Indexing and Retrieval of Devanagari Text in Printed Documents

Mudit Agrawal, M. N. S. S. K. Pavan Kumar, C. V. Jawahar
Centre for Visual Information Technology
International Institute of Information Technology,
Gachibowli, Hyderabad - 500 019, INDIA
{mudit,pavan}@students.iiit.net
jawahar@iiit.net

Abstract

This paper describes a document database system used for indexing and retrieval of Devanagari document images. These images are acquired by scanning the printed text documents from various sources. The document images are segmented into textual and image regions. The textual regions are annotated automatically using an OCR, and are stored in a database with relevant auxiliary information. The document is generated back according to the formulated queries. Results from various experiments are provided.

1 Introduction

Wide variety of information, which has been conventionally stored on paper is now being converted into electronic form for better storage and intelligent processing. The document database indexes and retrieves the images of documents containing text and graphics. The primary purpose of such systems is to facilitate retrieval of information which is relevant to a given query. Representation of documents as images is undesirable, because of impossibility of many common user-level operations like editing, searching and large storage requirements. These limitations can be overcome by representing the content as text, which takes very less space, and is also convenient for processing. As an illustration, a digitized document of A4 size may take 11 MB when represented as image in a database. A typical document may contain a lot of textual content and a few images. Representing the non-image portions as text format like ISCII brings down the storage requirement for the whole document, to a few KB and also makes them suitable for further processing. This kind of conversion can be achieved using an Optical Character Recognizer (OCR) for the corresponding language used in the documents. This approach aims at indexing the text in documents in a cost and labor effective manner. The document database system discussed here, uses an OCR for Devanagari [1] developed in-house for indexing. This OCR gives an accuracy of about 97% and hence was found extremely suitabe for indexing purposes.

This paper discusses the building of a Devanagari document database system, along with a few complexities involved in Devanagari OCRs. Section 2 describes the general issues involved in processing the document images for applying an OCR on these images. Section 3 deals with training and classification of the OCR. This is followed by Section 4, describing how this classifier is used in handling the splitting of samyuktaksharas. Issues regarding the indexing and retrieval of a large number of pages efficiently are discussed in section 5.

2 Document Image Processing

A document image is processed for extraction of robust and meaningful features for recognition. This involves noise reduction, binarization, skew correction and most importantly segmentation.

Preprocessing The scanned image usually contains noise. To reduce the effect of noise, a mean filter is applied to the image. In this process, the intensity of each pixel in the image is replaced by the average of the intensities of surrounding pixels. The resulting image is then subjected to binarization and skew correction. The binariztion method used in the system is a global thresholding algorithm

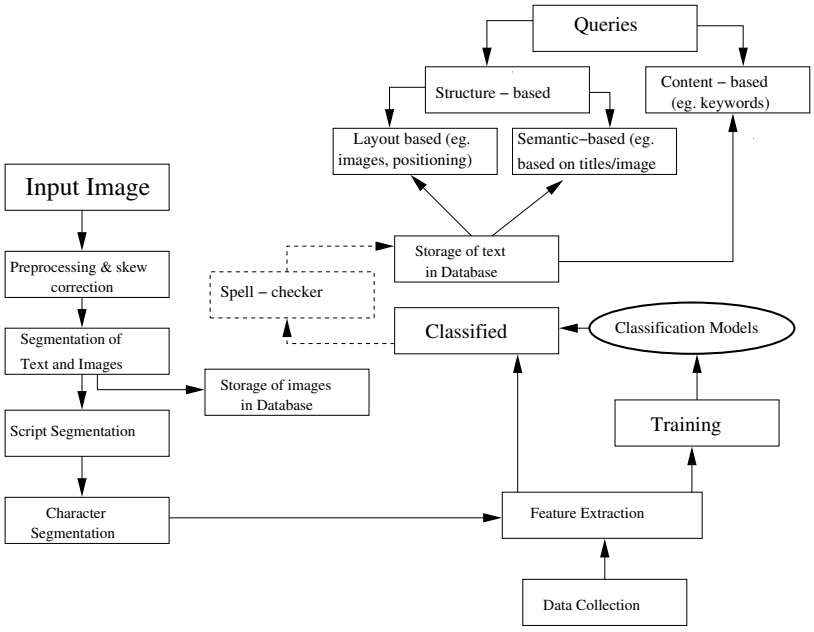


Figure 1: A block diagram of different modules in an Document Database system. The modules enclosed in dashed lines denote a future implementation.

based on the *K-Means* or *ISODATA* algorithm [2, 3]. Document images are often misaligned to the standard axes, due to improper placement of the paper on the scanner bed. Skew correction is performed to align the document axes properly to the coordinate axes. Since Devanagari text has a sirorekha, projection profile-based method is found to be appropriate [1, 3].

2.1 Page Segmentation

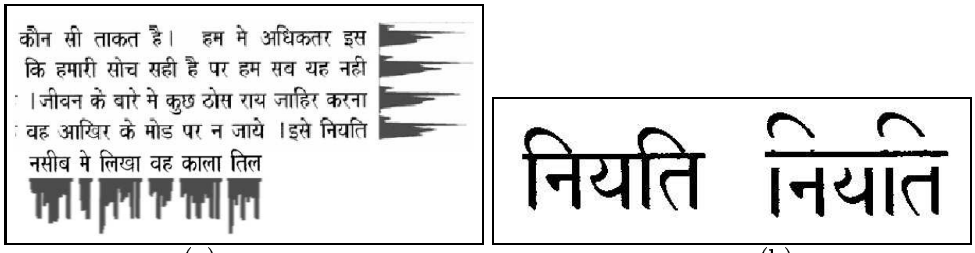
Segmentation is the process of subdividing the image into its constituent parts based on some predefined criteria [4, 5]. A logical subdivision of a document image would be initially into text and images. Since OCR has to operate on an image of a text only, it is necessary to give isolated text portions of the document image for OCR, and to store the image portions directly in the database with appropriate location-details. A method based on horizontal and vertical projection profiling is used for this.

The histogram of pixels along horizontal and vertical axes gives a clue of the boundaries of the images. The boundaries are generally darkened and their projections on the axes produce a high value. The images are separated from text by a margin generally greater than the line-spacing between the text. The line spacing is calculated for a given font by noting the difference between the first scan line which has horizontal projection greater than a threshold and the first scan line after the above one which has the horizontal projection less than the given threshold. If any spacing is found greater than this difference, it denotes the boundary of a graph or of an image. The pixel locations (x, y) of these bounding rectangles are stored along with the image so that the page can be regenerated exactly at the time of retrieval.

The portions containing text are given to the OCR. The processing of the OCR [1] is described below in brief.

Line Segmentation: Identification of the lines in a document image is called line segmentation. To perform this, the horizontal projection of the character pixels in the document image is computed. The valleys in the histogram correspond to the gaps between the lines in the image [1](see Figure 2(a)).

Word Segmentation: After the above step, the boundaries of each line - its top and bottom are known. Word segmentation is extracting the boundaries of words from these lines. Lines are processed in the similar way as in line segmentation to get the words but with vertical profiling [1]. Each word is width delimited by the starting and end points of such valleys (see Figure 2(a)).



(a) (b)

Figure 2: Horizontal and Vertical projection profiles of a section in a Devanagari document image. The valleys correspond to gaps between lines/words and are used to isolate them.

These words are further split into sub-zonal components using projections on the words after deleting the sirorekha, and dividing the line into 3 zones, the upper zone, middle zone and the lower zone. An example segmentation of a word is shown in Figure 2(b). Features are then extracted from these components. The current OCR uses images scaled to a standard size, with their rows concatenated to form a vector, as features. The classifier used in the OCR is a Multi class Support Vector Machine classifier called Directed Acyclic Graph SVM(DAGSVM). The theory of SVM and DAGSVM is briefly discussed in section 3.

Once a page is stored in the database, it is generated by rendering the contents matching a formulated query. The details stored along with the text and images in the database are used to generate the page in its original form, by pasting them as and where required.

3 Training and Classification

The OCR uses a SVM classifier. SVMs are a good example of classifiers which can perform generalization. Also, they are better suited to problems where the dimensionality of the input data and number of classes are very high as in this case. A brief discussion on SVMs is given below.

3.1 Support Vector Machines

Since their introduction by Vapnik and his co-workers, SVMs have been successfully applied to a number of problems like isolated hand-written character recognition (Cortes and Vapnik, 1995), object recognition [6], and face detection. A detailed description of SVMs is given in [7]. SVMs are defined for two class classification problems, which are later extended to multi-class classification problems. To start with, consider the following two-class classification problem. Given a training dataset of l independently and identically distributed samples. SVM constructs the decision function by finding the hyperplane that has the maximum margin of separation from the closest data points in each class. Such a hyperplane is called an optimal hyperplane. Training an SVM requires solving a quadratic optimization problem, and the classification is done just by a simple evaluation of a kernel function over a dot product of two feature vectors. This makes SVMs very fast when compared to other classifiers like KNN.

Although the theory for two-class SVMs is well developed, multi-class SVM is still an unsolved research problem. Multi-class SVMs are usually implemented as combinations of two-class SVMs. The one-Vs-one method, which constructs a classifier for each pair of classes is used here. A directed acyclic graph (DAG) is constructed, where each node corresponds to a two-class classifier for a pair of classes [8]. The multi-class classifier built by this algorithm is a Rooted Binary DAG. It can be observed that the number of binary classifiers to be built for a N class classification problem is $N(N-1)/2$, one for each pair of classes.

Based on our experience we have used a quadratic polynomial kernel SVM in the OCR.

A syllable in Indian languages, may be composed of a few consonant and vowel modifiers. If the syllables are not separated, a prohibitively large number of classes have to be handled in the system, but separation of them into base and modifier classes yields very less classes for the classification. Separation of consonant modifiers from the base modifier is a nontrivial task as they are attached to the base character.

Multiple Classifiers: A Devanagari word can be divided into 5 components: The upper matras, the lower matras, numerals plus special symbols which do not have *sirorekha*, full characters and

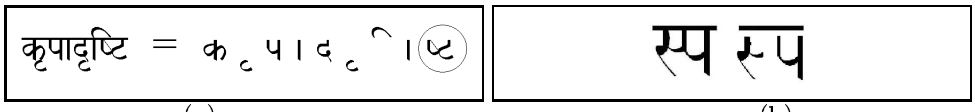
samyuktaksharas (refer Figure 4(a)). All these classes sum up to 168 in Devanagari. If just one classifier is used to classify all of them, misclassifications may occur. e.g. the numeral 2 and Hindi "r" or numeral 7 and "u matra" can be easily misclassified (see Figure 3), due to their resemblance.

Figure 3: Similarity of numeral r & 2

The difference in them lies on the mere fact that no numeral has a sirorekha. Once the is absence of sirorekha is observed, the numeral is given to the first classifier hence removing any ambiguity. Thus, numerals and other special symbols which do not have *sirorekha* are recognized by the first classifier. The numbers from 0 to 9 plus the special symbols like "- , ." makes up 16 classes. Word segmentation easily extracts upper and lower matras as explained in the previous section. They are recognized by one of the classifiers. Full characters with height to width ratio below a specific threshold are recognized by the third classifier. Number of classes are highest in this category and sum up to 115. And finally the wide characters, called samyuktaksharas (21 in Devanagari), are recognized by the fourth classifier. This classifier recognizes the samyuktaksharas as discussed in the next section.

4 Recognition of Samyuktaksharas

The Devanagari Optical Character Reader poses a lot of interesting problems due to a large set of Devanagari letters (called "Aksharas"). The akshara is the basic unit or quantum from a linguistic point of view and computer programs processing text in Indian languages should be able to efficiently deal with this quantum. The words in Devanagari are not only built from basic set of these akshara, but also from vowels, consonants and conjunct characters called "samyuktaksharas". Once we have broken a document page into a number of sentences, each sentence into words, then the removal of the sirorekha separates all the characters from a word. Similarly if a word has some matras, then the separation of matras above the sirorekha, the akshara and the matras below them separates the glyphs apart. In case of samyuktaksharas, the characters separated out by the removal of sirorekha will not be an akshara alone but a conjunct! i.e. a combination of a half character with a full character(see Figure 4(b)).



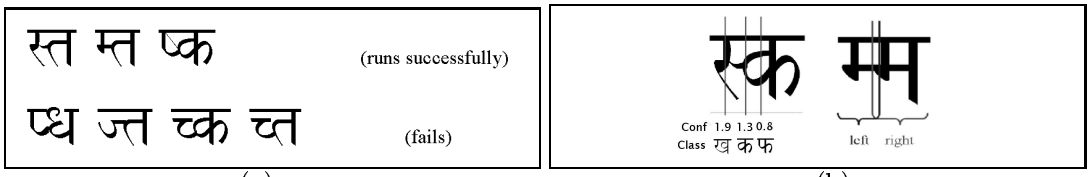
(a) (b)

Figure 4: (a)A Devanagari word (left) and the corresponding components of the word - the sirorekha, matras and the samyuktaksharas(right). (b)A samyuktakshara(left) and the corresponding components of the samyuktakshar(right).

The main task is to separate the half character and the full character in the samyuktakshara so that they can be analyzed (classified) separately. A few heuristic approaches, and a classification confidence based algorithm, which gave very good results on all samyuktaksharas, are discussed below along with their merits and demerits.

Iso-data Algorithm Iso-data algorithm was first applied to separate the two characters. The projection along each column was calculated and iso-data techniques were used to identify the 'valley' or depression in the projection histogram, which probably gives the point where the separation is to be done. The results were good for characters for which the projection-value was minimum at the point of conjunction of half character & full character. But the algorithm failed on cases where the projection-value reached its minima either before or after the 'actual' conjunct point (Figure 5(a)).

Single Classifier Approach The cut is performed at the point where the classification confidence of the right-side character (Full character) went to maximum. The cut-point is slided from left to



(a) (b)

Figure 5: (a)The first line shows words which were correctly segmented by isodata algorithm, whereas the second line shows the words which were not. (b) The confidences and classifications of a samyuktakshara by one-classifier approach (left). Segmentation of a samyuktakshara using 'two-cut technique' in two-classifier approach (right).

right and the classification confidence of the full character are noted.This is not a robust approach as many characters when cut partially, resemble another full/half character in Devanagari (Figure 5(a)). This emphasizes the importance of having a separate classifier for half characters.

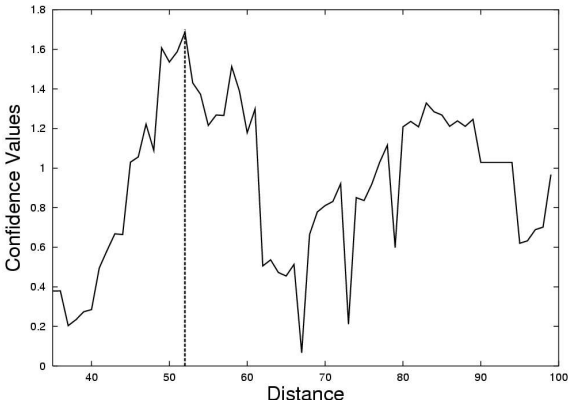


Figure 6: A plot showing confidences of cuts at various positions in a samyuktakshara.

Two Classifier Approach In this approach, left part (half character) is sent to its classifier and the maximum of sum of individual confidences is found. Moreover, the whole scan of the samyuktakshara was avoided as the actual cut occurs in the middle-portion only. Thus only the middle 40% of the samyuktakshar is scanned, confidence values are noted and max. is found among them. Due to the problem discussed above, the confidences were high in some cuts, so the "consecutive" occurrences of the classes was also taken into account. i.e. if the confidence of left & right part is high and also if the classes return have occurred consecutively above a threshold (the threshold depends only on the width to height ratio of samyuktakshara and hence is font independent to an extent).

Two-classifier - Two-cut technique: Analysis showed that a single cut is not enough for cutting the samyuktaksharas into their constituent characters.The cut where the left character is fully recognized (i.e. a high confidence is achieved), engulfs some part of right character too, and thus the identification of right character is poor. Hence, two cuts were moved from left to right of the samyuktakshara such that the separation between them is minimum and they give the appropriate portion of the characters to the classifier (see Figure 5(b)).

The isodata algorithm was not combined with it as it shifts the cut towards right or left thus giving wrong results. The height-width ratio of the full characters can not be used to aid in determination of cut as this makes it segmentation dependent.

5 Indexing and Retrieval

After segmenting the document image into text and images, applying OCR on text-blocks, followed by this text-storage, the major picture of Document Database comes into limelight. Once we have

a large collection of document pages in the database, intelligent retrieval is necessary in order to filter the required text (and images) from the rest.The indexing and retrieval of text and images and generating the same page structure depends upon the nature of the data to be accessed (fielded, full text or image) and hence different techniques must be used for creating indexes, formulating queries and retrieving records.

5.1 Typical Search Categories

Queries in a document database can be either structure based or content based for data retrieval. Having the ability to capture an accurate representation of both the content and structure is especially important considering the complex, multi-modal documents that are generally found in real-life situations.

Structure-based database There are two ways to index a document collection based on structure: using Physical(layout) structure or using Logical(semantic) structure. The physical structure of a document generally deals with the organization of text and images in it. Logical structure deals with meaning the layout conveys, and is a high level representation of the low level structure features. for example: Structure for news-items implies a "date" in the beginning of every news-item which marks the day and time of the event.

Content-based Document Database Content-based Document Database deals with the keyword searches and searches based on the storage-names of various documents. Since the recognized text need not be 100% accurate, some key-word searches may fail. This is prevented by adopting a wildcard searching. eg. if no results are found for a keyword "abc" then he should be able to replace any character by a ? and give the query again, viz. a?c might return what user was looking for. The block diagram of the Document Database is shown in Fig. 1.

5.2 Searches supported by the system

Page search The data in stored in the system as different pages. During the storage, when OCR is applied to the page, each page is stored with (i) a page number (ii) the magazine/book name (iii) other details like issue number, etc. Once the user types in page "x", the pages numbered "x" of all the magazines/books will be displayed. Searches on specific magazine name, or issue number are also supported. The pages are rendered from the content and location information and are shown.

Line search If the user's query word matches with a word in any line of any page in whole of the document database, the line containing it is generated back.The keyword entered by the user is matched against all the words in the database. The sentences with this word (i.e. the set of words preceded and followed by a "viraam") are displayed.

Layout search Layout based search involves retrieval based on the information related to the arrangement of text, images, graphs, subsections and headings/titles. Approximate locations of the components can be specified by their location, or spatial arrangement etc. For example, user can retrieve those pages, in which images are on the left/right/top/bottom part of the page. The location variables (x,y) stored with every image are used to aid in the search.

Word-based search Keyword based search is one of the typically encountered facility in a document database. Whenever a word is given, it is searched for in the database and the pages containing that are generated. If the search contains more than one word, occurrence of all those words is considered. Unless it is specifically mentioned, the order of the words to be searched is not maintained. Examples

As the first example to demonstrate some of the capabilities of the system, we scanned document pages which contain the following text in Devanagari script. We applied our OCR and recognized text were stored in the database.

1. kal pradhanmantri rashtrapati se holi milne rashtrapati bhavan padhare.
2. Rashtrapati ne pradhanmantri ko hardik badhai di
3. Bharat ke naye Rashtrapati mananiye dr. abdul kalaam hoyenge.

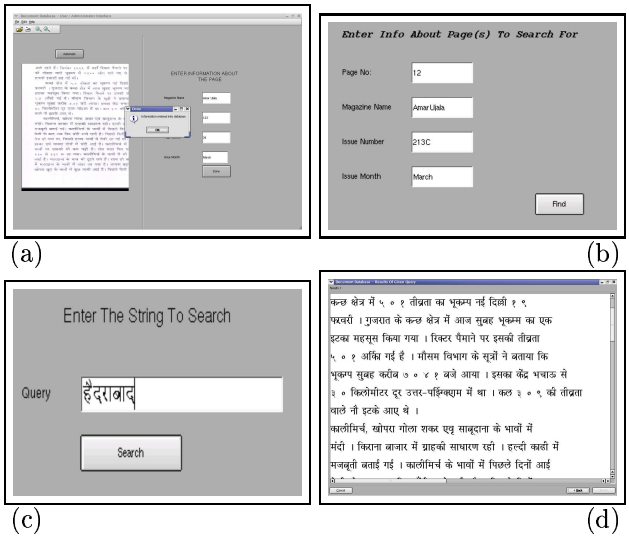


Figure 7: Various steps involved in a typical Document Database system.(a) Shows the storage of OCR'd text in the database. (b) Shows the page-wise query-processing. (c) Shows the word-based query-processing. (d) Shows the Document page generated from the database on the fly.

The outputs of various search-strings are shown below.

- Search string: pradhanmantri rashtrapati

  (a) When a search was made for "all the words", 1 and 2 sentences were matched. (b) When a search was made for "any one word", all the sentences were matched.

- Search string: "pradhanmantri rashtrapati" Output: sentence 1 only. This is because, enclosing a string within quotes forces it to be matched with sentences which have its occurrence exactly in the same manner.

- Search string: pradhanm* Output: 1,2 and 3. The use of wildcard '*' implies zero or more characters following the word 'pradhanm'.

Enclosing a string in single-quotes gives way to partial string matching. This avoids spelling mistakes by the OCR or the user.

5.3 Image-based search

Images have been a centre of focus in many areas, both in searches and vision combined. Image searches enable us to locate the relevant matter by just using the caption of the image. Whenever the image is segmented out of the text its caption is given to the OCR, the results are stored in the special "caption" entity of Image. During the image-search, a search is made only on these captions and a match results on the generation of the page containing the image. Its not always true that an image has a caption, and even when it has, it is not necessary that it is what a user may desire. In such cases, user is presented by two options of image-searches. One by searches on captions and other when the keyword occurs in a page containing an image. For example, we have a photograph of bomb-shelling in Afganistan by US. The document page is likely to contain words like "laden","bomb", "afghanistan". Using these keywords, the page containing such an image can be retrieved.

5.4 System Description

A brief presentation on the performance of the system is given below. The documents used for testing were scanned on a Hewlett-Packard HP7670 scanner. The system has been developed completely in C++. The interface of the system was designed using Qt - a C++ based GUI library. The database is maintained in MYSQL. Hardware interfacing with scanner has been done using the SANE API. The machine used was a Hewlett-Packard running on 128 MB RAM.

The system consists of four main components – an image editor, a text editor, and the core OCR engine and the back-end mysql support. Scanned images are displayed in the image editor along with the segmented parts of the image into different text blocks and images. User selected portions (Fig7(a)) of the noise free images of the text blocks are sent to the OCR engine. The OCR engine then performs line segmentation, word segmentation, and character segmentation. The OCR converted text is shown in the text editor allowing the user to perform any necessary editing. The "automate" option does this automatically for the user. All the text-blocks are given to the OCR. The search options allow the user to go for Structure-based as well as Content-based searches(Fig7(b) & (c)). The resulting page is created on the fly from the database and is shown to the user(Fig7(d)). The Document Database system was applied to many datasets, say, one of them has been described as under:

Cricket pages: Nearly 40 to 50 pages on printed text from cricket news (source: Hindi newspapers Dainik Jagran and Amar Ujala) were scanned and stored. The common queries like the name of the cricketers e.g. (1) sachin, generated all the pages (refer previous section) containing the word "sachin". (2) "World Cup" generated all the news ranging from present world cup to the old world cups. (3). "Hyderabad cricket" produced the pages containing the name of Mohd. Azzaruddin, and various matches played there.

Other queries included: bharat pakistan, scorecard (searched the images of scorecards by using the caption "scorecard" under them and displayed them), "jayasurya sachin", "ganguly apne form ko sudharne ki koshish", "dravid bharat ke reed ki haddi" etc.

6 Conclusions

In this paper we have presented an indexing and retrieval system for Devanagari printed documents. The system provides excellent results for structure and content based queries. We are yet to formally evaluate the performance (precision, recall etc.) on a large database. Presently we are working on the indexing of Hindi books, adding a new dimension to the digital archival process.

Acknowledgements

The authors would like to thank Uday Kumar Visesh, Arvind Upadhyay, and Puspendra for their help in implementation and testing of some parts of the system.

References

[1] C. V. Jawahar, M. N. S. S. K. Pavan Kumar, and S. S. Ravi Kiran, "A Bilingual OCR for Hindi-Telugu Documents and its Applications," in *International Conference on Document Analysis and Recognition (ICDAR)*, 2003.

[2] T.W.Ridler and S.Calvard, "Picture thresholding using an iterative selection method," in *IEEE Trans. System, Man and Cybernetics*, vol. SMC-8, pp. 630–632, 1978.

[3] C. V. Jawahar, M. N. S. S. K. Pavan Kumar, and S. S. Ravi Kiran, "Recognition of Indian Language Characters using Support Vectors Machines," *Technical Report TR-CVIT-22, International Institute of Information Technology,Hyderabad*, 2002.

[4] R.C.Gonzalez and R.E.Woods, *Digital Image Processing.* Prentice-Hall, 1994.

[5] R. G.Casey and E. Lecolinet, "A survey of methods and strategies in character segmentation," *Pattern Analysis and Machine Intelligence*, pp. 690–706, 1996.

[6] E. Osuna, R. Freund, and F. Girosi, "Training support vector machines: Application to face detection," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 130–136, 1997.

[7] C. J.C.Burges, "A tutorial on support vector machines for pattern recognition," in *Data Mining and Knowledge Discovery*, 1998.

[8] John.C.Platt, N. Cristianini, and J. Shawe-Taylor, "Large margin dags for multi-class classification," in *Advances in Neural Information Processing Systems 12*, pp. 547–553, 2000.