

# OCR for Printed Telugu Documents

## M. Tech. Stage 2 Project Report

Submitted in partial fulfillment of the requirements  
for the degree of

## Master of Technology

by

**Arja Rajesh Babu**

**Roll No: 09305914**

under the guidance of

**Prof. J.Saketha Nath and Prof. Parag Chaudhuri**



Department of Computer Science and Engineering  
Indian Institute of Technology Bombay  
Mumbai

## Dissertation Approval

This dissertation entitled **OCR for Printed Telugu Documents** by **ARJA RAJESH BABU** is approved for the degree of **Master of Technology**.

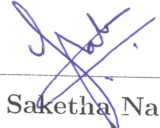
Examiner-I

  
Prof. Sunita Sarawagi

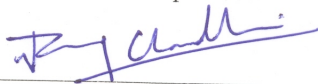
Examiner-II

  
Prof. Ganesh Ramakrishnan

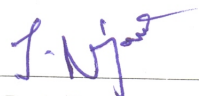
Supervisor

  
Prof. J. Saketha Nath

Co-Supervisor

  
Prof. Parag Chaudhuri

Chairman

  
Prof. J. Adinarayana

Date: 21/06/2012

Place: Mumbai

## Declaration

I declare that this written submission represents my ideas in my own words and where others ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misinterpreted or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will cause for disciplinary action by the institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

A. Rajesh

Signature

ARJA. RAJESH BARU

Name

09305914

Roll No.

Date: 21/06/2012

Place: Mumbai

## **Abstract**

Optical character recognition(OCR) is a well known process for converting text images to machine editable text format. Applications of OCR include preserving old/historical documents in electronic format, library cataloging, automatic reading for sorting of postal mail, bank cheques and forms as base for many applications in Natural Language Processing(NLP). OCR is a difficult problem to solve on real world data. In specific for Telugu language it is very difficult problem as single character formed by a single vowel or consonant or it can be compound character consists of combination of vowel and consonants. Drishti is the only one significant OCR available for Telugu language which will work on high resolution, good quality, noise free documents and specific input formats which is making use of OCR unrealistic. Our aim is to create a OCR which can eliminate all the constraints on input formats, quality which will work on real world documents.

We have created a basic end to end functioning OCR. We have used basic 0/1 features we obtained accuracy of 30% in stage1. We have observed the problems with OCR are font dependence, joint characters and broken characters. We have experimented with different types of features like gabor, wavelets, skeleton and circular features in stage2 for solving font dependence problem. We got improvement in accuracy with wavelet features, we got 48% accuracy. We have given our solution for joint character problem. We have observed distorted characters formed by binarization methods, we have studied some methods for solving binarization problem.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Stage1 Summary</b>	<b>3</b>
<b>3</b>	<b>Feature Extraction Methodologies</b>	<b>5</b>
3.1	Wavelet Features . . . . .	5
3.2	Gabor Features . . . . .	7
3.3	Circular zonal features . . . . .	9
3.4	Skelton features . . . . .	10
3.5	Comparisons . . . . .	11
3.5.1	Wavelet features . . . . .	11
3.5.2	Gabor features . . . . .	11
3.5.3	Circular features . . . . .	12
<b>4</b>	<b>Handling OCR Problems</b>	<b>13</b>
4.1	Joint character problem . . . . .	13
4.1.1	Improved formulation . . . . .	17
4.2	Distorted characters . . . . .	19
<b>5</b>	<b>Conclusion and Future Work</b>	<b>20</b>

# List of Figures

3.1	Down sampling of image by wavelet function [27] . . . . .	6
3.2	Applying gabor filter with different $\theta$ values [26] . . . . .	8
3.3	Zonal features used by Negi[22] . . . . .	9
3.4	Circular zonal features [1] . . . . .	9
3.5	Skelton structure of two different font characters [17] . . . . .	10
3.6	Skelton structures of characters generated using voronoiSkel . . . . .	10
3.7	Images with similar shape with different pixel density . . . . .	11
4.1	Joint character experiment input/output and basis images . . . . .	16
4.2	With $\lambda$ value 1 . . . . .	18
4.3	With $\lambda$ value 20 . . . . .	18

# List of Tables

3.1	Comparison of accuracies . . . . .	12
-----	------------------------------------	----

# Chapter 1

## Introduction

Optical character recognition(OCR) is a well known process for converting text images to machine editable text format. During past few decades significant research work is reported in the OCR area. In English there are many commercial OCR applications are available [7]. Apart from English there is significant amount of research have been done for languages like Chinese [33] and Japanese [20]. OCR gained so much research interest because of its potential applications in post offices, banks and defense organizations. Other applications involve reading aid for the blind, preserving old/historical documents in electronic format, library cataloging, automatic reading for sorting bank cheques and applications in natural language processing(NLP) area.

OCR implemented for English language can not be applied for Indian languages as the single character formation in the Indian language can be either simple character formed by single vowel or consonant or compound character formed by combination of the vowel and consonants. So in the Indian context OCR is very difficult problem compared to English. Recently, work has been done in the development of OCR systems for Indian languages. This includes work on recognition of Devanagari characters [2], Bengali characters [3], Kannada characters [1], Tamil characters [28] and Telugu characters [23].

Telugu is the second largest speaking language in India there 74 million native speakers of Telugu in India in 2001 [6]. First Telugu OCR developed by Deekshatulu and Rajasekaran in 1977 [32]. This OCR is a two-stage syntax-aided character recognition system which identifies 50 primitive features. Using the circular characteristic of the Telugu character new OCR system was proposed by Rao



and Ajitha in 1995 [29]. Sukhaswami developed a Telugu OCR in 1995 which uses neural networks for the recognition of the characters. Atul Negi developed a Telugu OCR using template matching method based on fringe distance measure for recognition in 2001 [23]. Pujari [27] developed a OCR in 2002 using wavelet multi resolution analysis for extracting features and associative memory model for the recognition. A multi-font OCR for Telugu developed by Vasntha Lakshmi and Patvardhan in 2002 using pixel gradient direction as the feature vector [17]. DRISHTI is a complete Optical Character Recognition system for Telugu language developed by the Resource Center for Indian Language Technology Solutions (RCILTS), at the University of Hyderabad in 2003 [9]. Dristhi is the significant Telugu OCR available till date which is having many constraints on resolution of the image, quality of the image and input/output formats of the image.

**Motivation:** The research in OCR area all other languages like English, Chinese and Japanese are far a head compared to the research done in Indian languages. In particular the research done in OCR for Telugu language is not significant. We want to develop an OCR which can eliminate all the constraints on input formats, quality which will work on real world documents. We want to digitalize the old/historical documents of Telugu language which are available in digital library of India [4].

Rest of the report is organized as follows. In Chapter 2 we will describe the summary of the stage1 work. In chapter 3 we will describe various feature extraction methods we studied, experimented for solving font dependency problem. In chapter 4 we will describe the proposed solutions of problems in OCR. In chapter 5 we will describe the conclusion and future work.

## Chapter 2

### Stage1 Summary

In this chapter we will describe the work done in stage1 in brief. In stage1 we have implemented basic OCR application for Telugu which can perform end to end operations like binarize image, correct the skew angle, finds the connected components, segment the lines, finds feature vector, recognizes the character and renders into text file using Unicode. We will describe the methodologies we used in each step in the following paragraphs.

We have used Java Imaging API for binarization of the image. First we will find the histogram of the image using the JAI class from Java Imaging API. After getting histogram we will find the Maximum Variance Threshold which will maximizes the ratio of the between-class variance to the within-class variance for each band [5]. We will make use of the threshold and using the JAI class we will binarize the image. We have used Hough's transformation for skew detection and correction. We have customized the existing implementation for the deskew available [10].

We have implemented connected component generation algorithm by checking the 8-way connectivity in the image. We have labeled each connected component with a different label which we will use for further processing. We formed data structure called cluster which contains all the points with same lable and their boundary positions.

We are working on the historical documents which will contain noise implicitly. Border noise is one major we will encounter most of the cases when we are dealing with distorted documents. We implemented our own method for removing the noise using one of the most prominent machine learning algorithm, Expectation Maximization (EM) algorithm. We experimented on different kind

of features of an connected component (CC) like length/breadth ratio, area of the CC, density of the dark pixels in CC and position of the CC. We got better results when we use length/breadth ratio, area and density of cluster as noise clusters will have comparatively large size and will have more density of black pixels compared to normal clusters. We have experimented with different clustering algorithms in machine learning like k-means clustering and EM algorithm with different number of clusters. We got better noise removal when we used EM algorithm with 2 clusters overall. So we are using EM algorithm with 2 cluster method with the above three features for our noise removal. We have used WEKA API [8] for EM algorithm.

We have implemented line segmentation by using Hidden Markov Model(HMM). We have observed that in Telugu document there is fixed pattern followed by the property of Telugu writing script. We can categorize the document into four different parts. First part with complete white pixels, second part with low density black pixels, third part with the high density of the black pixels and fourth part is the low density black pixel part. This pattern repeats for each line. With this observation we have mapped each part with a state in HMM. We created model with 4-state HMM with the density of the black pixels in each line in the document. We have used SVM Torch [15] HMM implementation for our line segmentation.

We have implemented a basic prototype of the OCR for Telugu. We have used basic feature vector 0/1 feature vector which will consider white pixel as value 0 and black pixel as 1. We have rescaled all the connected components to  $41 \times 41$  size and found 0/1 feature vector. We have prepared synthetic test data dataset with same size used euclidean distance measure for finding the closest label for the testing data.

We have experimented with Scale-invariant feature transform(SIFT) [19] features for finding the feature vectors. As we are working on the historical documents characters will not be complete there will be many broken characters because of which we did not got significant results with SIFT. We used 0/1 features for classification in stage1.

From the results of stage1 we have observed the major problems causing poor accuracy are font dependency, joint characters and broken characters present in the document. We tried solving the problems of font dependency and joint characters in stage2. We have tried different feature vectors for solving the font dependency problem in the documents. In chapter3 we will explain the different types feature vectors we tried for improving accuracy.

## Chapter 3

# Feature Extraction Methodologies

Feature extraction is important stage in OCR after preprocessing. Features extracted in this stage determines the accuracy of the OCR system. Ideal features should should give good accuracy across different fonts and different scales. Features can be classified into two categories structural features and frequency features [26]. Structural features include directional features [33], direction change features [24], skeleton features [16]. Structural features fail to give good accuracies in case of low resolution characters, broken or distorted characters. Frequency features are extracted using Fourier transform and wavelet transform which are robust with resolution of the document. In this chapter we will be explaining examples of the frequency features and structural features. We will give the comparison of the all feature methods at the end of the chapter.

### 3.1 Wavelet Features

Wavelet analysis can capture the invariant features of scripts [27]. Wavelet analysis down samples the image to capture the inherent directional features of the image. All the characters in the Telugu script are combination of circular or semi circular shapes of different sizes. This nature motivates to usage of wavelet analysis for Telugu script as it captures the directional features [27].

Wavelet analysis will encodes image into average image and three detailed images. Three detailed images will respectively encode the directional features in vertical, horizontal and diagonol directions. In the following image shows how wavelet function down samples image into four images. In the

following image  $f_{LL}$  represents the average image,  $f_{LH}$  represents horizontal image features,  $f_{HL}$  represents vertical image features and  $f_{HH}$  represents diagonal features.

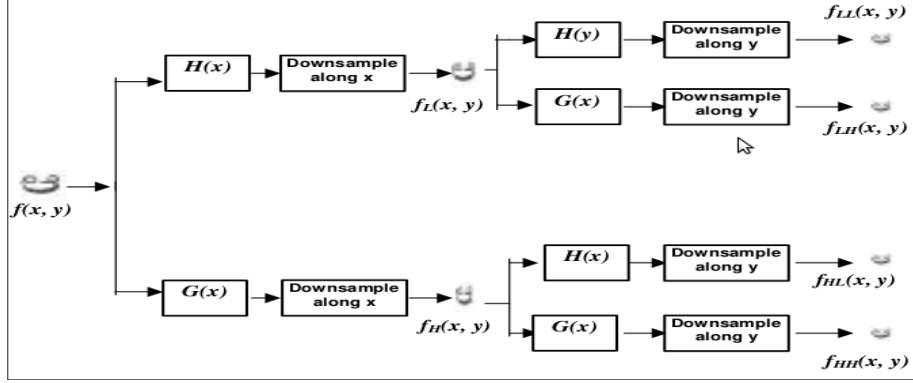


Figure 3.1: Down sampling of image by wavelet function [27]

Wavelet features depends on the wavelet basis function used. According to Pujari [27] not all wavelet basis functions captures the features for Telugu script. Pujari tried with two wavelet basis functions for extracting features for Telugu characters. Battle-Lemarie Filter [31] is proved as the best basis function useful for Telugu script by Pujari [27].

In our experiments we have normalized our template images to  $32 \times 32$  size. We have used the discrete wavelet transform(dwt2) [14] method of matlab. We need to provide low-pass filter, high-pass filter and image as an input for dwt2 function. We have used Battle-Lemarie Filter [31] as wavelet basis function. We have used lemarie [11] matlab function with number of coefficients as 3 and generated low-pass and high-pass filters. Output generated by dwt2 function contains three directional components mentioned above and one average image component. We have created our feature vector by concatenating the vectorized three directional image matrices and average image matrix.

We have observed wavelet features are giving better accuracy than 0/1 features which we used in stage1. The improvement in accuracy is achieved in wavelets is because they capture the directional changes as the features. Many confusing characters got misclassified in case of 0/1 features got classified correctly while using wavelet features. We will provide more comparisons in results section.

## 3.2 Gabor Features

Gabor filter performs a spatial frequency analysis on image [26]. It can extract oriental-dependent frequency contents as small as possible. Gabor filters have given good accuracy for Chinese characters [34]. In Indian language context gabor filters are used for Tamil script and gave good accuracy [30]. Gabor filter can extract features from the documents having low resolution [26]. Our goal is to implement OCR system independent of constraints like resolution. This motivated us to use the Gabor Filter for feature extraction.

Gabor features are extracted using by convoluting gabor filters on the input images. Two dimensional gabor filter looks as follows [26].

$$h(x, y) = g(x, y)e^{j\lambda(x\cos\theta+y\sin\theta)} \quad (3.1)$$

Where  $g(x, y)$  is Gaussian function given by:

$$g(x, y) = \frac{1}{\sqrt{2\pi\sigma_x\sigma_y}} e^{-\frac{((\frac{x}{\sigma_x})^2 + (\frac{y}{\sigma_y})^2)}{2}} \quad (3.2)$$

In the above equation  $\lambda$  is the wavelength of Gabor filter and  $\theta$  is the orientation angle of Gabor filter.  $\sigma_x, \sigma_y$  is the standard deviation of Gaussian along the  $x$ -direction and  $y$ -direction. If we set  $\sigma_x = \sigma_y = \sigma$ , we can rewrite above equation as follows

$$h(x, y) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x^2+y^2)}{2\sigma^2}} e^{j\lambda(x\cos\theta+y\sin\theta)} \quad (3.3)$$

Suppose  $u(x, y)$  as an image and  $I(x, y, \theta)$  is the response image obtained after convoluting above Gabor filter with orientation angle  $\theta$  on the input image as follows where  $M, N$  are dimensions of gabor filter.

$$I(x, y, \theta) = \sum_{x_1=x-\frac{M}{2}}^{x+\frac{M}{2}} \sum_{y_1=y-\frac{N}{2}}^{y+\frac{N}{2}} u(x_1, y_1) \cdot e^{-\frac{(x_1-x)^2+(y_1-y)^2}{2\sigma^2}} \cdot e^{j\lambda(\cos\theta(x-x_1)+\sin\theta(y_1-y))} \quad (3.4)$$

In our experiments we have used Gabor filter matlab code [12]. We normalized binary image to  $32 \times 32$  image then we applied gabor filters with different  $\theta$  values. Following image gives the steps we followed for extracting the gabor features.

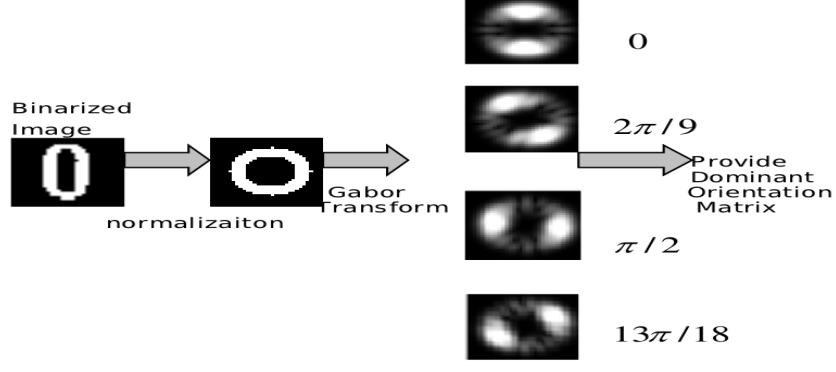


Figure 3.2: Applying gabor filter with different  $\theta$  values [26]

We have tuned the parameters in the above equation to get the gabor filter size  $11 \times 11$  suggested by Yannan [26]. There are many ways for extracting features once we got the gabor filtered images with different  $\theta$  values. In case of Tamil character recognition features are mean value and standard deviation calculated for each filtered images [30]. Yannan suggested dominant orientation matrix method for feature extraction [26] which gave good results on low resolution images so in our experiments we have implemented dominant orientation matrix method for feature extraction as our goal is to achieve the same.

In the above convolution formulation we will get different  $I(x, y, \theta)$  value for different  $\theta$  values. Suppose we have  $n$  different orientations and we numbered them from  $0, 1, 2, 3, \dots, n - 1$ . Dominant orientation matrix contains the values between  $0$  and  $n - 1$ . Each value of dominant orientation matrix  $DM(x, y)$  can be obtained as follows.

$$DM(x, y) = k \text{ where } I(x, y, \theta_k) = \max(I(x, y, \theta_k)) \forall k$$

After obtaining the dominant feature vector matrix as above we will reshape the matrix to vector for forming the feature vector for our experiments.

We have observed gabor features not giving good accuracies compared to wavelet features for our test data. As gabor filter capturing the spacial dependent frequency features it can not give good accuracies characters with different font. Gabor features are giving poor accuracies compared to 0/1 features. We will provide detailed comparison in the comparisons section.

### 3.3 Circular zonal features

Zonal features are extracted by dividing the image into different zones and extracting features from each zone. Zonal features have significant usage in character recognition. Lakshmi [17] divided each image into  $3 \times 3$  and extracted directional features from each zone. Concatenated features from all zones form the final feature vector. Negi [22] used zonal features for Telugu OCR. Following image is shows the zoning done by Negi [22]. In the previous two methods they have divided into image in square or rectangular zones.



Figure 3.3: Zonal features used by Negi[22]

Kannada writing script is similar to Telugu script both these languages have characters in circular shape. Therefore features which can extract the distribution of the ON pixels in the radial and the angular directions will be effective in capturing the shapes of the characters [1]. Circular zonal features gave good recognition accuracies for Kannada script [1]. This motivated us to use circular features for Telugu script. Following image shows how the sectors and tracks are drawn for extracting circular zonal features.

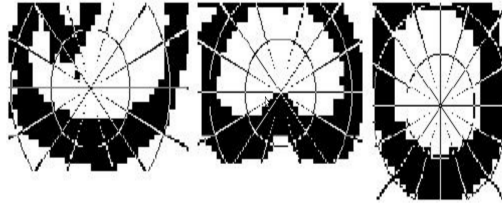


Figure 3.4: Circular zonal features [1]

In our experiments we have divided image into three tracks and each track will have six sectors. Image got divided into 18 circular zones. In our experiments we have used only density or count of ON pixels as the feature. Our feature vector contains the densities of 18 zones. Because we have



used basic density features we didn't get good accuracies in case of circular zonal features. We will give more details in comparisons section.

### 3.4 Skelton features

One of our main goal is to create font independent OCR. Creating a multi-font or font independent OCR is still a difficult problem, but human can recognize character which was printed in different fonts and written by different people. The main reason behind this our mind reads the underlying skeleton. This motivates us to use the skeleton structure of character instead of character itself for extracting features. Following image shows the skeleton structures of a character in two different fonts.



Figure 3.5: Skelton structure of two different font characters [17]

Skelton features we mean the features generated over the skeleton of the image. We have used previous three feature generation methods for extracting the final features over skeleton characters used for classification. For generating skeleton image we have used matlab method voronoiSkel [13]. voronoiSkel uses only the pixel on the boundary of the objects and therefore is very efficient for thick objects (efficiency scales with the object's radius, rather than its area). However, it might be sensitive to small defects. Sample skeleton structure formed using voronoiSkel are as follows.



Figure 3.6: Skelton structures of characters generated using voronoiSkel

## 3.5 Comparisons

### 3.5.1 Wavelet features

We have implemented wavelet feature extraction method with Battle-Lemarie Filter [31] and dwt2 matlab method. When we compared the accuracy with 0/1 feature vectors we got better accuracy with wavelet features. Wavelet extracts the directional features which are giving more accuracy. We have observed wavelet features classified characters which are having similar shape but different dark pixel density correctly. Following example image shows two characters KHA with same shape but with different pixel density.



Figure 3.7: Images with similar shape with different pixel density

Some confusion characters are classified correctly. For example character "NAA" is less confused with "VAA". Wavelet features are giving over all accuracy of **47.81**. We have tried applying wavelet features on the skeleton images created using the algorithm mentioned in skeleton features section. We observed that skeleton features caused the reduction in the accuracy. The reason can be the algorithm for skeletonization causing the loss of information for finding the directional features for wavelets. We got accuracy with wavelet features on skeleton images as **35.24**.

### 3.5.2 Gabor features

We have used Gabor filter matlab code [12] for gabor feature extraction. We have implemented a method for dominant dominant orientation matrix. We have observed Gabor features are giving less accuracy compared to wavelet features. Gabor features are spacial dependent frequency features they suffer font dependency problem. It performed got less accuracy compared to 0/1 features also. Gabor features got overall accuracy **35.09**. We have applied gabor features on skeleton images which performed worst compared to gabor features on normal images. We got accuracy of **26.23**.

### 3.5.3 Circular features

Circular features are extracted by dividing image into 3 tracks and 6 sectors. We have extracted 18 length feature vector consists of the density of dark pixels in each zone. We have observed circular zonal features with density are not performed well when compared to 0/1. This feature is highly font dependent. Little shift in character orientation may effect the features which causes further decrease in accuracy. Circular features gave poor accuracy of **26.90** over all.

We have applied circular zonal features on skeleton images. In case of circular features on normal images it is highly font dependent but with skeleton images it becomes font independent. We have observed significant increase in accuracy. We got accuracy as **41.93**.

Following table gives complete statistics of the all the types of feature vectors we have used. We have compared simple characters accuracy and over all accuracy.

Table 3.1: Comparison of accuracies

Type of features	Simple characters accuracy	Over all Accuracy
wavelet	38.96	47.81
gabor	25.66	35.09
circular	18.89	26.90
0/1	31.57	40.58
wavelet+skel	23.07	35.24
Gabor+ skel	18.36	26.23
Circular + skel	24.67	41.93

We have observed the time taken for classification with above types of feature vectors. We have observed all methodologies are taking nearly same time. When we compare the accuracies given by the above methods we can conclude Wavelet features are giving better accuracy over all compared to all other methods.

## Chapter 4

# Handling OCR Problems

Developing OCR which gives good accuracy is possible only when we solve all the problems which are causing the poor accuracy. We have seen the major problems for poor accuracy are font dependence, joint characters and distorted characters. If we take real world data it can have different fonts, different scales and noise embedded in the data. Multiple scale problem can be solved using normalization of character to a specified size. We tried solving multi font character recognition or font dependency problem by trying with different feature vector types in chapter3. In this chapter we will provide our methodology for solving joint character problem. We will also provide one method for solving font dependency problem. Distorted characters are formed by noise or bad quality of printer. We will provide observations on distorted character problem in this chapter.

### 4.1 Joint character problem

Joint characters are formed due to bad quality of printer. While processing a text image in OCR we will find connected components assuming each connected component represents a character or part of character. Some times due to bad printing some characters have dark pixels connecting because of which they form a single connected component. These joint character connected component will not find any matching template in the training templates which will results the poor accuracy.

By observing connected components having joint characters they will comparatively have more size than ordinary connected component(CC). As the solution to this problem we need to first sepa-

rate those connected components from normal connected components. Joint character component formed due to more than one normal character template. With this observation we can say instead of comparing these templates with training templates directly we need to find the templates whose combination gives us this connected component. In naive method, we need to find the existence of basis character at each position of the joint character image for that we need to find a co-efficient matrix which will tell us whether the basis image is present in that position. Joint character image is the image formed by the sum of all basis images with all corresponding positions. Following equation gives the naive formula.

$$\operatorname{argmin}_{\alpha}(X - \sum_j \sum_i B_i \times \alpha_{ij})^2 + \beta \sum_{i,j,u,v} |\alpha_{ij}(u,v)| \quad (4.1)$$

where  $X$  is input image,  $B$  is set of basis images,  $\alpha$  is co-efficient matrix and  $\beta$  trade-off parameter. For reconstructing total image we need to find all the co-efficient matrices corresponding to each image in the basis. The number of the matrices to be found are equal to the product of number of positions of the image and number of basis images. The number of variables are very high in this case. Computational time required for computing the matrices is also too high, storage space required is also high.

Convolution is similar operation which will do the cyclic multiplication on image. We want to find co-efficient matrix which will give the position of the basis image on the joint character image. Using convolution we can find the similar thing. Convolution can be computed very fast using Fast Fourier Transformation(FFT). Number of variables will be minimized. The number of variables required are equal to the number of basis images. We need one co-efficient matrix per image to find the positions of the basis image on joint character image.

Morten [21] is trying to solve the similar problem in which given image they try to find the basis, co-efficient matrix and channel mixing parameters in the context of Non-negative matrix factorization(NMF). Our problem is sub problem with some variations. According to Morten the given image is written as the sum of the basis images convoluted over the co-efficient matrix. The formulation is as follows.

$$J_c(x, y) \approx L_c(x, y) = \sum_d s_{c,d} \sum_{u,v} \alpha_d(u, v) B_d(x - u, y - v) \quad (4.2)$$

where  $L_c(x, y)$  is the reconstructed image,  $\alpha_d$  is the co-efficient matrix for basis image  $B_d$  and  $s_{c,d}$  is the channel parameter. In case of Morten basis images, channel parameters and co-efficient matrix are variables. They are learning all these three parameters while solving following least square objective.

$$Z(\alpha, B, S) = \frac{1}{2} \sum_{c,x,y} (X_c(x, y) - L_c(x, y))^2 + \beta \sum_{d,u,v} |\alpha_d(u, v)| \quad (4.3)$$

where  $\alpha$  is multi dimensional co-efficient matrix,  $B$  is multi dimensional matrix contains all basis images,  $S$  is the channel parameter matrix and  $\beta$  is trade off parameter. Above objective function has least square error function, a regularize over  $\alpha$  and  $\beta$  is trade-off parameter with three parameters  $\alpha, S$  and  $B$ . In our case we have fixed basis images and we have used uniformly distributed values for channel parameters and only parameter is  $\alpha$ .

Morten assumed all the basis images are of same size. In our case joint characters need not have all the characters in that from same size. We have included this constraint also in our code. The objective mentioned above minimized with respect to  $\alpha$  and multiplicative update method [18] is followed for updating the variable after each iteration. Following is the algorithm followed.

---

**Algorithm 1:** Shift Invariant Sparse Coding [21]

---

**Initialization.**  $t = 0, \alpha_{c,d}^0$  random uniform initialized

**repeat until convergence**

**begin**

**Update sparse code**

$$A_d(u, v) = \sum_c s_{c,d} \sum_{x,y} X_c(x, y) B_d(x - u, y - v),$$

$$B_d(u, v) = \sum_c s_{c,d} \sum_{x,y} L_c(x, y) B_d(x - u, y - v),$$

$$\alpha_d^{t+1}(u, v) \leftarrow \alpha_d^t(u, v) \left( \frac{A_d(u, v)}{B_d(u, v) + \lambda'} \right)^{\mu_\alpha}$$

**end**

---

Non-negativity condition needs to be ensured on co-efficient matrix as we are dealing with image data. Co-efficient matrix is called sparse code matrix as we are imposing the sparsity constraint on

the matrix  $\alpha$  in our objective. In current objective function  $L_1$  norm is applied over entire matrix  $\alpha$  which ensures the sparsity across all the co-efficient matrices.

We have experimented using various size basis images using above formulation. Our assumption is we are having huge training space containing various font and various scale basis images. With this assumption we are trying to fit the best basis image in the joint character image. We have created toy dataset and experimented on it. We have tried for several values for trade-off parameter. As we increased the value of  $\beta$  more than 3 we are getting output image containing characters with less resolution. If we take value between 0.01 and 1 we are getting nearly same result with some resolution changes. We found best value for our data with  $\beta$  as 0 when we take trade off between computational time for convergence and the output image. We have used 4 basis images of size  $16 \times 16$  and six basis images of size  $32 \times 32$ . We have created a test image which consists of 7 characters out of these 10. Following image shows the input image, output image generated with  $\beta$  value one and the 10 basis images.

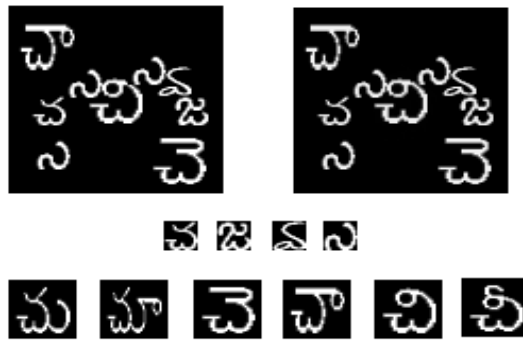


Figure 4.1: Joint character experiment input/output and basis images

We have applied above method on the joint characters present in our data set. We have observed there are 25 joint characters in case of our dataset. Number of joint characters are more when observed in case online news paper articles. The increase of the accuracy by solving them using above algorithm is less than 1%. Time taken to converge with real data is more. We have few observations from our dataset. Most frequent joint character in our dataset is variation of the character KSHA in Telugu. Using these observations we can implement some optimizations in the existing code. We can give weight to some characters and also we can add some more heuristics for extracting characters after finding co-efficient matrix.

### 4.1.1 Improved formulation

We have tried similar formulation with change of norm. We have observed due to  $L-1$  norm we are losing information in joint character formulation. If change the norm to  $L-2$  norm over co-efficient matrix of one basis image and  $L-1$  norm over all the co-efficient matrices then we are giving equal important to presence of character in a image. In this case character in a image can be formed by the parts of basis images. This will give some extent font independence. Basis character which is very near input image is identified as the corresponding image.

We have used following formulation for font independent problem.

$$Z(\alpha) = \frac{1}{2} \sum_{c,x,y} (X_c(x,y) - L_c(x,y))^2 + \beta \sum_d |(\sum_{u,v} \alpha_d(u,v))^2| \quad (4.4)$$

In case of joint character problem we have used multiplicative update method. Multiplicative method ensures the non-negativity constraint. We have used projected gradient method for updating the co-efficient matrix. We have used gradient decent method for updating co-efficient matrix. We projected it to non-negative space. We have used Armijo's rule for finding step length. Following is the algorithm.

---

**Algorithm 2:** Algorithm for Updating Co-efficient matrix

---

**Initialization.**  $t = 0, \alpha_{c,d}^0$  random uniform initialized

**repeat until convergence**

**begin**

**Update sparse code**

$$A_d(u,v) = \sum_c s_{c,d} \sum_{x,y} X_c(x,y) B_d(x-u, y-v),$$

$$B_d(u,v) = \sum_c s_{c,d} \sum_{x,y} L_c(x,y) B_d(x-u, y-v),$$

$$\alpha_d^{t+1}(u,v) \leftarrow \alpha_d^t(u,v) - \mu_\alpha * Z'(\alpha)$$

$$\alpha_d^{t+1}(u,v) \leftarrow 0 \text{ if } \alpha_d^{t+1}(u,v) < 0$$

**end**

$$\alpha_d^{t+1}(u,v) \leftarrow 0 \text{ if } \alpha_d^{t+1}(u,v) < threshold$$


---

In the above algorithm  $\mu_\alpha$  is step length find using Armijo's rule.  $Z'(\alpha)$  is the gradient of objective function at  $\alpha$  value. First we updated with gradient decent and in the next step we projected it into non-negative space. We have applied sparsity constraint at the end of the algorithm. We have



observed at  $\lambda$  value 0 we are getting good results. When we increase  $\lambda$  value we sparsity is increased and observed some characters in the input image are not formed in the output image. Following image shows the input, output and basis images for  $\lambda$  value 1.

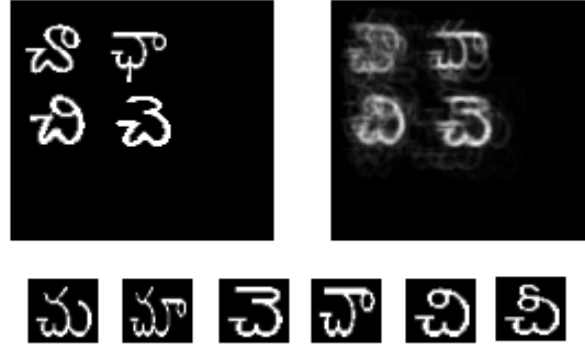


Figure 4.2: With  $\lambda$  value 1

We can observe fourth letter is which is different font character formed by the existing characters. By convoluting basis image with corresponding co-efficient matrix we can find the existence of any basis character in the final image. By finding connected components in this image we can find the positions of the basis character existence in the image.

In the following image we can see some characters not formed completely when we give more  $\lambda$  value. Following image shows input image, basis images and output image generated at  $\lambda$  value 20.



Figure 4.3: With  $\lambda$  value 20

We have observed the time taken by above method for convergence is high compared to other font independence methods. We can improve the solution with further heuristics.

## 4.2 Distorted characters

Distorted characters are major reason for poor accuracy of the OCR. Distorted characters can be formed due to the noise present in the document or bad printer. We have observed one more reason causing distorted characters which is binarization method. In this section we will explain the problems we observed due to binarization.

One of our main goal of the project is to develop OCR which can work on low resolution images. We tried working on e-paper image with our OCR which is having very less resolution. We observed the standard binarization method we used, Otsu's method [25] which takes global threshold for binarizing image is causing the distorted characters. When we used adaptive thresholding method which uses local thresholding value we observed number of broken or distorted characters are less. We have to experiment more in binarization methods.

## Chapter 5

# Conclusion and Future Work

We have implemented OCR system with 0/1 features in our first stage. We have observed the major problems causing the poor accuracy are font dependency, joint characters and distorted characters. In stage two we have experimented with different types of feature vectors for solving font dependency. We have extracted features using wavelet features, gabor features and circular zonal features. We have observed for Telugu characters we got better accuracy using wavelet feature vectors. We have observed circular zonal features with density are performing worst. We have repeated same feature extraction methods on skeleton images generated using voronoiSkel [13] method. We have observed decrease in accuracies in case of gabor and wavelet features but in case of circular zonal features we have observed very good improvement in accuracies. From stage2 results we can conclude wavelet features are giving better accuracy and are solving font dependency problems compared to other methods.

We have tried solving other problem in OCR, joint character problem. We have observed joint character problem can be solved by finding co-efficient matrices for with which we can convolute the basis image and their sum will give the reconstructed image. Our objective is to minimize reconstruction error and solve for co-efficient matrices. Paper by Morten is solving similar problem in case of NMF. We have used their code and implemented solution for our problem. We observed we can solve the joint character problem using this when we have huge training dataset. We have observed the number of joint characters are less in our dataset which is causing less improvement in our accuracy i.e less than 1%. We have observed the time taken for solving joint characters is

more. We have used similar formulation with change in the regularizer for improving the solution. We have observed slight improvement and this can also be used for font independence.

We observed other major problem for poor accuracy is distorted characters. Distorted characters caused due to bad printers or noise, which can not be solved completely using general font independence methods. But we observed one more reason for distorted characters in case of low resolution input images, binarization. We have used Otsu's method binarization which is also causing some broken characters. We have observed using adaptive thresholding methods can decrease number of broken/distorted characters.

Our future work involves improving accuracy by trying more types feature vectors. We also need to try using combinations of the different types of feature vectors. We need to improve the solutions for joint character problem in terms of execution time. We need to find some optimizations, by observing dataset we can say there are repetitions in joint characters which we can use for giving weights for some of the basis images and optimize the execution time. We can use additional heuristics for optimizing the solution. We have observed broken characters problem can be caused by binarization methods also. We need to experiment more on different binarization methods to decrease the number of broken characters. We have to apply some more filters which can remove remaining noise from image after our noise removal step.

# Bibliography

- [1] T. V. Ashwin and P. S. Sastry. A font and size independent ocr system for printed kannada documents using support vector machines. *Saadhana, Vol. 27, Part 1*, page 3558, 2002.
- [2] R.M.K. Bansal, V.; Sinha. A complete ocr for printed hindi text in devanagari script. *Document Analysis and Recognition, 2001. Proceedings. Sixth International Conference on Digital Object Identifier: 10.1109/ICDAR.2001.953898*, pages 800–804, 2001.
- [3] B. B. Chaudhuri and U. Pal. A complete printed bangla ocr system. *Pattern Recognition, vol.31*, pages 531–549, 1998.
- [4] Web Reference <http://202.41.82.144/>. Digital library of india.
- [5] Web Reference [http://download.oracle.com/docs/cd/E17802\\_01/products/products/java-media/jai/forDevelopers/jai\\_apidocs/javax/media/jai/Histogram.html](http://download.oracle.com/docs/cd/E17802_01/products/products/java-media/jai/forDevelopers/jai_apidocs/javax/media/jai/Histogram.html). Histogram java api.
- [6] Web Reference [http://en.wikipedia.org/wiki/Telugu\\_language](http://en.wikipedia.org/wiki/Telugu_language). Telugu.
- [7] Web Reference <http://http://www.abbyy.com/>. abbyy english ocr.
- [8] Web Reference <http://www.cs.waikato.ac.nz/ml/weka/>. Weka java api.
- [9] Web Reference [http://www.ildc.in/Telugu/htm/lin\\_ocr\\_spell.htm](http://www.ildc.in/Telugu/htm/lin_ocr_spell.htm). Drishti telugu ocr.
- [10] Web Reference <http://www.jdeskew.com>. Java deskew by hough trasformation.
- [11] Web Reference [http://www.math.rutgers.edu/courses/357/Uvi\\_Wave/wfilter/lemarie.m](http://www.math.rutgers.edu/courses/357/Uvi_Wave/wfilter/lemarie.m). lemarie matlab function.
- [12] Web Reference <http://www.mathworks.com/matlabcentral/fileexchange/23253>. Gabor filter matlab code.

- [13] Web Reference [http://www.mathworks.com/matlabcentral/fileexchange/27543-skeletonizationusing\\_voronoi](http://www.mathworks.com/matlabcentral/fileexchange/27543-skeletonizationusing_voronoi). Skeletonization using voronoi.
- [14] Web Reference <http://www.mathworks.in/help/toolbox/wavelet/ref/dwt2.html>. Dwt2 matlab function.
- [15] Web Reference <http://www.torch.ch/>. Svm torch.
- [16] JianMing Hu and Hong Yan. Structural primitive extraction and coding for handwritten numerical recognition. *Pattern Recognition*, vol.31, no.5, May 1998.
- [17] C. Vasantha Lakshmi and C. Patvardhan. A multi-font ocr system for printed telugu text. *Language Engineering Conference*, page 3036, 2002.
- [18] D. D. Lee and H. S. Seung. Algorithms for nonnegative matrix factorization. *NIPS*, page 556562, 2000.
- [19] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, pages 91–110, 2004.
- [20] Decerbo.M MacRostie.E, Natarajan.P and Prasad R. The bbn byblos japanese ocr system. *Pattern Recognition*, pages 650–653 Vol.2, 2004.
- [21] Mikkel N. Schmidt Morten Mrup and Lars K. Hansen. Shift invariant sparse coding of image and music data. *Journal of Machine Learning Research*, 2007.
- [22] Atul Negi and Chandra Kanth Chereddi. Candidate search and elimination approach for telugu ocr. *TENCON 2003. Conference on Convergent Technologies for Asia-Pacific Region*, pages 745 – 748 Vol.2, 2004.
- [23] Chakravarthy Bhagvati and.Krishna B Negi Atul. An ocr system for telugu. *Proc. Of 6th Int. Conf. on Document Analysis and Recognition IEEE Comp. Soc. Press, USA*, pages 1110–1114, 2001.
- [24] Masayoshi Okamoto and Kazuhiko Yamamoto. On-line handwriting character recognition using direction-change features that consider imaginary strokes. *Pattern Recognition vol.32, no.7*, July 1999.

- [25] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man and Cybernetics*, pages 62–66, 1979.
- [26] Zehong Yang Peifeng Hu, Yannan Zhao and Jiaqin Wang. Recognition of gray character using gabor filters. *Proceedings of the Fifth International Conference on In Information Fusion, Vol. 1*, pages 419–424, 2002.
- [27] C Dhanunjaya Naidu Pujari Arun K and B C Jinaga. An adaptive character recognizer for telugu scripts using multiresolution analysis and associative memory. *ICVGIP, Ahmedabad*, 2002.
- [28] T.Balachandar Abnikant Singh Markandey Singh Ritwaj Ratan R. Seethalakshmi., T.R. Sreeranjani and Sarvesh Kumar. Optical character recognition for printed tamil text using unicode. *Journal of Zhejiang University SCI 6A(11)*, pages 1297–1305, 2005.
- [29] P.V.S Rao and Ajitha T.M. Telugu script recognition-a feature based approach. *ICDAR*, pages 323–326, 1995.
- [30] L.Thaneshwaran S.Ponmathavan N.Valliappan R.Ramanathan, Arun.S.Nair and Dr. K.P.Soman. Robust feature extraction technique for optical character recognition. *International Conference on Advances in Computing, Control, and Telecommunication Technologies*, 2009.
- [31] S.Mallat. Multiresolution approximations and wavelet orthonormal bases of  $l_2(\mathbb{R})$ . *Transactions of the American Mathematical Society., vol3, No.1*, pages 69–87, 1989.
- [32] Rajasekaran S.N.S and Deekshatulu B.L. Comput. graphics image processing,6. *First International Conference on Engineering and Technology*, pages 335–360, 1977.
- [33] A.B. Wang and K.C. Fan. Optical recognition of handwritten chineses character by hierarchical radical matching method. *Pattern Recognition*, pages 15–35, 2001.
- [34] Xiaoqing Ding Xuewen Wang and Changsong Liu. Gabor filters-based feature extraction for character recognition. *Pattern Recognition 38*, pages 369–379, 2005.

## Acknowledgements

I would sincerely like to thank my guide, **Prof. J.Saketha Nath** for his motivating support throughout the year and the consistent directions that he has fed into my work. I would like to thank my co-guide, **Prof. Parag Chaudhuri** for his valuable suggestions and discussions. I would like to thank my friends Kolli Sarath, Balaji MMS and Ashok Kumar for their support and suggestions.