Tutorial

Character Recognition Systems for the non-expert

by Nawwaf N. Kharma & Rabab K. Ward
University of British Columbia

This tutorial paper presents an overview of the field of character recognition by providing answers to the following questions:
- What does a character recognition system do?
- How does it do it i.e. what are its functional components?

The answers are meant to shed some light onto the field. Finally, what the authors believe are the two main open problems of character recognition are briefly described.

Cet article présente un aperçu des technologies de reconnaissance de caractères en vous fournissant les réponses aux questions suivantes:

- Quel est le rôle d'un système de reconnaissance de caractères?

- Quel en est le fonctionnement?

Les réponses vous aideront à comprendre cette nouvelle technologie. De plus, deux gros problèmes connus de la reconnaissance de caractères seront décrits brièvement.

1.0 What is a Character Recognition System?

Character recognition is the processing by machine of (2-dimensional) text-based input patterns to produce some meaningful output. As such, character recognition systems are a subset of pattern recognition systems.

Input may come from on-line or off-line devices. On-line devices are stylus based, and they include tablet-displays and digitizing tablets. They are able to provide the temporal order of the points, which make up the lines of text. Some tablets provide additional information, such as speed (of writing) and pressure (exerted by the writer). Off-line devices, on the other hand, include scanners of the flat-bed, paper-fed, and hand-held types. They return an image as (basically) a bit-map of pixels.

A character recognition system (CRS) accepts the output of the on/off-line device as input, processes it, and then produces some meaningful output. Possible output forms includes: sequences of symbols (e.g. 'Y E S'), the date on a cheque (e.g. 'Feb. 14, '94'), and the validity or otherwise of a signature.

2.0 What are the 'Functional' Components of a Character Recognition System?

Functionally, a CRS may be broken up into components. One component carries out 'pre-processing' functions, such as normalization [1] and thinning [2]. Another component accepts the pre-processed input pattern and extracts characteristic features [3]. The extracted features are used by a 'classification' component (such as a Neural Net [4]) to assign a label to the pattern. All functions carried out after (initial) classification fall under 'post-processing'. It is worth noting that the functional components are:

- Not present in every CRS. A CRS may carry out the function of classification without first explicitly extracting features by (for example) using some form of Template Matching [5].
- Not always implemented as mutually exclusive components. For example, a software object may extract features and classify simultaneously [6].
- Not necessarily sequentially executed. Indeed, in many applications, a substantial amount of feature extraction is carried out prior to segmentation.

Nevertheless, the greater majority of CRSs include at least three of the four functions described above. Each of the four functional components of a CRS are described below in more detail.

2.1 Pre-Processing

Pre-processing covers all those functions carried out prior to feature extraction to produce a 'cleaned-up' version of the original image so that it can be used directly and efficiently by the feature extraction component of the CRS. Hence, pre-processing includes the following functions:

A. Noise Reduction (Figure 1)

Noise is a random error in pixel value, usually introduced as a result of reproduction, digitalization and transmission of the original image. Noise may be placed under three categories: signal-independent, signal-dependent, and salt & pepper noise. Noise cannot always be totally eliminated; but smoothing is a widely used procedure for replacing the value of a pixel by the average of the values of the pixels around (and including) the original pixel. This, in scanned images may cause blurring, and when applied to on-line text causes end-point clipping.

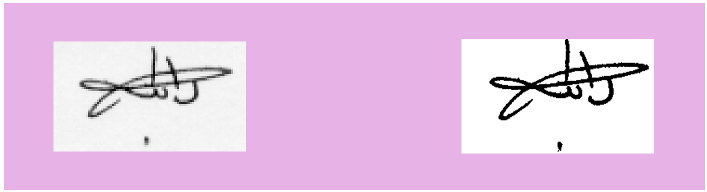


Figure 1: An Image of a Signature before and after noise reduction

B. Skeletonisation (Figure 2)

Text consists of lines. These lines may be 1-point thick, as is the case with most on-line sources, such as notepads. Line images coming from scanners, however, are normally several points thick. Most relevant information in lines is not related to the thickness of the line. Hence, thinning of lines by removing all redundant pixels, until they become just 1-point thick can be a very useful procedure. The question becomes; what are the redundant pixels, and how can they be removed from the original line?

In general, a thinning procedure is judged by how well it is able to control lines of the original image without at the same time:

- Fragmenting a previously continuous line by breaking it into a number of isolated lines,
- Clipping the ends of the central line,
- Introducing new features (e.g. a cusp) which were not there originally, or
- Eliminating/replacing a feature (i.e. by replacing a loop with a single line).

A good algorithm for thinning is described in [7]. This algorithm basically peals off layers of pixels from the boundary of the original line image, while avoiding end-point clipping and line fragmentation.

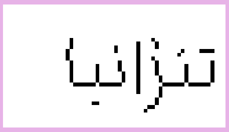


Figure 2: The Word 'Tanzania' in Arabic, after the application of a Thinning Algorithm [8]

C. Normalization

Patterns (i.e. words) can have different sizes, appear at different positions (within an image), and are often rotated by up to 180 degrees. It is often required to carry out a normalization operation before any feature extraction (or pattern matching) is carried out. Normalization routines may be broken down into the following groups.

- Moment Invariant Techniques [9].
- Fourier Descriptors [10].
- Boundary-Based Techniques [1].
- Vector Analysis [11].

These routines have in common the following features:

- They normalize the character size by dividing whatever size-related feature they are using by the total length of the character.
- They normalize the position of the character by moving the centre of co-ordinates to a point, which is at a fixed position on/about the character, e.g. the centroid, or the starting point of that character.

Normalizing the orientation of the character is, however, a more complicated procedure than the two above, and one that is done in fundamentally varied ways.

D. Segmentation (Figure 3)

Characters can be written cursively. They may also overlap. For a CRS that is required to identify individual characters (as opposed to just whole words) there is a need to identify where (roughly) a character starts and ends. This is essentially what segmentation aims to do. There are various methods for tackling the segmentation problem:

- Pre-segmentation (often) means characters that arrive already separated from each other. This is normally the case when the text is printed, or when the writer is required to write the characters in boxes or without connecting them together (e.g. the Palm III).
- Finding Gaps; to find out the gaps between the letters or, at least, the connecting lines. For an overview of gap based techniques, such as: stograms [16] bounding boxes, run-length, and convex hulls, see [13]. All these techniques function by analyzing the geometric relationships between the various components of the text.
- There are systems [14] which classify without explicitly segmenting the word.

.

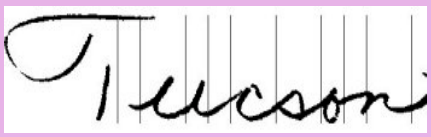
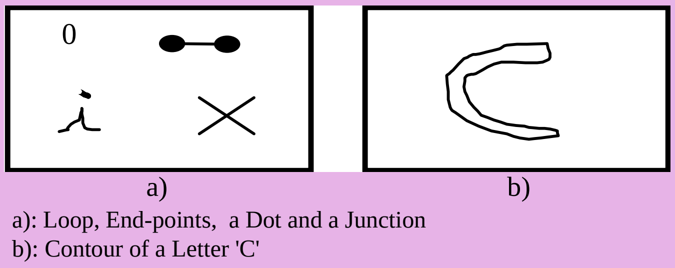


Figure 3: A Word Segmented using Algorithm (after all incorrect segmentation points are removed)

2.2 Feature Extraction

Feature extraction is one of the two most basic functions of a CRS. It involves measuring those features of the (cleaned-up) input pattern that are relevant to classification. After feature extraction, the pattern is represented by the set of extracted features.



0
a)
b)
a): Loop, End-points, a Dot and a Junction
b): Contour of a Letter 'C'

Figure 4 (a & b): Some Features of Characters

There is an infinite number of potential features that one can extract from a finite 2D pattern. However, only those features that are of possible relevance to classification need to be considered. This entails that during the design stage, the expert is focused on those features, which, given a certain classification technique, will produce the most certain and efficient classification results.

For example, in a 2-symbol alphabet, containing '0' and '1', the height of the input pattern (digit) is not a differentiating feature and hence is insignificant. On the other hand, the number of acute angles in the pattern is (potentially) a differentiating feature, with '0' having none and '1' having exactly one (in their printed forms).

Various types of features proposed in the literature include:

- Horizontal and vertical histograms.
- Curvature information (e.g. slope), and local extrema of curvature (of the line making-up/fitting a word) [15].
- Topological features, such as Loops (a group of white pixels surrounded by black ones), end-points (points with exactly 1 neighboring point), dots (a cluster of, say, 1-3 pixels), and junctions (points with more than 2 neighbors) - all in thinned black & white images.
- Parameters of polynomial (or other) curve-fitting functions [30].
- Contour information. Where a contour is the outside boundary of a pattern- see Fig. 4 (b).

Abstractly, feature extraction maps the whole of each input pattern from its original spatial (e.g. Euclidean) system of coordinates onto a single point in a 'feature' space. This feature space is defined by the N extracted features, and hence has N dimensions. The N axes delineating the feature space are orthogonal, only if the features are independent of each other (such as the height and width of a digit).

In summary, if the goal of feature extraction is to map input patterns onto points in feature space (see Fig. 5), then the purpose of classification is to assign each point in the space with a correct label (e.g. a '1'). Hence, once a pattern is mapped, the problem becomes one of classical classification.

.

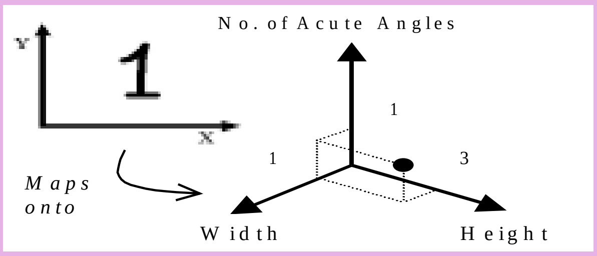


Figure 5: Feature Extraction as Mapping from Euclidean to Feature Space

2.3 Pattern Classification

Once an input pattern is mapped onto a point in feature space, the next step is to label each of the points. Classification techniques include:

- Rule-based systems
- Decision Trees [18]
- Clustering techniques [19]
- Artificial Neural Networks
- Hidden Markov Models [20]

All the above techniques share one common characteristic: they all divide the feature space into smaller sub-spaces, each of which typically contains points of the same classification. Some of the above techniques are explained in more detail below.

- **Rule-based systems** (including Expert Systems) typically use a set of If-Then rules to decide the class of a pattern based on how well the conditions in the If-part fit the pattern. In rule-based systems, it is possible for two (or more) rules (with different classification recommendations) to be applicable to the same input pattern. This causes conflicts, and hence calls for conflict-resolution machinery. Again, in a CRS for a '0'/'1' alphabet, one can imagine a rule such as: IF (pattern has big loop) THEN (class = '0')
- **Decision Trees** (DT's) may be viewed as a tree data structure used for decision making. A tree has a single entry point at its top, and any number of single-class leaf nodes at its bottom. ID3 [18] is a very popular method for automatically building a DT from a pre-classified set of patterns. Once a decision tree is constructed, it can be used to classify new (unknown) patterns.

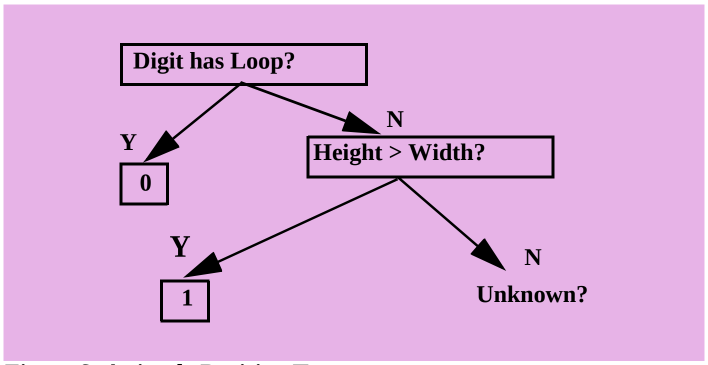


Figure 6: A simple Decision Tree

- **Clustering** basically attempts to look for points in feature space that are close to each other and place them in the same class. One way of doing this is to arbitrarily assign a class to each point in a set of unclassified points, then find the centre (or mean) of each group of similarly classed points. Hence, each point is re-assigned to the class of the centre-point closest to it. This is followed by a re-computation of the centre points of the various classes, and so on the process repeats (iteratively) until no more re-assignment is necessary (see K-means in [21]).

2.4 Post-processing

Post-processing, covers verification, action execution, and adaptation. The goal of verification is to increase the level of confidence in the classification made. This is done in various ways. One way is to use a database of 2 or 3 letter combinations to check that the sequence of letters recognized does not contain impossible combinations (e.g., 'zdh'). Alternatively, a word dictionary could be used to check that a certain string of characters constitutes a valid word. This method, in general, suffers from the shortcoming that correct words not in the dictionary will get rejected. In addition to letter/word dictionaries, a higher-level formal grammatical model can be used to verify the correctness of whole phrases or sentences [4].

Besides verification, a CRS system always carries out some action in response to recognition. For example, the system in [22] attempts to deduce certain personal characteristics of the writer by applying a rule-base to the set of features extracted from a sample of his/her own handwriting (e.g. an increasing left margin entails a tendency to fatigue as work progresses).

Also, most interestingly, some advanced CRS's alter their own weights (ANN's), or probabilistic parameters (HMM's) in an act of adaptation to reduce the gap between expected and actual performance, in an effort to improve future performance.

3.0 Summation & Future Challenges

The problem of character recognition is a sub-set of pattern recognition in that it is confined to text-based patterns. The aim of all character recognition systems is to automatically extract some meaning out of a 2D image (or trace) of some text-based input. There are many types of CRS's. Some read cheques, others recognize printed words from a scanned image. All, however, may be viewed as consisting of four functional parts.

The four functional parts of a CRS are: pre-processing, feature extraction, pattern classification, and post-processing. Not all character recognition systems have all these parts, and some have additional components. Almost all, however, must somehow measure features, and decide on a meaningful class for the input pattern.

Regardless of the techniques used, all character recognition methods, the authors believe, face two big problems: segmentation, and adaptation. Segmentation, or the lack of it, is the biggest impediment in the face of designers trying to build a totally unrestricted character recognition system. Characters overlap, words overlap, and unwanted info. (e.g. noise) overlaps both. It is quite hard, however, to determine where a character/word starts and finishes, without recognizing the character/word, in the first place. But, of course, segmentation is often needed prior to recognition (or classification). Hence, we have a chicken-and-egg type problem, which, the authors believe, can only be solved either incrementally, or by not requiring segmentation in the first place.

The other big remaining open problem in character recognition, is that of adaptation, especially in the absence of direct corrective feedback (from a human being). This means that the learning would have to be unsupervised, and hence, uncertain. This is because the machine will have to be required to decide for itself where and what error in recognition has occurred; a task, that in the best of circumstances can be hard. There are, however, a good number of unsupervised machine learning techniques available in the literature [24], which may prove of use to researchers of this problem. Both problems, just described, are intimately related to Mori's level 3-1 problems [25].

4.0 References

[1]. S. Di Zenzo et al. (1992): “Optical Recognition of Hand-printed Characters of any Size, Position, and Orientation”, in IBM Journal of Research & Development. Vol. 36, No. 3.

[2]. Sabri A. Mahmoud (1991): “Skeletonization of Arabic Characters using Clustering Based Skeletonisation Algorithm (CBSA)”, in Pattern Recognition, Vol. 24, No. 5, pp. 453-464. Pergamon Press.

[3]. A. C. Downton & S. Impedovo (ed.'s) (1997): “Progress in Handwriting Recognition”, Feature Extraction chapter, published by World Scientific.

[4]. J J Hull (1994): “Language Level Syntactic and Semantic Constraints Applied to Visual Word Recognition”, in “Fundamentals of Handwriting Recognition”, by Sebastiano Impedovo (ed.), published by Springer-Verlag.

[5]. Sargur Srihari (1997): “Recent Advances in Off-line Handwriting Recognition at CEDAR”, in A. C. Downton & S. Impedovo (ed.'s): “Progress in Handwriting Recognition”, published by World Scientific.

[6]. Boris Alexandrovsky (1997): “A Thalamocortical Algorithm That Performs Handwritten Character Recognition”, in A. C. Downton & S. Impedovo (ed.'s): “Progress in Handwriting Recognition”, published by World Scientific.

[7]. T Y Zhyang & C Y Suen (1985): “A Fast Parallel Algorithm for Thinning Digital Patterns”, in Communications of the ACM, Vol. 27, No. 3.

[8]. M. Ahmed & R. Ward, “A rule-based system for thinning symbols to their central lines,” submitted to the IEEE journal of PAMI in June 1998.

[9]. Bailey R R and Srinath M (1996): “Orthogonal Moment Features for Use with Parametric and Non-Parametric Classifiers”, in IEEE Transactions on Pattern Analysis and Machine Intelligence; Vol.18, No.4.

[10]. Kauppinen, et al (1995): “An experimental comparison of autoregressive and Fourier-based descriptors in 2D shape classification”, in IEEE Transactions on Pattern Analysis and Machine Intelligence; Vol.17, No.2.

[11]. Wilfong G, et al (1996): “On-Line Recognition of Handwritten Symbols”, in IEEE Transactions on Pattern Analysis and Machine Intelligence; Vol.18, No. 9.

[12]. H Al-Yousefi & S S Udpa (1992): “Recognition of Arabic Characters”, in IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 14, No. 8.

[13]. U Mahedavan & S N Srihari (1995): “Gap Metrics for Word Separation in Handwritten Lines”, in the 3rd International Conference on Document Analysis and Recognition, Montreal, Canada.

[14]. T Caesra et al. (1994): “Handwriting Recognition by statistical Methods”, in “Fundamentals of Handwriting Recognition”, by Sebastiano Impedovo (ed.), published by Springer-Verlag.

[15]. X Li, R Parizeau, & R Plamondon (1997): “Detection of Extreme Points of On-Line Handwritten scripts”, in A. C. Downton & S. Impedovo (ed.'s): “Progress in Handwriting Recognition”, published by World Scientific.

[16]. H Beigi (1997): “Pre-Processing the Dynamics of On-line Handwriting Data, Feature Extraction and Recognition”, in A. C. Downton & S. Impedovo (ed.'s): “Progress in Handwriting Recognition”, published by World Scientific.

[17]. O Due Trier, et al. (1995): “Feature Extraction Methods for Character Recognition”, in Pattern Recognition, Vol. 29, No.4.

[18]. J R Quinlan (1986): “Induction of Decision Trees”, in Machine Learning, 1:81-106.

[19]. http://www.ee.ic.ac.uk/hp/staff/sjrob/ Projects/cluster.html. Last Accessed: June 15th, 1999.

[20]. L R Rabiner (1989): “A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition”, in Proceedings of the IEEE, Vol. 77, No. 2.

[21]. K. Fukunaga (1990): “Introduction to Statistical Pattern Recognition”, published by Academic Press.

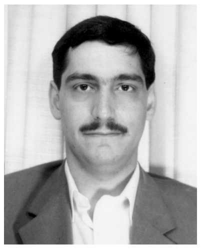
[22]. G Sheikholeslami et al (1997): “Computer Aided Graphology”, in A. C. Downton & S. Impedovo (ed.'s): “Progress in Handwriting Recognition”, published by World Scientific.

[23]. M. Blumenstein, and B. Verma (1998): “A Neural Based Segmentation and Recognition Technique for Handwritten Words”, World Congress on Computational Intelligence, Anchorage, Alaska.

[24]. G Brisco & T Caelli (1996): “A Compendium of Machine Learning, Volume 1: Symbolic Machine Learning”, published by Ablex Publishing Corporation, Norwood, NJ.

[25]. S. Mori (1994): “Historical Review of Theiry and Practice of Handwritten Character Recognition”, in “Fundamentals of Handwriting Recognition”, by Sebastiano Impedovo (ed.), published by Springer-Verlag.

About the Authors



Nawwaf Kharma is currently working as a lecturer at the Electrical & Computer Eng. Dept. of the University of British Columbia in Vancouver, where he carries out research in on-line character recognition, and genetic optimization of pattern recognition systems. He had previously worked as an assistant professor, teaching AI and Software Engineering at the University of Paisley in Scotland. His Ph.D. work, which was carried out at Imperial College London, involved the development of a psychologically inspired incremental machine learning algorithm for robotic applications.

Rabab Ward is a professor in the Electrical & Computer Eng. Dept. of the University of British Columbia and the Director of the Centre for Integrated Computer Systems Research (CICSR). She is the chief scientist of Ward Laboratories Inc., a company incorporated in BC to transfer the technology from her laboratory to industry. Her expertise lies in digital signal processing and applications to cable TV, HDTV, video compression and medical images, including mammography, microscopy and cell images. She also holds six patents and has published around 150 papers and chapters in scientific books.

Professional Engineers
Ontario

Engineering Association Embraces Software Practitioners

Toronto - (September 7, 1999) - Professional Engineers Ontario (PEO), the regulatory body for engineering in Ontario, announced today it will license, as professional engineers, software practitioners who meet specific criteria. Now, individuals whose work experience is mainly in the area of software design and development, but whose academic background is in something other than an accredited computer engineering or other information technology-related engineering program, will be eligible for licensure, provided they meet other licensing requirements.

PEO strongly supports development by university faculties of engineering of software engineering programs designed to meet the national accreditation standards set by the Canadian Engineering Accreditation Board, and PEO's licensing criteria for software practitioners.

For additional information, please contact:

Dave Pearce
Communications Coordinator
(416) 224-1100, Ext. 402
1 (800) 339-3716, Ext. 402