# Tools for Developing OCRs for Indian Scripts

M N S S K Pavan Kumar, S S Ravikiran, Abhishek Nayani, C V Jawahar, P J Narayanan
Centre for Visual Information Technology
International Institute of Information Technology
Gachibowli, Hyderabad 500019, INDIA

## Abstract

Development of OCRs for Indian script is an active area of activity today. Indian scripts present great challenges to an OCR designer due to the large number of letters in the alphabet, the sophisticated ways in which they combine, and the complicated graphemes they result in. The problem is compounded by the unstructured manner in which popular fonts are designed. There is a lot of common structure in the different Indian scripts. In this paper, we argue that a number of automatic and semi-automatic tools can ease the development of recognizers for new font styles and new scripts. We discuss briefly three such tools we developed and show how they have helped build new OCRs. An integrated approach to the design of OCRs for all Indian scripts has great benefits. We are building OCRs for all Indian languages following this approach as part of a system to provide tools to create content in them.

## 1. Introduction

A lot of activity is taking place today on the development of OCRs for the scripts of Indian languages [1, 2, 3, 4, 5, 6]. Electronic representation and access of documents and other material in Indian languages have been lagging behind for many years. The recent wide adoption of the computer and the channels provided by Information Technology in India has resulted in a big spurt in the use of electronic representation, storage, and access of information in Indian languages. The demand for content in the local languages is ballooning as Information Technology reaches beyond the English-speaking sections of the populace. This trend is likely to continue in a populous country like India. A similar trend is expected in other developing nations also. An Optical Character Reader is a critical tool in creating electronic content in Indian languages as a rich tradition of literature and a vibrant publishing industry have existed in all of them for a long time.

Indian languages and the scripts used to write them pose great challenges to an OCR designer. Four factors make the task considerably harder than the design of an OCR for the Roman script: (a) the larger number of letters in the alphabet, the typical number being 50, (b) richer structure of the basic unit in the language as the combination of upto 3 consonants and 1 vowel forms the basic unit, (c) the variations in the graphical form of the different combinations even within the same script, and (d) non-adherence to any standard structure while designing the fonts.

All Indian languages have a common alphabet derived from the Sanskrit alphabet. The scripts used to express the alphabet differ widely from language to language, however. The code for electronic representation of Indian languages, called Indian Standard Code for Information Interchange or ISCII, exploits this feature. Thus, ISCII is a compact and script-independent code for all Indian languages. This design of ISCII has been adopted by the Unicode representation of Indian languages in an expanded form. ISCII representation is expressed in different scripts using script-dependent drivers for each language. This is a feature that can reduce the complexity of designing an OCR for a new script by bootstrapping from an existing OCR system. This aspect has not been explicitly recognized by the designers of OCRs in Indian scripts.

We are developing a system containing OCRs for all major Indian scripts. The goals of the overall project is to provide electronic content in all Indian languages. A unified approach to the individual tools such as OCRs for all languages is necessary for this. Thus, our design naturally provides a framework to exploit the commonality as well as differences between different Indian languages and their scripts. Most other efforts towards building Indian language OCRs have focussed on single language/script, making it difficult to recognize inter-lingual structures. [7, 8, 9, 10, 11, 12]

In this paper, we argue that automatic and semi-automatic tools can aid the design of an OCR for a new script or a new font family, given an OCR for another script or font family. We built a number of tools for the generation of extensive test data, to analyze the effectiveness of an existing segmentation scheme, to judge the classification accuracy of an OCR, etc. The tools perform extensive analysis on the graphical form and the ground truth in combination. The results are presented in a form that makes it

easy for the human designer to adjust its various parameters. This iterative procedure for OCR design has helped us build OCRs tuned for new font sizes and new scripts considerably quickly. We present two case studies here.

Section 2 summarizes the features of Indian languages and scripts that have bearings on OCR design for them. Section 3 describes two of the tools we developed for the bootstrapped design of OCRs. Section 4 presents the case studies of designing an OCR for a new font style in Devanagari and for a new script Telugu, starting with an OCR for a specific Devanagari script. Section 5 presents some concluding remarks.

# 2. Features of Indian Languages and Scripts

We look at some of the features of Indian languages and the scripts used to express them that can affect the way OCRs are designed for them.

**Common Alphabet:** All Indian languages have essentially the same alphabet derived from the Sanskrit alphabet. This common alphabet contains 33 consonants and 15 vowels in common practice. Additional 3-4 consonants and 2-3 vowels are used in specific languages or in the classical forms of others. This difference is not very significant in practice. Individual consonants and vowels form the basic letters of the alphabet. The most notable exception to the common alphabet is the southern language of Tamil, which uses about 12 fewer consonants. However, the structure is not too different in Tamil also, as the change can be modelled as dropping some of the consonants from the master list.

The now standard Indian Standard Code for Information Interchange (ISCII) takes explicit cognizance of the commonality in the alphabet. The same design has also been adopted by Unicode for Indian languages.

**Basic Character Unit: Akshara** Indian languages also have a more sophisticated notion of a character unit or *akshara* that forms the fundamental linguistic unit, akin to a character in English. An akshara consists of 0, 1, 2, or 3 consonants and a vowel. Words are made up of one or more aksharas. Each akshara can be pronounced independently as the languages are completely phonetic. Aksharas with more than one consonants are called *samyuktaksharas* or combo-characters. The last of the consonants is the main one in a samyuktakshara.

The ISCII standard recognizes and encodes this structure of an akshara. Thus, the basic representational unit in ISCII can range in length from 1 to 6 bytes. A string of bytes of encoded text in ISCII can easily be split into its constituent aksharas uniquely.

**Different Graphemes** The commonality in the alphabet does not extend the graphic forms used to express them in print. Each language uses different scripts consisting of dissimilar graphemes for printing. Thus, printed matter in other scripts are inaccessible to readers of one script. There are 10-12 major scripts in India. The Devanagari script is the widest used one, being used to write Hindi (the most spoken language), Marathi, Konkani, and Nepali, the language of the neighbouring Nepal. Different scripts use different philosophies for the individual graphemes and their combinations. Some have a head-line or *shirorekha* that persists for a whole word. Others have non-touching graphemes. The grapheme of one of the consonants is usually at the heart of the printed akshara. The vowel appears as a *matra* or vowel modifier. These can appear to the left, right, above or below it or in combinations. The supporting consonants of a samyuktashara also appear as modifier graphemes to the left, right, above, or below of the main one. These modifiers could be truncated or scaled down forms of the basic consonant, but could also be completely different. They may touch each other or the main consonant in some cases or may be separate. These rules are not consistent even within a script and certainly not across scripts.

The script and font dependent aspects of a language are pushed to a language driver responsible for printing ISCII strings in a specific script and font. Different scripts use different drivers. Transliterating a block of text from one language to another involves sending it through another driver.

**Unstructured Font Design** Different fonts have been designed for each Indian script in the past few years. The fonts are built from glyphs and follow the graphical structure of each script, which is different for different languages. It is not possible to use a consistent set of rules for this step for all scripts. Unfortunately, no conventions have been followed in defining the glyphs for different font families of the same script, which is quite possible. To increase the confusion, the computer fonts have been defined for specific purposes (for instance, each electronic newspaper defines its own font for its electronic site). They do not come with converters from ISCII to their glyph set. Since the computer usage in local languages is abysmally low, the standards encoded in ISCII have not been adopted widely enough by the different players such as the individual newspapers.

## 2.1. OCR Design for Indian Scripts

Machine reading of printed or hand-written characters through an OCR has to map the glyphs as it appears on paper to the component graphemes and ultimately to aksharas represented in ISCII. The above factors make design of OCRs for Indian scripts tedious and manual effort intensive. Much of the work done for one font family has to be repeated for another of the same script. Sometimes, many

changes have to be made for a font of larger or smaller point size within the same family. It would be advantageous to reuse the work done for one script when designing an OCR for another.

An example will illustrate this point. The akshara `tri` is composed of the consonants `ta`, `ra` and the vowel `e`. The combination of `ta` and `ra` in the Devanagari script assumes a form that is totally different from either of them. In Telugu and Malayalam scripts, a modifier-grapheme corresponding to `ra` appears to the left of the grapheme for `ta`. The matra for the vowel `e` in Devanagari appears on the left of the grapheme of the consonant, in Telugu above it, and in Malayalam to the right of it!

It must be noted that human readers have no difficulty in reading a complicated expression of such a rich alphabet structure even when multiple scripts are mixed. Such mixing is quite common in printing, in India.

# 3. Bootstrapping Tools for OCRs

We have embarked on a project to produce processing resources in multiple Indian languages. The goal of the project are to create tools for building a repository of electronic documents with proper indexing in all major Indian languages. Such resources are not available today partly because the computer processing in these languages are not very popular. English dominates the network connected and computer aware world in India, although the population that is literate in English is an extremely small fraction.

A critical component in our project is a reliable OCR system in the languages of interest. The OCRs need to be font size and style independent to be effective on a wide range of documents. This task necessitates the reuse of as much of the work done for one script and one font to build OCRs for other scripts and fonts. We believe automatic and semi-automatic tools can make the task of building a new OCR easier.

Development of any OCR system involves 5 major stages: data collection, segmentation, feature selection, training and testing, and postprocessing. Internal representation formats, display technologies, font conversion, etc., are also crucial for building a complete working system. Each of these stages involve a lot of font-dependent, language-dependent parameters, especially in Indian languages. The design of completely generalized procedures to solve these problems may not be possible. We look at some of the problems now and discuss how appropriate tools can help.

## 3.1. Analysis of Script Characteristics

A detailed analysis of the characteristics of the script is the first step in the design of an OCR for a new font or script. This is tied to the script segmentation module. The results of the analysis will be used for data collection, training, testing and the final OCR system. Script analysis is difficult in Indian languages as explained in Section 2. Script analysis contains two major steps. The first is the selection of the segmentation algorithm and its parameters for accurate segmentation of the characters. The second is the construction of the required mapping tables between labels, glyphs and aksharas. We look at these steps now. The relationships between different representations of the data from ISCII to the image and back are given in Figure 3.2.

The script analysis tool works on an electronically generated image of a document, containing each valid glyph of the script written once. Since the alphabet is the same for all Indian languages, all possible combinations of aksharas expressed in ISCII can be the contents of this document. However, half a million aksharas are possible (33 * 33 * 33 * 15), but most are improbable. The document containing all aksharas of interest has to be prepared with manual supervision. Another alternative is to pick the aksharas from a representative corpus in the target language. Most of the aksharas of interest are same in all languages. Therefore a document in ISCII can serve most of the needs in all languages. This document can be rendered using a specific language and font driver to get the electronic image for analyzing the script features.

The segments generated by the segmentation need not be the same as the glyphs in the font/script typically. For example, segmentation of Devanagari script has the removal of the shirorekha using the projection profile as the first step. The akshara `ki` gets segmented into three segments, where as, the akshara is made up of only two graphemes in Devanagari – the vowel-modifier for `e` and the consonant `ka`. Only these two glyphs are used by all the fonts. Thus, a map between the segments generated by the segmentation algorithm and the corresponding script components needs to be built.

The script components or graphemes identified by the above process needs to be mapped to the aksharas represented in ISCII. This requires another set of mapping tables. Thus, final text output in ISCII can be provided by the combination of these mapping tables.

These tables are usually generated manually because of the complexity of script features. However, it is an intensive task since the number of possible grapheme combinations could be very large.

We also have the ground truth in terms of font specific glyphs since the image is electronically generated. Segmentation could split each glyph into one or more segments that are recognized by a classifier. It is necessary to label each segment using a unique number. These segments may be shared by other glyphs since they are based on the image alone. For example, one of the segments the matra (vowel modifier) for `e` splits into in Devanagari is a vertical line

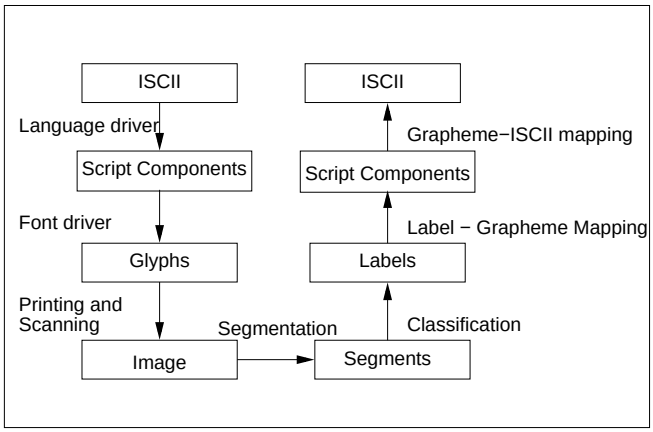


Figure 1: The steps involved in going from ISCII to print and back

which is also a segment in the matra for the long aa as well as part of characters like ga, Na. This makes the segment to script-component mapping complex.

Effective tools can ease this task. For instance, the segments generated for a new glyph can be compared with already labelled segments using an appropriate image matching scheme. Close ones can be shown to the user graphically for final selection. Tools do not automate this critical process, but can ease the manual task and improve the speed of building the segment to script component mapping.

## 3.2. Data Collection

Large amounts of data with sufficient variations are required to train an OCR that should work under different conditions. These variations can be on size, font type, scanning resolution, etc. All expected inputs to the system should be covered. Creating and collecting data for such conditions is a time consuming and tiresome task.

The block diagram of the data collection tool is shown in Figure 3.2. A document containing all graphemes present in a script of a language is created electronically. This is possible by using a font driver. We refer the set of graphemes as the standard grapheme set. A structured document containing a predetermined (large) number of each grapheme in a separate page can be generated as an image by rendering the required characters using a language driver. For example, a training page might contain 100 samples of each character possible in Devanagari in a known sequence. This can be generated for different font styles and sizes automatically. These images can have unique labels (as a bar-code, for instance) on the page for easy identification. The training documents so generated can be printed on a printer to get hard-copy versions of the data set.

The printed data set can be scanned in batch mode under the control of the tool to get images for training and

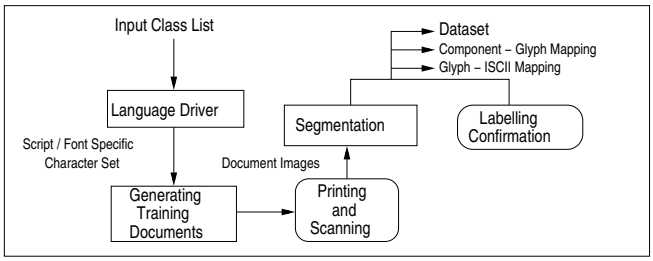


Figure 2: Block diagram of the data collection tool

testing. The scanning can be performed at different resolutions to get more variations. These scanned pages are then subjected to preprocessing algorithms like noise reduction, skew correction and thresholding The preprinted label helps identify the page. The ground truth for the page is then available from tables created while generating and printing these documents. Thus, we get a large number of data for training or testing with realistic images and ground truth.

The last step performed by the data collector is the segmentation of the image using the pre-decided segmentation procedure for the script. The ground truth information helps in labelling each segmented image with the class ID, the font style and size, and the scanning resolution. The results of the segmentation are displayed nicely for quick manual inspection for verification. This is to guard against the program missing an odd grapheme, throwing the ground truth sequence into disarray.

The data collection tool, thus, creates labelled images of all the graphemes in a script with the manual intervention kept to a minimum of verifying the segmentation visually.

## 3.3. Segmentation Analysis and Feature Selection

Selection of the segmentation strategy, features, and classifier are very important steps in the construction of an OCR. Semi-automatic tools can play an important role in these steps also. We built a segmentation analysis tool that helps tune the segmentation parameters. The block diagram of the tool is shown in Figure 3.3. This tool starts with ground truthed pages such as the ones created by the Data Collection Tool. The images generated by the tool are segmented using a segmentation procedure for another font of the same script. The OCR for the other font is applied on the segmented data. The results of classification are compared with the ground truth data and the discrepancy or error is displayed graphically for manual verification. Based on the results, the segmentation parameters can be tuned manually and the OCR process can be repeated. If no segmentation gives adequate results, manual verification might indicate that the input class list, adopted from the font for which an OCR is available, is inadequate for this font. A new class

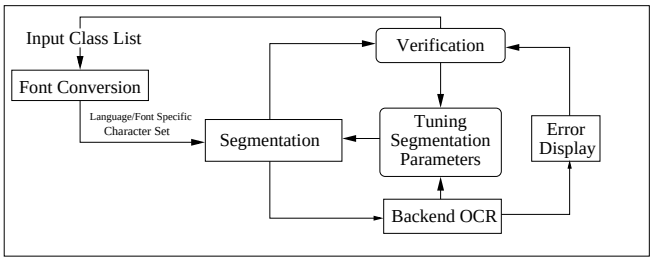


Figure 3: Block diagram of the segmentation analysis tool

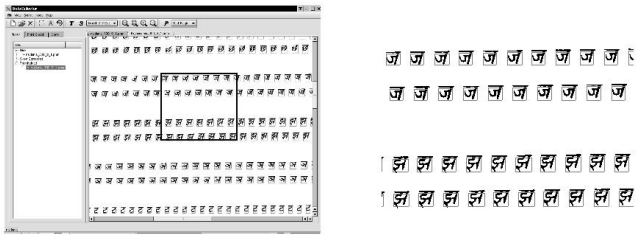

Figure 4: Data collector at work. The image on the right is a zoomed version of the selected region.

can be added manually to the list and the whole process can be repeated.

A similar tool can be built to select the best features semi-automatically also. The block diagram remains the same with a difference that the efficacy of a set of features is being tested and not the segmentation. The availability of a large amount of ground truth data and the human in the loop iterative tuning of the features used help select an optimal set of features very rapidly.

### 3.4. Automatic Training and Testing Tool

The labelled output of the Data Collector can be used for automatic training and testing. The parameters that can be changed for batch mode training were the feature set, classifier, the percentages of the data set to be used for training and testing, mixing the data set for training and testing etc. The results reported in our ICDAR paper [2] for different combinations of the above were generated using this tool. The options for the feature set currently are PCAs, scaled images, and discriminant vectors. The classifiers available are SVM and kNN. The percentage of the samples to be used for training could be varied from 20% to 100%. The results of this tool include the accuracies obtained, the confusion matrix, etc.

## 4. Results

The tools described above ease the process of building an OCR for a new script or a new font. We present two experiences as results.

A Devanagari OCR for a font called Naidunia, is extended to an OCR that can handle the same font of smaller sizes. It was observed that the problem could not be solved just by selecting shape independent features, as the segmentation algorithm must be more robust for smaller size fonts, as the edge errors and noise matter a lot in these cases. Also, redesigning the OCR to handle both the fonts, was equally easy with the help of the tools available.

### 4.1 Extension of an OCR to a Smaller Font

The following section describes the extension of an existing Devanagari OCR tuned for 24pt Naidunia font to handle a size of 16pt of the same. The segmentation, feature selection, etc., are already available for the larger size. Most of the problems for smaller size are in segmentation of the components.

An electronically-generated image containing all glyphs in 16pt is generated using a tool as described earlier from such a document for the font. This is the initial document for this script. The segmentation analysis tool is used on this image with the OCR for the 24pt font used as the backend. The segmentation algorithm does not work perfectly due to the problems introduced in smaller size print. Some components start touching each other in the lower size. The higher relative importance of noise makes segmentation less effective. The segmentation analysis tool brings out such discrepancies graphically to the user and allows for tuning of the segmentation parameters. The segmentation algorithm used for this purpose was a mixture of projection profiling [12] and region growing, both of which have font and size dependent parameters. The task of tuning the segmentation for the new font size was completed in less than 30 minutes using the tool.

The training data for 16 point was generated using the data collector tool. Training data for 137 components, with 100 samples for each component, was generated as described earlier. Extraction of 13,700 components for training took just 3 hours of time using the data collector tool. A screenshot of the tool is shown in figure 4.1.

The data collected was put to extensive testing using the Training and Testing Tool described earlier using SVM and kNN classifiers, and PCA, Discriminant Vectors, and images as features. The Support Vector Machine classifier with scaled image as the feature worked best for the OCR. The training time was approximately 30 minutes of system time. This process of extension resulted in an OCR with classification accuracy of 85% when the backend OCR had an accuracy of 94%. Repeated use of classification analysis with the ground truth improved the accuracy to 94% in a couple of iterations. The segmentation thresholds and pa-

rameters were changed accordingly and the algorithm was soon tuned to handle both the fonts with same efficiency.

### 4.2 Building a Telugu OCR

We used many of the tools described earlier to develop an OCR for the Telugu script. All possible script-components in Telugu were identified from the ISCII class list using a language driver for ISCII. The segmentation analyzer was run with the segmentation algorithm not removing the shirorekha but using projection profiles. The number of disjoint classes was found using the segment analysis tool to be around 420. The Data Collection tool was used to generate a large number of data, to the tune of 40000 components. These were generated synthetically as documents, printed, scanned and thresholded. The collected data is verified quickly for correctness with a graphical interface. The data-collector facilitates renumbering of such samples and removal of noisy ones. Once the necessary corrections are made, the components are saved as images.

The data thus obtained was used for feature extraction, training and testing. In the feature selection, dimensionality reduction was performed using PCA analysis and the reduced dimension vectors were used as feature vectors. This is necessary to ensure a reasonable speed for the classifier as the number of classes is high. This set of features, with a SVM classifier, gave an initial accuracy of 90%. Iteration using the ground truthing tool improved it to about 97semi-automatic tools developed is demonstrated by the drastic reduction in the data-collection, training and recognition times. The construction of the OCR with the 33600 segments took 6 man hours to collect and about 90 minutes for training.

## 5. Conclusions

In this paper, we presented the unique problems of designing OCRs for Indian scripts. We also described the concept and design of a couple of semi-automatic tools that will help with the process, bootstrapping using existing OCRs or segmentation algorithms wherever possible. The main goal of designing such tools is to reduce the tedium of manual processes and to ensure very good results. We presented the Data Collection tool that can quickly create thousands of sets of data with ground truth, the Segmentation Analysis tool that enables quick tuning of the segmentation algorithm to a new script given one for another, and the Feature Selection tool that makes it possible to tune the features used for classification for a particular font or script given a set for another. We also presented how we could create an OCR for a different font size for Devanagari and for the Telugu script using these tools. There are a number of other tools that would ease the process of making OCRs in many languages, a task we are currently involved with. We are in the process of generating some of them currently.

## References


[1] "Document image processing of Indian scripts," *Special Issue of Sadhana*, 2002.

[2] C. V. Jawahar, M. N. S. S. K. Pavan Kumar and S. S. Ravi Kiran, "A bilingual OCR for hindi-telugu documents and its applications," in *International Conference on Document Analysis and Recognition*, 2003.

[3] C. V. Jawahar, M. N. S. S. K. Pavan Kumar, and S. S. Ravi Kiran, "Recognition of Indian Language Characters using Support Vectors Machines," *Technical Report TR-CVIT-22, International Institute of Information Technology,Hyderabad*, 2002.

[4] V. Bansal, "Integrating knowledge sources in devanagari text recognition," doctoral thesis, IIT Kanpur, Department of Computer Science and Engineering, March 1999.

[5] V. Bansal and R.M.K.Sinha, "A devanagari OCR and a brief overview of OCR research for Indian scripts,"

[6] "http://www.cedar.buffalo.edu/ilt/research.htm." Home Page Denanagari OCR.

[7] B. B. Chaudhuri and U. Pal, "An OCR system to read two Indian language scripts: Bangla and devnagari (hindi)," *in Proc of ICDAR*, pp. 1011–1015, 1997.

[8] A. Negi, Chakravarthy Bhagvathi, and B. Krishna, "An OCR system for telugu," in *Int. Conf. Document Analysis and Recognition (ICDAR)*, 2001.

[9] T.V.Ashwin and P.S.Sastry, "A font and size-independent OCR system for printed kannada documents using support vector machines," *Sadhana*, vol. 27, pp. 35–58, February 2002.

[10] B.B.Chaudhuri, U.Pal, and M.Mitra, "Automatic recognition of printed oriya script," *Sadhana*, vol. 27, pp. 23–34, February 2002.

[11] G. S. Lehal and C. Singh, "A gurmukhi script recognition system," in *Proccedings of International Conference in Pattern Recognition,Barcelona,Spain*, vol. 2, pp. 557–560, 2000.

[12] U. Pal and B. B. Chaudhuri, "Automatic separation of words in multi-lingual multi-script Indian document," in *Proceedings of International Conference on Document Analysis and Recognition (ICDAR)*, pp. 576–583, 1997.