Journal of Zhejiang University SCIENCE
ISSN 1009-3095
http://www.zju.edu.cn/jzus
E-mail: jzus@zju.edu.cn

# Optical Character Recognition for printed Tamil text using Unicode

SEETHALAKSHMI R.[†], SREERANJANI T.R.[†], BALACHANDAR T.,
Abnikant Singh, Markandey Singh, Ritwaj Ratan, Sarvesh Kumar
(*Shanmugha Arts Science Technology and Research Academy, Thirumalaisamudram, Thanjavur, Tamil Nadu, India*)
[†]E-mail: rseetha123@cse.sastra.edu; trsree@yahoo.com

Received Aug. 5, 2005; revision accepted Sept. 10, 2005

**Abstract:** Optical Character Recognition (OCR) refers to the process of converting printed Tamil text documents into software translated Unicode Tamil Text. The printed documents available in the form of books, papers, magazines, etc. are scanned using standard scanners which produce an image of the scanned document. As part of the preprocessing phase the image file is checked for skewing. If the image is skewed, it is corrected by a simple rotation technique in the appropriate direction. Then the image is passed through a noise elimination phase and is binarized. The preprocessed image is segmented using an algorithm which decomposes the scanned text into paragraphs using special space detection technique and then the paragraphs into lines using vertical histograms, and lines into words using horizontal histograms, and words into character image glyphs using horizontal histograms. Each image glyph is comprised of 32×32 pixels. Thus a database of character image glyphs is created out of the segmentation phase. Then all the image glyphs are considered for recognition using Unicode mapping. Each image glyph is passed through various routines which extract the features of the glyph. The various features that are considered for classification are the character height, character width, the number of horizontal lines (long and short), the number of vertical lines (long and short), the horizontally oriented curves, the vertically oriented curves, the number of circles, number of slope lines, image centroid and special dots. The glyphs are now set ready for classification based on these features. The extracted features are passed to a Support Vector Machine (SVM) where the characters are classified by Supervised Learning Algorithm. These classes are mapped onto Unicode for recognition. Then the text is reconstructed using Unicode fonts.

**Key words:** OCR, Unicode, Features, Support Vector Machine (SVM), Artificial Neural Networks
**doi:**10.1631/jzus.2005.A1297 **Document code:** A **CLC number:** TP391

## INTRODUCTION

Optical Character Recognition (OCR) 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 Tamil documents from the ancient and old era to the latest, which helps in sharing the data through the Internet.

### Tamil language

Tamil is a South Indian language spoken widely in TamilNadu in India. Tamil has the longest unbroken literary tradition amongst the Dravidian languages. Tamil is inherited from Brahmi script. The earliest available text is the Tolkaappiyam, a work describing the language of the classical period. There are several other famous works in Tamil like Kambar Ramayanam and Silapathigaram but few support in Tamil which speaks about the greatness of the language. For example the Thirukural is translated into most other languages due to its richness in content. It is a collection of two sentence poems efficiently conveying and few other things in a hidden language called Slaydai in Tamil. Tamil has 12 vowels and 18 consonants. These are combined with each other to yield 216 composite characters and 1 special character (aayatha ezhuthu) counting to a total of (12+18+216+1) 247 characters.

**Vowels**

Vowels in Tamil are otherwise called UyirEzhuthu and are of two types short (Kuril) and long (Nedil).

**Consonants**

Consonants are classified into three classes with 6 in each class and are called Vallinam, Idaiyinam, and Mellinam.

**Tamil Unicode**

The Unicode Standard (http://www.unicode.org) is the Universal Character encoding scheme for written characters and text. It defines the uniform way of encoding multilingual text that enables the exchange of text data internationally and creates the foundation of global software. The Tamil Unicode range is U+0B80 to U+0BFF. The Unicode characters are comprised of 2 bytes in nature. For example, the Unicode for the character அ is 0B85; the Unicode for the character மீ is 0BAE+0BC0. The Unicode is designed for various other Tamil characters.

## OCR FUNCTIONAL BLOCK DIAGRAM

The block diagram of OCR consists of various states as shown in Fig.1. They are scanning phase, preprocessing, segmentation, feature extraction, classification (SVM, rule based, and ANN), Unicode mapping and recognition and output verification.

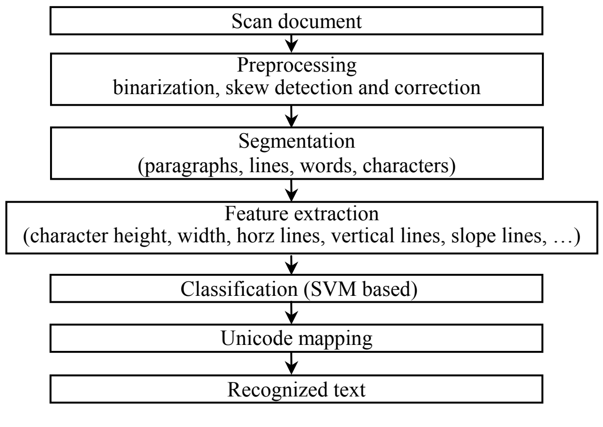

Scan document
Preprocessing
binarization, skew detection and correction
Segmentation
(paragraphs, lines, words, characters)
Feature extraction
(character height, width, horz lines, vertical lines, slope lines, …)
Classification (SVM based)
Unicode mapping
Recognized text

**Fig.1 Block diagram of OCR**

## OCR FUNCTIONS PHASE I

This phase includes the scanning state, preprocessing block, segmentation and feature extraction.

**Scanning the document**

A properly printed document is chosen for scanning. It is placed over the scanner. A scanner software is invoked which scans the document. The document is sent to a program that saves it in preferably TIF, JPG or GIF format, so that the image of the document can be obtained when needed. This is the first step in OCR (VijayaKumar, 2001). The size of the input image is as specified by the user and can be of any length but is inherently restricted by the scope of the vision and by the scanner software length.

**Preprocessing**

This is the first step in the processing of scanned image. The scanned image is checked for skewing. There are possibilities of image getting skewed with either left or right orientation. Here the image is first brightened and binarized. The function for skew detection checks for an angle of orientation between ±15 degrees and if detected then a simple image rotation is carried out till the lines match with the true horizontal axis, which produces a skew corrected image. Fig.2a shows the skewed image and skew corrected image histograms and Fig.2b shows skewed and skew corrected image. Skew correction is done by rotating the image around an angle $\theta$ (−2.0) as shown in Fig.2c (LTG, 2003; VijayaKumar, 2001).

**Segmentation**

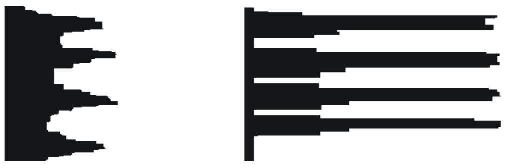
After pre-processing, the noise free image is passed to the segmentation phase, where the image is decomposed into individual characters. Fig.3 shows the image and various steps in segmentation.

Algorithm for segmentation:

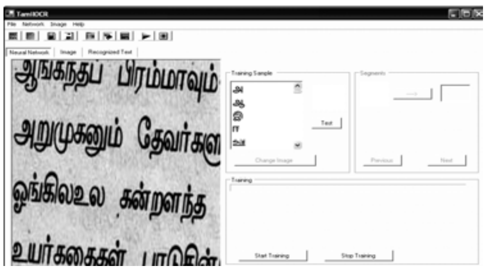
(1) The binarized image is checked for inter line spaces.

(2) If inter line spaces are detected then the image is segmented into sets of paragraphs across the interline gap.

(3) The lines in the paragraphs are scanned for horizontal space intersection with respect to the background. Histogram of the image is used to detect

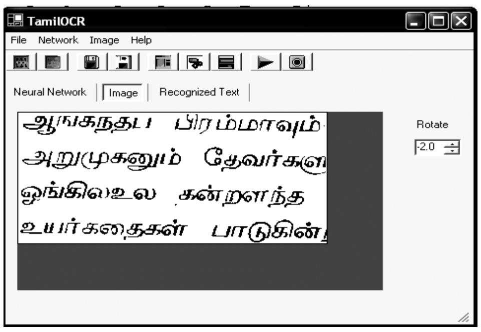


(a)



ஆங்கநதப் பிரம்மாவும்
அறுமுகனும் தேவர்களு
ஓங்கிஉலவ கன்றளந்த
உயர்கதைகள் பாடுகின்

(b)



Rotate: 2.0

ஆங்கநதப் பிரம்மாவும்
அறுமுகனும் தேவர்களு
ஓங்கிஉலவ கன்றளந்த
உயர்கதைகள் பாடுகின்

(c)

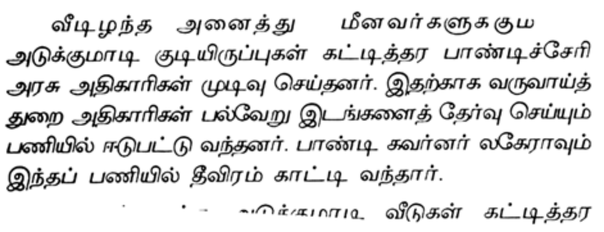
**Fig.2 (a) Histograms for skewed and skew corrected images; (b) Skewed image; (c) Skew corrected and preprocessed image**

the width of the horizontal lines. Then the lines are scanned vertically for vertical space intersection. Here histograms are used to detect the width of the words. Then the words are decomposed into characters using character width computation. Fig.3a shows the original image and Fig.3b shows the decomposition of image into single image character glyphs of size 32×32 (LTG, 2003; VijayaKumar, 2001).

## Feature extraction

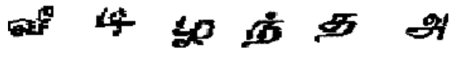
This follows the segmentation phase of OCR where the individual image glyph is considered and extracted for features.

First a character glyph is defined by the following attributes: (1) Height of the character; (2) Width of the character; (3) Numbers of horizontal lines present—short and long; (4) Numbers of vertical lines present—short and long; (5) Numbers of circles present; (6) Numbers of horizontally oriented arcs; (7) Numbers of vertically oriented arcs; (8) Centroid of the image; (9) Position of the various features; (10) Pixels in the various regions.



வீடிழந்த அனைத்து மீனவர்களுககுய
அடுக்குமாடி குடியிருப்புகள் கட்டித்தர பாண்டிச்சேரி
அரசு அதிகாரிகள் முடிவு செய்தனர். இதற்காக வருவாய்த்
துறை அதிகாரிகள் பல்வேறு இடங்களைத் தேர்வு செய்யும்
பணியில் ஈடுபட்டு வந்தனர். பாண்டி கவர்னர் லகேராவும்
இந்தப் பணியில் தீவிரம் காட்டி வந்தார்.
[illegible] வீடுகள் கட்டித்தர

(a)



(b)

**Fig.3 (a) Original text; (b) Character segmentation (32×32 glyph)**

The various feature extraction algorithms are as follows:

Detection of character height and character width: This is detected by simply scanning the image glyph and finding the boundary of the glyph in the horizontal and vertical directions.

அ Height=30 and Width=20

1. Horizontal line detection

Here a mask is run over the entire image glyph and thresholded which detects the horizontal line (Gonzalez *et al.*, 2004).

The mask is

| | | |
|---|---|---|
| −1 | −1 | −1 |
| 2 | 2 | 2 |
| −1 | −1 | −1 |

Image glyph: அ
Result:
No_of_Horiz=1.
Algorithm for Horizontal Line Detection:

```
c[i][j]=img[i−1][j−1]×(−1)+img[i−1][j]×(−1)
      +img[i−1][j+1]×(−1)+img[i][j−1]×2
      +img[i][j]×2+img[i][j+1]×2
      +img[i+1][j−1]×(−1)+img[i+1][j]×(−1)
      +img[i+1][j+1]×(−1)
If (c[i][j]>0)
   Increment q[i];
If (q[i]≥smallThreshold)
```

```
  Small horizontal line is detected
Else
  If (q[i]≥longThreshold)
    Long Horizontal line is detected;
```

2. Vertical line detection

Here a mask is applied over the entire range of pixels and thresholded which detects the vertical line (Gonzalez *et al.*, 2004).

The mask is
$$\boxed{\begin{array}{ccc} -1 & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & -1 \end{array}}$$

Image glyph: அ

Result:

The number of vertical lines is 1.

Algorithm for vertical line detection:

```
c[i][j]=img[i−1][j−1]×(−1)+img[i−1][j]×2
        +img[i−1][j+1]×(−1)+img[i][j−1]×(−1)
        +img[i][j]×2+img[i][j+1]×(−1)
        +img[i+1][j−1]×(−1)+img[i+1][j]×2
        +img[i+1][j+1]×(−1)
If (c[i][j]>0)
  Increment q[i];
If (q[i]≥smallThreshold)
  Short vertical line is detected
Else
  If (q[i]≥longThreshold)
    Long vertical line is detected;
```

3. Slope lines detection

The masks for the slope lines are as follows:

$$\begin{array}{cc} +45 & -45 \\ \boxed{\begin{array}{ccc} -1 & -1 & 2 \\ -1 & 2 & -1 \\ 2 & -1 & -1 \end{array}} & \boxed{\begin{array}{ccc} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{array}} \end{array}$$

To detect slope lines these masks are applied over the image and then thresholded suitably.

4. Detection of circles and arcs

Here a new mask is derived and operated on each and every pixel in the image glyph and thresholded which detects the circle. A 5×5 mask is taken. The arcs are detected using the circle detection algorithm and checked for the semi-circle and diameter (Gonzalez *et al.*, 2004).

The mask for circle is

$$\boxed{\begin{array}{ccccc} 2 & 2 & 2 & 2 & 2 \\ 2 & -1 & -1 & -1 & 2 \\ 2 & -1 & -1 & -1 & 2 \\ 2 & -1 & -1 & -1 & 2 \\ 2 & 2 & 2 & 2 & 2 \end{array}}$$

## OCR FUNCTIONS—PHASE II

The second phase of the OCR functions consists of classification and Unicode mapping and recognition strategies.

### Classification

Classification is done using the features extracted in the previous step, which corresponds to each character glyph. These features are analyzed using the set of rules and labelled as belonging to different classes. This classification is generalized such that it works for all the fonts' types (Rosenfeld and Kak, 1969).

1. A typical rule based classifier

If ((Numbers of short horizontal lines==0) and (No of long horizontal lines==1) and (Numbers of short vertical lines==0) and (Numbers of long vertical lines==1) and (Numbers of circles==1) and (Numbers of Horizontally oriented Arcs==1) and (Numbers of Vertically Oriented Arcs==1))

Then the character is அ.

The height of the character and the width of the character, various distance metrics are chosen as the candidate for classification when conflict occurs. Similarly the classification rules are written for other characters. This method is a generic one since it extracts the shape of the characters and need not be trained. When a new glyph is given to this classifier block it extracts the features and compares the features as per the rules and then recognizes the character and labels it.

2. Backpropagation based Classifier

Here a Backpropagation based Artificial Neural Network is chosen for classification because of its simplicity and ease of implementation. The architecture consists of three layers: Input, hidden and output. The features extracted are passed through various layers and the BPN algorithm is used to determine the output of each node, error back propagated and corrected. Thus after few iterations the error is reduced and the character glyphs are recognized.

3. Support Vector Machine (SVM) based Classifier

The next architecture chosen for classification, which in turn involves training and testing is Support Vector Machines (SVM). The use of Support Vector Machine (SVM) classifiers has gained immense popularity in recent years. SVMs have achieved excellent recognition results in various pattern recognition applications. Also in off-line optical character recognition (OCR) they have been shown to be comparable or even superior to the standard techniques like Bayesian classifiers or multilayer perceptions. SVMs are discriminative classifiers based on Vapnik's structural risk minimization principle. They can implement flexible decision boundaries in high dimensional feature spaces. The implicit regularization of the classifier's complexity avoids overfitting and mostly this leads to good generalizations. Some more properties are commonly seen as reasons for the success of SVMs in real-world problems. The optimality of the training result is guaranteed, fast training algorithms exist and little a-priori knowledge is required, i.e. only a labelled training set.

## Classification using SVM

Support Vector Machines are based on the concept of decision planes that define decision boundaries. A decision plane is one that separates a set of objects having different class memberships. A typical example is shown in Fig.4 where it is used to classify different types of character glyphs belonging to different Tamil fonts.

Support Vector Machine (SVM) is primarily a classier method that performs classification tasks by constructing hyperplanes in a multidimensional space that separates cases of different class labels. SVM supports both regression and classification tasks and can handle multiple continuous and categorical variables. To construct an optimal hyperplane, SVM employees an iterative training algorithm, which is used to minimize an error function. According to the form of the error function, SVM models can be classified into two distinct groups: Classification SVM Type 1 (also known as C-SVM classification); Classification SVM Type 2 (also known as nu-SVM classification).

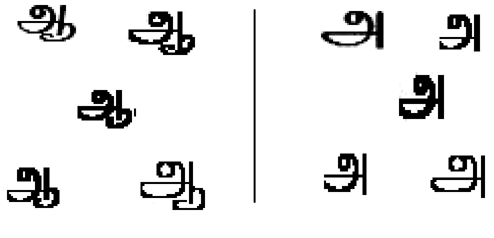


**Fig.4 Classification using SVM**

## Classification SVM Type 1

For this type of SVM, training involves the minimization of the error function:

$$\frac{1}{2}\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w}+C\sum_{i=1}^{N}\xi_i$$

subject to the constraints:

$$y_i(\boldsymbol{w}^{\mathrm{T}}\phi(x_i)+b)\geq 1-\xi_i \text{ and } \xi_i\geq 0,\ i=1,\ \ldots,\ N$$

where $C$ is the capacity constant, $\boldsymbol{w}$ is the vector of coefficients, $b$ a constant and $\xi_i$ are parameters for handling nonseparable data (inputs). The index $i$ label the $N$ training cases. Note that $y\in\pm 1$ represents the class labels and $x_i$ is the independent variables. The kernel $\phi$ is used to transform data from the input (independent) to the feature space. It should be noted that the larger the $C$, the more the error is penalized. Thus, $C$ should be chosen with care to avoid over fitting.

## Classification SVM Type 2

In contrast to Classification SVM Type 1, the Classification SVM Type 2 model minimizes the error function:

$$\frac{1}{2}\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w}-\nu\rho+\frac{1}{N}\sum_{i=1}^{N}\xi_i$$

subject to the constraints:

$$y_i(\boldsymbol{w}^{\mathrm{T}}\phi(x_i)+b)\geq\rho-\xi_i \text{ and } \xi_i\geq 0,\ i=1,\ \ldots,\ N;\ \rho\geq 0$$

## Kernel functions

There are a number of kernels that can be used in Support Vector Machines models. These include linear, polynomial, radial basis function (RBF) and sigmoid:

$$\phi=\begin{cases} x_i x_j & \text{Linear} \\ (\gamma x_i x_j+\text{coefficient})^{\text{degree}} & \text{Polynomial} \\ \exp(-\gamma\left|x_i-x_j\right|^2) & \text{RBF} \\ \tanh(\gamma x_i x_j+\text{coefficient}) & \text{Sigmoid} \end{cases}$$

The RBF is by far the most popular choice of kernel type used in Support Vector Machines. This is mainly because of their localized and finite responses across the entire range of the real *x*-axis.

SVM consists of a learning module (svm_learn) and a classification module (svm_classify). The training model takes the input file, target file and trains the network. In the classification model, the various class labels like class1, 2, and 3, …, 247 are given. Thus the SVM learns and produces correct labels of the classes.

SVM has the following points added to its credit, such as, learning is much faster especially for large training sets; working set selection based on steepest feasible descent; “shrinking” heuristics; improved caching; new solver for intermediate queries; let anyone set the size of the cache; simplified output format of svm_classify and data files may contain comments.

### Unicode mapping

The Unicode standard reflects the basic principle which emphasizes that each character code has a width of 16 bits. Unicode text is simple to parse and process, and Unicode characters have well defined semantics. Hence Unicode is chosen as the encoding scheme for the current work (Unicode, 2000). After classification the characters are recognized and a mapping table is created in which the unicodes for the corresponding characters are mapped. Table 1 shows one such rule based classifier based on Tamil Unicode.

### Character recognition

The scanned image is passed through various blocks of functions and finally compared with the recognition details from the mapping table from which corresponding unicodes are accessed (Table 1) and printed using standard Unicode fonts so that the OCR is achieved (Unicode, 2000).

## EXPERIMENTAL RESULTS

The OCR is implemented in Microsoft .NET using C#. Various experimental results are discussed below. Fig.5 of the segmentation phase output displays on the left side the original image and on the right side the segmented image. Fig.6 represents the feature extraction and rule based classification of vowels with Unicode and Fig.7 the feature extraction and rule based classification of consonants with Unicode. Figs.8, 9 and 10 represent the typical scenarios in the recognition of Tamil text using C# in the Microsoft .NET environment using Microsoft DLLs.

## OCR ANALYSIS

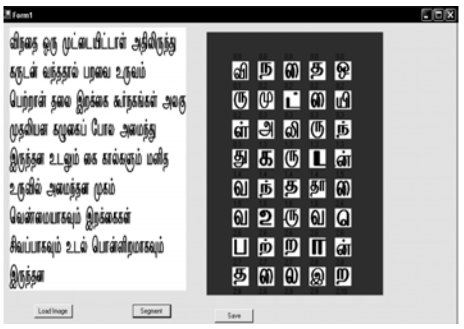
A complete analysis is done for the classification and recognition stages and various charts are depicted for clarity. A comparative study on various classifiers was also conducted.

### Analysis of classifiers

1. BPN based classifier output

An ANN based classifier used for classification is tested with minimum features (horizontal lines, vertical lines, circles and arcs). The target is the Unicode. The RMSE is shown to be tolerable.

Training samples:



விநதை ஒரு முட்டையிட்டாள் அதிலிருந்து
கருடன் வந்ததால் பறவை உருவம்
பெற்றான் தலை இறக்கை கூர்நகங்கள் அலகு
முதலியன கழுகைப் போல அமைந்து
இருந்தன உடலும் கை கால்களும் மனித
உருவில் அமைந்தன முகம்
வெண்மையாகவும் இறக்கைகள்
சிவப்பாகவும் உடல் பொன்னிறமாகவும்
இருந்தன

Fig.5 Output of segmentation

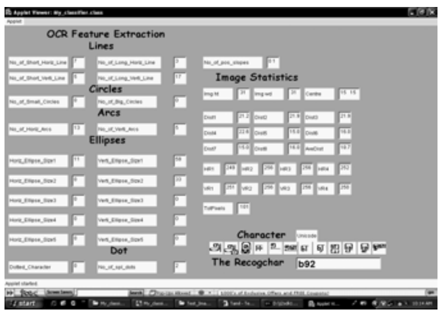


Fig.6 Output of feature extraction and recognition —vowels

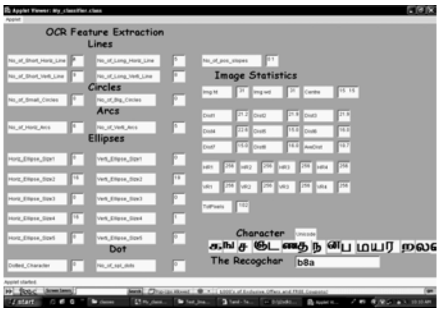


Fig.7 Output of feature extraction and recognition —consonants

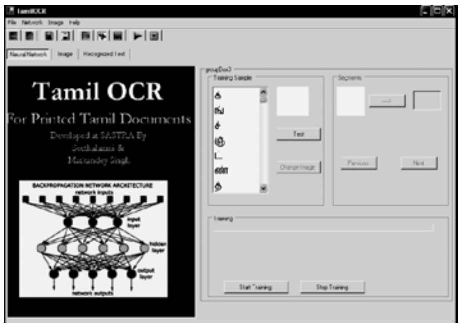

Tamil OCR
For Printed Tamil Documents
BACKPROPAGATION NETWORK ARCHITECTURE
Test
Start Training
Stop Training

Fig.8 OCR using C# in Microsoft .NET framework

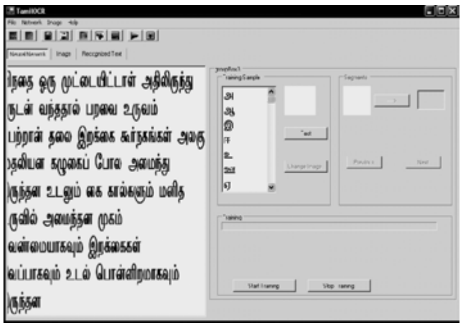


Fig.9 A scenario depicting recognition of Tamil text

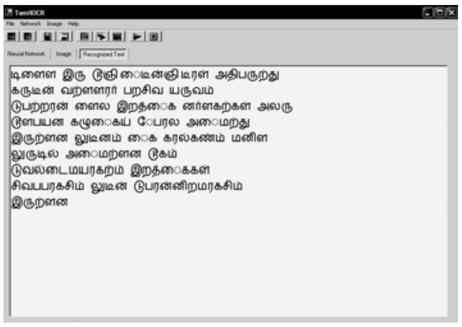


Fig.10 A scenario depicting recognition of Tamil text

Table 1 Sample mapping

| Features extracted | Classified character | Class label | Unicode |
|---|---|---|---|
| No of short horizontal lines==0 | அ | 1 | 0x0b85 |
| No of short vertical lines==0 | அ | 1 | |
| No of long horizontal lines==1 | அ | 1 | |
| No of vertically oriented arcs==1 | … | | |
| No of circles==1 | | | |
| No of horizontally oriented arcs==1 | | | |
| Height=23 | | | |
| Width=20 | | | |
| If input==target then the character is அ | | | |
| அ + one horizontal arc and one vertical arc | ஆ | 2 | 0x0b86 |
| | ஆ | 2 | |
| | ஆ | 2 | |
| | … | | |
| … Next character | | | |

1; 1; 1; 1; 1; 0; 0b85; 0;
1; 1; 2; 1; 1; 0; 0b87; 1;
0; 1; 1; 3; 2; 1; 0b88; 1;
1; 2; 0; 0; 0; 0; 0b89; 1;

0.0014195762512018836
0.001387192298036475
0.0013230704791920018
0.0015672780143078655

2. Output RMSE

All the patterns are recognized with a root mean square error of 0.001 (tolerance).

Comparison of various classifiers given in Table 2 and Fig.11 of the chart comparing strategies like SVM, BPN, Hybrid, Conventional Rule based Classifiers, shows that SVM gives consistently good performance. Five font types like Arial Unicode MS, Anjal (Nalinam), Amudham, Elango, Shree_tam fonts are considered for recognition. Table 3 shows the SVM learning results and Fig.12 the corresponding chart and Table 4 shows the SVM classification results and Fig.13 their chart.

## CONCLUSION

OCR is aimed at recognizing printed Tamil document. The input document is read preprocessed, feature extracted and recognized and the recognized text is displayed in a picture box. Thus the Tamil OCR is implemented using a C# Neural Network library. A complete tool bar is also provided for training, recognizing and editing options.

Tamil is an ancient language. There are millions and billions of books which are written by numerous well known authors. Maintaining and getting the contents from and to the books is very difficult. OCR eliminates the difficulty by making the data available in printed format. In a way OCR provides a paperless environment. OCR provides knowledge exchange by easier means. If the knowledgebase of rich Tamil contents is created, it can be accessed by people of varying category with ease and comfort. Still there are scholars who are interested in accessing the contents to look for knowledge.

OCR is currently used to maintain the history of students in universities. If OCR is available then processing and maintaining the students' records become easier. The students' forms can be directly

**Table 2 Comparison of classifiers**

| Type of classifier | Fonts | Error | Effi. |
|---|---|---|---|
| Typical Rule Based | 1 | 0 | 100% |
| Typical Rule Based | 2 or more | 0.2 | 80% |
| BPN | 1 | 0.00457 | 100% |
| BPN | 2 or more | 0.5000 | 50% |
| SVM | Single font | 0.001 | 100% |
| SVM | Multifont | 0.0001 | 100% |

**Table 3 SVM learning results**

| Number of fonts | Number of SVs | Empirical risks | L1 loss | Object value |
|---|---|---|---|---|
| 1 | 3 | 0.666667 | 7.12879E–5 | –0.405021 |
| 2 | 4 | 0.8 | 0.802707 | –1.30004 |
| 3 | 4 | 0.142857 | 0.000262048 | –0.405014 |
| 4 | 4 | 0.111111 | 0.000332225 | –0.405017 |
| 5 | 5 | 0.75 | 0.00199266 | –0.405015 |

**Table 4 SVM classify results**

| Number of fonts | Accuracy | Precision | Recall |
|---|---|---|---|
| 1 | 66.66667% | 50.00000% | 100.000% |
| 2 | 60.00000% | 100.00000% | 100.000% |
| 3 | 57.14286% | 50.00000% | 100.000% |
| 4 | 55.55556% | 50.00000% | 100.000% |
| 5 | 50.00000% | 45.45455% | 100.000% |

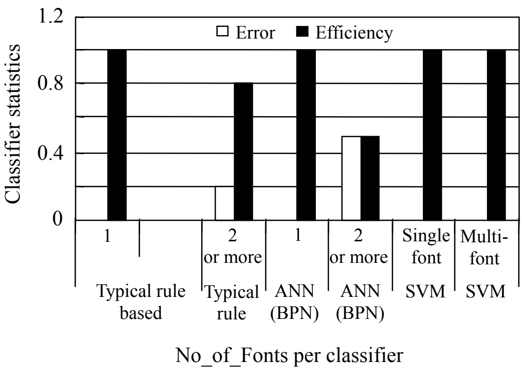


**Fig.11 Chart comparing No_of_Fonts and classifier efficiency**

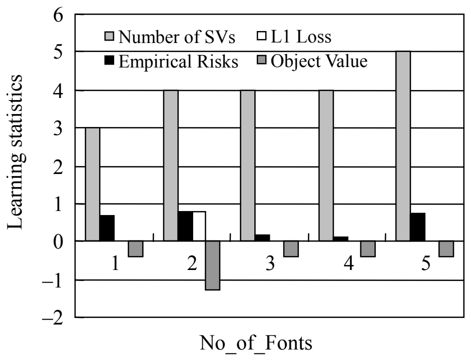


**Fig.12 SVM learning results**

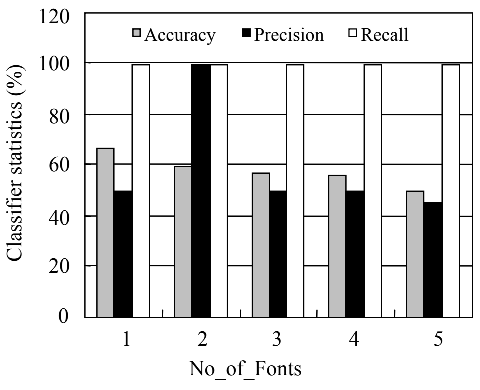


**Fig.13 Chart for classification results**

scanned, extracted for details and directly transformed into a Student Database.

OCR can also play a major role in the business environment. OCR reduces cost and effort by eliminating manual data entry, etc. If OCR is available, it becomes easier to extract and transform the data into business BASE and promote business without the need for large mobility (data, people). The increasing number of faxes and paper documents received by businesses often originate from the same suppliers or customers and have a format and layout that have not changed for some time. The data within these documents have to be manually interpreted and re-keyed into business applications as part of key business processes (e.g. Purchase Orders and Invoices into Accounting Systems for Accounts Receivables and Payables, students data etc.). The larger the volume of documents received, the greater the manual resource required entering the data into business applications. The scope for errors and delay to critical business processes also increases as volume increases, if it is handled manually. By scanning the documents to create TIFF image files and automatically routing electronic fax images to OCR, the errors, cost and delay of manual data entry can be avoided, as OCR can automatically extract data from the documents and format the data for onward delivery to other applications. Thus this paper discusses the various strategies and techniques involved in the recognition of Tamil text.

## ACKNOWLEDGEMENTS

We thank Microsoft Corporation of India (P) Ltd., for funding our project on Tamil OCR towards Digital Library Project at SASTRA. We would like to convey our gratitude to SERC-IISc, Bangalore for giving support to acquire this project. We also thank SASTRA for the infrastructure, other facilities and support provided towards the completion of this project.

## References

LTG (Language Technologies Group), 2003. Optical Character Recognition for Printed Kannada Text Documents. SERC, IISc Bangalore.
VijayaKumar, B., 2001. Machine Recognition of Printed Kannada Text. IISc Bangalore. The Unicode Standard Version 3.0, Addison Wesley.
Gonzalez, R.C., Woods, R.E., Eddins, S.L., 2004. Digital Image Processing Using MATLAB. PHI Pearson.
Unicode, 2000. The Unicode Standard Version 3.0. Addison Wesley.