Lipi Gnani - A Versatile OCR for Documents in any Language Printed in Kannada Script

H. R. SHIVA KUMAR, Indian Institute of Science
A. G. RAMAKRISHNAN, Indian Institute of Science

A Kannada OCR, named Lipi Gnani, has been designed and developed from scratch, with the motivation of it being able to convert printed text or poetry in Kannada script, without any restriction on vocabulary. The training and test sets have been collected from over 35 books published between the period 1970 to 2002, and this includes books written in Halegannada and pages containing Sanskrit slokas written in Kannada script. The coverage of the OCR is nearly complete in the sense that it recognizes all the punctuation marks, special symbols, Indo-Arabic and Kannada numerals. Several minor and major original contributions have been done in developing this OCR at the different processing stages such as binarization, character segmentation, recognition and Unicode mapping. This has created a Kannada OCR that performs as good as, and in some cases, better than the Google's Tesseract OCR, as shown by the results. To the knowledge of the authors, this is the maiden report of a complete Kannada OCR, handling all the issues involved. Currently, there is no dictionary based postprocessing, and the obtained results are due solely to the recognition process. Four benchmark test databases containing scanned pages from books in Kannada, Sanskrit, Konkani and Tulu languages, but all of them printed in Kannada script, have been created. The word level recognition accuracy of Lipi Gnani is 5.3% higher on the Kannada dataset than that of Google's Tesseract OCR, 8.5% higher on the Sanskrit dataset, and 23.4% higher on the datasets of Konkani and Tulu.

CCS Concepts: • **Applied computing** → **Optical character recognition**; *Document analysis*;

Additional Key Words and Phrases: Kannada, Sanskrit, Tulu, Konkani, Segmentation, OCR, Halegannada, SVM

ACM Reference format:
H. R. Shiva Kumar and A. G. Ramakrishnan. 2020. Lipi Gnani - A Versatile OCR for Documents in any Language Printed in Kannada Script. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.* 1, 1, Article 1 (February 2020), 24 pages.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The OCR for printed text can be used to digitize old, classical books, stories, educational and research treatise and convert them to searchable text. This will make these classical and educative literature immediately available around the world. Good recognition technology for printed text can also make school, college and other books readily available for the

2375-4699/2020/02-ART1 $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

blind. Digitization of historical collections of newspapers and magazines will make a wealth of information readily available to the new generation. In financial technology (FinTech) industry, automated extraction of key fields of importance from documents such as agreements, mortgage and sale documents can drastically reduce the processing cost and time, thereby creating an enhanced customer experience.

While so much work has been reported on Indic OCRs in the past two decades, the recognition accuracies have not reached the maturity levels needed for exploitation in real-life applications. Motivated by the passion to enable the visually challenged people to access any printed material in Indian languages, we have created OCR and text-to-speech systems for two of the South Indian languages, Kannada and Tamil, which are already being used by hundreds of blind individuals and numerous institutions in Karnataka and Tamil Nadu.

This paper describes the design of our Kannada OCR called Lipi Gnani. Section 1.1 gives a brief survey of the work reported on Indic OCRs, with particular reference to Kannada. Section 2 gives an overview of Kannada script from an OCR perspective. Section 3 gives the details of development of the various modules of the OCR. Sections 4 and 5 give the results of performance evaluation on the open benchmarking datasets and compare them with those of Google's Tesseract OCR.

1.1 Literature Survey

The first contribution in the development of OCR technology for Indic languages is by Rajasekaran and Deekshatulu [Rajasekaran and Deekshatulu 1977] for Telugu in 1977! Then, in 1979, Sinha and Mahabala [Sinha and Mahabala 1979] reported an OCR for Devanagari. In 1991, Chaudhury et al. reported work on Telugu [Chaudhuri et al. 1991] and then in 1994, on Bangla [Pal and Chaudhuri 1994]. This was shortly followed by Nagabhushan for Kannada [Nagabhushan and Pai 1999], Lehal for Gurmukhi [Lehal and Singh 2000], and Veena and Sinha for Devanagari [Bansal and Sinha 2000, 2002]. In 2000, Pati and Ramakrishnan reported the first OCR work on Odiya [Pati and Ramakrishnan 2000; Pati et al. 2000]. In 2001, Negi and Chakravarthy worked on Telugu [Negi et al. 2001]. Then, Sadhana had a special issue in 2002, where there were articles on the overview of document analysis [Kasturi et al. 2002], Kannada OCR [Ashwin and Sastry 2002] leveraging support vector machine (SVM), post-processing of Gurmukhi OCR [Lehal and Singh 2002] and also identification of Tamil and Roman scripts at the level of words [Dhanya et al. 2002]. A Tamil OCR was also reported by two different groups [Aparna and Ramakrishnan 2002; Kokku and Chakravarthy 2009]. BBN reported a Hindi OCR in 2005 [Natarajan et al. 2005], employing hidden Markov model (HMM). A lot of work has been carried out by Manmatha related to historical documents [Rath and Manmatha 2007]. There was an update on most of the above works in the book edited by Govindaraju and Setlur [Govindaraju and Setlur 2009], and a Malayalam OCR was also reported [Neeba et al. 2009]. [Krishnan et al. 2014; Mathew et al. 2016] made use of deep neural networks for the first time for Indic OCRs.

The above approaches to Indic OCRs could generally be grouped into three generations. The first generation approaches made use of structural features, geometric moments, etc. with basic classifiers such as nearest neighbour (NN), k-NN and decision trees. The second generation techniques employed transform features (discrete cosine transform, Karhunen-Loeve transform, discrete wavelet transform (DWT), etc.) with improved classifiers such as SVM and HMM. The third generation methods perform automatic feature extraction using convolutional neural networks (CNN) and classify using recurrent neural networks (RNN). Our OCR improves on the second generation approaches to Indic OCRs.

There are a number of reports in the literature on Kannada OCR [Ashwin and Sastry 2002; Nagabhushan and Pai 1999; Vijay Kumar and Ramakrishnan 2002, 2004]. However, most of these reports deal only with Kannada characters and none of them deal with the recognition of numerals, punctuation and other symbols normally present in any printed document. This is true also for the other papers surveyed above. Thus, to the knowledge of the authors, all of the above research work in Indic language OCRs involve theoretical contributions only and none of them have actually been used to digitize real-life printed matter. Recently, Mathew et al. [Mathew et al. 2016] reported the use of RNNs with bidirectional long-short-term memory (LSTM) cells in the hidden layers and connectionist temporal classification in the output layer. The dataset used by the authors has 5000 printed pages for many languages, and for Kannada, the authors report the performance of 5.6% character error rate on 3500 pages. However, that database is not publicly available.

1.2 Contributions of this Paper

- A realistic Kannada OCR has been designed and developed from scratch, which can actually recognize any old, printed page in Kannada, with a very good performance. It recognizes Indo-Arabic and Kannada numerals, Halegannada characters, as well as all the frequently used punctuation marks.
- A rich collection of benchmarking test datasets have been created, with annotated ground truth for Kannada, Konkani, Tulu and Sanskrit languages, all printed using Kannada script, and have been made openly available. Thus, researchers working on this topic can now compare their performance on standardized datasets.
- Results of our Lipi Gnani OCR on the above public datasets of Kannada, Tulu, Konkani and Sanskrit languages have been compared with those of Google's Tesseract OCR. These results will form reference results for future research on Kannada OCR.
- Unicode generation in Kannada is involved, and occasionally one needs to deal with a sequence of four to five graphemes. In some cases, the sequence of Unicodes follows an order different from the order in which the corresponding symbols occur in the text. We have come out with systematic rules and flow-charts for Unicode generation from any sequence of recognized Kannada primitives.

2 KANNADA RECOGNITION UNITS

2.1 Kannada Aksharas

Table 1 lists all the vowels (V), anusvara used for representing pure nasal consonants, visarga, consonants (C), and Kannada numerals. Table 2 lists the modifiers associated with each of the vowels, which normally are connected to any of the consonants, to result in the corresponding C-V combination occurring as a single connected component. As an example, all the C-V combinations of the consonant ಕ /ka/ are listed.

2.2 Ottus and OCR Challenges Posed by Them

In Kannada, two or three consonants can combine together with a vowel and form a compound character. In such a case, the second and the third consonants become consonant conjuncts, called ottus, and appear at the bottom or right bottom of the first consonant. The symbols present in the main line are referred to as the base symbols.

For example, in the word ಅಕ್ಕ /akka/, the symbol ◌್ಕ /ka_ottu/ is the ottu form of ಕ /ka/. Similarly, in the word ರಾಷ್ಟ್ರ /raassttra/, the symbols ◌್ಟ /tta_ottu/ and ◌್ರ /ra_ottu/ are the ottu forms of ಟ /tta/ and ರ /ra/, respectively.

Table 1. Kannada aksharas and numerals. The Roman names shown for each character are as per the Kannada Unicode chart [Unicode 2018] except for the names Ru, am & ah, which are used in place of the Unicode names vocalic_r, anusvara & visarga for the sake of saving space.

ಅ	ಆ	ಇ	ಈ	ಉ	ಊ	ಋ	ಎ	ಏ	ಐ	ಒ	ಓ	ಔ	ಂ	ಃ
a	aa	i	ii	u	uu	Ru	e	ee	ai	o	oo	au	am	ah

ಕ ka	ಖ kha	ಗ ga	ಘ gha	ಙ nga
ಚ ca	ಛ cha	ಜ ja	ಝ jha	ಞ nya
ಟ tta	ಠ ttha	ಡ dda	ಢ ddha	ಣ nna
ತ ta	ಥ tha	ದ da	ಧ dha	ನ na
ಪ pa	ಫ pha	ಬ ba	ಭ bha	ಮ ma

ಯ ya	ರ ra	ಱ rra	ಲ la	ವ va	ಶ sha	ಷ ssa	ಸ sa	ಹ ha	ಳ lla	ೞ llla

೦	೧	೨	೩	೪	೫	೬	೭	೮	೯
zero	one	two	three	four	five	six	seven	eight	nine

Table 2. Illustration of the formation of Kannada compound characters. The first row lists the Kannada vowels, and the second, their dependent forms (a.k.a. vowel signs). The third row shows the compound characters formed by the joining of the consonant ಕ್ /k/ with each of the vowel signs.

ಅ	ಆ	ಇ	ಈ	ಉ	ಊ	ಋ	ಎ	ಏ	ಐ	ಒ	ಓ	ಔ	ಂ	ಃ	್	
	ಾ	ಿ	ೀ	ು	ೂ	ೃ	ೆ	ೇ	ೈ	ೊ	ೋ	ೌ	ಂ	ಃ	್	
ಕ	ಕಾ	ಕಿ	ಕೀ	ಕು	ಕೂ	ಕೃ	ಕೆ	ಕೇ	ಕೈ	ಕೊ	ಕೋ	ಕೌ	ಕಂ	ಕಃ	ಕ್	್ಕ
ka	kaa	ki	kii	ku	kuu	kRu	ke	kee	kai	ko	koo	kau	kam	kah	k	k_ottu

Some of the ottus (e.g. ್ತ /ta_ottu/, ್ಗ /ga_ottu/, ್ದ /da_ottu/) are short and hence, when a text line contains only such short ottus, it is possible that the ottus form a separate horizontal line, not vertically overlapping with the main text line of base symbols, as shown by the example in Fig. 1b. On the other hand, some of the ottus have more height (e.g. ್ಕ /ka_ottu/, ್ಯ /ya_ottu/, ್ರ /ra_ottu/) and when one or more such conjuncts are present, they vertically overlap with the bottom parts of the base characters and form a more complex line, as shown in the example in Fig. 1a. When the ್ರ /ra_ottu/ follows ್ತ /ta_ottu/ or ್ಟ /tta_ottu/, the two ottus are printed one below the other, forming a ottu complex, as shown in Fig. 1c. The situation becomes more complex, when a ottu or its ascender touches the base character above and forms a merged character, as shown in Fig. 1d.

Interestingly, the vowel at the end of the consonant cluster modifies the shape of the first consonant present in the main text line. For example, in the word ಪ್ರಾಯಶ್ಚಿತ್ತ /praayashcittaa/, for the first consonant cluster ಪ್ರ /praa/, the vowel at the end ಾ /aa/ modifies the grapheme of the first consonant ಪ್ /p/. Similarly, for the second consonant cluster ಶ್ಚಿ /shci/, the vowel at the end ಿ /i/ modifies the grapheme of the first consonant ಶ್ /sh/ in the text, but the phoneme of the second consonant in the uttered word.

2.3 Recognition Units for a Realistic Kannada OCR

Table 3 lists the various distinct symbols used in Kannada script. Each of these symbols normally appears as a single connected component in the scanned image. The top row lists the independent vowels. The last row lists the Kannada numerals, visarga and anuswara/zero. The rest of the rows list the consonants with their various vowel combinations. The last

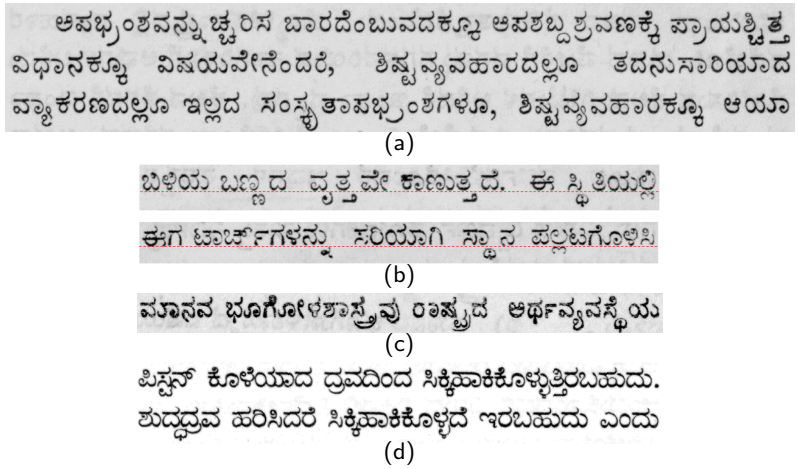


Fig. 1. Sample Kannada images, showing the consonant conjuncts (ottus), their varying sizes and spatial positions with respect to the main text line. (a) Ottus protrude up into the main text line. (b) Ottus form a separate horizontal line. (c) Two ottus stacked vertically one below the other. (d) Some ottus and base characters touch the other characters in the main text line and form merged characters.

column lists the consonants in their *ottu* form. The last but one column lists the pure consonants, which are consonants without any vowel attached to them. The vowel remover (a.k.a. halant) is denoted as /-a/, and ottu as /_ottu/.

The symbol ೦ shown in the last row of Table 3 can represent either anusvara (as in ಅಂ /am/, ಕಂ /kam/, ಚಂ /cam/) or numeral zero and needs disambiguation based on the word context. The symbol ೯ can represent either numeral nine or an alternate ottu form of ರ /ra/ called *arkaa-ottu* as in the word ಸೂರ್ಯ /suurya/. The symbol ೕ, henceforth referred to as dheergha, is used to represent the long (stressed) form of some of the dependent vowels as in ಕೀ /kii/, ಕೇ /kee/ and ಕೋ /koo/. Similarly, the symbol ೖ represents the dependent form of the vowel ಋ /vocalic_r/ as in ಕೃ /kRu/, and the symbol ೈ is the dependent form of vowel ಐ /ai/ as in ಕೈ /kai/. These are illustrated in Table 4.

The symbol ೠ /vocalic_rr/ which was earlier taught as representing the longer form of vowel ಋ /vocalic_r/, is practically non-existent in actual text. Hence we have dropped it from our symbol set.

The Halegannada (meaning old Kannada) had other consonants such as ಱ /rra/ (the stressed ರ /ra/) and ೞ /lla/ (the stressed ಳ /lla/), equivalents of which are still present in modern Tamil and Malayalam. So, the older poems and printed books contain these additional consonants and their vowel-modified forms. In old letterpress printed documents, like the example shown in Fig. 6c, the symbols representing some of the dependent vowels have been printed as separate characters after the base components. For example, ಕಾ /kaa/, ಕು /ku/, ಕೂ /kuu/, ಕೊ /ko/ and ಕೌ /kau/ are printed as ಕ ಾ , ಕ ು, ಕ ೂ, ಕ ೊ and ಕ ೌ, respectively. Similarly, the modifier symbol *halant* used for indicating any pure consonant has been printed separate from the base component (consonant part of CVs) as in ಕ ್. Also, the base components printed before ಾ, ೌ and ್ appear in a slightly cut form, as shown in the first column of Table 5.

Table 6 lists the additional recognition units necessitated by the touching of multiple ottus in an inseparable way, in some of the old books. Table 7 lists the non-Kannada recognition units handled by the OCR. This includes the Indo-Arabic numerals from 1 to 9, eight punctuation symbols, the exclamation and question marks, forward slash as well as a few

Table 3. Kannada recognition units: independent vowels, consonants with/without vowel modifiers, pure consonants and ottus.

ಅ	ಆ	ಇ	ಈ	ಉ	ಊ	ಋ	ಎ	ಏ	ಐ	ಒ	ಓ	ಔ		
a	aa	i	ii	u	uu	Ru	e	ee	ai	o	oo	au	-a	_ottu
ಕ /ka/	ಕಾ	ಕಿ	ಕೀ	ಕು	ಕೂ	ಕೃ	ಕೆ	ಕೇ	ಕೈ	ಕೊ	ಕೋ	ಕೌ	ಕ್	್ಕ
ಖ /kha/	ಖಾ	ಖಿ		ಖು	ಖೂ		ಖೆ			ಖೊ		ಖೌ	ಖ್	್ಖ
ಗ /ga/	ಗಾ	ಗಿ		ಗು	ಗೂ		ಗೆ			ಗೊ		ಗೌ	ಗ್	್ಗ
ಘ /gha/	ಘಾ	ಘಿ		ಘು	ಘೂ		ಘೆ			ಘೊ		ಘೌ	ಘ್	್ಘ
ಙ /nga/	ಙಾ	ಙಿ		ಙು	ಙೂ		ಙೆ			ಙೊ		ಙೌ	ಙ್	್ಙ
ಚ /ca/	ಚಾ	ಚಿ		ಚು	ಚೂ		ಚೆ			ಚೊ		ಚೌ	ಚ್	್ಚ
ಛ /cha/	ಛಾ	ಛಿ		ಛು	ಛೂ		ಛೆ			ಛೊ		ಛೌ	ಛ್	್ಛ
ಜ /ja/	ಜಾ	ಜಿ		ಜು	ಜೂ		ಜೆ			ಜೊ		ಜೌ	ಜ್	್ಜ
ಝ /jha/	ಝಾ	ಝಿ		ಝು	ಝೂ		ಝೆ			ಝೊ		ಝೌ	ಝ್	್ಝ
ಞ /nya/	ಞಾ	ಞಿ		ಞು	ಞೂ		ಞೆ			ಞೊ		ಞೌ	ಞ್	್ಞ
ಟ /tta/	ಟಾ	ಟಿ		ಟು	ಟೂ		ಟೆ			ಟೊ		ಟೌ	ಟ್	್ಟ
ಠ /ttha/	ಠಾ	ಠಿ		ಠು	ಠೂ		ಠೆ			ಠೊ		ಠೌ	ಠ್	್ಠ
ಡ /dda/	ಡಾ	ಡಿ		ಡು	ಡೂ		ಡೆ			ಡೊ		ಡೌ	ಡ್	್ಡ
ಢ /ddha/	ಢಾ	ಢಿ		ಢು	ಢೂ		ಢೆ			ಢೊ		ಢೌ	ಢ್	್ಢ
ಣ /nna/	ಣಾ	ಣಿ		ಣು	ಣೂ		ಣೆ			ಣೊ		ಣೌ	ಣ್	್ಣ
ತ /ta/	ತಾ	ತಿ		ತು	ತೂ		ತೆ			ತೊ		ತೌ	ತ್	್ತ
ಥ /tha/	ಥಾ	ಥಿ		ಥು	ಥೂ		ಥೆ			ಥೊ		ಥೌ	ಥ್	್ಥ
ದ /da/	ದಾ	ದಿ		ದು	ದೂ		ದೆ			ದೊ		ದೌ	ದ್	್ದ
ಧ /dha/	ಧಾ	ಧಿ		ಧು	ಧೂ		ಧೆ			ಧೊ		ಧೌ	ಧ್	್ಧ
ನ /na/	ನಾ	ನಿ		ನು	ನೂ		ನೆ			ನೊ		ನೌ	ನ್	್ನ
ಪ /pa/	ಪಾ	ಪಿ		ಪು	ಪೂ		ಪೆ			ಪೊ		ಪೌ	ಪ್	್ಪ
ಫ /pha/	ಫಾ	ಫಿ		ಫು	ಫೂ		ಫೆ			ಫೊ		ಫೌ	ಫ್	್ಫ
ಬ /ba/	ಬಾ	ಬಿ		ಬು	ಬೂ		ಬೆ			ಬೊ		ಬೌ	ಬ್	್ಬ
ಭ /bha/	ಭಾ	ಭಿ		ಭು	ಭೂ		ಭೆ			ಭೊ		ಭೌ	ಭ್	್ಭ
ಮ /ma/	ಮಾ	ಮಿ		ಮು	ಮೂ		ಮೆ			ಮೊ		ಮೌ	ಮ್	್ಮ
ಯ /ya/	ಯಾ	ಯಿ		ಯು	ಯೂ		ಯೆ			ಯೊ		ಯೌ	ಯ್	್ಯ
ರ /ra/	ರಾ	ರಿ		ರು	ರೂ		ರೆ			ರೊ		ರೌ	ರ್	್ರ
ಱ /rra/	ಱಾ	ಱಿ		ಱು	ಱೂ		ಱೆ			ಱೊ		ಱೌ	ಱ್	್ಱ
ಲ /la/	ಲಾ	ಲಿ		ಲು	ಲೂ		ಲೆ			ಲೊ		ಲೌ	ಲ್	್ಲ
ಳ /lla/	ಳಾ	ಳಿ		ಳು	ಳೂ		ಳೆ			ಳೊ		ಳೌ	ಳ್	್ಳ
ೞ /llla/	ೞಾ	ೞಿ		ೞು	ೞೂ		ೞೆ			ೞೊ		ೞೌ	ೞ್	್ೞ
ವ /va/	ವಾ	ವಿ		ವು	ವೂ		ವೆ			ವೊ		ವೌ	ವ್	್ವ
ಶ /sha/	ಶಾ	ಶಿ		ಶು	ಶೂ		ಶೆ			ಶೊ		ಶೌ	ಶ್	್ಶ
ಷ /ssa/	ಷಾ	ಷಿ		ಷು	ಷೂ		ಷೆ			ಷೊ		ಷೌ	ಷ್	್ಷ
ಸ /sa/	ಸಾ	ಸಿ		ಸು	ಸೂ		ಸೆ			ಸೊ		ಸೌ	ಸ್	್ಸ
ಹ /ha/	ಹಾ	ಹಿ		ಹು	ಹೂ		ಹೆ			ಹೊ		ಹೌ	ಹ್	್ಹ
ಃ	ಂ			೧/1	೨/2	೩/3	೪/4	೫/5	೬/6	೭/7		೮/8	೯/9	
visarga	anusvara/zero			one	two	three	four	five	six	seven		eight	nine / ರ್	

Table 4. Kannada recognition units that need disambiguation based on the preceding unit and the dependent vowels that appear as separate symbols.

Glyph	Possible Characters	ೀ /ii/	ೃ /Ru/	ೈ /ai/
೦	Anusvara or Kannada digit 0	ಕಿ /ki/ ಕೀ /kii/	ಕ /ka/ ಕೃ /kRu/	ಕೈ /ke/ ಕೈೃ /kai/
೯	Kannada digit 9 or arkaa-ottu (ರ ್)	ಕೈ /ke/ ಕೇ /kee/		
		ಕೊ /ko/ ಕೋ /koo/		

other symbols used as sentence-ending periods (danda or double-danda used in poetic text) and avagraha. The avagraha symbol ऽ is used in Kannada script for representing elongated vowels in poetic text, mainly borrowed from Sanskrit. In addition, all the Roman characters, both upper and lower case, are also recognized by the OCR.

An OCR is most useful to digitize old, rather than new books. Thus, a good Kannada OCR needs to handle all of these different possibilities. All the symbols shown in Tables 3, 5, 6 and 7 together form the set of 390 units that are recognized by our OCR.

3 THE DESIGN OF LIPI GNANI

Figure 2 gives the block diagram of the various steps performed by our Kannada OCR. The input image is first binarized. The skew angle is detected by Hough transform and the original gray image is corrected by bilinear interpolation and re-binarized. Then the skew-corrected binarized image is segmented into lines, words, and characters/symbols. The segmented words are tested to detect and correct italic formatting, if any. The symbols in our case are horizontally overlapping connected components as shown in Tables 3, 5, 6 and 7. Symbol segmentation is followed by feature extraction, classification and Viterbi based selection of the best sequence of classifier outputs using symbol-level bigram probability. Finally, the Unicodes of the recognized symbols are concatenated in the correct order to generate the word level Unicode string.

3.1 Text Block Selection, Skew Correction and Binarization

In our OCR, we have skipped the implementation of automatic text block extraction, but instead provide an intuitive GUI, where users can draw rectangles and specify text blocks for a multi-column input page. If no text blocks are specified, then we assume the input image to contain a single text block.

As explained in [Ashwin and Sastry 2002], the presence of short horizontal line on top of most of the Kannada characters ensures that the Hough transform technique for detecting skew [Srihari and Govindaraju 1989] works well for Kannada. Hence, we employ Hough transform to detect any skew introduced during scanning. The resolution with which the skew angle is detected is 0.1°. Our experiments show that the classifier's performance is not affected by a very small skew angle of 0.25°. Hence, if the detected skew angle θ_S is greater than 0.25°, then we correct the skew by rotating the original gray-level image [Aparna and Ramakrishnan 2002] in the opposite direction (by an angle of $-\theta_S$) and, then binarize the skew corrected image as explained below.

To make sure there is no loss of information during scanning, we recommend the input document to be scanned as a gray-level image at a minimum resolution of 300 dpi and the scanned image to be saved in uncompressed image format (like uncompressed TIFF). Since information we need for text recognition is innately binary (text-ink present or not), we binarize the input image using an effective binarization algorithm to make sure that

Table 5. Additional Kannada recognition units arising due to the nature of printing in some old letterpress printed documents. The glyphs in the first column and the title row are additional symbols that are used to form the aksharas shown in the rest of the rows and columns.

	ಾ /aa/	ೌ /au/	್ /halant/	ು /u/	ೂ /uu/		ೂ /uu/	
ಕ /k/	ಕಾ /kaa/	ಕೌ /kau/	ಕ್ /k/	ಕ /ka/	ಕು /ku/	ಕೂ /kuu/	ಕೆ /ke/	ಕೊ /ko/
ಖ /kh/	ಖಾ	ಖೌ	ಖ್	ಖ	ಖು	ಖೂ	ಖೆ	ಖೊ
ಗ /g/	ಗಾ	ಗೌ	ಗ್	ಗ	ಗು	ಗೂ	ಗೆ	ಗೊ
ಘ /gh/	ಘಾ	ಘೌ	ಘ್	ಘ	ಘು	ಘೂ	ಘೆ	ಘೊ
ಙ /ng/	ಙಾ	ಙೌ	ಙ್	ಙ	ಙು	ಙೂ	ಙೆ	ಙೊ
ಚ /c/	ಚಾ	ಚೌ	ಚ್	ಚ	ಚು	ಚೂ	ಚೆ	ಚೊ
ಛ /ch/	ಛಾ	ಛೌ	ಛ್	ಛ	ಛು	ಛೂ	ಛೆ	ಛೊ
ಜ /j/	ಜಾ	ಜೌ	ಜ್	ಜ	ಜು	ಜೂ	ಜೆ	ಜೊ
ಞ /ny/	ಞಾ	ಞೌ	ಞ್	ಞ	ಞು	ಞೂ	ಞೆ	ಞೊ
ಟ /tt/	ಟಾ	ಟೌ	ಟ್	ಟ	ಟು	ಟೂ	ಟೆ	ಟೊ
ಠ /tth/	ಠಾ	ಠೌ	ಠ್	ಠ	ಠು	ಠೂ	ಠೆ	ಠೊ
ಡ /dd/	ಡಾ	ಡೌ	ಡ್	ಡ	ಡು	ಡೂ	ಡೆ	ಡೊ
ಢ /ddh/	ಢಾ	ಢೌ	ಢ್	ಢ	ಢು	ಢೂ	ಢೆ	ಢೊ
ಣ /nn/	ಣಾ	ಣೌ	ಣ್	ಣ	ಣು	ಣೂ	ಣೆ	ಣೊ
ತ /t/	ತಾ	ತೌ	ತ್	ತ	ತು	ತೂ	ತೆ	ತೊ
ಥ /th/	ಥಾ	ಥೌ	ಥ್	ಥ	ಥು	ಥೂ	ಥೆ	ಥೊ
ದ /d/	ದಾ	ದೌ	ದ್	ದ	ದು	ದೂ	ದೆ	ದೊ
ಧ /dh/	ಧಾ	ಧೌ	ಧ್	ಧ	ಧು	ಧೂ	ಧೆ	ಧೊ
ನ /n/	ನಾ	ನೌ	ನ್	ನ	ನು	ನೂ	ನೆ	ನೊ
ಪ /p/	ಪಾ	ಪೌ	ಪ್	-	-	-	-	-
ಫ /ph/	ಫಾ	ಫೌ	ಫ್	-	-	-	-	-
ಬ /b/	ಬಾ	ಬೌ	ಬ್	ಬ	ಬು	ಬೂ	ಬೆ	ಬೊ
ಭ /bh/	ಭಾ	ಭೌ	ಭ್	ಭ	ಭು	ಭೂ	ಭೆ	ಭೊ
ರ /r/	ರಾ	ರೌ	ರ್	ರ	ರು	ರೂ	ರೆ	ರೊ
ಲ /l/	ಲಾ	ಲೌ	ಲ್	ಲ	ಲು	ಲೂ	ಲೆ	ಲೊ
ಳ /ll/	ಳಾ	ಳೌ	ಳ್	ಳ	ಳು	ಳೂ	ಳೆ	ಳೊ
ವ /v/	ವಾ	ವೌ	ವ್	-	-	-	-	-
ಶ	ಶಾ	ಶೌ	ಶ್	ಶ	ಶು	ಶೂ	ಶೆ	ಶೊ
ಷ /ss/	ಷಾ	ಷೌ	ಷ್	ಷ	ಷು	ಷೂ	ಷೆ	ಷೊ
ಸ /s/	ಸಾ	ಸೌ	ಸ್	ಸ	ಸು	ಸೂ	ಸೆ	ಸೊ
ಹ /h/	ಹಾ	ಹೌ	ಹ್	ಹ	ಹು	ಹೂ	ಹೆ	ಹೊ

	ು /u/	ೂ /aa*/	ೣ /u*/	ೡ /uu*/	ೌ /au*/	್ /halant*/
ಝ /jha_part/	ಝು /jha/	ಝೂ /jhaa/	ಝೣ /jhu/	ಝೡ /jhuu/	ಝೌ /jhau/	ಝ್ /jh/
ವ /va/	ಮ /ma/	ಮಾ /maa/	ಮು /mu/	ಮೂ /muu/	ಮೌ /mau/	ಮ್ /m/
ಯ /ya_part/	ಯ /ya/	ಯಾ /yaa/	ಯು /yu/	ಯೂ /yuu/	ಯೌ /yau/	ಯ್ /y/

	ು /u/		ು /u/	ೂ /uu/		ು /u/	ೂ /uu/	
ಝಿ /jhi_part/	ಝಿ /jhi/	ಝೆ /jhe_part/	ಝು /jhe/	ಝೊ /jho/	ಝು /jha/	ಝು /jhu/	ಝೂ /jhuu/
ವಿ /vi/	ಮಿ /mi/	ವೆ /ve/	ಮು /me/	ಮೊ /mo/	ಮ /ma/	ಮು /mu/	ಮೂ /muu/
ಯಿ /yi_part/	ಯಿ /yi/	ಯೆ /ye_part/	ಯು /ye/	ಯೊ /yo/	ಯ /ya/	ಯು /yu/	ಯೂ /yuu/

Table 6. Additional Kannada recognition units formed by the touching of multiple ottus.

Glyph	Unicode Sequence	Example
೬ /kRu_ottu/	್/-a/ ಕ/ka/ ೃ/Ru/	ಸಂಸ್ಕೃತಿ /samskRuti/
೨ /ttra_ottu/	್/-a/ ಟ/tta/ ್/-a/ ರ/ra/	ರಾಷ್ಟ್ರ /rassttra/
೯ /tra_ottu/	್/-a/ ತ/ta/ ್/-a/ ರ/ra/	ಶಾಸ್ತ್ರ /shastra/
೮ /pra_ottu/	್/-a/ ಪ/pa/ ್/-a/ ರ/ra/	ಜಗತ್ಪ್ರಸಿದ್ಧ /jagatprasidda/
೭ /vra_ottu/	್/-a/ ವ/va/ ್/-a/ ರ/ra/	ದಿಗ್ವ್ರತ /digvrata/
೫ /rai_ottu/	್/-a/ ರ/ra/ ೈ/ai/	ಕ್ರೈಸ್ತ /kraista/

Table 7. Indo-Arabic numerals and special symbols handled.

1	2	3	4	5	6	7	8	9	।	॥
?	(	)	[	]	:	;	!	/	|	ऽ
.	,	‘	’	'	-					

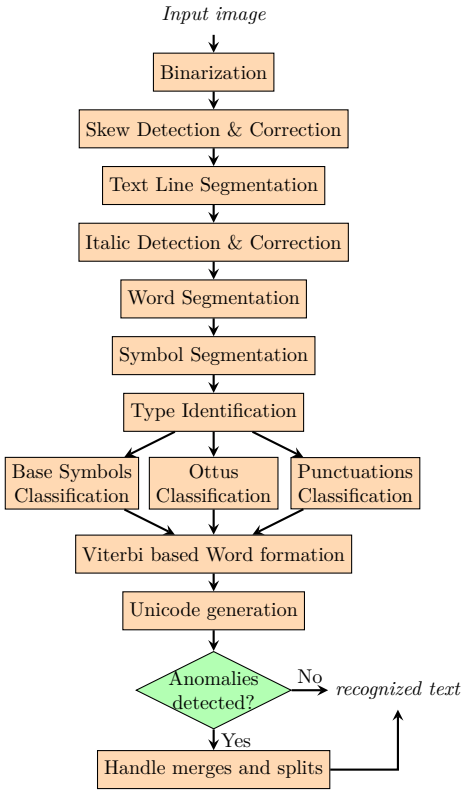


Fig. 2. Block Diagram of Lipi Gnani - the Kannada OCR.

individual characters/symbols are not cut and consecutive symbols are not merged. Otsu's global thresholding algorithm [Otsu 1979] performs well for good quality documents and

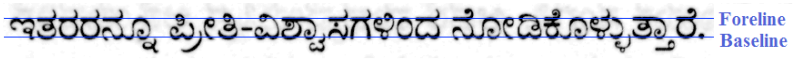


Fig. 3. Sample Kannada text line showing ottus and their position with respect to foreline and baseline.

that is what we have implemented at the first level. However, in the case of degraded documents, we need better binarization techniques and this is addressed in Section 3.7.

3.2 Line, Word and Character Segmentation and Italic Correction

For text line segmentation, we use a bottom up approach based on the sizes and the relative positions of the connected components in the document page.

It is well known that the angle of tilt in italic text is normally around 15°. Hence, for slant/italic detection and correction, the line image is sheared [Sun and Si 1997] at different angles in steps of 2° up to 20° and the vertical projection profile (VPP) is taken at every step. The image corresponding to the VPP with maximum character/word gaps is considered as the italics corrected image and is used for word segmentation.

The character and the word gaps generally increase with the font size of the characters. The height of a text line is directly proportional to the font size. However, in Kannada, the complete line height varies, since it includes the height of the ottus, if present. Hence, we use the foreline to baseline height as the reference. Empirically, the character gap threshold is obtained as 0.35 times this reference height. Thus, we classify the gaps between symbols (zeros in vertical projection profile) into either inter-word gaps or intra-word gaps by using this adaptive threshold.

Kannada has a few consonants such as ಪ /pa/ and ಸ /sa/, which can have 3 unconnected components one above the other. Therefore, for symbol segmentation, we extract the connected components (CCs) and group CCs that have significant horizontal overlap above the base-line into one symbol. For example, in the word ಸ್ವಂತ, the segmented symbols are ಸ, ◌್ವ, ಂ and ತ. Incidentally, this step takes care of any split in the character in the vertical direction.

3.3 Handling of Ottus and Special Symbols

Ottu symbols are printed below the base symbols and most of them occur completely below the baseline. However, in the case of some ottus, a small portion extends up above the baseline, as shown in Figures 3 and 1a. This relative position of the ottu symbols with respect to the text line is leveraged for recognizing them separately from the rest of the symbols. Once a connected component is identified as belonging to the ottu group, we recognize it by extracting features from its 32×32 normalized image and passing them to a separate SVM classifier trained only on ottu symbols.

The numerals and the special symbols listed in the first and second rows of Table 7 are recognized along with the rest of the base symbols by the primary SVM classifier. However, the punctuation marks listed in the third row of Table 7, namely the dot(.), comma(,), left single quotation(‘), right single quotation(’), apostrophe(') and hyphen(-) are recognized directly, independent of the rest of the symbols, using their position relative to the baseline and foreline, size relative to the line height and aspect ratio.

3.4 Feature Extraction and Classification

It was shown in [Vijay Kumar and Ramakrishnan 2002] that Haar wavelet features give best recognition results for Kannada. Accordingly, the segmented components are resized

to 32×32 size and DWT features are extracted from them using Haar wavelet. In addition, horizontal and vertical autocorrelation-based features [Madhavaraj et al. 2014] are also extracted and all the above three are concatenated to form a feature vector of dimension 3072. Two SVM classifiers [Chang and Lin 2011; Cortes and Vapnik 1995] with linear kernel [Fan et al. 2008] are used for classifying the base symbols and ottus. The base SVM has 351 classes and the other SVM has 39 ottus. Both the SVMs are trained with 5-fold cross-validation to choose the best value of the parameter C.

We have a total of 390 classes handled by the two SVMs together. On the average, we have collected 200 training samples for each class. These training samples have been collected from over 35 books published between the period 1970 to 2002, and this includes books written in Halegannada and pages containing Sanskrit slokas written in Kannada script. Thus, the training set includes many different font styles, printing styles (letter press Vs. DTP), and various kinds of degradations. In addition, there is a third SVM, which recognizes only the Roman characters, which has 47 classes.

3.5 Viterbi Algorithm for Word Formation

In old books with poor print quality, portions of some of the characters are lost either directly in print or during binarization, as illustrated by the word image in Fig. 4. When such symbols are fed into the classifier, instead of getting a single recognized class with very high confidence level, we might get multiple labels, all with low confidence levels. This is illustrated in Fig. 4, where each of the segmented components is shown above the horizontal line and the recognized classes, as output by the classifier, are shown below the horizontal line. The confidence level output by the classifier is shown below each of the recognized classes. In such cases, mere selection of the top recognized class may yield sub-optimal recognition results. Thus, in this example, the incorrect Unicode sequence of ಇಲ್ಯದಿಜ್ಜಲ್ಲಿ will result for the word image shown in Fig. 4.

Leveraging symbol level bigram probabilities and the Viterbi algorithm [Forney 1973] for selecting the optimal sequence of recognized classes increases the chances of correct recognition, as shown in Fig. 4. Here, the symbol level bigram probabilities are shown on the edges joining the recognized classes and the optimal path selected by the Viterbi algorithm is shown using thick edges, yielding the correct Unicode output of ಇಲ್ಲದಿದ್ದಲ್ಲಿ. For extracting the symbol level bigram probabilities, we have used a dump of Kannada Wikipedia text and our algorithm for mapping Kannada Unicode text to the corresponding sequence of Kannada OCR symbols (recognition units).

While the regular Viterbi algorithm uses a fixed number (top-N) of recognition classes for each segmented symbol, we prune the number of recognition classes considered for each symbol based on the distribution of confidence levels. Thus, for the first symbol in Fig. 4, we consider only the best class, since its confidence level is greater than 0.75. However, for the second symbol, we choose the top-2, since their confidence levels are comparable, while that of the third class is too low. For the fifth symbol, we go all the way up to the top-6 classes, since the confidence levels decrease slowly and we stop when two successive confidence values have a ratio exceeding five.

3.6 Kannada Unicode Generation Module

After symbol segmentation and classification, the recognized labels need to be put together in the correct order so as to generate a valid Unicode sequence [Unicode 2018] at the word level. Table 8 gives the Unicode sequences for a few recognition units used in our OCR. Table 9 lists the accurate Unicode sequences for some sample Kannada words. For the word

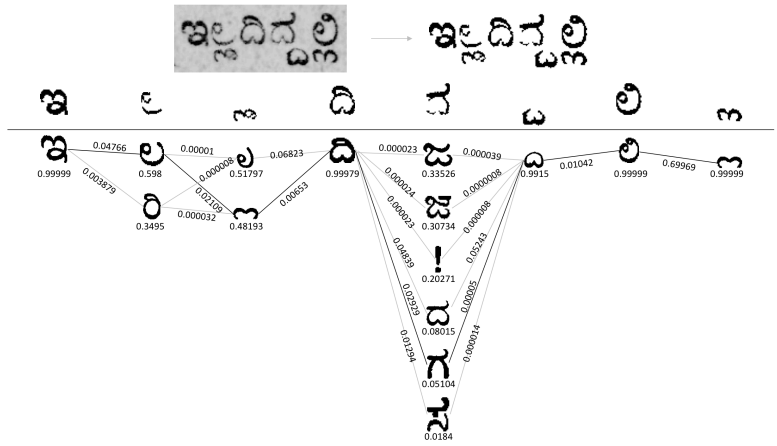


Fig. 4. Illustration of Viterbi-based word formation, using a sample degraded word image.

Table 8. Unicodes corresponding to some sample recognition units of our Kannada OCR.

Symbol	ಕ/ka/	ಕಾ/kaa/	ಕಿ/ki/	ಕು/ku/	ಕೂ/kuu/	ಕೆ/ke/	ಕೊ/ko/	ಕೌ/kau/	ಕ್/k/	್ಕ/k_ottu/
Unicodes	ಕ	ಕ ಾ	ಕ ಿ	ಕ ು	ಕ ೂ	ಕ ೆ	ಕ ೊ	ಕ ೌ	ಕ ್	್ ಕ

Table 9. The correct Unicode sequences for some sample Kannada words.

Word	Unicode Sequence
ರಾಷ್ಟ್ರಪತಿ /raassttrapati/	ರ/ra/ ಾ /aa/ ಷ/ssa/ ್/-a/ ಟ/tta/ ್/-a/ ರ/ra/ ಪ/pa/ ತ/ta/ ಿ /i/
ಸಂಸ್ಕೃತಿ /samskRuti/	ಸ/sa/ ಂ/am/ ಸ/sa/ ್/-a/ ಕ/ka/ ೃ/Ru/ ತ /ta/ ಿ /i/
ಕ್ರೈಸ್ತ /kraista/	ಕ/ka/ ್/-a/ ರ/ra/ ೈ/ai/ ಸ/sa/ ್/-a/ ತ/ta/
ಸೂರ್ಯ /suurya/	ಸ/sa/ ೂ /uu/ ರ/ra/ ್/-a/ ಯ/ya/

ರಾಷ್ಟ್ರಪತಿ /raassttrapati/, the sequence of segmented and recognized symbols are ರಾ /raa/, ಷ /ssa/, ್ಟ /tta_ottu/, ್ರ /r_ottu/, ಪ /pa/ and ತಿ /ti/. Putting together the Unicodes associated with these symbols in the same order, we get ರ ಾ ಷ ್ ಟ ್ ರ ಪ ತ ಿ, which matches with the expected Unicode for that word.

However, for the word ಕ್ರೈಸ್ತ /kraista/, the sequence of segmented and recognized symbols are ಕೆ /ke/, ್ರ /r_ottu/, ೈ /vowel_sign_ai/, ಸ /sa/ and ್ತ /t_ottu/. Putting their associated Unicodes in the same order would give us ಕ ೆ ್ ರ ೈ ಸ ್ ತ, but the correct Unicode for that word is ಕ ್ ರ ೈ ಸ ್ ತ. Similarly, for the word ಸೂರ್ಯ /suurya/, the sequence of segmented and recognized symbols are ಸೂ /suu/, ಯ /ya/ and ರ್ /arkaaottu/. Putting their associated Unicodes in the same order would give us ಸ ೂ ಯ ರ ್, but the correct Unicode for that word is ಸ ೂ ರ ್ ಯ.

To generate the correct Unicode string at the word level, we use the logic illustrated by the flowcharts of Fig. 5. From the list of recognized symbols, we first combine part characters as per Table 5. For example, if the recognized symbols have the sequence ಕ and ಾ, then we replace it by ಕಾ. Similarly, ನ and ು is replaced by ನು, and, ಪ and ಾ is replaced by ಮಾ. The next step, as shown in Fig. 5a, is to group the sequence of recognized symbols into aksharas. Symbols that represent independent vowels, consonants with/without dependent

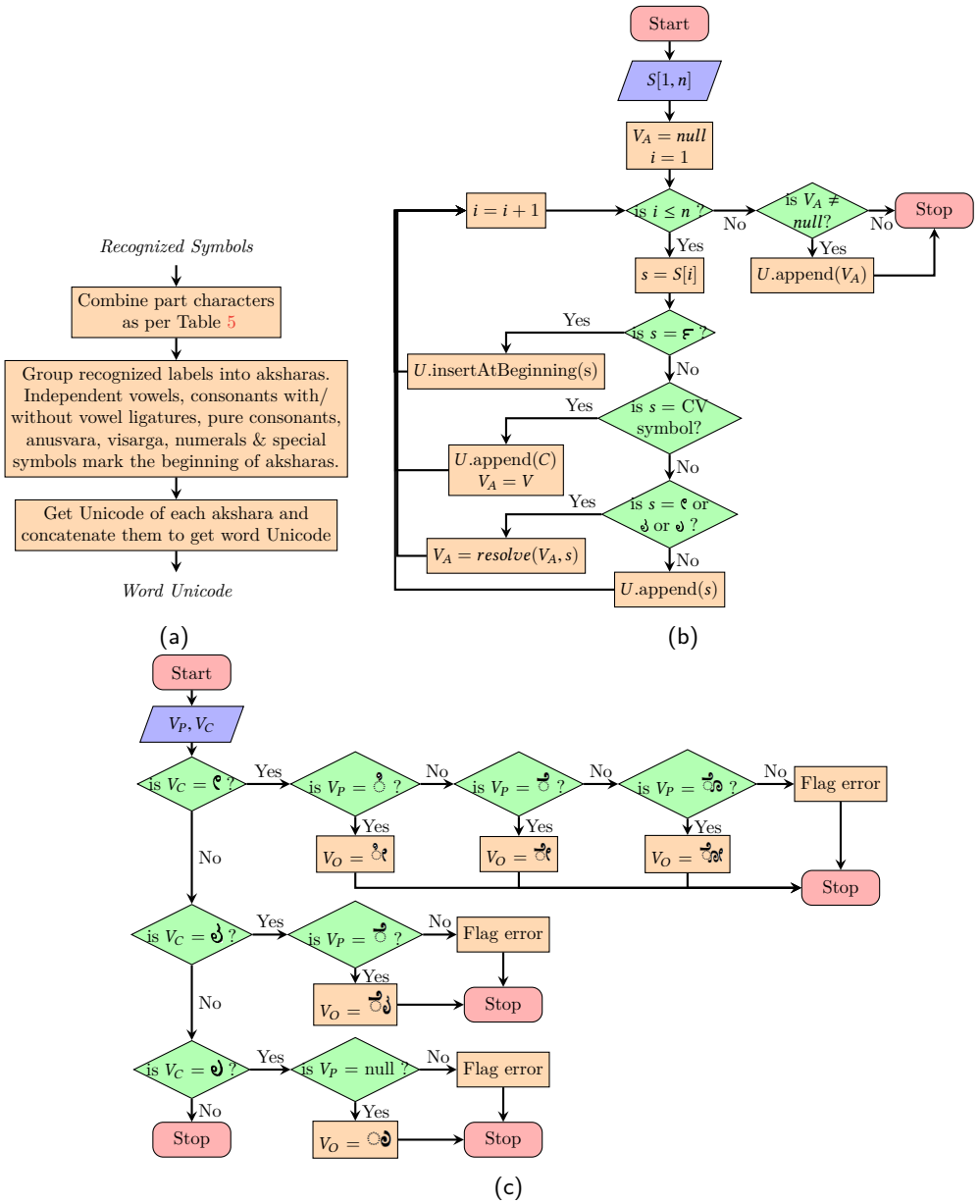


Fig. 5. (a) Block diagram for getting the word Unicode from the recognized labels. (b) Flowchart to generate Unicode for any akshara. The input is the list of recognized labels S. The output is the list of Unicodes U. V_A is used for storing the vowel to be inserted at the end of the akshara. (c) Flowchart for resolving the vowel sign within the akshara as per Table 4. V_P and V_C are the labels of the previous and current vowel modifiers. V_O is the resolved label of the output vowel modifier.

vowels, numerals and special symbols mark the beginning of aksharas. Symbols ೇ, ೆ, ೈ and ೯ and ottu symbols get added to the previous akshara. In order to keep the logic simple, we consider anusvara and visarga as separate aksharas. Then, we generate the Unicode for each akshara as shown in Fig. 5b, and concatenate them to get the word level Unicode.

As shown in Fig. 5b, the idea for generating the correct Unicode at the akshara level is to concatenate the Unicodes of all the symbols in it with the following exceptions:

- If ೯ is present, then insert it at the beginning of the akshara. For example, in the word ಸೂರ್ಯ, for the second akshara ರ್ಯ, we now get the correct Unicode ರ ್ ಯ.
- If the akshara begins with the symbol of a consonant-vowel combination, then we take the dependent vowel out and insert it at the end of the akshara. If the akshara has other vowel modifier symbols in it (such as ೇ, ೆ or ೈ), then we combine it with the first symbol's dependent vowel as per Fig. 5c, and insert the combined vowel modifier at the end of the akshara. For example, in the word ಕ್ರೈಸ್ತ, for the first akshara ಕ್ರೈ, the first symbol's vowel modifier ಿ is combined with the vowel modifier symbol ೆ to give the Unicode ೈ, which is then inserted at the end of the akshara, to get the correct Unicode sequence ಕ ್ ರ ೈ.

3.7 Handling of Merges and Splits

Old printed documents often contain broken and merged characters, which bring down the performance of the OCR, since they give rise to segmented components that do not belong to any of the recognition units defined earlier. The performance of the OCR is thus further improved, by handling the issue of broken characters using gamma enhanced binarization [Shiva Kumar and Ramakrishnan 2019] and splitting the merged characters using simplified paired valleys and L-cut algorithm [Shiva Kumar et al. 2019].

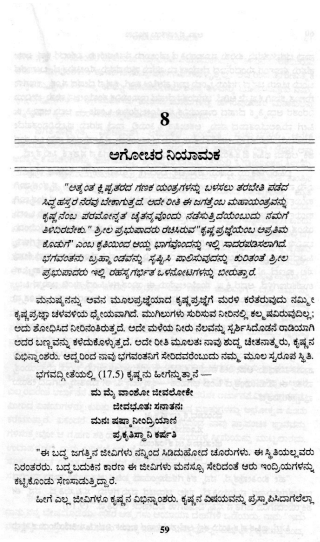
4 PERFORMANCE EVALUATION - OPEN DATASETS AND METRICS

4.1 Benchmark Dataset of Kannada Printed Pages

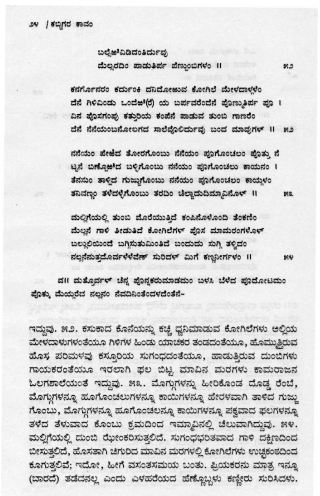
The Kannada dataset [MILE lab 2019] contains 250 images, carefully chosen to have various kinds of recognition challenges. All of these pages have been scanned as gray level images at 300 dpi resolution, and have Unicode ground truth, together with line and word coordinate information. Figure 6 gives some examples from this benchmark dataset created by us. Some of the pages have italics and bold characters (see Fig. 6a); some of them have Halegannada poems and text (see Fig. 6b); others are letterpress-printed pages, where the vowel modifiers appear as separate symbols and do not touch the consonants they go with (see Fig. 6c). Some pages have interspersed English words (see Fig. 6d); still others have Tables with a lot of numeric data (see Fig. 6e). In addition, there are old pages containing either a lot of broken characters or many words with two or more characters merged into a single connected component.

4.2 Multilingual Datasets Printed in Kannada Script

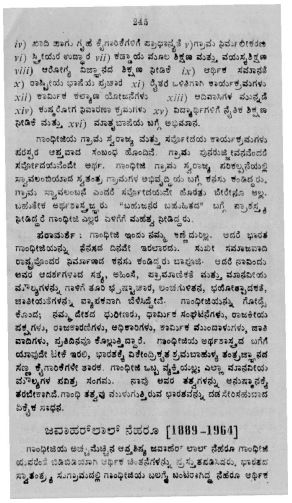
Many Kannadigas read the Sanskrit *slokas* (prayers) printed in Kannada script. In addition, the languages of Tulu [Wikipedia 2018b] and Konkani [Wikipedia 2018a] do not have dedicated scripts of their own. Almost all of the Tulu works are printed in Kannada script, and Konkani works are printed in Devanagari, Kannada or Malayalam scripts. Our OCR can cater to the digitization of documents from any of the above three languages printed in Kannada script. Accordingly, we have created benchmarking datasets for the above languages also.



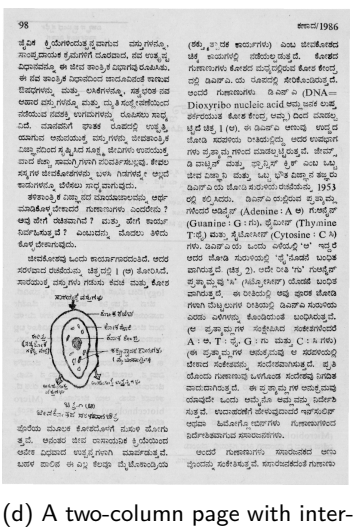
(a) A page with italic and bold text.



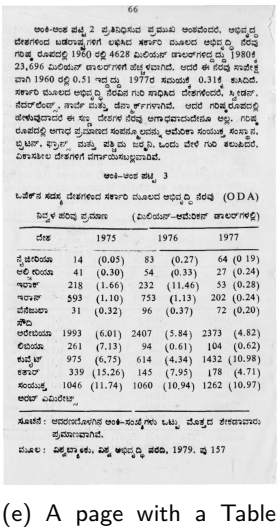
(b) A page with Halegannada letters and text.



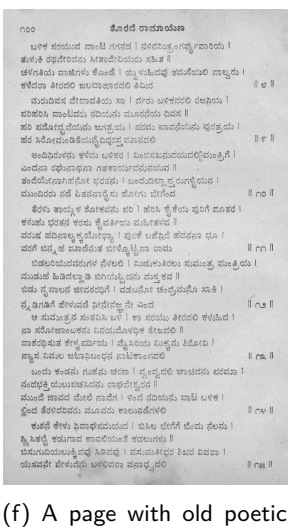
(c) A letterpress printed page: modifiers not connected to consonants.



(d) A two-column page with interspersed English words.



(e) A page with a Table and Indo-Arabic numerals.



(f) A page with old poetic text, containiing Kannada numerals.

Fig. 6. Samples from the Kannada benchmarking test dataset of 250 images.

The Konkani dataset [MILE lab 2018a] contains 40 pages, scanned from different books. The Tulu [MILE lab 2018c] and the Sanskrit [MILE lab 2018b] datasets have 60 pages each. These datasets with Kannada Unicode text coming from other languages are good testing ground for the raw recognition accuracy of the Kannada OCR, since many of the words in them may not be present in the dictionary of a typical Kannada OCR.

4.3 Computation of Recognition Accuracy

Let N be the size of the ground truth (number of Unicodes for Unicode level accuracy OR number of words for word level accuracy) & M, the size of the OCR output. Both at the character (Unicode) and the word levels, we compute the number of substitutions S, insertions I and deletions D, using Levenshtein distance [Wagner and Fischer 1974].

The total error measured and the recognition accuracy are defined as:

$$Error = S + I + D \tag{1}$$

$$Accuracy = (N - Error)/N \tag{2}$$

5 RESULTS AND DISCUSSION

5.1 Tesseract Version Compared

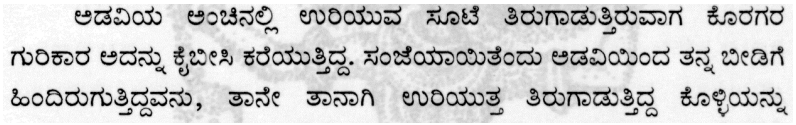
We compare our recognition results on all the test sets with those of the Google's Tesseract OCR. Tesseract was originally developed at Hewlett Packard (HP) between 1985 and 1994, and was open sourced by HP in 2005. Since 2006, it is being developed by Google. Thus, this OCR development has a history of over 30 years, backed by two multinational companies. In a DAS-2016 tutorial [Smith 2016], Tesseract's T-LSTM engine (available in v4.0.0) was reported to have the best Unicode and word-level recognition accuracy for Kannada language. Hence, we tested on the latest version (v4.0.0) of Tesseract with command line option `--oem 1` [Tesseract 2018].

The focus of our testing is on Unicode and word level recognition accuracies. Hence, to make sure that any possible mistakes in the layout analysis (like text/image block segmentation) does not impact the recognition accuracies, we manually pass the boundary coordinates of the text region of the test images to Tesseract through uzn [White 2014] files.

5.2 Results on Kannada Benchmarking Dataset

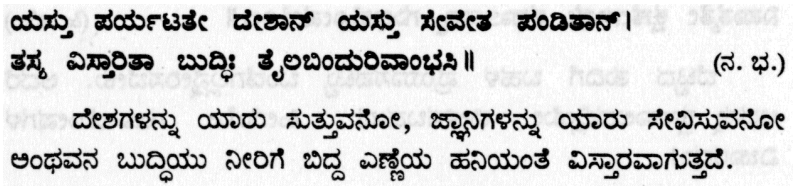
Figures 7 to 10 display the results obtained from our OCR for some sample image segments from the test database, as well as the recognition accuracy at Unicode level (UA) and word level (WA). Figure 7 shows a small part of the scanned image of a Kannada printed page, where the picture on the other side of the paper is clearly seen through. Our OCR perfectly recognizes the text present (see bottom part of Fig. 7). Figure 8 shows a portion of the scanned image of a Kannada printed page with half of the text bold and the rest normal. The text recognized by Lipi Gnani OCR has only two errors (see bottom part of Fig. 8). Figure 9 shows a segment of the scanned image of a Kannada printed page with all italic text. The text recognized by our OCR has very few errors (see bottom part of Fig. 9). Finally, Fig. 10 shows a small section of the scanned image of a highly degraded Kannada printed page. This is letterpress printed, and hence, all the vowel modifiers appear clearly separated from their corresponding consonants. In spite of the degradation present, the text recognized by our OCR has just three errors (see bottom part of Fig. 10).

Figure 11 compares the performances of Lipi Gnani and Tesseract OCRs on the benchmark Kannada dataset. Plotted in the two subfigures are the Unicode and word error rates obtained by the two OCRs on each of the pages. The Unicode error rate and hence, the word error rate obtained by our OCR are lower in nearly 200 of the 250 images in the database. Table 10 compares the average accuracy of the two OCRs on the whole dataset. Lipi Gnani performs marginally better, with an improvement of 0.84% in UA and 5.36%



ಅಡವಿಯ ಅಂಚಿನಲ್ಲಿ ಉರಿಯುವ ಸೂಟೆ ತಿರುಗಾಡುತ್ತಿರುವಾಗ ಕೊರಗರ
ಗುರಿಕಾರ ಅದನ್ನು ಕೈಬೀಸಿ ಕರೆಯುತ್ತಿದ್ದ. ಸಂಜೆಯಾಯಿತೆಂದು ಅಡವಿಯಿಂದ ತನ್ನ ಬೀಡಿಗೆ
ಹಿಂದಿರುಗುತ್ತಿದ್ದವನು, ತಾನೇ ತಾನಾಗಿ ಉರಿಯುತ್ತ ತಿರುಗಾಡುತ್ತಿದ್ದ ಕೊಳ್ಳಿಯನ್ನು

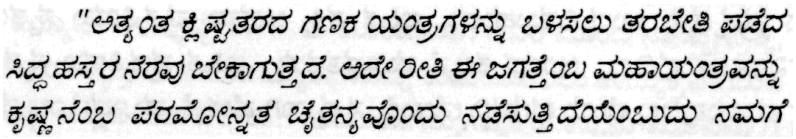
Fig. 7. A segment of a Kannada document, where the picture from the other side of the paper is seen through in the scanned image and the text recognized by Lipi Gnani. UA = 99.09%, WA = 95.12% for the entire page.



ಯಸ್ತು ಪರ್ಯಟತೇ ದೇಶಾನ್ ಯಸ್ತು ಸೇವೇತ ಪಂಡಿತಾನ್।
ತಸ್ಯ ವಿಸ್ತಾರಿತಾ ಬುದ್ಧಿಃ ತೈಲಬಿಂದುರಿದಾಂಭಸಿ ।। (ನ. ಭ.)

ದೇಶಗಳನ್ನು ಯಾರು ಸುತ್ತುವನೋ, ಜ್ಞಾವಿಗಳನ್ನು ಯಾರು ಸೇವಿಸುವನೋ
ಅಂಥವನ ಬುದ್ಧಿಯು ನೀರಿಗೆ ಬಿದ್ದ ಎಣ್ಣೆಯ ಹನಿಯುತೆ ವಿಸ್ತಾರವಾಗುತ್ತದೆ.

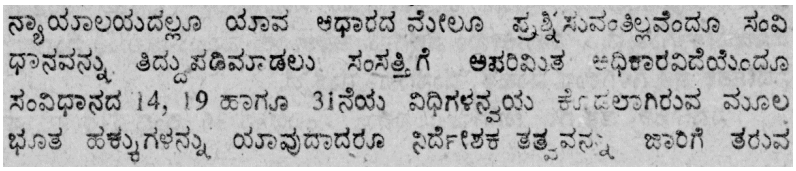
Fig. 8. A small part of the scanned image of a Kannada document with both bold and normal text, and the text recognized by Lipi Gnani. UA = 96.08%, WA = 78.91% for the complete page. The two, red bounding boxes indicate wrongly recognized characters.



" ಅತ್ಯಂತ ಕ್ಲಿಷ್ಟತರದ ಗಣಕ ಯಂತ್ರಗಳನ್ನು ಬಳಸಲು ತರಬೇತಿ ಪಡೆದ
ಸಿದ್ಧಹಸ್ತರ ನೆರವು ಬೇಕಾಗುತ್ತದೆ. ಅದೇ ರೀತಿ ಈ ಜಗತ್ತೆಂಬ ಮಹಾಯಂತ್ರವನ್ನು
ಕೃಷ್ಣನೆಂಬ ಪರಮೋವ್ನತ ಚೈತವ್ಯವೊಂದು ನಡೆಸುತ್ತಿದಯೆಂಬುದು ನಮೂ

Fig. 9. A portion of a Kannada document image with italic text and the output text from the Lipi Gnani OCR, with very few errors. UA = 98.64%, WA = 89.47% for the entire page.

in WA. Table 11 compares the processing times of the two OCRs, on a standard Windows desktop PC (running on Intel i7 Octacore CPU @3.4 GHz with 16 GB RAM), and we can see that Lipi Gnani takes only 1.53 seconds per page, as against Tesseract's 5.46 seconds per page on the same machine.



ನ್ಯಾಯಾಲಯದಲ್ಲೂ ಯಾವ ಆಧಾರದ ಮೇಲೂ ಪ್ರಶ್ನಿಸುವಂತಿಲ್ಲವೆಂದೂ ಸಂವಿ
ಧಾನವನ್ನು ತಿದ್ದುಪಡಿಮಾಡಲು ಸಂಸತ್ತಿಗೆ ಅಪರಿಮಿತ ಅಧಿಕಾರವಿದೆಯೆಂದೂ
ಸಂವಿಧಾನದ 14; 1 9 ಹಾಗೂ 31ನೆಯ ವಿಧಿಗಳನ್ವಯ ಕೊ ಶ್ಲಾಗಿರುವ ಮೂಲ
ಭೂತ ಹಕ್ಕುಗಳನ್ನು ಯಾವುದಾದರೂ ನಿರ್ದೇಶಕ ತತ್ವವನ್ನ ಜಾರಿಗೆ ತರುವ

Fig. 10. A segment of an old, letterpress printed, Kannada document page with severe localized degradation, and the text output by Lipi Gnani with negligible recognition errors. UA = 95.35% and WA = 76.96% for the complete page.

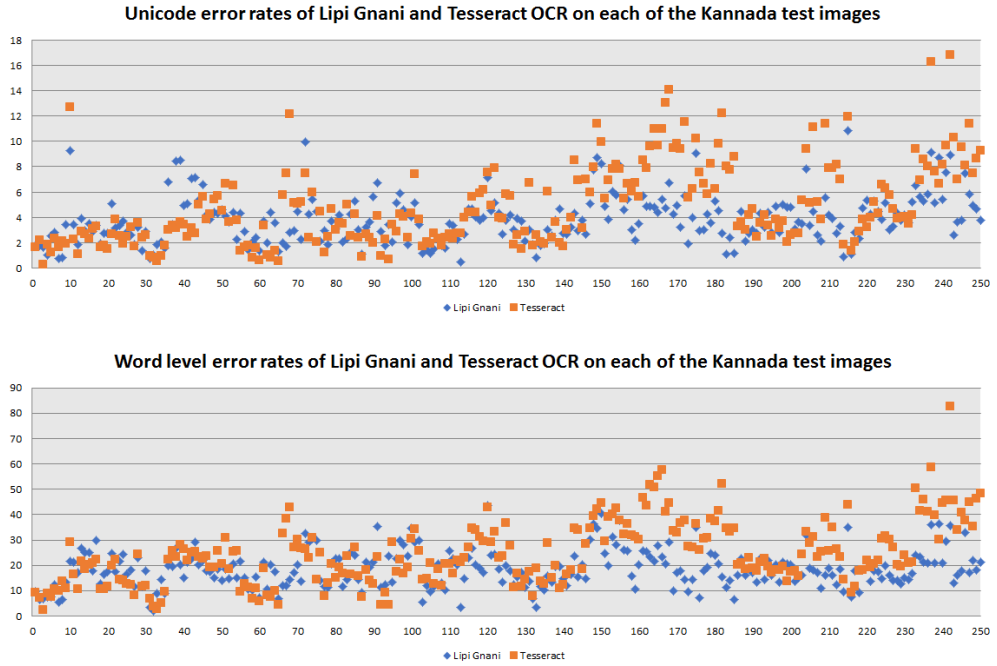


Fig. 11. Comparison of Unicode and word level error rates of Lipi Gnani and Tesseract OCRs on each page of the Kannada benchmarking dataset.

To our pleasant surprise, Lipi Gnani runs as-it-is on Raspberry Pi 4 (with 4 GB RAM and Raspbian OS) and takes an average of 6.2 seconds per page. We have been using this setup to digitize hundreds of Kannada books.

5.3 Results on Konkani, Tulu, and Sanskrit Benchmarking Datasets

Figures 12 to 15 display the recognition results from our OCR for segments from one or two sample pages from each of the non-Kannada language text printed in Kannada. Along with

Table 10. Unicode and word recognition accuracies of Lipi Gnani and Tesseract OCRs on the Kannada test dataset of 250 images. UA, WA: Unicode and word recognition accuracies. N, M: # of Unicodes in the ground-truth and the OCR output, respectively. S, I and D: # of Unicode substitutions, insertions and deletions w.r.t. the ground truth. N_W, M_W: # of words in the ground-truth and the OCR output, respectively. S_W, I_W, D_W: # of word substitutions, insertions and deletions w.r.t. the ground truth. N = 4,07,490; N_W = 48,879.

	UA (%)	M	S	I	D	M_w	S_w	I_w	D_w	WA(%)
Lipi Gnani	**96.25**	4,06,554	7,823	3,252	4,188	48,865	7,520	758	772	**81.48**
Tesseract	**95.41**	4,12,030	8,970	7,140	2,600	49,757	9,725	1,412	534	**76.12**

Table 11. Processing times of Lipi Gnani and Tesseract OCRs on Kannada benchmarking dataset of 250 images, run on a Windows 7 desktop having Intel i7 Octacore CPU @3.4 GHz with 16 GB RAM.

	Total time (250 images)	Average time per page
Lipi Gnani	6 mins 23 secs	1.53 secs
Tesseract	22 mins 45 secs	5.46 secs

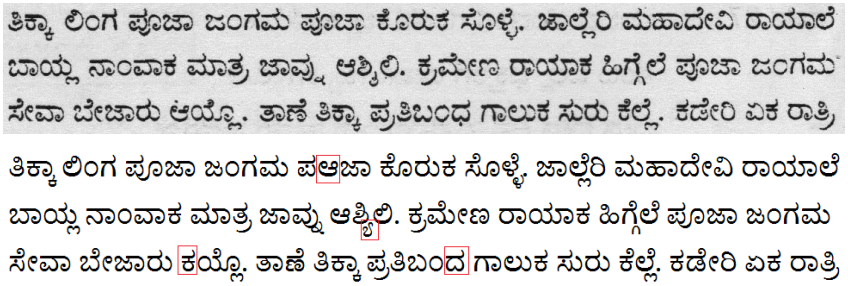


Fig. 12. A segment from the scanned image of a Konkani document and the corresponding text recognized by Lipi Gnani. UA = 94.38% and WA = 79.15% for the full page.

the recognized text, each figure also shows the overall UA and WA achieved on the whole page, from which the segment being displayed arises. A small part of the scanned image of a Konkani document is shown at the top of Fig. 12. Also shown is the text output from our OCR, at the bottom part of the figure. Figure 13 shows a portion of the scanned image of another Konkani printed page. The text recognized by Lipi Gnani OCR, shown on the right, is nearly perfect. Figure 14 shows an example from a Sanskrit printed page. The text recognized by our OCR has only one wrong character (see the bottom of Fig. 14). Another Sanskrit example is shown in Fig. 15. Again, there is a single error in the text recognized by our OCR (see the right hand side of Fig. 15).

Figure 16 compares the performances of Lipi Gnani and Tesseract OCRs on the pages of the other language datasets. Plotted in the two subfigures are the Unicode and word error rates obtained by the two OCRs on each of the test pages. It is clear that Lipi Gnani consistently has lower Unicode and word error rates, except for very few pages.

Table 12 compares the average accuracy of the two OCRs on the three datasets. Though the mean UA for both the OCRs is above 90% on all the datasets, the WA performance of Lipi Gnani is significantly better for Konkani and Tulu, being higher by 23.4% and

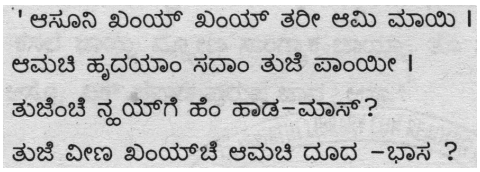


Fig. 13. A part of Konkani document page containing poetic text from another book, and the text recognized by our OCR. UA = 96.11% and WA = 79.26% for the entire page.

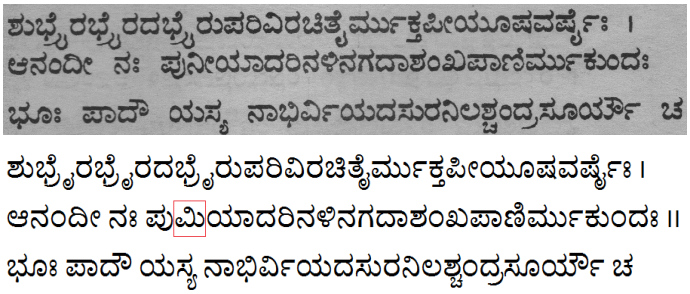


Fig. 14. A snippet from a Sanskrit document and the text recognized by Lipi Gnani. UA = 92.13% and WA = 71.88% for the full page.

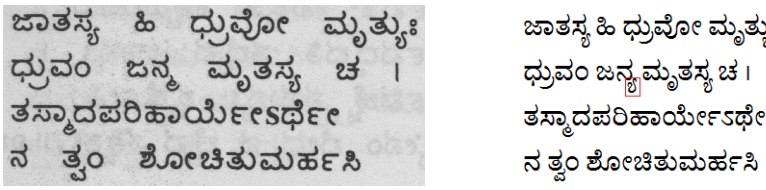


Fig. 15. A snippet from another Sanskrit document and the text recognized by Lipi Gnani. UA = 93.6% and WA = 67.35% for the complete page.

24% respectively, than that of Tesseract. Even for Sanskrit, the increase in word accuracy is more than 8.5%. This shows that a dictionary-based postprocessing or a vocabulary-learning classifier such as LSTM is a disadvantage, when dealing with out-of-vocabulary text, which typically happens with text that is transliterated from another language.

6 CONCLUSION

A full-fledged, Kannada OCR has been developed, with all the image processing routines implemented using Java. This has resulted in a computationally efficient, readily usable OCR on Windows, Linux and Mac OS. The OCR is invoked inside a user-friendly GUI, also developed by us in Java. The GUI has facility to recognize individual scanned pages or all the pages of an entire book. The latter facility was specifically added to help the NGO's create Braille versions of school texts for the use of blind children. Thus, the output text of the OCR can be saved in .rtf, .xml or Braille format. It also has provision to save the recognized Unicode text, and the line and word boundaries in the industry standard METS/ALTO XML [Standards 2015, 2019] format.

This has led to the use of our Lipi Gnani OCR by several NGO's working for the blind as well as by www.kannadapustaka.org, which has made available all the Kannada school books

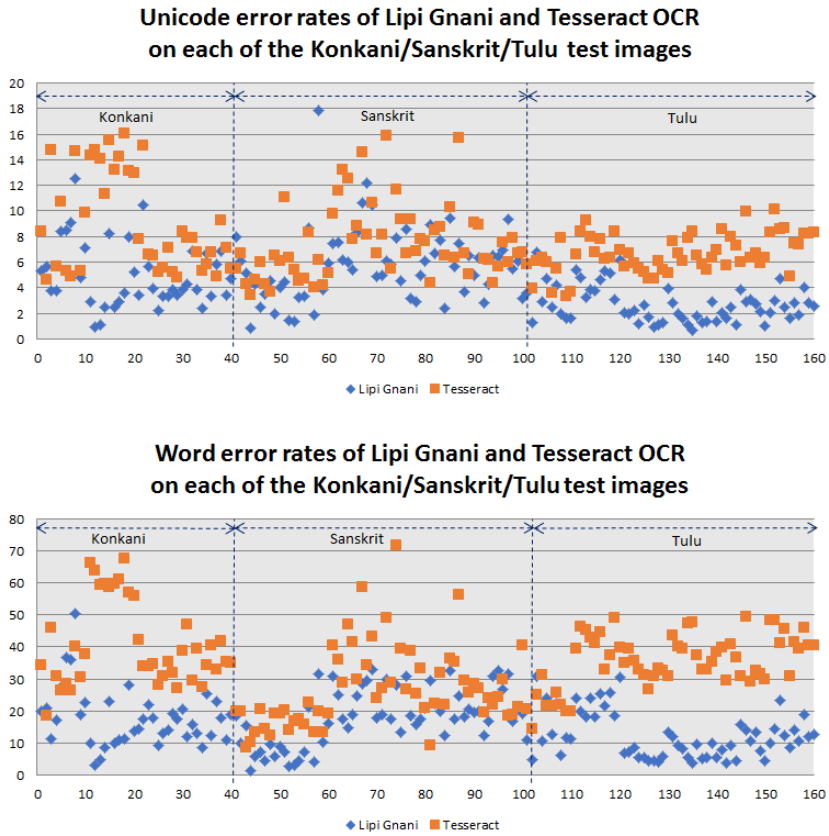


Fig. 16. Comparison of Unicode and word level error rates of Lipi Gnani and Tesseract OCRs on each page of Konkani, Sanskrit and Tulu benchmark datasets.

Table 12. Unicode and word level accuracies of Lipi Gnani and Tesseract OCRs on Konkani (40 pages; N = 45,063; N_W = 6,889), Sanskrit (60 pages; N = 25,196; N_W = 3,606) and Tulu (60 pages; N = 91,023; N_W = 12,372) datasets. All the notations are the same as in Table 10.

Dataset	OCR	UA (%)	M	S	I	D	M_W	S_W	I_W	D_W	WA (%)
Konkani	Lipi Gnani	**94.92**	44,616	1,371	236	683	6,749	1,036	26	166	**82.17**
	Tesseract	**91.23**	45,512	1,774	1,314	865	6,479	2,176	127	537	**58.77**
Sanskrit	Lipi Gnani	**94.13**	24,983	807	230	443	3,618	577	32	20	**82.56**
	Tesseract	**92.36**	25,340	807	631	487	3,685	756	130	51	**74.02**
Tulu	Lipi Gnani	**97.24**	90,511	1,232	384	896	12,277	1,406	10	105	**87.71**
	Tesseract	**93.30**	94,722	1,624	4,089	390	12,387	4,371	65	50	**63.74**

from eighth standard to PUC as both Braille books and audio books. This contribution was also recognized by the Digital Empowerment Foundation in giving us the Manthan Award in 2014 under the category of E-Inclusion & Accessibility [Ramakrishnan and Shiva Kumar

2014]. Together with the PrintToBraille GUI developed by us, it has already been used by some blind students to convert their text books into Unicode text and for reading by a text-to-speech engine. This OCR has now been licensed to RaGaVeRa Indic Technologies by the Indian Institute of Science and is being used to digitize hundreds of Kannada books by another startup, BhaShiNi Digitization Services.

In future, it is intended to incorporate separate dictionaries for each of Kannada, Konkani, Sanskrit and Tulu, and provide the option to the user to select the appropriate dictionary, to obtain the best possible performance for each of the languages.

7 ACKNOWLEDGMENTS

We thank Dr. Jayachandra Shenoy and the Kashi Mutt, Malleswaram, Bangalore for helping us with Konkani books. We thank Mrs. Bhargavi Shivakumar for verifying and correcting the ground truth for all the test images used in this study.

REFERENCES

K.G. Aparna and A.G. Ramakrishnan. 2002. A complete Tamil optical character recognition system. In *Fifth IAPR International Workshop on Document Analysis Systems (DAS 2002)*. Springer-Verlag Berlin, 53–57.

T.V. Ashwin and P.S. Sastry. 2002. A font and size-independent OCR system for printed Kannada documents using support vector machines. *Sadhana* 27, 1 (Feb 2002), 35–58.

Veena Bansal and R.M.K. Sinha. 2000. Integrating knowledge sources in Devanagari text recognition system. *IEEE Trans. on Systems, Man, and Cybernetics-Part A: Systems and Humans* 30, 4 (2000), 500–505.

Veena Bansal and R.M.K. Sinha. 2002. Segmentation of touching and fused Devanagari characters. *Pattern recognition* 35, 4 (2002), 875–893.

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology* 2 (2011). Issue 3. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

B.B. Chaudhuri, O. A. Kumar, and K.V. Ramana. 1991. Automatic generation and recognition of Telugu script characters. *IETE Journal of Research* 37 (1991), 499–511.

Corinna Cortes and Vladimir Vapnik. 1995. Support-Vector Networks. *Machine learning* 20, 3 (1995), 273–297.

D. Dhanya, A.G. Ramakrishnan, and Peeta Basa Pati. 2002. Script identification in printed bilingual documents. *Sadhana* 27, 1 (2002), 73–82.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A Library for Large Linear Classification. *Journal of Machine Learning Research* 9 (2008), 1871–1874.

G. David Forney. 1973. The Viterbi algorithm. *Proc. IEEE* 61, 3 (1973), 268–278.

Venu Govindaraju and Srirangaraj Ranga Setlur. 2009. *Guide to OCR for Indic Scripts: Document Recognition and Retrieval*. Springer Science & Business Media.

Rangachar Kasturi, Lawrence O'gorman, and Venu Govindaraju. 2002. Document image analysis: A primer. *Sadhana* 27, 1 (2002), 3–22.

Aparna Kokku and Srinivasa Chakravarthy. 2009. A Complete OCR System for Tamil Magazine Documents. In *Guide to OCR for Indic Scripts*. Springer, 147–162.

Praveen Krishnan, Naveen Sankaran, Ajeet Kumar Singh, and C.V. Jawahar. 2014. Towards a Robust OCR System for Indic Scripts. In *2014 11th IAPR International Workshop on Document Analysis Systems (DAS 2014)*. IEEE, 141–145.

Gurpreet Singh Lehal and Chandan Singh. 2000. A Gurmukhi script recognition system. In *Proc. 15th International Conf. on Pattern Recognition (ICPR)*, Vol. 2. IEEE, 557–560.

Gurpreet Singh Lehal and Chandan Singh. 2002. A post-processor for Gurmukhi OCR. *Sadhana* 27, 1 (2002), 99–111.

A. Madhavaraj, A.G. Ramakrishnan, H.R. Shiva Kumar, and Nagaraj Bhat. 2014. Improved recognition of aged Kannada documents by effective segmentation of merged characters. In *Signal Processing and Communications (SPCOM), 2014 International Conference on*. IEEE, 1–6.

Minesh Mathew, Ajeet Kumar Singh, and C.V. Jawahar. 2016. Multilingual OCR for Indic scripts. In *12th IAPR Workshop on Document Analysis Systems (DAS 2016)*. IEEE, 186–191.

IISc MILE lab. 2018a. Dataset of Konkani documents printed using Kannada script. (Dec 2018). Retrieved December 30, 2018 from https://github.com/MILE-IISc/KonkaniDocumentsInKannadaScript

IISc MILE lab. 2018b. Dataset of scanned images of Sanskrit text printed using Kannada script. (Dec 2018). Retrieved December 30, 2018 from https://github.com/MILE-IISc/SanskritPagesUsingKannadaScript

IISc MILE lab. 2018c. Dataset of scanned pages of Tulu books. (Dec 2018). Retrieved December 30, 2018 from https://github.com/MILE-IISc/TuluDocuments

IISc MILE lab. 2019. Dataset of 250 Kannada documents carefully chosen to have various kinds of recognition challenges. (May 2019). Retrieved May 28, 2019 from https://github.com/MILE-IISc/Kannada-OCR-test-images-with-ground-truth

P. Nagabhushan and Radhika M. Pai. 1999. Modified region decomposition method and optimal depth decision tree in the recognition of non-uniform sized characters–An experimentation with Kannada characters. *Pattern Recognition Letters* 20, 14 (1999), 1467–1475.

Premkumar S. Natarajan, Ehry MacRostie, and Michael Decerbo. 2005. The BBN Byblos Hindi OCR system. In *Document Recognition and Retrieval XII (DRR 2005)*, Vol. 5676. SPIE, 10–17.

N.V. Neeba, Anoop Namboodiri, C.V. Jawahar, and P.J. Narayanan. 2009. Recognition of Malayalam Documents. In *Guide to OCR for Indic Scripts*. Springer, 125–146.

Atul Negi, Chakravarthy Bhagvati, and B. Krishna. 2001. An OCR system for Telugu. In *Proceedings of Sixth International Conference on Document Analysis and Recognition (ICDAR 2001)*. IEEE, 1110–1114.

Nobuyuki Otsu. 1979. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics* 9, 1 (1979), 62–66.

Umapada Pal and B.B. Chaudhuri. 1994. OCR in Bangla: an Indo-Bangladeshi language. In *Proceedings of the 12th IAPR International Conference on Pattern Recognition*.

Peeta Basa Pati and A. G. Ramakrishnan. 2000. Machine recognition of printed Odiya text. *Master's Thesis, Department of EE, Indian Institute of Science, Bangalore* (2000).

Peeta Basa Pati, A. G. Ramakrishnan, and U.K.A. Rao. 2000. Machine recognition of printed Odiya characters. In *Proc. III International Conference on Information Technology (ICIT 2000)*.

S.N.S. Rajasekaran and B.L. Deekshatulu. 1977. Recognition of printed Telugu characters. *Computer graphics and image processing* 6, 4 (1977), 335–360.

A.G. Ramakrishnan and H.R. Shiva Kumar. 2014. Manthan Award 2014. *E-Inclusion & Accessibilty - Winner* (2014). http://manthanaward.org/e-inclusion-accessibilty-winner-2014/

Tony M. Rath and Rudrapatna Manmatha. 2007. Word spotting for historical documents. *International Journal of Document Analysis and Recognition (IJDAR)* 9, 2-4 (2007), 139–152.

H.R. Shiva Kumar, A. Madhavaraj, and A.G. Ramakrishnan. 2019. Splitting Merged Characters of Kannada Benchmark Dataset using Simplified Paired-Valleys and L-Cut. In *Proc. 25th National Conference on Communication*.

H.R. Shiva Kumar and A.G. Ramakrishnan. 2019. Gamma Enhanced Binarization - An Adaptive Nonlinear Enhancement of Degraded Word Images for Improved Recognition of Split Characters. In *Proc. 25th National Conference on Communication*.

R.M.K. Sinha and H.N. Mahabala. 1979. Machine recognition of Devanagari script. *IEEE Trans. on Systems, Man, and Cybernetics* 9, 8 (1979), 535–441.

Ray Smith. 2016. Building a Multilingual OCR Engine. (Jun 2016). Retrieved June 20, 2016 from https://github.com/tesseract-ocr/docs/blob/master/das_tutorial2016/7Building%20a%20Multi-Lingual%20OCR%20Engine.pdf

Sargur N Srihari and Venugopal Govindaraju. 1989. Analysis of textual images using the Hough transform. *Machine vision and Applications* 2, 3 (1989), 141–153.

The Library of Congress Standards. 2015. METS: Metadata Encoding & Transmission Standard. (Jan 2015). Retrieved June 19, 2019 from http://www.loc.gov/standards/mets/

The Library of Congress Standards. 2019. ALTO: Technical Metadata for Layout and Text Objects. (May 2019). Retrieved June 19, 2019 from http://www.loc.gov/standards/alto/

Changming Sun and Deyi Si. 1997. Skew and slant correction for document images using gradient direction. In *Proceedings of the Fourth International Conference on Document Analysis and Recognition*, Vol. 1. IEEE, 142–146.

Tesseract. 2018. Tesseract Manual. (Dec 2018). Retrieved December 15, 2018 from https://github.com/tesseract-ocr/tesseract/blob/master/doc/tesseract.1.asc

Consortium Unicode. 2018. The Unicode Standard v11.0 U0C80. (Jun 2018). Retrieved June 5, 2018 from https://unicode.org/charts/PDF/U0C80.pdf

B. Vijay Kumar and A.G. Ramakrishnan. 2002. Machine recognition of printed Kannada text. In *Fifth IAPR International Workshop on Document Analysis Systems (DAS-2002)*, Vol. 5. Springer Verlag, Berlin, 37 – 48.

B. Vijay Kumar and A.G. Ramakrishnan. 2004. Radial basis function and subspace approach for printed Kannada text recognition. In *International Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, Vol. 5. IEEE, 321 – 324.

Robert A Wagner and Michael J Fischer. 1974. The string-to-string correction problem. *Journal of the ACM (JACM)* 21, 1 (1974), 168–173.

Nick White. 2014. uzn format. (Aug 2014). Retrieved August 21, 2014 from https://github.com/OpenGreekAndLatin/greek-dev/wiki/uzn-format

Wikipedia. 2018a. Konkani language. (Oct 2018). Retrieved October 3, 2018 from https://en.wikipedia.org/wiki/Konkani_language

Wikipedia. 2018b. Tulu language. (Oct 2018). Retrieved October 3, 2018 from https://en.wikipedia.org/wiki/Tulu_language

Received December 2018