an object method for assigning a null value to the second structured information format attribute value using the software object for the rule to be created, when the source which has been input by the user indicates no source is to be used.

22. A system according to Claim 20, further comprising:

a reference to a software object for a rule to be created;
an object method for examining the source which has been input by the user; and
an object method for assigning a system value to the second structured information format attribute value using the software object for the rule to be created, when the source which has been input by the user indicates a system source is to be used.

23. A system according to Claim 20, further comprising:

a reference to a software object for a rule to be created;
an object method for examining the source which has been input by the user; and
an object method for assigning a first structured information format attribute value to the second structured information format attribute value using the software object for the rule to be created, when the source which has been input by the user indicates a first structured information format attribute source is to be used.

24. A system according to Claim 20, further comprising:

a reference to a software object for a rule to be created;
an object method for examining the source which has been input by the user; and
an object method for assigning a first structured information format content value to the second structured information format attribute value using the software object for the rule to be created, when the source which has been input by the user indicates a first structured information format content source is to be used.

25. A system according to Claim 20, further comprising:

a reference to a storage buffer for the source which has been input by the user;
an object method for examining the source which has been input by the user using the storage buffer for the source which has been input by the user;
an object method for interactively inputting a user input value, when the source which has been input by the user indicates a user input source is to be used; and
an object method for assigning the user input value to the second structured information format attribute value, when the source which has been input by the user indicates a user input source is to be used.

26. A system according to Claim 1, wherein the user comprises:

a software object.

27. An object-oriented computer program product for processing structured information for implementation by a computer in an object-oriented framework, comprising:

a storage means;
a first obtaining means for obtaining an interactive input from a user;
a second obtaining means for obtaining a first structural description of a first structured information format;
a third obtaining means for obtaining a second structural description of a second structured information format;
means for creating a rule to transform an element of the first structured information format into an element of the second structured information format utilizing the interactive input from the user, the first structural description, and the second structural description; and means for outputting the rule,
wherein at least one of the first obtaining means, the second obtaining means, the third obtaining means, the means for creating, and the means for outputting includes a software object.

28. A computer program product according to Claim 27, wherein the first structured information format includes ISO/IEC 9070 public identifier naming format, the second structured information format includes an operating system file name format and the means for creating comprises:

means for creating a rule to transform an element of a first structured information format which includes an ISO/IEC 9070 public identifier element into an element of a second structured information format which includes an operating system file name format element utilizing the interactive input from the user, the first structural description which includes a structural description of the ISO/IEC 9070 public identifier format, and the second structural description which includes a structural description of the operating system file name format.

29. A computer program product according to Claim 27, wherein the structured information includes database variable names, the first structured information format includes a first database variable name format, the second structured information format includes a second database variable name format, and the means for creating comprises:

means for creating a rule to transform an element of a first structured information format which includes a first database variable name format element into an element of a second structured information format which includes a second database variable name format element utilizing the interactive input from the user, the first structural description which includes a structural description of the first database variable name format, and the second structural description which includes a structural description of the second database variable name format.

30. A computer program product according to Claim 27, wherein the structured information includes markup language, the first structured information format includes a first markup language, the second structured information format includes a second markup language, and the means for creating comprises:

means for creating a rule to transform an element of a first structured information format which includes a first markup language element into an element of a second structured information format which includes a second markup language element utilizing the interactive input from the user, the first structural description which includes a structural description of the first markup language, and the second structural description which includes a structural description of the second markup language.

31. A computer program product according to Claim 30, wherein the first markup language includes SGML, the second markup language includes HTML, and the means for creating further comprises:

means for creating a rule to transform an element of a first markup language which includes an SGML element into an element of a second markup language which includes an HTML element utilizing the interactive input from the user, the first structural description which includes an SGML DTD, and the second structural description which includes an HTML DTD.

32. A computer implemented method to provide a graphical user interface for creating a mapping of a first structural description to a second structural description, comprising the steps of:

displaying an element for transformation of a first structural description;
displaying a list of candidate elements of a second structural description;
inputting, from a user, a first selection of one of the candidate elements of the second structural description which defines a correspondence between the element for transformation of the first structural description and the first selection of one of the candidate elements of the second structural description for a transformation of the element of the first structural description to the second structural description; and
storing the correspondence between the element for transformation of the first structural description and the first selection of one of the candidate elements of the second structural description as a rule.

33. A method according to Claim 32, wherein the first structural description includes an ISO/IEC 9070 public identifier naming format, the second structural description includes an operating system file name format, and the step of displaying the element for transformation comprises:

displaying the element for transformation which includes an element of the ISO/IEC 9070 public identifier naming format; and
the step of displaying the list of candidate elements comprises:
displaying the list of candidate elements which includes a list of operating system file name candidate elements.

34. A method according to Claim 32, wherein the first structural description includes a first database variable name format, the second structural description includes a second database variable name format, and the step of displaying the element for transformation comprises:

displaying the element for transformation which includes an element of the first database variable name format; and
the step of displaying the list of candidate elements comprises:
displaying the list of candidate elements which includes a list of second database variable name format candidate elements.

35. A method according to Claim 32, further comprising the steps of:

obtaining the stored rule;
displaying the element for transformation of the first structural description;
displaying the first selected element of the second structural description which defines a correspondence between the element for transformation of the first structural description and the first selection of one of the candidate elements of the second structural description for a transformation of the element of the first structural description to the second structural description;
displaying the list of candidate elements of the second structural description;
inputting, from the user, a second selection of one of the candidate elements of the second structural description which defines a correspondence between the element for transformation of the first structural description and the second selection of one of the candidate elements of the second structural description for a transformation of the element of the first structural description to the second structural description; and
storing the correspondence between the element for transformation of the first structural description and the second selection of one of the candidate elements of the second structural description as a rule.

36. A method according to Claim 32, further comprising the steps of:

displaying an icon for the user to input a request to clear the first selection which is being displayed;
inputting the request to clear the first selection which is being displayed, when the user inputs the request to clear the first selection which is being displayed; and
clearing the first selection which is being displayed, when the user inputs the request to clear the first selection which is being displayed.

37. A method according to Claim 32, wherein the first structural description includes a first markup language, the second structural description includes a second markup language, and the step of displaying the element for transformation comprises:

displaying the element for transformation which includes an element of the first markup language; and
the step of displaying the list of candidate elements comprises:
displaying the list of candidate elements which includes a list of second markup language candidate elements.

38. A method according to Claim 37, wherein the first markup language includes a Standard Generalized Markup Language ("SGML"), the second markup language includes a HyperText Markup Language ("HTML"), and the step of displaying the element for transformation comprises:

displaying the element for transformation which includes an element of SGML; and
the step of displaying the list of candidate elements comprises:
displaying the list of candidate elements which includes a list of HTML candidate elements.

39. A method according to Claim 32, wherein the storing step comprises:

storing the correspondence between the element for transformation of the first structural description and the first selection of one of the candidate elements of the second structural description as a rule in a list of rules for transformation.

40. A method according to Claim 39, further comprising the steps of:

displaying a second element for transformation of the first structural description;
displaying a list of candidate elements of the second structural description;
inputting, from the user, a second selection of one of the candidate elements of the second structural description which defines a correspondence between the second element for transformation of the first structural description and the second selection of one of the candidate elements of the second structural description for a transformation of the second element of the first structural description to the second structural description; and
storing the correspondence between the second element for transformation of the first structural description and the second selection of one of the candidate elements of the second structural description as a rule in a list of rules for transformation.

41. A method according to Claim 39, wherein the inputting step further comprises:

displaying an icon for the user to input a request to store the first selection correspondence as a rule in the list of rules for transformation; and
the storing step further comprises:
inputting the request to store the correspondence as a rule in the list of rules for transformation, when the user inputs the request to store the first selection as a rule; and
displaying a second element for transformation of the first structural description, when the user inputs the request to store the first selection as a rule.

42. A method according to Claim 39, wherein the inputting step further comprises:

displaying an icon for the user to input a request to store the correspondence as a rule in the list of rules for transformation; and
the storing step further comprises:
inputting the request to store the correspondence as a rule in the list of rules for transformation, when the user inputs the request to store the first selection as a rule in the list of rules for transformation; and
storing the list of rules for transformation as a map.

43. A method according to Claim 39, further comprising the steps of:

displaying an icon for the user to input a request to delete the list of rules for transformation;
inputting the request to delete the list of rules for transformation, when the user inputs the request to delete the list or rules for transformation; and
deleting the list of rules for transformation, when the user inputs a request to delete the list of rules for transformation.

44. A method according to Claim 32, further comprising the steps of:

displaying the first selection of one of the candidate elements of the second structural description which defines the correspondence between the element for transformation of the first structural description and the first selection of one of the candidate elements of the second structural description for a transformation of the element of the first structural description to the second structural description.

45. A method according to Claim 44, wherein the inputting step comprises:

inputting, from the user, a first ordered list of a plurality of the candidate elements of the second structural description which defines a correspondence between the element for transformation of the first structural description and the first ordered list of the plurality of the candidate elements of the second structural description for a transformation of the element of the first structural description to the second structural description;
the step of displaying the first selection comprises:
displaying the first ordered list of the plurality of the candidate elements of the second markup language which defines a correspondence between the element for transformation of the first structural description and the first ordered list of the plurality of the candidate elements of the second structural description for a transformation of the element of the first structural description to the second structural description; and
the storing step comprises:
storing the correspondence between the element for transformation of the first structural description and the

first ordered list of the plurality of the candidate elements of the second structural description for a transformation of the element of the first structural description to the second structural description as a rule.

46. A method according to Claim 45, further comprising the steps of:

obtaining the stored rule;
displaying the element for transformation of the first structural description;
displaying the first ordered list of elements of the second structural description which defines a correspondence between the element for transformation of the first structural description and the first ordered list of the plurality of the candidate elements of the second structural description for a transformation of the element of the first structural description to the second structural description;
displaying the list of candidate elements of the second structural description;
inputting, from the user, a second ordered list of a plurality of the candidate elements of the second structural description which defines a correspondence between the element for transformation of the first structural description and the second ordered list of the plurality of the candidate elements of the second structural description for a transformation of the element of the first structural description to the second structural description; and
storing the correspondence between the element for transformation of the first structural description and the second ordered list of the plurality of the candidate elements of the second structural description for a transformation of the element of the first structural description to the second structural description as a rule.

47. A method according to Claim 45, further comprising the steps of:

displaying an icon for the user to input a request to clear the first ordered list which is being displayed;
inputting the request to clear the first ordered list which is being displayed, when the user inputs the request to clear the first ordered list which is being displayed; and
clearing the first ordered list which is being displayed, when the user inputs the request to clear the first ordered list which is being displayed.

48. A method according to Claim 32, further comprising the steps of:

displaying an attribute of the first selection of one of the candidate elements of the second structural description which corresponds to a transformation of the element of the first structural description to the second structural description, for assignment of an attribute value of the second structural description;
displaying a plurality of icons representing sources for obtaining the attribute value to be assigned to the attribute of the first selection which is being displayed;
displaying the element of the first structural description;
displaying an attribute list of the element of the first structural description;
inputting a user input of a selection of sources for obtaining the attribute value to be assigned to the attribute of the first selection which is being displayed; and
processing the user input of the selection of sources for obtaining the attribute value to be assigned to the attribute of the first selection which is being displayed, when the user inputs the selection of sources for obtaining the attribute value to be assigned to the attribute of the first selection which is being displayed.

49. A method according to Claim 48, wherein the processing step further comprises:

assigning a null value to the attribute of the first selection which is being displayed, when the selection of sources which has been input by the user indicates no source is to be used.

50. A method according to Claim 48, wherein the processing step further comprises:

assigning a system value to the attribute of the first selection which is being displayed, when the selection of sources which has been input by the user indicates a system value is to be used.

51. A method according to Claim 48, wherein the processing step further comprises:

assigning a first structural description attribute value to the attribute of the first selection which is being displayed, when the selection of sources which has been input by the user indicates a first structural description

attribute value source is to be used.

52. A method according to Claim 48, wherein the processing step further comprises:

assigning a first structural description content value to the attribute of the first selection which is being displayed, when the selection of sources which has been input by the user indicates a first structural description content value source is to be used.

53. A method according to Claim 48, wherein the processing step further comprises:

assigning a user input value to the attribute of the first selection which is being displayed, when the selection of sources which has been input by the user indicates a user input value source is to be used.

54. A method according to Claim 53, wherein the assigning step comprises:

displaying a text input area for the user to input a value to be assigned;
inputting the value entered by the user in the text input area; and
assigning the value input by the user to the attribute of the first selection which is being displayed.

55. A method according to Claim 32, wherein the step of displaying a list of candidate elements of a second structural description further comprises:

displaying a candidate for requesting removal of the first structural description element in the transformation; and
displaying a candidate for requesting ignoring of the first structural description element in the transformation.

56. An apparatus for providing a graphical user interface for creating a mapping of a first structural description to a second structural description, comprising:

an element displaying means for displaying an element for transformation of a first structural description;
a list displaying means for displaying a list of candidate elements of a second structural description;
a user inputting means for inputting, from a user, a first selection of one of the candidate elements of the second structural description which defines a correspondence between the element for transformation of the first structural description and the first selection of one of the candidate elements of the second structural description for a transformation of the element of the first structural description to the second structural description; and
a storing means for storing the correspondence between the element for transformation of the first structural description and the first selection of one of the candidate elements of the second structural description as a rule.

57. An apparatus according to Claim 56, wherein the first structural description includes an ISO/IEC 9070 public identifier naming format, the second structural description includes an operating system file name format, and the element displaying means further comprises:

means for displaying the element for transformation which includes an element of the ISO/IEC 9070 public identifier naming format; and
the list displaying means further comprises:
means for displaying the list of candidate elements which includes a list of operating system file name candidate elements.

58. An apparatus according to Claim 56, wherein the first structural description includes a first database variable name format, the second structural description includes a second database variable name format, and the element displaying means further comprises:

means for displaying the element for transformation which includes an element of the first database variable name format; and
the list displaying means further comprises:
means for displaying the list of candidate elements which includes a list of second database variable name for-

mat candidate elements.

59. An apparatus according to Claim 56, further comprising:

means for obtaining the stored rule;
means for displaying the element for transformation of the first structural description;
means for displaying the first selected element of the second structural description which defines a correspondence between the element for transformation of the first structural description and the first selection of one of the candidate elements of the second structural description for a transformation of the element of the first structural description to the second structural description;
means for displaying the list of candidate elements of the second structural description;
means for inputting, from the user, a second selection of one of the candidate elements of the second structural description which defines a correspondence between the element for transformation of the first structural description and the second selection of one of the candidate elements of the second structural description for a transformation of the element of the first structural description to the second structural description; and
means for storing the correspondence between the element for transformation of the first structural description and the second selection of one of the candidate elements of the second structural description as a rule.

60. An apparatus according to Claim 56, further comprising:

means for displaying an icon for the user to input a request to clear the first selection which is being displayed;
means for inputting the request to clear the first selection which is being displayed, when the user inputs the request to clear the first selection which is being displayed; and
means for clearing the first selection which is being displayed, when the user inputs the request to clear the first selection which is being displayed.

61. An apparatus according to Claim 56, wherein the first structural description includes a first markup language, the second structural description includes a second markup language, and the element displaying means further comprises:

means for displaying the element for transformation which includes an element of the first markup language; and
the list displaying means further comprises:
means for displaying the list of candidate elements which includes a list of second markup language candidate elements.

62. An apparatus according to Claim 61, wherein the first markup language includes an SGML, the second markup language includes an HTML, and the element displaying means further comprises:

means for displaying the element for transformation which includes an element of SGML; and
the list displaying means comprises:
means for displaying the list of candidate elements which includes a list of HTML candidate elements.

63. An apparatus according to Claim 56, wherein the storing means comprises:

means for storing the correspondence between the element for transformation of the first structural description and the first selection of one of the candidate elements of the second structural description as a rule in a list of rules for transformation.

64. An apparatus according to Claim 63, further comprising:

means for displaying a second element for transformation of the first structural description;
means for displaying a list of candidate elements of the second structural description;
means for inputting, from the user, a second selection of one of the candidate elements of the second structural description which defines a correspondence between the second element for transformation of the first structural description and the second selection of one of the candidate elements of the second structural description for a transformation of the second element of the first structural description to the second structural description; and

means for storing the correspondence between the second element for transformation of the first structural description and the second selection of one of the candidate elements of the second structural description as a rule in a list of rules for transformation.

65. An apparatus according to Claim 63, further comprising:

means for displaying an icon for the user to input a request to store the first selection correspondence as a rule in the list of rules for transformation;
means for inputting the request to store the correspondence as a rule in the list of rules for transformation, when the user inputs the request to store the first selection as a rule; and
means for displaying a second element for transformation of the first structural description, when the user inputs the request to store the first selection as a rule.

66. An apparatus according to Claim 63, further comprising:

means for displaying an icon for the user to input a request to store the correspondence as a rule in the list of rules for transformation;
means for inputting the request to store the correspondence as a rule in the list of rules for transformation, when the user inputs the request to store the first selection as a rule in the list of rules for transformation; and
means for storing the list of rules for transformation as a map.

67. An apparatus according to Claim 63, further comprising:

means for displaying an icon for the user to input a request to delete the list of rules for transformation;
means for inputting the request to delete the list of rules for transformation, when the user inputs the request to delete the list of rules for transformation; and
means for deleting the list of rules for transformation, when the user inputs a request to delete the list of rules for transformation.

68. An apparatus according to Claim 56, further comprising:

means for displaying the first selection of one of the candidate elements of the second structural description which defines the correspondence between the element for transformation of the first structural description and the first selection of one of the candidate elements of the second structural description for a transformation of the element of the first structural description to the second structural description.

69. An apparatus according to Claim 68, wherein the user inputting means comprises:

means for inputting, from the user, a first ordered list of a plurality of the candidate elements of the second structural description which defines a correspondence between the element for transformation of the first structural description and the first ordered list of the plurality of the candidate elements of the second structural description for a transformation of the element of the first structural description to the second structural description;
the means for displaying the first selection comprises:
means for displaying the first ordered list of the plurality of the candidate elements of the second markup language which defines a correspondence between the element for transformation of the first structural description and the first ordered list of the plurality of the candidate elements of the second structural description for a transformation of the element of the first structural description to the second structural description; and
the storing means comprises:
means for storing the correspondence between the element for transformation of the first structural description and the first ordered list of the plurality of the candidate elements of the second structural description for a transformation of the element of the first structural description to the second structural description as a rule.

70. An apparatus according to Claim 69, further comprising:

means for obtaining the stored rule;
means for displaying the element for transformation of the first structural description;
means for displaying the first ordered list of elements of the second structural description which defines a cor-

respondence between the element for transformation of the first structural description and the first ordered list of the plurality of the candidate elements of the second structural description for a transformation of the element of the first structural description to the second structural description;

means for displaying the list of candidate elements of the second structural description;

means for inputting, from the user, a second ordered list of a plurality of the candidate elements of the second structural description which defines a correspondence between the element for transformation of the first structural description and the second ordered list of the plurality of the candidate elements of the second structural description for a transformation of the element of the first structural description to the second structural description; and

means for storing the correspondence between the element for transformation of the first structural description and the second ordered list of the plurality of the candidate elements of the second structural description for a transformation of the element of the first structural description to the second structural description as a rule.

71. An apparatus according to Claim 69, further comprising:

means for displaying an icon for the user to input a request to clear the first ordered list which is being displayed;

means for inputting the request to clear the first ordered list which is being displayed, when the user inputs the request to clear the first ordered list which is being displayed; and

means for clearing the first ordered list which is being displayed, when the user inputs the request to clear the first ordered list which is being displayed.

72. An apparatus according to Claim 56, further comprising:

means for displaying an attribute of the first selection of one of the candidate elements of the second structural description which corresponds to a transformation of the element of the first structural description to the second structural description, for assignment of an attribute value of the second structural description;

means for displaying a plurality of icons representing sources for obtaining the attribute value to be assigned to the attribute of the first selection which is being displayed;

means for displaying the element of the first structural description;

means for displaying an attribute list of the element of the first structural description;

means for inputting a user input of a selection of sources for obtaining the attribute value to be assigned to the attribute of the first selection which is being displayed; and

an input processing means for processing the user input of the selection of sources for obtaining the attribute value to be assigned to the attribute of the first selection which is being displayed, when the user inputs the selection of sources for obtaining the attribute value to be assigned to the attribute of the first selection which is being displayed.

73. An apparatus according to Claim 72, wherein the input processing means further comprises:

means for assigning a null value to the attribute of the first selection which is being displayed, when the selection of sources which has been input by the user indicates no source is to be used.

74. An apparatus according to Claim 72, wherein the input processing means further comprises:

means for assigning a system value to the attribute of the first selection which is being displayed, when the selection of sources which has been input by the user indicates a system value is to be used.

75. An apparatus according to Claim 72, wherein the input processing means further comprises:

means for assigning a first structural description attribute value to the attribute of the first selection which is being displayed, when the selection of sources which has been input by the user indicates a first structural description attribute value source is to be used.

76. An apparatus according to Claim 72, wherein the input processing means further comprises:

an assigning means for assigning a first structural description content value to the attribute of the first selection which is being displayed, when the selection of sources which has been input by the user indicates a first struc-

tural description content value source is to be used.

77. An apparatus according to Claim 72, wherein the input processing means further comprises:

means for assigning a user input value to the attribute of the first selection which is being displayed, when the selection of sources which has been input by the user indicates a user input value source is to be used.

78. An apparatus according to Claim 77, wherein the assigning means comprises:

means for displaying a text input area for the user to input a value to be assigned;
means for inputting the value entered by the user in the text input area; and
means for assigning the value input by the user to the attribute of the first selection which is being displayed.

79. An apparatus according to Claim 56, wherein the list displaying means further comprises:

means for displaying a candidate for requesting removal of the element for transformation of the first structural description in the transformation; and
means for displaying a candidate for requesting ignoring of the element for transformation of the first structural description in the transformation.

80. A computer program product including a computer readable medium for providing a graphical user interface for creating a mapping of a first structural description to a second structural description, comprising:

element displaying means for displaying an element for transformation of a first structural description;
list displaying means for displaying a list of candidate elements of a second structural description;
user inputting means for inputting, from a user, a first selection of one of the candidate elements of the second structural description which defines a correspondence between the element for transformation of the first structural description and the first selection of one of the candidate elements of the second structural description for a transformation of the element of the first structural description to the second structural description; and
storing means for storing the correspondence between the element for transformation of the first structural description and the first selection of one of the candidate elements of the second structural description as a rule.

81. A computer program product according to Claim 80, wherein the first structural description includes ISO/IEC 9070 public identifier format, the second structural description includes an operating system file name format, and the element displaying means comprises:

means for displaying the element for transformation which includes an element of the ISO/IEC 9070 public identifier format; and
the list displaying means comprises:
means for displaying the list of candidate elements which includes a list of operating system file name candidate elements.

82. A computer program product according to Claim 80, wherein the first structural description includes a first database variable name format, the second structural description includes a second database variable name format, and the element displaying means comprises:

means for displaying the element for transformation which includes an element of the first database variable name format; and
the list displaying means comprises:
means for displaying the list of candidate elements which includes a list of second database variable name format candidate elements.

83. A computer program product according to Claim 80, further comprising:

means for obtaining the stored rule;
means for displaying the element for transformation of the first structural description;
means for displaying the first selected element of the second structural description which defines a corre-

spondence between the element for transformation of the first structural description and the first selection of one of the candidate elements of the second structural description for a transformation of the element of the first structural description to the second structural description;

means for displaying the list of candidate elements of the second structural description;

means for inputting, from the user, a second selection of one of the candidate elements of the second structural description which defines a correspondence between the element for transformation of the first structural description and the second selection of one of the candidate elements of the second structural description for a transformation of the element of the first structural description to the second structural description; and

means for storing the correspondence between the element for transformation of the first structural description and the second selection of one of the candidate elements of the second structural description as a rule.

84. A computer program product according to Claim 80, further comprising:

means for displaying an icon for the user to input a request to clear the first selection which is being displayed;

means for inputting the request to clear the first selection which is being displayed, when the user inputs the request to clear the first selection which is being displayed; and

means for clearing the first selection which is being displayed, when the user inputs the request to clear the first selection which is being displayed.

85. A computer program product according to Claim 80, wherein the first structural description includes a first markup language, the second structural description includes a second markup language, and the element displaying means comprises:

means for displaying the element for transformation which includes an element of the first markup language; and

the list displaying means comprises:

means for displaying the list of candidate elements which includes a list of second markup language candidate elements.

86. A computer program product according to Claim 85, wherein the first markup language includes SGML, the second markup language includes HTML, and the element displaying means further comprises:

means for displaying the element for transformation which includes an element of SGML; and

the list displaying means further comprises:

means for displaying the list of candidate elements which includes a list of HTML candidate elements.

```
22 <!--    sample1.dtd   >      -->
24 <!ELEMENT t - - (t1?, t2?, t3?, t4?) ->
26 <!ELEMENT t1 - - CDATA>
28 <!ATTLIST t1 name CDATA #REQUIRED>
30 <!ELEMENT t2 - - CDATA>
32 <!ELEMENT t3 - - CDATA>
34 <!ELEMENT t4 - - CDATA>
```

Fig. 1A

```
42 t--> <html><title>Title</title>
44 t1--><H3><A NAME="the source is t1's name">
46 t2--> <P>
48 t3--> <P>
50 t4--> <P><A HREF="# the source is t1's name">
```

Fig. 1B

```
60 <!DOCTYPE t system "sample1.dtd">
62 <t>
64 <t1 name="hilarry">1.  Hi Larry</t1>
66 <t2> This is the most fun I have ever had.</t2>
68 <t3> It must be great for you as well.</t3>
70 <t4> (Back to the Hi Larry greeting.)</t4>
72 </t>
```

Fig. 1C

```
80 <DOCTYPE HTML Public "-//W3C//DTD HTML 3.2 Final//EN">
82 <html>
84 <title>Title</title>
86 <H3><A name="hilarry">1.  Hi Larry</A></H3>
88 <P> This is the most fun I have ever had.</P>
90 <P> It must be great for you as well.</P>
92 <P><A HREF="#hilarry"> (Back to the Hi Larry greeting.)</A></P>
94 </html>
```

Fig. 1D

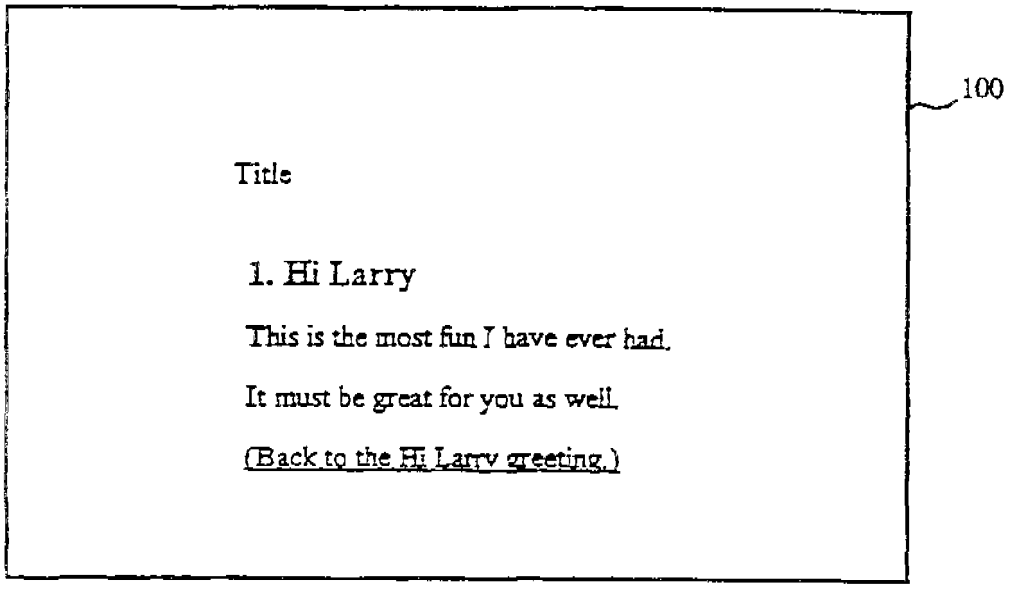

100
Title
1. Hi Larry
This is the most fun I have ever had.
It must be great for you as well.
(Back to the Hi Larry greeting.)

Fig. 2

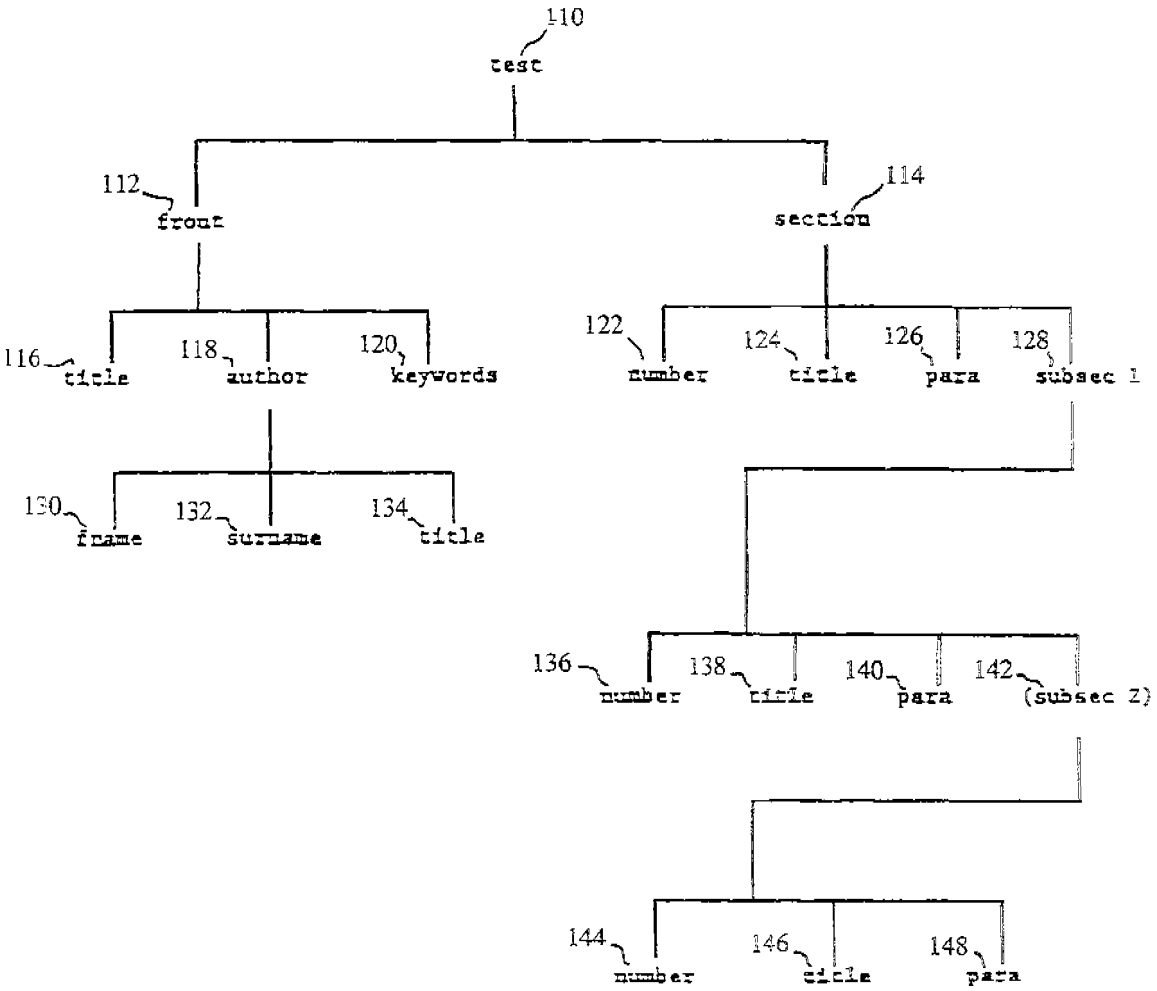

110
test
112
front
114
section
116
title
118
author
120
keywords
122
number
124
title
126
para
128
subsec 1
130
fname
132
surname
134
title
136
number
138
title
140
para
142
(subsec 2)
144
number
146
title
148
para

Fig. 3A

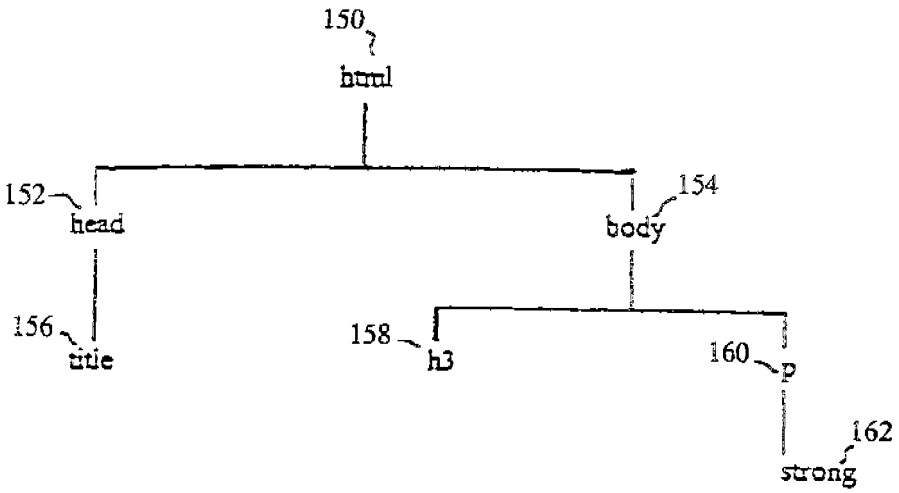


Fig. 3B

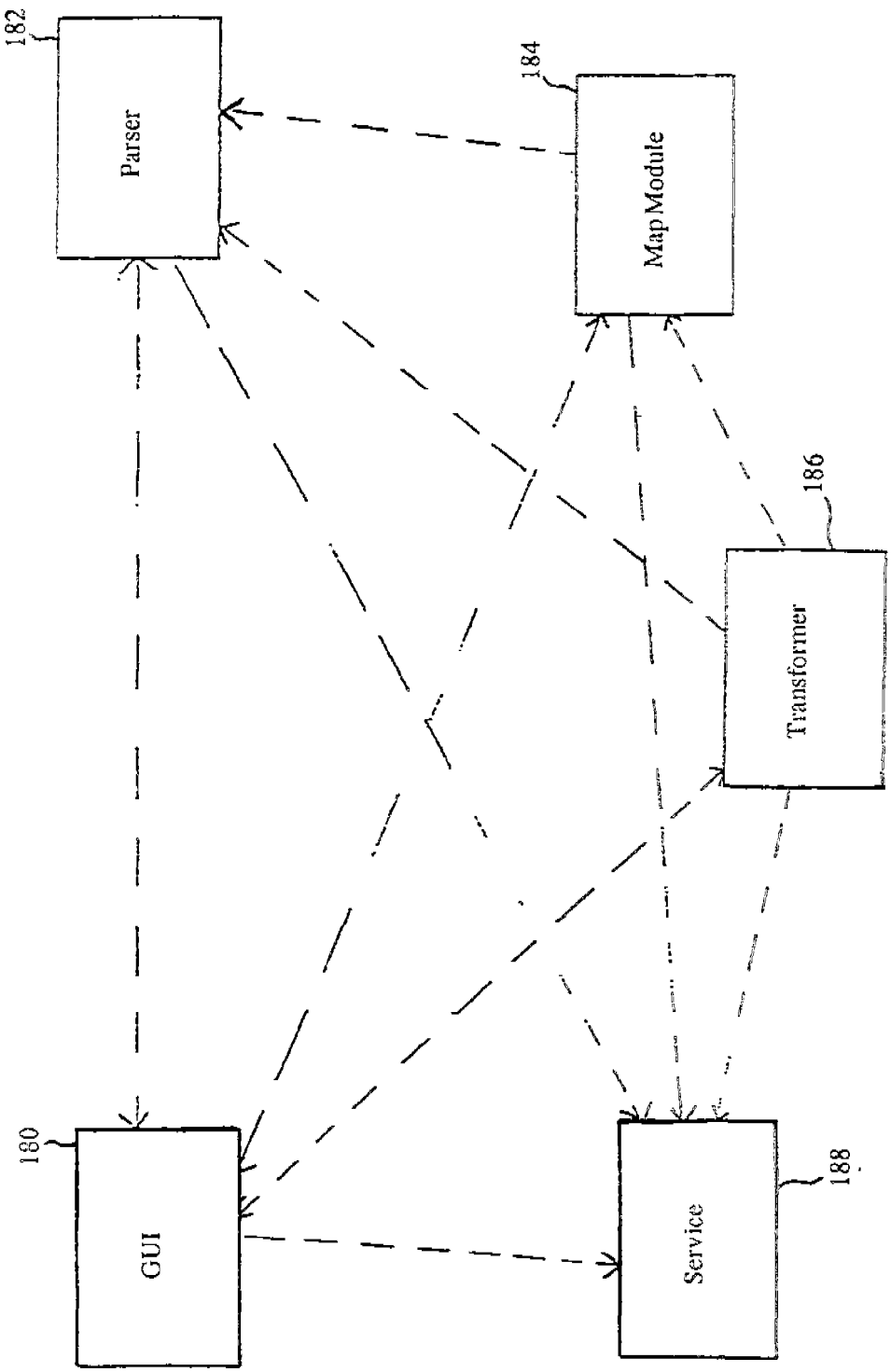

182
Parser
184
Map Module
186
Transformer
180
GUI
188
Service

Fig. 4

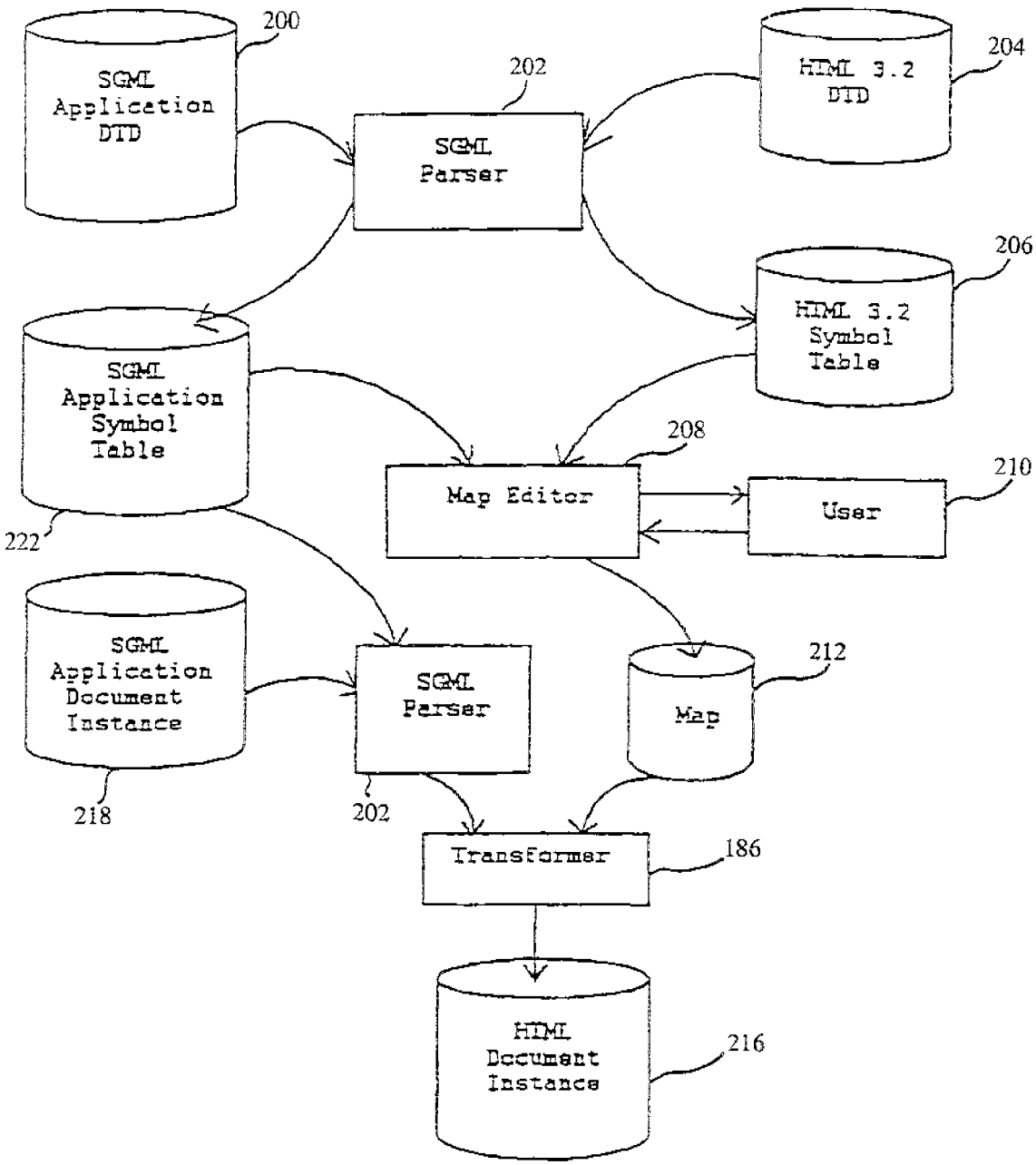

200
SGML Application DTD
202
SGML Parser
204
HTML 3.2 DTD
206
HTML 3.2 Symbol Table
SGML Application Symbol Table
208
Map Editor
210
User
222
SGML Application Document Instance
SGML Parser
212
Map
218
202
Transformer
186
HTML Document Instance
216

Fig. 5

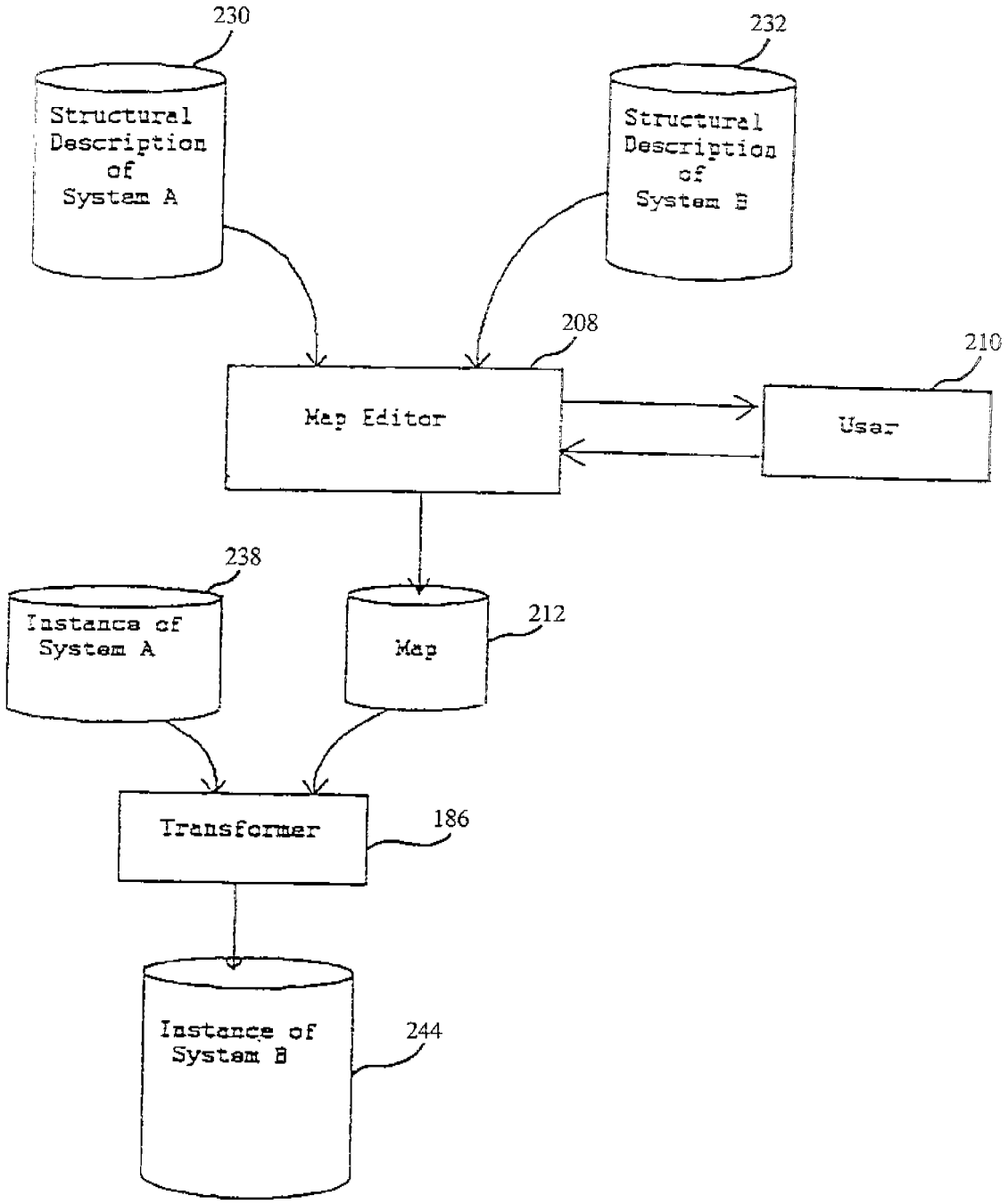

230
Structural Description of System A
232
Structural Description of System B
208
Map Editor
210
User
238
Instance of System A
212
Map
Transformer
186
Instance of System B
244

Fig. 6A

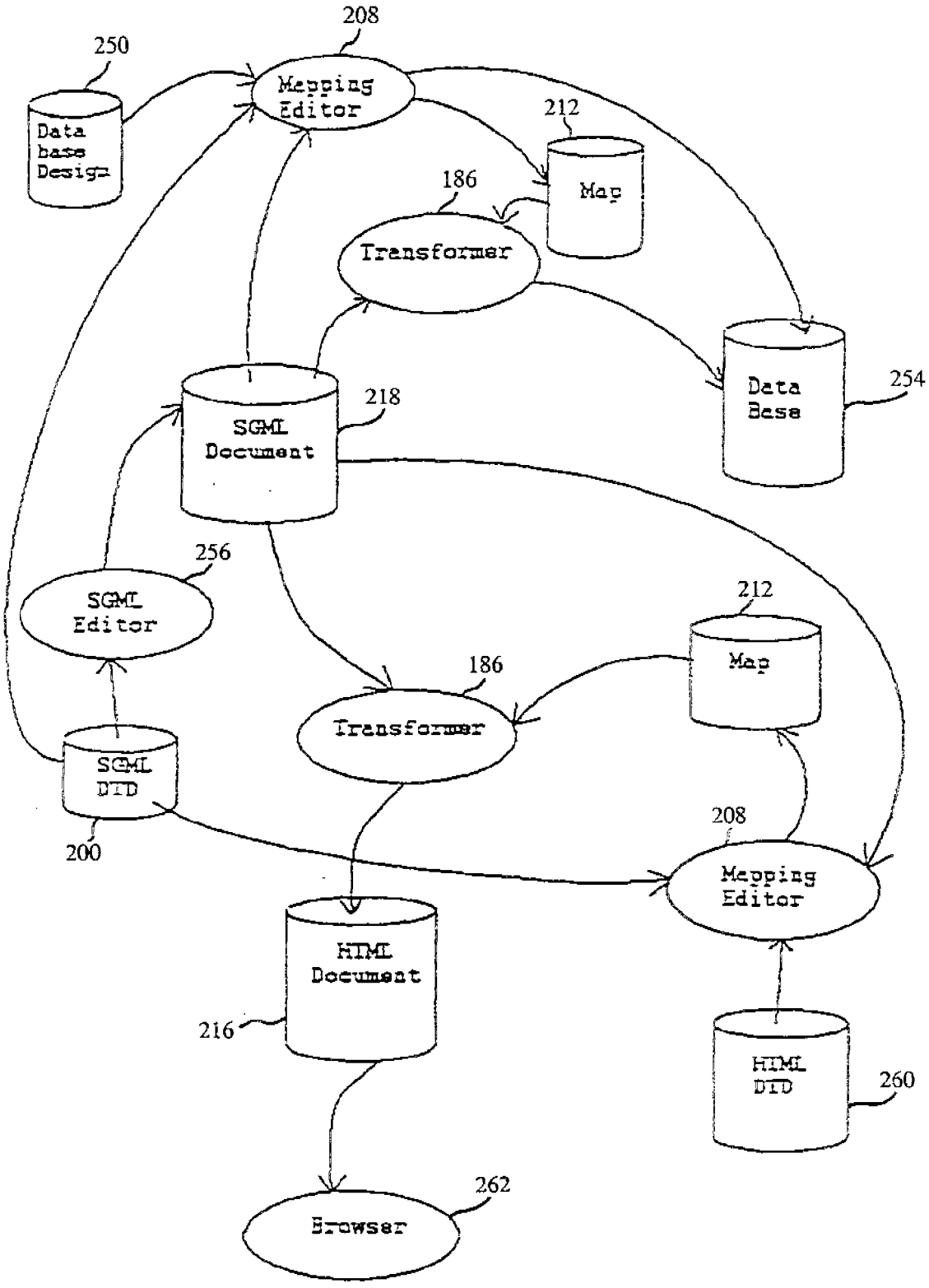

208
250
Mapping Editor
Data base Design
212
Map
186
Transformer
Data Base
254
218
SGML Document
256
SGML Editor
212
Map
186
Transformer
SGML DTD
200
208
Mapping Editor
HTML Document
216
HTML DTD
260
262
Browser

Fig. 6B

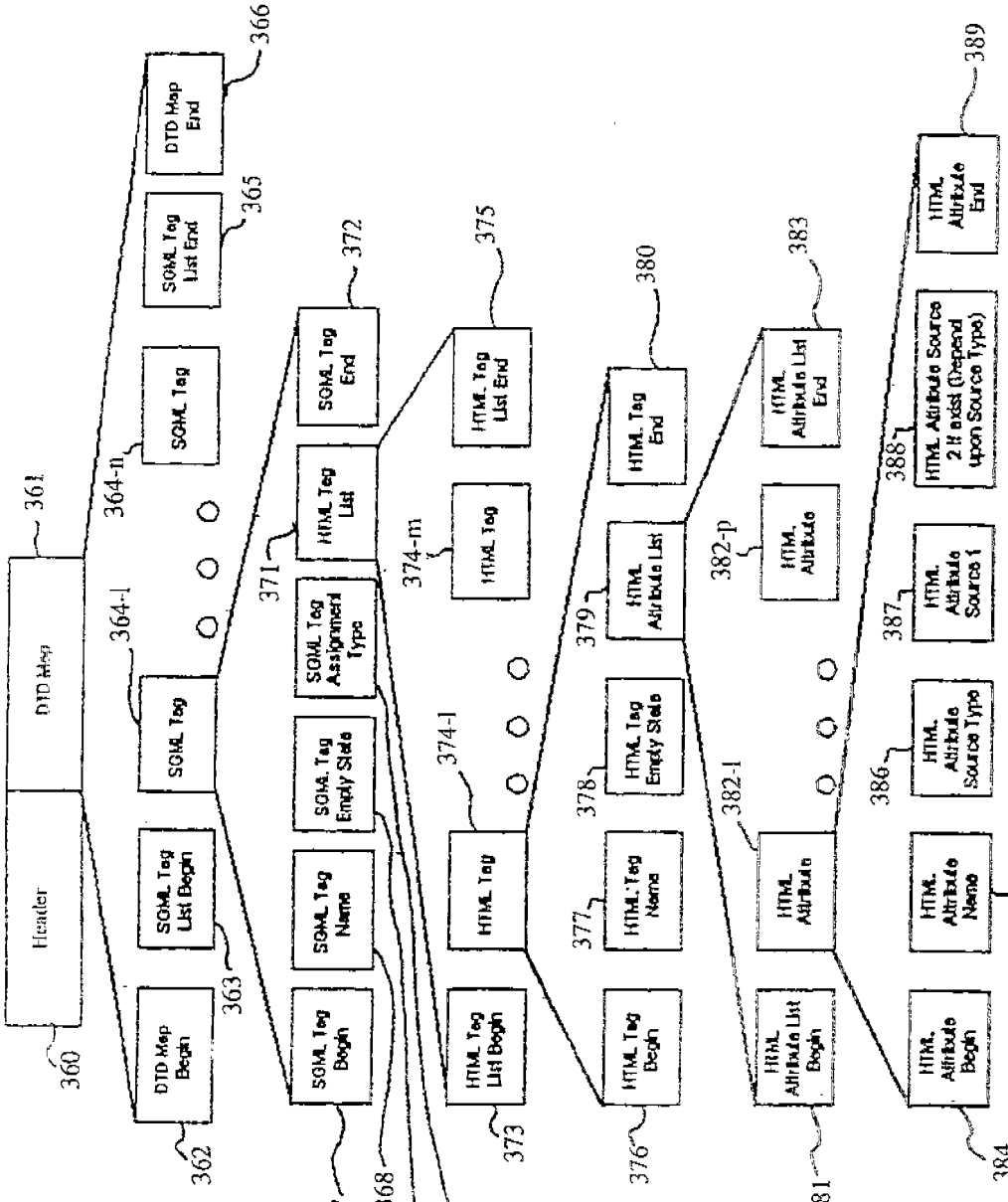

Header
DTD Map
DTD Map Begin
SGML Tag List Begin
SGML Tag
SGML Tag
SGML Tag List End
DTD Map End
SGML Tag Begin
SGML Tag Name
SGML Tag Empty Slate
SGML Tag Assignment Type
HTML Tag List
SGML Tag End
HTML Tag List Begin
HTML Tag
HTML Tag
HTML Tag List End
HTML Tag Begin
HTML Tag Name
HTML Tag Empty Slate
HTML Attribute List
HTML Tag End
HTML Attribute List Begin
HTML Attribute
HTML Attribute
HTML Attribute List End
HTML Attribute Begin
HTML Attribute Name
HTML Attribute Source Type
HTML Attribute Source 1
HTML Attribute Source 2 if exist (Depend upon Source Type)
HTML Attribute End
360
361
362
363
364-1
364-n
365
366
367
368
369
370
371
372
373
374-1
374-m
375
376
377
378
379
380
381
382-1
382-p
383
384
385
386
387
388
389

Fig. 7

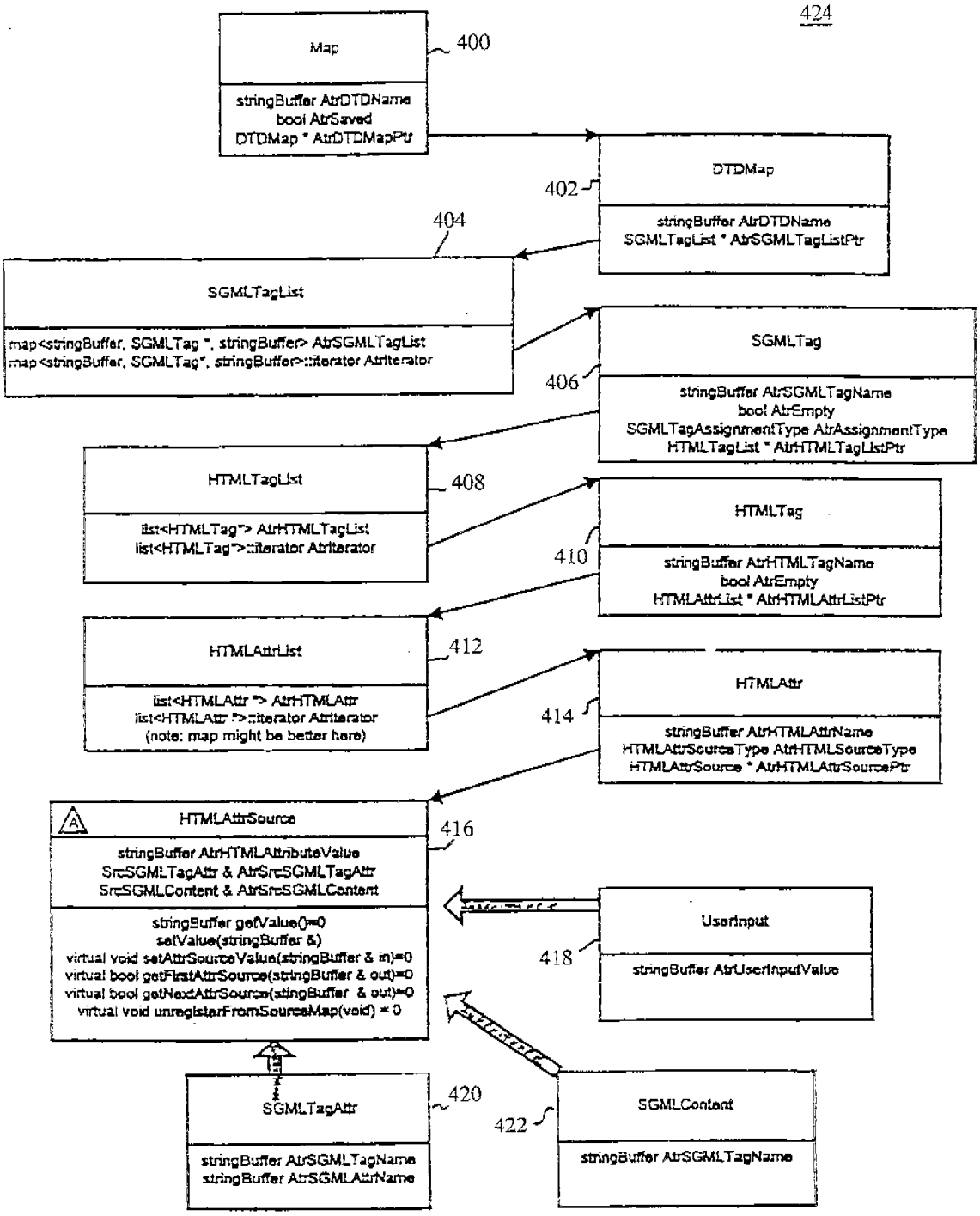

424
Map
400
stringBuffer AtrDTDName
bool AtrSaved
DTDMap * AtrDTDMapPtr
DTDMap
402
stringBuffer AtrDTDName
SGMLTagList * AtrSGMLTagListPtr
404
SGMLTagList
map<stringBuffer, SGMLTag *, stringBuffer> AtrSGMLTagList
map<stringBuffer, SGMLTag*, stringBuffer>::iterator AtrIterator
SGMLTag
406
stringBuffer AtrSGMLTagName
bool AtrEmpty
SGMLTagAssignmentType AtrAssignmentType
HTMLTagList * AtrHTMLTagListPtr
HTMLTagList
408
list<HTMLTag*> AtrHTMLTagList
list<HTMLTag*>::iterator AtrIterator
HTMLTag
410
stringBuffer AtrHTMLTagName
bool AtrEmpty
HTMLAttrList * AtrHTMLAttrListPtr
HTMLAttrList
412
list<HTMLAttr *> AtrHTMLAttr
list<HTMLAttr *>::iterator AtrIterator
(note: map might be better here)
HTMLAttr
414
stringBuffer AtrHTMLAttrName
HTMLAttrSourceType AtrHTMLSourceType
HTMLAttrSource * AtrHTMLAttrSourcePtr
HTMLAttrSource
416
stringBuffer AtrHTMLAttributeValue
SrcSGMLTagAttr & AtrSrcSGMLTagAttr
SrcSGMLContent & AtrSrcSGMLContent
stringBuffer getValue()=0
setValue(stringBuffer &)
virtual void setAttrSourceValue(stringBuffer & in)=0
virtual bool getFirstAttrSource(stringBuffer & out)=0
virtual bool getNextAttrSource(stringBuffer & out)=0
virtual void unregisterFromSourceMap(void) = 0
UserInput
418
stringBuffer AtrUserInputValue
SGMLTagAttr
420
stringBuffer AtrSGMLTagName
stringBuffer AtrSGMLAttrName
SGMLContent
422
stringBuffer AtrSGMLTagName

Fig. 8A

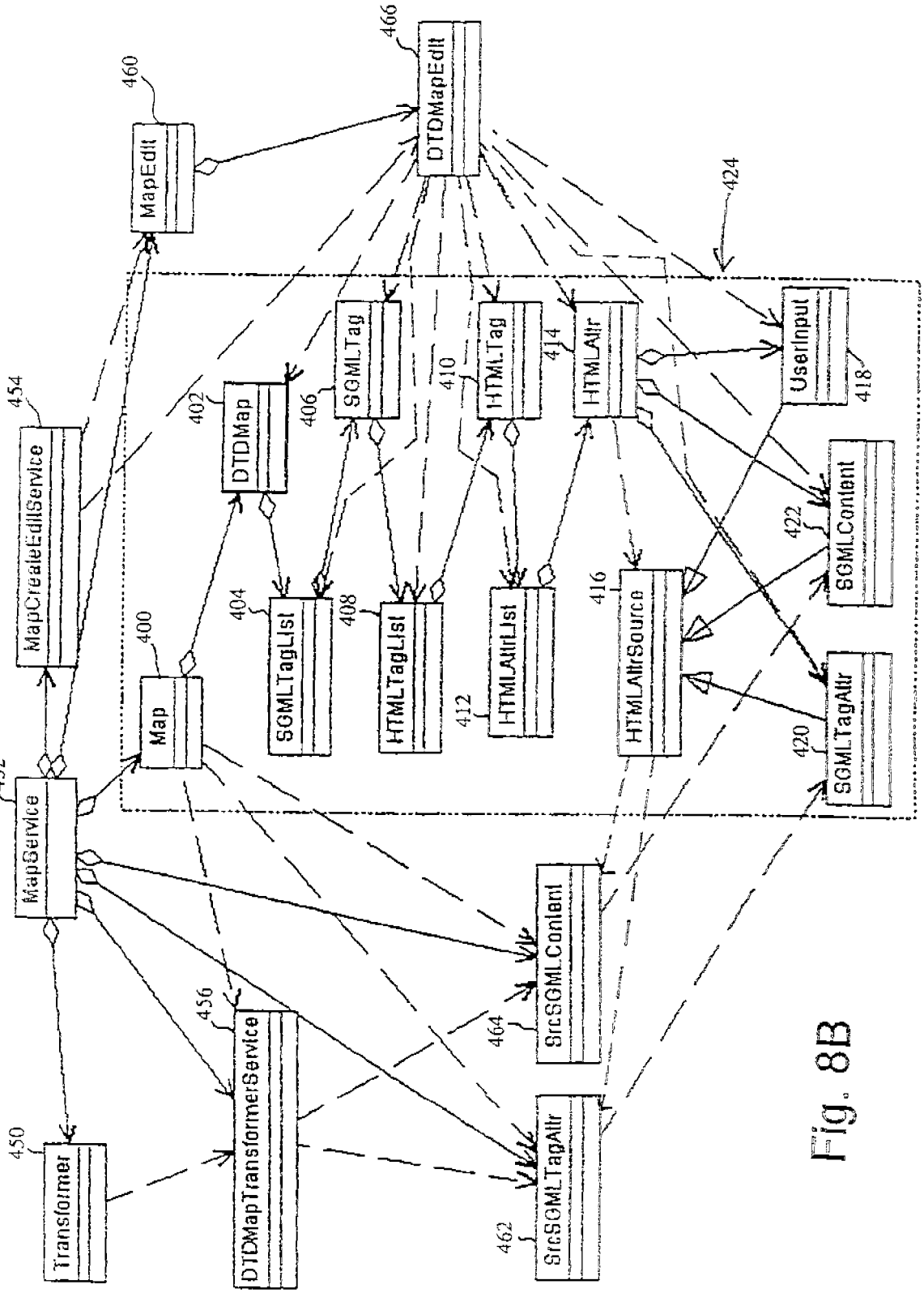

MapEdit
460
DTDMapEdit
466
424
MapCreateEditService
454
DTDMap
402
SGMLTag
406
HTMLTag
410
HTMLAttr
414
UserInput
418
SGMLContent
422
Map
400
SGMLTagList
404
HTMLTagList
408
HTMLAttrList
412
HTMLAttrSource
416
SGMLTagAttr
420
MapService
452
SrcSGMLContent
464
DTDMapTransformerService
456
SrcSGMLTagAttr
462
Transformer
450

Fig. 8B

462

SrcSGMLTagAttr
stringBuffer AtrCurrentSGMLTagName stringBuffer AtrCurrentAttribute stringBuffer AtrCurrentHTMLAttributeSourceValue stringBuffer AtrTagAttrKey map<stringBuffer, list<SGMLTagAttr *>, stringBuffer> AtrKeyOfSGMLTagAndAttribute
void registerSGMLTagNameAndAttributeName(SGMLTagAttr *) void unregisterTagAttrKeyAndMapEntry(SGMLTagAttr *) void setValueForAttributeOfTag (stringBuffer & Tag, stringBuffer & Attribute, stringBuffer & value) void reset(void)

Fig. 8C(1)

464

SrcSGMLContent
stringBuffer AtrCurrentSGMLTagName stringBuffer AtrCurrentHTMLAttributeSourceVal map<stringBuffer, list<SGMLContent *>, stringBuffer> AtrTableWithSGMLTagKey
void registerSGMLTagName(SGMLContent *) void unregisterSGMLTagName(SGMLContent *) void setValueForTag(stringBuffer & Tag, stringBuffer & value) void reset(void)

Fig. 8C(2)

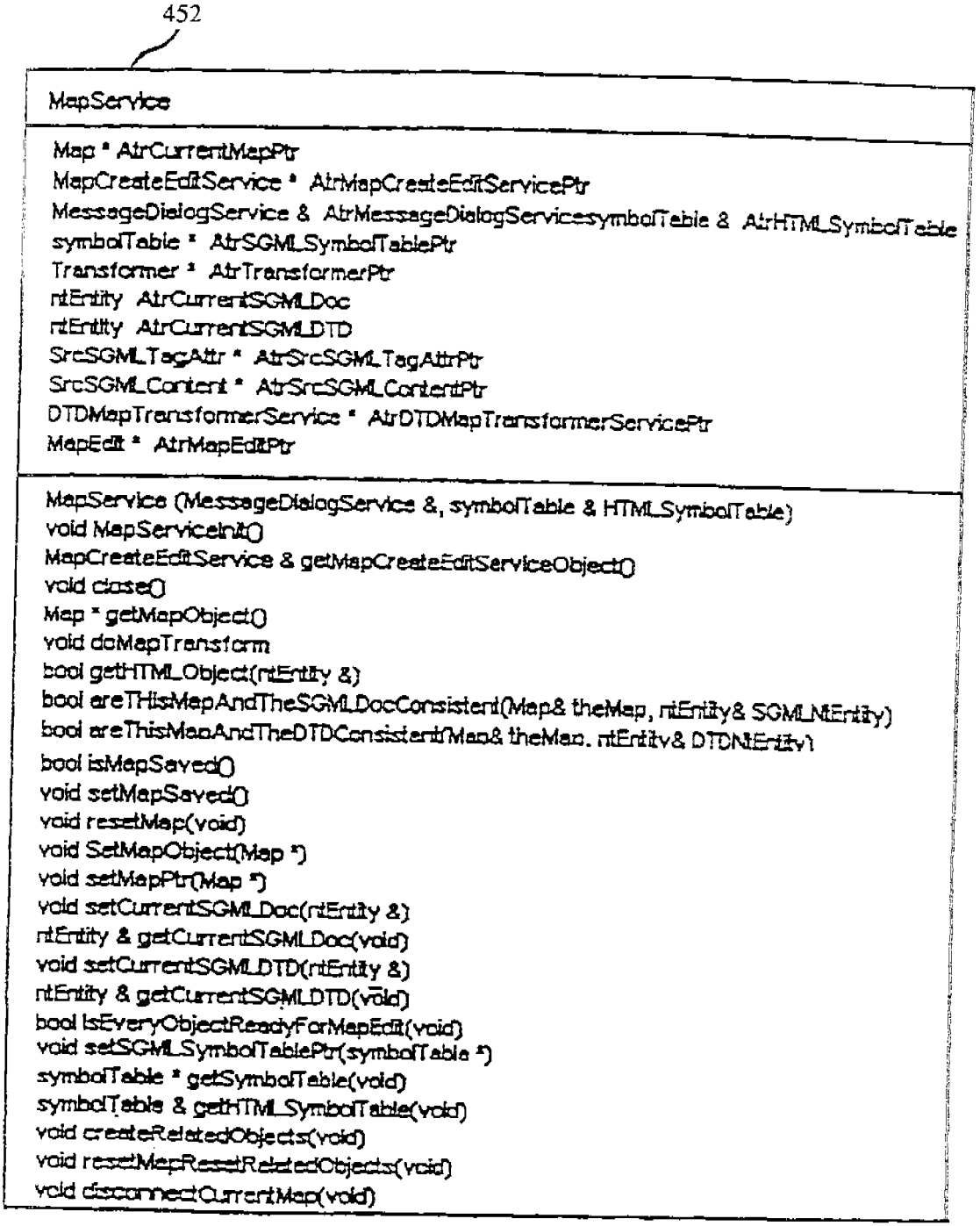

452
MapService
Map * AtrCurrentMapPtr
MapCreateEditService * AtrMapCreateEditServicePtr
MessageDialogService & AtrMessageDialogServicesymbolTable & AtrHTML_SymbolTable
symbolTable * AtrSGML_SymbolTablePtr
Transformer * AtrTransformerPtr
ntEntity AtrCurrentSGMLDoc
ntEntity AtrCurrentSGMLDTD
SrcSGMLTagAttr * AtrSrcSGMLTagAttrPtr
SrcSGMLContent * AtrSrcSGMLContentPtr
DTDMapTransformerService * AtrDTDMapTransformerServicePtr
MapEdit * AtrMapEditPtr
MapService (MessageDialogService &, symbolTable & HTML_SymbolTable)
void MapServiceInit()
MapCreateEditService & getMapCreateEditServiceObject()
void close()
Map * getMapObject()
void doMapTransform
bool getHTML_Object(ntEntity &)
bool areThisMapAndTheSGMLDocConsistent(Map& theMap, ntEntity& SGMLNtEntity)
bool areThisMapAndTheDTDConsistent(Map& theMap, ntEntity& DTDNtEntity)
bool isMapSaved()
void setMapSaved()
void resetMap(void)
void SetMapObject(Map *)
void setMapPtr(Map *)
void setCurrentSGMLDoc(ntEntity &)
ntEntity & getCurrentSGMLDoc(void)
void setCurrentSGMLDTD(ntEntity &)
ntEntity & getCurrentSGMLDTD(void)
bool isEveryObjectReadyForMapEdit(void)
void setSGML_SymbolTablePtr(symbolTable *)
symbolTable * getSymbolTable(void)
symbolTable & getHTML_SymbolTable(void)
void createRelatedObjects(void)
void resetMapResetRelatedObjects(void)
void disconnectCurrentMap(void)

Fig. 8C(3)

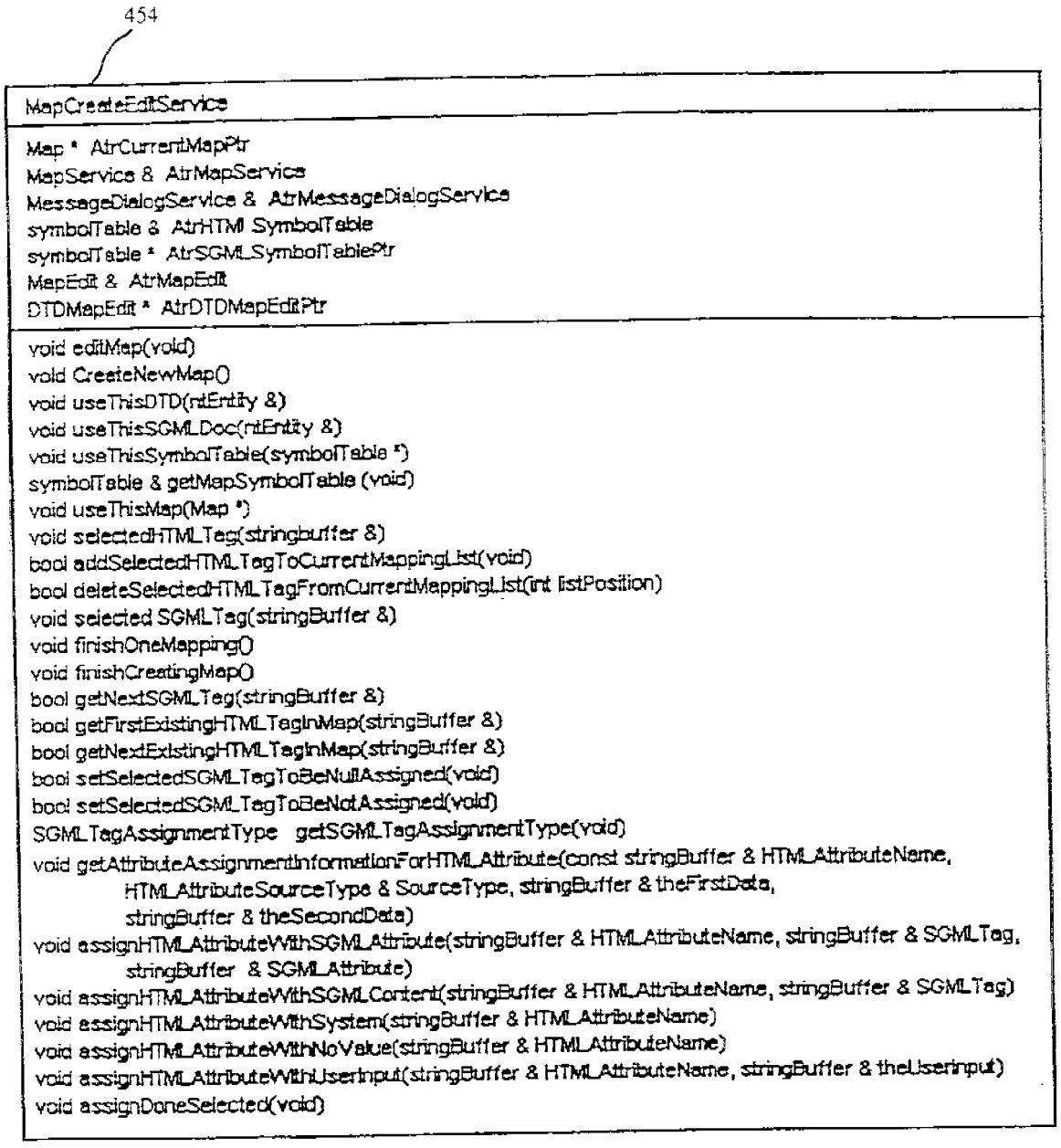

454
MapCreateEditService
Map * AtrCurrentMapPtr
MapService & AtrMapService
MessageDialogService & AtrMessageDialogService
symbolTable & AtrHTMl SymbolTable
symbolTable * AtrSGMLSymbolTablePtr
MapEdit & AtrMapEdit
DTDMapEdit * AtrDTDMapEditPtr
void editMap(void)
void CreateNewMap()
void useThisDTD(ntEntity &)
void useThisSGMLDoc(ntEntity &)
void useThisSymbolTable(symbolTable *)
symbolTable & getMapSymbolTable (void)
void useThisMap(Map *)
void selectedHTMLTag(stringbuffer &)
bool addSelectedHTMLTagToCurrentMappingList(void)
bool deleteSelectedHTMLTagFromCurrentMappingList(int listPosition)
void selected SGMLTag(stringBuffer &)
void finishOneMapping()
void finishCreatingMap()
bool getNextSGMLTag(stringBuffer &)
bool getFirstExistingHTMLTagInMap(stringBuffer &)
bool getNextExistingHTMLTagInMap(stringBuffer &)
bool setSelectedSGMLTagToBeNullAssigned(void)
bool setSelectedSGMLTagToBeNotAssigned(void)
SGMLTagAssignmentType getSGMLTagAssignmentType(void)
void getAttributeAssignmentInformationForHTMLAttribute(const stringBuffer & HTMLAttributeName,
HTMLAttributeSourceType & SourceType, stringBuffer & theFirstData,
stringBuffer & theSecondData)
void assignHTMLAttributeWithSGMLAttribute(stringBuffer & HTMLAttributeName, stringBuffer & SGMLTag,
stringBuffer & SGMLAttribute)
void assignHTMLAttributeWithSGMLContent(stringBuffer & HTMLAttributeName, stringBuffer & SGMLTag)
void assignHTMLAttributeWithSystem(stringBuffer & HTMLAttributeName)
void assignHTMLAttributeWithNoValue(stringBuffer & HTMLAttributeName)
void assignHTMLAttributeWithUserInput(stringBuffer & HTMLAttributeName, stringBuffer & theUserInput)
void assignDoneSelected(void)

Fig. 8C(4)

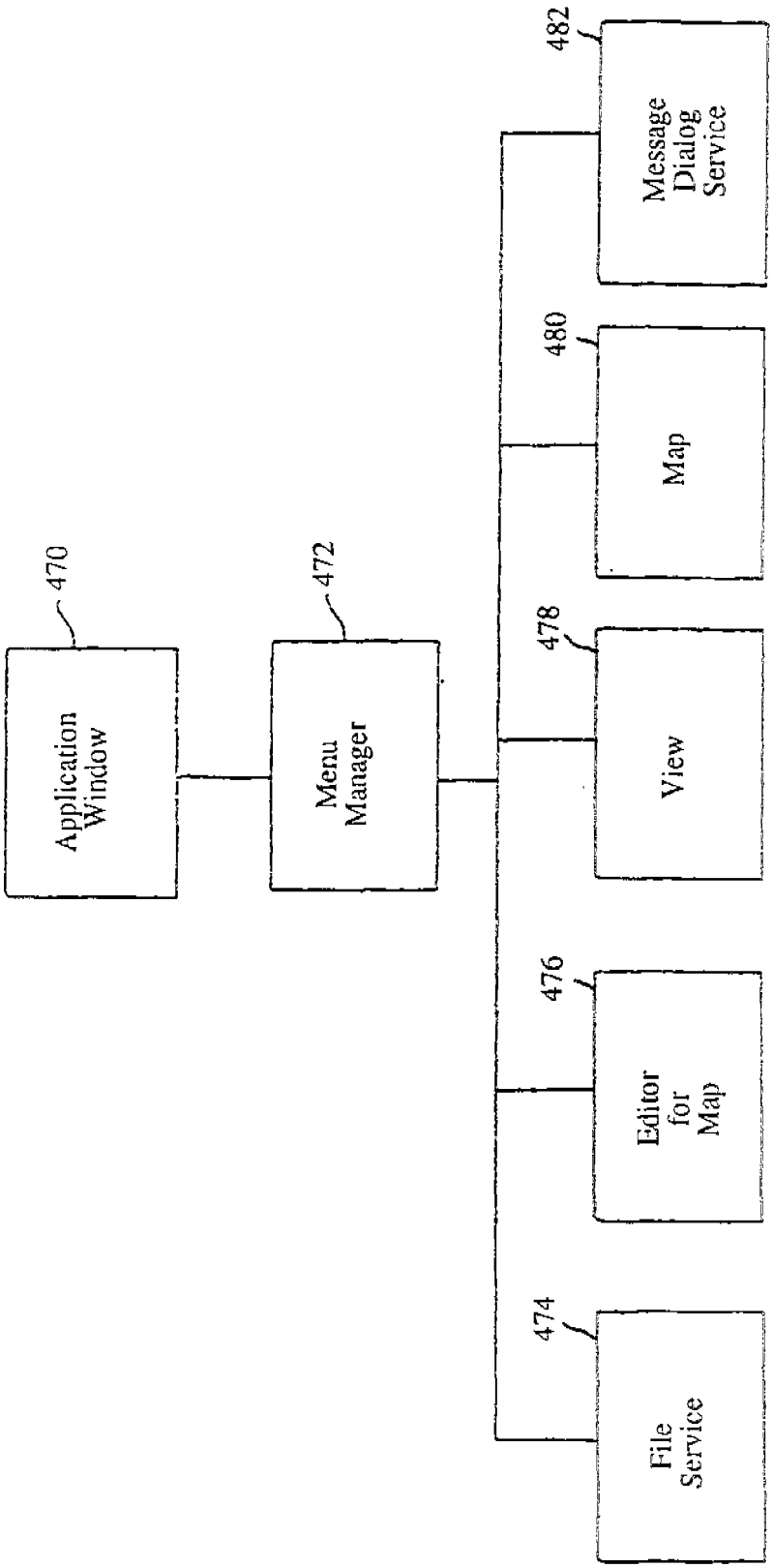

470
Application Window
472
Menu Manager
474
File Service
476
Editor for Map
478
View
480
Map
482
Message Dialog Service

Fig. 9

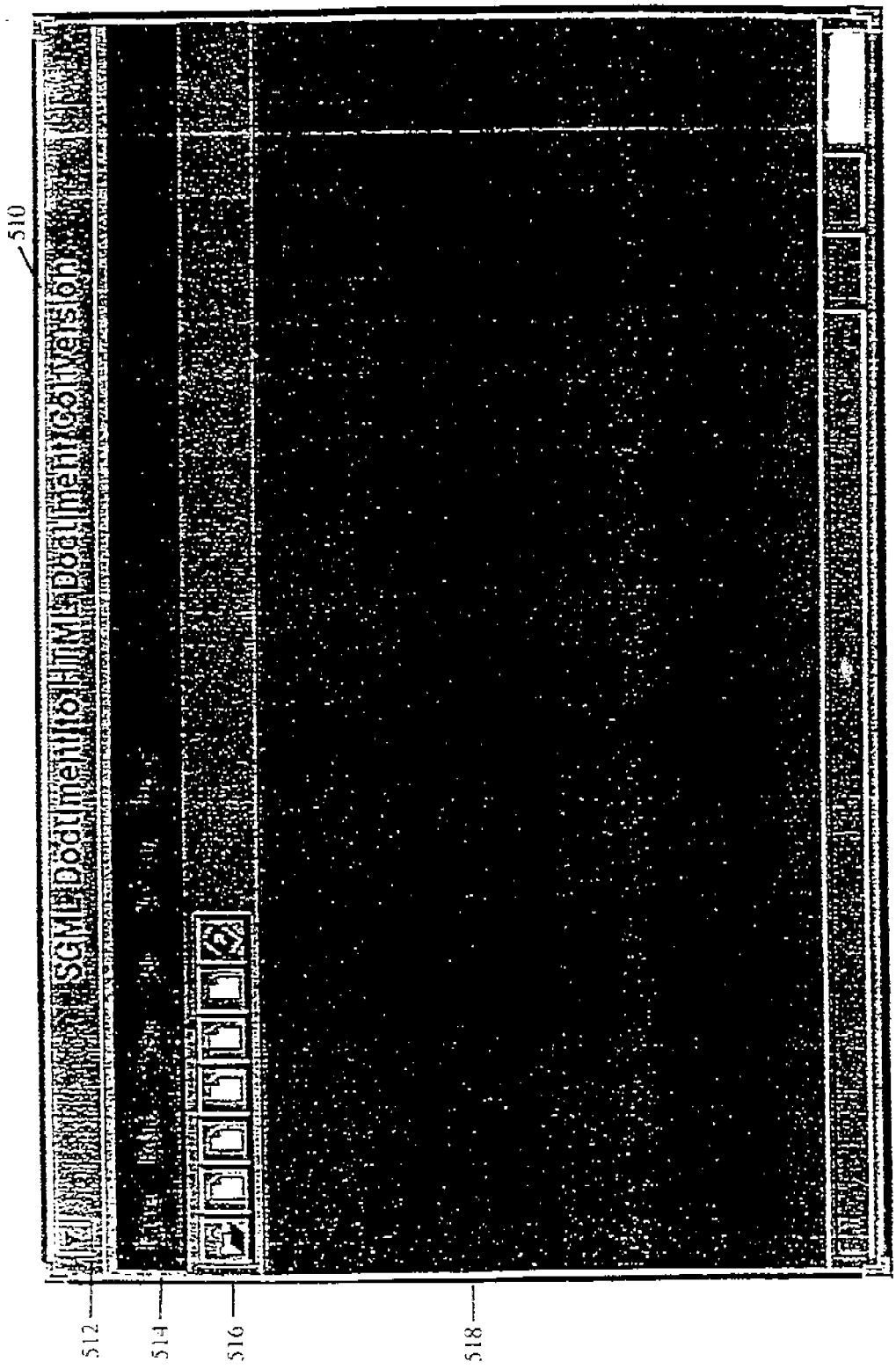


Fig. 10

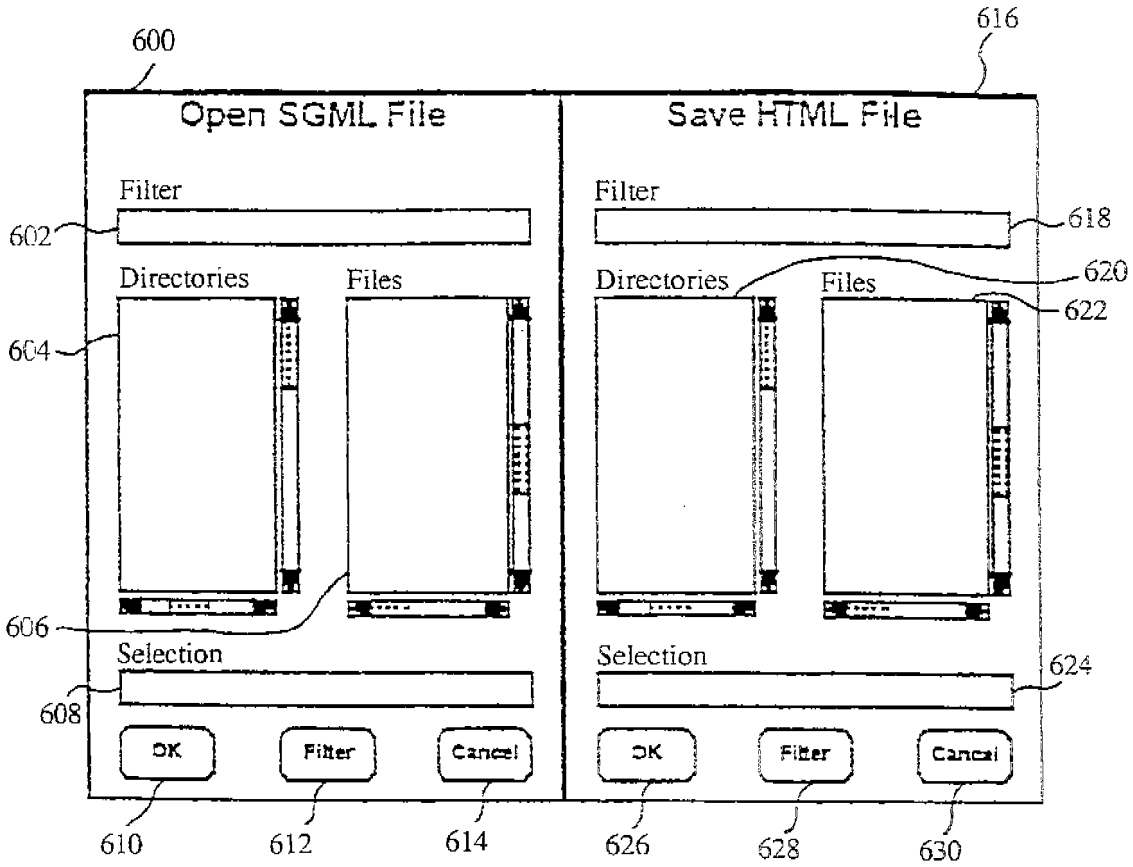

600
616
Open SGML File
Save HTML File
Filter
Filter
602
618
Directories
Files
Directories
Files
620
622
604
606
Selection
Selection
624
608
OK
Filter
Cancel
OK
Filter
Cancel
610
612
614
626
628
630

Fig. 11

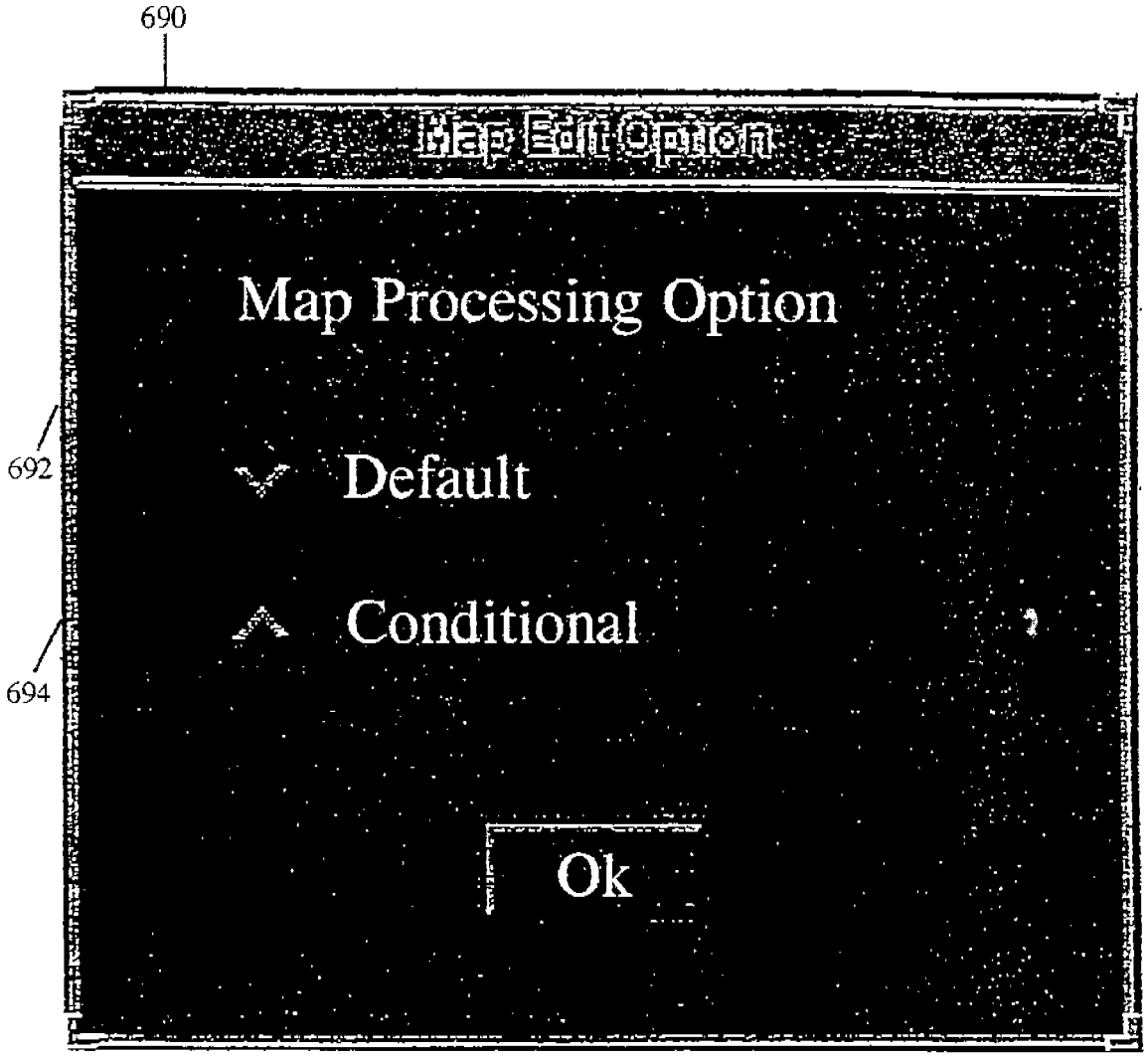

690
Map Edit Option
Map Processing Option
692
Default
Conditional
694
Ok

Fig. 12A

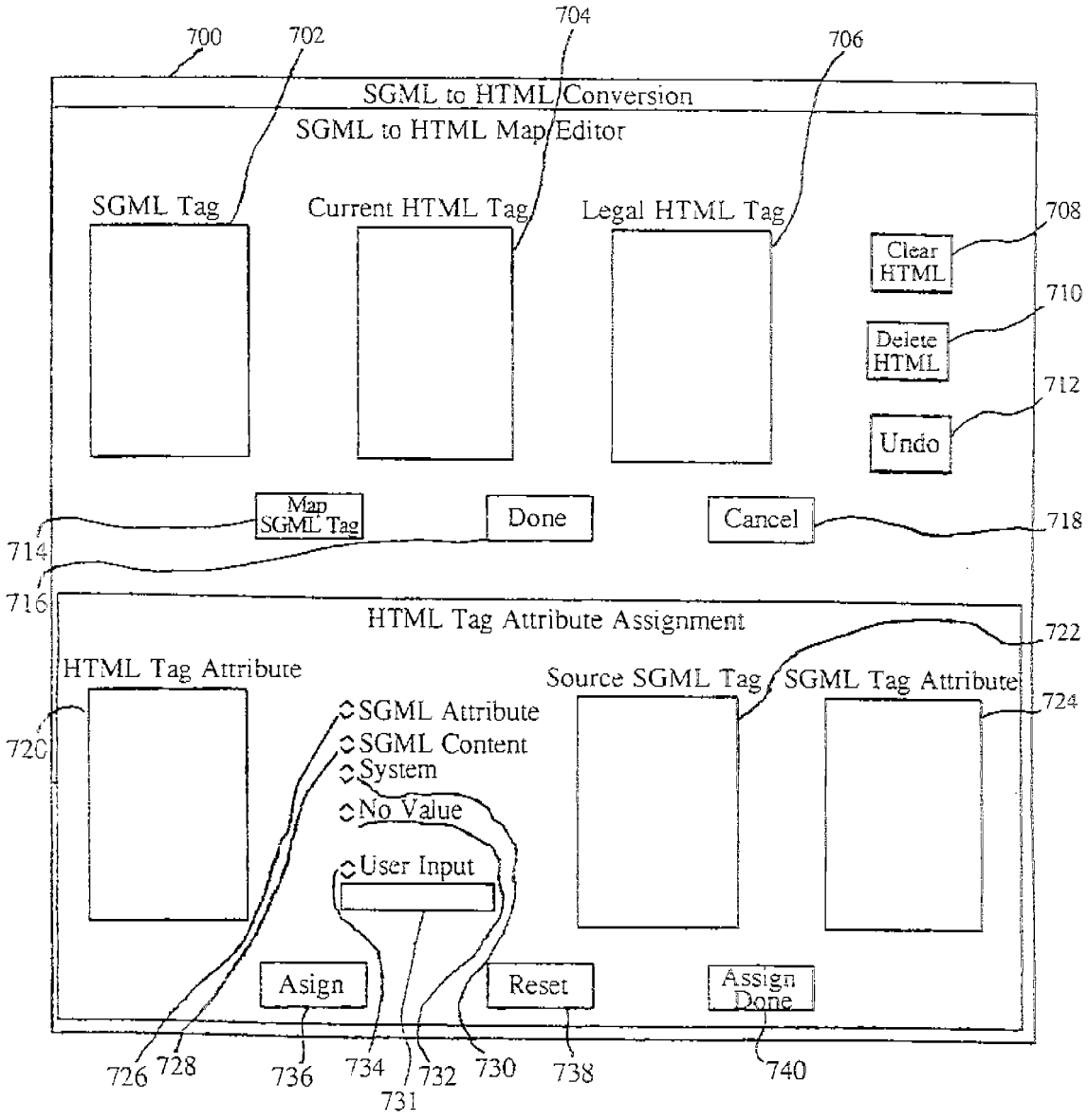

700
702
704
706
SGML to HTML Conversion
SGML to HTML Map Editor
SGML Tag
Current HTML Tag
Legal HTML Tag
Clear HTML
708
Delete HTML
710
Undo
712
Map SGML Tag
Done
Cancel
718
714
716
HTML Tag Attribute Assignment
722
HTML Tag Attribute
Source SGML Tag
SGML Tag Attribute
724
720
SGML Attribute
SGML Content
System
No Value
User Input
Asign
Reset
Assign Done
726
728
736
734
731
732
730
738
740

Fig. 12B

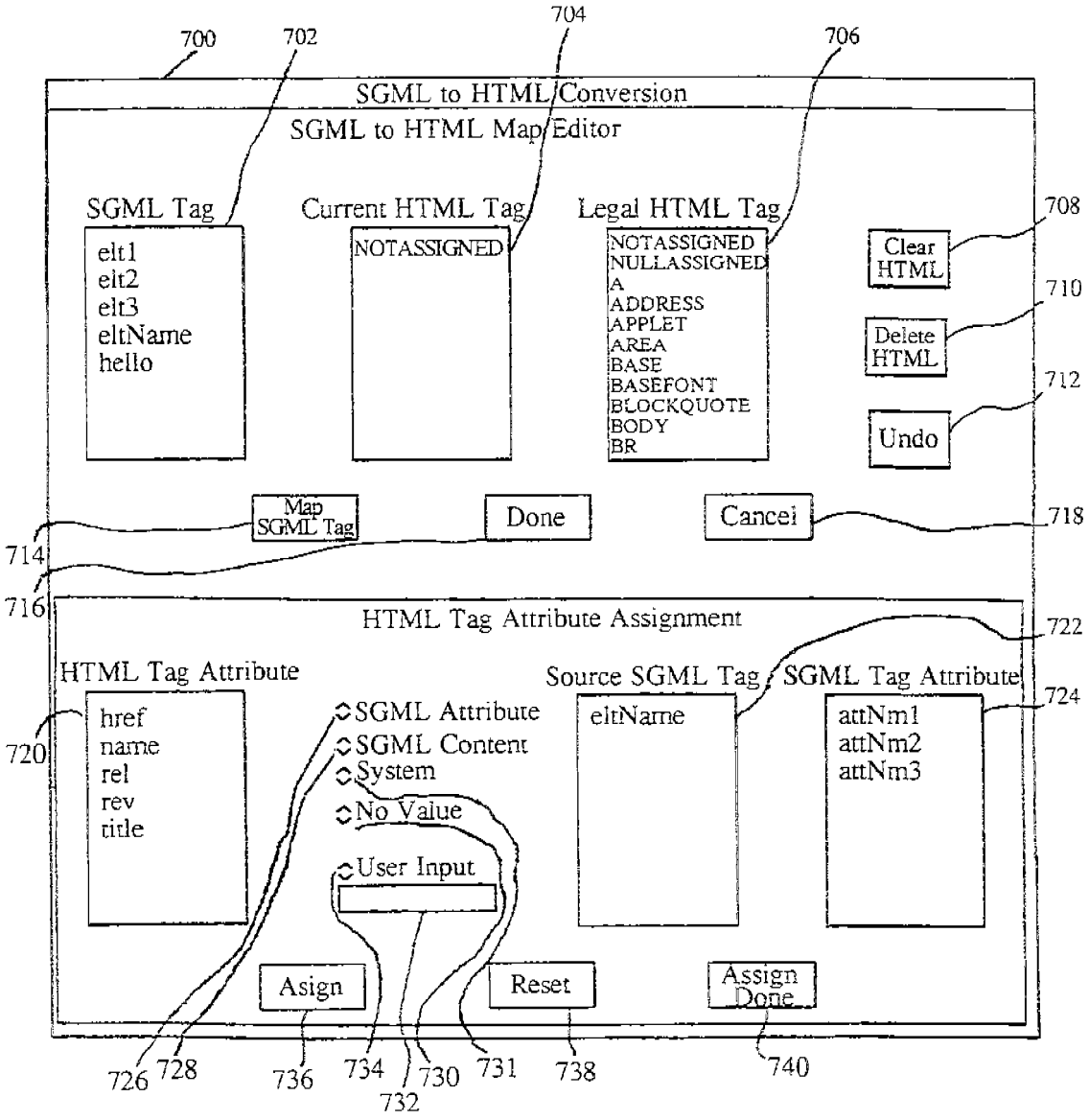

700
702
704
706
SGML to HTML Conversion
SGML to HTML Map Editor
SGML Tag
Current HTML Tag
Legal HTML Tag
elt1
elt2
elt3
eltName
hello
NOTASSIGNED
NOTASSIGNED
NULLASSIGNED
A
ADDRESS
APPLET
AREA
BASE
BASEFONT
BLOCKQUOTE
BODY
BR
Clear HTML
Delete HTML
Undo
708
710
712
Map SGML Tag
Done
Cancel
718
714
716
HTML Tag Attribute Assignment
722
HTML Tag Attribute
Source SGML Tag
SGML Tag Attribute
724
720
href
name
rel
rev
title
SGML Attribute
SGML Content
System
No Value
User Input
eltName
attNm1
attNm2
attNm3
Asign
Reset
Assign Done
726
728
736
734
730
731
738
740
732

Fig. 12C

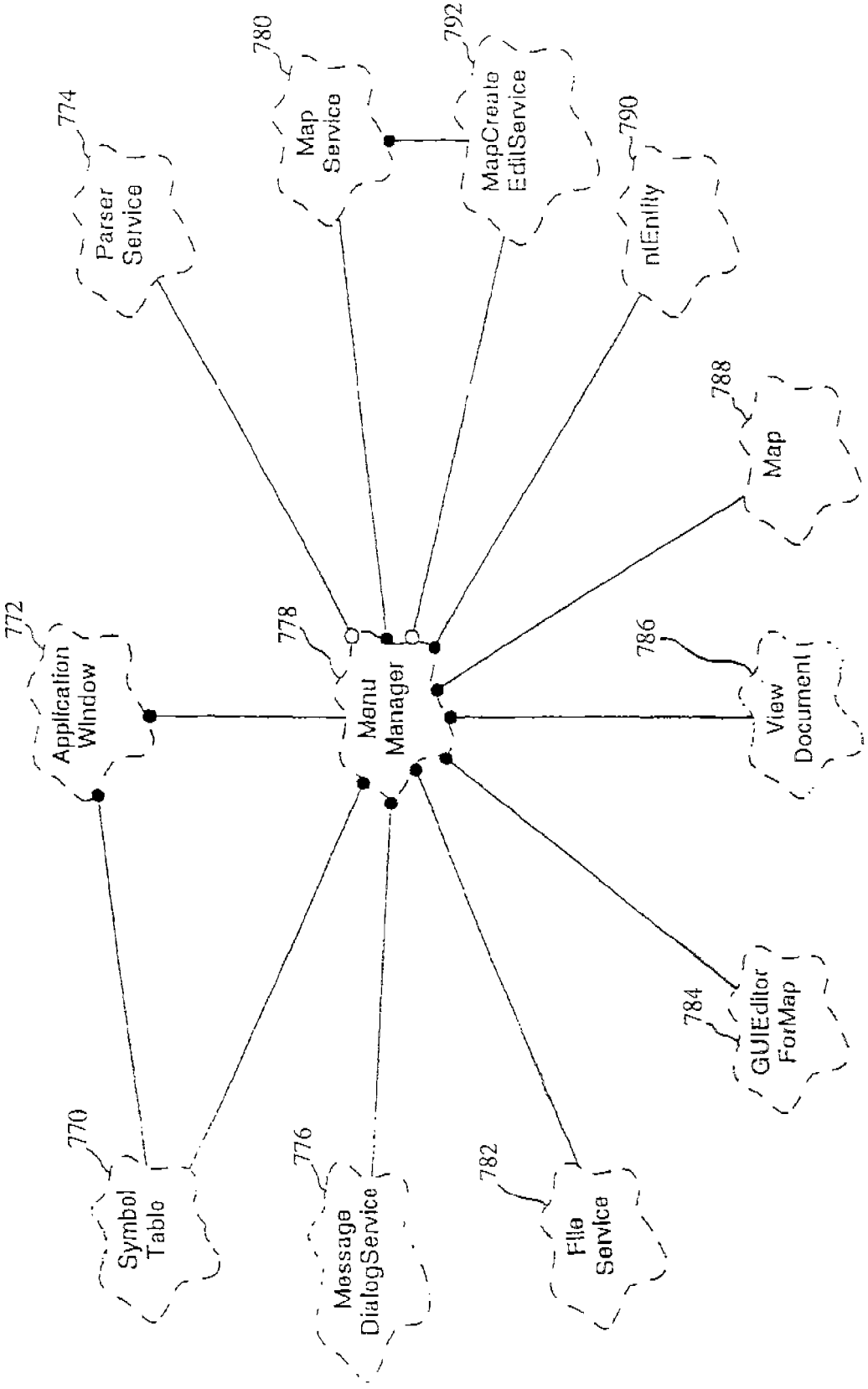

770
Symbol Table
772
Application Window
774
Parser Service
776
Message DialogService
778
Menu Manager
780
Map Service
782
File Service
784
GUIEditor ForMap
786
View Document
788
Map
790
nEntity
792
MapCreate EditService

Fig. 13

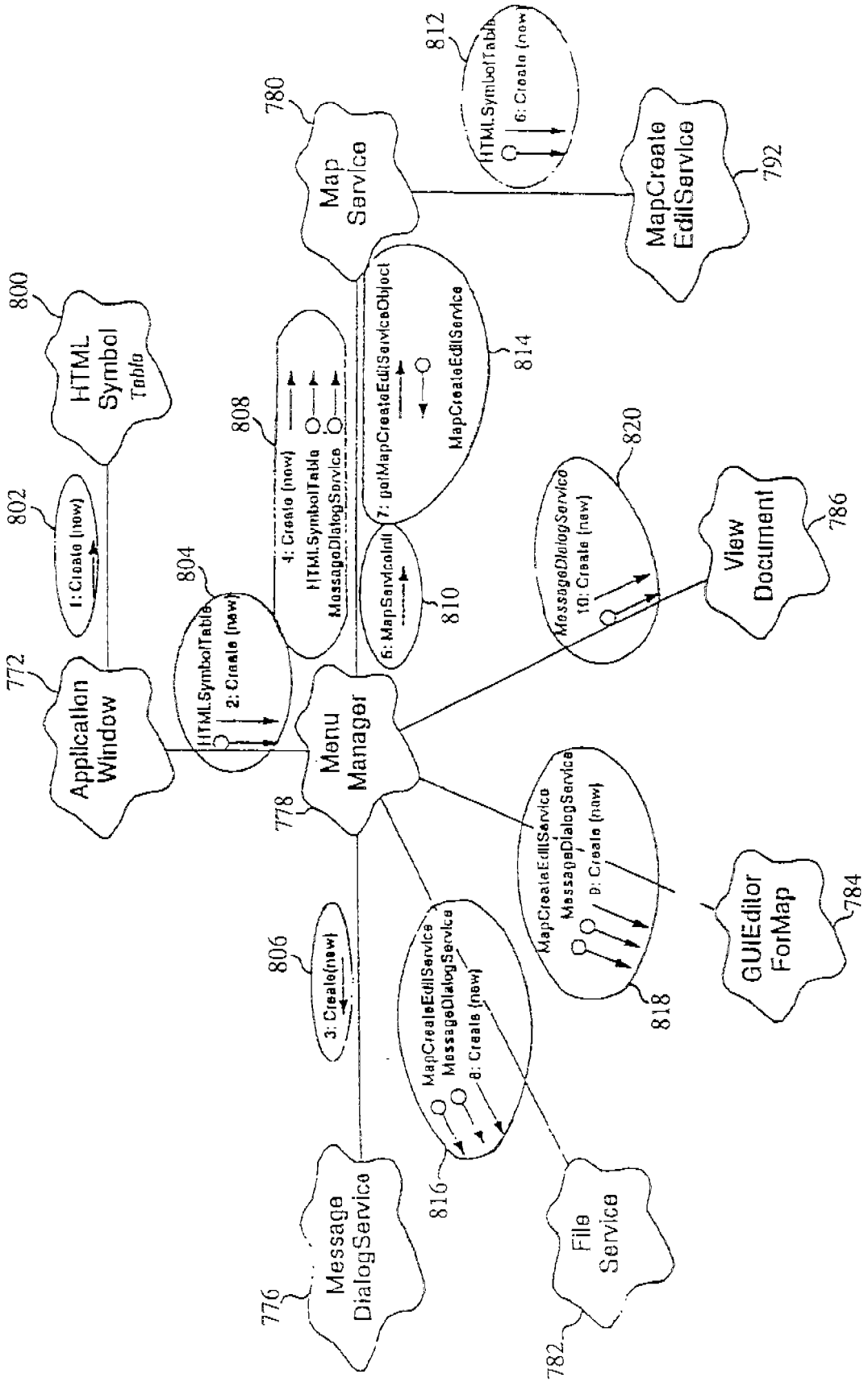

Application Window
772
HTML Symbol Table
800
1: Create (new)
802
HTMLSymbolTable
2: Create (new)
804
Menu Manager
778
3: Create(new)
806
4: Create (new)
HTMLSymbolTable
MessageDialogService
808
6: MapServiceInit
810
Map Service
780
7: getMapCreateEditServiceObject
MapCreateEditService
814
HTMLSymbolTable
6: Create (new)
812
MapCreate EditService
792
MessageDialogService
10: Create (new)
820
View Document
786
MapCreateEditService
MessageDialogService
9: Create (new)
818
GUIEditor ForMap
784
MapCreateEditService
MessageDialogService
8: Create (new)
816
Message DialogService
776
File Service
782

Fig. 14

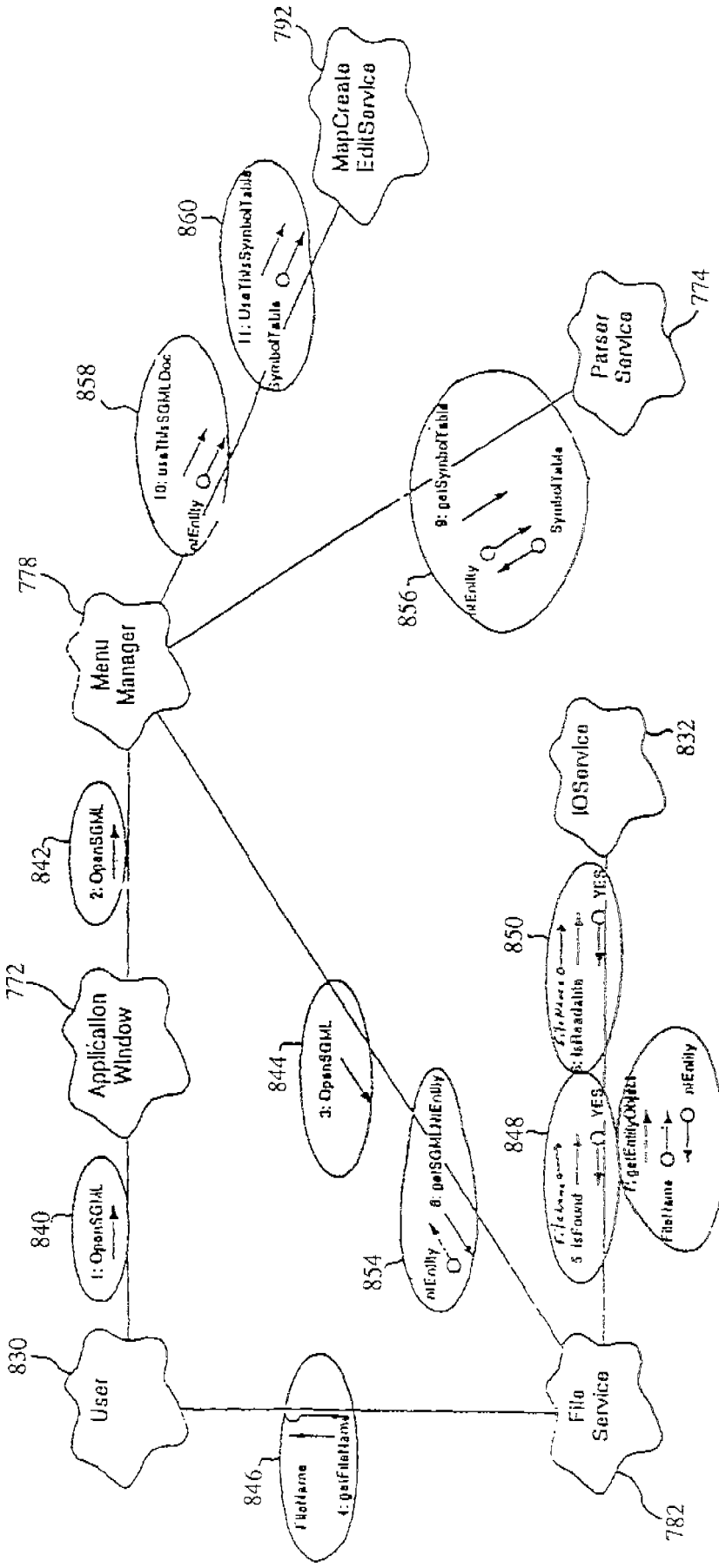

User
830
Application Window
772
Menu Manager
778
File Service
782
MapCreate EditService
792
Parser Service
774
IOService
832
1: OpenSGML
840
2: OpenSGML
842
3: OpenSGML
844
4: getFileName
846
5: IsFound
YES
6: getSGMLNtEntity
854
8: getSymbolTable
856
9: getSymbolTable
SymbolTable
10: useThisSGMLDoc
858
11: UseThisSymbolTable
860
FileName
ntEntity
FileName
ntEntity
848
852
850
YES
IsReadable
FileName
ntEntity

Fig. 15

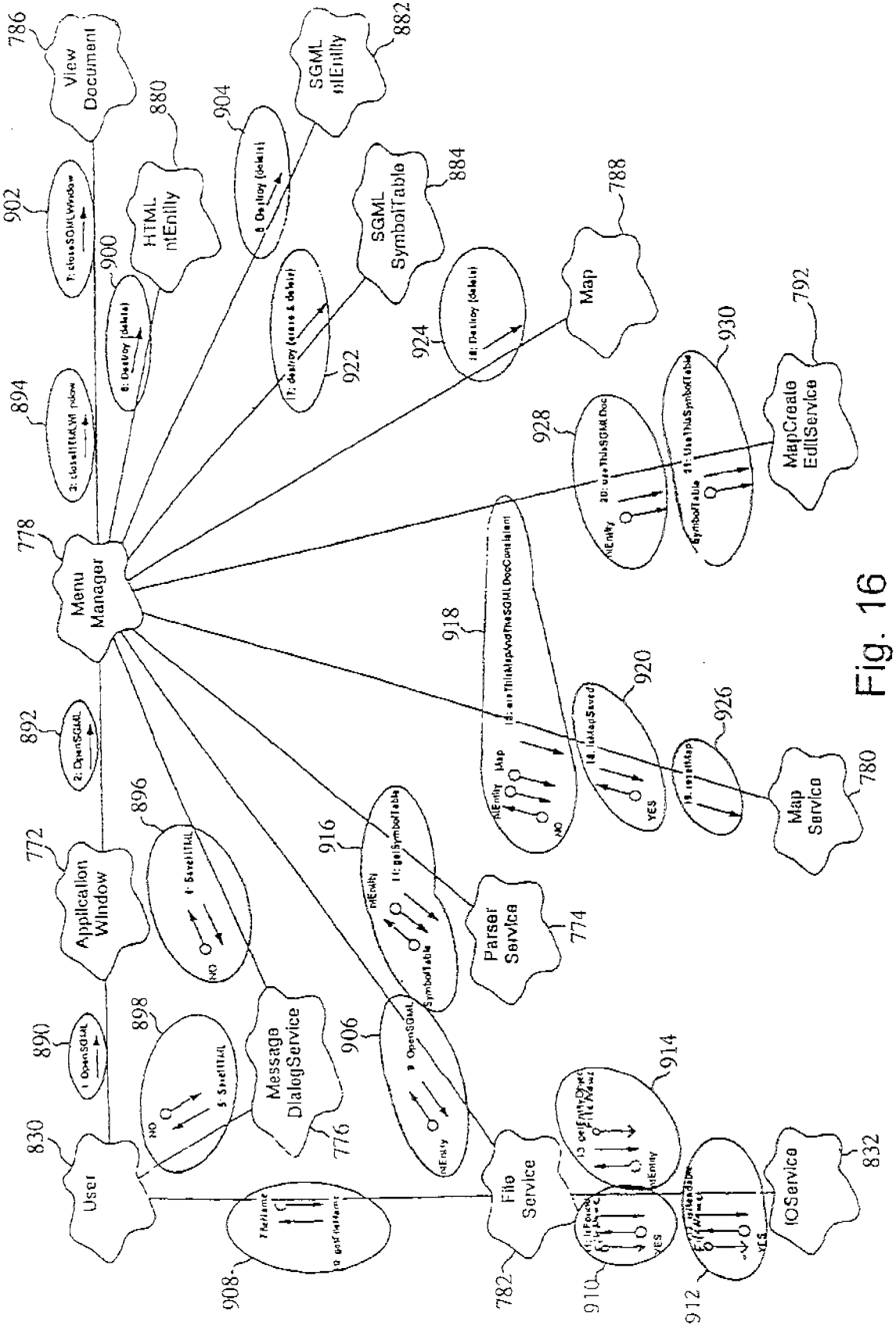

View Document
786
HTML ntEntity
880
SGML ntEntity
882
SGML SymbolTable
884
Map
788
MapCreate EditService
792
Menu Manager
778
Application Window
772
User
830
Message DialogService
776
Parser Service
774
File Service
782
IOService
832
Map Service
780
902
900
904
894
922
924
928
930
918
920
926
892
896
916
890
898
906
914
908
910
912
YES
NO

Fig. 16

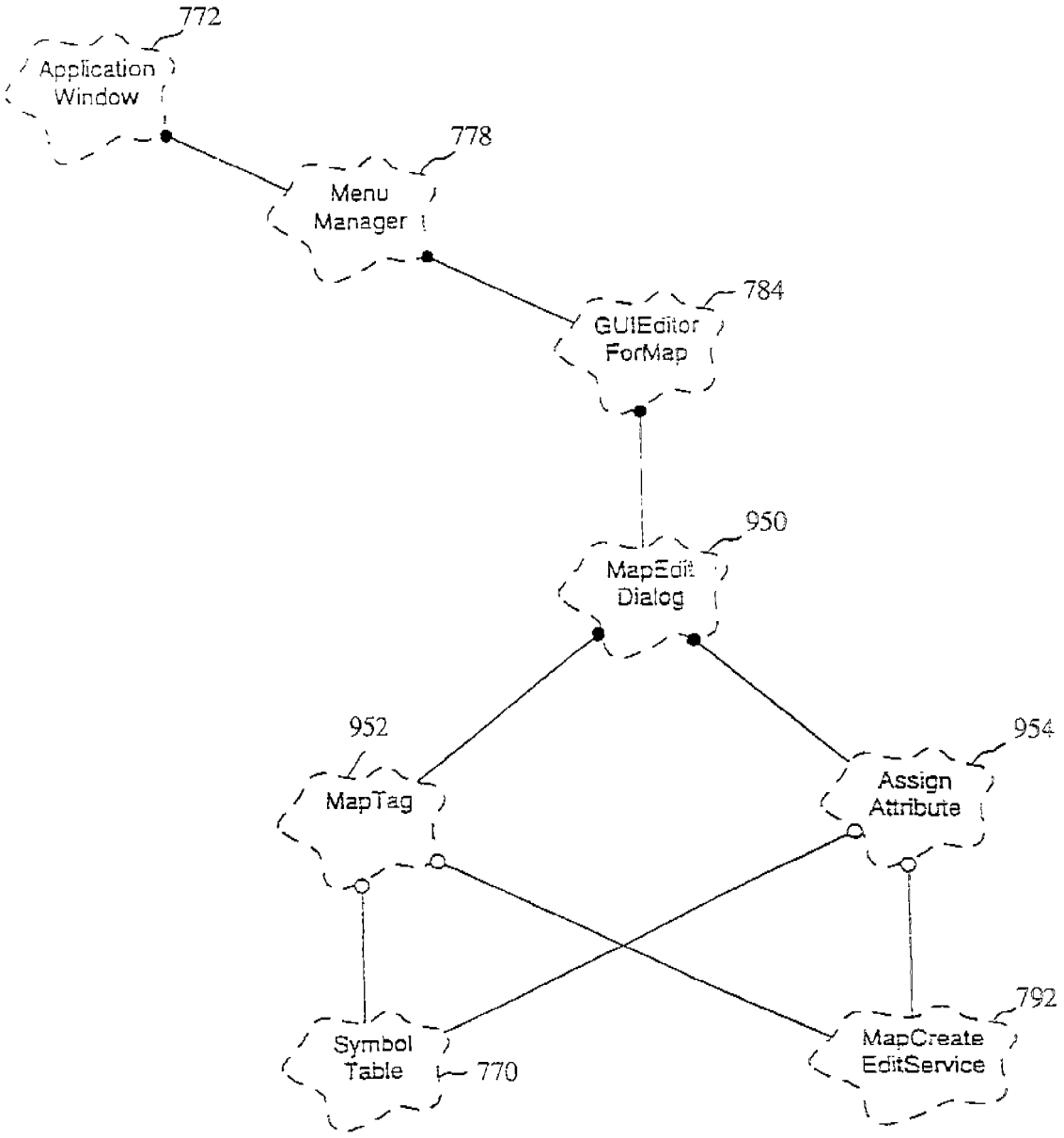

772
Application Window
778
Menu Manager
784
GUIEditor ForMap
950
MapEdit Dialog
952
MapTag
954
Assign Attribute
792
MapCreate EditService
Symbol Table
770

Fig. 17

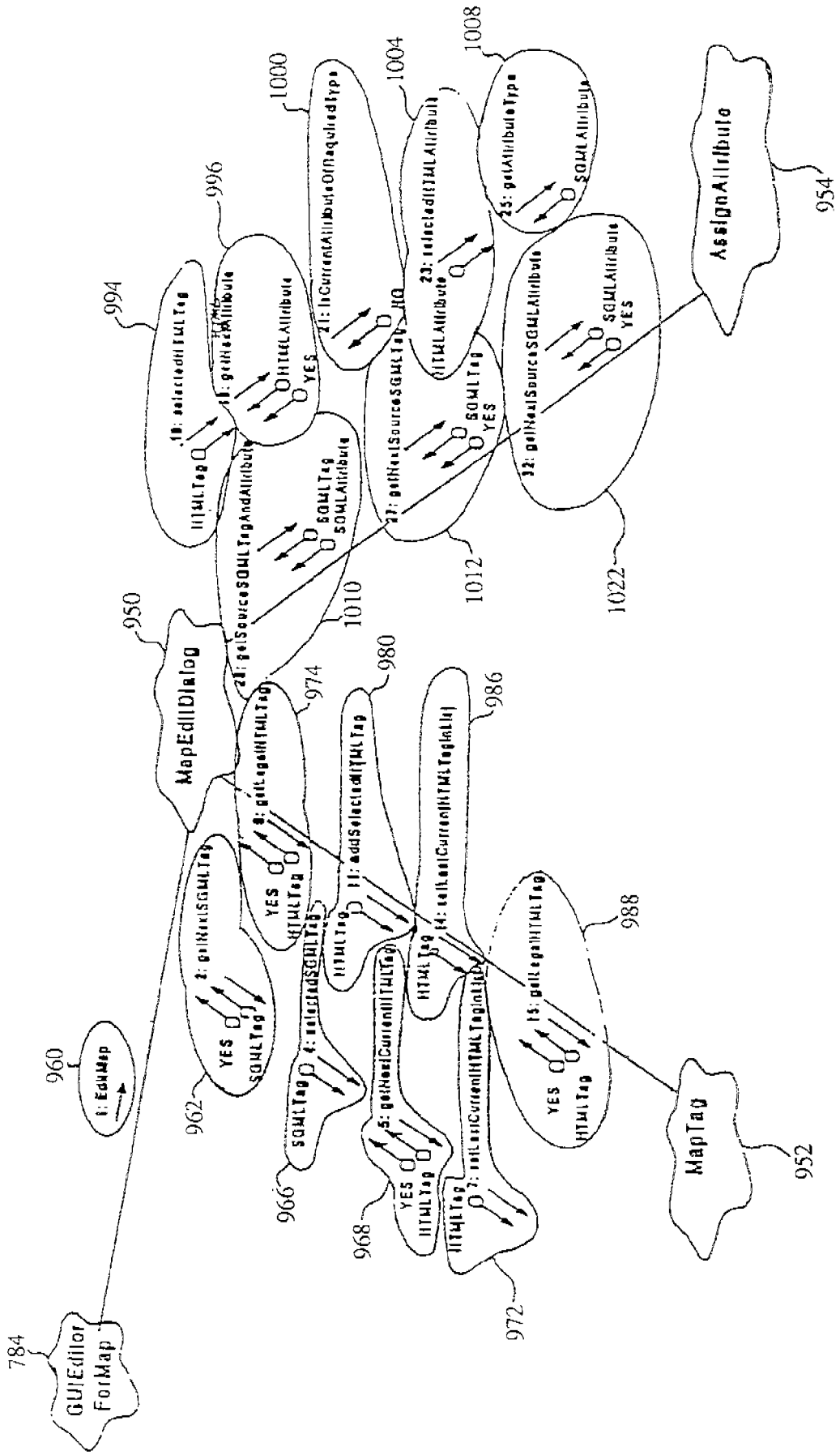

784
GUIEditor ForMap
960
1: EditMap
950
MapEditDialog
962
966
968
972
974
980
986
988
994
996
1000
1004
1008
1010
1012
1022
954
952
MapTag
AssignAttribute
YES
HTMLTag
SGMLTag
HTMLAttribute
SGMLAttribute

Fig. 18A(1)

Fig. 18A(2)

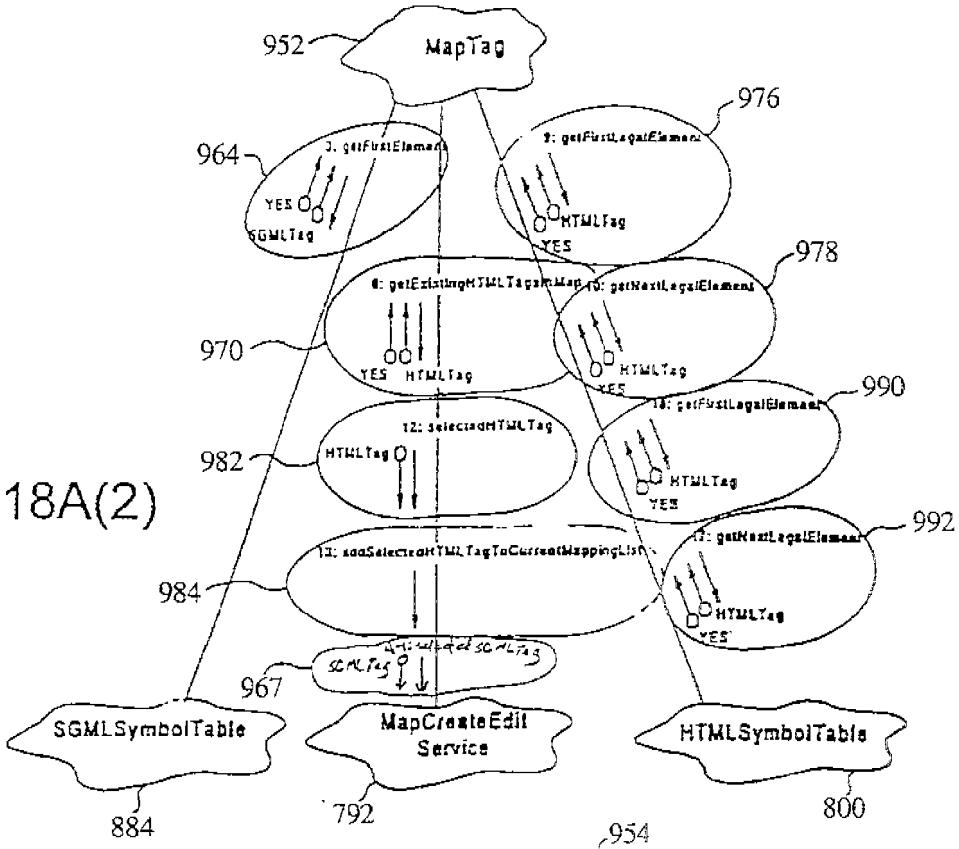
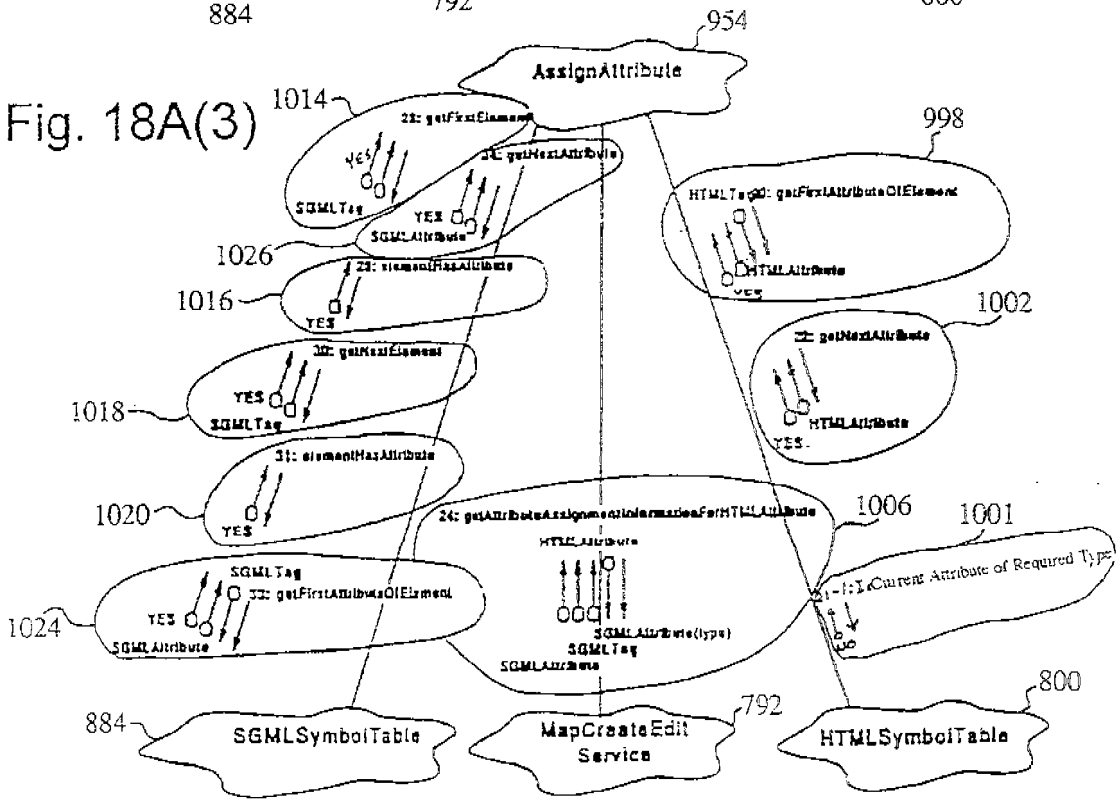


Fig. 18A(3)



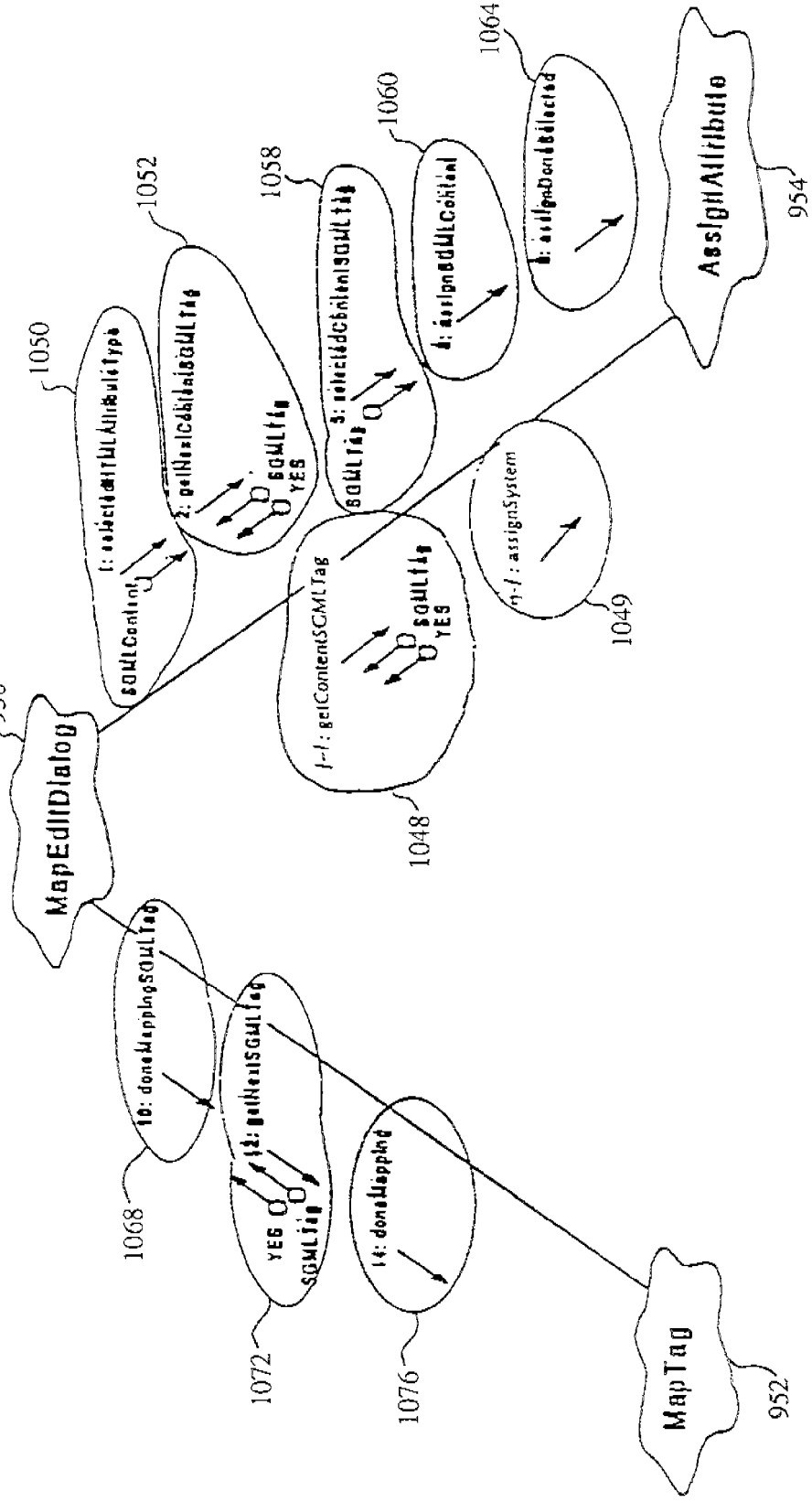

MapEditDialog
950
1050
1052
1058
1060
1064
954
AssignAttribute
1048
1049
1068
1072
1076
MapTag
952
SGMLContent
SGMLTag
YES
getContentSGMLTag
assignSystem
doneMappingSGMLTag
getNextSGMLTag
doneMapping

Fig. 18B(1)

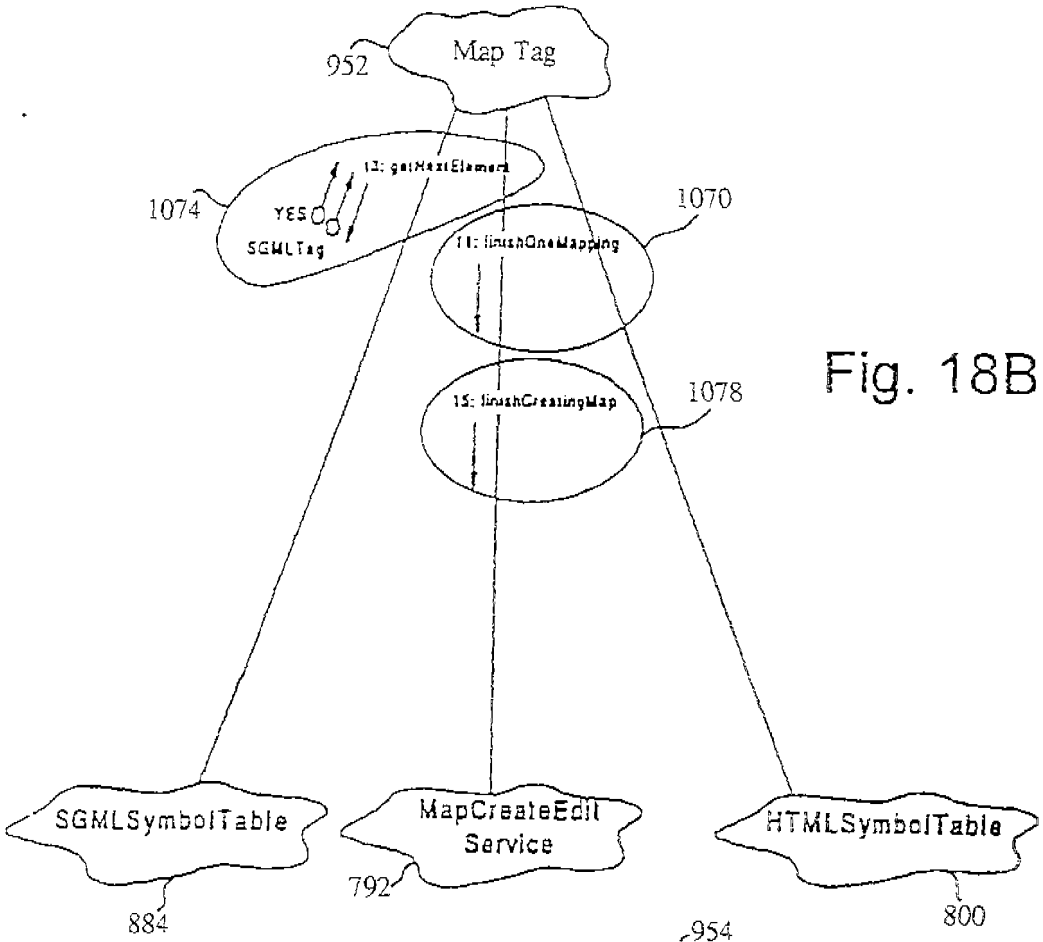


Fig. 18B(2)

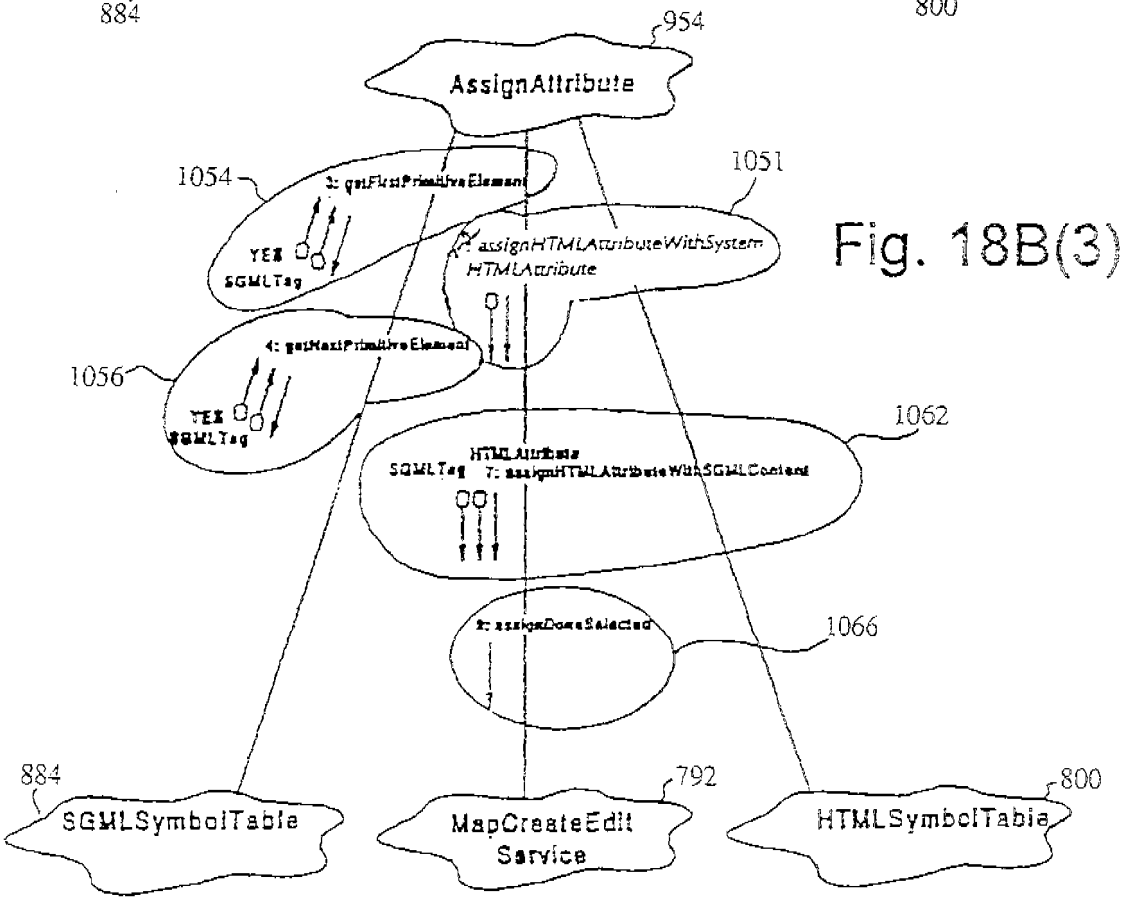


Fig. 18B(3)

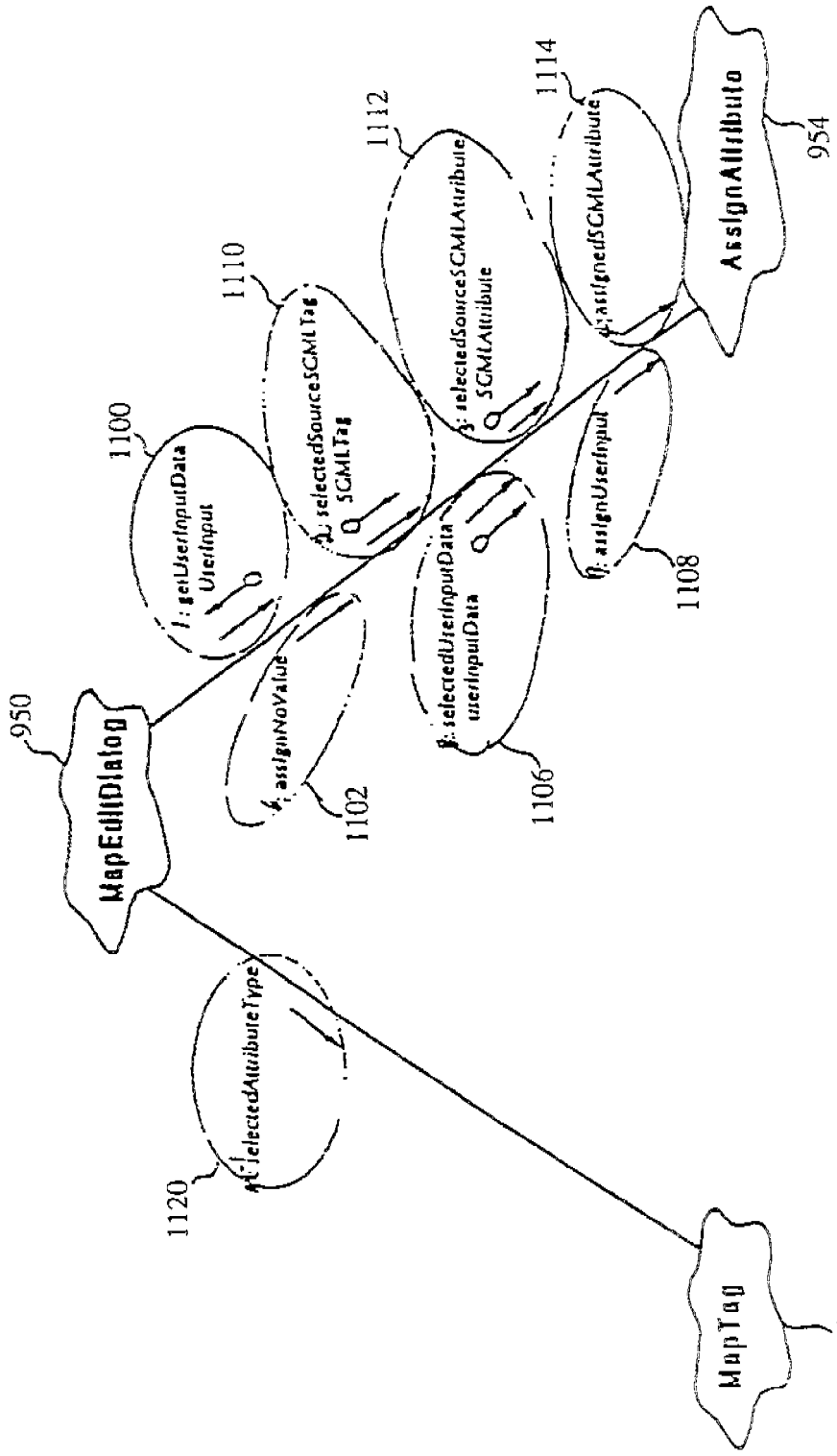

MapEditDialog
950
MapTag
952
AssignAttribute
954
1: getUserInputData
UserInput
1100
2: selectedSourceSGMLTag
SGMLTag
1110
3: selectedSourceSGMLAttribute
SGMLAttribute
1112
4: assignNoValue
1102
5: selectedUserInputData
userInputData
1106
6: assignUserInput
1108
7: assignedSGMLAttribute
1114
8: selectedAttributeType
1120

Fig. 18C(1)

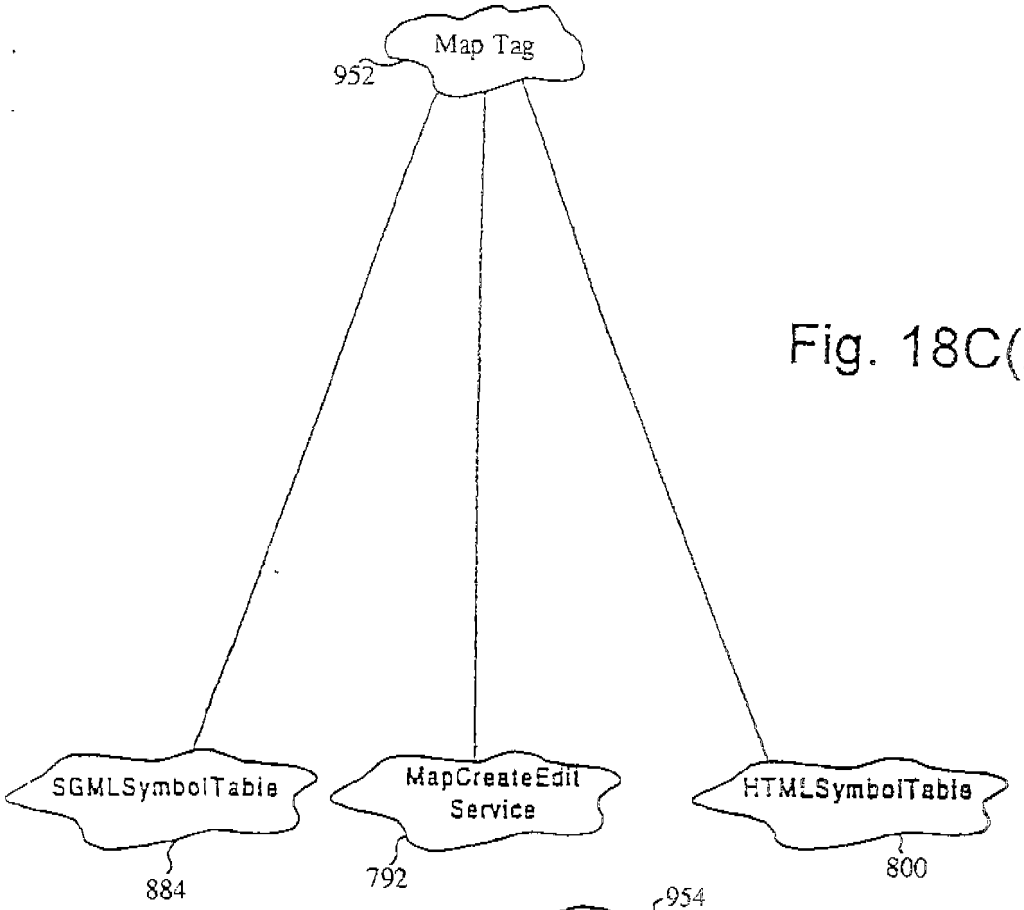


Fig. 18C(2)

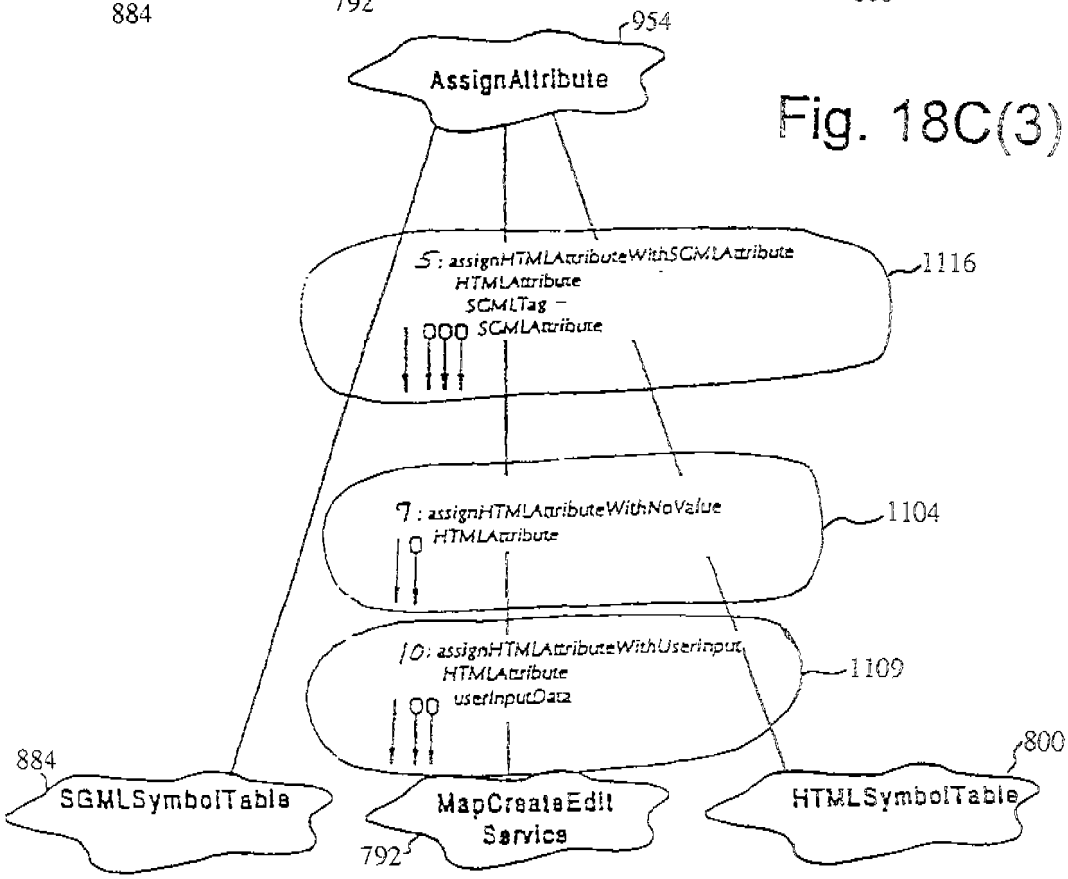


Fig. 18C(3)

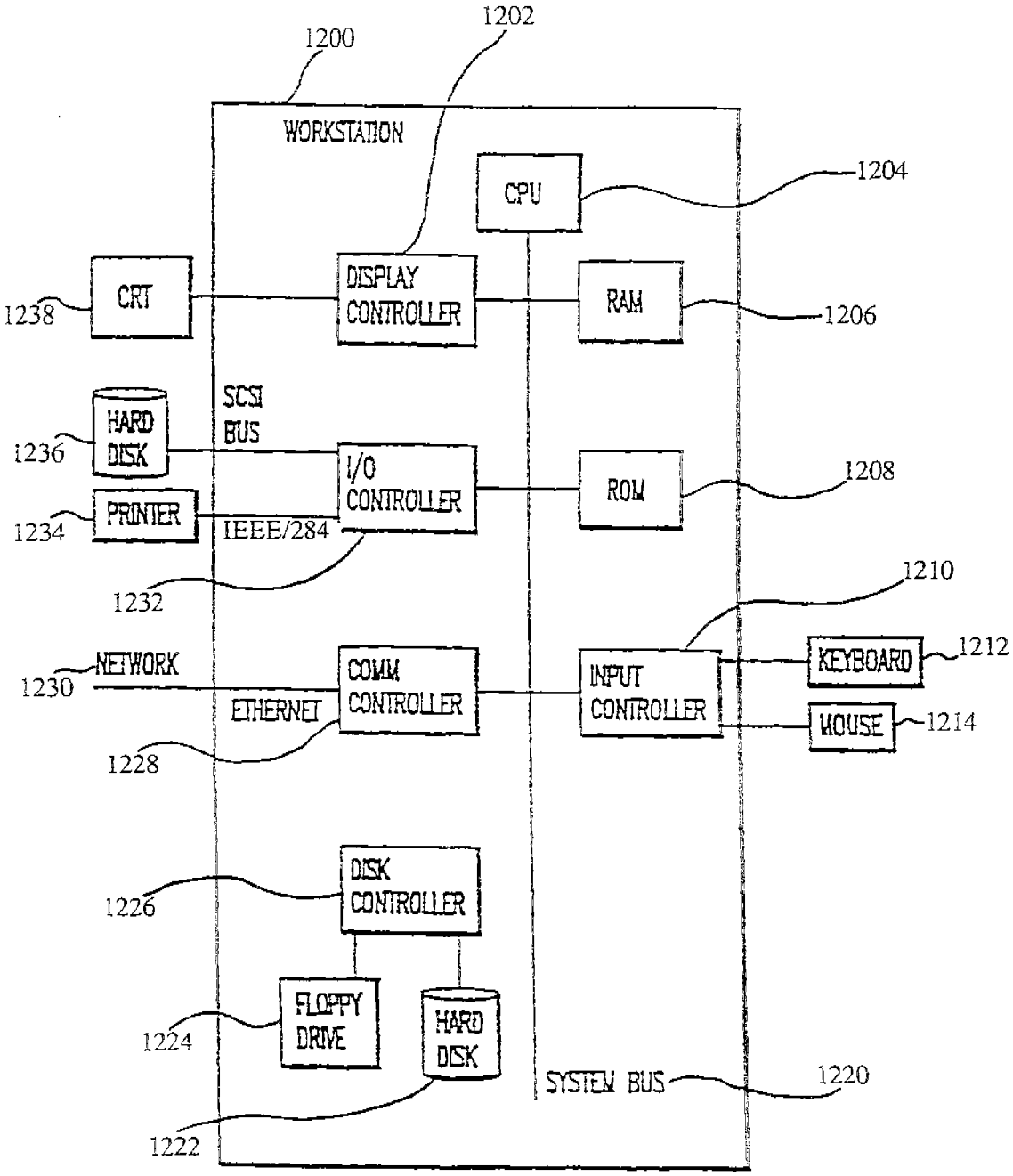

1200
1202
WORKSTATION
CPU
1204
CRT
1238
DISPLAY CONTROLLER
RAM
1206
HARD DISK
1236
SCSI BUS
I/O CONTROLLER
ROM
1208
PRINTER
1234
IEEE/284
1232
1210
NETWORK
1230
COMM. CONTROLLER
INPUT CONTROLLER
KEYBOARD
1212
ETHERNET
MOUSE
1214
1228
DISK CONTROLLER
1226
FLOPPY DRIVE
1224
HARD DISK
SYSTEM BUS
1220
1222

Fig. 19

1400 XYZ//font::metric::x-offset::622

Fig. 20A

1420	ownername	->	ownername
1422	ownerdescription	->	ownerdescription
1424	objectname	->	objectname
1426	objectdescription	->	objectdescription
1428	'::'	->	' '
1430	'//'	->	'_'

Fig. 20B

1440 XYZ__font_metric_x-offset_622

Fig. 20C

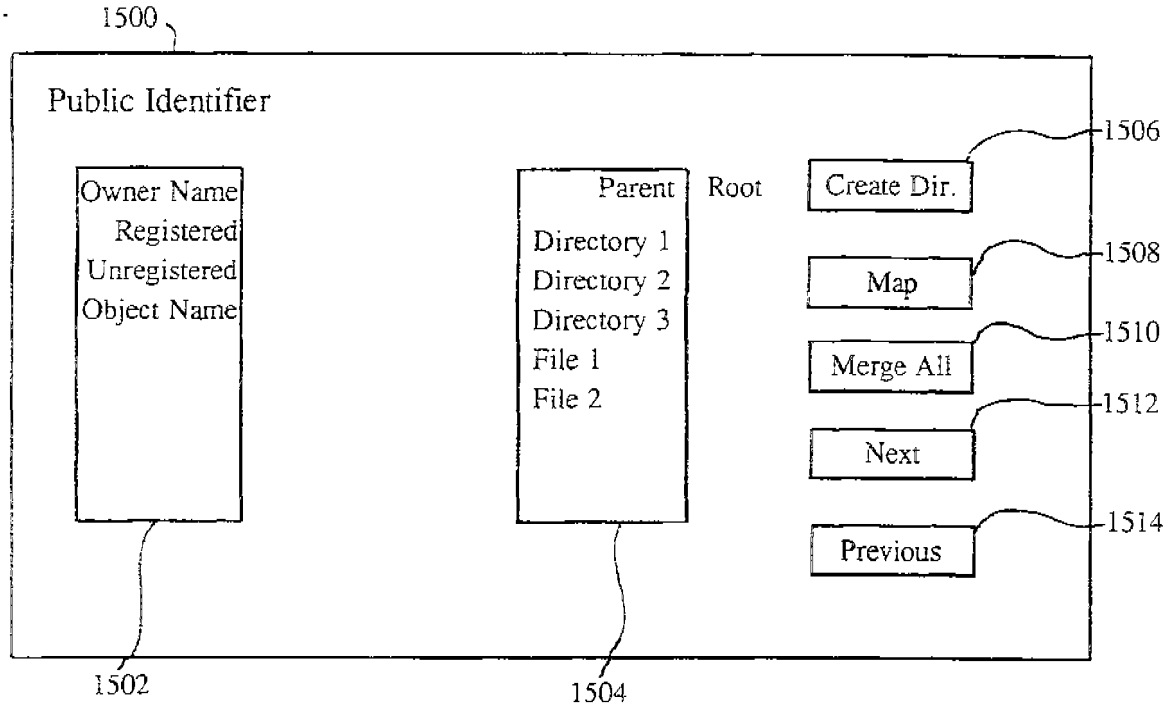


Fig. 20D

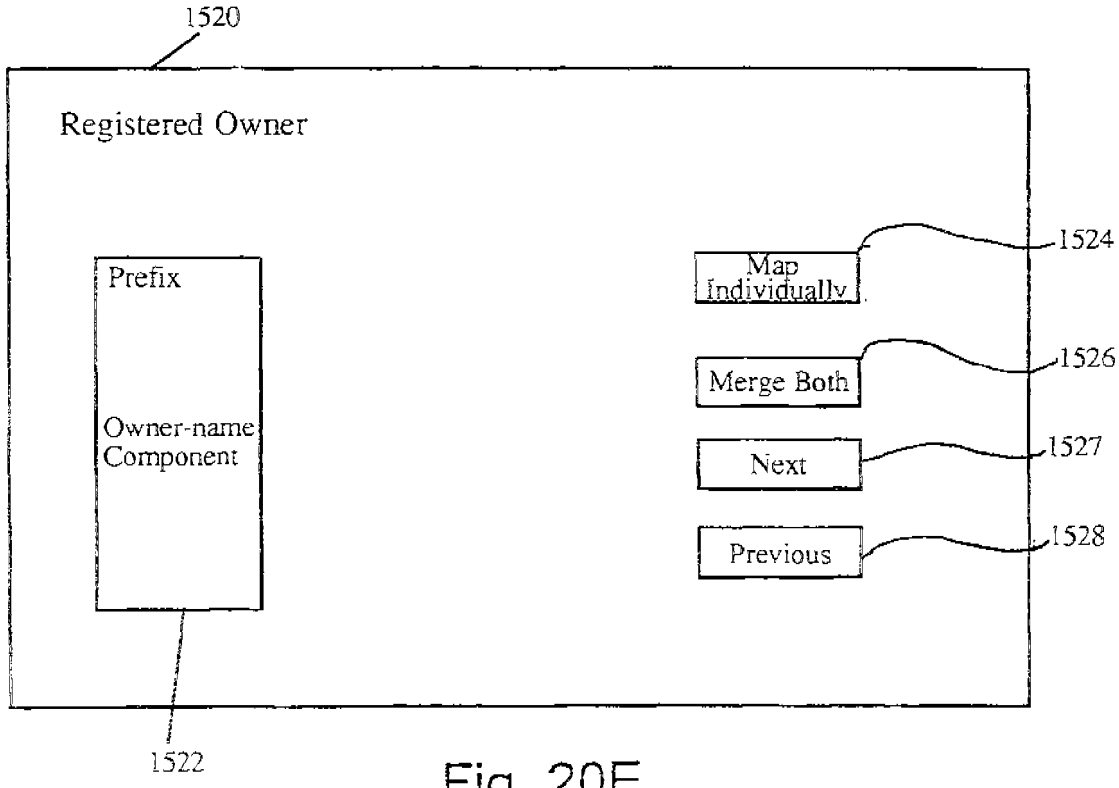


Fig. 20E

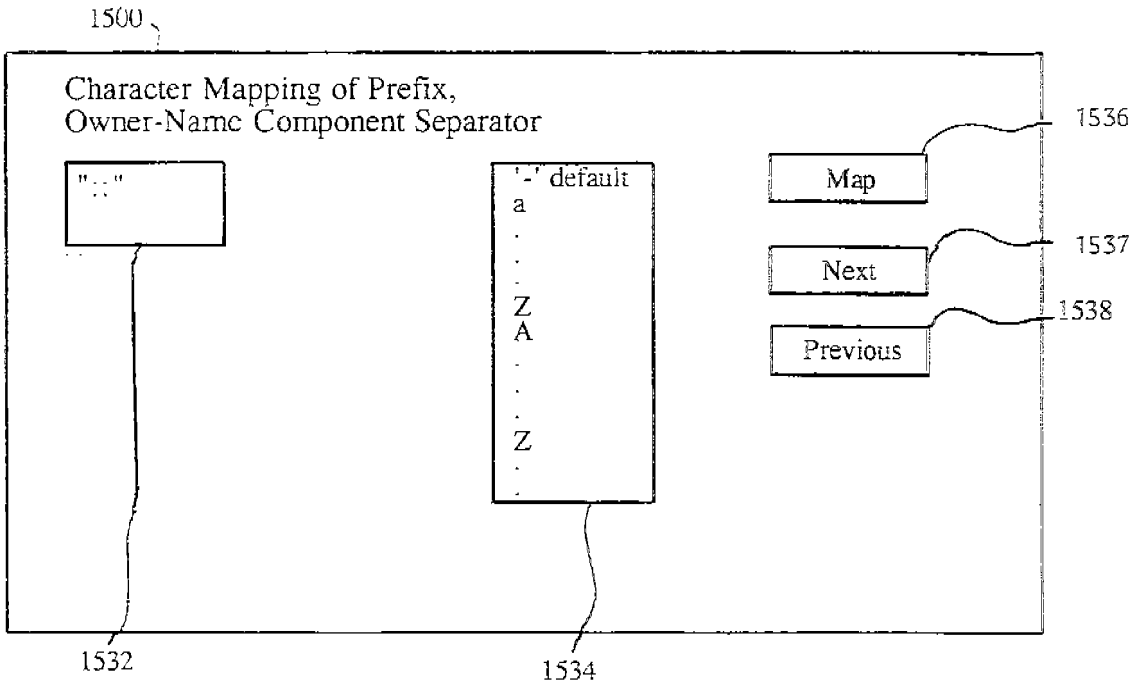

1500
Character Mapping of Prefix,
Owner-Name Component Separator
"."
'-' default
a
Z
A
Z
Map
Next
Previous
1536
1537
1538
1532
1534

Fig. 20F

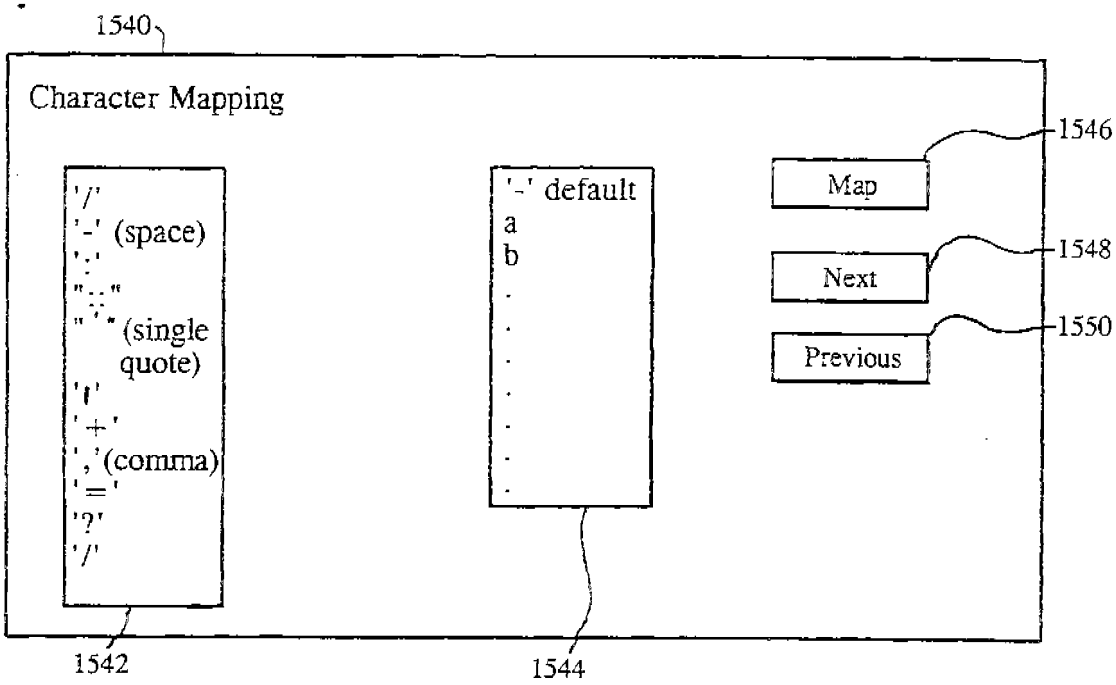


Fig. 20G

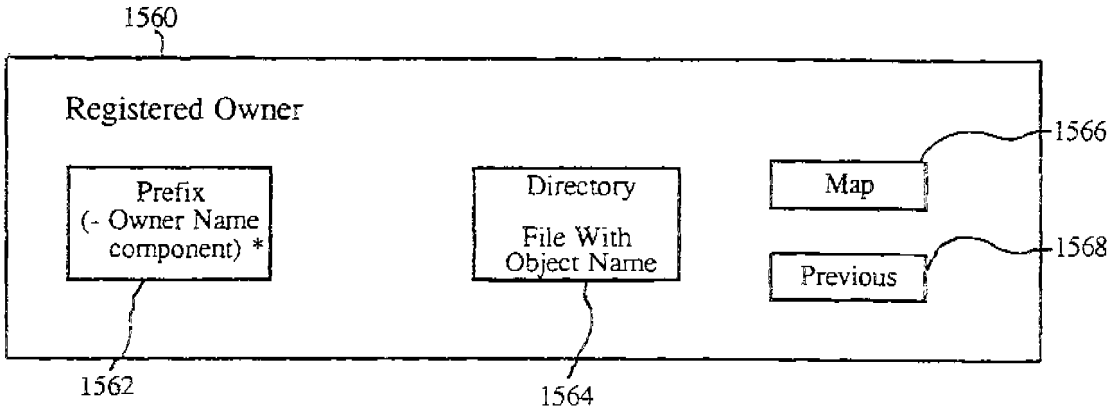


Fig. 20H

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 949 571 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
13.10.1999 Bulletin 1999/41

(51) Int Cl.[6]: G06F 17/22

(21) Application number: 99302718.4

(22) Date of filing: 07.04.1999

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: 07.04.1998 US 80909 P
29.01.1999 US 239295

(71) Applicant: XEROX CORPORATION
Rochester, New York 14644 (US)

(72) Inventors:
- Bickmore, Timothy W.
Somerville, Massachusetts 02144 (US)
- Girgensohn, Andreas
Menlo Park, California (US)
- Schilit, William N.
Menlo Park, California 94025 (US)
- Sullivan, Joseph W.
San Francisco, California 94107 (US)

(74) Representative: Skone James, Robert Edmund
GILL JENNINGS & EVERY
Broadgate House
7 Eldon Street
London EC2M 7LH (GB)

(54) Document re-authoring systems and methods for providing device-independent access to the world wide web

(57) An automatic re-authoring system and method re-author a document originally designed for display on a desktop computer screen for display on a smaller display screen, such as those used with a PDA or a cellular telephone. The automatic re-authoring system and method input a document to be re-authored and re-authoring parameters, such as display screen size, default font and the like. The automatic re-authoring system and method convert the document into a number of pages, where each page is fully displayable with only at most a minimal amount of scrolling on the display screen of the PDA or cellular phone. At each stage of the re-authoring, a number of different transformations are applied to the original document or a selected re-authored page. The selected re-authored page is the best page resulting from the previous re-authoring stage. The best page at each stage is determined based on the re-authoring parameters and the content of the document being re-authored.

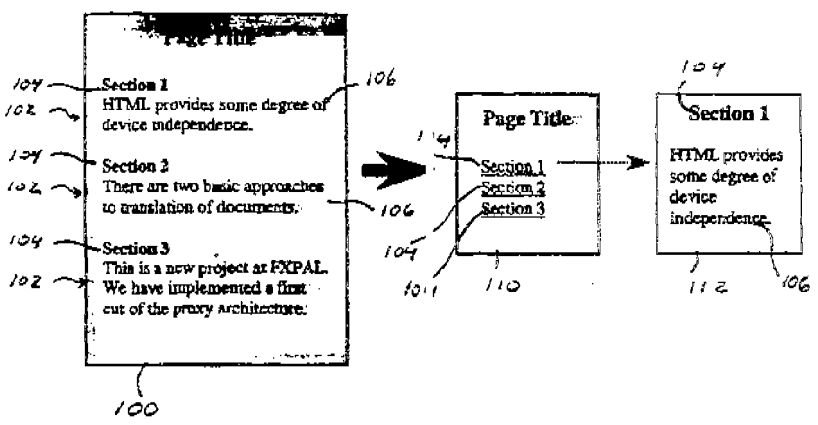


FIG. 1

EP 0 949 571 A2

Description

[0001] This invention is directed to document re-authoring systems and methods that automatically re-author arbitrary documents from the world-wide web to display the documents appropriately on small screen devices, such as personal digital assistants (PDAs) and cellular phones, providing device-independent access to the web.

[0002] Access to world-wide web documents from personal electronic devices has been demonstrated in research projects such as those described in J. Bartlett, "Experience with a Wireless World Wide Web Client", IEEE COMPCON 95, San Francisco, CA, March 1995; S. Gessler et al., "PDAs as Mobile WWW Browsers", Second International World Wide Web Conference. Chicago, IL, October 1994; G. Voelker et al., "Mobisaic: An Information System for a Mobile Wireless Computing Environment", Workshop on Mobile Computing Systems and Applications. Santa Cruz, CA, December 1994; and T. Watson, "Application Design for Wireless Computing", 1994 Mobile Computing Systems and Applications Workshop Position Paper, August 1994. Such access is now a commercial reality. General Magic's Presto! Links for Sony's MagicLink, and AllPen's NetHopper for the Newton and Sharp's MI-10 all provide WWW browsers for PDA class devices, while the Nokia 9000 Communicator and Samsung's Duett provide web access capabilities from cellular phones.

[0003] Unfortunately, most documents on the world-wide web and other distributed networks are designed for display on desktop computers with color monitors having at least 640x480 resolution. Many pages are designed with even larger resolution monitors in mind. In contrast, most PDA class devices and cellular phone displays are much smaller. This difference in display area can lead to a ratio of designed vs. available display area from 4-to-1 to 100-to-1, or greater, making direct presentation of most worldwide web documents on these small devices aesthetically unpleasant, un-navigable, and in the worst case, completely undecipherable. This presents a central problem in accessing world-wide web pages using these small devices: how to display arbitrary web documents, such as HTML documents, that have been designed for desktop systems on personal electronic devices that have much more limited display capabilities.

[0004] Technologies already provide computational mobility and wireless connectivity, but the standard solutions to viewing documents and web pages on tiny screens are to either increase the screen resolution, which is great if the user happens to carry a magnifying glass, or to provide the ability to FAX or print to a local hardcopy device, which is both inconvenient and contradicts the rationale for having electronic documents in the first place. There are five general approaches to displaying web documents on small screen devices: device-specific authoring; multiple-device authoring; client-side navigation; automatic re-authoring; and web page filtering. Device-specific authoring involves authoring a set of web documents for a particular display device, such as, for example, a cellular phone outfitted with a display and communications software, such as the Nokia 9000. The basic philosophy in this approach is that users of such specialty devices will only have access to a select set of services. Thus, the document for these services must be designed up-front for the accessing device's particular display system. Information may be provided from the distributed network at large, but the desired pages must be pre-defined, and custom information extraction and page formatting software must be written to deliver the information to the small device. This is the approach taken in Unwired Planet's UP.Link service, which uses a proprietary mark-up language (HDML).

[0005] In multiple-device authoring, a range of target devices is identified. Then, mappings from a single source document to a set of rendered documents are defined to cover the devices within the identified range. One example of this is the StretchText approach discussed in I. Cooper et al., "PDA Web Browsers: Implementation Issues" University of Kent at Canterbury Computing Laboratory WWW Page, November 1995. In StretchText, portions of the document, potentially down to the word level, can be tagged with a 'level of abstraction' measure. Upon receiving the document, users can specify the level of abstraction they wish to view and are presented with the corresponding detail or lack of detail.

[0006] Another example of multiple-device authoring is HTML cascading style sheets (CSS), as described in H. Lie et al. "Cascading Style Sheets", WWW Consortium, September 1996. In cascading style sheets, a single style sheet defines a set of display attributes for different structural portions of a document. For example, all top-level section headings can be defined to be displayed in red 18-point Times font. A series of style sheets may be attached to a document, each with a weight describing that style sheet's desirability to the document's author. The user can also specify a default style sheet. The browser used by the user to access the distributed network can also define a "default" style sheet. Although the author's style sheets normally override the user's style sheets, the user can selectively enable or disable the author's style sheets, providing the user with the ability to tailor the rendering of the document to the user's particular display.

[0007] In client-side navigation, the user is given the ability to interactively navigate within a single web page by altering the portion of the single web page that is displayed at any given time. A very trivial example of this is the use of scroll bars in the document display area. A much more sophisticated approach is that taken in the PAD++ system, as described in B. Bederson et al., "Pad++: A Zooming Graphical Interface for Exploring Alternate Interface Physics", Proceedings of ACM UIST '94, ACM Press, 1994, in which the user is free to zoom and pan the device display over

the document with infinite resolution. Active Outlining, as described in J. Hsu et al., "Active Outlining for HTML Documents: An X-Mosaic Implementation", Second International World Wide Web Conference, Chicago, IL, October 1994, has also been implemented as a client-side navigation technique, in which the user can dynamically expand and collapse sections of the document under the respective section headings. Other techniques that fall into this category include semi-transparent widgets, as described in T. Kamba et al., "Using small screen space more efficiently", Proceedings, Computer-Human Interactions: CHI 96, Vancouver, BC, Canada, April 1996, and the Magic Lens system, as described in E. Bier et al., "Toolglass and Magic Lenses: The See-through Interface", SIGGRAPH '93 Conference Proceedings 1993.

[0008] Automatic document re-authoring involves developing software that can take an arbitrary document, such as an HTML document, designed to be displayed on a desktop-sized monitor, along with characteristics of the target display device, and re-author the arbitrary document through a series of transformations, so that the arbitrary document can be appropriately displayed on the target display device. This process can be performed either by the client, by the server, or by an intermediary proxy server, such as an HTTP proxy server , that exists solely to provide these transformation services. An example of this latter approach is the UC Berkeley Pythia proxy server, as described in A. Fox et al., "Reducing WWW Latency and Bandwidth Requirements by Real-Time Distillation", Fifth International World Wide Web Conference, Paris, France, May 1996, which performs transformations on web page images. However, the focus of the Pythia proxy server is solely on minimizing page retrieval time. Spyglass Prism is a commercial product that performs automatic re-authoring of HTML documents using fixed transformations associated with page tags or embedded object types. For example, Prism will reduce all JPEG images by 50%.

[0009] Finally, web page filtering lets a user see only those portions of a page that user is interested in. Filtering may be performed on an intermediate server, such as an HTTP proxy server, to conserve wireless bandwidth and device memory. However, filtering could also be performed by the client device as a display-management technique. Filter specifications can be based on keyword or regular expression matching, or on page structure navigation and extraction commands. Filtering can be either specified using visual tools or using a scripting language.

[0010] Each of the five approaches, device-specific authoring, multiple-device authoring, client-side navigation, automatic re-authoring and web page filtering, has specific benefits and drawbacks. Device-specific authoring will typically yield the best-looking results due to the direct involvement of human designer. However, device-specific authoring limits the user's access to a small, select set of documents that have been authored for that specific device. Multiple-device authoring, while requiring less total effort per document than device-specific authoring, still requires significantly more manual design work than simply authoring a single version of a document for a single desktop platform. Client-side navigation will work well if a good set of viewing techniques can be developed. However, client-side navigation requires that the entire document be delivered to the client device at once, which may waste valuable wireless bandwidth and memory. Furthermore, the 'peephole' approach taken in PAD++ seems very awkward to use for large documents, and the active outlining technique has limited applicability, as most web pages do not use a strict section/subsection organization, or use that organization incorrectly.

[0011] Automatic re-authoring is thus the ideal approach to providing broad access to web documents or other web content from a wide range of devices, if automatic re-authoring can be made to produce legible, navigable and aesthetically pleasing re-authored documents without loss of information.

[0012] This invention provides systems and methods that automatically re-author documents designed for a larger display area for display on a smaller display area.

[0013] This invention separately provides systems and methods that automatically transform a document into a plurality of linked subdocuments, where each subdocument requires less display area.

[0014] This invention separately provides systems and methods that automatically apply a plurality of different transforms to an original document to generate a plurality of sets of linked subdocuments.

[0015] This invention further provides systems and methods that automatically apply the plurality of different transforms to at least one of the plurality of sets of linked subdocuments to generate additional linked subdocuments.

[0016] This invention further provides systems and methods that analyze a main subdocument of each set of linked subdocuments to determine a best one of the main subdocuments.

[0017] This invention additionally provides systems and methods that determine if the best main subdocument can be displayed in the smaller display area, and if not, that apply further transforms to that main subdocument to further reduce the required display area.

[0018] This invention separately provides systems and methods that filter a document to extract a desired portion of the document that is displayable in a smaller display area.

[0019] This invention separately provides systems and methods that filter a document to extract a described portion based on a predefined script.

[0020] This invention separately provides systems and methods that generate scripts usable to filter a document to extract a desired portion.

[0021] This invention separately provides a scripting language usable to write scripts for filtering a document to

extract a desired portion.

[0022] In one exemplary embodiment, the document re-authoring systems and methods of this invention are implemented on an HTTP proxy that dynamically re-authors requested web pages using a heuristic planning technique and a set of structural page transformations to achieve the best-looking document for a given display size. The automatic document re-authoring according to the systems and methods of this invention can be performed either by the client, by the server, or, in one exemplary embodiment, by an intermediary HTTP proxy server that exists solely to provide these transformation services. Additionally, the automatic document re-authoring systems and methods according to this invention can be performed on a combination of these devices.

[0023] The automatic document re-authoring systems and methods of this invention work well with displays found in PDAs. However, when the document re-authoring systems and methods of this invention are applied to the very limited displays found on current cellular phones, the document re-authoring systems and methods of this invention sometimes produces pages that are difficult to navigate. When accessing a distributed network, such as the Internet or an intranet, from a cellular phone, most users are mainly interested in accessing very specific information. The document filtering systems and methods of this invention provide those users with manual control in defining the information they would like to be displayed. The document filtering systems and methods of this invention return only a small portion of a page that is easily navigable. The document filtering systems and methods of this invention are ideal in those situations in which the user is monitoring a particular page whose layout is fixed but whose content is changing, since those users can tune the filters to the format of the page.

[0024] The automatic document re-authoring and document filtering systems and methods of this invention provide an automatic document re-authoring capability coupled with document filtering to provide access to arbitrary documents on a distributed network, such as the Internet or an intranet, to devices with limited communications bandwidth and small displays.

[0025] The automatic document re-authoring and document filtering systems and methods ofthis invention intercept requests for documents from a distributed network and return re-authored versions of the requested documents rather than the original requested documents.

[0026] In the larger context of mobile and ubiquitous computing, the automatic document re-authoring and document filtering systems and methods of this invention provide a key technology for giving users view-mobility over platforms.

[0027] These and other features and advantages of this invention are described in or are apparent from the following detailed description of the preferred embodiments.

[0028] The preferred embodiments of this invention will be described in detail, with reference to the following figures, wherein:

Fig. 1 illustrates re-authoring of a document into a section list page and a number of section pages according to one exemplary embodiment of the document re-authoring systems and methods of this invention;

Fig. 2 illustrates a layout table that can be re-authored into a plurality of linked cells according to one exemplary embodiment of the document re-authoring systems and methods of this invention;

Fig. 3 illustrates how a document can be re-authored into different re-authored states based on applying different transformations according to one exemplary embodiment of the re-authoring systems and methods of this invention;

Fig. 4 illustrates one exemplary embodiment of a control form for supplying display information to the HTTP proxy server according to the document re-authoring system and method of this invention;

Fig. 5 illustrates one exemplary embodiment of re-authoring an exemplary document according to the document re-authoring systems and methods of this invention;

Fig. 6 is a block diagram outlining one exemplary embodiment of the invention in which the document re-authoring systems and methods of this invention are used;

Fig. 7 is a block diagram outlining one exemplary embodiment of the document flow in the document re-authoring systems and methods of this invention;

Fig. 8 is a functional block diagram outlining one exemplary embodiment of a document re-authoring system according to this invention;

Fig. 9 is one exemplary embodiment of the document version search space of the document re-authoring systems and methods of this invention;

Fig. 10 is one exemplary embodiment of an image and the abstract syntax tree generated from that image according to this invention;

Figs. 11A and 11B outline one exemplary embodiment of a method for document re-authoring according to this invention;

Fig. 12 is one exemplary embodiment of a method for performing elision transformation according to this invention;

Fig. 13 is one exemplary embodiment of a method for performing table transformation according to this invention;

Fig. 14 is one exemplary embodiment of a method for performing image reduction transformation according to this

invention;

Fig. 15 is a functional block diagram outlining one exemplary embodiment of a document re-authoring system 600 of this invention including the document filtering according to this invention;

Fig. 16 is one exemplary embodiment of the document flow during document filtering and re-authoring according to this invention;

Fig. 17 shows an exemplary embodiment of using the document filtering systems and methods of this invention to navigate within the abstract syntax tree generated from the image shown in Fig. 10; and

Fig. 18 illustrates further navigation within the abstract syntax tree of Fig. 10 according to the document filtering systems and methods of this invention.

[0029] In the following discussion of the document re-authoring and document filtering systems and methods of this invention, the terms "web page", "web document" and "document" are intended to encompass any set of information retrieved as a single entity from a distributed network, such as an intranet, the Internet, the World Wide Web portion of the Internet or any other known or later developed distributed network. This information can include text strings, images, tables of text strings and images, links to other web pages and formatting information that defines the layout of the text strings, images, tables and links within the web page.

[0030] There are many possible automatic document re-authoring techniques, which can be categorized along two dimensions: syntactic vs. semantic techniques and transformation vs. elision techniques. Syntactic techniques operate on the structure of the document, while semantic techniques rely on some understanding of the content. Elision techniques basically remove some information, leaving everything else untouched, while transformation techniques involve modifying some aspect of the document's presentation or content. Table 1 illustrates these dimensions, along with examples of each category:

TABLE 1

Examples of different types of automatic document re-authoring techniques		
	Elide	Transform
Syntactic	Section Outlining	Image Reduction
Semantic	Removing Irrelevant Content	Text Summarizing

[0031] In order to gain an understanding of the processes required by an automated document re-authoring system, a study was conducted to assess the characteristics of typical web pages, and to identify candidate re-authoring techniques through the process of re-authoring several web pages by hand.

[0032] A collection of'typical' web pages, the Xerox Corporate web site, was initially selected to focus the study. This collection of 3,188 web pages is representative of a state-of-the-art, professionally-designed web site. A variety of statistics were collected on these pages using a web crawler, to help gain an understanding of the structure and content of a typical page. These statistics generally agree with other, larger-scale studies that have been performed across the entire web.

[0033] Next, a subset of the pages in the Xerox web site was selected for manual re-authoring. A set of pages from the Xerox 1995 Annual Report was selected and converted by hand for display on a Sharp Zaurus PDA with a 320x240 pixel screen. Detailed notes were kept of the design strategies and techniques used.

Some of the design heuristics learned during this process were:

[0034] Keeping at least some of the original images is important to maintain the look and feel of the original document. Common techniques include keeping only the first image, or keeping only the first and last images, i.e., the bookend images, and eliding the rest.

[0035] Section headers, i.e., the H1 - H6 tags in HTML, are not often used correctly. The section headers are more frequently used to achieve a particular font size and style, such as, for example, bold, if the section headers are used at all. Thus, the section headers cannot be relied upon to provide a structural outline for most documents. Instead, documents with many text blocks can be reduced by replacing each text block with the first sentence or phrase of each block, i.e., first sentence elision.

[0036] An initial rule of thumb for images is to reduce all images in size by a standard percentage, dictated by the ratio of the display area that the document was authored for to the display area of the target device. However, images which contain text or numbers can only be reduced by a small amount before their contents become illegible.

[0037] Semantic elision can be performed on sidebars that present information which is tangential to the main concepts presented in a page. Many of the Xerox pages had such sidebars, which were simply eliminated in the reduced

versions.

[0038] Semantic elision can also be performed on images that do not contribute any information to the page, but serve only to improve its aesthetics.

[0039] Pages can be categorized, and then re-authored based on their category. Two examples of these are banners and link tables. Banners primarily contain a set of images and a small number of navigation links, often only one, that serve to establish an aesthetic look, but contain little or no content. When space is at a premium, these can usually be omitted entirely. Link table pages are primarily sets of hypertext links to other pages, and thus contain very little additional content These link table pages can usually be re-formatted into a more compact form that just lists the links in a text block.

[0040] Whitespace, which is taken for granted on a large display, is at a premium on small devices. Several techniques were discovered for reducing the amount of whitespace in a page. Sequences of paragraphs , i.e., HTML "P" tags, or breaks, i.e., HTML "BR" tags, can be collapsed into one such paragraph or break. Lists, i.e., HTML "UL", "OL", and/or "DL" tags, take up valuable horizontal space with their indenting and bullets. These lists can be re-formatted into simple text blocks with breaks between successive items, as described in Cooper et al.

[0041] In conclusion, to perform document re-authoring two things are required: a set of re-authoring techniques, i. e., a set of page transformations, and a strategy for applying the page transformations. Of the techniques used in the manual re-authoring study, those most amenable to codification were the syntactic elision techniques, including section outlining, first sentence elision, and image elision, and the syntactic transformation techniques, including image size reduction and font size reduction. The design strategy learned during the study included a ranking of the transformation techniques, i.e., try this before that, and a set of conditions under which each transformation or combination of transformations should be applied.

[0042] Following the results of the study discussed above, there are two major elements to the document re-authoring systems and methods of this invention: a collection of individual re-authoring techniques that transform documents in various ways; and automated document re-authoring systems and methods that implement a design strategy by selecting the best combination of techniques for a given document/display size pair.

[0043] The Section Header Outlining transform provide a very good method for reducing the required display size for structured documents, such as technical papers and reports. The outlining process is shown in Fig. 1.

[0044] As shown in Fig. 1, the document 100 is converted into a list of sections page 110 and each section is elided into a page 111. That is, the contents 106 of each section 102 of the document 100 is elided from the document 100 and each section header 104 is converted in ɔ a hypertext link. When the hypertext link for any section is selected, the corresponding page 111 of elided content is loaded into the browser. When confronted with multiple section levels (sections, sub-sections, sub-subsections, etc.), there are two approaches to performing the elision. The first approach is full outlining, which works by keeping only the section headers and eliding all content, with the results looking like a table of contents for a book. The second approach is to-level outlining. In the to-level outlining, a cutoff level in the section hierarchy is determined and all content below that level, including lower-level section headers, is elided, but all content above that level is kept.

[0045] Since most pages have text blocks, even when no section headers are present, the First Sentence Elision transform can be a good way of reducing required screen area. In this technique, each text block is replaced with its first sentence, or, altematively, its first phrase up to some natural break point. This first sentence or phrase is also made into a hypertext link to the original text block.

[0046] The Indexed Segment transform first attempts to find page elements that can logically be partitioned, such as ordered or unordered lists, sequences of paragraphs on tables. This transform takes an input page, segments the content into sub-pages by allocating some number of items to each, and builds and prepends an index page to the collection of sub-pages. The Indexed Segment transform then starts filling output pages with these elements in order until each page is "full" relative to the client's display size. If a single logical element cannot fit on a single output page, then the Indexed Segment transform performs a secondary partitioning that partitions text blocks on paragraph or sentence boundaries.

[0047] In the Indexed Segment transform, as much style information as possible is retained for the output elements, by outputting each element embedded within all of its ancestor partitions' HTML tags. The Indexed Segment transform then constructs an index page by copying a section header or first sentence from each element to be output, concatenating the copied portion onto an index page, and creating a hypertext link from each copied portion to the appropriate sub-page. It should be appreciated that the index page itself may also need to be segmented. In the Indexed Segment transform, "Next" and "Previous" navigation links between sequential sub-pages are also added for navigational convenience.

[0048] The Table transform recognizes when a table, i.e., the presentation of information arranged in a rectangular grid, on a page cannot be directly sent to the client. In these cases, the Table transform generates one sub-page per table cell, using a top-down, left-to-right order. Tables nested within tables are processed in the same manner. The Table transform uses heuristics to determine when table columns are being used as "navigational sidebars," which is

a common practice in commercial HTML web pages. In this case, the Table transform moves these cells to the end of the list of sub-pages as these cells tend to carry very little content.

[0049] Fig. 2 shows a nested table, marking tables with thicker borders than table cells. In the table 120 show in Fig. 2, the cell 122 is identified a as sidebar and will be placed after the cell 128. All of the other cells are placed in their natural order. The six portions of the cell 124, such as the subcells 125 and 126, are each placed in their own subpage between the subpages containing the subcells 123 and 127, unless they contain only whitespace.

[0050] As one can see from the example, nested tables and sidebars complicate the processing of tables. This is especially true if the sidebar is part of an inner table. In that situation, the sidebar should be moved to the end of the inner table, rather than to the end of any surrounding tables. In one exemplary embodiment of the document re-authoring systems and methods of this inventory, the sidebars are moved one table at a time and then all table cells are processed at once, rather than grouping the cells by table.

[0051] Images present one of the most difficult problems for automatic document re-authoring, because the decision of whether to keep, reduce, or eliminate a given image should be based on an understanding of the content and role of the image on the page. However, Image Reduction transforms and Image Elision transforms can be applied without content understanding, as long as users are provided a mechanism by which the users can retrieve the original images. In one exemplary embodiment of the systems and methods of this invention, the Image Reduction transform reduces all images in a page by one of a set of pre-defined scaling factors, such as 25%, 50%, and 75%, and making the reduced images into hypertext links that link the reduced images back to the original images.

[0052] In addition to the Image Reduction transform, three Syntactic Elision transforms have also been developed for image, the Elide All transform, the First Image Only transform, and the Bookends transform. In the Elide All transform, all images are elided from the document. In the First Image Only transform, all but the first image are elided from the document. In the Bookends transform, all but the first and last images are elided from the document. The elided images are each replaced with their HTML "ALT" text when it is available. Alternatively, the elided images are each replaced with a standard icon when no ALT text is available. The ALT text or standard icon for each elided image is also made into a hypertext link to that original image.

[0053] In one exemplary embodiment of the document re-authoring systems and methods of this inventory, if screen space is too limited or the client device cannot display images, the images are removed from the document. However, the removed images may be used as anchors for hypertext links via a client-side image map. It should also be appreciated that if such images are removed, the web site represented by the HTML document can be rendered non-navigable. To accommodate this, in one exemplary embodiment of the document re-authoring systems and methods of this inventory, a transform that extracts the hypertext links from such images and formats them into a text list of link anchors is used. The labels for the text list are extracted from the HTML "ALT" tags of the image map, if present, or from part of the Uniform Resource Locator of the link. This transformation preserves links attached to images for navigation when removing the images.

[0054] The overall process of deciding which combination of transforms to apply to a given page for a given client display seems at first to require some form of human artistic ability. However, the automatic document re-authoring systems and methods of this invention capture many of the heuristics used in the manual re-authoring exercise, and do a fairly good job of producing good-looking pages for a given display.

[0055] Individual page transformations are ordered by their desirability. In order to determine which combination of transformations should be applied to a given document, the document re-authoring systems and methods of this invention performs a depth-first search of the document transformation space, using many heuristics that describe preconditions for transformations and combinations of transformations. The depth-first search ensures that a "good enough" version of the document is found by using a combination of the most desirable transformations. Only if the more desirable transformations are not applicable or do not reduce the document enough, are the less favored transformations used.

[0056] The document re-authoring systems and methods of this invention search a document transformation space in a best-first manner. Each state in this search space represents a version of the document, with the initial state representing the original 'as-authored' document. Each state is tagged with a number representing a measure of merit that represents the quality of the document version at that state. The measure of merit, i.e., the evaluation function or value, for each state is a rough estimate of the screen area required to display the entire document as that document exists in that state. A state can be expanded into a successor state by applying a single transformation technique to the re-authored document as it exists in that state.

[0057] At every step in the search process, the most-promising state of the document, i.e., the state with the smallest current display area requirements, is selected and a transformation is applied to transform the document from its current state to a more promising state of the document, if possible. As soon as a state is created that contains a document version that is 'good enough', the search can be halted and that version of the document is returned to the client device for rendering. Alternatively, the search is continued until all content of the original page is contained or represented in a set of good-enough subpages. If the search is exhausted and no document version can be found that is good enough,

then the best document found during the search is returned to the client device for rendering. If there are hard size constraints that are not met by the best document, a more destructive transformation is applied that breaks documents up in the middle of paragraphs.

[0058] Fig. 3 shows how different transformations applied to a document 200 result in different resulting re-authored sub-pages 210, 220 and 230. Depending on the information supplied by the user to the systems and methods of this invention, one of the sub-pages 210, 220 and 230 would be selected as the "best" re-authored page. Then, if further re-authoring is required, for example, to generate good-enough subpages for the content removed from the first sub-page, or if the best sub-page is not yet "good enough", additional transformations could be applied to the subpages resulting from the selected best re-authored sub-page 210, 220 or 230 or to further re-author the selected best re-authored subpage 210, 220 or 230.

[0059] Heuristic information is used in several places by the document re-authoring systems and methods according to this invention, including: the order in which various transformation techniques are applied to a given state; the pre-conditions for each transformation technique; and the determination of when a document version or subpage is 'good enough'. In general, transformations which make minor changes to the document are preferred over those which make more extensive changes. For example, reducing images by 25% is preferable to reducing the images by 75%.

[0060] The pre-conditions for each transformation technique specify the other transformations with which that transformation can be combined. For example, it makes no sense to apply both full outlining and first sentence elision to the same document. The pre-conditions also specify the requirements on the content and structure of the document that the technique is being applied to. For example, the Full Outlining transform should be applied only when there are at least three section headers in the document or sub-page being re-authored. The current condition for 'good enough' is fairly simplistic. That is, the search is stopped when the area required by a document or sub-page is a predetermined multiple of the screen area of the client display. In general, this predetermined multiple is greater than I, and, in one exemplary embodiment, is 2.5. This higher multiple merely assumes that the user doesn't mind scrolling the display a little in one direction.

[0061] When a transformation is applied to a document it can result in the document's contents being split into multiple, smaller "sub-pages", as shown in Fig. 2. However, each of these sub-pages may still be too large to download and display on the client. To address this problem, the document re-authoring systems and methods of this invention keep a list of the sub-pages generated by each sequence of transformations attached to the state representing the resulting document version. Once the good-enough version of the document is selected, which is really only a good-enough version of the *first* sub-page delivered to the client, the list of generated sub-pages for that version is added to a global list of pages to be re-authored. The document re-authoring systems and methods of this invention then re-author each of these to-be-re-authored pages until all of the resulting sub-pages can be delivered to the client. This procedure is shown in pseudocode below, where "reauthor" refers to the best-first re-authoring process described above for a single input page.

```
Digestor(initial_page)
    to_be_reauthored = { initial_page }
    to_deliver = {}
    while(to_be_reauthored != {})
        next_page = pop(to_be_reauthored)
        best_version_state = reauthor(next_page)
        to_deliver.append(best_version_state.page)
        to_be_reauthored.append(best_version_state.sub_pages)
return to_deliver
```

[0062] All re-authored sub-pages are cached as transformed parse trees. As the user navigates a transformed document and requests subpages, the corresponding parse trees are rendered and sent to the client.

[0063] The document re-authoring systems and methods of this invention re-author document by first parsing the document and constructing a parse tree or abstract syntax tree (AST) representation of the document. The document re-authoring systems and methods of this invention then apply a series of transformations to the parse tree. Then, the document re-authoring systems and methods of this invention map each resulting transformed parse tree back into a document representation, which may be in a document format that is different from the input format of the original document.

[0064] Document transforms are implemented using a standard procedure that includes a condition function that takes a state node in the document version space and returns true if the transform should be applied to the state, and

an action function that is called when the transform is actually applied to a state to produce a new state containing a new document version, a new measure of quality, and the resulting sub-pages. Three types of transforms can be defined—1) those which are always run on a page before the planning process starts; 2) those used in the best-first planning process; and 3) those which are always run on a page before it translated from the final abstract syntax tree back into a surface form such as HTML.

[0065] Transformations manipulate the parse tree, in the state they are applied to, in order to produce a new version of the document. The manipulations are similar to those described in S. Bonhomme et al., "Interactively Restructuring HTML Documents", Fifth International World Wide Web Conference, Paris, France, May 1996. Whenever portions ofthe parse tree are elided or transformed, an HTML hypertext link is added into the parse tree to reference the node identifiers of all affected parse tree subtrees, enabling users to request the original portions of the document that have been modified during re-authoring.

[0066] The document re-authoring systems and methods of this invention also keep track of which combinations of transforms have already been tried, via a global list of transform sets, assuming that all transformations are commutative, to ensure that no duplicate states are ever constructed.

[0067] One exemplary document re-authoring system and method according to this invention, as described above, has been implemented as an HTTP proxy server. The HTTP proxy server accepts a request for an HTML document, retrieves the document from the specified HTTP server, parses the HTML document, constructs the parse tree, or abstract syntax tree, from the retrieved HTML document, labels each of the parse tree nodes with a unique identifier, and then retrieves any embedded images so that the size of the retrieved images can be determined, as necessary. Once this has been accomplished, the document re-authoring systems and methods of this invention are initialized with a state containing the parse tree for the original retrieved document. During each re-authoring cycle, the document re-authoring systems and methods of this invention select the state with the best document version so far, then select the best applicable transformation technique and apply the selected transformation, resulting in a new state and a new document version being generated. It is assumed that the convolution of transformations is always commutative, and several checks are used by the re-authoring software systems and methods of this invention to ensure that redundant states are not constructed.

[0068] In one exemplary embodiment of the document re-authoring systems and methods of this invention, fifteen transformation techniques were implemented: FullOutline, Outline ToH1, Outline ToH2, Outline ToH3, Outline ToH4, Outline ToH5, Outline ToH6, FirstsentenceElision, ReduceImages25%, ReduceImages50%, ReduceImages75%, ElideAllInmges, FirstImageOnly, BookendImages, and ReduceFontSize.

[0069] This exemplary embodiment of the document re-authoring systems and methods of this invention has been implemented in the Java programming language. In addition to functioning as a true proxy server, this HTTP proxy server system can also respond to requests for certain uniform resource locators with documents generated by the HTTP proxy server itself. This is used to provide the user with forms-based control over the HTTP proxy server and the document re-authoring systems and methods. This exemplary embodiment of the document re-authoring system can process even very compex pages in less than 2 seconds on a 200Mhz Pentium, using Symantec's Java JIT compiler.

[0070] The first thing that a user of the document re-authoring software systems and methods of this invention must do is specify the size of display for the device being used and indicate the font size of the default browser font being used. This information is needed in order to estimate the screen area requirements of text blocks. To do this, the user requests a specific control uniform resource locator from the HTTP proxy server, resulting in delivery of the form 300 shown in Fig. 4.

[0071] Once a user has configured the document re-authoring system, the user can start retrieving documents from a distributed network, such as the World Wide Web. The original page 400 and the re-authored page 410 shown in Fig. 5 illustrate the re-authoring capability of the document re-authoring systems and methods of this invention. In this example, this exemplary embodiment of the document re-authoring systems and methods of this invention chose to use 25% image reduction in combination with first sentence elision to render the displayed page 410 from the original page 400. The re-authored page 410 is then displayed on a browser window 420. In this exemplary embodiment of the re-authoring systems and methods of this invention, immediately following retrieval of a page, the user can request a trace of the re-authoring session to determine which transformations had been applied, by requesting another control uniform resource locator from the HTTP proxy server.

[0072] Fig. 6 shows one exemplary embodiment of an environment 500 in which the automatic document re-authoring systems and methods and/or the automatic document filtering systems and methods of this invention will be implemented. As shown in Fig. 6, the environment 500 includes a limited display area device 510 that has a display having a display area that is significantly limited relative to the display area of a monitor for a desktop or a laptop computer. As shown in Fig. 6, the environment 500 further includes a transmitter/receiver communication system 550, a host node 570 of a distributed network and the remaining portions 590 of the distributed network.

[0073] In the environment 500, the limited display area device 510 will normally be a personal digital assistance

(PDA), a cellular phone or the like that is connected by a wireless communication channel 530 to the transmitter/receiver communication system 550. Thus, as shown in Fig. 6, the limited display area device 510 will normally include an antenna 520, while the transmitter/receiver communication system 550 will normally include a corresponding antenna 540. The limited display area device 510 will normally communicate with the transmitter/recciver communication system 550 over the wireless communications channel 530 using radio frequency signals transmitted between the antennas 520 and 540.

[0074] The transmitter/receiver communication system 550 converts the analog or digital signals received from the limited display area device 510 over the communications channel 530 in to a form usable by the host node 570 of the distributed network. The transmitter/receiver communication system 550 then outputs the signals received over the communications channel 530 over a communication link 560 to the host node 570 of the distributed network. It should be appreciated that the communication link 560 can be any known or later-developed communication structure capable of transmitting the appropriate signals between the transmitter/receiver communication system 550 and the host node of the distributed network 570. Because the exact structure of the transmitter/receiver communication system 550 and the communication link 560 will be a matter of design choice depending upon how these elements are implemented, but such design choices will be readily apparent and predictable to those of ordinary skill in the art, these elements will not be further described.

[0075] It should also be appreciated that the limited display area device 510 can also be connected to the host node 570 of the distributed network by other than the wireless communication channel 530, such as a communication link 522. That is, the communication link 522 could be any other known communications structure, such as a local area network, a wide area network, a modem connection over the public switched telephone network or a cable television system, or the like. For example, the user of the limited display area device 510, rather than communicating over the wireless communication channel 530, could connect the limited display area device 510 to the public switch telephone network using a modem. The user would then dial directly into the host node 570 of the distributed network.

[0076] Regardless of how the host node 570 of the distributed network is ultimately connected to the limited display area device 510, once the host node 570 of the distributed network receives a request for a document to be transmitted to the limited display area device 510, the host node 570 of the distributed network first determines if the requested document is located locally on the host node 570 of the distributed network. If the requested document is not located locally, the host node 570 of the distributed network communicates over a communication structure 580 to the remaining portions 590 of the distributed network to request the document. The particular node of the remaining portions 590 of the distributed network storing that document ultimately will receive the request from the host node 570 over the communication structure 580 and will return the requested document to the host node 570 over the communication structure 580. It should be appreciated that the communication structure 580 can be any known or later-developed communication structure and protocol system for linking together widely located nodes of a distributed network.

[0077] Once the host node 570 of the distributed network receives the requested document, an HTTP proxy server executing on the host node 570 of the distributed network re-authors the requested document based on the previously-provided information about the limited display area device 510. A first re-authored page is ther transmitted by the host node 570 over either the wireless communication link 530 or the communication link 522 to the limited display area device 510. As the user reviews the delivered page, the user may determine that viewing additional information removed from the re-authored page is required. In this case, the user will send a request over one of the wireless communication link 530 or the communication link 522 to the host node 570 of the distributed network to obtain the desired re-authored sub-page. The host node 570, in response to this request, transmits a further re-authored sub-page of the original document to the limited display area device 510 over one of the wireless communication channel 530 or the communication link 522.

[0078] Fig. 7 shows this information flow in greater detail. As shown in Fig. 7, when the user of the limited display area device 510 wishes to review a particular document presiding on a distributed network, the user sends a request for the particular document from the limited display area device 510 to an HTTP proxy server 571 residing on the host node 570 of the distributed network. The HTTP proxy server 571 then transmits the request for the particular document to the particular remote node 591 on the distributed network that stores the requested page. The particular remote node 591 returns the requested original document to a document re-authoring system 600 residing on the HTTP proxy server 571. The document re-authoring system 600 re-authors the original document into a plurality of subdocuments that are each capable, as closely as possible, of being displayed on the limited display area device 510. The document re-authoring system 600 then delivers the first re-authored to page to the limited display area device 510, while the other re-authored sub-pages are stored in a re-authored sub-page cache 636 of the document re-authoring system 600. Thus, when the user of the limited display area device 510 wishes to view information residing on one of the re-authored sub-pages stored in the re-authored sub-page cache 636, the user causes the limited display area device 510 to transmit a request for that sub-page. The requested cached sub-pages are delivered from the re-authored sub-page cache 636 to the limited display area device 510.

[0079] It should be appreciated that, while the HTTP server 571, the document re-authoring system 600 and the re-

authored subpage cache 636 are shown in Fig. 7 as independent elements, in general, these elements will be implemented as different portions of a single entity, such as different modules of a single software application.

[0080] Fig. 8 is a functional block diagram outlining in greater detail one exemplary embodiment of the document re-authoring system 600. As shown in Fig. 8, the document re-authoring system 600 includes a controller 610, an input/output interface 620, a memory 630, an abstract syntax tree generating circuit 640, a document size evaluation circuit 650, a transform circuit 660 and a tree-to-document remap circuit 670. each interconnected by a data/control bus 680. The communication links 522, 560 and 580 discussed above with respect to Fig. 6 are each connected to the input/output interface 620.

[0081] The memory 630 includes a number of functionally distinct portions, including an original page memory portion 631, a display device size memory portion 632, an abstract syntax tree memory portion 633, a search space portion 634, a transform memory 635, the re-authored page cache 636 described above with respect to Fig. 7, and a sub-pages to be re-authored list 637. The original page memory portion 631 stores the returned original document returned from the remote node 591 of the distributed network that stores the page requested by the limited display area device 510.

[0082] The display device size memory 632 stores a number of form documents used by the document re-authoring system 600 to obtain various parameters about the limited display area device 510 used by the document re-authoring system 600 to re-author a page for a particular limited display area device 510. The display device size memory 632 also stores the particular size parameters for at least one limited display area device 510. It should be appreciated there are a number of different possible ways of implementing the document re-authoring system 600 relative to the various parameters about the limited display area device 510. In one exemplary embodiment, the document re-authoring system 600 can store the various parameters for a particular limited display area device 510 only for as long as that limited display area device 510 remains continuously connected to the document re-authoring system 600. In this case, each time a particular limited area device 510 is reconnected to the document re-authoring 600, the document re-authoring system 600 would send the various forms used to obtain the various parameters about the limited display area device 510 and the user would be required to re-supply these various parameters each time the document re-authoring system 610 was initially accessed.

[0083] While this reduces the required size for the display device size memory 632 and does not require any system for identifying a particular limited display area device 510, this system places a larger burden on the user of the limited display area device 510 or requires a process for automating the supply of information from the limited display area device 510 to the document re-authoring system 600. This automation could be provided, for example, by the document re-authoring system 600 requesting the information from the limited display area device 510. If the information has already been entered by the user during a previous session with the document re-authonng system 600, and that information was stored at that time on the limited display area device 510, the user would not need to be actively involved in re supplying the information to document re-authoring system 600.

[0084] Alternatively, the information could be stored in the display device size memory 632, along with an identification code that the user can cause to be supplied from the limited display area device 510 when beginning a session with the documen re-authoring system 600. By supplying the identification code to the document re-authoring system 600, the user again would not be required to re-supply all of the various parameters about the limited display area device 510 each time the document re-authoring system 600 is accessed.

[0085] In any case, the document re-authoring system 600 uses the various parameter about the limited display area device 510, as described above, when re-authoring the original page stored in the original page memory 631 so that each re-authored page will fit, as closely as possible, on to the small display area of the limited display area device 510.

[0086] The abstract syntax tree memory portion 633 stores the abstract syntax tree generated form the original document stored in the original page memory 631 by the abstract syntax tree generating circuit 640. The transform memory portion 635 stores the various transforms described above, as well as the conditions under which each transform can be applied and the conditions regarding which transforms are not usable with various other ones of the transforms. The transform memory 635 also stores an indication of the desirability of applying any particular transform to a particular original or re-authored page. That is, as described above, the various transforms have general order that emphasis applying a more limited transform, such as reducing an image by a small about, over a more radical transform, such as reducing an image by a large amount or removing the image completely.

[0087] The re-authored page cache 636 stores the abstract syntax tree corresponding to each re-authored page or sub-page as the document size evaluation circuit indicates that the abstract syntax tree for a particular re-authored page or sub-page is good enough, based on the various parameters about the limited display area device 510 stored in the display device size memory 632. The sub-pages to be re-authored list 637 stores the abstract syntax trees for those sub-pages generated by transforming the original document or an earlier sub-page. These sub-pages will generally contain the images of any reduced-size images or any elided images, as well as the full text of any text segments that have had content elided from them.

[0088] Finally, the search space memory 634 stores a number of states generated by the transform circuit 660 as it

applies the various transforms stored in the transform 635 to either the original document stored in the original page memory 631 or to various sub-pages stored in the sub-pages to be re-authored list 637, based on the particular state of the search space currently being manipulated.

[0089] In particular, each state i in the search space 634 includes an evaluation value portion, a transformed abstract syntax tree portion and a sub-page list portion. The evaluation valued portion stores the evaluation value generated for the re-authored page or sub-page corresponding to the state i generated by the document size evaluation circuit 650. The transformed abstract syntax tree portion stores the transformed abstract syntax tree for the state i generated by the transform circuit 660 by applying one of the transforms in the transform memory 635 to the parent state to the state i. The sub-page list portion stores the list of sub-pages generated to store any original content removed from the page corresponding to the state i when the transform circuit 660 applies the particular transform used to generate that state i.

[0090] It should be appreciated that state 0 corresponds to the original document stored in the original page memory 631. In particular, the evaluation value portion of state 0 corresponds to the evaluation value generated for the original document before any re-authoring. In this state 0, the transformed abstract syntax tree portion stores the original un-transformed abstract syntax tree generated by the abstract syntax tree generating circuit for the original document. Finally, before state 0, the sub-page list will be empty, as the original document contains all ofthe original information and therefore, no sub-pages are required.

[0091] Fig. 9 graphically illustrates various states stored in the search space memory portion 634. In particular, Fig. 9 shows a document comprising a section header, a text paragraph, and an image. As shown in Fig. 9, in the initial state, i.e., state 0, the original document has not been transformed. This initial state also shows the original rating, i. e., the evaluation value, generated for the original document. Fig. 9 also shows the state 1 generated from the state 0 by applying the "elide all images" transformed to the document of state 0. As shown in state 1, the re-authored sub-page of state 1 contains the section header and the text but does not contain the image. Rather, in place of the image, the re-authored sub-page of state 1 contains a link labeled "IMG" that links the re-authored page of state 1 to the sub-page storing the image elided from the re-authored subpage of state 1. State 1 also indicates the evaluation value for this re-authored document. As shown in Fig. 9, the size requirements for the re-authored page are now one-quarter of the size requirements of the original, un-re-authored page.

[0092] Fig. 9 also indicates that two additional states, state 2 and state 3, were generated by applying other transforms to the document of state 0. Finally, Fig. 9 shows three additional states, state 4, state 5 and state 6, generated by applying additional transforms to the re-authored document of state 1 or to the sub-page of state 1. For example, if the sub-page containing the image is still too large to be displayed on the limited display area device 510, an intermediate sub-page generated by applying the "reduced image by 25%", the "reduce image by 50%", or the "reduce image by 75%" transforms to the image to obtain a re-authored document good enough to be displayed on the limited display area device 510.

[0093] Currently, in operation, the document re-authoring system 600 of Fig. 8 receives the returned original document over the communication link 580. The received or general document is input through the input/output interface 620 and is stored in the original page memory 631 under the control of the controller 610. Then, the abstract syntax tree generating circuit 640, under control of the controller 610, inputs the original document from the original page memory portion 631 and generates an abstract syntax tree from that original document. The abstract syntax tree generated by the abstract syntax tree generating circuit 640 is then stored in the abstract syntax tree memory portion 633 of the memory 630 under control of the controller 610.

[0094] The document size evaluation circuit 650 then inputs, under control of the controller 610, the abstract syntax tree corresponding to the original document stored in the original page memory 631 and the various parameters from the display device size memory 632 about the particular limited display area device 510 to which the re-authored documents are to be returned. The document size evaluation circuit 650 then generates an evaluation value and stores that evaluation value in state 0 of the search space memory portion 634. The document size evaluation circuit 650 also outputs an indication to the controller 610 whether the document of state 0 is good enough for outputting it to the limited display area device 510 over one of the communication links 522 or 560. If the original document is already good enough, the original document is immediately returned without further transformation.

[0095] Then, the transform circuit 660, under control of the controller 610, inputs the document of state 0, as represented by the abstract syntax tree for that state, and applies one of the transforms stored in the transform memory 635 to the abstract syntax tree of the input state. In particular, the transform circuit 660 first determines, for the current state i, whether the selected transform should be applied to the current state i of the document. For example, as described above, if the current state i of the document does not contain any images, there is no point in applying any of the image reduction or elision transforms to this state of the document. Furthermore, if the "elide all but first image" transform has already been applied to obtain the current state i of the image, there is no point of applying the "elide all but first and last images" transform to this current state i.

[0096] Assuming the current transform selected by the transform circuit 660 is properly applicable to the current state

i of the document, as indicated by the transformed abstract syntax tree for the current state i, the transform circuit 660 applies that transform to the abstract syntax tree for that state to generate a child state j. The child state j includes the further transformed abstract syntax tree and a sub-page list indicating the sub-pages that remain to be transformed based on the content elided from the original document necessary to reach this child state j. Finally, the document size evaluation circuit 650, under control of the controller 610, evaluates the document obtained in the child state j to determine if that resulting document is good enough for outputting to the limited display area device 510. That evaluation value is then stored in the newly-created child state j.

[0097] After the transform circuit 660 has generated the new child state j, the transform abstract syntax tree for that state j is output to the document size evaluation circuit 650 for evaluating the size requirements of the document corresponding to the state j.

[0098] Once the abstract syntax tree for the first page of the transformed document is determined to be good enough, that abstract syntax tree is output to the tree-to-document remap circuit 670, which renders the first re-authored sub-page from that abstract syntax tree. That first re-authored sub-page is output from the tree-to-document remap circuit 670 to the input/output interface 620 and ultimately is transmitted to the limited area display device 510. At the same time, the transform circuit 660 continues to apply additional transforms to any subpages resulting from transforming the original document into the first good-enough re-authored subpage. As each such subpage is transformed into a good-enough subpage, the abstract syntax tree for each such good-enough subpage is stored in the re-authored page cache 636 until a request for that subpage is received by the document re-authoring system 600 from the limited area display device 510.

[0099] Once a request for that subpage is received by the document re-authoring system 600, the abstract syntax tree for that requested subpage is output to the tree-to-document remap circuit 670, which renders the requested re-authored sub-page from that abstract syntax tree. That requested re-authored sub-page is output from the treeto-document remap circuit 670 to the input/output interface 620 and ultimately is transmitted to the limited area display device 510.

[0100] It should be understood that each ofthe circuits and other elements shown in Figs. 6-8 can be implemented as portions of suitably programmed general purpose computers. Alternatively, each of the circuits shown in Figs. 6-8 can be implemented as physically distinct hardware circuits within one or more ASICs, or using FPGAs, PDLs, PLAs, or PALs, or using discreet logic elements or discreet circuit elements. The particular form each of the circuits shown in Figs. 6-8 will take is a design choice and will be obvious and predictable to those of ordinary skill in the art.

[0101] It should also be appreciated that the links 522, 560 and 580 can by any known or later-developed device or system for connecting the limited display area device 510 to the host node 570 or the host node 570 to the transmitter/receiver communication system 550 or the remaining portions 590 of the distributed network. Thus, the links 522, 560 and 580 can each be implemented as a direct cable connection, a connection over a wide-area network or a local-area network, a connection over an intranet, or a connection over the Internet. In general, the links 522, 560 and 580 can be any known or later-developed connection system or structure usable to connect the corresponding apparatus to the host node 570 over the distributed network.

[0102] It should further be appreciated that the document re-authoring system 600 is preferably implemented on a programmed general purpose computer. However, the document re-authoring system 600 can also be implemented on special purpose computer, a programmed microprocessor or microcontroller as a peripheral integrated circuit elements, and ASIC or other integrated circuit, a digital signal processor, a hardwired electronic or logic circuit such as a discreet element circuit, a programmable logic device such as PLD, PLA, FPGA or PAL, or the like. In general, any device, capable of implementing a finite state machine that is in turn capable of implementing the flowcharts shown in Figs. 11A-14, can be used to implement the document re-authoring system 600.

[0103] The memory 630 shown in Fig. 8 is preferably implemented using static or dynamic RAM. However, the memory 630 can also be implemented using a floppy disk and disk drive, a writeable optical disk and disk drive, a hard drive, flash memory or any other know or later-developed volatile or non-volatile alterable memory. In addition, the memory 630 can further include one or more portions storing control programs for the controller 610. In general, such control programs are preferably stored using non-volatile memory, such as flash memory, a ROM, a PROM, and EPROM or EEPROM, a CD-ROM and disk drive, or any other known or later-developed alterable or non-alterable non-volatile memory.

[0104] Fig. 10 shows another exemplary original document and the abstract syntax tree that is generated from that document. As shown in Fig. 10, the document includes an image, a table having two rows of three columns each, and a text paragraph. The resulting abstract syntax tree generated from this page includes a root node labeled "Page". Three intermediate nodes, "Image", "Table" and "Paragraph" corresponding to each of the image, the table and the text paragraph, respectively, extend from the root "Page" node. Furthermore, as shown in Fig. 10, two intermediate nodes, "Row 1" and "Row 2", corresponding to each of the two rows, respectivley, extend from the intermediate "table" node. Finally, three nodes, corresponding to each of the three cells in each row, respectively, extend from each of the "Row 1" and "Row 2" nodes.

[0105] To re-author the page shown in Fig. 10, for example, the first transform to be applied would generally replace the full size image with a node representing an image reduced by 25%. Then, a new abstract syntax tree having a root node corresponding to the full-sized image would be formed and linked by a hypertext link to the reduced image node of the transformed abstract syntax tree. If the re-authored page having the image reduced by 25% is not yet good enough, the image reduction transformation reducing the image by 50%, 75% and then completely removing the image would be applied.in turn to the original document until a good-enough image was obtained. In each case, the abstract syntax tree would contain a link from the transformed node corresponding to the image to the separate abstract syntax tree containing the full-sized image. If removing the image completely is still insufficient to result in a good-enough re-authored document, the table transform can be applied, as described above, to transform the table into a set of linked individual cells, or the First Sentence Elision transform can be applied to move the text paragraph into a separate subpage.

[0106] Figs. 11A and 11B are a flowchart outlining one exemplary method for re-authoring a page according to this invention. As shown in Fig. 11, control begins in step S100 and continues to step S110, where a user connects a device having a limited display area to a re-authoring system according to this invention. Then, in step S120, the re-authoring system transmits one or more parameter forms to the user to obtain the necessary information about the limited display area necessary to be able to re-author a requested page for display on the limited display area device. Then, in step S130, the re-authoring system inputs the parameter information from the user and stores the input parameter information in a memory. Control then continues to step S140.

[0107] As indicated above with respect to Figs. 6 and 7, the parameter information gathering process outlined in steps S120 and S130 can be automated so that the user does not have to be actively involved in performing steps S120 and S130. Alternatively, as shown in optional step S135, steps S120 and 130 can be replaced by step S135. In step S135, the user either actively inputs, or the limited display area device automatically outputs, an identification code to the re-authoring system identifying previously-stored parameter information for this particular limited display area device. Control then again continues to step S140.

[0108] In step S140, a request for a document on the distributed network is output to the re-authoring system from the user using the limited display area device. Then, in step S150, the re-authoring system obtains the requested document from the distributed network. Next, in step S160, the obtained document is parsed to build an abstract syntax tree of that document. Then, in step S170, an evaluation value for the obtained original document is generated from the abstract syntax tree. Control then continues to step S180.

[0109] In step S180, the evaluation value is analyzed to determine if the obtained document is good enough to b displayed on the limited display area device without any re-authoring. If so, control jumps to step S340. Otherwise, control continues to step S190.

[0110] In step S190, one or more pre-re-authoring transforms are applied to the abstract syntax tree of the obtained, original document. These pre-re-authoring transforms are used, for example, to remove portions of the original document that do not contain any content but that consume display area For example, such portions of the obtained document include banners and other graphical elements that are merely identifying links to other pages or portions of the page. These contentless images are replaced by text links. However, because such transforms do not actually remove any content from the image, re-authoring the page in this way does not require the removed portions to be retained. Other portions that can be removed without effecting the content of the original document include formatting commands that add whitespace and other contentless esthetic formatting to the original document. Finally, other transforms can be applied that convert the various fonts of a document to a single standard font to eliminate unnecessary display area requirements of large and complicated fonts.

[0111] Once the pre-re-authoring transforms are applied in step S190, control continues to step S200, where an evaluation value for the pre-re-authored original document is generated. Then, in step S210, the pre-re-authored documents evaluation value is checked to determine if the pre-re-authored document is good enough to be displayed on the limited display area device. If so, control again jumps to step S340. Otherwise, control continues to step S220.

[0112] In step S220, state 0 of the search space, corresponding to the pre-re-authored document, is selected as the current state of the search space. Then, in step S230, a first transform is selected as the current transform. Then, in step S240, a determination is made whether the current transform can be applied to the abstract syntax tree of the current state. As outlined above, various ones of the transforms have conditions that indicate whether that transform can be efficiently applied to the current re-authored document or whether the current transform is properly combinable with previously applied transforms. If the current re-authored document corresponding to the current state is such that the current transform can be efficiently applied and does not conflict with any previously applied transforms, control continues to step S250. Otherwise, control jumps to step S290.

[0113] In step S250, the current state is transformed to a child state using the current transform and the resulting child state, including the transformed abstract syntax tree and any resulting sub-pages, are added to the search space. Then, in step S260, an evaluation value is generated for the document corresponding to the transformed abstract syntax tree corresponding to the child state generated in step S250. Next, in step S270, the evaluation value is analyzed

to determine if the document corresponding to the child state generated in step S250 is good enough to be displayed on the limited display area device. If the evaluation value indicates the re-authored document or sub-page is good enough, control jumps to step S310. Otherwise, control continues to step S280.

[0114] In step S280, a determination is made whether all transforms have been applied to the current state. If all of the transforms have not been applied, control continues to step S290. Otherwise, control jumps to step S300.

[0115] In step S290, the next transform is selected as the current transform and control jumps back to step S240. In contrast, in step S300, the state of the search space having the best evaluation value is selected as the current state. Control then jumps back to step S230.

[0116] In step S310, the document or sub-page defined by the current state is added to the re-authored page cache as a first re-authored page or a next re-authored sub-page suitable for delivery to the requesting limited display area device. Then, in step S320, a determination is made whether there are any sub-pages resulting from the good-enough sub-page that has been added to the re-authored page cache. If there are any such sub-pages that still need to be re-authored, control continues to step S330. Otherwise, control jumps to step S340.

[0117] In step S330, a state of the search space corresponding to one of the sub-pages to be re-authored is selected as the current state. Control then jumps back to step S230. In contrast, since there are no further sub-pages that need to be re-authored, in step S340, the first re-authored page is output to the requesting limited display area device. Then, in step S350, the control routine ends.

[0118] Fig. 12 outlines one exemplary embodiment of an elision transform according to this invention. As shown in Fig. 12, the elision transform routine begins in step S400, and continues to step S410, where a portion of a current page or sub-page to be removed is selected. Then, in step S420, the selected portion is copied into a new sub-page. Next, in step S430, an identifier is generated for the selected portion. In general, the identifier will be generated using some content of the selected portion. For example, if the selected portion is a paragraph or other text string, the identifier will be the first sentence or the first portion of the first sentence of the selected text portion. If the selected portion is an image, the identifier could be a portion of text used to identify the image in the web document. Control then continues to step S440.

[0119] In step S440, a link is generated to link the current page or sub-page with generated sub-page. Then, in step S450, the selected portion is removed from the current page or sub-page and the identifier and the link are added to the current page. Next, in step S640, the control routine stops.

[0120] Fig. 13 outlines one exemplary embodiment of a table transform according to this invention. As shown in Fig. 13, the table transform begins in step S500 and continues to step S505, where a top level table is selected as the current table. Then, in step S510, the current table is checked to determine if there are any nested tables in the current table. If so, control continues to step S515. Otherwise, control jumps to step S520. In step S515, one nested table of the current table is selected as the new current table. Control then jumps back to step S510, to determine if there are nested tables in this nested table selected as the current table.

[0121] Once there are no nested tables in the current table, in step S520, the current table is checked to determine if there are any sidebars in the current table. If so, control continues to step S525. Otherwise, control jumps to step S535. In step S525, a link list is generated from all of the links in all of the sidebars of the current table. Then, in step S530, the link list is placed at the end of the current table. Control then continues to step S535.

[0122] In step S535, the current table is divided into two or more portions. In particular, as indicated above, one method for dividing the current table into portions is to divide each cell of the table into a separate portion. Then, in step S540, each portion of the current table is copied into a separate new sub-page, and "Next" and "Previous" links are added to each such sub-page. Next, in step S545, the current table is replaced with the set of linked sub-pages generated in step S540. Control then continues to step S550.

[0123] In step S550, the current table is checked to determine if it is the top level table. If not, there is at least one higher level table that still needs to be divided into portions. Accordingly, control continues to step S555. Otherwise, control jumps to step S560.

[0124] In step S555, the table that contains the current table is selected as the new current table. Control then jumps back to step S510, to determine if there any more nested tables in the current table. In contrast, in step S560, the control routine ends.

[0125] Fig. 14 is a flowchart outlining one exemplary embodiment of an image reduction transformation according to this invention. Beginning in step S600, the image reduction transformation continues to step S610, where the image to be reduced in the current sub-page is selected. Then, the reduced image is generated based on the reduction factor associated with the particular image reduction transformation being applied. Then, in step S630, the current sub-page is analyzed to determine if the selected image has been previously reduced. If so, control jumps to step S670. Otherwise, control continues to step S640.

[0126] In step S640, the selected image is copied to a new sub-page. Next, in step S650, a link to the new sub-page is generated. Then, in step S660, the full-size image is removed from the current page or sub-page, and the reduced image and the generated link are added to the current page to form the re-authored page. Control then jumps to step

S680.

[0127] In contrast, in step S670, rather than moving the full-sized image from the current sub-page, the old previously reduced image is removed from the current sub-page and the new reduced image is added to the current sub-page. However, because the current sub-page should already have a link to the previously-created sub-page containing the full-size image, it is not necessary to again add the link to the current sub-page or to create a new sub-page storing that full-sized image. Control then continues to step S680, where the control routine ends.

[0128] Even with perfect automatic re-authoring of documents, there is often simply too much information in a typical web document to make serendipitous cellular phone web browsing a pleasurable or profitable past-time, due to the very small, text-only-type display used in cellular phones. Typically, these devices and services will be used to find and present information that the user is specifically looking for. That is, these devices and services will be used for targeted information search and extraction. The document filtering systems and methods of this invention allow users to extract only portions of documents that they are interested in, via a simple, end-user scripting language that combines structural page navigation commands with regular expression pattern matching and report generation functions.

[0129] The SPHINX system, as described in R. Miller et al., "SPHINX: a framework for creating personal, site-specific Web crawlers", Seventh International World-Wide Web Conference, Brisbane, Australia, April 1998, provides a visual tool that lets users create custom "personal" web crawlers that are similar in functionality to the filtering mechanism of the systems and methods of this invention. The Internet Scrapbook, as described in A. Sugiura et al., "Internet Scrapbook: automating Web browsing tasks by programming-by-demonstration", Seventh International World-Wide Web Conference, Brisbane, Australia, April 1998, allow users to visually select elements from web pages and then updates these elements in a "scrapbook" when the web pages change, providing a function that is similar to the page element retrieval for a particular page of the systems and methods of this invention. Several commercial products also provide similar functionality for other applications, such as, for example, corporate reporting or database population. Lanacom's Headliner Pro, as described in Lanacom, Inc., http://www.headliner.com, and OnDisplay's CenterStage, as described in OnDisplay, Inc., http://www.ondisplay.com, both provide visual editors that let users specify which structural parts of web pages to extract. However, neither of these systems provide users with any ability to extract content based on regular expressions or keywords.

[0130] The document filtering systems and methods of this invention have the capability to extract partial information from a document based on commands written by a user in a high-level scripting language. The document filtering systems and methods of this invention combine page structure navigation, regular expression matching, site traversal, i.e., web crawling, and iterative matching, in addition to re-authoring of the extracted information using the document re-authoring systems and methods of this invention described above.

[0131] A filter script is simply entered into a text file and saved on a web server. The filter script is executed whenever a user requests its Uniform Resource Locator. A filter script will typically load a target web page, traverse to particular locations within that web page, which are described structurally and/or by regular expressions, extract the content found at those locations, and then send the extracted content through the document re-authoring system to be properly formatted before being returned to the user.

[0132] The document filtering systems and methods of this invention take advantage of the parse tree creation and navigation of the document re-authoring systems and methods of this invention, by providing a simple set of HTML document navigation options that use the concept of a "current context" in the HTML document. The current context is analogous to a "cursor" in database programming, in that it refers to a location within HTML the document.

[0133] In actuality, the current context refers to a node in the HTML parse tree. The navigation commands serve to move this reference around within the tree until a desired part of the HTML document is found, at which time the desired part can be extracted. For example, Fig. 10 shows an HTML document and its corresponding parse tree. When the document is first loaded, by executing a "GO URL" command, the current context is pointing at the root node of the parse tree, which essentially refers to the entire document.

[0134] Fig. 15 shows one exemplary embodiment of the document re-authoring system 600 further including a filter circuit 690 that implements the document filtering systems and methods outlined herein. In particular, the filter circuit 690, under control of the controller 610, inputs a requested filter, requested by the user over one of the communication links 522 or 560, that is supplied from a node of the distributed network storing such a filter over the communication link 580. The filter circuit 690 then inputs the requested document from the node of the distributed network storing the requested document and filters the requested document to extract the requested page elements. The filter circuit 690 stores these extracted page elements in the original page memory 631 in place of the original document initially stored there. The document re-authoring system 600 then operates on these extracted page elements as if they were the original document to be re-authored.

[0135] In extracting the page elements from the original document, the filter circuit 690 uses the abstract syntax tree generated by the abstract syntax tree generating circuit from the original document and stored in the abstract syntax tree memory 633.

[0136] Fig. 16 outlines one exemplary embodiment of the information flow when the requested document is also to

be filtered. As shown in Fig. 16, after a request for filter is output by the limited display area device 510 to the HTTP proxy server 571, the request for filter is forwarded by the HTTP proxy server 571 to a remote node 592 of the new distributed network that stores the requested filter. The remote node 592 storing the requested filter returns the requested filter to the document filter 690. The document filter 690 then requests, under control of the controller 610, the document from the remote node 591 of the distributed network that stores the request page. The remote node 591 storing the requested page returns the document to the document filter 690. The document filter 690 then filters the returned document using the filter returned from the remote node 592 and the abstract syntax tree generated by the abstract syntax tree generating circuit 640. The document filter 690 returns the extracted page elements to the document re-authoring system 600 where the extracted page elements are treated as an original document for re-authoring as described above.

[0137] There are three types of page navigation commands, those which go *into* the current context to select more specific content, those which go *out* from the current context to enclosing structures, and those which traverse the page sequentially from the start of the current context, for example, to navigate to the *next* structure of some kind, which may or may not be properly contained within the current context.

[0138] The simplest type of navigation command goes *into* the current context. For example, given the document and current context shown in Fig. 10, executing the command "GO ROW 2" results in the current context being moved to the second table row object within the current context, as shown in Fig. 17.

[0139] The current context can also be enlarged, i.e., moved up the parse tree towards the root node, by using a "GO ENCLOSING" command. For example, given the document and context shown in Fig. 17, a "GO ENCLOSING TABLE" command results in the current context shown in Fig. 18.

[0140] Finally, the current context can be moved forwards or backwards among the objects in a page in a sequential manner, as they appear to a user. This is accomplished by moving the current context forwards or backwards from its current location within a prefix traversal of the parse tree. This results in a search that first is performed within the current context, then continues with the objects that follow the current context on the page. For example, a "GO PREVIOUS IMAGE" command moves to the previous image found sequentially from the current context.

[0141] In addition to named page elements, navigation commands can also be specified using regular expressions. For example, a "GO NEXT" "DOW\sJONES\s*(\d+)\s*POINTS"" command moves the current context to the next match of the specified regular expression, using a prefix traversal of text blocks on the page. The filtering systems and methods of this invention are able to demarcate sub-expressions and recall them into output strings.

[0142] The simple navigation commands described above can also be used to navigate among a set of linked web pages through the use of the "LINKEDPAGE" page object type. For example, a "GO FIRST LINKEDPAGE" command moves to the first hypertext link within the current context, loads the referenced page and moves the current context to the root of that document's parse tree, while a "GO ENCLOSING LINKEDPAGE" command returns the current context to the hypertext link that led to the document currently being processed.

[0143] Traversal between pages is handled by a stack of script activations, each of which pairs script state information (including current context) with a particular Uniform Resource Locator and a parse tree. This facilitates rapid navigation back and forth among linked pages and is required to support the "GO ENCLOSING LINKEDPAGE" command.

[0144] Once the current context has been moved to a page object that is of interest, a "REPORT" command is used to extract it. The "REPORT" command can be issued several times within a filter script, in which case the extracted page elements are concatenated. The "REPORT" command can also be used to insert arbitrary strings into the output, which can contain sub-strings from regular expression pattern matching. For example, the "REPORT "Dow:\1"" command adds the string "Dow:" plus a substring identified by the identifier "1" extracted during a regular expression match to the filter's output.

[0145] Often the user does not know in advance how many page elements of a particular kind will exist on a web page. For example, the number of news article paragraphs in a daily e-zine will generally not be known in advance. The "FOREACH" command addresses this lack of information by executing a sequence of commands for every page element found within the current context that meets a specified criteria. When used with a "LINKEDPAGE" target, this provides the functionality of a web spider that can visit all of the linked pages within a web site. In the following examples the ellipses represent sequences of valid filter commands:

[0146] A "FOREACH PARAGRAPH" command moves to each paragraph within the current context in turn DO... END and executes the specified commands.

[0147] A "FOREACH LINKEDPAGE" command loads each page that is reachable through hypertext links from the DO... END current page in turn and executes the specified commands.

[0148] Whenever a filter encounters any kind of error, including navigation failures, regular expression matching failures, or web page retrieval error, it simply begins the next iteration of the innermost "FOREACH" loop in which the offending command is embedded. If the error occurred at the top level of a filter, the filter halts execution and produces any pending output.

[0149] The document re-authoring systems and methods of this invention do a good job of automatically re-authoring

documents for display on devices with small screens. One exemplary embodiment of the document re-authoring systems and methods of this invention have been informally tested on a wide range of pages for a number of screen sizes. This exemplary embodiment of the document re-authoring systems and methods of this invention produced output that is legible and navigable.

[0150] In one exemplary embodiment, the document re-authoring systems and methods of this invention simply add up the space requirements of all images and text to arrive at an estimate of the screen area requirements for a document. This is adequate for fairly dense documents with minimal structure, such as those in a Xerox Annual Report, but works poorly for documents with a lot of whitespace or which use advanced layout techniques, such as, for example, tables. In a second exemplary embodiment, the document re-authoring systems and methods of this invention includes a size estimator that performs much of the work performed by a browser in formatting each document version onto a display area. Factors other than required screen area may also need to be included, such as actual width requirements of the re-authored document, because users don't like to scroll horizontally, bandwidth requirements, and aesthetic measures.

[0151] Users should be able to adjust the various heuristics used in the document re-authoring systems and methods of this invention to suit their taste. For example, the user could specify the relative preference of the transformation techniques, or specify that some transforms not be used at all. At a higher level of abstraction, the user could express their preferences for a set of trade-offs, such as 'more content' vs. 'larger representation'. In addition, the re-authoring systems and methods of this invention could be moved to the client and coupled with the browser so that the user could dynamically apply and undo different transformations until the user achieves a result the user likes.

[0152] The automatic document re-authoring systems and methods of this invention, and in particular, the exemplary embodiment of the HTTP proxy server described above, are preferably implemented on a programmed general purpose computer. However, the automatic document re-authoring systems and methods of this invention, and in particular, the HTTP proxy server described above, can also be implemented on a special purpose computer, a programmed microprocessor or microcontroller and peripheral integrated circuit elements, an ASIC or other integrated circuit, a digital signal processor, a hardwired electronic or logic circuit such as a discrete element circuit, a programmable logic device such as a PLD, PLA, FPGA or PAL, or the like. In general, any device, capable of implementing a finite state machine, can be used to implement the automatic document re-authoring system and method of this invention, and in particular, the HTTP proxy server described above.

[0153] The automatic document re-authoring systems and methods according to this invention can be performed by invoking a stand-alone re-authoring program running on the HTTP proxy server described above, or can be performed through a plug-in to a conventional web browser, such as Netscape Navigator or the like.

[0154] Furthermore, while the automatic document re-authoring systems and methods of this invention have been described in relation to re-authoring documents obtained from the world-wide web, the automatic re-authoring systems and methods of this invention can be used to re-author documents obtained from any distributed network, such as a local area network, a wide area network, an intranet, the Internet, or any other distributed processing and storage network.

[0155] While this invention has been described in conjunction with the specific embodiments outlined above, it is evident that many alternatives, modifications and variations will be apparent to those skilled in the art. Accordingly, the preferred embodiments of the invention set forth above are intended to be illustrative, not limiting. Various changes may be made without departing from the spirit and scope of the invention.

Claims

1. A method for automatically re-authoring a document, comprising:

parsing the document;
transforming the parsed document into a transformed document;
generating an evaluation value from the transformed document;
determining if the evaluation value meets at least one evaluation criterion,
if the evalution value for the transformed document does not meet the at least one criterion, repeating the transforming, generating and determining steps using a different transform; and
if the evaluation value for the transformed document meets the at least one criterion, outputting the transformed document.

2. The method of claim 1, wherein:

parsing the document comprises generating an abstract syntax tree from the document; and
transforming the parsed document comprises transforming the abstract syntax tree into at least one trans-

formed abstract syntax tree.

3. The method of claim 1 or claim 2, wherein transforming the parsed document comprises:

selecting a transform:
determining if the transform can properly be applied to the parsed document;
if the transform can properly be applied, transforming the parsed document into the transformed document using the selected transform; and
if the transform cannot properly be appliet, repeating the selecting and determining steps for a different transform.

4. The method of any of claims 1-3, wherein transforming the parsed document into the transformed document comprises at least one of outlining sections of the document, removing contentless portions from the document, removing content from the document, reducing a size of ad least one image within the document, removing at least one image from the document, removing at least one table cell from the document, and summarizing text within the document.

5. The method of claim 4. wherein:

outlining sections ofthe document preferably comprises:

identifying sections within the document,
identifying a section header and a document portion for each section,
placing each identified document portion into a separate subpage,
removing the identified document portions from the parsed document to form a transformed document containing only the identified sections headers,
converting each of the identified section headers into a link to the corresponding subpage, and
linking the separate subpages together and to the transformed document;

educing a size of at least one image within the document preferably comprises:

identifying at lead one image within the document,
placing each identified image into a separate subpage,
generating a reduced version of each identified image,
removing each identified image from the document and inserting the reduced version of each removed image to form the transformed document, and
adding, for each removed image, a link into the reduced version of that image to the subpage containing that removed image;

removing at least one image from the document preferably comprises one of removing all images from the document, removing all but the first image from the document, and removing all but the first and last images from the document;
removing at least one table cell from the document preferably comprises:

determining if the table contains any sidebars of links,
if the table contains any sidebars, converting the sidebars into a list of links as a last cell of the table,
identifying all but the first cell of the table,
adding each identified cell to a separate subpage,
replacing the table with the first cell to form the transformed document, and
linking the separate subpages together and to the transformed document, and
removing at least one table cell from the document preferably further comprises:

determining if that cell is a nested table,
if that cell is not a nested table, adding that cell to the separate subpage, and
if that cell is a nested table, repeating the determining, converting, identifying, adding, replacing and linking steps; and

removing contentless portions from the document preferably comprises at least one of replacing sequenc-

es of page breaks or paragraph breaks with a single page break or paragraph break, removing indenting from the document; converting text strings of the document to at least one of a single font and font size, removing bullets from the document, removing background space from the document and removing banner images from the document.

6. The method of any of claims 1-5, wherein, if no transform results in a transformed document that has an evaluation value that meets the a least one evaluation criterion, the method further comprises:

selecting the transformed document having the evaluation value that most closely meets the evaluation value; and
repeating the transforming, generating and determining steps on the selected transformed document using art additiond transform.

7. The method of any of claims 1-6, wherein:

transforming the parsed document into a transformed document comprises generating at least one subpage; and
when a transformed document meets the at least one evaluation criterion, the method further comprises:
generating an evaluation value for each generated subpage for that transformed document;
detemining, for each subpage, if the evaluation value for that subpage meets the at least one evaluation criterion;
for each subpage, if the evaluation value for that subpage does not meet the at least one criterion, performing the transforming, generating and determining steps on that subpage using an additional one of the transforms to generate a transformed subpage; and
for each subpage, if that subpage meets the at least one criterion, identifying that subpage as ready to be output.

8. The method of claim 1, further comprising, after parsing the document:

optionally removing contentless portions from the document to form a pre-transformed document
generating an evaluation value from the document or the pre-transformed document;
determining if the evaluation value meets at least one evaluation criterion;
if the evaluation value for the document or the pre-transformed document does not meet the at least one criterion, performing the transforming, generating and determining steps using a first one of the transforms; and
if the evaluation valve for the document or the pre-transformed document meets the at least one criterion, outputting the document or the pre-transformed document without removing any content from the document.

9. The method of any of claims 1-8, wherein transforming the document comprises:

filtering the document to extract desired portions of the document; and
replacing the document with the extracted portions.

10. A document re-authoring system that automatically re-authors a document, comprising

a parse tree generating circuit that parses the document to generate a parse tree;
a transform circuit that transforms the parse tree using a first transform to generate a transformed parse tree representing a transformed document, and that preferably transforms the parse tree or the transformed parse tree using another transform to generate another transformed parse tree representing another transformed document; and
a document size evaluation circuit that evaluates the parse tree or the another transformed parse tree to determine if the document, the transformed document or the another transformed document meets at least one evaluation criterion;
wherein, when the document, the transformed document or the another transformed document meets the at least one evaluation criterion; the document, the transformed document or the another transformed document is output to a display device.

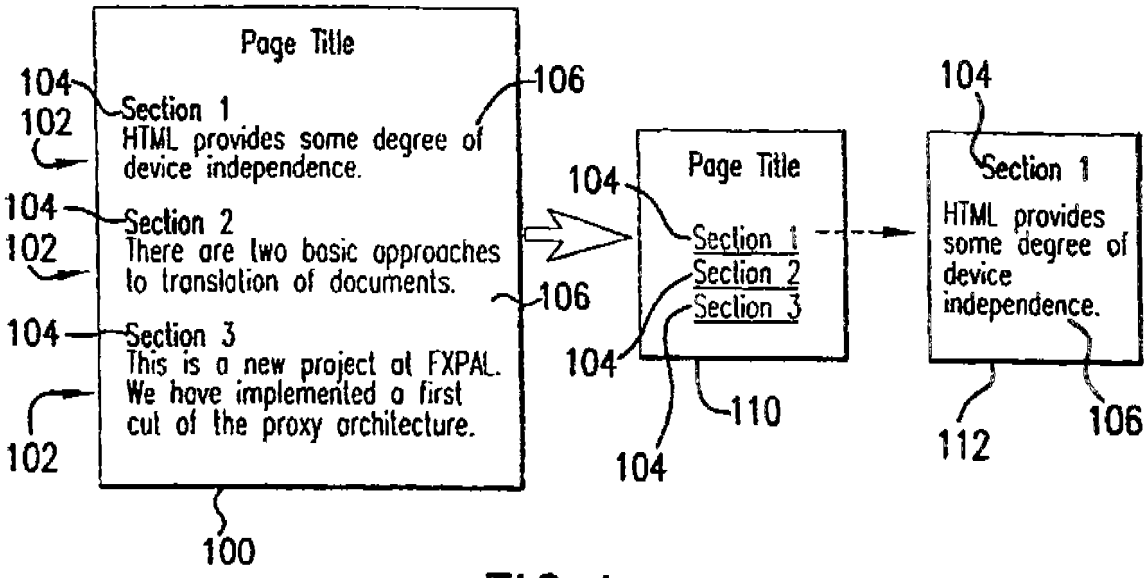

Page Title
104
Section 1
102
HTML provides some degree of device independence.
106
104
Section 2
102
There are two basic approaches to translation of documents.
106
104
Section 3
This is a new project at FXPAL. We have implemented a first cut of the proxy architecture.
102
100
104
Page Title
Section 1
Section 2
Section 3
104
104
110
104
104
Section 1
HTML provides some degree of device independence.
112
106

FIG.1

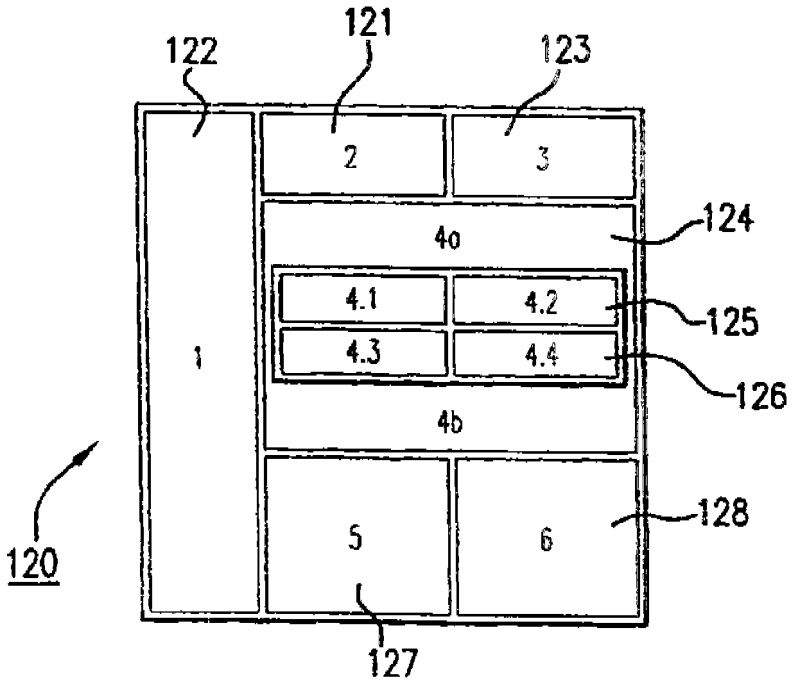

121
122
123
2
3
4a
124
4.1
4.2
125
1
4.3
4.4
126
4b
5
6
128
120
127

FIG.2

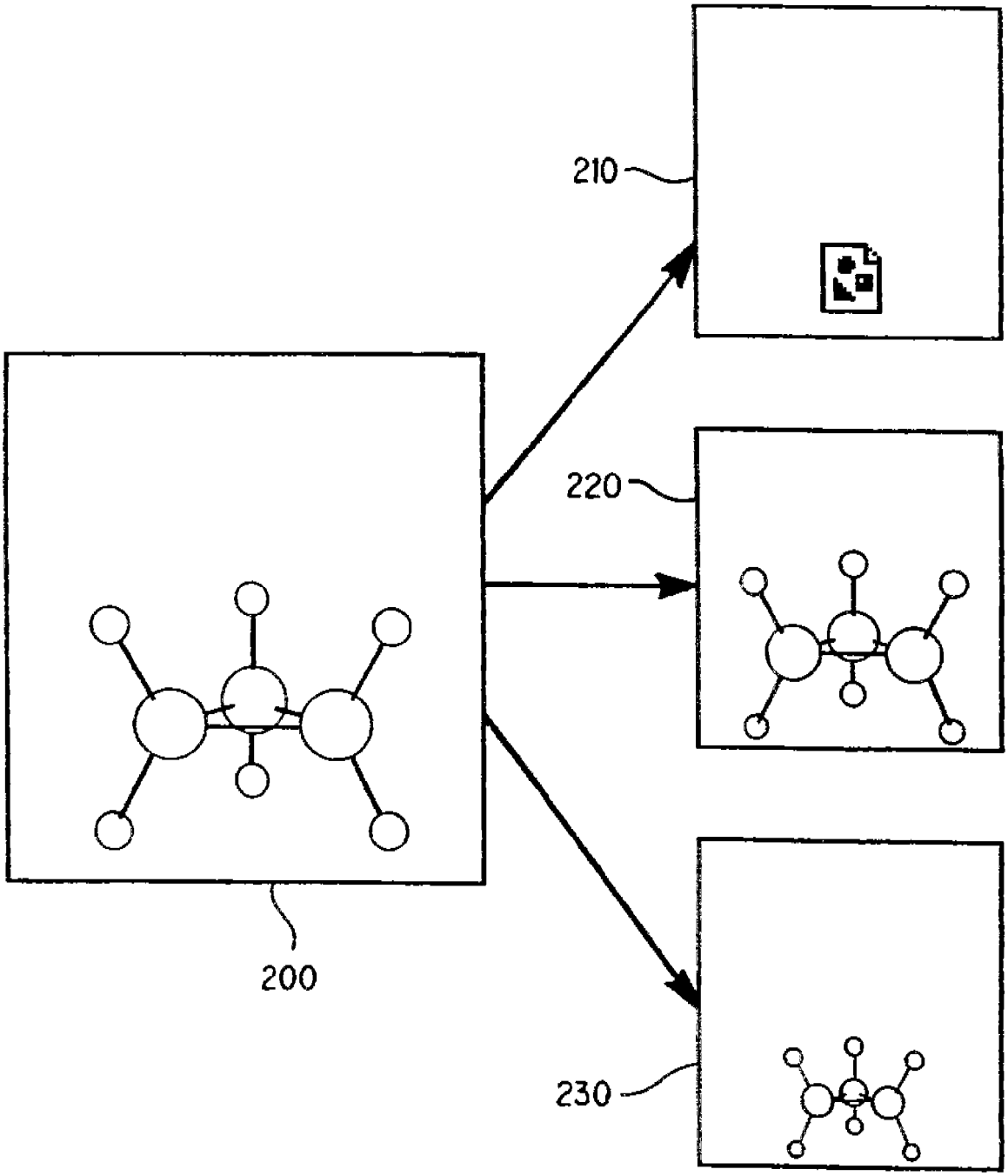


FIG. 3

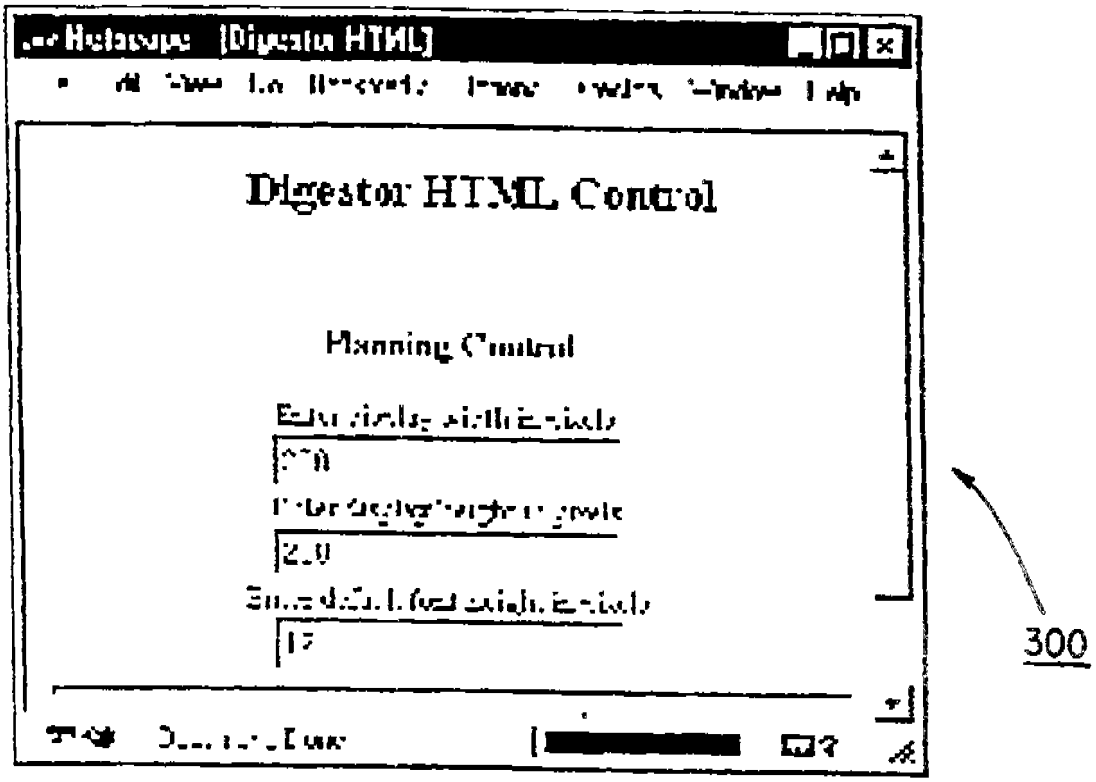


FIG. 4

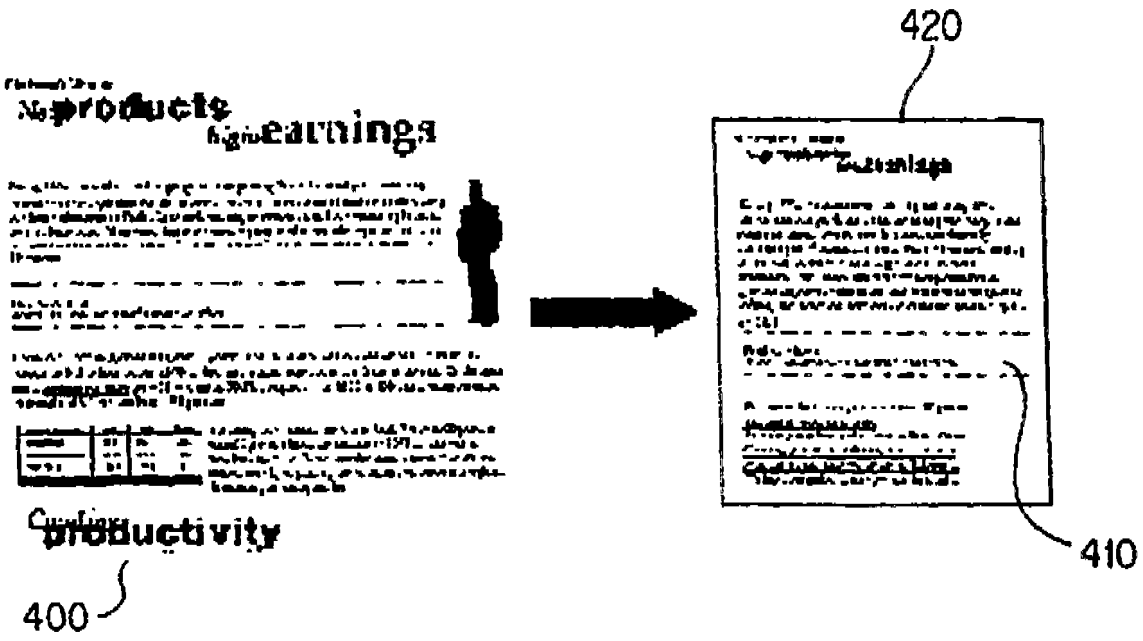


FIG. 5

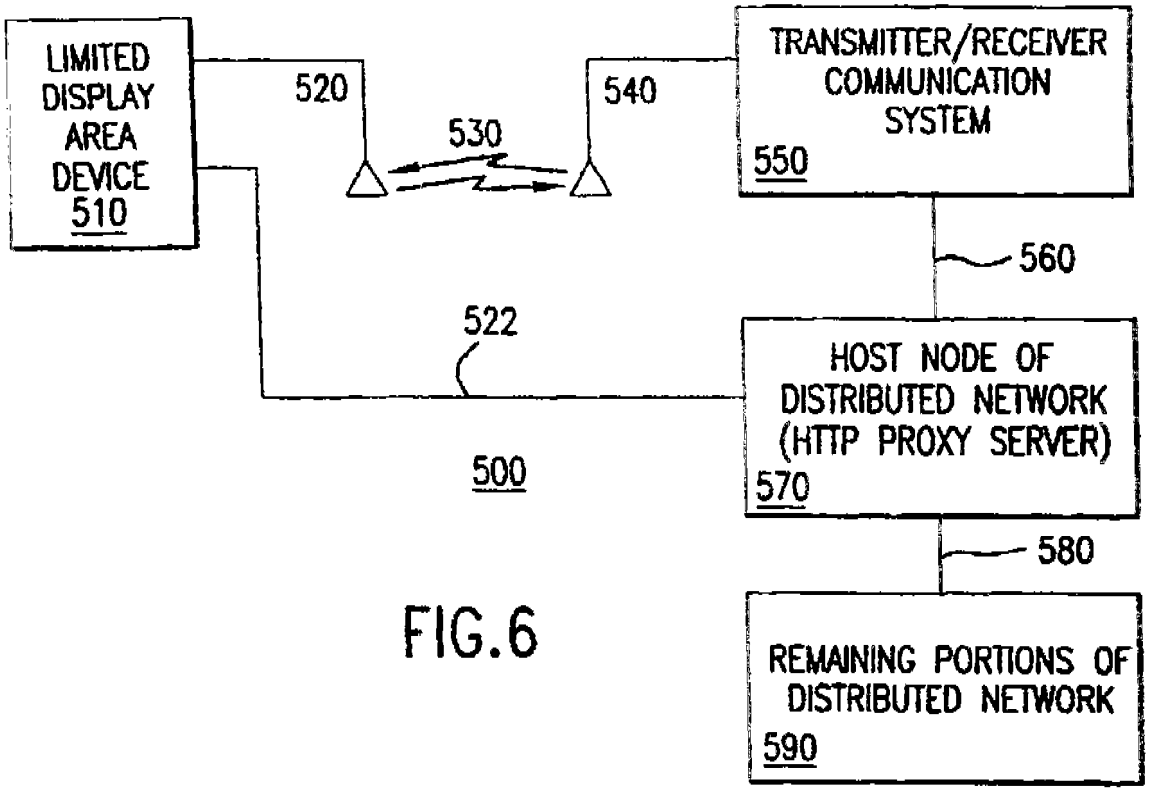

LIMITED DISPLAY AREA DEVICE 510
520
530
540
TRANSMITTER/RECEIVER COMMUNICATION SYSTEM 550
560
522
HOST NODE OF DISTRIBUTED NETWORK (HTTP PROXY SERVER) 570
500
580
REMAINING PORTIONS OF DISTRIBUTED NETWORK 590

FIG.6

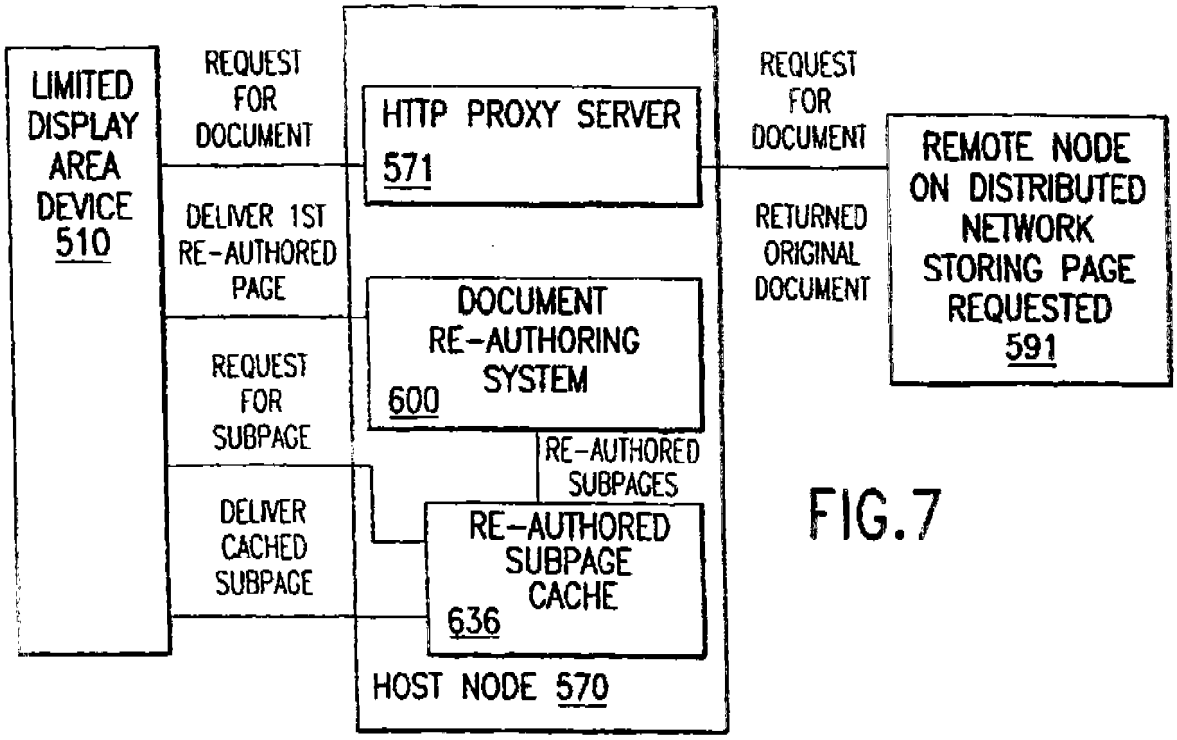

LIMITED DISPLAY AREA DEVICE 510
REQUEST FOR DOCUMENT
DELIVER 1ST RE-AUTHORED PAGE
REQUEST FOR SUBPAGE
DELIVER CACHED SUBPAGE
HTTP PROXY SERVER 571
DOCUMENT RE-AUTHORING SYSTEM 600
RE-AUTHORED SUBPAGES
RE-AUTHORED SUBPAGE CACHE 636
HOST NODE 570
REQUEST FOR DOCUMENT
RETURNED ORIGINAL DOCUMENT
REMOTE NODE ON DISTRIBUTED NETWORK STORING PAGE REQUESTED 591

FIG.7

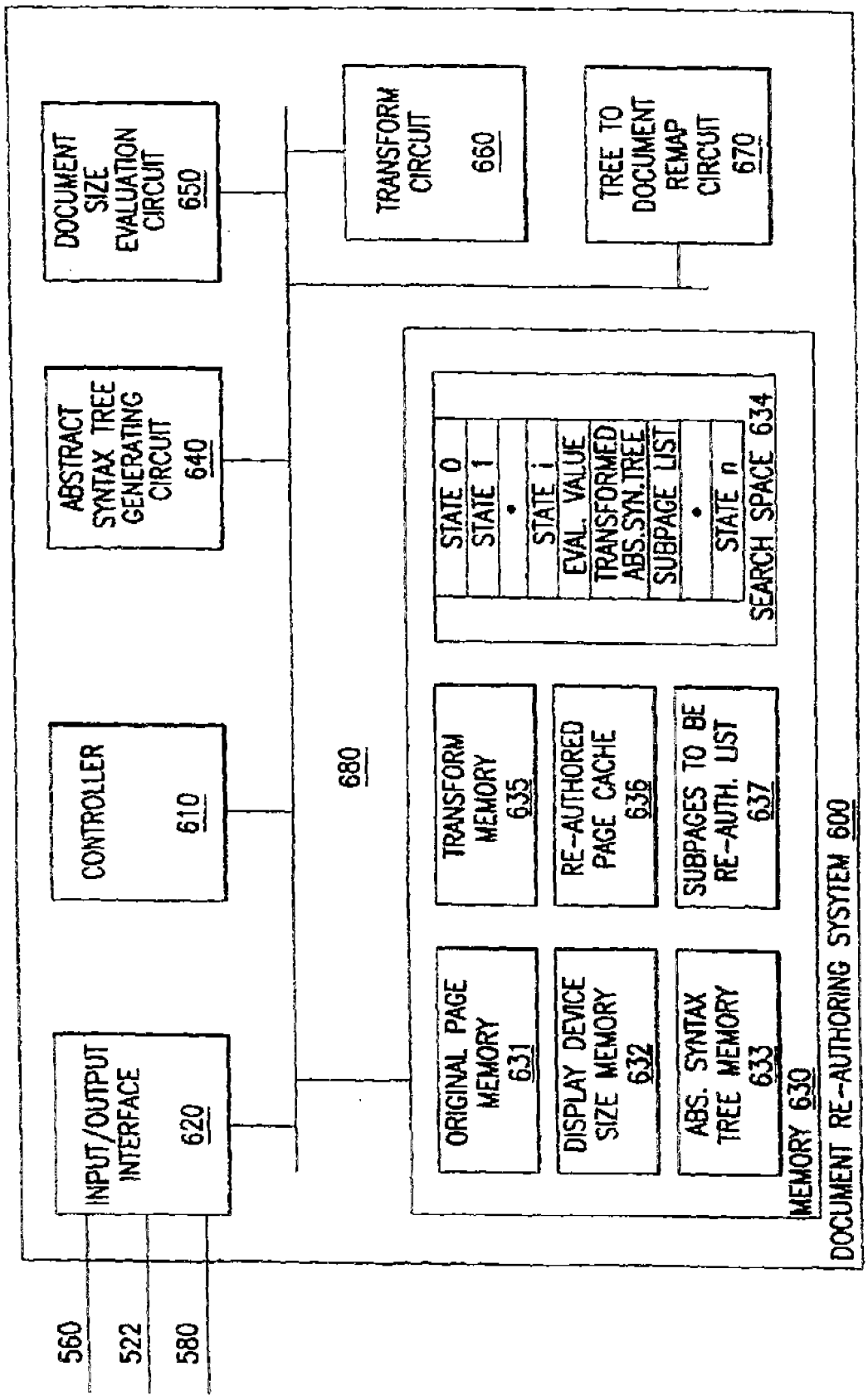

DOCUMENT RE-AUTHORING SYSTEM 600
MEMORY 630
ORIGINAL PAGE MEMORY 631
DISPLAY DEVICE SIZE MEMORY 632
ABS. SYNTAX TREE MEMORY 633
TRANSFORM MEMORY 635
RE-AUTHORED PAGE CACHE 636
SUBPAGES TO BE RE-AUTH. LIST 637
SEARCH SPACE 634
STATE 0
STATE 1
STATE i
EVAL. VALUE
TRANSFORMED ABS.SYN.TREE
SUBPAGE LIST
STATE n
680
INPUT/OUTPUT INTERFACE 620
CONTROLLER 610
ABSTRACT SYNTAX TREE GENERATING CIRCUIT 640
DOCUMENT SIZE EVALUATION CIRCUIT 650
TRANSFORM CIRCUIT 660
TREE TO DOCUMENT REMAP CIRCUIT 670
560
522
580

FIG.8

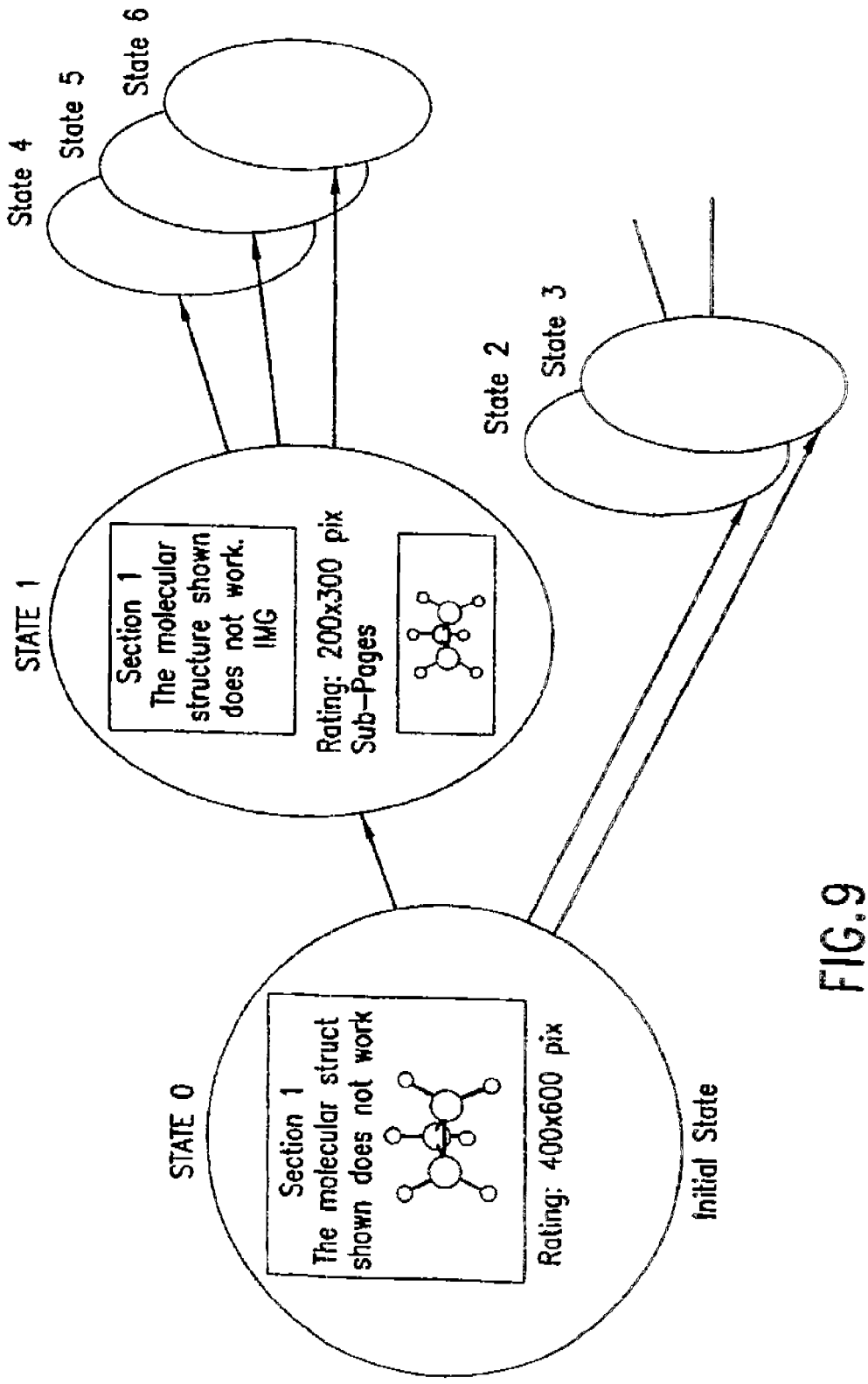

State 4
State 5
State 6
State 2
State 3
STATE 1
Section 1
The molecular structure shown does not work.
IMG
Rating: 200x300 pix
Sub-Pages
STATE 0
Section 1
The molecular struct shown does not work
Rating: 400x600 pix
Initial State

FIG.9

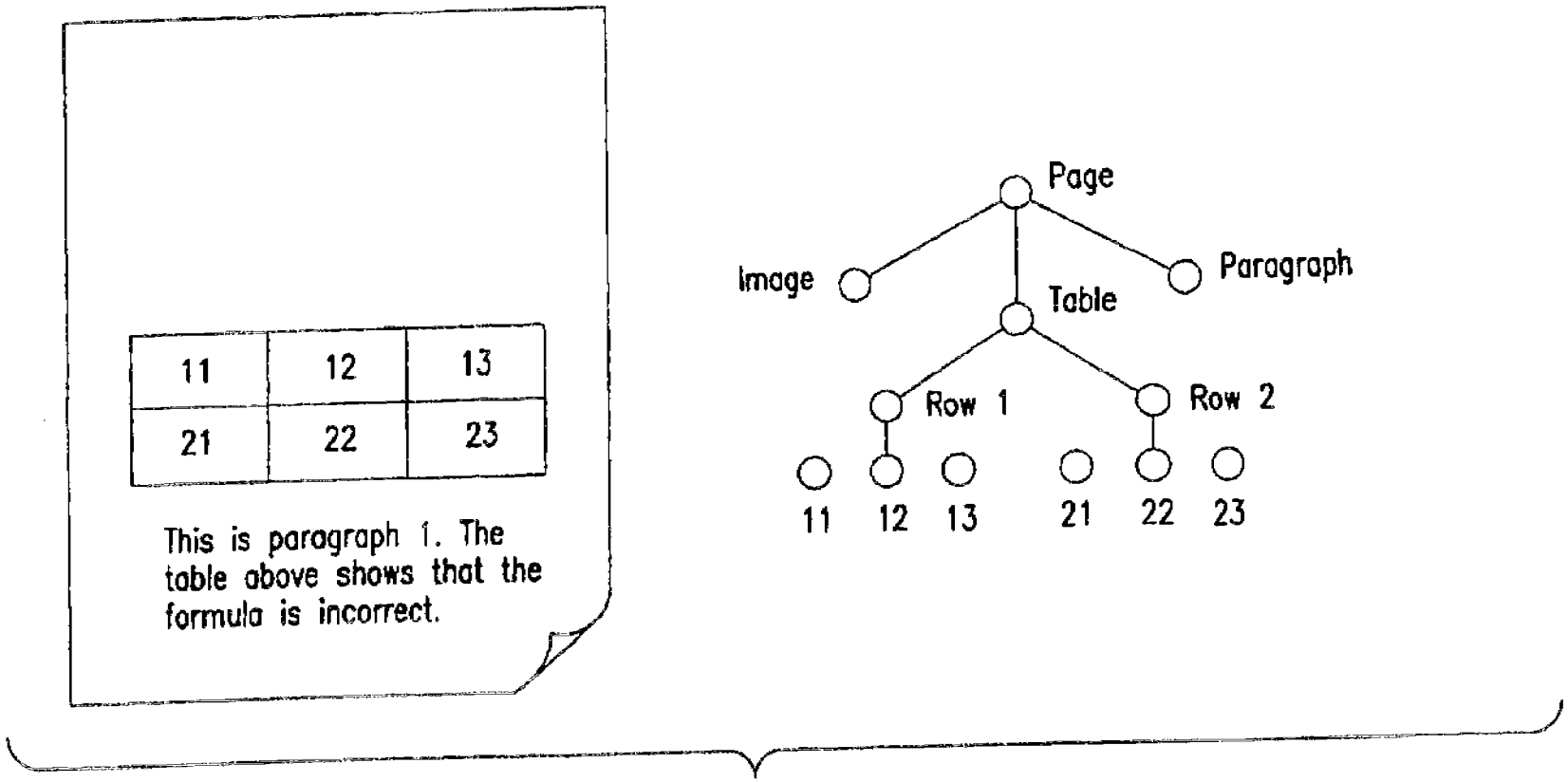


FIG.10

EP 0 949 571 A2

IPR2023-00330 Page 00804

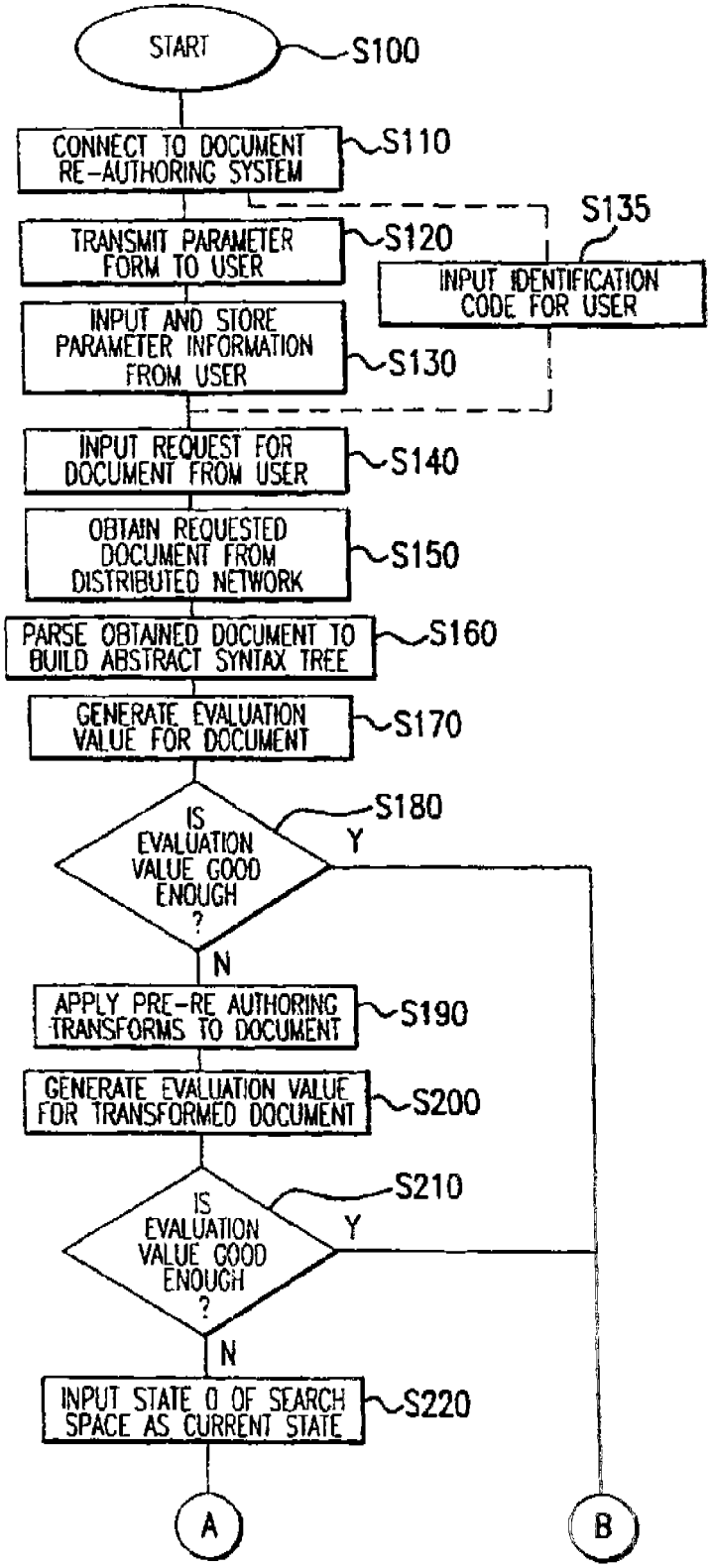

START
S100
CONNECT TO DOCUMENT RE-AUTHORING SYSTEM
S110
TRANSMIT PARAMETER FORM TO USER
S120
INPUT AND STORE PARAMETER INFORMATION FROM USER
S130
S135
INPUT IDENTIFICATION CODE FOR USER
INPUT REQUEST FOR DOCUMENT FROM USER
S140
OBTAIN REQUESTED DOCUMENT FROM DISTRIBUTED NETWORK
S150
PARSE OBTAINED DOCUMENT TO BUILD ABSTRACT SYNTAX TREE
S160
GENERATE EVALUATION VALUE FOR DOCUMENT
S170
IS EVALUATION VALUE GOOD ENOUGH ?
S180
Y
N
APPLY PRE-RE AUTHORING TRANSFORMS TO DOCUMENT
S190
GENERATE EVALUATION VALUE FOR TRANSFORMED DOCUMENT
S200
IS EVALUATION VALUE GOOD ENOUGH ?
S210
Y
N
INPUT STATE 0 OF SEARCH SPACE AS CURRENT STATE
S220
A
B

FIG.11A

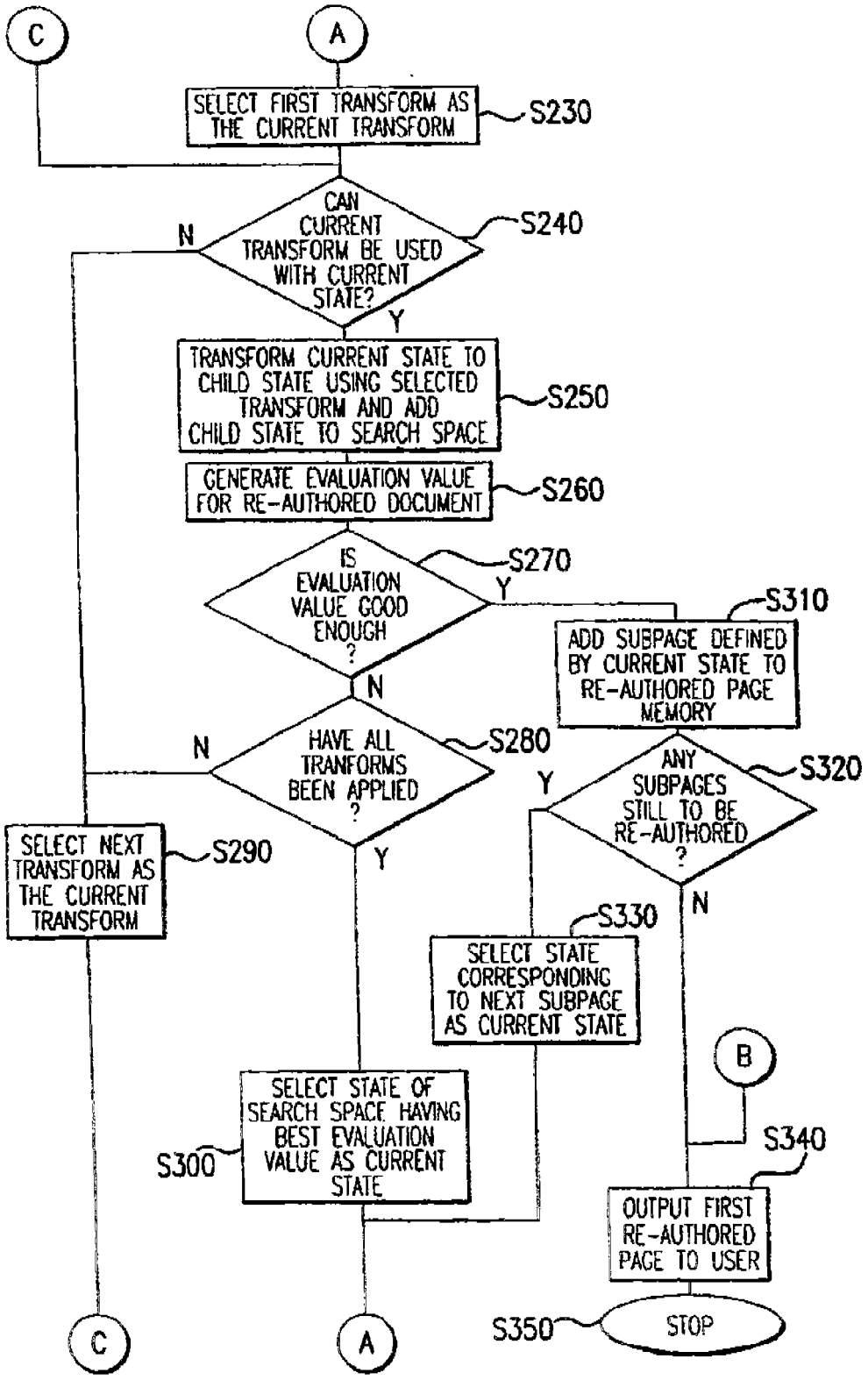

C
A
SELECT FIRST TRANSFORM AS THE CURRENT TRANSFORM
S230
CAN CURRENT TRANSFORM BE USED WITH CURRENT STATE?
S240
N
Y
TRANSFORM CURRENT STATE TO CHILD STATE USING SELECTED TRANSFORM AND ADD CHILD STATE TO SEARCH SPACE
S250
GENERATE EVALUATION VALUE FOR RE-AUTHORED DOCUMENT
S260
IS EVALUATION VALUE GOOD ENOUGH ?
S270
Y
N
S310
ADD SUBPAGE DEFINED BY CURRENT STATE TO RE-AUTHORED PAGE MEMORY
HAVE ALL TRANFORMS BEEN APPLIED ?
S280
N
Y
ANY SUBPAGES STILL TO BE RE-AUTHORED ?
S320
Y
N
SELECT NEXT TRANSFORM AS THE CURRENT TRANSFORM
S290
S330
SELECT STATE CORRESPONDING TO NEXT SUBPAGE AS CURRENT STATE
B
SELECT STATE OF SEARCH SPACE HAVING BEST EVALUATION VALUE AS CURRENT STATE
S300
S340
OUTPUT FIRST RE-AUTHORED PAGE TO USER
S350
STOP
C
A

FIG.11B

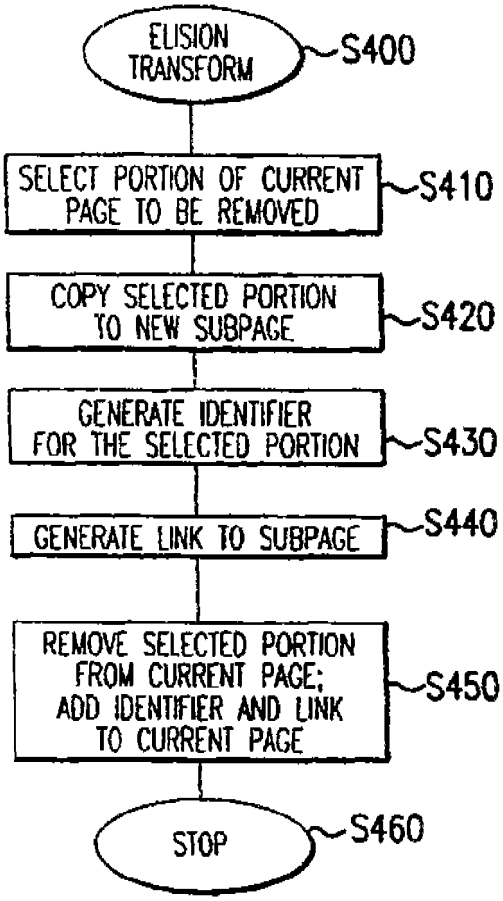

ELISION TRANSFORM
S400
SELECT PORTION OF CURRENT PAGE TO BE REMOVED
S410
COPY SELECTED PORTION TO NEW SUBPAGE
S420
GENERATE IDENTIFIER FOR THE SELECTED PORTION
S430
GENERATE LINK TO SUBPAGE
S440
REMOVE SELECTED PORTION FROM CURRENT PAGE; ADD IDENTIFIER AND LINK TO CURRENT PAGE
S450
STOP
S460

FIG.12

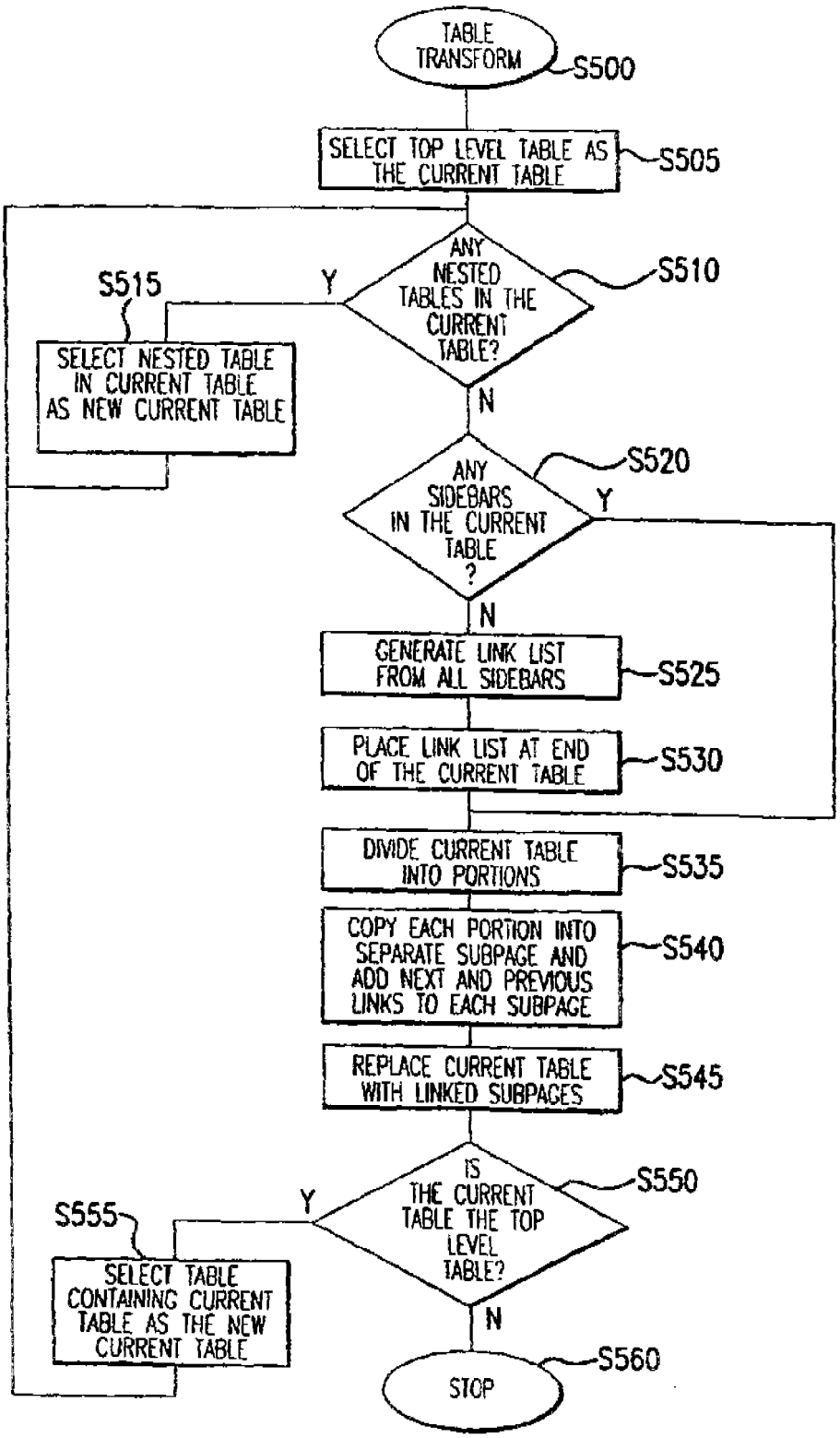


TABLE TRANSFORM
S500
SELECT TOP LEVEL TABLE AS THE CURRENT TABLE
S505
ANY NESTED TABLES IN THE CURRENT TABLE?
S510
Y
S515
SELECT NESTED TABLE IN CURRENT TABLE AS NEW CURRENT TABLE
N
ANY SIDEBARS IN THE CURRENT TABLE ?
S520
Y
N
GENERATE LINK LIST FROM ALL SIDEBARS
S525
PLACE LINK LIST AT END OF THE CURRENT TABLE
S530
DIVIDE CURRENT TABLE INTO PORTIONS
S535
COPY EACH PORTION INTO SEPARATE SUBPAGE AND ADD NEXT AND PREVIOUS LINKS TO EACH SUBPAGE
S540
REPLACE CURRENT TABLE WITH LINKED SUBPAGES
S545
IS THE CURRENT TABLE THE TOP LEVEL TABLE?
S550
Y
S555
SELECT TABLE CONTAINING CURRENT TABLE AS THE NEW CURRENT TABLE
N
STOP
S560

FIG.13

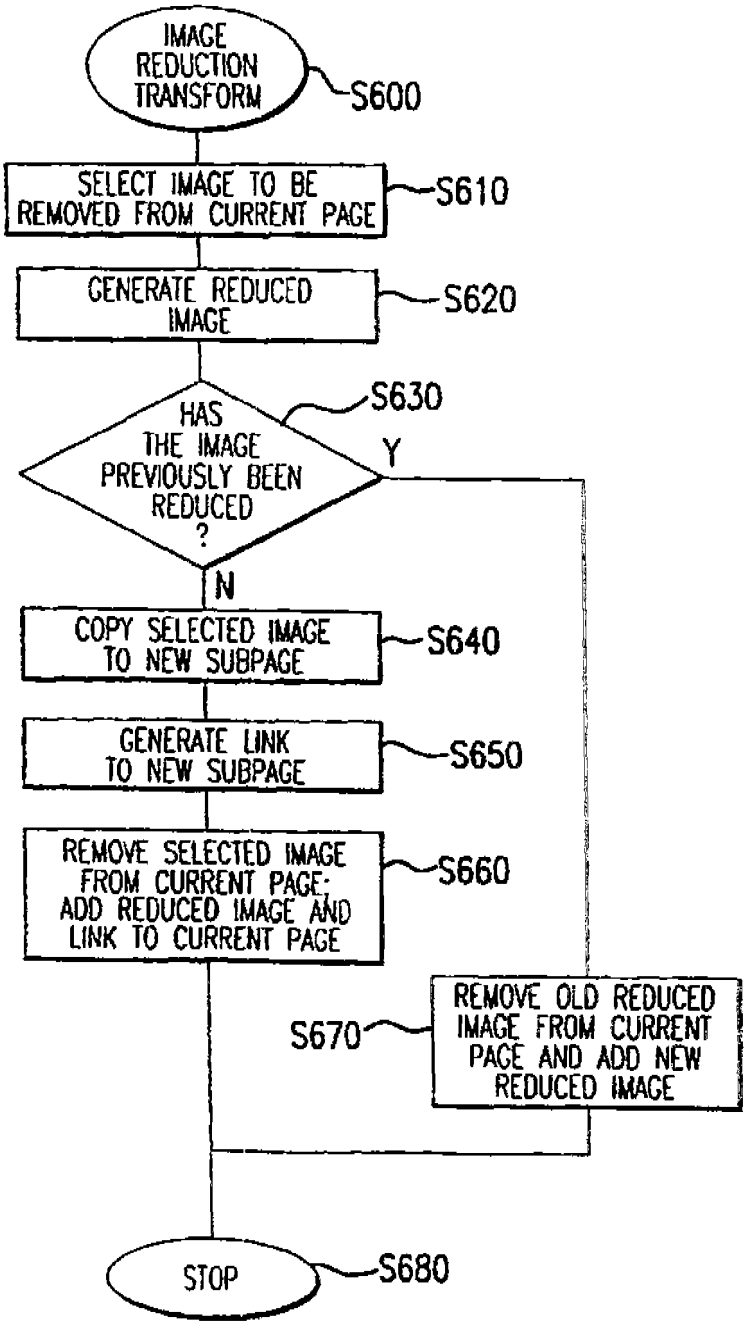


IMAGE REDUCTION TRANSFORM
S600
SELECT IMAGE TO BE REMOVED FROM CURRENT PAGE
S610
GENERATE REDUCED IMAGE
S620
HAS THE IMAGE PREVIOUSLY BEEN REDUCED ?
S630
Y
N
COPY SELECTED IMAGE TO NEW SUBPAGE
S640
GENERATE LINK TO NEW SUBPAGE
S650
REMOVE SELECTED IMAGE FROM CURRENT PAGE; ADD REDUCED IMAGE AND LINK TO CURRENT PAGE
S660
S670
REMOVE OLD REDUCED IMAGE FROM CURRENT PAGE AND ADD NEW REDUCED IMAGE
STOP
S680

FIG.14

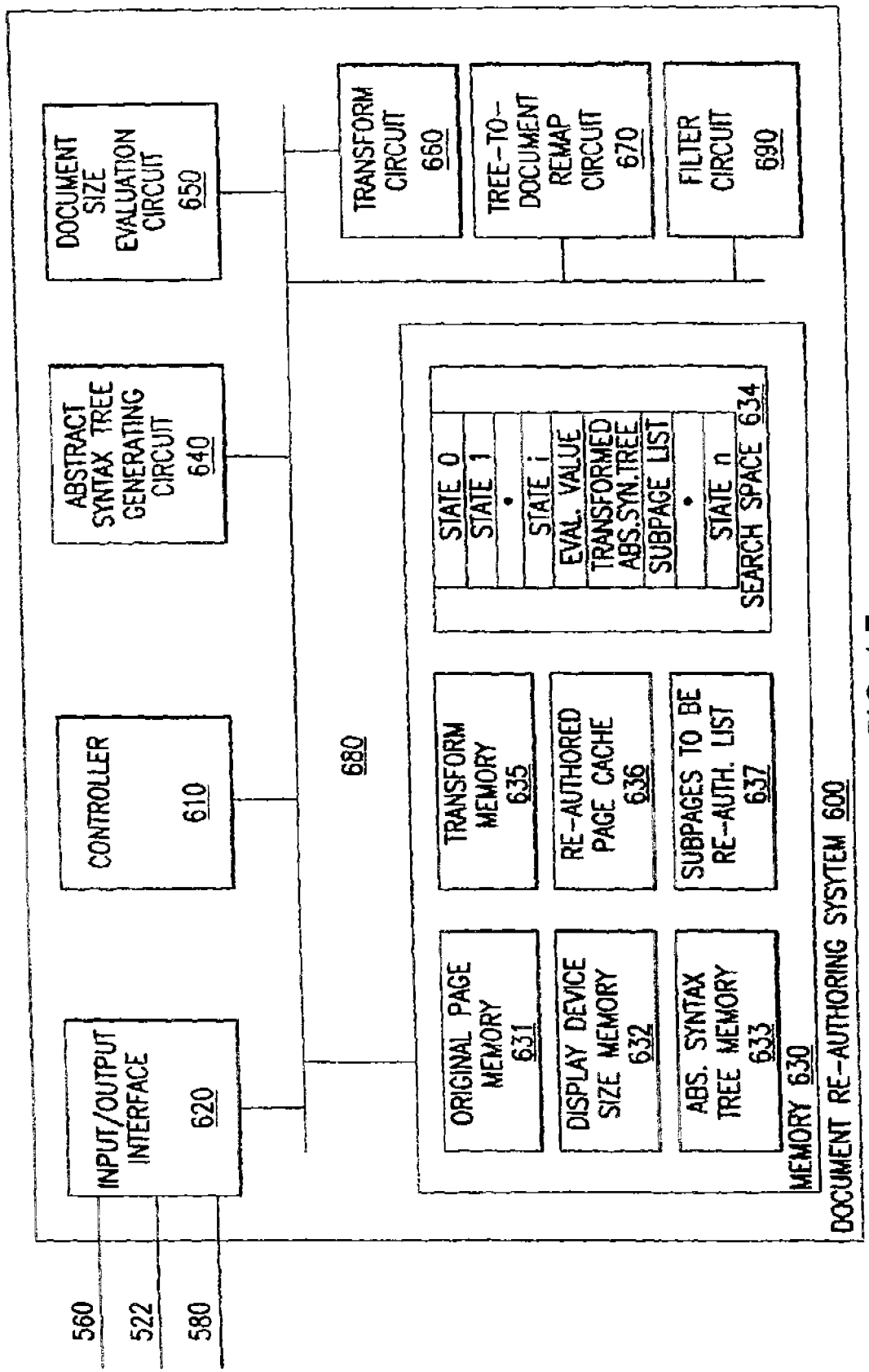

DOCUMENT RE-AUTHORING SYSTEM 600
MEMORY 630
ORIGINAL PAGE MEMORY 631
DISPLAY DEVICE SIZE MEMORY 632
ABS. SYNTAX TREE MEMORY 633
TRANSFORM MEMORY 635
RE-AUTHORED PAGE CACHE 636
SUBPAGES TO BE RE-AUTH. LIST 637
SEARCH SPACE 634
STATE 0
STATE 1
STATE i
EVAL. VALUE
TRANSFORMED ABS.SYN.TREE
SUBPAGE LIST
STATE n
680
INPUT/OUTPUT INTERFACE 620
CONTROLLER 610
ABSTRACT SYNTAX TREE GENERATING CIRCUIT 640
DOCUMENT SIZE EVALUATION CIRCUIT 650
TRANSFORM CIRCUIT 660
TREE-TO-DOCUMENT REMAP CIRCUIT 670
FILTER CIRCUIT 690
560
522
580

FIG.15

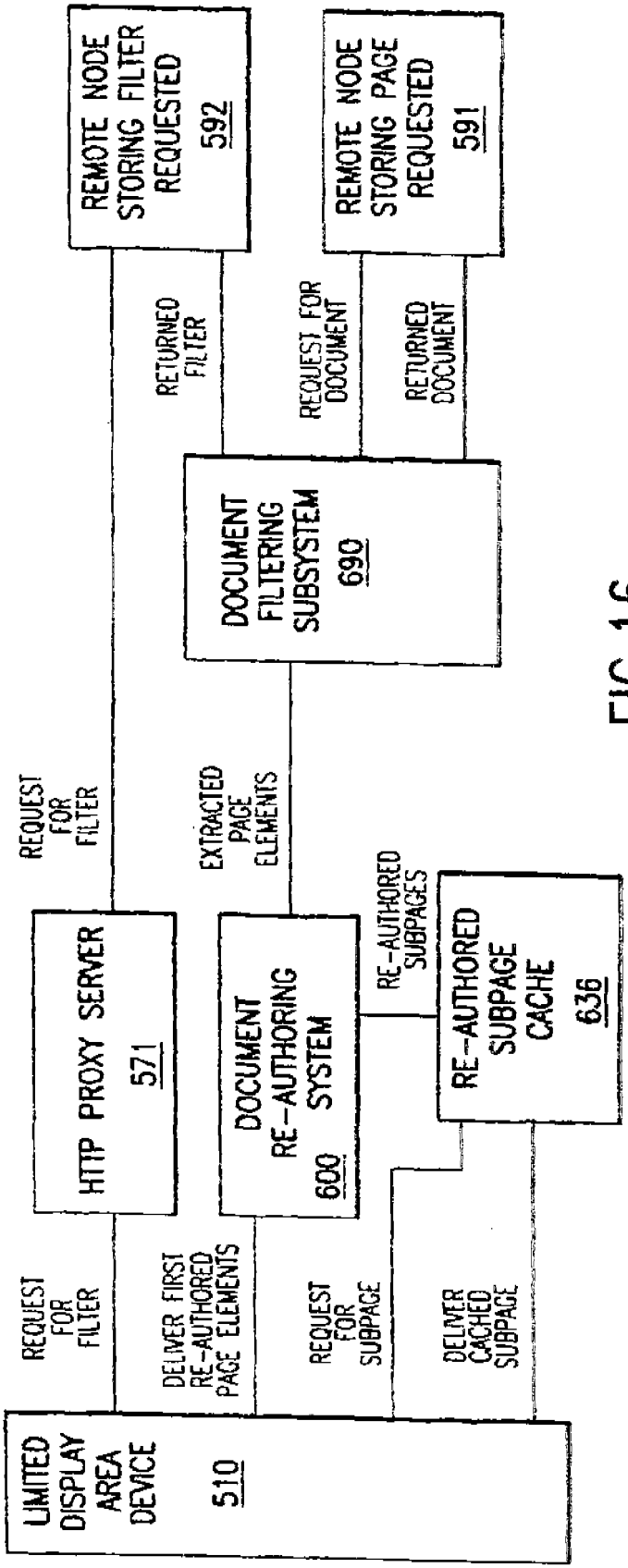

REMOTE NODE STORING FILTER REQUESTED 592
REMOTE NODE STORING PAGE REQUESTED 591
RETURNED FILTER
REQUEST FOR DOCUMENT
RETURNED DOCUMENT
DOCUMENT FILTERING SUBSYSTEM 690
REQUEST FOR FILTER
EXTRACTED PAGE ELEMENTS
RE-AUTHORED SUBPAGES
HTTP PROXY SERVER 571
DOCUMENT RE-AUTHORING SYSTEM 600
RE-AUTHORED SUBPAGE CACHE 636
REQUEST FOR FILTER
DELIVER FIRST RE-AUTHORED PAGE ELEMENTS
REQUEST FOR SUBPAGE
DELIVER CACHED SUBPAGE
LIMITED DISPLAY AREA DEVICE 510

FIG. 16

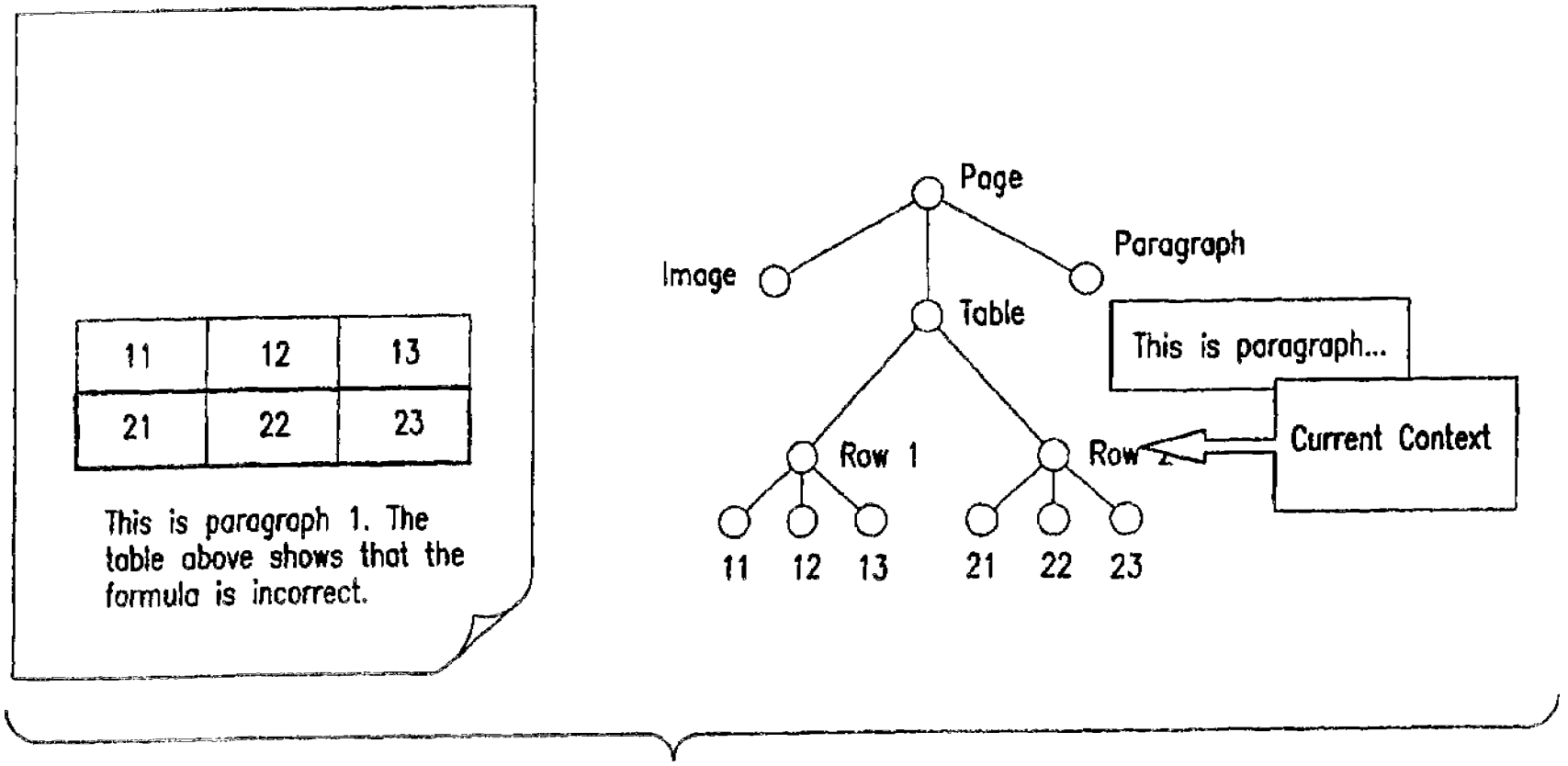


FIG.17

EP 0 949 571 A2

IPR2023-00330 Page 00812

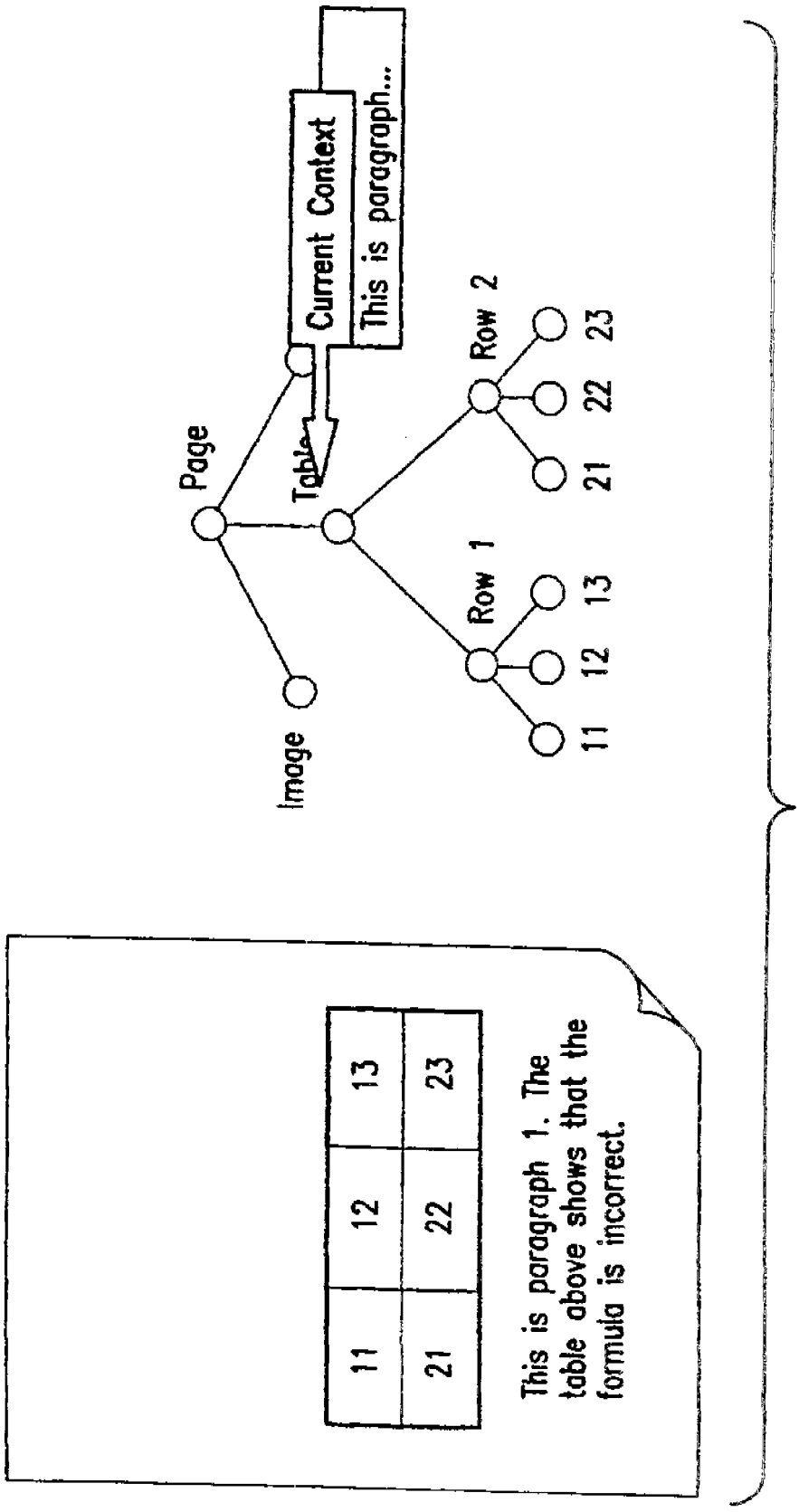

Page
Table
Current Context
This is paragraph...
Image
Row 1
11 12 13
Row 2
21 22 23
11
12
13
21
22
23
This is paragraph 1. The table above shows that the formula is incorrect.

FIG.18

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 949 571 A3

(12) **EUROPEAN PATENT APPLICATION**

(88) Date of publication A3:
05.01.2000 Bulletin 2000/01

(43) Date of publication A2:
13.10.1999 Bulletin 1999/41

(21) Application number: **99302718.4**

(22) Date of filing: **07.04.1999**

(51) Int Cl.[7]: **G06F 17/22**, G06F 17/30

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: **07.04.1998 US 80909 P**
29.01.1999 US 239295

(71) Applicant: **XEROX CORPORATION**
Rochester, New York 14644 (US)

(72) Inventors:
- **Bickmore, Timothy W.**
  Somerville, Massachusetts 02144 (US)
- **Girgensohn, Andreas**
  Menlo Park, California (US)
- **Schilit, William N.**
  Menlo Park, California 94025 (US)
- **Sullivan, Joseph W.**
  San Francisco, California 94107 (US)

(74) Representative: **Skone James, Robert Edmund**
GILL JENNINGS & EVERY
Broadgate House
7 Eldon Street
London EC2M 7LH (GB)

(54) Document re-authoring systems and methods for providing device-independent access to the world wide web

(57) An automatic re-authoring system and method re-author a document originally designed for display on a desktop computer screen for display on a smaller display screen, such as those used with a PDA or a cellular telephone. The automatic re-authoring system and method input a document to be re-authored and re-authoring parameters, such as display screen size, default font and the like. The automatic re-authoring system and method convert the document into a number of pages, where each page is fully displayable with only at most a minimal amount of scrolling on the display screen of the PDA or cellular phone. At each stage of the re-authoring, a number of different transformations are applied to the original document or a selected re-authored page. The selected re-authored page is the best page resulting from the previous re-authoring stage. The best page at each stage is determined based on the re-authoring parameters and the content of the document being re-authored.

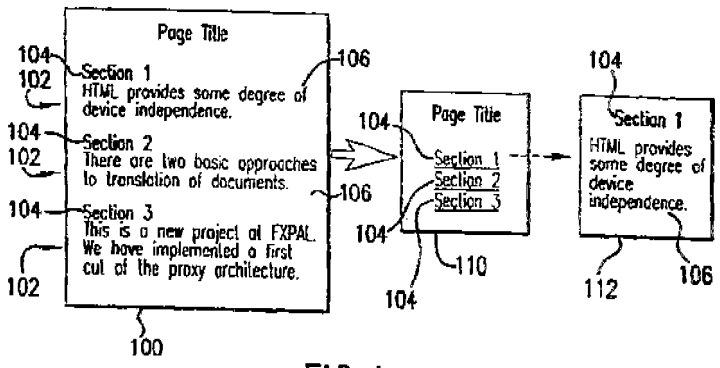


FIG.1

EP 0 949 571 A3

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 99 30 2718

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.6)
X	BICKMORE T W ET AL: "Digestor: device-independent access to the World Wide Web" COMPUTER NETWORKS AND ISDN SYSTEMS,NL,NORTH HOLLAND PUBLISHING. AMSTERDAM, vol. 29, no. 8-13, September 1997 (1997-09), page 1075-1082 XP004095305 ISSN: 0169-7552 * the whole document *	1-10	G06F17/22 G06F17/30
A	JOHNSON D: "Converting PC GUIs for nonPC devices" CIRCUIT CELLAR INK, FEB. 1998, CIRCUIT CELLAR INC, USA, no. 91, pages 40-42, 44 - 45, XP000852859 ISSN: 0896-8985 * page 40, left-hand column, line 1 - page 42, right-hand column, line 43 *	1,10	

TECHNICAL FIELDS SEARCHED (Int.Cl.6)

G06F

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	15 November 1999	Fournier, C

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06K 9/00

A1

(11) International Publication Number: WO 98/40842

(43) International Publication Date: 17 September 1998 (17.09.98)

(21) International Application Number: PCT/US98/04700

(22) International Filing Date: 11 March 1998 (11.03.98)

(30) Priority Data:

60/040,241	11 March 1997 (11.03.97)	US
09/038,562	10 March 1998 (10.03.98)	US

(71) Applicant: COMPUTER INFORMATION AND SCIENCES, INC. [US/US]; Suite 104, 3401 East University, Denton, TX 76208 (US).

(72) Inventors: CHAO, Hongyang; 1011 Chestnut #10, Denton, TX 76201 (US). HUA, Zeyi; 2197 S. Uecker #357, Denton, TX 76201 (US). FISCHER, Howard, P.; 3106 Saints Circle, Denton, TX 76201 (US). FISCHER, Paul, S.; 34 Timbergreen Circle, Denton, TX 76205 (US).

(74) Agents: SAMPLES, Kenneth, H. et al.; Fitch, Even, Tabin & Flannery, Room 900, 135 S. LaSalle, Chicago, IL 60603 (US).

(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, GM, GW, HU, ID, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, UZ, VN, YU, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SZ, UG, ZW), European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

Published

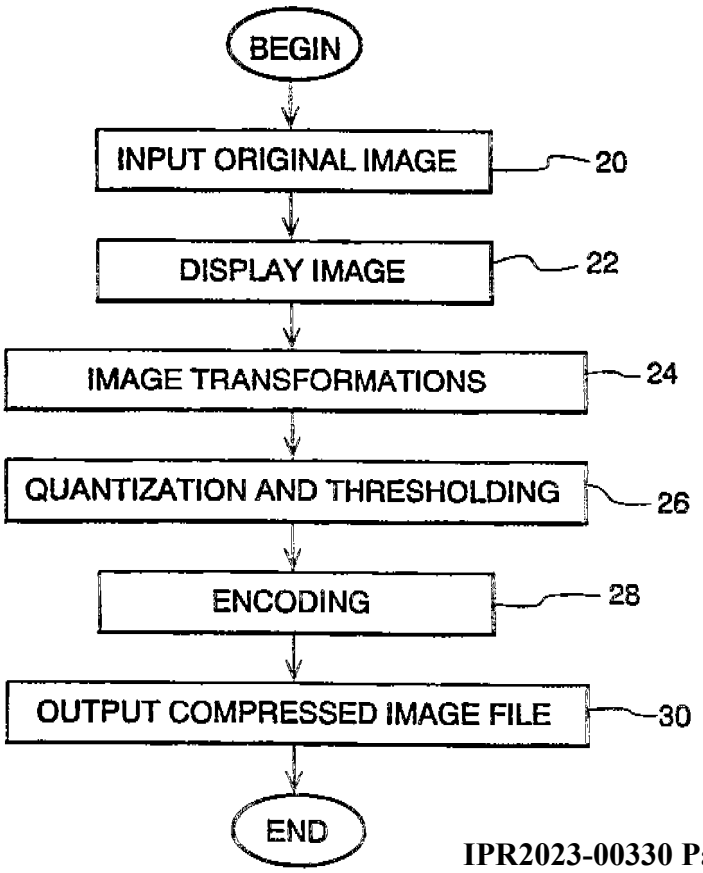
With international search report.

Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: SYSTEM AND METHOD FOR IMAGE COMPRESSION AND DECOMPRESSION

(57) Abstract

A wavelet-based image compression system and method are presented (24). Compression is accomplished by performing a wavelet transformation of an input digital image (20). The resulting wavelet coefficients are compared to a threshold value. Coefficients falling below the threshold are discarded. The remaining coefficients are quantized (26). The quantized coefficients are then compressed using an entropy encoding technique (28).



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

SYSTEM AND METHOD FOR IMAGE COMPRESSION AND DECOMPRESSION

This application claims the benefit of U.S. Provisional Application No. 60/040,241, filed March 11, 1997, System and Method for Still Image Compression, which is incorporated herein by reference.

Field of the Invention

The present invention relates generally to digital image compression/decompression, and particularly, to a wavelet-based system and method of image compression and decompression.

Background of the Invention

Nearly every computer user needs to store, transfer, and view images. These images include still images, or pictures, as well as video images, which are sequences of still images displayed in a manner that depicts motion. The enormous size of image files leads to serious file management limitations. For example, a single still image (equivalent to a video frame) displayed by a rectangular array of picture elements (pixels) arranged in 640 rows and 800 columns, with the color of each pixel represented by twenty-four bits, would require over 1.5 megabytes of digital memory to store. One solution to this problem is high-quality data compression technology. Essentially, image compression mathematically transforms a grid of image pixels into a new, much smaller set of digital values holding the information needed to regenerate the original image or data file.

In addition imaging systems, compression technology can be incorporated into "video on demand" systems, such as video servers. Compression technology can also be applied to streaming video, which is the real-time capture and display of video images over a communications link. Applications for streaming video

include video telephones, remote security systems, and other types of monitoring systems.

Several standards for compressing real-time video currently exist. The H.263 standard for real-time video is an industry standard based upon the discrete co-sign transform (DCT). DCT is also the basis for both of the public domain image compression standards, MPEG (Motion Picture Experts Group) and JPEG (Joint Photographic Experts Group). Although the DCT approach performs interframe coding adequately, its compression ratio and speed can be improved upon.

Various other types of data compression have been developed in recent years. Conventional data compression techniques are generally referred to as being either "lossless" or "lossy", depending upon whether data is discarded in the compression process. Examples of conventional lossless compression techniques include Huffman encoding, arithmetic encoding, and Fano-Shannon encoding. With a lossless compression, the decompression process will reproduce all bits of the original image. Lossless compression is important for images found in such applications as medical and space science. In such situations, the designer of the compression algorithm must be very careful to avoid discarding any information that may be required or even useful at some later point.

Lossy compression, in contrast, provides greater efficiency over lossless compression in terms of speed and storage, as some data is discarded. As a result, lossy techniques are employed where some degree of inaccuracy relative to the input data is tolerable. Accordingly, lossy compression is frequently used in video or commercial image processing. Two popular lossy image compression standards are the MPEG and JPEG compression methods.

The wavelet transform has proven to be one of the most powerful tools in the field of data compression. Theoretically, the wavelet transformation is lossless,

but since all computers have only finite precision even when using floating point calculations, most of the transformations are lossy in practice. On the other hand, integer calculations are much faster than floating point for virtually all computers; and integer computations are much easier to implement in hardware, which is more important in some applications. While integers require less memory than real numbers, the direct use of integers in conventional wavelet transforms and their inverses typically causes an unacceptable loss of accuracy. Accordingly, there is a need for a wavelet-based compression technique that permits lossless or near-lossless data compression, yet retains the speed and memory advantages of integer arithmetic.

Summary of the Invention

It is an advantage of the present invention to provide a system and method of wavelet-based data compression that permits integer computations in a computer without significant loss of accuracy. This is accomplished by using an integer reversible wavelet transform that possesses a property of precision preservation (PPP). The integer reversible transform greatly reduces the computer resources needed to compress and decompress images, as well as the time required to perform the same.

It is an advantage of the present invention to provide a system and method of wavelet-based image compression that is suitable for both still and video images.

It is also an advantage of the present invention to provide a system and method of image compression that is capable of selectively performing lossless and lossy compression of either color or gray-scale images.

According to one aspect of the invention, a wavelet-based image compression method can be implemented using a software program. Compression is accomplished by

performing a wavelet transform on an input digital image. The resulting wavelet components are compared to a threshold value; coefficients falling below the threshold are discarded. The remaining coefficients are quantized. The quantized coefficients are then compressed using an entropy encoding technique, such as arithmetic, run length, or Huffman encoding, or a combination of Huffman and run length encoding. The wavelet transform can be an integer reversible wavelet transform derived using a lifting scheme or correction method, while the quantization scheme can be sub-band oriented. To further enhance the speed of the compression scheme, input color image pixels can be reduced using a color table. In addition, color pixels can be transformed between color spaces prior to wavelet transformation.

According to another aspect of the invention, a corresponding method of decompression is provided.

According to another aspect of the present invention, a compression method is provided that allows user selected portions of an image to compressed to different image qualities, whereby permitting non-uniform image compression.

According to another aspect of the present invention, a compression method is provided that permits compression quality to be based on image specific parameters.

According to another aspect of the present invention, a method of compressing images using a "split and merge" technique is provided.

According to further aspect of the present invention, an image compression system includes a compressor configured to generate a compressed image based on an integer wavelet transform derived using either a lifting scheme or correction method. The compressor can be implemented using one or more electronic components, such as application specific integrated circuits (ASICs), microprocessors, discrete

logic components, or any combination of the aforementioned.

According to another aspect of the present invention, a corresponding image decompression system is provided.

Brief Description of the Drawings

The invention is pointed out with particularity in the appended claims. However, other features of the invention will become more apparent, and the invention will be best understood by referring to the following detailed description in conjunction with the accompanying drawings, in which:

FIG. 1 illustrates a flow diagram for a method of compressing an image that is in accordance with an embodiment of the present invention;

FIGS. 2-4 depict wavelet coefficients for various levels of decomposition;

FIG. 5 illustrates a flow diagram of a method of decompressing an image that has been compressed using the method of FIG. 1;

FIG. 6 is a block diagram of a system that can incorporate a software program implementing any of the methods shown in FIGS. 1, 5, and 8-13 in accordance with a second embodiment of the present invention;

FIG. 7 is a block diagram of a system for compressing and decompressing an image in accordance with another embodiment of the present invention;

FIG. 8 illustrates a flow diagram of a method compressing an image that is in accordance with a further embodiment of the present invention;

FIG. 9 illustrates a flow diagram of a method for decompressing an image that has been compressed according to the method of FIG. 8;

FIG. 10 illustrates a flow diagram of a method of compressing an image in accordance with a further embodiment of the present invention;

FIG. 11 illustrates a flow diagram of a method of decompressing an image that has been compressed according to the method of FIG. 10;

FIG. 12 illustrates a flow diagram of a method of compressing an image that is in accordance with a further embodiment of the present invention; and

FIG. 13 illustrates a flow diagram of a method for decompressing an image that has been compressed according to the method of FIG. 12.

Detailed Description of the Preferred Embodiments

Referring now to the drawings, and in particular to FIG. 1, there is shown a flow diagram of a method for compressing an image that conforms to a first embodiment of the invention. In step 20, a digital image is received from an image source. The digital image consists of a matrix of values representing an array of pixels. Specifically, the array of pixels represents a still image or a frame from a video image. In step 22, the image is optionally displayed on an appropriate viewing device, such as a computer or video display unit having a flat panel or cathode ray tube (CRT). Next, in step 24, color and wavelet transformations of the image take place. The image transformations involved in this step include color transform for color images only, and wavelet transform for both gray level images and color images. In step 26, the values representing the transformed images are quantized and compared to thresholds. Values falling outside the threshold are discarded. In step 28, the remaining quantized values are encoded to remove redundant information, creating a compressed image file. Next, in step 30 the compressed image file is generated as output.

Referring to the color transformation of step 24, digital color images are typically based on an RGB color model, such as is commonly used with TIFF or BMP images. In order to get a higher compression ratio, the

RGB pixels are transformed to other color models, such as YIQ or YUV models. The method can convert RGB inputs into YIQ or YUV color spaces according to the following relationships.

RGB to YIQ:

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.596 & -0.275 & 0.321 \\ 0.212 & -0.523 & 0.311 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

RBG to YUV:

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.148 & -0.289 & 0.439 \\ 0.615 & -0.515 & -0.1 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

In the YIQ color space, there is one luminescence (Y) and two color planes (I, Q). The Y component is critical, while the I-Q components are less sensitive to error introduced by data compression.

The wavelet transform (also referred to as wavelet decomposition) operates on the converted color space signals. The purpose of the wavelet transform is to represent the original image by a different basis to achieve the objective of decorrelation. There are many different wavelet transforms that can be used in this step. For instance, the reversible integer wavelet transform described herein below is a preferred wavelet transform. However, to develop a better understanding of the preferred transform, the following alternative wavelet transform is first described.

Let $C^0 = [C^0_{jk}]$ (j - 0, ..., M-1; k = 0, ..., N-1) represent the original, uncompressed image, where M and N are integers which have the common factor 2^L (L is a positive integer). A one-level wavelet decomposition, where L = 1, results in the four coefficient quadrants as

shown in Figure 2. Each quadrant represents a set of wavelet coefficients.

Quadrant C^1 represents the blurred image of the original image C^0, where $C^1 = [C^1_{jk}]\ (j=0,\ldots,\frac{M}{2}-1;\ k=0,\ldots,\frac{N}{2}-1)$. HD^1 represents the horizontal high frequency part of C^0, while VD^1 represents the vertical high frequency part of C^0, and DD^1 represents the diagonal high frequency part of C^0. The decomposition can be iteratively repeated L times to obtain different levels of decomposition. For example, for L = 2, C^0 is set to equal C^1. The iterative formula for computing a decomposition is given as follows:

(1) Let $\overline{C}^0 = rC^0$, r>0 is a factor which can be changed for different needs.

(2) Transform for image columns:

For k=0,...,N-1, calculate

$$\begin{cases} d^1_{0k} = \dfrac{\overline{C}^0_{1k} - \overline{C}^0_{0K}}{2}, \\ d^1_{jk} = \dfrac{1}{4}\left(\overline{C}^{-0}_{2j-1,k} - 2\overline{C}^{-0}_{2j,k} + \overline{C}_{2j+1,k}\right), j=1,\ldots,\dfrac{M}{2}-1. \end{cases} \tag{3.1.1}$$

For k=0,...,N-1, calculate

$$\begin{cases} \tilde{C}^1_{0k} = \overline{C}^0_{1,k} - \dfrac{d_{ok} + d_{1,k}}{2}, \\ \tilde{C}^1_{jk} = \overline{C}_{2j+1,k} - \dfrac{d^1_{jk} + d_{j+1,k}}{2}, j=1,\ldots,\dfrac{M}{2}, \\ \qquad \dfrac{\tilde{C}^{-1}_{N-2}}{2}, k = \tilde{C}^{-0}_{N-1,k} - \dfrac{d_{N-2}}{2}, k. \end{cases} \tag{3.1.2}$$

(3) Transform for rows:

For j = 0, . . . M/2 - 1, computing

$$\begin{cases} hd^1_{j,0}=\dfrac{\tilde{c}_{j1}-\tilde{c}^1_{j0}}{2}, \\ hd^1_{jk}=\dfrac{1}{4}\left(\tilde{c}^1_{j,2k-1}-2\tilde{c}_{j,2k+1},+\tilde{c}_{j,2k+1}\right), k=1\ldots\dfrac{N}{2}-1. \end{cases} \tag{3.1.3}$$

and

$$\begin{cases} c^1_{jo}=\tilde{c}^1_{j,1}-\dfrac{hd^1_{j0}+hd^1_{j1}}{2}, \\ c^1_{jk}=c^1_{j,2k+1}-\dfrac{hd^1_{j,k}+hd^1_{j,k+1},}{2},\ldots,\dfrac{M}{2}-2, \\ c_{j,\frac{N-2}{2}}=\tilde{c}^1_{j,N-1}hd_{j,\frac{N-2}{2}}, \end{cases} \tag{3.1.4}$$

For j = 0, . . . , M/2 - 1, computing

$$\begin{cases} dd^1_{j,0}=\dfrac{\tilde{d}^1_{j,1}-\tilde{d}^1_{j0}}{2}, \\ dd^1_{jk}=\dfrac{1}{4}\left(\tilde{d}^1_{j,2k-1}-2\tilde{d}_{j,2k}+\tilde{d}^1_{j,2k+1}\right), k=1\ldots\dfrac{N}{2}-1. \end{cases} \tag{3.1.5}$$

and

$$\begin{cases} vd^1_{j0}=\tilde{d}^1_{j,1}-\dfrac{dd^1_{j0}+dd^1_{j1}}{2}, \\ vd^1_{jk}=\tilde{d}^1_{j,2k+1}-\dfrac{dd^1_{j,k}-dd^1_{j,k+1}}{2}, k=1,\ldots,\dfrac{M}{2}-2, \\ vd_{j,\frac{M-2}{2}}=\tilde{d}_{N-1,k}-dd^1_{j,\frac{N-2}{2}}, \end{cases} \tag{3.1.6}$$

(4) $C^1=[c^1_{j,k}]$, $HD^1=[hd^1_{j,k}]$, $VD^1[vd^1_{j,k}]$ and $DD=[dd^1_{j,k}]$, $j=0,\ldots,\dfrac{M}{2}-1$ $k=0,\ldots,\dfrac{M}{2}-1$.

Remark: If it is necessary, we also can use matrix multiply Wavelet Coefficient Image of 1 levels=$W_1 C^{\circ} W_1^T$.

Here, W^1 is the transform matrix for 1 level wavelet decomposition.

FIG. 3 depicts a three-level wavelet decomposition, where L = 3.

In step 26, the first loss in accuracy occurs. Both thresholding and quantization reduce accuracy with which the wavelet coefficients are represented. In step 26, the wavelet coefficients are matched against threshold values, and if the values are less than the established threshold values specified, then the resultant value is set to zero.

An important feature of the invention is that the wavelet coefficients are then quantized to a number of levels depending upon which quadrant is being processed, and the desired compression or quality factor. This can be very important in image compression, as it tends to make many coefficients zeros, especially those for high spatial frequencies, which reduces the size of a compressed image.

A multilevel uniform thresholding method can be used as described below.

Let $T = (t_1, \ldots, t_L, t_{L+1})$ be the chosen thresholds, where t_t is the threshold for 1 the (1=I, ..., L) level and t_{L+1} is a threshold for blurred image C^L. Thresholding sets every entry in the blocks C^L, HD^1, VD^1 and DD^1 (1 = I, ... L) to be zero if its absolute value is not greater than the corresponding threshold.

For color images, three threshold vectors which correspond three different color planes, such as y, I and Q, are used.

The step of quantization essentially scales the wavelet coefficients and truncates them to a predetermined set of integer values. The quantization table shown in Table 1 can be used.

q^1_{HD} q^1_{VD} q^1_{DD}	q^2_{HD} q^2_{VD} q^2_{DD}	... ...	q^L_{HD} q^L_{VD} q^L_{DD}	q_c^{L+1}

TABLE 1

In Table 1, the entries q^1_{HD} are quantization factors for blocks HD^1 (1 = I, ... L), q^1_{VD} and q^1_{DD} for blocks VD^1 and DD^1 (1 = I, ..., L) respectively, and the factor q_c^{L+1} is for the most blurred image C^L. The factors can be integers between 0 and 255. The quantization scheme for the block HD^1 (1 = I, ..., L) is

$$\overline{hd}^1_{j,k}=round\ \frac{hd^1_{j,k}\cdot q^1_{HD}}{\max^1_{HD}},\ j=0,\ldots,\frac{M}{2^1}-1;k=0,\ldots,\frac{N}{2^1}-1. \quad (3.2.1)$$

Here, $\overline{hd}^1_{j,k}(j=0,\ldots,\frac{M}{2^1}-1;\ ,k=0,\ldots,\frac{N}{2^1}-1$ *are quantized wavelet coefficients of block* $HD^1(1=1,\ldots,L)$

$$\max^1_{HD}=\max_{\substack{0\le j\le (M/2^1-1)\\ 0\le k\le (N/2^1-1)}}(|hd^1_{j,k}|),$$

and the function round(x) gives the nearest integer of x. Equation (3.2.1) is used for quantization of the other blocks (quadrants).

For color images, there are three separate quantization tables for the different color bands.

In step 28, entropy compression is applied to the resultant coefficients using either Arithmetic, Run Length, or Huffman, or Huffman and Run Length combined. The compression algorithm can be selected at run-time by the user, based on the desired compression ratio and the amount of time required to get the selected level of compression. The encoding step includes the entropy compression as well as coefficient rearranging.

An alternative process to that shown in FIG. 1 includes an optional down sampling of the IQ color planes. This down sampling may be done once or twice to produce two image planes either one-fourth or one-sixteenth the size of the original plane. If the down sampling is done, it will be accomplished prior to the wavelet transform of step 24. The down sampling reduces the compression time and size of the image file.

FIG. 5 shows a corresponding method for decompressing an image compressed using the method of FIG. 1. In step 40, the compressed image file is input. In step 42, the image is decoded. Next, in step 44 the values are de-quantized. Next, in step 46 inverse color and wavelet transformations are performed on the de-quantized data. In step 48, optional image post-processing takes place to refine the decompressed image. In step 50, the decompressed image is displayed.

The decoding of step 42 is the inverse operation of the encoding of step 28. Similarly, it can be divided into two parts: Entropy decoding (Huffman or arithmetic), and coefficient rearranging.

The decoding step produces quantized wavelet coefficients in 3*L+1 blocks. Dequantizing (step 44) uses the same quantization table as quantizing (Table 1), and the scheme as follows: for l = I, ..., L

$$\underline{hd}^{l}_{j,k} = \frac{\overline{hd}^{l}_{j,k} \cdot \max^{l}_{HD}}{q_{HD}}, j=0,\ldots,\frac{M}{2^{l}}-1;\ k=0,\ldots,\frac{N}{2^{l}}-1. \quad \textbf{(4.2.1)}$$

Equation (4.2.1) produces the approximate coefficients for the blocks HD^{l} (l = I, ..., L), which are shown in FIG. 3. The dequantizing scheme for other blocks is similar to 4.1.2).

In step 46, the inverse wavelet transform, also referred to as wavelet reconstruction, is performed prior to the inverse color transformation. FIG. 4 depicts a one-level wavelet reconstruction.

The wavelet reconstruction can be iteratively performed for various levels of decomposition, according to the following equations.

(1) Inverse transform for rows:

For j=0, ..., $\frac{M}{2}$ -1, calculate

$$\begin{cases} \tilde{d}^1_{j,1}=\upsilon d^1_{j,0}+\dfrac{dd^1_{j,0}+dd^1_{j,1}}{2} \\ \tilde{d}^1_{j,2k+1}=\upsilon d^1_{jk}+\dfrac{dd^1_{j,k}-dd^1_{j,k+1}}{2}, k=1,\ldots,\dfrac{N}{2}-2, \\ \tilde{d}_{N-1,k}=\upsilon d_{j,\frac{N-1}{2}}+dd^1_{j,\frac{N-2}{2}}. \end{cases} \quad (4.3.1)$$

and

$$\begin{cases} \tilde{d}^1_{j,0}=d^1_{j,1}-2dd^1_{j,0}, \\ \tilde{d}^1_{j,2k}=\dfrac{\tilde{d}^1_{j,2k-1}+\tilde{d}^1_{j,2k+1}}{2}-2dd^1_{j,k},\ k=1,\ldots,\dfrac{N}{2}-1. \end{cases} \quad (4.3.2)$$

For j=0, ..., M/2 - 1, calculate

$$\begin{cases} \tilde{c}^1_{j,1}=c^1_{j0}+\dfrac{hd^1_{j0}+hd^1_{j1}}{2} \\ \tilde{c}^1_{j,2k+1}=c^1_{jk}+\dfrac{hd^1_{j,k}-hd^1_{j,k+1}}{2}, k=1,\ldots,\dfrac{N}{2}-2, \\ \tilde{c}_{j,N-1}=c_{j,\frac{N-2}{2}}+hd^1_{j,\frac{N-2}{2}}. \end{cases} \quad (4.3.3)$$

and

$$\begin{cases} \tilde{c}^1_{j,0}=c^1_{j,1}-2hd^1_{j,0}, \\ \tilde{c}^1_{j,2k}=\dfrac{1}{2}(\tilde{c}^1_{j,2k-1}+\tilde{c}^1_{j,2k+1})-2hd^1_{jk},\ k=1\ldots,\dfrac{N}{2}-1. \end{cases} \quad (4.3.4)$$

(2) Inverse transform for column:

For k=0, ..., N - 1, calculate

and

$$\begin{cases} \overline{c}^0_{1,k}=\tilde{c}^1_{0k}+\dfrac{\tilde{d}^1_{0k}+\tilde{d}^1_{1,k}}{2}, \\ \overline{c}^0_{2j+1,k}=\tilde{c}^1_{jk}+\dfrac{\tilde{d}^1_{j,k}-\tilde{d}^1_{j+1,k}}{2}, j=1,\ldots,\dfrac{M}{2}-2, \\ \overline{c}^0_{N-1,k}=\tilde{c}^1_{\frac{N-2}{2},k}+\tilde{d}^1_{\frac{N-2}{2},k}. \end{cases} \quad (4.3.5)$$

$$\begin{cases} \overline{c}^0_{0k}=c^0_{1k}-2\tilde{d}^1_{0k}, \\ \overline{c}^0_{2j,k}=\dfrac{1}{2}(\overline{c}^0_{2j-1,k}+\overline{c}^0_{2j+1k},)-2\tilde{d}^1_{jk},\ j=1,\ldots,\dfrac{M}{2}-1. \end{cases} \quad (4.3.6)$$

(3) $c^0_{j,k}=\tilde{c}^0_{j,k}/r,\ j=0,\ldots,M-1;\ k=0,\ldots,N-1.\ C^0=[c^0_{j,k}]^N_1 x^N_2.$

Following the inverse wavelet transformation, an inverse color transform is performed. Equations (5)-(6) give the inverse transforms for the YIQ and YUV color spaces.

For YIQ to RGB:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.000 & 0.956 & 0.621 \\ 1.000 & -0.272 & -0.647 \\ 1.000 & -1.106 & 1.703 \end{bmatrix} \begin{bmatrix} Y \\ I \\ Q \end{bmatrix} \quad (5)$$

For YUV to RGB:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.000 & 0.000 & 1.140 \\ 1.000 & -0.395 & -0.581 \\ 1.000 & 2.032 & 0.000 \end{bmatrix} \begin{bmatrix} Y \\ U \\ V \end{bmatrix} \quad (6)$$

In step 48, a user can optionally apply image filtering to improve the image quality. Filters are known in the art for sharpening, smoothing and brightening images. Users can choose any number of processing filters at compression time. Information defining the selected filters can be stored in the coded image file, in a form such as a one byte flag in a file header. In addition to optionally applying the filters,

the method can also be implemented to automatically detect and apply the selected filters following decompression.

To sharpen an image, a filter is used that weights the eight pixels adjacent to the current pixel, as well as the current pixel, by one or more predetermined values. The weighted values of the nine pixels are then summed to derive a new value for the current pixel. For example, the surrounding eight pixel values can be weighted by the value -35/800, while the current pixel is weighted by 1.35. The sharpening filter is applied to every pixel in the image.

To smooth images, for every pixel, the average of the pixel and the eight adjacent pixels is calculated. Then the pixel value and the average is compared. The smaller of the two replaces the original pixel and is output as the smoothed pixel value.

To brighten images, the weighted sum of each pixel and the correspond eight adjacent pixels is calculated. For example, each of the adjacent pixels can be multiplied by the value 1/90 and the summed with the current pixel to obtain a brighten current pixel.

Another filter that can be used is one that adds a random value between [-12, 12] to each of the pixels in the image.

In FIG. 6 there is displayed a preferred hardware platform that can execute software for implementing an embodiment of the present invention. The computer system of FIG. 3 includes a CPU 62, a main memory 64, an I/O subsystem 66, and a display 68, all coupled to a CPU bus 70. The I/O subsystem 66 communicates with peripheral devices that include an image source 72, an image storage device 74, and a mass storage memory 76. Although shown as three separate devices, peripherals 72-76 can be implemented using a single memory device, such as a hard disk drive commonly found in computers.

The image source 72 may be a digital still image or video source, such as a CD-ROM drive, scanner, or network connection. In addition, the image source 85 can include analog video sources, such as a video camera, VCR, television broadcast or cable receiver. The analog video signals would be converted to a digital form by the image source 85 using conventional conversion techniques. Alternatively, an image source 72 can include a video camera and communications systems for transmitting real-time video to the I/O subsystem 66.

The image storage 74 can be a computer disk, such as a that used by a hard drive, or a portable memory medium, such as a floppy or ZIP disk, or a read/write optical CD.

In operation, a computer program, which implements aspects of the invention, is retrieved from the mass storage memory 76 into the main memory 64 for execution by the CPU 62. Upon execution of the compression aspect of the invention, the compressed image file can be stored in the image storage 74; while upon execution of the decompression aspect of the invention, the decompressed image can be viewed on the display 68. Operating under the control of the computer program, the CPU 62 can process images according to the methods set forth herein, as shown in FIGS. 1-2 and 6-10.

FIG. 7 illustrates an alternative hardware platform implementing a system in accordance with a further embodiment of the present invention. System 80 can be implemented using a variety of different hardware components, such as ASIC (Application Specific Integrated Circuits), or a combination of discrete digital components, such as microprocessors, standard logic components, and other programmable logic devices. The system 80 includes a compression system 81 and a decompression system 82. The compression system 81 can be configured to perform any one or combination of the compression methods set forth in FIGS. 1, 8, 10, and 12;

while the decompression system can be configured to perform any one or combination of the decompression methods set forth in FIGS. 5, 9, 11, and 13.

An image source 85 provides digital pixel values to a color converter 84. The image source 85 can provide the same functionality as described earlier for the image source 72 of FIG. 6.

The color converter 84 performs a color space transformation on the input pixels, such as any of those described herein for FIG. 1. The converter functionality can be provided by conventional integrated circuits that are readily available from various manufacturers. Compressor 86 compresses the transformed pixels, removing redundant data. The compressed image file generated by the compressor 86 can be transferred directly to the decompression system 82 over a transmission medium 91. The transmission medium 91 can be a radio-link, computer network, cable television network, or satellite link. Alternatively, the compressor 86 can transmit its output to a portable storage medium 92, such as an optical, floppy, or ZIP disk; or to a mass storage device 94 such as a computer hard disk or archival system.

The decompressor 88 expands the compressed image file by applying an inverse wavelet transformation, as well as de-quantization and de-encoding functions. The decompressed data is then passed to an inverse color converter 90 that applies an inverse color space transformation to generate pixel values in a color space and format appropriate for the image display 89. Standard electronic components are readily available for performing the function of the inverse color converter 90.

FIG. 8 illustrates a flow diagram of a method of compressing an image in accordance with an alternative embodiment of the present invention. In step 100, a digital image is input. In step 102, a color space transformation is performed on the input image pixels.

In step 104, the pixels are subjected to a wavelet transformation. In step 106, sub-band quantization is performed on the wavelet coefficients. Next, in step 108 the quantized sub-bands are respectively entropy encoded. In step 110, the coded image file is output.

Sub-band oriented quantization and entropy coding are well suited for wavelet-based image compression. The main idea is to take the advantage of different quantizations at different sub-bands (wavelet quadrant) and encode each band accordingly. Quadrants having a high variance in wavelet values can be allocated a finer mesh size for quantization, while those quadrants with smaller variances will be assigned fewer levels of quantization. That is, the number of bits one wishes to allocate to the output could be varied by quadrant. Those quadrants with large variances will utilize more bits, while those with low variants will utilize fewer bits. In this way, the number of bits resulting from quantization will remain the same, but their allocation will differ depending upon the nature of the image. This technique greatly improves image quality while maintaining a high compression ratio.

FIG. 9 illustrates a flow diagram of a method of decompressing an image compressed according to the methods shown in FIG. 8. Step 120, the compressed file is input. In step 122, the input image is entropy decoded. In step 124, de-quantization is performed on the decoded image file. Next, in step 126, an inverse wavelet transform is performed on the image. In step 128, an inverse color transformation is performed. In step 130, post-processing altering is optionally performed. In step 132, the decompressed image file is then displayed.

FIG. 10 illustrates a flow diagram of a method of compressing an image in accordance with another embodiment of the present invention. This method performs color-bit depth compression, which essentially

reduces the number of colors in the image to achieve compression. In step 140, the image is input with its original color. For example, each color pixel could be represented by a standard 24-bit value. Next, in step 142, a color table is created corresponding to the image. The color table is a set of quantized color values. The quantized color values represent a smaller number of colors with correspondingly fewer bits. Each of the input pixels is mapped to the color table. In step 144, an index is calculated for each pixel in the image by dithering the pixel values. Dithering is accomplished by weighting pixels adjacent to the current pixel in a frame and then arithmetically combining the weighted values with the current pixel value to produce the index, which then represents the current pixel. The dithering process is repeated for each pixel in a frame. In step 146, the indexes are wavelet transformed. In step 148, the wavelet coefficients are entropy coded. In step 150, the coded image file is output.

FIG. 11 illustrates a flow diagram of a method of decompressing an image that has been compressed according to the method shown in FIG. 10. In step 160, a compressed image file is received. Next, in step 162, the image file is entropy decoded. In step 164, an inverse wavelet transform is applied to the decoded data. Next, in step 166, post-processing filtering of the image is optionally applied. Next, in step 168, the decompressed image is displayed.

FIG. 12 illustrates another method of compressing an image in accordance with another embodiment of the present invention. In this method, a user can selective vary compression parameters (step 173) to obtain a lossless or near-lossless compressed image at a desired compression ratio. In step 170, the image is input. In step 172, an integer color transform is performed on the input image. In step 173, compression parameters are selected by the user using a software

interface. These parameters can include those described herein below in the subsection title "Peak Signal to Noise Ratio (PSNR) Controlled Compression". In step 174, an integer wavelet transform is performed on the color transformed pixels. In step 176, the wavelet coefficients are entropy coded. Next, in step 178, the compressed image file is then output from the system.

The integer color transformation of step 172 is an integer reversible transform which can be used in color image compression to reduce processing time and image size. Step 172 transforms RGB color components to a set of color components Y-Nb-Nr, which are known.

The RGB to Y-Nb-Nr transform is given by the equations:

$$Y = G + \mathrm{Int}(R/2 + B/2),$$
$$Nb = B - \mathrm{Int}(Y/2),$$
$$Nr = R - \mathrm{Int}(Y/2).$$

The integer wavelet transform of step 174 is described below in detail.

FIG. 13 illustrates a method of decompressing an image file that has been compressed according to the method shown in FIG. 12. In step 180, a compressed image file is input. In step 182, the image is entropy decoded. Next, in step 184, an inverse integer wavelet transform is performed on the decoded data. In step 186, an inverse integer color transform is performed. Next, in step 188 optional post-processing filtering is performed on the image. Next, in step 190, the decompressed image is displayed.

The Y-Nb-Nr to RGB transform of step 186 is given by the equations:

$$R = Nr + \mathrm{Int}(Y/2),$$
$$B = Nb + \mathrm{Int}(Y/2),$$
$$G = Y - \mathrm{Int}\ (R/2 + B/2)$$

The inverse integer wavelet transform of step 184 is described in detail below.

Reversible Integer Wavelet Transform

This method allows a series of transformations which are very close to the corresponding biorthogonal wavelet transforms or some non-orthogonal wavelet transforms, but can be calculated with only integer addition and bit-shift operations. In addition, the integer wavelet transforms created disclosed herein possess a property of precision preservation (PPP). This property is very useful for conserving memory in both compression and decompression, and speed up the whole procedure in some applications. Two general methods from which one can get the integer wavelet transform desired are disclosed.

Basic Integer Wavelet Transformations

Two examples are provided as the starting point for the unique method. For the sake of convenience, length, and simplicity, presented is only the algorithm for a one level decomposition and reconstruction and only for a one dimensional signal. The extension to two dimensions is immediate as the rows and columns can be treated into a sequence of one dimensional signals. For the following examples, assume that $\{C_n^0\}_{n=0}^{N-1}$ is the original signal where the superscript indicates level and the subscript indicates a particular point in the signal. Also, $\{c_n^1\}_{n=0}^{N_1-1}$ and $\{d_n^1\}_{n=0}^{M_1-1}$ are its decomposition parts at the first level. Here

$$N_1 = \begin{cases} \frac{N}{2}, & \text{if } N \text{ is an even number,} \\ \frac{N+1}{2}, & \text{if } N \text{ is an odd number;} \end{cases}$$

$$M_1 = N - N_1$$

$\{c_n^1\}_{n=0}^{N_1-1}$ and $\{d_n^1\}_{n=0}^{M_1-1}$ are its low frequency (l) part and high frequency (h) part, respectively. For multi-levels, we just create $\{c_n^1\}_{n=0}^{N_1-1}$ as $\{c_n^0\}_{n=0}^{N_1}$ and repeat the procedure again.

Example 1: A (2,2)-wavelet transform by integer calculation.

This transformation is similar to a variation of the Haar wavelet transform which uses low and high pass analysis (decomposition) filters given as:

n	0	1
$\tilde{h}_n$	1/2	1/2
$\tilde{g}_n$	1/2	-1/2

(1) Compute

$$d_k^1 = c_{2k}^0 - c_{2k+1}^0, \quad k = 0, \ldots, M_1 - 1. \qquad (2.1)$$

(2) Compute

$$c_k^1 = Int\left(\frac{d_k^1}{2}\right) + c_{2k+1}^0, \quad k=0,\ldots,N_1-2,$$

$$c_{N_1-1}^1 = \begin{cases} Int\left(\frac{d_{M_1-1}^1}{2}\right) + C_{2k+1}^0, & \text{if } N \text{ is an even number,} \\ c_{N-1}^0, & \text{if } N \text{ is an odd number.} \end{cases} \qquad (2.2)$$

Here, Int(x) is an arbitrary rounding function which may have different interpretations. For example, Int(x) can be the integer which is nearest to x, or Int(x) may be any integer which satisfies $x-1<Int(x)\leq x$, etc. It is easy to see that all entries in both

$\{c_n^1\}_{n=0}^{N_1-1}$ and $\{d_n^1\}_{n=0}^{M_1-1}$ are integers.

From (2.1)-(2.2), we can easily get the following integer reconstruction algorithm:

(b) Reconstruction

(1) If N is an even number, compute:

$$c_{2k+1}^0 = c_k^1 - Int\left(\frac{d_k^1}{2}\right), \quad k=0,\ldots,N_1-1; \qquad (2.3)$$

or, if N is an odd number, we have

$$c_{2k+1}^0 = c_k^1 - Int\left(\frac{d_k^1}{2}\right). \quad k=0,\ldots,N_1-2; \qquad (2.4)$$

$$c_{N-1}^0 = c_{N_1}^1.$$

(2) Compute

$$c_{2k}^0 = d_k^1 + c_{2k+1}^0 \quad k=0,\ldots,M_1-1. \qquad (2.5)$$

Remark. Since (2.1)-(2.6) are not linear because of the rounding operation Int(x), this means the transformation order becomes significant. For instance, if the decomposition was applied first to the columns and then to the rows, the inverse transformation must be applied first to the rows and then to the columns.

Example 2: Lazy wavelet transform.
The lazy wavelet transform is used to illustrate an important concept. The corresponding inverse transform is nothing else but sub-sampling the even and odd indexed samples. Decomposition and reconstruction can use the same formula as follows:

$$c_k^1 = c_{2k}^0, \quad k=0,\ldots,N_1-1;$$
$$d_k^1 = c_{2k+1}^0, \quad k=0,\ldots,M_1-1.$$

Examples 1 and 2 are not good transforms for image compression, but they are simple. Much better transforms can be achieved from these two. As suggested above, they are considered only as a starting point for the integer, reversible, wavelet transform algorithm of the disclosed invention.

It is noted that there is another interesting property in the above two transforms which may not be easily seen. If the values of the signal pixels are represented by a finite number of bits, say one bit or one byte, the same number of bits can be used to represent the result of the forward transform within the computer itself because of the complementary code property. While, from the reconstruction algorithm, the computer will get back the exact original signal through the same complementary code property. This property is called a *Property of Precision Preservation (PPP)* for these wavelets.

It is known that the general values for the high frequency wavelet coefficients are small, and all higher levels of the decomposition also provide generally small

values in the high frequency band. This allows the preservation of precision during the computational stage of the wavelet coefficients. Now, the complementary code property, the other aspect of the *PPP* property is a well known characteristic of integer arithmetic as done by the computer. Consider the computation of the difference of two integers given as $c = b - a$ and the inverse computation of $a = b - c$. The nature of the computation within the computer can be specified as follows:

$$c_m = \begin{cases} b-a \ \text{if} \ -2^{q-1} \le b-a < 2^{q-1}-1 \\ -2^q+b-a \ \text{if} \ b-a \ge 2^{q-1} \\ 2^q+b-a \ \text{if} \ b-a < -2^{q-1} \end{cases}$$

and the inverse is

$$a_m = \begin{cases} b-c_m \ \text{if} \ -2^{q-1} \le b-a < 2^{q-1}-1 \\ -2^q+b-c_m \ \text{if} \ b-c_m \ge 2^{q-1} \\ 2^q+b-c_m \ \text{if} \ b-c_m < -2^{q-1} \end{cases}$$

where the m subscript indicates the internal representation, and the range of the integers a, b, c is $[-2^{q-1}, 2^{q-1}-1]$. The internal representation of c_m when it is outside the range, its appearance is as a two's complement number, so the representation may not be the same as the external representation of c. However, the same complementary code for the a_m will cause the internal representation to be identical to the external representation of a. For example, if we let b = 2 (00000010) and a = -127 (10000001) then c_m has the internal binary value of (10000001) when q=4. With a value of -127 for c_m the inverse value for a_m will just be *a*.

In fact, for Example 2, this property is obviously true. While for Example 1, if the range of the pixel values is within a finite number of bits, say q, we can only use q bits as the working unit, which means the

value of transform coefficients will also be within the interval with length 2^q, say $[-2^{q-1}, 2^{q-1} - 1]$. Due to the nature of computation on a machine, most machines will implement (2.1)-(2.2) automatically as follows (the complementary code property):

$$d_k^1=\begin{cases} c_{2k}^0-c_{2k+1}^0, & if\ -2^{q-1}\le c_{2k}^0-c_{2k+1}^0<2^{q-1}, \\ c_{2k}^0-c_{2k+1}^0-2^q, & if\ c_{2k}^0-c_{2k+1}^0\ge 2^{q-1}, \\ 2^q+(c_{2k}^0-c_{2k+1}^0), & if\ c_{2k}^0-c_{2k+1}^0<-2^{q-1}. \end{cases} \tag{2.6}$$

$$c_k^1=\begin{cases} Int(\frac{d_k^1}{2})+c_{2k+1}^0, & if\ -2^{q-1}\le Int(\frac{d_k^1}{2})+c_{2k+1}^0<2^{q-1}, \\ Int(\frac{d_k^1}{2})+c_{2k+1}^0-2^q, & if\ Int(\frac{d_k^1}{2})+c_{2k+1}^0\ge 2^{q-1}, \\ Int(\frac{d_k^1}{2})+c_{2k+1}^0+2^q, & if\ Int(\frac{d_k^1}{2})+c_{2k+1}^0<-2^{q-1}. \end{cases} \tag{2.7}$$

While the reconstruction algorithm (2.3) and (2.5) will be implemented by the computer itself as

$$c_{2k+1}^0=\begin{cases} c_k^1-Int(\frac{d_k^1}{2}), & if\ -2^{q-1}\le c_k^1-Int(\frac{d_k^1}{2})<2^{q-1}, \\ 2^q+\left(c_k^1-Int(\frac{d_k^1}{2})\right), & if\ c_k^1-Int(\frac{d_k^1}{2})<-2^{q-1}, \\ \left(c_k^1-Int(\frac{d_k^1}{2})\right)-2^q, & if\ c_k^1-Int(\frac{d_k^1}{2})>-2^{q-1}. \end{cases} \tag{2.8}$$

$$c_{2k}^0=\begin{cases} d_k^1+c_{2k+1}^0, & if\ -2^{q-1}\le d_k^1+c_{2k+1}^0<2^{q-1}, \\ d_k^1+c_{2k+1}^0+2^q, & if\ d_k^1+c_{2k+1}^0<-2^{q-1}, \\ d_k^1+c_{2k+1}^0-2^q, & if\ d_k^1+c_{2k+1}^0\ge -2^{q-1}. \end{cases} \tag{2.9}$$

It is obvious that (2.8)-(2.9) are just the reverse of (2.6)-(2.7). It is also easy to see that if we properly take advantage of the bound in the coefficient size mentioned above, the algorithm can be implemented using a minimal amount of storage.

The following are examples which give motivation for our new approach.

Example 3: A (2.6) wavelet transform by integer calculation (2).

This transformation is similar to using the following analysis filters:

n	-2	-1	0	1	2	3
$\tilde{h}_n$	0	0	1/2	1/2	0	0
$\tilde{g}_n$	-1/16	-1/16	1/2	-1/2	1/16	1/16

(a) Decomposition

Decomposition starts with Example 1 at step (1) and (2), and then upgrades the high frequency component at step (3):

(1) Compute

$$d_k^{1,0}=c_{2k}^0-c_{2k+1}^0, k=0,\ldots,M_1-1.$$

(2) Compute

$$c_k^1=Int(\frac{d_k^{1,0}}{2})+c_{2k+1}^0,\ k=0,\ldots,N_1-2,$$

$$c_{N_1-1}^1=\begin{cases}Int(\frac{d_{M_1-1}^{1,0}}{2}),+c_{N-1}^0, & \text{if } N \text{ is an even number},\\ c_{N-1}^0, & \text{if } N \text{ is an odd number};\end{cases}$$

(3) Compute

$$\begin{cases}d_0^1=Int(\frac{c_0-c_1^1}{4}),+d_0^{1,0}\\ d_k^1,=Int(\frac{c_{k-1}^1-c_{k+1}^1}{4})-d_k^{1,0}, k=1,\ldots,M_1-2,\end{cases}$$

and then, if N is even, calculate

$$d_{M_1-1}^{1}=Int(\frac{C_{N_1-2}^{1}-C_{N_1-1}^{1}}{4})-d_{M_1-1}^{1,0},$$

else, calculate

$$d_{M_1-1}^{1}=Int(\frac{C_{N_1-3}^{1}-C_{N_1-1}^{1}}{4})-d_{M_1-1}^{1,0}.$$

(b) Reconstruction

The reconstruction algorithm is identical to the decomposition algorithm, except it is now running "backwards".

(1) Compute

$$\begin{cases} d_0^{1,0}=Int(\frac{C_0^1-C_1^1}{4})-d_0^1 \\ d_k^{1,0}=Int(\frac{C_{k-1}^1-C_{k+1}^1}{4})-d_k^1, k=1,\ldots,M_1-2, \end{cases}$$

and then, if N is even, calculate

$$d_{M_1-1}^{1,0}=Int(\frac{C_{N_1-2}^{1}-C_{N_1-1}^{1}}{4})-d_{M_1-1}^{1,1},$$

else calculate

$$d_{M_1-1}^{1,0}=Int(\frac{C_{N_1-3}^{1}-C_{N_1-1}^{1}}{4})-d_{M_1-1}^{1,1},$$

(2) If N is an even number, compute

$$c_{2k+1}^{0}=c_k^1-Int(\frac{d_k^{1,0}}{2}),\ k=0,\ldots,N_1-1;$$

or, if N is an odd number, we have

$$c_{2k+1}^{0}=c_k^1-Int(\frac{d_k^{1,0}}{2}),\ k=0,\ldots,N_1-2;$$

$$c_{N-1}^{0}=c_{N_1}^{1}.$$

(3) Compute

$$c_{2k}^0=d_k^{1,0}+c_{2k+1}^0,\ k=0,\ldots,M_1-1.$$

We see in step (2)-(3) above, that they are just the same as shown for the reconstruction of the (2.2)-wavelet transform (Example 1).

Example 4: A (1,3)-wavelet transform by integer calculation.

The following nonlinear transform is a variation of the transform which uses biorthogonal analysis filters:

n	-1	0	1
$\tilde{h}_n$	1	0	0
$\tilde{g}_n$	1/4	-1/2	1/4

(a) Decomposition

This decomposition starts with the *Lazy wavelet* at step (1) and upgrades the high frequency component at step (2):

(1) Set

$$c_k^1=c_{2k}^0,\ k=0,\ldots,N_1-1;$$

$$d_k^1=c_{2k+1}^0,\ k=0,\ldots,M_1-1.$$

(2) If N is an even number, calculate

$$\begin{cases} d_k^1=Int\left(\frac{c_k^1-c_{k+1}^1}{2}\right)-d_k^{1,0},\ k=0,\ldots,M_1-2, \\ d_{M_1-1}^1,=c_{N_1-1}^1-d_{M_1-1}^{1,0}. \end{cases}$$

Otherwise, if N is an odd number, calculate

$$d_k^1 = Int\left(\frac{c_{2k}^0 + c_{2k+2}^0}{2}\right) - c_{2k+1}, k=0,\ldots,M_1-1.$$

(b) Reconstruction

(1) Set

$$c_{2k}^0 = c_k^1,\ k=0,\ldots,N_1-1;$$

(2) If N is an even number, calculate

$$\begin{cases} c_{2k+1}^0 = Int\left(\frac{c_{2k}^0 + c_{2k+2}^0}{2}\right) - d_k^1, k=0,\ldots,M_1-2, \\ c_{N-1}^0, = c_{N-2}^0 - d_{M_1-1}^1. \end{cases}$$

Otherwise, if N is an odd number, calculate

$$c_{2k+1}^0 = Int\left(\frac{c_{2k}^0 + c_{2k+2}^0}{2}\right) - d_k^1,\ k=0,\ldots,M_1-1.$$

Example 5: (5,3)-wavelet transform by integer calculation.

This transformation is also similar in function to using the biorthogonal analysis filters. It is given by

n	-2	-1	0	1	2
$\tilde{h}_n$	-1/8	1/4	3/4	1/4	-1/8
$\tilde{g}_n$	1/4	-1/2	1/4	0	0

(a) Decomposition

This decomposition starts with Example 3 at step (1) and upgrade low frequency components at step (2):

(1) Set

$$c_k^{1,0}=c_{2k}^0,\ k=0,\ldots,N_1-1;$$

If N is an even number, calculate

$$\begin{cases} d_k^1=Int(\frac{c_{2k}^0+c_{2k+2}^0}{2})-c_{2k+1}^0,\ k=0,\ldots,M_1-2, \\ d_{M_1-1}^1=c_{N-2}^0-c_{N-1}^1. \end{cases}$$

Otherwise, if N is an odd number, calculate

$$d_k^1=Int(\frac{c_{2k}^0+c_{2k+2}^0}{2})-c_{2k+1},\ k=0,\ldots,M_1-1.$$

(2) If N is an even number, compute

$$\begin{cases} c_0^1=c_0^{1,0}-Int(\frac{d_0^1}{2}), \\ c_k^1=c_k^{1,0}-Int(\frac{d_{k-1}^1+d_k^1}{4}),\ k=1,\ldots,N_1-2, \\ c_{N-1}^1=c_{N_1-2}^{1,0}-Int(\frac{d_{N_1-2}^1+d_{N_1-1}^1}{4}). \end{cases}$$

Otherwise, if N is an odd number, calculate

$$\begin{cases} c_0^1=c_0^{1,0}-Int(\frac{d_0^1}{2}), \\ c_k^1=c_k^{1,0}-\frac{d_{k-1}^1+d_k^1}{4},\ k=1,\ldots,N_1-2, \\ c_{N_1-1}^1=c_{N_1-1}^{1,0}-Int(\frac{d_{M_1-1}^1}{2}). \end{cases}$$

(b) Reconstruction

(1) Compute

$$c_0^0 = c_0^1 + Int(\frac{d_0^1}{2}),$$

$$c_{2K}^0 = c_K^1 + Int(\frac{d_{K-1}^1 + d_K^1}{4}),\ K=1,\ldots,N_1-2,$$

Then, if N is even, calculate

$$c_{N-2}^0 = c_{N_1-1}^1 + Int(\frac{d_{N_1-2}^1 + d_{N_1-1}^1}{4}).$$

else calculate

$$c_{N-1}^0 = c_{N_1-1}^1 + Int(\frac{d_{M_1-1}^1}{2}).$$

(2) Compute

$$c_{2k+1}^0 = Int(\frac{c_{2k}^0 + c_{2k+2}^0}{2}) - d_k^1,\ k=0,\ldots,M_1-2,$$

Then, if N is even, calculate

$$c_{N-1}^0 = c_{N-2}^0 - d_{M_1-1}^1.$$

The *PPP* property for Examples 1-2 mentioned at the end of the previous section is also applicable for these three examples. It is obvious these three transformations are not really linear, but they are similar to the one using the corresponding filters given above. Especially, the filters in Example 3 and Example 5 belong to, with minor modification, the group of the best biorthogonal filters for image compression.

Also, from the above three examples, we can note that if we begin with integer (linear or nonlinear) wavelet transformations and then use some proper upgrading formulas, we can get other, much better integer, wavelet transformations for image compression.

Lifting Scheme and Integer Biorthogonal Filtering

The *Lifting scheme*, discussed by W. Sweldens in "The Lifting Scheme: A Custom-Designed Construction of Biorthogonal Wavelet", Applied and Computational Harmonic Analysis, Vol. 3, No. 2, April 1996, is a recently developed approach for constructing biorthogonal wavelets with compact support. It can be used, with minor modifications, to create integer biorthogonal wavelet transformations. The following is an adaptation of the lifting scheme.

Definition 1. The set of filters {h, h, g, g}, a set of biorthogonal filters if the following formula is satisfied:

$$\forall \omega \in \mathbf{R} : \tilde{m}(\omega)\overline{M'(\omega)} = 1.$$

where $$m(\omega) = \begin{bmatrix} h(\omega) & h(\omega+\pi) \\ g(\omega) & g(\omega+\pi) \end{bmatrix},$$

and $$h(\omega) = \sum_k h_k e^{-k\omega} \text{ and } g(\omega) = \sum_k g_k e^{-k\omega},$$

and *similarly for* $\tilde{m}(\omega)$, $\tilde{h}(\omega)$ *and* $\tilde{g}(\omega)$.

The following lemma is the main result of the lifting scheme [1] reported as corollary 6 in that paper.

Lemma 1. Take an initial set of finite biorthogonal filters {h, $\tilde{h}^{\circ}$, g, $\tilde{g}^{\circ}$}, then a new set of finite biorthogonal filters {h, $\tilde{h}$, g, $\tilde{g}$} can be found as

$$\tilde{h}(\omega) = \tilde{h}^0(\omega) + \tilde{g}(\omega)\overline{s(2\omega)}$$
$$g(\omega) = g^0(\omega) - h(\omega)s(2\omega).$$

Similarly if we take {h°, $\tilde{h}$, g, $\tilde{g}^{\circ}$} as an initial set of biorthogonal filters, a new set of biorthogonal filters {h, $\tilde{h}$, g, $\tilde{g}$} can be found as can be found as

$$h(\omega)=h^0(\omega)+g(\omega)\overline{s(2\omega)}$$
$$\tilde{g}(\omega)=\tilde{g}^0(\omega)-\tilde{h}(\omega)\tilde{s}(2\omega).$$

Here $s(\omega)$ is a trigonometric polynomial and the corresponding filter s is finite, and so is $\tilde{s}(\omega)$. Actually, regarding the filters (4.1) is equivalent to

$$\tilde{h}_k=\tilde{h}_k^0+\sum_l \tilde{g}_{k+2l}\tilde{s}_l$$
$$g_k=g_k^0-\sum_l h_{k-2l}\tilde{s}_l$$

or

$$h_k=h_k^0+\sum_l g_{k+2l}\tilde{s}_l$$
$$\tilde{g}_k=\tilde{g}_k^0-\sum_l \tilde{h}_{k-2l}\tilde{s}_l \qquad (4.26)$$

Next we use the lifting scheme with minor modifications to create an integer, nonlinear, quasi-biorthogonal, wavelet algorithm. Suppose $[c_n^0]$ is a original signal, $[c_n^1]$ and $[d_n^1]$ are again its low and high frequency decomposition parts, obtained by using the filters $\{h, \tilde{h}, g, \tilde{g}\}$.

If we use filters $\{\tilde{h}, \tilde{g}\}$ for decomposition (analysis), the corresponding decomposition algorithm is

$$\begin{cases} c_k^1=\alpha_c\sum_n c_n^0\tilde{h}_{n-2k}, \\ d_k^1=\alpha_d\sum_n c_n^0\tilde{g}_{n-2k}. \end{cases}$$

While the reconstruction algorithm will be

$$c_n^0=2\sum_k\left(\frac{c_k^1 h_{n-2k}}{\alpha_c}+\frac{d_k^1 g_{n-2k}}{\alpha_d}\right),$$

related to the synthesis filter {h,g}. Here, parameters α_c and α_d are positive constants with $\alpha_c \cdot \alpha_d=2$. For example, in the situation of regular biorthogonal decomposition and reconstruction, $\alpha_c=\alpha_d=\sqrt{2}$; and for Example 1 through Example 5 above, $\alpha_c=1$ and $\alpha_d=2$.

If the set of filters {h, $\tilde{h}$, g, $\tilde{g}$} is from {h, $\tilde{h}^0$, g^0, $\tilde{g}$} by (4.2b), then decomposition can be accomplished as follows:

1. Calculate

$$\begin{cases} c_k^{1,0}=\alpha_c\sum_n c_n^0 \tilde{h}_{n-2k}^0, \\ d_k^1=\alpha_d\sum_n c_n^0 \tilde{g}_{n-2k}. \end{cases}$$

2. Calculate

$$c_k^1=c_k^{1,0}+\frac{\alpha_c}{\alpha_d}\sum_i d_{k-1}^1 s_i. \qquad (4.4)$$

The relative reconstruction scheme will be:

1. Calculate

$$c_k^{1,0}=c_k^1\frac{\alpha_c}{\alpha_d}\sum_i d_{k-1}^1 s_i \qquad (4.5)$$

2. Calculate

$$c_n^0=2\sum_k\left(\frac{c_k^{1,0} h_{n-2k}}{\alpha_c}+\frac{d_k^1 g_{n-2k}^0}{\alpha_d}\right). \qquad (4.6)$$

Here, equations (4.3) and (4.6) are just the wavelet (inverse) transforms using biorthogonal filters {h, $\tilde{h}^0$, g^0,

$\tilde{g}$). While (4.4) and (4.5) are forward and backward upgrading formulas.

Similarly if the set of filters {h, $\tilde{h}$, g, $\tilde{g}$} is from the initial set of filters {h, $\tilde{h}$, g, $\tilde{g}$} by using (4.2b), the relative decomposition is:

1. Calculate

$$\begin{cases} c_k^1=\alpha_c\sum_n c_n^0 h_{n-2k} \\ d_k^{1,0}=\alpha_d\sum_n c_n^0 \overline{g}_{n-2k}^0. \end{cases}$$

2. Calculate

$$d_k^1=d_k^{1,0}\frac{\alpha_c}{\alpha_d}\sum_1 c_{k-1^{s_1}}^1$$

The reconstruction scheme is:

1. Calculate

1. Calculate

$$d_k^{1,0}=d_k^1\frac{\alpha_c}{\alpha_d}\sum_1 c_{k-1^{s_1}}^1$$

2. Calculate

$$c_n^0=2\sum_k\left(\frac{c_k^{1,0}h_{n-2k}}{\alpha_c}+\frac{d_k^1 g_{n-2k}^0}{\alpha_d}\right).$$

Corollary 4.1. *Suppose biorthoganal filters* $\{h,\tilde{h},g,\tilde{g}\}$ *are from initial filters* $\{h,\tilde{h}^0,g^0,\tilde{g}\}$ *by the lifting scheme* (4.1a) *or* (4.2a). *If the decomposition and reconstruction by filters* $\{h,\tilde{h}^0,g^0,\tilde{g}\}$ *can be accomplished only by integer calculation, such as Example* 2, *we also can create a corresponding integer wavelet decomposition and reconstruction scheme which is very "close" to the original one by using filters* $\{h,\tilde{h},g,\tilde{g},\}$. *Here the word "close" means that the difference of the two decompostion schemes is just some rounding error, and this rounding error will l corrected by the integer reconstruction scheme.*

In fact, if $\{c_k^{1,0}\}$ *and* $\{d_k^1\}$ *are integer after* (4.3), *we can calculate* $\{c_k^1\}$ *by*

$$c_k^1=c_k^{1,0}+Int\left(\frac{\alpha_c}{\alpha_d}\sum d_{k-l}s_l\right)$$

instead of (4.4). Here Int(x), as described in Section 2, is an arbitrary rounding up function which satisfies $x-1 \le Int(x) \le x+1$. It is obvious that (4.7) is very close to (4.4), and the exact reconstruction scheme can easily be obtained from

$$c_k^{1,0}=c_k^1-Int\left(\frac{\alpha_c}{\alpha_d}\sum_l d_{k-l}s_l\right)$$

and (4.6). There will be a similar result, if the set of biorthogonal filters $\{h, \tilde{h}, g, \tilde{g}\}$ is obtained from the initial set of filters $\{h^*, \tilde{h}, g, \tilde{g}^*\}$ by using (4.2b).

Except for the example shown in the *Lazy wavelet* (Example 2), most standard biorthogonal wavelet forms cannot be performed directly by integer, even for one of the simplest wavelets, the *Harr wavelet*. However, if the parameters α_c, and α_d are properly chosen and the transform algorithms, such as Example 1 and Example 3, are slightly changed, a variation of the original

biorthogonal wavelet transforms with respect to the set of filters {h, h, g, g} is created. On the other hand, the parameters should be also chosen carefully to guarantee that only addition and shift operations are needed by the algorithm.

If the set of filters $\{h, \tilde{h}, g, \tilde{g}\}$ is obtained from a set of filters $\{h^{\circ}, \tilde{h}, g, \tilde{g}^{\circ}\}$ by the lifting scheme, and the set $\{h^{\bullet}, \tilde{h}, g, \tilde{g}^{\circ}\}$ is also obtained from a filter set $\{h^{\circ}, \tilde{h}^{\circ}, g^{\circ}, \tilde{g}^{\circ}\}$, one can repeatedly use Corollary 1 to get a "close" integer wavelet transformation.

The Correction Method for Creating Integer Wavelet Transforms

Another approach for obtaining integer wavelets is using the so-called *Correction method*. The motivation of this method is from the S+P transform. The lifting scheme for generating biorthogonal wavelets can be considered as a special case of the correction method. From this can be derived complicated filters with fast decomposition and reconstruction algorithms.

Assuming a simple integer wavelet transform, such as Examples 1 through 3, the decomposition and reconstruction scheme of which can be formulated as follows:

Decomposition

$$c_1^{1,0}=df_c\left(\{c_n^0\}\right)$$
$$d_1^{1,0}=df_d\left(\{c_n^0\}\right) \qquad (5.1)$$

Reconstruction

$$c_n^0=rf\left(\{c_1^{1,0}\},\{d_k^{1,0}\}\right) \qquad (5.2)$$

Here, (5.1) and (5.2) can be the same as (4.3) and (4.6) or other algorithms.

In general, after the above decomposition, one may not be satisfied with the result. There may still be some correlation among the high pass components because

of the aliasing from the low pass components, or the low pass components do not carry enough of the expected information from the original signal. Hence, one could make an improvement by putting some correction part on the high pass components or low pass components. There are many ways to accomplish this. However, for the sake of the integer calculation, it is preferable to use following correction method. To make a correction for the high pass part, the corresponding formula would be:

$$d_2^1=d_k^{1,0}-Int(dc_{2,k}^1)k=\cdots,0,1,2\cdots \qquad (5.3)$$

Here, dc_k^1 is a correction quantity for d_k^1

$$d_k^1=\sum_{j=1}^{S_1}\sigma_j c_{k+1}^{1,0}+\sum_{j=1}^{1}\tau_j d_{k+j}^{1,0},k=\cdots,0,1,2,\cdots \qquad (5.4)$$

and $\{\sigma_i\}_{j=s_0}^{s_1}$ and $\{\zeta_j\}_{j=1}^{T}$ are given parameters which have been chosen for the user's purpose such as reducing the redundancy among high pass components or some other special requirement. To preserve the integer calculation, any entries in both $\{\sigma_i\}_{i=s_0}^{s_1}$ and $\{\zeta_j\}_{j=1}^{T}$ should be rational numbers with denominators being powers of 2.

From (5.1), (5.3) and (5.4), it is easy to see the perfect reconstruction algorithm can be

$$d_k^{1,0}=d_k^1+Int(dc_k),k=\cdots,m,m-1,m-2\cdots, \qquad (5.5)$$

combined with (5.2).

As mentioned above, the Lifting scheme is a special condition of the correction method. Examples 3 through 5 can also be considered as the examples of this method. We next give an example of the Correction method which cannot be included in the group of Lifting scheme, and also which does not result in a closed form of compact support for biorthogonal filters.

Example 6: S+P transform, which is similar to using following analysis filters.

n	-2	-1	0	1	2	3
$\tilde{h}_n$	0	0	1/2	1/2	0	0
$\hat{g}_n$	-1/16	-1/16	15/32	-17/32	7/32	-1/32

While the synthesis filters do not have compact support, the S+P transform can be implemented as follows:

(a) Decomposition

(1) Take the decomposition step of Example 1, that is, compute

$$d_k^{1,0}=c_{2k}^0-c_{2k+1}^0, k=-0,1,\cdots,M_1-1;$$

and

$$c_k^1=Int\left(\frac{d_k^{1,0}}{2}\right)+c_{2k+1}^0, k=0,\cdots,N_1-2,$$

$$c_{n_1-1}^1=\begin{cases} Int\left(\frac{d_{M_1-1}^{1,0}}{2}\right)+c_{N-1}^0, \\ c_{n-1}^0 \end{cases}$$

(2) Correction Step: Define $S_0 = -1$, $S_1 = 1$, $T = 1$ and

$$\sigma_{-1}=-\frac{1}{4}, \sigma_0=-\frac{1}{6}, \sigma_1=\frac{1}{6};$$

$$\tau_1=\frac{1}{4}.$$

and now compute

$$d_0^1 = d_0^{1,0} - Int\left(\frac{c_0^1 - c_1^1}{4}\right);$$

$$d_k^1 = d_k^{1,0} - Int\left(\frac{2c_{k-1}^1 + c_k^1 - 3c_{k+1}^1 - 2d_{k+1}^{1,0}}{8}\right), k=1,\cdots,M_1-2;$$

$$d_{M_1-1}^1 = d_{M_1-1}^{1,0} - Int\left(\frac{C_{M_1-2}^1 - C_{M_1-1}^1}{4}\right).$$

(b) Reconstruction

(1) Compute

$$d_{M_1-1}^{1,0} = d_{M_1-1}^1 + Int\left(\frac{C_{M_1-2}^1 - C_{M_1-1}^1}{4}\right)$$

$$d_k^{1,0} = d_k^1 + Int\left(\frac{2c_{k-1}^1 + c_k^1 - c_{k+1}^1 - 2d_{k+1}^{1,0}}{8}\right), k=M_1-2,\cdots,1;$$

$$d_0^{1,0} = d_0^1 + Int\left(\frac{c_0^1 - c_1^1}{4}\right).$$

(2) If N is an even number, compute

$$c_{2k+1}^0 = c_k^1 - Int\left(\frac{d_k^1}{2}\right), k=0,\cdots,N_1-1$$

or, if N is an odd number, we have

$$c_{2k+1}^0 = c_k^1 - Int\left(\frac{d_k^1}{2}\right), k=0,\cdots,N_1-2,$$

$$c_{N-1}^0 = c_{N_1}^1.$$

(3) Compute

$$c_{2k}^0 = d_k^1 + c_{2k+1}^0, k=0,\cdots,M_1-1.$$

Boundary Conditions

There are two issues dealing with boundary filtering if the Lifting scheme or the Correction method

is used to generate the integer wavelet transformations. The first is how to process the boundaries which occur in the start-up wavelet transformations. The second is how to deal with the boundaries in the deductive formula. If the boundaries in the start-up wavelet transform have already been established, then those in the upgrading formula are relatively easy to establish. For the *Lifting scheme*, the boundaries in both steps should be processed in the same way. While, for the Correction method, according to (5.3)-(5.4), one has more choices to process boundaries in the second step. Therefore, the process by which the boundaries in the start-up wavelet transformations are established is discussed. Assume compact supported biorthogonal wavelets.

Suppose the original signal is $\{c_n^0\}_{n=0}^{N}$. For creating integer biorthogonal wavelet transformations, use the following symmetric extension:

(1) If current biorthogonal filters have even length, the boundaries of the signal are extended as $c_{-k}^0 = c_{k-1}^0$, $k = 1, 2, \cdots$;

(2) If the filters have odd length, the following extension is performed $c_{-k}^0 = c_k^0$, $k = 1, 2, \cdots$.

Examples 1 through 5 use the boundaries given above. In Example 6, the start up wavelet transform uses the above boundaries but in the upgrading step, another boundary filtering is used. In addition, for arbitrarily sized images or signals, one can use the same technique described in the above examples to deal with this condition.

As mentioned earlier, for many applications, lossless image compression is more important than lossy compression. The integer wavelet transforms described above provide the opportunity to compress without loss. It is also obvious that the integer wavelet algorithms

can be used wherever ordinary wavelets are used, especially in signal and image compression. However, for most computers, the integer wavelet transform is much faster than other wavelets and it uses much less memory.

Peak Signal to Noise Ratio (PSNR) Controlled Compression

Peak Signal to Noise Ratio (PSNR) is a widely used quality measurement. PSNR controlled compression allows users to choose their desired PSNR for the compressed image. In each of the compression methods set forth herein, a user can selectively set the PSNR and the desired compression ratio, as well as the initial quantization and threshold levels for each quadrant of wavelet coefficients, to obtain the desired image quality.

For example, the wavelet map of FIG. 3 shows a total of 10 regions (quadrants). Each of these ten quadrants can have two additional parameters associated with them. The parameters define the quantization and threshold values for that particular quadrant. Since there are three planes for color (only one for gray level) the maximum number of parameters that the user can control is 60 -- 10 for quantization and 10 for thresholding for each of the three color layers. In the case of a gray level image, there are only 20 parameters.

If a compression ratio, or a quality factor which indirectly defines a compression ratio, is specified, then the user wants the compression ratio to remain identical over the changes in the parameters. In order to accomplish this, two parameters are monitored: the compression ratio and PSNR (peak signal to noise ratio). The PSNR is defined as $PSNR = 20 \log_{10} (X/MSE)$, where the X is the average absolute value of the pixels in the compressed image and MSE is the mean squared error measured between the compressed and original image. Holding the compression ratio constant, the PSNR needs to

increase to improve image quality. The way to increase the PSNR is to reduce the MSE.

An iterative method can be used to adjust parameters to achieve the desired PSNR. The step are as follows:

(a) Pick an initial parameter setting P_0;
(b) Quantize the wavelet coefficients with P_0 and calculate the corresponding PSNR;
(c) If the PSNR is close to the desired one, stop and output the coded file; otherwise, get an adjusted vector ΔP_0 and set $P_0 \leftarrow P_0 + \Delta P_0$, go to step (b).

Progressive Decomposition

Progressive decompression allows users to decode images at varying degrees of resolution, starting from the lowest resolution and progressing to the highest resolution. The advantage of this feature is that users can download small pieces of the coded file and view the image at lower resolution to determine if they want to download the whole image. Progressive decomposition can be used with any of the decompression methods previously disclosed herein. Progressive decomposition is accomplished according to the following steps:

(a) Input the lowest bandpass component C^1 of the coded file and reconstruct the lowest resolution image I^0;
(b) Display image I^0;
(c) If the user is not satisfied with the image quality or the resolution is big enough for stop; otherwise, go to step (d);
(d) Input the lowest three band-pass components HD^1, VD^1, and DD^1 successively in the current image file. Reconstruct the new image I^1 from C^1, HD^1, VD^1, and DD^1. Let $I^0 = I^1$; go to step (b).

Image Map Editor

The image map editor creates an image map over a compressed image file. This permits an image compressed according to one of the methods set forth herein to be easily integrated into a web page using an http link. A user selects one or several areas of compressed image, assigns one or more http links to the areas. The image map editor calculates the coordinates of the areas and outputs the HTML associate with the image. The user can add such information into program source code. Following is an example of such image map:

```
<EMBED SRC="cow.cod" type="image/cis-cod"
WIDTH="257" poly= "44, 45, 103, 78, 103, 86, 54,
86, 54, 78",
href="http://www.infinop.com"></EMBED>
```

Non-Uniform Image Compression

The present invention allows a user to perform non-uniform image compression. Essentially, non-uniform compression is accomplished by dividing an image into one or more rectangles, each representing a matrix of image pixels. Each rectangle can be compressed by any of the methods disclosed herein.

For instance, referring to the compression method of FIG. 8, integrating the non-uniform compression feature with the method allows a user to partition the image into several parts with different interests. The user can then compress these areas with different image and/or compression qualities. The parts can have any shape.

The non-uniform compression feature can be incorporated in to the method of FIG. 8 as follows. Steps 100-102 are performed. Then, the user creates bitmap matrices defining the partitioned areas. Each area is then wavelet transformed. Different quantizations are then applied to the different areas according to the transformed matrices obtained above.

Split and Merge Wavelet Algorithm
for Big Image Compression

This algorithm allows users to compress large images by partitioning them into smaller pieces. The key is to divide the original image into several smaller pieces and compress/decompress them separately by using overlap and de-overlap technique. With this technique, the individually compressed pieces are equivalent to compressed whole image. The user does not see any edge effects in the decompressed image, which normally occur with conventional split and merge methods.

Also, with this algorithm, users can selectively decompress the whole image or choose a specific part to decompress according to an image map created during the compression phase. The algorithm is preferably implemented as a software program executing on a general purpose computer.

There are two ways to compress an image by splitting it: automatically or interactively. The automatic approach is transparent to users since the algorithm will automatically split to the image according to the characteristics of the computer used to perform the compression. Using the automated method, the algorithm first detects the size of the source image and the memory size of the host computer. Next, the image is split into several pieces with a predetermined number of pixels overlapping according to the image size and computer's memory. Overlapping pixels are those that appear in more than one piece of the split image.

Each piece of image is compressed in order according to any of the methods disclosed herein from the image resource.

The split image is decompressed as follows. First, the headers of the compressed image pieces are read to determine their order and compression parameters, such as quantization thresholds and decomposition levels. Next, each piece of the image is decompressed and de-

overlapped. Merge all pieces together in the proper place for display.

Using the interactive method, a user can indicate how many blocks they want to divide the image into and how many pixels they want for overlap. To compress an image according to this approach, the size of the source image is first detected. Then, the user's choice for the number of blocks and number of overlapping pixels is entered. Next, the image is divided into the pieces according to the user's choice and the size of the image. Finally, the individual pieces are compressed according to one of the methods disclosed herein.

The interactively split image is decompressed as follows. First, the header of the coded image is read. Next, an image map is displayed for the user to look at what the image context is about. The user can then choose to display entire image or a specific piece of image. If user chooses to display a single piece of image, the algorithm finds the position of this coded piece and decompresses it. If the user instead chooses to display the entire image, the algorithm decompress each piece of image and de-overlaps it. All pieces are then merged together in the appropriate display location.

Example A, below, shows further technical details related to the present invention.

While specific embodiments of the present invention have been shown and described, it will be apparent to this skilled in the art that the disclosed invention may be modified in numerous ways and may assume many embodiments other than the preferred form specifically set out and described above. Accordingly, it is intended by the appended claims to cover all modifications of the invention which fall within the true spirit and scope of the invention.

EXAMPLE A

1.0 Quality Compression Optimization

The following sections deal with the improvement of the compressed image quality. As stated in the report introduction, we believe this is the last significant major problem to be addressed in the still imagery compression system.

1.1 Introduction

The issue is the mechanism which can be used to improve the quality of the image based upon the nature of the parameters used in the compression process. We think this is an important issue, and in our experimentation, we have found fairly striking results. For example, if we hold the compression ratio constant, we can compress an image and obtain on a subjective scale of C, where

- A is no observable defect,
- B is observable but not noticeable,
- C is quite noticeable, but not distracting from the image,
- D is very noticeable and detracts from the image,
- E is unacceptable.

If we now change the parameters by a hand optimization, we find we can change the subjective evaluation from a C to a B with the compression ratio left alone. We believe this is significant, in that it guarantees a compression system which is tailored for each image independently, rather than have each image compressed by the same set of parameters irrespective of the image content.

To review the process of compression, consider the following figure, Figure 49.

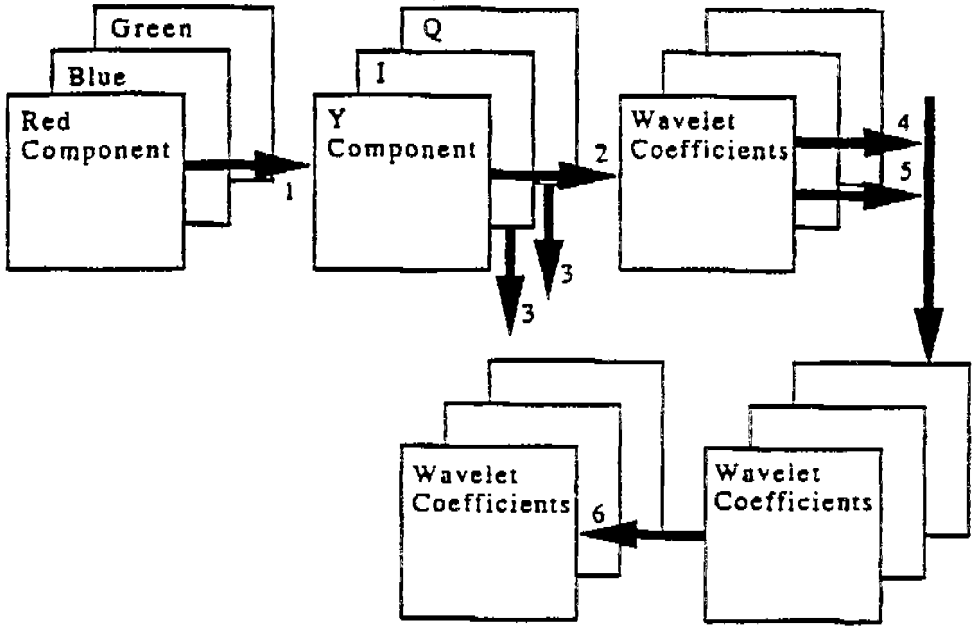


Figure 49
RGB Image in 24 Bit Color Depth Showing Transform to YIQ

Figure 49 shows the relationship between the RGB and YIQ color representation and the processes which take place in the algorithm as this data is transformed into the final lossy set of wavelet coefficients.. The processes which apply to the imagery are shown as heavy arrows with the process identifier shown as part of the arrow. If no process number is present, then the arrow is just a passive link between processes and data. The important point to note is that in color imagery, we deal with three images (one luminescence, and two color planes). The Y component is critical, while the IQ components are less sensitive to error introduced by the compression system.

The processes shown in Figure 49 are as follows:

- (1) Transform the RGB format into YIQ format, using long integers. This format is only an approximation of the YIQ format.

- (2) Transform the YIQ planes into a wavelet decomposition using our own integer wavelet transform. This produces the result shown in Figure 50, but for each plane.

- (3) We have shown an alternative process which is an optional down sampling for the IQ color planes. This down sampling may

be done once or twice to produce two image planes either one fourth or one sixteenth the size of the original plane. If this process is to be done, it will be accomplished prior to the wavelet transform of process (2).

- (4) Here the first step in the loss occurs. The wavelet coefficients are now quantized to a number of levels depending upon which quadrant is being processed, and the desired compression or quality factor.

- (5) Simultaneously with step (4), the wavelet coefficients are being matched against threshold values, and if the values are less than the established threshold values specified, then the resultant value is set to zero.

- (6) The last step in the process is to entropy compress the resultant coefficients using either Arithmetic, Run Length, or Huffman, of Huffman and Run Length combined. The key issue is the amount of compression desired against the invested time required to get that level of compression.

The issue now, irrespective of the down sampling or not of the IQ components, is the fact that we process the three planes into five levels in a decomposition as shown in Figure 50. From this figure, one can see a total of 16 regions (quadrants) which are defined by the numbers 1 through 16. Each of these sixteen quadrants have two additional parameters associated with them. The parameters define the quantization and threshold values for that particular quadrant. Since there are three planes for color (only one for gray level) the maximum number of parameters that the user can control is 96 -- 16 for quantization and 16 for thresholding for each of the three color layers. In the case of a gray level image, there are only parameters for one layer.

Our experience has shown the parameters for an image are very sensitive in some cases, and not in other cases. In order to measure this sensitivity, we generated the variance of the wavelet coefficients in the 16 quadrants. Table 1 provides these values for each of the three planes.

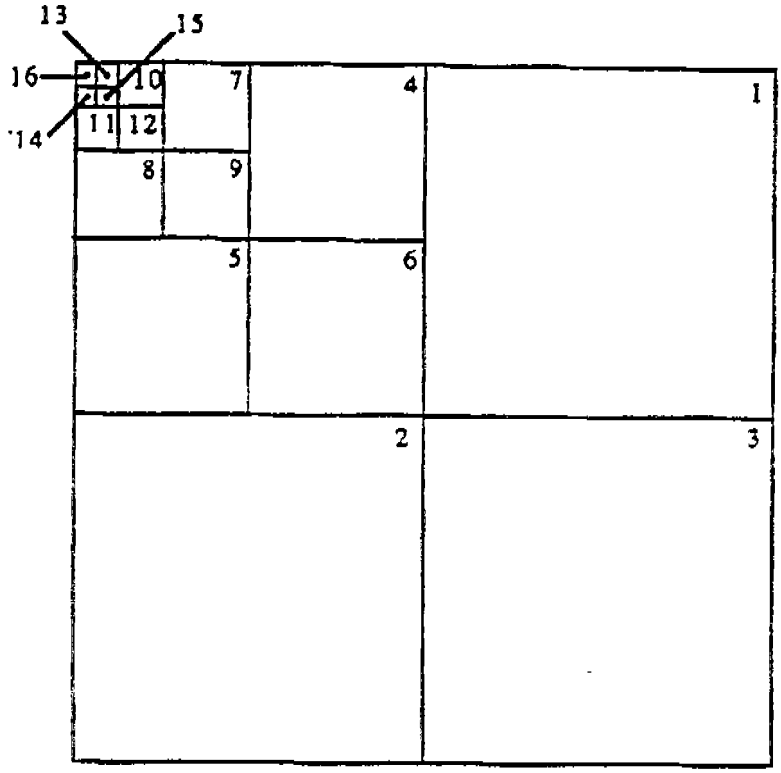


Figure 50
Decomposition of a Plane into Five Levels

Image	Lenna Wavelet Variance			Tulips Wavelet Variance		
Quadrant	Y	I	Q	Y	I	Q
1	13	4.7	1.4	164.2	27.3	6.6
2	6	3.2	1.1	134.6	24.9	5.0
3	1	1.3	0.6	28.9	4.6	1.1
4	56	5.8	1.7	276.1	57.7	12.9
5	23	4.5	1.1	274.3	67.3	12.4
6	10	2.8	0.8	124.8	22.7	5.0
7	128	11.4	2.6	225.7	73.2	13.7
8	55	7.4	1.1	298.1	107.3	16.2
9	37	3.3	1.0	114.2	35.5	6.8
10	253	27.5	4.1	174.8	73.9	10.7
11	75	11.8	1.5	285.1	128.2	14.9
12	64	6.5	1.1	107.9	47.8	6.7
13	499	41.7	10.1	135.4	75.6	8.7
14	138	11.2	2.5	245.0	144.0	11.2
15	156	12.2	2.0	89.3	49.1	5.1
16	14161	1,281.8	295.6	7,408.6	690.4	77.9

Table 1
Wavelet Coefficient Variance by Numbered Quadrant for Two Images

The clear information to be gained by the numbers in Table 1 is the images are quite different, and the quantization or threshold levels set for all images will only be an approximate solution at best. The optimum solution would be to have the quantization and threshold values set according to the variance values. Such settings could be found in a table with various ranges, and for each such range, the parameters of interest could be defined. This would give a more optimal solution, but still not the optimal solution. In order to get optimality, one would need to search over the variable space using the near optimal settings to find the actual best values for the parameters.

As the values in Table 1 grow, the implication is a finer mesh size for quantization. That is, the number of bits one wishes to allocate to the output could be varied by quadrant. Those quadrants with large variances will utilize more bits, while those with low variants will utilize fewer bits. In this way, the number of bits resulting from quantization will remain the same, but their allocation will differ depending upon the nature of the image.

1.2 Approach

The solution to the problem posed in the previous section is to determine how the ninety-six parameters interact with one another. The problem is a bit more sophisticated than just measuring parameters, however. The issue is with each parameter change, the compression ratio will change. If a compression ratio, or a quality factor which indirectly defines a compression ratio, is specified, then the user wants the compression ratio to remain identical over the changes in the parameters. In order to accomplish this, there are two parameters which we must monitor: PSNR (peak signal to noise ratio which is defined to be PSNR = 20 $\log_{10}$ (X/MSE) where the X is the average absolute value of the pixels in the after image and MSE is the mean squared error measured between the before and after image) and the compression ratio. The compression ratio must be held constant, and the PSNR needs to increase, and the way to increase the PSNR is to reduce the MSE.

The difficulty with this system as described is in many cases, small changes in the parameters introduce significant changes in the MSE. Also, we believe, the parameters are not independent. We have also seen images where the parameters can be changed in one way, then

altered, and the results are exactly the same. This indicates the optimal value is not a single point, but rather something like a plane with little slope.

1.3 Status

We have established the rules under which the optimal solution will need to exist, and are at the moment writing software to measure the variance within the Lightning Strike environment. Once this is done, we will being examining many images to see how close we can determine the optimal parameters for a defined set of variances.

CIS-2 Image Compression Algorithm

HONGYANG CHAO

Part I: Brief Review of LSIC 3.0

1. Introduction

CIS-2 (temporary name), which has been being used in **Lightning Strike 3.0** image compression software, is a wavelet based image compression algorithm. CIS-2 has following inventions:

- Integer reversible wavelet algorithm with Property of Precision Preservation;
- Subband oriented quantization and related entropy coding;
- Wavelet lossless compression for color and gray images;
- Progressive transmission algorithm for color bit compression;
- Progressive transmission and decompression algorithms;
- Non-uniform image compression algorithm;
- Quality based wavelet coefficient quantization tables;
- Attached optional post-processing filters;
- Image map editor;
- Optional peak signal noise ratio controlled compression;
- Special split and merge wavelet compression algorithm for very big image compression without any boundary effects ;
- Image dependent parameter optimization .

2. Main steps of the algorithm

In Lsic 3.0, three kinds of different image compression methods are included:

Method 1: Quality controlled wavelet based compression

Method 2: Color bit depth compression

Method 3: Wavelet lossless compression

Section 2.1-2.3 will give brief description of above method. The details will discuss later.

2.1. Main step of method 1

Figure 1 and Figure 2 give the flow charts of the image compression and decompression of method1 respectively. Every step in both compression and decompression has lot of details, which will be described later.

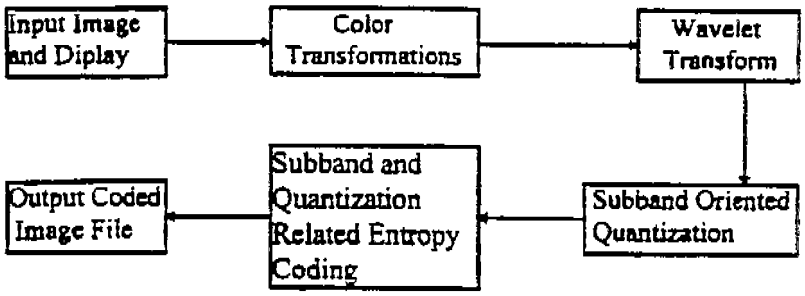


Figure 1: Compression flow chart for Method 1

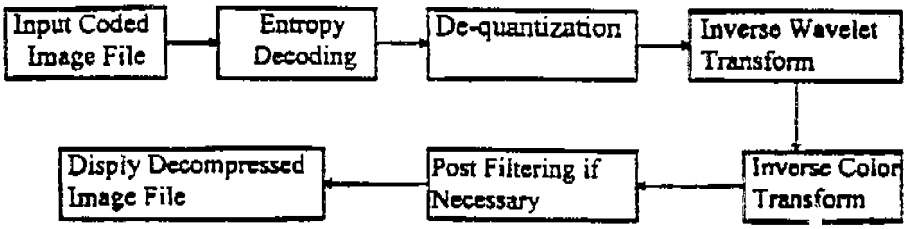


Figure 2: Decompression flow chart for Method 1

2.2. Main step of method 2

This method based on using less number of colors to approximately represent original images. Following figure gives the main step of the algorithm for the compression and decompression.

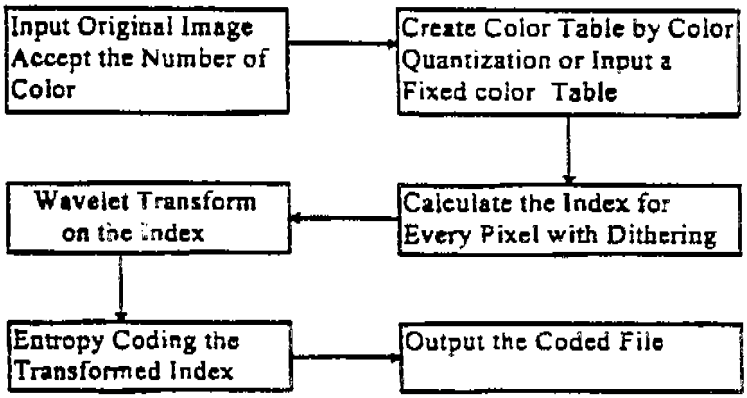


Figure 3: Compression flow chart for Method 2

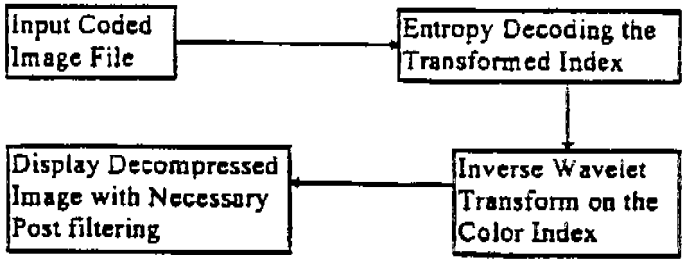


Figure 4: Decompression flow chart for Method 2

2.3. Main steps of method 3

The main steps of method 3 is almost as same as method 1. However, at the every step we use different methods. Following are its compression and decompression flow charts:

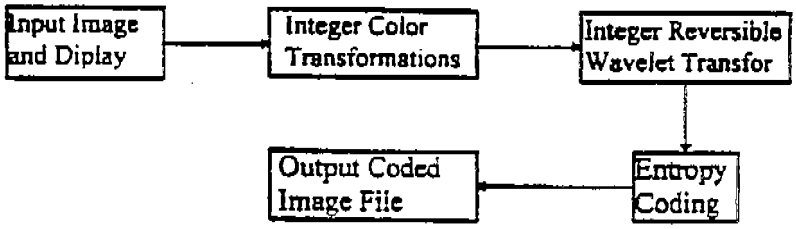


Figure 5: Compression flow chart for Method 3

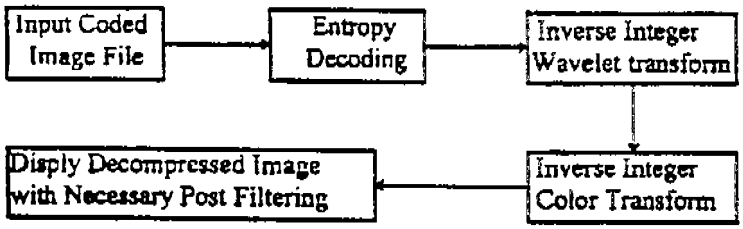


Figure 6: Decompression flow chart for Method 3

3. Brief Description for Other Features

Most Features (or inventions) are included in above three compression methods. Following is the brief introduction for some inventions list above.

3.1. Progressive decompression

This algorithm allows users to decode images from the lowest resolution to highest resolution. The advantage of this feature is that users can download small piece of the coded file and view the image at lower resolution to determine if they want to download the whole image.

Steps:

(a) Input the lowest pass component LL^0 of the coded file and reconstruct the lowest resolution image I^0;

(b) Display image I^0. If the user doesn't like it or the resolution is big enough for , stop; otherwise, go to next step;

(c) Input the lowest three band-pass components HL^0, LH^0 and HH^0 successively in the current coded file. Reconstruct the new image I^1 from LL^0, HL^0, LH^0 and HH^0. Let $I^0 = I^1$, go to step (b).

3.2. Non-uniform image compression

The algorithm allows users to divide the image into several parts with different interests and compress these areas with different qualities. The areas can have any shape. This algorithm is only available for method 1.

Original image goes though all of the procedure except quantization part, which follows the steps below:

(a) Creating the bitmap matrices related to the areas chosen by the user;

(b) Wavelet transform to every bitmap matrix;

(c) Different quantization in different areas according to the transformed matrices obtained above step.

3.3. Peak Signal Noise Ratio (PSNR) controlled compression

Peak Signal Noise Ratio (PSNR) is an image quality measurement used by most professional people. PSNR controlled compression allows users to choose their desired PSNR for the compressed image.

The related algorithm is an iterated system:

(a) Picking an initial parameter setting P_0;

(b) Quantize the wavelet coefficients with P_0 and calculate the corresponding PSNR;

(c) If the PSNR is close to desired one, stop and output the coded file; otherwise, get an adjusted vector ΔP_0 and set $P_0 \leftarrow P_0 + \Delta P_0$, go to step (b);

3.4. Attached optional post-processing filters

Users can choose any number of following processing filters at their compressing time. The desired results can stored in the coded image file, and, anyone who decompress the coded file will see the same result immediately.

- Sharpening images
- Smoothing images
- Improving the visual quality
- Brightening the images

3.5. Image map editor

Image Map Editor create a image map over Lsic3.0 compressed image file. User selects one or several areas of compressed image, assigns the http links to the areas, then, Image Map Editor calculates the coordinates of the areas and outputs a HTML associate with the image. User can add such information into his/her source code.

Following is an example of such image map:

<EMBED SRC="cow.cod" type = "image/cis-cod" WIDTH= "257" poly= "44, 45, 103, 78, 103, 86, 54, 86, 54, 78", href= "http://www.infinop.com"></EMBED>

3.6. Split and merge wavelet algorithm for very big image compression

This algorithm allows users to compress very big image by an ordinary machine. The key is to divide the original image into several smaller pieces and compress/decompress them separately by using overlap and dis-overlap technique. With this technique, the compression/decompression piece by piece is equal to compress/decompress the whole image together, which means users won't see any edge effect at decompressed image which appears at general split method.

Also, with this algorithm, users can either decompress the whole image or choose the specific part to decompress according to a image map we create for the division .

3.7. Image dependent optimized parameter setting

This algorithm allows user to get the best (or almost best) image quality at the desired compression ratio by choosing image related parameter setting.

3.8. Integer reversible wavelet algorithm with PPP property

See attached unpublished paper titled as "An Approach to Fast Integer Reversible Wavelet Transformations for Image Compression"

3.9. Integer color transformation

This algorithm is a integer reversible transform which has been used in lossless color image compression (Method 3) for Lsic 3.0.

The algorithm transform RGB color components to a new set of color components Y-Nb-Nr:

Forward transform RGB to Y-Nb-Nr:

$$Y = G + \mathrm{Int}\left(\frac{R+B}{2}\right),$$
$$Nb = B - \mathrm{Int}\left(\frac{Y}{2}\right),$$
$$Nr = R - \mathrm{Int}\left(\frac{Y}{2}\right).$$

Inverse transform Y-Nb-Nr to RGB:

$$R = Nr + \mathrm{Int}\left(\frac{Y}{2}\right),$$
$$B = Nb + \mathrm{Int}\left(\frac{Y}{2}\right),$$
$$G = Y - \mathrm{Int}\left(\frac{R+B}{2}\right).$$

3.10. Subband related Quantization and entropy coding

This entropy coding method is just designed for wavelet based image compression. The main idea is to take the advantage of different quantization at different subbands and encode each band according to its content. This method reduce the coding cost greatly.

CIS-1 Image Compression Algorithm

HONGYANG CHAO*
Computer and Information Science Inc.

1. Introduction

CIS-1, which has been being used in **Lightning Strike** image compression software, is a wavelet based image compression algorithm. CIS-1 has following advantages:

- Reach almost optimal compression ratio;
- Keep the major characteristics as more as possible. In other words, it reduce insignificant components gradually according to human visual system, so that people can still accept the image quality at the extremely high compression ratio;
- Fast.

2. Main steps of the algorithm

Figure 1 and Figure 2 give the flow charts of the image compression and decompression respectively. Every st p in both compression and decompression has lot of details, which will be described later.

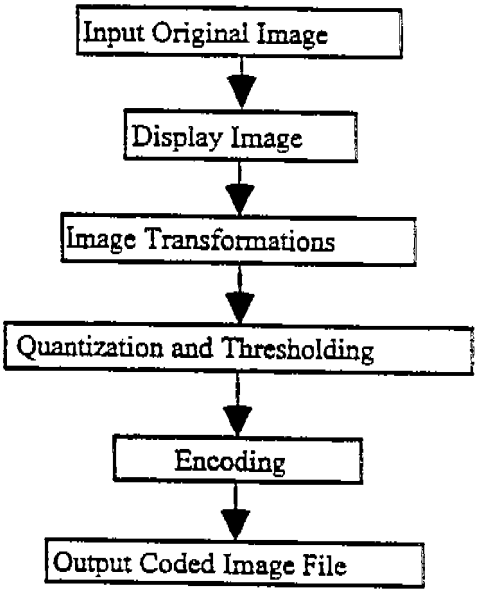


Figure 1: Image compression

*

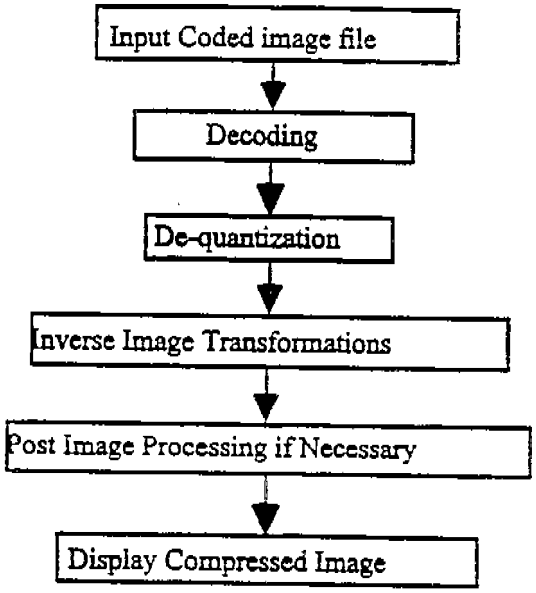


Figure 2: Image decompression

3. Image Compression

For the compression, we only have to describe three parts: image transformations, quantization and thresholding, and entropy coding.

3.1. Image Transformations

The image transformations involved in this algorithm include color transform (for color images) and wavelet decomposition (for both gray level images and color images).

(a) Color transform

In general, input color images are based on RGB color model, such as TIFF or BMP images. In order to get high compression ratio, it is better to change RGB color model to other color models, such as YIQ or YUV models.

RGB to YIQ:

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.275 & -0.321 \\ 0.212 & -0.523 & 0.311 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}.$$

RGB to YUV:

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.148 & -0.289 & 0.439 \\ 0.615 & -0.515 & -0.1 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}.$$

(b) Wavelet decomposition

The purpose of wavelet transform is to represent the original image by different basis to achieve the objective of decorrelation. There are a lot of wavelets which can be used in this step. In CIS-1, we use a wavelet which results in the following algorithm: Suppose $\mathbf{C}^0 = \left[c_{jk}^0\right]_{M\times N}$ $(j = 0,\cdots,M-1;\quad k = 0,\cdots,N-1)$ is original image, where M and N are integers which have the common factor 2^L (L is a positive integer). After one-level wavelet decomposition, we will get four parts as shown in figure 3.

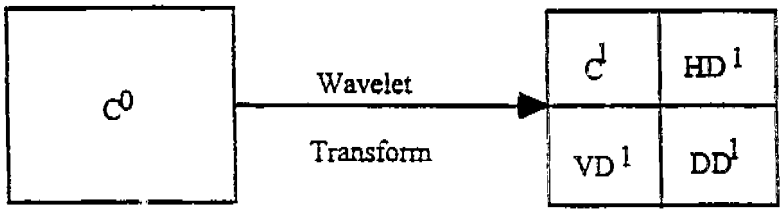


Figure 3. Wavelet image decomposition

We call $\mathbf{C}^1 = \left[c_{jk}^1\right]$ $(j = 0,\cdots,\frac{M}{2}-1;\ k = 0,\cdots,\frac{N}{2}-1)$ the blurred image of $\mathbf{C}^0$, $\mathbf{HD}^1$ the horizontal high frequency part of $\mathbf{C}^0$, $\mathbf{VD}^1$ the vertical high frequency part, and $\mathbf{DD}^1$ the diagonal ones. Setting $\mathbf{C}^0 = \mathbf{C}^1$, we can repeat the same procedure L times or until the size of the new blurred image $\mathbf{C}^1$ is small enough. Therefore, we only have to give the algorithm for one level decomposition:

(1) Let $\overline{\mathbf{C}}^0 = r\mathbf{C}^0$, $r>0$ is a factor which can be changed for different needs.

(2) Transform for image columns:

- For $k = 0,\cdots,N-1$, calculate

$$\begin{cases} \tilde{d}_{0k}^1 = \dfrac{\bar{c}_{1k}^0 - \bar{c}_{0k}^0}{2}, \\ \tilde{d}_{jk}^1 = \dfrac{1}{4}(\bar{c}_{2j-1,k}^0 - 2\bar{c}_{2j,k}^0 + \bar{c}_{2j+1,k}^0),\quad j = 1,\cdots,\frac{M}{2}-1. \end{cases} \tag{3.1.1}$$

- For $k = 0,\cdots,N-1$, calculate

$$\begin{cases} \tilde{c}^1_{0,k} = \tilde{c}^0_{1,k} - \dfrac{\tilde{d}^1_{0,k} + \tilde{d}^1_{1,k}}{2}, \\ \tilde{c}^1_{j,k} = \tilde{c}^0_{2j+1,k} - \dfrac{\tilde{d}^1_{j,k} + \tilde{d}^1_{j+1,k}}{2}, \quad j = 1,\cdots,\frac{M}{2}-2, \\ \tilde{c}^1_{\frac{N}{2}-1,k} = \tilde{c}^0_{N-1,k} - \tilde{d}^1_{\frac{N}{2}-1,k}. \end{cases} \qquad (3.1.2)$$

(3) Transform for rows:

o For $j = 0,\cdots,\frac{M}{2}-1$, computing

$$\begin{cases} hd^1_{j,0} = \dfrac{\tilde{c}^1_{j,1} - \tilde{c}^1_{j,0}}{2}, \\ hd^1_{j,k} = \dfrac{1}{4}(\tilde{c}^1_{j,2k-1} - 2\tilde{c}^1_{j,2k} + \tilde{c}^1_{j,2k+1}), \quad k = 1,\cdots\frac{N}{2}-1. \end{cases} \qquad (3.1.3)$$

and

$$\begin{cases} c^1_{j,0} = \tilde{c}^1_{j,1} - \dfrac{hd^1_{j,0} + hd^1_{j,1}}{2}, \\ c^1_{j,k} = \tilde{c}^1_{j,2k+1} - \dfrac{hd^1_{j,k} + hd^1_{j,k+1}}{2}, \quad k = 1,\cdots,\frac{M}{2}-2, \\ c^1_{j,\frac{N}{2}-1} = \tilde{c}^1_{j,N-1} - hd^1_{j,\frac{N}{2}-1}. \end{cases} \qquad (3.1.4)$$

o For $j = 0,\cdots,\frac{M}{2}-1$, computing

$$\begin{cases} dd^1_{j,0} = \dfrac{\tilde{d}^1_{j,1} - \tilde{d}^1_{j,0}}{2}, \\ dd^1_{j,k} = \dfrac{1}{4}(\tilde{d}^1_{j,2k-1} - 2\tilde{d}^1_{j,2k} + \tilde{d}^1_{j,2k+1}), \quad k = 1,\cdots,\frac{N}{2}-1. \end{cases} \qquad (3.1.5)$$

and

$$\begin{cases} vd^1_{j,0} = \tilde{d}^1_{j,1} - \dfrac{dd^1_{j,0} + dd^1_{j,1}}{2}, \\ vd^1_{j,k} = \tilde{d}^1_{j,2k+1} - \dfrac{dd^1_{j,k} - dd^1_{j,k+1}}{2}, \quad k = 1,\cdots,\frac{M}{2}-2, \\ vd^1_{j,\frac{N}{2}-1} = \tilde{d}^1_{N-1,k} - dd^1_{j,\frac{N}{2}-1}. \end{cases} \qquad (3.1.6)$$

(4) $\mathbf{C}^1 = \left[c^1_{j,k}\right]$, $\mathbf{HD}^1 = \left[hd^1_{j,k}\right]$, $\mathbf{VD}^1 = \left[vd^1_{j,k}\right]$ and $\mathbf{DD}^1 = \left[dd^1_{j,k}\right]$, $j = 0,\cdots,\frac{M}{2}-1$; $k = 0,\cdots,\frac{M}{2}-1$.

Remark: If it is necessary, we also can use matrix multiply

Wavelet Coefficient Image of l levels = $\mathbf{W}_l\mathbf{C}^0\mathbf{W}_l^T$.

Here, W_l is the transform matrix for l level wavelet decomposition.

3.2. Thresholding and Quantization

Both thresholding and Quantization allow us to reduce accuracy with which the wavelet coefficients are represented when converting the wavelet decomposition to an integer representation. This can be very important in image compression, as it tends to make many coefficients zeros--especially those for high spatial frequencies.

After L level, for example L=3, wavelet decomposition, we get the wavelet coefficients of the original image as plotted in Figure 4:

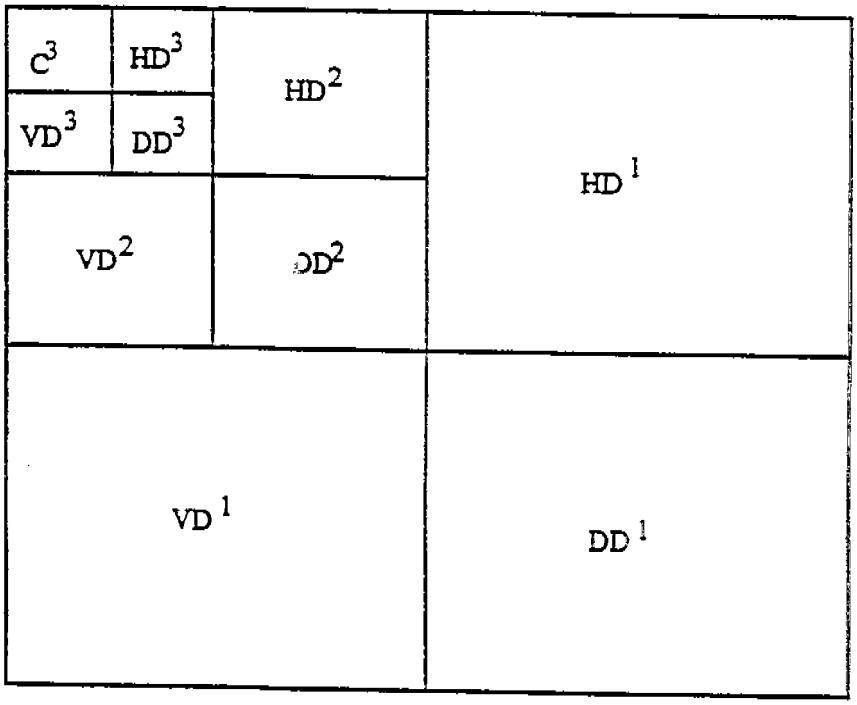


Figure 4. L=3 wavelet coefficients distribution

(a) Thresholding

In algorithm CIS-1, we use multilevel uniform thresholding method: Let

$$\mathbf{T} = (t_1, \cdots, t_L, t_{L+1})$$

be the chosen thresholds, where t_l is the threshold for l th $(l = 1, \cdots, L)$ level and t_{L+1} is a threshold for blurred image $\mathbf{C}^L$. Thresholding is to set every entry in the blocks $\mathbf{C}^L$,

$\mathbf{HD}^l$, $\mathbf{VD}^l$ and $\mathbf{DD}^l$ $(l=1,\cdots,L)$ to be zeros if its absolute value is not greater than the corresponding threshold.

Remark. For color image, we can have three threshold vectors which correspond three different color bands, such as Y, I and Q.

(b) Quantization

Quantization is to scale the wavelet coefficients and truncate them to integer values. In CIS-1, we use the quantization table shown in Table 1 to implement it.

q^1_{HD}	q^2_{HD}	$\bullet\bullet\bullet$	q^L_{HD}	
q^1_{VD}	q^2_{VD}	$\bullet\bullet\bullet$	q^L_{VD}	q_C^{L+1}
q^1_{DD}	q^2_{DD}	$\bullet\bullet\bullet$	q^L_{DD}	

Table 1. Quantization table

Here, the entries q^l_{HD} are quantization factors for blocks $\mathbf{HD}^l$ $(l=1,\cdots,L)$, q^l_{VD} and q^l_{DD} for blocks $\mathbf{VD}^l$ and $\mathbf{DD}^l$ $(l=1,\cdots,L)$ respectively, and the factor q_C^{L+1} is for the most blurred image $\mathbf{C}^L$. All the factors are integers between 0 and 255. The quantization scheme for the block $\mathbf{HD}^l$ $(l=1,\cdots,L)$ is

$$\overline{hd}^l_{j,k} = round\left(\frac{hd^l_{j,k} \bullet q^l_{HD}}{\max^l_{HD}}\right),\ j=0,\cdots,\tfrac{M}{2^l}-1;\quad k=0,\cdots,\tfrac{N}{2^l}-1. \quad (3.2.1)$$

Here, $\overline{hd}^l_{j,k}$ $(j=0,\cdots,\frac{M}{2^l}-1;\quad k=0,\cdots,\frac{N}{2^l}-1)$ are quantized wavelet coefficients in block $\mathbf{HD}^l$ $(l=1,\cdots,L)$,

$$\max^l_{HD} = \max_{\substack{0\le j\le(M/2^l-1)\\ 0\le k\le(N/2^l-1)}}\left(\left|hd^l_{j,k}\right|\right),$$

and the function *round* (x) gives the nearest integer of x. The scheme of quantization for other blocks are the similar to (3.2.1).

Remark. For color image, as the same as thresholding, we can have three separate quantization tables for different color bands.

3.3. Entropy Coding

Here, the encoding means the lossless compression for the wavelet coefficients. It is divided into two parts: Coefficient arrangement and entropy coding (Huffman or arithmetic).

4. Decompression

4.1 Decoding

Decoding, just as encoding, can be divided into two parts: Entropy decoding (Huffman or arithmetic), and coefficient rearranging.

4.2 Dequantization

After Decoding, we get quantized wavelet coefficients in $3*L+1$ Blocks. Dequantizing uses the same quantization table as quantizing, and the scheme as follow: for $l = 1,\cdots,L$

$$\widetilde{hd}^l_{j,k} = \frac{\widetilde{hd}^l_{j,k} * \max^l_{HD}}{q^l_{HD}},\ j = 0,\cdots,\frac{M}{2^l}-1;\quad k = 0,\cdots,\frac{N}{2^l}-1. \qquad (4.2.1)$$

(4.2.1) allows us to get the approximate coefficients for the blocks $\mathbf{HD}^l$ $(l = 1,\cdots,L)$, which is shown in Figure 4 . The dequantizing scheme for other blocks are similar to (4.1.2).

4.3 Inverse Image Transformations

(a) Wavelet reconstruction

We are going to describe the algorithm for one-level reconstruction which is plotted in Figure 5.

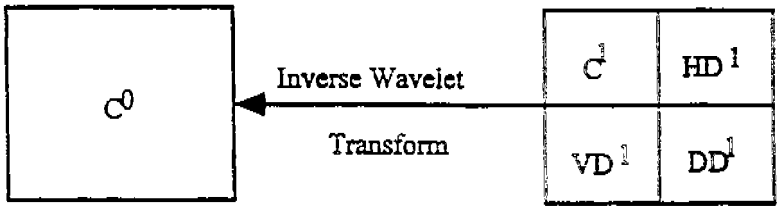


Figure 5. One-level wavelet reconstruction

(1) Inverse transform for rows:

o For $j = 0,\cdots,\frac{M}{2}-1$, calculate

$$\begin{cases} \tilde{d}^1_{j,1} = vd^1_{j,0} + \dfrac{dd^1_{j,0} + dd^1_{j,1}}{2} \\ \tilde{d}^1_{j,2k+1} = vd^1_{jk} + \dfrac{dd^1_{j,k} - dd^1_{j,k+1}}{2}, \quad k = 1,\cdots,\frac{N}{2}-2, \\ \tilde{d}^1_{N-1,k} = vd^1_{j,\frac{N-2}{2}} + dd^1_{j,\frac{N-2}{2}}. \end{cases} \qquad (4.3.1)$$

and

$$\begin{cases} \tilde{d}^1_{j,0} = \tilde{d}^1_{j,1} - 2dd^1_{j,0}, \\ \tilde{d}^1_{j,2k} = \dfrac{(\tilde{d}^1_{j,2k-1} + \tilde{d}^1_{j,2k+1})}{2} - 2dd^1_{j,k}, \quad k = 1, \cdots, \frac{N}{2} - 1. \end{cases} \tag{4.3.2}$$

o For $j = 0, \cdots, \frac{M}{2} - 1$, calculate

$$\begin{cases} \tilde{c}^1_{j,1} = c^1_{j0} + \dfrac{hd^1_{j0} + hd^1_{j1}}{2}, \\ \tilde{c}^1_{j,2k+1} = c^1_{jk} + \dfrac{hd^1_{j,k} + hd^1_{j,k+1}}{2}, \quad k = 1, \cdots, \frac{N}{2} - 2, \\ \tilde{c}^1_{j,N-1} = c^1_{j,\frac{N}{2}-1} + hd^1_{j,\frac{N}{2}-1}. \end{cases} \tag{4.3.3}$$

and

$$\begin{cases} \tilde{c}^1_{j0} = \tilde{c}^1_{j1} - 2hd^1_{j,0}, \\ \tilde{c}^1_{j,2k} = \dfrac{1}{2}(\tilde{c}^1_{j,2k-1} + \tilde{c}^1_{j,2k+1}) - 2hd^1_{jk}, \quad k = 1, \cdots \frac{N}{2} - 1. \end{cases} \tag{4.3.4}$$

(2) Inverse transform for column:

o For $k = 0, \cdots, N-1$, calculate

$$\begin{cases} \tilde{c}^0_{1,k} = \tilde{c}^1_{0k} + \dfrac{\tilde{d}^1_{0,k} + \tilde{d}^1_{1,k}}{2}, \\ \tilde{c}^0_{2j+1,k} = \tilde{c}^1_{jk} + \dfrac{\tilde{d}^1_{jk} + \tilde{d}^1_{j+1,k}}{2}, \quad j = 1, \cdots, \frac{M}{2} - 2, \\ \tilde{c}^0_{N-1,k} = \tilde{c}^1_{\frac{N}{2}-1,k} + \tilde{d}^1_{\frac{N}{2}-1,k}. \end{cases} \tag{4.3.5}$$

and

$$\begin{cases} \tilde{c}^0_{0k} = \tilde{c}^0_{1k} - 2\tilde{d}^1_{0k}, \\ \tilde{c}^0_{2j,k} = \dfrac{1}{2}(\tilde{c}^0_{2j-1,k} + \tilde{c}^0_{2j+1,k}) - 2\tilde{d}^1_{jk}, \quad j = 1, \cdots, \frac{M}{2} - 1. \end{cases} \tag{4.3.6}$$

(3) $c^0_{j,k} = \tilde{c}^0_{j,k} / r, \quad j = 0, \cdots, M-1; \quad k = 0, \cdots, N-1.$ $C^0 = \left[c^0_{j,k}\right]_{M \times N}$.

(b) Inverse color transform

For color image, we have to do inverse color transform

o YIQ to RGB

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.000 & 0.956 & 0.621 \\ 1.000 & -0.272 & -0.647 \\ 1.000 & -1.106 & 1.703 \end{bmatrix} \begin{bmatrix} Y \\ I \\ Q \end{bmatrix}.$$

$$\begin{cases} \tilde{d}^1_{j,0} = \tilde{d}^1_{j,1} - 2dd^1_{j,0}, \\ \tilde{d}^1_{j,2k} = \dfrac{(\tilde{d}^1_{j,2k-1} + \tilde{d}^1_{j,2k+1})}{2} - 2dd^1_{j,k}, \quad k = 1, \cdots, \frac{N}{2} - 1. \end{cases} \tag{4.3.2}$$

o For $j = 0, \cdots, \frac{M}{2} - 1$, calculate

$$\begin{cases} \tilde{c}^1_{j,1} = c^1_{j0} + \dfrac{hd^1_{j0} + hd^1_{j1}}{2}, \\ \tilde{c}^1_{j,2k+1} = c^1_{jk} + \dfrac{hd^1_{j,k} + hd^1_{j,k+1}}{2}, \quad k = 1, \cdots, \frac{N}{2} - 2, \\ \tilde{c}^1_{j,N-1} = c^1_{j,\frac{N}{2}-1} + hd^1_{j,\frac{N}{2}-1}. \end{cases} \tag{4.3.3}$$

and

$$\begin{cases} \tilde{c}^1_{j0} = \tilde{c}^1_{j1} - 2hd^1_{j,0}, \\ \tilde{c}^1_{j,2k} = \dfrac{1}{2}(\tilde{c}^1_{j,2k-1} + \tilde{c}^1_{j,2k+1}) - 2hd^1_{jk}, \quad k = 1, \cdots \frac{N}{2} - 1. \end{cases} \tag{4.3.4}$$

(2) Inverse transform for column:

o For $k = 0, \cdots, N-1$, calculate

$$\begin{cases} \tilde{c}^0_{1,k} = \tilde{c}^1_{0k} + \dfrac{\tilde{d}^1_{0,k} + \tilde{d}^1_{1,k}}{2}, \\ \tilde{c}^0_{2j+1,k} = \tilde{c}^1_{jk} + \dfrac{\tilde{d}^1_{jk} + \tilde{d}^1_{j+1,k}}{2}, \quad j = 1, \cdots, \frac{M}{2} - 2, \\ \tilde{c}^0_{N-1,k} = \tilde{c}^1_{\frac{N}{2}-1,k} + \tilde{d}^1_{\frac{N}{2}-1,k}. \end{cases} \tag{4.3.5}$$

and

$$\begin{cases} \tilde{c}^0_{0k} = \tilde{c}^0_{1k} - 2\tilde{d}^1_{0k}, \\ \tilde{c}^0_{2j,k} = \dfrac{1}{2}(\tilde{c}^0_{2j-1,k} + \tilde{c}^0_{2j+1,k}) - 2\tilde{d}^1_{jk}, \quad j = 1, \cdots, \frac{M}{2} - 1. \end{cases} \tag{4.3.6}$$

(3) $c^0_{j,k} = \tilde{c}^0_{j,k} / r$, $j = 0, \cdots, M-1$; $k = 0, \cdots, N-1$. $\mathbf{C}^0 = \left[c^0_{j,k}\right]_{M \times N}$.

(b) Inverse color transform

For color image, we have to do inverse color transform

o YIQ to RGB

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.000 & 0.956 & 0.621 \\ 1.000 & -0.272 & -0.647 \\ 1.000 & -1.106 & 1.703 \end{bmatrix} \begin{bmatrix} Y \\ I \\ Q \end{bmatrix}.$$

o YUV to RGB

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.000 & 0.000 & 1.140 \\ 1.000 & -0.395 & -0.581 \\ 1.000 & 2.032 & 0.000 \end{bmatrix} \begin{bmatrix} Y \\ U \\ V \end{bmatrix}$$

4.4 Necessary Post Image Processing

(a) Color Quatization

An Approach to Fast Integer Reversible Wavelet Transforms for Image Compression*

Hongyang Chao**
Computer and Information Science Inc.
3401 E. University, Suite 104, Denton, TX 76208
AND
Dept. of Computer Science, Zhongshan Univ.
Guangzhou 510275, P.R. China

Paul Fisher
Computer and Information Science Inc.
3401 E. University, Suite 104, Denton, TX 76208
AND
Dept. of Computer Science, Univ. of North Texas
Denton, TX 76205

Abstract

In this paper, we propose a general method for creating integer wavelet transforms which can be used in both lossless (reversible) and lossy compression of signals and images with arbitrary size. This method allows us to get a series of transformations which are very close to the corresponding biorthogonal wavelet transforms or some non-orthogonal wavelet transforms, but can be calculated with only integer addition and bit-shift operations. In addition, the integer wavelet transforms created in this paper possess a property of precision preservation (PPP). This property is very useful for conserving memory in both compression and decompression, and speed up the whole procedure in some applications. The motivation of this paper comes from the lifting scheme [1] and S+P transform [3].

* This work has been partially supported by the US Navy under SBIR Contract N00039-94-C-0013.

** The author is partially supported by the National Science Foundation of P. R. China and the Science Foundation of Zhongshan University.

1. Introduction

The wavelet transform has proven to be one of the most powerful tools in the field of image compression. Theoretically, the wavelet transformation is lossless, but since all computers have only finite precision, most of transformations are lossy in practice, even when we use floating point calculations. On the other hand, integer calculations are much faster than floating point for virtually all computers; and integer computations are much easier to implement in hardware which is more important in some applications. The memory utilization of integers is also a positive consideration. The difficulty is, if we directly use integers in the wavelet transform and its inverse without some proper considerations, it will cause the loss of accuracy. For some important image applications, the user wants to have complete control of the precision in which the image pixels are represented during the compression process, and thus prefers to have the image compressed from lossless to lossy.

Lossless compression is also very important for images found in such applications as medical and space science. In such situations, the designer of the compression algorithm must be very careful to avoid discarding any information that may be required or even useful at some later point From the academic point of view, it is also very interesting to have a compression scheme which has very fast performance, and which can exactly reconstruct the image when necessary. In addition, it is also very useful to have some wavelet transforms which exhibit the property of precision preservation (PPP), which can be utilized in the computer which has limited precision and limited memory without losing any precision during the computation.

In this paper, we are going to describe two general methods from which one can get the integer wavelet transform desired. All of the wavelet transforms from the methods given in this paper possess the property of precision preservation (PPP). We draw on the work of several other authors who have already contributed to this area [2-3], where some specific examples were developed. However, this paper presents a more general method which allows one to see several new results as well as those presented and acknowledged prior to this work.

This paper is organized as follows: Section 2 and 3 give some examples of integer wavelet transforms. The examples in section 2 are the starting point for our approach, and the examples in Section 3 show the steps and motivation of our general method. Section 4 indicates how one can use the lifting technique to create an integer biorthogonal wavelet transform. The Correction technique to generate more general integer wavelet transforms is described in Section 5. Section 6 describes how to process boundaries in order to apply the integer calculation in finite sized images or signals. In Section 7, we prove the integer wavelet transforms developed by both the lifting and correction method possess the property of precision preservation (PPP). Some example images are also shown in this section. The last section, Section 8, provides the conclusion to this paper.

2. Basic integer wavelet transformations

We provide the following two examples as the starting point for our new method. For the sake of convenience, length, and simplicity, we only discuss the algorithm for a one level decomposition and reconstruction and only for a one dimensional signal. The extension to two dimensions is immediate as the rows and columns can be treated into a sequence of one dimensional signals. For the following examples, assume that $\{c_n^0\}_{n=0}^{N-1}$ is the original signal where the superscript indicates level and the subscript indicates a particular point in the signal. Also, $\{c_n^1\}_{n=0}^{N_1-1}$ and $\{d_n^1\}_{n=0}^{M_1-1}$ are its decomposition parts at the first level. Here

$$\begin{cases} N_1 = \begin{cases} \frac{N}{2}, & \text{if } N \text{ is an even number,} \\ \frac{N+1}{2}, & \text{if } N \text{ is an odd number;} \end{cases} \\ M_1 = N - N_1. \end{cases}$$

$\{c_n^1\}_{n=0}^{N_1-1}$ and $\{d_n^1\}_{n=0}^{M_1-1}$ are its low frequency (l) part and high frequency (h) part, respectively. For multi-levels, we just treat $\{c_n^1\}_{n=0}^{N_1-1}$ as $\{c_n^0\}_{n=0}^{N-1}$ and repeat the procedure again.

Example 1: A (2,2)-wavelet transform by integer calculation.

This transformation is similar to a variation of the Haar wavelet transform which uses low and high pass analysis (decomposition) filters given as:

n	0	1
$\tilde{h}_n$	1/2	1/2
$\tilde{g}_n$	1/2	-1/2

(1) Compute

$$d_k^1 = c_{2k}^0 - c_{2k+1}^0, \quad k = 0, \cdots, M_1 - 1. \tag{2.1}$$

(2) Compute

$$\begin{aligned} c_k^1 &= \mathrm{Int}(\frac{d_k^1}{2}) + c_{2k+1}^0, \quad k = 0, \cdots, N_1 - 2, \\ c_{N_1-1}^1 &= \begin{cases} \mathrm{Int}(\frac{d_{N_1-1}^1}{2}) + c_{N-1}^0, & \text{if } N \text{ is an even number,} \\ c_{N-1}^0, & \text{if } N \text{ is an odd number.} \end{cases} \end{aligned} \tag{2.2}$$

Here, $\mathrm{Int}(x)$ is an arbitrary rounding function which may have different interpretations. For example, $\mathrm{Int}(x)$ can be the integer which is nearest to x, or $\mathrm{Int}(x)$ may be any integer which satisfies $x - 1 < \mathrm{Int}(x) \le x$, etc. It is easy to see that all entries in both $\{c_n^1\}_{n=0}^{N_1-1}$ and $\{d_n^1\}_{n=0}^{M_1-1}$ are integers.

IPR2023-00330 Page 00891

From (2.1)-(2.2), we can easily get the following integer reconstruction algorithm:

(b) Reconstruction

(1) If N is an even number, compute

$$c^0_{2k+1} = c^1_k - \mathrm{Int}(\frac{d^1_k}{2}), \quad k = 0, \cdots, N_1 - 1 \ ; \tag{2.3}$$

or, if N is an odd number, we have

$$\begin{aligned} c^0_{2k+1} &= c^1_k - \mathrm{Int}(\frac{d^1_k}{2}), \quad k = 0, \cdots, N_1 - 2, \\ c^0_{N-1} &= c^1_{N_1}. \end{aligned} \tag{2.4}$$

(2) Compute

$$c^0_{2k} = d^1_k + c^0_{2k+1}, \quad k = 0, \cdots, M_1 - 1. \tag{2.5}$$

Remark Since (2.1)-(2.6) are not linear because of the rounding operation $\mathrm{Int}(x)$, this means the transformation order becomes significant. For instance, if the decomposition was applied first to the columns and then to the rows, the inverse transformation must be applied first to the rows and then to the columns.

Example 2: Lazy wavelet transform.

The Lazy wavelet transform *does not do anything*. However, this illustrates an important concept. The corresponding inverse transform is nothing else but sub-sampling the even and odd indexed samples. Decomposition and reconstruction can use same formula as follows:

$$\begin{aligned} c^1_k &= c^0_{2k}, \ k = 0, \cdots, N_1 - 1; \\ d^1_k &= c^0_{2k+1}, \ k = 0, \cdots, M_1 - 1. \end{aligned}$$

Examples 1 and 2 are not good transforms for image compression, but they are simple. Much better transforms can be achieved from these two. As suggested above, we consider them only as a starting point for our integer, reversible, wavelet transform algorithm.

We must mention that there is another interesting property in the above two transforms which may not be easily seen. If the values of the signal pixels are represented by a finite number of bits, say one bit or one byte, we can still use the same number of bits to represent the result of the forward transform within the computer itself because of the complementary code property. While, from the reconstruction algorithm, the computer will get back the exact original signal through the same complementary code property. We call this property *a Property of Precision Preservation (PPP)* for these wavelets.

It is known that the general values of the high frequency wavelet coefficients are small, and all higher levels in the decomposition also provide generally small values in the high frequency band. This allows the preservation of precision during the computational stage of the wavelet coefficients. Now, the complementary code property, the other aspect of the *PPP* property is a well known characteristic of integer arithmetic as done by the computer. Consider the computation of the difference of two integers given as $c = b\text{-}a$ and the inverse computation of $a = b - c$. The nature of the computation within the computer can be specified as follows:

$$c_m = \begin{cases} b-a & \text{if } -2^{q-1} \le b-a < 2^{q-1}-1 \\ -2^q + b - a & \text{if } b-a \ge 2^{q-1} \\ 2^q + b - a & \text{if } b-a < -2^{q-1} \end{cases}$$

and the inverse is

$$a_m = \begin{cases} b-c_m & \text{if } -2^{q-1} \le b-c_m < 2^{q-1}-1 \\ -2^q + b - c_m & \text{if } b-c_m \ge 2^{q-1} \\ 2^q + b - c_m & \text{if } b-c_m < -2^{q-1} \end{cases}$$

where the m subscript indicates the internal representation, and the range of the integers a, b, c is $\left[-2^{q-1},\ 2^{q-1}-1\right]$. The internal representation of c_m when it is outside the range, its appearance is as a two's complement number, so the representation may not be the same as the external representation of c. However, the same complementary code for the a_m will cause the internal representation to be identical to the external representation of a. For example, if we let $b=2$ (00000010) and $a=-127$ (10000001) then c_m has the internal binary value of (10000001) when $q=4$. With a value of -127 for c_m, the inverse value for a_m will just be a.

In fact, for Example 2, this property is obviously true. While for Example 1, if the range of the pixel values is within a finite number of bits, say q, we can only use q bits as the working unit, which means the value of transform coefficients will also be within the interval with length 2^q, say $\left[-2^{q-1},\ 2^{q-1}-1\right]$. Due to the nature of computation on a machine, most machines will implement (2.1)-(2.2) automatically as follows (the complementary code property):

$$d_k^1 = \begin{cases} c_{2k}^0 - c_{2k+1}^0, & \text{if } -2^{q-1} \le c_{2k}^0 - c_{2k+1}^0 < 2^{q-1}, \\ c_{2k}^0 - c_{2k+1}^0 - 2^q, & \text{if } c_{2k}^0 - c_{2k+1}^0 \ge 2^{q-1}, \\ 2^q + \left(c_{2k}^0 - c_{2k+1}^0\right), & \text{if } c_{2k}^0 - c_{2k+1}^0 < -2^{q-1}. \end{cases} \tag{2.6}$$

$$c_k^1 = \begin{cases} \mathrm{Int}(\frac{d_k^1}{2}) + c_{2k+1}^0, & \text{if } -2^{q-1} \le \mathrm{Int}(\frac{d_k^1}{2}) + c_{2k+1}^0 < 2^{q-1}, \\ \mathrm{Int}(\frac{d_k^1}{2}) + c_{2k+1}^0 - 2^q, & \text{if } \mathrm{Int}(\frac{d_k^1}{2}) + c_{2k+1}^0 \ge 2^{q-1}, \\ \mathrm{Int}(\frac{d_k^1}{2}) + c_{2k+1}^0 + 2^q, & \text{if } \mathrm{Int}(\frac{d_k^1}{2}) + c_{2k+1}^0 < -2^{q-1}. \end{cases} \tag{2.7}$$

While the reconstruction algorithm (2.3) and (2.5) will be implemented by the computer itself as

$$c_{2k+1}^0 = \begin{cases} c_k^1 - \mathrm{Int}(\frac{d_k^1}{2}), & \text{if } -2^{q-1} \le c_k^1 - \mathrm{Int}(\frac{d_k^1}{2}) < 2^{q-1}, \\ 2^q + \left(c_k^1 - \mathrm{Int}(\frac{d_k^1}{2}) \right), & \text{if } c_k^1 - \mathrm{Int}(\frac{d_k^1}{2}) < -2^{q-1}, \\ \left(c_k^1 - \mathrm{Int}(\frac{d_k^1}{2}) \right) - 2^q, & \text{if } c_k^1 - \mathrm{Int}(\frac{d_k^1}{2}) > 2^{q-1}. \end{cases} \tag{2.8}$$

$$c_{2k}^0 = \begin{cases} d_k^1 + c_{2k+1}^0, & \text{if } -2^{q-1} \le d_k^1 + c_{2k+1}^0 < 2^{q-1}, \\ d_k^1 + c_{2k+1}^0 + 2^q, & \text{if } d_k^1 + c_{2k+1}^0 < -2^{q-1}, \\ d_k^1 + c_{2k+1}^0 - 2^q, & \text{if } d_k^1 + c_{2k+1}^0 \ge 2^{q-1}. \end{cases} \tag{2.9}$$

It is obvious that (2.8)-(2.9) are just the reverse of (2.6)-(2.7). It is also easy to see that if we properly take advantage of the bound in the coefficient size mentioned above, the algorithm can be implemented using a minimal amount of storage.

3. More Examples and Additional Analysis

In this section we are going to give more examples which will give some motivation for our new approach.

Example 3: A (2,6)-wavelet transform by integer calculation [2].

This transformation is similar to using following analysis filters

n	-2	-1	0	1	2	3
h_n	0	0	1/2	1/2	0	0
$\tilde{h}_n$	-1/16	-1/16	1/2	-1/2	1/16	1/16

(a) Decomposition

Decomposition starts with Example 1 at step (1) and (2), and then upgrades the high frequency component at step (3):

(1) Compute

$$d_k^{1,0} = c_{2k}^0 - c_{2k+1}^0, \quad k = 0, \cdots, M_1 - 1.$$

(2) Compute

$$c_k^1 = \mathrm{Int}(\frac{d_k^{1,0}}{2}) + c_{2k+1}^0, \quad k = 0, \cdots, N_1 - 2,$$

$$c_{N_1-1}^1 = \begin{cases} \mathrm{Int}(\frac{d_{M_1-1}^{1,0}}{2}) + c_{N-1}^0, & \text{if } N \text{ is an even number,} \\ c_{N-1}^0, & \text{if } N \text{ is an odd number.} \end{cases}$$

(3) Compute

$$\begin{cases} d_0^1 = \mathrm{Int}(\frac{c_0^1 - c_1^1}{4}) - d_0^{1,0} \\ d_k^1 = \mathrm{Int}(\frac{c_{k-1}^1 - c_{k+1}^1}{4}) - d_k^{1,0}, \quad k = 1, \cdots, M_1 - 2. \end{cases}$$

and then, if N is even, calculate

$$d_{M_1-1}^1 = \mathrm{Int}(\frac{c_{N_1-2}^1 - c_{N_1-1}^1}{4}) - d_{M_1-1}^{1,0},$$

else, calculate

$$d_{M_1-1}^1 = \mathrm{Int}(\frac{c_{N_1-3}^1 - c_{N_1-1}^1}{4}) - d_{M_1-1}^{1,0}.$$

(b) Reconstruction

The reconstruction algorithm is identical to the decomposition algorithm, except it is now running "backwards".

(1) Compute

$$\begin{cases} d_0^{1,0} = \mathrm{Int}(\frac{c_0^1 - c_1^1}{4}) - d_0^1 \\ d_k^{1,0} = \mathrm{Int}(\frac{c_{k-1}^1 - c_{k+1}^1}{4}) - d_k^1, \quad k = 1, \cdots, M_1 - 2, \end{cases}$$

and then, if N is even, calculate

$$d_{M_1-1}^{1,0} = \mathrm{Int}(\frac{c_{N_1-2}^1 - c_{N_1-1}^1}{4}) - d_{M_1-1}^{1,1},$$

else, calculate

$$d_{M_1-1}^{1,0} = \mathrm{Int}(\frac{c_{N_1-3}^1 - c_{N_1-1}^1}{4}) - d_{M_1-1}^{1,1}.$$

(2) If N is an even number, compute

$$c_{2k+1}^0 = c_k^1 - \mathrm{Int}(\frac{d_k^{1,0}}{2}), \quad k = 0, \cdots, N_1 - 1 \ ;$$

or, if N is an odd number, we have

$$c^0_{2k+1} = c^1_k - \mathrm{Int}(\frac{d^{1,0}_k}{2}), \quad k = 0, \cdots, N_1 - 2,$$

$$c^0_{N-1} = c^1_{N_1}.$$

(3) Compute

$$c^0_{2k} = d^{1,0}_k + c^0_{2k+1}, \quad k = 0, \cdots, M_1 - 1.$$

We see in step (2)-(3) above, that they are just the same as shown for the reconstruction of the (2,2)-wavelet transform (Example 1).

Example 4: A (1,3)-wavelet transform by integer calculation.
The following nonlinear transform is a variation of the transform which uses biorthogonal analysis filters:

n	-1	0	1
$\bar{h}_n$	1	0	0
$\bar{g}_n$	1/4	-1/2	1/4

(a) Decomposition

This decomposition starts with the *Lazy wavelet* at step (1) and upgrades the high frequency component at step (2):

(1) Set

$$c^1_k = c^0_{2k}, \ k = 0, \cdots, N_1 - 1;$$

$$d^1_k = c^0_{2k+1}, \ k = 0, \cdots, M_1 - 1.$$

(2) If N is an even number, calculate

$$\begin{cases} d^1_k = \mathrm{Int}(\frac{c^1_k + c^1_{k+1}}{2}) - d^{1,0}_k, & k = 0, \cdots, M_1 - 2, \\ d^1_{M_1-1} = c^1_{N_1-1} - d^{1,0}_{M_1-1}. \end{cases}$$

Otherwise, if N is an odd number, calculate

$$d^1_k = \mathrm{Int}(\frac{c^0_{2k} + c^0_{2k+2}}{2}) - c^0_{2k+1}, \quad k = 0, \cdots, M_1 - 1.$$

(b) Reconstruction

(1) Set

$$c^0_{2k} = c^1_k, \quad k = 0, \cdots, N_1 - 1;$$

(2) If N is an even number, calculate

$$\begin{cases} c^0_{2k+1} = \mathrm{Int}(\frac{c^0_{2k} + c^0_{2k+2}}{2}) - d^1_k, & k = 0, \cdots, M_1 - 2, \\ c^0_{N-1} = c^0_{N-2} - d^1_{M_1-1}. \end{cases}$$

IPR2023-00330 Page 00896

Otherwise, if N is an odd number, calculate

$$c^0_{2k+1} = \mathrm{Int}(\frac{c^0_{2k} + c^0_{2k+2}}{2}) - d^1_k, \quad k = 0, \cdots, M_1 - 1.$$

Example 5: A (5,3)-wavelet transform by integer calculation.

This transformation is also similar in function to using the biorthogonal analysis filters. It is given by

n	-2	-1	0	1	2
$\tilde{h}_n$	-1/8	1/4	3/4	1/4	-1/8
$\tilde{g}_n$	1/4	-1/2	1/4	0	0

(a) Decomposition

This decomposition starts with Example 3 at step (1) and upgrade low frequency components at step (2):

(1) Set $c^{1,0}_k = c^0_{2k}, \quad k=0, \cdots, N_1 - 1;$

If N is an even number, calculate

$$\begin{cases} d^1_k = \mathrm{Int}(\frac{c^0_{2k} + c^0_{2k+2}}{2}) - c^0_{2k+1}, & k = 0, \cdots, M_1 - 2, \\ d^1_{M_1-1} = c^0_{N-2} - c^0_{N-1}. \end{cases}$$

Otherwise, if N is an odd number, calculate

$$d^1_k = \mathrm{Int}(\frac{c^0_{2k} + c^0_{2k+2}}{2}) - c^0_{2k+1}, \quad k = 0, \cdots, M_1 - 1.$$

(2) If N is an even number, compute

$$\begin{cases} c^1_0 = c^{1,0}_0 - \mathrm{Int}(\frac{d^1_0}{2}), \\ c^1_k = c^{1,0}_k - \mathrm{Int}(\frac{d^1_{k-1} + d^1_k}{4}), & k = 1, \cdots, N_1 - 2, \\ c^1_{N_1-1} = c^{1,0}_{N_1-2} - \mathrm{Int}(\frac{d^1_{N_1-2} + d^1_{N_1-1}}{4}). \end{cases}$$

Otherwise, if N is an odd number, calculate

IPR2023-00330 Page 00897

$$\begin{cases} c_0^1 = c_0^{1,0} - \mathrm{Int}(\frac{d_0^1}{2}), \\ c_k^1 = c_k^{1,0} - \mathrm{Int}(\frac{d_{k-1}^1 + d_k^1}{4}), \quad k = 1, \cdots, N_1 - 2, \\ c_{N_1-1}^1 = c_{N_1-1}^{1,0} - \mathrm{Int}(\frac{d_{M_1-1}^1}{2}). \end{cases}$$

(b) Reconstruction

(1) Compute

$$c_0^0 = c_0^1 + \mathrm{Int}(\frac{d_0^1}{2}),$$

$$c_{2k}^0 = c_k^1 + \mathrm{Int}(\frac{d_{k-1}^1 + d_k^1}{4}), \quad k = 1, \cdots, N_1 - 2.$$

Then, if N is even, calculate

$$c_{N-2}^0 = c_{N_1-1}^1 + \mathrm{Int}(\frac{d_{N_1-2}^1 + d_{N_1-1}^1}{4}).$$

else calculate $$c_{N-1}^0 = c_{N_1-1}^1 + \mathrm{Int}(\frac{d_{M_1-1}^1}{2}).$$

(2) Compute $$c_{2k+1}^0 = \mathrm{Int}(\frac{c_{2k}^0 + c_{2k+2}^0}{2}) - d_k^1, \quad k = 0, \cdots, M_1 - 2.$$

Then, if N is even, calculate $$c_{N-1}^0 = c_{N-2}^0 - d_{M_1-1}^1.$$

The *PPP* property for Example 1-2 mentioned at the end of the previous section is also applicable for these three examples. It is obvious these three transformations are not really linear, but they are similar to the one using the corresponding filters given above. Especially, the filters in Example 3 and Example 5 belong to, with minor modification, the group of the best biorthogonal filters for image compression according to both our experience and the conclusion of [4].

Also, from the above three examples, we can note that if we begin with integer (linear or nonlinear) wavelet transformations and then use some proper upgrading formulas, we can get other, much better integer, wavelet transformations for image compression. Now, the key problem is: What kind of deductive formulas should be used? We provide an answer to this question in the following two sections, Section 4 and Section 5.

4. Lifting Scheme and Integer Biorthogonal Filtering

The *Lifting scheme*, discovered by Sweldens [1], is a new approach for constructing biorthogonal wavelets with compact support. However, the most interesting part of this method

for us is: it can be used, with minor modification, to create integer biorthogonal wavelet transformations. The following is an adaptation of the technique of [1].

Definition 1. The set of filters $\{h, \tilde{h}, g, \tilde{g}\}$ is a set of *biorthogonal filters* if the following formula is satisfied:

$$\forall \omega \in \mathbf{R}: \ \tilde{m}(\omega)\overline{m^t(\omega)} = 1.$$

where
$$m(\omega) = \begin{bmatrix} h(\omega) & h(\omega+\pi) \\ g(\omega) & g(\omega+\pi) \end{bmatrix},$$

and
$$h(\omega) = \sum_k h_k e^{-ik\omega} \text{ and } g(\omega) = \sum_k g_k e^{-ik\omega},$$

and similarly for
$$\tilde{m}(\omega), \ \tilde{h}(\omega) \text{ and } \tilde{g}(\omega).$$

The following lemma is the main result of the *lifting scheme* [1] reported as corollary 6 in that paper.

Lemma 1. Take an initial set of finite biorthogonal filters $\{h, \tilde{h}^0, g^0, \tilde{g}\}$, then a new set of finite biorthogonal filters $\{h, \tilde{h}, g, \tilde{g}\}$ can be found as

$$\begin{aligned} \tilde{h}(\omega) &= \tilde{h}^0(\omega) + \tilde{g}(\omega)\overline{s(2\omega)} \\ g(\omega) &= g^0(\omega) - h(\omega)s(2\omega). \end{aligned} \tag{4.1a}$$

Similarly, if we take $\{h^0, \tilde{h}, g, \tilde{g}^0\}$ as an initial set of biorthogonal filters, a new set of finite biorthogonal filters $\{h, \tilde{h}, g, \tilde{g}\}$ can be found as

$$\begin{aligned} h(\omega) &= h^0(\omega) + g(\omega)\overline{\tilde{s}(2\omega)} \\ \tilde{g}(\omega) &= \tilde{g}^0(\omega) - \tilde{h}(\omega)\tilde{s}(2\omega). \end{aligned} \tag{4.1b}$$

Here, $s(\omega)$ is a trigonometric polynomial and the corresponding filter s is finite, and so is $\tilde{s}(\omega)$. Actually, regarding the filters, (4.1) is equivalent to

$$\begin{aligned} \tilde{h}_k &= \tilde{h}_k^0 + \sum_l \tilde{g}_{k+2l} s_l \\ g_k &= g_k^0 - \sum_l h_{k-2l} s_l. \end{aligned} \tag{4.2a}$$

or

$$\begin{aligned} h_k &= h_k^0 + \sum_l g_{k+2l} \tilde{s}_l \\ \tilde{g}_k &= \tilde{g}_k^0 - \sum_l \tilde{h}_{k-2l} \tilde{s}_l. \end{aligned} \tag{4.2b}$$

Next, we use the lifting scheme with minor modification to create an integer, nonlinear, quasi-biorthogonal, wavelet algorithm. Suppose $\{c_n^0\}$ is a original signal, $\{c_n^1\}$ and $\{d_n^1\}$ are again its low and high frequency decomposition parts, obtained by using the filters $\{h, \tilde{h}, g, \tilde{g}\}$.

IPR2023-00330 Page 00899

If we use filters $\{\tilde{h}, \tilde{g}\}$ for decomposition (analysis), the corresponding decomposition algorithm is

$$\begin{cases} c_k^1 = \alpha_c \sum_n c_n^0 \bar{h}_{n-2k}, \\ d_k^1 = \alpha_d \sum_n c_n^0 \bar{g}_{n-2k}. \end{cases}$$

While the reconstruction algorithm will be

$$c_n^0 = 2\sum_k \left(\frac{c_k^1 h_{n-2k}}{\alpha_c} + \frac{d_k^1 g_{n-2k}}{\alpha_d} \right),$$

related to the synthesis filter $\{h, g\}$. Here, parameters α_c and α_d are positive constants with $\alpha_c \bullet \alpha_d = 2$. For example, in the situation of regular biorthogonal decomposition and reconstruction, $\alpha_c = \alpha_d = \sqrt{2}$; and for Example 1 through Example 5 above, $\alpha_c = 1$ and $\alpha_d = 2$.

If the set of filters $\{h, \tilde{h}, g, \tilde{g}\}$ is from $\{h, \tilde{h}^0, g^0, \tilde{g}\}$ by (4.2b), then the decomposition can be accomplished as follows:

1. Calculate

$$\begin{cases} c_k^{1,0} = \alpha_c \sum_n c_n^0 \bar{h}_{n-2k}^0, \\ d_k^1 = \alpha_d \sum_n c_n^0 \bar{g}_{n-2k}. \end{cases} \tag{4.3}$$

2. Calculate

$$c_k^1 = c_k^{1,0} + \frac{\alpha_c}{\alpha_d} \sum_l d_{k-l}^1 s_l. \tag{4.4}$$

The relative reconstruction scheme will be:

1. Calculate

$$c_k^{1,0} = c_k^1 - \frac{\alpha_c}{\alpha_d} \sum_l d_{k-l}^1 s_l, \tag{4.5}$$

2. Calculate

$$c_n^0 = 2\sum_k \left(\frac{c_k^{1,0} h_{n-2k}}{\alpha_c} + \frac{d_k^1 g_{n-2k}^0}{\alpha_d} \right). \tag{4.6}$$

Here, equations (4.3) and (4.6) are just the wavelet (inverse) transforms using biorthogonal filters $\{h, \tilde{h}^0, g^0, \tilde{g}\}$. While (4.4) and (4.5) are forward and backward upgrading formulas.

Similarly, if the set of filters $\{h, \tilde{h}, g, \tilde{g}\}$ is from the initial set of filters $\{h^0, \tilde{h}, g, \tilde{g}^0\}$ by using (4.2b), the relative decomposition is:

1. Calculate

$$\begin{cases} c_k^1 = \alpha_c \sum_n c_n^0 \bar{h}_{n-2k}, \\ d_k^{1,0} = \alpha_d \sum_n c_n^0 \bar{g}_{n-2k}^0. \end{cases}$$

2. Calculate $$d_k^1 = d_k^{1,0} - \frac{\alpha_d}{\alpha_c}\sum_l c_{k-l}^1 s_l .$$

The reconstruction scheme is:

1. Calculate $$d_k^{1,0} = d_k^1 + \frac{\alpha_d}{\alpha_c}\sum_l c_{k-l}^1 s_l$$

2. Calculate $$c_n^0 = 2\sum_k \left(\frac{c_k^1 h_{n-2k}^0}{\alpha_c} + \frac{d_k^1 g_{n-2k}}{\alpha_d} \right)$$

For the sake of clarity, we haven't considered the boundary situation, but we will address this later.

Corollary 4.1. Suppose biorthogonal filters $\{h, \bar{h}, g, \tilde{g}\}$ are from initial filters $\{h, \bar{h}^0, g^0, \tilde{g}\}$ by the lifting scheme (4.1a) or (4.2a). If the decomposition and reconstruction by filters $\{h, \bar{h}^0, g^0, \tilde{g}\}$ can be accomplished only by integer calculation, such as Example 2, we also can create a corresponding integer wavelet decomposition and reconstruction scheme which is very "close" to the original one by using filters $\{h, \bar{h}, g, \tilde{g}\}$. Here the word "close" means that the difference of the two decomposition schemes is just some rounding error, and this rounding error will be corrected by the integer reconstruction scheme.

In fact, if $\{c_k^{1,0}\}$ and $\{d_k^1\}$ are integer after (4.3), we can calculate $\{c_k^1\}$ by

$$c_k^1 = c_k^{1,0} + \mathrm{Int}\left(\frac{\alpha_c}{\alpha_d}\sum_l d_{k-l}^1 s_l \right). \tag{4.7}$$

instead of (4.4). Here $\mathrm{Int}(x)$, as described in Section 2, is an arbitrary rounding up function which satisfies $x - 1 \le \mathrm{Int}(x) \le x + 1$. It is obvious that (4.7) is very "close" to (4.4), and the exact reconstruction scheme can easily be obtained from

$$c_k^{1,0} = c_k^1 - \mathrm{Int}\left(\frac{\alpha_c}{\alpha_d}\sum_l d_{k-l}^1 s_l \right) \tag{4.8}$$

and (4.6). There will be a similar result, if the set of biorthogonal filters $\{h, \bar{h}, g, \tilde{g}\}$ is obtained from the initial set of filters $\{h^0, \bar{h}, g, \tilde{g}^0\}$ by using (4.2b).

We can now note, except for the example shown in the Lazy *wavelet*, (Example 2) most standard biorthogonal wavelet transforms cannot be performed directly by integer, even for one of the simplest wavelets, the *Haar wavelet*. However, if we properly choose the parameters α_c and α_d, and slightly change the transform algorithms, such as Example 1 and Example 3, we can have a variation of the original biorthogonal wavelet transforms with respect to the set of filters

IPR2023-00330 Page 00901

$\{h, \tilde{h}^0, g^0, \tilde{g}\}$ (or $\{h^0, \tilde{h}, g, \tilde{g}^0\}$). On the other hand, the parameters $\{s_i\}$ should be also chosen carefully to guarantee that only addition and shift operations are needed by the algorithm.

Another observation: if the set of filters $\{h, \tilde{h}, g, \tilde{g}\}$ is obtained from a set of filters $\{h^0, \tilde{h}, g, \tilde{g}^0\}$ by the lifting scheme, and the set $\{h^0, \tilde{h}, g, \tilde{g}^0\}$ is also obtained from a filter set $\{h^0, \tilde{h}^0, g^0, \tilde{g}^0\}$, we can repeatedly use Corollary 1 to get a "close" integer wavelet transformation.

5. The Correction Method for Creating Integer Wavelet Transforms

In this section, we will describe another approach for obtaining integer wavelets by using the so called *Correction method*. The motivation of this method is from the S+P transform, and we will now generalize this approach. Actually, the lifting scheme for generating biorthogonal wavelets can be considered as a special case of the correction method. From this method we can get some even complicated filters with fast decomposition and reconstruction algorithm.

Suppose that we already have a simple integer wavelet transform, such as Examples 1 through 3, the decomposition and reconstruction scheme of which can be formulated as follows:

Decomposition

$$c_i^{1,0} = df_c\left(\{c_n^0\}\right)$$
$$d_i^{1,0} = df_d\left(\{c_n^0\}\right) \tag{5.1}$$

Reconstruction

$$c_n^0 = rf\left(\{c_i^{1,0}\}, \{d_i^{1,0}\}\right) \tag{5.2}$$

Here, (5.1) and (5.2) can be the same as (4.3) and (4.6) or other algorithms.

In general, after the above decomposition, one may not be satisfied with the result. There may still be some correlation among the highpass components because of the aliasing from the lowpass components, or the lowpass components do not carry enough of the expected information from the original signal. Hence, we could make an improvement by putting some correction part on the highpass components or lowpass components. There are many ways to accomplish this. However, for the sake of the integer calculation, we prefer to use following correction method. For example, if we want to make a correction for the highpass part, the corresponding formula would be:

$$d_i^1 = d_i^{1,0} - \mathrm{Int}\left(dc_i^1\right) \quad k = \cdots, 0, 1, 2, \cdots. \tag{5.3}$$

Here, dc_i^1 is a correction quantity for d_i^1

$$dc_i^1 = \sum_{\mu=s_1}^{s_2} \sigma_\mu c_{i+\mu}^{1,0} + \sum_{\mu=1}^{\tau} \tau_\mu d_{i+\mu}^{1,0}, \quad k = \cdots, 0, 1, 2, \cdots \tag{5.4}$$

IPR2023-00330 Page 00902

and, $\{\sigma_i\}_{i=-S_0}^{S_1}$ and $\{\tau_j\}_{j=1}^{T}$ are given parameters which have been chosen for the user's purpose, such as reducing the redundancy among highpass components or some other special requirement. We are not going to discuss how to choose these parameters, but one can refer to the references [3, 5, 6] for clarification of this process. The only thing we need to mention is, for the sake of the integer calculation, any entries in both $\{\sigma_i\}_{i=-S_0}^{S_1}$ and $\{\tau_j\}_{j=1}^{T}$ should be rational numbers with denominators being powers of 2.

From (5.1), (5.3) and (5.4), it is easy to see the perfect reconstruction algorithm can be

$$d_k^{1,0} = d_k^1 + \mathrm{Int}(dc_k), \; k = \cdots, \; m, \; m-1, \; m-2 \cdots, \tag{5.5}$$

combined with (5.2).

As mentioned above, the Lifting scheme is a special condition of the Correction method. Examples 3 through 5 can also be considered as the examples of this method. We next give an example of the Correction method which cannot be included in the group of Lifting scheme, and also which does not result in a closed form of compact support for biorthogonal filters.

Example 6 S+P transform [3], which is similar to using following analysis filters

n	-2	-1	0	1	2	3
h_n	0	0	1/2	1/2	0	0
$\tilde{h}_n$	-1/16	-1/16	15/32	-17/32	7/32	-1/32

While, the synthesis filters do not have compact support. However, the S+P transform can be implemented as follows:

(a) Decomposition

(1) Take the decomposition step of Example 1, that is, compute

$$d_k^{1,0} = c_{2k}^0 - c_{2k+1}^0, \quad k = 0, \; 1, \cdots, M_1 - 1;$$

and

$$c_k^1 = \mathrm{Int}\left(\frac{d_k^{1,0}}{2}\right) + c_{2k+1}^0, \quad k = 0, \cdots, N_1 - 2,$$

$$c_{N_1-1}^1 = \begin{cases} \mathrm{Int}\left(\frac{d_{M_1-1}^{1,0}}{2}\right) + c_{N-1}^0, & \text{if } N \text{ is an even number,} \\ c_{N-1}^0, & \text{if } N \text{ is an odd number.} \end{cases}$$

(2) Correction Step: Define $S_0 = -1$, $S_1 = 1$, $T = 1$ and

$$\sigma_{-1} = -\tfrac{1}{16}, \; \sigma_0 = -\tfrac{1}{16}, \; \sigma_1 = \tfrac{1}{16};$$

$$\tau_1 = \tfrac{1}{2}.$$

and now compute

IPR2023-00330 Page 00903

In fact, the quantity b_k would have the same value in both (4.7) and (4.8) if we calculate it in the same way. On the other hand, if the working unit for b_k is q bits, the machine will give b_k another value, say $\bar{b}_k$ ($-2^{q-1} \le \bar{b}_k < 2^{q-1}$), where $\bar{b}_k$ is not equal to b_k in the sense of mathematics if the value of b_k is beyond the interval $[-2^{q-1}, 2^{q-1}-1]$. However, $\bar{b}_k$ will be the same in both (4.7) and (4.8). Therefore, the machine will automatically implement (7.1) and (7.2) as

$$c_k^1 = \begin{cases} c_k^{1,0} + \bar{b}_k, & \text{if } -2^{q-1} \le c_k^{1,0} + \bar{b}_k < 2^{q-1}, \\ c_k^{1,0} + \bar{b}_k - 2^q, & \text{if } c_k^{1,0} + \bar{b}_k \ge 2^{q-1}, \\ 2^q + c_k^{1,0} + \bar{b}_k & \text{if } c_k^{1,0} + \bar{b}_k < -2^{q-1}. \end{cases} \tag{7.1m}$$

and

$$c_k^{1,0} = \begin{cases} c_k^1 - \bar{b}_k, & \text{if } -2^{q-1} \le c_k^1 - \bar{b}_k < 2^{q-1}, \\ 2^q + c_k^1 - \bar{b}_k, & \text{if } c_k^1 - \bar{b}_k < -2^{q-1}, \\ c_k^1 - \bar{b}_k - 2^q, & \text{if } c_k^1 - \bar{b}_k \ge 2^{q-1}. \end{cases} \tag{7.2m}$$

It is easy to see that (7.2m) is just the backward operation of (7.1m), which provides the evidence that the conclusion of this lemma is correct.

It should be mentioned that the coefficients $\{c_k^1\}$ obtained by (4.3) and (7.1m) might not be the "real" wavelet coefficients using common sense. However, if we still use the working unit with q bits precision at the reconstruction step, (7.2m) and (4.6) will give the exact original signal back. On the other hand, the coefficients $\{c_k^1\}$ still keep the most continuity of the "real" wavelet coefficients. Therefore, when we repeat the decomposition step on $\{c_k^1\}$, most small coefficients in its high frequency part $\{d_k^2\}$ will be almost the same as the "real" coefficients (within some rounding error), which allows us to still take advantage of the "real" wavelet transform in image compression.

A similar argument can show the same *PPP* property will hold for the integer wavelet transforms generated by the Correction method in Section 5.

As we mentioned before, for many applications, the lossless image compression is as important as lossy compression. The integer wavelet transforms give the opportunity to compress without loss. It is also obvious that the integer wavelet algorithms can be used wherever ordinary wavelets are used, especially in signal and image compression. However, for most computers, the integer wavelet transform is much faster than the ordinary one and it uses much less memory. The following are some applications illustrating these types of transforms.

IPR2023-00330 Page 00904

$$\begin{cases} d_0^1 = d_0^{1,0} - \mathrm{Int}(\frac{c_0^1 - c_1^1}{4}); \\ d_k^1 = d_k^{1,0} - \mathrm{Int}(\frac{2c_{k-1}^1 + c_k^1 - 3c_{k+1}^1 - 2d_{k+1}^{1,0}}{8}), \quad k = 1, \cdots, M_1 - 2; \\ d_{M_1-1}^1 = d_{M_1-1}^{1,0} - \mathrm{Int}(\frac{c_{M_1-2}^1 - c_{M_1-1}^1}{4}). \end{cases}$$

(b) Reconstruction

(1) Compute

$$\begin{cases} d_{M_1-1}^{1,0} = d_{M_1-1}^1 + \mathrm{Int}(\frac{c_{M_1-2}^1 - c_{M_1-1}^1}{4}); \\ d_k^{1,0} = d_k^1 + \mathrm{Int}(\frac{2c_{k-1}^1 + c_k^1 - c_{k+1}^1 - 2d_{k+1}^{1,0}}{8}), k = M_1 - 2, \cdots, 1; \\ d_0^{1,0} = d_0^1 + \mathrm{Int}(\frac{c_0^1 - c_1^1}{4}). \end{cases}$$

(2) If N is an even number, compute

$$c_{2k+1}^0 = c_k^1 - \mathrm{Int}(\frac{d_k^1}{2}), \quad k = 0, \cdots, N_1 - 1$$

or, if N is an odd number, we have

$$c_{2k+1}^0 = c_k^1 - \mathrm{Int}(\frac{d_k^1}{2}), \quad k = 0, \cdots, N_1 - 2,$$

$$c_{N-1}^0 = c_{N_1}^1.$$

(3) Compute

$$c_{2k}^0 = d_k^1 + c_{2k+1}^0, \quad k = 0, \cdots, M_1 - 1.$$

6. Boundary Conditions

In the previous two sections, we did not show how to get the integer, wavelet transform at the boundaries of signals. However, for all of the examples given above, the boundaries have been considered. There are two issues dealing with boundary filtering if we use the *Lifting scheme* or the *Correction method* to generate the integer wavelet transformations. The first is how to process the boundaries which occur in the start-up wavelet transformations. The second is how to deal with the boundaries in the deductive formula. If the boundaries in the start-up wavelet transform have already been established, then those in the upgrading formula are easy to establish. In fact, for the *Lifting scheme*, the boundaries in both steps should be processed in the same way. While, for the Correction *method*, it is easy to see from (5.3)-(5.4) that one has more choices to process boundaries in the second step. Therefore, the only thing we need to discuss here is the process by which the boundaries in the start up wavelet transformations are established. Assume we begin with compact supported biorthogonal wavelets.

Suppose the original signal is $\{c_n^0\}_{n=0}^N$. For creating integer biorthogonal wavelet transformations we can use the following symmetric extension [7]:

(1). If current biorthogonal filters have even length, we extend the boundaries of the signal as $c_{-k}^0 = c_{k-1}^0$, $k = 1,2,\cdots$;

(2). If the filters have odd length, we do the extension as $c_{-k}^0 = c_k^0$, $k = 1,2,\cdots$

Example 1 through 5 use the boundaries give above. In Example 6, the start up wavelet transform uses the above boundaries, but in the upgrading step, another boundary filtering is used. In addition, for arbitrarily sized images or signals, one can use the same technique which we described in the above examples to deal with this condition.

7. Some Applications

Before talking about any applications of the integer wavelet transform given above, we first prove that a nice *property of precision preservation (PPP)*, which is similar to the one mentioned in Section 2, holds for both the Lifting and Correction upgrading technique. This property is very important for many applications.

Lemma 7.1 Suppose that our integer wavelet transform starts with a pair of biorthogonal filters with the *PPP* property discussed in Section 2, that is, (4.3) and (4.6) possess this property. Then, the same property will be preserved in the whole algorithm if we adopt the Lifting scheme to be the upgrading formula.

In other words, Lemma 7.1 states if we only use the working units with the same precision as the original signal or image to calculate the wavelet transform developed in Section 4, the equations (4.8) and (4.6) are still the backward operations of the equations (4.3) and (4.7).

Proof. Assume that we only use q bits to represent images or signals, say, the range of the pixel values is within $[-2^{q-1}, 2^{q-1}-1]$. According to the hypothesis of the lemma, the equations (4.3) and its inverse (4.6) have the *PPP* property. Therefore, what we have to verify here is that the equation (4.7) and its inverse (4.8) can preserve the same property. We rewrite (4.7) and (4.8) as follow:

$$c_k^1 = c_k^{1,0} + b_k, \tag{7.1}$$

and its inverse

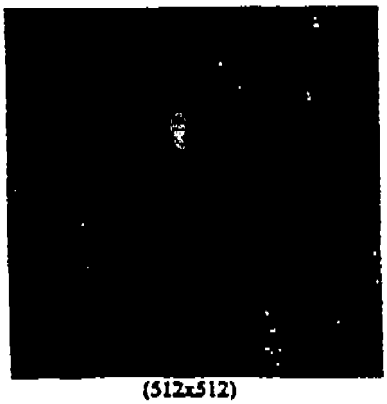
$$c_k^{1,0} = c_k^1 - b_k. \tag{7.2}$$

Here,

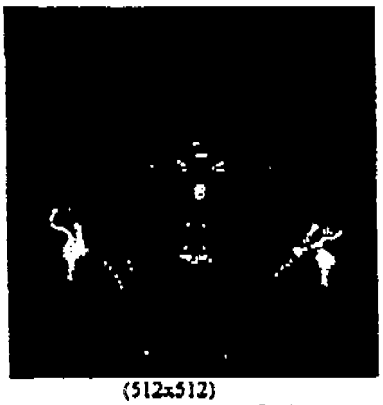
$$b_k = \mathrm{Int}\left(\frac{\alpha_r}{\alpha_d}\sum_i d_{k-i}^1 s_i\right).$$

Application 1. Lossless image compression

As mentioned at the beginning of this paper, the integer wavelet transformation established by the techniques described in this paper can always be used for lossless image compression because of the reversible ability. Especially, we can use the *PPP* property discussed in Lemma 7.1. We have used this wavelet lossless technology (WLT) for gray scale lossless image compression, and we have tried several images. For most natural images, the size of wavelet lossless compressed images is much smaller than corresponding GIF images. Figure 1 through 4 give some examples. Figure 1 is a standard image for compression, Figure 2 and 4 are X-ray images and Figure 3 is a two-value image but we treat it as a 8 bit gray scale image in order to compare with the GIF format. In fact, if we convert Figure 2 to a binary image, better result can be obtained by the JBIG technique.



(512x512)
Figure 1. Compression Ratio
WLT: 1.9:1/GIF: 1.05:1



(512x512)
Figure 2. Compression Ratio:
WLT 4.5:1/GIF: 2.72:1

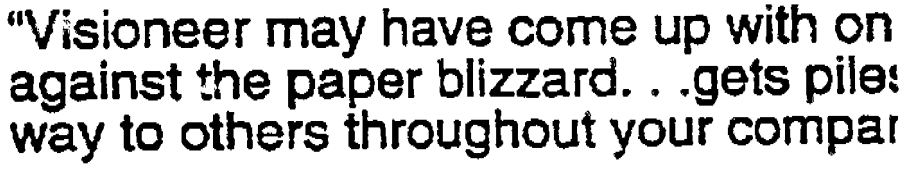


Figure 3. Compression Ratio: WLT: 20.8/ GIF 17.8 (152x794)

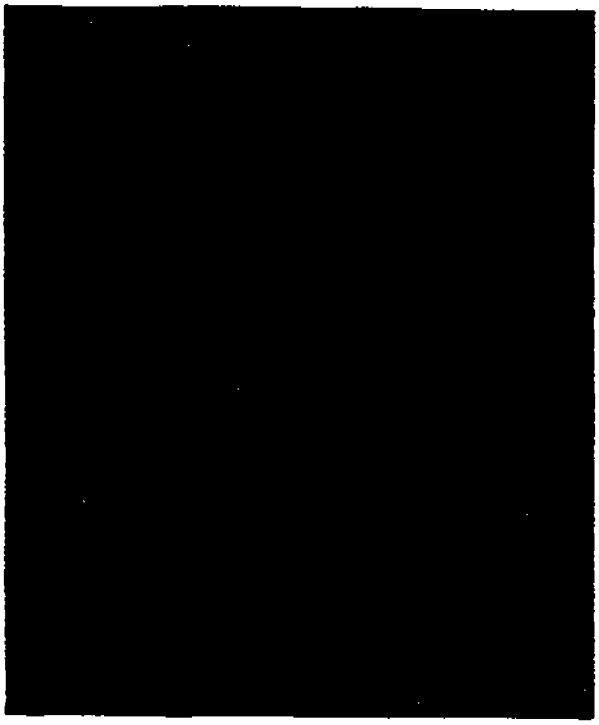


Figure 4. Compression Ratio: WLT 3.8:1/GIF 1.98:1 (1232x1024)

Application 2. Large scale medical image compression

Usually, 12 bits are used to represent one pixel in medical images. In this situation, the values of the pixels vary from 0 to 4095. Such images require careful treatment when a transform coding method is used for compression. If we use ordinary biorthogonal wavelets, the range of the transform coefficients will expand to $[-2^{16}, 2^{16}]$ when five levels of transform are used. Therefore, a longer working unit has to be employed, which consumes significant computer resources. However, the integer wavelet technique developed in this paper will solve this problem. For example, if we use the transforms given in Example 3, 5 and 6, the values of transform coefficients will be limited to the range of $[-2^{13}, 2^{13}]$. Even if we do not use the *PPP* property for these wavelets, *16* bits for the working unit is sufficient for all computations.

8. Conclusion

This paper has shown the processes necessary in order to obtain a non-linear, integer, biorthogonal, or non-biorthogonal reversible wavelet transform suitable for signal or image processing. We have shown how such a transform can be obtained either using the Lifting method, or the Correction method. For example, all interpolation wavelets can be modified to be corresponding integer wavelets without loosing any properties of original wavelets. In addition,

we have shown under certain conditions, the precision of the transform computation on the computer can remain at the same precision of the data, thus reducing the need for additional computer memory during the transform computation. These are extremely powerful techniques when the target data are large images, or the requirements establish a need for speed.

Although this paper establishes the structure for the integer transform based upon the biorthogonal wavelet or some non-biorthogonal wavelet, we do not imply the examples in this paper are necessarily the best wavelets for any particular application. However, we do claim if one is going to use such a technique, the ideas suggested in this paper will provide the best implementation.

References

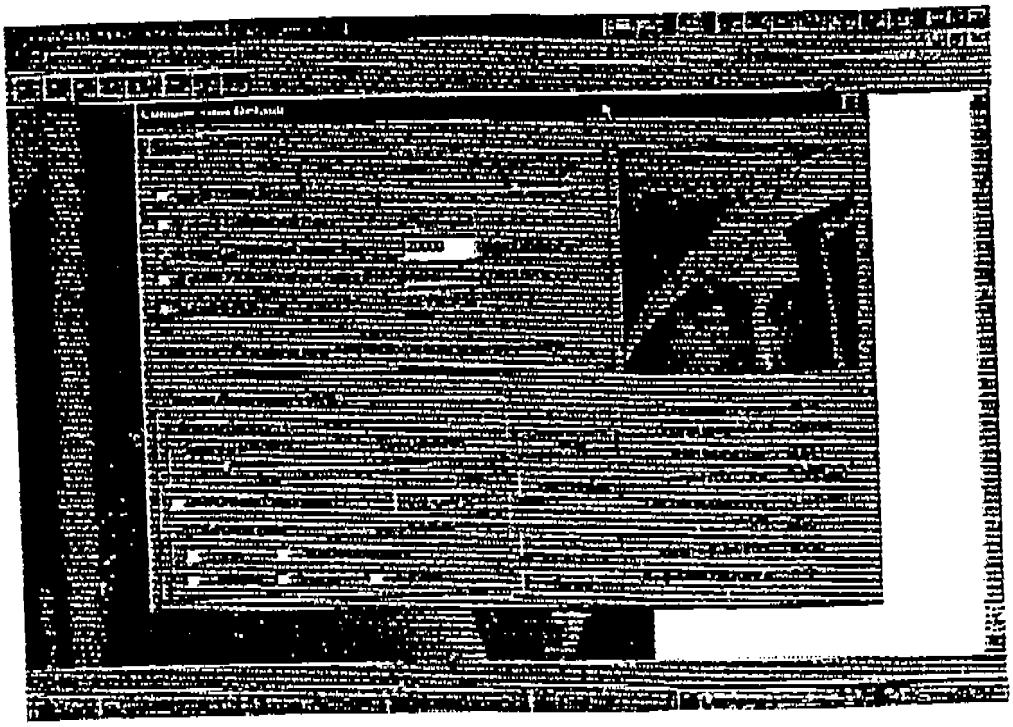
1. Wim Sweldens, *The lifting scheme: A custom-design construction of biorthogonal wavelets*, Applied and Computational Harmonic Analysis, Vol. 3, No. 2, April 1996.
2. A. Zandi, J. Allen, E. Schwartz and M. Boliek, *CREW: Compression with reversible embedded wavelets*, in IEEE Data Compression Conference, (Snowbird, Utah), pp.212-221, March 1995.
3. Amir Said, *An image multiresolution representation for lossless and lossy compression*, Submitted to the IEEE Transactions on Image Processing.
4. J. Villasenor, B. Belzer, and J. Liao, *Wavelet filter evaluation for image compression*, IEEE Trans. Image Processing, Vol. 4, pp.1053-1060.
5. G.R. Kuduvalli and R.M. Rangayyan, *Performance analysis of reversible image compression techniques for high-resolution digital teleradiology*, IEEE Trans. Med. Imaging, Vol. 11, pp. 430-445, Sept. 1992.
6. W.H. Press, B.P Flannery, S.A. Teukolsky, and W.T. Vetterling, *Numerical Recipes: the Art of Science Programming*, Cambridge University Press, Cambridge, New York, 1986.
7. G. Strang and Truong Nguyen, *Wavelets and filter banks*, Wellesley-Cambridge Press, 1996.

INFINITRON

Windows Image Compressor V3.0

Announcing, Lightning Strike™ Image Compressor (LSIC) version 3.0, a Windows 95 tool that compresses still images from 50:1 to 200:1 using INFINITRON's proprietary wavelet technology. LSIC is a versatile, easy to use tool for Web and Graphic designers that can handle a wide variety of digital image formats, and it includes filters and convenient web tools. Images can be compressed 3 to 5 times more than JPEG, while maintaining similar or better image fidelity. Images can be viewed in 2 to 4 seconds over the Internet rather than 10 to 20 seconds for images compressed under JPEG. This has enormous benefits for reducing bottlenecks on corporate networks and the web, and in addition, requires less storage space.

Lightning Strike



Lightning Strike Features

Compression. Images can be compressed as high as 200:1 using wavelet technology.

Compression Control. An EASY mode allows the user to compress images with minimal input, requiring only a decision between more quality or more compression. An ADVANCED mode enables the user to select; 1) image file size, 2) compression ratio, 3) PSNR, or 4) master level. Web designers will like the one step process to control the size of their image files, thus insuring the speed an image may be viewed on a browser.

Non Uniform Compression. Regions of an image can be selected for less compression to preserve a higher image quality while the rest of the image is compressed to the specified compression ratio. In this way important parts of a picture maintain crucial details while the over all picture file can be made as small as possible

Post Reconstruction Filters. Filters are available to enhance the reconstructed image. At compression time the user can preset a control to have these filters operate automatically during reconstruction. The filters include: quality improvement, sharpen (edge enhancement), smoothing, and brighten.

Transparencies. The user will have the ability to set pixels transparent so that a color in the background (already on the page) can be seen through the picture. This is useful for creative web site developers.

Progressive Compression. An image can be compressed so that when it is viewed it will appear quickly, first with low resolution, and then progressively building up in detail as it is downloaded. This insures the viewer does not lose interest while the image is downloaded.

INFINITRON

Windows Image Compressor V3.0

Lightning Strike

Cow, Bitmap, No Compression

Cow, Lightning Strike, 85:1 compression

Cow, JPEG 85:1 Compression

Applications Lightning Strike Image Compression

- Images on the Web
- Photo Stock
- Data Warehousing
- Catalogues
- Video Games
- CDs (Encyclopedias, Museums, Science, and Medicine)
- Archives (Art, History, Genealogy)
- Medical Imaging

Performance

Encode time typically less than 3 seconds for a 320 X 240 pixel, 24 bit color image on a 133 MHz. Pentium with 16 MB RAM.

Minimum Recommended System

Windows 95/NT OS
Pentium 100 MHz, 8 MB RAM
2 MB for program files
10 MB plus to swap image files

Ancillary INFINITRON Products

- Netscape Navigator Plug-in
- Java Applet
- ActiveX Control
- Web Site Image Converter
- Lightning Strike SDK
- GML Banner Generator

About INFINITRON, Inc.

Founded in 1992, INFINITRON, Inc. is a private company that specializes in the design and marketing of high quality image and video compression solutions for a wide array of markets. INFINITRON is based in Vancouver, BC with labs in Regina, Saskatchewan and Denton Texas.

Download a FREE demo version of Lightning Strike Windows Compressor from:
www.infinitron.com

INFINITRON USA Office 3401 East University, #104, Denton, TX, 76208
Tel: 817.484.1165 FAX: 817.484.0588

INFINITRON Canada Office 10th Flr 1199 W. Hastings, Vancouver, BC, V6E 3T5
Tel: 604.688.9789 FAX: 604.688.9798

IPR2023-00330 Page 00911

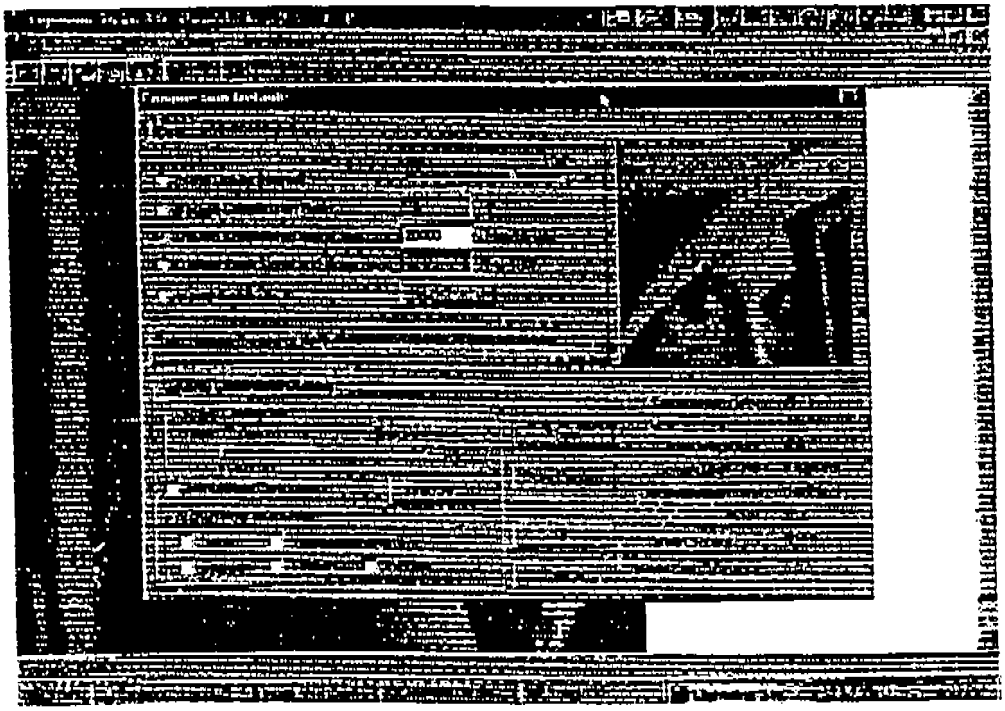
INFINITRON Research International, Inc., All Rights Reserved

INFINITRON

Windows Image Compressor V3.0

Lightning Strike

Announcing, Lightning Strike Image Compressor (LSIC) version 3.0, a Windows 95 tool that compresses still images from 50:1 to 200:1 using Infinitron's proprietary Wavelet technology. LSIC is a versatile, easy to use tool for Web and Graphic designers that can handle a wide variety of digital image formats and includes filters, and convenient web tools. Images can be compressed to files 4 times smaller than JPEG, while maintaining similar or better image fidelity. Images can be viewed in 2 to 5 seconds over the internet rather than 10 to 20 seconds for images compressed under JPEG. This has enormous benefits for reducing bottlenecks on corporate networks and the web, and in addition, requires less storage space.



Lightning Strike Features

Compression. Images can be compressed to over ~~100~~:1 using Wavelet Technology.

Compression Control. An EASY mode allows the user to compress images with minimal input, requiring only a decision between more quality or more compression. An ADVANCED mode enables the user to select; 1) image file size, 2) compression ratio, 3) PSNR, or 4) master level. Web designers will like the one step process to control the size of their image files, thus insuring the speed an image may be viewed on a browser.

Non Uniform Compression. Regions of an image can be selected for less compression to preserve a higher image quality while the rest of the image is compressed to the specified compression ratio. In this way important parts of a picture maintain crucial details while the over all picture file can be made as small as possible

Post Reconstruction Filters. Filters are available to enhance the reconstructed image. At compression time the user can preset a control to have these filters operate automatically during reconstruction. The filters include: quality improvement, sharpen (edge enhancement), smoothing, and brighten.

Transparencies. The user will have the ability to set pixels transparent so that a color in the background (already on the page) can be seen through the picture. This is useful for creative web site developers.

Progressive Compression. An image can be compressed so that when it is viewed it will appear quickly, first with low resolution, and then progressively building up in detail as it is downloaded. This insures the viewer does not loose interest while the image is downloaded.

INFINITRON

windows Image Compressor V3.0

Lightning Strike

Lenna, JPEG 115:1 Compression

Lenna .bmp, No Compression

Applications Lightning Strike Image Compression

- Images on the Web
- Photo Stock,
- Data Warehousing
- Catalogues.
- Video Games
- CDs (Encyclopedias, Museums, Science, and Medicine)
- Archives (Art, History, Genealogy)
- Medical Imaging

Performance

Encode time typically less than 3 seconds for a 320 X 240 pixel, 24 bit color image on a 133 MHz. Pentium with 16 MB RAM.

Minimum Recommended System

Windows 95/NT OS
Pentium 100 MHz, 8 MB RAM
2 MB for program files
10 MB plus to swap image files

Ancillary INFINITRON Products

- Netscape Navigator Plug-in
- Java Applet
- ActiveX Control
- Web Site Image Converter
- Lightning Strike SDK
- GMI Banner Generator

About INFINITRON, Inc.

Founded in 1992, INFINITRON, Inc. is a private company that specializes in the design and marketing of high quality image and video compression solutions for a wide array of markets. INFINITRON is based in Vancouver, BC with labs in Regina, Saskatchewan and Denton Texas.

Download a FREE demo version of Lightning Strike Windows Compressor from:
www.infinitron.com

INFINITRON USA Office 3401 East University, #104, Denton, TX. 76208
Tel: 817.484.1165 FAX: 817.484.0588

INFINITRON Canada Office 10th Flr 1199 W. Hastings, Vancouver, BC, V6E 3T5
Tel: 604.688.9789 FAX: 604.688.9798

INFINITRON
Lightning Strike

Product Fact Sheet
Windows Compressor

Compressor Version 3.0

The Lightning Strike Compressor is a Windows tool that compresses still images from a wide variety of digital image formats using Infinitron's proprietary Wavelet algorithm. Images can be compressed to files 5 times smaller than JPEG, while maintaining similar or better image fidelity. Images can be viewed in 1 or 2 seconds over the internet rather than 10 to 20 seconds for images compressed under JPEG. This has enormous benefits for transmitting over corporate networks or the web, and in addition, saves space required for storing all those images.

Lightning Strike is a collection of tools in a user friendly environment. Two levels of user control are offered, one quick and easy for most applications, the other a master level for the advanced user who wishes to control parameters to maximize image quality.

The compression approach used by Lightning Strike is based upon integer wavelets. This technology is acknowledged by leading experts as a superior compression technique as compared to discrete cosine transform used in JPEG.

Lightning Strike Windows Compressor Features

Image Compression Options and Control

Compression Technique Options

Both Infinitron's Wavelet Compression and other frequently used compression methods are included in the product so users need only have Lightning Strike on their work station to perform all image compressions. Images can be compressed to Wavelet, JPEG, PNG, and GIF.

Compression Quality Versus Speed Options

The user can select one of two encoding processes that trade quality for speed of compression and ease of use. With the "Advanced" option selected, the optimum compression parameters are set by the user to give the best possible images for selected compression ratio. With "Easy" selected you get the fastest compression without having to know details of parameter selection.

Compression Ratio Control

The compressed image file size or compression ratio may be specified rather than the quality factor. This enables a web designer to control the size of their image files or the speed an image may be viewed, in a one step process.

Region of Interest Focusing

Regions of an image can be selected for less compression to preserve a higher image quality while the rest of the image is compressed to the specified compression ratio. In this way important parts of a picture maintain crucial details while the over all picture file can be made as small as possible. This is also known as Non-Uniform Compression.

INFINITRON
Lightning Strike

Product Fact Sheet
Windows Compressor

Split and Merge

Very large images, which could not otherwise be compressed due to their large size, can be split into smaller images and compressed individually. This process has the side advantage of using RAM more effectively speeding time for compression. The split images can be reassembled using the merge aspect of the feature.

Post Reconstruction Filters

Filters are available to enhance the reconstructed image. At compression time the user can preset a control to have these filters operate automatically during reconstruction. The filters include; quality improvement, sharpen (edge enhancement), smoothing, and brighten.

Transparencies

The user will have the ability to set pixels transparent so that a color in the background (already on the page) can be seen through the picture. This is useful for creative web site developers. This gives the ability to display pictures other than the rectangular shape allotted on the web page, i.e. circles, polygons etc.. Also, designers often use this feature for shadowing, letters and objects.

Progressive Decompression

An image can be compressed so that when it is viewed it will appear quickly, first with low resolution, and then progressively building up in detail as it is downloaded. This insures the viewer does not loose interest while the image is downloaded.

INFINITRON
Lightning Strike

Product Fact Sheet
Windows Compressor

Image Comparisons

Picture of Lena, with no Compression (512 X 512 Image)

INFINITRON
Lightning Strike

Product Fact Sheet
Windows Compressor

Picture of Lena Compressed 100:1 with Lightning Strike

INFINITRON
Lightning Strike

Product Fact Sheet
Windows Compressor

File Functions

PNG File Structure
The compressed images are stored in file format compliant with the PNG standard. In the future this file format will replace the GIF format used today.

Batch Compress
The user is able to compress many images at once by adding or deleting image files (or paths) to a list box.

Image Statistics
The compressor stores image statistics on each compressed image which may be viewed by the user. The following information is provided; image dimensions, compression ratio, file sizes, MSE, PSNR, maximum pixel difference, compression and decompression times.

Performance and System Requirements

Encoding and Decode Time.
The typical time to encode or decode a 320X240, 24 bit color image is 1 second on a Pentium running at 133 MHz with 16 Meg RAM.

Minimum Recommended System
The minimum system requirements for an IBM PC Compatible are:

Hard Disc Drive	2 Mbytes free for program files, 10 Mb plus to swap image files.
Operating System	MS Windows 3.1(Win32)/ 95/ NT
RAM	8 Mbytes

This software is also available on the Apple MAC, Solaris, and UNIX platforms.

INFINITRON
Lightning Strike

Product Fact Sheet
Windows Compressor

Ancillary Infinitron Products

Netscape Navigator Plug-in
Netscape plug-ins are available for the Mac68k, Mac PPC, Widows 3.1 and 95/NT.

Java Applet
Java Applets are available for the Mac68k, Mac PPC, Widows 3.1 and 95/NT.

ActiveX Control
The Lightning Strike decompression software is available for such applications as Microsoft's Internet Explorer, as an ActiveX control.

Web Site Image Converter
This utility will automate the conversion of web pages from JPEG to the Lightning Strike format. The utility searches an HTML file and replicates it replacing any JPEG image tags with Lightning Strike tags and converting the JPEG image files to Lightning Strike. The utility can follow link tags to recursively convert and replicate an entire web site or sub-section of a web site to the Lightning Strike format. This utility will be available for Windows NT and most flavors of the UNIX operating system.

Lightning Strike Software Developers Kit
Using the SDK, a developer can integrate the highly efficient Lightning Strike module libraries into their own applications.

Download a FREE demo version of Lightning Strike Windows Compressor from: www.infinitron.com

Infinitron USA Office: 3401 East University, #104, Denton, TX. 76208
Tel: 817.484.1165 FAX: 817.484.0588

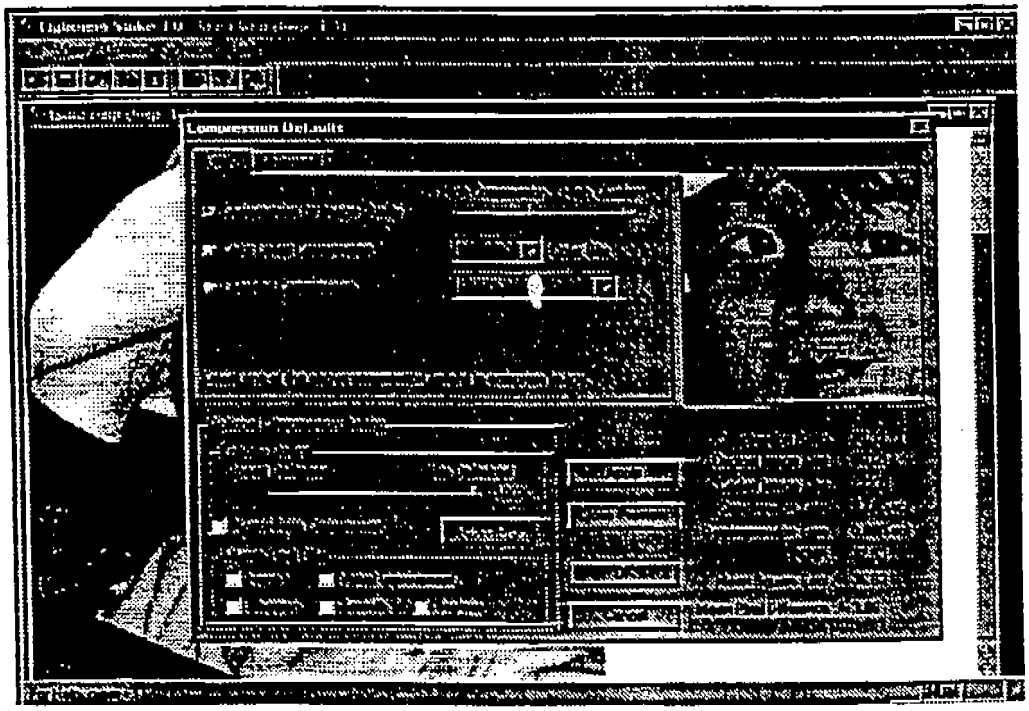
Infinitron Canada Office: 10th Flr 1199 W. Hastings, Vancouver, BC, V6E 3T5
Tel: 604.688.9789 FAX: 604.688.9789

INFINITRON

Windows Image Compressor V3.0

Lightning Strike

Announcing, Lightning Strike™ Image Compressor (LSIC) version 3.0, a Windows 95 tool that compresses still images from 20:1 to 200:1 using INFINITRON's proprietary wavelet technology. LSIC is a versatile, easy to use tool for Web and Graphic designers that can handle a wide variety of digital image formats, and it includes filters and convenient web tools. Images can be compressed at ratios well in excess of present JPEG ratios while maintaining comparable image fidelity. This translates to much shorter image down load time on the web. This has enormous benefits for reducing bottlenecks on corporate networks and the web, and in addition, requires less storage space.



Lightning Strike Features

Compression. Uses a proprietary integer wavelet.

Compression Control. An EASY mode allows the user to compress images with minimal input, requiring only a decision between quality and compression. An ADVANCED mode enables the user to select: 1) image file size, 2) compression ratio, 3) PSNR, or 4) master level for professionals where every parameter can be altered. We also provide the highest, wavelet lossless compression for users wishing this capability. Web designers will like the one step process to control the size of their image files allowing control over the delivery time of an image over a network.

Non Uniform Compression. Regions can be selected for less compression to preserve image quality while the rest of the image is compressed to the specified compression ratio. In this way, important parts of a picture maintain as small as possible

Post Reconstruction Filters. Filters are available to enhance the reconstructed image. At compression time the user can preset a control to have these filters operate automatically during reconstruction. The filters include: visual quality improvement, sharpen (edge enhancement), smoothing, and brighten.

Transparencies. The user will have the ability to set pixels transparent, so that a color in the background (already on the page) can be seen through the picture. This is useful for creative web site developers.

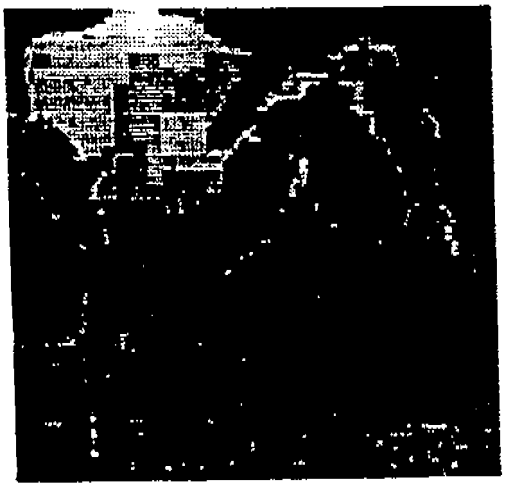
Progressive Compression. An image can be compressed so that when it is viewed it will appear quickly, first with low resolution, and then progressively building up in detail as it is downloaded. This insures the viewer gets

INFINITRON

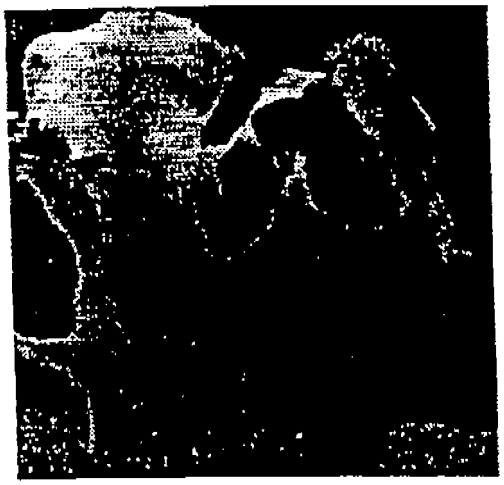
Windows Image Compressor V3.0

Lightning Strike

Cow. Bitmap. No Compression



Cow. JPEG 85:1 Compression



Cow. Lightning Strike. 85:1 compression

Applications for Lightning Strike Image Compression

- Images on the Web
- Photo Stock
- Data Warehousing
- Catalogues
- Video Games
- CDs (Encyclopedias. Museums. Science. and Medicine)
- Archives (Art. History. Genealogy)
- Medical Imaging

Performance

Encode time typically less than 2.5 seconds for a 640 X 480 pixel. 24 bit color image on a 133 MHz. Penuum with 16 MB RAM. Decode time is less than .75 seconds.

Minimum Recommended System

Windows 95/NT OS
Pentium 100 MHz. 8 MB RAM
2 MB for program files
10 MB plus to swap image files

Ancillary INFINITRON Products

- Netscape Navigator Plug-in
- Java Applet
- ActiveX Control
- Web Site Image Converter
- Lightning Strike SDK
- GML Banner Animation/Compression
- Black and White Image Compression

About INFINITRON. Inc.

Founded in 1992. INFINITRON. Inc. is a private company that specializes in the design and marketing of high quality image and video compression solutions for a wide array of markets. INFINITRON is based in Vancouver. BC with offices in Regina. Saskatchewan and Denton Texas.

Download a FREE demo version of Lightning Strike Windows Compressor from:
www.infinitron.com

INFINITRON USA Office 3401 East University. #104. Denton. TX. 76208
Tel: 817.484.1165 FAX: 817.484.0586

INFINITRON 10th Flr 1199 W. Hastings. Vancouver. BC. V6E 3T5

IPR2023-00330 Page 00921

What is claimed is:

1. A method for compressing an image, comprising the steps of:

performing a wavelet transformation of the image;

quantizing the wavelet transformed image;

applying entropy coding to the quantized image; and

outputting a file that includes the entropy coded image.

2. The method of claim 1, further comprising the following step:

performing a color transformation of the image.

3. The method of claim 1, further comprising the following step:

performing the wavelet transformation using an integer wavelet transform.

4. The method of claim 3, further comprising:

deriving the integer wavelet transform using a lifting scheme.

5. The method of claim 3, further comprising:

deriving the integer wavelet transform using a correction method.

6. The method of claim 1, wherein the step of quantizing includes the sub-step of:

processing the wavelet transformed image using sub-band oriented quantization.

7. The method of claim 1, further comprising:

comparing the wavelet transformed image to at least one predetermined threshold value.

8. A method for wavelet-based image compression using reduced color components, comprising the steps of:

creating a color table for an input image having a plurality of pixels;

calculating an index for each of the pixels, whereby generating a plurality of indices;

performing a wavelet transformation on the indices;

applying entropy coding on the transformed indices; and

outputting a file that includes the entropy coded indices.

9. The method of claim 8, further comprising:

dithering the pixels to generate the indices.

10. The method of claim 8, further comprising:

partitioning a large image into a plurality of small images to produce the input image.

11. The method of claim 10, wherein the large image is selectively partitioned.

12. An image processing system, comprising:

means for performing a wavelet transformation on an input image;

means for quantizing the wavelet transformed image;

means for entropy coding to the quantized image; and

means for outputting the entropy coded image.

13. The image processing system of claim 12, further comprising:

means for receiving the entropy coded image;

means for entropy decoding the received image;

means for de-quantizing the decoded image; and

means for performing an inverse wavelet transformation on the de-quantized image to produce an output image.

14. The image processing system of claim 12, further comprising:

means for displaying the output image.

15. The image processing system of claim 12, further comprising:

means for transmitting the entropy encoded image over a communications medium.

16. An image compression system, comprising:

a compressor configured to generate a compressed image based on an integer wavelet transform derived using a technique selected from a lifting scheme and a correction method.

17. The image compression system of claim 16, wherein the compressor quantizes a wavelet transformed image to produce the compressed image.

18. The image compression system of claim 16, wherein the compressor entropy encodes a quantized image to produce the compressed image.

19. The image compression system of claim 16, wherein the compressor performs a color transformation to produce the compressed image.

20. An image decompression system, comprising:

a decompressor configured to generate a decompressed image based on an integer inverse wavelet transform derived using a technique selected from a lifting scheme and a correction method.

21. A computer-readable memory storing a computer program for directing a computer system to perform image compression, wherein the computer program implements steps for performing a wavelet transformation of an input image, quantizing the wavelet transformed image, applying entropy coding to the quantized image, and outputting a file that includes the entropy coded image.

22. A method of compressing a data file, comprising the steps of:

performing a wavelet transformation of the data file to provide a series of wavelet coefficients;

quantizing those wavelet coefficients which fall above a predetermined threshold value to provide a quantized series of wavelet coefficients; and

compressing the quantized series of wavelet coefficients to provide a compressed data file.

23. The method of claim 22 wherein the compressing step comprises the step of applying an entropy coding to the quantized series of wavelet coefficients.

24. The method of claim 23 wherein the entropy coding is selected from the group of arithmetic, Huffman, run length and Huffman run length combined.

25. The method of claim 23 further comprising the step of performing a color transformation of the data file prior to the wavelet transformation step.

26. The method of claim 25 wherein the quantizing step comprises sub-band orientation quantization.

27. The method of claim 26 wherein the wavelet transformation step comprises integer wavelet transformation.

28. The method of claim 22 further comprising the step of filtering the data file prior to the wavelet transformation step.

29. The method of claim 27 wherein the integer wavelet transformation comprises biorthogonal filter method.

30. The method of claim 27 wherein the integer wavelet transformation comprises the correction method.

31. A compressed data file comprising a wavelet transformation of a data file having a series of compressed, quantized wavelet coefficients, the quantized wavelet coefficients having a value above a predetermined threshold value to provide a quantized series of wavelet coefficients.

32. A program for compressing a data file comprising:

a routine for performing a wavelet transformation of the data file to provide a series of wavelet coefficients;

a routine for quantizing those wavelet coefficients which fall above a predetermined threshold value to provide a quantized series of wavelet coefficients; and

a routine for compressing the quantized series of wavelet coefficients to provide a compressed data file.

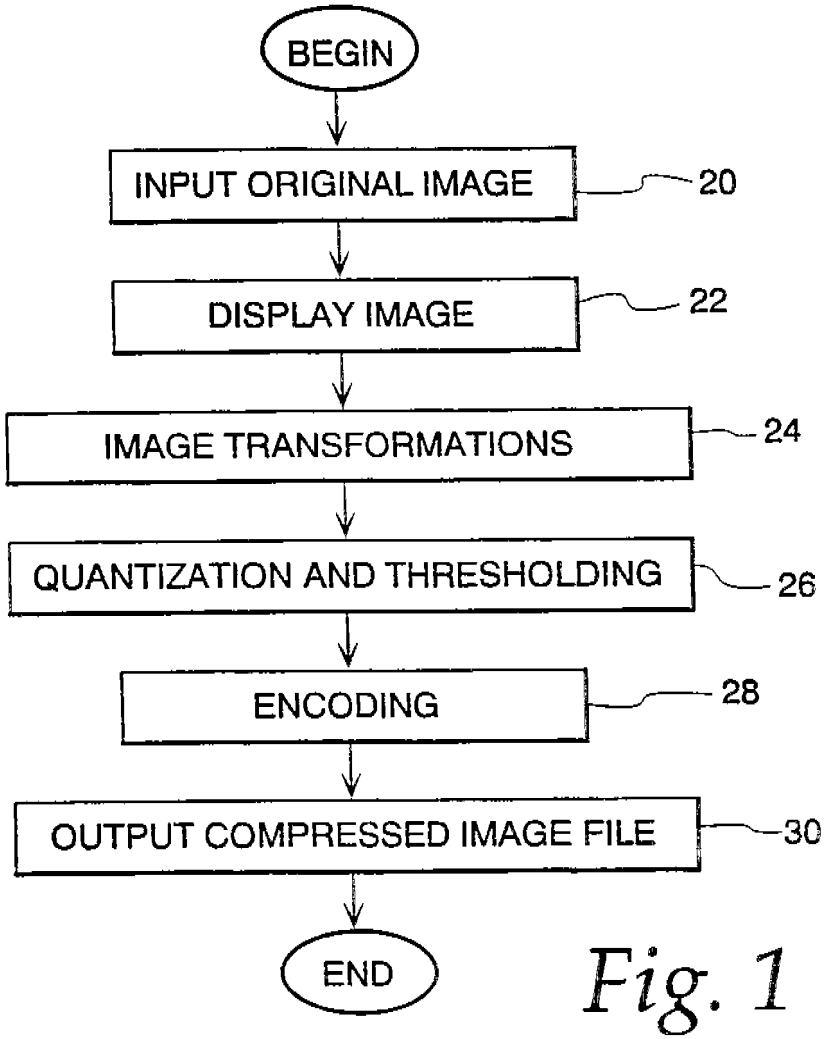

BEGIN
INPUT ORIGINAL IMAGE
20
DISPLAY IMAGE
22
IMAGE TRANSFORMATIONS
24
QUANTIZATION AND THRESHOLDING
26
ENCODING
28
OUTPUT COMPRESSED IMAGE FILE
30
END

Fig. 1

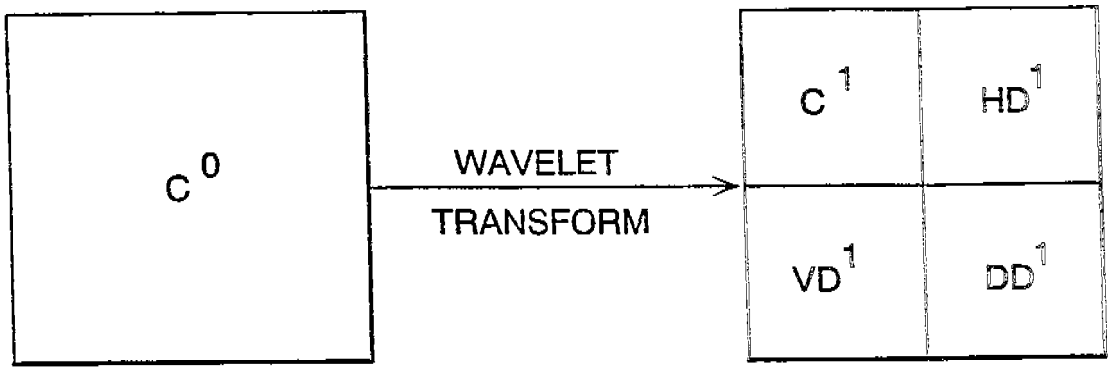

C 0
WAVELET
TRANSFORM
C 1
HD 1
VD 1
DD 1

Fig. 2

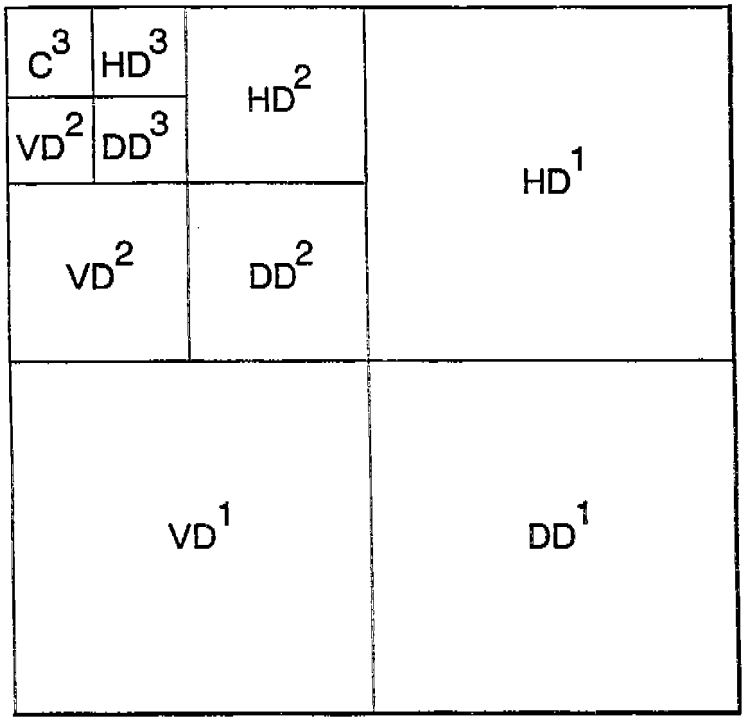

C3
HD3
HD2
VD2
DD3
HD1
VD2
DD2
VD1
DD1

Fig. 3

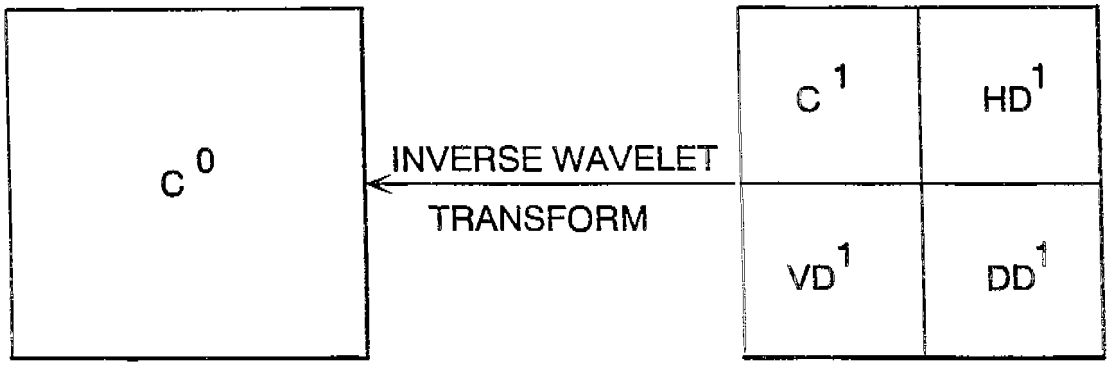

C 0
INVERSE WAVELET
TRANSFORM
C 1
HD1
VD1
DD1

Fig. 4

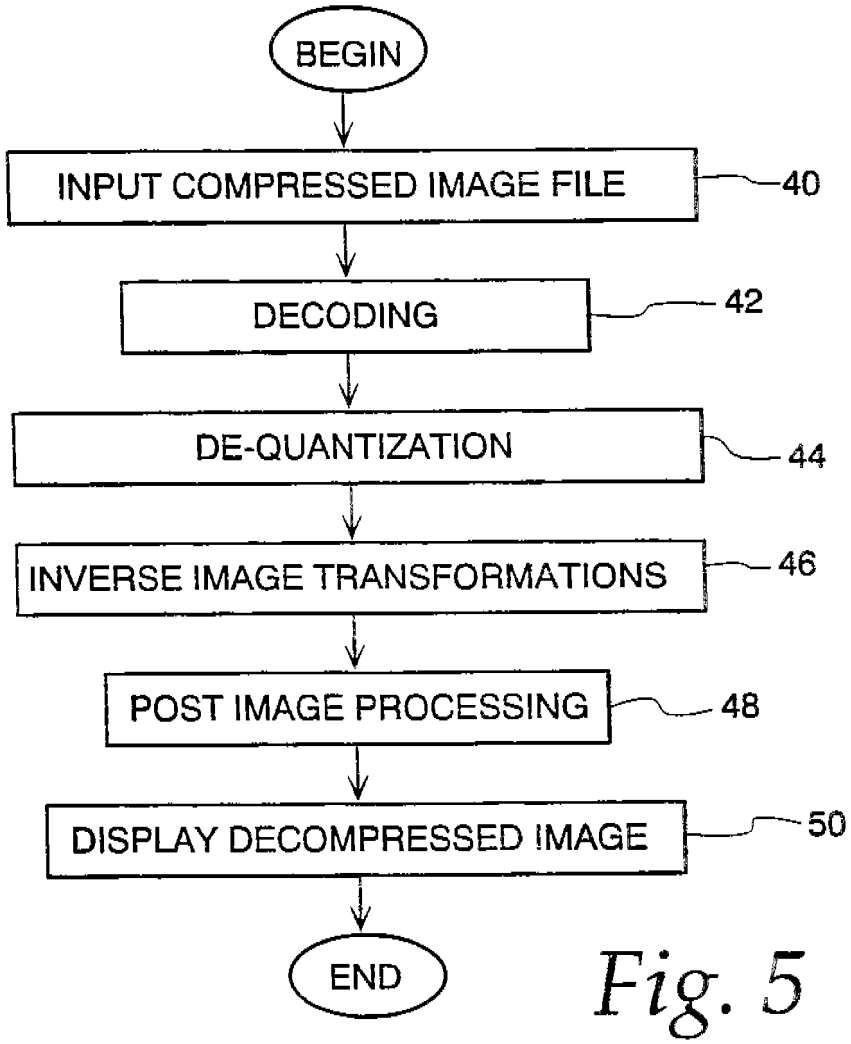

BEGIN
INPUT COMPRESSED IMAGE FILE
40
DECODING
42
DE-QUANTIZATION
44
INVERSE IMAGE TRANSFORMATIONS
46
POST IMAGE PROCESSING
48
DISPLAY DECOMPRESSED IMAGE
50
END

Fig. 5

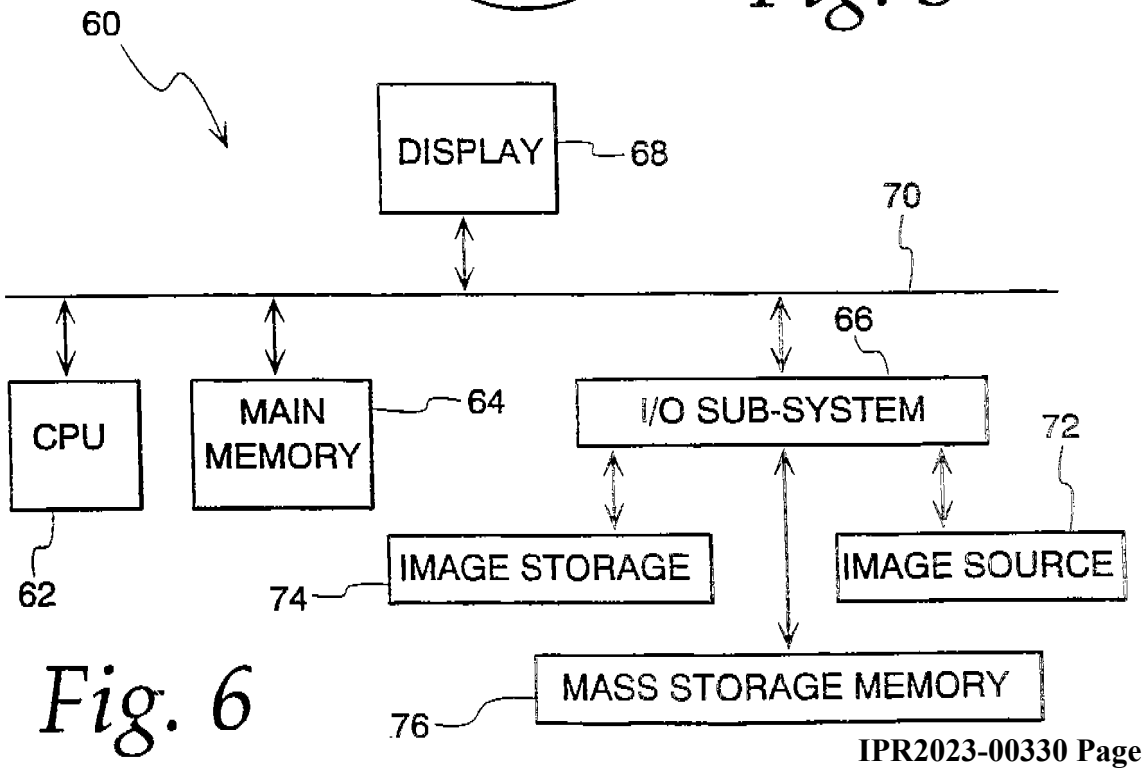

60
DISPLAY
68
70
66
CPU
MAIN MEMORY
64
I/O SUB-SYSTEM
72
62
74
IMAGE STORAGE
IMAGE SOURCE
MASS STORAGE MEMORY
76

Fig. 6

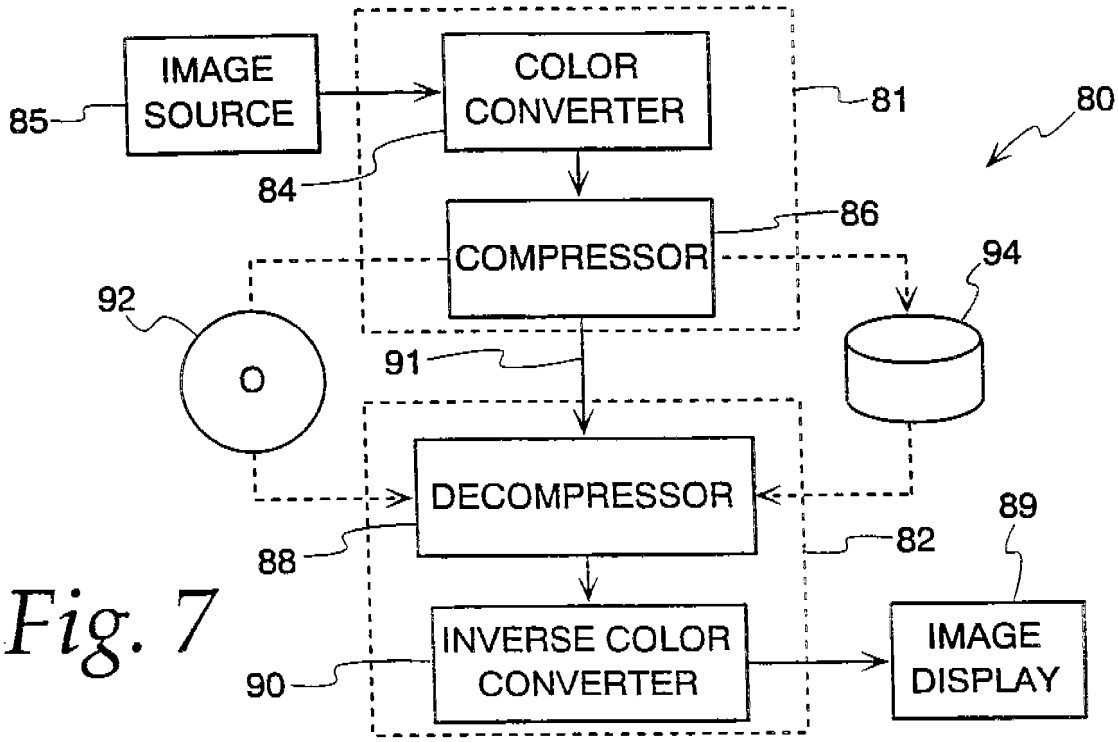


IMAGE SOURCE
85
COLOR CONVERTER
81
80
84
COMPRESSOR
86
94
92
O
91
DECOMPRESSOR
88
82
89
INVERSE COLOR CONVERTER
90
IMAGE DISPLAY

Fig. 7

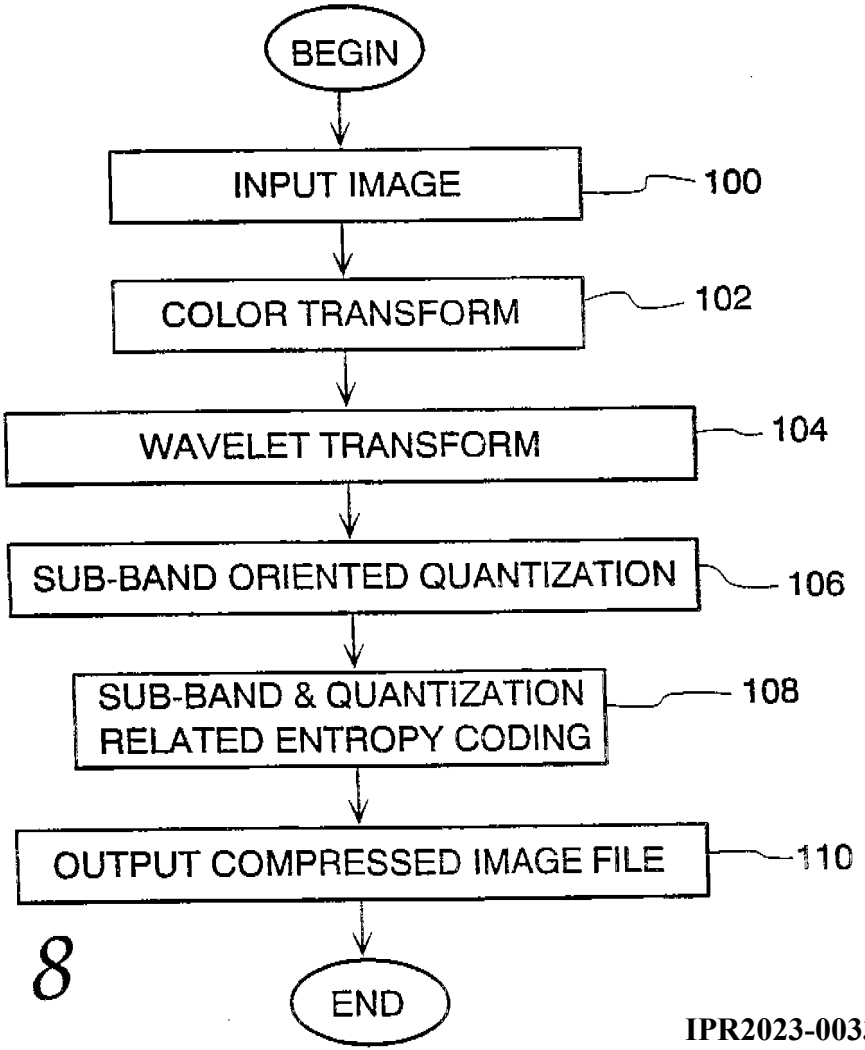

BEGIN
INPUT IMAGE
100
COLOR TRANSFORM
102
WAVELET TRANSFORM
104
SUB-BAND ORIENTED QUANTIZATION
106
SUB-BAND & QUANTIZATION RELATED ENTROPY CODING
108
OUTPUT COMPRESSED IMAGE FILE
110
END

Fig. 8

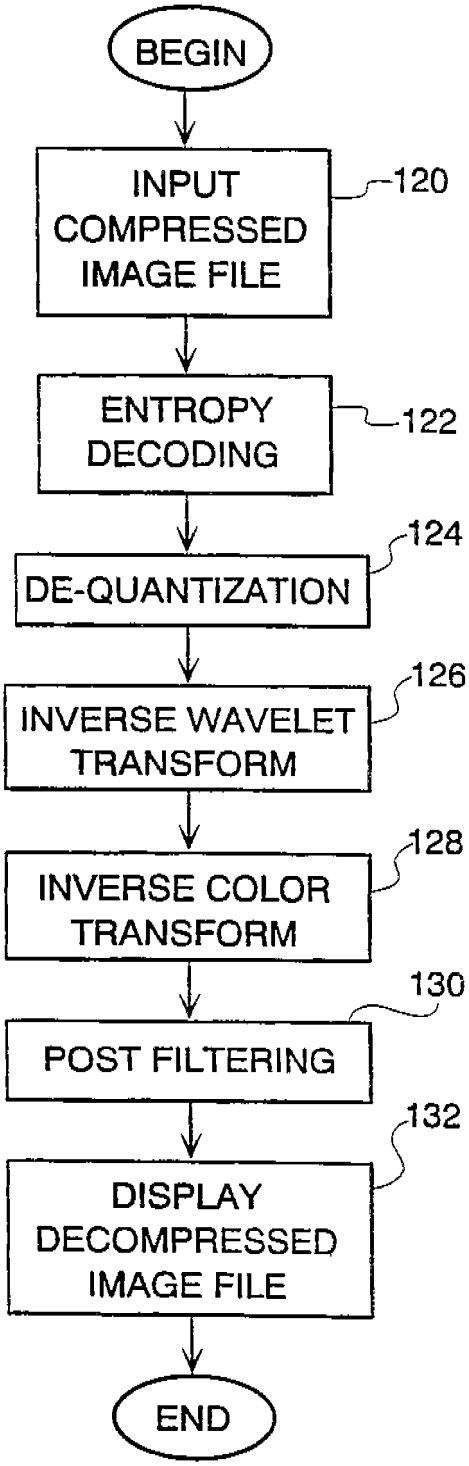

BEGIN
INPUT COMPRESSED IMAGE FILE
120
ENTROPY DECODING
122
DE-QUANTIZATION
124
INVERSE WAVELET TRANSFORM
126
INVERSE COLOR TRANSFORM
128
POST FILTERING
130
DISPLAY DECOMPRESSED IMAGE FILE
132
END

Fig. 9

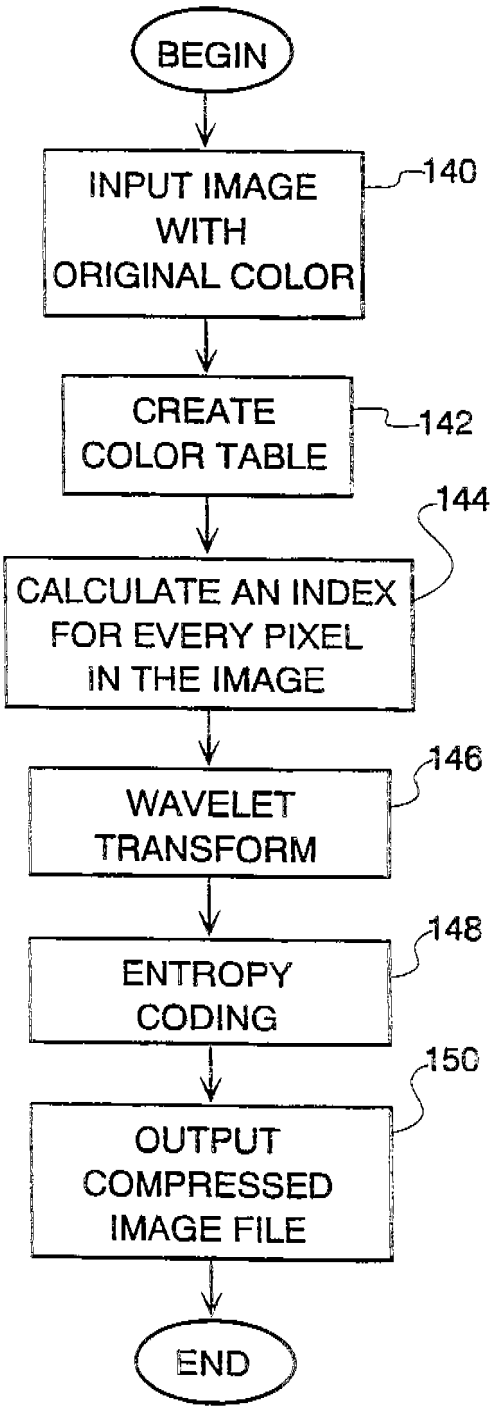

BEGIN
INPUT IMAGE WITH ORIGINAL COLOR
140
CREATE COLOR TABLE
142
CALCULATE AN INDEX FOR EVERY PIXEL IN THE IMAGE
144
WAVELET TRANSFORM
146
ENTROPY CODING
148
OUTPUT COMPRESSED IMAGE FILE
150
END

Fig. 10

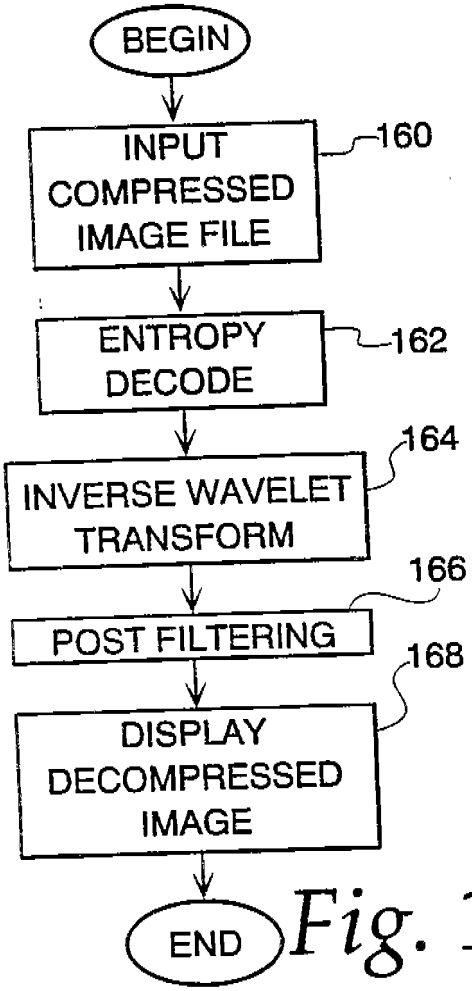


Fig. 11

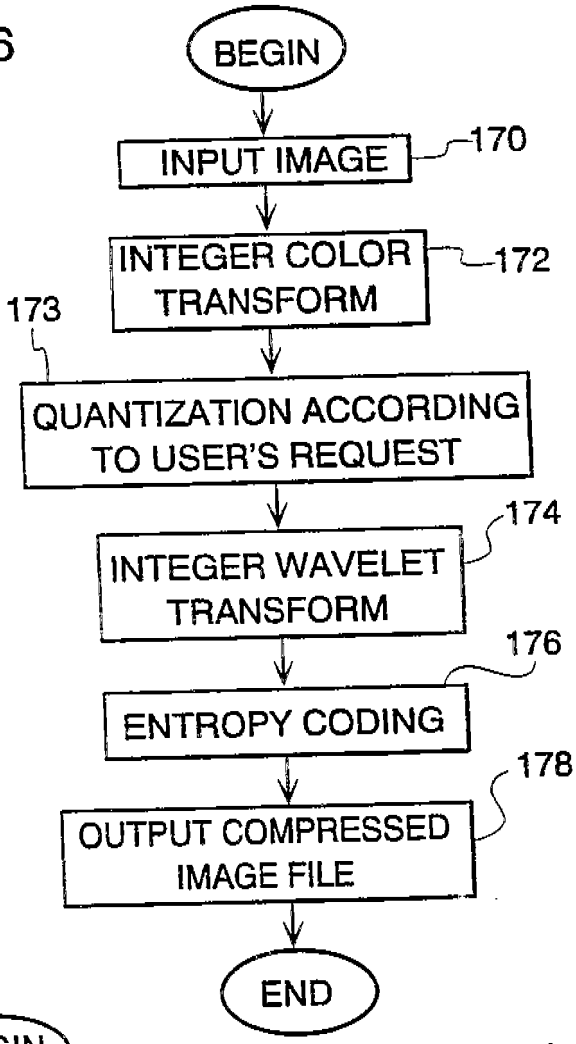


Fig. 12

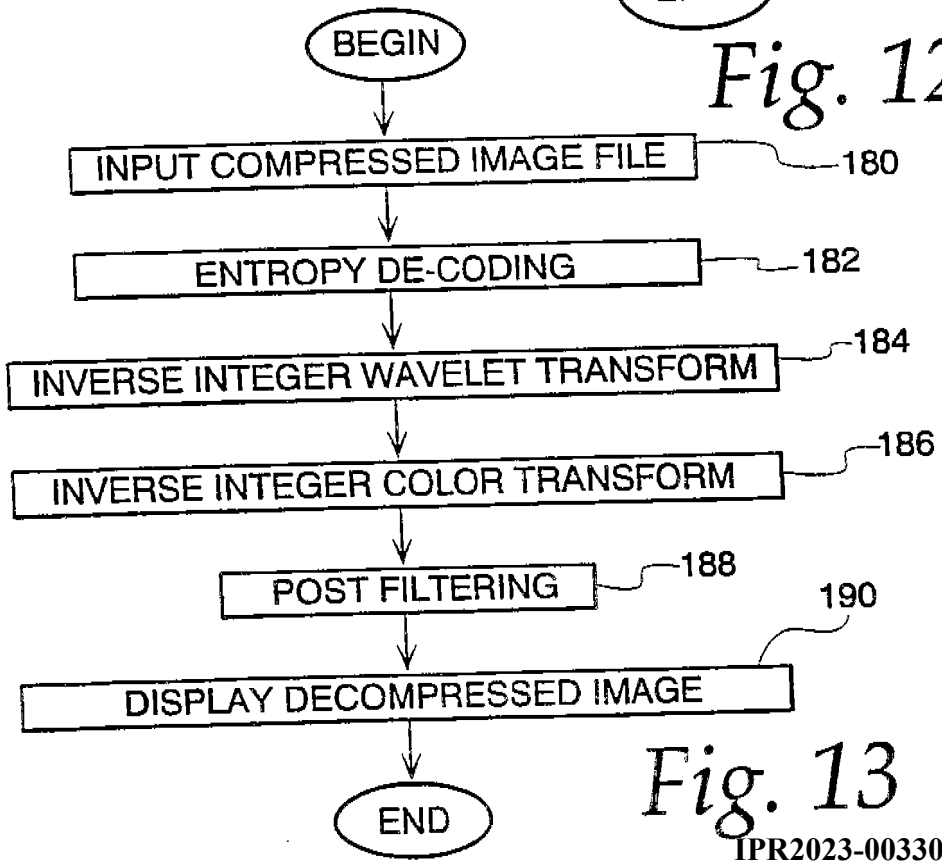


Fig. 13

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US98/04700

A. CLASSIFICATION OF SUBJECT MATTER

IPC(6) :G06K 9/00
US CL :Please See Extra Sheet.
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 382/232, 233, 236, 238, 239, 240, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 5,495,292 A (ZHANG et al) 27 February 1996, col. 3, lines 1-68.	1-32
Y	US 5,414,780 A (CARNAHAN) 09 May 1995, col. 4, lines 1-68.	1-32

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier document published on or after the international filing date	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&"	document member of the same patent family
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search	Date of mailing of the international search report
09 JUNE 1998	24 AUG 1998
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231	Authorized officer LEO BOUDREAU

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US98/04700

A. CLASSIFICATION OF SUBJECT MATTER:
US CL :

382/232, 233, 236, 238, 239, 240, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 13/38, 15/17 — A1

(11) International Publication Number: WO 98/43177

(43) International Publication Date: 1 October 1998 (01.10.98)

(21) International Application Number: PCT/US98/05304

(22) International Filing Date: 19 March 1998 (19.03.98)

(30) Priority Data:

60/041,366	25 March 1997 (25.03.97)	US
08/925,275	8 September 1997 (08.09.97)	US

(71) Applicant: INTEL CORPORATION [US/US]; 2200 Mission College Boulevard, P.O. Box 58119, Santa Clara, CA 95052–8119 (US).

(72) Inventors: TSO, Michael, Man–Hak; 5744 S.E. Preston Court, Hillsboro, OR 97123 (US). WILLIS, Thomas, G.; 619 S.W. Arboretum Circle, Portland, OR 97221 (US). RICHARDSON, John, W.; 2748 N.E. 19th Avenue, Portland, OR 97212 (US). KNAUERHASE, Robert, Conrad; 4926 S.W. Corbett Avenue #108, Portland, OR 97201 (US). MACIELINSKI, Damien; 415 S.W. 121st Place, Portland, OR 97225 (US).

(74) Agents: ALTMILLER, John, C. et al.; Kenyon & Kenyon, 1025 Connecticut Avenue, N.W., Washington, DC 20036 (US).

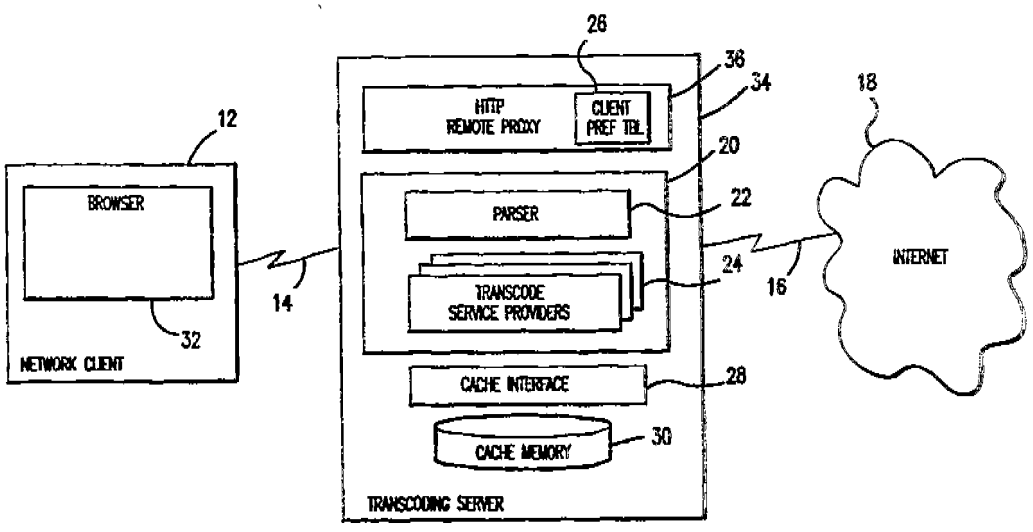
(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, GM, GW, HU, ID, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, UZ, VN, YU, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

Published

With international search report.

Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: SYSTEM FOR DYNAMICALLY TRANSCODING DATA TRANSMITTED BETWEEN COMPUTERS



(57) Abstract

A system for dynamically transcoding data transmitted between computers is implemented in an apparatus for use in transmitting data between a network server (10) and a network client (12) over a communications link (14). The apparatus includes a parser (22) coupled to a transcode service provider (24). The parser (22) is configured to selectively invoke the transcode service provider (24) in response to a predetermined selection criterion.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

SYSTEM FOR DYNAMICALLY TRANSCODING DATA TRANSMITTED BETWEEN COMPUTERS

Background of the Invention

This application claims the benefit of U.S. Provisional Application No. 60/041,366, filed March 25, 1997.

Field of the Invention

The present invention relates generally to the field of data communications for personal computers (PCs), and in particular to a system for dynamically transcoding data transmitted between two computers over a communications link.

Related Art

The Internet is quickly becoming the preferred data communications medium for a broad class of computer users ranging from private individuals to large multi-national corporations. Such users now routinely employ the Internet to access information, distribute information, correspond electronically, and even conduct personal conferencing. An ever-growing number of individuals, organizations and businesses have established a presence on the Internet through "web pages" on the World-Wide Web (WWW).

For a wide variety of reasons, it may be desirable to manipulate data transmitted between a local client computer and a network server computer. For

IPR2023-00330 Page 00937

example, in certain instances it may be advantageous to dynamically add, modify or delete content retrieved from an Internet server computer before that content is provided to a client computer. Conversely, it may be advantageous to modify a content request from a client computer prior to transmitting the request to an Internet server computer. While such dynamic manipulation of requests and responses is desirable, it is impractical to expect the expansive Internet infrastructure to quickly change to accommodate such a new capability. For this reason, it is desirable to implement such new capabilities in a way that does not require changes to either existing client computers or Internet server computers.

It is known to deploy a proxy server, or network proxy, as an intermediary between one or more client computers and an external network such as the Internet. Network proxies are described generally in Ian S. Graham, HTML Source Book: A Complete Guide to HTML 3.0 403 (2d ed. 1996). One common application for a proxy server is as a so-called "firewall," wherein the proxy server is responsible for all communications with the outside world. In other words, local devices are not permitted to communicate directly with external network computers, such as Internet servers. Instead, each local device directs requests for network-resident data to the proxy server. When the proxy server receives such a request, it forwards the request to the appropriate external computer, receives the response from the external computer, and then forwards the response to the local device. The external computer thus has no knowledge of the local devices. In this way, the local devices are protected from potential dangers such as unauthorized access.

Existing proxy servers do not manipulate the data passing through them. In essence, proxy servers are merely blind conduits for requests and responses. This limitation of existing proxy servers restricts these devices from being used to full advantage when facilitating communications between local devices and network devices. There is therefore a need for a so-called "smart" proxy capable of examining the data passing through it, whether it be a request intended for an external network device or network content being returned to a local device, and dynamically acting

IPR2023-00330 Page 00938

upon that data. Such a device can be used to transparently provide a wide range of services that were heretofore impossible without modifying existing Internet infrastructure.

Summary of the Invention

Embodiments of the present invention relate to devices, systems and methods for transcoding information transmitted between computers, such as a network server computer and a network client computer.

According to one embodiment, an apparatus for use in transmitting data between a network server and a network client over a communications link includes a parser coupled to a transcode service provider. The parser is configured to selectively invoke the transcode service provider in response to a predetermined selection criterion.

Brief Description of the Drawings

Fig. 1 is a schematic diagram illustrating an environment in which embodiments of the present invention may be applied.

Fig. 2 is a schematic diagram illustrating a transcoder module according to an embodiment of the present invention.

Fig. 3 is a schematic diagram illustrating an embodiment of the present invention for a non-enabled network client.

Fig. 4 is a schematic diagram illustrating an example of a user interface for providing a non-enabled network client with control over transcoding functionality.

Fig. 5 is a schematic diagram illustrating an embodiment of the present invention for an enabled network client.

Fig. 6 is a schematic diagram illustrating a network client with transcoding functionality integrated in a browser according to an embodiment of the present invention.

IPR2023-00330 Page 00939

Figs. 7-9 are flow charts illustrating logic for presenting a requested URL object to a network client according to an embodiment of the present invention.

Detailed Description

Embodiments of the present invention provide the ability to dynamically transcode information transmitted between, for example, a network server computer and a network client computer. As used herein, the term "transcode" applies to virtually any manipulation of data including, but not limited to, adding, modifying or deleting data.

Referring now to **Fig. 1**, which illustrates an environment in which embodiments of the present invention may be advantageously applied, a network server **10** manages the transfer of data from the Internet **18** to a network client **12**. Network client **12** may be any computer having suitable data communications capability.

Network client **12** communicates requests for information to, and receives information from, network server **10** over a client/server communications link **14**. Client/server communications link **14** may comprise, for example, a so-called "slow network" using, for example, POTS (Plain Old Telephone System) dial-up technology or wireless connections. Alternatively, client/server communications link **14** may comprise a so-called "fast network," such as a LAN or WAN (Wide Area Network), which is capable of operating at much higher speeds than are possible with slow networks. Combinations of these access methods are also possible. For example, network client **12** may use a POTS or wireless dial-up connection to a modem bank maintained by an ISP (Internet Service Provider), which is in turn connected to network server **10** over a LAN. Network server **10** communicates with computers resident on Internet **18** through server/network communications link **16**, which may comprise any suitable communications medium known in the art.

IPR2023-00330 Page 00940

According to a first general embodiment of the present invention, illustrated schematically in **Fig. 2,** a transcoder **20** includes a parser **22** and a plurality of transcode service providers **24**. Parser **22** is configured to act upon data received by transcoder **20**, such as a request for a network object generated by a client device or a reply to such a request provided by a content server device. In this particular embodiment, parser **22** is responsible for selectively invoking one or more of transcode service providers **24** based upon a predetermined selection criterion.

Transcoder **20** may be implemented, for example, as a software module installed in a network proxy, in a client device, in a network server device, or in a content server device. In one particular implementation, illustrated in **Fig. 3**, transcoder **20** is installed in a remote transcoding server **34** arranged between network client **12** and Internet **18**. Transcoding server **34** may comprise, or be a part of, a network server, a stand-alone computer in communication with a network server, or a distributed system of computers. Remote transcoding server **34** may be coupled, for example, to an ISP's network, a corporate network, or anywhere on Internet **18**, and may provide multiple users (i.e., clients) with a means to obtain content on Internet **18**.

In the particular embodiment illustrated in **Fig. 3**, transcoding server **34** includes an HTTP (HyperText Transfer Protocol) remote proxy **36**, capable of accessing Internet **18** over server/network communications link **16**. HTTP remote proxy **36** differs from known network proxies, which generally are little more than a conduit for requests to, and replies from, external Internet resources, in that it is capable not only of examining such requests and replies, but also of acting upon commands in the requests by, for example, determining whether or not to transcode content. Moreover, using transcoder **20**, HTTP remote proxy **36** is capable of changing content received from Internet **18** prior to returning it to a requesting network client **12**, as is explained further below.

IPR2023-00330 Page 00941

Looking more closely at the embodiment in **Fig. 3**, transcoder **20** is coupled to HTTP remote proxy **36**. Parser **22** manages the transcoding of data to be transmitted from transcoding server **34** to network client **12**. To this end, parser **22** controls transcode service providers **24** to selectively transcode content based on a predetermined selection criterion. For example, one or more transcode service providers **24** may provide the capability to compress and/or scale different types of data content, such as image, video, or HTML (HyperText Markup Language). Such uses are described further in co-pending U.S. patent applications Serial No. 08/772,164 entitled "System for Enhancing Data Access Over a Communications Link," filed on December 20, 1996, and Serial No. 08/799,654 entitled "Method and Apparatus for Scaling Image Data," filed on February 11, 1997, both of which are assigned to Intel Corporation. For purposes of illustrating certain features of the present invention, a number of embodiments are described below in terms of content scaling/compression; however, as is explained, transcode service providers **24** may provide a wide variety of transcoding functions.

As shown in **Fig. 3**, transcoding server **34** may also include a server-side cache memory **30** managed by a server-side cache interface **28**. Server-side cache memory **30** may be used to store both original and transcoded versions of content for later transmission to network client **12** without the need to re-retrieve the content from Internet **18** or to re-transcode the content.

Transcoding server **34** is coupled to network client **12** by client/server communications link **14**. Network client **12** includes a browser **32**, such as the Netscape Navigator v.3.0 browser (although the invention is not limited in this respect), which manages the presentation of data to a user. In this embodiment, network client **12** is "non-enabled," meaning no specialized transcoding software is preloaded on network client **12**.

Parser **22** may comprise a relatively simple, uniform interface to

IPR2023-00330 Page 00942

HTTP remote proxy **36**, and may provide an API (Application Programming Interface) for transcoding data received by HTTP remote proxy **36**. Parser **22** manages one or more transcode service providers **24** that are accessed through a common SPI (Service Provider Interface). In this particular embodiment, parser **22** is designed in compliance with the Windows Open Systems Architecture (WOSA), and may be implemented as a Win32 DLL (Dynamic Link Library). The WOSA architecture, described in Readings on Microsoft Windows and WOSA (Microsoft Corp. 1995), enables additional transcode service providers **24** to be dynamically added to the system to provide new features and/or better transcoding algorithms, while at the same time not requiring changing or retesting other software components in the system. This feature is especially beneficial where transcoding server **34** also interacts with "enabled" network clients equipped with specialized transcoding software. It should be noted that some of the features of parser **22** described below may be inapplicable to the non-enabled client embodiment of **Fig. 3**; however, transcoding server **34** may advantageously be configured flexibly enough to process requests from both non-enabled and enabled network clients.

Like parser **22**, server-side cache interface **28** may be modeled after a standard Get/Set interface. Server-side cache memory **30** essentially "owns" all cached objects, in that it manages the properties and storage of the objects and may invalidate any non-locked object at any time; however, the actual format of any given cached object is known only by parser **22** and its associated transcode service providers **24**. Thus, for data integrity and transcoding efficiency purposes, all access to server-side cache memory **30** in this embodiment is through parser **22**.

Server-side cache interface **28** may include the following calls:

```
CreateEntry(URL, &Entry, ...);
GetEntry(URL, &Entry);
CreateStream(Entry, &StreamEntry, ...);
GetStream(Entry, &StreamEntry, ...);
CloseEntry(Entry);
CloseStreamEntry(StreamEntry);
```


IPR2023-00330 Page 00943

```
GetProperties(Entry, &Properties, ...);
SetProperties(Entry, &Properties, ...);
Read(StreamEntry, &OutStream, ...);
Write(StreamEntry, &InStream, ...).
```

Unlike most cache memories, server-side cache interface **28** and server-side cache memory **30** enable maintenance of multiple representations of a given cached object, with descriptive information about each representation included in server-side cache memory **30**. In addition, server-side cache interface **28** and server-side cache memory **30** serve as a synchronization point for multi-threaded accesses to cached objects. It should be noted that the illustrated embodiment does not require any particular configuration for server-side cache interface **28** and/or server-side cache memory **30**. Indeed, functionality attributed to these components in the various embodiments described herein may be readily implemented in other system components.

The CreateEntry() call creates and returns a cache entry for a specified hypertext object. This call also creates an entry stream for an original version of the hypertext object. Similarly, the GetEntry() call obtains a cache entry for a hypertext object already existing in cache memory **30**. Both the CreateEntry() and GetEntry() calls set locks on associated cached objects until a CloseEntry() call is invoked. Once a lock is set, the cached object will not be replaced or invalidated by cache interface **28**, permitting one or more transcode service providers **24** to safely perform any required cache operations, such as object retrieval and/or storage.

After a cache entry is created or opened by a CreateEntry() or GetEntry() call, the CreateStream() or GetStream() calls may respectively create or open an extra stream entry for the cached object. Each extra stream entry is associated with a different transcoded version of the hypertext object, which may be retrieved or appended to by one of transcode service providers **24**. Stream-based processing of cached objects makes it possible for transcoding server **34** to begin transmitting a transcoded version of a hypertext object to a requesting network client

IPR2023-00330 Page 00944

12 even while transcode service provider 24 is appending additional transcoded content to that same version. Advantages of this stream-based processing include reducing user latency through incremental painting of objects and avoiding unnecessary idle time on client/server communications link 14, thereby providing users with a more responsive "feel."

The GetProperties() and SetProperties() calls retrieve and store information about cached objects, including information maintained by transcode service provider 24 used to determine transcoding properties and transcoding status of a cached object. Transcode service provider 24 may use such information, for example, to determine current compression progress for scaled data access and staged refinements.

The Read() call reads data from a specified cached object data stream. For example, transcode service provider 24 may invoke this call and tunnel stream data through HTTP remote proxy 36 directly to network client 12. The Write() call caches data from a new HTTP data stream. This call will append an incoming data stream received from, for example, a Web server or transcode service provider 24, to an opened cache stream which may be concurrently read using the Read() call.

In the present embodiment, parser 22 includes the following calls:

GetObject(URL, InParams, &OutParams, &OutStream, ...);
GetScaledObject(URL, InParams, &OutParams, &OutStream, Stage, ...);
PutObject(URL, InParamStruct, &InStream, &OutParams, &OutStream, ...).

As detailed below, parser 22 uses these calls to manage the provision of requested content to network client 12.

The GetObject() call is used to service non-enabled client requests, and returns a non-transcoded (i.e., original) version of a specified hypertext object. In this embodiment, transcoding server 34 assumes that each HTTP request has a unique thread that may be blocked until the request is satisfied. Accordingly, the

GetObject() call will block until it either returns the requested data stream or indicates failure with a cause (e.g., object does not exist). This ability to return a so-called standard hypertext object is advantageous for compatibility reasons, enabling embodiments of the present invention to be used with existing browsers that do not include support for certain transcoding functionality (e.g., advanced data compression), and enabling users to selectively retrieve non-transcoded versions.

The GetScaledObject() call is similar to GetObject(), and is also used to request an object from server-side cache memory **30**; however, it adds support for requesting a particular version of that object, such as a high-quality rendition. Unlike traditional caching proxies, transcode service providers **24** can use server-side cache memory **30** to store several different versions of an object to support clients with different communications and/or presentation capabilities. Thus, an additional "Stage" parameter may be used to indicate which version of the cached object is to be returned to network client **12**. Where transcode service provider **24** is configured to scale network content, it may use this parameter to request a version of a cached object having, for example, a default scaled quality, a refinement to a better-quality version, or the original non-scaled version.

In this embodiment, when network client **12** requests a hypertext object, HTTP remote proxy **36** uses either the GetObject() or GetScaledObject() call (depending on if network client **12** is capable of receiving scaled/transcoded datatypes) to retrieve the hypertext object from parser **22**. If the hypertext object is not found, parser **22** uses the CreateEntry() call to create an entry (in effect, a placeholder) in server-side cache memory **30** for the new object. The new entry is returned to HTTP remote proxy **36**, which requests the hypertext object from Internet **18**. As a data stream for the hypertext object is returned, HTTP remote proxy **36** calls parser **22** using the PutObject() call, passing into this call the new entry and the handle to the data stream to be placed into the entry. Parser **22** selects an appropriate transcode service provider **24** based, for example, on the content type

IPR2023-00330 Page 00946

of the data stream. In this context, the term content type encompasses a datatype, an HTTP MIME (Multipurpose Internet Mail Extensions) type, a content format, and so on. The selected transcode service provider **24** uses a separate thread to read the incoming data stream, transcode it, and place it within the entry of server-side cache memory **30**. The current thread immediately returns to HTTP remote proxy **36**, which once again calls GetScaledObject() (or GetObject()). This case will always result in a cache hit. This thread then works simultaneously with the separate thread in the PutObject() to tunnel data (either original or transcoded) from transcoding server **34** to network client **12**.

Multiple-thread processing improves the efficiency of the present embodiment by not waiting for a hypertext object to be received in its entirety by HTTP remote proxy **36,** or added in its entirety to server-side cache memory **30**, before beginning to send the object to network client **12**. Another benefit of multiple-thread processing is that parser **22** may efficiently process requests for the same hypertext object from multiple network clients **12**. The hypertext object need only be retrieved from Internet **18** once, and appropriate versions may be transmitted to such multiple network clients **12** concurrently. It should be noted, however, that embodiments of the present invention may be implemented without multiple-thread processing.

As noted above, parser **22** may selectively invoke one of transcode service providers **24** based upon satisfaction of a predetermined selection criterion. Such selection criterion may comprise, for example, information contained in a header portion of a data packet received by transcoding server **34**, such as a MIME type, a URL (Uniform Resource Locator), a last modified time indicator and so on. Alternatively, the predetermined selection criterion may comprise information contained in a data portion of such a data packet, such as particular content, key words, structures (for example, heading levels), and so on. Still further, the predetermined selection criterion may comprise a condition of the device on which

IPR2023-00330 Page 00947

transcoding server **34** is installed (for example, a current processing load), a condition of a device to which transcoding server **34** is coupled, or a condition of a communications link. Transcoding server **34** may provide the ability to dynamically update such predetermined selection criteria.

The following discussion provides still more examples of the types of information which may be used to dictate which of transcode service providers **24** are invoked. It should be noted, however, that these examples are provided by way of illustration only, and are not intended to limit in any way the scope of the invention claimed herein. The predetermined selection criterion may comprise: (1) network client **12**, such as a display dimension, resolution, number of colors, processor type, memory/disk configuration, modem or network interface type, installed add-in boards (for example, hardware compression/decompression), software configuration (for example, availability of pre-installed software decompression modules), physical location/proximity (for example, as determined by a telephone area code), and user identity; (2) characteristics of transcoding server **34** or some other network server, including system load and identification information (for example, the owner of the server); (3) content characteristics, such as its data type, type of encoding/compression, size, and dimension; (4) network characteristics, including best-case, worst-case and average latency, bandwidth and/or error rates (for example, for wireless communications) between network client **12** and a proxy, and/or between a proxy and a server (this may be predetermined for guaranteed bandwidth links like ATM (Asynchronous Transfer Mode), or dynamically measured/predicted for so-called "best effort" links like many IP (Internet Protocol) links); (5) proxy characteristics, including system load, available storage, physical location/proximity, and identity (owner); (6) user preferences, including preferred content quality/speed tradeoff, language, content rating, exclusion list, inclusion list, data type-specific preferences (for example, "never download" images), include/exclude advertising, amount of advertising desired, offensive language removal, whether the user's defined or learned preferences may be disclosed (and to whom), custom rules or

IPR2023-00330 Page 00948

programs for filtering/transcoding/processing data, and shared preferences with either another user or a group of users (any of the foregoing user preferences may be explicitly defined or system predicated, such as based on usage statistics compiled over time); (7) group preferences, including results from collaborative rating systems, whether manual (for example, a prior user manually assigned a rating to a Web page after viewing it) or automatic (for example, given a large number of users who accessed a link on a given page, the probability of any given user subsequently following that link); (8) content provider preferences, including the degree of alteration desired for its content, the prioritization for download and display of different content types, cache restriction or prioritization parameters such as update frequency or replacement preferences, the types of users to target, rules or programs to run for customizing content (for example, news or advertising, custom language translation software) based on user or client characteristics, desire for receiving certain types of user or group data collected (for example, demographics or access patterns), and type of payment/reward offered in exchange for such information; and (9) other preferences, including software vendor rules or programs for dynamically checking content created or distributed using unauthorized software and companies' desire to enforce correct usage of certain types of content (for example, trademarks and logos).

Applying the above-listed selection criteria, or combinations thereof, embodiments of the present invention may be used to provide a virtually limitless range of dynamic transcoding services. For example, client and/or proxy physical proximity, in combination with demographic data, may be used for extremely targeted advertising. Such advertising may be added to any content passing through a proxy, for example, or some other mechanism. This can in turn be tailored even further based upon the user's willingness to tolerate advertising or share demographic information, as well as the advertiser's ability/willingness to subsidize or otherwise reward the user for participation.

IPR2023-00330 Page 00949

Embodiments of the present invention may be advantageously used to reduce the amount of data that is transmitted to network client **12**, thereby promoting faster downloading and rendering of content. Suitable transcoding techniques include lossy compression and transcoding to a more efficient (and perhaps not widely supported) format specifically for the transmission. Similarly, HTTP remote proxy **36** may be configured to "predigest" Web sites or groups of sites to produce extremely condensed overviews of large amounts of content (for example, a tree structure, pages with only first-level or first- and second-level headings, thumbnails of pages, or only parts of a page or site that have changed since the user's last visit). Such applications may be especially advantageous for poorly-connected or computationally limited devices such as PDAs (Personal Digital Assistant), since this predigestion can be performed on a well-connected proxy server with an abundance of computational power, and the concise result can be easily downloaded and rendered on the more limited device.

Embodiments of the present invention may alternatively be used for dynamic translation of data, such as Web pages, to a user's native language (determined by user preference or automatically by the physical location of network client **12** or transcoding server **34**). Such a capability greatly simplifies the task of making content truly global, as well as reduces storage and maintenance required at the content provider (that is, only one copy of the content need be maintained, rather than different copies for each of a plurality of different languages).

Embodiments of the present invention may be used to block certain types of content or to automatically censor offensive language (similar to a "beep" used for television broadcasts). Only the particular offensive parts of the content (for example, obscene words) may be removed, or entire Web sites may be blocked. Similarly, transcoding server **34** may be configured to scan content for certain words or phrases to ensure that trademarks or logos are used correctly (for example, as a source identifier rather than a generic product designation). This feature may be offered as a service to companies or organizations, who would supply a list of words

IPR2023-00330 Page 00950

or phrases to flag. A similar capability could be used to automatically insert links into the content upon detection of certain words or phrases. For example, Intel Corporation might want to automatically add a link to its corporate Website whenever the name "Intel" is used in a Web page. Using an embodiment of the present invention, such links can be dynamically added to the content before it is displayed to a user. In a similar vein, an embodiment of the present invention may be used to scan for content that was created or distributed using unlicensed software. This feature may be implemented using special keys (binary bit patterns) embedded in the content or headers put in by the content creation or distribution software. The scanning logic and logic for taking a predetermined responsive action, such as denying service or posting a warning, may optionally be supplied by the vendor of the software in question or configured into transcoding server **34**.

Embodiments of the present invention may also be used to scan content for computer viruses prior to sending such content to network client **12**. For example, an existing virus scanning routine may be installed on transcoding server **34**, possibly as a plug-in module Transcoding server **34** may then be configured to invoke the virus scanning routine to ensure any content transmitted to network client **12** is free of viruses. A significant advantage provided by such an embodiment is that virus scanning software need only be maintained on transcoding server **34**, rather than on a plurality of network clients **12**. In this way, the benefit of upgrades to the virus checking software may be efficiently and timely provided to large numbers of users, thus avoiding the problem of any particular user relying on outdated virus scanning software.

Embodiments of the present invention may also be used to produce custom content on demand in accordance with user-specific preferences and/or associations with collaborative rating systems. In a variation on such an embodiment, transcoding server **34** can collect preferences and append them as part of a client request transmitted to a content provider so that the dynamic content generation can

IPR2023-00330 Page 00951

be done at the content server. Likewise, a proxy provider (for example, an Internet Service Provider (ISP)), can collect and make available to content providers information such as user preferences and data access statistics, as well as content provider specific statistics (for example, how many users from a given region or a given profile accessed a particular Web site, and at what time, in the past month). Such information may be used for applications such as targeted advertising.

Embodiments of the present invention may further be used to automatically check the validity of links in an object, and correct or remove invalid links, prior to transmitting the object to network client **12**. This capability may be provided, for example, as a service to content providers who may not have the most up-to-date information on Websites they are linked to which have moved or been deleted.

To further illustrate the general operation of the embodiment illustrated in **Fig. 3**, assume a user of network client **12** wishes to access a particular Web page, or URL (Uniform Resource Locator), on Internet **18**. Further assume that the desired URL resides on, or is accessible through, transcoding server **34**. Network client **12**, via browser **32**, transmits an HTTP request for the hypertext object to transcoding server **34** over client/server communications link **14**. Where browser **32** normally accesses Internet **18** through a proxy, browser **32** is configured to pass user requests through transcoding server **34** via browser's **32** standard proxy configuration procedures. As is well known in the art, browser **32** may actually transmit a plurality of additional HTTP requests corresponding to each of various distinct hypertext objects that may be embedded in the Web page. In such a case, transcoding server **34** may process each such request in the manner described below.

According to this embodiment, HTTP remote proxy **36** is capable of distinguishing between a non-enabled network client **12** and an enabled network client **12**. This may be accomplished, for example, using a private protocol to

transmit content requests from an enabled network client to transcoding server **34**, so that the use of some other communications protocol indicates network client **12** is non-enabled. This method of sending a private protocol in each request to HTTP remote proxy **36** is an improvement over a registration type process. The overhead involved in making the enabled/non-enabled determination on a per request basis is relatively small, while providing a significant advantage because it addresses the situation for HTTP remote proxy **36** where a first network client disconnects and a second network client, likely with different communications and/or presentation capabilities, reconnects using the same IP address.

Upon determining that network client **12** is non-enabled, HTTP remote proxy **36** may record the IP address of network client **12** in a client preference table **26** maintained in a local data store (client preference table **26** may improve performance of this or other embodiments, but is not required). HTTP remote proxy **36** then passes the hypertext object to parser **22**. HTTP remote proxy **36** may also inform parser **22** of any applicable user preferences (e.g., from client preference table **26**). Upon being invoked, parser **22** first calls cache interface **28** with the requested hypertext object to determine whether a copy of the required version already resides in server-side cache memory **30**. For purposes of illustration, assume no entry exists in server-side cache memory **30** for the requested hypertext object. HTTP remote proxy **36** then invokes a call to retrieve the hypertext object from Internet **18** over server/network communications link **16**. Assuming the requested hypertext object is found, HTTP remote proxy **36** begins receiving an HTTP data stream representing the hypertext object. HTTP remote proxy **36** passes the handle for this incoming data stream to parser **22**.

Parser **22** dynamically determines whether the data stream satisfies any applicable predetermined selection criteria. For example, where transcode service providers **24** are configured to scale data of different types, parser **22** may determine the content type for the data stream (e.g., image/jpeg, image/gif,

IPR2023-00330 Page 00953

video/mpeg) by interrogating a MIME type in the content-type header record that appears at the beginning of the incoming HTTP data stream. If parser **22** detects a match for a predetermined selection criterion, the HTTP stream handle is given to the appropriate transcode service provider **24**. Transcode service provider **24** then transcodes the data stream appropriately, and HTTP remote proxy **26** transmits the transcoded data stream to network client **12**.

A non-enabled network client **12** may optionally be provided with the ability to actively control aspects of the transcoding process, or indeed whether or not to transcode requested content at all. To provide this ability, HTTP remote proxy **36** may embed additional instructions at the beginning of the HTML header for the requested URL prior to transmitting the associated data stream to network client **12**. These embedded instructions may be implemented, for example, as JavaScript codes, VB Script codes or Java Applet codes. As browser **32** of network client **12** receives the data stream, the embedded instructions will automatically execute so long as browser **32** is equipped to support them. For example, if the embedded instructions are implemented as JavaScript codes, browser **32** may be a JavaScript-enabled browser such as a Netscape Navigator v.2.0 or above browser, or an Internet Explorer v.3.0 or above browser. If browser **32** is not equipped for such HTML scripting, the embedded instructions will not interfere with the browser's **32** normal processing, as such browsers **32** are typically configured to ignore any data they cannot interpret.

The embedded instructions transmitted to network client **12** may enable the user to manipulate some of the transcoding capabilities of transcoding server **34**. As illustrated in **Fig. 4**, the embedded instructions may drive a user interface in the form of a pop-up window **40** that is displayed at the top of a browser window **38**. Pop-up window **40** includes a three-state switch **42** having "ON," "OFF" and "AUTO" settings, and may also include a hypertext link **44** which the user may follow to download specialized client software supporting, for example, more

IPR2023-00330 Page 00954

sophisticated transcoding functionality (i.e., become "enabled"). The initial setting of three-state switch **42** may be based upon a prior determination by HTTP remote proxy **36** as to whether network client **12** has an established preference for reception of transcoded content. If so, three-state switch **42** may be set to "ON;" if not, three-state switch **42** may be set to "OFF." A goal of this feature is to provide the user with some means for communicating a preference to HTTP remote proxy **36** with regard to aspects of particular transcoding features, such as a content quality/latency tradeoff where the transcoding comprises data compression/scaling. Persons skilled in the art will recognize that many other means for providing this capability are possible, and such other means could enable the user to communicate preferences beyond simply a yes/no indication for transcoding.

In the illustrated in **Fig. 4**, pop-up window **40** enables the user to change his or her preference as to whether transcoded or original content is desired, and communicates such changes to HTTP remote proxy **36**. Pop-up window **40** may or may not interact with browser **32**, meaning the user's preference will only take effect after setting three-state switch **42** and clicking on the browser's **32** "RELOAD" button **46** to cause browser **32** to request the (transcoded or un-transcoded) content for presentation to the user. Subsequent pages in the current session may then be rendered in accordance with the new setting of three-state switch **42** without further user intervention. Upon receipt, HTTP remote proxy **36** may update user preference table **26** accordingly. As an alternative, pop-up window **40** may be configured to automatically invoke the "RELOAD" operation when the user indicates a change (such as by flipping three-state switch **42**). Where browser **32** is a JavaScript-enabled browser, JavaScript instructions inserted by HTTP remote proxy **36** in the HTML document may "POST" the state of three-state switch **42** to HTTP remote proxy **36** and also cause browser **32** to "RELOAD" the current URL.

It is possible to allow a non-enabled network client **12** to save the state of three-state switch **42** on network client **12** across multiple sessions of

IPR2023-00330 Page 00955

browser **32** using what is known in the art as a "cookie." In other words, a cookie may be used to store the state of three-state switch **42** persistently. When a new session of browser **32** is initiated by a user, this state information may be read from network client **12** and "POSTed" by the JavaScript code (inserted at the beginning of the HTML document) to HTTP remote proxy **36** before any content for the requested hypertext object is actually sent to network client **12**. This will allow HTTP remote proxy **36** to update user preference table **26** with the correct state of three-state switch **42**, and hence send correctly-transcoded content to network client **12**. In such an embodiment, the state information may be "POSTed" to HTTP remote proxy **36** each time a given URL is requested by browser **32**. This will allow network client **12** to receive the correctly-transcoded content even if the HTTP remote proxy **36** to which it is coupled changes due to, for example, a change in geographical location of network client **12** or network load-balancing procedures.

The embodiment shown in **Fig. 3** may also be used for network clients **12** that already access Internet **18** through a standard proxy. JavaScript-enabled browsers **32** may query the local IP address of network client **12** and "POST" this information to HTTP remote proxy **36**. The HTTP header of this "POST" message will contain the IP address of the standard proxy, which will now be different from the IP address of network client **12** (which is included in the contents of the message). A comparison of the two IP addresses will determine whether network client **12** resides behind a standard proxy. HTTP remote proxy may then use this information to update transcoding information about network client **12** in user preference table **26**.

According to another embodiment of the present invention, illustrated in **Fig. 5**, network client **12** may be "enabled," containing specialized software to support, for example, more sophisticated transcoding features than are provided by the above-described embodiments, or to perform some or all of the transcoding functions on the client side. As illustrated, network client **12** includes an HTTP local

IPR2023-00330 Page 00956

proxy **48** coupled to a client-side parser **50** which, similar to parser **22** of transcoding server **34**, controls one or more client-side transcode service providers **52**. Each transcode service provider **52** may be configured, for example, to transcode content before it is rendered to a user or to perform a counterpart transcoding function (e.g., decoding, decompression) with respect to a function performed by a corresponding transcode service provider **24** of transcoding server **34**. As in transcoding server **34**, network client **12** may include a client-side cache memory **56** managed by a client-side cache interface **54**. Client-side cache interface **54** may be an already-existing facility supported by the operating system, such as WININET. Using an existing caching facility reduces the amount of software that is to be downloaded to network client **12** to implement this embodiment, and also allows other applications, such as disconnected browsers, to share client-side cache memory **56**.

HTTP local proxy **48**, client-side parser **50** and client-side transcode service providers **52** (collectively, the client software) may be downloaded to network client **12** on demand, such as by clicking on hypertext link **44** presented by pop-up window **38** illustrated in **Fig. 4**. Alternatively, the client software could be distributed to users on a portable storage medium, such as a diskette or CD-ROM, or it may be preloaded on an off-the-shelf personal computer. In the embodiment of **Fig. 5**, the client software is separate from browser **32**; however, in yet another embodiment the client software may be integrated in browser **32** (see **Fig. 6**).

The enabled client embodiments provide network client **12** with expanded flexibility for rendering hypertext objects. As in the non-enabled client embodiments described above, enabled network client **12** may receive a transcoded data stream from HTTP remote proxy **36** in a format that is already supported by the standard internal rendering software of browser **32** (e.g., JPG, GIF). This would be the case where, for example, the transcoding process involved adding or deleting text to the hypertext object. In addition, HTTP remote proxy **36** may transcode a hypertext object to a data stream having a new MIME type, such as where the

IPR2023-00330 Page 00957

transcoding process comprised scaling or data compression, in which case a client-side transcode service provider **52** could be provided to convert the data stream back to a MIME type supported by browser **32**. For example, HTTP remote proxy **36** could transmit a file compressed using a non-standard, not well-supported but leading-edge compression algorithm to network client **12**, and client-side transcode service provider **52** could uncompress the file back to its original format. This approach has the benefit of relieving HTTP local proxy **48** from having to provide a user interface, and eliminates restrictions imposed by limitations as to the data types supported by browser **32**. In this way, the transcoding process can remain transparent to users, browsers and Web servers even when it involves changing content to different datatypes.

Yet another possibility is that enabled network client **12** includes one or more add-ins **46** specifically configured to transcode, render or playback content received by network client **12**. Add-ins **46** may be implemented, for example, using Netscape plug-ins or ActiveX controls. Moreover, add-ins **46** may be installed as part of the client software, as illustrated in **Fig. 5**, or integrated with browser **32**. Such add-ins **46** are beneficial in that they generally may be configured to permit a user to click on a specific object to obtain a different version (e.g., higher quality) representation. Add-ins **46** are also beneficial in that they appear to a user to be well-integrated with browser **32**, and are easily upgradeable. Combinations of the above-described presentation facilities are also possible.

In an advantageous optional application of add-ins **46**, network client **12** may be configured to request that an appropriate add-in **46** be downloaded from HTTP remote proxy **36** in the event that network client **12** determines it is unable to transcode a particular received data stream. HTTP remote proxy **36** could then download the necessary add-in **46** or, alternatively, resend the data stream in a different format. This facility provides for automatic extension of the system, ensuring that client software is as current as possible.

IPR2023-00330 Page 00958

In the embodiment of **Fig. 5**, browser **32** is configured to send all HTTP requests through HTTP local proxy **48**, thus allowing HTTP local proxy **48** to improve retrieval and rendering of requested hypertext objects. For example, when HTTP local proxy **48** receives an HTTP request from browser **32** for a hypertext object associated with a Web page, it passes the URL to client-side cache interface **54** to check whether a copy of the hypertext object already exists in client-side cache memory **56**. If the hypertext object is cached, HTTP local proxy **48** passes the cached object to browser **32** for rendering. If the requested hypertext object is not cached, HTTP local proxy **48** transmits an HTTP request to transcoding server **34** for processing. HTTP local proxy **48** may use a custom Get() request for this purpose to enable transcoding server **34** to identify network client **12** as enabled. Performing the processing described above with reference to other embodiments, transcoding server **34** will return a data stream for the hypertext object to HTTP local proxy **48**.

To further illustrate the features and benefits of embodiments of the present invention, the flow charts provided in **Figs. 7-9** illustrate the logic for an embodiment of a method by which an enabled network client may render a hypertext object resident on the Internet. The flow charts are not intended to be comprehensive of all processing that is performed, but rather are intended to describe the overall flow of the method. Detailed descriptions of the various processes have been provided above with reference to various disclosed embodiments. Where practical, the following description includes reference numbers for previously-described structural elements, although the method is not limited to those structures.

Referring now to **Fig. 7**, processing begins when a user on network client **12** requests a hypertext object from browser **32 (Step 100)**. This could be in the form of a request for a specific Web page, in which case a plurality of hypertext objects will likely be displayed to the user, or in the form of a click on an image already being displayed to the user. Browser **32** may be configured to pass all HTTP

IPR2023-00330 Page 00959

requests through HTTP local proxy **48**, so HTTP local proxy **48** may intercept the HTTP(URL) request from browser **32** (**Step 110**).

In this particular embodiment, HTTP local proxy **48** first checks whether the requested hypertext object exists in client-side cache memory **56** (**Step 120**). To do this, HTTP local proxy **48** may invoke client-side parser **50** using a GetScaledObject(URL) call, which in turn issues a GetEntry call to client-side cache interface **54** to open a stream for the cached object. This effectively "retrieves" the cached object from client-side cache memory **56** if it exists (**Step 140**). HTTP local proxy **48** then passes the stream to browser **32**, which displays the cached object to the user (**Step 150**).

Referring now to **Fig. 8**, if the requested URL object is not found in client-side cache memory **56**, HTTP local proxy **48** transmits a request for the object to transcoding server **34** using, for example, a Post of a GetStage(URL, Stage=0) call (**Step 160**). Upon receipt of this call, HTTP remote proxy **36** invokes parser **22**, which ir turn issues a GetScaledObject() call to server-side cache interface **28** to determine whether a non-transcoded version of the requested hypertext object already exists in the server-side cache memory **30** (**Step 170**). If the hypertext object is cached, server-side cache interface **28** issues a GetEntry call to open a stream for the cached object (**Step 200**). In addition, parser **22** may issue a GetProperties(URL, ...) call to server-side cache interface **28** to retrieve information about the transcoding properties and transcoded status (such as the refinement level) of the cached object.

If parser **22** determines that the requested hypertext object does not exist in the server-side cache memory **30**, HTTP remote proxy **36** issues an HTTP request to retrieve the hypertext object from Internet **18** (**Step 190**). If the object is not found, HTTP remote proxy **36** returns an error to network client **12** which browser **32** will communicate to the user (**Step 220**); if the object is found, HTTP remote proxy **36** passes the handle for the incoming data stream to parser **22**, which

IPR2023-00330 Page 00960

in turn initiates caching of an original version of the retrieved hypertext object (**Step 230**).

Referring now to **Fig. 9**, once the requested hypertext object has started to be obtained, parser **22** determines whether (and how) to transcode the object before transmitting it to network client **12** (**Step 240**). Both this decision-making process and exemplary transcoding processes are described in detail above. For purposes of the present illustration, assume parser **22** determined that transcoding was appropriate and therefore generated a transcoded version of the requested hypertext object (**Step 250**). HTTP remote proxy **36** transmits a data stream for the transcoded hypertext object to network client **12** (**Step 260**). Upon receipt, HTTP local proxy **48** initiates caching of the transcoded hypertext object (**Step 270**). In addition, client-side parser **50** determines whether any further processing is required before the hypertext object is rendered (e.g., a new MIME type has been established by transcoding server **34**) (**Step 280**).

If no additional transcoding is required, HTTP local proxy **48** passes the handle for the received data stream to browser **32** for display to the user (**Step 290**). If additional transcoding is required, client-side parser **50** passes the handle to an appropriate transcode service provider **52** (**Step 300**). The result of this latter processing may be a hypertext object which browser **32** can readily display to the user (**Step 320**), or the result may be a hypertext object having a non-standard MIME type, in which case browser **32** may invoke add-in **46** to display the object (**Step 330**).

According to another embodiment of the present invention, additional data or programs need not necessarily be inserted as part of a response to a client request. Rather, data and programs may be transparently "pushed" to network client **12** without the user or the browser **32** software's detection or intervention. One advantage of this approach is that transcoding server **34** is able to detect when

IPR2023-00330 Page 00961

client/server communications link **14** is underutilized, and can thus push data to network client **12** with limited risk of interfering with other transactions. An especially advantageous implementation uses at least a local proxy, which could issue its own requests (rather than being user-driven) to content providers or networked proxy servers, or receive unsolicited data pushed to it from the network. The local proxy may store the data in a client-side cache, install it as a program, or prompt the user to take some further action. Many potential uses for such an embodiment are possible. For example, an advertiser of software products or music can preload network client **12** with trial versions of products before prompting the user with an advertisement, thus enabling instant playback capability without the user having to wait for a demo to be downloaded (and possibly losing interest in the meantime).

A number of different configurations are possible for implementing embodiments of the present invention. In a first configuration, the only additional apparatus required is a remote proxy. That is, no new software needs to be installed on network client **12**. The remote proxy may reside anywhere on a suitable network, such as the Internet, including at particular content provider sites. Alternatively, the remote proxy may be located at ISP local POPs (Point of Presence), for example, if location-specific characteristics are to be used as predetermined selection criteria. Of course, such information can be gathered by other methods as well, such as user-preference settings or assigning location-specific domain names to proxies. In a second configuration, a new piece of client software acting as a local proxy may be installed, for example, on a client device. The user would then point the client application's proxy to the local host. Combinations of these exemplary configurations are likewise possible, as well as simultaneously having multiple modes active (for example, a local proxy acting as a pass-through for some requests and a non-pass-through for others that require the use of a remote proxy).

Where network client **12** connects to a remote proxy over a relatively slow communications link, it may be particularly advantageous to implement

IPR2023-00330 Page 00962

transcoding and link validity checking on remote proxies. Combinations of remote and local proxies can sometimes give more efficient implementations of certain applications, such as automatic data/program download and interactively displaying predigested content. Other applications, such as translation and trademark enforcement, can be done efficiently on local proxies alone, but may be more advantageously done on remote proxies because the results can be cached for use by others, thereby saving resources for future requests. Still other applications, such as clickstream analysis, are generally better implemented on a local proxy because there are more resources available locally to the individual user, and also for privacy reasons.

In view of the foregoing description, it should be apparent that it is possible for there to be more than one so-called "smart" proxy arranged between a client device and a content server device. If left unchecked, such a condition can result in content being altered excessively (for example, too many ads inserted, multiple lossy compressions resulting in indecipherable images). To address this problem, an embodiment of the present invention may use a special proxy-to-proxy protocol that extends the existing request/response structures to indicate whether and what sort of transcoding has already been performed on the content. Such a specialized protocol, in addition to other proxy-to-proxy messages which may be implemented on an as-needed basis, enables multiple proxies to work collaboratively, yet still transparently to users, client software, existing "standard" proxies and content servers.

According to yet another embodiment of the present invention, a proxy server may be used to provide certain Internet proxy or server users with so-called "VIP" treatment, identifying users who are entitled (either through payment or based on some other selection criterion, such as extent of usage) to have a higher priority when competing with other users for proxy resources. By contrast, with existing Internet proxies and servers, users are serviced either on a random or first-come/first-served basis.

IPR2023-00330 Page 00963

In one particular implementation of such an embodiment, transcoding server **34** may be configured to extract user IP addresses from requests it processes and maintain information such as how frequently, or for what duration, a user is browsing a particular Web site. Such information could be used to determine "frequent browser miles" at particular Web sites. Users can then be rewarded with faster response times for subsequent visits to the site, or the site owner could choose to reward the user with improved performance on all sites reached through the same proxy. Still another possibility is that users may pay for such preferred service, being assigned a password which may be provided to transcoding server **34**. Yet another possibility is that a Web site owner can pay a proxy provider to improve the performance of all users while visiting the owner's site.

In another particular implementation, information identifying users to be given VIP treatment may be passed to transcoding server **34** in the form of a Web page. Upon receipt of such a Web page, the proxy may subsequently allow servicing threads to perform work for requests generated by VIP users first. To do this, transcoding server **34** may boost thread scheduling priorities (within the operating system) for the VIP service, while ensuring there is no starvation of any thread (that is, no user should be denied access entirely by VIP users). In addition, transcoding server **34** may permit preferential caching for particular Web sites and more aggressive pre-fetching for VIP users. Still further, transcoding server **34** may use more resource-intensive compression algorithms, for example, to provide better quality content for the same latency at the expense of slowing down access for non-VIP users.

It is possible that certain content providers or users will not wish to have their content dynamically altered in any manner. Accordingly, embodiments of the present invention may be implemented in such a way that either content providers or users are given the capability to override any potentially content-altering service.

IPR2023-00330 Page 00964

This may be accomplished, for example, using a pass-through technique triggered by a special tag embedded within the content.

As the foregoing description demonstrates, embodiments of the present invention may be used to provide a system for improving the communications capabilities of computers accessing networks such as the Internet. Embodiments of the invention may be advantageously applied to computers having limited communications bandwidth available, such as mobile computers or personal computers accessing a network over a modem connection. The unique features of such embodiments enhance the ability of these computers to access data on the network in a timely fashion with reduced user-visible latencies, thereby enabling content authors to produce rich content without fear that only users with highly-sophisticated data communications and display capabilities are able to enjoy it. Embodiments of the present invention may also be advantageously used for purposes other than, or in addition to, reducing latency. Such purposes include, for example, converting color images to greyscale images for users lacking a color display; filtering and/or deleting undesired content, such as pornography; adding content, such as advertising; and language translation.

Although the present invention has been described with reference to embodiments for accessing data from the Internet, persons skilled in the art will recognize that it is equally applicable to other networking environments. For example, embodiments of the present invention may be used to enhance data communications between a network client computer and an "intranet." An intranet typically is a secure corporate network modeled after the Internet architecture, and generally includes mechanisms for communicating with external networks such as the Internet.

The foregoing is a detailed description of particular embodiments of the present invention. The invention embraces all alternatives, modifications and variations that fall within the letter and spirit of the claims, as well as all equivalents

IPR2023-00330 Page 00965

of the claimed subject matter. For example, some or all of the features described above as being provided by a remote proxy may be implemented in a content server. Likewise, some or all of the features described above as being provided by a local proxy may be implemented in a browser application. Persons skilled in the art will recognize from the foregoing detailed description that many other alternatives, modifications and variations are possible.

IPR2023-00330 Page 00966

What Is Claimed Is:

1. An apparatus for use in transmitting data between a network server and a network client over a communications link, said apparatus comprising a parser coupled to a transcode service provider, said parser being configured to selectively invoke said transcode service provider in response to a predetermined selection criterion.

2. The apparatus of claim 1, wherein said predetermined selection criterion comprises a characteristic of the data being transmitted.

3. The apparatus of claim 1, wherein said predetermined selection criterion comprises a characteristic of the communications link.

4. The apparatus of claim 1, wherein said predetermined selection criterion comprises a characteristic of the network server.

5. The apparatus of claim 1, wherein said predetermined selection criterion comprises a characteristic of the network client.

6. The apparatus of claim 1, wherein said predetermined selection criterion comprises a user preference.

7. The apparatus of claim 1, wherein data is transmitted from the network server to the network client in response to a request by the network client, said predetermined selection criterion being included in said request.

8. A method for providing a network client with a data object residing on a network server, wherein the network client and the network server are coupled by a communications link, said method comprising the steps of:

IPR2023-00330 Page 00967

receiving a data object from the network server;

selectively transcoding the data object according to a predetermined selection criterion; and

providing the data object to the network client.

9. The method of claim 8, wherein said transcoding step further comprises comparing a characteristic of the received data object to the predetermined selection criterion.

10. The method of claim 8, wherein said step of selectively transcoding the data object further comprises determining whether the data object includes content created with an unregistered software product.

11. The method of claim 10, wherein said step of selectively transcoding the data object further comprises adding a message to the data object corresponding to said detection of content created with an unregistered software product.

12. The method of claim 8, wherein said step of selectively transcoding the data object comprises compressing a portion of the data object.

13. The method of claim 8, wherein said step of selectively transcoding the data object comprises translating a portion of the data object from a first language to a second language.

14. The method of claim 8, wherein said step of selectively transcoding the data object further comprises determining whether the data object includes offensive content.

15. The method of claim 14, wherein said step of selectively transcoding the data object further comprises modifying the data object to prevent offensive content from being rendered by the network client.

IPR2023-00330 Page 00968

16. The method of claim 8, wherein said step of selectively transcoding the data object further comprises adding advertising information into the data object.

17. The method of claim 16, wherein said advertising information is selected in accordance with user profile information.

18. The method of claim 8, wherein said step of selectively transcoding the data object further comprises determining whether the data object includes a link to a second data object.

19. The method of claim 18, further comprising the step of validating the link to a second data object.

20. The method of claim 19, wherein said step of selectively transcoding the data object further comprises correcting an invalid link.

21. The method of claim 8, wherein said step of selectively transcoding the data object further comprises communicating information relating to said transcoding to the network server.

22. The method of claim 8, wherein said step of selectively transcoding the data object further comprises determining whether the network client is preconfigured to receive preferential treatment of requests.

23. A set of instructions residing on a storage medium for execution by a computer, the computer being coupled to a device for rendering a data object to a user, said set of instructions comprising instructions for:

parsing a data object to be rendered to detect content corresponding to a predetermined selection criterion;

IPR2023-00330 Page 00969

selectively transcoding the data object in response to said detection prior to rendering the data object.

24. The set of instructions of claim 23, wherein the storage medium comprises a magnetic storage device.

25. The set of instructions of claim 23, wherein the storage medium comprises a memory installed in a computer.

IPR2023-00330 Page 00970

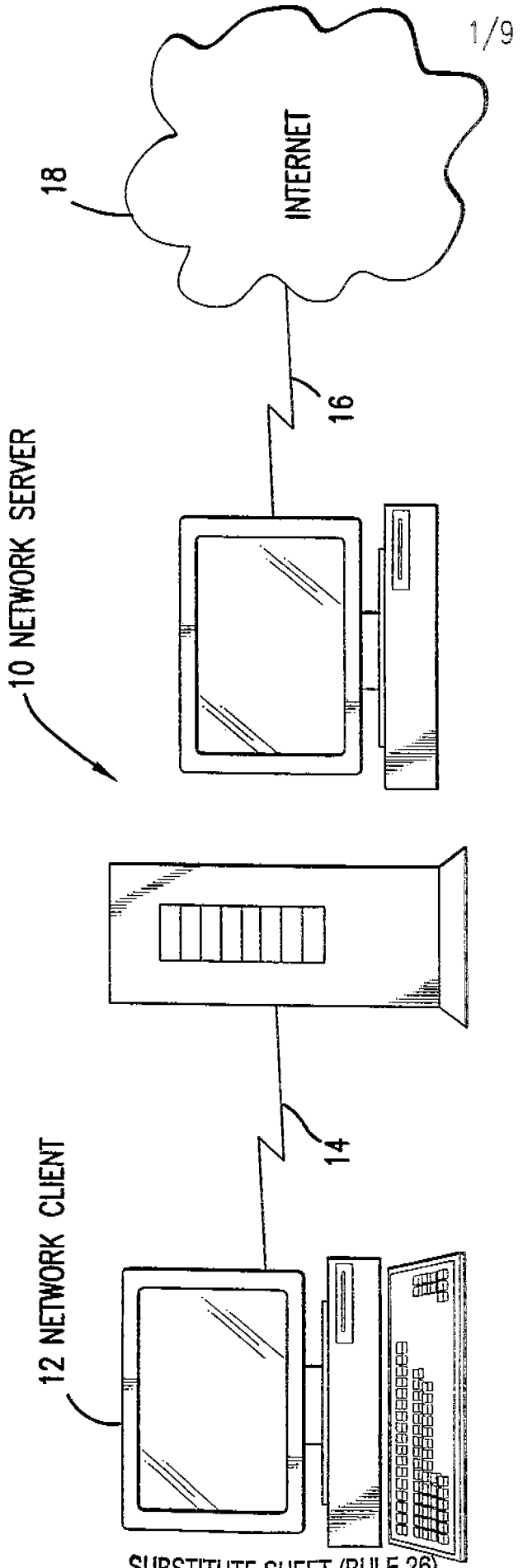


FIG. 1

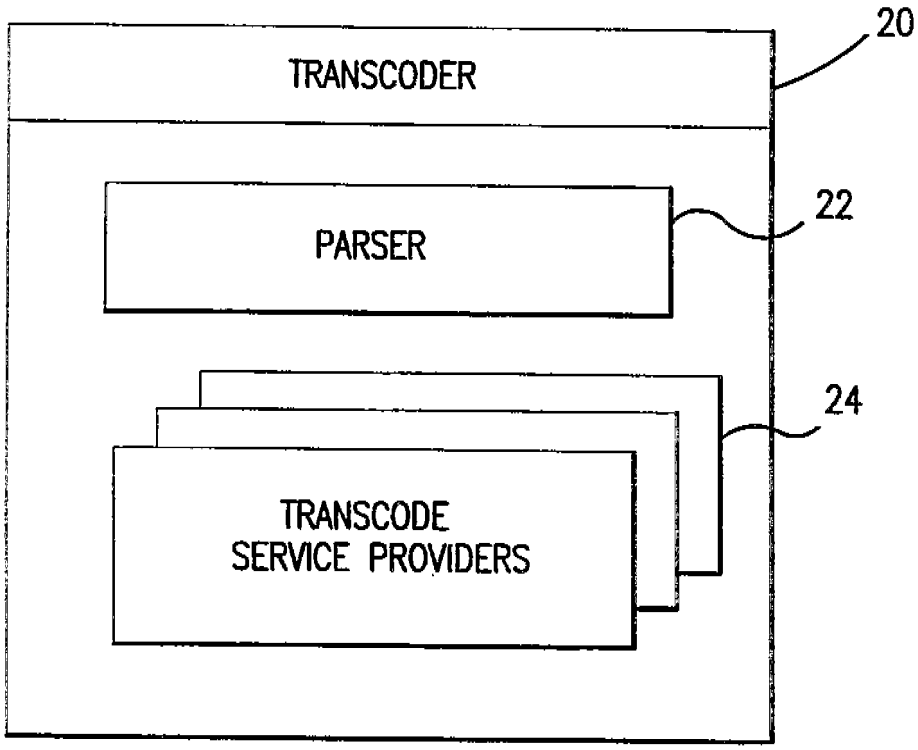


FIG.2

IPR2023-00330 Page 00972

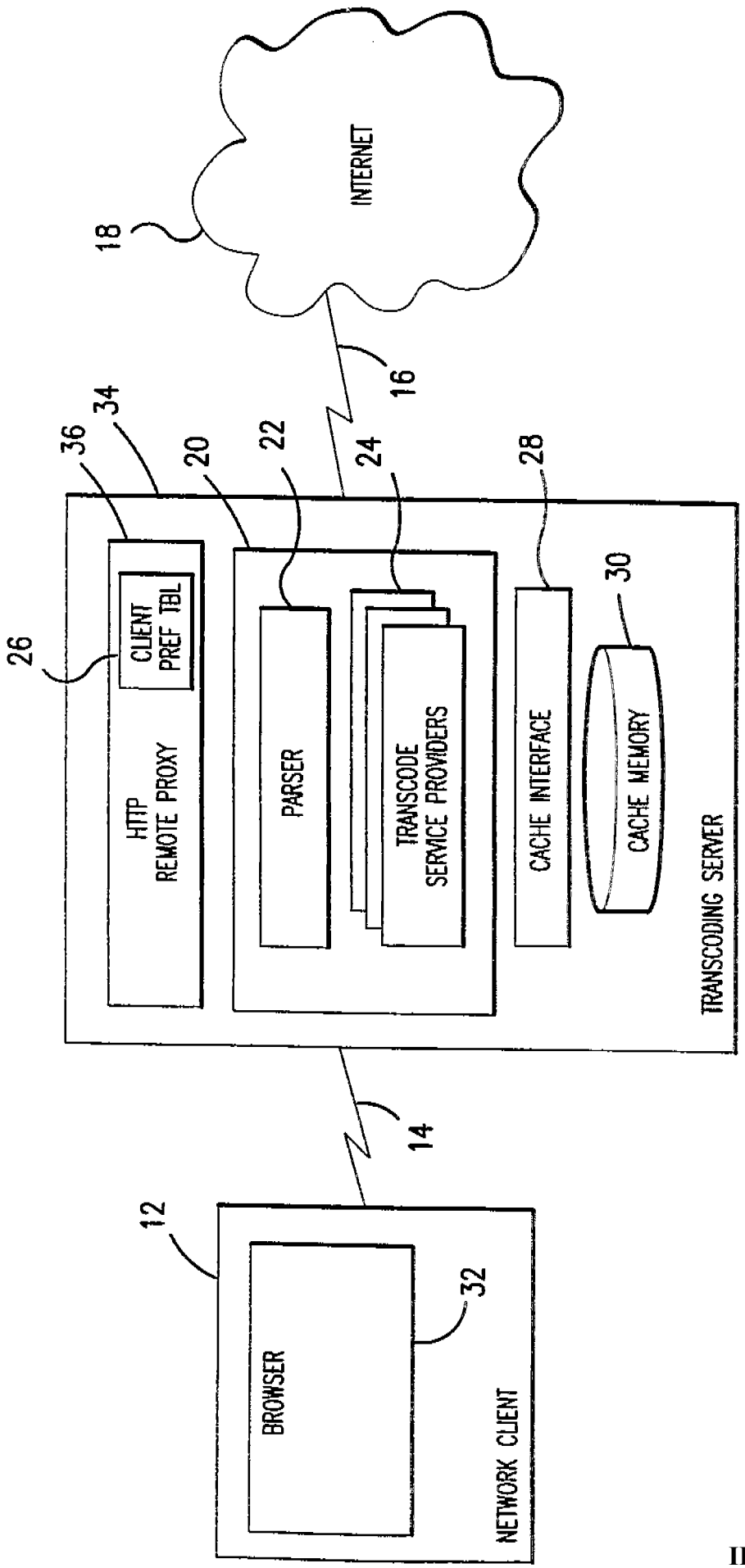

INTERNET
18
16
34
36
20
22
24
28
26
CLIENT PREF TBL
HTTP REMOTE PROXY
PARSER
TRANSCODE SERVICE PROVIDERS
CACHE INTERFACE
30
CACHE MEMORY
TRANSCODING SERVER
14
12
BROWSER
32
NETWORK CLIENT

FIG.3

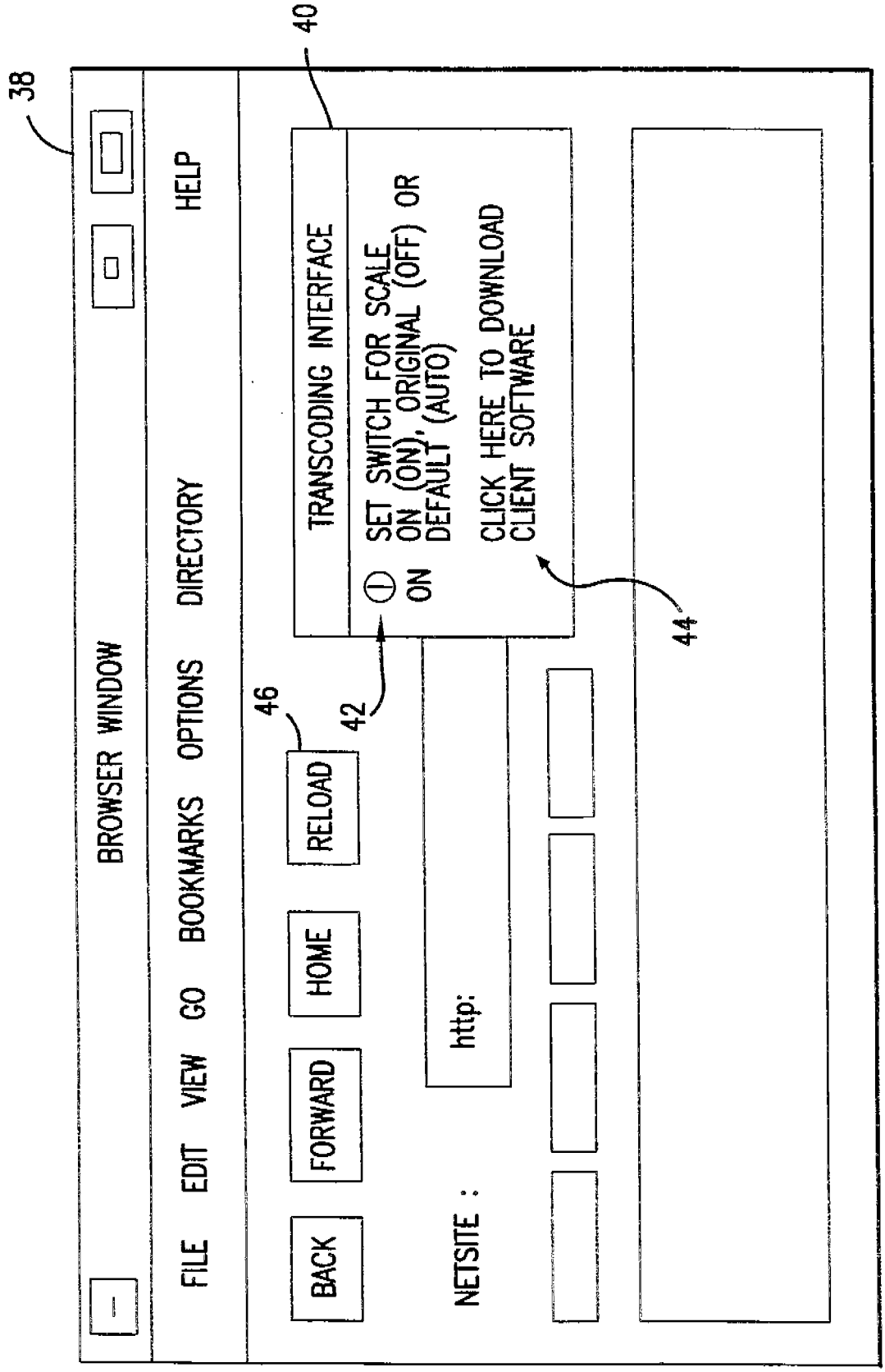


FIG.4

IPR2023-00330 Page 00974

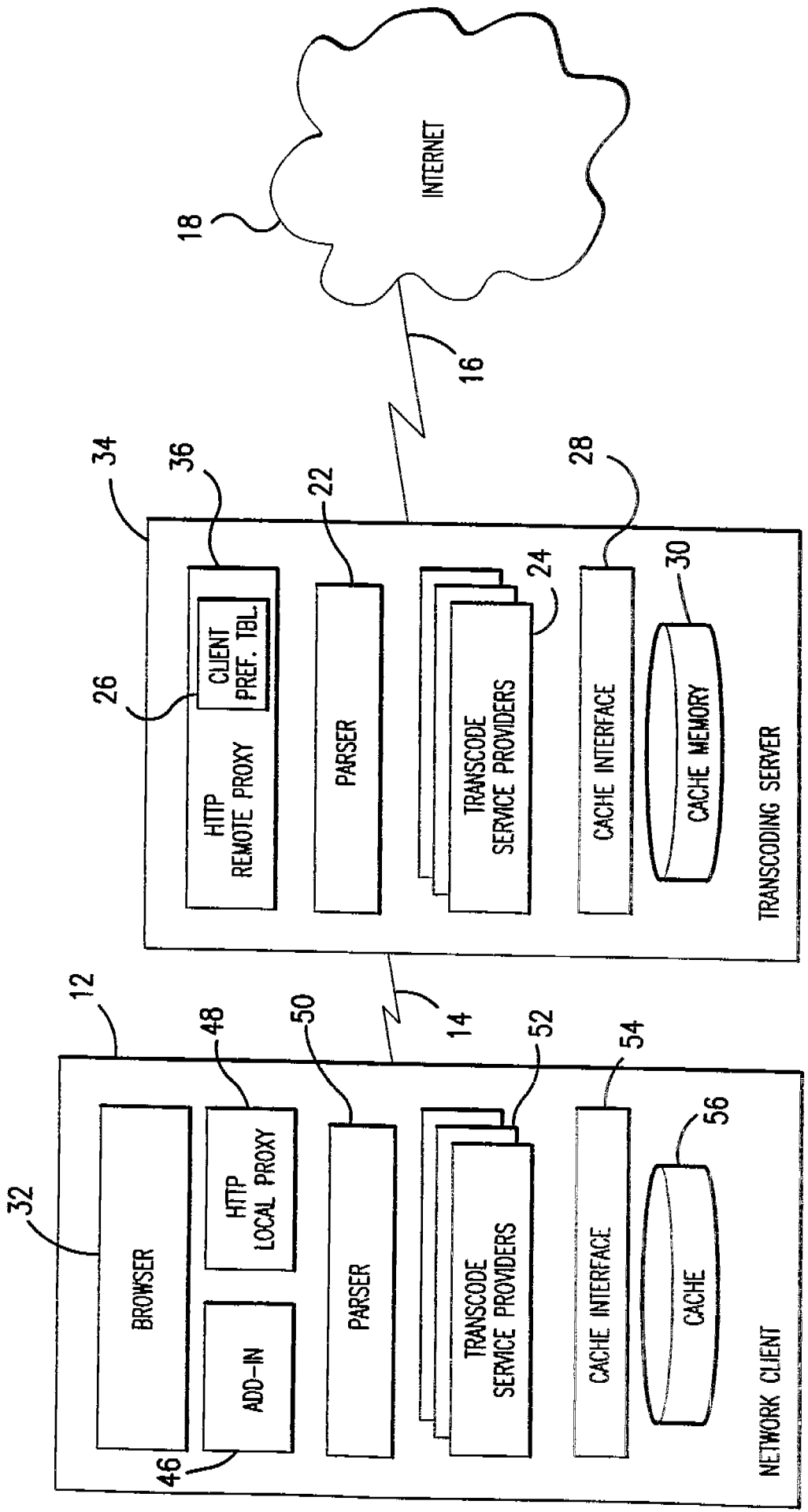

INTERNET
18
16
34
36
22
28
CLIENT PREF. TBL
HTTP REMOTE PROXY
26
PARSER
TRANSCODE SERVICE PROVIDERS
24
CACHE INTERFACE
CACHE MEMORY
30
TRANSCODING SERVER
12
14
48
50
52
54
56
32
BROWSER
HTTP LOCAL PROXY
ADD-IN
46
PARSER
TRANSCODE SERVICE PROVIDERS
CACHE INTERFACE
CACHE
NETWORK CLIENT

FIG.5

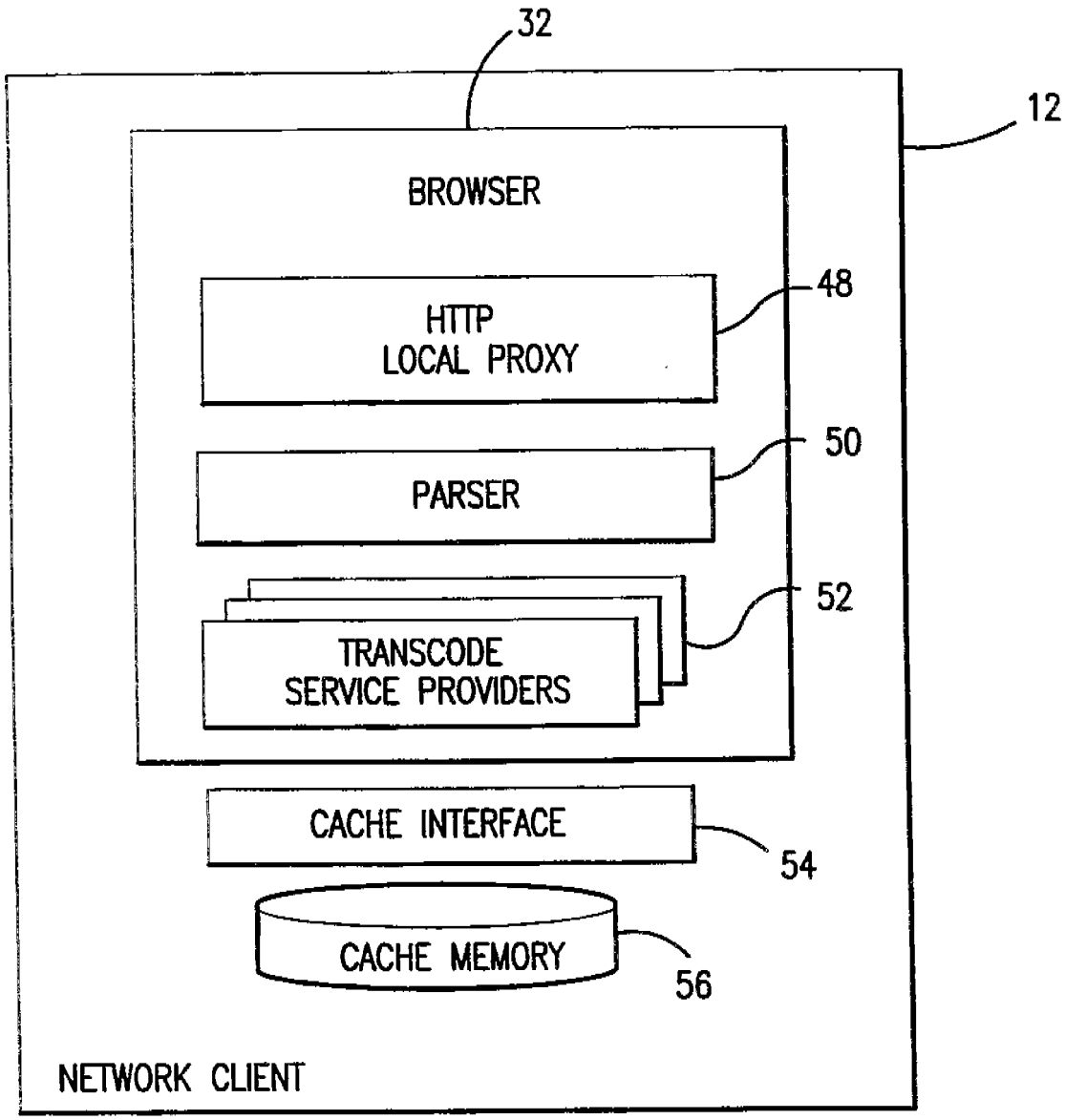

32
12
BROWSER
48
HTTP
LOCAL PROXY
50
PARSER
52
TRANSCODE
SERVICE PROVIDERS
CACHE INTERFACE
54
CACHE MEMORY
56
NETWORK CLIENT

FIG.6

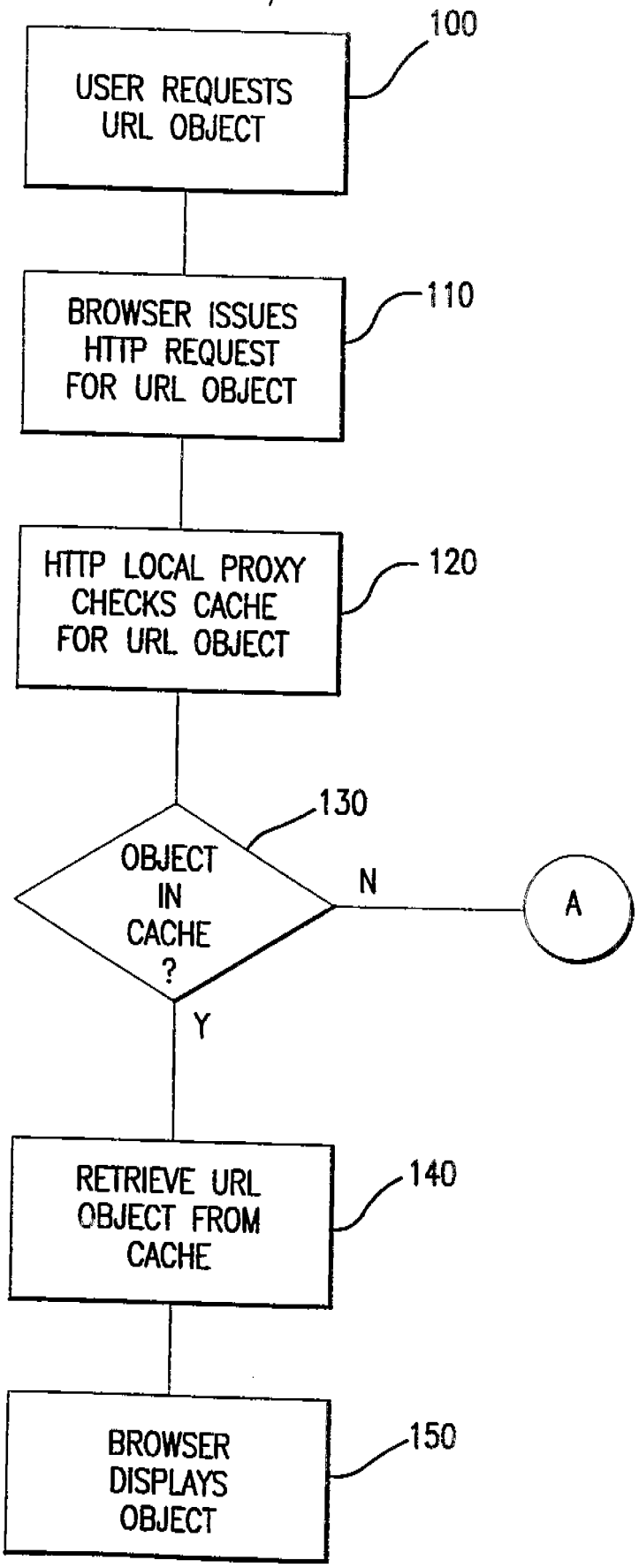


FIG.7

IPR2023-00330 Page 00977

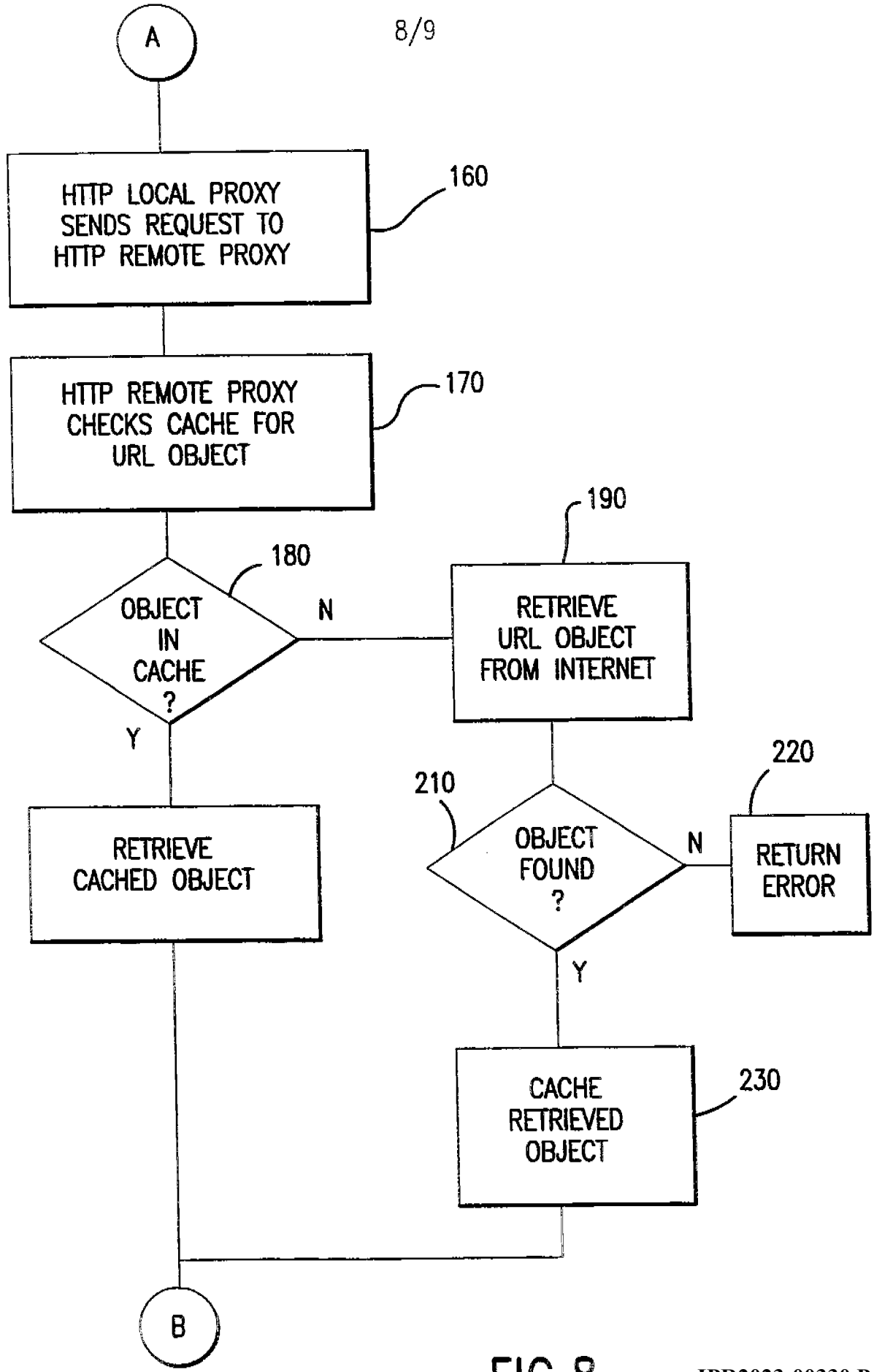

A
HTTP LOCAL PROXY SENDS REQUEST TO HTTP REMOTE PROXY
160
HTTP REMOTE PROXY CHECKS CACHE FOR URL OBJECT
170
190
180
OBJECT IN CACHE ?
N
RETRIEVE URL OBJECT FROM INTERNET
Y
220
210
RETRIEVE CACHED OBJECT
OBJECT FOUND ?
N
RETURN ERROR
Y
CACHE RETRIEVED OBJECT
230
B

FIG.8

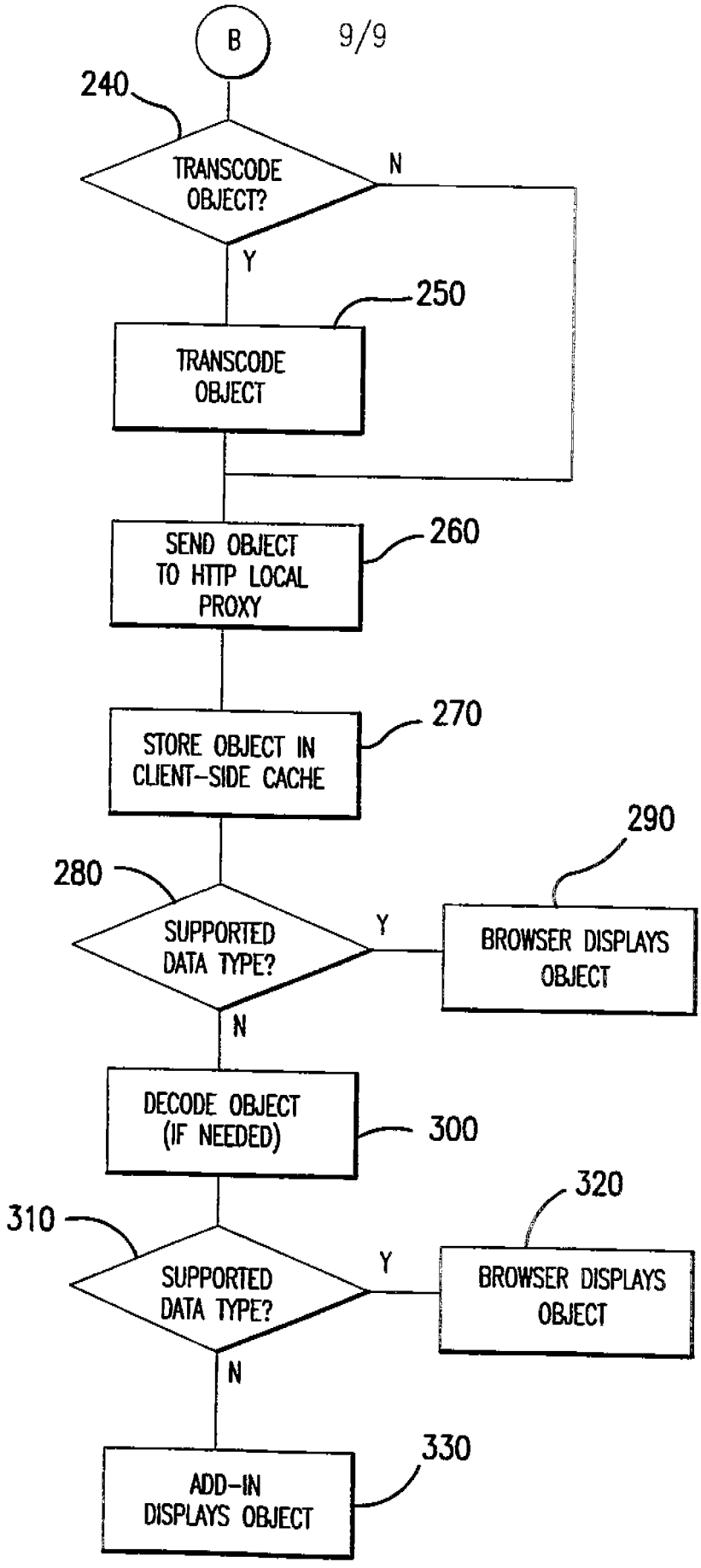

B
240
TRANSCODE OBJECT?
N
Y
250
TRANSCODE OBJECT
SEND OBJECT TO HTTP LOCAL PROXY
260
STORE OBJECT IN CLIENT-SIDE CACHE
270
290
280
SUPPORTED DATA TYPE?
Y
BROWSER DISPLAYS OBJECT
N
DECODE OBJECT (IF NEEDED)
300
320
310
SUPPORTED DATA TYPE?
Y
BROWSER DISPLAYS OBJECT
N
330
ADD-IN DISPLAYS OBJECT

FIG.9

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US98/05304

A. CLASSIFICATION OF SUBJECT MATTER

IPC(6) :G06F 13/38, 15/17

US CL :395/200.76, 200.48, 200.59; 345/335

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 395/200.76, 200.48, 200.59, 200.33, 200.47, 200.32, 200.58, 200.77; 345/335

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

NONE

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

USPTO APS

search terms: WWW, URL, HTML, HTTP, parser, configure, data object, selection criteria, forward

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 5,544,320 A (KONRAD) 06 August 1996 Abstract; Fig. 2; C16:14-C17:18	1-25
Y, P	US 5,706,434 A (KREMEN et al.) 06 January 1998 Abstract; Fig. 1, 5, 6; C5:21-59; C10:23-C11:50	1-25
A, P	US 5,724,556 A (SOUDER et al.) 03 March 1998 Abstract; Fig. 6, 9	1-25
X, E	US 5,768,510 A (GISH) 16 June 1998 Abstract; Fig. 6-7, 11-13, 26; C5:24-50; C18:60-C20:18; C22:49-C23:38; C25:66-C26:20	1-25

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier document published on or after the international filing date	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&"	document member of the same patent family
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search

03 JULY 1998

Date of mailing of the international search report

13 AUG 1998

Name and mailing address of the ISA/US
Commissioner of Patents and Trademarks
Box PCT
Washington, D.C. 20231

Authorized officer

MARK H. RINEHART

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : H04Q | A2

(11) International Publication Number: **WO 97/49252**

(43) International Publication Date: 24 December 1997 (24.12.97)

(21) International Application Number: PCT/US97/10758

(22) International Filing Date: 20 June 1997 (20.06.97)

(30) Priority Data:
60/020,094 21 June 1996 (21.06.96) US

(71) Applicant *(for all designated States except US)*: INTEGRATED COMPUTING ENGINES, INC. [US/US]; 460 Totten Pond Road, Waltham, MA 02154 (US).

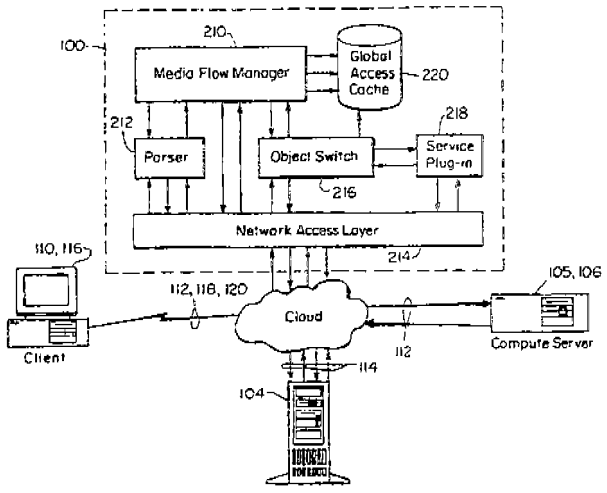
(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: SHAH, Ashesh, C. [US/US]; 567 Tremont Avenue, No. 31, Boston, MA 02118 (US). PEDERSEN, Palle [DK/US]; 82 Commonwealth Avenue, No. 10, Boston, MA 02116 (US). RADOVIC, Niksa [HR/US]; 19 Mountain Avenue, Somerville, MA 02143 (US). MANICKAVASAGAM, Senthilkumar [IN/US]; 11 Highland Glen Drive, No. 17, Randolph, MA 02368 (US).

(74) Agents: SMITH, James, M. et al.; Hamilton, Brook, Smith & Reynolds, P.C., Two Militia Drive, Lexington, MA 02173 (US).

(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, HU, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, TJ, TM, TR, TT, UA, UG, US, UZ, VN, YU, ARIPO patent (GH, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

Published
Without international search report and to be republished upon receipt of that report.

(54) Title: NETWORK BASED PROGRAMMABLE MEDIA MANIPULATOR



(57) Abstract

The media manipulator is a middle layer between the clients, and the remote data servers is the common client-server organization. It transforms the network into a more flexible three-tiered configuration. Requests generated by the clients for media objects from media resources are routed to the media manipulator. It processes the requests and determines if the media objects may be found locally, either cached in the media manipulator itself or in the local data servers. When the media objects are obtained, the media manipulator can be used to perform operations on those objects such as format translations, to apply protective mechanisms for the clients, to speed communications between the remote servers and the clients, or perform compute operations for the clients. In one example, a parser of the manipulator searches for images in the media objects so that service devices can be called to perform data compression or pornography detection on the images. The parser can also search for executable or data files in the media objects and to perform virus scanning or format conversion, respectively.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

NETWORK BASED PROGRAMMABLE MEDIA MANIPULATOR

RELATED APPLICATIONS

This application claims priority to U.S. Provisional Application No. 60/020,094, filed June 21, 1996, the contents of which is incorporated herein by reference in its entirety.

BACKGROUND OF THE INVENTION

In a client-server network, on one hand there are clients, typically personal computers, IBM-compatible computers and/or UNIX workstations, for example, equipped with information browsers. On the other hand, there are data servers and compute servers. Data servers are computers with a large storage capacity containing information in different media formats: data records, plain text documents, word processing documents, still pictures, compressed audio and video, and executable files, for example. Compute servers are computers that carry out intensive computational tasks that would typically require too much time for the client to complete. Each compute server might use a single or many processors to complete the given task.

Users interact with their clients in a natural way with a mouse, keyboard, screen, printer, or by some other input/output device. The users need not be concerned about what happens after they make their selection within their clients. Clients then make service requests to geographically dispersed servers. Upon receiving requests from the clients, the servers

perform the desired operations and return the retrieved or computed media stream back to the client for display.

SUMMARY OF THE INVENTION

The present invention is connected into the ubiquitous two-tiered client-server network of computers. It is designed as a middle layer, middleware, between the clients and the remote data servers. It transforms the network into a more flexible three-tiered configuration. Requests generated by the clients for media objects from media resources are routed to the media manipulator. It processes the requests and determines if the media objects may be found locally, either cached in the media manipulator itself or in local/remote data servers. When the media objects are obtained, the media manipulator can be used to perform operations on those objects such as format translations, to apply protective mechanisms for the clients such as virus scanning, to speed communications between the remote servers and the clients using compression operations, or perform compute operations for the clients.

In general, according to one aspect, the invention features a middle-ware computing system. It includes a network access system that supports communications with media resources and client computers and a media manipulation system that operates on media objects received from the media resources via the network access system prior to forwarding the media objects to the client computers.

In specific embodiments, a parser is used to identify different media types within the media objects

so that service devices may be called to operate on the media types. In one example, the parser searches for images in the media objects and service devices include an image compressor for performing data compression or pornography detection on the images. The parser can also search for executable or data files in the media objects and the service devices then called to perform virus scanning or format conversion, respectively.

In further specifics, a cache is used to store media objects. A media flow manager receives requests for media objects and checks for the presence of the media objects in the cache to preclude the necessity of obtaining the objects from the remote media resources.

The above and other features of the invention including various novel details of construction and combinations of parts, and other advantages, will now be more particularly described with reference to the accompanying drawings and pointed out in the claims. It will be understood that the particular method and device embodying the invention are shown by way of illustration and not as a limitation of the invention. The principles and features of this invention may be employed in various and numerous embodiments without departing from the scope of the invention.

BRIEF DESCRIPTION OF THE DRAWINGS

In the accompanying drawings, reference characters refer to the same parts throughout the different views. The drawings are not necessarily to scale; emphasis has instead been placed upon illustrating the principles of the invention. Of the drawings:

Fig. 1 is a schematic block diagram illustrating the context in which the inventive media manipulator operates;

Fig. 2 is a block diagram illustrating the interaction between components of the media manipulator according to the invention;

Fig. 3 is an object interaction diagram illustrating the operation of the components of the media manipulator;

Figs. 4A, 4B, and 4C show the message formats for transmitting tasks to compute servers;

Fig. 5 is a block diagram showing the programming of the media manipulator using m-script;

Fig. 6 is another object interaction diagram showing the order of creation of the components of the manipulator; and

Fig. 7 is a block diagram showing another embodiment of the media manipulator.

DETAILED DESCRIPTION OF THE DRAWINGS

Fig. 1 illustrates the context in which the media manipulator 100 operates. In many applications, it is important for users to access remote media resources 108 such as the data servers 104 of content providers on the Internet. These users may be at client computers 110 that are inter-connected by a local area network (LAN) 112. The clients 110 access the media resources 108 through a gateway 114 linking the LAN 112 to the Internet. The user's also require access to the media resources 108 remotely at remote clients 116 through, for example, telephone dial-up connections 118 or through cellular/wireless links 120.

The media manipulator 100 is connected into this two-tiered client-server network of computers as a

middle layer between the clients 110, 116 and the remote data servers 104 of the media resources 108.

Fig. 2 is a block diagram illustrating the internal organization of the media manipulator 100. It comprises six basic components: media flow manager 210, media parser 212, network access layer 214, object switch 216, multiple service plugins 218, and global access cache 220. In one embodiment, these components are implemented as separate software objects that run on a common microprocessor or multiprocessor system.

The media flow manager 210 serves as the principle controller for the media manipulator 100. It has access to the various components and can alter their behavior. It specifies the operations to be formed on the received media objects. It is also the storehouse for the information on the media objects as they are received from the media parser. The media flow manager 210 also tracks the physical resources that are functional and available in the media manipulator and on the surrounding LAN in order to determine to which of the resources the media objects flow.

The network access layer 214 makes the media manipulator 100 accessible through many different types of network devices and the protocols running on top of them. In one implementation, the network access layer communicates through the Internet gateway 114 using the TCP/IP protocol, connects to the compute or data server using the protocol of the local area network 112, and communicates with the clients using either the LAN or the protocols necessary to communicate with the remote clients 116 over low-bandwidth connections 118, 120.

When communicating with the remote data servers 104 of the content providers, the network accepts the incoming data streams and assembles them into media objects. These media objects are then made available to the media parser 212, object switch 216, and service plugins 218. The media parser 212 analyses all incoming media objects to extract the relevant media types. These media types include executable files, data files, and images, for example. Information concerning the detected media types is forwarded to the media flow manager 210, which decides what operations should be performed on the media.

The object switch 216 supports a number of incoming and outgoing object gates. Media objects enter into the object switch from the network access layer 214 and from the service plugins' output links. The media objects leaving the object switch 216 go into the network access layer 214 and the service plugins' input links. The object switch routes the objects based on the media manager instructions, either directly or indirectly.

The global access cache is an intelligent mechanism that speeds the operation from the perspective of the user at the clients 110, 116. It determines which media objects are most likely to be used in the future and stores them in the fastest available memory. Media objects that are somewhat less likely to be required again are stored in slower memory or a secondary cache. There can be as many levels of the cache as the physical infrastructure allows, and the caching may take place on data servers that are remote from the main computational resources of the media manipulator. This caching minimizes the time

that different users need to wait for requests to be processed.

The media manipulator 100 is a programmable device. A system administrator can change its behavior by giving it m-script commands. It is also an extendable device. By adding new service plugins, new capabilities can be added to the device. The construction of the components of the media manipulator allows for redundancy and fault tolerance. A hardware failure does not bring the entire system to a halt. The system will keep working and simply notify the administrator that one of its components needs to be replaced.

Fig. 3 is an object flow diagram illustrating the communication between the client 110, 116, content provider's data server 104, and the components of the multimedia manipulator 100.

The first step is the initial connection 1, Connect, between the client 110, 116 and the media manipulator 100 via the network access layer 214. The network access layer 214 accepts this request 2. In one implementation, it accepts by calling a new incidence of itself such that each incidence of the network access layer object supports a single connection outside the media manipulator 100.

After establishing the connection, the client makes a request 4 for a media object. In the typical example, this will be a universal resource locator (URL) to a data server 104 of a content provider on the Internet. The network access layer then calls the media parser 212 and passes the client request 6.

The media parser 212 looks to two sources for the media object simultaneously. The ProcessURL request 8 is passed to the media flow manager 212, which has knowledge of the contents of the global access cache 220. The parser also issues a request 10, GetPage, to the network access layer.

The media flow manager 212 searches for the object in the cache 220. If the cache returns a cache-miss, the request to the provider has not been delayed waiting for the miss status, whereas in the case of a cache-hit, the request to the provider is simply terminated after verifying the validity of the cached page. Using this scheme, there is little increased latency associated with the use of the manipulator 100 in the worst-case cache-miss scenario.

In the illustrated example, a cache-miss occurred. Thus, rather than supplying the object, the cache 220 is prepared 12 to receive the media object, PutInCache. Also, the network access layer 214 connects 14 to the content provider and retrieves 16 the media object or page.

As the media object is being received by the network access layer from the content provider 104, the parser begins to parse 17 the object. As parsing proceeds, the parser also begins to update 18 the global access cache 220 with the parsed portions of the object. Simultaneously, the parser begins the reply 20, 22 to the client via the network access layer.

In one implementation, the parser searches for images in the media objects to perform compression or pornography detection, for example. On encountering an

image, the parser 214 passes a call to the media flow manager to process the image 24 while continuing to parse 26 the media object.

The media flow manager 210 gets the image 28 via the network access layer 214. The fact images are not stored with the page but must be separately requested is an artifact of the HTTP protocol. The network access layer 214 then connects 30 to the content provider 104 and retrieves 32 the image.

When the image is retrieved, the media flow manager 210 places it in the cache with the other portions of the media object and makes a function call to the object switch to process the image 34. The object switch knows the various service plugins that are available and the actions that must be performed on the media types that are discovered by the media parser, which in this example is an image. When called by the object switch 216 to process 36 the media type, the particular service plugin, or multiple plugins when serial operations are required, retrieves 38 the media type, *i.e.*, image, and performs the desired operation on or processes 40 the image. For example, in one instance, this can be compression or thinning to expedite communication to the client. In another case, it can detect the probability of pornography by detecting the percentage of flesh-tone colors in the picture. Once the processing is complete, the new image or revised media object may be placed 42 in the cache 220 or used in a reply to the client 110, 116.

In many instances, the service plugin functionality will be performed by a separate compute

server 105. This computer may be directly accessible by the media manipulator 100 or accessible through the local area network 112. Generally out-sourcing this functionality is desirable, rather than running on the same device with the other components of the media manipulator 100, to avoid depriving those other components of processing bandwidth.

When the plugin does utilize the external compute server, it issues a request message. Fig. 4A illustrates the formatting of the message to the compute server. The message has a number of different fields. It has a version field and a length field defining the length of the content. The type field indicates the type of the message, and the message ID is assigned by the network access layer. The source type indicates the media type. In the context of image files, the type indicates whether the image is in a GIF or JPEG type compression format, for example. The source path is the path to where the image is stored in the global access cache 220, to which the compute server has access. The destination type, path length, path, and parameters define the transformed media type and where it is to be sent.

Fig. 4B illustrates the reply message from the compute server. It again has version, length, type, and message ID fields. The reply code indicates whether or not the service was successful. The destination type, path length, and path indicate the type of the final image after the transform of the compute server has been implemented and where that final image is stored in the global access cache or otherwise.

Fig. 4C shows the error message issued by the compute server when service was unsuccessful or error occurred. To contain this information, the message has a computer server error code identifying the server and a field holding the reason for the error.

As illustrated in Fig. 5, the administrator or Internet application developer specifies the actions of the media manipulator by supplying an m-script language to the media flow manager 210. This is a quasi-configuration, script file which forms a high level programming language of the media manipulator. The following illustrates the general structure of the language with examples showing its use in the media manipulator 100.

name := definition

The name of a rule is simply the name itself (without any enclosing "<" and ">") and is separated from its definition by the colon-equal (":=") character.

"literal"

Quotation marks surround literal text. Unless stated otherwise, the text is case-sensitive.

rule1 | rule2

Elements separated by a bar ("|") are alternatives, e.g.,

"yes | no" will accept "yes" or "no".

{rule1 rule2}

Elements enclosed in parentheses are treated as a single element. Thus, "{elem {foo | bar} elem}" allows the token sequences "elem foo elem" and "elem bar elem".

rule*

The character "*" following an element indicates repetition. For example, "foo bar*", implies, "foo" followed by zero or more of "bars".

[rule]

Square brackets enclose optional elements. For example, "foo [bar]" implies, "foo" followed by zero or one of "bar".

The BNF grammar of the m-script is grouped under three logical groups.

Basic

Alphabets	:=	{abc...zABC..Z}
Variable-chars	:=	{abc...zABC...Z-_}
Numbers	:=	{0...9}
Variable-name	:=	Variable-chars {Variable-chars | Numbers}*
Host-name	:=	Variable-chars {Variable-chars | Numbers | "."}*
Path-name	:=	{Variable-chars | Numbers | "." | "/" | "\" | "~" | ":"}*
Others	:=	{!@#$%^&*()"{}-=|\<>,}
EOLN	:=	"\n"
Comment	:=	"#" {Variable-chars | Others | "." | "/" | "~" | ":" }* EOLN

This section describes the basic rules used: Alphabets are composed of letters "a" through "z", "A" through "Z"; Numbers are composed of digits zero through nine. Variables-chars are alphabets, dash ("-") and underscore ("_"). A variable name must start with a Variable-char and followed by zero or many variable-chars or numbers. Host-name is similar to variable-name and in-addition can have periods ("."). Path-name is a generic path used for locating files. EOLN is ASCII 13. A comment must start with "#" character and ends with an EOLN.

Generic

m-script := { comment | section }*

Section := section-key "{" section-Desc "}"

Section-Desc := section-line*

Section-line := section-desc-key "=" section-desc-value [EOLN]

Section := server-section | cache-section | service-section | filter-section | action-section

This section describes a generic m-script file. An m-script is a comment or a section. A section must start with a Section-key, followed by a Section-description enclosed in parentheses. The section description is made up of zero or many section lines. A section line starts with a section description key followed by an equal sign ("=") and the section description key's value. There are five types of sections, *viz.*, server, cache, service, filter and action.

Detail

Server-section := "server" "{" server-sec-desc "}"

Server-sec-desc := server-name-line | server-port-line

Server-name-line := "name" "=" host-name [EOLN]

Server-port-line := "port" "=" numbers [EOLN]

A server section starts with the key "server". This section consists of two lines: Name and port lines. The name line specifies the name of the host on which the MM 100 is run. The port line specifies the main port number on which the MM awaits requests from clients.

Cache-section := "cache" "{" cache-sec-desc "}"

Cache-sec-desc := cache-clean-line | cache-direc-line

Cache-clean-line := "cleanup" "=" { number | "no" }

Cache-direc-line := "directory" "=" Path-name

A cache section starts with the key "Cache". This section consists of two lines as well: Cache-clean and directory lines. The cache-clean line specifies the time interval after which the cache cleaning is performed. It takes two values: a positive number (time interval in seconds) or the string "no"(implying never to be cleaned). The directory line specifies the directory in which the cached files need to be stored.

Service-section := "service" "{" Service-sec-desc "}"

Service-sec-desc := Service -id-line | Service-host-line | Service-port-line

Service-id-line := "id" "=" variable-name

Service-host-line := "host" "=" host-name

Service-port-line := "port" "=" numbers

A service section is for service plugins. There must be a service section for each service that has to be used by the MM 100. This section starts with the key "service". The section consists of three lines: Id, Host and port lines. The id line specifies a user defined identifier that can be used in other sections. The host and port lines respectively specify the name of the host and port number on which the service is available.

Filter-sec := "filter" "{" filter-desc "}"

Filter-desc := filter-object-line | filter-action-line

Filter-Object-line := "object" "=" Filter-Object-Name

Filter-Object-Name := "image" | "video" | "java"

Filter-Action-line := "action" "=" variable-name

A filter section starts with the key "filter". This section consists of two lines: object and action line. The object line specifies the name of the object to be identified and filtered. The action line identifies the rule to be applied on the object. Currently, the objects identified are images. In future, objects like video and Java applets can be identified.

Action-section := "action" "{" action-desc "}"

Action-desc := action-id-line | action-cond-line | action-proc-line

Action-id-line := "id" "=" variable-name

Action-cond-line := "cond" "=" Action-cond-exp

Action-cond-exp := [Action-exp-bin-op] Action-exp-var [Action-cond-exp-op Action-cond-exp]

Action-exp-bin-op := "!"

Action-cond-exp-op := "&&" | "||" | "==" | "!=" | ">" | "<" | ">=" | "<="

Action-exp-var := { Filter-Object-Name "." Parameter } | { variable-name "." "result" }

Parameter := "any" | "transparent" | "animated"

Action-proc-line := "process" "=" Action-proc-exp

Action-proc-exp := { variable-name| Method-exp } [Action-connect Action-proc-exp]

Method-exp := Filter-Object-Name "." Method-name "(" Method-Param* ")"

Method-name := "replace"

Method-Param := """ Path-name """

Action-connect := "&" | "|"

The action section is the most complicated section. The action sections can be linked to other action sections forming a list of actions to be applied in tandem. The section starts with the key "action". This section consists of three lines: id, condition and process lines. The id, as before, is a user assigned identifier. The condition line specifies a condition when the process has to be performed. The condition is like a standard "C" expression. It uses object's properties (e.g., image.transparent - image that has a transparent bit), or result of other rules (e.g. rule1.result). The process can be a service identifier or another rule identifier. Several identifiers can be connected using action connectors: "&"(and) or "|"(or). The "&"(and) connector implies both the rules have to be applied in succession (e.g.: rule1 & rule2 - implies apply rule1 and then rule2). The "|" (or) connector implies that apply either of the process (e.g.: compress1 | compress2 - implies, apply compress1 or compress2).

An example is as shown below:

```
1   #m-script for manipulating HTML files
2
3   #listening host name and port
4   server {
5   name = center
6   port = 8001
7   }
8
9   #cache parameters
10  cache{
11  cleanup = no
12  directory = "/opt/mm/cache/images/"
```

```
13      }
14
15      #compress service server 1
16      service{
17      id = compress1
18      host = center
19      port = 7002
20      }
21
22      #compress service server 2
23      service{
24      id = compress2
25      host = center
26      port = 7003
27      }
28
29      filter{
30      object = image
31      action = rule1
32      }
33
34      action{
35      id = rule1
36      cond = image.any && ! image.transparent
37      process = compress1 | compress2
38      }
```

Line #	Explanation
1.	A comment line starts with a "#" character. Everything to the end of that line is ignored.
5 & 6	The media manipulator listens on the host "center" and on port "8001"
11 & 12	The files are cached (global cache) on the server. Keep them longer. Store them in the directory specified.
17-19	Compute server id is "Compress1". The host address is "center" and is listening on port "7002"
24-26	Compute server id is "Compress2". The host address is "center" and is listening on port "7003"
30 & 31	Filter the images and apply rule1

IPR2023-00330 Page 00999

```
35      This section is rule1
36      Do the process for any image that is not transparent.
37      Process  images by sending to compress1 or to compress2
```

Example #2

Apart from compressing the images, the images can be tested for pornography. For this a service section has to be added and the action section has to be modified. The following m-script accomplishes this.

```
1       #m-script for manipulating HTML files
2       #This compresses the images and detects them for pornography
3
4       #listening host name and port
5       server {
6       name = center
7       port = 8001
8       }
9
10      #cache parameters
11      cache{
12      cleanup = no
13      directory = "/opt/mm/cache/images/"
14      }
15
16      #compress service server 1
17      service{
18      id = compress1
19      host = center
20      port = 7002
21      }
22
23      #compress service server 2
24      service{
25      id = compress2
26      host = center
27      port = 7003
28      }
29
30      #pornography detect service server 1
31      service{
```

```
32      id = porno1
33      host = center
        port = 7010
34      }
35
36      #pornography detect service server 2
37      service{
38      id = porno2
39      host = center
40      port = 7011
41      }
42
43      filter{
44      object = image
45      action = all_image_rule
46      }
47
48      action{
49      id = all_image_rule
50      cond = image.any
51      process = compress_rule & porno_rule & destroy_rule
52      }
53
54      action{
55      id = compress_rule
56      cond = ! image.transparent
57      process = compress1 | compress2
58      }
59
60      action{
61      id = porno_rule
62      cond = compress_rule.result == 1
63      process = porno1 | porno2
64      }
65
66      action{
67      id = destroy_rule
68      cond = porno_rule.result >= 75
69      process = image.replace("/opt/mm/lib/images/forbidden.gif")
70      }
```

Line	Explanation
#	

5 - 8 Server section
11 - 14 Cache section
16 - 28 Compute servers "Compress1" and "Compress2"
31 - 35 Pornography detection service "porno1" is running in "center" and listening on port 7010
38 - 42 Pornography detection service "porno1" is running in "center" and listening on port 7010
44 - 47 Filter the images and apply all_image_rule
49 - 52 Apply action "all_image_rule" to all images. First apply compress_rule, followed by porno_rule and then by destroy_rule.
55 - 59 Apply action "compress_rule" to non-transparent images. Pass the images to either compress1 or compress2.
61 - 65 Apply action "porno_rule" to images, if compress_rule returned 1. Pass the compressed images to porno1 or porno2.
67 - 71 Apply action "destroy_rule" to images, if porno_rule returned a value greater than or equal to 75(probability of a pornographic image). Replace the image with "forbidden.gif".

Media Flow Manager (MFM)

MFM reads m-script and configures itself and other components based on the m-script. The MFM can be implemented as a multi-threaded object as shown below:

```
class MFM{
private:
        int iPort;
        char *strHostName;
        char *strMFileName;
        MediaParser *pMP;
        GAC *pGAC;
        ObjSw *pOS;
        NAC *pNAC;
        ...
```


IPR2023-00330 Page 01002

```
public:
        MFM();
        ~MFM();
        int Configure(char *strMFileName);
        int ProcessURL(char *strURL, ...);
        int ProcessImage(char *strSrcURL, int iHeight, int iWidth, ...);
        int CheckCacheUpdate(...);
        int CreateInstance();
        ...
};
```

Configure(...)	Parses the m-Script file specified and configures the rest of the components.
ProcessURL(...)	This is called by the parser, when it encounters a new image. This initiates the cache insertion on GAC.
ProcessImage(...)	Mainly invoked by the MediaParser, when it encounters an image tag. This passes the command to the appropriate object switch.
CreateInstance(...)	This creates a new instance of the MFM by first copying the internal data structures and then creating a new thread.

```
Media Parser - HTML Parser
The media parser can be implemented using generic tools
like lex and yacc. The core of the parser can then be
packaged to make parser objects.

class MediaParser {
private:
        MFM *pMFM;
        GAC *pGAC;
        ...
public:
```


IPR2023-00330 Page 01003

```
        MediaParser(MFM *pMFM, GAC *pGAC, ...);
        ~MediaParser();
        int AddFilter(int iObjectType, ...);
        int Parse(...);
        ...
};
```

AddFilter(...)	Called by the MFM.Configure, adds to the list of objects that the MediaParser has to look for.
Parse(...)	This is called by the NAL, when it successfully establishes a connection with the client. This parses the media. When it encounters the object to be filtered, the parser notifies the MFM by invoking the appropriate function.

Global Access Cache

The global access cache is a specialized cache system, specifically tuned to keep HTML pages and the images. The images can have multiple versions. These have to be cached separately. The cache is also cleaned regularly as described in the cache section of the m-script.

```
class GAC{
private:
        char *apMainBuckets[MAX_HASH_KEY];
        int Hash(char *strURL);
        ...
public:
        GAC(MFM *pMFM, char *strPath, ...);
        ~GAC();
        int SearchCache(char *strURL, ...);
```

```
        int PutInCache(char *strURL, char *strLocalFilename, FILE *fp, ...);
        int UpdateCache(char *strURL, ...);
        int GetFromCache(char *strURL, int iKey, ...);
        ...
};
```

Hash(...)	This is used to create the Hash key based on an URL.
SearchCache(...)	This searches the cache for the given URL.
PutInCache(...)	First searches the cache(SearchCache()) and if not found, inserts the URL int the cache.
UpdateCache(...)	Updates the cache entry with related entries. For example, the URL entry can be updated with image entries that are related to the URL.
GetFromCache(...)	Retrieves an URL or an Image.

```
Network Access Layer
The network access layer for handling HTML pages, primarily
deals with HTTP(Hyper Text Transmission Protocol). It
accepts connection from the clients; makes connection to
the content provider; requests and receives pages and
images from the content provider. In addition to these the
layer also provides connection to compute servers.

class NAL{
private:
        int iPort;
        char *strHostName;
        int iNumCharsRead;
        int iNumCharsWritten;
        char *strURL;
        ...
public:
        NAL(MFM *pMFM, MediaParser *pMP, ...);
```


IPR2023-00330 Page 01005

```
    ~NAL();
    int Listen(char *strHostName, int iPort, ...);
    int Accept(...);
    int Connect(char *strHostName, int iPort, ...);
    int AcceptClients(char *strHostName, int iPort, ...);
    int GetImage(char *strHostName, int iPort, char *strURL, ...);
    int GetURL(char *strHostName, int iPort, char *strURL, ...);
    int SendRequest(...);
    int ReceiveReply(...);
    ...
};
```

Listen(...)	Creates a listening port.
Accept(...)	Accepts any client requesting a connect.
Connect(...)	Connects to the specified host and port number. Usually called by the GetImage() or GetURL()
GetImage(...)	Connects to the ContentProvider and requests the image specified by the URL. This is responsible for building the appropriate request header etc
GetURL(...)	Connects to the ContentProvider and requests the page specified by the URL. This is responsible for building the appropriate request header etc.
SendRequest(...)	Sends a formatted message to the compute server. The format of the message is shown in the following section.
ReceiveReply(...)	Receives a formatted message that is a reply to the message sent earlier.

Service Plugin

The service sections describe the various servers available for the MM. Each service server is an instance of this object.

IPR2023-00330 Page 01006

```
class ServPlugin{
private:
        char *strId;
        int  iPort;
        char *strHostName;
        ...
public:
        ServPlugin(NAL *pNAL, char *strId, char *strHostName, int iPort, ...);
        ~ServPlugin();
        int Request(char *strSrcPath, char *strDestPath, ...);
        ...
};
```

Request(...)	This initiates the request through the NAL. NAL sends the formatted message to the appropriate Compute Server.

Object Switch

The object switch interfaces the MFM and the service plugins. The object switch mostly implements the rules specified in the action section of the m-script, as instructed by the MFM.

```
class ObjSw{
private:
        MFM *pMFM;
        GAC *pGAC;
        ServPlugin *aSP; //array of service plugins
        ActionList *alAction; //linked list of actions
        –
public:
```

```
        ObjSw(MFM *pMFM, GAC *pGAC, _);
        ~ObjSw();
        int AddServicePlugin(ServPlugin *pSP, _);
        int AddAction(char *strId, char *strCond, char *strProcess, _);
        int ProcessImage(_);
        _
};
```

AddServicePlugin(_)	This is invoked by the MFM during configuration phase. This adds the service plugin to its internal list.
AddAction(_)	This is also invoked by the MFM during the configuration phase. This adds the actions specified in the m-Script
ProcessImage(_)	Invoked by the MFM, this executes the actions in the specified order.

Compute Server

The compute server executes as a separate processor or on a different machine itself. It can be implemented as an object as well.

```
class CompServ{
private:
        int iPort;
        char *strHostName;
        ...
public:
        CompServ();
        ~CompServ();
        int ReceiveRequest(char *strSrcPath, char *strDestPath, ...);
        int ProcessRequest(...);
        int Reply(...);
```


IPR2023-00330 Page 01008

...

};

ReceiveRequest(...)	This receives the formatted message.
ProcessRequest(...)	This processes the request. The user can extend the compute server by adding capabilities to this method.
Reply(...)	Sends the reply.

The compute server can also use the NAL to send and receive messages.

Fig. 6 is an object interaction diagram showing the order of creation of the objects/components of the manipulator 100 and the order in which the m-script is processed or read. Of note is the fact that the object switch 216 is called after service plugins 218. This order ensures that the services are all declared. AddAction takes the pointer to those service plugins, and AddServicePlugin identifies the compute server executing the plugin, its host name, and its port. ObjSw ensures the GAC 220 may be updated by the object switch with the results of the service, once executed.

Fig. 7 illustrates another embodiment of the inventive media manipulator 100. The media manipulator described in the previous sections was used as an intermediate processor between the client 106, 116 and the content provider server 104. In this alternative embodiment, an additional, stripped down tunneler version of the manipulator 100' can be used to interact between the client 106, 116 and the media manipulator 100 as described previously. These two instances of the manipulator 100, 100' can now perform in unison to further enhance the user experience.

IPR2023-00330 Page 01009

The tunneler media manipulator 100' and the media manipulator 100 exchange a compressed format suitable for the transmission over a low-bandwidth connection, while the tunneler 100' and the browser(client) exchange information in the client's native format. Apart from these, the client 106, 116 can be inside a firewall f and still use the services of a main media manipulator 100, which may be outside the firewall f. The tunneler 100' can also be used to set various options such as compression quality, specific to the client's need. These options are forwarded to the main media manipulator 100 along with the client's request. The main media manipulator 100 can categorically act on both the tunneler's and client's request.

Apart from compressing images, the tunneler 100' and main media manipulator 100 combination can be used to compress the HTML page itself. The HTML page is a media, and if the service is available to compress it, the m-script can be modified appropriately to send the page to the text-compress-plugin before sending towards the client. The tunneler can intercept this and decompress the page.

The tunneler 100' has following components of the media manipulator: 1) media flow manager 210, 2) media parser 212, 3) object switch 216, 4) network access layer 214, and 5) service plugin 218. It does not the global access cache 220. The service plugin in the tunneler 100' is the compliment of what is used in the media manipulator to decompress the images.

IPR2023-00330 Page 01010

While this invention has been particularly shown and described with references to preferred embodiments thereof, it will be understood by those skilled in the art that various changes in form and detail may be made therein without departing from the spirit and scope of the invention as defined by the appended claims.

IPR2023-00330 Page 01011

CLAIMS

What is claimed is:

1. A middle-ware computing system comprising:

a network access system that supports communications with media resources and client computers; and

a media manipulation system that operates on media objects received from the media resources via the network access system prior to forwarding the media objects to the client computers.

2. The computing system described in Claim 1, wherein the media manipulation system comprises:

a parser that identifies different media types within the media objects; and

service devices that operate on the media types.

3. The computing system described in Claim 2, wherein the parser searches for images in the media objects and service devices include an image compressor for performing data compression on the images.

4. The computing system described in any of Claims 2-3, wherein the parser searches for executable files in the media objects and service devices include a virus scanner that searches for computer viruses in the files.

5. The computing system described in any of Claims 2-4, wherein the parser searches for images in the media objects and service devices include a pornography detector for assessing a probability that the images are pornographic.

6. The computing system described in any of Claims 2-5, wherein the parser searches for data files in the media objects and service devices include an format converter for changing a format of the data files.

7. The computing system described in any of Claims 2-6, wherein the media manipulation system further comprises an object switch that passes the media types to the service devices to determine operations performed on the different media types.

8. The computing system described in any of Claims 2-7, wherein the media manipulation system further comprises a media flow manager that reassembles the media objects for forwarding to the clients after the manipulation of the media types.

9. The computing system described in Claim 8, further comprising a cache that stores media objects, the media flow manager receiving requests for media objects and checking for the presence of the media objects in the cache to preclude obtaining the objects from the media resources.

10. A middle-ware computing system comprising:

a network access system that supports communications with media resources to obtain media objects from client computers;

a parser that identifies different media types within the media objects;

service devices that manipulate the media types;

an object switch that passes the media types to the service devices to determine operations performed on the different media types; and

a media flow manager that reassembles the media objects for forwarding to the clients after the manipulation of the media types.

11. The computing system described in Claim 10, further comprising a cache that stores media objects, the media flow manager receiving requests for media objects and checking for the presence of the media objects in the cache to preclude obtaining the objects from the media resources.

12. A method for facilitating transmission of media objects between media resources and client computers, the method comprising:

receiving requests for media objects from the client computers to the media resources;

obtaining the media objects;

manipulating the media objects;

forwarding the manipulated media objects to the client computers.

13. The method described in Claim 12, wherein manipulating the media objects comprises:

identifying different media types within the media objects; and

performing separate operations on the different media types.

14. The method described in Claim 13, wherein the step of identifying different media types comprises searching for images in the media objects and the step of performing operations comprises data compressing the images.

15. The method described in any of Claims 13-14, wherein the step of identifying different media types comprises searching for executable files in the media objects and the step of performing operations comprises scanning the files for computer viruses.

16. The method described in any of Claims 13-15, wherein the step of identifying different media types comprises searching for images in the media objects and the step of performing operations comprises assessing a probability that the images are pornographic.

17. The method described in any of Claims 13-16, wherein the step of identifying different media types comprises searching for data files in the media objects and the step of performing operations changing a format of the data files.

18. The method described in any of Claims 13-17, further comprising reassembling the media objects for forwarding to the clients after the manipulation of the media types.

19. The method described in any of Claims 13-18, further comprising routing the media types to form successive operations on the media types.

20. The method described in any of Claims 13-19, further comprising caching media objects that have been received from the media resources and later obtaining the media objects from the cache.

21. The method described in Claim 20, wherein the step of obtaining the media objects comprises requesting the media objects from the media resources while checking for the objects in a cache; and obtaining the media objects from the cache if present.

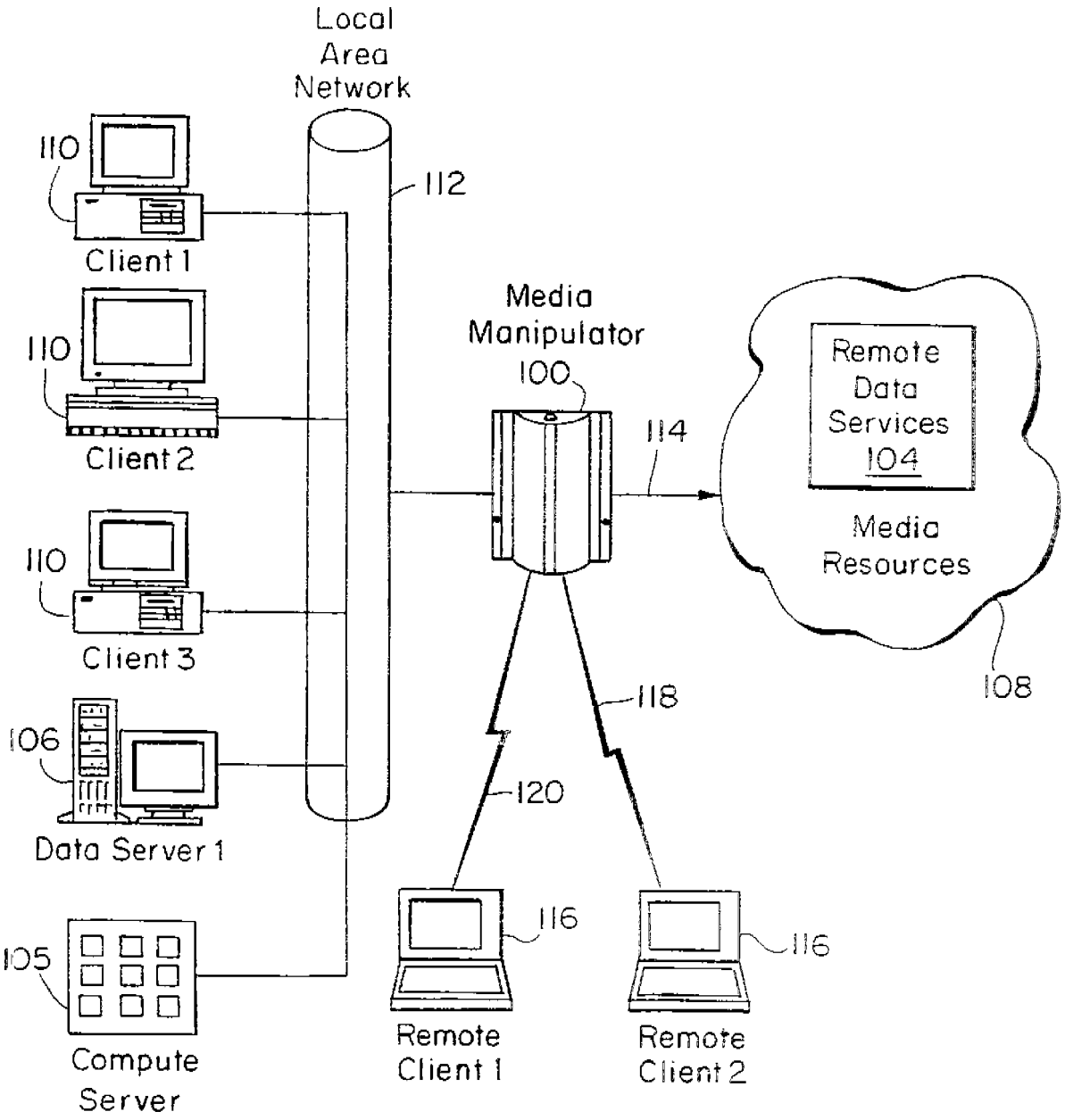


FIG. 1

IPR2023-00330 Page 01017

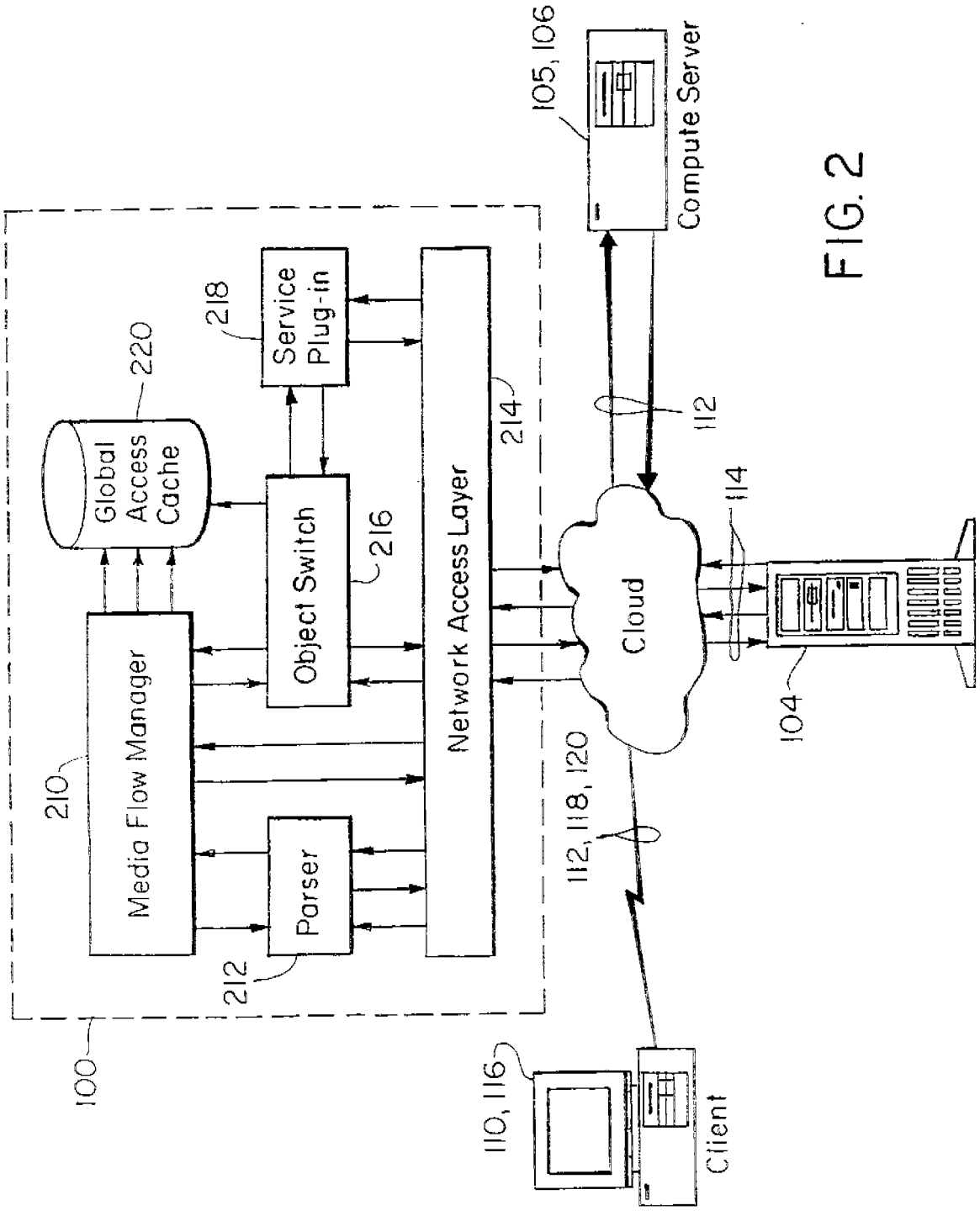

105, 106
Compute Server
218
Service Plug-in
220
Global Access Cache
214
Network Access Layer
112
216
Object Switch
114
Cloud
210
Media Flow Manager
104
112, 118, 120
212
Parser
100
110, 116
Client

FIG. 2

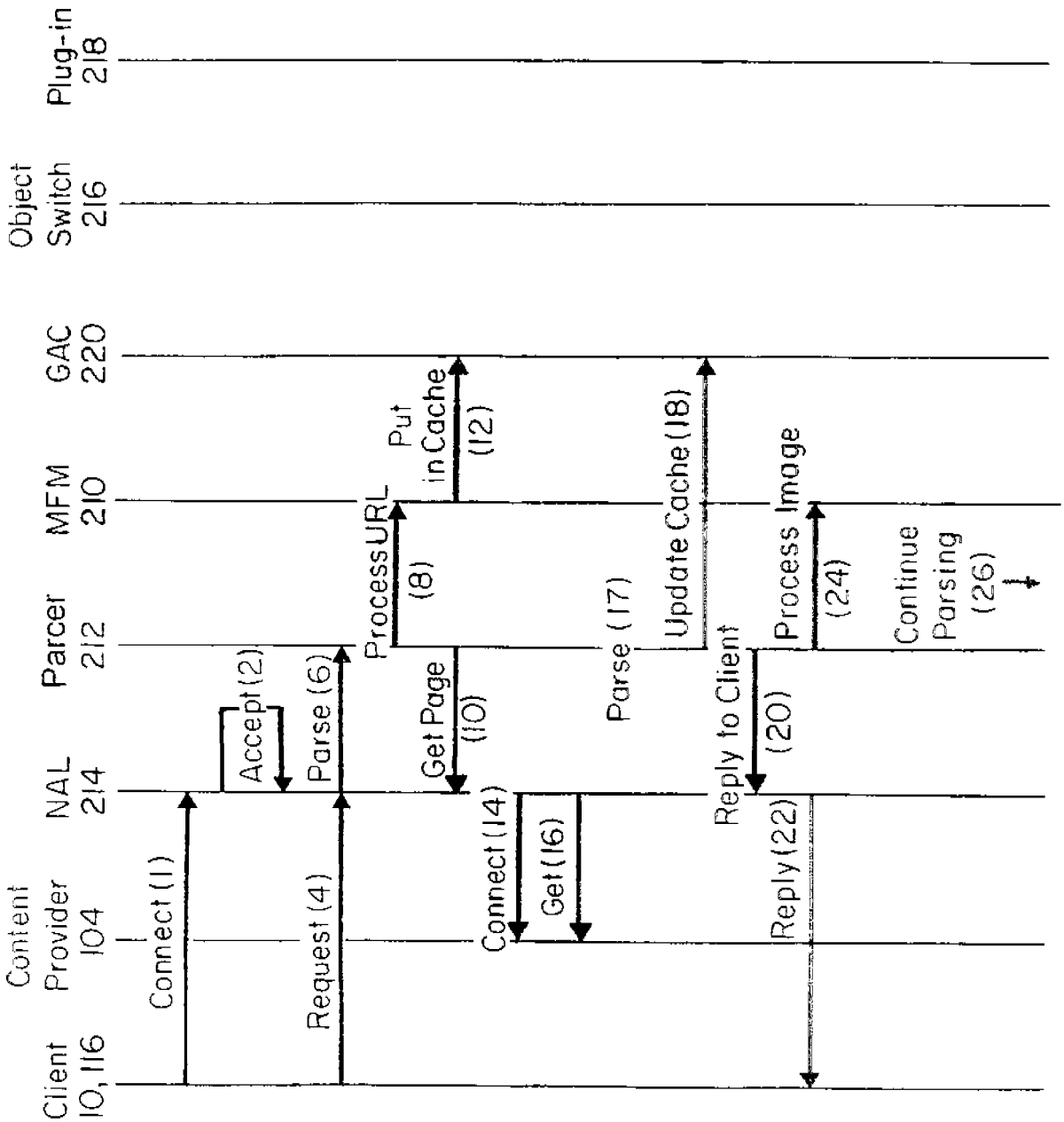


FIG. 3A

IPR2023-00330 Page 01019

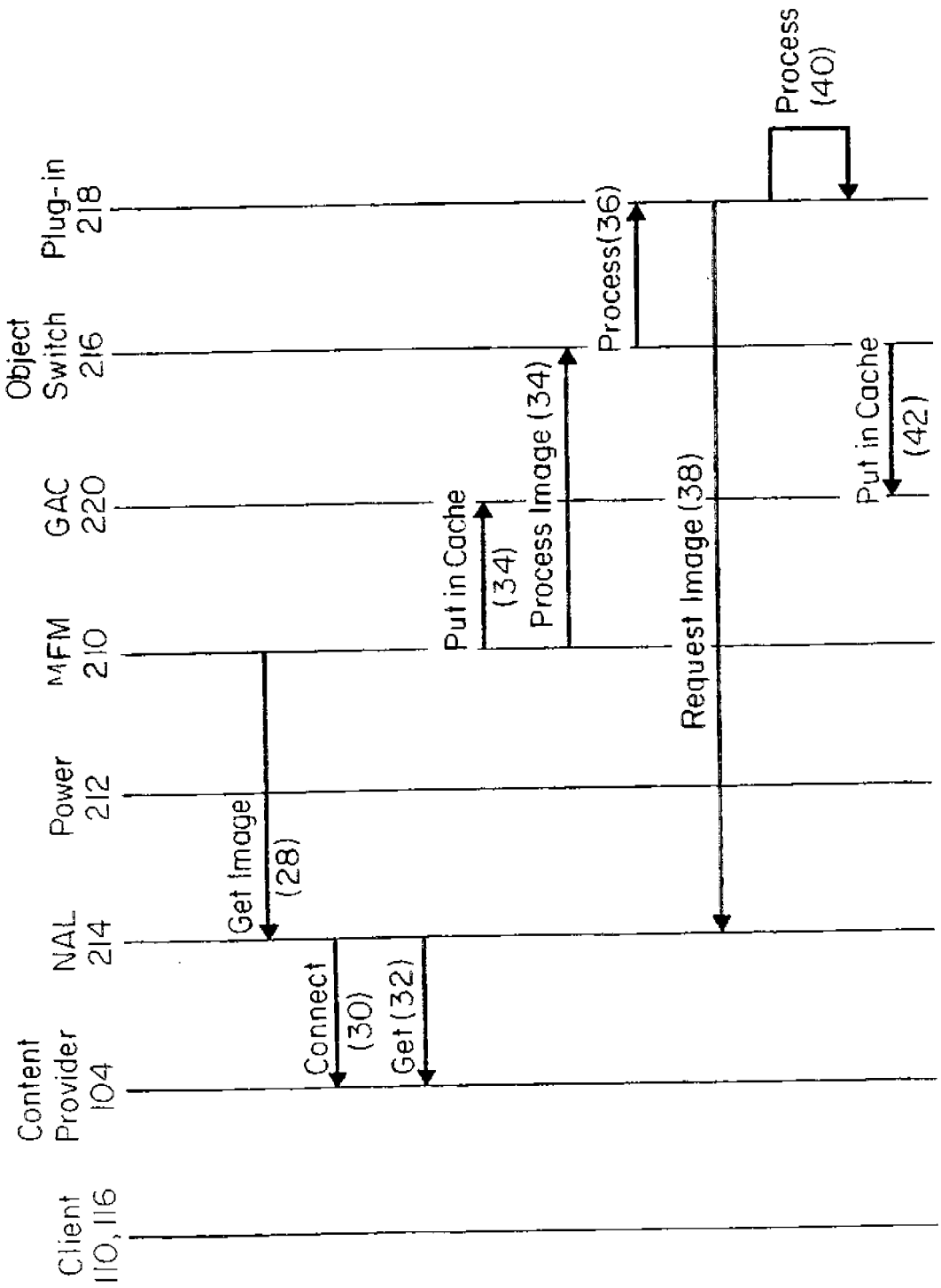


FIG. 3B

IPR2023-00330 Page 01020

Header				Content						
Version	Length	Type	Message Id	Src Type	Src Path Len	Src Path	Dest Type	Dest Path Len	Dest Path	Dest Param

Field	Field Length	Description
Version	4	Message Version Number. E.g. 0100, implies 1.0
Length	4	Length of the content
Type	4	Type of the Message: 1-for request, 2-for reply, 3-for error
Message Id	4	Numeric ID of the message assigned by the NAL
Src Type	4	Numeric type of the source image: 1-GIF, 2-JPEG, 3-MM Compress Format 1
Src Path Len	4	Length of the Src Path
Src Path	-	Path where the image is stored. Can be a network path as well.
Dest Type	4	Numeric type of the final image: 1-GIF, 2-JPEG, 3-MM Compress Format 1
Dest Path Len	4	Length of the Dest Path
Dest Path	-	Path where the final image has to be stored
Dest Param	4	Can be used to set an optional parameter

FIG. 4A

Header				Content			
Version	Length	Type	Message Id	Reply Code	Dest Type	Dest Path Len	Dest Path

Field	Field Length	Description
Version	4	Message Version Number. E.g. 0100, implies 1.0
Length	4	Length of the content
Type	4	Type of the Message: 1-for request, 2-for reply, 3-for error
Message Id	4	Numeric ID of the message assigned by the NAL
Reply Code	4	The success or failure of the service: 1- success, 0- error
Dest Type	4	Numeric type of the final image: 1 - GIF, 2 - JPEG, 3 - MM Compress Format 1
Dest Path Len	4	Length of the Dest Path
Dest Path	-	Path where the final image has to be stored

FIG. 4B

IPR2023-00330 Page 01021

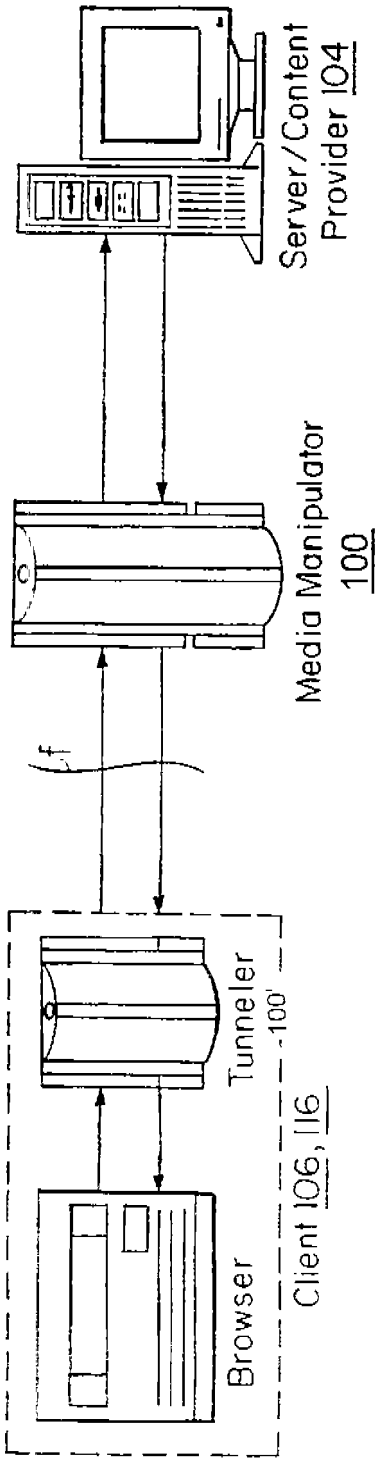
Header				Content			
Version	Length	Type	Message Id	Reply Code	Error Code	Error Reason Len	Error Reason

Field	Field Length	Description
Version	4	Message Version Number. E.g. 0100, implies 1.0
Length	4	Length of the content
Type	4	Type of the Message: 1-for request, 2 for reply, 3 for error
Message Id	4	Numeric ID of the message assigned by the NAL
Reply Code	4	The success or failure of the service: 0-error
Error Code	4	Numeric Error Code assigned by the compute server
Error Reason Len	4	Length of the reason, the next field
Error Reason	-	String describing the error

FIG. 4C

IPR2023-00330 Page 01022

FIG. 7



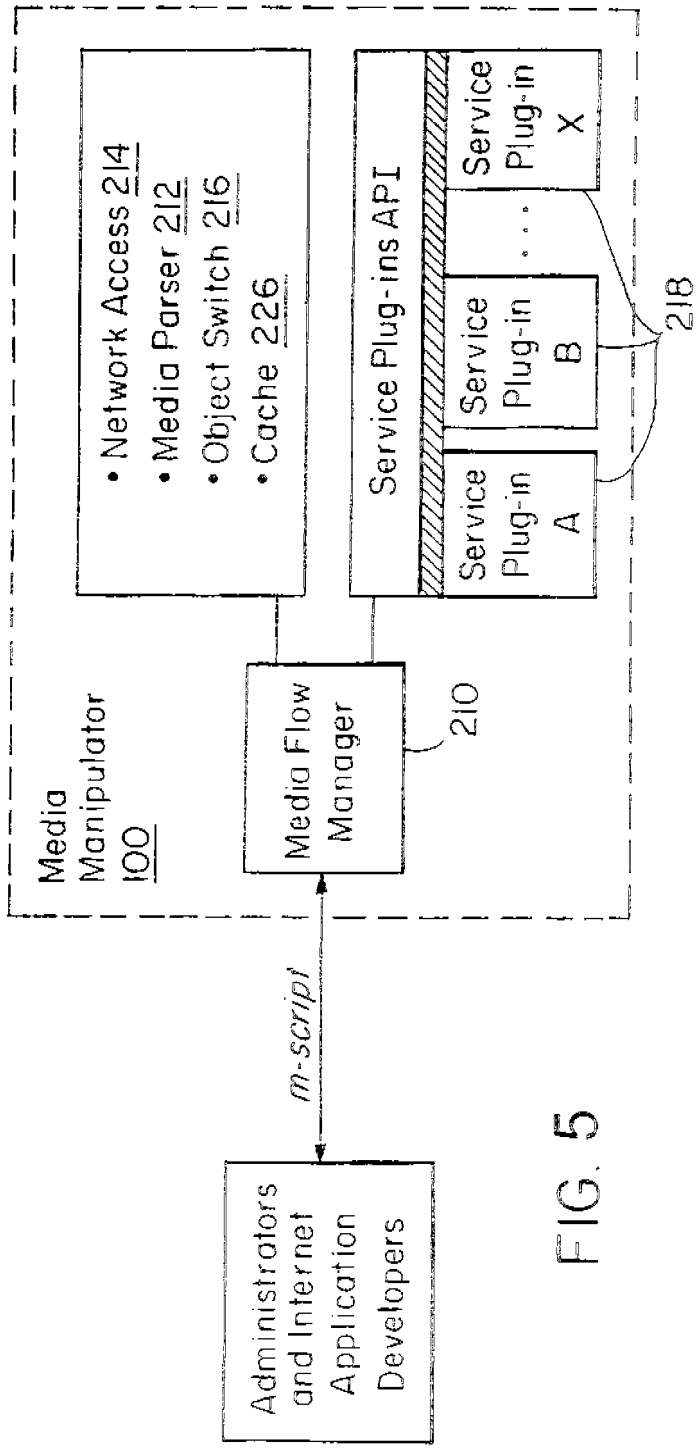


FIG. 5

IPR2023-00330 Page 01023

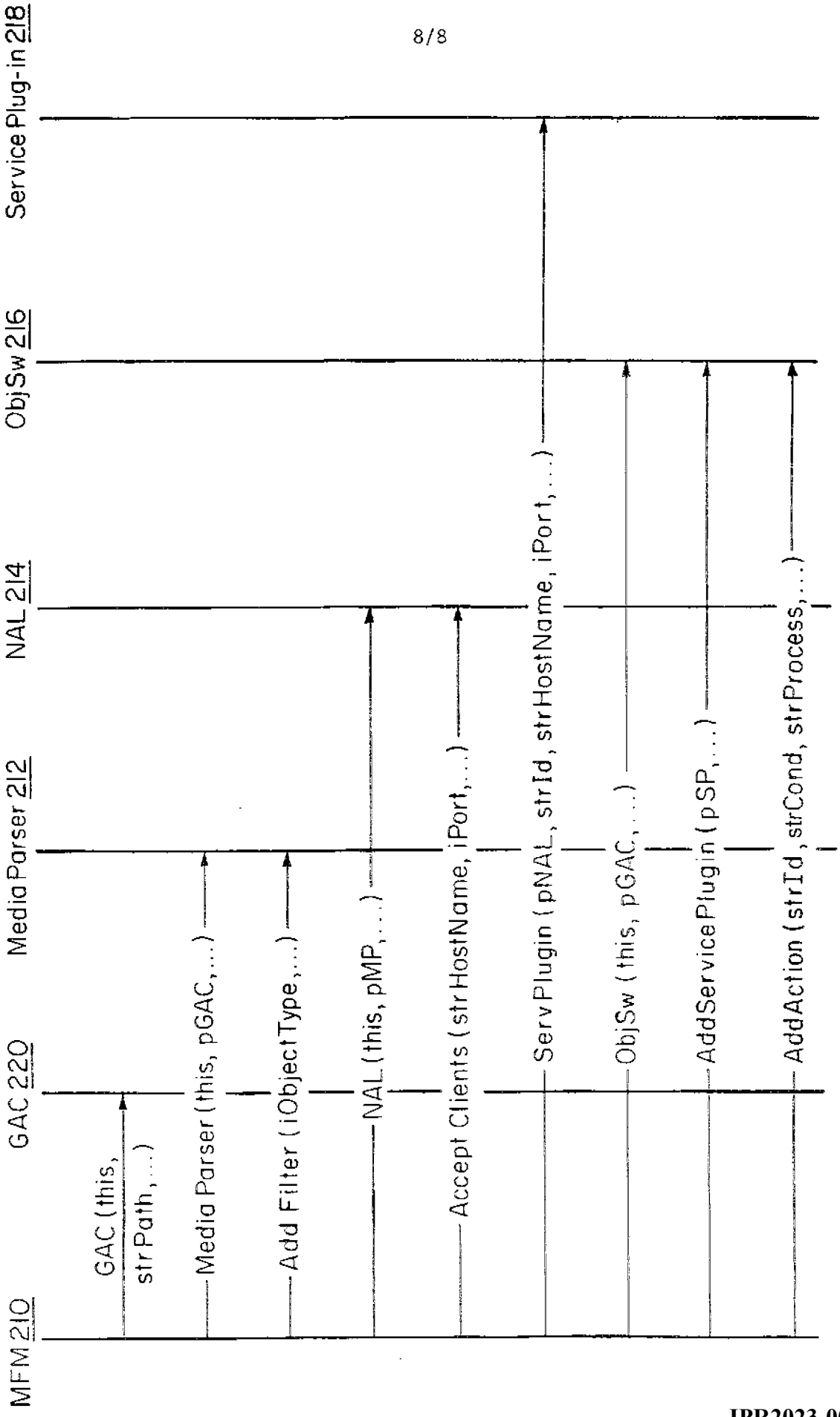


FIG. 6

IPR2023-00330 Page 01024

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 :
H04L 29/06 | **A3**

(11) International Publication Number: **WO 97/49252**

(43) International Publication Date: 24 December 1997 (24.12.97)

(21) International Application Number: PCT/US97/10758

(22) International Filing Date: 20 June 1997 (20.06.97)

(30) Priority Data:
60/020,094 21 June 1996 (21.06.96) US

(71) Applicant *(for all designated States except US)*: INTEGRATED COMPUTING ENGINES, INC. [US/US]; 460 Totten Pond Road, Waltham, MA 02154 (US).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: SHAH, Ashesh, C. [US/US]; 567 Tremont Avenue, No. 31, Boston, MA 02118 (US). PEDERSEN, Palle [DK/US]; 82 Commonwealth Avenue, No. 10, Boston, MA 02116 (US). RADOVIC, Niksa [HR/US]; 19 Mountain Avenue, Somerville, MA 02143 (US). MANICKAVASAGAM, Senthilkumar [IN/US]; 11 Highland Glen Drive, No. 17, Randolph, MA 02368 (US).

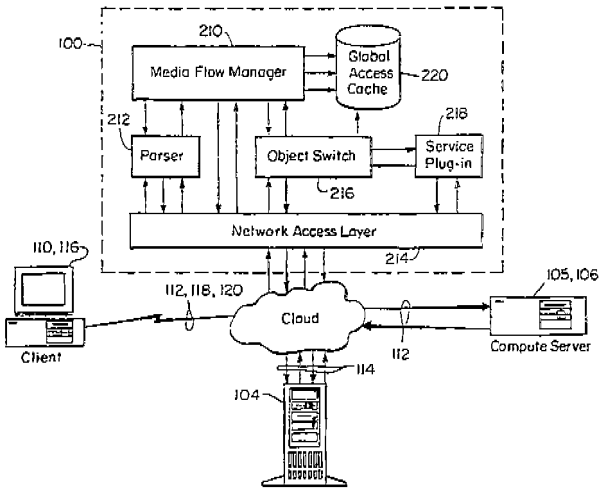
(74) Agents: SMITH, James, M. et al.; Hamilton, Brook, Smith & Reynolds, P.C., Two Militia Drive, Lexington, MA 02173 (US).

(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, HU, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, TJ, TM, TR, TT, UA, UG, US, UZ, VN, YU, ARIPO patent (GH, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

Published
With international search report.

(88) Date of publication of the international search report:
30 April 1998 (30.04.98)

(54) Title: NETWORK BASED PROGRAMMABLE MEDIA MANIPULATOR



(57) Abstract

The media manipulator is a middle layer between the clients (110, 116) and the remote data servers (104) is the common client-server organization. It transforms the network into a more flexible three-tiered configuration. Requests generated by the clients (110) for media objects from media resources are routed to the media manipulator (100). It processes the requests and determines if the media objects may be found locally, either cached (220) in the media manipulator (100) itself or in the local data servers (106). When the media objects are obtained, the media manipulator (100) can be used to perform operations on those objects such as format translations, to apply protective mechanisms for the clients (110), to speed communications between the remote servers (104) and the clients (110), or perform compute operations for the clients (110). In one example, a parser (112) of the manipulator (100) searches for images in the media objects so that service devices (218) can be called to perform data compression or pornography detection on the images. The parser can also search for executable or data files in the media objects and to perform virus scanning or format conversion, respectively.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

INTERNATIONAL SEARCH REPORT

Inter ›nal Application No
PCT/US 97/10758

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 H04L29/06

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 H04L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	EP 0 669 587 A (AT & T CORP) 30 August 1995	1,12
A	see column 4, line 20 - column 5, line 14; figure 1 see column 7, line 13-34; figure 2 see column 9, line 17-29 see column 16, line 39-51	2,3,7,8, 10,13, 14,18
A	THAU R: "Design considerations for the Apache Server API" COMPUTER NETWORKS AND ISDN SYSTEMS, vol. 28, no. 11, May 1996, pages 1113-1122, XP002046988 see paragraph 4	2,10,13

-/--

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
20 November 1997	18.12.97

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Dupuis, H

INTERNATIONAL SEARCH REPORT

Inter onal Application No
PCT/US 97/10758

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	TREVOR J ET AL: "Exorcising daemons: a moduiar and lightweight approach to deploying applications on the Web" COMPUTER NETWORKS AND ISDN SYSTEMS, vol. 28, no. 11, May 1996, pages 1053-1062, XP002046968 see paragraph 3 - paragraph 3.1 ---	2,3,7,8, 10,13, 14,18
A	WO 96 17306 A (ORACLE CORP) 6 June 1996 see page 8, line 16-21 see page 11, line 29 - page 13, line 22; figure 1 see page 15, line 6-16 see page 17, line 27 - page 18, line 15 see page 34, line 2-28 ---	1,6, 9-12,17, 20,21
A	HOWLETT D: "Protection on the Web" COMPUTERS AND SECURITY, vol. 15, no. 4, 1996, page 319 XP002046969 see the whole document -----	4,15

7

INTERNATIONAL SEARCH REPORT

Information on patent family members

Inter onal Application No
PCT/US 97/10758

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 0669587 A	30-08-95	CA 2140850 A	25-08-95
WO 9617306 A	06-06-96	NONE	

ORIGINAL

AUSTRALIA

Patents Act 1990

COMPLETE SPECIFICATION

APPLICANT: Dudley John MILLS

ADDRESS: 30 Hutchison Crescent,
Kambah, ACT 2902

ACTUAL INVENTOR: Dudley John MILLS

ADDRESS FOR SERVICE: Dudley J. Mills,
30 Hutchison Crescent,
Kambah, ACT2902

ASSOCIATED PROVISIONAL: PO5254 Filed 21 February 1997

INVENTION TITLE: NETWORK-BASED CLASSIFIED INFORMATION SYSTEMS

The following is a full description of the invention including the best method of performing it known to me:

AU9853031

(12) PATENT ABSTRACT (11) Document No. AU-A-53031/98
(19) AUSTRALIAN PATENT OFFICE

(54) Title
NETWORK-BASED CLASSIFIED INFORMATION SYSTEMS

(51)[6] International Patent Classification(s)
G06F 017/30

(21) Application No. : **53031/98** (22) Application Date : **10/02/98**

(30) Priority Data

(31) Number	(32) Date	(33) Country
PO5254	**21/02/97**	**AU AUSTRALIA**

(43) Publication Date : **27/08/98**

(71) Applicant(s)
DUDLEY JOHN MILLS

(72) Inventor(s)
DUDLEY JOHN MILLS

(57)

A system for automatically creating databases containing industry, service, product and subject classification data, contact data, geographic location data (CCG-data) and links to web pages from HTML, XML or SGML encoded web pages posted on computer networks such as the Internet or Intranets. The web pages containing HTML, XML or SGML encoded CCG-data, database update controls and web browser display controls are created and modified by using simple text editors, HTML, XML or SGML editors or purpose built editors. The CCG databases may be searched for references (URLs) to web pages by use of enquiries which reference one or more of the items of the CCG-data. Alternatively, enquiries referencing the CCG-data in the databases may supply contact data without web page references. Data duplication and coordination is reduced by including in the web page CCG-data display controls which are used by web browsers to format for display the same data that is used to automatically update the databases.

TITLE: NETWORK BASED CLASSIFIED INFORMATION SYSTEMS

FIELD OF INVENTION

This invention relates to network based classified information systems, to methods of automatically building searchable databases of classified information derived from web pages posted on a network, and, to web pages for use in such systems and methods.

The information systems and databases of most relevance to this invention are those which include classified product and service catalogues similar to the Yellow Pages telephone books, contact indexes similar to the White Pages telephone books, and/or subject indexes similar to Library catalogues. Such information systems and databases typically include sets of associated classification, contact and/or geographic items of information. For convenience, classification, contact and/or geographic information will be hereinafter called CCG-data.

The networks with which this invention is concerned are the worldwide public computer/communications network commonly known as the Internet and private networks – sometimes called intranets – which allow common access to markup documents on computers connected to the network. Markup documents are text files prepared using various markup languages such as HyperText Markup Language (HTML) and Extensible Markup Language (XML) which are implementations (or dialects) of the Standard Generalised Markup Language (SGML). The system of accessible files on the Internet is called the World Wide Web (WWW) and the markup documents themselves are commonly called 'web pages'. A web page is said to be 'posted' on a network when it is stored on computer-readable media of a host network computer as a file which is generally accessible to network users. A web page is transported from the host computer to a requesting computer through intermediate network computers as a computer-readable signal embodied in a carrier wave. Though this invention is not limited to Internet based information systems, these terms are used for convenience.

BACKGROUND TO THE INVENTION

It has been estimated that there are about 100 million web pages on the Internet and that the number is doubling every two years. Many of these pages include information concerning commercially offered goods and services and often include contact details. But the difficulty of locating such information is increasing faster than the growth in the number of web pages.

To assist network users locate web pages of interest, certain network service providers create indexes (or databases) of the contents of web pages posted (stored on computer readable media so as to be generally accessible) on the network and provide 'search engines' to use the indexes. These indexes are often created automatically by the use of 'web crawlers' which (i) interrogate computer after computer on the network to locate successive web pages and (ii) index the words in each web page encountered against the network address (eg Internet Protocol Address or IPA) and filing system path or universal resource locator (URL) at which the web page is accessible. Hereinafter the terms URL and URI (Uniform Resource Identifier) are taken to be identical in meaning and to signify network addresses and filing system paths. Usually, the indexes consist of a list of unique words with each word having an associated list of URLs of the web pages wherein the word was found to occur during interrogation. The URL serves as a 'hyperlink' which, if selected by a user/searcher, results in the associated web page being automatically transmitted from the computer where it is posted on the network to the user/searcher's computer where it may be displayed or otherwise processed. The sending and receiving of files in this way is greatly assisted by user interface programs called 'web browsers' (or more simply, 'browsers') such as Netscape and Microsoft Internet Explorer.

The search for web pages of interest using search engines leaves much to be desired:

- simple searches (those using a few keywords in simple combinations) often yield far too many web page references (URLs) to permit them to be interrogated one-by-one,
- complex searches (those using many keywords and/or complex Boolean expressions) require considerable expertise to undertake,
- even using optimum search criteria, many irrelevant web pages are referenced because of inconsistent use of terminology by those who author the original web pages,
- even using optimum search criteria, many relevant pages are missed, again because of inconsistent use of terminology by web page authors, and
- because items of information included in the body of web pages cannot be 'understood' or associated in useful ways by web crawlers; that is recognised as, say, a surname, a street name, a geographic locality, or type of goods or services and, say, a surname strongly associated with a street name, a geographic locality, or a type of goods or service.

The result is that information provided by search engines from databases which are automatically compiled using web crawlers is a very poor equivalent of the common Yellow Pages and White Pages directories which serve the telephone industry (though these directories are not, of course, automatically compiled from web pages).

In an attempt to improve the usefulness of automatically compiled network databases, some search engine providers make use of information contained in URLs, such as the country code and top level domain name codes such as 'com', 'edu', 'net' and 'org' which is sometimes used to signify the subject matter of web pages. It has been proposed to add more content classifying codes to URLs (eg, "chem" to signify chemical subject matter) to allow specialised databases – national, commercial, chemical, etc – to be generated. However, this proposal has serious drawbacks:

- URLs are Internet addresses and it is in principle undesirable to confuse the address function of a URL with that of representing a list of web page classifications or contact details.
- A URL is an inappropriate container of multiple web page classification codes and contact details because the length of the URL would cause it to become unwieldy as an Internet address.
- Including in a URL classification codes drawn from a list of thousands of codes would compromise the mnemonic quality of Internet addresses such as "www.yellowpages.com".
- There is substantial overlap in the subject matter contained in web pages having the various top level domain name codes.
- There is no consensus on, or standard for, content classification codes in URLs.

Another proposal to add content classification data to web pages has arisen from the wish to identify pages containing material that may be offensive to some viewers, or should not be accessed by minors. The Platform for Internet Content Selection (PICS) (see http://www.w3.org/pub/WWW/PICS and other documents at www.w3.org) is a web page ratings standard similar in principle to the ratings systems for motion pictures. This system allows page authors to "internally" self classify their pages through use of the "<meta...>" HTML element. Alternatively, "external" PICS ratings of web pages may be obtained from ratings service providers accessed each time a URL is selected. In practice, the ratings service providers have adopted very limited range of web page classifications. For example, Ararat Software's Commercial Rating System (see http://www.ararat.com.ratings/ararat10.html) provides just 5 categories of web page content; commercial content, technical/customer support, ordering information, downloading information and contact information. In other

examples, CyberPatrol (http://www.microsys.com/pics/pics_msi.htm) provides 16 categories, the Recreational Software Advisory Council (http://www.rsac.org/faq.html) provides 4 categories, SafeSurf (http://www.safesurf.com/ssplan.htm) provides 11 categories and Vancouver Webpages Rating Service (http://vancouver-webpages.com/VWP1.0/ provides 11 categories. None of the categories provide classification of web pages by industry, service, product or subject with sufficient specificity to be useful when searching for web pages. Rather, the categories are intended to prevent web browsers from displaying web pages unsuitable for particular types of web browser users. Such rating systems are not intended to be used for the automated creation of Yellow or White pages like databases from web pages and are unsuitable for that purpose because they can not represent contact details. Further, the ratings data may only be encoded in the <meta...> element in the <head> of an HTML document drastically limiting the type and usefulness of the data that can be encoded.

Another proposal for classifying the content of web pages, the "Meta Content Framework" (MCF - see http://mcf.research.apple.com/mcf.html"), requires the content of web pages to be classified and the classification data to be held in a separate non-HTML data file with a MIME type of text/mcf. Storing data in non-HTML encoded documents which describes the content of HTML encoded documents is a technical and economic barrier to the adoption by search engine providers of the proposal. The MCF proposal is thus entirely unsuited to the automated creation of Yellow or White pages like databases from HTML encoded web pages (MIME type text/html) because data stored according to the MCF proposal is not stored in HTML encoded web pages.

The "Electronic Business Card", vCard, (see "vCard The Electronic Business Card" Version 2.1, versit Consortium Specification, Sept 18, 1996 or ftp://ds.internic.net/internet-drafts/draft-ietf-asid-mime-vcard-01.txt) uses non-HTML data file (MIME Content Types of "text/plain" or the non-standard "text/X-vCard") containing contact information equivalent to an extended White Pages entry which can be exchanged on a network using Simple Mail Transfer Protocol (SMTP) or using HTTP. It can be associated with a web page by use of a URL in the web page which refers to the vCard information (eg My vCard). Version 2.1 vCard standard data file format (published 18 September 1996) provides for the inclusion of many items of contact information. The vCard specification recommends that, where possible, there should be consistent mapping of vCard property names to HTML "<input>" element attribute names (eg vCard property name "TITLE" maps to HTML "<input name= 'title'>"). The intention is to facilitate the transfer of vCard data into web page input forms by pasting from a clipboard or by dragging from other computer applications. The VCard proposal is unsuited to the automated creation of Yellow or White pages like databases from HTML encoded web pages because data stored according to the VCard proposal is not stored in HTML encoded web pages.

The inclusion of classified information in separate documents (such as Meta Content files or vCards) has the disadvantage that there is necessarily much duplication of data and coordination of modifications between the separate documents and the web pages. This must be done to allow a person who has accessed a web page using an HTML compliant browser to determine whether it is worth calling up the associated file or vice versa. Also, to allow portions of web pages to be classified, web page contextual information would have to be duplicated in the separate document. vCards in particular do not provide this functionality. Another disadvantage is that non-HTML documents such as vCards contain no details as to how the data they contain is to be displayed. In the display of HTML documents the position, font, size, colour of the text and other elements of the document are of great importance. The

restriction of address data in a vCard to untagged ordinally organised fields is inflexible. For example, multiple instances of extended parts of the address are not possible. Also components of names, addresses and telephone numbers and so forth are insufficiently identified.

The Online Computer Library Center Inc (OCLC, Dublin, Ohio, USA) proposal, known as the "Dublin Core", proposes to classifying scholarly web pages by subject (topic of the work, or keywords that describe the content of the work), title, author, publisher, other agent, date, object type (genre of the object such as home page, novel, poem etc), form, identifier, source, language, relationship and coverage (spatial and temporal) (see http://www.oclc.org:5046/~weibel/html-meta.html and other documents at www.oclc.org). This proposal does not include industry, service, product or subject classifications. It also does not include contact details. Names such as that of the author are not specified in sufficient detail to avoid ambiguities such as which is the author's first and last names. The proposal specifies that the details are encoded using the <meta...> element in the <head> of web pages. The proposal is unsuited to the automated creation of Yellow or White pages like databases from web pages because the proposal does not provide for classification of web pages and does not provide adequate contact details. Further, the use of keywords for describing the content of the work adds very little to the effectiveness of indexing of web pages since the web pages are usually indexed on every word of their content and most often the key words would simply be a duplication of words already contained in the document.

It has also been proposed to use the Dewey Decimal System (see http://orc.rsch.oclc.org:6109/eval_dc.html and http://orc.rsch.oclc.org:6109/bintro.html) to rank electronic documents against a Dewey Decimal subject classification. The proposal suggests automatically assigning Dewey Decimal subject classification codes to documents during automated indexing and cataloguing but does not specify the exact nature of the assignment although it is implied that the codes are stored separately from the documents. The proposal admits that such automated classification is less satisfactory than human classification. The proposal is unsuited to the automated creation of Yellow or White pages like databases from web pages because the accuracy of classification is inadequate, does not provide for inclusion of industry, service or product classifications and does not provide for inclusion of contact details. Deriving a subject classification code from an analysis of every word and phrase in a web page is computationally expensive.

The HTML 3.0 standard (see page 23 of the www.w3.org document "draft-ietf-html-specv3-00.txt") provides "class" as an attribute of almost all HTML "<body>" elements. The "class" attribute is intended to be used with style sheets. Style sheets provide a means by which the display of HTML documents may be altered to suit the needs of different classes of browser users. For example, could be used to define a division that acts as an appendix, <h2 class="section"> could be used to define a level 2 header that acts as a section header, although, of course, any string of characters could be defined for those purposes. The "class" attribute, although never having been suggested for holding goods and services classifications, is not suited for such a use as it is, in any case, undesirable to confuse the style sheet function of the "class" attribute.

The HTML 3.0 and earlier standards provided the HTML elements "<person>" and "<address>" but do not specify the form of the content or method of validating the content of those elements. A person's name may be written as first name followed by last name or last name followed by first name. Similarly, different conventions exist for writing addresses. Similar

ambiguities arise in the ill defined format of the HTML elements "<person>" and "<address>". As such they are of little use in the automatic compilation of searchable databases.

The XML language (see: http://textuality.com/sgml-erb/WD-xml.html) was developed to extend HTML so that software vendors can add new elements and new element attributes to HTML which are not specifically defined in any HTML standard. The intention is to ensure that all new elements and attributes could be parsed by all XML parsers even if the new elements held no significance for any particular XML parser. However, like HTML, XML does not provide a standard for the representation of industry, service, product or subject classification, contact or geographic location details within an web page.

Of course, many useful databases of the Yellow Pages or White Pages type are made available by service providers on networks, but they are not compiled automatically by using web crawlers to scan HTML web pages posted on a network. For example, http://www.yellowpages.com.au and http://www.mcp.com provide classified advertisements of the Yellow Pages type with links to the web pages of paying advertisers or subscribers. There are also directories of email addresses which approximate the White Pages directories, listing the names of individuals and organisations and contact details, (eg http://www.bigbook.com and http://query1.whowhere.com). However, these email directories require listers to manually add their directory entries and enquirers to be aware of and to find the directory enquiry web page. They cannot be automatically generated by scanning web pages using web crawlers since there is no adequate mechanism to relate email addresses to the names of people and organisations and their other contact details which may also exist in the same web page.

OBJECTIVES OF THE INVENTION

The general object of the invention is to provide improved methods for automatically building searchable databases of classification, contact, and/or geographical information by using web crawlers to interrogate web pages posted on a network. [For convenience, this information is collectively referred to as CCG-data].

Other non-essential objectives are to provide methods for including and/or displaying CCG-data within web pages accessed by browsers, for automatically extracting CCG-data from web pages posted on a network and for using the same, and/or to provide methods for searching automatically compiled databases using such data.

Another subsidiary objective of the invention is to provide a new form of web page which is better suited to the automatic compilation (using web crawlers) of databases constructed by the automatic scanning of many such pages posted on a network.

OUTLINE OF THE INVENTION

The invention is based upon the realisation that highly useful databases can be automatically built by successively interrogating web pages posted on a network if one or more HTML encoded CCG phrases are included in the web pages. A CCG phrase is one containing CCG-data in a form which is directly accessible and identifiable. CCG phrases may also include one or more items which provide the web page author with control over how the CCG-data is applied to the database.

Data duplication can be reduced if some of the CCG-data in the coded CCG phrases can be displayed by browsers as well as being used to update databases. Errors due to inexactly duplicated data are also eliminated. Accordingly, it is envisaged that CCG phrases may include

one or more items which provide the web page author with control over how the CCG-data is displayed by a browser.

HTML (including version 2 and version 3) and XML are evolving applications (sub-sets or dialects) of ISO Standard 8879 1986 known as Standard Generalised Markup Language (SGML). HTML, in large part, is a language used to describe how text (unstructured data) and graphics is to be formatted for display. The HTML language consists of a finite number of "elements" (for example: "<BR>" where "BR" is the element name, also called the tag name) which may contain "attributes" (for example; "<DL COMPACT>" where "COMPACT" is an attribute named "COMPACT") and may contain values associated with attributes (for example; "<FONT SIZE=+1>" where +1 is the attribute value of the attribute named "SIZE"). XML is a language used to describe structured data. The XML language is similarly composed of elements, attributes and values with a similar syntax to HTML but unlike HTML the element names which may be used are not restricted and the meaning of the XML data may be interpreted in any convenient manner. While the XML language is mute about how data described by XML is to be formatted for display, the data may be used by computer programs for any purpose including description of how XML coded data is displayed. However, due to its historic importance in connection with web pages, the term "HTML" is herein used to refer to all markup languages which are subsets or complete sets of the SGML language. In particular, the term "HTML encoded CCG phrase" and the synonymous term "CCG phrase" are herein used to refer to CCG-data encoded in a subset or complete set of the SGML language. Herein, a "web page" is a document adapted to be or actually accessible through a network and encoded in a subset or complete set of the SGML language.

For convenience, CCG items in HTML encoded CCG phrases, whether they are syntactically represented as elements or as attributes, will be referred to hereinafter as CCG attributes.

A CCG phrase includes at least one of the following identifiable types of CCG-data attributes:

- industry, product, service, and/or subject classifications,
- contact categories, contact person(s) and/or organisation(s) names, titles or associations, contact details including physical and postal addresses, telephone and fax numbers, email and Internet or network addresses or locations, public keys, and
- geographic location details.

A CCG phrase may also include any of the following identifiable types of CCG control attributes:

- database control attributes to indicate which parts of the data are to be used to update databases, and
- display control attributes to indicate how browsers are to display the data.

By virtue of occurring in the same CCG phrase, a plurality of CCG-data attributes are associated with each other.

By virtue of their occurrence in the same CCG phrase, CCG-data attributes are idententified as a set of associated attributes. However the degree of association between attributes can be controlled by the inclusion in the phrase of database control attributes.

The start and end of CCG phrases should be identifiable to clearly distinguish these phrases from other data. To identify the beginning and end of a CCG phrase, at least one HTML element should have a CCG specific HTML element name or CCG specific attribute name or

CCG specific value. Each CCG attribute may consist, with or without other incidental characters, of a CCG attribute name and/or a CCG value or values. Preferably, each CCG phrase is contained in the "<body>" of the web page.

Two examples of a CCG specific HTML element are: "<CCG ...>" or "<CCG ... />" or "<CCG>...</CCG>". (Where a CCG phrase is coded in XML, the elements "<XML>" and "</XML>" may also be needed at the start and end of the CCG phrase.) A less satisfactory example is: "<!–CCG ...–> where the characters "CCG" after HTML comment element name "!–" are used to signify that the comment contains CCG-data. An example of the use of a CCG specific attribute name is: "<START CCG>"..."<END CCG>". An example of the use of a CCG specific value is: "<START TYPE='CCG'>"..."<END TYPE='CCG'>". Obviously, other character strings could be substituted for the element name, element attribute name or element attribute value "CCG" string of the examples.

The codes "<CCG ...>" and "<CCG ... />" are compatible with most HTML specifications, but being non-standard HTML, most web browsers do not display any text or attributes (eg PQ="AQD") within the angle brackets "<" and ">". These codes are preferred where display of the CCG data is not required and compatibility with older browsers is required (eg CCG phrases containing only classification values).

From one aspect, therefore, the invention comprises a web page for posting on a network, the web page being characterised by the inclusion of at least one CCG phrase in the "<body>" of the page, the CCG phrase being such that the CCG attributes contained therein are accessible and identifiable by (i) HTML compliant editors and/or (ii) HTML compliant web crawlers for the automatic construction of databases of classified information, and/or (iii) HTML compliant browsers for display on the computer screens of network users.

From another aspect, the invention comprises a method of constructing web pages of the above described type. The web pages may be constructed on digital computers using simple text editors such as Microsoft Windows Notepad, or preferably, purpose built human controlled editors or automated composing programs which embody knowledge of HTML and CCG syntax and grammar. Which ever process is used, CCG attributes are selected and inserted, modified, deleted and/or organised to form a valid CCG phrases in HTML encoded documents and the documents are posted on computer readable storage devices of computers connected to a computer network so that the documents are generally available to computers on the network.

From another aspect, the invention comprises a method of populating a database with CCG-data extracted from web pages. Web pages posted on a network are successively retrieved by a digital computer program (eg: a web crawler) and CCG phrases contained therein are identified and at least some of the CCG attributes found within the CCG phrases are extracted. The CCG attribute names are used to determine the type of data in the associated values. Generally the CCG attributes of interest are those relating to classification, contact and geographic data and database update controls while the attributes of little or no of interest in relation to database updating are those relating to display controls. Of course, the CCG-data extracted need only be that relevant to the particular database being updated. For example, one database may have been designed to index only web page classifications and URLs while another database may have been designed to index only contact details. Databases also differ in their internal representation of data and means of associating data. For example, some use

"flat file" tables, others use pointers to data to create network associations while others use hashing and buckets.

The conventional nomenclature differs considerably between different types of database. Depending on the particular database nomenclature, data of the same type is said to be stored in table columns, fields, attributes and properties. The terms column and field are somewhat related to the physical representation of the data in files while attribute and property is more related to the logical representation of data. To avoid confusion, with the terms "HTML attribute", "CCG attribute" or just "attribute", hereinafter a database property means both a type of data stored in the database and a place in the database where data of the same type is stored. Database properties are referred to by a name ("property name") or similar reference and contain values. For example, a database property with the name "City name" and which contains values which are all the names of cities may be defined as a "City name" type database property.

Whichever style of database is used, it is preferred that the database update program relate the CCG attributes to corresponding database properties used by the database update process so that the database property values are updated with CCG values in a manner which preserves the distinctness, content and meaning of the CCG values and, preferably, preserves the CCG value associations expressed in the CCG phrase as sets of associated database property values of different types.

In some cases, it is desired to know the address of the web page from which the CCG values were extracted. For example, the purpose of building a database might be to allow searching of the database by web page classification to provide a list URLs of web pages or URLs of portions of web pages which contain matching CCG classifications. The URLs could then be inserted in an HTML document and transmitted to a web browser as a list of references to web pages matching a search expression. In that example, associating the URL of a web page or the URL of a portion of a web page with the CCG values extracted from the same web page or web page portion is important and the URL or means of reconstructing it must be available and supplied to the database update process. In one style of database, the values of the same type are held separate rows in a column (property) of a database table, and pointers held in another column (property) are associated with the values by sharing the same table row. The table row constitutes a set of associated property values. Each pointer points to a bucket (block of data) containing a list of URLs or pointers to URLs held in a separate bucket or table. In another style of database, values of different types are held in different tables together with a set number, pointer or similar code which is used to indicate which values are associated as members of the same set. In one variation, the values of set members are prefixed with a code indicating the type of value and all values are held in the same column of a table. If the purpose of the database is to hold contact data, recording the web page URL in the database might not be required although if the URL is not present in the database, updating changes in the CCG contact details contained within a web page is more difficult. Of course, one database may be used to record all types of CCG values contained in web pages and associate with each other any and all values extracted from the same web page or even from other web pages.

From another aspect, the invention comprises a method of searching the databases constructed as outlined above. These databases may be used for a variety of searching purposes. For example, to find web page URLs by using the association of web page URLs with industry, service, product or subject classification or a person's or organisation's name or

address or geographic location values or any combination thereof. In another example, the databases may be used to find the contact details for people or organisations by name or location of industry, service, product or web page subject type and so forth by using the association between items of the contact details in the database without having to retrieve web pages associated with the contact details.

More particularly, the searching method involves finding URL references, or finding sets of associated database property values, from databases containing CCG-data. The method including steps of parsing a query phrase received from a computer network to extract query relational expressions and, from each expression, deriving a query field name, query relational operator and query value, determining the type of the query field by reference to its name, relating the query field to a corresponding database property according to type and locating CCG-data database property values in the database property which return a true value when tested against the query value using the query relational operator. Finally, the URL references or the sets of property values associated with the so located CCG-data database property values are extracted.

Database queries are usually expressed in a query language in the form of a phrase or sentence. In query by example style enquiry systems, the user types values into input fields on a form and a program extracts the input values and uses the values to automatically compose a query phrase or sentence. There are many existing examples of query languages used in connection with databases. Generally, they consist of relational expressions (eg Field=Value), logical expressions and grouping of relational and logical expressions by means such as parentheses. They may also contain sorting and output formatting expressions. Often abbreviated notation is used in the expressions such as leaving out field names or relational operators which are then inferred from the value in the expression or implied by default. In an enquiry the nature and format of the output may also be implied, such as a list of URLs of web pages or a list of contact details. Whatever is the mechanism of any particular database, the query expression needs to be parsed and fields in the query expression, explicit, default, implied or inferred, need be related to database properties of similar type. In some styles of database enquiry the query expression is evaluated against each row of a table or record of a file to find rows or records (ie a set of associated property values) which match the query expression. In other styles, sub-sets of the values of the properties are selected according to the interpretation of relational expressions in the query expression and the sub-sets are combined according to logical and grouping expressions in the query to find the sets of associated property values which match the query expression. Often, to make logical operations which combine the selected sub-sets more efficient, it is not the values which are selected but pointers to the values (eg Table name and table row) or unique keys (eg URLs or pointers to URLs) associated with the values. For example, the AND logical operator is often used to combine two lists so that only values or pointers or keys common to both lists are found in the combined list. Usually, the query produces a result list which is then provided to other processes. For example, a list of URLs of web pages is processed to produce an attractively formatted HTML encoded document containing the URLs and is sent to a web browser to allow an enquirer to retrieve interesting web pages. In another example, the contact details associated in the database with each value or pointer in the result list are retrieved from the database and presented as a report in the form of an HTML encoded document and is sent to a web browser for viewing.

From another aspect, the invention comprises a method of displaying CCG-data contained in CCG phrases within web pages which are displayed by a web browser executing on a digital

computer. While a web page is loading or has loaded in a web browser, the web browser parses the web page and displays the text (or data) of the web page on a display device connected to the computer. When the web browser parser encounters CCG phrases, the web browser may display the CCG-data (element and/or attribute names (or translations of element and/or attribute names) and/or values) in a number of browser specific ways. For example, the web browser may by default not display any CCG-data, display all CCG-data, not display any CCG-data until a CCG display control attribute explicitly states that subsequent data should be displayed or display all CCG-data until a CCG display control attribute explicitly states that subsequent data should not be displayed. The web browser may also use CGA display controls specifying the size, font, position and so forth to alter the display of the CCG-data.

DESCRIPTION OF EXAMPLES

Having indicated the nature of the present invention, examples or embodiments thereof will now be described by way of illustration only.

Example 1: HTML Syntax Suitable for Representing a CCG Phrase

The following is an example of HTML element syntax suitable for representing CCG phrases in which a control (e.g. "SHOW") may be "good until countermanded" and thus apply to more than one field:

```
<CCG HREF="url"
{{NAME="label"  |  ID="identifier_code"}  &|  {LANG="language_code"  &
CLASS="Class_name"}
{
    {SET_SEPARATOR} &|
    {INDEX | NOINDEX} &|
    {SHOW | HIDE} &|
    {XPOS="horizontal_position_number"} &|
    {YPOS="vertical_position_number"} &|
    {NEWLINE} &|
    {ALIGN=centre | left | right | justify} &|
    {SIZE=[+/-]1 | 2 | 3 | 4 | 5 | 6 | 7} &|
    {COLOR="#rrggbb" | "colour_name"} &|
    {FACE="type_face_name"} &|
    {BLINK &| BOLD &| UNDERLINE &| ITALIC &| STRIKE} &|
    {SUBSCRIPT | SUPERSCRIPT} &|
    {CLEAR{=left | right | all}}
    {NORMAL} &|
    {{{CONTACT &| COPYRIGHT &| DEVELOPER} &|
    {PERSONAL &| BUSINESS &| ASSOCIATION} &|
    {attribute_name="attribute_value(s)"}
}
...
>
```

where: the ellipsis "..." implies optional repetition of the braced ("{" "}") items; the braces are used to group items and are not CCG syntactic elements; "&" (and) implies items must occur together; "|" (or) implies only one item must occur; and "&|" .(and/or) implies any including none of the items may appear together.

Using the syntax of this example, each CCG phrase is represented as an HTML element, the element name being "CCG" and the CCG-data (eg attribute_name="attribute_value") and CCG

controls (eg SIZE=+1) are represented as attributes of the HTML element. Some of the attributes (eg SIZE) having explicit values (eg +1) and some attributes have implied values depending on the presence or absence in a CCG phrase (eg when the attribute BUSINESS is present it has the implied value of True and the implied value of False when absent).

Representation in XML syntax requires, at most, only a simple translation. All the items, such as "NORMAL" and "attribute_name" may remain unchanged as attributes of the element named "CCG" (eg <CCG size=+1/>). However, when a CCG phrase is encoded in XML, it is preferred that the items are represented as XML elements. For example attribute "SIZE=+1" can be represented as element "<size>+1</size>" or "<size value=+1/>" and "NORMAL" can be represented as "<normal/>.

In this example, the attributes, ID, LANG and CLASS take their meanings from HTML 3.0. The "url" in HREF="url" or may be a link with or without destination anchor labels. For example the URL http://www.w3.org/docs.html does not contain a destination anchor label (or identifier) while http://www.w3.org/docs.html#searching does contain the destination anchor label "#searching" which is intended refer to an anchor in docs.html such as <A NAME="searching">...</A>. There is some confusion in various HTML standards documentation about the distinction between the expression NAME="label" and the expression ID="identifier_code". For most practical purposes the two expressions have the same function or meaning: to uniquely identify within a document a position in or portion of that document.

Database control attributes:
"Set_separator" indicates the end of association between preceding and following data other than through the weaker mutual association with the same CCG phrase or web page; the data are divided into sets. "Index | Noindex" indicates that the following data are / are not to be indexed by a web crawler. These attributes have an implied attribute value of 'True' if present in and 'False' when absent from a CCG phrase.

Display control attributes:
"Show | Hide" indicates that a browser should show / not show the following data. Xpos and Ypos indicate the position (for example in pixel or physical units) on the browser screen where the data is to be displayed. "Newline" may be used in addition or as an alternative method of placing text on a browser screen. "Align" indicates the positioning of data on a browser screen relative to the cursor position set by "Xpos", "Ypos" or "Newline". "Size", "Colour" and "Face" indicates the size, colour and type face or font of the following data when displayed on an browser screen. "Blink", "Bold", "Underline", "Italic", "Strike", "Superscript" and "Subscript" indicates that the following data should be displayed blinking, bold, underlined, italicised, struck through, superscripted or subscripted. "Clear" indicates that the browser screen in the region where data will be displayed should be cleared to background before displaying the following data. "Normal" indicates the data is to be displayed without the "Blink", ..., "Clear" characteristics. The display controls which consist of an attribute name without an explicit value have an implied value of 'True' when present and 'False' when absent.

CCG-data attributes:
"Contact &| Copyright &| Developer" indicates that the following CCG-data refers to details for a person or organisation and/or to the copyright owner and/or to the HTML or web page developer. "Personal &| Business &| Association" indicates that the following data refers to details for a person and/or business and/or association. The previous CCG-data attributes have an implied attribute value of 'True' if present in a CCG phrase or set and 'False' when

absent from a CCG phrase or set. The attribute_name could be standard CCG attribute names or synonyms of standard CCG attribute names or abbreviations of CCG attribute names which refer to the following types of CCG attribute values where square brackets "[" and "]" surround suggested attribute names:

- industry or service or product or subject classifications and sub-classifications:
  - classification name [CN],
  - classification codes [CC].
- display only text [TEXT].
- contact:
  - person:
    - courtesy title [PNC],
    - first given name [PNG],
    - other given names [PNO],
    - family name [PNF],
    - name suffix [PNS],
    - qualifications [PQ],
    - associations [PA],
    - contact person title [PT],
    - contact person role [PR].
  - organisation:
    - name [ON],
    - unit [OU],
    - identifier [OID].
  - physical or post or delivery address:
    - type [AT] (= "PHYSICAL" &| "POST-OFFICE" &| "POSTAL" &| "DELIVERY")
    - post office box number [AP#]
    - post office name [APN]
    - room or suite or office or unit or flat or apartment name &| number [AB#],
    - floor name &| number [ABF],
    - building name [ABN],
    - lane or street or road or highway number [AS#],
    - lane or street or road or highway name [ASN],
    - suburb or town or city name [ACN],
    - region or state or territory or province name [ARN],
    - post code [APC],
    - country or nation name [ANN],
  - telephone:
    - type [TT] (= "PREFERRED" &| "VOICE" &| "MOBILE" &| "CAR" &| "MESSAGE" &|"PAGER" &| "FACSIMILE" &| "MODEM" &| "ISDN" &| "VIDEO")
    - nation or country code number [TC#],
    - trunk access number [TT#],
    - area code number [TA#],
    - local number [TL#],
  - email:
    - type [ET] (= "INTERNET" | {other}),
    - mailer [EM],
    - address [EA],
  - Internet address:
    - url [IURL].
  - date & time:

- date & time from [DTF],
- date & time to [DTT],
- weekday from [DTWF],
- weekday to [DTWT],
- weekday time from [DTWFT],
- weekday time to [DTWTT],
- time zone [DTZ].

- brand name [BN].
- public key:
  - key type[KT].
  - key [K],
- geographical:
  - location units [GLU],
  - location [GL],
  - serviced region units [GLRU].
  - serviced region [GLR],

Suggested attribute name [CN] is the name of an attribute associated with the attribute value containing "classification name" type data. For example, the [CN] attribute value could be the name of a proprietary or national or international or other industry classification standard such as the Australian and New Zealand Standard Industry Classification or "ANZSIC" for short or the U.S. Bureau of the Census Industrial Classifications (USBCIC). The associated classification codes [CC] attribute value could contain the codes and/or descriptions of the codes of the named standard with or without modifications, deletions or extensions. For example: CN="ANZSIC" CC="61;Road transport" or CN="USBCIC" CC="581;Hardware store". Service classifications such as the International Standard Classification of Occupations could be used. For example: CN="ISCOO" CC="4430;Auctioneer" Product classifications such as the Harmonised Commodity Description And Coding System could be used. For example: CN="HSC" CC="8411;Turbojets, turbo-propellers & other gas turbines; parts thereof" For subject classifications, Dewey Decimal, and/or Universal Decimal and/or Library of Congress and/or Bliss and/or Colon Classification could be used. For example: CN="DDC" CC="577.699;Sea shore ecology" The inclusion of subject classifications provides a very simple, straightforward method of classifying the subject matter of an HTML document which could be attractive to commercially oriented copyright owners.

The text ([TEXT]), person ([PNC] - [PR]), organisation ([ON] - [OID]), physical or post or delivery address ([AT] - [ANN]), telephone ([TT] - [TL#]), email address ([ET] - [EA]) and Internet address [IURL] are intended to be associated with each other in the obvious manner. Date & time(s) ([DTF] - [DTZ]) are intended to indicate the times at which the address and/or telephone and/or email will be serviced by the associated person(s) and/or organisation(s). The brand name ([BN]) attribute is intended to hold commercial brand names. Public key ([KT] - [K]) is intended to hold public encryption keys for secure communication with the contact person or organisation.

The geographical location [GL] could be a latitude and longitude (eg E148D31'12.5",S36D40',09.6" or E148.5201,S36.6693 or -148.5201,-36.6693), or a Universal Grid Reference (eg 55FV364402) or other global, national, regional or local location reference with units as specified [GLU], which is typed in or obtained by pointing to a digitally encoded map or other methods. In more populated regions of some countries such as the U.S., street addresses and post codes are associated with a moderately accurate geographic location and

can be used to interpolate geographic location data where geographic location data is not explicitly stated in the CCG-data. Using a universally recognised code such as latitude and longitude has advantages when used with international mediums like the Internet. Geographical location is intended to be associated with a post, delivery address or physical address such as place of business or residence. A CCG compliant browser could use this reference to display a map centred on that geographic location. The purpose of the geographical location data is to allow browser users to specify search engine search criteria which will result in the search engine selecting only those Internet accessible documents which provide details about providers which are within a specified region. The serviced region [GLR] is intended to indicate the preferred area of operation of providers expressed in terms of serviced region units [GLRU]. A radial distance (eg in kilometres) or alternate means of expressing an area of interest around a geographic point, such as polygons, are envisaged.

It is envisaged that the CCG attribute_value could be composed of more than one value (actually sub-value) wherein specific characters or character strings separate individual values.

While specific instances of element names and types have been given in this example, of more importance is the type of data and type controls over the display and indexing of the data. As an alternative to the preferred immediately following example where the CCG-data is lumped together under the HTML element named "CCG", certain elements of the data, for example the classification data, could be lumped under separate HTML elements with distinctly different names thereby separating CCG classification data from CCG contact data. However, this is not preferred because the strength of association between the two types of data is weakened.

Example 2: Classification of Portion of a Web Page.

Where it is desired to classify a portion of a web page, such as a paragraph about a product, simple CCG-data may be used in conjunction with the syntax of Example1. For example:

```
<A NAME="Radios">AM-FM radio receivers: </A>
    <CCG HREF="#Radios">
        CN="ANZSIC"
        CC="E23.34.78;Electrical equipment - radio receivers AM"
        CC="E23.34.79;Electrical equipment - radio receivers FM"
    </CCG>
We won't be beaten on the price of these high quality receivers ....
```

In this example, the CCG prase appears after the related anchor (<A NAME=...</A>). However, while such proximity visually provides an obvious association between the anchor and related CCG phrase, it is intended that CCG phrase containing the attribute HREF related to a specific anchor could appear anywhere within the body of a web page and remain related to the named anchor. The CCG phrase containing the attribute HREF could appear in a separate document and thereby relate the CCG-data to the entire document or to a named anchor although, as previously noted, coordinating separate documents can be problematic. In the absence of the HREF and NAME attributes, it is also intended that the CCG-data apply to the whole web page.

Example 3 Classification of Portion of a Web Page using XML Syntax

Using XML syntax and similar attribute names to those of Example 2 the HTML fragment of Example 2 may be rewritten as:

```
<A NAME="Radios">AM-FM radio receivers: </A>
    <XML>
```

```
<CCG>
    <HREF>"#Radios"</HREF>
    <CN>"ANZSIC"</CN>
    <CC>"E23.34.78;Electrical equipment - radio receivers AM"</CC>
    <CC>"E23.34.79;Electrical equipment - radio receivers FM"</CC>
</CCG>
</XML>
```

We won't be beaten on the price of these high quality receivers ....

This example demonstrates that the translation of CCG-data from HTML to XML (and the reverse) involves simple syntactical and grammatical translations. Of course, the resulting HTML and XML, while "well formed" might not be recognised or, if recognised, might not be understood by some parsers.

Example 4: Constructing a Web Page Containing CCG-data

As an example, a web page developer, Alice Jamieson, is preparing an advertisement for a local electrician John Williams, trading as Kelso Electrical, who wants to advertise on the web for business within 30 kilometres from his office located at 18 Raglan Street, Kelso, New South Wales. Alice uses a graphical user interface web page authoring tool capable of creating and modifying web pages containing HTML (and XML) CCG phrases by accepting inputs from a user. The tool executes on a digital computer having input devices such as a keyboard, mouse, light pen and touch pad, display devices such as a CRT, LED arrays, liquid crystal arrays and computer-readable media such as magnetic and optical disks, memory arrays, magnetic tape and the like.

The authoring tool also embodies knowledge of the content and structure of CCG phrases such as the attribute names, valid ranges and sets of associated attribute values, the normal order of the attributes in the CCG phrase and interdependencies between attribute values. The tool provides a window where web pages may be viewed in layout (browser) mode and another window where the HTML code may be viewed in editing mode. The tool also provides means of inserting, deleting, modifying and organising HTML elements, changing font size, face and colour and so forth. The tool provides means for the user to build CCG phrases by using input devices to select an edit control representing various types of CCG attributes from a list which the tool then inserts in the body of a web page together with, when not already present, HTML code indicative of the start and end of a CCG phrase. The user then types in the value in the attribute. Similarly, the tool provides means of converting web page text to CCG attributes. Using input devices, the user selects the text to be converted to a CCG attribute then selects an edit control from a list; the tool then inserts the HTML code necessary to encode the text as a CCG attribute. However, these semi-manual methods of creating and modifying CCG phrases are inefficient and error prone. The tool also provides a button, which can be activated by using input devices, for access to CCG phrase editing functions. The CCG editing functions consist of a means of extracting the CCG values from existing CCG phrases in the web page being edited, forms for entering and modifying the extracted CCG values, a layout view browser window for altering how the CCG-data displays (position, font size, face, colour, bold, normal, hiding or showing and so forth), a data view browser window to alter which CCG-data values are to be indexed or not indexed in search engine databases, and a means of deleting existing CCG phrases from web pages and inserting new or changed CCG phrases in web pages. Editing cursors marking the current location at which text and/or data may be inserted, deleted or modified are provided in each window and form.

In the current example, the web page initially contains no CCG phrase. Clicking the CCG editing function button of the authoring tool causes a form to appear. The form contains prompts related to CCG attribute names and associated data input fields related to the CCG attribute values associated with the CCG attribute names, that is CCG-data. The fields are blank because, in the web page layout view, the edit cursor is not over a CCG phrase (and can not be since the web page initially contains no CCG phrase). The service classifications relevant to the web age, John Williams physical business contact address, phone and fax numbers, email address and geographic location and his post office business contact addresses are entered into the forms using a keyboard and mouse. The developer, Alice Jamieson, also includes her basic contact details where provided for on the form. The forms use drop down lists to select address blocks (eg physical and post office) for editing. Logic associated with the forms validates the CCG attribute values and interdependencies. Input devices are then used to control the CCG-data layout view browser to modify the appearance of the CCG-data such as font size and colour and positioning. In the layout browser, input devices communicating with the edit cursor are used to highlight individual items and blocks of items to be changed. The post office address is highlighted as a block and moved into position in line with the physical address. The CCG-data view window is then used to check which data items are to be indexed by search engines. In this example all CCG-data (ie all CCG attribute values except display control values and database control values) are to be indexed. Input devices are used to control the edit cursor to highlight the entire data and a mouse is used to click (activate) a button to mark all the data for indexing. Then another button is clicked which builds an HML encoded CCG phrase of CCG attributes derived from the CCG-data values, display control values and database control values and inserts the CCG phrase in the web page at the location pointed to in the web page layout browser window.

The HTML code editing mode window was called up which revealed the following HTML encoded CCG phrase in the web page:

```
<XML>
<CCG>
    <INDEX/>
    <HIDE/>
    <CN>ANZSIC</CN>
    <CC>D36.11.45;Electrical contractors - residential</CC>
    <CC>D36.11.46;Electrical contractors - industrial</CC>
    <SHOW/>
    <CONTACT/> <COPYRIGHT/>
    <BUSINESS/>
    <XPOS>50</XPOS>
    <YPOS>320</YPOS>
    <ALIGN>centre</ALIGN>
    <SIZE>3</SIZE>
    <COLOR>black</COLOR>
    <FACE>Times New Roman</FACE>
    <BOLD/>
    <CLEAR>all</CLEAR>
    <TEXT>Contact :</TEXT>
    <PNC>Mr</PNC>
    <PNG>John</PNG>
    <PNF>Williams</PNF>
    <PQ>AIE</PQ>
```

```
<PA>ARUC</PA>
<NEWLINE/>
<PT>Managing Director</PT>
<NEWLINE/>
<ON>Kelso Electrical Pty. Ltd.</ON>
<NEWLINE/>
<NORMAL/> <ITALIC/>
<SIZE>-2</SIZE>
<TEXT>NSW License 45678C</TEXT>
<NEWLINE/>
<NORMAL/> <BOLD/>
<SIZE>+2</SIZE>
<AT>PHYSICAL</AT>
<AS#>18<AS#>
<ASN>Raglan Street<ASN>
<NEWLINE/>
<ACN>Kelso</CAN>
<NEWLINE/>
<ARN>NSW<ARN>
<NEWLINE/>
<HIDE/>
<ANN>Australia</ANN>
<NEWLINE/>
<SHOW/>
<TEXT>Phone:</TEXT>
<TT>PREFERRED ; VOICE ; MESSAGE</TT>
<HIDE/>
<TC#>61</TC>
<SHOW/>
<TT#>0</TT#>
<TA#>63</TA#>
<TL#>456-7828</TL#>
<TEXT>  Fax:</TEXT>
<TT>FACSIMILE</TT>
<HIDE/>
<TC#>61</TC#>
<SHOW/>
<TT#>0</TT#>
<TA#>63</TA#>
<TL#>456-7829</TL#>
<NEWLINE/>
<ET>INTERNET</ET>
<EA>johnw@firefly.com.au<EA>
<TEXT>  </TEXT>
<GLU>LatLong</GLU>
<GL>="33.3978S;148.5679E</GL>
<GLRU>Km</GLRU>
<GLR>30 </GLR>
<SET_SEPARATOR/>
<XPOS>250</XPOS>
```

```
        <YPOS>320</YPOS>
        <NEWLINE/>
        <NEWLINE/>
        <TEXT>Or write to us at :</TEXT>
        <NEWLINE/>
        <ON>Kelso Electrical Pty. Ltd.</ON>
        <NEWLINE/>
        <AT>POST-OFFICE</AT>
        <AP#>P.O. Box 187</AP#>
        <NEWLINE/>
        <APN>Sunny Corner</APN>
        <TEXT> </TEXT>
        <APC>2795</APC>
        <NEWLINE/>
        <HIDE/>
        <ANN>Australia</ANN>
        <SET_SEPARATOR/>
        <HIDE/>
        <DEVELOPER/>
        <BUSINESS/>
        <PNG>Alice</PNG>
        <PNF>Jamieson</PNF>
        <ET>INTERNET</ET>
        <EA>alijam@firefly.com.au</EA>
        <IURL>http://www.firefly.com.au/~aljam/<IURL>
    </CCG>
    </XML>
```

In the web page layout browser window the CCG-data displayed as follows:

Contact :
Mr John Williams, AIE, ARUC,
Managing Director
Kelso Electrical Pty. Ltd.
NSW License 45678C
18 Raglan Street
Kelso
NSW
Phone: 063-456-7828 **Fax**: 063-456-7829
Email: johnw@firefly.com.au Map

Or write to us at:

Kelso Electrical Pty Ltd
P.O. Box 187
Sunny Corner 2795

Having encoded the web page in this way, Alice then posts it on the storage device of a digital computer connected to the Internet from where it can be retrieved through the Internet using the URL "http://www.firefly.com.au/~johnw/index.html"

Example 4: Constructing a Database from Web Pages Containing CCG-data

During a routine sweep of Internet connected web page servers, a web crawler (or robot) operating on a server named "ccg.search.com" executing on an Internet connected digital computer discovers the URL "http://www.firefly.com.au/~johnw/index.html" in a document it had previously retrieved through the Internet. The web crawler decides that the URL matches it's selection criteria because the URL contains the suffix ".html". The web crawler then

successfully retrieves the document by extracting from the URL the address of the computer hosting the document, addressing and sending a message (including the address of the web crawler) requesting the web page through the network to the web page host computer using TCP/IP protocol, the host computer then reads the document, addresses and sends the document to the web crawler using TCP/IP protocol, the web crawler then waiting until it has received all parts of the web page from the host computer before proceeding. It inspects the contents of the document and finds that it matches the additional selection criteria that it is an HTML encoded document. The web crawler program, depending on its state and logic, then parses the document, strips out and saves some or all of the URLs in the document for future examination. The web crawler program then passes the document, together with the URL of the document through a network communications channel to an indexing program executing on a different computer. The indexing computer has database updating software which manipulates a database stored on computer-readable media.

The indexing program parses the document, from first to last character, indexing some of the meta data in the <head> of the document and the words in the text of the document with respect to the document URL. In the database of this example, unique words extracted from the documents already indexed are held in separate rows of a column of a database table and in another column of the same table on each row is an associated pointer to the first bucket or block of URLs of documents containing the word associated with the pointer. As new words are found, the new word is added as a new row in the word column of the table, a new bucket is created, the URL of the document containing the new word is inserted into the bucket and a pointer to the new bucket is written in the new row pointer column. When the same word is found in another document, the row in the table of the word is found, the pointer is retrieved from the table, the bucket pointed to by the pointer is retrieved and the URL of the other document is inserted in the bucket. Where a bucket becomes full of URLs, a new bucket is created and a pointer to the new bucket for holding additional URLs is placed in the full bucket. Deletion of words and URLs of changed or no longer existing documents is also provided for.

In addition to indexing words extracted from the text of the document, the indexing program also indexes the CCG-data in the document as well as indexing words found in the CCG-data. When the parser finds HTML element "<XML>" in the document it switches into XML parsing mode and switches out of that mode when "</XML> is found. When the element "<CCG>" is found, the parser switches into the CCG parsing mode and switches out of that mode when "</CCG>" is found.

The example database has a CCG-data attribute name to database property name correspondence table to show the relationship between the CCG-data attribute names and the database tables and columns (properties) where the CCG-data attribute values are to be stored in the database as database property values. The database property values and associated URLs are stored in much the same way as for words extracted from text as outlined above. However, CCG contact data, for example, which consists of several distinct CCG-data attributes which are related (eg street name, city), is stored in a database table having a column (property) related to each distinct CCG contact attribute name and each separate CCG contact data set (eg person's name, address, telephone number) as separated by "<CCG>", "<SET_SEPARATOR>" and "</CCG>" is held in a separate row in the table. The values stored in each row are considered to be a set of associated property values of different types.

The indexing program, during parsing the document of Example 2 above, encounters the "<CCG>" element and enters the CCG parsing mode. The parser knows to ignore display control attributes and to consider database control elements in the CCG phrase. The example indexing program opts to index all other CCG-data contained in the attribute values until explicitly instructed not to index the attribute values by encountering the "<NOINDEX/>" database control element and then to recommence indexing when the "<INDEX/>" database control element is encountered.

Taking each CCG-data attribute name and associated attribute value(s) in succession, the example indexing program uses the correspondence table to translate the CCG-data attribute name to the database table and column (property) names where the CCG-data attribute value(s) are to be stored as database property value(s). The indexing program may opt to translate the CCG-data attribute values to database property values by, for example, converting character strings of digits to binary encoded decimal representation, the string "True" to a single bit representation and the like. The indexing program then adds or updates the database property value(s), using the database table and column (property) names (or similar references) obtained by translation, in much the same manner as outlined above for the update of the database using words extracted from the document text, including associating the data to the document URL where desired. Where the CCG-data contains a "HREF" attribute (or similar), the URL associated with the other CCG-data is a URL taken from the "HREF" attribute value or composed of the document URL and the "HREF" attribute value if the attribute value is a partial or relative URL. Some CCG attributes, such as "<BUSINESS/> have only an implied value of true if the attribute is present and false if the attribute is absent. the "<SET_SEPARATOR/>", "<CCG>" and "</CCG>" resetting such values to false. However, where attribute value(s) associated with different attribute names are still related, such as a person's name and a street name, the related values of different types are stored on the same row of the same database table but in a different column (database property) to preserve the relationship. "<SET_SEPARATOR/>" limits the degree of relatedness between, for example, a person's name occuring before the separator and a street name occurring after the separator. Using the example document and using the same database column (property) names as used for the CCG-data attribute names a portion of the table constructed database table would look like:

	PNC	PNG	PNF	PQ	PA	PT		URL
...	...	...	...	...	...	...	...	...
...	Mr	John	Williams	AIE	ARUC	Managing Director	...	(pointer)
...	...	...	...	...	...	...	...	...

Difficulties not highlighted by this example are the need to handle properties having multiple values of the same type, "sparse rows" where only a few values are not null (blank) and tables with extremely large numbers of rows. For example, the CCG-data of this example could have contained multiple values of personal qualifications ("PQ"). To represent this type of data using a 2 dimensional table database system, the database would be "normalised" so that the multiple values were stored in a separate table and keys or pointers were used to relate the relate the items in the two tables. Numerous alternate database systems, for example those based on key hashing and data buckets, or tagging data values with prefixes or suffixes related to the type of data value may be used. Preferably, however, whatever database system is used, it should preserve the associations of CCG-data items present in the CCG phrases.

Because the geographic location data was missing from the postal address of the CCG-data in the example document, but a post code was present, the indexing program inferred the geographic location from the post code.

Example 6: Finding Web Page References Using a CCG Database

As an example, Kevin Robson lives in Sydney but owns and has rented out a house in Bathurst. He wants to use the web to find some electricians based in the general Bathurst region (not only in Bathurst City) to contact for estimating the cost of modifying the wiring in the house. He uses his web browser to open the web page "http://www.ausline.com.au/web_search.html" containing AusLine's search engine web page search criteria input form encoded using the HTML "<form>" element.

The search criteria input form contains several input fields including those labelled "Service classification", "Key words", "City./Suburb/Town", "Country", "Lat/Long" and "Radius". The form also displays a button labelled "Map" to allow latitude and longitude to be selected by pointing to map images. The word "electrician" is typed into the "Service classification" field, "house wiring" into the "Keywords" field, "Bathurst" into the "City/Suburb/Town" field and "10" into the field "Radius". The country "Australia" was already showing in the country field because the web page server had received cookie data from the browser indicating that that was the country used when the browser last used the web page. The "submit search" button on the web page was clicked. The browser transmitted a message using TCP/IP protocol to the AusLine server containing the input field values encoded in the header of the message.

After a short delay, the search result HTML encoded web page was returned. Clicking on the "Service classification" input field drop down list box to check the classifications used in the search revealed three items:

- Electrical contractors - residential
- Electrical contractors - industrial
- Electrical engineers

The search engine attached to the server obtained those classifications by using word stemming and searching the text of the service classifications held in it's database. The Lat/Long field contained the value "33.3856S;148.5743E" which the search engine obtained by looking up the latitude and longitude of the town "Bathurst" in the country "Australia" in it's database. Clicking on the "Map" button retrieved a web page having the image of a map centred on the town of Bathurst and showing the area 20 Km around it. The search engine obtained the map by making a request to another Internet connected server and supplying the latitude, longitude and radius. Clicking on the browser "Back" button returned to the search results page.

The search results contained 8 titles, brief descriptions and URLs including a reference containing the URL "http://www.firefly.com.au/~johnw/index.html". Retrieving each in turn revealed that all were well focused according to the search criteria being related to electricians, electrical contractors and engineers in the Bathurst area. The search engine obtained these references to web pages by:

- searching it's database of service classification titles with words stemming from "electrician" which resulted in three service classification codes,
- searching it's database using the three service classification codes to obtain an intermediate list of URLs of web pages containing those CCG codes

- searching it's database for the two keywords to obtain an intermediate list of URLs of web pages containing those words in the web page text,
- Searching it's database to find the latitude and longitude of Bathurst, Australia,
- searching it's database to obtain an intermediate list of web pages which contain latitude and longitude data lying within 10 Km of the latitude and longitude of Bathurst, Australia,
- producing as a result list, a list of URLs which are common to all the intermediate lists,
- obtaining from it's database the title and brief description of the web pages,
- formatting the titles, descriptions and URLs into an HTML encoded report,
- transmitting the report to the enquiring web browser.

Example 7: Finding Contact Details Using a CCG Database

As an example, Jim Jones of Jones and Sons wants to send a recall notice about a faulty batch of UV stabilised electrical power cable to all Electrical contractors and Electrical wholesalers in Australia who have email addresses. He uses his web browser to open the web page "http://www.ausline.com.au/contact_search.html" containing AusLine's search engine contact search criteria input form encoded using the HTML "<form>" element.

The search criteria input form contains several input fields including those labelled "Service classification", "Country" and "Output format". The word "electric*" is typed into the "Service classification" field, the word "Australia" is typed into the "Country" field and the "Tabular - Name & Email" option in the "Output format" drop down list box is selected. The "Submit search" button on the web page is clicked. The browser transmits a message using TCP/IP protocol to the AusLine server containing the input field values encoded in the header of the message.

After a short delay, the search result HTML encoded web page is returned. Clicking on the "Service classification" input field drop down list box to check the classifications used in the search revealed too many classifications for the result to be sufficiently focused. The following four classifications were selected from the list:

- Electric cable - ducting systems
- Electrical contractors - residential
- Electrical contractors - industrial
- Electrical wholesalers

and the "Submit search" button is pressed again to refine the search.

The search results contained 3,473 names and associated email addresses and URLs to full contact details. Jim saved the search result page on his computer so that he could use his email program to send the recall notice to each email address in the list. The email address "johnw@firefly.com.au" was included in the list.

The search engine obtained these references to web pages by:

- searching it's database using the four service classification titles which resulted in four service classification codes,
- searching it's database using the four service classification codes to obtain an intermediate list of database primary keys of database table rows containing those service classification codes in the database Service classification attribute,
- searching it's database using the country name "Australia" to obtain an intermediate list of database primary keys of database table rows containing that word in the database Country attribute,

- producing as a result list, a list of database primary keys which are common to both the intermediate lists,
- obtaining from it's database using the result list the values of the name and email attributes,
- using the HTML <table> element to format the name values, email values and full detail URLs into an HTML encoded report,
- transmitting the report to the enquiring web browser.

This example relates to finding sets of associated database contact values without requiring references to web pages. However, finding other sets of associated database values such as sets of associated industry classification values and geographic location values might also be useful for some purposes.

Thus it is appreciated that the afore stated goals, advantages and objectives are achieved by the teachings herein. In particular it is seen that, unlike the prior art, efficiently searchable Yellow pages and White pages databases and the like may be automatically constructed from HTML encoded web pages. Additionally the database entries may be automatically linked to specific web pages and portions of web pages allowing convenient methods of indexing of product and service catalogues and the like. It is also appreciated that simpler methods of constructing databases suited to a variety of other uses such as industry and subject directories are also provided.

From the foregoing teachings and with the knowledge of those skilled in the art, it is apparent that other modifications and adaptations of the invention will become apparent. For example, the method steps disclosed and claimed herein may be practiced in a variety of different orders. CCG-data may take on a variety of different forms within the meaning of the claims. Thus, it is our intention to include within the scope of the claims not only the invention literally embraced by the language of the claims but to include all such modifications and adaptations which may come to those skilled in the art.

What I claim is:

1. An HTML encoded web page embodied on a computer-readable medium, said web page comprising at least one HTML encoded CCG phrase, each CCG phrase comprising:
   a) HTML code indicative of the start of a CCG phrase,
   b) at least one CCG-data attribute, and
   c) HTML code indicative of the end of a CCG phrase.

2. An HTML encoded web page embodied on a computer-readable medium, said web page comprising at least one HTML encoded CCG phrase, each CCG phrase comprising:
   a) HTML code indicative of the start of a CCG phrase,
   b) at least two CCG-data attributes,
   c) at least one database control attribute separating said CCG-data attributes into at least two sets of CCG attributes, and
   d) HTML code indicative of the end of a CCG phrase.

3. An HTML encoded web page embodied on a computer-readable medium, said web page comprising at least one HTML encoded CCG phrase, each CCG phrase comprising:
   a) HTML code indicative of the start of a CCG phrase,
   b) at least one CCG-data attributes,
   c) at least one attribute of: database control attributes, display control attributes; and
   d) HTML code indicative of the end of a CCG phrase.

4. A computer implemented method of building a web page comprising at least one HTML encoded CCG phrase, the method comprising the steps of:
   a) displaying a web page on a computer display device,
   b) displaying an edit cursor indicating a character position on said display device and a corresponding character position in said web page, said edit cursor being positionable within the display of said web page by use of computer input devices,
   c) separately displaying on said computer display device a set of edit controls representing CCG-data attribute types,
   d) positioning said edit cursor within said display of said web page using said input devices,
   e) selecting an edit control from said set of edit controls using said input devices,
   f) relating said selected edit control to a corresponding CCG-data attribute name,
   g) constructing a CCG-data attribute character string comprising a character string representing said attribute name and another character string representing an empty CCG-data value,
   h) if the said edit cursor is positioned outside a CCG phrase,
      i) inserting into said web page, at the character position indicated by said edit cursor, a start character string comprising HTML code indicative of the start of a CCG phrase,
      ii) inserting into said web page, immediately after the end of said start character string, an end character string comprising HTML code indicative of the end of a CCG phrase, and
      iii) positioning said edit cursor between said start and end character strings,

i) inserting said CCG-data attribute character string into said web page at the character position indicated by said edit cursor,

j) positioning said edit cursor at the character position in said web page of the CCG-data value of said inserted CCG-data attribute character string,

k) inputting characters using a keyboard,

l) inserting said input characters into said web page at the character position indicated by said edit cursor, thereby converting said empty CCG-data value to a non-empty CCG-data value, and

m) writing said web page on computer-readable media.

5. A computer implemented method of building a web page comprising at least one HTML encoded CCG phrase, the method comprising the steps of:

a) displaying a web page on a computer display device,

b) displaying a start edit cursor and an end edit cursor on said display device, each said edit cursors indicating a character position on said display device and a corresponding character position in said web page, said edit cursors being positionable within the display of said web page by use of computer input devices,

c) separately displaying on said computer display device a set of edit controls representing CCG-data attribute types,

d) selecting a string of web page characters on said display device using said input devices to position said start edit cursor to indicate the start said string of web page characters and said end edit cursor to indicate the end of said string of web page characters,

e) selecting an edit control from said set of edit controls using said input devices,

f) relating said selected CCG-data control to a corresponding CCG-data attribute name,

g) constructing a CCG-data attribute character string comprising a character string representing said attribute name and another character string representing a CCG-data value containing said string of web page characters,

h) deleting said string of web page characters from said wen page,

i) if the said start edit cursor is positioned outside a CCG phrase,

i) inserting into said web page, at the character position indicated by said start edit cursor, a start character string comprising HTML code indicative of the start of a CCG phrase,

ii) inserting into said web page, immediately after the end of said start character string, an end character string comprising HTML code indicative of the end of a CCG phrase, and

iii) positioning said start edit cursor between said start and end character strings,

j) inserting said CCG-data attribute character string into said web page at the character position indicated by said start edit cursor, thereby converting said string of web page characters to a CCG-data attribute value contained within a CCG-data attribute contained within CCG-phrase, and

k) writing said web page on computer-readable media.

6. A computer implemented method of building a web page comprising at least one HTML encoded CCG phrase, the method comprising the steps of:

a) displaying a CCG-data input form on a computer display device,

b) inputting CCG-data values into fields of said data input form using computer input devices,

c) inserting into the body of a web page a start character string comprising HTML code indicative of the start of a CCG phrase,
d) inserting into said web page body immediately after the end of said start character string an end character string comprising HTML code indicative of the end of a CCG phrase,
e) extracting successive field values from said data entry form together with related field value type information,
f) relating the type of each extracted field value to a corresponding CCG-data attribute name,
g) constructing a CCG-data attribute character string comprising a character string representing said attribute name and another character string representing said field value,
h) inserting said CCG-data attribute character string into said web page between said start and end character strings.
i) writing said web page on computer-readable media.

7. A computer implemented method of building a database which comprises sets of associated property values wherein each set includes at least two property values of different types, the property values being any of classification values, contact values, geographic location values, hereinafter collectively referred to as CCG-data, the method comprising the steps of:
a) retrieving successive web pages from a computer network, each web page being identified by a URL,
b) searching each web page for a CCG phrase that includes a plurality of different types of CCG-data attributes,
c) extracting a plurality of said attributes from said phrase,
d) from each extracted attribute, deriving an attribute name and a related attribute value,
e) determining the type of said extracted attribute and said attribute value by reference to said attribute name,
f) relating said type of attribute value so determined to a corresponding type of database property value,
g) relating the URL of said web page to an other type of database property value,
h) writing said derived attribute value to the database property value of said determined corresponding type in a set of associated property values, and
i) writing the URL of said web page to a database property value of said other type in said set of associated property values.

8. A computer implemented method of building a database which comprises sets of associated property values wherein each set includes at least two property values of different types, the property values being any of classification values, contact values, geographic location values, hereinafter collectively referred to as CCG-data, the method comprising the steps of:
a) retrieving successive web pages from a computer network, each web page being identified by a URL,
b) searching each web page for a CCG phrase that includes at least one type of CCG-data attribute,
c) extracting at least one said attribute from said phrase,
d) from each extracted attribute, deriving an attribute name and a related attribute value,

e) determining the type of said extracted attribute and said attribute value by reference to said attribute name,
f) relating said type of attribute value so determined to a corresponding type of database property value,
g) relating the URL of said web page to an other type of database property value,
h) writing said derived attribute value to the database property value of said determined corresponding type in a set of associated property values, and
i) writing the URL of said web page to a database property value of said other type in said set of associated property values.

9. A computer implemented method of building a database which comprises sets of associated property values wherein each set includes at least two property values of different types, the property values being any of classification values, contact values, geographic location values, hereinafter collectively referred to as CCG-data, the method comprising the steps of:
a) retrieving successive web pages from a computer network,
b) searching each web page for a CCG phrase that includes a plurality of different types of CCG-data attributes.
c) extracting a plurality of said attributes from said phrase,
d) from each extracted attribute, deriving an attribute name and a related attribute value,
e) determining the type of said extracted attribute and said attribute value by reference to said attribute name,
f) relating said type of attribute value so determined to a corresponding type of database property value, and
g) writing said derived attribute value to the database property value of said determined corresponding type in a set of associated property values.

10. A computer implemented method of finding references to web pages posted on computer network the method using a database comprising sets of associated property values, the property values being any of classification values, contact values, geographic location values, hereinafter collectively referred to as CCG-data, and URL references, the method comprising the steps of:
a) receiving a query phrase including query relational expressions from a computer network,
b) parsing said query phrase and extracting each of said query relational expressions included therein,
c) from each extracted query relational expression, deriving a query field name,
d) determining the type of said query relational expression by reference to its derived query field name,
e) relating said type of query relational expression so determined to one of the following query relational expression types: CCG-data type, other type,
f) provided said query relational expression is a CCG-data type, deriving a query relational operator and query value related to its query field name from said query relational expression,
g) determining the type of said query value by reference to said query field name,
h) relating said type of query value so determined to a corresponding type of database property value,

i) locating database property values of said determined corresponding type which return a true value when tested against said query value using said query relational operator,

j) extracting from said database a list of the URL references associated with the so located database property values.

11. A computer implemented method of finding sets of associated database property values the method using a database comprising sets of associated property values wherein each set includes at least two property values of different types, the property values being any of classification values, contact values, geographic values, hereinafter collectively referred to as CCG-data, the method comprising the steps of:

a) receiving a query phrase including query relational expressions from a computer network,

b) parsing said query phrase and extracting each of said query relational expressions included therein,

c) from each extracted query relational expression, deriving a query field name,

d) determining the type of said query relational expression by reference to its derived query field name,

e) relating said type of query relational expression so determined to one of the following query relational expression types: CCG-data type, other type,

f) provided said query relational expression is a CCG-data type, deriving a query relational operator and query value related to its query field name from said query relational expression,

g) determining the type of said query value by reference to said query field name,

h) relating said type of query value so determined to a corresponding type of database property value,

i) locating database property values of said determined corresponding type which return a true value when tested against said query value using said query relational operator,

j) extracting from said database sets of associated database property values associated with the so located database property values.

12. A method of displaying a web page comprising at least one HTML encoded CCG phrase, the method comprising the steps of:

a) retrieving a web page from a computer network,

b) parsing said retrieved web page to locate an HTML code indicative of the start of a CCG phrase,

c) parsing said located CCG phrase and extracting successive CCG attributes contained therein until an HTML code indicative of the end of said CCG phrase is found,

d) from each extracted attribute, deriving an attribute name,

e) determining the type of said extracted attribute by reference to its derived attribute name,

f) relating said type of attribute so determined to one of the following attribute types: database control, display control, CCG-data,

g) provided said extracted attribute is not a database control type, deriving an attribute value related to its attribute name from said extracted attribute,

h) determining the type of said attribute value by reference to said attribute name,

i) relating said type of attribute value so determined to a corresponding type of parameter of a display-device-control-program,

j) writing said attribute value to said parameter, and

k) where said type of attribute is a CCG-data type, causing said display-device-control-program to effect display of said attribute value on a display device, formatted and positioned according said display-device-control-program parameters whereby successive values of CCG-data of the CCG phrase are displayed.

ABSTRACT

A system for automatically creating databases containing industry, service, product and subject classification data, contact data, geographic location data (CCG-data) and links to web pages from HTML, XML or SGML encoded web pages posted on computer networks such as the Internet or Intranets. The web pages containing HTML, XML or SGML encoded CCG-data, database update controls and web browser display controls are created and modified by using simple text editors, HTML, XML or SGML editors or purpose built editors. The CCG databases may be searched for references (URLs) to web pages by use of enquiries which reference one or more of the items of the CCG-data. Alternatively, enquiries referencing the CCG-data in the databases may supply contact data without web page references. Data duplication and coordination is reduced by including in the web page CCG-data display controls which are used by web browsers to format for display the same data that is used to automatically update the databases.

Electronic Acknowledgement Receipt

EFS ID:	4809513
Application Number:	11269916
International Application Number:	
Confirmation Number:	5707
Title of Invention:	Automated media delivery system
First Named Inventor/Applicant Name:	Sean Barger
Customer Number:	22862
Filer:	Michael Glenn/Christine Ortt
Filer Authorized By:	Michael Glenn
Attorney Docket Number:	EQUI0001CIP-C
Receipt Date:	17-FEB-2009
Filing Date:	07-NOV-2005
Time Stamp:	19:30:46
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1		IDSEQUI0001CIP-C.pdf	245188 5a36601a0d26e823e5e9aeb6f5387b217ac08ae1	yes	6

	Multipart Description/PDF files in .zip description		
	Document Description	**Start**	**End**
	Miscellaneous Incoming Letter	1	1
	Information Disclosure Statement Letter	2	3
	Information Disclosure Statement (IDS) Filed (SB/08)	4	6

Warnings:					
Information:					
2	Foreign Reference	35-EP0747842.pdf	1664575 2af9850cc271a2199bf6d59e70568e93d507f6c1	no	27
Warnings:					
Information:					
3	Foreign Reference	36-EP0782085.pdf	791335 2b7c787340dafd993b77b5b55fa1e4343e211d36	no	18
Warnings:					
Information:					
4	Foreign Reference	37-EP0818907.pdf	542895 b690be55020c3094e256e0988aa06af4a7fe681d	no	8
Warnings:					
Information:					
5	Foreign Reference	38-EP0843276.pdf	1854757 eb56a7b43f7c18e6a7d7c9a1e2b128bac6789fe3	no	42
Warnings:					
Information:					
6	Foreign Reference	39-EP0876034.pdf	1026073 26f5619a632d73360d25feb2506f03da590b7130	no	16
Warnings:					
Information:					
7	Foreign Reference	40-EP0883068.pdf	1172032 fa47e50bf1595b8f41ebc3335f55a7d26cb432ef	no	22
Warnings:					
Information:					
8	Foreign Reference	41-EP0886409.pdf	1365504 dbb211b2c33c03334e6171f7d45569c580bcc3b4	no	28
Warnings:					

Information:					
9	Foreign Reference	42-EP0895171.pdf	2463238 d4e5c48419de6f5f3edab9a42a57b6686ef1603e	no	39
Warnings:					
Information:					
10	Foreign Reference	43-EPO926607.pdf	3783168 82f99a0149416291f9775b3e6a903a8202ac47b0	no	74
Warnings:					
Information:					
11	Foreign Reference	44-EPO949571.pdf	2246210 e090ac4df027a18cb51a141b53a1f824a9b3788b	no	40
Warnings:					
Information:					
12	Foreign Reference	45-WO98-40842.pdf	3996847 f631e807a428f3eb336305478b1ce88ffbff0020	no	118
Warnings:					
Information:					
13	Foreign Reference	46-WO98-43177.PDF	1839850 054919e44072c80d458b1a87557159880acbaef1	no	46
Warnings:					
Information:					
14	Foreign Reference	47-WO97-49252.pdf	1511607 7bfa5f299ba517e99c8f27e90c68e0efe07676b3	no	50
Warnings:					
Information:					
15	Foreign Reference	48-AUA53031-98.pdf	1705632 410c5e3f5665f1ac14f8282911acfdba41cf54a1	no	32
Warnings:					
Information:					
16	NPL Documents	A-Sakaguchi-ABrowsingTool.PDF	494426 1d96d1dc1240a3508d5539ad0bc687b614dae8fd	no	9
Warnings:					
Information:					
17	NPL Documents	B-Zaiane-MiningMultimediaData.pdf	1370264 564270bf9ee4122477794da53486be356ccdad6d	no	19
Warnings:					

Information:					
18	NPL Documents	C-Bulterman-ModelsMediaAndMotion.PDF	690629 0345df97b463fa006f0df427272df739d9fad73d	no	12
Warnings:					
Information:					
19	NPL Documents	D-Mohler-MigratingCourseMaterials.PDF	77628 7f940080689bd2c8d05a89945320fcabae8e32e6	no	2
Warnings:					
Information:					
20	NPL Documents	E-Dobson-AnimatingYourWebPages.PDF	293322 759a70d5547c34da189436bcd3acf15a66c914a7	no	5
Warnings:					
Information:					
21	NPL Documents	F-Berinstein-TheBigPicture.PDf	526864 055c120228c026c7ce046cec7039c80a2b8b6968	no	11
Warnings:					
Information:					
22	NPL Documents	G-McNeil-ResearchInterests.PDF	100387 a9668ca00802930bbdd9f94836d40b2e61c13419	no	3
Warnings:					
Information:					
23	NPL Documents	H-GeoSciences-TableOfContents.PDF	45795 c41562a3bd6be90a4d9142499ca7dbeb799677e2	no	2
Warnings:					
Information:					
			Total Files Size (in bytes):	29808226	

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111

If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371

If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office

If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

ELECTRONIC TRANSMITTAL COVER SHEET

Application Serial No. 11/269,916 Attorney Docket No. EQUI0001CIP-C

I hereby certify that this correspondence is being ELECTRONICALLY TRANSMITTED to the United States Patent and Trademark Office

From: GLENN PATENT GROUP
Customer No.: 22,862
Tel: (650) 474-8400
Fax: (650) 474-8401

on February 17, 2009 .
Date

Signature

Christine Ortt

Typed or printed name of person signing Certificate

Note: Each paper must have its own certificate of transmission, or this certificate must identify each submitted paper.

Attached to this cover sheet please find the following documents:

- Information Disclosure Statement (2 pages);
- 1449 (3 pages); and
- Cited References

This collection of information is required by 37 CFR 1.8. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 1.8 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

First Named Inventor	:	Sean Barger
Serial No.	:	11/269,916
Filed	:	November 7, 2005
Art Unit	:	4112
Confirmation Number	:	5707
Examiner	:	Christopher D. Bryant
Title	:	Automated Media Delivery System
Attorney Docket No.	:	EQUI0001CIP-C

February 17, 2009

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

INFORMATION DISCLOSURE STATEMENT

Examiner:

This Information Disclosure Statement is submitted:

(X) under 37 CFR 1.97(b), or
(Within three months of filing national application; or date of entry of international application; or before mailing date of first office action on the merits; whichever occurs last)

( ) under 37 CFR 1.97(c) together with either a:
- ( ) Certification under 37 CFR 1.97(e), or
- ( ) a $180.00 fee under 37 CFR 1.17(p), or

(After the CFR 1.97(b) time period, but before final action or notice of allowance, whichever occurs first)

( ) under 37 CFR 1.97(d) together with a:
- ( ) Certification under 37 CFR 1.97(e), and
- ( ) a $180.00 fee under 37 CFR 1.17(p).

(Filed after final action or notice of allowance, whichever occurs first, but before payment of the issue fee)

(X) The Commissioner is authorized to charge any additional fees or credit any overpayment to Deposit Account No. 07-1445 (Order No. EQUI0001CIP-C).

(X) Applicant(s) submit herewith PTO Form 1449 (Modified) -- Information Disclosure Citation together with copies of patents, publications or other information of which applicant(s) are aware, which applicant(s) believe(s) may be material to the examination of this application and for which there may be a duty to disclose in accordance with 37 CFR 1.25.

( ) A concise explanation of the relevance of foreign language patents, foreign language publications and other foreign language information listed on PTO Form 1449 (Modified), as presently understood by the individual(s) designated in 37 CFR 156(c) most knowledgeable about the content is given on the attached sheet, or where a foreign language patent is cited in a search report or other action by a foreign patent office in a counterpart foreign application, an English language version of the search report or action which indicates the degree of relevance found by the foreign office is listed on form PTO Form 1449 (Modified) and is enclosed herewith.

It is requested that the information disclosed herein be made of record in this application.

Respectfully Submitted,

Michael A. Glenn
Reg. No. 30,176

Customer No. 22862

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
11/269,916	11/07/2005	Sean Barger	EQUI0001CIP-C	5707

22862 7590 03/30/2009

GLENN PATENT GROUP
3475 EDISON WAY, SUITE L
MENLO PARK, CA 94025

EXAMINER	
BRYANT, CHRISTOPHER D	
ART UNIT	PAPER NUMBER
4112	
MAIL DATE	DELIVERY MODE
03/30/2009	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	11/269,916	BARGER ET AL.	
	Examiner	Art Unit	
	CHRISTOPHER BRYANT	4112	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☐ Responsive to communication(s) filed on *07 November 2005*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-16* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-16* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *2/17/2009, 10/11/2006*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

Claim Rejections - 35 USC § 112

1. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

2. Regarding claims 2, and 10 the phrase "or the like" renders the claim(s) indefinite because the claim(s) include(s) elements not actually disclosed (those encompassed by "or the like"), thereby rendering the scope of the claim(s) unascertainable. See MPEP § 2173.05(d).

Claim Rejections - 35 USC § 103

3. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

Claims 1, 3-9, and 11-16 are rejected under 35 U.S.C. 103(a) as being unpatentable over Jammes et al. (United States Patent 6,484,149), hereinafter referenced as Jammes, in view of Filepp et al. (United States Patent 5,442,771), hereinafter referenced as Filepp.

Regarding claim 1, Jammes discloses a method of parsing tags to determine if any of the content generation procedure to execute and corresponding input to a procedure, “a standard coding convention and set of codes for attaching presentation and linking attributes to informational content within documents. During a document authoring stage, the HTML codes (referred

to as "tags") are embedded within the informational content of the document. When the Web document (or "HTML document") is subsequently transmitted by a Web server to a Web browser, the codes are interpreted by the browser and used to parse and display the document. In addition to specifying how the Web browser is to display the document, HTML tags can be used create hyperlinks to other Web documents" (column 7, lines 15-26). Jammes also discloses a method of separating tags from content generators, generating a unique image lookup, checking the media cache using an intermediate image lookup key, (Figure 17). Jammes also discloses determining valid content, and also converting negative content to valid type, customizing content for specified browser or profile system, and attaching any specified cache-control directives to a response and delivering the content, "a variety of methods exist for embedding a customize reference in a template file. According to one embodiment of the present invention, a script of commands is embedded in a template file specifying at least one query to perform on the traffic database, comparison of the query result against preferred customization rules, and, if customization is warranted, a translation to perform on the query result to convert the result to HTML format" (column 43, lines 23-30).

However, the examiner maintains that it was well known in the art to have dynamic modification to perform on content, user profile characters and cache control and generating a unique key for content, check a cache and performing an action based on the result, as taught by Filepp.

In a similar field of endeavor Filepp discloses having dynamic modification to performs on media, user profile characteristics and cache control, “standard for encoding graphics data, or text code, such as ASCII, which are displayed on monitor 412 of the user's personal computer 405 as pictorial codes. Codes for other presentation media, such as audio, can be specified by using the appropriate type code in the presentation data segments” (column 21, lines 5-15), generating a unique final lookup key for media, checking the media cache with the lookup key, (Figure 2, Item 302), passing control to cache control, and delivering media, (Figure 2, Item 302 passing to 301).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to modify Jammes by adding unique lookups and checking caches as taught by Filepp for delivering media (content).

Regarding claim 3, Jammes discloses a method of adding content to a system, creating a procedure containing instructions for processing the content, and creating a document referring to the processed content by pointing to the new procedure, “in an additional step 2026, the HTML page engine scans the template file and determines whether there is a customize reference in the template file. If not, then, in a next step 2027, the HTML page engine determines whether any cross sale related product information is associated with the consumer and, if so, generates HTML hyperlink tags to Web pages describing the cross sale products. After adding these cross sale product hyperlink tags to the requested HTML file or template file, the HTML page engine deletes any association between cross sale product information and the consumer. In a

further step 2028, the Web server 106 transmits the template file (a compliant HTML file) to the Web browser 102" (column 52, lines 11-20, corresponding to Fig.20B).

Regarding claim 4, Jammes discloses a system that assists with choosing parameters and with content generation, "an HTML authoring system 110 is used to create and modify HTML template files 108" (column 8, lines 10-17, corresponding to Figure 1).

Regarding claim 5, Jammes discloses an apparatus which has a server for receiving request and delivering content (Figure 1), a content generation procedure containing instructions (Figure 17). Jammes also discloses a link parser which determines any of content generation procedure to execute and any corresponding input parameters to be used for generating a primary media cached, "one of ordinary skill in the art will understand that Web servers possess parsing routines to extract data parameters from HTTP Post messages in name/value pair format and that applications, such as the ISAPI query application, may be identified by a portion of a URL" (column 18, lines 0-5), however, Jammes does not disclose processing based on user profile information as well as containing cache control headers. Jammes also fails to disclose having a primary and secondary lookup key.

However, the examiner maintains that it was well known in the art to have additional functionality using cache control headers with primary and secondary lookups, as taught by Filepp.

In a similar field of endeavor Filepp teaches processing to be performed with user profile information modifying a resulting image, dynamic content processing, and cache control headers with the resulting image, "in operation, as data is supplied to the store; for example, a reception system store during a user interactive session, data is retained at the store based on the available cache space within the store; i.e., reception system available RAM and designated disk file. Particularly, data items designated by a data identification number are placed on a list of recently called data items, the most recently called items being at the top of the list. As new data is called, it pushes previously called data down on the list, with the result that a data item pushed below the list capacity forfeits its presence on the list if not recalled before being pushed off. If data is recalled during a session, it once more is promoted to the top of the data list. At the end of a session, data items at the cache are written to a stage least-recently-used list, the stage retaining data items between sessions in the same fashion the cache retains data during a session. The result is, over a series of sessions, the stage automatically configures itself; i.e., self-configures, with the data most often called" (column 3, lines 45-60). Filepp also discloses having primary and secondary keys for parsing within a cache, "cacheable objects can be retained during the current user session, but cannot be retained between sessions. These objects usually have a moderate update frequency. Object storage facility 439 retains objects in the cache according to the LRU storage retention algorithm. Object storage facility 439 uses the LRU algorithm to ensure that objects that are

least frequently used forfeit their storage to objects that are more frequently used" (column 15, lines 60-65).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to modify Jammes by adding a link tag parser with additional functionality, as taught by Filepp for delivering media (content).

Regarding claim 6, Jammes discloses a method of link tags being modified and updated, (Figure 17).

Regarding claim 7, Jammes discloses an apparatus with a server infrastructure providing requests, (Figure 2).

Regarding claim 8, Jammes discloses an apparatus which able to operate on-site or off-site within server infrastructure, (Figure 2).

Regarding claim 9, Jammes discloses an apparatus which primary cache memory' is composed of original content acquired from a repository, (Figure 2).

Regarding claim 11, Jammes discloses delivering content by disclosing tags, being accessible by a browser, links request for an image system, and process links through interpretation of tags and executing an image generation procedure with the procedure located within the tags, "during a document authoring stage, the HTML codes (referred to as "tags") are embedded within the informational content of the document. When the Web document (or "HTML document") is subsequently transmitted by a Web server to a Web browser, the codes are interpreted by the browser and used to parse and display the document. In addition to specifying how the Web browser is to display the document, HTML tags can be used create hyperlinks to other Web documents"

(column 7, lines 15-25). The art further teaches delivering the content (Figure 16).

Regarding claim 12, all limitations were addressed above in claim 11.

Regarding claim 13, all limitations were addressed above in claim 5.

Regarding claim 14, Jammes provides an apparatus which provides changed media to a user (Fig.17, item 1722 consumer).

Regarding claim 15, Jammes provides an apparatus that places links within tags that are used to execute commands with regards to the property of link, “during a document authoring stage, the HTML codes (referred to as "tags") are embedded within the informational content of the document. When the Web document (or "HTML document") is subsequently transmitted by a Web server to a Web browser, the codes are interpreted by the browser and used to parse and display the document. In addition to specifying how the Web browser is to display the document, HTML tags can be used create hyperlinks to other Web documents” (column 7, lines 15-25). The art further teaches delivering the content (Figure 16), and delivering media to the user, (Fig.17, item 1722 consumer).

Regarding claim 16, all limitations were addressed above claim 14.

Claims 2 and 10 are rejected under 35 U.S.C. 103(a) as being unpatentable over Jammes in view of Filepp, in further view of Blumberg et al. (United States Patent 6,449,639), hereinafter referenced as Blumberg.

Regarding claim 2, Jammes and Filepp claim everything claimed above, however they fail to disclose processing with directives for zoom, pan, and slice.

However, the examiner maintains that it was well known in the art to include such directives for dynamic processing, as taught by Blumberg.

In a similar field of endeavor Blumberg teaches a method of a server which processes an HTML page. During the processing the page goes through a series of calculations, and modifications (Figure 6, Box 570 and 540).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to modify Jammes and Filepp by adding directives for dynamic processing, as taught by Blumberg for delivering media (content).

Regarding claim 10, all limitations were addressed above in claim 2.

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to CHRISTOPHER BRYANT whose telephone number is (571)270-7260. The examiner can normally be reached on Monday-Friday 8:00 A.M. to 5:00 P.M. EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Jeffrey Harold can be reached on (571)272-7519. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Jefferey F Harold/
Supervisory Patent Examiner, Art Unit 4112

Notice of References Cited	Application/Control No. 11/269,916	Applicant(s)/Patent Under Reexamination BARGER ET AL.	
	Examiner CHRISTOPHER BRYANT	Art Unit 4112	Page 1 of 1

U.S. PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Name	Classification
*	A	US-5,442,771	08-1995	Filepp et al.	709/219
*	B	US-6,484,149	11-2002	Jammes et al.	705/26
	C	US-			
	D	US-			
	E	US-			
	F	US-			
	G	US-			
	H	US-			
	I	US-			
	J	US-			
	K	US-			
	L	US-			
	M	US-			

FOREIGN PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Country	Name	Classification
	N					
	O					
	P					
	Q					
	R					
	S					
	T					

NON-PATENT DOCUMENTS

*		Include as applicable: Author, Title Date, Publisher, Edition or Volume, Pertinent Pages)
	U	
	V	
	W	
	X	

*A copy of this reference is not being furnished with this Office action. (See MPEP § 707.05(a).)
Dates in MM-YYYY format are publication dates. Classifications may be US or foreign.

OIPE
OCT 11 2006
PATENT & TRADEMARK OFFICE

Attorney Docket No. EQUI0001CIP-C

Form 1449 (Modified)	Atty. Docket No. EQUI0001CIP-C	Serial No.:11/269,916
Information Disclosure Statement By Applicant	Applicant: Barger et al.	
(Use Several Sheets if Necessary)	Filing Date:11/07/2005	Group: 2176

U.S. Patent Documents

Examiner Initial	No.	Patent No.	Date	Patentee	Class	Sub-class	Filing Date
/C.B./	A	5,758,110					
/C.B./	B	5,761,655					
/C.B./	C	5,819,261					
/C.B./	D	5,864,337					
/C.B./	E	5,870,552					
/C.B./	F	5,880,740					
/C.B./	G	5,890,170					
/C.B./	H	5,895,476					
/C.B./	I	5,937,160					
/C.B./	J	5,943,680					
/C.B./	K	5,956,737					
/C.B./	L	6,009,436					
/C.B./	M	6,456,305					
/C.B./	N	6,563,517					
/C.B./	O	6,591,280					
/C.B./	P	6,623,529					

Foreign Patent or Published Foreign Patent Application

Examiner Initial	No.	Document No.	Publication Date	Country or Patent Office	Class	Sub-class	Translation Yes	Translation No
/C.B./	Q	EP0926607						
/C.B./	R	EP0949571						
/C.B./	S	WO97/49252						
/C.B./	T	WO98/40842						
/C.B./	U	WO98/43177						

Other Documents

Examiner Initial	No.	Author, Title, Date, Place (e.g. Journal) of Publication

Examiner: Initial citation considered. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

/Christopher Bryant/ 03/16/2009

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /C.B./

Search Notes	Application/Control No.	Applicant(s)/Patent Under Reexamination
	11269916	BARGER ET AL.
	Examiner	Art Unit
	CHRISTOPHER BRYANT	4112

SEARCHED			
Class	Subclass	Date	Examiner
715	255	3/19/2009	cb

SEARCH NOTES		
Search Notes	Date	Examiner
See File EastSearchHistory	3/19/2009	CB

INTERFERENCE SEARCH			
Class	Subclass	Date	Examiner

/CHRISTOPHER BRYANT/ Examiner.Art Unit 4112	

Form 1449 (Modified)	Atty. Docket No.	Serial No.:
	EQUI0001CIP-C	11/269,916
Information Disclosure Statement By Applicant	**Applicant:** Sean Barger, et al.	
	Filing Date:	**Group:** 4112
(Use Several Sheets if Necessary)	November 7, 2005	**Confirmation No:** 5707

U.S. Patent Documents

Examiner Initial	No.	Patent No.	Issue Date	Patentee	Class	Sub-class	Filing Date
/C.B./	1	5,088,052	2/1/1992	Spielman, et al.			
/C.B./	2	5,355,472	10/1/1994	Lewis			
/C.B./	3	5,530,852	6/1/1996	Meske, Jr., et al.			
/C.B./	4	5,701,451	12/1/1997	Rogers, et al.			
/C.B./	5	5,708,845	1/1/1998	Wistendahl, et al.			
/C.B./	6	5,710,918	1/1/1998	Lagarde, et al.			
/C.B./	7	5,737,619	4/1/1998	Judson			
/C.B./	8	5,745,908	4/1/1998	Anderson, et al.			
/C.B./	9	5,758,110	5/26/1998	Boss, et al.			
/C.B./	10	5,761,655	6/2/1998	Hoffman, Michael T.			
/C.B./	11	5,793,964	8/1/1998	Rogers, et al.			
/C.B./	12	5,819,261	10/6/1998	Takahashi, et al.			
/C.B./	13	5,822,436	10/1/1998	Rhoads			
/C.B./	14	5,845,084	12/1/1998	Cordell, et al.			
/C.B./	15	5,845,299	12/1/1998	Arora, et al.			
/C.B./	16	5,860,068	1/1/1999	Cook			
/C.B./	17	5,860,073	1/1/1999	Ferrel, et al.			
/C.B./	18	5,861,881	1/1/1999	Freeman, et al.			
/C.B./	19	5,862,325	1/1/1999	Reed, et al.			
/C.B./	20	5,864,337	1/26/1999	Marvin, John			
/C.B./	21	5,870,552	2/1/1999	Dozier, et al.			
/C.B./	22	5,880,740	3/1/1999	Halliday, et al.			
/C.B./	23	5,890,170	3/1/1999	Sidana			
/C.B./	24	5,895,476	4/1/1999	Orr, et al.			
/C.B./	25	5,895,477	4/20/1999	Orr, et al.			
/C.B./	26	5,903,892	5/11/1999	Hoffert, et al.			
/C.B./	27	5,937,160	8/1/1999	Davis, et al.			
/C.B./	28	5,943,680	8/24/1999	Ohga, et al.			
/C.B./	29	5,956,737	9/21/1999	King, et al.			
/C.B./	30	6,009,436	12/1/1999	Motoyama, et al.			
/C.B./	31	6,456,305	9/1/2002	Qureshi, et al.			
/C.B./	32	6,563,517	5/1/2003	Bhagwat, et al.			
/C.B./	33	6,591,280	7/1/2003	Orr			
/C.B./	34	6,623,529	9/1/2003	Lakritz			

Published U.S. Patent Application

Examiner Initial	No.	Document No.	Publication Date	Assignee	Class	Sub-class	Translation Yes	Translation No

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /C.B./

Foreign Patent or Published Foreign Patent Application

Examiner Initial	No.	Document No.	Publication Date	Assignee	Class	Sub-class	Translation Yes	Translation No
/C.B./	35	EP 0747842	12/11/1996	International Business Machines Corp.				
/C.B./	36	EP 0782085	7/2/1997	International Business Machines Corp.				
/C.B./	37	EP 0818907	1/14/1998	AT&T Corp				
/C.B./	38	EP 0843276	5/20/1998	Canon Information Systems Inc.				
/C.B./	39	EP 0876034	11/4/1998	International Business Machines Corp.				
/C.B./	40	EP 0883068	12/9/1998	Home Information Services, Inc.				
/C.B./	41	EP 0886409	12/23/1998	Digital Vision Laboratories Corporation				
/C.B./	42	EP 0895171	2/3/1999	Neoforma, Inc.				
/C.B./	43	EP 0926607	6/30/1999	Ricoh KK				
/C.B./	44	EP 0949571	10/13/1999	Xerox Corp				
/C.B./	45	WO 98/40842	9/17/1998	Computer Information and Sciences, Inc.				
/C.B./	46	WO 98/43177	10/1/1998	Intel Corp				
/C.B./	47	WO 97/49252	12/24/1997	Senthilkumar, et al.				
/C.B./	48	AU-A-53031/98	8/27/1998	Dudley, John Mills				

Other Documents

Examiner Initial	No.	Author, Title, Date, Place (e.g. Journal) of Publication
/C.B./	A	Sakaguchi, et al.; " A browsing tool for multi-lingual documents for users without multi-lingual fonts"; 1996; ACM International Conference On Digital Libraries, pp. 63-71
/C.B./	B	Zaiane, et al.; "Mining multimedia data"; Nov. 1998; ACM Conference of the Center for Advanced Studies on Collaborative research, pp. 1-18
/C.B./	C	BULTERMAN, DICK.C.A.; Models, Media and Motion: Using the Web to Support Multimedia Documents; Proceedings of 1997 Int'l Conf on Multimedia Modeling; p. 17-20; November 1997; SINGAPORE
/C.B./	D	MOHLER, J.L.; Migrating Course Materials to the World Wide Web: A Case Study of the Department of Technical Graphics at Purdue University; Computer Networks and ISDN Systems; Vol. 30, Issues 20-21, p,1981-1990; November 12, 1988
/C.B./	E	DOBSON, R.; Animating Your Web Pages with Direct Animation; Web Techniques; vol.3, no. 6, p. 49-52; June 1998
/C.B./	F	BERINSTEIN, Paula; "The Big Picture; Text and Graphics on UMI's ProQuest Direct: The Best (Yet) of Both Words"; March 1997; retrieved on 3/23/04 from website: http://www.infotoday.com/online/MarOL97/picture3.html

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /C.B./

/C.B./	G	McNeil, Sara; Research Interests; retrieved on March 18, 2004 from website: http//www.coe.uh.edu/`smcneil/research.htm
/C.B./	H	Tables of Contents service for Computers & Geosciences; Copyright 1997; Computers and GeoSciences, Volume 23, Issue 5, retrieved on 3/18/04 from website: http://library.iem.ac.ru/comp&geo/00983004/sz977014.html

Examiner's Signature /Christopher Bryant/ Date 03/16/2009

Examiner: Initial citation considered. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /C.B./

EAST Search History

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
S1	416	barger.in.	US-PGPUB; USPAT	OR	ON	2009/03/17 09:42
S2	1	S1 and samaniego.in.	US-PGPUB; USPAT	OR	ON	2009/03/17 09:42
S3	77	S1 and media	US-PGPUB; USPAT	OR	ON	2009/03/17 09:43
S4	1934	715/255.ccls.	US-PGPUB; USPAT	OR	ON	2009/03/17 09:45
S5	1470	S4 and media	US-PGPUB; USPAT	OR	ON	2009/03/17 09:46
S7	322	S5 and delivery	US-PGPUB; USPAT	OR	ON	2009/03/17 09:47
S8	195	S7 and graphics	US-PGPUB; USPAT	OR	ON	2009/03/17 09:47
S9	64	S8 and cache	US-PGPUB; USPAT	OR	ON	2009/03/17 11:38
S10	17412	content with deliver	US-PGPUB; USPAT	OR	ON	2009/03/17 11:48
S11	135	S10 with tag	US-PGPUB; USPAT	OR	ON	2009/03/17 11:49
S12	3	S11 with cache	US-PGPUB; USPAT	OR	ON	2009/03/17 11:49
S13	14036	media with deliver	US-PGPUB; USPAT	OR	ON	2009/03/17 11:51
S15	53	S13 with cache	US-PGPUB; USPAT	OR	ON	2009/03/17 11:52

S17	26427	tags and html	US-PGPUB; USPAT	OR	ON	2009/03/17 11:59
S18	5540	tags near html	US-PGPUB; USPAT	OR	ON	2009/03/17 12:00
S19	525	S18 same parse	US-PGPUB; USPAT	OR	ON	2009/03/17 12:00
S20	36	S19 same media	US-PGPUB; USPAT	OR	ON	2009/03/17 12:00

3/19/2009 7:51:44 PM
C:\Documents and Settings\cbryant2\My Documents\EAST\Workspaces\11072005automedideliv.wsp

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

BIB DATA SHEET

CONFIRMATION NO. 5707

SERIAL NUMBER	FILING or 371(c) DATE	CLASS	GROUP ART UNIT	ATTORNEY DOCKET NO.
11/269,916	11/07/2005 **RULE** 1.47	707	4112	EQUI0001CIP-C

APPLICANTS

Sean Barger, M. Valley, CA;
Christopher Samaniego, San Francisco, CA;
Nelson H. Rocky Offner, Kensington, CA;
Adrian D. Thewlis, Sausalito, CA;
David R. Boyd, San Francisco, CA;
David C. Salmon, San Rafael, CA;
Joshua N. Devan, Kentfield, CA;

**** CONTINUING DATA ***************************

This application is a CIP of 09/929,904 08/14/2001 PAT 6,964,009
which is a CON of 09/425,326 10/21/1999 PAT 6,792,575

**** FOREIGN APPLICATIONS ***************************

**** IF REQUIRED, FOREIGN FILING LICENSE GRANTED **** ** SMALL ENTITY **
12/08/2005

Foreign Priority claimed	☐ Yes ☑ No		STATE OR COUNTRY	SHEETS DRAWINGS	TOTAL CLAIMS	INDEPENDENT CLAIMS
35 USC 119(a-d) conditions met	☐ Yes ☑ No	☐ Met after Allowance				
Verified and Acknowledged	/CHRISTOPHER D BRYANT/ Examiner's Signature	CB Initials	CA	23	16	5

ADDRESS

GLENN PATENT GROUP
3475 EDISON WAY, SUITE L
MENLO PARK, CA 94025
UNITED STATES

TITLE

Automated media delivery system

FILING FEE RECEIVED		
765	FEES: Authority has been given in Paper No._____ to charge/credit DEPOSIT ACCOUNT No._____ for following:	☐ All Fees ☐ 1.16 Fees (Filing) ☐ 1.17 Fees (Processing Ext. of time) ☐ 1.18 Fees (Issue) ☐ Other _____ ☐ Credit

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
11/269,916	11/07/2005	Sean Barger	EQUI0001CIP-C	5707

22862 7590 10/15/2009

GLENN PATENT GROUP
3475 EDISON WAY, SUITE L
MENLO PARK, CA 94025

EXAMINER	
BRYANT, CHRISTOPHER D	

ART UNIT	PAPER NUMBER
2178	

NOTIFICATION DATE	DELIVERY MODE
10/15/2009	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

eptomatters@glenn-law.com

Notice of Abandonment	Application No.	Applicant(s)	
	11/269,916	BARGER ET AL.	
	Examiner	Art Unit	
	CHRISTOPHER BRYANT	2178	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address--

This application is abandoned in view of:

1. ☒ Applicant's failure to timely file a proper reply to the Office letter mailed on *30 March 2009*.
   (a) ☐ A reply was received on _____ (with a Certificate of Mailing or Transmission dated _____), which is after the expiration of the period for reply (including a total extension of time of _____ month(s)) which expired on _____.
   (b) ☐ A proposed reply was received on _____, but it does not constitute a proper reply under 37 CFR 1.113 (a) to the final rejection.
   (A proper reply under 37 CFR 1.113 to a final rejection consists only of: (1) a timely filed amendment which places the application in condition for allowance; (2) a timely filed Notice of Appeal (with appeal fee); or (3) a timely filed Request for Continued Examination (RCE) in compliance with 37 CFR 1.114).
   (c) ☐ A reply was received on _____ but it does not constitute a proper reply, or a bona fide attempt at a proper reply, to the non-final rejection. See 37 CFR 1.85(a) and 1.111. (See explanation in box 7 below).
   (d) ☒ No reply has been received.

2. ☐ Applicant's failure to timely pay the required issue fee and publication fee, if applicable, within the statutory period of three months from the mailing date of the Notice of Allowance (PTOL-85).
   (a) ☐ The issue fee and publication fee, if applicable, was received on _____ (with a Certificate of Mailing or Transmission dated _____), which is after the expiration of the statutory period for payment of the issue fee (and publication fee) set in the Notice of Allowance (PTOL-85).
   (b) ☐ The submitted fee of $_____ is insufficient. A balance of $_____ is due.
   The issue fee required by 37 CFR 1.18 is $_____. The publication fee, if required by 37 CFR 1.18(d), is $_____.
   (c) ☐ The issue fee and publication fee, if applicable, has not been received.

3. ☐ Applicant's failure to timely file corrected drawings as required by, and within the three-month period set in, the Notice of Allowability (PTO-37).
   (a) ☐ Proposed corrected drawings were received on _____ (with a Certificate of Mailing or Transmission dated _____), which is after the expiration of the period for reply.
   (b) ☐ No corrected drawings have been received.

4. ☐ The letter of express abandonment which is signed by the attorney or agent of record, the assignee of the entire interest, or all of the applicants.

5. ☐ The letter of express abandonment which is signed by an attorney or agent (acting in a representative capacity under 37 CFR 1.34(a)) upon the filing of a continuing application.

6. ☐ The decision by the Board of Patent Appeals and Interference rendered on _____ and because the period for seeking court review of the decision has expired and there are no allowed claims.

7. ☒ The reason(s) below:

Representation was contacted last week. The response was that they were unable to contact applicant. Representation has not since responded.

/Joshua D Campbell/
Primary Examiner, Art Unit 2178

Petitions to revive under 37 CFR 1.137(a) or (b), or requests to withdraw the holding of abandonment under 37 CFR 1.181, should be promptly filed to minimize any negative effects on patent term.