A Block DCT based Printed Character Recognition System

Dissertation submitted in partial fulfillment for the award of the degree of

Master of Science (Mathematics)
with Specialization in Computer Science.

by

Sai Charan K.

(Regd.No.: 04006)

DEPARTMENT OF MATHEMATICS & COMPUTER SCIENCE

Sri Sathya Sai Institute of Higher Learning

(Deemed University)

Prashanthi Nilayam, March 2006

Dedicated to *Mother Sai* who chose to

spread Her message through *Telugu* natively...

Sri Sathya Sai Institute of Higher Learning

(Deemed University)

DEPARTMENT OF MATHEMATICS & COMPUTER SCIENCE

Certificate

This is to certify that this thesis entitled **"A Block DCT based Printed Character Recognition System"** being submitted by **Sri. Koduru Sai Charan** in partial fulfillment of the requirements for the award of the degree **Master of Science (Mathematics) with Specialization in Computer Science** to Sri Sathya Sai Institute of Higher Learning, Prashanthi Nilayam (Deemed University), is a record of bonafide research work carried out by him under my supervision and guidance. It is a record of the work done by him in the four-month winter semester of the academic year 2005-06 in the **Department of Mathematics and Computer Science** of the Institute. The results embodied in this dissertation have not been submitted to any other University or Institute for the award of any Diploma or Degree.

Prof. G. V. Prabhakar Rao
Department of Mathematics and Computer Science,
Sri Sathya Sai Institute of Higher Learning,
Prashanthi Nilayam.

Place: Prashanthi Nilayam
Date: 19^{th} March 2006

ACKNOWLEDGEMENTS

At the outset, I express my deepest gratitude to **Beloved Bhagawan Baba** for His constant presence through this work. Also, I lovingly remember my parents for their support, as always.

I am grateful to my supervisor **Prof. G. V. Prabhakar Rao**, Department of Mathematics & Computer Science, Sri Sathya Sai Institute of Higher Learning, for his constant guidance through this work and the direction that he has provided me all through.

I would like to thank the university administration, in particular, the Vice Chancellor Sri A.V.Gokak, the Registrar, Dr. Lakshmi Narasimham, the Controller of Examinations, Prof. M. Nanjundaiah & the Pricipal of the campus, Prof. U.S. Rao, for kindly permitting me to pursue this work.

I would like to thank Prof. V.Chandrasekaran and Prof. C.J.M. Rao, Department of Mathematics & Computer Science, Sri Sathya Sai Institute of Higher Learning, for the excellent introductory course on Pattern Recognition Systems and the brief discussions through the work; they were of great help in making this project a success.

I express my thankfulness to the course coordinator, Dr. Pallav Baruah, for his timely reviews and guidelines which have gone a long way in completing the work in time, to Sri. Raghunath Sarma, coordinator, AI Center, SSSIHL to Sri. Ravi Iyer, who has been an inspiration in perfection and also to Prof. Panchanathan, Arizona State University, for his enlightening Power Points.

I am extremely grateful to all my classmates, seniors and juniors for all the help they have rendered during the course of this work. However, the acknowledgement would be incomplete without mentioning some people who have rendered the greatest help I could hope for; that of lending their precious time to hear and advise me on many aspects of this work. They are Sri. Sai Suhas, Sri. Hanumanth Rao Naidu, Sri. Srikanth Khanna, Sri. Shiva Kumar K., Sri. Prashanth Sai G. & Sri. Ramprasad G..

Contents

1 Introduction **1**

1.1 The Context of this work 1

1.2 Brief History of OCR 3

1.3 Design of the Pattern Recognition System 3

1.3.1 Sensing 4

1.3.2 Segmentation & Grouping 4

1.3.3 Feature Extraction 4

1.3.4 Classification 5

1.3.5 Post-Processing 5

1.4 OCR: The Problem, Methods, & Techniques 6

1.4.1 Scanning & Digitization 6

1.4.2 Preprocessing 6

1.4.3 Segmentation 7

1.4.4 Feature Extraction 7

1.4.5 Classification 8

1.4.6 Post-processing 8

1.5 The present work 8

1.6 Applications 9

1.7 Chapterization 9

2 Previous Work on OCR for Indian Languages **11**

2.1 Introduction 11
2.2 Preprocessing − A Brief survey 12
2.2.1 Noise Filtering 12
2.2.2 Deskewing 13
2.2.3 Page Segmentation 14
2.2.4 Font Recognition 14
2.3 Feature Extraction & Classification methods 15
2.3.1 Template Matching 16
2.3.2 Deformable Templates 16
2.3.3 Unitary Transforms 17
2.3.4 Zoning 17
2.3.5 Geometric Moments 17
2.3.6 Zernike Moments 17
2.3.7 Projection Histograms 18
2.4 Existing OCRs for Indian Languages 19
2.4.1 The DMACS OCR 19

3 The DCT Based Feature Extractor 22
3.1 Introduction 22
3.1.1 Motivation 22
3.1.2 The Discrete Cosine Transform: The Mathematics 24
3.1.3 The Discrete Cosine Transform: The Pictorial Description 25
3.2 The Feature Extractor(s) 28
3.2.1 The Design Cycle 28

4 Results, Implementation & Discussion 39
4.1 The Tests and Results 39
4.2 Discussion 41

4.2.1 Implementation 41
4.2.2 Other Considerations 43
4.3 Future Developments 43
4.4 An analysis of the Dictionary 45
4.5 Conclusions 47

A Sample Code **48**
A.1 Binarization 48
A.2 Zig-Zag Feature Vector 48
A.3 Feature Extraction for Blockwise DC coefficients 51
A.4 Metrics 52
A.5 Run the Tests 52
A.6 Append the dictionary 57

ABSTRACT

Optical Character Recognition of Indian languages is a a research field that is socially very relevant and challenging. The social relevance lies in fact that the OCR can help preserve documents of the past for posterity. Many ancient manuscripts can be digitized and stored away for future editing and utilization using OCR.

One important motivation for this work has been the digitization of the Telugu spiritual monthly magazine, "*Sanaathana Sarathi*", published from Prashanthi Nilayam.

The objective of the present work is to propose a prototype for an efficient feature extractor for recognizing printed Telugu characters using the **Blocked Discrete Cosine Transform**. The motivation to use this comes from the fact that the **Blocked Discrete Cosine Transform** efficiently encodes the energy / the significant details of the image in a few coefficients. This has also been one of the main reasons for its use in the JPEG compression algorithms. A further motivation for using the DCT is that it is a faster algorithm. Its computational complexity is of $O(Nlog_2N)$.

In this work we propose a prototype feature extractor for *Printed Telugu Characters* using the DC and the first AC coefficients of the **Block Discrete Cosine Transform** applied on the micro-blocks of the character image. The prototype is coded and tested on some sample text using Matlab and the classification was accurate upto 95.6%. It was observed that for fixed font printed character recognition, the DC coefficients of the micro-blocks were reasonbly adequate. An interesting feature of the model developed is that it can be easily extended to any other script in a very short time.

Chapter 1

Introduction

What is it?

-Bodhidharma

1.1 The Context of this work

Optical Character Recognition, popularly referred to as the OCR, is an active area of research in this information age. It is a special case of the Pattern Recognition Systems and is one of the key issues regarding the use of information systems in the acquisition of new information, which often resides in paper documents. In order to provide a suitable solution to this problem, information systems will have to be integrated with paper document processing systems, which are devised to transform printed or handwritten documents into a computer-revisable form [MAC].

Optical Character Recognition (OCR) is perhaps the most successful application of Pattern Recognition, an allied problem of information systems. It is one of the oldest tasks that has engaged computer scientists. Initially the goal was to recognize machine printed text formed from characters drawn from a single font

and of a single size. With success in recognizing single-font, single-size text, the goal was expanded to reading multi-font text in which a range of character sizes may be present. Success bred more challenging problems such as hand-printed character recognition, and, of late, the recognition of cursive scripts both printed and handwritten [CSR].

Optical Character Recognition is regarded as one of the most challenging steps in the process of digitization of literature. It deals with machine recognition of characters present in an input image obtained using scanning operation. It refers to the process by which scanned images are electronically processed and converted to an editable text. The need for OCR arises in the context of digitizing documents from the ancient and old era to the latest, which helps in sharing the data through the Internet. OCRs are now used to generate machine-readable text from printed documents. These are generally legacy documents from the pre-electronic publishing era, but may also be printed documents for which the original machine-readable text was discarded or lost.

One of the aims of the Universal Digital Library project is to digitize the vast corpus of Indian literature available in the regional scripts. Due to climatic effects and poor storage conditions, even the which are books barely 50 to 100 years old are significantly damaged to the extent that conventional OCRs are not very successful [IYER]. According to Wikipedia [WIK], Optical Character Recognition, "involves computer software designed to translate images of typewritten text (usually captured by a scanner) into machine-editable text, or to translate pictures of characters into a standard encoding scheme representing them in ASCII or Unicode."

According to Sachwani [SAC] and Raghavan [RAG], "OCR is the process of converting scanned images of machine printed or handwritten text into a computer processable format."

1.2 Brief History of OCR

Optical Character Recognition is a problem recognized as being as old as the computer itself. There have been many papers and technical reports published reviewing the history of OCR technologies [TRI]. Modern OCR was said to have begun in 1951 due to an invention by M. Sheppard called GISMO, a robot reader-writer. In 1954, a prototype machine developed by J. Rainbow was used to read uppercase typewritten letters at very slow speeds. By 1967, companies such as IBM finally marketed OCR systems [VER].

1.3 Design of the Pattern Recognition System

According to Duda, Hart and Stork [DUD], Pattern Recognition Systems (PRS) can be thought of consisting of the following problems: Sensing, Segmentation & Grouping, Feature Extraction, Classification & Post-Processing. The diagrammatic representation is given below :

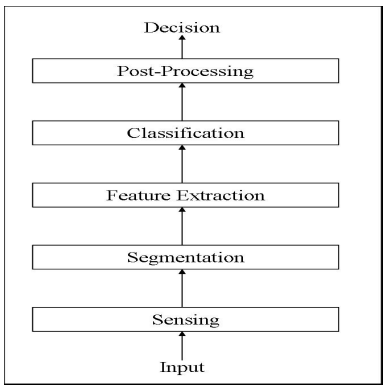


Figure 1.1: Conceptual view of the Pattern Recognition System

1.3.1 Sensing

Sensing is the perceiving of the input by the PR system. It could be by way of a camera or some transducer or some other signal sensing device. The design of the *Sensor* is by itself a field of study. There are many factors about the Sensor that can influence the efficiency of the PRS. For eg., the resolution of the camera plays an important role in the quality of the image that may be given as input to the PRS.

1.3.2 Segmentation & Grouping

Consider a PRS for the automated segregation of bottles. The input may be through a video camera that is focused on a conveyor belt that is carrying the bottles. The bottles may have different colors, shapes, sizes, texture etc. The bottles may also be overlapping as they come across the conveyor belt. The first task in recognition is to *identify* the bottle, the 'boundary' that defines the bottle. This task is called segmentation. According to Duda & Hart [DUD], "Segmentation is one of the deepest problems of pattern recognition." The problems of segmentation in the field of Optical Character Recognition systems will be very different from those of the Text-to-Speech PR systems. Lot of literature is available on this subject for specialized domain, and no single technique can be guaranteed to work on all domains.

1.3.3 Feature Extraction

For reasons more practical than theoretical, we distinguish between Feature Extraction and Feature Classification. An "omnipotent classifier"[1] should not need the help of a sophisticated feature extractor and conversely, an ideal feature extractor would should make the job of a classifier trivial. The choice of the feature will very much affect the time and space complexity of the PR System. But, the

[1] see [DUD], p.11

quality of the PR System should not be compromised, and hence, there has to be a trade off between the two. Coming to our example of the PR system for segregation of bottles, an intuitive feature, as was mentioned earlier, could be the color or size or shape or texture of the bottle. A study has to be made as to which of these features are "*distinguishing*", ie., which feature(s) differentiate one *type* of bottle from the other *types*. For example, if the blue and green bottles have the same size, the size cannot be a reliable feature! One another aspect remains to mentioned here: ***that of invariance or tolerance***. The feature should be tolerant to some distortions in the input. For example, the distance of the camera from the bottle (which could lead to projective distortion) should not affect the classification process.

1.3.4 Classification

The classification uses the features extracted in the previous step to decide the group to which the feature belongs to. This is usually in the form of some function of the feature(s). For example, some kind of *metric* can be used to measure the *distance* between various features. The distance (or a function of it), can be treated as the *classifier code*. This code is the output of the classification phase. Some times, many classifiers could be used and the democratic policy can be used to make the final decision. Some popular classifiers are the 1-Nearest Neighbour (or the 1-NN) classifier, the n-Nearest Neighbour (or the n-NN) classifier, the Neural Net Classifier. This phase also may use the Artificial Intelligence methods of ***learning***. More on this in later chapters. Note: The neural net classifier has "learning" incorporated into itself!

1.3.5 Post-Processing

The input to this phase is the classifier code that was introduced in the previous section. The role of this phase is to make decisions. And, once there is decision

making, there is bound to be error. So, the aim of this phase can now be defined to be the process of making decisions with minimum average error.

1.4 OCR: The Problem, Methods, & Techniques

As a special case of the PR systems, we will take a look at the OCR, which is one of the many sub problems of PRS, and is of special interest in this work. The OCR problem, consists of: Image Measurement, Feature Extraction & Classification [TOU, PAT]. This scheme is much the same as the one proposed by [DUD] with the Image Measurement referring to Sensing, Segmentation & Grouping and the others being the same.

1.4.1 Scanning & Digitization

The Document should first be scanned & saved in a standard format so as to be processable by any standard image processing tool.

1.4.2 Preprocessing

The scanned image is subjected to certain image processing algorithms such as cropping and alignment. The inherent imperfections of the image are to be removed before the next phase so that the processing is simple later. The various processes involved here are : noise filtering, deskewing, page segmentation & font recognition [RAG]. In OCR parlance, these steps form the Pre-Processing section. For example, if the font is recognized, we can map from the document font to some standard font for post processing purposes.

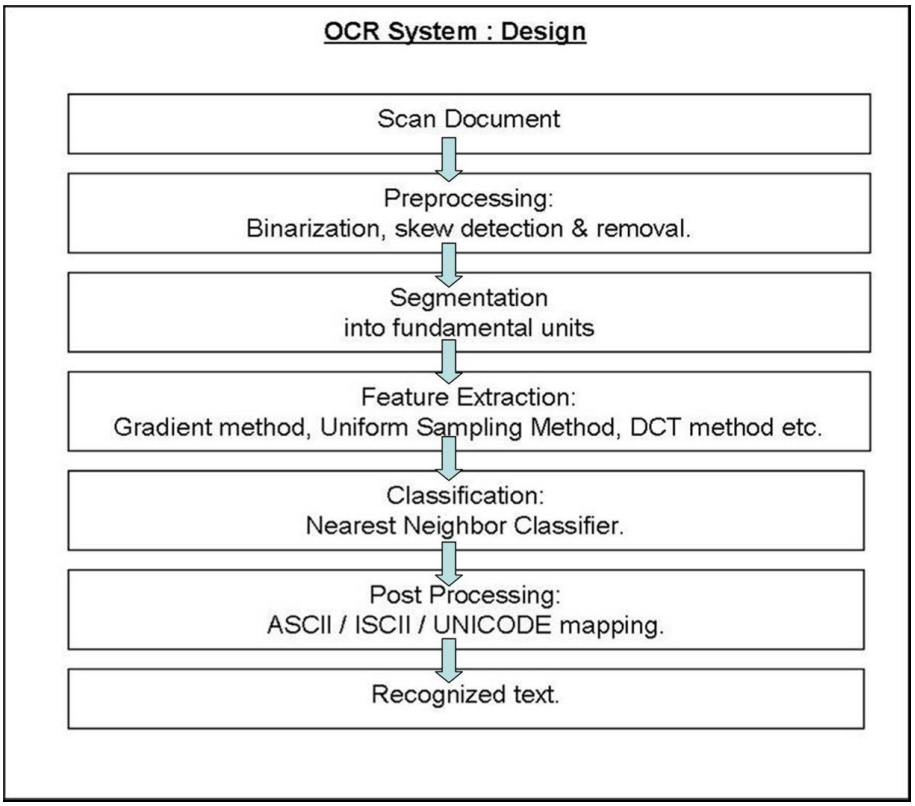


Figure 1.2: The OCR System Design

1.4.3 Segmentation

Segmentation in the context of character recognition can be defined as the process of extracting from the preprocessed image the smallest possible character units suitable for recognition; i.e. to segment the fundamental units of the document.

1.4.4 Feature Extraction

Feature extraction refers to the process of characterizing the sub images generated as the output of the segmentation procedure based on certain specific parameters. There are many previous approaches like: template matching, zoning

etc. For more information, please refer to [TRI].

1.4.5 Classification

Classification refers to placing each of the recognition units into one of a predetermined set of classes, based on the characteristics of that unit as evaluated by the feature extraction procedure. Thus, each character image is mapped to a textual representation.

1.4.6 Post-processing

Here, the output of the classification stage is converted into ASCII or ISCII or other standard coding schemes so that words, sentences and paragraphs are reconstructed from the outputs of the classification stage. A well-structured dictionary may also be used to resolve ambiguities in recognition.

1.5 The present work

This project seeks to propose an OCR feature extraction system using Fourier methods, the DCT in particular.

As in the OCR implemented by Sachwani[SAC] and Raghavan[RAG], in this work also, the scanned image is saved as a 256-color bitmap image. This image is then segmented into characters. These segments are then resized to a 32-by-32 matrix. This matrix is to be further divided into 16, 8-by-8 matrices. The DCT is applied to these 16 matrices to get the DCT coefficients. Of these coefficients, the first coefficient, being the DC component, and the next two to three AC coefficients are considered as the significant features for each of the 16 sub matrices. The feature vector for the entire image is then to be constructed from these chosen coefficients. The problem now is to construct a feature vector and check the 'discriminating' power of this feature vector for character recognition. This gives

the 'Average Pixel Intensity' feature vector.

An enhancement of the above procedure was also tested. The above procedure was carried out on a block size of 4-by-4. This was with the hope that it would yield a more localized and discriminating feature vector, as we are concentrating at a smaller area. This would definitely increase the feature vector size and hence the time and space complexity of the recognition process, but it may be worth the effort.

1.6 Applications

1. Automated fax reader for the blind [ALN].
2. Text to speech recognition.
3. Document reader for offline scanned text images or Document Image Decoding (DID) [BRU].
4. Assigning zip codes to letter mail.
5. Reading data entered in forms (tax applications etc).
6. Account number verification on bank drafts & checks.
7. Automatic validation of passports.
8. Valuation of exams [SAC, RAG].
9. Digitization of Spiritual literature for ease of access over the internet or other media.

1.7 Chapterization

The report is organized as follows:

The first chapter introduces Pattern Recognition Systems and their special case, the Optical Character Recognition Systems from a utilitarian and historical perspectives. It also provides some applications.

The second chapter is a survey of the existing methods and techniques in use. It then mentions some of the existing OCRs for the Indian languages and gives some implementation details of the OCR developed at DMACS, SSSIHL.

The third chapter presents the details of the methodology adopted in this work, the mathematics and the implementation of the algorithms used.

The last chapter presents the results and analyses the results obtained. It also discusses the results and proposes some future enhancements.

———

Chapter 2

Previous Work on OCR for Indian Languages

...echoes from eons...

This chapter presents an overview of the existing important feature extraction techniques and classification methods. Along with a summary of the existing OCRs for the Indian Languages (in particular the Telugu and the Tamil scripts), it also discusses in some detail, the techniques used by the OCR implemented at the *Department of Mathematics & Computer Science, DMACS, SSSIHL.*

2.1 Introduction

In the following sections, we will describe some of the existing preprocessing, feature extraction and classification techniques. But, considerable attention is given to the techniques used by the OCR developed at the Department of Mathematics & Computer Science, SSSIHL, by Sachwani [SAC], Easwar [EAS], Raghavan [RAG] & Sudheer. The present is an attempt to improve the performance of the

feature extraction component of the above OCR by adding a new prototype based on an approach using the Block DCT, implemented in Matlab.

2.2 Preprocessing – A Brief survey

The following methods pertaining to the preprocessing techniques were implemented by Raghavan & Sudheer [RAG]. The following subtasks form the core gamut of the preprocessing phase:

1. Noise Filtering
2. Deskewing
3. Page Segmentation
4. Font Recognition

2.2.1 Noise Filtering

Noise is the unnecessary 'information' that is present in the image, which may have been inadvertently introduced. This may be because of inefficient input devices used etc. For example, when an image is scanned, a black color border may be introduced. For removing the noise, which may interfere with the working of the PR system, *filters* are used. Filters are essentially mathematical functions which help to 'differentiate' the image from the noise. There are two types of filter widely used:

1. Spatial filters, which work in the spatial domain. Examples are the Order, Mean & Median filters.
2. Frequency filters which work in the frequency domain. Examples include Inverse, Weiner, Constrained, Geometric filters among the others.

Further, this step also includes:

1. Thresholding and Binarization.

2. Black border removal.

Thresholding and Binarization

In this step, the gray image is converted to a *'binary'* image. By binary, we mean that only black or white pixels are present in the image. This helps us to work more efficiently with the image than with the gray or color image. Binarization is achieved through the process of thresholding. In thresholding, a *'threshold'* value is chosen. Any pixel with a value greater (or less) than the threshold, is converted to a text (or background) pixel. That is, its value is made either 0 or 255.

Black border removal

In this process, the black borders are identified and clipped. For the identification process, an adaptive threshold value is chosen and neighborhoods of continuous black pixels are chosen and clipped. A ratio of black pixels to the number of pixels in each row is compared against an adaptively determined threshold value. Those rows with the greater than the threshold are marked as black border rows. This process is continued till a non-black border row is encountered. Then, all that needs to be done is to remove those rows.

2.2.2 Deskewing

While scanning the image, if the paper/source document is not aligned properly, it may cause the components to be tilted. This could lead to erroneous behaviour of the PR system. To prevent this, deskewing methods have been devised, which *detect & remove* the skew from the image. As defined by Raghavan [RAG][1], "Skew is the counter-clockwise orientation of the dominant text base-line with respect to the x-axis". Some methods for deskewing are:

[1] see page 31

1. Based on Projection Profiles (both horizontal and vertical).

2. Based on Hough transforms.

3. Fourier based methods.

The DMACS OCR has two methods of skew detection. One is a variant of the Hough transform and the other is based on the observation that the gradient of the text base-line should be perpendicular to the text line. For details, please see [RAG][2].

2.2.3 Page Segmentation

Most of the time, the documents scanned have multiple *regions*. A region is an area of continuous homogeneity. By this we could refer to a section/block of text or images or a paragraph, the start and end of the page etc. So essentially, we are *partitioning* the document into its components.

2.2.4 Font Recognition

This is a very important phase of the OCR system. It affects the Automatic Document Processing (ADP), which is what the OCR aims to attain, in two important ways [RAG][3]:

1. It reduces the number of alternate shapes for each class, leading to single font recognition.

2. The font information along with the OCR output can be used for typesetting the document by the native text processor.

The DMACS OCR uses three methods of font detection. The first one is based on the *'most frequently occurring character'*. This character in different fonts is

[2]see pages 45-54

[3]see page 7

used as a feature and is compared against a dictionary which contains features for the most common fonts. The second approach is based on *'typographical structure of text lines'*. In this method, the characters are divided into three zones: upper, central and lower. This method takes into consideration the space between words, the proportion of the text in the upper and lower zones. The third method is based on *'global texture features'*.

2.3 Feature Extraction & Classification methods

Feature Extraction has been defined as, the process of extracting from the raw data that information which is most useful for the purposes of classification. That is to reduce the feature variation between members of the same class while maximizing the feature variation between members of different classes.

Trier et.al.,[TRI] refer to the problem of dimensionality that must be considered when we propose to use a statistical classifier. In essence, this principle warns us that the number of features must be kept reasonably small when the training set is small. As a rule of the thumb, they propose that we use the number of training samples around 5-10 times the dimensionality of the feature vector for any given class.

The following are some common techniques of feature extraction:

1. Template Matching
2. Deformable Templates
3. Unitary Transforms
4. Zoning
5. Geometric Moments

6. Zernike Moments

A brief description of these methods is provided in the next few pages. Before that, a word on **invariance** is in order. Invariant features are the features that do not significantly change with variations in the characters. The variations in characters could be translation, rotation, skew, scale, mirroring etc. The feature should be more or less the same inspite of the above occurrences.

Also, sometimes, the characters can be reconstructed from the feature itself. A good example is that of the DCT. *If* we choose the DCT of a character or of some sub-image of it as the feature, we can recover the entire character by taking the IDCT of the corresponding features! But, the sheer size of the feature in this case can be prohibitive.

2.3.1 Template Matching

This is perhaps the most basic form of feature extraction [TOU, TRI]. Actually, in this case, there is *no extraction.* Instead, template characters are stored in the database and the entire input character is compared to every template in the database. The closest one is chosen based on some similarity measure like the mean squared distance:

$$D = \sum_{i=1}^{M} (Z(x_i, y_i) - T_j(x_i, y_i))^2 \tag{2.1}$$

where, Z is the input character and T_j is the j^{th} template in the database. The limitations of this method are apparent.

2.3.2 Deformable Templates

Although this technique has been used in Object Recognition, it is not clear how the templates are chosen for the characters[4].

[4]please see page 5 of [TRI]

2.3.3 Unitary Transforms

The transforms used in image compression are generally Unitary. By this we mean that if A is the matrix form of the image transform, the $A^{-1} = A^{*T}$. Where A^* is the adjoint matrix to A and the superscripted T stands for the transpose. In most cases, these transforms can be represented by a (finite or truncated) series of orthogonal *basis* functions. The classic example is that of the *Fourier Transform*. Other transforms include the KL and Hadamard transforms. It has been shown that the KL transform is the statistically optimal (in the sense of minimum mean-square error) transform [TOU], but it is computationally expensive.

2.3.4 Zoning

In this method, the image is divided into $m \times n$ blocks and the average gray level is computed for each block. This is taken as the feature.

2.3.5 Geometric Moments

The word 'moment' here refers to the some of the characteristics that can be calculated from the images. There are moments of different orders that are used in pattern recognition as they are in statistics and elsewhere. The regular moments of order $(p + q)$ for a given image of M pixels Z are given by:

$$m_{pq} = \sum_{i=1} MZ(x_i, y_i) \, (x_i)^p \, (y_i)^q \tag{2.2}$$

Similar definition are also provided for the *central, relative* moments.

2.3.6 Zernike Moments

The Zernike moments are projections of the input image on the space spanned by the orthogonal V-functions defined by:

$$V_{nm}(x, y) = R_{nm}(x, y) \exp\left(j \; m \; tan^{-1}\left(\frac{y}{x}\right)\right) \tag{2.3}$$

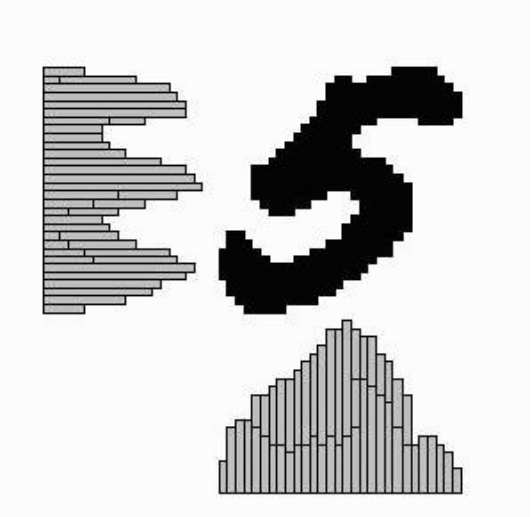


Figure 2.1: Example of a projection.

where $j = \sqrt{-1}$, $n \geq 0$, $|m| \leq n$, $n - |m|$ even and

$$R_{nm}(x,y) = \sum_{s=0}^{\frac{n-|m|}{2}} \frac{(-1)^s (x^2+y^2)^{\frac{n}{2}-s} (n-s)!}{s!\left(\frac{n+|m|}{2}-s\right)!\left(\frac{n-|m|}{2}-s\right)!} \tag{2.4}$$

2.3.7 Projection Histograms

For binary images, this is another technique used for feature extraction. For a horizontal projection of the image, $y(x_i)$ is the number of pixels with $x = x_i$. This along with vertical projections can make useful feature vectors. But note that the horizontal projection is **not** slant invariant, while the vertical projection is. Recently, to compare histograms, [DHN] has successfully used the Battacharya measure. It is defined as:

$$\sum_i \sqrt{X_i}\sqrt{Y_i} \tag{2.5}$$

where,

X_i and Y_i are the i^{th} frequency values of the two histograms under consideration.

Some common classification techniques are Nearest Neighbour and its variants [TOU, DUD][5], Neural Net Classifiers [DUD][6].

2.4 Existing OCRs for Indian Languages

Perhaps the most active agency in India for OCR development is the Ministry of Information Technology's Technology Development for Indian Languages or the TDIL [TDL]. The Vishwabharathi team periodically announces the advances made in this regard. The report says, "OCR with more than 97% accuracy has been developed for seven Indian Languages viz Hindi, Marathi, Bangla, Tamil, Telugu, Punjabi, Malayalam. The OCR technologies for Assamese, Oriya, Malayalam and Gujarati scripts are in the advanced stages of development". Examples of these are DEVDRISHTI (for devanagari script), Gurmukhi OCR (for punjabi script).

2.4.1 The DMACS OCR

The DMACS OCR was developed in Visual C++ 6.0 for the Windows platform. The preprocessing phase was developed chronologically after the main OCR program was developed. The preprocessing techniques used in this OCR have already been mentioned in section 2.2.

The next phase is that of segmentation. In this OCR system, a simple segmentation technique has been implemented. The first task is that of line segmentation. For this, The document is searched from top to down for presence of background rows, i.e., for rows that contain only background rows. An appropriate threshold number of rows would indicate the presence of the text-line separator. This may lead to discrepancies in the segmentation of Telugu scripts because of the presence of a mini line just below the main text. See figures 2.3. But, for segmenting the connected components, a region-growing algorithm is used. It starts at the first

[5]see chapter: 4, section: 4.5 of [TRI]

[6]see page 376

encountered text pixel and grows the character by looking for the presence of background pixels until some threshold level is reached. The algorithm is presented in [SAC].

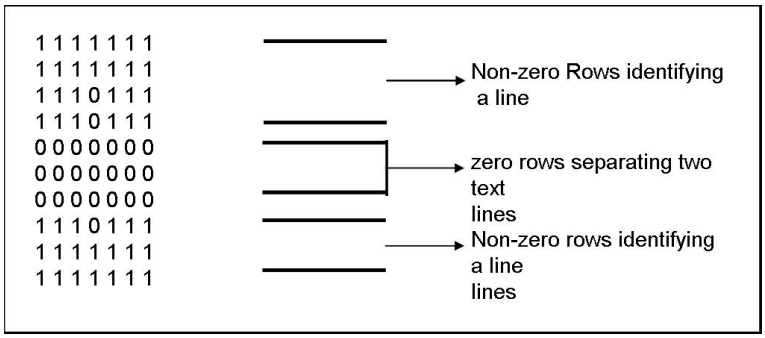


Figure 2.2: How segmentation is done.

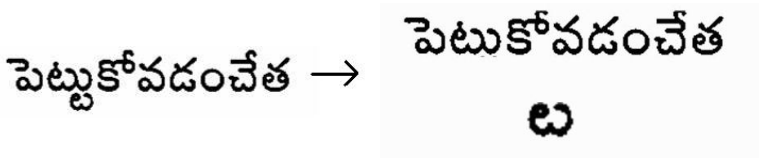


Figure 2.3: Sample Telugu scripts and the problem of segmentation.

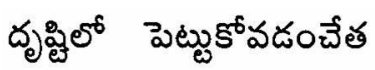


Figure 2.4: No problems in segmenting in this case.

The next phase is that of Feature Extraction and Classification. The DMACS OCR uses the Uniform sampling method and the Gradient Based Contour Encoding techniques. The Uniform sampling method, which is a rather intuitive and appealing method was proposed by Sachwani [SAC]. It consists of uniformly sampling the character horizontally and vertically and using the number of times the line crosses the character as the feature. The process is depicted in figure 2.5.

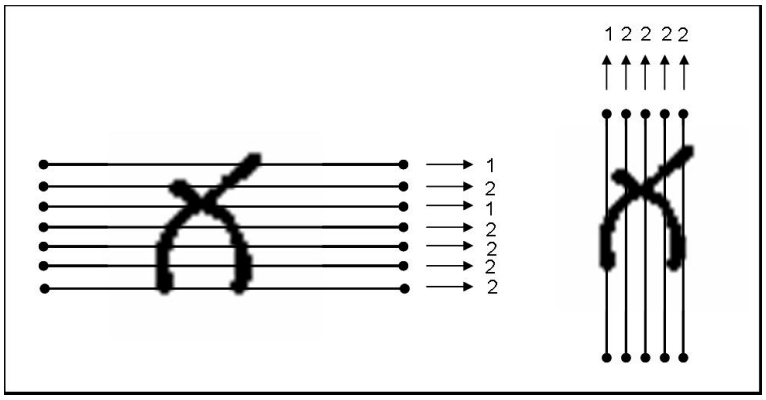


Figure 2.5: The Uniform Sampling technique.

The other method, the Gradient Based Contour Encoding is based on the observation that the object contour and the structure are encoded in the gradient direction and magnitude at each pixel of the image. The technique uses the Sobel operator. The features used here are of two types:

1. Binary Directional Features
2. Real valued Directional Features

The classifier implemented is the 1-Nearest Neighbour classifier.

The last phase is that of the post-processing. In this phase, the DMACS OCR conforms to the ISCII or the Indian Script Code for Information Interchange. The out code of the classifier is used to convert the characters into ISCII codes which are then added to a .aci file. The format of the aci file which are recognizable by standard Telugu text processors like *i*-Leap is provided in the appendix of [SAC].

The performance results of these OCRs are around 96.60% on the average.

Chapter 3

The DCT Based Feature Extractor

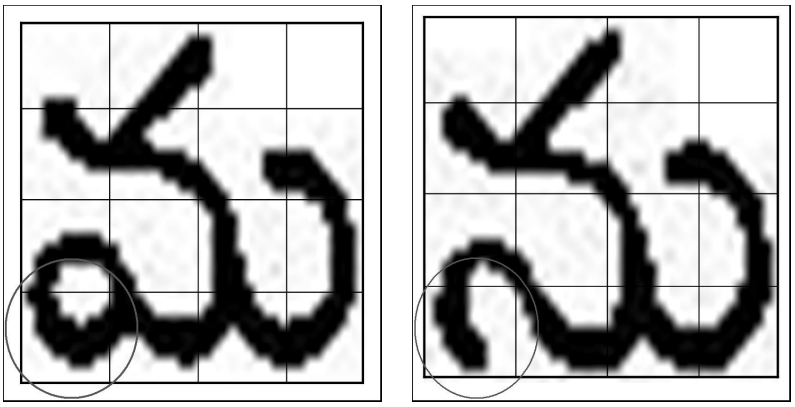
Everything should be as simple as possible, but no simpler.
-Albert Einstein

This chapter describes the core part of the present project,a Block DCT based feature extractor, that was developed as part of this work.

3.1 Introduction

3.1.1 Motivation

Consider the following images in figure 3.1. The idea for this feature extractor is very intuitive. Observe that the average number of text-pixels in any of the 16 grids varies for different characters. Consider now the following so called 'confusion-pairs' – characters which are similar and many OCRs tend to classify one for the other. The only way in which the two characters differ is the bottom-right curve in the 13^{th} block. That is, they are the same except for the small curve in the bottom



(a) 'MA' (b) 'NU'

Figure 3.1: The characters 'MA' & 'NU'. Note the circled portion.

block. One first observation is that there are text pixels at the bottom-right of the 13^{th} block of the first image while there are no text pixels at the bottom-right of the 13^{th} block of the second image. Clearly, the DC & AC frequencies of the two images definitely differ in the 13^{th} block. This is the source of the idea. For a single font, just the average pixel intensity may be adequate. But, for some fortuitous cases the average number of the pixels may turn out to be the same. Hence, we will have to use some of the AC frequencies of the image also. For this, we need to find a mathematical tool that will give us the DC and the AC components of the image.

According to P.Z. Myers, "the variations in intensity across a complex image can be treated as a harmonic function, which can be decomposed into a series of simpler waves − a set ranging from low frequency waves that change slowly across the width of the image, to high frequency waves that oscillate many times across it. We can think of an image as a set of spatial frequencies. If there is a slight gradient of intensity, where the left edge is a little bit darker than the right edge, that may be represented by a sine wave with a very long wavelength. If the image contains very sharp edges, where we have rapid transitions from dark to light in

the space of a few pixels, that has to be represented by sine waves with a very short wavelength, or we say that the image contains high spatial frequencies" [ANL].

Typically, to get the (*spatial*) frequency content of an image, the image transform that is used is the 2-dimensional Discrete Cosine Transform [RIC] which was proposed by Ahmed, Natarajan and Rao in 1974 [AHM].

In effect, the DCT expresses the image block as an array of weights for a set of two-dimensional repeating patterns, called basis functions.

3.1.2 The Discrete Cosine Transform: The Mathematics

Thus, the ideal mathematical tool for this is the Discrete Cosine Transform or the so called DCT. The DCT is defined as follows:

$$D(i,j) = C(i)C(j) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} p(x,y) cos\left[\frac{(2x+1)}{2N} i\pi\right] cos\left[\frac{(2y+1)}{2N} j\pi\right] \tag{3.1}$$

where,

$$C(u) = \begin{cases} \sqrt{\frac{1}{N}} & \text{if } u = 0 \\ \sqrt{\frac{2}{N}} & \text{if } u > 0 \end{cases} \tag{3.2}$$

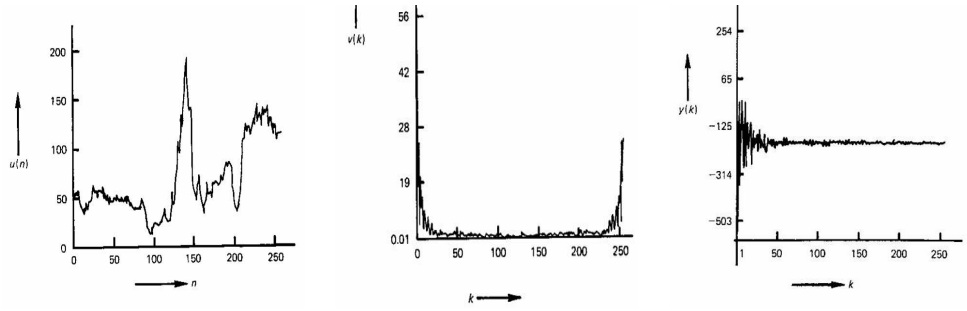
The matrix form of equation 3.1 is described by:

$$D(i,j) = \begin{cases} \frac{1}{\sqrt{N}} & \text{if } i = 0 \\ \sqrt{\frac{2}{N}}\ cos\left(\frac{(2j+1)}{2N} i\pi\right) & \text{if } i > 0 \end{cases} \tag{3.3}$$

The following are some of the salient properties of the DCT [ANL]:

1. The DCT is real and orthogonal. ie, $D = D^* \Rightarrow D^{-1} = D^T$
2. The DCT is a *fast* transform. That is, the time complexity of the DCT is comparable to that of the Fast Fourier Transform (FFT). The DCT of a vector of N elements can be calculated in O($Nlog_2N$) operations[1].
3. The DCT concentrates most of the image energy in very few coefficients, as can be seen from figure 3.2.

[1] see p.152 [ANL]



(a) The Source (b) DFT (c) DCT

Figure 3.2: Energy Distributions

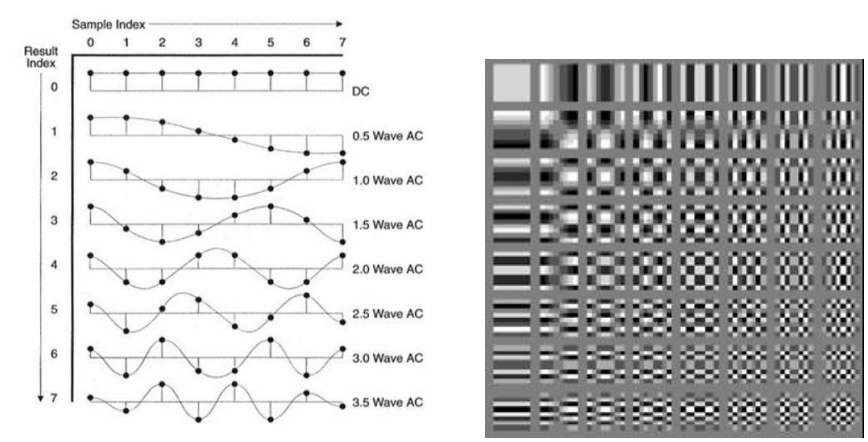
4. The $N \times N$ DCT is the next best choice to the KL transform, which is statistically the the most efficient transform [TOU].

3.1.3 The Discrete Cosine Transform: The Pictorial Description

In the case of 1-dimensional transform of a vector, the vector can be expressed as a linear combination of the *basis vectors*. Similarly, in the case of 2-dimensional transform of a matrix, the original matrix can be expressed as a combination of the *basis images*. The basis functions for the 1-dim DCT are shown in figure 3.3(a) and the basis functions for the 2-dim DCT are shown in the image 3.3(b).

Pictorially, the process of 1-dim DCT can be depicted as follows. The first or the DC coefficient is the simple wighted sum (which turns out to be an average) of all the elements of the vector, the weights being the discrete amplitude values of the cosine functions of the frequency equal to the index of the coefficient being calculated. The process is shown in figure 3.4 [FRD].

The 2-dimensional DCT uses the one-dimensional DCT as the basic operation. It works on each of the rows of the matrix to get the 1-dimensional coefficients. The coefficients so obtained are arranged in the same order, i.e., the first column



(a) One-dimensional DCT Basis Vectors (b) Two-dimensional DCT Basis Vectors

Figure 3.3: Basis Images

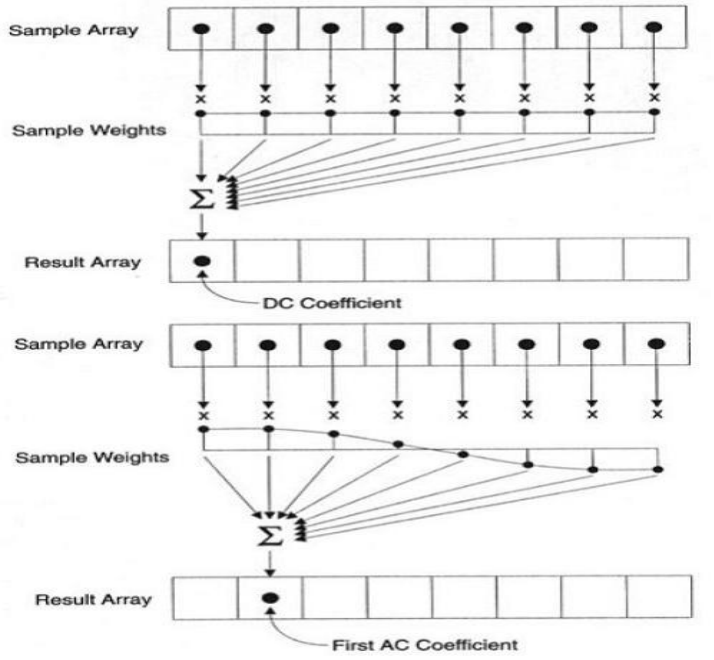


Figure 3.4: The process of DCT

contains the DC coefficients, the second column contains the first AC coefficients and so on. However, the *spatial* information is still retained because of the order of the arrangement of the coefficients. Thus it makes sense to take the 1-DCT of the columns now. This gives the 2-DCT of the original matrix, which, we conclude by the above discussion, contains the (*spatial*) frequency information in both the directions. The resulting frequency distribution is as in figure 3.5 [FRD].

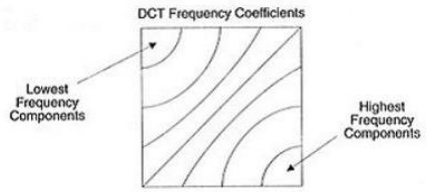


Figure 3.5: The frequency distribution of 2-DCT

Definition:

Spatial Frequencies: If f(x,y) denotes the luminance (intensity) of the image and x, y are the spatial coordinates, then, ξ_1, ξ_2 are the spatial frequencies that represent luminance changes with respect to spatial changes. The units of ξ_1 and ξ_2 are the reciprocals of the x and y respectively[2] [ANL].

The ordering of the 2-DCT coefficients used is the zig-zag ordering as shown in figure 3.6

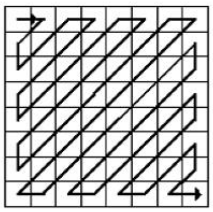


Figure 3.6: Zig-Zag ordering of 2-DCT coefficients

[2]see page 16 of [ANL]

3.2 The Feature Extractor(s)

Consider the figure 3.7. It shows the image of a character embedded in a 4-by-4 block. Each of these blocks is actually composed of 8-by-8 micro-blocks. In the following procedure, the image is first resized to a 32×32 block. Next, the 2DCT function is applied to the 16 micro-blocks of size 8×8 or size 4×4.

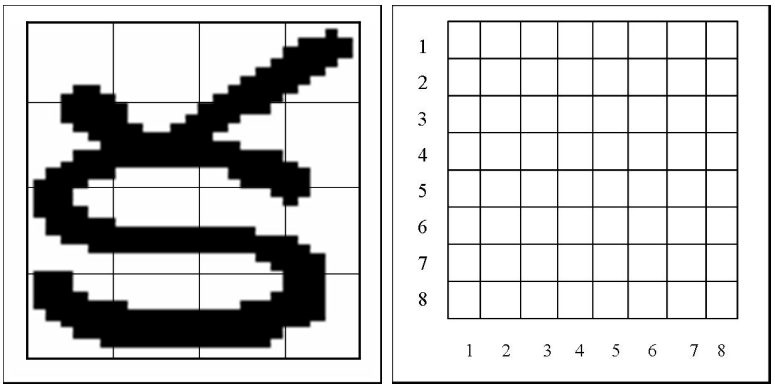


Figure 3.7: (a) A character framed in a 4-by-4 block and (b) a sample 8-by-8 sub block

3.2.1 The Design Cycle

The logical model that is used in designing the feature extractor[3] is shown in figure 3.8.

Data Collection

The first stage in PR system is that of data collection. The designer or developer must have enough data to study and come up with some conceptual model for the PRS. In our case, the typical samples were the character samples. Also, some sample passages and texts were obtained for the tests. Since the discrimination of confusion-pairs is critical to the process, many such pairs were also collected.

[3]The model followed here is as described by Duda, Hart and Stork in [DUD]

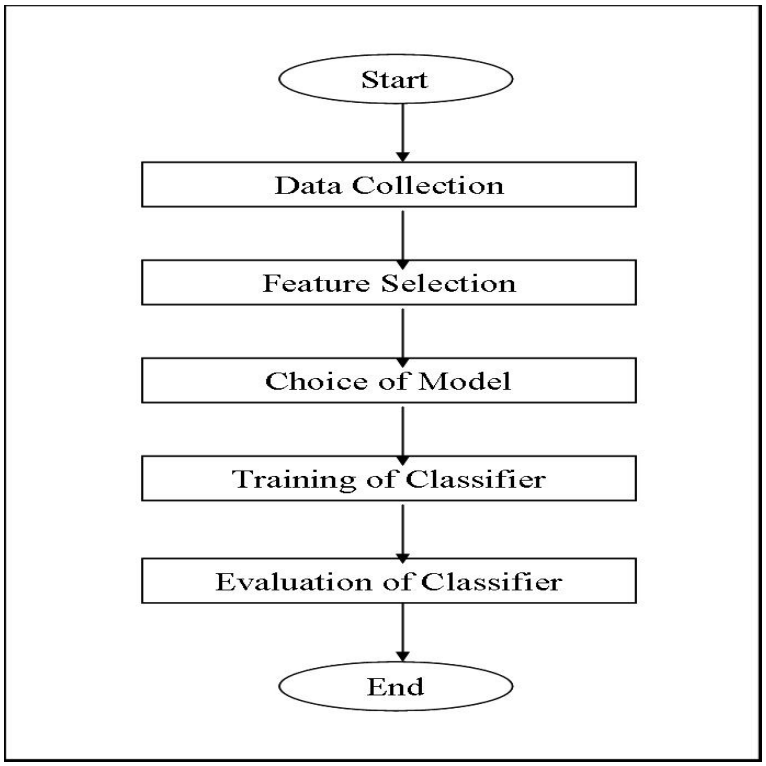


Figure 3.8: The Feature Extractor Design Cycle

Feature Choice

To start with, about fifteen different choices of features were made. All these were coded and tested on *typical* samples collected in the previous phase. The current phase is the **feasibility** or preliminary testing phase. It helped us decide if the feature(s) under consideration will make a reasonable candidate for deeper study. The features initially selected are enumerated here:

1. The DCT of the entire image was concatenated row-wise to get a single vector.

2. The DCT of the entire image was converted to a vector using the zig-zag ordering shown in 3.6.

3. The 1-DCT of each row was concatenated to obtain another feature.

4. The 1-DCT of each column was concatenated to obtain another feature.

5. The above two features were concatenated to obtain another feature.

6. The DC components of each of the 16 micro-blocks of the Block 2-DCT (blocksize being 8-by-8) were concatenated to get another feature.

7. The DC and the first AC component of the *Blockwise* 2-DCT (blocksize being 8-by-8) were concatenated to get another feature.

8. The DC and the first two AC component of the *Blockwise* 2-DCT (blocksize being 8-by-8) were concatenated to get another feature.

9. The DC components of the of the 1-DCT of each row were concatenated to get another feature.

10. The DC and the first AC components of the of the 1-DCT of each row were concatenated to get another feature.

11. The DC and the two first AC components of the of the 1-DCT of each row were concatenated to get another feature.

12. The DC components of the of the 1-DCT of each column were concatenated to get another feature.

13. The DC and the first AC components of the of the 1-DCT of each column were concatenated to get another feature.

14. The DC and the two first AC components of the of the 1-DCT of each column were concatenated to get another feature.

15. The concatenation of the above two in each case was also considered.

The features stated above implemented in Matlab. The in-built `dct()` & `dct2()` functions were used for calculating the 1- & 2-DCTs. As a sample, the

main steps of the algorithm for the feature extraction using Block DCT is described below:

Input: The input used was a 256 color bitmap image (in this case, the bitmap was created in MS Paint).

1. The input image is first binarized. This was achieved through the thresholding of the input (gray) image at the level of 128 intensity.

2. The binarized image is then clipped of all the margins. That is, the image is clipped so that there are no background pixels surrounding the image. The algorithm developed by Sachwani et.al., as part of [SAC] is used.

3. The binarized image is then segmented into its component characters.

4. The segmented image is then given as input to the 2DCT. To get the block-wise DCT of the 16 8-by-8 sub-matrices are given as input to the 2DCT function. The output DC coefficients of the 16 matrices are concatenated to form a 1-by-16 feature vector. This vector is saved for each of the input characters to build the dictionary.

5. The function was run on sample "*confusion-pairs*" or characters that are much alike. Some typical confusion-pairs are shown in figure 3.9.

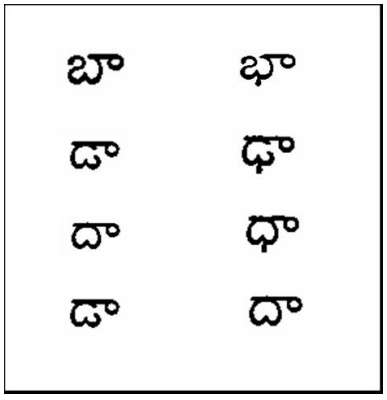


Figure 3.9: Some typical confusion pairs

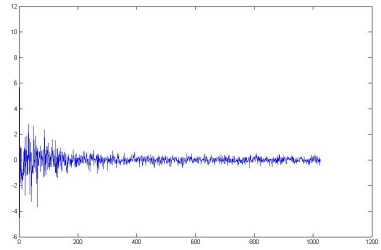
Some sample code implementing this algorithm is listed here.

```
function c=c_blockwise_DC_8size(image)
bin_image = image;
clipped_bin_image = clipallmargins(bin_image);
resized_clipped_bin_image=imresize(clipped_bin_image,[32,32]);
i=1;j=1;k=8;l=8;
step=8;
plotnumber=0;
for countx=0:3,
    for county=0:3,
         a=c_getsubmat(resized_clipped_bin_image,
         i+(countx*step),j+(county*step),
         k+(countx*step),l+(county*step));
         b=dct2(a);
         plotnumber=plotnumber+1;
         c(1,plotnumber)=b(1,1);
     end
end
return;
```

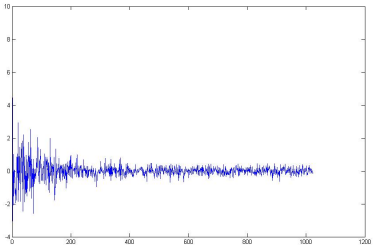
Model Choice

The above listed choices of the feature vectors were tested in the preliminary round for **feasibility**. Figure 3.10 shows some of the plots of the features proposed above. The plots show the vector form of the DCT coefficients for the characters 'MA' & 'NU' plotted against the magnitude of the coefficients. In the first case, all the coefficients are plotted. In the second case, only the DC components of the 16 micro-blocks are plotted. The third case plots the DC coefficients of the 64 micro-blocks. The last row shows the plots of the DC components of the the 32 columns of the character image. These show that the features vectors are different in each of the cases for both the characters.

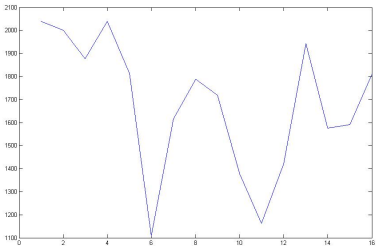
The greater the euclidean distance between the feature vectors, the more discriminated they are. It was observed that, all the feature vectors gave reasonably good classification. For example, for the DC component feature vectors, the euclidean distance between the feature vectors of the two characters 'MA' & 'NU'



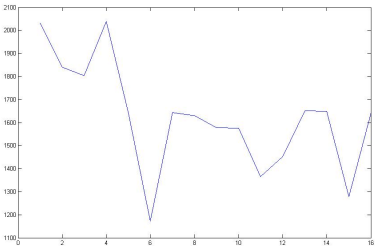
(a) DCT of Complete ‘Ma’



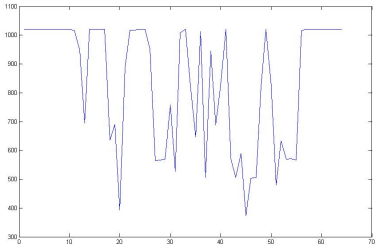
(b) DCT of Complete ‘Nu’



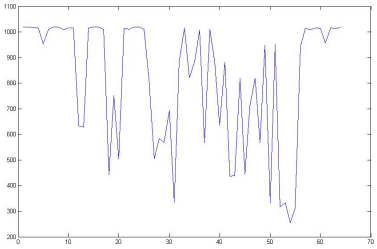
(c) ‘Ma’: DC components for blocksize 8



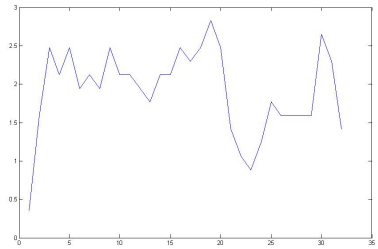
(d) ‘Nu’: DC components for blocksize 8



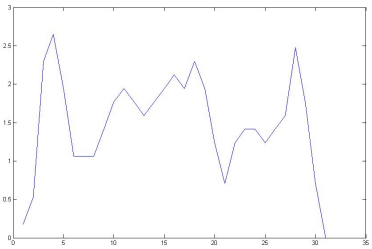
(e) ‘Ma’: DC components for blocksize 4



(f) ‘Nu’: DC components for blocksize 4



(g) ‘Ma’: DC components for columns



(h) ‘Nu’: DC components for columns

Figure 3.10: Feature Vector plots for ‘MA’ & ‘NU’ respectively.

was about 10 units and about 16 units for the case of the feature vector consisting of the DC and first AC coefficients for the blockwise DCT approach. This distance was much larger for the feature vectors obtained using the DC coefficients of the DCT applied to the rows and columns of the character image.

So for further investigation, we decided to test the feature vector which has minimal time and space complexity. For this, the choice would have to be the feature vector with the smallest size because the algorithm applied is otherwise the same. Thus, it will be computationally least demanding to choose a small feature vector. But, the results are scrutinized to check if this time and space efficiency has reduced the accuracy of the classifier. If necessary, we have to make some necessary changes to the feature extractor. To start with, the blockwise 2-DCT (see item 6 above) was chosen as it had the smallest size among all the contending feature vectors: 1-by-16 and also was quite discriminating.

The euclidean distance along with 1-NN (or the 1-Nearest Neighbour) classifier is used. The 1-NN is preferred over n-NN classifier because,

1. It is simple and fast to implement, and, more importantly,

2. It is the statistically better classifier, with a maximum of twice the error of Bayes classifier (which statistically gives the least error). The result in this regard is given in [TOU][4]:

$$p_B \leq p_{e_1} \leq p_B \left(2 - \frac{M}{M-1} p_B\right) \tag{3.4}$$

where,

p_{e_1} is the probability of error of 1-NN classifier,

p_B is the Bayes probability of error, and,

M is the number of pattern classes under consideration.

The simplicity and the discriminating power of this feature were striking. The results are discussed in the next chapter.

[4]see pages 81-83 of [TOU] for a detailed derivation of the proof

Building the Dictionary

There are two basic forms of training or learning:

1. Supervised
2. Unsupervised

For greater details, please refer to [DUD][5]. We use the unsupervised learning.

The next part of the work is to build a good dictionary. The Indian languages are rich in the variety of the character set. So, it is clearly not possible to build the dictionary in one go. To start with, all the common characters can be added to the dictionary. This forms the basis of the training phase. More sophisticated training algorithms include the Neural Net with its now legendary (multi-layer) **back-propagation** [TOU] algorithm so as to propagate the error to the source, which enables it to minimize the error at all *previous* levels of the classifier.

In our case, we had a basic set of characters from the data collection phase. The feature vectors of these characters were extracted and an initial dictionary was built. A basic method to improve this dictionary which was implemented is explained shortly.

In this work, we follow the unsupervised training, and use the **Similarity Measure** approach. We tested two similarity measures:

1. Euclidean metric, in the $\Re^n$ space, n being the dimension of the feature vector under consideration.
2. Canberra metric, which is known to have clustering properties.

Of these two, the canberra metric failed to give even reasonable results. So, the similarity measure we used for later phases of the work is the **euclidean metric in** $\Re^n$. We have also implemented (again, in Matlab) in this work a facility we thought up, and are of the opinion that it will help to improve the performance

[5]see p.414

of the extractor: a facility to add to the dictionary those characters that are not presently in the dictionary. So, we came up with a scheme where the classifier has two modes: *Classifier mode* and the *Dictionary mode*. In the classifier mode, it requires no input from the user except that of the image file to be OCR-ed. In the other mode, which is the **interactive** dictionary mode, the user is asked after each character recognized, whether to add the character to the dictionary or not. The user can refer to the character-dictionary (another file containing all the characters whose feature vectors are currently in the dictionary) and decide whether to add the character to the dictionary or not. Thus the user can use to improve the performance of the classifier. As another application of this facility, is the creation dictionaries for characters of different fonts or even different scripts from the scratch.

It is here that the classifier output, the results of the feature extraction process are put to test. The results in our case was a text file which contains the following parameters:

Evaluation

This is the most critical part of the feature extraction process, as it is the culmination of the above mentioned laborious phases.

1. The classifier code for the current character.
2. The Euclidean distance of the current character from the closest neighbour in the feature extractor's dictionary or database.
3. The Feature Vector of the current character under consideration.
4. The Feature Vector of the dictionary character to which the current character has got mapped.
5. A binary representation of the current character, and

6. A binary representation of the character to which the current input character has got mapped or classified.

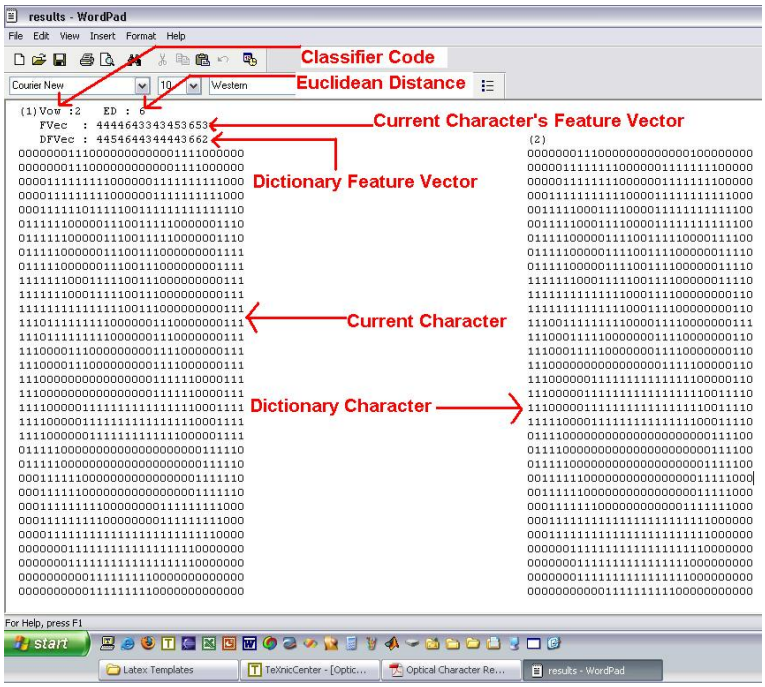


Figure 3.11: A screen shot of the results file. Note that the character 'Aa' is outlined by the 1s in the current and dictionary.

The format of the text file was chosen so as to contain all the necessary (but minimal information, so as to keep file I/O to the bare minimum, as this affects the time performance of the feature extractor) information for making a quick decision for evaluation of the results which could otherwise take a long time. A sample screen shot of the results file is given in figure 3.11.

For a simpler solution, the output can be in the form of the Unicode representation of the characters or output them in the ISCII format as a *.aci file which are recognizable by standard Telugu text editors like *i*-Leap [SAC], but this would require more sophisticated code. Since this work involved the prototyping of only

the extractor, this was not implemented as it was considered outside of the scope of this work.

All the algorithms were implemented in Matlab 7.0 R-14 and tested on an IBM Think Center R-51 running Windows XP (service pack 2).

Chapter 4

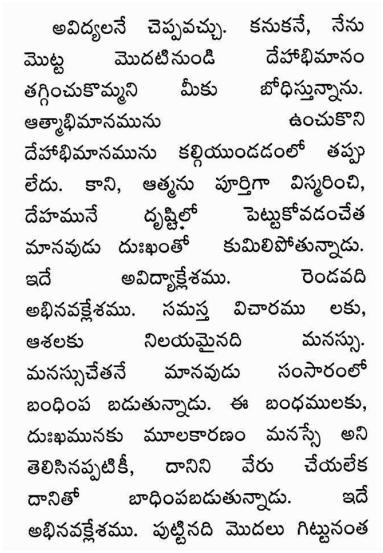
Results, Implementation & Discussion

Karmanyeva Adhikaraste, Ma Phaleshu Kadachana.
-Bhagwadgita

This chapter presents the results obtained, and discusses some of the limitations of the methods. It also discusses the future direction envisaged by the project.

4.1 The Tests and Results

The first test was to give a randomly selected sample page. Later, the sample used by Sachwani et.al., [SAC] was used to test the prototype. This sample is shown in figure 4.1. The sample page was randomly picked up from the April 2005 issue of the spiritual magazine "Sanathana Sarathi". A sample of 100 characters was considered at a time and analysed. Then all the results were consolidated to get the average performance. The samples are shown in figure 4.1.



(a) The text used by Sachwani et.al. [SAC]

అప్పుడు స్వామి అన్నారు,"నేను ఆ సంస్థలను, వాటి భవనాలను చూసి, ఉన్న డబ్బును చూసి స్వాధీనం చేసుకోవడం లేదు. ఆ సంస్థను నిర్వహిస్తున్న వ్యక్తులను చూసి అందుకు అంగీకరించాను. వారందరూ సేవాదురంధరులు, త్యాగశీలురు, ధర్మనిరతులు."

1977 విజయదశమినాడు ప్రశాంతినిలయంలో చిరస్మరణీయమైన ఒక పెద్ద సభ జరిగింది. ఆ పుణ్యదినంనాడే ఆ సంస్థల స్వాధీనాన్నిగురించిన ఒక ప్రకటనకూడా వెలువడింది. ట్రస్టు ఆ సంస్థలను స్వాధీనం చేసుకుంటున్నది, అన్నది ఆ ప్రకటనలోని ముఖ్య సారాంశం. ఆ ప్రకటనను విన్న మరుక్షణం, మా మనసులనిండా ఒకానొక దివ్యమైన అనుభూతి ఆక్రమించింది. మేము మా జీవితలక్ష్యాన్ని సాధించామన్న ఒక భావావేశం మమ్మల్ని పరవశుల్ని చేసింది.

అయితే, ట్రస్టు మా సంస్థలను సంపూర్ణంగా స్వాధీనం చేసుకోవడానికి ముందు కొన్ని అడ్డంకుల్ని అధిగమించవలసి వచ్చింది. ఆ ఆటంకాలు అళికె, ముద్దేనహళ్ళిలలో ఉన్న వ్యవసాయ, వ్యవసాయేతర భూములు పరులకు స్వాధీనం చేయడంలోగల న్యాయ సంబంధమైన చిక్కులకు సంబంధించినవి. ముద్దేన హళ్ళిలో ఉన్న భూముల స్వాధీనంగూర్చిన సమస్య చాలా జటిలమైనది. నేను జిల్లాలోనూ, రాష్ట్రప్రభుత్వ అధికార యంత్రాంగందగ్గర వెళ్ళనివారు లేరు, ఎదుర్కొనని ఆటంకం లేదు. అది ఒక బృహత్తర ప్రయత్నం. చాలాసార్లు, అధిగమించడానికి వీలు కానట్లుగా కనిపించే ఆ అడ్డుగోడల్ని చూసి నేను నీరుగారిపోయాను. అన్న లేని లోటు ప్రతి క్షణంలోనూ నాకు కనిపించింది. దానికితోడు అదే సమయంలో మా అమ్మగారిని బెంగుళూరులో ఒక ఆసుపత్రిలో చేర్చవలసి వచ్చింది. అనుకోకుండా ఆమెకు సర్జరీ అవసరమైంది. మా అమ్మగారికి ఆపరేషను ఇష్టం లేదు. నేను చాలాసార్లు ఆసుపత్రికి వెళ్ళి వస్తుండేవాడిని. ఒకసారి మా అమ్మగారు నన్ను అడిగింది, "బాబూ, స్వామిని ప్రార్థించరాదా నాకు ఆపరేషను లేకుండా ఈ జబ్బును నయం చేయమని?"

(b) The sample text from Sanathana Sarathi that was used for the runs

Figure 4.1: Sample texts used

The results discussed here pertain to the 1-NN classifier with Euclidean Distances. The tests were also performed with the Canberra metric but, the results were not satisfactory.

The results of the test are as tabulated in tables 4.1 and 4.3. We wish to point out that the character-set in the dictionary used for the study of the DC coefficient feature vector of size 8×8 are not the same as those used for the other cases. The former case underwent some *learning*. We also wish to mention that the image was rotated be about 3-5 degrees (introduced fortuitously at the scan phase). So, the results also show the tolerance (to rotation) of the method to a certain extent.

Description	Blocksize 8×8	Blocksize 4×4
Total # of characters	702	702
Total # of characters wrongly segmented	43	43
Total # of characters not in the dictionary	80	80
Total # of characters not being considered	123	123
Net # of characters to be considered	579	579
Total # of wrongly classified characters	55	24
Total # of correctly classified characters	524	555
% of characters correctly classified	90.50	95.86

Table 4.1: Results of the test for blocksizes of 8×8 and 4×4

4.2 Discussion

4.2.1 Implementation

This section analyses and explains the results obtained. The results show that the performance of the test with the blocksize 8×8 is is poorer than that with the blocksize 4×4. The results in the latter case are comparable to those obtained by Sachwani et.al., for the same fontsize. This may be attributed to the size of the region in focus. In the earlier case, we considered a neighbourhood of a larger size. This may be insensitive to the smaller changes in the neighbourhood. Changes in one place may cancel out those in another. In the latter case, changes much more minute are reflected in the feature vector.

The results are now analysed in the light of the implementation of the prototype itself. The prototype was implemented in Matlab. The following are the steps of the algorithm:

1. The input image is binarized with a threshold of 128.

Description	All other sizes	8×8 DC
Total # of characters	263	263
Total # of characters wrongly segmented	14	14
Total # of characters not in the dictionary	47	31
Total # of characters not being considered	61	45
Net # of characters to be considered	202	218

Table 4.2: Statistics for the tests on text used by Sachwani et.al.

Description	Blocksize 8×8		Blocksize 4×4	
	DC	DC & 1AC	DC	DC & 1AC
Total # of wrongly classified characters	14	10	10	11
Total # of correctly classified characters	204	192	192	191
% of characters correctly classified	93.58	95.05	95.55	94.56

Table 4.3: Results of the test on the text used by Sachwani et.al.

2. The binarized image is then segmented into characters.
3. The segmented characters are then normalized to a matrix of size 32×32.
4. This matrix is then subjected to the inbuilt `dct2()` function.
5. The integer part of the DC component of the DCT alone is taken and the feature vector is generated.

The above algorithm brings to light many of the approximations that are implicitly being assumed, some of which are listed here.

1. The threshold value may cause the binary image to be distorted and hence cause the input to be noisy.

2. The normalization of the characters uses intra/extrapolation. The Matlab function uses the "Linear" method. This also may be a source of distortion in the characters [TRI][1].

3. The `dct2()` function itself may be using a lot of approximations. Moreover, since the program is running on a 32 bit machine, this itself may affect the precision of the arithmetic involved.

4.2.2 Other Considerations

In our view, the most important part of the feature extractor is the *dictionary*. The size of the dictionary is crucial to the performance. The larger the dictionary, greater is the probability of misclassification. The method used in the prototype can be improved. The current method segments the entire character. But, for Indian languages like Telugu, each character can be composed of smaller components as explained in [SAC]. Their method uses the region-growing algorithm to segment the connected components. This way, the common sub-components will be in the database only once. Otherwise, all the permutations and combinations of the sub-components will bloat the size of the dictionary and also introduce unnecessary redundancy.

Another consideration regarding the size of the dictionary is the search time. Currently, the prototype searches the database sequentially. This may not be a good idea. For larger sizes of the database, the search time could grow exponentially. Hence in such cases, a small sized database is preferred. Otherwise, a better search algorithm like 'Hashing' or B^+- tree can be implemented.

4.3 Future Developments

Here we summarize the suggestions made through out the report.

[1] see page 4 of [TRI]

1. A more efficient DCT algorithm can be contemplated. Many fast algorithms have already been proposed in the literature.

2. The normalization procedure used in this algorithm to resize the images, needs to be reconsidered in light of what was mentioned earlier.

3. The segmentation procedure also needs to be reconsidered so as to efficiently segment the images and to minimize the size of the dictionary as was mentioned earlier.

4. An integration of this prototype with the existing DMACS OCR will facilitate the use of the tested segmentation techniques implemented there.

5. Note that the DC component of the DCT is a simple average. Since we are using only the DC component of the DCT, the DCT algorithm can be simple replaced by a simple average of the block pixels in order to make the code more efficient. A similar argument holds for the case of the DC & first AC components.

6. As was mentioned earlier, the DC component was chosen from among the many contending feature extractors for the simple reason that it seemed to work well and was computationally the least expensive. But, feature extractor formed by the concatenation of the row and column DC coefficient seemed to give the best results. This too can be investigated in the light of the previous comment.

7. Further coefficients of the DCT may also be considered in improving the performance of the feature extractor.

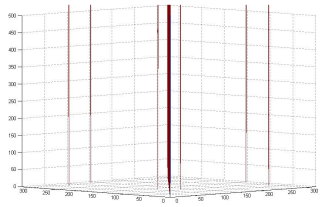
8. Note that the euclidean distance classifies equally the points that lie on the hypersphere in the $\Re^n$ space, where n is the size of the feature vector.

   Hence, an alternative metric to the Euclidean distance can be considered. In our opinion, a metric which would take into consideration the positional

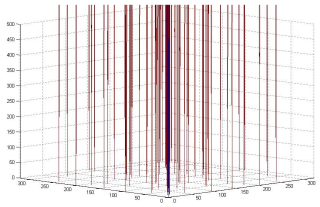
aspect of the elements of the feature vector should possibly yield better results.

9. An additional task would be the integration of this system with a text-to-speech system, which would be very relevant for use in educating blind/visually impaired people.

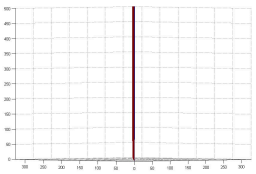
4.4 An analysis of the Dictionary



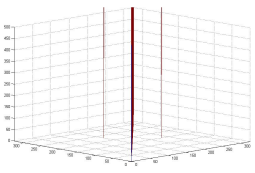
(a) DC component Blocksize=8, $\theta = 2$



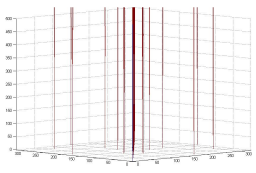
(b) DC component Blocksize=8, $\theta = 5$



(c) DC component Blocksize=4, $\theta = 2$



(d) DC component Blocksize=4, $\theta = 10$



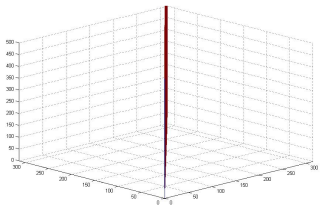
(e) DC component Blocksize=4, $\theta = 15$

Figure 4.2: Plots of the DC component feature vector for the dictionaries with blocksizes 8×8 & 4×4 ; θ is the eulidean distance.

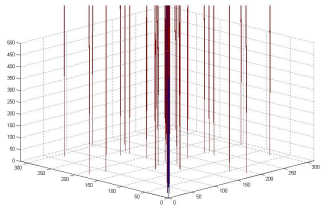
We conclude the discussion of the results with an analysis of the dictionaries themselves. The aim of the study is to check the difference between the feature vectors in the dictionary using the same distance metric as the classification does. If the dictionary reveals many feature vectors close together, it means that the feature vector cannot be very discriminating. The results obtained in the tests

confirm to this observation as is revealed in the following plots. The experiment we set up was as follows:

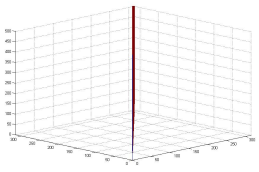
The distances of each feature vector from all the other feature vectors were calculated. This information was stored in a $n \times n$ matrix say Z. The element $Z(i, j)$ corresponds to the distance of the j^{th} vector *from* the i^{th} vector. That is, the i^{th} row of Z corresponds to the distances of the features vectors from the i^{th} feature vector. The vertical lines that are seen reveal that the distance between some feature vectors is below the threshold θ. The x- and y- axes are the indices of the feature vector in the dictionary (the order of the feature vectors in the dictionary is fixed), and the z-axis is the distance axis. The point (x,y) can be used to find out which of the feature vectors are close to each other within the threshold θ.



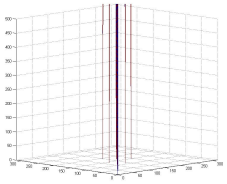
(a) DC & 1^{st}AC component Block-size=8, $\theta = 2$



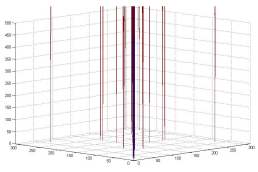
(b) DC & 1^{st}AC component Block-size=8, $\theta = 10$



(c) DC & 1^{st}AC component Blocksize=4, $\theta = 10$



(d) DC & 1^{st}AC component Blocksize=4, $\theta = 25$



(e) DC & 1^{st}AC component Blocksize=4, $\theta = 30$

Figure 4.3: Plots of the DC & $1^{st}AC$ component feature vector for the dictionaries with blocksizes 8×8 & 4×4; θ is the eulidean distance.

Note that the plots are symmetric about the plane x=y. So a study of one half of the graph would suffice.

The figure 4.44.4 shows the plots.

Notice that there are many feature vectors within a threshold $\theta = 5$ for the DC components with blocksize 8. Whereas, for blocksize 4, $\theta = 2$ and $\theta = 10$, has a negligible number of θ close feature vectors. Only at $\theta = 15$, do the numbers grow; but the number is still much less than that in the former case. The trend slows down when it comes to the feature vectors with the DC and the 1^{st} AC component. The least problematic dictionary appears to be the case of DC & 1^{st} AC component feature vectors with blocksize 4×4.

4.5 Conclusions

To summarize, the preprocessed image was segmented into its component characters and then resized to a 32×32 image. The Block DCT was then applied to the 16 micro-blocks of this image. The output DC coefficients were concatenated to get one feature vector. Also, the DC and first AC coefficients were concatenated to get another feature vector. These feature vectors were studied for their discriminating power. The results are of the order of 95.6% accuracy in discriminating. For characters of a single font, the DC coefficient feature vector seemed adequate for most of the characters.

In conclusion, the Block DCT was found to be a reasonably good tool for feature extraction purposes. Further investigation with the above mentioned enhancements would probably yield more optimum results.

Appendix A

Sample Code

This appendix lists some of the Matlab code that implements the Feature Extractor.

A.1 Binarization

```
function i=c_bin(image)
i=image;
[nx,ny]=size(i);
for countx=1:nx,
    for county=1:ny,
        if( i(countx,county) <= 128)
            i(countx,county) = 1;
        else i(countx,county) = 0;
        end
    end
end
return
```

A.2 Zig-Zag Feature Vector

```
function j=c_vectorize(image)

%initalize the first element
j=image(1,1);
```

```
[nrows,ncols] = size(image);
%check for SQUARE matrix
if(nrows ~= ncols)
    display 'Warning! : Not a square matrix..Aborting.'
    return;
end
%traverse the upper triangle
count=1;
lower_triangle=0;
while(count~=nrows)
    if( count~=nrows )
        count=count+1;
        j=c_xincr_ydecr(image,count,lower_triangle,j);
    end
    if(count~=nrows)
        count=count+1;
        j=c_xdecr_yincr(image,count,lower_triangle,j);
    end
end
%traverse the lower triangle
count=1;
lower_triangle=1;
while(count~=nrows)
    if( count~=nrows )
        count=count+1;
        if(rem(nrows,2)==0)
            j=c_xdecr_yincr(image,count,lower_triangle,j);
        else
            j=c_xincr_ydecr(image,count,lower_triangle,j);
        end
    end
    if(count~=nrows)
        count=count+1;
        if(rem(nrows,2)==0)
            j=c_xincr_ydecr(image,count,lower_triangle,j);
        else
            j=c_xincr_ydecr(image,count,lower_triangle,j);
        end
    end
end

return;
```

```
function j=c_xdecr_yincr(image,count,lower,j)

[nrows,ncols] = size(image);
if(lower==0)
    bound=0;
    countx=count;
    county=1;
    while(countx ~= bound)
        j=c_appendarray(j,image(countx,county));
        countx=countx-1;
        county=county+1;
    end
else
    bound=nrows;
    countx=nrows;
    county=count;
    while(county ~= bound+1)
        j=c_appendarray(j,image(countx,county));
        countx=countx-1;
        county=county+1;
    end
end
return;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function j=c_xincr_ydecr(image,count,lower,j)

[nrows,ncols]=size(image);
if(lower==0)
    bound=0;
    countx=1;
    county=count;
    while(county ~= bound)
        j=c_appendarray(j,image(countx,county));
        countx=countx+1;
        county=county-1;
    end
else
    bound=nrows;
    countx=count;
    county=nrows;
    while(countx ~= bound+1)
```

```
        j=c_appendarray(j,image(countx,county));
        countx=countx+1;
        county=county-1;
    end
end

return;
```

A.3 Feature Extraction for Blockwise DC coefficients

```
function c_blockwise(image)

bin_image = c_bin(image);
clipped_bin_image = clipallmargins(bin_image);
resized_clipped_bin_image=
 imresize(clipped_bin_image,[32,32],'bilinear');
i=1;j=1;k=8;l=8;
step=8;
x_direction = [1:1:8];
y_direction = [1:1:8];
plotnumber=0;
figure
for countx=0:3,
    for county=0:3,
        a=c_getsubmat(resized_clipped_bin_image,i+(countx*step),
         j+(county*step),k+(countx*step),l+(county*step));
        b=dct2(a);
        c=c_vectorize(b);
        plotnumber=plotnumber+1;
        subplot(4,4,plotnumber),plot(c);
    end
end
return;
```

A.4 Metrics

Canberra Metric

```
function dis = c_file_canberra(x,y)
dis=0;
[countx,county] = size(x);
for count=1:county
    if( double(x(count))+double(y(count)) ~= 0 )
        dis = dis + (abs((double(x(count))-double(y(count))))
         /(double(x(count))+double(y(count))));
    end
end
return
```

Euclidean Metric

```
function dis = c_file_euclid(x,y)
dis = sum((int8(x)-int8(y)).*(int8(x)-int8(y)));
return
```

A.5 Run the Tests

```
function temp=get_test(filename,mode)
global dir;
%set the dir to the current directory.
dir='D:\iimsc\scharan\Dissertation\Charan\M Files\
Modified For Files\My Files\Sample\';
disp('blockwiseDC 1AC size8');
global c_count;
c_count=0;
temp=get_lines(filename,mode);
return;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [character_line,count]=get_lines(filename,mode)
global dir;
%Obtains the characters from an image...
image=imread(filename);
image=c_bin(image);
[x,y]=size(image);
%initialize the character line
character_line=[];
```

```
character_line_count=1;
started_building=0;
finished_line=0;
count=0;
fid=fopen(strcat(dir,'sample.txt'),'w');
%get the lines...
while(finished_line~=1)
    for pixel_rows=1:x
        temp_row=image(pixel_rows,:);
        %reset the character row count
        character_row=0;
        for pixel_col=1:y
            %do not go in again if a text pixel was found
            if(character_row~=1)
                if(temp_row(pixel_col)~=1)
                    pixel_col=pixel_col+1;
                else
                    %found a text pixel !
                    character_row=1;
                end
            end
        end
        %Check if the pixel row had any text pixel...
        if character_row==1
            started_building=1;
            character_line(character_line_count,:)=temp_row;
            character_line_count=character_line_count+1;
        else
            %now we have populated the character line and found a
            %line of pixels with no text pixel ! So, the line ends...
            if(started_building==1)
                finished_line=1;
                count=count+1;
                started_building=0;
                for icount=1:character_line_count-1
                    fprintf(fid,'%d',character_line(icount,:));
                    fprintf(fid,'\n');
                end
                [a,b]=get_characters(character_line,mode);
                character_line=[];
                character_line_count=1;
            end
        end
```

```
    end
end
fclose(fid);
return;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [character,count]=get_characters(image,mode)
%initializations
[x,y]=size(image);
global c_count;
%initialize the character line
characters=[];
character_pixels_cols_count=1;
started_building=0;
finished_line=0;
count=0;
%get the characters, line by line...
while(finished_line~=1)
    for pixel_col=1:y
        temp_col=image(:,pixel_col);
        %reset the charcter column count
        character_col=0;
        for pixel_row=1:x
            %do not go in again if a text pixel was found
            if(character_col~=1)
                if(temp_col(pixel_row)~=1)
                    pixel_row=pixel_row+1;
                else
                    %found a text pixel !
                    character_col=1;
                end
            end
        end
        %Check if the pixel row had any text pixel...
        if character_col==1
            started_building=1;
            character(:,character_pixels_cols_count)=temp_col;
            character_pixels_cols_count=character_pixels_cols_count+1;
        else
            %now we have populated the character and found a column
            %of pixels with no text pixel ! So, the character ends...
            if(started_building==1)
```

```
                finished_line=1;
                count=count+1;
                started_building=0;
                [cx,cy]=size(character);
                c_count=c_count+1;
%CHOOSE THE CORRECT FILE HERE!
                z=test_character_DC_1ACsize8(character,mode);
                character_pixels_cols_count=1;
                character=[];
            end
        end
    end
end
return;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function z=test_character_DC_1ACsize8(image,mode)
global dir;
global c_count;
%TEST THE NEW FEATURE FOR THE FILE
fid=fopen(strcat(dir,'results.txt'),'a+');
%Get the blockwise DC feature vector
fv=int8(c_blockwise_DC_1AC_size8(image));
count=0;
global dictionary;
load(strcat(dir,'dictionary'));
[sx,sy]=size(dictionary);
for count=1:sx
    z(count)=c_file_euclid(dictionary(count,:),fv);
end
%%%%%%%% all this is just to print results properly...
    [a,b]=min(z);
    Distance=a;
    Index=b;
    dict_const_fid=0;
    if(b<=262)
        dict_cont_fid=fopen(strcat(dir,'TEST_blockwise_DC_const.txt'),'r');
        dict_const_fid=dict_cont_fid;
        c=b;
    else
        dict_cont_fid=fopen(strcat(dir,'TEST_blockwise_DC_vowel.txt'),'r');
        dict_const_fid=dict_cont_fid;
```

```
        c=b-262;
    end
    temp=0;
    for i=1:c-1
                temp=fscanf(dict_const_fid,'(%d)\n');
                for i=1:32
                    temp=fscanf(dict_const_fid,'%c\n',[1,32]);
                end
                temp=fscanf(dict_const_fid,'%c\n',[1,16]);
    end
    if (mode==0)
        fprintf('LEARNING MODE\n');
        append_dictionary(image);
    else
%       ONLY CLASSIFICATION...
        fprintf(fid,'=======================================\n');
        character=image;
        temp_character=clipallmargins(character);
        temp_character=imresize(temp_character,[32,32]);
        fprintf(fid,'(%d)',c_count);
        if(b>262)
            Character_Index=(b-262);
            fprintf(fid,'Vow :%d\t',Character_Index);
        else
            Character_Index=b;
            fprintf(fid,'Con :%d\t',Character_Index);
        end;
        fprintf(fid,'ED : %d',Distance);
        fprintf(fid,'\n   FVec  : ');
        fprintf(fid,'%d',fv);
        fprintf(fid,'\n   DFVec : ');
        fprintf(fid,'%d',dictionary(b,:));
        fprintf(fid,'\t\t\t\t\t\t\t\t');
        temp=fscanf(dict_const_fid,'(%d)\n');
        fprintf(fid,'(%d)\n',temp);
        for icount=1:32
            fprintf(fid,'%d',temp_character(icount,:));
            fprintf(fid,'\t\t\t\t\t\t\t');
            temp=fscanf(dict_const_fid,'%c\n',[1,32]);
            fprintf(fid,'%c',temp);
            fprintf(fid,'\n');

        end
```

```
        temp=fscanf(dict_const_fid,'%c\n',[1,16]);
        fprintf(fid,'\n');

    end
fclose(fid);
return;
```

A.6 Append the dictionary

```
function append_dictionary(image)
global dir;
global dictionary;
character=image;
temp_character=clipallmargins(character);
temp_character=imresize(temp_character,[32,32]);
for count=1:32
    fprintf('%d',temp_character(count,:));
    fprintf('\n');
end
% load t_dict;
load(strcat(dir,'dictionary'));
load(strcat(dir,'nchars'));
load(strcat(dir,'vowel_count'));
fprintf('Add to Dictionary ?(1-Yes, 0-No):');
option = input('');
fprintf('%d\n',option);
if (option == 1)
    fprintf('appendimg...\n');
    %CHANGE THE FUNCTION FOR CALCULATING THE FEATURE VECTOR HERE ALSO.
    fv=int8(c_blockwise_DC_1AC(image));
    nchars=nchars+1;
    dictionary(nchars,:)=fv;
    save((strcat(dir,'nchars')),'nchars');
    save((strcat(dir,'dictionary')),'dictionary');
    fid=fopen(strcat(dir,'\TEST_blockwise_DC_1ACvowel.txt'),'a+');
    fseek(fid,-1,'eof');
    vowel_count=vowel_count+1;
    save((strcat(dir,'vowel_count')),'vowel_count');
    fprintf(fid,'(%d)\n',vowel_count);
    for count=1:32
        fprintf(fid,'%d',temp_character(count,:));
        fprintf(fid,'\n');
    end
```

```
    fprintf(fid,'%d',fv);
    fprintf(fid,'\n');
    fclose(fid);
else
    fprintf('not adding...\n');
end
return;
```

Bibliography

[MAC] Machine Learning for Intelligent Processing of Printed Documents Floriana Esposito, Donato Malerba, Francesca A. Lisi, Journal of Intelligent Information Systems, 14, 175-198, 2000.

[CSR] CISRO
http : *//vision.cmit.csiro.au/expertise/ocr*

[IYER] Optical Character Recognition System for Noisy Images in Devanagari Script, Parvati Iyer Abhipsita Singh Dr.S.Sanyal UDL Workshop on Optical Character Recognition with Workflow and Document Summarization (OCR & DS-2005)

[WIK] Wikipedia.org
http : *//en.wikipedia.org/wiki/Optical_character_recognition*

[SAC] OCR system for printed dravidian scripts using Uniform Sampling Techniques, M.Tech project, SSSIHL, Sachwani, 2002.

[EAS] OCR system for printed Dravidian scripts using Gradient Based Contour Encoding, M.Tech project, SSSIHL, Easwar, 2002.

[RAG] Deskwing techniques for font recognition for printed tamil texts, M.Tech project, SSSIHL, Raghavan, 2003.

[TRI] Feature extraction methods for OCR - a survey, Ovind Due Trier, Anil K Jain & Torfinn Taxt, Pattern Recognition, Vol 29, No.4, 1996.

[VER] Recent Achievements In Off-Line Handwriting Recognition Systems, B. Verma, M. Blumenstein & S. kulkarni, School of Information Technology, Griffith University - Gold Coast Campus PMB 50, Gold Coast Mail Centre, Qld 9726 Australia.

[TOU] J. T. Tou and R C. Gonzalez, Pattern recognition principles, addison-Wesley Publishing Company, Inc., 1974.

[PAT] A comparative study of neural network algorithms applied to Optical Character Recognition, P. Patrick van & Smagt.

[ALN] A FAX Reader for the Blind, Allen E. Milewski & Henry S. Baird , 24th Annual Asilomar Conference on Signals, Systems, and Computers, Pacific Grove, California, November 5-6, 1990.

[DUD] Pattern Classification, Second edition, Richard O. Duda, Peter E. Hart, David G. Stork, John Wiley & sons, 2004.

[BRU] Recent Work in the Document Image Decoding Group at Xerox PARC, Thomas M. Breuel and Kris Popat, Xerox Palo Alto Research Center 3333 Coyote Hill Road Palo Alto, CA 94304.

[ANL] Fundamentals of Digital Image Processing, Anil K. Jain, Prentice Hall International.

[GON] Digital Image Processing, Second Edition, Rafael C. Gonzalez and Richard E. Woods, Pearson Education, 2005.

[MYR] PZ Myers' Own Original, Cosmic, and Eccentric Analogy for How the Genome Works -OR- High Geekology,Wednesday, September 29, 2004
http : *//pharyngula.org/index/science/comments/high_geekology/*

[RIC] Sidebar: Fundamentals of Frequency Conversion, Richard A. Quinnell, TechOnLine Publication Date: Sep. 15, 2004
http : *//www.techonline.com/community/ed_resource/feature_article/*37054

[FRD] Discete cosine transform (DCT): What is it?
http://www.fh-friedberg.de/fachbereiche/e2/telekom-labor/zinke/mk/mpeg2beg/whatisit.htm

[AHM] Ahmed, N., T. Natarajan, and K. R. Rao. On image processing and a discrete cosine transform. IEEE Transactions on Computers C-23(1): 90-93, 1974.

[DHN] Object Tracking Based Robot Navigation using Particle Filters, Dhanu M. Poulose ,M.Tech project, SSSIHL, march 2006.

[TDL] Technology Development for Indian Languages.
http://www.tdil.mit.gov.in