FEATURE EXTRACTION METHODS FOR CHARACTER RECOGNITION — A SURVEY

ØIVIND DUE TRIER,*† ANIL K. JAIN,§ and TORFINN TAXT†

†Department of Informatics, University of Oslo, P.O.Box 1080 Blindern, N-0316 Oslo, Norway
§Department of Computer Science, Michigan State University, A714 Wells Hall, East Lansing, MI 48824–1027, USA

Revised July 19, 1995

Abstract— This paper presents an overview of feature extraction methods for off-line recognition of segmented (isolated) characters. Selection of a feature extraction method is probably the single most important factor in achieving high recognition performance in character recognition systems. Different feature extraction methods are designed for different representations of the characters, such as solid binary characters, character contours, skeletons (thinned characters), or gray level subimages of each individual character. The feature extraction methods are discussed in terms of invariance properties, reconstructability, and expected distortions and variability of the characters. The problem of choosing the appropriate feature extraction method for a given application is also discussed. When a few promising feature extraction methods have been identified, they need to be evaluated experimentally to find the best method for the given application.

Feature extraction Optical character recognition Character representation Invariance Reconstructability

1 Introduction

Optical character recognition (OCR) is one of the most successful applications of automatic pattern recognition. Since the mid 1950's, OCR has been a very active field for research and development [1]. Today, reasonably good OCR packages can be bought for as little as $100. However, these are only able to recognize high quality printed text documents or neatly written hand-printed text. The current research in OCR is now addressing documents that are not well handled by the available systems, including severely degraded, omnifont machine printed text, and (unconstrained) handwritten text. Also, efforts are being made to achieve lower substitution error rates and reject rates even on good quality machine printed text, since an experienced human typist still has a much lower error rate, albeit at a slower speed.

Selection of a feature extraction method is probably the single most important factor in achieving high recognition performance. Our own interest in character recognition is to recognize hand-printed digits in hydrographic maps (Fig. 1), but we have tried not to emphasize this particular application in the paper. Given the large number of feature extraction methods reported in the literature, a newcomer to the field is faced with the following question: Which feature extraction method is the best for a given application? This question led us to characterize the available feature extraction methods, so that the most promising methods could be sorted out. An experimental evaluation of these few promising methods must still be performed to select the best method for a specific application. In this process, one might find that a specific feature extraction method needs to be further developed.

A full performance evaluation of each method in terms of classification accuracy and speed is not within the scope of this review paper. In order to study performance issues, we will have to implement all the feature extraction methods, which is an enormous task. In addition, the performance also depends on the type of classifier used. Different feature types may need different types of classifiers. Also, the classification results reported in the literature are not comparable because they are based on different data sets.

Given the vast number of papers published on OCR every year, it is impossible to include all the available feature extraction methods in this survey. Instead, we have tried to make a representative selection to illustrate the different principles that can be used.

Two-dimensional object classification has several applications in addition to character recognition. These include airplane recognition [2], recognition of mechanical parts and tools [3], and tissue classification in medical imaging [4]. Several of the feature extraction techniques described in this pa-

*Author to whom correspondence should be addressed. This work was done while Trier was visiting Michigan State University. The paper appeared in *Pattern Recognition*, Vol. 29, No. 4, pp. 641–662, 1996

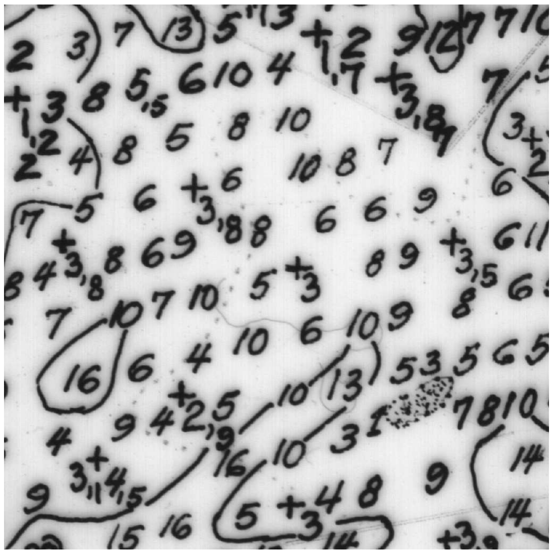

Figure 1: A gray scale image of a part of a hand-printed hydrographic map.

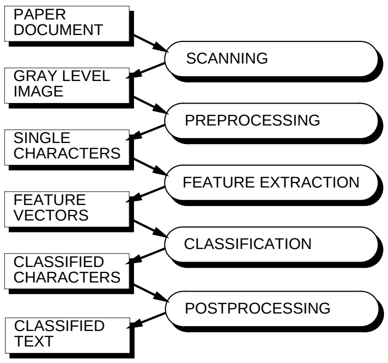

Figure 2: Steps in a character recognition system.

per for OCR have also been found to be useful in such applications.

An OCR system typically consists of the following processing steps (Fig. 2):

1. Gray level scanning at an appropriate resolution, typically 300–1000 dots per inch,
2. Preprocessing:
   (a) Binarization (two-level thresholding), using a global or a locally adaptive method,
   (b) Segmentation to isolate individual characters,
   (c) (Optional) conversion to another character representation (e.g., skeleton or contour curve),
3. Feature extraction
4. Recognition using one or more classifiers,
5. Contextual verification or postprocessing.

Survey papers [5–7], books [8–12], and evaluation studies [13–16] cover most of these subtasks, and several general surveys of OCR systems [1][17–22] also exist. However, to our knowledge, no thorough, up to date survey of feature extraction methods for OCR is available.

Devijver and Kittler define feature extraction (page 12 in [11]) as the problem of "extracting from the raw data the information which is most relevant for classification purposes, in the sense of minimizing the within-class pattern variability while enhancing the between-class pattern variability." It should be clear that different feature extraction methods fulfill this requirement to a varying degree, depending on the specific recognition problem and available data. A feature extraction method that proves to be successful in one application domain may turn out not to be very useful in another domain.

One could argue that there is only a limited number of independent features that can be extracted from a character image, so that which set of features is used is not so important. However, the extracted features must be invariant to the expected distortions and variations that the characters may have in a specific application. Also, the phenomenon called the *curse of dimensionality* [9, 23] cautions us that with a limited training set, the number of features must be kept reasonably small if a statistical classifier is to be used. A rule of thumb is to use five to ten times as many training patterns of each class as the dimensionality of the feature vector [23]. In practice, the requirements of a good feature extraction method makes selection of the best method for a given application a challenging task. One must also consider whether the characters to be recognized have known orientation and size, whether they are handwritten, machine printed or typed, and to what degree they are degraded. Also, more than one pattern class may be necessary to characterize characters that can be written in two or more distinct ways, as for example '*4*' and '*4*', and '*a*' and '*a*'.

Feature extraction is an important step in achieving good performance of OCR systems. However, the other steps in the system (Fig. 2) also need to be optimized to obtain the best possible performance, and these steps are not independent. The choice of feature extraction method limits or dictates the nature and output of the preprocessing step (Table 1). Some feature extraction methods work on gray level subimages of single characters (Fig. 3), while others work on solid 4-connected or 8-connected symbols segmented from the binary raster image (Fig. 4), thinned symbols or skeletons (Fig. 5), or symbol contours (Fig. 6). Further, the type or format of the extracted features must match the requirements of the chosen classifier. Graph

Table 1: Overview of feature extraction methods for the various representation forms (gray level, binary, vector).

Gray scale subimage	Binary		Vector (skeleton)
	solid character	outer contour	
Template matching	Template matching		Template matching
Deformable templates			Deformable templates
Unitary Transforms	Unitary transforms		Graph description
	Projection histograms	Contour profiles	Discrete features
Zoning	Zoning	Zoning	Zoning
Geometric moments	Geometric moments	Spline curve	
Zernike moments	Zernike moments	Fourier descriptors	Fourier descriptors

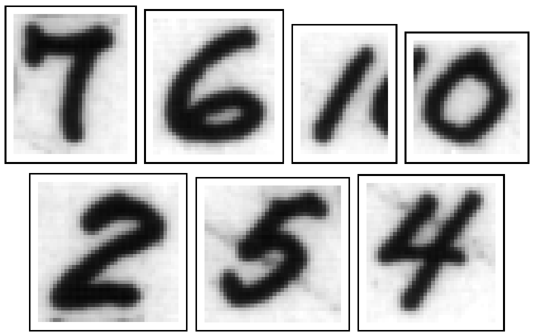

Figure 3: Gray scale subimages (≈ 30×30 pixels) of segmented characters. These digits were extracted from the top center portion of the map in Fig. 1. Note that for some of the digits, parts of other print objects are also present inside the character image.

descriptions or grammar-based descriptions of the characters are well suited for structural or syntactic classifiers. Discrete features that may assume only, say, two or three distinct values are ideal for decision trees. Real-valued feature vectors are ideal for statistical classifiers. However, multiple classifiers may be used, either as a multi-stage classification scheme [24, 25], or as parallel classifiers, where a combination of the individual classification results is used to decide the final classification [20, 26, 27]. In that case, features of more than one type or format may be extracted from the input characters.

1.1 Invariants

In order to recognize many variations of the same character, features that are invariant to certain transformations on the character need to be used. Invariants are features which have approximately the same values for samples of the same character that are, for example, translated, scaled, rotated, stretched, skewed, or mirrored (Fig. 7). However, not all variations among characters from the same character class (e.g., noise or degradation, and absence or presence of serifs) can be modelled by using

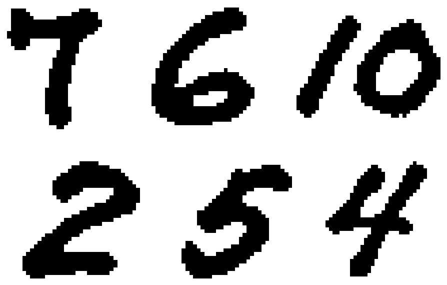

Figure 4: Digits from the hydrographic map in the binary raster representation.

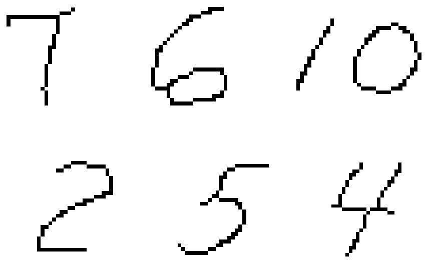

Figure 5: Skeletons of the digits in Fig. 4, thinned with the method of Zhang and Suen [28]. Note that junctions are displaced and a few short false branches occur.

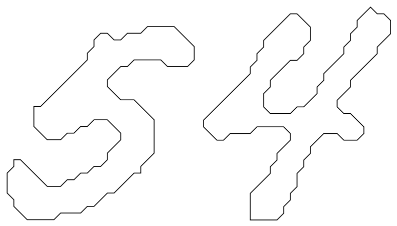

Figure 6: Contours of two of the digits in Fig. 4.

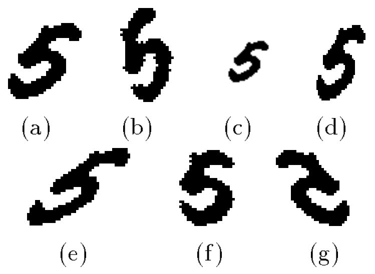

Figure 7: Transformed versions of digit '5'. **(a)** original, **(b)** rotated, **(c)** scaled, **(d)** stretched, **(e)** skewed, **(f)** de-skewed, **(g)** mirrored.

invariants.

Size- and translation invariance is easily achieved. The segmentation of individual characters can itself provide estimates of size and location, but the feature extraction method may often provide more accurate estimates.

Rotation invariance is important if the characters to be recognized can occur in any orientation. However, if all the characters are expected to have the same rotation, then rotation-variant features should be used to distinguish between such characters as '6' and '9', and 'n' and 'u'. Another alternative is to use rotation-invariant features, augmented with the detected rotation angle. If the rotation angle is restricted, say, to lie between $-45°$ and $45°$, characters that are, say $180°$ rotations of each other can be differentiated. The same principle may be used for size-invariant features, if one wants to recognize punctuation marks in addition to characters, and wants to distinguish between, say, '.', 'o', and 'O'; and ',' and '9'.

Skew-invariance may be useful for hand-printed text, where the characters may be more or less slanted, and multifont machine printed text, where some fonts are slanted and some are not. Invariance to mirror images is not desirable in character recognition, as the mirror image of a character may produce an illegitimate symbol or a different character.

For features extracted from gray scale subimages, invariance to contrast between print and background and to mean gray level may be needed, in addition to the other invariants mentioned above. Invariance to mean gray level is easily obtained by adding to each pixel the difference of the desired and the actual mean gray levels of the image [29].

If invariant features can not be found, an alternative is to normalize the input images to have standard size, rotation, contrast, and so on. However, one should keep in mind that this introduces new discretization errors.

1.2 Reconstructability

For some feature extraction methods, the characters can be reconstructed from the extracted features [30, 31]. This property ensures that complete information about the character shape is present in the extracted features. Although, for some methods, exact reconstruction may require an arbitrarily large number of features, reasonable approximations of the original character shape can usually be obtained by using only a small number of features with the highest information content. The hope is that these features also have high discrimination power.

By reconstructing the character images from the extracted features, one may visually check that a sufficient number of features is used to capture the essential structure of the characters. Reconstruction may also be used to informally control that the implementation is correct.

The rest of the paper is organized as follows. Sections 2–5 give a detailed review of feature extraction methods, grouped by the various representation forms of the characters. A short summary on neural network classifiers is given in Section 6. Section 7 gives guidelines for how one should choose the appropriate feature extraction method for a given application. Finally, a summary is given in Section 7.

2 Features Extracted From Gray Scale Images

A major challenge in gray scale image-based methods is to locate candidate character locations. One can use a locally adaptive binarization method to obtain a good binary raster image, and use connected components of the expected character size to locate the candidate characters. However, a gray scale-based method is typically used when recognition based on the binary raster representation fails, so the localization problem remains unsolved for difficult images. One may have to resort to the brute force approach of trying all possible locations in the image. However, one then has to assume a standard size for a character image, as the combination of all character sizes and locations is computationally prohibitive. This approach can not be used if the character size is expected to vary.

The desired result of the localization or segmentation step is a subimage containing one character, and, except for background pixels, *no other objects*. However, when print objects appear very close to each other in the input image, this goal can not always be achieved. Often, other characters or print objects may accidentally occur inside the subimage (Fig. 3), possibly distorting the extracted features. This is one of the reasons why every character recognition system has a *reject* option.

2.1 Template matching

We are not aware of OCR systems using template matching on gray scale character images. However, since template matching is a fairly standard image processing technique [32, 33], we have included this section for completeness.

In template matching the feature extraction step is left out altogether, and the character image itself is used as a "feature vector". In the recognition stage, a similarity (or dissimilarity) measure between each template T_j and the character image Z is computed. The template T_k which has the highest similarity measure is identified, and if this similarity is above a specified threshold, then the character is assigned the class label k. Else, the character remains unclassified. In the case of a dissimilarity measure, the template T_k having the *lowest* dissimilarity measure is identified, and if the dissimilarity is *below* a specified threshold, the character is given the class label k.

A common dissimilarity measure is the *mean square distance* D (Eq. 20.1-1 in Pratt [33]):

$$D_j = \sum_{i=1}^{M} (Z(x_i, y_i) - T_j(x_i, y_i))^2, \qquad (1)$$

where it is assumed that the template and the input character image are of the same size, and the sum is taken over the M pixels in the image.

Eqn. (1) can be rewritten as

$$D_j = E_Z - 2R_{ZT_j} + E_{T_j}, \qquad (2)$$

where

$$E_Z = \sum_{i=1}^{M} \left(Z^2(x_i, y_i)\right), \qquad (3)$$

$$R_{ZT_j} = \sum_{i=1}^{M} (Z(x_i, y_i) T_j(x_i, y_i)), \qquad (4)$$

$$E_{T_j} = \sum_{i=1}^{M} \left(T_j^2(x_i, y_i)\right). \qquad (5)$$

E_Z and E_{T_j} are the total character image energy and the total template energy, respectively. R_{ZT_j} is the cross-correlation between the character and the template, and could have been used as a similarity measure, but Pratt [33] points out that R_{ZT_j} may detect a false match if, say, Z contains mostly high values. In that case, E_Z also has a high value, and it could be used to normalize R_{ZT_j} by the expression $\tilde{R}_{ZT_j} = R_{ZT_j}/E_Z$. However, in Pratt's formulation of template matching, one wants to decide whether the template is present in the image (and get the locations of each occurrence). Our problem is the opposite: find the template that matches the character image best. Therefore, it is more relevant to normalize the cross-correlation by dividing it with the total template energy:

$$\hat{R}_{ZT_j} = \frac{R_{ZT_j}}{E_{T_j}}. \qquad (6)$$

Experiments are needed to decide wether D_j or $\hat{R}_{ZT_j}$ should be used for OCR.

Although simple, template matching suffers from some obvious limitations. One template is only capable of recognizing characters of the same size and rotation, is not illumination-invariant (invariant to contrast and to mean gray level), and is very vulnerable to noise and small variations that occur among characters from the same class. However, many templates may be used for each character class, but at the cost of higher computational time since every input character has to be compared with every template. The character candidates in the input image can be scaled to suit the template sizes, thus making the recognizer scale-independent.

2.2 Deformable Templates

Deformable templates have been used extensively in several object recognition applications [34, 35]. Recently, Del Bimbo et al. [36] proposed to use deformable templates for character recognition in gray scale images of credit card slips with poor print quality. The templates used were character skeletons. It is not clear how the initial positions of the templates were chosen. If all possible positions in the image were to be tried, then the computational time would be prohibitive.

2.3 Unitary Image Transforms

In template matching, all the pixels in the gray scale character image are used as features. Andrews [37] applies a unitary transform to character images, obtaining a reduction in the number of features while preserving most of the information about the character shape. In the transformed space, the pixels are ordered by their variance, and the pixels with the highest variance are used as features. The unitary transform has to be applied to a training set to obtain estimates of the variances of the pixels in the transformed space. Andrews investigated the Karhunen-Loeve (KL), Fourier, Hadamard (or Walsh), and Haar transforms in 1971 [37]. He concluded that the KL transform was too computationally demanding, so he recommended to use the Fourier or Hadamard transforms. However, the KL transform is the only (mean-squared error) optimal unitary transform in terms of information compression [38]. When the KL transform is used, the same amount of information about the input character image is contained in fewer features compared to any other unitary transform.

Other unitary transforms include the Cosine, Sine, and Slant transforms [38]. It has been shown that the Cosine transform is better in terms of information compression (e.g., see pp. 375–379 in [38]) than the other non-optimal unitary transforms. Its computational cost is comparable to that of the fast Fourier transform, so the Cosine transform has been coined "the method of choice for

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.