Shirorekha Chopping Integrated Tesseract OCR Engine for Enhanced Hindi Language Recognition

Nitin Mishra
Dept. of Phy. & Comp. Sc.
Dayalbagh Edu. Institute
Dayalbagh, Agra, India

C. Patvardhan
Dept. of Electrical Engg.
Dayalbagh Edu. Institute
Dayalbagh, Agra, India

C. Vasantha Lakshmi
Dept. of Phy. & Comp. Sc.
Dayalbagh Edu. Institute
Dayalbagh, Agra, India

Sarika Singh
Dept. of Phy. & Comp. Sc.
Dayalbagh Edu. Institute
Dayalbagh, Agra, India

ABSTRACT
Tesseract OCR Engine is one of the most efficient open source OCR engines currently available. Recently, Tesseract OCR 3.01 is capable of recognizing Hindi language but still it needs some enhancement to improve the performance. The Hindi language recognition accuracy is quite low even for the printed text, as the conjunct character combinations of Hindi Language are not easily separable due to partial overlapping. The proposed approach solves this problem, so that Devanagari conjunct characters can easily be segmented and recognized using Tesseract OCR Engine. This paper presents a complete methodology to improve The Hindi Language Recognition accuracy. This paper also presents comparison with other Devanagari OCR engines available on the basis of recognition accuracy, processing time, font variations and database size.

General Terms
Pattern Recognition

Keywords
Tesseract, Hindi, OCR, Shirorekha Chopping, Character Segmentation

1. INTRODUCTION
Today, Tesseract is considered one of the most accurate open source OCR engines available. Tesseract OCR Engine was one of the best 3 engines in 1995 UNLV Accuracy Test. Between 1995 and 2006 however; there was little activity in Tesseract, until it was open sourced by HP and UNLV in 2005. It was again re-released to the open source community in August of 2006 by Google [1]. Tesseract has ability to train for newer language and scripts as well [2]. A complete overview of Tesseract OCR engine can be found in [3]. While Tesseract was originally developed for English, it has since been extended to recognize French, Italian, Catalan, Czech, Danish, Polish, Bulgarian, Russian, Greek, Korean, Spanish, Japanese, Dutch, Chinese, Indonesian, Swedish, German, Thai, Arabic, and Hindi etc. Training the Tesseract OCR Engine for Hindi language requires in-depth knowledge of Devnagari script in order to collect the character set [4]. Moreover, Tesseract OCR Engine does not just require training of the collected dataset but also to tackle the character segmentation and clubbing issues based on the script specific features [5] i.e. Shirorekha, maatra etc. Hindi language has enormous number of character combinations [6]; it is not a good technique to train all the possible combinations of Hindi characters. It is highly desirable to choose a Smart Database having all basic characters, half characters, and the minimal set of conjunct character combinations that may occur in some word and left out all unfavorable combinations. The segmentation issues related to Shirorekha based scripts are presented in [7]. Basically the proposed Hindi Language Database consists of basic vowels, consonants, extensions, special symbols, punctuation marks, English numerals, Devnagari numerals and minimal set of favorable vowel-consonant combinations, bi-consonant combinations and bi-consonant-vowel combinations. Tesseract based researches have shown robust results on Bangla and Kannada languages [8, 9] but still no efficient recognition results had been shown for Hindi language. This paper presents an improvement in printed Devanagari script recognition using Tesseract OCR Engine.

Table 1: General Vowels

अ a	आ ा aa/A	इ ि e/i	ई ी ee/ii	उ ु u	ऊ ू oo/uu
ए े e	ऐ ै ai	ओ ो o	औ ौ ou	अं ं aM	अः ः aH

Table 2: Other Vowels

ऋ r^^	ऌ l^^	ॐ AUM

Table 3: Consonants

क ka	ख kha	ग ga	घ gha	ङ nga
च cha	छ chha	ज ja	झ jha	ञ nja
ट Ta	ठ Tha	ड Da	ढ Dha	ण Na
त ta	थ tha	द da	ध dha	न na
प pa	फ Pha/fa	ब ba	भ bha	म ma
य ya	र ra	ल la	व va/wa	श Sha
ष shh	स sa	ह ha	क्ष ksh	त्र tra
ज्ञ jnja				

Table 4: Dot+Consonants (Extensions)

ऩ .na	ऱ .ra	ऴ .La	क़ .ka	ख़ .kha	ग़ .ga
ज़ .ja	ड़ .Da	ढ़ .Dha	फ़ .fa	य़ .ya	

Table 5: Special Symbols

Anusvara ं	Visarga ः	Chandra Bindu ँ	Chandra ॅ
Nukta ़	Virama ्	Udatta ॑	Anudatta ॒
Purna virama ।	Deergha virama ॥	Avagraha ऽ	Grave Accent ॓
Accute Accent ॔			

Table 6: Punctuation Marks and Other Symbols

“	?	;	%	*	/	(	)	\
=	{	}	[	]	,	-	:	!

Table 7: Numerals

०	१	२	३	४	५	६	७	८	९
0	1	2	3	4	5	6	7	8	9

2. METHODOLOGY

As Fig 1 shows, the proposed approach can be divided into two major components described below:

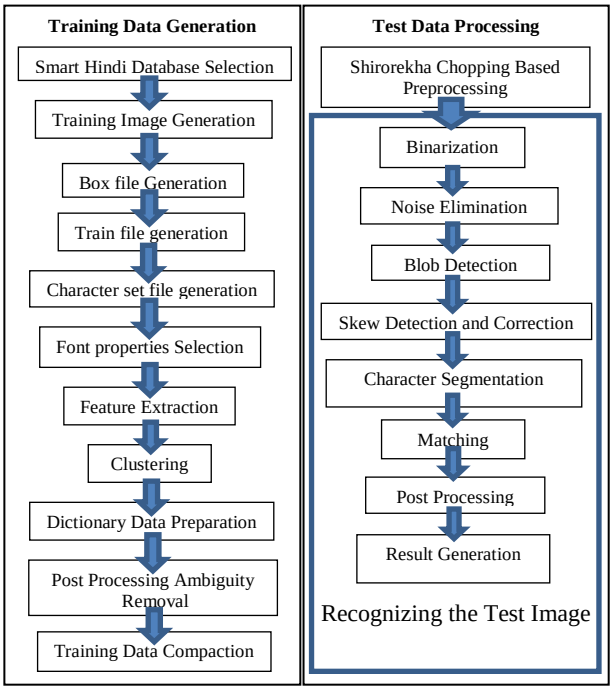

Fig 1: Block Level Diagram

2.1 Training Data Generation

The basic guideline to prepare training data has very clearly explained in [10], which is followed to prepare the customized training data. It has following phases described below:

2.1.1 Smart Hindi database selection

The Training database consists of 15 vowels, 36 consonants, 11 extensions, 13 special symbols, 18 punctuation marks and other symbols, 10 English numerals, 10 Devnagari numerals, a minimal set of 218 vowel-consonant combinations, 276 bi-consonant combinations and 179 bi-consonant-vowel combinations, providing a total of 786 character combinations of 18 pt. sized mangal font. The coarse classification of Hindi characters is presented in [11].

2.1.2 Training image generation

It involves the sufficiently spaced out single font specific text image creation. For each new font Tesseract OCR Engine suggests preparation of a new image file.

2.1.3 Box file generation

The information about the Bounding Boxes for all the characters present in the training image is generated for specifying Devanagari script components in the box file. The default generated Bounding boxes can easily be edited using box file editors i.e. cowboxer tool etc.

2.1.4 Train file generation

Box file editors also allow editing the corresponding Unicode characters against appropriate Bounding boxes.

2.1.5 Character set file generation

Character set file is required to specify the information like uppercase, lowercase, digits, punctuation marks etc. about the Unicode characters. Since Devanagari does not distinguish upper and lower case characters, only digits and punctuation marks have to be specified.

2.1.6 Font properties selection

Font properties like italic, bold, fixed, serif etc. are required to be specified before training the data. In this work only normal fonts have been considered.

2.1.7 Feature extraction

This phase extracts the features of the shape of characters from the Training Data Image.

2.1.8 Clustering

This phase clusters the character shape features into prototypes.

2.1.9 Dictionary data preparation

Tesseract may use up to 5 types of Dictionary files which are converted into Directed Acyclic Word Graph (DAWG) files.

2.1.10 Post processing ambiguity removal

Editing the unicharambigs file allows removing the intrinsic ambiguity between two similar looking characters or their combinations by using a substitution rule.

2.1.11 Training data compaction

Finally all the generated files are compacted into a single file.

OS used: **Ubuntu 10.04**
Tesseract OCR version used: **3.01**
Training image used: **hin.mangal.exp1.tif**

Commands used for Training Data Generation:

```
tesseract hin.mangal.exp1.tif hin.mangal.exp1 batch.nochop makebox
tesseract hin.mangal.exp1.tif hin.mangal.exp1 nobatch box.train
unicharset_extractor hin.mangal.exp1.box
cp unicharset hin.unicharset
echo mangal 0 0 0 0 0 > font_properties
mftraining –F font_properties –U hin.unicharset hin.mangal.exp1.tr
cntraining hin.mangal.exp1.tr
mv Microfeat hin.Microfeat
mv normproto hin.normproto
mv pffmtable hin.pffmtable
mv mfunicharset hin.mfunicharset
mv inttemp hin.inttemp
wordlist2dawg frequent_words_list hin.freq-dawg hin.unicharset
combine_tessdata hin.
```

Fig 2: Resources and Commands used

Fig 2 lists all the resources and commands used from the experimental point of view. The ***Mangal*** font was used in training image.

2.2 Test Data Processing

This component can be categorized basically in two sub components described below:

2.2.1 Shirorekha Chopping Algorithm

In the Preprocessing Phase, the horizontal and vertical histograms are generated for each line of the text identified in the test image. The Shirorekha of the Text in the image is chopped each time the distance between the bottom of the valley and the x-axis of corresponding vertical histogram goes below a threshold T, which is dependent on the font size. The motivation behind the Shirorekha Chopping is that by applying good segmentation techniques the performance of OCR can be increased [12].

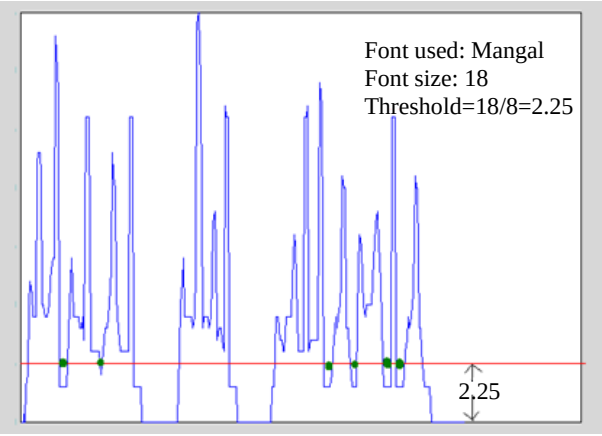

Fig 3: Shirorekha Chopping based on Font size specific threshold

The dots in Fig 3 represent the chopping points on the Shirorekha for corresponding word in the Test image.

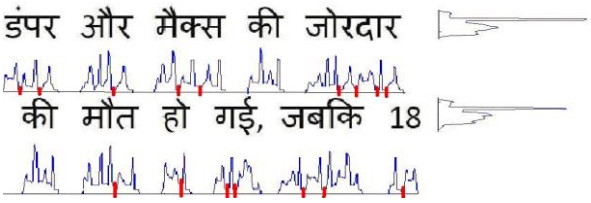

Fig 4: Shirorekha Chopping in Test Image

Fig 4 illustrates the Shirorekha Chopping. The small short lines highlight those valleys, at which distance between the bottom of the valley and the x-axis of corresponding vertical histogram goes below a threshold, T. Thus Shirorekha is chopped at these valleys. After the preprocessing gets completed, the Shirorekha Chopped test image as shown in Fig 5 is obtained.

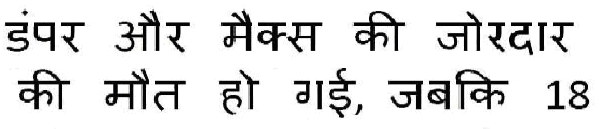

Fig 5: Shirorekha Chopped Test Image

The Shirorekha Chopped test image is now easily segmented using inbuilt segmentation technique of Tesseract OCR Engine as shown in Fig 6.

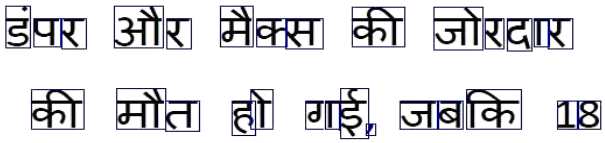

Fig 6: Shirorekha Chopping based Character Segmentation

2.2.2 Recognizing the Test Image

In this Phase, the preprocessed test image is recognized using Training Data.

Test image used: **test.tif**

Commands used for Test Data Processing:

```
tesseract test.tif result –l hin
```

3. EXPERIMENTAL RESULTS

The recognition accuracy, the processing time, and the size of database with preprocessing and font variations, was tested against Google's hin.traineddata [13] and Parichit's hin.traineddata [14].

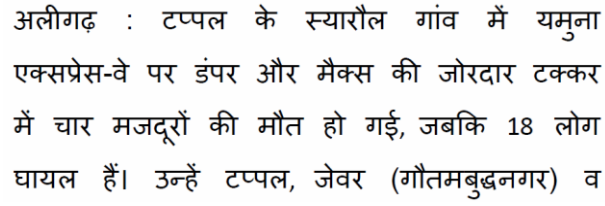
अलीगढ़ : टप्पल के स्यारौल गांव में यमुना एक्सप्रेस-वे पर डंपर और मैक्स की जोरदार टक्कर में चार मजदूरों की मौत हो गई, जबकि 18 लोग घायल हैं। उन्हें टप्पल, जेवर (गौतमबुद्धनगर) व

Fig 7: Test image

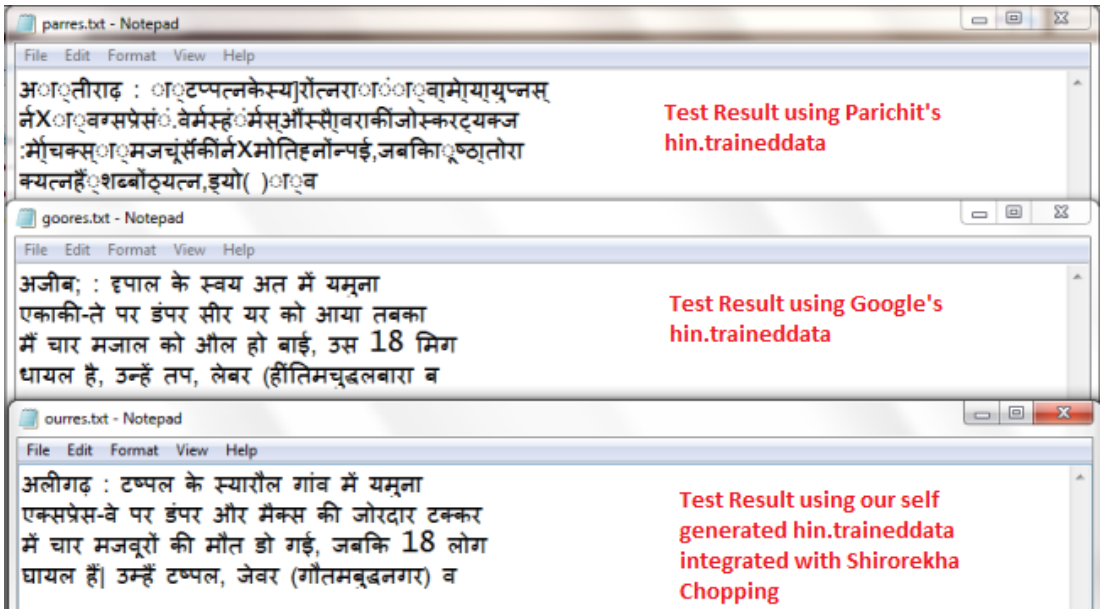

Fig 8: Experimental Results Comparison

The test image sample taken is shown in Fig 7. The Test Results can be compared by Fig 8. After a number of tests, the final results were obtained, which are described below:

Table 8: Font Variation Tolerance Comparison

	Recognition rate with Mangal as Testing font	Recognition rate with Krutidev as Testing font
Google's hin.traineddata	**45.6 %**	**44.8 %**
Parichit's hin.traineddata	**23.4 %**	**21.2 %**
Proposed hin.traineddata	**94.9 %**	**86.9 %**

Table 9: Average Recognition Rate Comparison

	Average Recognition Rate	Preprocessing used on Test Image
Google's hin.traineddata	**45.2 %**	**No preprocessing**
Parichit's hin.traineddata	**22.3 %**	**No preprocessing**
Proposed hin.traineddata	**90.9 %**	**Shirorekha Chopping**

Table 10: Processing Time Comparison

	Processing Time	Total Characters in Test Image
Google's hin.traineddata	**2000 ms**	**94**
Parichit's hin.traineddata	**1500 ms**	**94**
Proposed hin.traineddata	**1000 ms**	**94**

Table 11: Training Data Size Comparison

	Training Data size	Training font
Google's hin.traineddata	**13.8 MB**	**-**
Parichit's hin.traineddata	**13.1 MB**	**-**
Proposed hin.traineddata	**7.5 MB**	**Mangal**

4. CONCLUSIONS

There is a significant improvement in the recognition rate, processing time and the size of training database after integrating Shirorekha Chopping with Tesseract OCR Engine. Table 8 shows the higher accuracy for testing font being same as that of training font but lower accuracy for testing font being different from the training font, but still the font variation tolerance is quite better than existing ones. Table 9 shows the average recognition rate is quite enhanced using Shirorekha Chopping. The proposed Shirorekha chopping based preprocessing approach does not just improve the recognition rate but also allows training only two or more touching conjunct characters along with basic characters and isolated half characters. The single touching conjunct characters may be left out as these conjunct characters can easily be segmented using Shirorekha Chopping into those basic components that were trained. This leads to the generation of comparatively smaller training database (Table 11). The proposed Approach runs faster than that of Google and Parichit (Table 10). The extension to multiple fonts is being done, from the perspective of Future scope.

5. REFERENCES

[1] Google code : http://googlecode.blogspot.com/2006/08/ announcing-tesseract-ocr.html (last accessed 8 January, 2012)

[2] http://code.google.com/p/tesseract-ocr/ (last accessed 8 January, 2012)

[3] Smith, R. "An Overview of the Tesseract OCR" in proc. ICDAR 2007, Curitiba, Paraná, Brazil.

[4] Bansal, V. and Sinha, R.M.K. "A Complete OCR for Printed Hindi Text in Devnagari Script", Sixth International Conference on Document Analysis and Recognition, IEEE Publication, Seatle USA, 2001, Page(s):800-804.

[5] Jindal, M.K., Sharma, R.K., lehal, G.S. "A Study of Different Kinds of Degradation in Printed Gurmukhi Script", Proceedings of the International Conference on Computing: Theory and Applications (ICCTA'07),2007.

[6] Yadav, D., Sharma, A.K. and Gupta, J.P. Optical character recognition for printed Hindi text in Devanagari using soft-computing technique, *IASTED International Multi-Conference: Artificial Intelligence and Applications, Innsbruck, Austria*, 2007, pp. 102-107

[7] Chaudhuri, B. B. and Pal, U. "An OCR System to Read Two Indian Language Scripts: Bangla and Devnagari (Hindi)", Proc. of 4th ICDAR vol.2, Ulm, Germany, 1997, Page(s): 1011 -1015

[8] Hasnat, A., Chowdhury, M. and Khan, M. "Integrating Bangla script recognition support in Tesseract OCR", Proc. of the Conference on Language and Technology 2009 (CLT09), Lahore, Pakistan, 2009.

[9] Pal, U., Chaudhuri, B. B. "Indian Script Character recognition: A survey", *Pattern Recognition*, vol. 37, pp. 1887-1899, 2004..

[10] tesseract-ocr An OCR Engine that was developed at HP Labs between 1985 and 1995... and now at Google. Available at: http://code.google.com/p/tesseract-ocr/wiki/TrainingTesseract3/ (last accessed 8 January, 2012)

[11] Agrawal, P., Hanmandlu, M. and Lall, B., "Coarse Classification of Handwritten Hindi Characters", International Journal of Advanced Science and Technology,Vol. 10, September, 2009.

[12] Saba, T., Sulong, G. and Rehman, A. "A Survey on Methods and Strategies on Touched Characters Segmentation", International Journal of Research and Reviews in Computer Science (IJRRCS) Vol. 1, No. 2, June 2010.

[13] tesseract-ocr, available at: http://code.google.com/p/ tesseract-ocr/downloads/detail?name=tesseract-ocr-3.01 .hin.tar.gz&can=2&q= (last accessed 8 January, 2012)

[14] parichit The best open source OCR for Indian Languages...yet, available at: http://code.google.com/p/parichit/downloads/detail?name =hin.traineddata (last accessed 8 January, 2012)