An Effort Towards OCR From Ancient Indian Manuscripts Using Grid Search

[1]Anu Pallavi R, [2]Dr Chandrashekara S N

[1]M.Tech, [2]Professor and HOD

[1, 2] Dept. of Computer Science & Engineering, C. Byregowda Institute of Technology, Srinivasapur Road, Kolar, Karnataka, India.

Abstract : Ancient Indian scripts are a perfect evidence of the advanced languages and their grammarian rules existed tens of thousands of years. Though most of these Indian languages are still in use today, the scripts of these languages had undergone a significant change over these years due to the introduction of variations and local slangs, resulting in new children languages. While this being the case, the ancient Indians had taken immense trouble in preserving the history and culture through engravings on walls and through writings on palm leaves, which were traditionally passed on to next generations. These scriptures now are deteriorated either through the years of exposure to natural elements, or through human invasions of carelessness, in the process loosing huge ancient knowledge being lost forever. Understanding the value of this, academia in recent times started to digitalize these data into digital format, predominantly as images. To extract the characters from these scripts has been a tough work since the OCR techniques are inefficient towards such images. Thus, an approach is proposed in this paper for multilingual OCR of characters and retrieve from these ancient scripts. This paper proposes an image processing technique to retrieve the font pixels and removes the noise using Gaussian filter. And the characters are identified using a grid search approach using a weighted dataset

IndexTerms **- Indian language OCR, ancient script, character recognition.**

I. INTRODUCTION

Indian languages are among the earliest perfect scripture languages. The earliest scriptures of Indian languages date back to days before 12000 BCE. While the dates are still under debate, the scriptures themselves indicate the times of thousands of years before the last sea level rise, in about 7000 BCE. These languages, the early Devanagari and the early Dravidian are considered parent of other Indian languages. Thus, a majority of the current day languages take its root to these few ancient languages.

The scholars of the ancient civilizations had used stones and processed palm leaves as a medium to engrave their scriptures upon, owing to the fact that these can survive for a long time. While the monuments stood the test of time, they had been ruined due to the various invasions made by the kings who invaded India during the last few centuries. What remained after the invasions, are currently being vandalized by local people and tourist when they found to remove these stone monuments to construct their houses, or when they seem to damage the engravings by scribbling over them.

While the monuments are being destroyed this way, the palm leaves also started to face the deterioration due to the mishandling of these delicate fragment of leaves. Recent studies indicate that only twenty eight percent of the ancient scripts are existing today, while all the other literature are lost to time.

Recently the archaeological department and the research scholars took to digitalization of the remaining contents, so as to preserve these them for future studies. These digital copies are generally stored in the form of images which are either taken of these monuments or the scanned images of the palm leaves. These digital copies then require a huge manual effort to extract the text content, and many OCR technologies that are currently available fail to retrieve the contents.

Indian languages poses a unique set of challenges for the OCR to be performed. Each Indian languages has anywhere between forty to sixty-five alphabets which includes the vowels and consonants, while another two hundred can be formed when each character is conjoint with the each vowel. Thus the dataset for each Indian language grows profoundly. This also adds up to the other main complication is that the characters can be a complex set of connected curves, where, a single character can have multiple disconnected curve to represent the syllable, or a multiple-character conjunction can have a single set of connected curves, as shown below, where joining of three characters forms one set of connected curves.

श् + र् + ई = श्री

Thus, there is no predefined rule of how the character might shape after conjunction, like how the rules are fixed in newer languages like English. Thus, the dataset needs to include each of these characters and their mix.

Predominantly, current Indian languages follow two main categories, the Devanagari, where each character has a header line, or the Dravidian, where the characters are purely curve based. Each word in Devanagari are in turn joined to each other through these header lines, making the character splitting also a huge challenge.

Thus the commercially available OCR techniques are not efficient towards the detection of Indian language, and the few available OCR are purely language dependent and do not support cross scripting style.

Thus, an effort is made here to provide a common and an effective method to retrieve the characters for all Indian languages.

II. RELATED WORK

OCR for Indian script started in the late 1960s when Sinha and Mahabala produced their research [1] provided a pattern analysis for the Devanagari system to perform OCR to detect the embedded characters from an image using structural description using primitives and their relation with the neighboring characters. An advanced algorithm for recognition of multilingual Indian document images was given in [2] which was based on Devanagari scripts. Accodingly, the primary segmentations are obtained using structural property of characters and are then applied with the principles of graph theory to create character segmentations. These are then run with over a SVM classifier for validation. Then three features are retrieved based on the center pixel of character

and its neighboring pixel. These are then fed into a k-NN classifier that uses the databases containing printed as well as handwritten texts to perform OCR. However this does not support the conjunction of characters and does not support the Dravidian languages.

An adaptive model for OCR was proposed in [3] which mainly focused on the offline character recognition of Chinese handwritten homologous image. The recognition is first done to retrieve the bigram model for each character. These are then matched adaptively with the database. The contents are again passed through a third iteration of best fit match on topic language model. The algorithm was tested on over a hundred Chinese images and the results were published. This method has improved over the earlier OCR for Chinese, but still rendered inefficient, as the Chinese script are similar in complexity as Indian scripts.

These methods provide a basic features and recent advances in the field of OCR towards complex scripting languages. But the efficiency of the algorithms are tightly coupled with the processing technique to remove the noises in the input image. Recent improvement to the computational efficiency and the GPU the processing techniques for image has become more powerful and efficient.

Nityananda et al [4] provided an advanced algorithm on the image quality enhancement. Accordingly, image quality was enhanced for better visibility of features to be able to differentiate using human eyes. The resulted image is an output from the equalization algorithm which uses histogram as the basis of reference to shift the contrast range.

Moore neighborhood [5] is outlined on a two-dimensional sq. lattice and consists of a central cell and therefore the eight cells that surround it. it's one in every of the 2 most typically used neighborhood varieties, the opposite one being the von Neumann neighborhood. The renowned Conway's Game of Life, as an example, uses the Moore neighborhood. it's just like the notion of 8-connected pixels in camera work. the concept behind the formulation of Moore neighborhood is to search out the contour of a given graph. this concept was a good challenge for many analysts of the eighteenth century, Associate in Nursingd as a result an algorithmic program was derived from the Moore graph that was later known as the Moore Neighborhood algorithmic program.

III. PROPOSED WORK

This paper starts with the assumption that the ancient scripts are basically handwritten text but the font and background are not in conjuncture with the current handwritten contents. This requires processing the image such that the image is converted to a black and white format and run the detection technique over it. Hence, the process is divided in two categories.

3.1 Pre-Processing

This stage works predominantly on the image processing techniques. It is important to get this step right as this Some of the processing techniques are listed below.

3.1.1 Rasterizing

The current day cameras come in all sort of image sensors and the image color scheme. In addition, the image sometimes are rendered to be ineffective for image reading. Thus, the image first needs to be rasterized. That means the image first needs to be converted into a system recognizable color scheme like sRGB Adobe color.
This step would convert the image into a raster image, which is a matrix data of dot or line pixels arranged in a rectangular pattern of colored pixels. The rasterization is performed using a code as shown in below pseudo code.

```
foreach (pixel ← image)
    Ray R = RayTrace(pixel);
    float nearest = infinity;
    Triangle nearestTriangle;
    foreach (triangle ← scene)
      float hit;
      if (intersect(R, object, hit))
        if (hit < closest)
              nearestTriangle = triangle
    if (nearestTriangle != null)
       NewPixel= ColorAtHit(nearestTriangle, nearest);
```

3.1.2 Monochrome

Monochrome is termed to an image which is represented only by one color. This paper uses Gray as the monochrome color. That means the image after the monochrome algorithm (3.1) is applied will only have two colors, white and a wide range of gray. The white indicates the pixel with full luminosity and the darkest gray would be the absence of color.

$$Gray{:}\,Y \leftarrow 0.299 * \mathrm{R} + 0.587 * \mathrm{G} + 0.114 * \mathrm{B} \qquad (3.1)$$

Where R, G, and B represents the color value of Red, Green and Blue.

The colors of each pixel is passed through this equation which results in the monochrome value of that color. The color is set back to the same pixel. This would render the pixel in the a shade of gray. When all the pixels is run through the same steps the resultant image will be a black and white image.

3.1.3 Filters

Few filters are applied over the monochrome obtained in previous step. The main purpose is to make the image devoid of any unwanted noises and render the image in a medium regularity of the color shift.

First, the image is passed through a contrast correction filter [6], this would correct the image from any defect due to incorrect lighting in the image. The core equation behind the contrast correction (3.2) forms a linear regression.

$$A = A1 + (\mathrm{A2} - \mathrm{A1})/(\mathrm{G2} - \mathrm{G1})[\mathrm{G} - \mathrm{G1}] \qquad (3.2)$$

Where G is, the gray pixel values and the A are the desired level of contrast.

3.1.4 Gradient

This involves splitting the foreground and background from an image. Since different images requires a different threshold value, the current peak value is first calculated using the swarm technique. The gradient value returned from the swarm localization technique indicates the localized areas with its max gray gradient. An average of the localized max gradient is calculated and 80% of the max value is then used to determine the gradient threshold to retrieve the foreground.

3.1.5 Segmentation

The image thus coming from the previous filters may still contain small blotch of pixels arising from the damaged portion of the pictured object. These blotches are termed as noise and needs to be removed from the image to prevent any inefficiency that might result from these. To detect these, the image undergoes Prewitt edge detection. The detected edges are categorized into two, the closed loop connected edges and the open unconnected edges. The closed loop edges are then run with Moores algorithm, which calculates the bounding box of the detected edges. Based on the bounding rectangles the edges spread area are detected, and the smaller area based edges, and the open edges are masked from the input image. This step leaves with only bigger, connected and closed-set of edges. It is upon these images the detection is performed.

3.2 Character Detection

While the first step focuses predominantly on the processing of the image. This step runs the OCR algorithm over the image.

3.2.1 Dataset

The detection is performed using a grid search approach of the template images that are used in the training dataset creation. Opposing to the approaches in some of the earlier researches, this paper provides the weightage to each data item in the training dataset an integer that indicates the complexity of the shape, rather than the order of the character found in the dataset. For instance, the character (र) Ra and (स) Sa falls at the twenty-seventh and thirty-second consonant respectively, of the Devanagari script. And, the character (सु) Su comes when the consonant Sa conjoins with the vowel U. If an OCR is conducted over the image as in fig 3.1a, The character detection of Ra, Sa, and Su will all have a hundred percent match for the first character as shown in 3.1b, 3.1c and 3.1d.

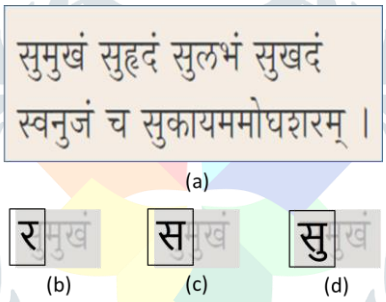

Fig. 3.1. Grid search without prioritized dataset

Systematic risk is the only independent variable for the CAPM and inflation, interest rate, oil prices and exchange rate are the independent variables for APT model.

From the above figure, it can be noted that all the three characters give a 100% match, but only the last character is the required output. Thus, the traditional approach of running the search according to their occurrence in the alphabets does not yield the required result.

To get the required output, a weighted dataset is recommended. Accordingly, the each dataset item is provided with a number to indicate the shape complexity of the character. According to that, the character Su will have higher weightage compared to Sa which is higher to Ra. The data architect who prepares the data for training dataset assigns this weightage. And the data is trained over this set of dataset, which typically contains the template image, its encoded value and the weightage.

When this dataset is imported, the template are sorted according to their weightage and the image with higher weight is first used to detect the OCR

3.2.1 Grid Search

For each template in the dataset, a search is performed on the source image. To do this, the template is run with the Moore's boundary detection algorithm. This would provide a tight fit rectangle for each of the template image. This rectangle provides the parameters required for the grid size composition.

Each grid pixels are then run through a histogram generation and all the histograms are collected and compared with the histogram of the template image. The histogram overlap percentage determines the probability of the template to be matched in a particular grid. If the percentage is more than the threshold value, the grid is considered to represent the character of the template. The detected character is saved in an array, and the image is masked to remove the detected grid. This would ensure that other characters following the current template character are not detecting the same grid.

Once the search is completed with all the dataset templates, or if all no more grid exists to detect, these collected grids along with their encoded value is sorted according to their position along X and Y axis of a graph, from left to right, top to bottom. This would put the characters in their respective word and line sequence.

This method also accommodates scaling. This means, that the grid size before histogram generation are scaled, so that a bigger or smaller grid is placed and the histogram is compared. Since this method does not depend on the language as a factor,

this method works for any language of any country. And can also be used to detect the hieroglyphics or pictograms from ancient paintings.

IV. RESULTS AND DISCUSSION

The algorithms and methodology thus proposed was coded in C-sharp. Emgu wrapper to the OpenCV [7] library was used for image processing. The performance was tested on a single thread in CPU. A future enhancement can be to use the GPU of the system to run the algorithm.

4.1 Test case 1

The test case one involves recognizing the characters from a handwritten Tamil script. This test is basically performed to test the effectiveness of the machine learning algorithm to perform the matching. Since any script, either from palm leaves or from temple wall, is typically a handwritten script, with just a difference in the medium it was written.

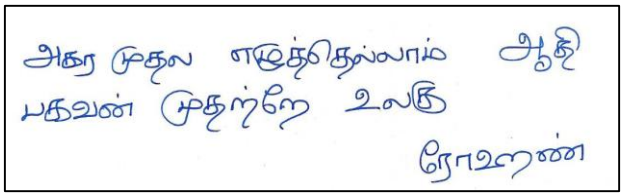

Fig. 4.1. Grid search without prioritized dataset

The above test case was run successfully with all characters detected

4.2 Test case 2

The next test is an acceptance test to ensure the foresaid algorithms work in a tandem to provide a complete OCR for an ancient script. To test this, an image of old Sanskrit script from Mahabharata was passed through the algorithm. The application took eighty seconds to detect the hundred and forty-nine characters, missing three. The results were in the same order and sequence of the input script, as shown in fig 5. This test case had out the efficiency to 98%. The training dataset for the above test had over four hundred and-fifty images that were given appropriate weights.

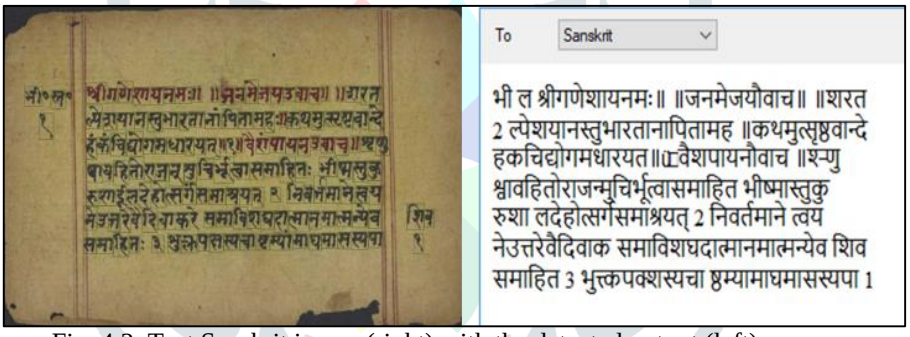

Fig. 4.2. Test Sanskrit image (right) with the detected output (left)

Another test was performed on a Kannada script, containing about a hundred and fifteen characters. The code ran with a success rate of 94% which recognized a hundred and nine of them.

V. RESULTS AND DISCUSSION

The paper provides one solution for multiple Indian languages and is found efficient to detect the character conjunction among vowels and consonants. The dataset containing the weighted images ensures that the recognition happens in the right order and the characters are not miss-detected. The histogram search along with Moore's boundary detection helps in determining the right scaling for default run, while still supporting any scaling of the image. The image processing ensures that the image provided for the OCR, is cleaned up before recognition.

Though a higher accuracy is obtained, the time taken for the detection is one of the factor that can be worked upon in the future. Since the number of characters in a typical Indian language is almost about two-hundred and fifty, still excluding consonant-consonant juncture, the dataset required even for few fonts and hand written can easily go high. To improve the effectiveness in terms of time, the grids can be run in multiple threads and if possible, using GPU. Once this is achieved, the algorithm can be used to detect the language automatically and run OCR over it.

REFERENCES

[1] R. M. K. Sinha and H. N. Mahabala, "Machine Recognition of Devanagari Script," IEEE Transactions on Systems, Man, and Cybernetics, Vol. 8, 1979, pp. 435-441.

[2] Sahare, Parul & Dhok, Sanjay. (2018). "Multilingual Character Segmentation and Recognition Schemes for Indian Document Images". IEEE Access. PP. 1-1. 10.1109/ACCESS.2018.2795104.

[3] Wang, Yanwei & Ding, Xiaoqing & Liu, Changsong. (2014). "Topic Language Model Adaption for Recognition of Homologous Offline Handwritten Chinese Text Image. Signal Processing Letters", IEEE. 21. 550-553. 10.1109/LSP.2014.2308572

[4] Nithyananda CR, Ramachandra C & Preethi "Surve on Histogram Equalization method based Image Enhancement techniques" International Conference on Data Mining and advanced Computing, Ernakulam, 2016, pp. 150-158

[5] http://www.imageprocessingplace.com/downloads_V3/root_downloads/tutorials/contour_tracing_Abeer_George_Ghuneim/moore.html

[6] S. Mori, et al., "Research on machine recognition of handprinted characters", IEEE Trans. Pattern Anal. Mach. Intell. 6, 4 (1984) 386–405

[7] https://docs.opencv.org/