UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

Axis Communications AB, Canon Inc., and Canon U.S.A., Inc.,

Petitioner

v.

Avigilon Fortress Corporation,

Patent Owner

Cases: IPR2019-00235 & IPR2019-00236

U.S. Patent No. 7,868,912
Issue Date: January 11, 2011

Title: Video Surveillance System Employing Video Primitives

DECLARATION OF EMILY R. FLORIO

I, Emily R. Florio, state and declare as follows:

1. I have prepared this Declaration in connection with the Petitions of Axis Communications AB, Canon Inc., and Canon U.S.A., Inc. (collectively "Petitioner") for two *inter partes* reviews of U.S. Patent No. 7,868,912 ("the '912 patent"), Case Nos. IPR2019-00235 and IPR2019-00236, which I understand will be filed concurrently with this Declaration.

2. I am currently the Director of Research & Information Services at Finnegan, Henderson, Farabow, Garrett & Dunner LLP, 901 New York Avenue NW, Washington, DC 20001-4413.

3. I am over 18 years of age and am competent to make this Declaration. I make this Declaration based on my own personal knowledge, based on my knowledge of library science practices, as well as my knowledge of the practices at the Massachusetts Institute of Technology ("MIT") Libraries.

4. I earned a Master's of Library Science ("MLS") from Simmons College in 2006, and I have worked as a librarian for over a decade. I have been employed in the Research & Information Services (formerly Library) Department of Finnegan since 2013, and from 2005-2013, I worked in the Library Department of Fish & Richardson P.C.

5. I am currently the Vice-President Elect of the American Association of Law Libraries and the President of the Law Librarians' Society of Washington, DC, and a member of the International Legal Technology Association.

Attachments

6. Attached as Exhibit A (Exhibit 1003 to the Petition in IPR2019-00235) is a true and correct copy of "Visual Memory," May 1993, pp. 1-92, by Christopher James Kellogg ("*Kellogg*"), obtained from the MIT Libraries.

7. Attached as Exhibit B is a true and correct copy of the "Standard" record from the MIT Libraries' catalog system (known as the Barton Catalog) for its copy of *Kellogg.*

8. Attached as Exhibit C is a true and correct copy of the MARC record of the MIT Libraries for its copy of *Kellogg.*

9. Attached as Exhibit D (Exhibit 1005 to the Petition in IPR2019-00235) is a true and accurate copy of B. Flinchbaugh et al., "Autonomous Scene Monitoring System," Proceedings, 10th Annual Joint Government-Industry Security Technology Symposium & Exhibition, June 20-23, 1994, pp. 205-209 ("*Flinchbaugh*"), obtained from the British Library.

10. Attached as Exhibit E is a true and correct copy of the MARC record of the British Library for its copy of the Proceedings publication that includes *Flinchbaugh.*

11. Attached as Exhibit F are documents further showing the public availability of *Flinchbaugh.* Exhibit F-1 is a true and correct copy of U.S. Patent No. 7,023,469 (the '469 Patent), and the list of references filed by the applicant of the '469 Patent with its filing on April 15, 1999, obtained from the U.S. Patent and Trademark Office's PAIR files. Exhibit F-2 is a true and correct copy of R. Collins et al., "A System for Video Surveillance and Monitoring," Robotics Institute, Carnegie Mellon University, Pittsburgh, PA ("*Collins*") and information concerning this 1999 publication available at https://www.semanticscholar.org.

12. Attached as Exhibit G (Exhibit 1004 in each Petition in IPR2019-00235 and IPR2019-00236) is a true and correct copy of Brill et al., "Event Recognition and Reliability Improvements for the Autonomous Video Surveillance System," Proceedings of the Image Understanding Workshop, Monterey, CA, Nov. 20-23, 1998, Vol. 1, pp. 267-283 ("*Brill*"), obtained from the Duderstadt Center, formerly known as the University of Michigan Media Union (UMMU).

13. Attached as Exhibit H is a true and correct copy of the MARC record of the University of Virginia Library for its copy of *Brill.*

14. Attached as Exhibit I is a true and correct copy of the MARC record of the North Carolina State University library for its copy of *Brill.*

The MARC Cataloging System

15. The MAchine-Readable Cataloging ("MARC") system is used by libraries to catalog materials. The MARC system was developed in the 1960s to standardize bibliographic records so they could be read by computers and shared among libraries. By the mid-1970's, MARC had become the international standard for bibliographic data, and it is still used today.

16. Each field in a MARC record provides information about the cataloged item. MARC uses a simple three-digit numeric code (from 001-999) to identify each field in the record.

17. For example, field 245 lists the title of the work and field 260 lists publisher information. In addition, field 008 provides the date the item was cataloged. The first six characters of the field 008 are always in the "YYMMDD" format.

18. It is standard library practice that once an item is cataloged using the MARC system, it is shelved. This process may take a relatively nominal amount of time (i.e., a few days or weeks). During the time between the cataloging and shelving of an item, the public may still find the item by searching the catalog and requesting the item from the library.

Kellogg

19. As indicated in Exhibit A (Exhibit 1003 to the Petition in IPR2019-00235), *Kellogg* has an MIT Libraries date stamp of "JUL 09 1993" on page 1, indicating that the MIT Libraries received *Kellogg* on July 9, 1993. Further, as indicated in Exhibit B, the Standard record of the Barton Catalog confirms that *Kellogg* is shelved at the MIT Libraries and was published in 1993. In view of the above and the following, *Kellogg* was published and accessible to the public in 1993, years before October 1999.

20. As indicated in Exhibit C, *Kellogg* has a cataloging date of September 28, 1993 (shown as "930928" in field 008). This confirms that *Kellogg* was entered into the OCLC database, in which MIT does its cataloging, on September 28, 1993. This is also consistent with its noted year of publication in the MARC record (shown as "1993" in field 260). The OCLC database (also referred to as "WorldCat") is the largest online public access catalog (OPAC) in the world.

21. Soon after *Kellogg* received a cataloging date, a record of its existence would have appeared in and been keyword-searchable through the Barton Catalog of the MIT Libraries. The Barton Catalog is currently available online to any user of the World Wide Web. Before it was accessible by Web (i.e., at the time the *Kellogg* thesis was received by the MIT Libraries in July 1993), it would have been

accessible to anyone on the MIT campus *and* anyone who had access to the OCLC database.

22. During the time period from September 1993 through October 1999, the Barton Catalog allowed keyword searching for words in the thesis title, and *Kellogg* would have appeared in a relevant Barton Catalog search conducted on or shortly after September 28, 1993.

23. After being cataloged, a document such as *Kellogg* will undergo a process of being labeled and then shelved at the MIT Libraries. Based on my knowledge of MIT Libraries' current and prior practices, *Kellogg* would have been shelved in a relatively nominal amount of time (i.e., a few days or weeks). Thus, *Kellogg* was cataloged and shelved at the MIT Libraries at least before the end of 1993.

24. Once shelved, *Kellogg* can be borrowed by any member of the MIT community. Furthermore, a copy of *Kellogg* can be purchased from MIT by any member of the public. Indeed, the first page of *Kellogg* confirms that there were no restrictions placed on its publication, as it states that "[t]he author hereby grants to MIT permission *to reproduce and to distribute copies of this thesis document* in whole or in part, and to grant others the right to do so."

25. Further evidence of the public availability of *Kellogg* before October 1999 is provided in Exhibit D, which is a copy of *Flinchbaugh*. In its

"References" section, *Flinchbaugh* cites to *Kellogg* (reference [1] on p. 209). It also states that *Kellogg* is available as a Technical Report, CSL-93-05-20, from Texas Instruments. As addressed below, *Flinchbaugh* was published in the Proceedings from the 10th Annual Joint Government-Industry Security Technology Symposium and Exhibition. The Symposium was held June 20-23, 1994, and the Proceedings were published by at least 1997. Thus, *Kellogg* was at least available to members of the public in 1997, as shown by its citation in *Flinchbaugh.*

26. For the avoidance of any doubt, I note that on June 23, 2001, *Kellogg* was also cataloged in the MIT Archive Noncirculating Collection 1, Noncirculating Collection 3, and in microfiche form in the Barker Library, as indicated in the three entries for PST8 and in the second, third, and fourth instances of field 008 on page 1 of Exhibit C. However, none of this alters the fact that *Kellogg* was published and accessible to the public in 1993, as indicated above.

Flinchbaugh

27. As indicated in Exhibit D, *Flinchbaugh* (Exhibit 1005 to the Petition in IPR2019-00235) was published in the Proceedings of the 10th Annual Joint Government-Industry Security Technology Symposium and Exhibition. The Symposium was held in Williamsburg, VA during June 20-23, 1994, and the Proceedings was published by the American Defense Preparedness Association.

Ex. D at 1. In view of the above and the following, *Flinchbaugh* was published and accessible to the public before October 1999.

28. First, as noted above, the Symposium was held in Williamsburg, VA during June 20-23, 1994. Second, a copy of *Flinchbaugh* was received and cataloged by the British Library in February 1997. *See* Ex. E at 1. Exhibit E is the MARC record for the Proceedings, including *Flinchbaugh*, that was obtained from the British Library. As shown in field 008 on page 1 of Exhibit E, *Flinchbaugh* was cataloged by the library on February 17, 1997 (shown as 970217 in field 008). Based on standard library practices, this reference would have been shelved shortly after it was cataloged (i.e., within a few days or weeks). The above is corroborated by the dates stamped on the cover of the Proceedings, including the "12 Feb 1997" date at the top of the page indicating that it was received into the British Library's "serials file" and the "17 Feb 1997" date on the middle-left side of the page indicating "conference indexed." Ex. D at 1. Field 85241, subfield j of Exhibit E also has the number 1086.803000, which matches the stamp in the upper right corner of *Flinchbaugh*. Collectively, Exhibits D and E show that *Flinchbaugh* was published and accessible to the public years before October 1999.

29. Further evidence of the publication and public availability of *Flinchbaugh* can be found in Exhibit F. For example, Exhibit F-1 is a copy of the '469 Patent, which lists Thomas J. Olson as the inventor and shows that

Flinchbaugh was cited to the Patent Office. *See* Ex. F-1 at 1-2. Based on the Patent Office's prosecution file of the '469 Patent, *Flinchbaugh* was cited by the applicant with the filing of the application on April 15, 1999, more than a year before the earliest possible filing date of the '912 Patent. Ex. F-1 at 20.

30. As further evidence of the publication and public availability of *Flinchbaugh*, Exhibit F-2 is a copy of *Collins,* which is co-authored by one of the inventors of the '912 Patent, Alan J. Lipton. Ex. F-2 at 1. As part of the "References" section in *Collins*, *Flinchbaugh* is listed and *Collins* states that *Flinchbaugh* was published in "June 1994." Ex. F-2 at 15.

31. For the avoidance of any doubt, I note that *Collins* itself was published in 1999 and is a highly cited paper. For example, as indicated on the website "www.semanticscholar.org," *Collins* was published in "1999" and is a highly cited paper, having "1,676 citations." *See* Ex. F-2 at 16. However, none of this alters the fact that *Flinchbaugh* was published and accessible to the public years before October 1999, as indicated above.

Brill

32. As indicated in Exhibit G, *Brill* (Exhibit 1004 to each Petition in IPR2019-00235 and IPR2019-00236) is part of the published Proceedings of the 1998 Image Understanding Workshop. The Workshop was held in Monterey, California during November 20-23, 1998, and the Proceedings were "APPROVED

FOR PUBLIC RELEASE" with "DISTRIBUTION UNLIMITED." Ex. G at 1. In view of the above and the following, the Proceedings, including *Brill*, was published and accessible to the public before October 1999.

33. Evidence of *Brill's* publication and availability to the public includes the hand-written receipt date of "8-13-99" at the top of page 3 of Exhibit G. This indicates it was received by the UMMU (the University of Michigan Media Union, now known as the Duderstadt Center) on August 13, 1999. In my experience as a librarian and knowledge of standard library practices, the hand-written information at the top of p. 2 of Exhibit G appears to be the catalog record information for *Brill*. Based on standard library practices, this reference would have been shelved shortly after being received and cataloged by UMMU.

34. Further evidence of the publication and accessibility of *Brill* to the public can be found in Exhibit H, which is the MARC record for the Proceedings, including *Brill*, that was obtained from the University of Virginia Library. As shown in field 008 near the top of page 2 of Exhibit H, *Brill* was cataloged by the library on December 15, 1998. Based on standard library practices, this reference would have been shelved shortly after (i.e., within a few days or weeks) and been accessible to the public prior to October 1999.

35. Further evidence of the publication and public availability of *Brill* can be found in Exhibit I, which is the MARC record for the Proceedings, including

Brill, that was obtained from North Carolina State University. As shown in field 008 on page 1 of Exhibit I, *Brill* was cataloged by the library on December 15, 1998. Based on standard library practices, this reference would have been shelved shortly after (i.e., within a few days or weeks) and been accessible to the public prior to October 1999.

I declare under penalty of perjury that the foregoing is true and correct. Executed on April 1, 2019 in Washington, D.C.

Emily R. Florio

EXHIBIT A

Visual Memory

by

Christopher James Kellogg

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degrees of

Bachelor of Science
and
Master of Science in Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 1993

Signature redacted
Author ....
Department of Electrical Engineering and Computer Science
April 21, 1993

Signature redacted
Certified by ....
Alex P. Pentland
Associate Professor of Media Arts and Sciences
Thesis Supervisor

Signature redacted
Certified by ....
Bruce E. Flinchbaugh
Manager of Image Understanding Branch at Texas Instruments
Thesis Supervisor

Signature redacted
Accepted by ....
Campbell L. Searle
Chairperson, Departmental Committee on Graduate Students

Visual Memory

by

Christopher James Kellogg

Submitted to the Department of Electrical Engineering and Computer Science
on April 21, 1993, in partial fulfillment of the
requirements for the degrees of
Bachelor of Science
and
Master of Science in Computer Science

Abstract

Visual memory supports computer vision applications by efficiently storing and retrieving spatiotemporal information. It is a unique combination of databases, spatial representation and indexing, and temporal representation and indexing. This thesis designs a visual memory architecture that meets the requirements of a number of computer vision applications. It also presents an implementation of part of this design in support of a scene monitoring prototype.

Thesis Supervisor: Alex P. Pentland
Title: Associate Professor of Media Arts and Sciences

Thesis Supervisor: Bruce E. Flinchbaugh
Title: Manager of Image Understanding Branch at Texas Instruments

Acknowledgements

My primary thanks goes to my two thesis supervisors, Bruce Flinchbaugh at Texas Instruments and Sandy Pentland at MIT. Bruce pointed me to the visual memory project that he was starting and guided my research at Texas Instruments. Sandy provided useful feedback throughout the research stage. They were both very helpful in critiquing the thesis document.

I'd also like to thank the other people at Texas Instruments who helped me with this project. Steve Ford and Tom Bannon were especially helpful in developing the visual memory design. In addition, I don't think I would have survived the bugs in PC++ without Steve's expertise. Tom Bannon and Tom O'Donnell provided a nice tracking system with which to test the visual memory prototype.

Finally, I'd like to thank my family, Fred, Jeannette, and Mark Kellogg, my fiancée Christine Bailey, and my brothers at Phi Kappa Sigma for their support throughout my MIT career.

Contents

List of Figures

Chapter 1

Introduction

Visual memory supports computer vision applications by efficiently storing and retrieving spatiotemporal information. It is a unique combination of databases, spatial representation and indexing, and temporal representation and indexing. Visual memory provides representational flexibility and high-performance information access to meet the requirements of a variety of computer vision applications.

1.1 Needs for Visual Memory

Applications use spatiotemporal data in many different ways and place many different demands on a visual memory. Studying possible uses helps to clarify the concept of a visual memory and to identify the functionality it provides.

Visual memory could serve as the repository for static information, such as object descriptions, maps, and environment models, that applications reference during execution. For example, a vehicle navigator could store maps and images to help it later recognize its location. A large amount of such information could be established prior to application execution, and the visual memory would subsequently provide an application with efficient access to desired pieces of information.

An application could store dynamic information in the visual memory. For example, a vehicle navigator's input systems could maintain in the visual memory a description of the vehicle's local environment, updating it as the vehicle moved. The visual memory could provide the navigator's planning processes with information about the vehicle's latest state and could analyze its progress to help determine a course of action. The high performance of the visual memory allows it to handle the frequent updates and queries needed by such dynamic, real-time systems.

Visual memory could manipulate spatiotemporal information about objects and collections of objects too large to fit into volatile memory. For example, a computer-aided design and modeling system could use the visual memory in building up a large design layout and simulating its execution over time; a photo interpretation system could similarly construct in the visual memory a complex representation of a scene. The visual memory would retrieve into main memory only a manageable part of a large representation at a time.

Visual memory could act as the interface between inputs and applications in a computer vision system. For example, computer vision algorithms for a security system could analyze data provided by various cameras and store information in the visual memory. Applications could then retrieve this data to track objects, watch for suspicious events, and respond to user queries. The visual memory would coordinate the information from its inputs and eliminate the need for full connectivity between inputs and applications.

Finally, visual memory could serve as a means for data transfer. A computer vision application could store spatiotemporal information in the visual memory for other applications to retrieve at any time in the future. To run comparative studies, different algorithms could use common data stored in the visual memory.

1.2 Goals

This thesis explores visual memory design and implementation. The primary goal of the thesis is to design a visual memory architecture that meets the requirements of various computer vision applications. A secondary goal is to implement a visual memory prototype to support a real-time scene monitoring prototype.

Chapter 2

Background

Visual memory builds on research in database design, spatial representation and indexing, and temporal representation and indexing. While there has been significant research in each of these areas, no previous project has combined them in this manner. The visual memory design uses knowledge gained from research projects in all these areas. This chapter summarizes and discusses some especially relevant projects.

2.1 Database Research

Visual memory must address concerns that a great deal of database research has already investigated. It must provide everything from information storage techniques to concurrency control for multiple inputs and outputs. Visual memory should build on the results of research into these topics. Presented here are two databases that address a number of the issues important to visual memory and that could be the basis for a visual memory system.

2.1.1 DARPA Open OODB

The DARPA Open Object-Oriented Database (Open OODB) project at Texas Instruments outlines an extensible architecture that allows "...tailoring database functionality for particular applications in the framework of an incrementally improvable system ...." [25] The architecture meets functional requirements such as an object data model and concurrent access, along with "meta requirements" including openness and reusability. The open architecture lets separate modules handle extensions to the basic storage mechanism. These extensions cover standard database issues such as transactions, versions, and queries.

The Open OODB architecture is very suitable for visual memory. The object-oriented model can flexibly and intuitively represent the information used by computer vision applications. Following the Open OODB architecture, visual memory could avoid confronting standard database issues by letting other modules support those features. Instead, visual memory would consist only of those extensions necessary to support efficient manipulation of spatiotemporal information. If new features were needed, extra modules could easily be added to the architecture.

2.1.2 POSTGRES

The POSTGRES database [23] expands the relational database model to meet the needs of complex applications. Because it builds on traditional relational databases, it provides a number of standard features, such as transactions, a query language, and recovery processing. In addition, it allows applications to specify new data types, operators, and access methods. POSTGRES supports active databases and rules, letting applications set up daemons in the database that react to changes in the data. A versioning mechanism keeps track of old data and works with the query language to let applications retrieve this information. Finally, the POSTGRES storage server can "vacuum" old data onto archival media.

POSTGRES supplies many features useful to a visual memory, such as transactions, queries, and application-defined access methods. However, the relational model might not be sufficiently expressive to meet the representational needs of complex computer vision applications. In addition, the POSTGRES design does not support application-specific extensions to the database, so it would be hard for the visual memory to expand to meet future requirements.

2.2 Spatial Research

There are many ways to describe spatial objects and to handle their storage and retrieval. Visual memory must consider how well different spatial models meet the representational needs of computer vision applications and how efficiently information in these models can be stored and retrieved.

2.2.1 CODGER

Researchers at Carnegie Mellon University developed the CODGER (COmmunications Database with GEometric Reasoning) "whiteboard" database and communication system to support the autonomous NAVLAB vehicle [20]. CODGER stores data to be communicated among the various modules that control vehicle navigation. It represents this information as tokens consisting of attributes and values.

CODGER uses a fairly simple spatial model. Token attributes represent basic spatial information such as position and object extent. The tokens support some standard geometric operations like area calculation. A query mechanism can answer some spatial queries like the proximity query "Return the tokens with location within 5 units of (45,32)." CODGER does not provide an indexing mechanism, and spatial operations and queries are performed in memory.

2.2.2 Core Knowledge System

The Core Knowledge System (CKS) [24], developed at SRI International, stores information for a robot. Like CODGER, it encodes this information as attribute-value tokens. CKS introduces special support for the uncertainty that results from inconsistent or incomplete information provided to the database. Its query mechanism includes keywords such as *apparently* and *possibly* to discern multiple opinions. Since spatial information is often imprecise, this support for uncertainty would be very useful in a visual memory context. However, CKS does not provide any special spatial operations or query constructs.

2.2.3 ISR

The ISR project at the University of Massachusetts at Amherst [3] defines a spatial representation (the Intermediate Symbolic Representation) and a management system for accessing data represented this way. The intermediate symbolic representation includes tokens for basic spatial objects such as lines, regions, and sets of parallel lines, but not for higher-level spatial objects such as people and vehicles. The data management system manipulates these tokens in an efficient manner. Applications built with ISR perform classification and in-memory spatial indexing.

2.2.4 Image Understanding Environments

The Image Understanding Environments (IUE) program [16] specifies a spatial representation to meet the needs of a wide variety of computer vision applications. An IUE spatial object is defined by a set of points; this point set can be concrete (a list of all the points) or abstract (an equation defining the points in the object). IUE spatial objects are manipulated through set operations – complex objects can be constructed through conjunction and disjunction of point sets. In addition to its point set, each spatial object also defines a bounding box, a centroid, and other attributes for different, and perhaps more efficient, methods of spatial manipulation. The IUE specification only briefly discusses data transfer and does not provide database support for storage and retrieval of spatial information.

2.2.5 PROBE

The PROBE database [15], developed at the Computer Corporation of America, extends an object-oriented database management system to meet the requirements of a variety of computer vision applications. It implements a number of spatial data types and supports operations on sets of points. It outlines a query language with some support for spatial queries. To provide more efficient spatial access, it also provides what the authors call *approximate geometry*, a limited form of spatial indexing.

2.2.6 Spatial Indices

A large number of spatial data structures can provide efficient access to spatial information. Samet [18] describes a number of these, including quadtrees, hash tables, grid files, range trees, and R trees. Each index is specialized for specific storage and retrieval characteristics; visual memory would benefit from including a number of different indices to efficiently manipulate data for different applications.

2.3 Temporal Research

Databases manipulate two different types of time: *transaction time*, specifying when updates for events are stored in the database, and *valid time*, specifying when events actually happen. *Rollback databases* implement transaction time, *historical databases* implement valid time, and *temporal databases* implement both. Sometimes historical and rollback databases are informally called temporal databases to indicate their concern with time. Since the computer vision applications discussed in the Introduction are concerned with the times at which events happen, visual memory should be a historical database.

A number of different historical and temporal databases represent and store temporal information. Each addresses a different set of concerns, and some designs suit visual memory better than others. The following research projects address many of the issues that visual memory must consider.

2.3.1 TQuel

The temporal database TQuel [21] is a temporal extension to a relational database. TQuel associates with each database record the slots *valid-from* and *valid-to*, defining an interval during which the record is valid. For example, the Employees relation might have three records for Frank, one valid from 0 to 1/1/93, another valid from 1/1/93 to 5/7/93, and a third valid from 5/7/93 to ∞. If Frank were changed on 8/7/93, then the third record's *valid-to* slot would be changed to 8/7/93, and a new record valid from 8/7/93 to ∞ would be added.

TQuel extends the query language Quel [12] to support temporal access of records. A temporal query specifies an interval of interest; the database retrieves any record whose valid interval overlaps that interval. A query can also ask for records before, after, or as of a given moment. TQuel provides operators such as *overlaps* and *extend* to form complex query intervals.

2.3.2 Temporal Sequences

The temporal database outlined in [19] models object state changes with temporal sequences. A temporal sequence can be discrete or continuous; for example, sales per month could be modeled as a discrete temporal sequence, while the voltage in alternating current could be modeled as a continuous temporal sequence. A temporal sequence is always represented by a set of state snapshots; interpolating functions estimate continuous sequences. Characteristics such as granularity and regularity of state snapshots define each temporal sequence. Functions including selection, aggregation, and accumulation operate on sets of time sequences. The database also includes a powerful SQL-like [1] query language for retrieving temporal sequences.

2.3.3 Temporal Sets

Researchers at the University of Houston proposed some temporal additions [8] to the Extended Entity-Relationship Model. The basic temporal representation in this temporal model is a finite union of time intervals; for example, a particular state could be valid during the set of time intervals { [50,60), [90,230), [231,239) }. The database stores with each object a temporal element denoting its valid time. Basing temporal representation on sets of intervals preserves closure under set operations and provides a standard means for manipulating temporal information and querying the database.

This model was later augmented to better represent temporal uncertainty [13]. The extended model preserves the definition of a temporal element but modifies the definition of a temporal interval. Each endpoint in an interval specifies a valid time method that returns an ordered set of time points. The endpoint belongs to this set, but in order to allow for uncertainty, it is not explicitly specified. The model also modifies the standard set operations to manipulate uncertain temporal elements.

2.3.4 Relative Time

Some applications, such as computer-aided design systems, know how events are ordered but not the actual times of the events. Chaudhuri [5] proposes a temporal model to handle these cases. This model represents time as a graph rather than as a time line. Events are ordered with binary relations like *before* and *simultaneously*. These relations must obey properties such as transitivity and antisymmetry so that the database can navigate through a graph and infer additional relationships. The model supports temporal queries about event relations; for example, a query could ask for a lower time bound on an event or for common ancestors of two events. This capability could be useful in a visual memory to support efficient handling of temporal information for some applications.

2.3.5 Temporal Indices

Much of the spatial indexing research also applies to temporal indexing. For example, interval trees can store intervals in space or in time. To handle more complex, specialized temporal representations, however, requires additional research. Some of the databases described above provide their own temporal indices; [22] references many other systems with temporal indices.

Chapter 3

Design

This chapter presents a design for a visual memory system. It examines requirements and considerations that the design must take into account. It discusses key visual memory topics such as representation and indexing of spatial, temporal and spatiotemporal information. This chapter outlines a concrete, implementable system; the next chapter presents the prototype implementation of this design.

3.1 Requirements and Considerations

The design of a visual memory must address a number of concerns. Some of these come from anticipated uses of the visual memory, while others are common themes in spatial, temporal, and database research. This section covers a number of these requirements and considerations.

3.1.1 Database Considerations

One database issue relevant to visual memory is how to represent and store information. There are several standard models, including the relational model, the entity-relationship model, and the object-oriented model. The visual memory should use an object-oriented model to meet the broad representational requirements of a variety of applications. An object-oriented approach is intuitive and highly extensible, allowing applications to define new, complex objects at any time.

Another important consideration is concurrency control. The visual memory must be able to handle multiple, dynamic inputs and outputs. For example, in a scene-monitoring system, many different cameras could update the visual memory simultaneously. The visual memory must ensure data consistency.

Much database research involves well-defined program interfaces, including explicit storage mechanisms and query algebras. Applications using the visual memory do not need to know how it achieves its results, but they should know what results to expect. For example, performance-enhancing measures such as indexing and caching do not affect the objects returned by a query and can be added without affecting the query algebra.

Recoverability is another database issue important to some visual memory applications. The visual memory must work to guarantee that, even in the case of a system crash, it does not lose stored information. In addition, it must be able to remove inconsistencies resulting from system failure during information storage.

3.1.2 Spatial and Temporal Considerations

The purpose of visual memory is to store information about the history of a visual environment. Visual memory is not just a generic database — it must have spatiotemporal concerns at the heart of its design.

A visual memory must provide representational flexibility. Rather than forcing one spatiotemporal representation on all applications, the visual memory should be tailorable to an application's needs. Applications can trade off between representational power and performance.

A visual memory must handle dynamic objects. Some computer vision applications need to update spatial information in response to changes in the environment. The visual memory must define spatiotemporal representations to effectively handle such changes. It must provide a versioning mechanism to store and retrieve different state snapshots of objects.

A visual memory must provide a flexible, expressive query mechanism with extensive spatiotemporal support. This query mechanism should support a wide variety of spatiotemporal queries. For example, a security system might ask the visual memory to retrace a person's path over the past five minutes, a vehicle navigator might ask it to watch for objects entering the field of view, and a CAD system might ask for simulation results for everything electrically connected to a specific chip. The visual memory should let applications conveniently express such queries.

3.1.3 Performance Considerations

High performance is one of the key requirements for a visual memory. Some visual memory applications, such as a vehicle navigator, need to store and retrieve information very quickly. Many spatial and temporal models in the literature are very expressive but do not provide the necessary information throughput. A visual memory must be both expressive and fast enough to meet the demands of its applications.

Indexing can help a visual memory achieve high performance by quickly identifying objects satisfying given constraints. Visual memory indices should be *conservative*, never mistakenly omitting objects that satisfy a query. In this manner, indices can improve query performance but are guaranteed to not affect the results.

A visual memory must provide a variety of indices to meet the needs of different applications. For example, a real-time scene monitoring system could set up an index to track the centroids of moving objects, while a photo interpretation system could index the areas covered by objects. A visual memory indexing mechanism should be extensible, handling additional application-defined indices.

A visual memory must let applications control which objects are indexed. For example, an application could establish one index on all objects, another index on everything in the current session, and yet another index only on certain objects of interest. This would prevent the visual memory from wasting time and space updating unimportant indexing information.

Caching and look-ahead techniques can increase the performance of a visual memory. Caching improves storage performance by not requiring the visual memory to wait for information to be written to disk. Both caching and look-ahead improve retrieval performance by reducing the number of disk accesses.

Visual memory performance can be increased by letting applications tailor the visual memory to their specific requirements. For example, some applications can afford to lose a small amount of data, so they could eliminate recoverability information. Other applications could optimize specific storage and retrieval cases; for example, a vehicle navigator could optimize its real-time performance by sacrificing some historical performance.

3.2 Design Overview

The visual memory design consists of a set of extensions to an open database architecture like DARPA Open OODB [25]. An open architecture allows the visual memory to add spatiotemporal customizations to the database. The visual memory can take full advantage of other modules implementing features such as concurrency control, caching, and versioning, without having to handle these capabilities directly.

The visual memory design follows the object-oriented model discussed in the previous section. A class hierarchy defines representations for spatiotemporal information. Abstract superclasses define the interfaces for manipulating spatiotemporal information, and their subclasses extend the definitions to represent more specific types of objects. This document denotes classes in italics; for example, *SpatialObject* is the class representing spatial objects. A concrete member of this class is referred to as "a *SpatialObject* instance" or informally just as "a spatial object."

The visual memory design specifies a number of classes for representing spatiotemporal information. These classes provide methods through which computer vision applications and the visual memory can manipulate them. For example, the spatial class *Square* could include a method to return its area, the temporal class *TemporalInterval* could have a method to determine its duration, and the spatiotemporal class *Person* could implement a method plotting its space-time trajectory. Applications can design their own classes inheriting from these classes and extending them to meet additional needs.

The visual memory design extends the database's storage mechanism. It provides a mechanism for object identity and maintains a history for each object. Each version of an object specifies when it was valid, and the visual memory can manipulate versions based on valid time. The design lets applications customize the database storage server based on characteristics of the data they typically store.

The visual memory design extends the database's query mechanism to provide spatiotemporal support. The additional spatiotemporal constructs allow computer vision applications to flexibly and expressively specify objects of interest.

To achieve suitable query performance, the visual memory provides spatiotemporal indices that can efficiently identify objects satisfying query conditions. A visual memory index is an object that maintains information about other objects, allowing it to efficiently indicate those objects that meet certain constraints. For example, a visual memory spatial index might store object centroids so that it can quickly identify all the objects within a specified area. The visual memory provides a powerful and flexible indexing mechanism.

3.3 Spatial Representations

The visual memory spatial class hierarchy provides a powerful framework that allows applications flexibility in designing spatial representations while ensuring that the visual memory can access the information it requires. The class hierarchy draws on the research outlined in the Background chapter. It provides the basic framework for any visual memory application, and it allows applications to extend it to meet additional needs.

Spatial operations are often complex and require much computation. Spatial indices, described in Section 3.8, can increase the performance of these operations by maintaining information about sets of spatial objects. This chapter presents a number of spatial operations; Section 3.8 describes related performance issues.

3.3.1 Core Spatial Classes

SpatialObject

The *SpatialObject* class is the basis for all high-level spatial representations. Possible subclasses derived from *SpatialObject* include *Cube*, *QuestionMark*, and *Person*, depicted in Figure 3-1. *SpatialObject* captures the common representational requirements of a variety of such spatial objects. It provides a standard set of slots and methods to yield a consistent spatial interface. Applications can design additional spatial representations as long as they provide the same functionality.

A spatial object is defined by a set of points and a local coordinate system. This information is sufficient to fully represent a spatial object. The point set specifies what area of space the object fills. The coordinate system relates these points to the points in other spatial objects. Additional information, such as centroid, orientation, and bounding box, is derivable from this information.

SpatialObject provides a wide variety of methods to manipulate its data. These methods can translate and rotate an object, operate on its point set, and find the object's bounding box, among other things. Most of these are actually point set and coordinate system functions and will be discussed further below. Concrete spatial

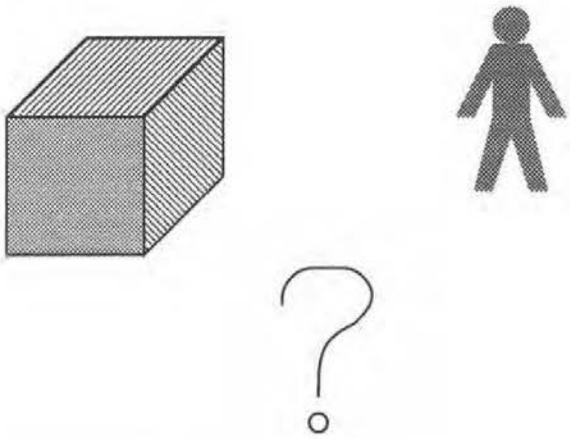


Figure 3-1: Spatial objects

objects can provide additional relevant information; for example, a cube could have functions returning the length of its side, its surface area, and its volume. Using the methods of *SpatialObject* and its subclasses, an application can manipulate spatial data in many different ways.

Several lower-level classes manipulate information for the high-level *SpatialObject* class. The following sections present these classes.

Point

The most elementary unit of spatial representation is the point. The visual memory provides the abstract class *Point* and subclasses *TwoDPoint*, *ThreeDPoint*, etc., to represent this elementary unit. *Point* is a fairly simple class, only storing and manipulating a coordinate in some space. As will be shown below, however, it is an important building block.

PointSet

Complex spatial information can be represented as a collection of points, or point set. The visual memory provides the class *PointSet*, derived from a generic *Set* class,

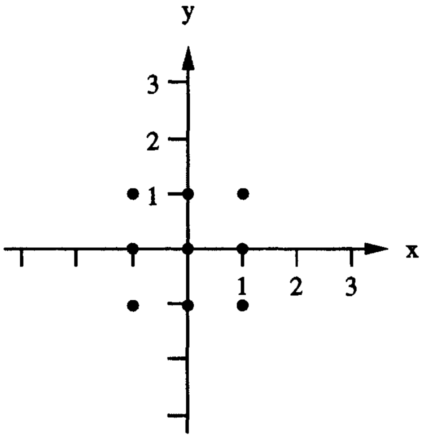


Figure 3-2: Discrete point set

to store and manipulate sets of points. Since *PointSet* is a kind of *Set*, it provides standard set operations, such as union, disjunction, member, and difference. This allows a powerful means for constructing complex objects. It also furnishes a well-defined and sound mathematical basis for spatial representation and manipulation.

The class *DiscretePointSet* represents a set of points simply as an exhaustive list of all desired points. This representation is feasible only for small point sets. For example, consider the task of representing the square area of the points plotted in Figure 3-2. A system could, by convention, represent a square area by such a discrete set of points. Standard set operations can easily manipulate this information. Unfortunately, the space required for this representation grows too quickly to be broadly applicable.

The class *AbstractPointSet* is a far more efficient means for representing large or even infinite point sets. It abandons an exhaustive list of all points in favor of a functional definition of the points in the set. An abstract point set specifies a function that returns TRUE for points in the set and FALSE for points not in the set. For example, the function for the continuous square in Figure 3-3 would check for $-1 <= x, y <= 1$. This fully represents the square area. A point set's representation function grows complex as the set is modified by operations such as conjunction and

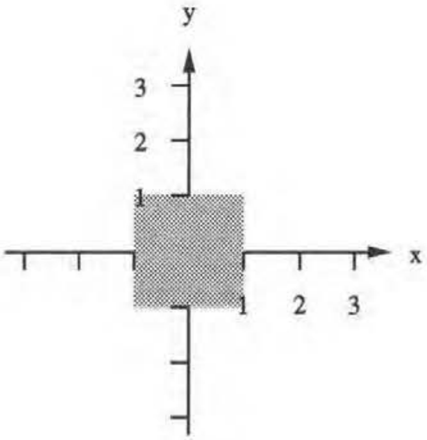


Figure 3-3: Abstract point set

disjunction, but a suitably complex function can represent any desired set of points.

Some visual memory applications start with discrete approximations to the continuous world but want to interpolate to a continuous description. For example, an application might recognize only the list of discrete points above that make up just a small part of an actual continuous square. For these applications, a subclass of *AbstractPointSet*, *InterpolatingAbstractPointSet*, can apply a specified interpolation function to that list of points to derive a continuous function. For example, an interpolation of the point set in Figure 3-2 could yield the point set in Figure 3-3.

An instance of *PointSet* is more than just a set of points; it also includes a number of methods deriving spatial information from this set. Important methods find a point set's centroid, boundary, bounding box, and surface normal, among other things. These methods extend the power of the point set and enhance the visual memory spatial support.

CoordinateSystem

A point in space is useful only in relation to other points. The *CoordinateSystem* class establishes relationships between points in the visual memory. Figure 3-4 shows a couple of possible coordinate systems. Each *CoordinateSystem* subclass must define

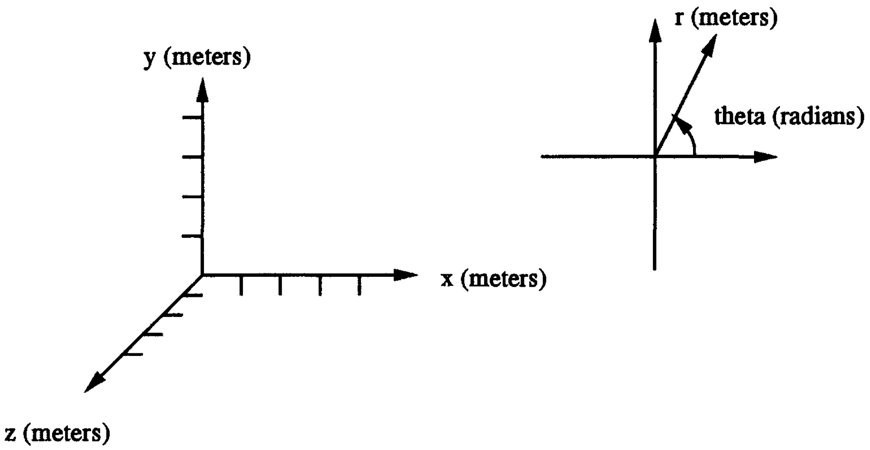


Figure 3-4: Coordinate systems

dimensions, axes, and other features of the space. These specifications give meaning to points and provide the basis for relating points. The visual memory defines a coordinate system for a set of points; the *SpatialObject* class associates a local coordinate system with each point set.

The main job of a coordinate system is to relate points. To do this, it maintains a list of coordinate transforms between it and other coordinate systems. To achieve high run-time speed efficiency, a coordinate system can maintain transforms between it and several other coordinate systems. Alternatively, it can trade off speed of operation for lower space requirements by storing only a few transforms and letting the visual memory follow a chain of transforms among related coordinate systems.

To reduce the cost of multiple transforms, an application can adopt a unified coordinate system to relate a number of nearby local coordinate systems. This unified coordinate system would maintain transforms to and from each local coordinate system. In this manner each coordinate system would not need to keep a large list of transforms, and only two transforms would be needed to relate points in one coordinate system to those in any other. A limitation of this approach is that it does not scale well for large distances, because the error induced by each transform could compound significantly.

The *CoordinateSystem* class provides methods to transform a coordinate system's relationship with other coordinate systems. For example, one coordinate system might translate and rotate with respect to others. Transforming a coordinate system modifies its list of coordinate transforms, and all coordinate transforms between it and other systems must be updated. This is automatically provided by the visual memory as part of the transformation method.

Transforms like translation and rotation are *CoordinateSystem* methods rather than *PointSet* methods for a number of reasons. The coordinate system relates the point set to other coordinate systems, and it is probably more efficient to store a transform than a transformed point set for each other coordinate system. It is also more efficient to accumulate a set of transforms into one transform than to repeatedly apply transforms to a whole set of points. If the points are represented by a function, it could be hard to determine how the transform should modify that function. The transform could be applied only when needed; if it were used repeatedly, the results could be cached.

Coordinate system transforms permit the construction of multiple-object scenes. Each spatial object is developed in its local coordinate system, and then coordinate system transforms construct relations between local coordinate systems. The opposite effect occurs when multiple sets of points in one coordinate system are split into separate spatial objects with local coordinate systems. In this case, the transformation from each local coordinate system to the original unified coordinate system is already defined. Standard computer graphics texts, such as [11], discuss coordinate system transforms and the construction of multiple-object scenes in further detail.

3.3.2 Relative Spatial Specification

In many cases, a coordinate system has explicitly-defined relationships to other coordinate systems. For example, one coordinate system might have an origin 3 units to the east of the origin of another coordinate system. In other instances, however, this information is not so clear. For example, an application might only need to know

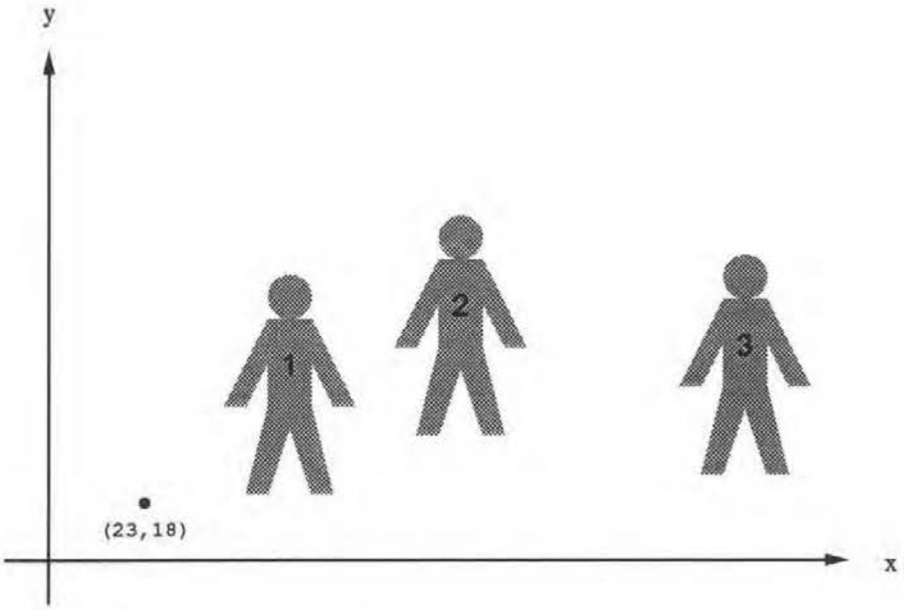


Figure 3-5: Relative spatial objects

that one block was to the east of another block, without knowing an explicit distance. In these cases a relative spatial specification is required.

There are actually two kinds of relative spatial specification: specification relative to a concrete position or object with concrete position, and specification relative to another relative spatial specification. For example, in Figure 3-5 the visual memory knows that object 1 is to the east of the point (23,18), object 2 is to the east of object 1, and object 3 is to the east of object 2. This description does not precisely specify the scene; for example, object 2 could be further to the north and east and still meet the specification.

The visual memory provides the class *RelativeSpatialObject*, a subclass of *SpatialObject*, to handle relative spatial specification. A relative spatial object simply keeps lists of objects relative to it in various ways. For example, one subclass of *RelativeSpatialObject* might provide lists for objects west, east, north, and south of it, while another subclass might provide a list for nearby objects, where "near" is defined by a method of the class. *RelativeSpatialObject* can represent both kinds of relative spatial specification mentioned above, since an instance can be defined relative to any spatial object, including a fixed position or another relative spatial object.

An application can construct arbitrary graphs of relative spatial objects. For example, in Figure 3-5, object 1 is to the west of object 2, which is to the west of object 3, and so forth. *RelativeSpatialObject* provides methods to trace through the transitive closure of a graph operation. In the above example, since object 1 is to the west of object 2 and object 2 is to the west of object 3, it follows that object 1 is to the west of object 3. Both objects must keep track of the relationship so that the connection can go in either direction; in the above example, it also follows that object 3 is to the east of object 1. If a large number of links separate two related objects, an application might want to establish a direct connection. Alternatively, the visual memory could cache this information.

The design of *RelativeSpatialObject* must determine how to handle transformation of an object in a relative object graph. In Figure 3-5, object 2 was to the east of object 1. If object 2 moved west, it could be either to the east or to the west of object 1, as shown in Figure 3-6 and Figure 3-7 respectively. When an object is transformed, the visual memory must eliminate all of its relative dependencies. If objects maintain their relationship after transformation, that relationship must be reasserted. If objects are somehow connected so that the relationship is always maintained, they should be established as subobjects of a larger object that maintains the relationship.

3.3.3 Uncertain Spatial Specification

Some computer vision applications do not know exactly where objects are located and exactly which points are in the point sets. They deal with approximate information and conflicting evidence from multiple sources. These applications require uncertain spatial specifications.

The visual memory class *ProbabilisticPointSet*, a subclass of *PointSet*, represents uncertain spatial information. *ProbabilisticPointSet* associates with each point the probability that it belongs to the point set. Thus instead of just knowing that point (3,4,5) was in a point set, a probabilistic point set would know that point (3,4,5) was

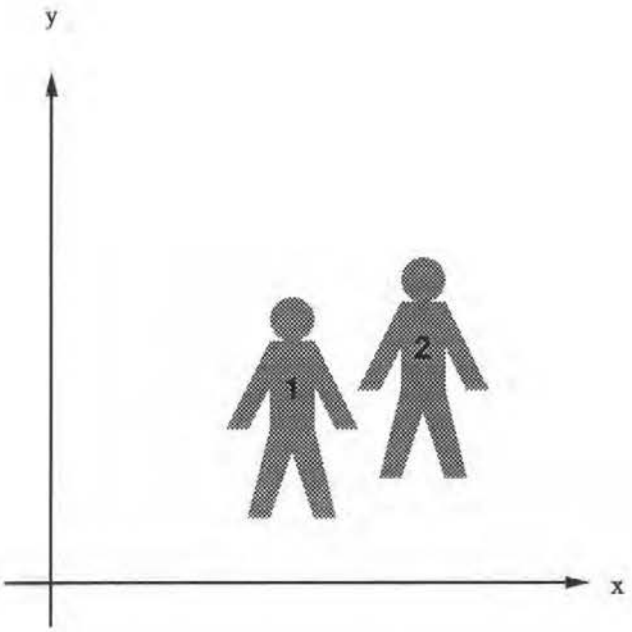


Figure 3-6: Breaking a relative spatial specification, part 1

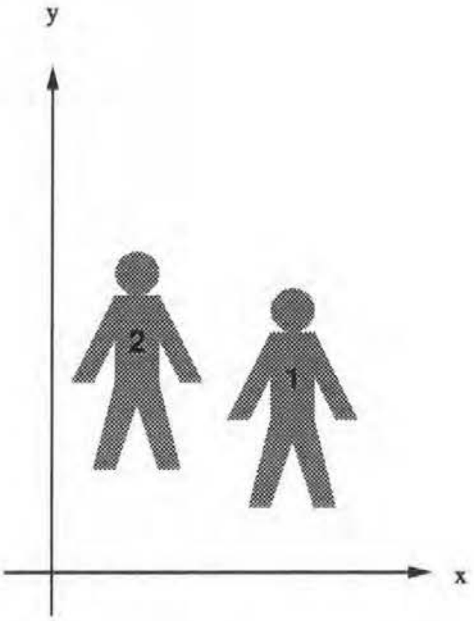


Figure 3-7: Breaking a relative spatial specification, part 2

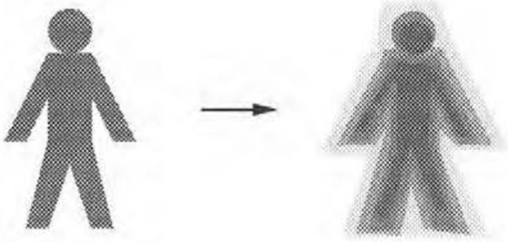


Figure 3-8: Uncertain edges

in the point set with probability 0.7. Point probabilities allow applications to specify empirical certainty factors for point sets. As will be shown below, this provides great flexibility in representing uncertain spatial information.

The standard *PointSet* methods must be modified to take point probabilities into account. The methods can reflect uncertainty in various ways. For example, there is no real boundary to a point set with point probabilities asymptotically approaching 0. However, an application can define the boundary as the set of points with probability 0.1. Subclasses of *ProbabilisticPointSet* support such variations.

Like *PointSet*, *ProbabilisticPointSet* has both discrete and abstract subclasses. The discrete subclass maintains a list of <point, probability> pairs, while the abstract subclass defines a function that returns a probability for a given point. Set operations can combine the point probabilities from different point sets by maximizing, minimizing, or averaging, among other operations.

Probabilistic point sets can handle a number of different types of uncertain spatial specification. The following sections examine a few of these.

Uncertain Edges

ProbabilisticPointSet lets applications be fuzzy about the region of space occupied by an object. For example, the probability function can decrease at the edges of an object, where a segmentation algorithm might be unsure of exactly how to separate regions. Figure 3-8 shows examples of a standard person and a person with uncertain edges, where darker regions have higher probability.

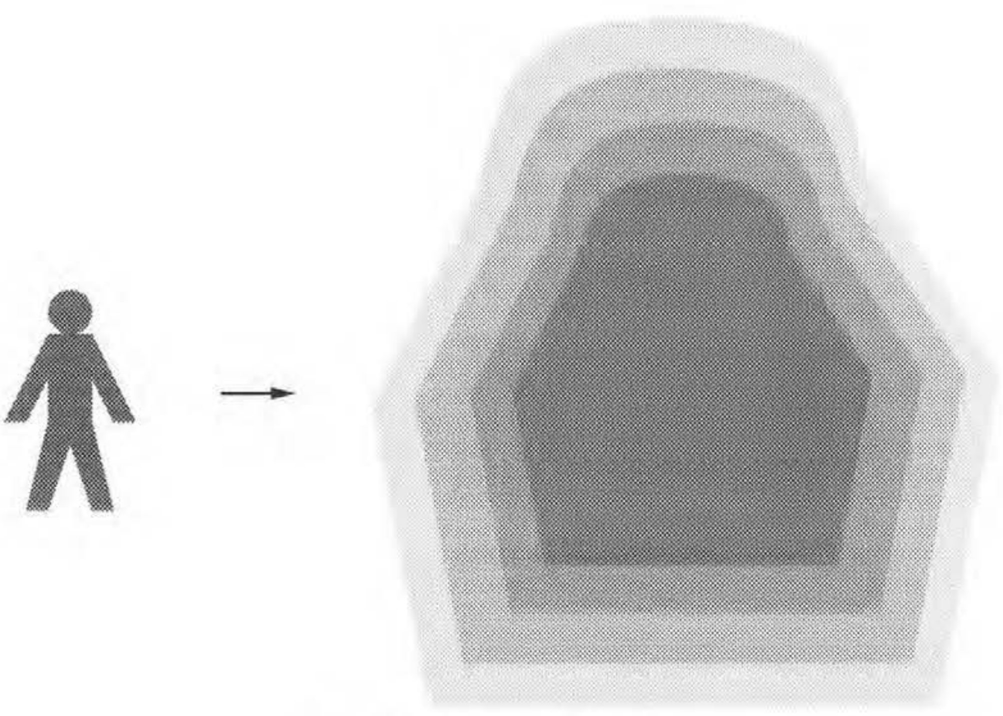


Figure 3-9: Uncertain location

Uncertain Location

An application might know the points in the point set but might not be sure of its exact location. For example, a tracking algorithm might identify a group of points as a person and decide to use the default point set to represent it. However, it might know the person's location only to within a meter. The point set for this particular person can be "spread out" to cover the range of possibilities. An application can indicate the areas most likely to contain the object by giving them the highest probability. Figure 3-9 shows a person and a "spread out" probabilistic point set, with darker regions for points with higher probability.

Conflicting Information

An application might have separate processes providing inconsistent information to the visual memory. For example, one process in a vehicle navigator might identify a

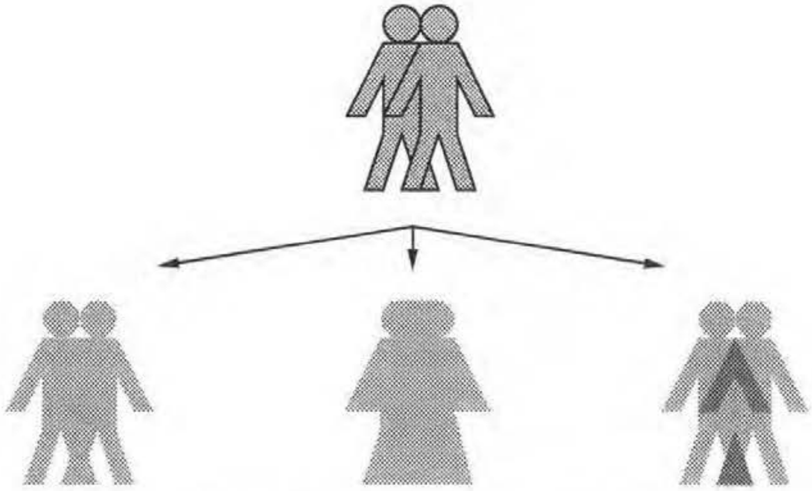


Figure 3-10: Conflicting information

car at one location, while another process might identify the same car at a slightly different location. *ProbabilisticPointSet* handles this situation similarly to uncertain location, but it must combine only a discrete set of point sets rather than spreading out one point set over a continuous area.

ProbabilisticPointSet provides several different ways to combine point sets. For example, it can combine point probabilities by maximizing, minimizing, summing, or differencing (with probabilities staying between 0 and 1), or it can interpolate between the point sets. These approaches to combining probabilities are similar to those taken by expert systems [7] and multi-valued logics [14].

Figure 3-10 shows two point sets to be combined and the results of three different types of combination. The points in the original point sets have the same probabilities. The leftmost combination is formed by maximization; since the probabilities are all the same, this is equivalent to a point set union operation. The middle combination interpolates horizontally between the two point sets. The rightmost combination adds probabilities, assigning higher probabilities to the areas where the point sets overlap.

3.4 Temporal Representations

Temporal representations fit into another branch of the visual memory class hierarchy. There are some parallels between the spatial branch and the temporal branch, but the temporal branch has many of its own requirements and features. This section presents the visual memory temporal representations. Like spatial operations, many temporal operations are complex and require the indexing mechanism of Section 3.8 to achieve high performance.

3.4.1 Core Temporal Classes

TemporalObject

The class *TemporalObject* is the basis for high-level representation of temporal information in the visual memory. Visual memory is concerned with *valid time*, the time at which events happen. *TemporalObject* provides slots and methods defining a standard interface for visual memory temporal support. Its subclasses extend the definition to handle additional temporal information. Any class that needs to keep track of its valid time should inherit from *TemporalObject*.

TemporalObject represents valid time as a set of time intervals and a local clock. It provides methods to manipulate this information, setting and retrieving the valid time, relating the clock to other clocks, and so forth. Most of these methods are furnished by the lower-level classes that make up *TemporalObject*, discussed further in the following sections.

VMTime

The most elementary temporal representation is the class *VMTime*, an abbreviation for *Visual Memory Time*. An instance of this class represents exactly one point in time. Like its spatial counterpart *Point*, *VMTime* is a fairly simple but essential building block in the visual memory class hierarchy.

TemporalInterval

Most objects are valid not for just one point in time but rather for some duration of time. The visual memory provides the class *TemporalInterval* to represent temporal extents. *TemporalInterval* is defined as an open interval $[t_1, t_2)$ to denote valid time from time t_1 to time t_2. The interval is open because applications generally recognize when an object is first valid (t_1) and when it is first invalid (t_2); its valid interval then extends from t_1 up to but not including t_2.

TemporalInterval provides a variety of methods for manipulating temporal information. Standard methods set and retrieve the starting and ending times of the interval. Additional methods check interval overlap, combine overlapping intervals, and check the equality of intervals.

TemporalElement

While some temporal databases use the temporal interval as the main temporal representation, that is insufficient for all visual memory applications. One problem is that the difference of two temporal intervals might not be a temporal interval: if interval 1 covered [10, 30) and interval 2 covered [15, 25), the difference would be [10,15) $\cup$ [25,30). The same problem occurs with disjunction, when an object is valid for multiple distinct intervals. The visual memory follows Elmasri [10] and goes one step further than *TemporalInterval* to provide a more powerful temporal representation.

The class *TemporalElement* maintains a temporal object's valid time in the visual memory. A temporal element consists of a set of temporal intervals. Thus it is closed under set operations and can represent complex temporal specifications. Each of the less expressive temporal representations is a subcase of *TemporalElement*: *TemporalInterval* is a singleton *TemporalElement* and *VMTime* is a singleton *TemporalElement* with the same starting and ending point. Figure 3-11 depicts an example temporal element.

TemporalElement furnishes many methods for manipulating its temporal information. It is a subclass of the generic *Set* class, so it provides standard set operations such as member, conjoin, disjoin, and difference. In addition, by using *TemporalEle-*

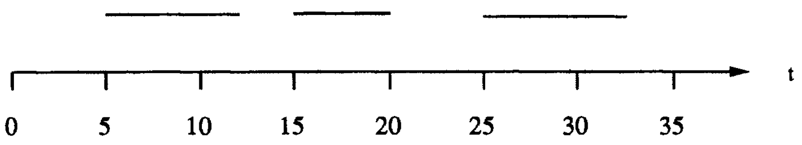


Figure 3-11: Temporal element

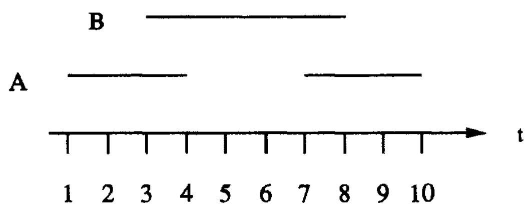


Figure 3-12: Overlapping temporal elements

ment methods, an application can set and retrieve valid times, compare valid times, combine overlapping intervals in a temporal element, and resolve two temporal elements, eliminating overlapping times from one in favor of the other.

Resolution of conflicting temporal elements is an important concept in the visual memory. An application can specify what to do in case of conflict between valid times: it can resolve in favor of the original valid time, it can resolve in favor of the new valid time, or it can leave them in an inconsistent state. Figure 3-12 shows two overlapping temporal elements, version A and version B; Figure 3-13 and Figure 3-14 show the two ways in which they can be resolved. Temporal resolution is especially useful for an application that is initially unsure of the full extent of an object's valid time. The application could assume that the object was valid from a given point until told otherwise and then later resolve that when it learned more information.

Like its spatial counterpart *PointSet*, *TemporalElement* has both discrete and abstract subclasses. The class *DiscreteTemporalElement* lists all the temporal intervals in the set, while the class *AbstractTemporalElement* uses a function to determine whether or not a given temporal interval is in the set. Since time is one-dimensional

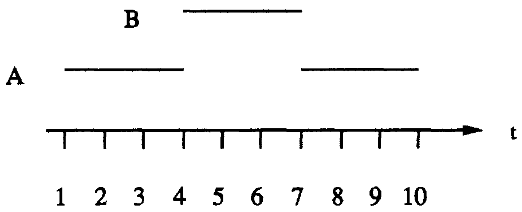


Figure 3-13: Temporal resolution in favor of version A

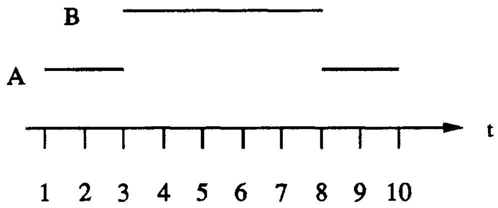


Figure 3-14: Temporal resolution in favor of version B

and most valid times are in just a few continuous blocks, the discrete class is probably more useful for most applications. The abstract class is available for applications that need to represent a large number of disjoint intervals.

Clock

A time point makes sense only with specification of the clock on which it was measured. The visual memory provides the class *Clock*, the temporal analog of the spatial class *CoordinateSystem*, to represent this information. Each clock can assign a different meaning to time points: one clock might use milliseconds since January 1, 1900, while another might use seconds since March 8, 1970. In addition, a *Clock* instance can specify the machine on which the clock is located so that applications can try to account for inaccuracies and differences between system clocks.

The *TemporalObject* class associates a clock with a temporal element. Clocks are associated at this level of granularity because *TemporalElement* is the main visual memory temporal representation. Using a finer granularity would hurt performance

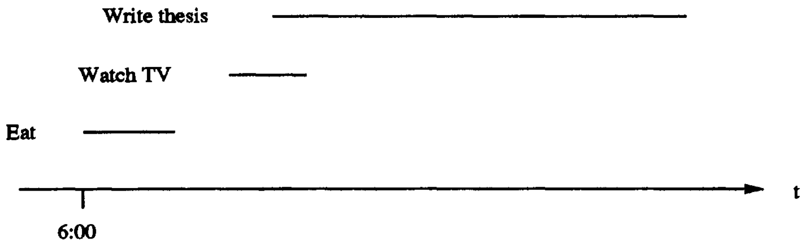


Figure 3-15: Relative temporal specification

for complex temporal specifications and would not greatly improve performance for simple temporal specifications that can be represented as trivial temporal elements.

Like each coordinate system, each clock provides a set of transforms between it and other clocks. This establishes meaning behind the time points associated with a clock and allows the visual memory to convert times among clocks. To increase performance, applications can use the same or compatible clocks.

3.4.2 Relative Temporal Specification

Some applications, such as planners and schedulers, do not know explicit temporal information but can specify some relative ordering of events. For example, a planner might know that it must move to the other side of the room, which will take 5 seconds, before it can pick up a block. To support these applications the visual memory provides classes representing relative temporal specifications.

There are two kinds of relative temporal specification: specification relative to a definite time or object with a definite time, and specification relative to another relative temporal specification. For example, Figure 3-15 illustrates that I plan to eat dinner after 6:00, watch TV after that, and start writing my thesis while I watch TV. This description does not precisely specify the times of these events; if I took a longer break between eating and watching TV the relative specification would be the same.

The visual memory class *RelativeTemporalObject*, a subclass of *TemporalObject*, supports relative temporal specification. A relative temporal object maintains a list of other objects to which it is temporally related. For example, one subclass of *RelativeTemporalObject* might keep track of events before and after it, while another might maintain a list of events happening at approximately the same time.

RelativeTemporalObject allows applications to build arbitrary graphs of temporal relations. For example, the specification in Figure 3-15 directly relates a time and three temporal objects. *RelativeTemporalObject* also provides methods to trace through the transitive closure of a graph. In this example, it could report that I will study after 6:00. Both related objects must keep track of the relationship so that the link can be traversed in either direction. In this manner, the visual memory could also report that 6:00 is before the time when I will study.

3.4.3 Uncertain Temporal Specification

In many cases an application might be unsure about the valid time of an object's state. This could happen, for instance, if the application did not notice an abrupt change of state or could not pinpoint the time of the state change. The visual memory provides two classes, *ProbabilisticTemporalInterval* and *ProbabilisticTemporalElement*, to support uncertain temporal information. Like their spatial counterpart *ProbabilisticPointSet*, these classes follow in the tradition of multi-valued logics and expert system certainty factors.

ProbabilisticTemporalInterval

ProbabilisticTemporalInterval extends the definition of an interval to include a function that, given a time, returns the probability that the interval includes that time. Thus, as shown in Figure 3-16, a probabilistic temporal interval can specify that the valid time most likely includes [10,25), is increasingly less likely to include times on the other sides of 10 and 25, and definitely does not include times outside of [5,30). This probability drop-off could indicate where the application was trying to determine state-change boundaries. The deterministic temporal interval is merely a special case where the probability is 1 during a specific interval and 0 elsewhere.

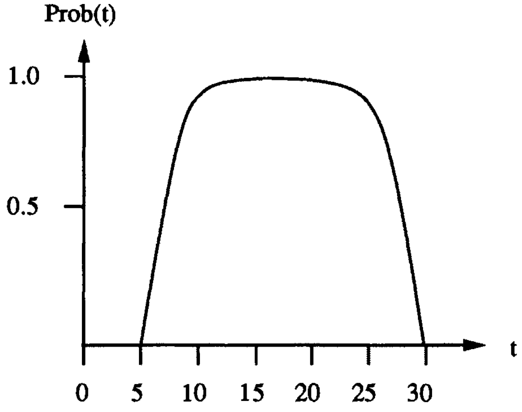


Figure 3-16: Probabilistic temporal interval

ProbabilisticTemporalInterval modifies standard *TemporalInterval* methods to use the temporal probability function. For example, a probabilistic temporal interval does not have clearly-defined endpoints; the method to find endpoints uses a threshold supplied by the application to separate points in the interval from those outside it.

ProbabilisticTemporalElement

ProbabilisticTemporalElement, a subclass of *TemporalElement*, contains a set of probabilistic temporal intervals rather than a set of temporal intervals. This allows a temporal object to represent the probability that it is valid during a time in a set of disjoint intervals.

The methods of *ProbabilisticTemporalElement* are specialized to handle temporal probability. For example, multi-valued logic systems often define probabilistic conjunction as a minimization operation and probabilistic disjunction as a maximization operation [14]. Figure 3-17 shows two overlapping temporal elements; Figure 3-18 demonstrates conjunction by minimization and Figure 3-19 shows disjunction by maximization.

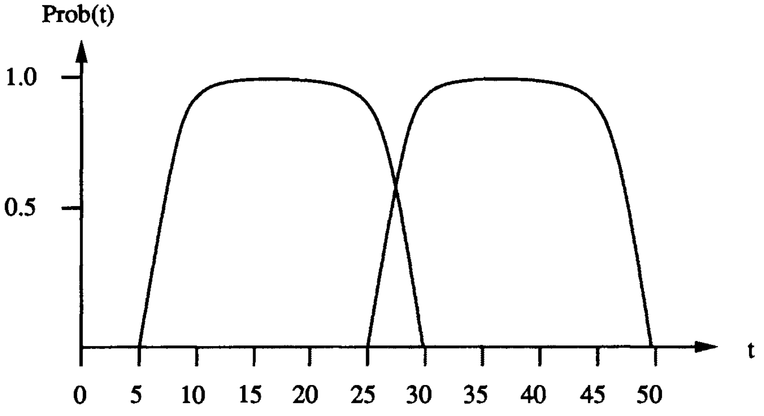


Figure 3-17: Overlapping probabilistic temporal intervals

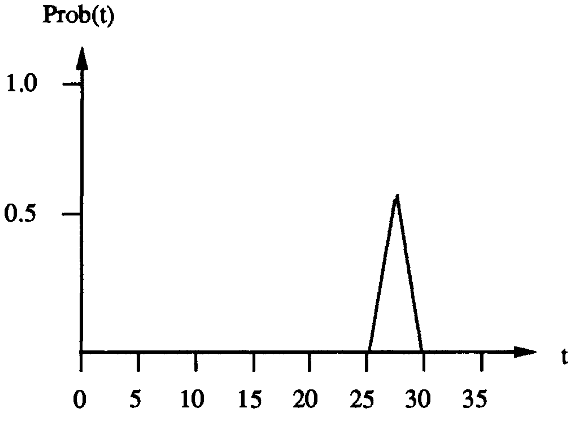


Figure 3-18: Probabilistic conjunction by minimization

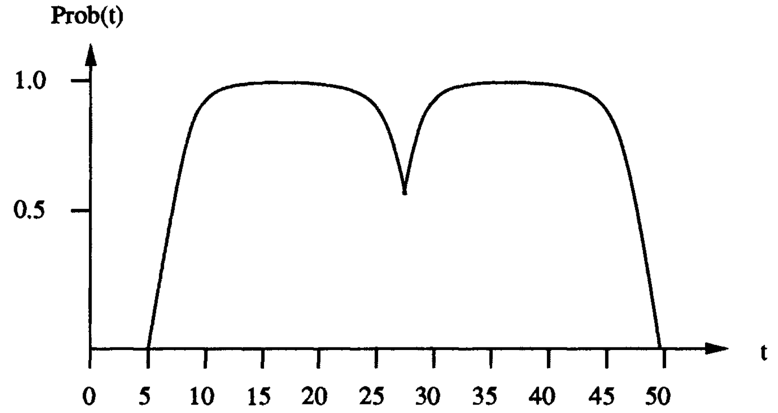


Figure 3-19: Probabilistic disjunction by maximization

3.5 Spatiotemporal Representations

Many objects stored in the visual memory have both spatial and temporal components. For example, a vehicle navigator might watch other vehicles driving nearby and a security system might track people walking in a hall. In both of these cases, objects are moving in space over an extent of time. The meshing of spatial and temporal information in these cases suggests that, in addition to spatial and temporal support, the visual memory should provide spatiotemporal support.

The class *SpatiotemporalObject*, a subclass of both *SpatialObject* and *TemporalObject*, represents spatiotemporal information in the visual memory. Because it is a subclass of both *SpatialObject* and *TemporalObject*, it contains the same information, including a point set, a coordinate system, a set of valid times, and a clock. It also supports all the *SpatialObject* and *TemporalObject* methods for manipulating this information.

The class *DiscreteSpatiotemporalObject*, a subclass of *SpatiotemporalObject*, stores state snapshots of objects. For example, a vehicle navigator could use an instance of this class to periodically store information indicating the spatial extent of the vehicle over some interval of time. In this way it could build up a whole history of the vehicle's motion.

DiscreteSpatiotemporalObject provides interpolation methods to estimate additional spatiotemporal information from existing information. For example, from the information in Figure 3-20, the visual memory could interpolate the snapshot of Figure 3-21. *DiscreteSpatiotemporalObject* subclasses implement a variety of interpolation procedures; for example, the circle in Figure 3-21 could be interpolated by radius or by area, and acceleration over several snapshots could be taken into account. Interpolation allows applications to store spatiotemporal information more sparsely and still closely approximate necessary information.

Like *SpatialObject* and *TemporalObject*, *SpatiotemporalObject* also provides an abstract subclass to represent information by means of a function. *AbstractSpatiotemporalObject* uses a trajectory method to determine which points are in its point set

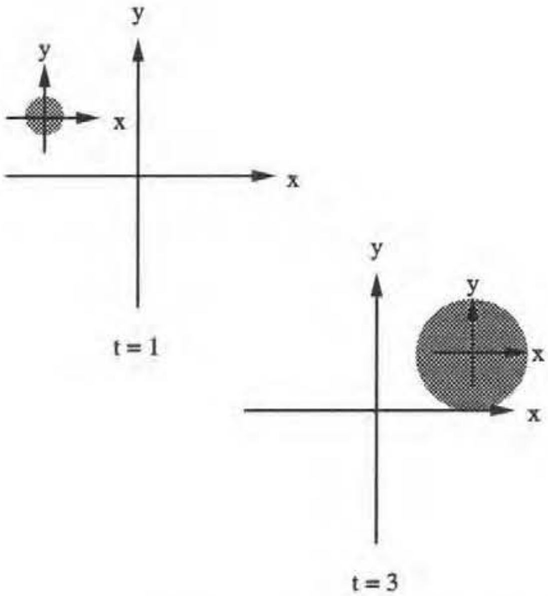


Figure 3-20: Discrete spatiotemporal information

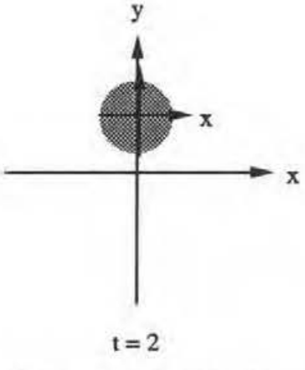


Figure 3-21: Interpolated spatiotemporal state

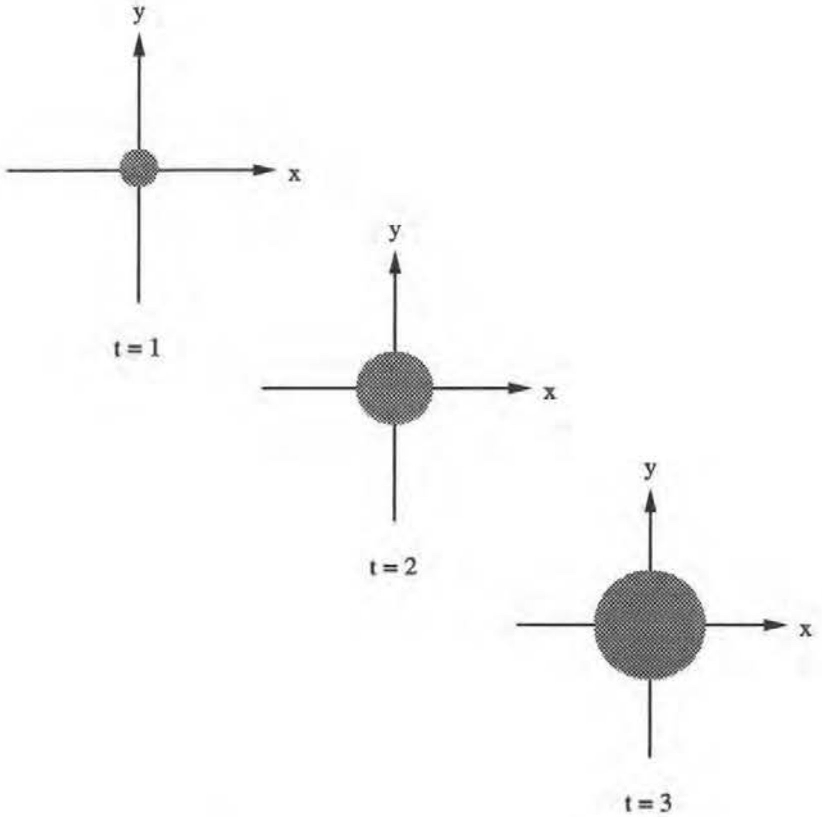


Figure 3-22: Point set trajectory

at specified times. The trajectory specifies how the object's point set and coordinate system change with time. Thus an abstract spatiotemporal object is equivalent to a set of discrete spatiotemporal state snapshots.

The spatial description of an object can change over time in two different ways: the point set itself can change, or the point set's relation with other points can change. The circle with an expanding radius shown in Figure 3-22 is an example of a changing point set, while the translating circle shown in Figure 3-23 demonstrates changing relationships. A trajectory for an abstract spatiotemporal object can handle either or both types of change.

The trajectory can modify a point set over time by supplying a time point as an additional argument to the point set function. For example, the trajectory for the

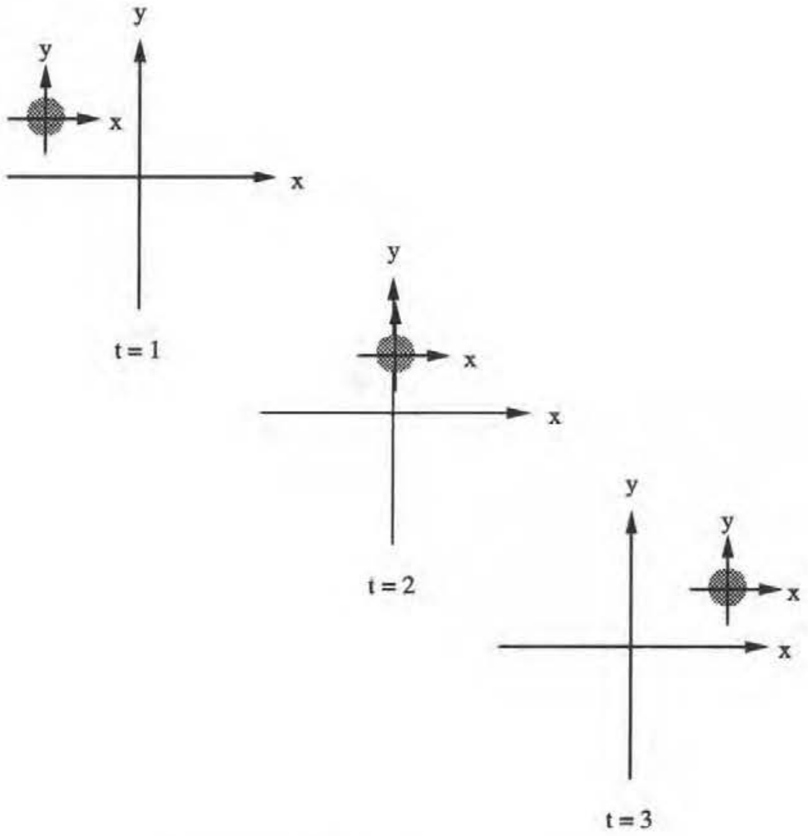


Figure 3-23: Coordinate system trajectory

circle with expanding radius in Figure 3-22 could check $x^2 + y^2 <= t$ to determine all points (x, y) in the point set at time t. If the changes in the point set follow some pattern, the spatiotemporal point set function can capture that pattern; otherwise, the discrete approach is probably more suitable.

The trajectory can change relationships between point sets over time by establishing a function to specify coordinate system transforms as a function of time. In Figure 3-23, the trajectory would translate the coordinate system one unit along the x-axis every second. This can be implemented by establishing an initial coordinate system and its relationships to other coordinate systems and then identifying differences between the coordinate system at a given time and the initial coordinate system. This way the trajectory does not have to establish all the coordinate system's relations at each time; instead, it can transform from a given coordinate system to the established coordinate system and from it to any other related coordinate system.

Visual memory provides the class *RelativeSpatiotemporalObject* to express spatiotemporal relationships. For example, an application could describe a relative spatiotemporal object as being to the right of another object sometime after 6:00. *RelativeSpatiotemporalObject* and its subclasses simply combine the relative spatial and temporal classes detailed in earlier sections.

Spatiotemporal representations can benefit from probabilistic methods. The visual memory class *ProbabilisticSpatiotemporalObject* combines the spatial and temporal probabilistic methods previously described. It allows applications to express uncertainty about both the spatial and temporal extents of spatiotemporal objects. Probabilistic functions are especially useful with spatiotemporal interpolation, allowing a measure of uncertainty to accompany an interpolated object description. Abstract spatiotemporal objects can establish probabilistic trajectories to be imprecise about the changes in an object's spatial description over time.

3.6 Object Storage

An important part of the visual memory design addresses how to store and retrieve spatiotemporal information. The object-oriented database on which the visual memory builds provides basic support for object storage and retrieval. This section discusses the concepts and issues most relevant to the visual memory design.

3.6.1 Identity

Each object in the visual memory has a unique identity. This identity does not necessarily correspond to physical identity; for example, an application might not recognize a person appearing in its view as the same person who disappeared moments ago, causing it to create a new object for the person. To preserve identity, the visual memory assigns each object an object identitifier (OID), a number that distinguishes that object from all others. The object maintains the same OID through all of its state changes.

Each object can have multiple versions. For example, a security system could track a person walking down a hall and store a new version describing that person's location every tenth of a second. The versions of an object maintain the same OID, but each has a different version number. Thus an <OID, version number> pair uniquely distinguishes a particular state snapshot of a particular object. By maintaining all of an object's versions, the visual memory can answer questions about the object's history.

Some visual memory applications might need to combine the histories of different objects to form the history of one object. This could happen, for example, if a tracking system lost sight of a person, found a new person and created a new object, and later realized that the two people were actually the same. The visual memory can consolidate object histories to create versions of one object from versions of other objects.

An application can report to the visual memory that an object has disappeared. Making an object disappear is quite different from deleting that object, which actually removes old versions of the object from the visual memory. Disappearing does not affect an object's history but instead removes it from the current state of the visual memory. Visual memory queries after an object's time of disappearance do not retrieve that object.

3.6.2 Storage Mechanism

The database underlying the visual memory decides how to store objects. To implement an appropriate storage policy, the database should consider the visual memory's storage needs and the characteristics of the objects that it stores. This section discusses how object storage should be tailored for the visual memory.

Many visual memory objects change very little from one version to the next. For example, a rigid object moving across the room changes only its coordinate system and valid time; the point set, clock, and other information remains the same. In cases like this, the database should store one base version of the object and then indicate differences for each new version.

The visual memory obeys a nondeletion policy: it creates a new version each time an object changes, and it never deletes old versions. Deleting a version would cause problems for other object versions containing references to it. The visual memory is not an append-only database since it actually modifies old versions, as discussed below in section 3.6.3. Only the visual memory can modify old versions, since uncontrolled modification could lead to inconsistencies. These considerations allow the database to implement a simpler storage policy.

Some visual memory applications store a great amount of data. Since old information might never be deleted, the available space can quickly fill. Once old information has settled down and will not be accessed or modified often, the database can move it onto long-term, high-capacity storage devices. This keeps the most useful information readily available while increasing the amount of information that can be stored.

3.6.3 Time

As a historical database, the visual memory keeps track of when events happened. It stores with each version of a temporal object information about that version's valid time. Since one version's valid time might conflict with the valid times of other versions, the visual memory attempts to ensure consistency by resolving these valid times. Section 3.4.1 discusses temporal resolution strategies.

The valid time of a new version could conflict with the valid times of many old versions. The indexing strategies discussed below in Section 3.8 allow the visual memory to quickly identify which old versions must be changed. The necessity of resolving old temporal information encourages the use of caching techniques to reduce the number of disk accesses.

Applications can improve the performance of temporal resolution by operating in "real-time mode." In real-time mode, the valid time of the latest version of an object is an infinite interval starting from the current time. Thus each new version must be resolved only with the previous version. For example, if the first version were valid $[0, \infty)$, then a second version valid $[5, \infty)$ would change the first version's valid time to $[0, 5)$, a third version valid $[10, \infty)$ would change the second version's valid time to $[5, 10)$, and so forth. Performing only one temporal resolution per object update can greatly improve storage performance.

3.7 Queries

The visual memory provides a powerful and expressive mechanism for retrieving information. This query mechanism is tailored to the spatial and temporal representations presented in earlier sections. It is also designed to meet a wide variety of retrieval needs, providing flexibility in specifying objects of interest. This section describes the query mechanism and the types of queries supported by the visual memory.

3.7.1 Query Mechanism

The visual memory query mechanism extends a standard SQL-based [1] object query language, such as OQL [2]. The queries below demonstrate the basic form and functionality of such a query language.

Find everyone with the same age as the object stored in program variable "me":

```
Select p from Person
  where p.age() == %me.age()
```

Find everyone named Larry who used to play professional basketball:

```
Select p from Person
  where p.firstname() == "Larry" and
    p.occupation().title() == "pro basketball player" and
    p.occupation().status() == "retired"
```

Find the children of the above people:

```
Select p from Person
  where p.father() in
    (Select p from Person
        where p.firstname() == "Larry" and
            p.occupation().title() == "pro basketball player" and
            p.occupation().status() == "retired")
```

The database literature contains many examples demonstrating the power of query languages. The visual memory query language extensions allow applications to construct complex spatiotemporal queries.

A query language provides flexibility and expressiveness but can be hard to use. For applications that do not need the full power of a query language, graphical query specification might be more suitable. A graphical query could be specified by outlining regions of space and intervals of time; objects satisfying the specification could also be displayed graphically. A graphical query language could be built over the visual memory query language by transforming graphical specifications into visual memory queries. Chapter 4 discusses an implementation of such a graphical query language.

A query mechanism works on two levels, on disk and in memory. The visual memory indices, discussed further in Section 3.8, provide information to help the query mechanism eliminate objects that do not satisfy a query before bringing them into memory. The query mechanism then further filters these objects to determine which objects satisfy the specification. A number of the query constructs outlined below could easily be performed in memory but are implemented as part of the query language to allow the query language to optimize object retrieval.

Rather than adding a large number of special spatial and temporal constructs to the query language, the visual memory bases its query support on instances of the spatial and temporal classes discussed in previous sections. Each query includes spatial or temporal keywords and a spatial or temporal object; the keyword describes how instances satisfying the query must interact with the given object. The specified spatial or temporal object could be a program variable, allowing the application to form a complex specification before posing the query. Alternatively, it could be the result of another query, allowing an application to compose queries. These mechanisms provide great flexibility in spatial and temporal query specification.

3.7.2 Spatial Queries

Instances of the class *SpatialObject* form the basis for all spatial queries. A query specifies a spatial object of interest and how objects satisfying the query must interact with that spatial object. Described below are the ways that applications can use a

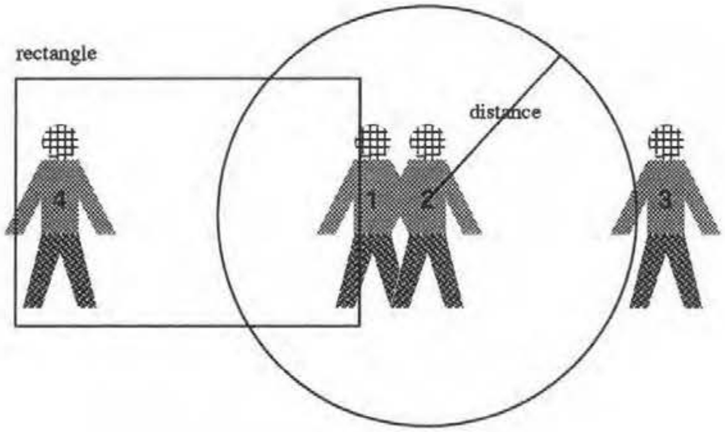


Figure 3-24: Spatial queries

specified spatial object to retrieve objects of interest. The accompanying examples demonstrate the query language syntax and reference objects of Figure 3-24.

Intersects Query

The *intersects* query looks for the intersection of spatial objects. For example, the following query returns the set { person-1, person-4 }:

```
Select p from Person
  where p intersects %rectangle
```

This is one of the most broadly useful spatial query constructs. The specified spatial object can be a point, line, pentagon, pyramid, or just about any other spatial specification imaginable. This construct is also useful when negated. For example, the set { person-2, person-3 } satisfies the following query:

```
Select p from Person
  where not p intersects %rectangle
```

A security system could use intersection to find all the objects within a room, and a vehicle navigator could use negated intersection to make sure that nothing was on the road in front of the vehicle.

Borders Query

The *borders* query checks for bordering objects. The set { person-4.torso() } satisfies the following query:

```
Select p from Person
  where p borders %person-4.head()
```

A VLSI system could use this query construct to look for electrical contact, and a photo interpretation system could use it in constructing a high-level representation of connected regions. Applications can use probabilistic point sets to specify imprecise borders for this query.

Centroid-Within Query

The *centroid-within* query ignores the spatial extent of objects and checks distances between centroids. For example, the following query returns the set { person-1 person-2 }:

```
Select p from Person
  where p centroid within %distance of %person-2
```

This distance parameter specifies within how many units, using the specified spatial object's coordinate system, an object must be to satisfy the query. With this query, applications can quickly gather objects roughly within a given distance from a specified object. The estimation is fairly accurate if the point sets are much smaller than the distance between them.

Point Set-Within Query

To select nearby objects with greater accuracy than the *centroid-within* query provides, applications can use the *point set-within* query. The following example selects the set { person-1, person-2, person-3 }:

```
Select p from Person
  where p point set within %distance of %person-2
```

This query is similar to the *centroid-within* query, but it retrieves all objects that have at least one point within the given distance of any point of the specified spatial object. To select objects meeting some specialized definition of nearness, an application can construct any spatial object and perform an *intersects* query; this is just a specialized, optimized version of that process.

Transitive-Closure Query

The *transitive-closure* query compounds any of the above specifications, applying a query to its results until there are no new results. It returns all objects identified in the process. For example, the transitive closure of a *borders* query shown below returns the set { person-4.torso(), person-4.legs() }:

```
Select p from Person
  where p borders by transitive closure %person-4.head()
```

This query retrieves any objects bordering the given object, any object bordering those objects, and so forth. A photo interpretation system could use it to find connected regions.

3.7.3 Temporal Queries

The visual memory temporal query mechanism retrieves all the versions of objects that satisfy some set of constraints. A temporal query specifies a *TemporalObject* instance to describe the times of interest and a keyword to describe how the valid time of a satisfying version must interact with those times. Described below are the visual memory temporal query specifications. Accompanying examples demonstrate the query language syntax using versions shown in Figure 3-25.

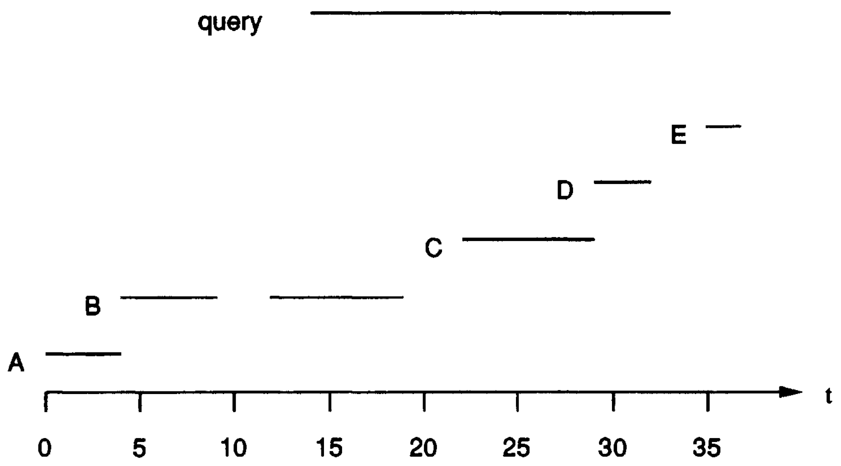


Figure 3-25: Temporal queries

During Query

The *during* query checks for versions whose valid times intersect the given valid time. For example, the following query returns the set { Version-B, Version-C, Version-D }:

```
Select p from Person during %query
```

This is a very powerful query, allowing applications to retrieve versions during any specified set of times. It is also useful in its negated form, where it returns versions whose valid times do not intersect the given valid time. The negated query below selects the set { Version-A, Version-E }:

```
Select p from Person
  not during %query
```

Latest-During Query

The *latest-during* query retrieves only the latest version of an object during some specified temporal element. For example, the set { Version-D } satisfies the following query:

```
Select p from Person
  latest during %query
```

An application could use this query to update a memory-resident model with the latest information in the visual memory. For example, a vehicle navigator could establish a model of static objects at the beginning of its execution and then use this query to update that model with the latest dynamic information stored by image processing software.

3.7.4 Spatiotemporal Queries

In addition to spatial and temporal queries, the visual memory supports spatiotemporal queries. Some of this support comes from the query language's natural ability to handle combined specifications. For example an application could pose the following query:

```
Select p from Person
  where p intersects %square
  during %times
```

This query retrieves all versions of all objects valid during the specified times and intersecting the specified square. Figure 3-26 depicts five states of a spatial object, at time $t = 1$ through $t = 5$. Figure 3-27 depicts a square valid over [1,5) and shows that the above query would return the third state of the object.

The joint spatial and temporal query checks a static spatial object over time, so it does not handle interactions between spatial and temporal information. Some applications want to track a moving object and retrieve versions near it at various times. To handle cases like this, the visual memory provides spatiotemporal queries.

A spatiotemporal query specifies a spatiotemporal object and a temporal object, and how objects must interact with these to satisfy the query. The spatiotemporal object's history describes where an object must be at given times, and the temporal object specifies a portion of the history of the spatiotemporal object. The query can use any of the spatial constructs discussed above to specify spatiotemporal interac-

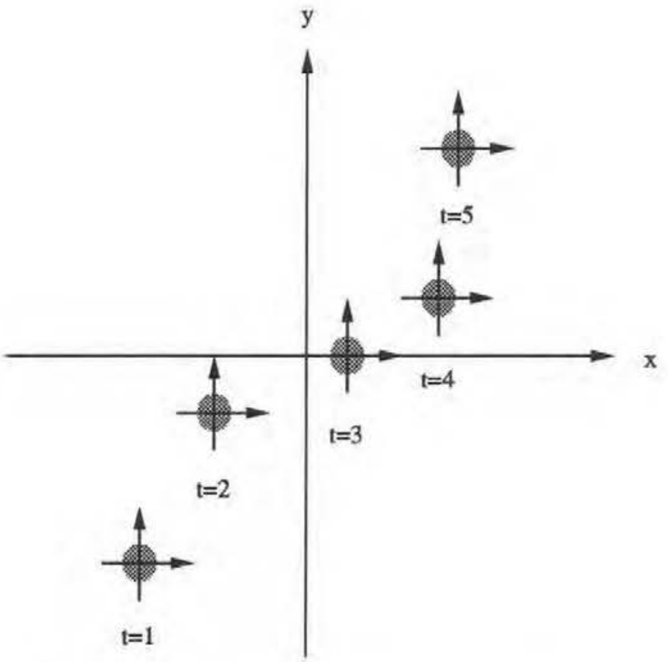


Figure 3-26: States of a spatiotemporal object

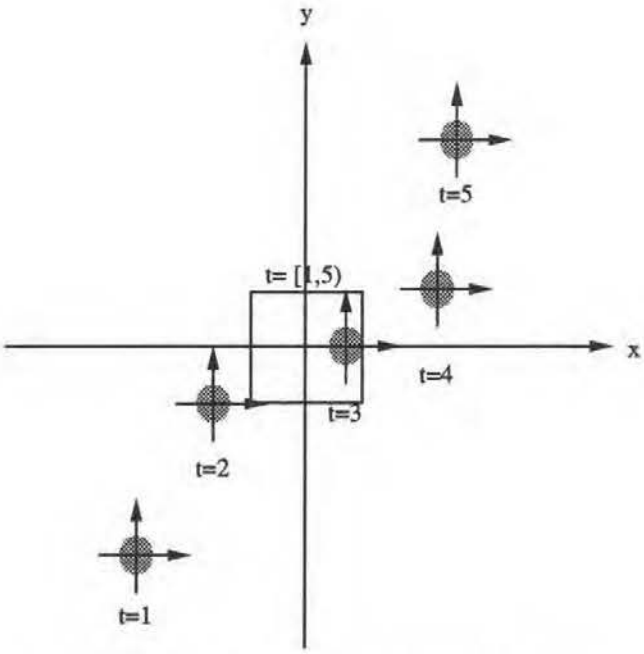


Figure 3-27: Joint spatial and temporal queries

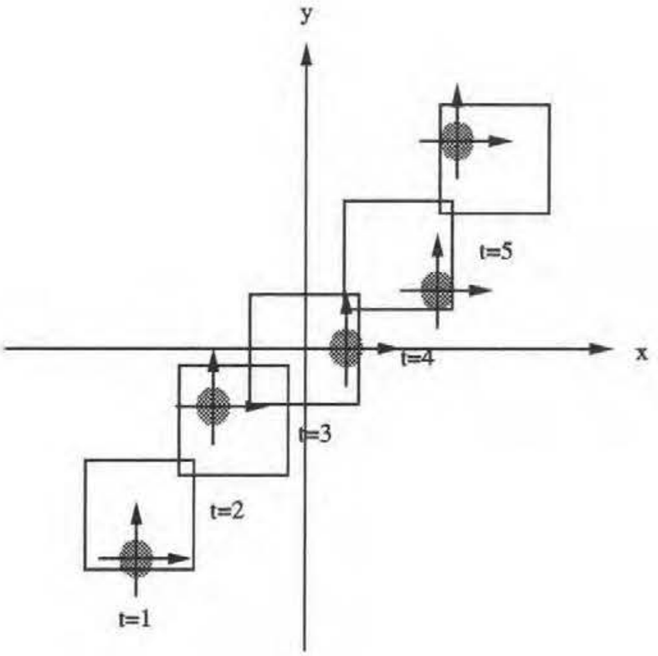


Figure 3-28: Spatiotemporal queries

tions. For example, consider the following query:

```
Select p from Person
  where p intersects %square
  during %times
```

This query construct retrieves versions of objects that intersect the square in its trajectory over a set of times. The query shown in Figure 3-28 uses as the spatiotemporal query object a square translating equally in the x and y dimensions over time. This query returns all five states of the object of Figure 3-26.

This powerful query construct can handle many complex queries, especially when combined with the join capability of the query language. For example,

```
Select p from Person
  during %times-1
  where p in
     (Select q from Person
       where q centroid within 3 of %spatiotemporal-spec
       during %times-2)
```

This query tracks all objects that came within 3 units of a given object on its trajectory during a certain set of valid times. Queries like this demonstrate the power of a query language extended with the visual memory spatiotemporal constructs.

3.8 Indices

The visual memory provides an indexing mechanism to quickly identify objects meeting sets of constraints. Indices tie in with both the query mechanism and the various spatial, temporal, and spatiotemporal operations described in preceding sections. For example, a spatial index can help identify solutions to an intersection query retrieving objects stored in the visual memory, and it can help identify intersecting memory-resident objects. The two types of indexing work similarly, so for conciseness this section primarily considers how indices can improve retrieval performance.

Indices maintain information allowing them to quickly eliminate objects that do not satisfy a query. They provide *conservative* approximate answers to queries; that is, they can mistakenly retrieve objects that do not satisfy a query, but they can never mistakenly leave out objects that do satisfy a query. The design of an index must trade off between how quickly the index can answer a query and how much overhead is necessary to maintain the indexing information. A well-designed index can greatly help query performance while adding minimal information overhead.

3.8.1 Mechanism

Visual memory indices are object-oriented: they are objects and they maintain information about objects. This yields a consistent approach to information representation. The database can store and retrieve indices just like other objects. Indices can keep track of other indices, a technique further discussed below. Finally, due to the extensible nature of the object-oriented approach, it provides flexibility in designing indices.

The purpose of an index is to maintain information to help it efficiently identify objects that might satisfy a query. In the visual memory design, this information consists of <OID, version number> records, each uniquely specifying a particular version of a particular object. An index structures these records so that it can quickly provide a set of records identifying the objects that meet specific constraints.

Indices maintain information in many different ways, such as tables, arrays, and trees. The visual memory can handle a very large index by retrieving only a necessary, manageable part at a time. However, an index must strive to minimize the amount of retrieval required to reach an answer, so that the the cost of using the index does not outweigh the query efficiency it yields.

An application can specify which sets of objects it wants to index and how it wants to index them. It simply specifies the class of the index desired and the set of objects for which it should maintain information. For example, consider the following examples of index specification:

```
Index temporal-btree on
  (Select p from Person)

Index spatial-grid on
  %my-set

Index spatial-quadtree on
  (Select o from Object
     where o intersects %my-room)
```

The first example establishes a temporal index for all people; the second establishes a spatial index on a specific set specified by a program variable; the third indexes all the objects in a certain scene. The visual memory maintains a list of all the indices in use and knows when to update them and for which queries they are appropriate.

The following sections present issues in the design of spatial, temporal, and spatiotemporal indices. Chapter 4 discusses additional indexing issues raised by one visual memory application and describes indices designed for the application.

3.8.2 Spatial Indices

Spatial indices organize information about the objects in a scene. The literature describes many different spatial indices; see [18] for descriptions of quite a few. Different spatial indices use different parts of an object's spatial representation and thus are most appropriate for different queries. For example, a point quadtree uses an object's centroid and works best with proximity queries, while an interval tree uses spatial intervals and is most suitable for intersection queries. An application must pick spatial indices applicable to its retrieval needs.

3.8.3 Temporal Indices

Temporal indices store information about object histories. The task of ordering the temporal component of an object is similar to that of ordering the spatial component, if time is viewed as just another dimension. Thus a lot of spatial indexing research applies to temporal indexing as well. For example, a spatial interval tree could store lists of versions valid during temporal intervals. However, temporal representation poses some concerns unique to temporal indexing.

Temporal indices must address the monotonicity of time. The visual memory allows applications to modify the past or predict the future, but some applications maintain an always-increasing sense of time. This could hurt the performance of some temporal indices; for example, a tree could become unbalanced. Temporal indices still need to support nonmonotonic temporal specification, but some could be optimized for the monotonic case.

Because a temporal index retains historic information, it constantly increases in size throughout its lifetime. A temporal index must not lose too much efficiency as it grows. Some temporal indices should even partition their data between short- and long-term storage, as in [9].

Temporal indices must be able to represent infinite temporal intervals. An infinite interval occurs, for example, when an application assumes that an object will be valid until otherwise notified and assigns the object a valid time extending to infinity. An infinite interval would cause problems for a temporal index representing intervals as collections of subintervals in a tree.

3.8.4 Spatiotemporal Indices

Spatiotemporal indices store spatial information about a scene as it varies in time. The interaction of space and time makes spatiotemporal indexing a complex problem.

There are two kinds of spatiotemporal indexing, corresponding to the discrete and abstract spatiotemporal classes discussed in Section 3.5.

The first type of spatiotemporal indexing stores information about versions of discrete spatiotemporal objects. The indexing is a two-step process: spatial indices maintain spatial descriptions of objects, and temporal indices maintain the temporal descriptions of the spatial indices. To perform a spatiotemporal query, the indexing mechanism finds the temporal description in the temporal indices, retrieves the corresponding versions of the spatial indices, finds the spatial description in them, and retrieves the corresponding spatiotemporal object versions.

Discrete spatiotemporal indexing must address some concerns. Spatiotemporal objects that move continuously cause constant index updates. This leads to large temporal indices, raising the issues previously discussed. The structure of a spatial index used in spatiotemporal indexing should not depend on the objects contained within it, since those objects move.

The second type of spatiotemporal indexing stores information about abstract spatiotemporal objects. An abstract spatiotemporal index could build up its own spatiotemporal function representing a set of object trajectories. Given a spatiotemporal specification, this function would return a list of those objects satisfying it. This function could grow very complex, so the index would have to devise some means of efficiently storing, retrieving, and evaluating it. In this manner an index could efficiently answer queries about abstract spatiotemporal objects.

Chapter 4

Implementation

To test the visual memory design, a subset of it was implemented in support of a real-time scene monitoring prototype. In this prototype, image processing using video cameras tracks objects and stores information about them in the visual memory. Through a graphical query interface, users can specify queries to the visual memory and view the results in various ways. Figure 4-1 shows the basic flow of information in the prototype. This chapter describes the implementation of the scene monitoring prototype and the visual memory supporting it.

Scene monitoring is a good testbed for the visual memory. Its constant updates and retrievals of information test the visual memory's performance. Multiple sensors and outputs test concurrency issues. The query interface tests the power of the query language by specifying a variety of queries, including spatial ("Watch for anything that comes within 3 feet of that button."), temporal ("Play back the last 10 seconds."), and spatiotemporal ("Did anybody come into the room between 12:00 and 1:00?"). Finally, the construction of such a prototype tests the usefulness of the visual memory spatiotemporal representations.

4.1 Database

An object-oriented database called Persistent C++, or PC++ for short [17], is the basis for the visual memory prototype. This database is a prototype for the DARPA

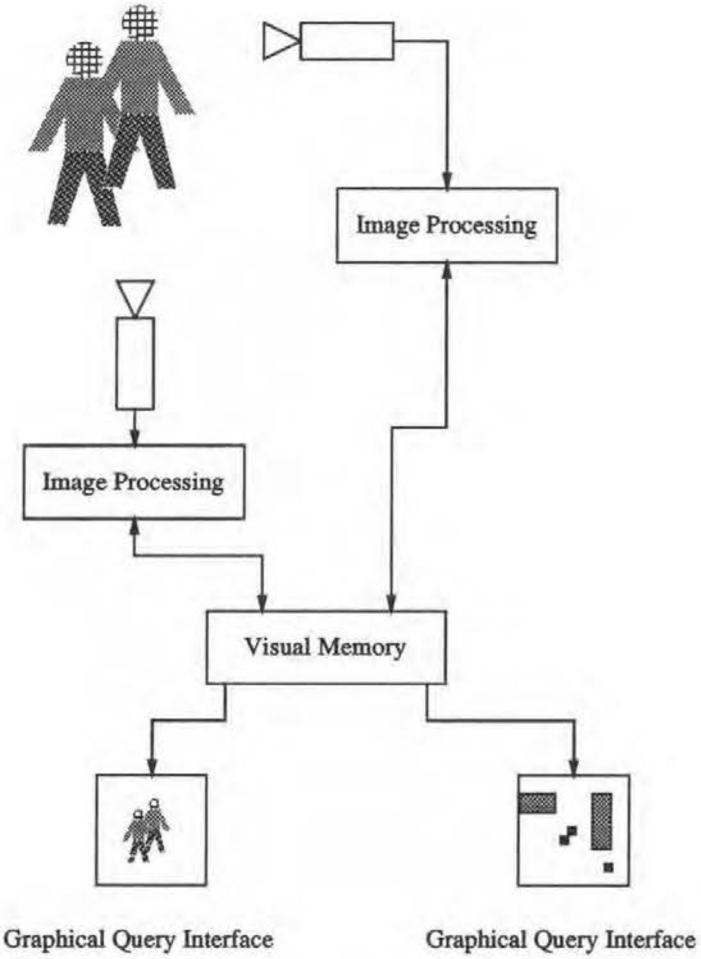


Figure 4-1: Scene monitoring prototype

Open Object-Oriented Database project at Texas Instruments [25]. PC++ has an open architecture, allowing the visual memory to add spatiotemporal extensions and take advantage of the features provided by other modules.

Some of the features provided by Persistent C++ are particularly useful to the visual memory. A versioning mechanism allows access to any previous state of any object. Transactions ensure atomicity, consistency, isolation, and durability. The object storage mechanism caches recently accessed information to increase performance.

A Persistent C++ preprocessor gathers information about the classes of objects to be stored in the database. This particular prototype preprocessor is somewhat limited, not allowing multiple inheritance or function pointers; these constraints limit the prototype in some situations. The preprocessor adds extra information to the class descriptions and forms actual C++ classes for an application to use. It adds function hooks into these classes so that the application can establish daemons to be executed when objects are stored or retrieved. Finally, when one object contains a pointer to another object, its class specification indicates either that the referenced object should be automatically retrieved with the referring object or that it should be retrieved only on demand.

Persistent C++ stores objects with the Exodus storage manager [4]. It stores a whole Exodus object for each version of a PC++ object, rather than storing differences between versions. This could hurt performance for objects that change very little from one version to the next. PC++ maintains a B-tree structure to map its OIDs to Exodus OIDs; this hurts performance as the number of OIDs grows large.

Persistent C++ can retrieve an object specified by OID and version or by a character string previously assigned to that object. It provides an object query language extension, OQL [2], but this query language does not interface well with the visual memory indexing mechanism. Thus the visual memory prototype has its own spatiotemporal query mechanism.

4.2 Spatiotemporal Representations

The prototype visual memory implements as Persistent C++ classes a number of the spatial, temporal, and spatiotemporal representations discussed in Chapter 3. These representations conform to the design except for some differences due to limitations in Persistent C++ and some optimizations and simplifications tailored to the scene monitoring application.

The prototype implements only the basic discrete classes. Since Persistent C++ cannot store functions, an instance cannot construct an arbitrary abstract function for its point set, temporal element, or trajectory function. In addition, the scene monitoring prototype does not need relative or probabilistic specifications.

To increase performance, the prototype uses a global coordinate system and a global clock. This eliminates the need for spatial transforms between coordinate systems and temporal transforms between clocks. Translation and rotation methods act on objects themselves rather than on their coordinate systems.

The prototype implements specific subclasses of the class *SpatiotemporalObject* to represent the objects tracked by the scene monitoring system. For example, the *Person* class adds a slot for estimations of the person's height; it could also store the person's name and other such information if it were connected to face recognition software.

4.3 Indices

4.3.1 Mechanism

Index updates occur in the visual memory prototype at transaction commit time, through Persistent C++ commit daemons. When the database stores an object, it automatically calls the object's commit daemons. The visual memory establishes commit daemons for all objects to update index information.

The visual memory prototype implements the discrete spatiotemporal indexing described in Section 3.8.4. Spatial indices store information about object locations, and temporal indices store information about the valid times of these spatial indices.

The visual memory prototype handles multiple indices. An application can create sets of indices and specify the types of information they should store and the types of queries they should answer. However, the prototype only implements start and stop control over indices; that is, an application can tell an index to start recording information about all objects committed, or to stop recording such information. This is a simpler approach than the specification of arbitrary index sets discussed in the design, but it is adequate for the prototype application.

4.3.2 Spatial Indices

The prototype spatial indices store information about the centroids of objects stored in the visual memory. This information allows them to efficiently answer locational and proximity queries, such as "Find everything in this square" and "Find everything within 5 units of this coordinate." Two such indices were implemented; this section describes the two-dimensional version of each.

The first spatial index is a simple fixed grid [18], dividing space into a number of cells. Each cell stores a list indicating those objects with centroids in the cell. The index can determine the correct cell for an object by rounding down the coordinates of the object's centroid, modulo the cell size. Figure 4-2 shows a fixed grid with a cell size of 5. Using the scheme described above, object G at spatial coordinate (14,18) belongs to cell (2,3).

To answer a spatial query, the grid determines relevant cells in the manner described above and retrieves the objects they list. A query for objects within the shaded rectangle in Figure 4-2 searches cells (2,3), (2,4), (3,3), (3,4), (4,3), and (4,4), and returns objects C, F, and G. The fixed grid index is most suitable for visual memory applications with unknown distributions of object positions and frequent needs for efficient updates.

The second spatial index implemented in the prototype is a bucket PR quadtree [18]. Each node in the tree keeps a bucket of object records for some region. The index initially consists of one node covering the entire indexed region and containing

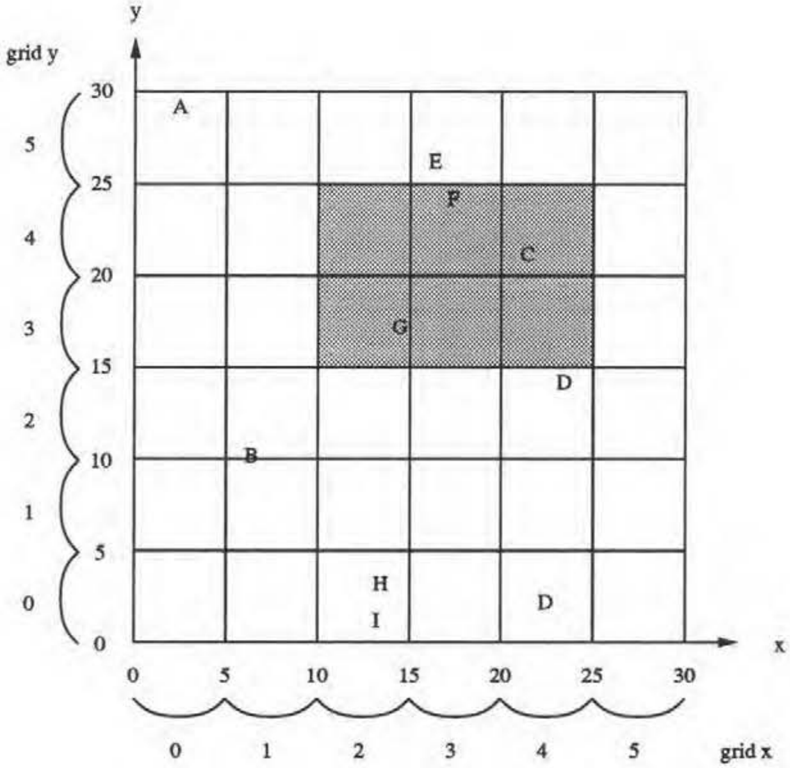


Figure 4-2: Fixed grid spatial index

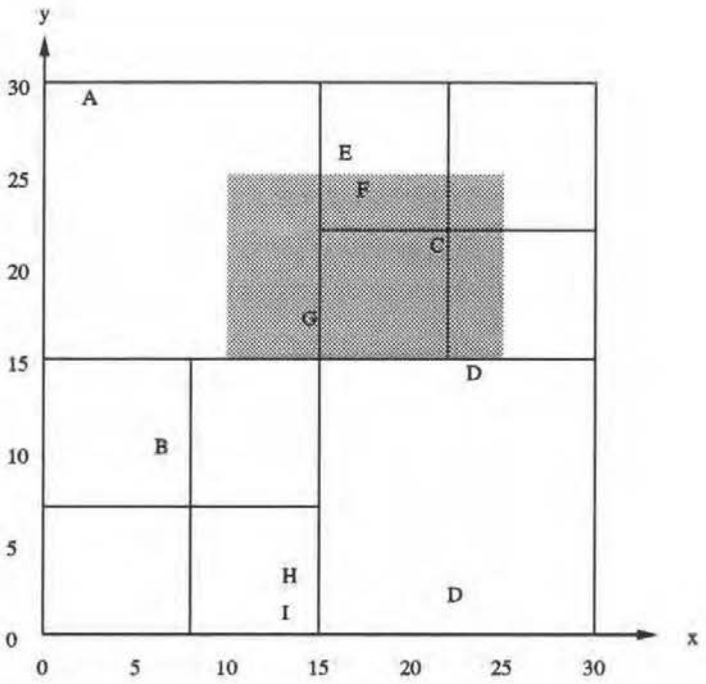


Figure 4-3: Segmented space for bucket PR quadtree

no records. As objects are added to a node's bucket, it eventually becomes full and the node must be split. A node is split into four children, one for each quadrant, and the node's bucket is appropriately divided among the children; full children are recursively split. Figure 4-3 shows how space would be segmented for a quadtree with bucket size of 2 and the given objects. Figure 4-4 shows the corresponding index structure.

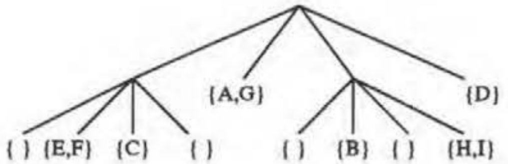


Figure 4-4: Data structure for bucket PR quadtree

A bucket PR quadtree answers a spatial query with a recursive search through all nodes intersecting the region of interest. A query for objects in the shaded rectangle in Figure 4-3 searches the left half of the tree in Figure 4-4, and it returns objects C, F, and G.

The bucket PR quadtree index is best suited for visual memory applications where objects are spread out and do not move often. In these cases, it has much less overhead than the fixed grid. Thus an application might use a quadtree to store static background information and a grid to store dynamic information.

4.3.3 Temporal Indices

The prototype temporal indices keep track of the valid times of object versions. They can efficiently answer temporal intersection queries, such as "Find all events that happened after work last Tuesday and Wednesday." The prototype implements two different temporal indices.

The first temporal index is a segment tree [18]. Each node in the tree represents a temporal interval and contains a list of all versions valid throughout the entire interval. The children of a node represent subintervals of their parent's interval, so that a version that is not valid throughout a node's interval can be stored in one of its descendants. For example, if version A were valid from time 35 to time 140, it would appear at the indicated nodes in Figure 4-5.

To answer a temporal intersection query, the temporal segment tree retrieves the versions referenced by all nodes with intervals intersecting the specified temporal element. To find all versions valid during [105, 118) in Figure 4-5, the index searches the darkened branches and returns versions A and E.

The second temporal index is a B+ tree [6] with times as its keys. Each leaf node maintains a start-list containing versions that become valid at the node's key time and a stop-list containing versions that stop being valid at that time. The keys in an internal node separate its children. Leaves are connected in a linked list, and the start-list for the first leaf of an internal node also indicates "carry-over" versions still valid after the last key in the previous node. In Figure 4-6, version A, valid from

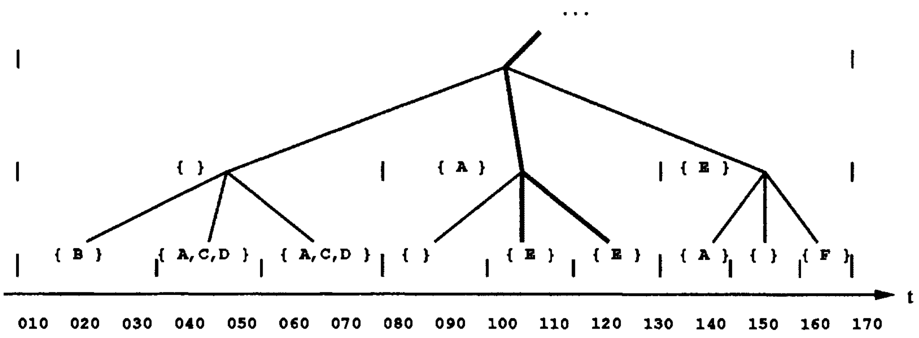


Figure 4-5: Temporal segment tree

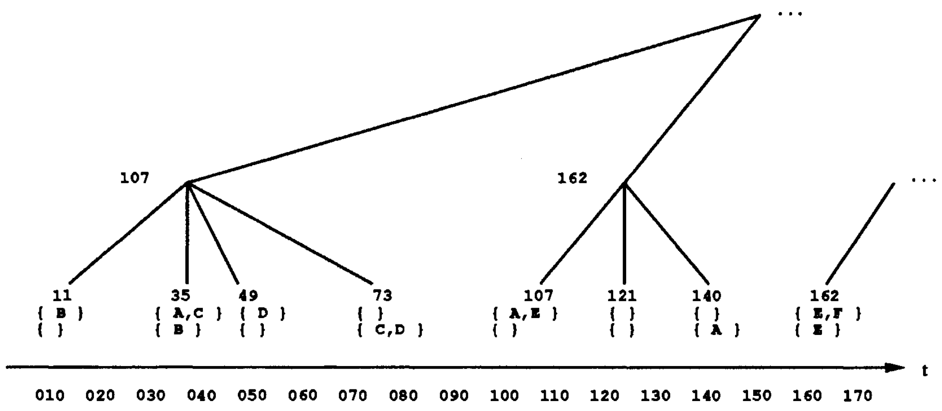


Figure 4-6: Temporal B+ tree

time 35 to time 140, has a start record at node 35, a stop record at node 140, and a carry-over record at node 107.

A temporal intersection query proceeds down the tree to the first leaf of an internal node with a time less than the earliest specified time. There it gathers the carry-over records and traverses the linked list to the earliest specified time to determine which carry-over versions are still valid then. Next it continues through the list to the latest specified time, noting which versions become valid during the temporal element. In Figure 4-6, a query for the interval [105,118) would go down to leaf node 11 and traverse the linked list to leaf node 107, noting that only version A was still valid at time 105. It would then proceed to leaf node 121 to find the remaining valid versions, finally returning versions A and E.

4.4 Queries

The prototype visual memory implements a functional query interface rather than a full query language. To pose a query, an application calls a visual memory function, passing it parameters specifying the query. For example, a spatial proximity query's parameters are a point and a radius, while a temporal intersection query takes a temporal element. The visual memory returns a set of <OID, version number> index records indicating objects that might satisfy the query. This set can be combined with other such sets to construct complex queries. Once a query has been fully specified, the query mechanism can retrieve the indicated objects. The indices provide only approximate answers, so the query mechanism filters the retrieved objects to return only those objects satisfying the specification. This query mechanism allows applications to pose fairly complex queries.

4.5 Input

The input for the scene monitoring prototype comes from real-time processing of CCD camera images. This software, which tracks people walking in its field of view, was implemented by Tom Bannon and Tom O'Donnell in the Image Understanding Branch at the Texas Instruments Computer Science Laboratory. Using a calibrated internal model of its field of view, the software estimates the positions and heights of people and updates the visual memory a few times per second. This yields enough information to test the visual memory's performance and to provide interesting data for queries to retrieve.

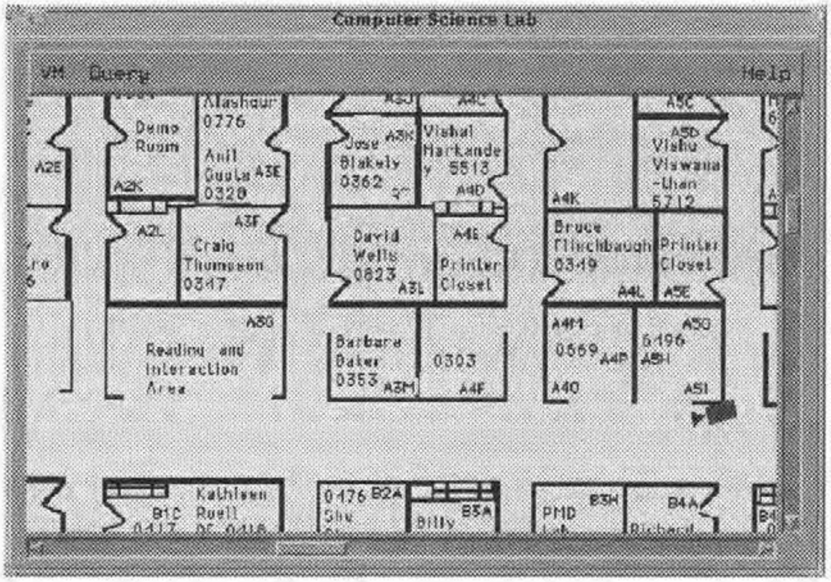


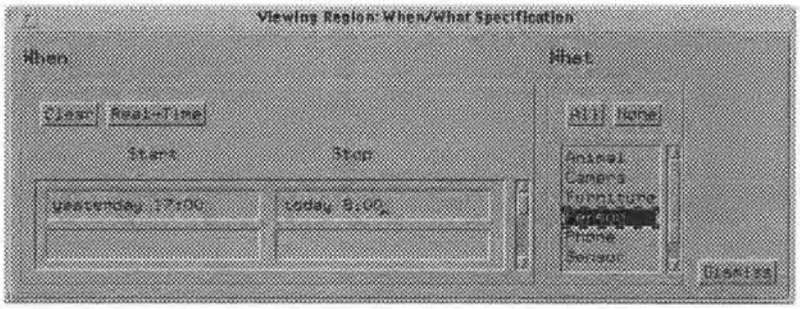
Figure 4-7: Graphical query interface viewing region

4.6 Graphical Query Interface

The scene monitoring prototype includes a graphical interface through which users can query the visual memory to retrieve information stored by the tracking software. A user establishes regions, times, and object types of interest, and the visual memory retrieves the corresponding objects. The query interface can display the results by dynamically stepping through the state changes of the objects, by displaying all the changes at once, or by displaying textual information about the objects.

The first step in posing a query is to select the query region. The query interface allows a user to step through a map hierarchy to select the map for the region of interest. The user can resize and scroll the query interface window to select an exact query region. This region specifies the spatial area for which objects should be retrieved. Figure 4-7 demonstrates a typical viewing region.

The next step is to establish alarm regions by shading rectangles on the map. In addition to displaying objects in the query region, the scene monitoring system alerts



Start: yesterday 17:00
Stop: today 8:00

Figure 4-8: Specification of query times and classes

the user to events in alarm regions. Alarm regions can be established all over the map, allowing the user to monitor a number of disjoint regions without having to watch them all.

Another step in query specification is to indicate a set of time intervals for each region, as shown in the left half of Figure 4-8. The system can parse times such as "3/8/93 8:00" and "today 13:00." It includes a special construct "..." to represent infinite queries retrieving all information after a given point. In addition, it provides the keyword "now" to signify a real-time query, one that constantly polls the database for new information.

An alarm region's temporal specification defaults to that of the query region. If an alarm region has an explicit temporal specification, that specification is conjoined with the query region's specification. This allows a user to specify, for example, that an alarm region should be active only during certain hours. The temporal specification for the query region identifies times of interest, and the temporal specification for an alarm region further restricts that specification to indicate exactly when the alarm should be active.

The user can specify for each region what types of objects are important, as shown in the right half of Figure 4-8. For example, the query region might return all objects, an indoor alarm region only people, and an outdoor alarm region both people and vehicles. Alarm regions default to the same type specification as the query region.

Associated with each alarm region is a delay specification that indicates how long

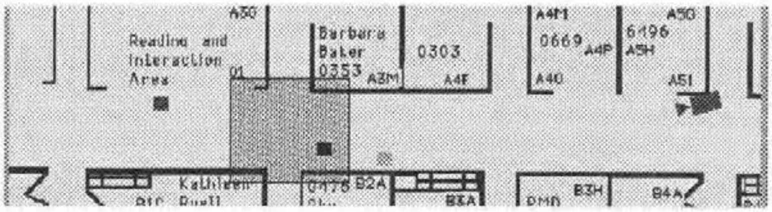


Figure 4-9: Graphical query results

an object must remain in that region before the system triggers an alarm. This lets the user specify that an alarm should fire only if an object remains in a region for a suspicious amount of time. The default value is 0 seconds, causing an alarm to be sounded as soon as an object enters the alarm region.

Once a query is fully specified, the results can be displayed in one of three ways: playback, event report, and trail trace. A playback steps through the retrieved information in temporal order, displaying moving blocks for moving objects and printing alarm information in another window. An event report textually describes alarms that were triggered. A trail trace displays blocks for all the retrieved information simultaneously and provides information about a certain object in response to a button press over its picture. Figure 4-9 shows part of a playback window, with one object inside an alarm region and two other objects also being monitored.

Chapter 5

Performance

One of the key requirements for the visual memory is to provide high-performance storage and retrieval of spatiotemporal information. The scene monitoring prototype described in Chapter 4 not only demonstrates the representational power of the visual memory design, it also provides a means for examining the performance of the prototype visual memory. This chapter studies some tests conducted to analyze the prototype's performance.

Visual memory performance can be measured in two main ways: by the number of objects stored and retrieved, and by the amount of time taken to store and retrieve those objects. The scene monitoring prototype is most concerned with how fast it can manipulate information, suggesting the use of temporal performance measurement. However, measuring the number of objects stored and retrieved can give an idea of the bottom-line visual memory performance and can help predict how changes in the storage and retrieval mechanism could affect the temporal performance. This chapter only discusses temporal performance, since both measurements follow approximately the same pattern and since timing measurements provide an intuitive benchmark.

The results of timing tests vary from machine to machine and from one execution to the next depending on system load, so they are most useful in providing comparative information. To reduce inaccuracy, times discussed here are the averages of three test executions. To provide more valid comparisons, the tests were run during the same time frame on a single machine with approximately the same system load.

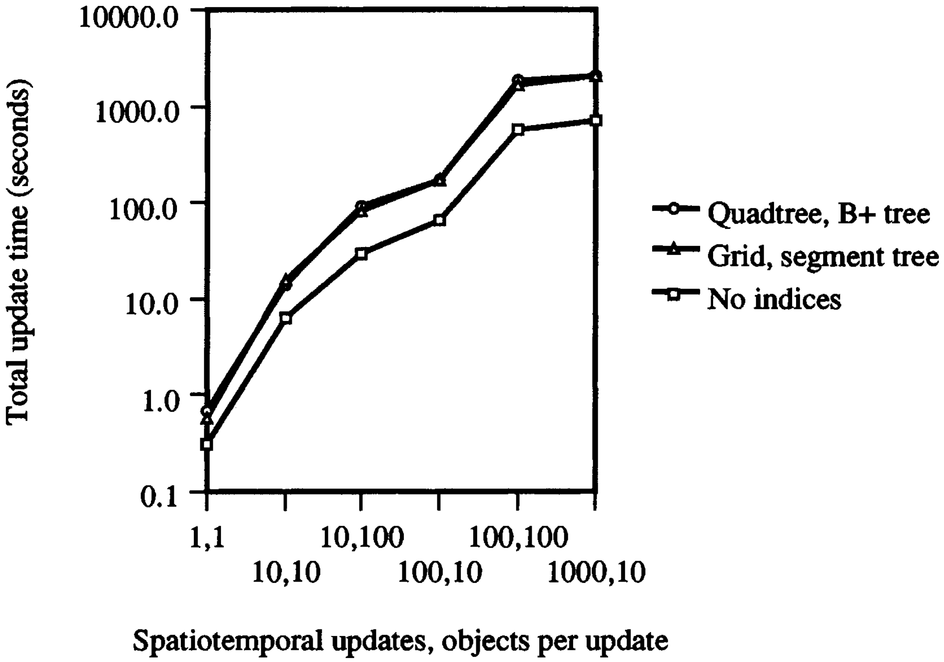


Figure 5-1: Spatiotemporal update performance

5.1 Spatiotemporal Object Storage and Retrieval

The prototype visual memory achieves reasonable spatiotemporal object storage performance. The underlying database limits the attainable performance, since it is responsible for actual object storage. With every spatiotemporal object update, the visual memory stores additional indexing information. A useful test of storage performance compares the time to store raw spatiotemporal objects with that to store both spatiotemporal objects and associated index information. The graph in Figure 5-1 shows storage times for spatiotemporal objects and different sets of indices as a function of the number of objects per update and the number of updates.

This graph shows that both raw storage time and indexed storage time steadily increase with the number of updates and the number of objects per update. Indexed storage costs a nearly constant factor of 2 to 3 times the raw update time. This overhead factor follows from the spatiotemporal indexing strategy discussed in Section 3.8.4, since for each update the visual memory stores spatiotemporal objects in a spatial index and the spatial index in a temporal index. While this seems to be a high price, it is necessary so that the visual memory can provide efficient spatiotemporal access to the stored information.

As a result of storing spatiotemporal index information, the visual memory can quickly answer spatiotemporal queries. Depending on indices, query complexity, and number of satisfying objects, the visual memory answered test spatiotemporal queries in 0.1 to 2.1 seconds. Clearly, retrieval performance is much better than storage performance.

5.2 Index Comparison

Chapter 4 describes two spatial and two temporal indices implemented in the visual memory prototype. The spatial indices can answer the same queries, but they differ in structure: the grid has a static structure built prior to execution, while the quadtree has a dynamic structure defined by the objects stored in it. Similarly, the temporal indices provide the same functionality, but the segment tree has a static structure and the B+ tree has a dynamic structure. The visual memory prototype provides a basis for comparing the performance of these indices.

Parameters such as branching factor and cell size affect index structure, so the tests must use comparable parameters. The spatial tests cover a 100-unit by 100-unit square. The quadtree has a bucket size of 10 objects and the fixed grid has a cell size of 10 units; this implies that the grid has 100 nodes and the quadtree has from 1 to a few hundred nodes. The temporal tests cover a time interval of up to 1000 seconds, and both temporal indices have a branching factor of 64.

In addition to the indices described above, each test also includes a "bucket" index. A bucket index simply maintains a list of all the objects stored in the visual memory. Since there is no overhead for the storage of complex index structure, a bucket index can achieve the highest update performance. A bucket index answers a query by retrieving all the objects in its list and checking them against the query specification.

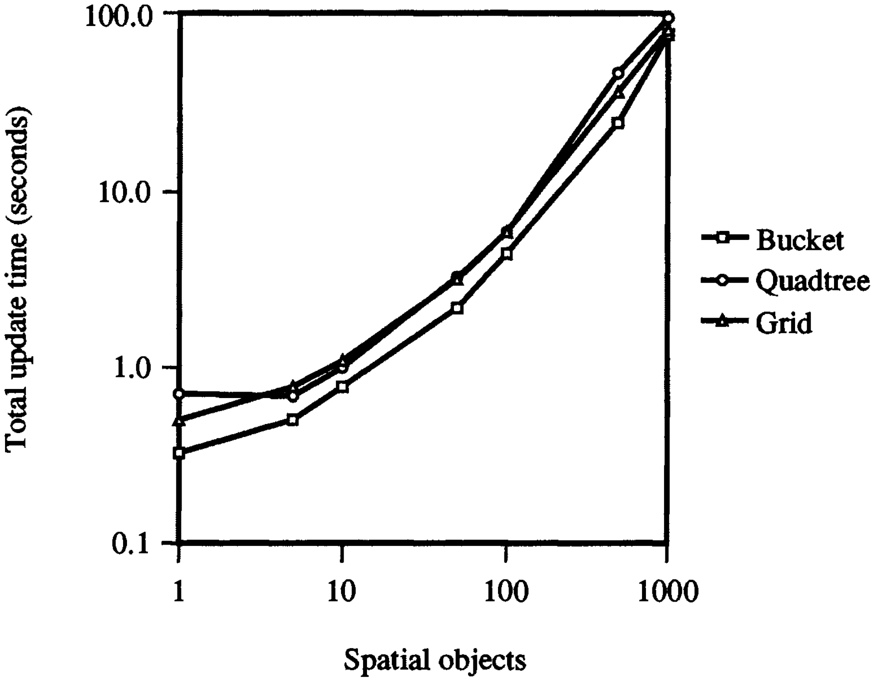


Figure 5-2: Spatial update performance

This is not an efficient query mechanism for large queries, but it provides a useful basis for comparing the performance of other indices.

One important performance measure compares how quickly indices can update information about objects. Figure 5-2 shows the update performance of spatial indices, and Figure 5-3 shows the update performance of temporal indices.

As expected, the bucket indices achieve the highest performance for small numbers of objects. However, the temporal bucket cannot store much more than 100 updates, since it saves an entire list with each update and quickly fills the database. Dynamic structures tend to perform slightly better than static structures for small numbers of objects, while static structures are better for large numbers of objects. This follows from the relative sizes of the structures; dynamic indices are initially small but grow as they store information about additional objects, while static indices maintain the same structure no matter how much information is stored.

Another important measure for index comparison is query performance. Timing

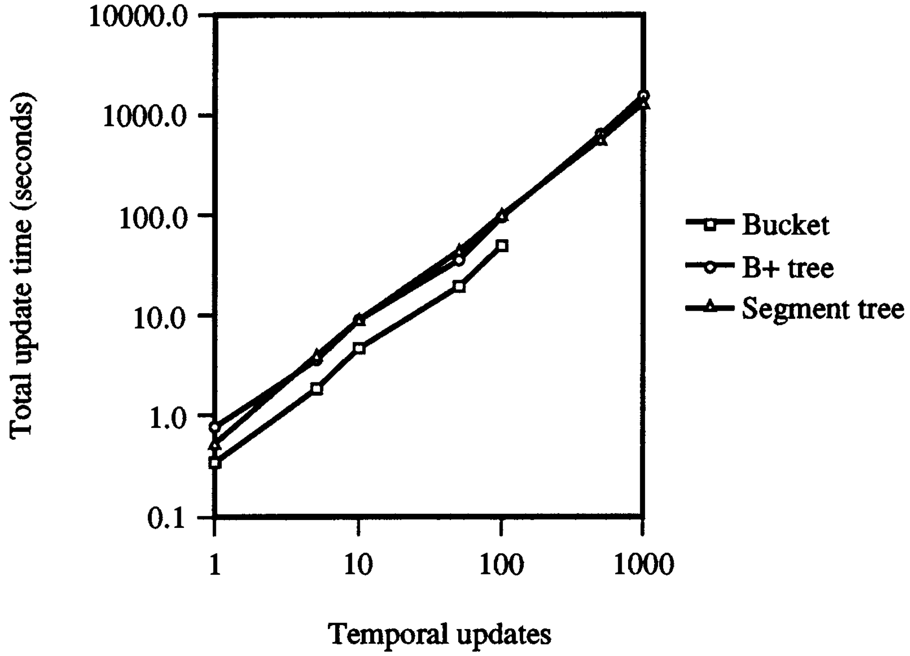


Figure 5-3: Temporal update performance

tests show that query performance follows a pattern similar to that of update performance: bucket indices achieve the best performance with small numbers of objects, dynamic structures work better than static structures with small numbers of objects, and static structures work better than dynamic structures with large number of objects. Figure 5-4 shows the performance for spatial indices with a 10-unit by 10-unit query square. Figure 5-5 shows the performance for temporal indices with a query interval of 10 seconds.

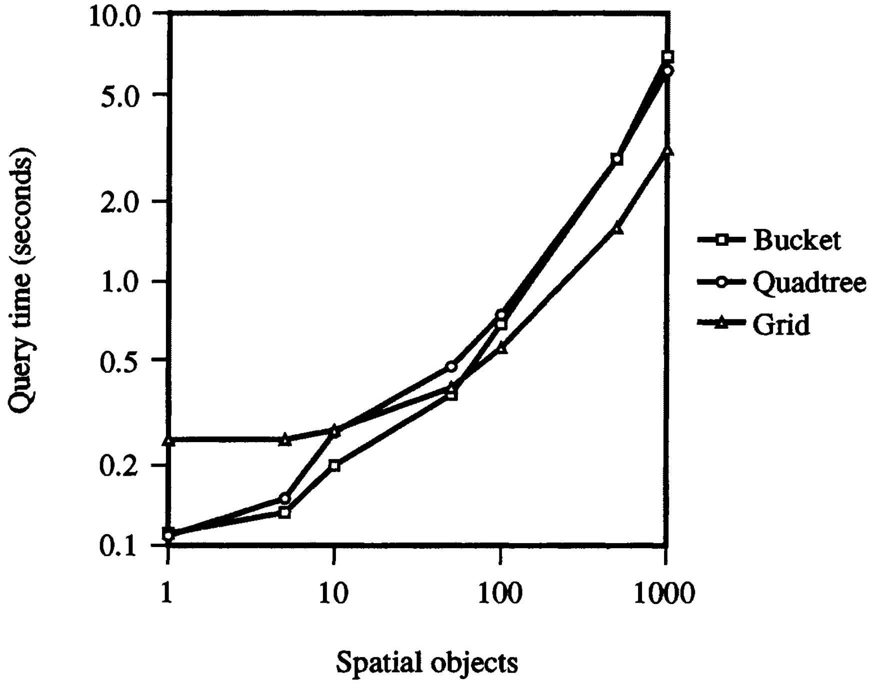


Figure 5-4: Spatial query performance

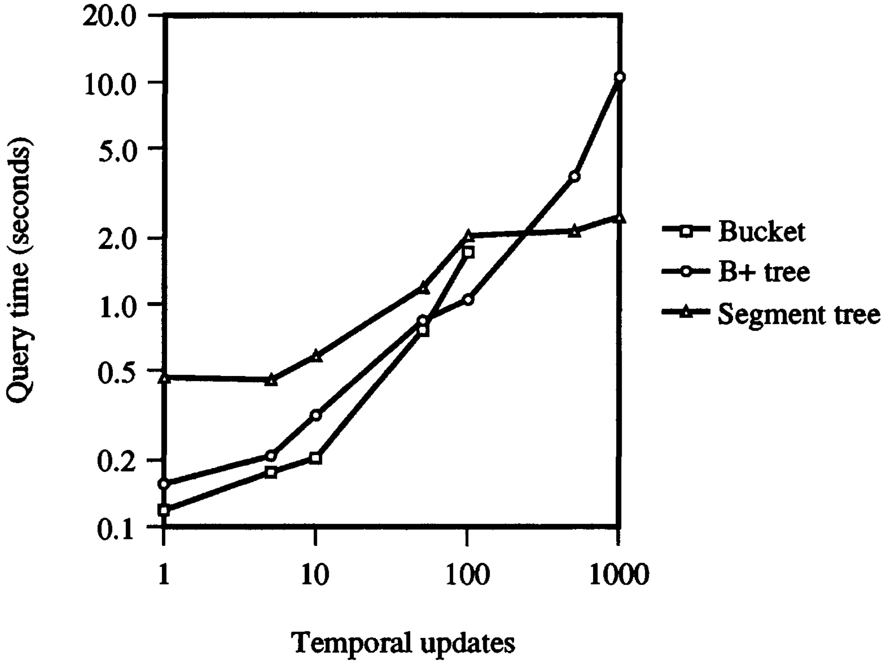


Figure 5-5: Temporal query performance

Chapter 6

Conclusion

The visual memory design presented in this thesis combines and extends spatial, temporal, and database research to meet the needs of a number of computer vision applications. It provides powerful and expressive spatiotemporal representations that it can efficiently manipulate, store, and retrieve. A prototype visual memory implemented in support of a scene monitoring prototype demonstrates the potential of this design. This prototype achieves useful storage and query performance and provides a basis for comparison of different indices.

Visual memory research could continue in many different directions. One step is to more fully implement the design. Some of the unimplemented spatiotemporal representations, such as probabilistic, relative, and abstract objects, could be beneficial to the scene monitoring prototype. The prototype visual memory could be connected to a number of different computer vision applications. Further implementation and testing would provide more feedback on the design and help identify areas for additional research.

The visual memory could furnish additional functionality if it used a different database. For example, if the database provided active rules, a security system could establish visual memory daemons to automatically check for alarms and to resolve old data. If the database provided real-time guarantees, a vehicle navigator could be sure that it would not crash because of visual memory performance. Finally, if the database provided data partitioning capabilities, applications that store large amounts of spatiotemporal data could make use of separate storage devices.

A number of extensions could improve the performance of the visual memory. Visual memory customization of caching and look-ahead could improve both storage and retrieval performance. Lightweight transactions could reduce overhead and increase storage performance for applications that continuously update the visual memory. Query optimization could increase retrieval performance by ordering parts of a query to reduce the number of retrievals. These extensions could help the visual memory reach its potential as high-performance system for manipulating spatiotemporal information.

Bibliography

[1] American National Standard for Information Systems. Database language SQL. Technical Report ANSI-X3.138-1986, American National Standards Institute, October 1986.

[2] José A. Blakeley. ZQL[C++]: Extending a persistent C++ language with a query capability. Technical Report 91-06-01, Texas Instruments Information Technologies Laboratory, 1991.

[3] John Brolio et al. ISR: A database for symbolic processing in computer vision. *IEEE Computer*, December 1989.

[4] M. Carey, D. DeWitt, and E. Shekita. Storage management for objects in EXODUS. In W. Kim and F. Lochovsky, editors, *Object-Oriented Concepts, Databases, and Applications.* Addison-Wesley Publishing Company, 1989.

[5] Surajit Chaudhuri. Temporal relationships in databases. In *Proceedings of the Conference on Very Large Databases*, 1988.

[6] Douglas Comer. The ubiquitous B-tree. *ACM Computing Surveys*, June 1979.

[7] Randall Davis, Bruce Buchanan, and Edward Shortliffe. Production rules as a representation for a knowledge-based consultation program. *Artificial Intelligence*, February 1977.

[8] Ramez Elmasri, Ihab El-Assal, and Vram Kouramajian. Semantics of temporal data in an extended ER model. In *Proceedings of the Entity-Relationship Conference*, October 1990.

[9] Ramez Elmasri, Muhammad Jaseemuddin, and Vram Kouramajian. Partitioning of time index for optical disks. In *IEEE Data Engineering Conference*, February 1992.

[10] Ramez Elmasri and G. Wiederhold. GORDAS: A formal high-level query language for the ER model. In *Proceedings of the Entity-Relationship Conference*, October 1981.

[11] James D. Foley and Andries Van Dam. *Fundamentals of Interactive Computer Graphics.* Addison-Wesley Publishing Company, 1984.

[12] G. D. Held, M. R. Stonebraker, and E. Wong. INGRES: A relational data base system. In *Proceedings of the National Computer Conference*, May 1975.

[13] Vram Kouramajian and Ramez Elmasri. Support for uncertainty in a generalized temporal model. Available from the authors at {kouramaj,elmasri}@cse.uta.edu.

[14] Jon Lukasiewicz. Numerical interpretation of the theory of propositions. In Borkowski, editor, *Jon Lukasiewicz: Selected Works.* North-Holland Publishing Company, 1970.

[15] Frank Manola and Jack A. Orenstein. Toward a general spatial data model for an object-oriented DBMS. In *Proceedings of the Conference on Very Large Databases*, August 1986.

[16] J. Mundy et al. *The Image Understanding Environments Program*, June 1992.

[17] Edward Perez and Robert W. Peterson. Zeitgeist Persistent C++ user manual. Technical Report 90-07-02, Texas Instruments Information Technologies Laboratory, February 1992.

[18] Hanan Samet. *The Design and Analysis of Spatial Data Structures.* Addison-Wesley Publishing Company, 1989.

[19] Arie Segev and Arie Shoshani. Logical modeling of temporal data. In *Proceedings of the ACM SIGMOD Conference*, June 1987.

[20] Steven A. Shafer, Anthony Stentz, and Charles E. Thorpe. An architecture for sensor fusion in a mobile robot. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 1986.

[21] Richard Snodgrass. The temporal query language TQuel. *ACM Transactions on Database Systems*, June 1989.

[22] Michael D. Soo. Bibliography on temporal databases. *ACM SIGMOD Record*, March 1991.

[23] Michael Stonebraker and Lawrence A. Rowe. The design of POSTGRES. In *Proceedings of the ACM-SIGMOD Conference on the Management of Data*, 1986.

[24] Thomas M. Strat and Grahame B. Smith. Core Knowledge System: Storage and retrieval of inconsistent information. In *Proceedings of the DARPA Image Understanding Workshop*, April 1988.

[25] David L. Wells, José A. Blakely, and Craig W. Thompson. Architecture of an open object-oriented database management system. *IEEE Computer*, October 1992.

EXHIBIT B

Full Record

Permalink for this record: http://library.mit.edu/item/000656176

Results List | **Add to Bookshelf** | **Save / Email**

Choose format: **Standard** | **Citation** | **MARC tags**

Record 1 out of 1

Author	Kollogg, Christopher James.
Title	Visual memory / by Christopher James Kellogg.
Shelf Access	Find it in the library/Request item
Shelf Location	Institute Archives - Noncirculating Collection 1 | Thesis E.E. 1993 M.S.
Shelf Location	Institute Archives - Noncirculating Collection 3 | Thesis E.E. 1993 M.S.
Shelf Location	Barker Library - Microforms | Thesis E.E. 1993 M.S.
Published	c1993.
Description	92 leaves : ill. ; 29 cm.
Format	Book
Thesis Note	Thesis (M.S.)--Massachusetts Institute of Technology, Dept. of Electrical Engineering and Computer Science, 1993.
Thesis Supervisor	Supervised by Alex P. Pentland.
Bibliography	Includes bibliographical references (leaves 90-92).
Local System Number	**000656176**

Basic Search of Full Catalog

Search type:

- Keyword
- Title begins with...
- Title Keyword
- Author (last name first)
- Author Keyword
- Call Number begins with...
- ----- Scroll down for more choices -----

Search for:

Search

EXHIBIT C

Barton MIT Libraries' Catalog

MIT Libraries

Search Full Catalog:
- Basic
- Advanced

Search only for:
- Conferences
- E-resources
- Journals
- MIT Theses
- Reserves
- more...

- Your Account
- Help with Your Account
- Your Bookshelf
- Previous Searches

Ask Us! Other Catalogs Help

Full Record

Permalink for this record: http://library.mit.edu/item/000656176

Results List | **Add to Bookshelf** | **Save/Email**

Choose format: **Standard** | **Citation** | **MARC tags**

Record 1 out of 1

FMT	BK
LDR	00968nam 2200265K 45q0
003	MCM
005	20010608211241.0
008	930928s1993 xx a b 000 0 eng d
035	|a MITb10656176
035	|a (OCoLC)28904769
035	|a GLIS00656176
040	|a MYG |c MYG
099	|a Thesis E.E. 1993 M.S. |a Thesis E.E. 1993 M.S. Mfch
099	|a Thesis |a E.E. |a 1993 |a M.S.
1001	|a Kollogg, Christopher James.
24510	|a Visual memory / |c by Christopher James Kellogg.
260	|c c1993.
300	|a 92 leaves : |b ill. ; |c 29 cm.
502	|a Thesis (M.S.)--Massachusetts Institute of Technology, Dept. of Electrical Engineering and Computer Science, 1993.
504	|a Includes bibliographical references (leaves 90-92).
59900	|a Supervised by Alex P. Pentland.
599	|a GRSN 656176
CAT	|a CONV |b 00 |c 20010620 |l MIT01 |h 1525
CAT	|a lti0904 |b 00 |c 20090523 |l MIT01 |h 2117
049	|a MYGG
910	|a jlh
949	|a MYTT |b 39080008599745 |a [Micro-] [fiche] MYTT |b 39080008664614 |a MYTE |b 39080008666296 |a [Micro-] [fiche] MYTE |b 39080008664663
PST8	|0 Z30 |1 000656176000020 |b ARC |c NOLN1 |o BOOK |d 02 |y 00000 |f N |r MIT60-000584976 |n 8 |h Thesis E.E. 1993 M.S. |k THESIS |a MCM |3 Book |4 Institute Archives |5 Noncirculating Collection 1 |6 Room Use Only |p Avail
PST8	|0 Z30 |1 000656176000010 |b ARC |c NOLN3 |o BOOK |d 02 |y 00000 |f N |r MIT60-000579310 |n 8 |h Thesis E.E. 1993 M.S. |k THESIS |a MCM |3 Book |4 Institute Archives |5 Noncirculating Collection 3 |6 Room Use Only
PST8	|0 Z30 |1 000656176000030 |b ENG |c MFORM |o BOOK |d 12 |y 00000 |f N |r MIT60-000584981 |n 8 |h Thesis E.E. 1993 M.S. |k THESIS |a MCM |3 Book |4 Barker Library |5 Microforms |6 Thesis Loan
LDR	nx 22 zn 4500
008	0106230u 0 4 uu 1
004	000656176
8528	|a MCM |b ARC |c NOLN3 |h Thesis E.E. 1993 M.S. |k THESIS |z
LDR	nx 22 zn 4500
008	0106230u 0 4 uu 1
004	000656176
8528	|a MCM |b ARC |c NOLN1 |h Thesis E.E. 1993 M.S. |k THESIS |z
LDR	nx 22 zn 4500
008	0106230u 0 4 uu 1
004	000656176
8528	|a MCM |b ENG |c MFORM |h Thesis E.E. 1993 M.S. |k THESIS |z

001 000656176
SFX01 |s 0-0-0-6-5-6-1-7-6 |l MIT01 |9 000 |z ~~~~~~~~~~ |p Avail |f 000
LU564 BK
SYS 000656176

Basic Search of Full Catalog

Search type:
Keyword
Title begins with...
Title Keyword
Author (last name first)
Author Keyword
Call Number begins with...
----- Scroll down for more choices -----

Search for:
Search

Barton Questions: Ask Us! | Contact Us
Massachusetts Institute of Technology
77 Massachusetts Avenue, Cambridge, MA 02139-4307 USA

-- Quick Links --

EXHIBIT D

American Defense
Preparedness Association's
Security Technology Division

10th Joint Annual Government-Industry Security Technology Symposium & Exhibition

*"Reducing Security Vulnerabilities
Through Innovation and Technology—
The Risk Management Challenge"*

PROCEEDINGS

Williamsburg, Virginia
June 20-23, 1994

TABLE OF CONTENTS

SESSION I
Government Security Perspectives

SESSION II
Transportation Security

SESSION III
Risk Management

TABLE OF CONTENTS (Continuation)

SESSION IV
Department of Defense Programs

TABLE OF CONTENTS (Continuation)

SESSION V
Information & Computer Security

SESSION VI
Department of Justice

SESSION VII
Arms Control & Non-Proliferation

TABLE OF CONTENTS (Continuation)

PRESENTATIONS ***PAGE***

SESSION VIII
Security Systems Engineering

TABLE OF CONTENTS (Continuation)

ALTERNATE PAPERS

SESSION VI

Department of Justice

Speakers will focus on the new technologies & policies developing in light of the modern challenges and risks in law enforcement and security policy

Session Chairperson:

James Mahan, Session Chairperson
Federal Bureau of Prisons - Washington, DC

Autonomous Scene Monitoring System

Bruce Flinchbaugh & Tom Bannon

Texas Instruments
Systems & Information Science Laboratory
P.O. Box 655474, MS 238
Dallas, TX 75265
Phone: 214-995-0349

Abstract

Autonomous scene monitoring poses substantial requirements for extracting, storing, and displaying information about three-dimensional (3-D) scenes. We have developed and demonstrated a data base system called visual memory that interfaces automated video camera monitoring systems with end-user applications requiring real-time and historical information about observed objects. This paper describes our visual memory and real-time camera monitoring capabilities. The visual memory prototype uses state-of-the-art object-oriented data base technology with spatio-temporal indexing extensions. The video monitoring system reports 3-D positions of people in the field of view of a CCD camera to visual memory, which dynamically maintains the information with respect to a map, and a user interface provides interactive access to the data via historical and real-time queries.

1 Motivation

CCTV surveillance cameras provide valuable data in many security monitoring situations. For example, in some cases CCTV images are recorded using time lapse video recorders. Security system operators use CCTV monitors for remote situation assessment when an alarm detector signals a potential problem. And in some systems video motion detectors are used to automatically signal alarms when changes are detected in the CCTV data.

Interestingly, practically all of the information available from CCTV cameras is essentially ignored. In many time-lapse video surveillance situations, the images recorded on tape are never viewed or used in any way unless a specific event occurs, such as an accident or a theft, and even then only a small portion of the data may be viewed. Although people are generally good at assessing situations using CCTV data, it is well known that operator performance deteriorates significantly with fatigue. And video motion detectors

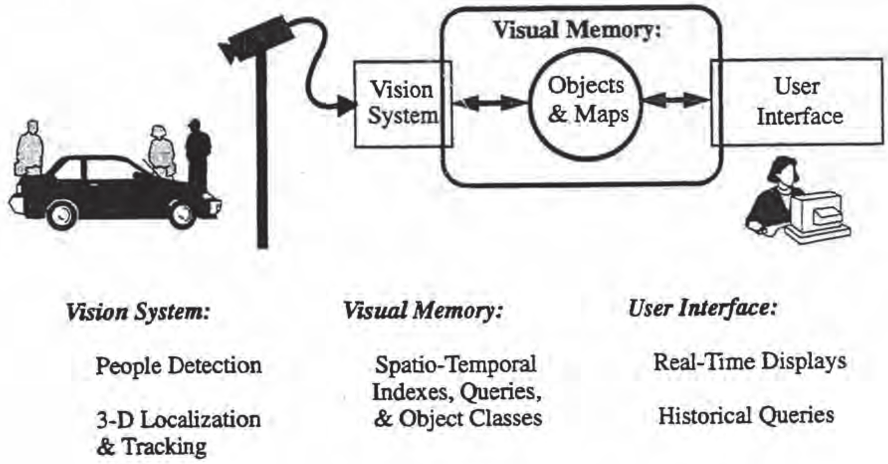


Figure 1: Autonomous Scene Monitoring System.

simply detect changes in the incoming light, relying on operators to visually assess the situation.

In principle, situation assessment can be performed automatically by computers, to continuously exploit available CCTV surveillance data all of the time. Computer vision systems can be used to extract information about the scene, such as how many people are in the field of view, and whether a particular kind of vehicle has entered the scene. In some cases this descriptive information may be all that is needed to facilitate efficient security monitoring and concise record keeping.

2 Autonomous Scene Monitoring Architecture

At Texas Instruments we have developed a prototype scene monitoring system that automatically extracts complex information about scenes from CCTV camera images and provides operators with convenient access to the information.

The overall scene monitoring system is illustrated in Figure 1. The *Vision System* detects people walking in the CCTV camera field of view and continuously reports their 3-D positions as they move. The *Visual Memory* at the center of the scene monitoring system is an object-oriented data base that stores the information reported by the vision system. As people walk, their current 3-D positions are updated in visual memory, and a history of their movement is maintained for future reference. The *User Interface* provides interactive graphical access to real-time and historical events stored in visual memory.

The Vision System, Visual Memory, and User Interface software are implemented on two computers. The Vision System uses a Datacube MaxVideo 20 real-time image processor and a Sun SPARCstation 10, while the Visual Memory and User Interface run on the Sun workstation. Users may also access Visual Memory over a network using an X Window System server or another workstation. The camera is a Texas Instruments monochrome MC-780PH CCD camera with a resolution of 755x484 8-bit pixels, of which 492x460 pixels are used by the Vision System. The camera is mounted in a pan/tilt/zoom unit about 8 feet high on a hallway ceiling in our laboratories. From this vantage point the camera can observe several hallways, including a section approximately 9 feet wide, 117 feet long, and 12 feet high. The hallway is illuminated by overhead and wall-mounted fluorescent lighting, with significant variations caused by natural light from large windows at one end of the hall.

3 Real-Time Scene Monitoring

Algorithms of the Vision System report where people are walking in the field of view. The algorithms use basic image processing techniques to continuously detect scene motion. Regions of motion are analyzed for consistent interpretations as people standing or walking. By using knowledge of the scene geometry, the algorithms estimate positions and heights of people in the scene. The system detects and updates the positions of people at a rate of ten frames per second.

4 Visual Memory

Our visual memory prototype project developed an architecture [1] to interface vision systems with applications requiring access to information about 3-D objects, events and their environment. Visual Memory requirements include:

- Storage of objects at high frame rates
- Retrieval from multi-gigabyte data volumes
- Support for diverse data structures

We selected an object-oriented data base (OODB) [2, 3] to use as the basis for the visual memory prototype. Relational data base technology is poorly suited for Visual Memory because it does not adequately support diverse data structures. The OODB is illustrated in Figure 2. For Visual Memory, the Indexing and Address Space modules of the OODB were extended to speed object storage and retrieval. In particular, several spatio-temporal indexing mechanisms were introduced as described below.

4.1 Spatial Indexing

Spatial indexing provides fast, efficient answers to questions such as, "Is anyone in area X?" Spatial indices typically provide conservative approximate answers, allowing false positives but not false negatives, and rely on further filtering for exact results.

Several different spatial indices are available, catering to different types of questions. For example, grids and point quadtrees are good for determining objects near a given

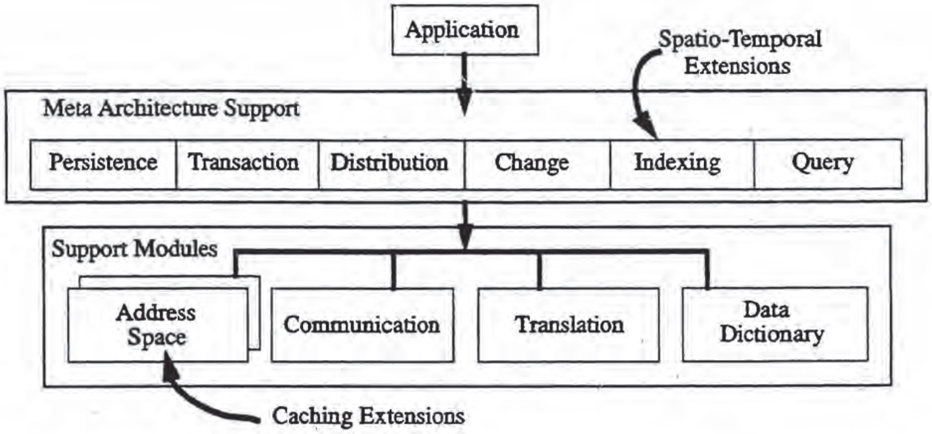


Figure 2: Object-Oriented Data Base.

point, while interval trees are better for finding object intersections [4]. The visual memory architecture supports multiple indices, and its query mechanism determines the appropriate index for a given question.

4.2 Temporal Indexing

Temporal indices provide means for accessing object histories by efficiently determining an object's state at a given time and how objects changed over a given interval of time. They help answer questions such as, "Was anyone in hall X during the night?" and "Where was object Y at 10:00am?"

Mathematically, temporal indexing mechanisms may be regarded as special cases of spatial indices for one dimension (time). Thus selection of a particular temporal index depends on the intended use. For example, image processing inputs to visual memory use only increasing time. Also, old information becomes increasingly unimportant and can be archived [5] for infrequent access. Tree indices support these requirements.

5 User Interface

The user interface provides interactive access to visual memory data via historical and real-time queries. To make historical queries, the user specifies periods of time, regions of space, and object types. Then the system retrieves the corresponding objects. To display real-time information, locations of people are indicated on a floor plan display and changed dynamically as the visual memory is updated.

Users may specify alarm regions on the map so that when someone enters that area an alarm is signaled. Digital snapshots of the scene may be kept and saved for future reference. Other user interface features provide interactive control of the camera pan, tilt, zoom, focus, and gain settings.

6 Concluding Remarks

The autonomous scene monitoring system currently operates in our laboratories with two remotely controlled pan/tilt/zoom cameras. We have also demonstrated operation of the system outdoors, using an infrared camera to map the position of a person walking along a sidewalk and across a street. The object-oriented data base architecture [3] underlying the Visual Memory facilitates expansion of the prototype scene monitoring system to handle hundreds of cameras, vision systems, and many users. As vision systems and visual memory evolve, increasingly sophisticated surveillance tasks will be automated to enhance security systems.

Acknowledgments

We thank Chris Kellogg, Steve Ford and Tom O'Donnell for their contributions in the conception, design, and development of the autonomous scene monitoring system.

References

[1] Kellogg, C., "Visual Memory", MIT Thesis, May 1993. *Also available as:* Technical Report CSL-93-05-20, Texas Instruments, Systems & Information Science Laboratory.

[2] S. Ford, et al, "ZEITGEIST: Database Support for Object-Oriented Programming," Advances in Object-Oriented Database Systems, 2nd International Workshop on Object-Oriented Database Systems, Springer-Verlag, Sep. 1988.

[3] Wells, D.L., J.A. Blakeley, & C.W. Thompson, "Architecture of the Open Object-Oriented Database Management System," *IEEE Computer*, Vol. 25, No. 10, Oct. 1992.

[4] Samet, H., The Design and Analysis of Spatial Data Structures. Addison-Wesley Publishing Company, Inc., 1989.

[5] Elmasri, R., M. Jaseemuddin, & V. Kouramajian, "Partioning of Time Index for Optical Disks," *IEEE Data Engineering Conference*, Feb. 1992.

EXHIBIT E

Close this window to return to the catalogue

EXPLORE THE BRITISH LIBRARY

Item Details

FMT	BK
LDR	caa a22002053u 4500
001	006374131
008	970217s1994 xxu || 100 ||eng
040	|a Uk |c Uk
CNF	|a Security technology (Annual joint government-industry symposium) |n (10th : |d 1994 Jun : |c Williamsburg, VA)
24510	|a 10th Annual joint government-industry security technology symposium and exhibition.
260	|b The Association, |c 1994.
336	|a text |2 rdacontent
337	|a unmediated |2 rdamedia
338	|a volume |2 rdacarrier
653	|a security technology
653	|a defense preparedness
7102	|a American Defense Preparedness Association. |b Security Technology Division.
945	|a Annual joint government industry security technology symposium and exhibition ; 10th
85241	|a British Library |b DSC |j 1086.803000 |i 10th 1994
SYS	006374131

P1

EXHIBIT F-1

US007023469B1

(12) **United States Patent**
Olson

(10) **Patent No.: US 7,023,469 B1**
(45) **Date of Patent: Apr. 4, 2006**

(54) **AUTOMATIC VIDEO MONITORING SYSTEM WHICH SELECTIVELY SAVES INFORMATION**

(75) Inventor: **Thomas J. Olson**, Plano, TX (US)

(73) Assignee: **Texas Instruments Incorporated**, Dallas, TX (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 444 days.

(21) Appl. No.: **09/292,265**

(22) Filed: **Apr. 15, 1999**

Related U.S. Application Data

(60) Provisional application No. 60/083,718, filed on Apr. 30, 1998.

(51) **Int. Cl.**
H04N 7/18 (2006.01)

(52) **U.S. Cl.** **348/152**; 348/143; 348/154

(58) **Field of Classification Search** 348/152, 348/154, 143, 700; 382/104; 379/40; 707/218; 340/565; 701/207; 345/474
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,249,207	A	*	2/1981	Harman et al.	348/152
4,943,854	A		7/1990	Shiota et al.	358/108
5,111,291	A		5/1992	Erickson et al.	358/108
5,289,275	A	*	2/1994	Ishii et al.	348/154
5,425,139	A	*	6/1995	Williams et al.	345/474
5,467,402	A	*	11/1995	Okuyama et al.	382/104
5,491,511	A		2/1996	Odle	349/153
5,572,643	A	*	11/1996	Judson	709/218
5,943,394	A	*	8/1999	Ader et al.	379/40
5,961,571	A	*	10/1999	Gorr et al.	701/207
5,966,074	A	*	10/1999	Baxter	340/565
5,969,755	A	*	10/1999	Courtney	348/143
6,049,363	A	*	4/2000	Courtney et al.	348/700
6,069,655	A	*	5/2000	Seeley et al.	348/154

OTHER PUBLICATIONS

U.S. Appl. No. 08/795,423, filed Feb. 5, 1997.

U.S. Appl. No. 08/866,789, filed May 30, 1997.

Jonathan D. Courtney, "Automatic Video Indexing via Object Motion Analysis", *Pattern Recognition*, Apr., 1997, cover page and pp 1-31.

Niyogi, et al., "Analyzing and Recognizing Walking Figures in XYT", 1994 IEEE, pp. 469-474.

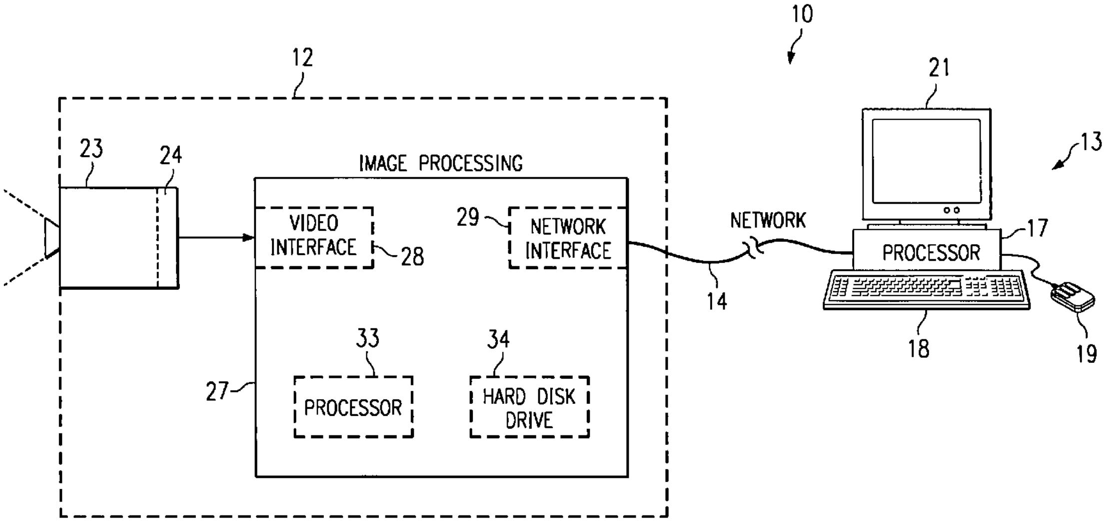
(Continued)

Primary Examiner—Allen Wong
(74) *Attorney, Agent, or Firm*—Robert D. Marshall, Jr.; W. James Brady, III; Frederick J. Telecky, Jr.

(57) **ABSTRACT**

A system (**10**) for automatically monitoring an area includes a camera unit (**12**) having therein a video camera (**23**) and an image processing section (**27**). The image processing section saves a reference image from the video camera, compares subsequent images to the reference image, and detects and tracks change regions in the subsequent images. For each change region, the image processing section saves a path of movement of the change region, and a selected image of the change region. Selection is carried out so as to optimize the selected image, for example so that a detected person is facing and close to the video camera. The camera unit is network-ready (**14**), so that a remote workstation (**13**) can access the images and other information saved in the camera unit.

28 Claims, 6 Drawing Sheets



OTHER PUBLICATIONS

Wren, et al., "Pfinder: Real-Time Tracking of the Human Body", M.I.T. Media Laboratory Perceptual Computing Section Technical Report No. 353, published in SPIE 1995 vol. 2615, pp. 1-9.
Turk, et al., "Eigenfaces for Recognition", 1991 Massachusetts Institute of Technology, Journal of Cognitive Neuroscience vol. 3, No. 1, pp. 71-86.
Flinchbaugh, et al., "Autonomous Scene Monitoring System", pp. 205-209.
Norris, et al., "Algorithmic Surveillance—The future of automated visual surveillance", CCTV, Surveillance and Social Control Conference, Jul. 9, 1996, pp. 1-21.
Jonathan D. Courtney, "Automatic Object-Based Indexing for Assisted Analysis of Video Data", (1995) pp. 1-25.
Bruce Flinchbaugh, "Robust Video Motion Detection and Event Recognition", May 11, 1997, pp. 51-54.
Kashi Rao, "A Computer Vision System to Detect 3-D Rectangular Solids", 6 unnumbered pages.
Jain, et al., "Segmentation through the Detection of Changes Due to Motion", Computer Graphics and Image Processing 11, (1979), pp. 13-34.
"DTS-1000 DTS-1000VB Video Motion Detection & Tracking System" Stellar Products Incorporated, 4 unnumbered pages, 1995.
Olson, et al., "Moving Object Detection and Event Recognition Algorithms for Smart Cameras", May 11, 1997, pp. 159-175.

* cited by examiner

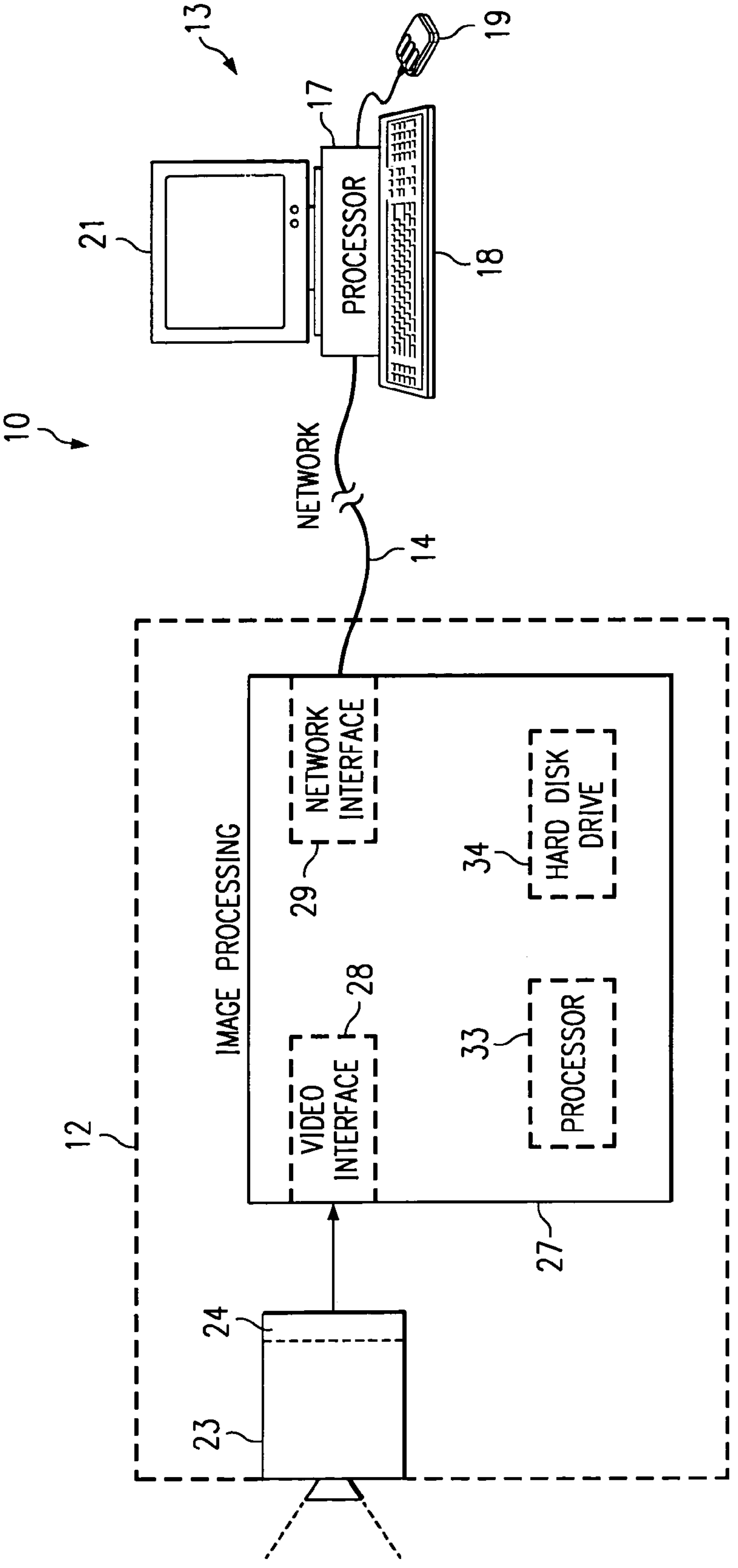

10
13
21
17
19
PROCESSOR
18
NETWORK
14
12
IMAGE PROCESSING
VIDEO INTERFACE
28
NETWORK INTERFACE
29
33
PROCESSOR
34
HARD DISK DRIVE
27
24
23

FIG. 1

FIG. 2A

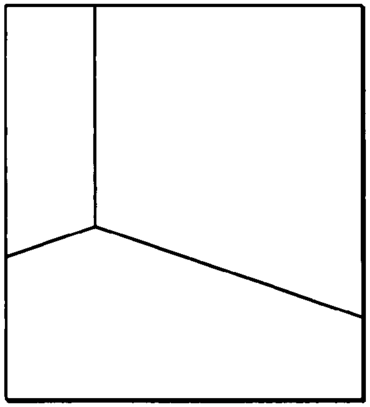


FIG. 2B

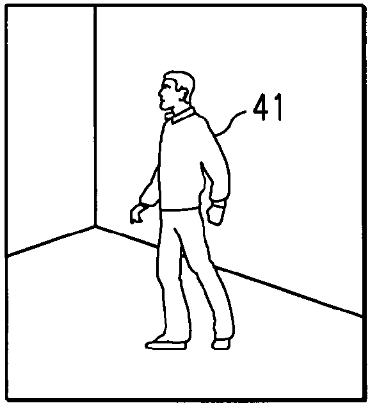


FIG. 2C

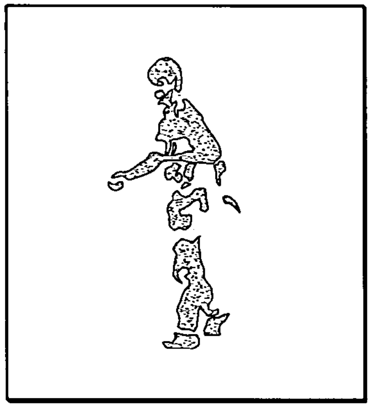
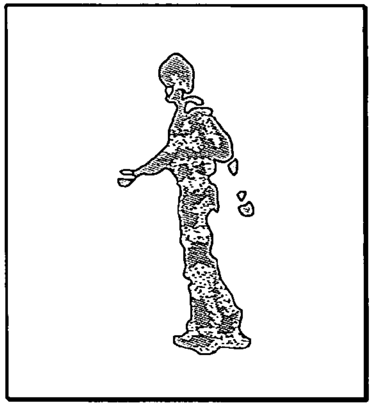


FIG. 2D



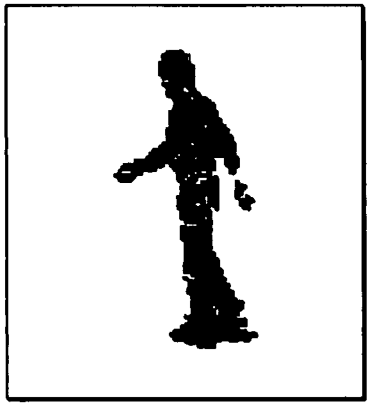


FIG. 2E

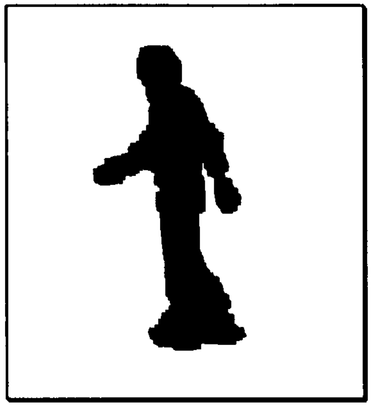


FIG. 2F

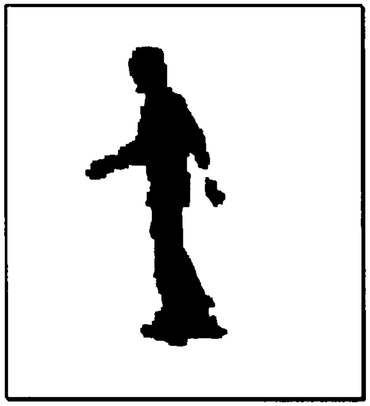


FIG. 2G

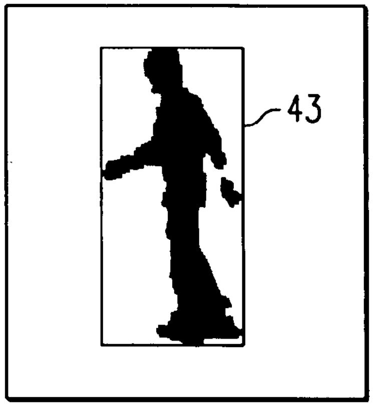


FIG. 2H

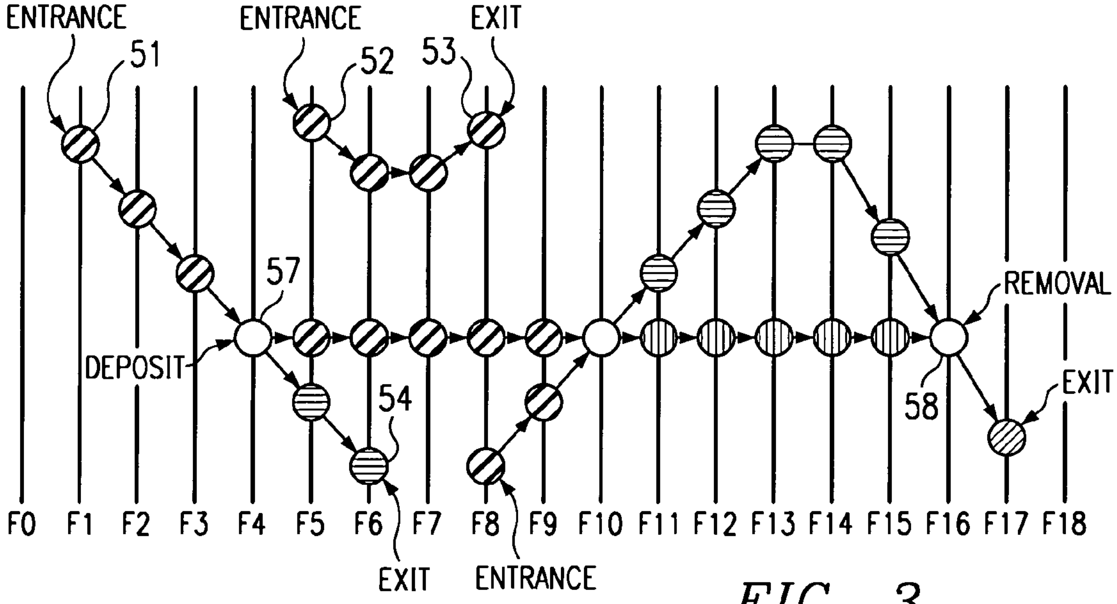

ENTRANCE
51
ENTRANCE
52
53
EXIT
57
DEPOSIT
54
REMOVAL
58
EXIT
F0
F1
F2
F3
F4
F5
F6
F7
F8
F9
F10
F11
F12
F13
F14
F15
F16
F17
F18
EXIT
ENTRANCE

FIG. 3

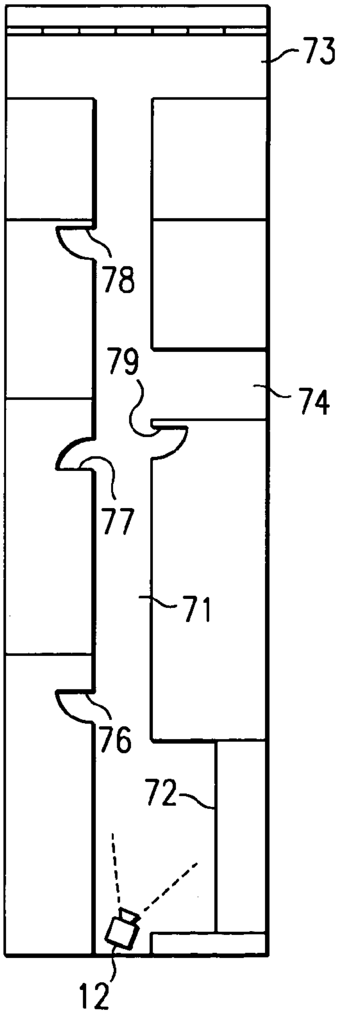

73
78
79
74
77
71
76
72
12

FIG. 4

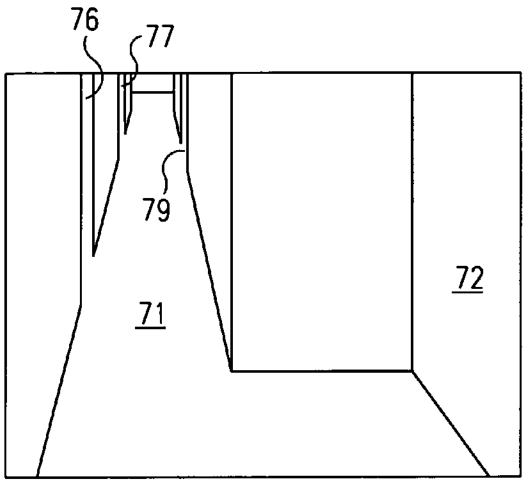


FIG. 5

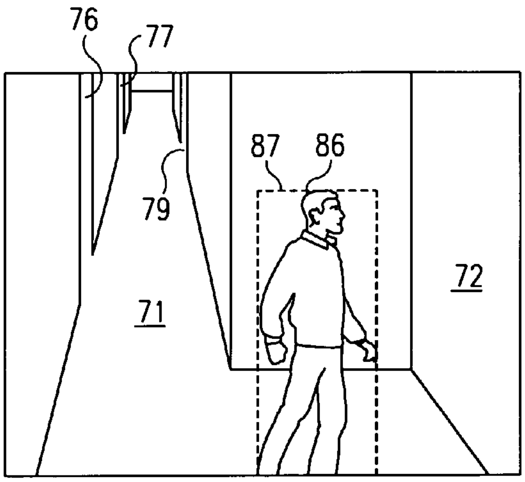


FIG. 6

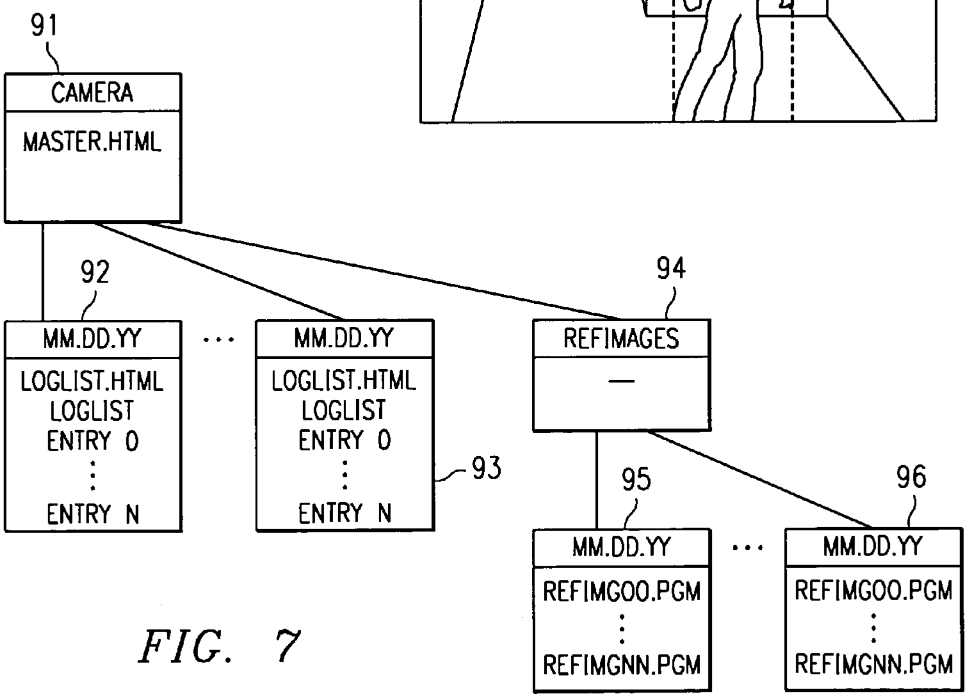
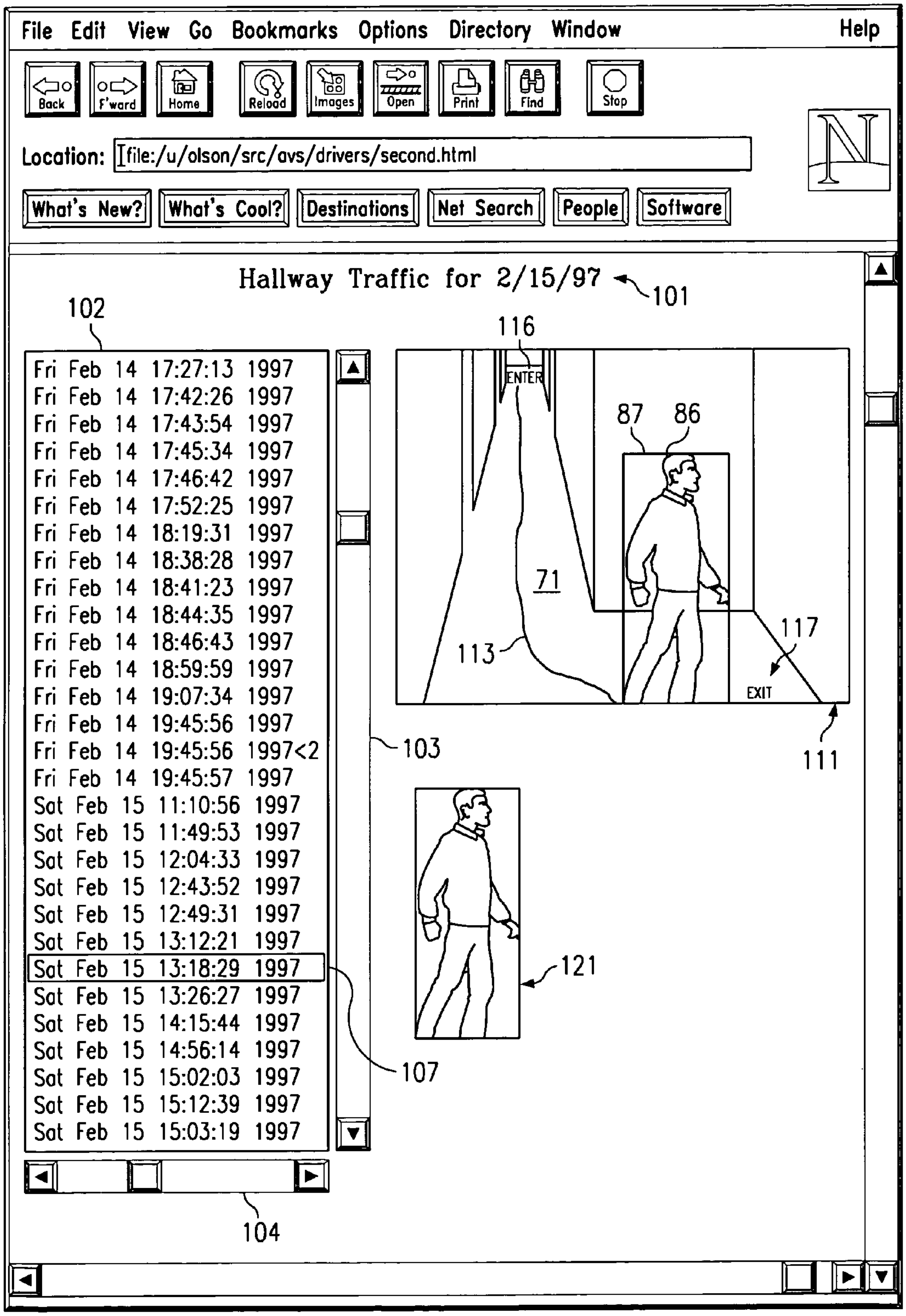


FIG. 7

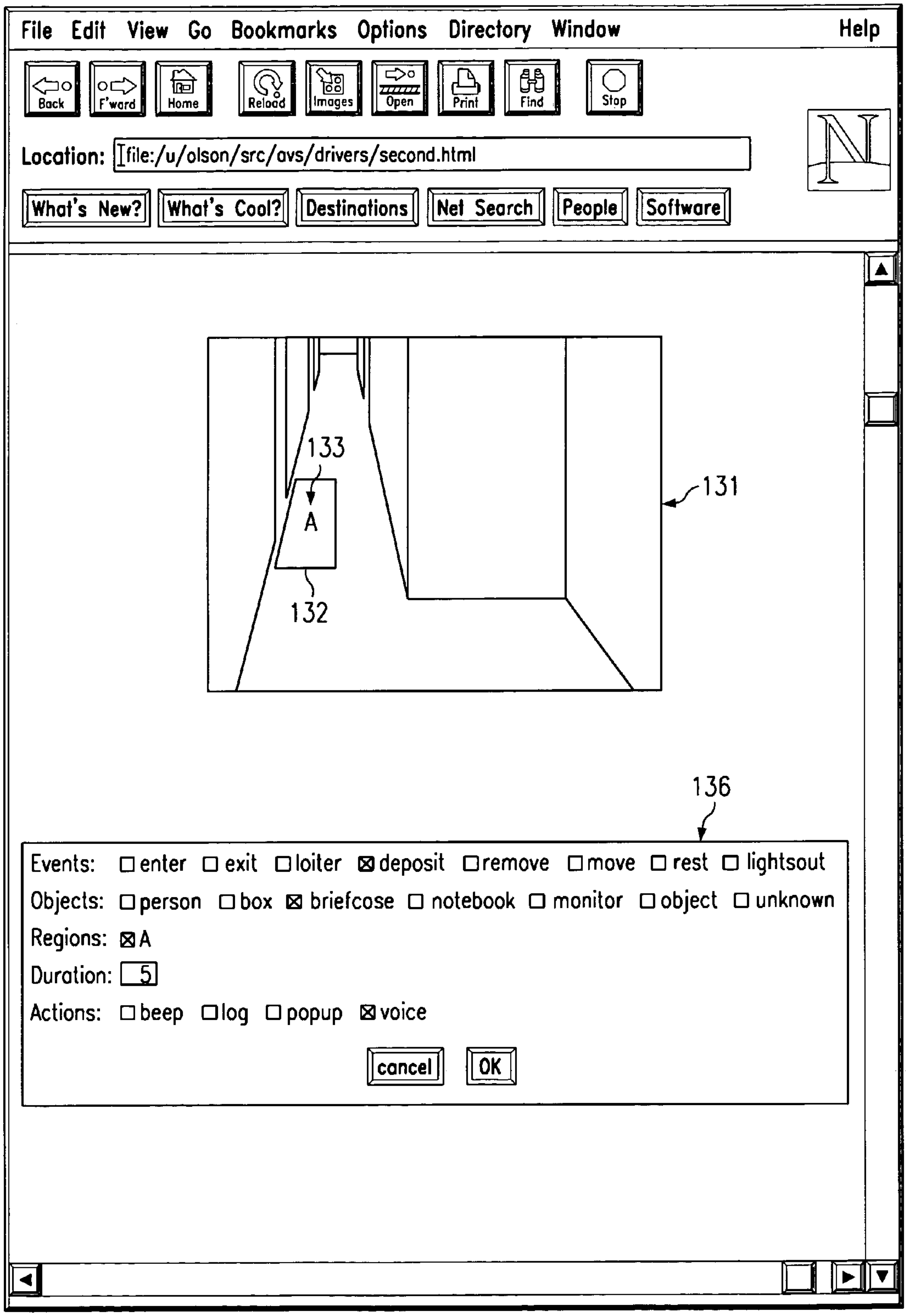


Location: file:/u/olson/src/avs/drivers/second.html

Hallway Traffic for 2/15/97

```
Fri Feb 14 17:27:13 1997
Fri Feb 14 17:42:26 1997
Fri Feb 14 17:43:54 1997
Fri Feb 14 17:45:34 1997
Fri Feb 14 17:46:42 1997
Fri Feb 14 17:52:25 1997
Fri Feb 14 18:19:31 1997
Fri Feb 14 18:38:28 1997
Fri Feb 14 18:41:23 1997
Fri Feb 14 18:44:35 1997
Fri Feb 14 18:46:43 1997
Fri Feb 14 18:59:59 1997
Fri Feb 14 19:07:34 1997
Fri Feb 14 19:45:56 1997
Fri Feb 14 19:45:56 1997<2
Fri Feb 14 19:45:57 1997
Sat Feb 15 11:10:56 1997
Sat Feb 15 11:49:53 1997
Sat Feb 15 12:04:33 1997
Sat Feb 15 12:43:52 1997
Sat Feb 15 12:49:31 1997
Sat Feb 15 13:12:21 1997
Sat Feb 15 13:18:29 1997
Sat Feb 15 13:26:27 1997
Sat Feb 15 14:15:44 1997
Sat Feb 15 14:56:14 1997
Sat Feb 15 15:02:03 1997
Sat Feb 15 15:12:39 1997
Sat Feb 15 15:03:19 1997
```

FIG. 8



Location: file:/u/olson/src/avs/drivers/second.html

Events: ☐ enter ☐ exit ☐ loiter ☒ deposit ☐ remove ☐ move ☐ rest ☐ lightsout

Objects: ☐ person ☐ box ☒ briefcase ☐ notebook ☐ monitor ☐ object ☐ unknown

Regions: ☒ A

Duration: 5

Actions: ☐ beep ☐ log ☐ popup ☒ voice

FIG. 9

AUTOMATIC VIDEO MONITORING SYSTEM WHICH SELECTIVELY SAVES INFORMATION

This application claims priority under 35 USC §119(e)(1) of Provisional Application No. 60/083,718, filed Apr. 30, 1998.

TECHNICAL FIELD OF THE INVENTION

This invention relates in general to automatic monitoring systems and, more particularly, to an automatic video monitoring system that selectively saves information derived from video images of a monitored area.

BACKGROUND OF THE INVENTION

Surveillance cameras are widely used as an aid in providing physical security for employees and property, such as commercial, industrial and government facilities. In many instances, the images from the camera are simply viewed in real-time by security guards.

It is also common to record the output of each camera on a time-lapse video cassette recorder (VCR). In the event of a problem or security incident, the resulting recording can then be examined. It is also possible to use a video or infrared motion detector, so that the VCR does not record anything except when there is motion in the observed area. This reduces the consumption of tape and makes it easier to find footage of interest. Nevertheless, it does not eliminate the need for the VCR, which is a relatively complex and expensive component that is subject to mechanical failure and that requires periodic maintenance, such as cleaning of the video heads. Moreover, infrared motion detectors have a tendency to produce false detections.

Another known approach is to use an all-digital video imaging system, which converts each video image to a compressed digital form immediately upon capture. The digital data is then saved in a conventional database (such as a disk farm backed up by a tape juke box). This approach is relatively expensive, requires a substantial amount of storage space, and does nothing to help an operator find frames of interest.

Another approach uses a video camera and personal computer to detect and track people, and saves the first image that satisfies some alarm condition. However, this system makes no attempt to select a good view of the person, as a result of which the saved image may show the person with his or her back to the camera, rendering it difficult or impossible to identify the particular person. Another known system displays a path of movement of a detected person who is in the observed area, but discards the path of movement after the person leaves the observed area.

All of these known approaches have been generally adequate for their intended purposes, but they have not been satisfactory in all respects. For example, they involve hardware which is relatively expensive and not particularly compact. They often use a VCR, which is subject to mechanical failure and requires periodic maintenance. Some systems store all incoming video information, which uses a substantial amount of storage capacity, and makes it difficult to find of events of interest.

SUMMARY OF THE INVENTION

From the foregoing, it may be appreciated that a need has arisen in the automatic monitoring field for a method and apparatus which are reliable, which intelligently save selected information that is meaningful but minimizes storage capacity, and which facilitate the location and review by an operator of events of interest. As to the apparatus, there is a need for physical compactness and low cost.

According to one form of the present invention, a method and apparatus are provided to address this need, and involve periodically detecting an image of the area, identifying and tracking a moving object in a succession of the detected images, automatically selecting an image of each identified object, and saving the selected image of each identified object.

A different form of the present invention involves periodically detecting an image of the area, identifying and tracking a moving object in a succession of the detected images, and automatically saving information which identifies the path and movement of the object, the information being retained after the object is no longer present in the detected images.

BRIEF DESCRIPTION OF THE DRAWINGS

A better understanding of the present invention will be realized from the detailed description which follows, taken in conjunction with the accompanying drawings, in which:

FIG. **1** is a diagrammatic view of an automatic monitoring system which embodies the present invention;

FIGS. **2**A, **2**B, **2**C, **2**D, **2**E, **2**F, **2**G and **2**H are diagrammatic views of two-dimensional images that represent successive steps carried out by the system of FIG. **1** when processing images obtained from a video camera;

FIG. **3** is a motion analysis diagram indicating how the motion of objects in a video image is analyzed by the system of FIG. **1**;

FIG. **4** is a diagrammatic top view of part of a floor plan of a building in which the system of FIG. **1** can be utilized;

FIG. **5** is a diagrammatic view of a reference image provided by the system of FIG. **1** for the building of FIG. **4**;

FIG. **6** is a diagrammatic view of a video image which is similar to the image of FIG. **5**, but which shows the presence of a person;

FIG. **7** is a diagrammatic view of a directory structure which is used on a hard disk drive in the system of FIG. **1**;

FIG. **8** is a diagrammatic view of a display presented on the screen of a computer monitor which is a component of the system of FIG. **1**; and

FIG. **9** is a diagrammatic view similar to FIG. **8** of a display presented on the screen of the computer monitor of FIG. **1**.

DETAILED DESCRIPTION OF THE INVENTION

FIG. **1** is a diagrammatic view of a monitoring system **10** which embodies the present invention, and which is used to monitor activity in a selected region or area. The monitoring system **10** includes a camera unit **12** and a workstation **13**, which are operatively coupled through a network shown diagrammatically at **14**. The network **14** may be a local area network, the Internet, some other type of network, a modem link, or a combination of such technologies. The workstation **13** may be a personal computer, including a processor **17**, a keyboard **18**, a mouse **19**, and a display **21**.

The camera unit **12** includes a video camera **23** which, in the disclosed embodiment, is a monochrome camera. However, the present invention is also suitable for use with a color video camera, or some other type of two-dimensional image detector, such as an infrared detector. Video camera **23** includes a detector **24**, which may be a charge coupled device (CCD), or a CMOS image sensor device. The video camera **23** also includes not-illustrated optics of a known type, which focus an image on the detector **24**.

The camera unit **12** further includes an image processing section **27**. The image processing section **27** includes a video interface circuit **28** which receives the output of the detector **24**, and a network interface circuit **29** which facilitates communication across the network **14**. The image processing section **27** could also include a modem, in addition to or in place of the interface circuit **29**, in order to facilitate communication through telephone lines. The image processing section **27** further includes a processor **33**, and a memory such as a hard disk drive **34**. The hard disk drive **34** could optionally be replaced with some other type of suitable non-volatile memory, such as a flash memory, or a memory with battery backup.

In the disclosed embodiment, the image processing section **27** is physically disposed within the housing of the camera unit **12**. Thus, the camera unit **12** is a standalone device which can coupled directly to a telephone line or a network, such as the network **14**. However, it will be recognized that the image processing section **27** could alternatively be implemented with a personal computer which is physically separate from the video camera **23**, which has a plug-in video capture card serving as the video interface circuit, and which has a plug-in network interface card serving as the network interface circuit. Further, although the disclosed system has just one video camera **23**, it would be possible to use two or more video cameras with a single image processing section.

The initial processing of video images by the image processing section **27** will now be described with reference to FIGS. **2**A–**2**H and FIG. **3**. More specifically, FIG. **2**A is a diagrammatic view of a video image produced by the detector **24** when the video camera **23** is directed toward an area which, in this example, has arbitrarily been selected to be the corner of a room. The video image of FIG. **2**A is saved as a reference image. FIG. **2**B is a similar video image, obtained from the detector **24** at a later point in time, after an object has been introduced into the image. In this case, the object is a person **41**, who has walked into the corner of the room and thus into the field of view of the video camera **23**. The video camera **23** is stationary, and thus the single difference between the images of FIGS. **2**A and **2**B is the presence of the person **41** in FIG. **2**B. The presence and movement of the person **41** is detected in the following manner.

First, the monochrome or gray scale image of FIG. **2**B is subtracted from the gray scale image of FIG. **2**A, on a pixel-by-pixel-basis. The absolute value of the difference for each pixel is then determined, and the result is the gray scale difference image of FIG. **2**C. Then, the difference image of FIG. **2**C is sub-sampled in order to reduce the number of pixels, for example to a 128 by 128 or 256 by 256 pixel image. The resulting low-resolution image is shown in FIG. **2**D. It will be recognized that it is alternatively possible to sub-sample each of the images of FIGS. **2**A and **2**B before determining the difference and absolute value for each pixel, which reduces the number of pixels that need to be processed, and therefore reduces the amount of time needed to obtain the image of FIG. **2**D.

The low-resolution difference image of FIG. **2**D is then thresholded. In other words, the gray scale value for each pixel in the image of FIG. **2**D is compared to a predetermined threshold, and is then set to be either on or off (black or white), depending on whether the value is above or below the threshold. The resulting threshold image is shown in FIG. **2**E. Each pixel in the threshold image of FIG. **2**E can be represented by a binary "1" or a binary "0", depending on whether the pixel is considered to be on or off.

Morphological processing is then carried out on each pixel of the threshold image of FIG. **2**E, by first performing a dilate operation, and then performing an erode operation. More specifically, each pixel is processed by viewing it as the center pixel in a three-by-three matrix of pixels. During the dilate operation for each pixel in the threshold image of FIG. **2**E, if any one of the eight neighboring pixels in that image is a logic "1", the pixel of interest is set to a logic "1". The resulting dilate image is shown in FIG. **2**F. During the subsequent erode operation for each pixel in the dilate image of FIG. **2**F, if any one of the eight neighboring pixels in that image is a logic "0", then the pixel of interest is set to a logic "0". The result is the erode image of FIG. **2**G.

The erode image of FIG. **2**G is then analyzed to identify each region of contiguous logic "1" pixels. Each such region of contiguous logic "1" pixels represents a change region, corresponding to an object which has been introduced in the image of FIG. **2**B and which was not present in the image of FIG. **2**A, such as the person **41**. This analysis can be carried out using known techniques, such as run-length encoding followed by connected-component analysis.

With respect to each detected change region, the image processing section **27** determines a bounding box for the change region. An example of a bounding box is shown at **43** in FIG. **2**H. It will be noted that the bounding box **43** is a rectangular box, just large enough to contain the entire change region. That is, no pixel of the change region lies outside the box, but every side of the box touches at least one pixel of the change region.

The above-described image processing is carried out for each image in a succession of images provided by the video camera **23**. That is, each of these successive images is processed with respect to the reference image of FIG. **2**A, in the same manner that was described above for the image of FIG. **2**B.

The image processing system **27** then carries out motion analysis, by tracking movement or non-movement of each identified change region through a succession of the frames or images from the video camera. For purposes of facilitating an understanding of the present invention, one known motion analysis technique will be briefly summarized with reference to FIG. **3**. Although it will be recognized that motion analysis in the video images is carried out in two dimensions, for purposes of convenience the diagram of FIG. **3** shows just one dimension.

In FIG. **3**, the nineteen vertical lines F**0** through F**18** each represent a respective frame or image in a series of successive images from the video camera **23**. In FIG. **3**, the horizontal dimension represents time, and the vertical dimension represents one dimension of movement of an object within a two-dimensional image. When an object which was not previously present first appears, for example at **51** or **52**, it is identified as an "entrance" or "enter" event. When an object which was previously present is found to no longer be present, for example at **53** or **54**, it is designated an "exit" event. If an existing object splits into two objects, one of which is moving and the other of which is stationary, for example as at **57**, it is designated a "deposit" event. This would occur, for example, when a person who is carrying a briefcase sets it down on a table, and then walks away.

If a moving object merges with a stationary object, and then continues to move while the stationary object disappears, as at **58**, it is designated a "remove" event. This would correspond to a situation where a person walks to a notebook resting on a table, and then picks up the notebook and walks away. Three other types of events, which are not specifically illustrated in FIG. **3**, are a "rest" event, a "move" event, and a "lightsout" event. A rest event occurs when a moving object comes to a stop but continues to be present without moving. A practical example is a situation where the objects being monitored are vehicles in a parking lot, and a car pulls into a parking space and thereafter remains stationary. A move event occurs when a detected object which has been stationary begins moving again, for example when a car that has been parked begins moving. A "lightsout" event occurs when the entire detected image suddenly changes, for example when the lights in a monitored room are turned out and the room becomes dark. A "lightsout" event can be detected without all of the image processing described above in association with FIGS. **2** and **3**.

It is optionally possible to also carry out an identification analysis, in an attempt to identify a detected object. For example, with a small amount of knowledge about the topography of the monitored area, the image processing system **27** can use the position in the image of the midpoint of the lower side of the object's bounding box in order to identify how far the object is from the camera. Then, knowing how tall a person that far from the camera would be, the image processing system **27** can evaluate the vertical height of the bounding box in the image, in order to determine whether the object generating the change region is tall enough to be a person. If the object is sufficiently tall to be a person, it can be assumed that it is a person.

If the object is not sufficiently tall to be a person, then the image processing section **27** can carry out an object analysis procedure, where the image of the object is compared to stored images of common objects, such as briefcases, notebooks, boxes, and computer monitors. If the object is not be specifically identified through this approach, then it is ultimately identified as an "unknown" object.

In order to facilitate an understanding of the present invention, a specific exemplary application for the system **10** of FIG. **1** will now be disclosed. However, it will be recognized that there are numerous other applications and environments in which the system **10** of FIG. **1** could be utilized. With respect to the exemplary application, FIG. **4** is a diagrammatic top view of a portion of a building which has a long hallway **71** with an alcove **72** near one end. The camera unit **12** of FIG. **1** is stationarily mounted just below the ceiling and at one end of the hallway **71**, so that it looks down the hallway **71** and slightly to the right. The camera unit **12** can thus observe the hallway **71** and the alcove **72**. At its far end, the hallway **71** dead-ends into a transverse further hallway **73**. Yet another transverse hallway **74** extends off to the right from hallway **71**, at a location intermediate the alcove **72** and the hallway **73**. There are three doors **76**–**78** disposed at spaced locations along the left side of the hallway **71**. A single door **79** is provided along the right side of the hallway **71**, adjacent the hallway **74** and on a side thereof nearest the camera unit **12**.

FIG. **5** is a diagrammatic view of a video image which was obtained from the camera unit **12** in the environment of FIG. **4**, and which thus shows the hallway **71** and the alcove **72**. For purposes of discussion, it is assumed that the image of FIG. **5** has been saved as a reference image, analogous to the reference image discussed above in association with FIG. **2**A. FIG. **6** is a diagrammatic view of a further video image from the camera unit **12**, but after the appearance in the monitored area of an object **86** which was not present in the reference image of FIG. **5**.

In this case, the object **86** is a person, who entered the hallway **71** at the far end, and then walked down the length of the hallway **71** to the alcove **72**. After the camera unit generated the video image of FIG. **6**, the person **86** continued down the hallway **71** toward the camera unit **12**, and then walked under the camera unit so as to disappear from the field of view of the camera unit. During the time that the person **86** was in the field of view of the camera unit **12**, the camera unit generated a succession of video images as the person walked down the hall **71**. A selected one of these video images is shown in FIG. **6**. Each of the video images in this succession of images was processed relative to the reference image of FIG. **5**, in a manner analogous to that described above in association with FIG. **2**. In association with the processing of each such image, the system determines for each image a bounding box around the change region which corresponds to the person **86**. The bounding box for the person **86** in the image of FIG. **6** is shown at **87**.

The image processing section **27** of FIG. **1** does not save each of the numerous images of the person **86** which are obtained while the person walks down the hallway **71**. While some known systems do this, it requires an extensive amount of memory to store all this video information. Instead, the system **10** stores just selected information, as discussed below.

More specifically, the image processing section **27** has already stored on the hard disk drive **34** the reference image of FIG. **5**. In the disclosed embodiment, the reference image of FIG. **5** is first sub-sampled, and then the resulting low-resolution version of the image is stored on the hard disk drive **34**, in order to reduce the amount of storage space needed for each such reference image. Objects which enter the observed area are of primary interest, rather than the observed area itself, and a low-resolution image of the observed area is thus sufficient for most applications.

For each detected object such as the person **86**, the image processing section **27** also determines the Cartesian coordinates within each image of the midpoint of the lower side of the bounding box for that detected object. This information is saved on the hard disk drive. In other words, for each detected object, a Cartesian coordinate pair for that object is saved for each video image in which the object is present. As to a given object, the set of Cartesian coordinate pairs for all of the images in which that object was present can serve as a trace of the movement of the object within the observed area, as will be discussed in more detail later.

The image processing section **27** also saves a selected image of each detected object. In the disclosed embodiment, this selected image is just a portion of the overall image from the video camera **23**. In particular, it is the portion of the image which is located within the bounding box for the object of interest. Thus, if the selected image for the person **86** was derived from the video image of FIG. **6**, it would be the portion of that image within the bounding box **87**. This selected image or image portion is stored at full resolution, in order to have a top-quality view of the detected object. This is because a top-quality view will often be useful at a later point in time, for example to facilitate identification of a particular individual. Since the selected image is just a portion of the overall video image, the amount of memory needed to store the selected image at full resolution is often less than the amount of memory which would be needed to store the overall video image at a reduced resolution.

The selection of the particular image to be saved is an automatic determination, which is effected with simple heuristics. In most applications, the objects of primary interest are humans, and it is therefore desirable to favor selection of an image in which the person is facing generally toward the camera unit **12**, and is reasonably close to the camera unit **12**. In this regard, if the lower side of the bounding box is moving downwardly in successive images, it is assumed that the person is moving toward and facing the camera. On the other hand, if the lower side of the bounding box is not moving downwardly or upwardly, the new view will nevertheless be favored over a prior view, if the subject appears to be larger, as reflected by an increase in the vertical size of the bounding box.

Thus, when an object such as a person first appears, the image processing system **27** temporarily saves the first video image containing the person, and tentatively designates this image as the selected image. Then, in each successive image, the image processing section **27** checks to see whether the lower side of the bounding box in the current image is lower than the lower side of the bounding box in the tentatively selected image. If it is, then the prior image is discarded and the current image is tentatively designated as the selected image.

On the other hand, if the lower side of the bounding box for the object is found to have the same vertical position in the current image as in the tentatively selected prior image, then the section **27** checks to see if the vertical height of the bounding box in the current image is larger than the vertical height of the bounding box in the tentatively selected image. If so, then the prior image is discarded and the current image is tentatively designated as the selected image.

When the object eventually exits the observed area, the image processing section **27** takes the tentatively selected video image, and saves on the hard disk drive **34** the portion of that video image which is within the bounding box. As discussed above, this portion of the image is saved at full resolution.

Although the disclosed embodiment uses the foregoing selection criteria in order to favor facial close-ups of humans, it will be recognized that other applications may require other selection criteria. For example, if the camera unit **12** was being used to monitor vehicles, and if it was desirable to favor close-ups of the rear license plates of the vehicles, the selection criteria could be adjusted to achieve this.

In association with each detected object, the image processing section **27** also saves on the hard disk drive **34** certain other information, including a human-readable timestamp which indicates the date and time that the object was detected, the name of the disk file containing the reference image which was in use while the object was present in the observed area, and a keyword indicating how the object entered the observed area. As to the latter, the allowable keywords in the disclosed embodiment are “enter”, “deposit” and “other”, but it will be recognized that there could be additional allowable keywords, or fewer allowable keywords.

Over time, changes may occur in the background of the observed area. For example, the ambient lighting may change, due to variations in the sunlight entering through windows, opening and closing of window blinds, opening and closing of interior doors, actuation and deactuation of interior lighting, and so forth. Similarly, people may deposit, remove or reposition objects in the observed area. Each such change creates a permanent region of difference between the original reference image and each current video image. Absent a periodic update of the reference image, the system will continue to track these difference or change regions as detected objects. Lighting changes would thus be treated as detected objects, resulting in the storage of images which are not really of interest, and which simply waste memory on the hard disk drive **34**.

In order to avoid this, the image processing section **27** checks for a condition in which nothing in the observed area has changed for a specified time interval, such as twenty seconds. In response to detection of this condition, the image processing section **27** terminates the tracking of all detected objects which were being actively tracked, saves the current video image as a new reference image, and then resumes monitoring of the observed area using the new reference image. In general, humans almost never remain completely still for more than a second or two, and there is thus little risk of selecting as the reference image a video image which has a human in it.

With reference to FIG. **1**, the image processing section **27** of the camera unit **12** has been designed so that it is Internet-compatible, and in particular is compatible with Internet standards commonly known as the World Wide Web (WWW). As a result, the camera unit **12** can be coupled directly to the network **14**, and the stored information which was discussed above can be accessed and viewed by a person using a web browser on a remote unit such as the workstation **13**. To facilitate this, the image processing section **27** stores the results of its monitoring activities on the hard disk drive **34** in a manner which will now be described with reference to FIG. **7**.

More specifically, FIG. **7** shows the directory organization of a portion of the hard disk drive **34**. In FIG. **7**, the rectangular boxes **91**–**96** are each a diagrammatic representation of respective directory. These directories store the information relating to monitoring activities of the image processing section **27**. The directory **91** is a subdirectory of a not-illustrated root directory, the directories **92**–**94** are subdirectories of the subdirectory **91**, and the directories **95** and **96** are subdirectories of the directory **94**.

The subdirectory **91** contains a file MASTER.HTML, and the subdirectories **92** and at **93** each contain a respective file named LOGLIST.HTML. The MASTER.HTML and LOGLIST.HTML files are each a WWW-compatible file in hypertext mark-up language (HTML) format, and facilitate access to other information stored in the directory structure of FIG. **7**. The MASTER.HTML file has hypertext links to each of the LOGLIST.HTML files, and the LOGLIST.HTML files are each an HTML shell which invokes an applet that facilitates access to files within the directory containing that particular LOGLIST.HTML file.

The directory **92** corresponds to a single day in which the camera unit **12** of FIG. **1** was operational. When the camera unit **12** first begins monitoring a given area, the subdirectory **91** exists, but the subdirectories **92** and **93** do not exist. During the first day of monitoring, the image processing section **27** creates the subdirectory **92**, and uses it to store information from that day's monitoring activities. Upon commencing each subsequent day of monitoring, the image processing section **27** creates a similar additional subdirectory, one of which is shown at **93**. The name of each such subdirectory is in the format MM.DD.YY, and identifies the month, day and year for which the directory contains information.

Each of the subdirectories **92**–**93** has therein the abovementioned LOGLIST.HTML file. Further, each such subdirectory includes a LOGLIST file, which is a summary list identifying all the log entries for the day in question, each log entry corresponding to a respective detected object. Each subdirectory also includes, for each log entry in its

LOGLIST file, a separate file with the name format ENTRYX, where X is an integer. Each ENTRYX file contains details associated with the specific detected object, including the name of the file which contains the reference image that was in effect when the object was present, the keyword indicating how the object entered the scene, the series of Cartesian coordinate pairs which trace the path of movement of the object within the image, the selected image of the object in a full-resolution PGM image format, and two Cartesian coordinate pairs which respectively identify the position in the video image of two opposite corners of the bounding box for the selected image.

The summary information in the LOGLIST file includes two elements for each detected object, namely a timestamp representing the date and time when the corresponding object was detected, and the name of the ENTRYX file containing details about that detected object. In the disclosed embodiment, this information in the LOGLIST file is in an ASCII format.

The subdirectories shown at **95** and **96** in FIG. **7** each correspond to a respective day, and each contain all of the reference images used during that day. More specifically, when the camera unit **12** first begins monitoring a selected area, the subdirectory **94** will exist, but the subdirectories **95** and **96** will not yet exist. During the first day of monitoring, the subdirectory **95** is created, and is used to store all of the reference images for that day. At the beginning of each subsequent day of monitoring, a new subdirectory is created, one of which is shown at **96**.

Each of the subdirectories **95** and **96** has a name format of MM.DD.YY, representing the date corresponding to the information stored in the subdirectory. Each of the subdirectories **95** and **96** contains a plurality of files with the name format REFIMGXX.PGM, where XX is a unique integer. Each REFIMGXX.PGM file contains a respective reference image. Each time a new reference image is saved during the day, a new REFIMGXX.PGM file is created, and is named using the next highest unused XX integer.

FIG. **8** is a diagrammatic view of the display **21** of FIG. **1** when an operator is using the workstation **13** to observe information stored on the hard disk drive **34** by the image processing section **27**. In FIG. **8**, the operator is using a web browser program which is sold under the tradename NETSCAPE by Netscape Communications Corporation of Mountainview, Calif. However, it will be recognized that some other equivalent web browser could alternatively be used. In FIG. **8**, the user has invoked the WWW capabilities of the Internet to access the WWW-compatible file MASTER.HTML in the directory **91** (FIG. **7**), which in turn has used the various LOGLIST.HTML files in the subdirectories **92**–**93** to access information in each of the respective LOGLIST files. The MASTER.HTML file may optionally require an operator to provide a valid password before giving the operator access to the information stored on the hard disk drive **34**.

At the top of the displayed web page is a title **101**, which is provided by the MASTER.HTML file, and which reflects the particular installation or application. Along the left side of the page is a scroll box **102**, in which the MASTER.HTML and LOGLIST.HTML files display a list of the timestamps from all of the LOGLIST files, each timestamp including both a date and a time. Vertical and horizontal scroll bars **103** and **104** are provided if the number of timestamp entries or the length of any single timestamp entry is larger than can be displayed at one time within the scroll box **102**. In the scroll box **102**, the operator has highlighted one entry, which corresponds to a detected object that was present at the specified time on Feb. 15, 1997.

To the right of the scroll box **102**, information from the ENTRYX file corresponding to the selected log entry is displayed. More specifically, a video image **111** is presented, which represents the event that was discussed above in association with FIGS. **5** and **6**, namely the detection and tracking of the person **86**. The image **111** is created by first retrieving and displaying the REFIMGXX.PGM file corresponding to the selected log entry **107**. Then, the selected image corresponding to the log entry **107** is retrieved from the ENTRYX file, sub-sampled so as to have the same resolution as the reference image, and displayed in place of the corresponding portion of the reference image. Thereafter, the bounding box **87** associated with the selected image is superimposed on image **111**.

Then, using the series of Cartesian coordinate pairs stored in the corresponding ENTRYX file, a trace **113** of the movement of the detected object is overlaid on the image **111**. As discussed above, the trace **113** represents the movement of the midpoint of the lower side of the bounding box **87**, and thus is an accurate representation of where the person **86** walked. Then, labels are superimposed on the image **111**, as at **116** and **117**, based on the information stored in the ENTRYX file. In FIG. **8**, the label **116** is the word "ENTER", and indicates that the person **86** entered the observed area at approximately the location of this label, or in other words at the far end of the hallway **71**. The label **117** is the word "EXIT", and indicates where the person **86** exited the observed area, in this case by continuing down the hallway **71** and underneath the camera unit **12**. The bounding box **87**, trace **113** and/or labels **116** and **117** may optionally be displayed in one or more different colors, so that they are more readily visible.

To the right of the scroll box **102**, and below the image **111**, the image processing section **27** displays a further image **121**, which is smaller than the image **111**. The image **121** corresponds to the portion of the image **111** within the bounding box **87**, but is displayed at full resolution rather than at the lower resolution used for the larger image **111**. Thus, if an attempt is being made to identify a particular person, the features of that person may be more clearly visible in the high resolution image **121** than in the reduced resolution image **111**. Since the saved image **121** was selected using the criteria discussed above, which are intended to favor facial close-ups of humans, it will be noted that the face of the person **86** is visible, and that the person is closer to the camera than would have been the case if the system had simply stored the first image in which the person **86** had been detected, without attempting to apply any selection criteria.

FIG. **9** is a diagrammatic view similar to FIG. **8**, but showing a different web page provided by the MASTER.HTML file. This web page includes an image **131**, which is the current reference image, for example the reference image shown in FIG. **5**. The user can then use a mouse to identify one or more regions in this image, for example the region **132**. The user may define the region by using the mouse pointer to identify the corners of the region, while clicking on each corner. Each time the user defines a region, it is automatically given a label, which is a letter. For example, the region **132** in FIG. **9** has been given the label "A". As discussed above, the image processing section **27** maintains a history of the movement of the midpoint of the lower side of the bounding box for each object. If this midpoint were to remain within a given region, such as the region **132**, for a predefined period of time, it might represent loitering, and could be detected by the image processing section **27**.

The web page of FIG. **9** also includes an event selection box **136**, which the operator can use to indicate that the imaging processing section **27** is to check for a specified event, and to indicate what action is to be taken if the specified event occurs. In this regard, the operator can use a mouse to select one of several events identified in box **136**, including an enter event, an exit event, a loiter event, a deposit event, a remove event, a move event, a rest event, and a lightsout event. The event selection box **136** allows the user to optionally restrict the monitoring for the specified event to certain types of detected objects, including a person, a box, a briefcase, a notebook, a computer monitor, any type of object, or just an unknown object. Event selection box **136** also allows the user to restrict the monitoring event to a particular region by identifying its label letter, such as the region **132** identified by the label letter "A".

For certain events, the event selection box **136** allows the user to specify a time duration in seconds. For example, if the user is instructing the system to monitor for a loiter event within a specified region, the user may specify that the loiter event is to be detected only if the specified object remains within the specified region for a period of at least five seconds. The event selection box **136** also allows the operator to specify the action to be taken if the specified event occurs, including an audible beep, the creation of a log entry on the hard disk drive **34**, a pop-up window on the display **21** of the workstation **13**, or a synthesized voice announcement which indicates that the event of interest has occurred, such as a synthesized announcement of the word "loiter". It will be recognized that the event selection box **136** could be modified to allow the identification of other events, objects, conditions, or actions. For example, actions could also include making a phone call to a specified number such as that of a security agency, or sending an electronic mail message to a specified electronic mail address.

The present invention provides a number of technical advantages. One such advantage is that, by periodically saving reference images, by saving these reference images at a reduced resolution, by saving just selected images of objects of interest, and by saving just portions of the overall image, the amount of memory needed to store images is greatly reduced in comparison to known systems. A related advantage is that the amount of stored information which an operator would have to review in response to the occurrence of an event is greatly reduced in comparison to known systems. A further advantage is that the available information is presented with timestamp information, so that an operator can rapidly identify the events of interest within a time frame of interest, and can quickly and easily review those events.

Yet another advantage is the storage of a trace representing the movement of a detected object, so as to later provide a readily understandable visible image of the object's movement, without storing numerous video images corresponding to the entire time interval while the detected object was present in an observed area. Another advantage is that the use of a web browser to access information logged by the system permits a person to access the information from virtually anywhere that a computer is available, including a WWW-compatible cellular phone.

Another advantage results from the fact that the selection of an image to save is based on criteria which are intended to optimize the image, for example to make it likely that a detected person is facing and close to the camera. Another advantage is that the disclosed system can be a self-contained camera unit which is WWW-compatible. A further advantage is that the disclosed system is more reliable than certain known technologies, such as known systems having a video cassette recorder (VCR) that is subject to mechanical breakdowns and that has heads which need to be periodically cleaned.

Although one embodiment has been illustrated and described in detail, it will be understood that various changes, substitutions and alternations can be made thereto without departing from the scope of the present invention. For example, as mentioned above, the disclosed embodiment has a camera unit which includes both a video camera and image processing circuitry, but a similar system could be implemented with a video camera and a physically separate personal computer. Further, the disclosed embodiment has one video camera, but it will be recognized that a single image processing circuit could support two or more video cameras.

In addition, the disclosed embodiment has been discussed in the context of one specific exemplary application, which involves the monitoring of activity in a hallway. However, there are many other applications to which the present invention could be applied. For example, a working couple might place a camera unit in their home, and could use the Internet to consult its LOGFILES from work, in order to verify that their children arrived safely home from school. A camera unit located over the front door of a residence could store pictures of everyone who comes to the door and, like a telephone answering machine, would give the owners a log of who tried to contact them while they were away. A system at a vacation home could telephone the owner and send an image of someone who is in the home, so that the owner could inspect the image and take the appropriate action.

A system located at a traffic intersection could store one or more selected images covering an automobile accident. For example, if it was detected through motion analysis that any vehicle decelerated more rapidly than would be possible by braking, it could be interpreted as a possible accident, and the system could respond by storing a selected image from a point in time approximately one-half second before the accident. In fact, the system could select and save several different images from a time interval just before the accident.

Although the disclosed embodiment uses a selection criteria optimized for detecting humans, a different image selection criteria could be used for optimal results in other applications. For example, if the system were monitoring a parking lot and it was desirable to store a selected image showing a license plate on the rear of a vehicle, the criteria would favor images in which the vehicle was close to the camera but moving away from the camera rather than moving toward the camera. Although the disclosed embodiment stores reference images and selected images with different resolutions, it will be recognized that all imagescould be stored with the same resolution.

Other changes, substitutions and alterations are possible without departing from the spirit and scope of the present invention, as defined by the following claims.

What is claimed is:

1. A method of monitoring an area, comprising the steps of:

periodically detecting an image of the area;

identifying and tracking a moving object in a succession of the detected images;

automatically selecting a portion of a single image of the succession of detected images for each identified moving object using selection criteria;

saving the selected portion of the single image of the succession of detected images for each identified object; and

discarding and not saving detected images of the succession of the detected images other than said single image of each identified object.

2. A method according to claim **1**, wherein said step of automatically selecting is carried out by using image selection criteria which are intended to lead to the selection of an image in which the face of a detected person is visible and large.

3. A method according to claim **2**, wherein said step of automatically selecting includes the steps of:

saving one of the detected images as a reference image;

carrying out said step of identifying by evaluating images detected subsequent to the reference image in order to identify therein each change region where the evaluated image differs from the reference image;

determining a bounding box subset of the single image for a given change region in each image of a set of images in which the given change region appears; and

selecting the selected portion of the single image for the given change region by discarding images from the set in which a lowermost side of the bounding box is higher than in other images of the set, and by selecting from the remaining images of the set an image in which a size of the bounding box is larger than in the other remaining images of the set.

4. A method according to claim **1**, wherein said step of automatically selecting is carried out using image selection criteria which cause a current image to be selected over a prior image if a lowermost point of a detected change region is lower in the current image than in the prior image.

5. A method according to claim **4**, wherein said step of automatically selecting is carried out using image selection criteria which cause a current image to be selected over a prior image if a detected change region has increased in size relative to a prior image.

6. A method according to claim **1**, wherein said selecting step is carried out in response to detection of the absence of a previously detected object.

7. A method according to claim **1**, wherein said selecting step is carried out in response to detection of a situation in which an object has remained within a predefined region of the area for a specified time interval.

8. A method according to claim **1**, wherein said selecting step is carried out in response to a determination that a previously moving object has become stationary.

9. A method according to claim **1**, wherein said selecting step is carried out in response to a determination a previously stationary object has started moving.

10. A method according to claim **1**, wherein said saving step is carried out by determining a bounding box subset of the single image just large enough to completely contain a corresponding detected object and saving a portion of a detected image corresponding to the bounding box.

11. A method according to claim **10**, including the step of saving one of the detected images as a reference image at a first resolution, and wherein said step of saving the selected portion of the single image is carried out by saving the bounding box enclosing the selected portion of the single image at a second resolution which is higher than the first resolution.

12. A method according to claim **1**, including the step of saving one of the detected images as a reference image having a first resolution, wherein said step of saving the selected portion of the single image is carried out by determining a bounding box subset of the single image just large enough to completely contain a corresponding detected object and saving at a second resolution the bounding box enclosing the selected portion of the single image, the second resolution being greater than the first resolution, and including the step of displaying the reference image at the first resolution, displaying the bounding box enclosing the selected portion of the single image within the reference image at the first resolution, and displaying the bounding box enclosing the selected portion of the single image separately from the reference image and at the second resolution.

13. An apparatus for monitoring an area, comprising:

a detector which is operative to periodically detect an image of the area; and

an image processing section which is responsive to the detector, said image processing section being operative to:

identify and track a moving object in a succession of the detected images;

automatically select a portion of a single image of the succession of detected images for each identified object utilizing selection criteria;

save the selected portion of the single image of the succession of detected images for each identified object; and

discard and not save detected images other than said single image of the succession of detected images for each identified object.

14. An apparatus according to claim **13**, wherein:

said image processing section being further operative to:

use image selection criteria which are intended to lead to the selection of an image in which the face of a detected person is visible and large.

15. An apparatus according to claim **14**, wherein:

said image processing section being further operative to:

save one of the detected images as a reference image;

identify a moving object by evaluating images detected subsequent to the reference image in order to identify therein each change region where the evaluated image differs from the reference image;

determine a bounding box subset of the selected image for a given change region in each image of a set of images in which the given change region appears; and

select the selected portion of the single image for the given change region by discarding images from the set in which a lowermost side of the bounding box is higher than in other images of the set, and by selecting from the remaining images of the set an image in which a size of the bounding box is larger than in the other remaining images of the set.

16. An apparatus according to claim **13**, wherein:

said image processing section being further operative to:

automatically select an image using image selection criteria which cause a current image to be selected over a prior image if a lowermost point of a detected change region is lower in the current image than in the prior image.

17. An apparatus according to claim **16**, wherein:

said image processing section being further operative to:

automatically select an image out using image selection criteria which cause a current image to be selected over a prior image if a detected change region has increased in size relative to a prior image.

18. An apparatus according to claim **13**, wherein:

said image processing section being further operative to:

select an image in response to detection of the absence of a previously detected object.

19. An apparatus according to claim **13**, wherein:

said image processing section being further operative to:

select an image in response to detection of a situation in which an object has remained within a predefined region of the area for a specified time interval.

20. An apparatus according to claim **13**, wherein:

said image processing section being further operative to:

select an image in response to a determination that a previously moving object has become stationary.

21. An apparatus according to claim **13**, wherein:

said image processing section being further operative to:

select an image in response to a determination a previously stationary object has started moving.

22. An apparatus according to claim **13**, wherein:

said image processing section being further operative to:

save said selected portion of the single image by determining a bounding box subset of the single image just large enough to completely contain a corresponding detected object and saving a portion of a detected image corresponding to the bounding box.

23. An apparatus according to claim **22**, wherein:

said image processing section being further operative to:

save one of the detected images as a reference image at a first resolution; and

save the selected portion of the single image by saving a bounding box enclosing the selected portion of the single image at a second resolution which is higher than the first resolution.

24. An apparatus according to claim **13**, further comprising:

a display device; and

wherein said image processing section being connected to the display device and being further operative to:

save one of the detected images as a reference image having a first resolution;

save the selected portion of the single image by saving a bounding box subset of the single image enclosing a corresponding detected object at a second resolution which is higher than the first resolution;

display via said display device said reference image at the first resolution and said bounding box enclosing the selected portion of the single image within said reference image at said first resolution, and

display via said display device said bounding box separately from said reference image at said second resolution.

25. A method of monitoring an area, comprising the steps of:

periodically detecting an image of the area;

identifying and tracking a moving object in a succession of the detected images;

automatically selecting a portion of single image of the succession of detected images for each identified object using selection criteria;

saving the selected portion of the single image of the succession of detected images for each identified object; and

discarding and not saving detected images other than said single image of the succession of detected images for each identified object; and

automatically saving a series of Cartesian coordinate pairs which identifies the path of movement of the object, said information being retained after the object is no longer present in newly detected images.

26. A method according to claim **25**, including the steps of saving an identification of an event associated with the detected object, saving one of the detected images as a reference image, displaying the reference image, displaying on the reference image the path of movement of the object, and displaying on the reference image the identification of the event at a location on the reference image corresponding to a location of the event.

27. An apparatus according to claim **25**, further comprising:

a display device;

wherein said image processing section being connected to said display device and being further operative to:

save an identification of an event associated with said detected object;

save one of the detected images as a reference image; and

display via said display device said reference image, said path of movement of the object within said reference image, and said identification of said event on said reference image at a location on the reference image corresponding to a location of the event.

28. An apparatus for monitoring an area, comprising:

a detector which is operative to periodically detect an image of the area; and

an image processing section which is responsive to the detector and which is operative to:

identify and track a moving object in a succession of the detected images;

save the selected portion of the single image of the succession of detected images for each identified object; and

discard and not save detected images other than said single image of the succession of detected images for each identified object;

automatically save a series of Cartesian coordinate pairs which identifies the path of movement of each moving object, and to retain the information after the moving object ceases to be present in current detected images.

* * * * *

#3 S. Hoover
6/11/99 #2

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

jc525 U.S. PTO
09/292265
04/15/99

In re the Application of: TI-25771

Thomas J. Olson

Serial No:

Filed: April 15, 1999

For: Automatic Video Monitoring System Which Selectively Saves Information

INFORMATION DISCLOSURE STATEMENT

EXPRESS MAILING" Mailing Label No. EL255678769. Date of Deposit: **April 15, 1999**. I hereby certify that this paper is being deposited with the U.S. Postal Service Express Mail Post Office to Addressee Service under 37 CFR 1.10 on the date shown above and is addressed to: Ass't Commissioner for Patents, Washington, D.C. 20231.

Robin E. Barnum

Ass't Commissioner for Patents
Washington, DC 20231

Dear Sir:

Applicant wishes to bring to the attention of the Patent and Trademark Office the information noted on the enclosed PTO-1449. Copies of the noted references are enclosed herewith, except where noted (2 references are previously filed pending patent application).

Please charge Deposit Account 20-0668 for the necessary fees. **This form is submitted in triplicate**.

Respectfully submitted,

Robert D. Marshall, Jr.
Attorney for Applicant
Reg. No. 28,527

Texas Instruments Incorporated
P.O. Box 655474, MS 3999
Dallas, TX 75265
(972) 917-5290

#2

PTO/SB/08 (2-92)
Sheet 1 of 5

Form PTO-1449 **INFORMATION DISCLOSURE CITATION IN AN APPLICATION** *(Use several sheets if necessary)*	Docket Number (Optional) TI-25771	Application Number 09/292,265
	Applicant Thomas J. Olson	
	Filing Date ~~04/30/98~~ 4/15/99	Group Art Unit 2613

U.S. PATENT DOCUMENTS

EXAMINER INITIAL	DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE
AW	4 9 4 3 8 5 4	07/24/90	Shiota, *et al.*	358	108	
AW	5 1 1 1 2 9 1	05/05/92	Erickson, *et al.*	358	108	
AW	5 4 9 1 5 1 1	02/13/96	Odle	349	153	

FOREIGN PATENT DOCUMENTS

	DOCUMENT NUMBER	DATE	COUNTRY	CLASS	SUBCLASS	Translation YES	Translation NO

OTHER DOCUMENTS *(Including Author, Title, Date, Pertinent Pages, Etc.)*

AW	☐	Patent Application Serial No. 08/795,423, filed February 5, 1997 (Attorney Ref. TI-21441) NOT INCLUDED
AW	☐	Patent Application Serial No. 08/866,789, filed May 30, 1997 (Attorney Ref. TI-22548) NOT INCLUDED
AW	☐	Jonathan D. Courtney, "Automatic Video Indexing via Object Motion Analysis", *Pattern Recognition*, April, 1997, cover page and pp 1-31.

EXAMINER	DATE CONSIDERED 2/4/02

EXAMINER: Initial if citation considered, whether or not citation is in conformance with MPEP § 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to the applicant.

Form PTO–1449 **INFORMATION DISCLOSURE CITATION IN AN APPLICATION** *(Use several sheets if necessary)*	Docket Number (Optional) TI-25771	Application Number
	Applicant Thomas J. Olsen	
	Filing Date ~~April 30, 1998~~ 4/15/99	Group Art Unit

U.S. PATENT DOCUMENTS

EXAMINER INITIAL	DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE

FOREIGN PATENT DOCUMENTS

	DOCUMENT NUMBER	DATE	COUNTRY	CLASS	SUBCLASS	Translation	
						YES	NO

OTHER DOCUMENTS *(Including Author, Title, Date, Pertinent Pages, Etc.)*

AW	Niyogi, *et al.*, "Analyzing and Recognizing Walking Figures in XYT", 1994 IEEE, pp. 469-474.
AW	Wren, *et al.*, "Pfinder: Real-Time Tracking of the Human Body", M.I.T. Media Laboratory Perceptual Computing Section Technical Report No. 353, published in SPIE 1995 Vol. 2615, pp. 1-9.
AW	Turk, *et al.*, "Eigenfaces for Recognition", 1991 Massachusetts Institute of Technology, Journal of Cognitive Neuroscience Volume 3, Number 1, pp. 71-86.

EXAMINER	DATE CONSIDERED 2/4/02

EXAMINER: Initial if citation considered, whether or not citation is in conformance with MPEP § 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to the applicant.

Form PTO–1449 **INFORMATION DISCLOSURE CITATION IN AN APPLICATION** *(Use several sheets if necessary)*	Docket Number (Optional) TI-25771	Application Number
	Applicant Thomas J. Olson	
	Filing Date ~~04/30/98~~ 4/15/99	Group Art Unit

U.S. PATENT DOCUMENTS

EXAMINER INITIAL	DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE

FOREIGN PATENT DOCUMENTS

	DOCUMENT NUMBER	DATE	COUNTRY	CLASS	SUBCLASS	Translation	
						YES	NO

OTHER DOCUMENTS *(Including Author, Title, Date, Pertinent Pages, Etc.)*

Aw		Flinchbaugh, *et al.*, "Autonomous Scene Monitoring System", pp. 205-209.
Aw		Norris, *et al.*, "Algorithmic Surveillance-The future of automated visual surveillance", CCTV, Surveillance and Social Control Conference, July 9, 1996, pp. 1-21.
Aw		Jonathan D. Courtney, "Automatic Object-Based Indexing for Assisted Analysis of Video Data", (1995) pp. 1-25.

EXAMINER	DATE CONSIDERED 2/4/02

EXAMINER: Initial if citation considered, whether or not citation is in conformance with MPEP § 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to the applicant.

PTO/SB/08 (2-92)
Sheet 4 of 5

Form PTO-1449

INFORMATION DISCLOSURE CITATION IN AN APPLICATION

(Use several sheets if necessary)

Docket Number (Optional): TI-25771
Application Number:
Applicant: Thomas J. Olsen
Filing Date: ~~04/30/98~~ 4/15/99
Group Art Unit:

U.S. PATENT DOCUMENTS

EXAMINER INITIAL	DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE

FOREIGN PATENT DOCUMENTS

	DOCUMENT NUMBER	DATE	COUNTRY	CLASS	SUBCLASS	Translation	
						YES	NO

OTHER DOCUMENTS *(Including Author, Title, Date, Pertinent Pages, Etc.)*

AW	Bruce Flinchbaugh, "Robust Video Motion Detection and Event Recognition", May 11, 1997, pp. 51-54.
AW	Kashi Rao, "A Computer Vision System to Detect 3-D Rectangular Solids", 6 unnumbered pages.
AW	Jain, *et al.*, "Segmentation through the Detection of Changes Due to Motion", Computer Graphics and Image Processing 11, (1979), pp. 13-34.

EXAMINER:
DATE CONSIDERED: 2/4/02

EXAMINER: Initial if citation considered, whether or not citation is in conformance with MPEP § 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to the applicant.

PTO/SB/08 (2-92)
Sheet 5 of 5

Form PTO-1449 **INFORMATION DISCLOSURE CITATION IN AN APPLICATION** *(Use several sheets if necessary)*	Docket Number (Optional) TI-25771	Application Number
	Applicant Thomas J. Olsen	
	Filing Date ~~04/30/98~~ 4/15/99	Group Art Unit

U.S. PATENT DOCUMENTS

EXAMINER INITIAL	DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE

FOREIGN PATENT DOCUMENTS

	DOCUMENT NUMBER	DATE	COUNTRY	CLASS	SUBCLASS	Translation	
						YES	NO

OTHER DOCUMENTS *(Including Author, Title, Date, Pertinent Pages, Etc.)*

AW	"DTS-1000 DTS-1000VB Video Motion Detection & Tracking System" Stellar Security Products Incorporated, 4 unnumbered pages, 1995.
AW	Olson, *et al.*, "Moving Object Detection and Event Recognition Algorithms for Smart Cameras", May 11, 1997, pp. 159-175.

EXAMINER	DATE CONSIDERED 2/4/02

EXAMINER: Initial if citation considered, whether or not citation is in conformance with MPEP § 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to the applicant.

EXHIBIT F-2

A System for Video Surveillance and Monitoring [*]

Robert T. Collins, Alan J. Lipton and Takeo Kanade
Robotics Institute, Carnegie Mellon University, Pittsburgh, PA
E-MAIL: {rcollins,ajl,kanade}@cs.cmu.edu PHONE: 412-268-1450
HOMEPAGE: http://www.cs.cmu.edu/~vsam

Abstract

The Robotics Institute at Carnegie Mellon University (CMU) and the Sarnoff Corporation are developing a system for autonomous Video Surveillance and Monitoring. The technical objective is to use multiple, cooperative video sensors to provide continuous coverage of people and vehicles in cluttered environments. This paper presents an overview of the system and significant results achieved to date.

1 Introduction

The DARPA Image Understanding (IU) program is funding basic research in the area of Video Surveillance and Monitoring (VSAM) to provide battlefield awareness. The thrust of CMU's VSAM research is to develop automated video understanding algorithms that allow a network of active video sensors to automatically monitor objects and events within a complex, urban environment. We have developed video understanding technology that can automatically detect and track multiple people and vehicles within cluttered scenes, and to monitor their activities over long periods of time. Human and vehicle targets are seamlessly tracked through the environment using a network of active sensors to cooperatively track targets over areas that cannot be viewed continuously by a single sensor alone. Each sensor transmits symbolic events and representative imagery back to a central operator control station, which provides a visual summary of activities detected over a broad area. The user interacts with the system using an intuitive map-based interface. For example, the user can specify that objects entering a region of interest should trigger an alert, relieving the burden of continually watching that area. The system automatically allocates sensors to optimize system performance while fulfilling user commands.

Although developed within a context of providing battlefield awareness, we believe this technology has great potential for applications in remote monitoring of nuclear facilities. Sample tasks that could be automated are verification that routine maintainance activities are being performed according to schedule, logging and tracking visitors and personnel as they enter and move through the site, and providing security against unauthorized intrusion. Other applications in military and law enforcement scenarios include providing perimeter security for troops, monitoring peace treaties or refugee movements using unmanned air vehicles, providing security for embassies or airports, and staking out suspected drug or terrorist hide-outs by collecting time-stamped pictures of everyone entering and exiting the building.

The following sections present an overview of video surveillance algorithms developed at CMU over the last two years (Section 2) and their incorporation into a prototype system for remote surveillance and monitoring (Section 3).

---

[*] This work is funded by the DARPA IU program under VSAM contract number DAAB07-97-C-J031.

2 Video Understanding Technologies

Keeping track of people, vehicles, and their interactions in a complex environment is a difficult task. The role of VSAM video understanding technology in achieving this goal is to automatically “parse” people and vehicles from raw video, determine their geolocations, and automatically insert them into a dynamic scene visualization. We have developed robust routines for detecting moving objects (Section 2.1) and tracking them through a video sequence (Section 2.2) using a combination of temporal differencing and template tracking. Detected objects are classified into semantic categories such as human, human group, car, and truck using shape and color analysis, and these labels are used to improve tracking using temporal consistency constraints (Section 2.3). Further classification of human activity, such as walking and running, has also been achieved (Section 2.4). Geolocations of labeled entities are determined from their image coordinates using either wide-baseline stereo from two or more overlapping camera views, or intersection of viewing rays with a terrain model from monocular views (Section 2.5). The computed geolocations are used to provide higher-level tracking capabilities, such as tasking multiple sensors with variable pan, tilt and zoom to cooperatively track an object through the scene (Section 2.6).

2.1 Moving Target Detection

The initial stage of the surveillance problem is the extraction of moving targets from a video stream. There are three conventional approaches to automated moving target detection: temporal differencing (two-frame or three-frame) [Anderson *et al.*, 1985]; background subtraction [Haritaoglu *et al.*, 1998, Wren *et al.*, 1997]; and optical flow (see [Barron *et al.*, 1994] for an excellent discussion). Temporal differencing is very adaptive to dynamic environments, but generally does a poor job of extracting all relevant feature pixels. Background subtraction provides the most complete feature data, but is extremely sensitive to dynamic scene changes due to lighting and extraneous events. Optical flow can be used to detect independently moving targets in the presence of camera motion; however, most optical flow computation methods are very complex and are inapplicable to real-time algorithms without specialized hardware.

The approach presented here is similar to that taken in [Grimson and Viola, 1997], and is an attempt to make background subtraction more robust to environmental dynamism. The key idea is to maintain an evolving statistical model of the background to provide a mechanism that adapts to slow changes in the environment. For each pixel value p_n in the n^{th} frame, a running average $\overline{p}_n$ and a form of standard deviation σ_{p_n} are maintained by temporal filtering, implemented as:

$$\begin{aligned} \overline{p}_{n+1} &= \alpha p_{n+1} + (1-\alpha)\overline{p}_n \\ \overline{\sigma}_{n+1} &= \alpha |p_{n+1} - \overline{p}_{n+1}| + (1-\alpha)\overline{\sigma}_n \end{aligned} \tag{1}$$

where $\alpha = \tau \times f$, f is the frame rate and τ is a time constant specifying how fast (responsive) the background model should be to intensity changes. the influence of old observations decays exponentially over time, and thus the background gradually adapts to reflect current environmental conditions.

If a pixel has a value which is more than 2σ from $\overline{p}_n$, then it is considered a foreground pixel. At this point a multiple hypothesis approach is used for determining its behavior. A new set of statistics $(\overline{p}', \sigma')$ is initialized for this pixel and the original set is remembered. If, after time $t = 3\tau$, the pixel value has not returned to its original statistical value, the new statistics are chosen as replacements for the old.

Foreground (moving) pixels are aggregated using a connected component approach so that individual target “blobs” can be extracted. Transient moving objects cause short term changes to the image stream that are not included in the background model, but are continually tracked, whereas more permanent changes are (after a time increment of 3τ has elapsed) absorbed into the background (see Figure 1).

The moving target detection algorithm described above is prone to three types of error: incomplete extraction of


(A) (B)

Figure 1: Example of moving target detection by dynamic background subtraction.

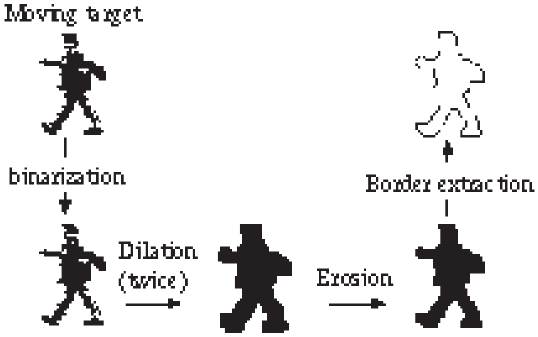


Figure 2: Target pre-processing. A moving target region is morphologically dilated (twice), eroded and then its border is extracted.

a moving object; erroneous extraction of non-moving pixels; and legitimate extraction of illegitimate targets (such as trees blowing in the wind). Incomplete targets are partially reconstructed by blob clustering and morphological dilation (Figure 2). Erroneously extracted "noise" is removed using a size filter whereby blobs below a certain critical size are ignored. Illegitimate targets must be removed by other means such as temporal consistency and domain knowledge. This is the purview of the target tracking algorithm.

2.2 Target Tracking

To begin to build a temporal model of activity, individual objects must be tracked over time. The first step in this process is to take the blobs generated by motion detection and match them between frames of a video sequence.

Many systems for target tracking are based on Kalman filters. However, as Isard and Blake point out, these are of limited use because they are based on unimodal Gaussian densities that cannot support simultaneous alternative motion hypotheses [Isard and Blake, 1996]. Isard and Blake present a new stochastic algorithm called CONDENSATION that does handle alternative hypotheses. Work on the problem of multiple data association in radar tracking contexts is also relevant [Bar-Shalom and Fortmann, 1988].

We employ a much simpler approach based on a frame-to-frame matching cost function. A record of each blob is kept with the following information:

- image trajectory (position $p(t)$ and velocity $v(t)$ as functions of time) of the object centroid,
- blob "appearance" in the form of an image template,

- blob size s in pixels,
- color histogram h of the blob.

The position and velocity of each blob T_i is determined from the last time step t_{last} and used to predict a new image position at the current time t_{now}:

$$p_i(t_{now}) \approx p_i(t_{last}) + v_i(t_{last}) \times (t_{now} - t_{last}) \tag{2}$$

Using this information a matching cost is determined between a known target T_i and a candidate moving blob R_j

$$C(T_i, R_j) = f(|p_i - p_j|, |s_i - s_j|, |h_i - h_j|). \tag{3}$$

Targets that are "close enough" in cost space are considered to be potential matches. To lend more robustness to changes in appearance and occlusions, the full tracking algorithm uses a combination of cost and adaptive template matching, as described in detail in [Lipton *et al.*, 1998]. Recent results from the system are shown in Figure 3.

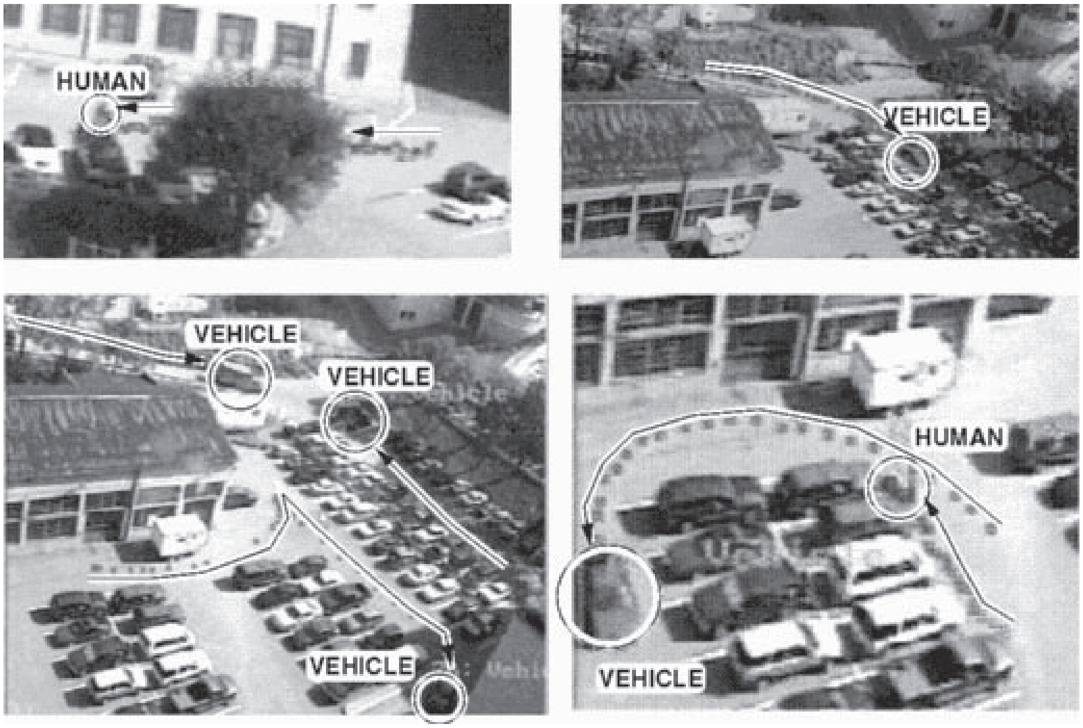


Figure 3: Recent results of moving entity detection and tracking showing detected objects and trajectories overlaid on original video imagery. Note that tracking persists even when targets are temporarily occluded or motionless.

2.3 Target Classification

The ultimate goal of the VSAM effort is to be able to identify individual entities (such as the "FedEx truck", the "4:15pm bus to Oakland" and "Fred Smith") and determine what they are doing. As a first step, entities are classified into specific class groupings such as "humans" and "vehicles".

Currently, we are experimenting with a neural network approach (Figure 4). The neural network is a standard three-layer network which uses a back propagation algorithm for hierarchical learning. Inputs to the network are a mixture of image-based and scene-based entity parameters: dispersedness (perimeter^2/area (pixels)); image area (pixels); aspect ratio (height/width); and camera zoom factor. Using a set of motion regions automatically extracted but labeled by hand, the network is trained to output one of three classes: human; vehicle; or human group (two or more humans walking close together). When teaching the network that an input entity is a human, all outputs are set to 0.0 except for "human", which is set to 1.0. Other classes are trained similarly. If the input does not fit any of the classes, such as a tree blowing in the wind, all outputs are set to 0.0.

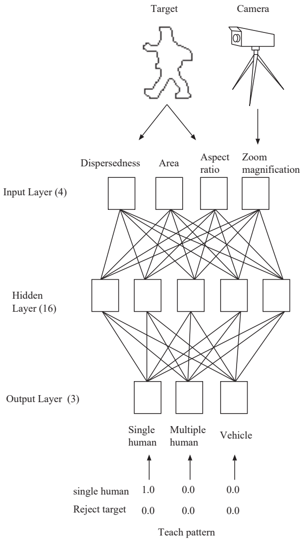


Figure 4: Neural network approach to target classification.

Results from the neural network are interpreted as follows:

```
if (output > THRESHOLD)
  classification = maximum NN output
else
  classification = REJECT
```

The results for this classification scheme are summarized in Table 1. This classification approach is effective for single images. However, one of the advantages of video is its temporal component. To exploit this, classification is performed on every entity at every frame and the results of classification are kept in a histogram with the ith bucket containing the number of times the object was classified as class i. At each time step, the class label that has been output most often for each object is chosen its most likely classification.

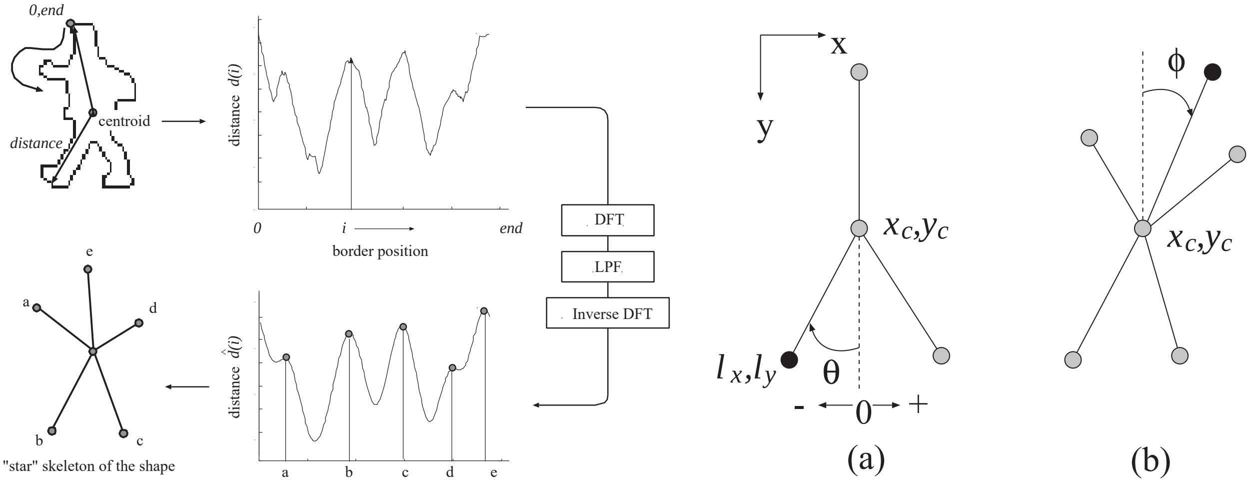
2.4 Activity Analysis

After classifying an object, we want to determine what it is doing. Understanding human activity is one of the most difficult open problems in the area of automated video surveillance. Detecting and analyzing human motion in real time from video imagery has only recently become viable with algorithms like *Pfinder* [Wren *et al.*, 1997] and W^4 [Haritaoglu *et al.*, 1998]. These algorithms represent a good first step to the problem of recognizing and analyzing humans, but they still have some drawbacks. In general, they work by detecting features (such as hands, feet and head), tracking them, and fitting them to some *a priori* human model such as the *cardboard model* of Ju *et al* [Ju *et al.*, 1996]. Therefore the human subject must dominate the image frame so that the individual body components can be reliably detected.

Class	Samples	% Correctly Classified
Human	430	99.5
Human group	96	88.5
Vehicle	508	99.4
False alarms	48	64.5
Total	1082	96.9

Table 1: Results for neural network classification algorithm.

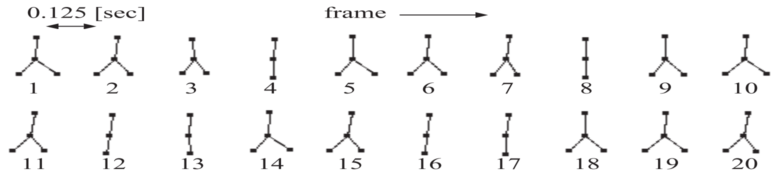
We use a “star” skeletonization procedure for analyzing the motion of humans that are relatively small in the image. Details can be found in [Fujiyoshi and Lipton, 1998]. The key idea is that a simple form of skeletonization that only extracts the broad internal motion features of a target can be employed to analyze its motion. This method provides a simple, real-time, robust way of detecting extremal points on the boundary of the target to produce a “star” skeleton. The “star” skeleton consists of the centroid of an entity and all of the local extremal points which can be recovered when traversing the boundary of the entity’s image (Figure 5a).



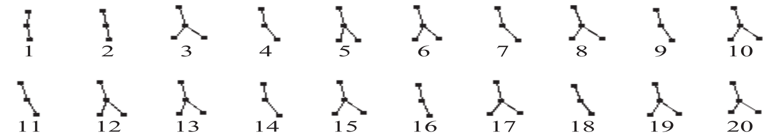
(A) (B)

Figure 5: (A) The star skeleton is formed by “unwrapping” a region boundary as a distance function from the centroid. This function is then smoothed and extremal points are extracted. (B) Determination of skeleton features measuring gait and posture. θ is the angle the leftmost leg makes with the vertical, and ϕ is the angle the torso makes with the vertical.

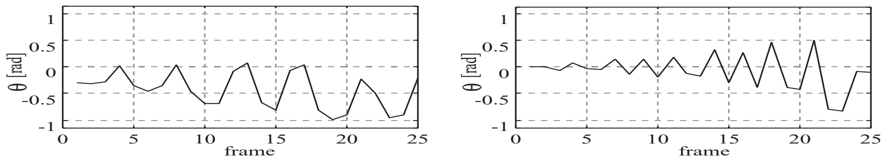
Using only measurements based on the “star” skeleton, it is possible to determine the gait and posture of a moving human being. Figure 5b shows how two angles ϕ_n and θ_n are extracted from the skeleton. The value ϕ_n represents the angle of the torso with respect to vertical, while θ_n represents the angle of the leftmost leg in the figure. Figure 6 shows skeleton motion for typical sequences of walking and running humans, along with the values of ϕ_n and θ_n. These data were acquired in real-time from a video stream with frame rate 8Hz. Comparing the average values $\overline{\phi}_n$ in figures 6(e)-(f) show that the posture of a running target can easily be distinguished from that of a walking one using the angle of the torso segment as a guide. Also, the frequency of cyclic motion of the leg segments provides cues to distinguishing running from walking.



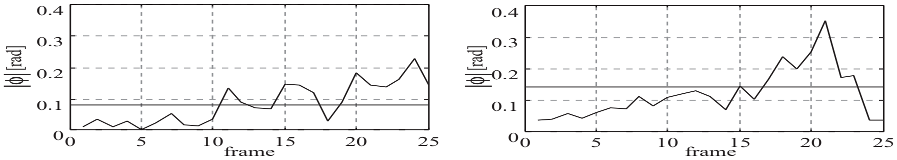
(a) skeleton motion of a walking person



(b) skeleton motion of a running person



(c) leg angle θ of a walking person (d) leg angle θ of a running person



(e) torso angle ϕ of a walking person (f) torso angle ϕ of a running person

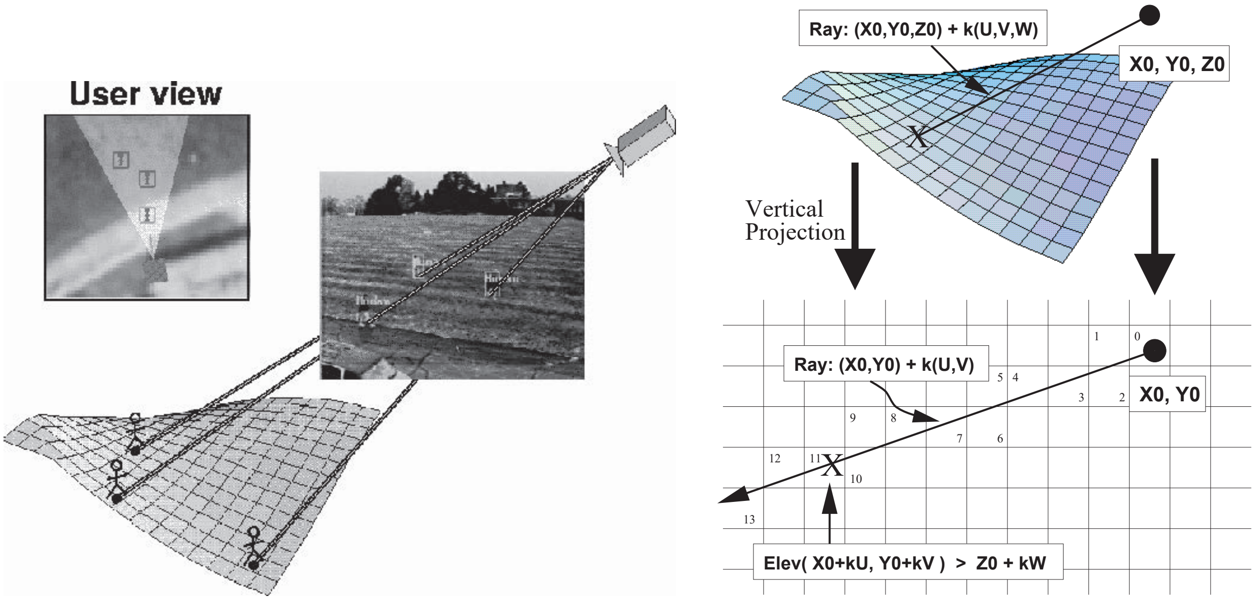
Figure 6: Skeleton motion sequences. Clearly, the periodic motion of θ_n provides cues to the target's motion as does the mean value of $\overline{\phi}_n$.

2.5 Model-based Geolocation

The video understanding techniques described so far have operated purely in image space. A large leap in terms of descriptive power can be made by transforming image blobs and measurements into 3D scene-based objects and descriptors. In particular, determination of object location in the scene allows us to infer the proper spatial relationships between sets of objects, and between objects and scene features such as roads and buildings. Furthermore, we believe the key to coherently integrating a large number of target hypotheses from multiple widely-spaced sensors is computation of target spatial geolocation.

In regions where multiple sensor viewpoints overlap, object locations can be determined very accurately by wide-baseline stereo triangulation. However, regions of the scene that can be simultaneously viewed by multiple sensors are likely to be a small percentage of the total area of regard in real outdoor surveillance applications, where it is desirable to maximize coverage of a large area given finite sensor resources. Determining target locations from a single sensor requires domain constraints, in this case the assumption that the object is in contact with the terrain. This contact location is estimated by passing a viewing ray through the bottom of the object in the image and intersecting it with a model representing the terrain (see Figure 7a). Sequences of location estimates over time are then assembled into consistent object trajectories.

Previous uses of the ray intersection technique for object localization in surveillance research have been restricted to small areas of planar terrain, where the relation between image pixels and terrain locations is a simple 2D homography [Bradshaw *et al.*, 1997, Flinchbaugh and Bannon, 1994, Koller *et al.*, 1993]. This has the benefit that no camera calibration is required to determine the back-projection of an image point onto the scene plane, provided the mappings of at least four coplanar scene points are known beforehand. However, large outdoor scene areas may



(A) (B)

Figure 7: (A) Estimating object geolocations by intersecting target viewing rays with a terrain model. (B) A Bresenham-like traversal algorithm determines which DEM cell contains the first intersection of a viewing ray and the terrain.

contain significantly varied terrain. To handle this situation, we perform geolocation using ray intersection with a full terrain model provided, for example, by a digital elevation map (DEM).

Given a calibrated sensor, and an image pixel corresponding to the assumed contact point between an object and the terrain, a viewing ray $(x_0 + ku, y_0 + kv, z_0 + kw)$ is constructed, where (x_0, y_0, z_0) is the 3D sensor location, (u, v, w) is a unit vector designating the direction of the viewing ray emanating from the sensor, and $k \geq 0$ is an arbitrary distance. General methods for determining where a viewing ray first intersects a 3D scene (for example, ray tracing) can be quite involved. However, when scene structure is stored as a DEM, a simple geometric traversal algorithm suggests itself, based on the well-known Bresenham algorithm for drawing digital line segments. Consider the vertical projection of the viewing ray onto the DEM grid (see Figure 7b). Starting at the grid cell (x_0, y_0) containing the sensor, each cell (x, y) that the ray passes through is examined in turn, progressing outward, until the elevation stored in that DEM cell exceeds the z-component of the 3D viewing ray at that location. The z-component of the view ray at location (x, y) is computed as either

$$z_0 + \frac{(x - x_0)}{u} w \quad \text{or} \quad z_0 + \frac{(y - y_0)}{v} w \tag{4}$$

depending on which direction cosine, u or v, is larger. This approach to viewing ray intersection localizes objects to lie within the boundaries of a single DEM grid cell. A more precise sub-cell location estimate can then be obtained by interpolation. If multiple intersections with the terrain beyond the first are required, this algorithm can be used to generate them in order of increasing distance from the sensor, out to some cut-off distance. See [Collins *et al.*, 1998] for more details.

2.6 Multi-Sensor Cooperation

In most complex outdoor scenes, it is impossible for a single sensor to maintain its view of an object for long periods of time. Objects become occluded by environmental features such as trees and buildings, and sensors have limited effective fields of regard. A promising solution to this problem is to use a network of video sensors to cooperatively track an object through the scene. Tracked objects are then *handed-off* between cameras to greatly extend the total effective area of surveillance coverage.

There has been little work done on autonomously coordinating multiple active video sensors to cooperatively track a moving target. One approach is presented by Matsuyama for a controlled indoor environment where four cameras lock onto onto a particular object moving across the floor [Matsuyama, 1998]. We approach the problem more generally by using the object's 3D geolocation as computed in the last section to determine where each sensor should look. The pan, tilt and zoom of the closest sensors are then controlled to bring the object within their fields of view, while a viewpoint independent cost function is used to determine which of the moving objects they find are the specific target of interest. These steps are described below.

Assume that at time t_0 a sensor with pan, tilt value (θ_0, ϕ_0) has been tasked to track a particular object with 3D ground location X_0 and velocity $\dot{X}$. Given a function $G(X)$ that converts a ground coordinate to a pan, tilt point (determined by camera calibration), the object's location X_0 is converted to a desired sensor pan, tilt value $(\theta_d, \phi_d) = G(X_0)$. The behavior of the pan, tilt unit is approximated by a linear system with infinite acceleration and maximum velocity $(\pm\dot{\theta}, \pm\dot{\phi})$ as

$$\begin{aligned} \theta(t) &= \theta_0 \pm \dot{\theta}(t - t_0) \\ \phi(t) &= \phi_0 \pm \dot{\phi}(t - t_0) \end{aligned} \tag{5}$$

Substituting the desired sensor pan, tilt (θ_d, ϕ_d) into the lefthand side of this equation and solving for $(t - t_0)$ yields a prediction of the acquisition time, that is, how long it would take for the pan, tilt device to point at the object's current location. However, the object will have moved further along its trajectory by that time. This new object position is estimated as

$$X(t) = X_0 + \dot{X}(t - t_0) \tag{6}$$

This predicted object position is then converted into a new desired sensor pan, tilt, and the whole procedure iterates until the time increments $(t - t_0)$ become small (convergence) or start to increase (divergence). This algorithm guarantees that if it converges, the sensor will be able to reacquire the object.

An appropriate camera zoom setting can be determined directly given a desired size of the object's projection in the image. Knowing the classification of the object C (as determined from Section 2.3), we employ the heuristic that humans are approximately 6 (2m) feet tall and vehicles are approximately 15 feet (5m) long to set the zoom. Given the position of the object and the sensor and, therefore the range r to the object, the angle ρ subtended by the image of the object is approximately

$$\rho = \begin{cases} \tan^{-1}\frac{2}{r}, & \text{human} \\ \tan^{-1}\frac{5}{r}, & \text{vehicle} \end{cases}$$

Knowing the focal length of the sensor as a function of zoom, as determined from camera calibration, the appropriate zoom setting is easily chosen.

Once the sensor is pointing in the right direction at the right zoom factor, all moving targets extracted are compared to the specific target of interest to see if they match. This need to re-acquire a specific target is a key feature necessary for multi-camera cooperative surveillance. Obviously viewpoint-specific appearance criteria are not useful, since the new view of the target may be significantly different from the previous view. Therefore, recognition features are needed that are independent of viewpoint. In our work we use two such criteria: the object's 3D scene trajectory as determined from geolocation, and a normalized color histogram of the object's image region. Candidate motion regions are tested by applying a matching cost function in a manner similar to that described in Section 2.2.

3 A VSAM Testbed System

We have built a prototype VSAM testbed system to demonstrate how the automated video understanding technology described in the last section can be combined into a coherent surveillance system that enables a single human operator to monitor a wide area. The testbed system consists of a central operator control unit (OCU) which receives video and ethernet data from multiple remote sensor processing units (SPUs) (see Figure 8). The OCU is responsible for integrating symbolic object trajectory information accumulated by each of the SPUs together with a 3D geometric site model, and presenting the results to the user on a map-based graphical user interface. Each component of the testbed system architecture is described briefly below.

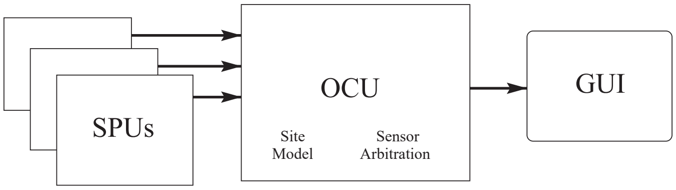


Figure 8: Overview of the VSAM testbed system.

3.1 Sensor Processing Units (SPUs)

Sensor processing units (SPUs) are the front-end sensing nodes of the testbed system. Their function is to automatically extract significant entities and events from video imagery, using the algorithms described in the last section. All video processing is performed on board the SPU, and the resulting object hypotheses are transmitted in symbolic form back to the OCU, significantly reducing the bandwidth requirements of the surveillance network. For example, we currently process NTSC color imagery with a frame size of 320x240 pixels at 10 frames per second on a Pentium II PC, so that data is streaming into the system through each SPU at a rate of roughly 2.3Mb per second per sensor. After VSAM processing, detected targets hypotheses contain information about object type, target location and velocity, as well as measurement statistics such as a time stamp and a description of the sensor (current pan, tilt, and zoom for example). Each target data packet takes up roughly 50 bytes. If a sensor tracks 3 targets for one second at 10 frames per second, it ends up transmitting 1500 bytes back to the OCU, well over a thousandfold reduction in data bandwidth.

The VSAM testbed can handle a wide variety of sensors and modalities. The current list of SPUs that we have successfully integrated into the system are:

- Five fixed-mount color CCD sensors with variable pan, tilt and zoom control, affixed to buildings around the CMU campus.
- One van-mounted relocatable SPU that can be moved from one point to another during a surveillance mission.
- A FLIR Systems camera turret mounted on an aircraft.
- A Columbia-Lehigh CycloVision ParaCamera with a hemispherical field of view.
- A Texas Instruments (TI) indoor surveillance system, which after some modifications is capable of directly interfacing with the VSAM network.

Logically, all of these SPUs are treated identically. In future years, it is hoped that other VSAM sensor modalities will be added, including thermal infrared sensors, multi-camera omnidirectional sensors, and stereo sensors.

3.2 Operator Control Unit (OCU)

Figure 9 shows the functional architecture of the VSAM OCU. It accepts video processing results from each of the SPUs and integrates the information with a site model and a database of known targets to infer activities that are of interest to the user. This data is sent to the GUI and other visualization tools as output from the system.

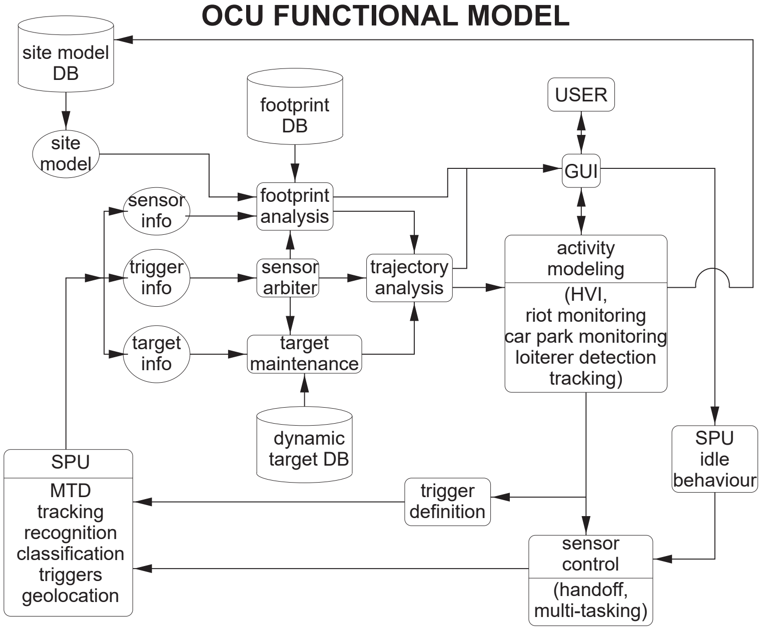


Figure 9: Functional architecture of the VSAM OCU.

One key piece of system functionality provided by the OCU is sensor arbitration. It is important in a surveillance system that sensor assets are not underutilised. Sensors must be allocated to surveillance tasks in such a way that all possible user-specified tasks can be performed, and if enough sensors are present, multiple sensors can be assigned to track important objects. At any given time, the OCU maintains a list of known objects and sensor parameters, as well as a set of "tasks" that may need attention. These tasks are explicitly indicated by the user through the GUI, and may include such things as specific targets to be tracked, specific regions to be watched, or specific events to be detected (such as a person loitering near a particular doorway). Sensor arbitration is performed by an arbitration cost function. The arbitration function determines the cost of assigning each of the SPUs to each of the tasks. These costs are based on the priority of the tasks, the load on the SPU, the visibility of the objects from a particular sensor, and so on. The system performs a greedy optimization of the cost to determine the best combination of SPU tasking to maximize overall system performance requirements.

The OCU also contains a site model representing VSAM-relevant information about the area being monitored. This includes both geometric and photometric information about the scene, represented using a combination of image and symbolic data. Site model representations have intentionally been kept as as simple as possible, with an eye towards efficiently supporting specific VSAM capabilities:

- target geolocation via intersection of viewing rays with the terrain.
- visibility analysis (predicting what what portions of the scene are visible from what sensors) so that sensors can be efficiently tasked.
- specification of of the geometric location and extent of relevant scene features. For example, we might directly

task a sensor to monitor the door of a building, or to look for vehicles passing through a particular intersection.

3.3 Graphical User Interface

One of the technical goals of the VSAM project is to demonstrate that a single human operator can effectively monitor a significant area of interest. Keeping track of multiple people, vehicles, and their interactions, within a complex urban environment is a difficult task. The user obviously shouldn't be looking at two dozen screens showing raw video output – that amount of sensory overload virtually guarantees that information will be ignored and would require a prohibitive amount of transmission bandwidth. Our approach is to provide an interactive, graphical user interface (GUI) that uses VSAM technology to automatically place dynamic agents representing people and vehicles into a synthetic view of the environment. This approach has the benefit that visualization of scene events is no longer tied to the original resolution and viewpoint of a single video sensor. The GUI currently consists of a map of the area, with all target and sensor platform locations overlaid on it.

In addition to scene visualization, the GUI is also used for sensor suite tasking. Through this interface, the operator can task individual sensor units, as well as the entire testbed sensor suite, to perform surveillance operations such as generating a quick summary of all target activities in the area.

4 VSAM Demonstrations

We have held two significant demonstrations of the VSAM system in the past two years. VSAM Demo I was held at CMU's Bushy Run research facility on November 12, 1997, roughly nine months into the program. The VSAM testbed system at that time consisted of an OCU with two ground-based and one airborne SPU. The two ground sensors cooperatively tracked a car as it entered the Bushy Run site, parked and let out two occupants. The two pedestrians were detected and tracked as they walked around and then returned to their car. The system continued tracking the car as it commenced its journey around the site, handing off control between cameras as the car left the field of view of each sensor. All entities were detected and tracking using temporal differencing motion detection and correlation-based tracking. Targets were classified into "vehicle" or "human" using a simple image-based property (aspect ratio) in conjunction with a temporal consistency constraint. Target geolocation was accomplished by intersection of back-projected viewing rays with the DEM terrain model. A synopsis of the vehicle trajectory computed automatically by the system is shown in Figure 10.

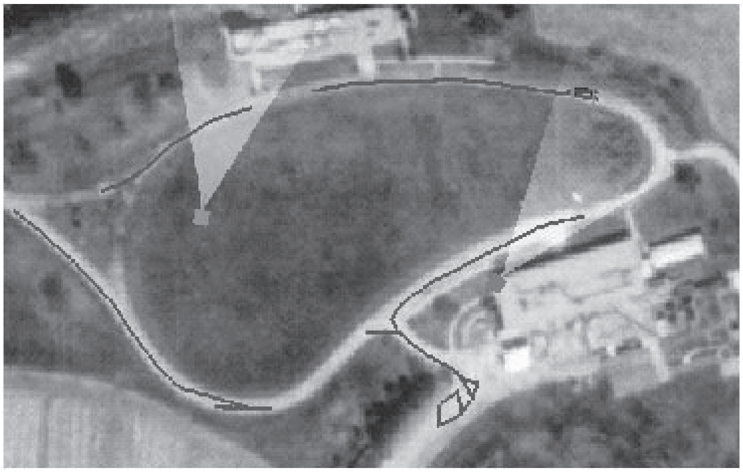


Figure 10: Synopsis of vehicle trajectory during the Bushy Run demo.

VSAM Demo II was help on October 8 on the urban campus of CMU. Five fixed-mount pan-tilt-zoom cameras were mounted on buildings around campus, a relocatable SPU mounted in a van, an airborne SPU operated by the Army's Night Vision and Electronic Sensor Directorate, a hemispherical field of view sensor operated by Lehigh and Columbia Universities, and an indoor video alarm system run by Texas Intruments. These sensors cooperated to track a vehicle from nearby Schenley Park onto campus, followed its path as it wound through campus and stopped at the OCU building, alerted the operator when one of the vehicle's occupants entered the building, and followed the ensuing car and foot chases as the vehicle and its occupants attempted to flee from the police. An example of multi-sensor tracking of the vehicle as it attempted to leave campus is shown in Figure 11. This diagram shows the continuous, autonomous tracking of a single object for a distance of approximately 400m and a time of approximately 3 minutes. In Figure 11(a) two sensors cooperatively track the object. At the time shown in Figure 11(b) the object is occluded from sensor 2, but is still visible from sensor 1, which continues to track it. When the object moves out of the occlusion area, sensor 2 is automatically retasked to track it, as shown in Figure 11(c).

Finally, when the object moves out of the field of regard of both sensors, a third sensor is automatically tasked to continue surveillance, as shown in Figure 11(d). By automatically managing multiple, redundant camera resources, the vehicle is continously tracked through a complex urban environment.

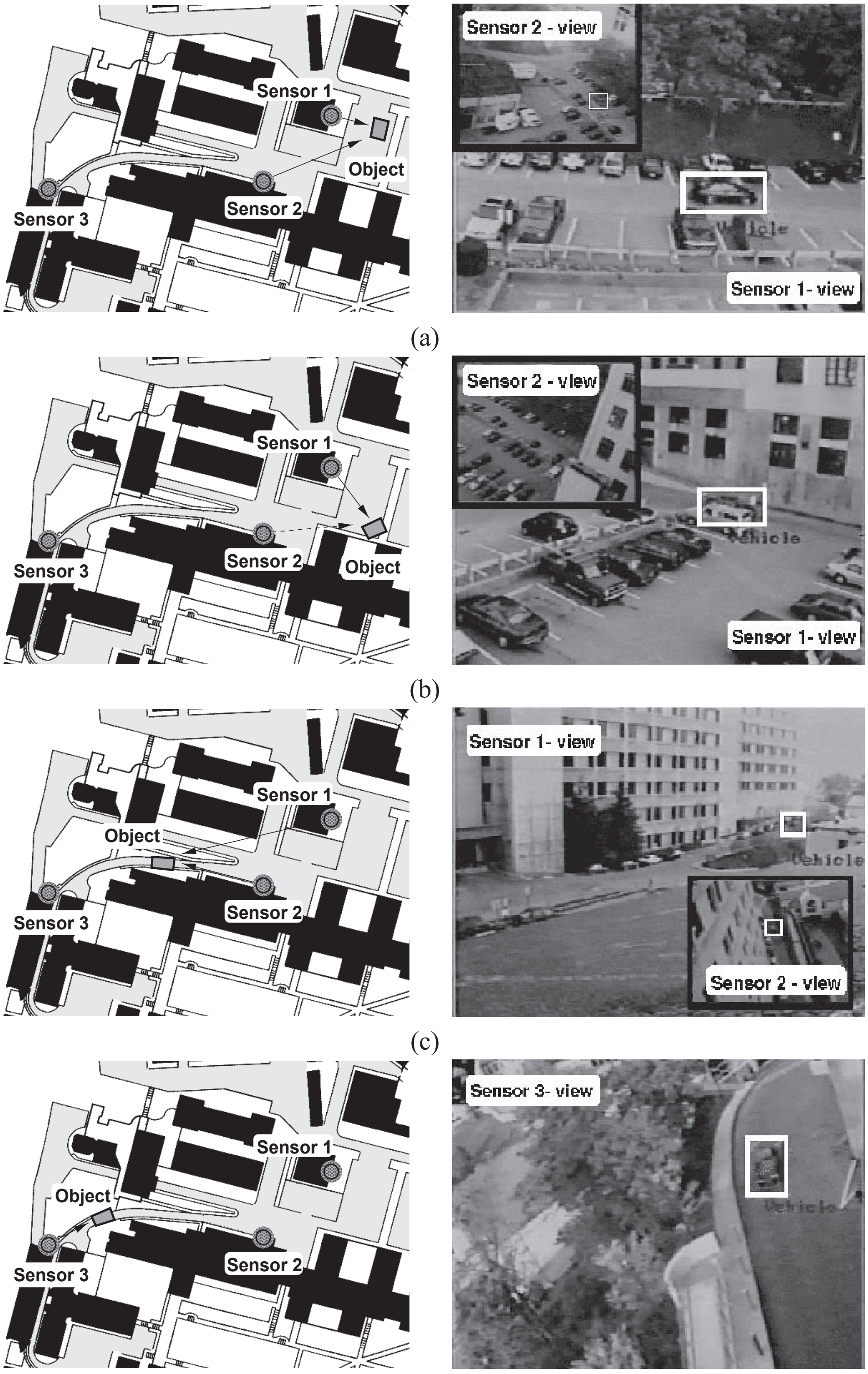
5 Conclusion

CMU and the Sarnoff Corporation have developed a testbed system for automated vision surveillance and monitoring. Multiple sensors cooperate to track moving objects through a complex, urban environment. More information on this project, particularly on the airborne processing component, can be found in [Kanade *et al.*, 1998].

As the VSAM effort enters its third year, we are focusing on making the video understanding algorithms more robust to lighting and environmental conditions, adding thermal cameras to the permanent test sensor suite located on the CMU campus, adding more complex ground sensor control strategies such as sensor multi-tasking, and the ability to perform unsupervised monitoring of limited domains such as parking lots, and expanding the testbed system's network architecture to handle Web-VSAM – remote site monitoring and SPU control over the internet using a JAVA DIS client.

Acknowledgments

The authors would like to thank the CMU VSAM team members: Hironobu Fujiyoshi, Dave Duggins, David Tolliver, Raju Patil, Yanghai Tsin, and Alan Lee, for their tireless efforts and good humor over the last two years.



(a)

(b)

(c)

(d)

Figure 11: Cooperative, multi-sensor tracking (see text for description).

References

[Anderson *et al.*, 1985] C. Anderson, Peter Burt, and G. van der Wal. Change detection and tracking using pyramid transformation techniques. In *Proceedings of SPIE - Intelligent Robots and Computer Vision*, volume 579, pages 72–78, 1985.

[Bar-Shalom and Fortmann, 1988] Yaakov Bar-Shalom and Thomas Fortmann. *Tacking and data association*. Academic Press, Boston, 1988.

[Barron *et al.*, 1994] J. Barron, D. Fleet, and S. Beauchemin. Performance of optical flow techniques. *International Journal of Computer Vision*, 12(1):42–77, 1994.

[Bradshaw *et al.*, 1997] K. Bradshaw, I. Reid, and D. Murray. The active recovery of 3d motion trajectories and their use in prediction. *PAMI*, 19(3):219–234, March 1997.

[Collins *et al.*, 1998] Robert Collins, Yanghai Tsin, J.Ryan Miller, and Alan Lipton. *Using a DEM to Determine Geospatial Object Trajectories*. CMU technical report CMU-RI-TR-98-19, 1998.

[Flinchbaugh and Bannon, 1994] B. Flinchbaugh and T. Bannon. Autonomous scene monitoring system. In *Proc. 10th Annual Joint Government-Industry Security Technology Symposium*. American Defense Preparedness Association, June 1994.

[Fujiyoshi and Lipton, 1998] Hironobu Fujiyoshi and Alan Lipton. Real-time human motion analysis by image skeletonization. In *Proceedings of IEEE WACV98*, 1998.

[Grimson and Viola, 1997] Eric Grimson and Paul Viola. A forest of sensors. In *Proceedings of DARP - VSAM workshop II*, November 1997.

[Haritaoglu *et al.*, 1998] I. Haritaoglu, Larry S. Davis, and D. Harwood. w^4 who? when? where? what? a real time system for detecing and tracking people. In *FGR98 (submitted)*, 1998.

[Isard and Blake, 1996] M. Isard and A. Blake. Contour tracking by stochastic propagation of conditional density. In *Proceedings of European Conference on Computer Vision 96*, pages 343–356, 1996.

[Ju *et al.*, 1996] S. Ju, M. Black, and Y. Yacoob. Cardboard people: A parameterized model of articulated image motion. In *Proceedings of International Conference on Face and Gesture Analysis*, 1996.

[Kanade *et al.*, 1998] Takeo Kanade, Robert Collins, Alan Lipton, Peter Burt, and Lambert Wixson. Advances in cooperative multi-sensor video surveillance. In *Proceedings of DARPA Image Understanding Workshop*, volume 1, pages 3–24, November 1998.

[Koller *et al.*, 1993] D. Koller, K. Daniilidis, and H. Nagel. Model-based object tracking in monocular image sequences of road traffic scenes. *IJCV*, 10(3):257–281, June 1993.

[Lipton *et al.*, 1998] Alan Lipton, Hironobu Fujiyoshi, and Raju S. Patil. Moving target detection and classification from real-time video. In *Proceedings of IEEE WACV98*, 1998.

[Matsuyama, 1998] Takashi Matsuyama. Cooperative distributed vision. In *Proceedings of DARPA Image Understanding Workshop*, volume 1, pages 365–384, November 1998.

[Wren *et al.*, 1997] C. Wren, A. Azarbayejani, T. Darrell, and Alex Pentland. Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):780–785, 1997.

All Fields ▼ Search 42,... FAQ Contact Sign in

A System for Video Surveillance and Monitoring

Robert T. Collins, Alan J. Lipton, Takeo Kanade • Published 1999

The Robotics Institute at Carnegie Mellon University (CMU) and the Sarnoff Corporation are developing a system for autonomous Video Surveillance and Monitoring. The technical objective is to use multiple, cooperative video sensors to provide continuous coverage of people and vehicles in cluttered environments. This paper presents an overview of the system and significant results achieved to date.

Highly Influential
This paper has highly influenced 50 other papers.
REVIEW HIGHLY INFLUENTIAL CITATIONS

Highly Cited
This paper has 1,676 citations.
REVIEW CITATIONS

VIEW PAPER

SAVE TO LIBRARY CREATE ALERT CITE

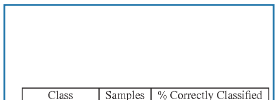
PAPER OUTLINE

ABSTRACT
FROM THIS PAPER
597 CITATIONS
11 REFERENCES
SIMILAR PAPERS

From This Paper

Figures, tables, and topics from this paper.

Figures & Tables


Class | Samples | % Correctly Classified

Explore Further: Topics Discussed in This Paper

- Testbed

Figure 1

Table 1

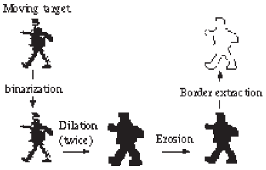


Figure 2

Figure 3

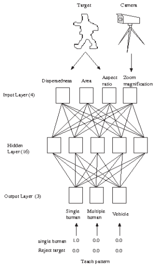


Figure 4

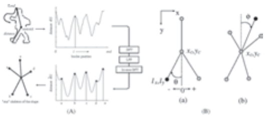


Figure 5

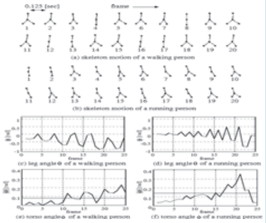


Figure 6

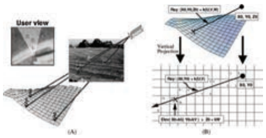


Figure 7

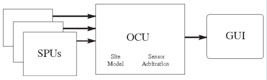


Figure 8

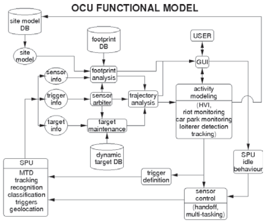


Figure 9

- polydrug use
- Computer multitasking
- Airborne Ranger
- Autonomous robot
- Algorithm
- Java
- Academic Medical Centers
- Numerous
- Closed-circuit television

Figure 10

Figure 11

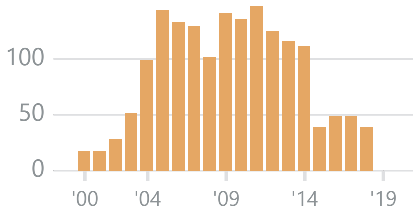
Citations

Publications citing this paper.

SHOWING 1-10 OF 597 CITATIONS, ESTIMATED 36% COVERAGE ?

Sort by: Influence ▼

Citation Statistics



Semantic Scholar estimates that this publication has **1,677** citations based on the available data.

See our **FAQ** for additional information.

Scene background initialization: A taxonomy

Thierry Bouwmans, Lucia Maddalena, Alfredo Petrosino • Pattern Recognition Letters • 2017

Highly Influenced 4 Excerpts

Long-term tracking of multiple interacting pedestrians using a single camera

Advice Seiphemo Keaikitse, Willie Brink, Natasha Govender • 2014

Highly Influenced 4 Excerpts

Real time hand detection in a complex background

Ekaterini Stergiopoulou, Kyriakos Sgouropoulos, Nikos A. Nikolaou, Nikos Papamarkos, Nikolaos Mitianoudis • Eng. Appl. of AI • 2014

Highly Influenced 6 Excerpts

Exploring Techniques for Vision Based Human Activity Recognition: Methods, Systems, and Evaluation

Xin Xu, Jinshan Tang, Xiaolong Zhang, Xiaoming Liu, Hong Zhang, Yimin Qiu • Sensors • 2013

Highly Influenced 8 Excerpts

Highly Influenced 5 Excerpts

---

1 A Block Matching Technique for Object Tracking Based on Peripheral Increment Sign Correlation Image

Hyoungseop Kim, Joo Kooi Tan, Seiji Ishikawa • 2012

Highly Influenced 6 Excerpts

Robust Motion Detection in Real-Life Scenarios

Ester Martínez-Martín, Angel P. Del Pobil • SpringerBriefs in Computer Science • 2012

Highly Influenced 8 Excerpts

An expert fuzzy system for predicting object collisions. Its application for avoiding pedestrian accidents

Juan Luis Castro, Miguel Delgado, Javier Medina, María Dolores Ruíz Lozano • Expert Syst. Appl. • 2011

Highly Influenced 9 Excerpts

Interested in publishing with us ? Contact book

Teddy Ko • 2011

Highly Influenced 5 Excerpts

An Architecture for a Task-Oriented Surveillance System: A Service- and Event-Based Approach

Jürgen Moßgraber, Frank Reinert, Hauke Vagts • Fifth International Conference on Systems • 2010

Highly Influenced 10 Excerpts

‹ 1 2 3 4 5 ›

References

Publications referenced by this paper.

Advances in Cooperative Multi-Sensor Video Surveillance
Takeo Kanade • 1998

Moving target detection and classification from real-time video
Lipton et al, 1998 Alan Lipton, Hironobu Fujiyoshi, Raju S. Patil • InProceedings of IEEE WACV98 • 1998

Real-time human motion analysis by image skeletonization
Hironobu Fujiyoshi, Alan J. Lipton • WACV • 1998

Pfinder: Real-Time Tracking of the Human Body
Christopher Richard Wren, Ali Azarbayejani, Trevor Darrell, Alex Pentland • IEEE Trans. Pattern Anal. Mach. Intell. • 1997

The Active Recovery of 3D Motion Trajectories and Their Use in Prediction
Kevin J. Bradshaw, Ian D. Reid, David William Murray • IEEE Trans. Pattern Anal. Mach. Intell. • 1997

and Paul Viola . A forest of sensors
Eric Grimson • Proceedings of DARP - VSAM workshop II , November... • 1997

Cardboard People: A Parameterized Model of Articulated Image Motion
Shanon X. Ju, Michael J. Black, Yaser Yacoob • FG • 1996

Contour Tracking by Stochastic Propagation of Conditional Density
Michael Isard, Andrew Blake • ECCV • 1996

Model-based object tracking in monocular image sequences of road traffic scenes
D. Roller, Kostas Daniilidis, Hans-Hellmut Nagel • International Journal of Computer 11263on • 1993

Performance of optical flow techniques
John L. Barron, David J. Fleet, Steven S. Beauchemin, T. A. Burkitt • CVPR • 1992

‹ 1 2 ›

Similar Papers

Content-Based Access to Algebraic Video
Ron Weiss, Andrzej Duda, David Kenneth Gifford • ICMCS • 1994

ClassView: hierarchical video shot classification, indexing, and accessing
Jianping Fan, Ahmed K. Elmagarmid, Xingquan Zhu, Walid G. Aref, Lide Wu • IEEE Transactions on Multimedia • 2004

Composition and Search with a Video Algebra
Ron Weiss, Andrzej Duda, David Kenneth Gifford • IEEE MultiMedia • 1995

The impact of YouTube recommendation system on video views
Renjie Zhou, Samamon Khemmarat, Lixin Gao • Internet Measurement Conference • 2010

Video Streaming: Concepts, Algorithms, and Systems
John G. Apostolopoulos, Wai-tian Tan • 2002

Using Webcast Text for Semantic Event Detection in Broadcast Sports Video
Changsheng Xu, Yifan Zhang, Guangyu Zhu, Yong Rui, Hanqing Lu, Qingming Huang • IEEE Transactions on Multimedia • 2008

InsightVideo: toward hierarchical video content organization for efficient browsing...
Xingquan Zhu, Ahmed K. Elmagarmid, Xiangyang Xue, Lide Wu, Ann Christine Catlin • IEEE Transactions on Multimedia • 2005

A hierarchical access control model for video database systems
Elisa Bertino, Jianping Fan, Elena Ferrari, Mohand-Said Hacid, Ahmed K. Elmagarmid, Xingquan Zhu • ACM Trans. Inf. Syst. • 2003

Video Summarization for Large Sports Video Archives
Yoshimasa Takahashi, Naoko Nitta, Noboru Babaguchi • IEEE International Conference on Multimedia and... • 2005

Search More

EXHIBIT G

Image Understanding Workshop

Proceedings of a Workshop
held in
Monterey, California

November 20–23, 1998

Volume I

Sponsored by:
Defense Advanced Research Projects Agency
Information Systems Office

This document contains copies of reports prepared for the DARPA Image Understanding Workshop. Included are Principal Investigator reports and technical results from the basic and strategic computing programs within DARPA/ISO-sponsored projects and certain technical reports from selected scientists from other organizations.

APPROVED FOR PUBLIC RELEASE
DISTRIBUTION UNLIMITED

Distributed by:
Morgan Kaufmann Publishers Inc.
340 Pine Street, 6th Floor
San Francisco, Calif. 94104-3205
ISBN: 1–55860–583–5
Printed in the United States of America

Table of Contents

Volume I

Section I — Video Surveillance and Monitoring (VSAM)

Video Surveillance and Monitoring – Principal Investigator Reports

Video Surveillance and Monitoring – Technical Papers

Event Recognition and Reliability Improvements for the Autonomous Video Surveillance System

Frank Z. Brill, Thomas J. Olson, and Christopher Tserng
Texas Instruments
P.O. Box 655303, MS 8374, Dallas, TX 75265
brill@csc.ti.com, olson@csc.ti.com, tserng@csc.ti.com

Abstract

This report describes recent progress in the development of the Autonomous Video Surveillance (AVS) system, a general-purpose system for moving object detection and event recognition. AVS analyses live video of a scene and builds a description of the activity in that scene. The recent enhancements to AVS described in this report are: (1) use of collateral information sources, (2) camera hand-off, (3) vehicle event recognition, and (4) complex-event recognition. Also described is a new segmentation and tracking technique and an evaluation of AVS performing the best-view selection task.

1. Introduction

The Autonomous Video Surveillance (AVS) system processes live video streams from surveillance cameras to automatically produce a real-time map-based display of the locations of people, objects and events in a monitored region. The system allows a user to specify alarm conditions interactively, based on the locations of people and objects in the scene, the types of objects in the scene, the events in which the people and objects are involved, and the times at which the events occur. Furthermore, the user can specify the action to take when an alarm is triggered, e.g., to generate an audio alarm or write a log file. For example, the user can specify that an audio alarm should be triggered if a person deposits a briefcase on a given table between 5:00pm and 7:00am on a weeknight. Section 2 below describes recent enhancements to the AVS system. Section 3 describes progress in improving the reliability of segmentation and tracking. Section 4 describes an experiment that quantifies the performance of the AVS "best view selection" capability.

2. New AVS functionality

The structure and function of the AVS system is described in detail in a previous IUW paper [Olson and Brill, 1997]. The primary purpose of the current paper is to describe recent enhancements to the AVS system. These enhancements are described in four sections below: (1) collateral information sources, (2) camera hand-off, (3) vehicle event recognition, and (4) complex-event recognition.

2.1. Collateral information sources

Figure 1 shows a diagram of the AVS system. One or more "smart" cameras process the video stream to recognize events. The resulting event streams are sent to a Video Surveillance Shell (VSS), which integrates the information and displays it on a map. The VSS can also generate alarms based on the information in the event streams. In recent work, the VSS was enhanced to accept information from other sources, or "recognition devices" which can identify the objects being reported on by the cameras. For example, a camera may report that there is a person near a door. A recognition device may report that the person near the door is Joe Smith. The recognition device may be a badge reader, a keypad in which a person types their PIN, a face recognition system, or other recognition system.

This research was sponsored in part by the DARPA Image Understanding Program.

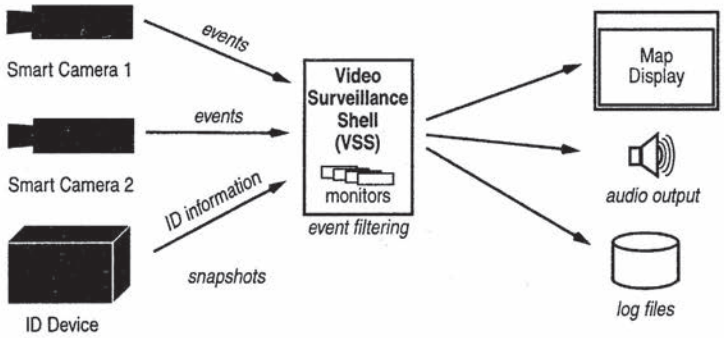


Figure 1: AVS system diagram

The recognition device we have incorporated is a voice verification system. The user stands in a predefined location in the room, and speaks his or her name. The system matches the utterance to previously captured examples of the person speaking their name, and reports to the VSS if there is a match. The VSS now knows the identity of the person being observed, and can customize alarms based on the person's identity.

A recognition device could identify things other than people, and could classify actions instead of objects. For example, the MIT Action Recognition System (MARS) recognizes actions of people in the scene, such as raising their arms or bending over. MARS is trained by observing examples of the action to be recognized and forming "temporal templates" that briefly describe the action [Davis and Bobick, 1997]. At run time, MARS observes the motion in the scene and determines when the motion matches one of the stored temporal templates. TI has obtained an evaluation copy of the MARS software and used it as an recognition device which identifies actions, and sends the result to the AVS VSS. We successfully trained MARS to recognize the actions of opening a door, and opening the drawer of a file cabinet. When MARS recognizes these actions, it sends a message to the AVS VSS, which can generate an appropriate alarm.

2.2. Camera hand-off

As depicted in Figure 1, the AVS system incorporates multiple cameras to enable surveillance of a wider area than can be monitored via a single camera. If the fields of view of these cameras are adjacent, a person can be tracked from one monitored area to another. When the person leaves the field of view of one camera and enters another, the process of maintaining the track from one camera view to another is termed *camera hand-off*. Figure 2 shows an area monitored by two cameras. Cam-

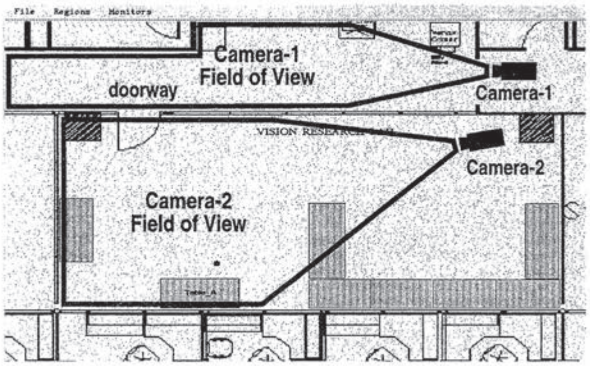


Figure 2: Multiple cameras with adjacent fields of view

era-1 monitors the hallway, and Camera-2 monitors the interior of the room. When a person moves through the doorway to enter the room from the hall or vice-versa, camera hand-off is necessary to enable the system to know that the person that was being monitored in the hall via Camera-1 is the *same* as the person being monitored in the room via Camera-2.

The AVS system accomplishes camera hand-off by integrating the information from the two cameras in the map coordinate system. The AVS "smart" cameras report the locations of the monitored objects and people in map coordinates, so that when the VSS receives reports about a person from two separate cameras, and both cameras are reporting the person's coordinates at about the same map location, the VSS can deduce that the two separate reports refer to the same person. In the example depicted in Figure 2, when a person is standing in the doorway, both cameras can see the person and report his or her location at nearly the same place. The VSS reports this as one person, using a minimum distance to allow for errors in location. When Camera-2 first sees a person at a location near the doorway and reports this to the VSS, the VSS checks to see if Camera-1 recently reported a person near the door. If so, the VSS reports the person in the room as the same one that Camera-1 had been tracking in the hall.

2.3. Vehicle event recognition

This section describes extensions to the existing AVS system that enable the recognition of events involving interactions of people with cars. These new capabilities enable smart security cameras to monitor streets, parking lots and driveways and report when suspicious events occur. For example, a smart camera signals an alarm when a person exits a car, deposits an object near a building, reenters the car, and drives away.

2.3.1. Scope and assumptions

Extending the AVS system to handle human-vehicle interactions reliably involved two separable subproblems. First, the system's vocabulary for events and objects must be extended to handle a new class of object (vehicle) and new event types. Second, the AVS moving object detection and tracking software must be modified to handle the outdoor environment, which features variable lighting, strong shadows, atmospheric disturbances, and dynamic backgrounds. The work described here in section 2.3 addresses the first problem, to extend the system for vehicle events in conditions of uniform overcast with little wind. Our approach to handling general outdoor lighting conditions is discussed in section 4.

The method is further specialized for imaging conditions in which:

1. The camera views cars laterally.
2. Cars are unoccluded by other cars.
3. When cars and people overlap, only one of the overlapping objects is moving
4. The events of interest are people getting into and out of cars.

2.3.2. Car detection

The first thing that was done to expand the event recognizing capability of the current system was to give the system the ability to distinguish between people and cars. The system classifies objects as cars by using their sizes and aspect ratios. The size of an object in feet is obtained using the AVS system's image coordinate to world coordinate mapping. Once the system has detected a car, it analyzes the motion graph to recognize new events.

2.3.3. Car event recognition

In principle, car exit and car entry events could be recognized by detecting characteristic interactions of blobs in difference images, in a manner similar to the way AVS recognizes DEPOSIT and REMOVE events. In early experiments, however, this method turned out to be unsatisfactory because the underlying motion segmentation method did not segment cars from people. Whenever the people pass near the car they appear to merge with it, and track is lost until they walk away from it.

To solve this problem, a new approach involving additional image differencing was developed. The technique allows objects to be detected and tracked even when their images overlap the image of the car. This method requires two reference images: one consists of the original background scene (background image), and the other is identical to the first except it includes the car. The system takes differences between the current video image and the original reference image as usual. However, it also differences the current video image with the reference image containing the car. This allows the system to detect objects which may be overlapping the car. Using this technique, it is easy to detect when people enter and exit a car. If an object disappears while overlapping with a car, it probably entered the car. Similarly, if an object appears overlapping a car, it probably exited the car.

2.3.4. Basic method

When a car comes to rest, the following steps are taken. First, the image of the car object is removed from its frame and stored. Then, the car image is merged with the background image, creating an updated reference image containing the car. (Terminology: a *reference car image* is the subregion of the updated reference image that contains the car.) Then, the *car background image*, the region of the original background image that is replaced by the car image, is stored.

For each successive frame, two difference images are generated. One difference image, the *foreground difference image*, is calculated by differencing the current video image with the updated reference image. The foreground difference image will contain all the blobs that represent objects other than the car, including ones that overlap the car. The second difference image, the *car difference image*, is calculated using the car background image. The car difference image is formed from the difference between the current frame and the car background image, and contains the large blob for the car itself. Figures 3 and 4 show the construction and use of these images.

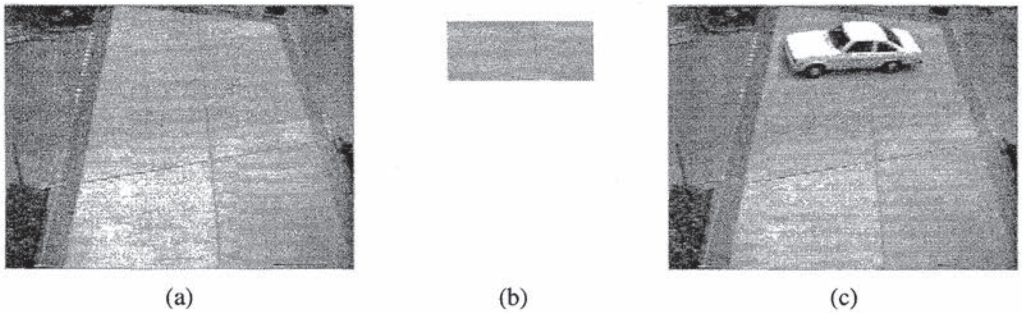

(a) (b) (c)

Figure 3: (a) Background image. (b) Car background image.
(c) Updated reference image

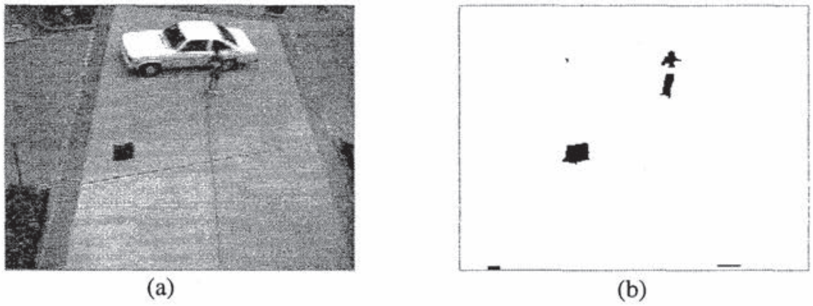

(a) (b)

Figure 4: (a) Current video image. (b) Foreground difference image

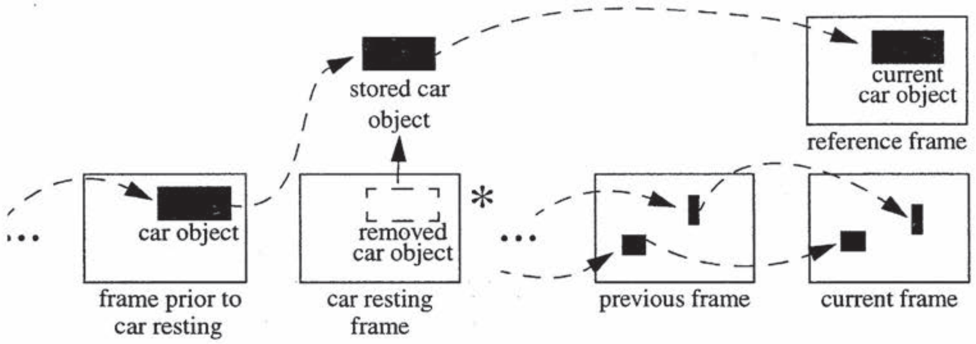


Figure 5: Creation of the motion graph.
The starred frame represents the frame prior to the background image being updated.

The blobs in the foreground difference image are grouped into objects using the normal grouping heuristics and placed in the current frame. The blobs in the car difference image necessarily represent the car, so they are all grouped into one current car object and placed in a special *reference frame*. Normal links occur between objects in the previous frame and objects in the current frame. Additionally, the stored car object, which was removed from its frame, (from Step 1) is linked to the current car object which is in the reference frame. In any given sequence, there is only one reference frame.

Figure 5 demonstrates the creation of this new motion graph. As indicated by the dotted lines, all objects maintain their tracks using this method. Notice that even though the car object disappears from future frames (due to the updated reference image), it is not detected to have exited because its track is maintained throughout every frame. Using this method, the system is able to keep track of the car object as well as any objects overlapping the car. If an object appears intersecting a car object, an INCAR event is reported. If an object disappears while intersecting a car object, an OUTCAR event is reported. Figure 6 shows the output of the system. The system will continue to operate in this manner until the car in the reference frame begins to move again.

When the car moves again, the system reverts to its normal single-reference-image state. The system detects the car's motion based on the movement of its centroid. It compares the position of the centroid of the stored car object with the centroid of the current car object. Figure 7 shows the slight movement of the car.

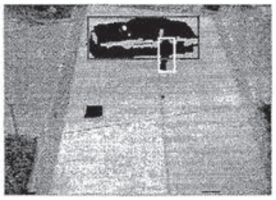


Figure 6: Final output of system

Figure 7: (a) Reference car image. (b) Moving car image.
(c) Reference car difference image. (d) Moving car difference image

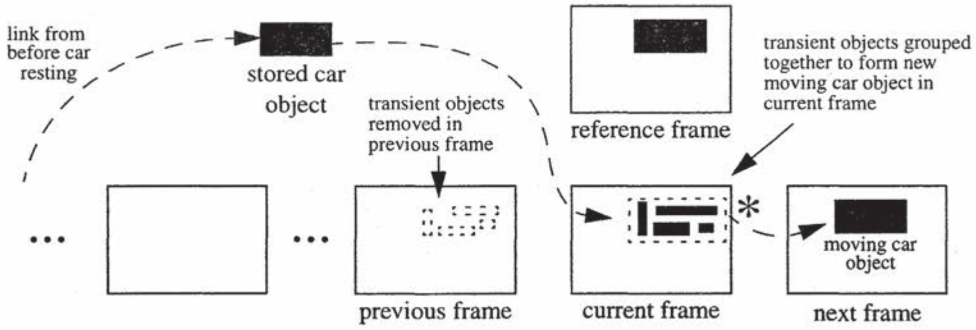


Figure 8: Restoration of normal differencing. The starred frame represents the last frame prior to the original reference image being restored.

If the centroid locations differ by more than a threshold, the following sequence of events occur to restore the system to its original state:

1. An object representing the moving car is created in the current frame.
2. The stored car object is linked to this new moving car object in the current frame.
3. Objects in the previous frame that intersect the moving car are removed from that frame.
4. The car background image is merged with the updated reference image to restore the original reference image.
5. Normal differencing continues.

Figure 8 demonstrates how the system is restored to its original state. Note that there is one continuous track that represents the path of the car throughout.

When the car begins to move again, transient blobs appear in the foreground difference image due to the fact that the car is in the updated reference image as seen in Figure 9. Therefore, to create a new moving car object in the current frame, these transient objects, which are identified by their intersection with the location of the resting car, are grouped together as one car object. If there are no transient objects, a copy of the stored car object is inserted into the current frame. This way, there is definitely a car object in the current frame to link with the stored car object. Transient objects might also appear in the previous frame when a car is moving. Therefore, these transient objects must be removed from their frame in order to prevent them from being linked to the new moving car object that was just created in the current frame. After the steps described above occur, the system continues as usual until another car comes to rest.

2.3.5. Experiments: disk-based sequences

To test the principles behind the modified AVS system, three sequences of video that represented interesting events were captured to disk. These sequences represented events which the modified system should be able to recognize. Capturing the sequences to disk reduces noise and ensures that the system processes the same frames on every run, making the results deterministic. In addition to these sequences, longer sequences were recorded and run directly from videotape to test how the system would work under less ideal conditions.

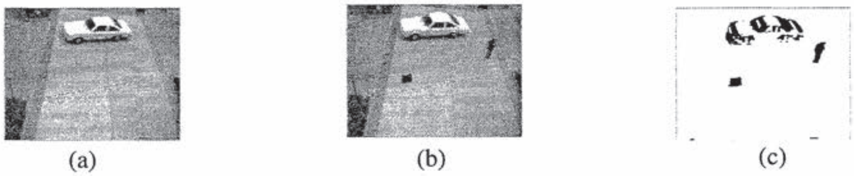

(a) (b) (c)

Figure 9: (a) Updated reference image. (b) Current video image. (c) Foreground difference image

2.3.5.1. Simple sequence. The first sequence was filmed from the 3rd story of an office building overlooking the driveway in front of the building. A car drives up and a person exits the car, walks away, deposits a briefcase, and finally reenters the car. Then, the car drives away. In this segment, the system successfully detects the person exiting the car. However, the person entering the car is missed because the person gets grouped with a second person walking near the car.

Further on in the sequence, the car drives up again and a person exits the car, walks away, removes the briefcase, and finally reenters the car. Again, the car drives away. In this segment, both the person entering and exiting the car are recognized. In both these sequences, there was only the one false negative mentioned earlier and no false positives.

2.3.5.2. Pickup sequence. This sequence was filmed in front of a house looking at the street in front of the house. In the sequence, a person walks into the scene and waits at the curb. A car drives up, picks up the person, and drives away. The system correctly detects the person entering the car. There are no false positives or negatives.

2.3.5.3. Drop off sequence. This sequence was filmed in the same location as the previous one. In this sequence, a car drives up and a person is dropped off. The car drives away with the person still standing in the same location. Then, the person walks off. The system correctly detects the person exiting the car and does not report a false enter event when the car moves away.

2.3.6. Experiments: videotaped sequences

These sequences were run on the system straight from videotape. These were all run at a higher threshold to accommodate noise on the videotape. However, this tended to decrease the performance of the system.

2.3.6.1. Dark day. This is a 15 minute sequence that was recorded from the 3rd floor of a building on a fairly dark day. In that time span, 8 cars passed through the camera's field of view. The system detected 6 cars correctly and one false car (due to people grouped together). One car that was not detected was due to its small size. The other car was undetected because the system slowed down (due to multiple events occurring) and missed the images with the car in them. In this sequence, two people entered a car. However, both events were missed because the car was not recognized as resting due to the dark lighting conditions on this rainy day.

2.3.6.2. Cloudy day. This is a 13 minute sequence in the same location as the previous sequence except it is a cloudy day. In this time span, 9 cars passed through the camera's field of view and all of them were detected by the system. There were a total of 2 people entering a car and 2 people exiting a car. The system successfully detected them all. Additionally, it incorrectly reported one person walking near a car as an instance of a person exiting a car.

2.3.6.3. Cloudy day—extended time. This is a 30 minute sequence in the same location as the previous two. In this time span, 28 cars pass through and all of them were detected. The system successfully detected one person exiting a car but missed two others. The two people were missed because the car was on the edge of the camera's field of view and so it was not recognized immediately as a car.

2.3.7. Evaluation of car-event recognition

The modified AVS system performs reasonably well on the test data. However, it has only been tested on a small number of videotaped sequences, in which much of the action was staged. Further experiments and further work with live, uncontrolled data will be required to make the system handle outdoor vehicle events as well as it handles indoor events. The technique of using multiple reference images is interesting and can be applied to other problems, e.g. handling repositioned furniture in indoor environments. For more detail on this method, see [Tserng, 1998].

2.4. Complex events

The AVS video monitoring technology enables the recognition of specific events such as when a person enters a room, deposits or picks up an object, or loiters for a while in a given area. Although these events are more sophisticated than those detected via simple motion detection, they are still unstructured events that are detected regardless of the context in which they occur. This can result in alarms being generated on events that are not of interest.

For example, if the system is monitoring a room or store with the intention of detecting theft, the system could be set up to generate an alarm whenever an object is picked up (i.e., whenever a REMOVE event occurs). However, no theft has occurred unless the person leaves the area with the object. A simple, unstructured event recognition system would generate an alarm every time someone picked up an object, resulting in many false alarms; whereas a system that can recognize complex events could be programmed to only generate an alarm when the REMOVE event is followed by an EXIT event. The EXIT event provides context for the REMOVE event that enables the system to filter out uninteresting cases in which the person does not leave the area with the object they picked up. This section describes the design and implementation of such a complex-event recognition system.

We use the term *simple event* to mean an unstructured atomic event. A *complex event* is structured, in that it is made up of one or more *sub-events*. The sub-events of a complex event may be simple events, or they may be complex, enabling the definition of event hierarchies. We will simply say *event* to refer to an event that may be either simple or complex. In our theft example above, REMOVE and EXIT are simple events, and THEFT is a complex event. A user may also define a further event, e.g., CRIME-SPREE, which may have one or more complex THEFT events as sub-events.

We created a user interface that enables definition of a complex event by constructing a list of sub-events. After one or more complex events have been defined, the sub-events of subsequently defined complex events can be complex events themselves.

2.4.1. Complex-event recognition

Once the user has defined the complex events and the actions to take when they occur, the event recognition system recognizes these events as they occur in the monitored area. For the purposes of this section, we assume *a priori* that the simple events can be recognized, and that the object involved in them can be tracked. In the implementation we will use the methods discussed in [Courtney, 1997, Olson and Brill, 1997] to track objects and recognize the simple events. In order to recognize a complex event, the system must keep a record of the sub-events that have occurred thus far, and the objects involved in them. Whenever the first sub-event in a complex event's sequence is recognized, an *activation* for that complex event is created. The activation contains the *ID* of the object involved in the event, and an *index*, which is the number of sub-events in the sequence that have been recognized thus far. The index is initialized to 1 when the activation is created, since the activation is only created when the first sub-event matches. The system maintains a list of current activations for each defined complex-event type. Whenever any new event is recognized, the list of current activations is consulted to see if the newly recognized (or *incoming*) event matches the next sub-event in the complex event. If so, the index is incremented. If the index reaches the total number of sub-events in the sequence, the complete complex event has been recognized, and any desired alarm can be generated. Also, since the complex event that was just recognized may also be a sub-event of another complex event, the activation lists are consulted again (recursively) to see if the indices of any other complex event activations can be advanced.

To return to our THEFT example, the complex THEFT event has two sub-events, REMOVE and EXIT. When a REMOVE event occurs, an activation for the THEFT event is created, containing the ID of the person involved in the REMOVE event, and an index set to 1. Later, when another event is recognized by the system, the activation is consulted to see if the event type of this new, incoming event matches the next sub-event in the sequence (in this case, EXIT). If the event type matches, the object ID is also checked, in this case to see if the person EXITing is the same as that of the person who REMOVEd the object earlier. This is to ensure that we do not signal a THEFT event when one person picks up an object and a different person exits the area. In a closed environment, the IDs used may merely be track-IDs, in which each object that enters the monitored area is assigned a unique track-ID, and the track-ID is discarded when the object is no longer being tracked. If both the event type and the object ID match, the activation's index is incremented to 2. Since there are only 2 sub-events in the complex event in this example, the entire complex-event has been recognized, and an alarm is generated if desired. Also, since the THEFT event has been recognized, this newly recognized THEFT event may be a sub-event of another complex event. When the complex THEFT event is recognized, the current activations are recursively checked to see if the theft is a part of another higher-level event, such as a CRIME-SPREE.

2.4.2. Variations and enhancements

We have described the basic mechanism of defining and recognizing complex events. There are several variations on this basic mechanism. One is to allow unordered events, i.e., complex events which are simply the conjunction or disjunction of their sub-events. Another is to allow negated sub-events, which can be used to cancel an activation when the negated sub-event occurs. For example, considering the definition for THEFT again, if the person pays for the item, it is not a theft. Also, if the person puts the item back down before leaving, no theft has occurred. A more complete definition of theft is one in which "a person picks up an item and then leaves without putting it back or paying." Assuming we can recognize the simple events REMOVE, DEPOSIT, PAY, and EXIT, the complex THEFT event can now be expressed as the ordered list (REMOVE, ~DEPOSIT, ~PAY, EXIT), where "~" indicates negation. Another application of the complex event with negated sub-events is to detect suspicious behavior in front of a building. The normal behavior may be for a person to park the car, get out of it, and then come up into the building. If the person parks the vehicle and leaves the area without coming up into the building, this may be a car bombing scenario. If we can detect the sub-events for PARK, OUTCAR, ENTER-BUILDING, and EXIT, we can define the car-bombing scenario as (PARK, OUTCAR, ~ENTER-BUILDING, EXIT).

Another variation is to allow the user to label the objects involved in the events, which facilitates the ability to specify that two object be different. Consider a different car bombing scenario in which two cars pull up in front of the building, and a person gets out of one car and into the other, which drives away. The event definition must specify that there are two *different* cars involved: the car-bomb and the getaway-car. This can be accomplished by labelling the object involved when defining the event, and giving different labels to objects which must be different.

Finally, one could allow multiple activations for the same event. For example, the desired behavior may be that a separate THEFT event should be signalled for each item stolen by a given person, e.g., if a person goes into a store and steals three things, three THEFT events are recognized. The basic mechanism described above signals a single THEFT event no matter how many objects are stolen. We can achieve the alternate behavior by creating multiple activations for a given event type, differing only in the ID's of the objects involved.

2.4.3. Implementation in AVS

We have described a method for defining and recognizing complex events. Most of this has been implemented and incorporated into the AVS system. This subsection describes the current implementation.

AVS analyzes the incoming video stream to detect and recognize events such as ENTER, EXIT, DEPOSIT, and REMOVE. The primary technique used by AVS for event recognition is motion graph matching as described in [Courtney, 1997]. The AVS system recognizes and reports these events in real time as illustrated in Figure 10. When the person enters the monitored area, an ENTER event is recognized as shown in the image on the left. When the person picks up an object, a REMOVE event is recognized, as depicted in the center image below. When the person exits the area, the EXIT

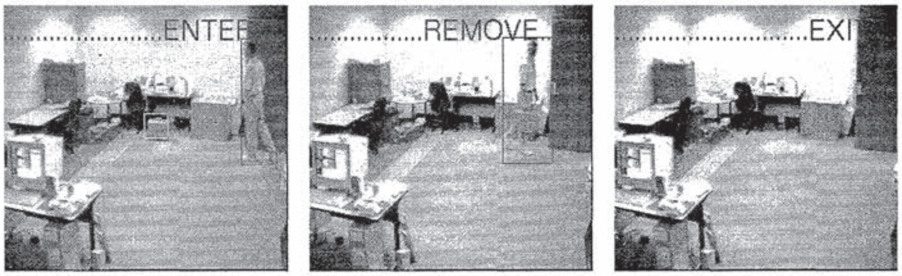


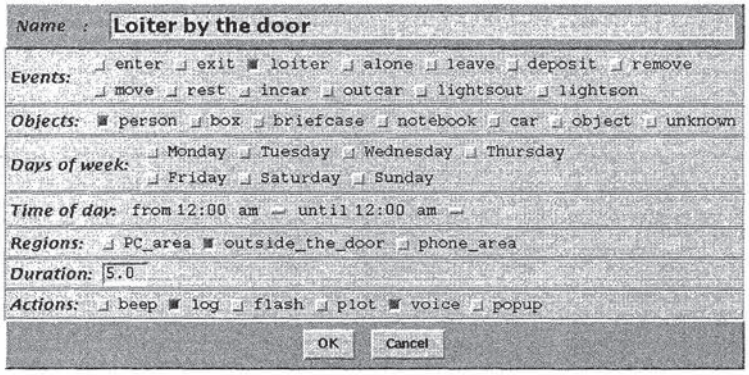
Figure 10: A series of simple events

event is signalled as shown in the image on the right

While the AVS system recognizes numerous events as shown above, the user can select which events are of interest by providing the dialog box interface illustrated in Figure 11. The user selects the event type, object type, time, location, and duration of the event of interest using a mouse. The user can also select an action for the AVS system to take when the event is recognized. This dialog box defines one type of simple event; an arbitrary number of different simple event types can be defined via multiple uses of the dialog box. The illustration in Figure 11 shows a dialog box defining an event called "Loiter by the door" which is triggered when a person loiters in the area near the door for more than 5 seconds.

AVS will generate a voice alarm and write a log entry when the specified event occurs. If the event is only being defined in order to be used as a sub-event in a complex event, the user might not check any action box, and no action will be taken when the event is recognized except to see if it matches the next sub-event in a complex-event activation, or generate a new activation if it matches the first sub-event in a complex event.

After one or more simple events have been defined, the user can define a complex event via the dialog box shown in Figure 12. This dialog box presents two lists: on the left is a scrolling list of all the event types that have been defined thus far, and on the right is a list of the sub-events of the complex event being defined. The sub-event list is initially blank when defining a new complex event. When the user double-clicks with the left mouse button on an item in the event list on the left, it is added as the next item in the sub-event list on the right. When the user double-clicks with the right mouse button on an item in the event list on the left, that item is also added to the sub-event list on the right, but as a negated sub-event. The event name is prefixed with a tilde (~) to indicate that the event is negated.



Name : **Loiter by the door**

Events: ☐ enter ☐ exit ☒ loiter ☐ alone ☐ leave ☐ deposit ☐ remove ☐ move ☐ rest ☐ incar ☐ outcar ☐ lightsout ☐ lightson

Objects: ☒ person ☐ box ☐ briefcase ☐ notebook ☐ car ☐ object ☐ unknown

Days of week: ☐ Monday ☐ Tuesday ☐ Wednesday ☐ Thursday ☐ Friday ☐ Saturday ☐ Sunday

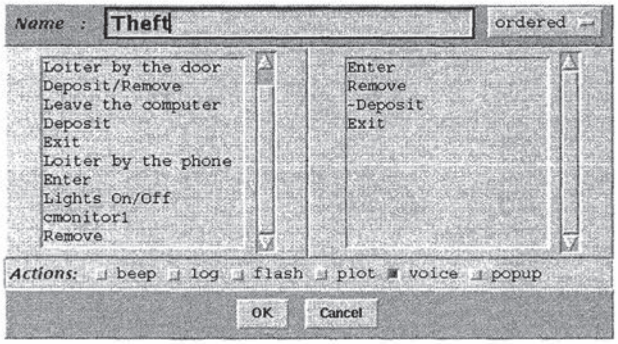
Time of day: from 12:00 am until 12:00 am

Regions: ☐ PC_area ☒ outside_the_door ☐ phone_area

Duration: 5.0

Actions: ☐ beep ☒ log ☐ flash ☐ plot ☒ voice ☐ popup

Figure 11: Selecting a type of simple event



Name : **Theft**

ordered

Loiter by the door
Deposit/Remove
Leave the computer
Deposit
Exit
Loiter by the phone
Enter
Lights On/Off
cmonitor1
Remove

Enter
Remove
~Deposit
Exit

Actions: ☐ beep ☐ log ☐ flash ☐ plot ☒ voice ☐ popup

Figure 12: Defining a complex event

In the upper right corner of the complex-event definition dialog box is an option menu via which the user indicates how the sub-events are to be combined. The default selection is "ordered" to indicate sequential processing of the sub-events. The other options are "all" and "any." If "all" is selected, the complex event will be signalled if all of the sub-events are matched, regardless of order, i.e., the complex event is simply the conjunction of the sub-events. If "any" is selected, the complex event occurs if any of the sub-events occurs, i.e., the complex event is the disjunction of the sub-events. At the bottom of the dialog box, the user can select the action to take when the complex event is recognized. The user can save the entire set of event definitions to a file so that they may be read back in at a later time.

Once a simple or complex event has been defined, the AVS system immediately begins recognition of the new events in real time, and taking the actions specified by the user. The AVS system, augmented as described, provides a functioning realization of the complex-event recognition method.

3. Advanced segmentation and tracking

In security applications, it is often necessary to track the movements of one or more people and objects in a scene monitored by a video camera. In real scenes, the objects move in unpredictable ways, may move close to one another, and may occlude each other. When a person moves, the shape of his or her image changes. These factors make it difficult to track the locations of individual objects throughout a scene containing multiple objects. The tracking capabilities of the original AVS system fail when there is mutual occlusion between the tracked objects. This section describes a new tracking method which overcomes this limitations of the previous tracking method, and maintains the integrity of the tracks of people even when they partially occlude one another.

The segmentation algorithm described here is related to tracking systems such as [Wren et al., 1997, Grimson et al., 1998, Cai et al., 1995] in that it extends the reference image to include a statistical model of the background. Our method further extends the tracking algorithm to reason explicitly about occlusion and maintain object tracks during mutual occlusion events. Unlike the capabilities described in previous sections, the new tracking method does not run in real time, and has not yet been integrated into the AVS system. Optimizations of the new method are expected to enable it to achieve real time operation in the future.

Figure 13 depicts an example scene containing two people. In (a), the two people are standing apart from each other, with Person-1 on the left, and Person-2 on the right. In (b), Person-1 moves to the right so that he is partially occluded by Person-2. Using a conventional technique such as background subtraction, it is difficult to maintain the separate tracks of the two people in the scene, since the images of the two people merge into a single large region.

Figure 14 shows a sequence of frames (in normal English reading order) in which it is particularly difficult to properly maintain the tracks of the two people in the scene. In this sequence, Person-2 moves from right to left and back again, crossing in front of Person-1. There are significant occlusions (e.g., in the third frame shown), and the orientations of both people with respect to the camera change significantly throughout the sequence,


(a) (b)

Figure 13: An example scene containing two people with occlusion

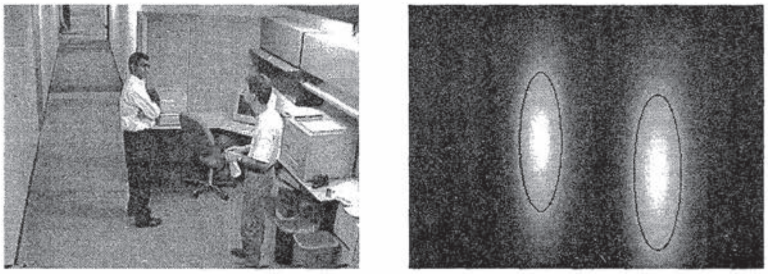
Figure 14: A difficult tracking sequence

making conventional template matching fail on this sequence.

A new tracking method is used to maintain tracks in sequences such as those depicted in Figures 13 and 14. The method maintains an estimate of the size and location of the objects being tracked, and creates an image which approximates the probability that the object intersects that pixel location. Figure 15b shows the probability images for the two person scene of Figure 13a, which is repeated here as 15a. The ellipse on the left indicates the estimated location of Person-1, and the ellipse on the right indicates the estimated location of Person-2. The brightness indicates the probability that the person's image intersects the given pixel, which is highest in the middle of the region, and falls off towards the edge. The black outlines represent the 50% probability contours. The size and shape of the regions are roughly the size and shape of a person standing at that location in the image.

We refer to the "person shaped" probability regions as *probabilistic templates* or simply *p-templates*. The path of the p-template through the scene represents the "track" of a given person which is maintained by the tracking system. P-templates can be used to reason about occlusion in a video sequence. While we only address the issue of p-templates for tracking people that are walking upright, the concept is applicable to tracking any object, e.g., vehicles and crawling people; although the shape of the p-template would need to be adapted to the type of object being tracked.

When the people in the scene overlap, the separate locations of the people can be maintained using the p-templates, and the region of partial occlusion can be detected. Figure 16 shows examples of such a situation. The two ellipses are maintained, even though the people are overlapping. The tracks of the people can be maintained through occlusions by tracking primarily on the basis of non-overlapping areas. This works for both the slight occlusion in Figures 16 (a) and (b), and often even for the very strong occlusions such as in Figures 16 (c) and (d). During the occlusions shown in Figure 14 and again in Figure 16 (c) and (d), the head of Person-1 is tracked, and the lower-body of Person-2 is tracked.



(a) (b)

Figure 15: Probability image for the locations of the people in the scene

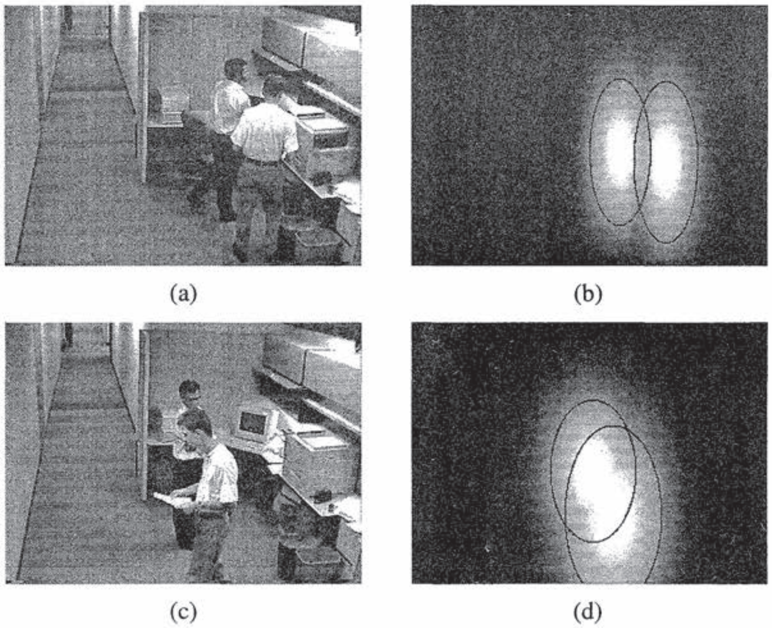


Figure 16: P-template images for partially occluding people

The new method requires a means of instantiating a new p-template when a person enters the scene, and updating the location of the region as the person moves through the scene. First we will describe the update mechanism, assuming that the p-templates have already been instantiated. The instantiation mechanism is described later.

The p-templates described above and depicted in Figures 15 and 16 represent the *prior* probabilities of the person locations, based on looking at the *previous* frame. These priors are then used to compute an estimate of the *posterior* probabilities of the person locations by looking at the new or *current* frame. The computation of the posterior probabilities takes into account both the prior probabilities and the information in the new frame. The posterior probabilities are used to update the locations of the people, and the new locations of the people are then used to compute the priors for the *next* frame.

Our current implementation computes the posteriors using a form of background differencing. Figure 17 shows the posteriors for the p-templates shown in Figure 16. Note that although there is significant overlap in the posterior estimates, especially in Figures 17 (e) and (f), there are significant differences in the brightnesses of the non-occluding areas. In Figure 17 (e), which represents the posteriors for Person-1, the head area of Person-1 is significantly brighter than in Figure 17 (f). Similarly, Figure 17 (f), which represents the posteriors for Person-2, is significantly brighter in the unoccluding area of Person-2's lower body.

Once the posteriors are computed, they are used to estimate the location of the tracked objects. In our implementation of a person tracker, we specifically need to estimate the location of the person's feet in the image, and their height in the image in pixels. Once the location and height are estimated, we can use the image-to-world coordinate transformation technique used in the original AVS system and described in [Olson and Brill, 1997]. That technique, called *quad-mapping*, computes the map locations of objects given the image locations of the bottom of the objects, e.g., in the case of a person, the location of the feet. Furthermore, if the scale of the map is known, the quad-mapping technique will estimate the size of the object, i.e., the height of a person being tracked.

If the lower portion of the p-template is unoccluded, foot locations are estimated directly from the image by looking at the bottom portion of the brightened region. If the upper portion is also unoccluded, the height can similarly obtained directly

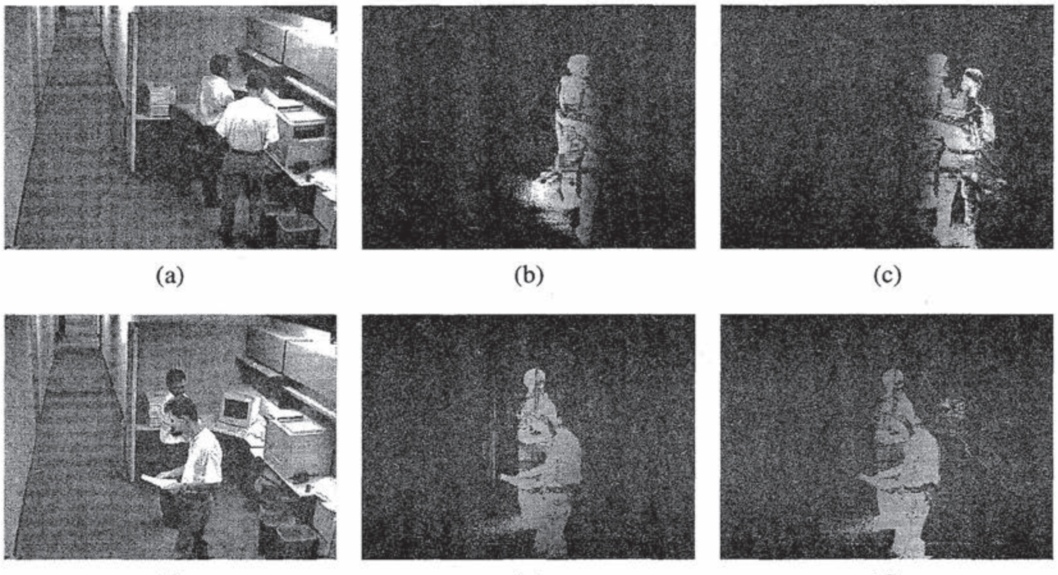


Figure 17: Posterior probability images for partially occluding people

from the image. If the upper part is occluded, but the lower part is not, the foot location is still determined directly from the image, but height is estimated using an estimate of the three-dimensional height of the person. The image height is then obtained by projecting the 3D height back into the image using the quad-mapping technique. If the lower portion is occluded, but the upper part is not, then the upper location is determined directly from the image, and then the 3D height is back-projected into the image to determine the foot location. If both the top and bottom are occluded, the location and height estimates are left unchanged from the previous frame.

Once the foot location and height of the person are computed, it is straightforward to compute the new location of the p-template, which is the Gaussian oval whose location and dimensions are determined by the foot location and image height computed above. The new p-template is then used to find the location of the person in the next frame, and the process repeats while the person remains in the scene.

A new p-template is instantiated whenever a new person enters the scene. Instantiation is best described in a Bayesian probabilistic framework. The p-templates constitute models of the objects in the environment. All of the pixels in the image are the result of a projection of some object in the environment—either from the background, or one of the people in the scene, or something else. The sum of the probabilities that the pixel is either from the background, from a person, or from "something else" must be 1.0. We maintain an "unknown" model to account for the probability that pixels may arise as a result of "something else." We compute the probability that each of the models caused the observed pixel value (where the unknown model is equally likely to produce any pixel value), and then use Bayes' formula to compute the inverse, i.e., the probability that the observed pixel value came from each of the models. When this computation is performed, for some of the pixels, the probability that the pixel came from the unknown model is the highest of all of the model probabilities. This results in a probability image for the unknown model, which represents pixels which probably came from something other than the objects the system knows about. At each frame, the probability image for the unknown model is computed, and this image is examined to see if adding a new person model would account for these unknown pixels. If so, a new person p-template is instantiated at the appropriate location, and the posteriors are recomputed.

Use of the procedure described above to track multiple people maintains tracks through occlusions where our previous technique could not. The robustness to occlusion of the new method enables video monitoring applications to improve tracking reliability in natural environments.

4. Best-view selection performance

Olson and Brill [1997] previously described the "best view selection" application of AVS technology. In this application, the system monitors and records the movements of humans in its field of view. For every person that it sees, it creates a log file that summarizes important information about the person, including a snapshot taken when the person was close to the camera and (if possible) facing it.

As the person is tracked through the scene, the tracker examines each image it captures of that person. If the new image is a better view of the person than the previously saved snapshot, the snapshot is replaced with the new view. In this manner, the system always contains the "best" view seen of the person thus far. When the person leaves the scene, the log entry is saved to a file. Each log entry records the time when the person entered the scene and a list of coordinate pairs showing their position in each video frame. The log entry also contains the "best" snapshot of the person while they were in the scene. Finally, the log entry file contains a pointer to the reference image that was in effect when the snapshot was taken. This information forms an extremely concise description of the person's movements and appearance while they were in the scene. An example of such a record in shown in Figure 18.

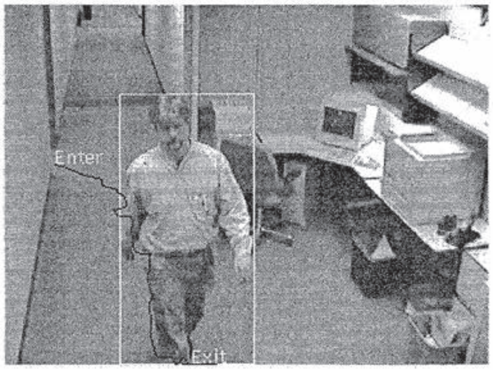


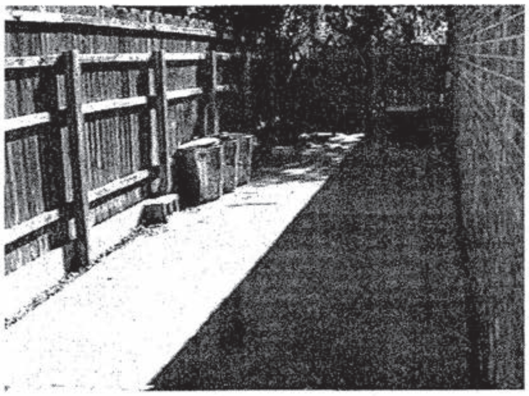
Figure 18: Example best view selection record

In an initial evaluation of this system, the system was installed in an uncontrolled office hallway and run for 118 hours. In this time, the system recorded 965 log entries in 35MB (uncompressed). The resulting records were examined to estimate the system performance, and we estimated 96% detection rate at 6% false alarm rate, with most errors due to segmentation and correspondence failure. However, for this initial experiment, there was no ground truth against which the performance could be measured.

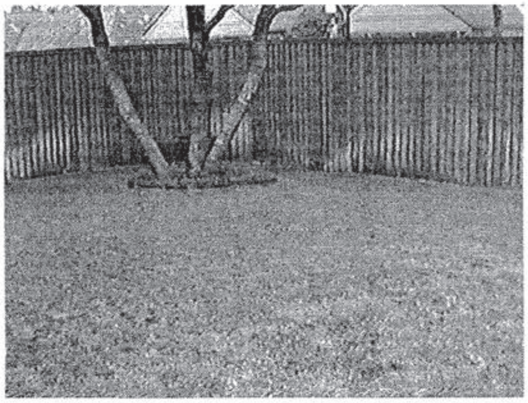
Recently, we have evaluated the system against ground truth observations. The performance of the system was initially evaluated on four hours of indoor video data. The video was manually annotated to obtain ground truth, and the surveillance system was evaluated against this ground truth. For situations in which only one person was in the scene, the system recorded exactly one record for each person, i.e., no person passed undetected though the field of view, and there is exactly one record for each such person. In the indoor condition, we observed a 100% detection rate.

For situations involving more than one person, the system occasionally failed to maintain track through partial occlusions. The result of this is that the system took extra pictures of these people when their track was re-acquired after the occlusion. On other occasions, the system failed to recognize that a motion region contained two people, and so it only took one picture that contained both people. We expect to reduce these errors via the use of the new tracking algorithms described above, once these algorithms are running in real time and are incorporated into the AVS system.

In order to evaluate and improve the system performance in outdoor monitoring environments, we have adopted an iterative research methodology in which we record representative videotape (2-3 hours), ground truth it with respect to the 'person events' that occur in the scene. One 'person event' is defined to be a video sequence in which one person enters monitored area completely, walking upright, and then exits field of view completely. We then run AVS system on the videotape and measure the false positives and negatives on person events. We then improve system as necessary to eliminate errors on video sequence and repeat the process.



(a)



(b)

Figure 19: Outdoor environments

Outdoor environments can be particularly difficult for video monitoring systems that operate based on change detection, due to the outdoor lighting variation. Figure 19 depicts two outdoor environments used to evaluate AVS best-view-selection performance. In Figure 19 (a), there is a strong shadow line running down the center of the field of view, which moves as the sun angle changes. The shadow motion here is sufficient to cause problems for a fixed background subtraction system within 5 minutes. There are also a number of trees in the background which move when the wind blows. Moreover, the shadows of these trees fall directly into the rear of the monitored area, and these shadows move with the wind as well. The shadow of the tree in Figure 19 (b) has a similar behavior. Cloud movement also causes large changes in brightness throughout the images.

Our initial outdoor evaluation was conducted in the environment depicted in Figure 19 (a). We captured two hours of outdoor video with extremely difficult imaging conditions caused by wind blown vegetation and strong shadows, which produced a large amount of "noise" motion. Additionally, the gate at the rear of the scene often blew open and closed. We manually ground-truthed the video to determine that a person entered the scene 20 times during the two hour sequence. The system recorded 16 of these events, for a detection rate of 75%. The undetected people were "lost in the noise." The system also produced 16 false detections in the two hour period, caused by noise from the moving shadows.

We were able to improve on this performance using our iterative research methodology to achieve a 100% detection rate for the 20 events in this two hour sequence. The system still recorded 8 false positives on this sequence. Four of these were caused by the gate blowing open and closed. The other four were cases in which the system lost track of the person in the field of view, and therefore took two pictures of the person, one before losing track, and another after picking up the track again. These cases are therefore more properly referred to as "extra pictures" rather than false positives.

Having achieved improved performance in the environment depicted in Figure 19 (a), we proceeded to test the system in the environment of Figure 19 (b). One three separate days we captured 1-2 hours of video, for a total of 4 hours of test video data in the environment of Figure 19 (b). We ground-truthed this video to determine that it contained 115 person events. The AVS system processed this video using the best-view-selection algorithm, and the results were compared to ground truth. We observed a 100% detection rate and a 2.6% false positive rate as a result of three false positives, all of which were "extra pictures."

In general, system performance was excellent in the indoor condition, with the exception of scenes containing multiple people, which produced extra records. We expect to address the multi-person problem using the p-template technique described in section 3. No person entered the scene without being recorded, even when there were multiple people. The system performance degrades in difficult outdoor lighting conditions, but it has improved significantly in recent work.

5. Conclusion

We have described several improvements in the video monitoring capabilities of the AVS system. Some improvements, such as vehicle event recognition, increase the functionality of the system to enable it to recognize new classes of events. Other improvements, such as the advanced segmentation and tracking, increase the robustness of the system's ability to recognize events in the presence of complications such as occlusion. We will continue to make improvements in the two categories of increased functionality and increased robustness. For the functionality improvements, we expect to recognize new classes of events, especially events regarding vehicles. For the robustness improvements, we are pursuing techniques that enable the system to be robust to lighting variation. As the techniques become more complex, additional effort will be needed to optimize the algorithms for real time operation. Our advanced segmentation and tracking will be the subject of optimization efforts in the near future.

References

[Cai, et al., 1995] Q. Cai, A. Mitiche, and J. K. Aggarwal. Tracking human motion in an indoor environment. In *Proc. of International Conference on Image Processing* (ICIP-95), 1995.

[Davis and Bobick, 1997] J. Davis and A. Bobick. Representation and recognition of human movement using temporal templates. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR-97)*, pp. 928-934, 1997.

[Courtney, 1997] J. D. Courtney. Automatic video indexing via object motion analysis. *Pattern Recognition*, **30**(4), April 1997

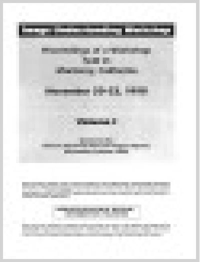
[Grimson et al., 1998] W. E. L. Grimson, C. Stauffer, R. Romano, and L. Lee. Using adaptive tracking to classify and monitor activities in a site. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR-98)*, 1998.

[Olson and Brill, 1997] T. J. Olson, and F. Z. Brill. Moving object detection and event recognition algorithms for smart cameras. *Proceedings of the 1997 Image Understanding Workshop*, New Orleans, LA, May 11-14, 1997, p. 159-175.

[Tserng, 1998] C. H. Tserng. Event recognition in scenes involving people and cars. Master's Thesis, MIT, May 1998.

[Wren et al., 1997] A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, July 1997, **19**(7), pp. 780-785.

EXHIBIT H



Related Names

- DARPA Image Understanding Workshop (26th : 1998 : Monterey, CA)
- Lukes, George E
- United States Defense Advanced Research Projects Agency Information Systems Office

Related Subjects

- Image Processing — Congresses
- Optical Pattern Recognition — Congresses
- Remote Sensing — Congresses
- Computer Vision — Congresses

Item Details (Item 1 of 1)

← Return to search results | Start over Add Star Item Actions

Image Understanding Workshop: Proceedings of a Workshop Held in Monterey, California, November 20-23, 1998

sponsored by Defense Advanced Research Projects Agency, Information Systems Office ; [edited by George E. Lukes]

Format	Book
Published	San Francisco, Calif. : Distributed by Morgan Kaufmann Publishers, [1998]
Language	English
Variant Title	Proceedings, 1998 Image Understanding Workshop, 20-23 November, Hyatt Regency, Monterey, CA

ISBN	1558605835
Description	2 v. : ill. ; 28 cm.
Notes	Includes bibliographical references and index.
Technical Details	Access in Virgo Classic Staff View

LEADER 01549nam a2200361Ia 4500

001			u2928081
003			SIRSI
005			19990805102441.0
008			981215s1998 cau 100 0 eng d
020			a| 1558605835
035			a| (Sirsi) i1558605835
035			a| (OCoLC)40497803
040			a| XRL c| XRL d| OCL
090			a| TA1632 b| .I57 1998
111	2		a| DARPA Image Understanding Workshop n| (26th : d| 1998 : c| Monterey, CA)
245	1	0	a| Image Understanding Workshop : b| proceedings of a workshop held in Monterey, California, November 20-23, 1998 / c| sponsored by Defense Advanced Research Projects Agency, Information Systems Office ; [edited by George E. Lukes].
246	1	4	a| Proceedings, 1998 Image Understanding Workshop, 20-23 November, Hyatt Regency, Monterey, CA
260			a| San Francisco, Calif. : b| Distributed by Morgan Kaufmann Publishers, c| [1998]
300			a| 2 v. : b| ill. ; c| 28 cm.
504			a| Includes bibliographical references and index.
596			a| 5
650		0	a| Image processing v| Congresses.
650		0	a| Optical pattern recognition v| Congresses.
650		0	a| Remote sensing v| Congresses.
650		0	a| Computer vision v| Congresses.
700	1		a| Lukes, George E.
994			a| Z0 b| VA@
710	1		a| United States. b| Defense Advanced Research Projects Agency. b| Information Systems Office.
852			b| SCI-ENG c| STACKS
866		0	8| 1 a| v.1-2
999			a| TA1632 .I57 1998 v.1 w| MONO-SER i| X004283491 l| STACKS m| SCI-ENG t|BOOK
999			a| TA1632 .I57 1998 v.2 w| MONO-SER i| X004283490 l| STACKS m| SCI-ENG t|BOOK

See fewer details

Availability

Request LEO delivery

Brown SEL

STACKS

v.1-2

Library	Location	Map	Availability	Call Number
Brown Science and Engineering	Stacks	N/A	AVAILABLE	TA1632 .I57 1998 v.1

Library	Location	Map	Availability	Call Number
Brown Science and Engineering	Stacks	N/A	AVAILABLE	TA1632 .I57 1998 v.2

EXHIBIT I

NCSU LIBRARIES

Library Catalog

Search for words: ▾ Search

☐ Search within results Start Over[1]

Image Understanding Workshop : proceedings of a workshop held in Monterey, California, November 20-23, 1998
sponsored by Defense Advanced Research Projects Agency, Information Systems Office ; [edited by George E. Lukes].

Author:	DARPA Image Understanding Workshop (26th : 1998 : Monterey, CA)[11]
Published:	San Francisco, Calif. : Distributed by Morgan Kaufmann Publishers, [1998]
Description:	2 v. : ill. ; 28 cm.
Format:	Book
Summary	Reports and technical results presented at the biennial workshops of the DARPA Image Understanding research program. (more) [5]

Browse Shelf [12]

Browse Related Subjects

- Image processing[13] -- Congresses[14]
- Optical pattern recognition[15] -- Congresses[16]
- Remote sensing[17] -- Congresses[18]
- Computer vision[19] -- Congresses[20]

MARC

000	01338cam a2200313Ia 4500		
003	OCoLC		
005	19990810132534.0		
008	981215s1998 caua b 100 0 eng d		
020			|a1558605835
035			|a(OCoLC)40497803
040			|aXRL |cXRL |dOCL
049			|aNRCC |xERJ
090			|aTA1632 |b.D35 1998
111	2		|aDARPA Image Understanding Workshop |n(26th : |d1998 : |cMonterey, CA)
245	1	0	|aImage Understanding Workshop : |bproceedings of a workshop held in Monterey, California, November 20-23, 1998 / |csponsored by Defense Advanced Research Projects Agency, Information Systems Office ; [edited by George E. Lukes].
246	1	4	|aProceedings, 1998 Image Understanding Workshop, 20-23 November, Hyatt Regency, Monterey, CA
260			|aSan Francisco, Calif. : |bDistributed by Morgan Kaufmann Publishers, |c[1998]
300			|a2 v. : |bill. ; |c28 cm.
504			|aIncludes bibliographical references and index.
596			|a14
650		0	|aImage processing |vCongresses.
650		0	|aOptical pattern recognition |vCongresses.
650		0	|aRemote sensing |vCongresses.
650		0	|aComputer vision |vCongresses.
700	1		|aLukes, George E.
710	1		|aUnited States. |bDefense Advanced Research Projects Agency. |bInformation Systems Office.
948			|a08/09/1999 |b10/06/1999
919			|aAKD-4329
918			|a1089089