Parallel Processing in OCR – A Multithreaded approach

G L Jayavardhana Rama, A G Ramakrishnan and Devaraj Gupta
Department of Electrical Engineering,
Indian Institute of Science, Bangalore – 560012, INDIA.

1. Introduction

Document Image processing has been a frontline research area in the field of human-machine interface for the last few decades. Optical character recognition (OCR) of Indian language scripts has been a topic of interest for quite some time. The earlier contributions were reported in [1] and [2]. More recent works are reported in [3] and [4]. The need for efficient and robust algorithms and systems for recognition is being felt in India, especially in the postal department for sorting mail and for preserving out-of-print old books by digitising them. Character recognition can also form a part in applications like intelligent scanning machines, text to speech converters, and automatic language-to-language translators. Very few OCRs are available commercially for Indian languages. Most of these systems are font-specific and require the knowledge of the font *a priori* in some way. Some of the systems available for Tamil were presented in earlier conferences [5][6][7]. The above software follow sequential processing of the methodologies as shown **in figure 1.**

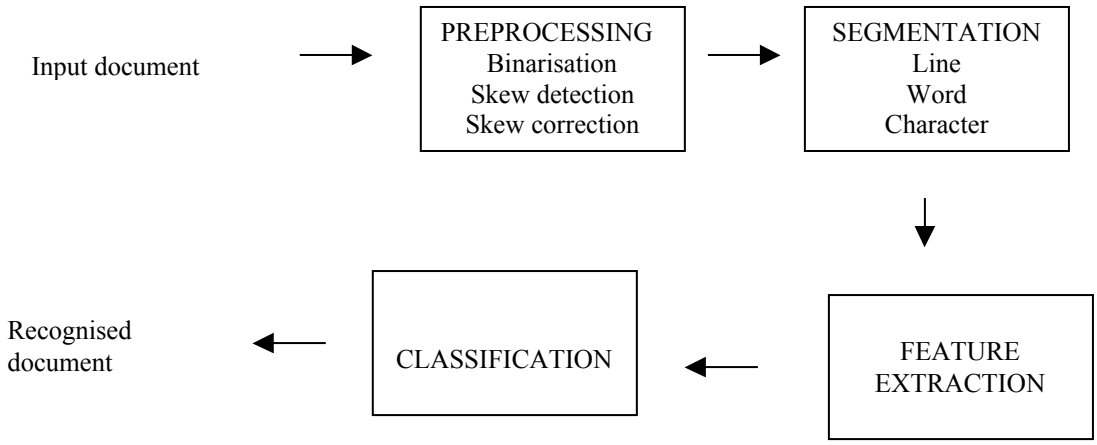

Figure 1: Block diagram of existing OCR Systems

From here onwards, we limit our discussion to Tamil Gnaani [6], which works reasonably well for any font. The sequence of steps followed in this system is explained below. Firstly, when we run the software, the database is loaded into the buffer followed by the display of GUI. The user is allowed to choose whether he wants to scan or he likes to run OCR on an already available document image. Then the file is read, and preliminary binarization, skew detection and correction are performed. Advanced binarization technique based on 3 sigma point [9] is then applied. The binarized document is then subjected to segmentation, feature extraction and classification. The recognized document in TAB format is displayed in the integrated text editor. The whole process from running the software to display of the result takes about 80 secs for an A4 sized document in a P3 machine. The idle time of the user is about 80 secs. To reduce the idle time of the user, we propose new modifications in the process flow of an OCR system. The technique uses parallel processing for the processes for which the data required do not overlap.

2. Multi-threading and Tamil Gnaani

Parallel processing is a term used to denote a large class of techniques that are used to handle multiple data-processing tasks simultaneously for the purpose of increasing the computational speed of a system. Instead of processing each instruction sequentially, a parallel processing system is able to perform concurrent data processing to achieve faster execution time. This is achieved using multithreaded programming. Multithreaded programming allows writing programs that do many things simultaneously.

A multithreaded program contains two or more parts that can run concurrently [8]. Each such part of a program is called a thread. A thread is a path of execution within a process. Each thread defines a separate path of execution. A process consists of one or more threads and the code, data, and other resources of a program in memory. Each thread in a process operates independently. Unless we make them visible to each other, the threads execute individually and are unaware of the other threads in the process. Threads sharing common resources, however, must coordinate their work by using semaphores or another method of inter-process communication. We use inter-process communication efficiently so that the race over problem is avoided. The flow chart representation of the OCR system employing multithreading is shown in **Fig. 2**. Here the reading of the database which is a worker thread (WT) [8] is processed in parallel with selection of a file which is a user interface (UI) thread [8]. When threshold is being arrived at, another WT for skew detection is employed. The odd and even lines are processed in parallel i.e. when second line is being recognised in a WT, another UI thread is used for displaying the recognised first line thus reducing the user idle time.

3. Conclusion

By adopting the above strategy, the idle time of the user decreases from 80 secs to 40 secs. The user is able to edit the recognised text using another thread while recognition is being performed on rest of the page. Use of dictionary in the same process facilitates in increasing the recognition accuracy.

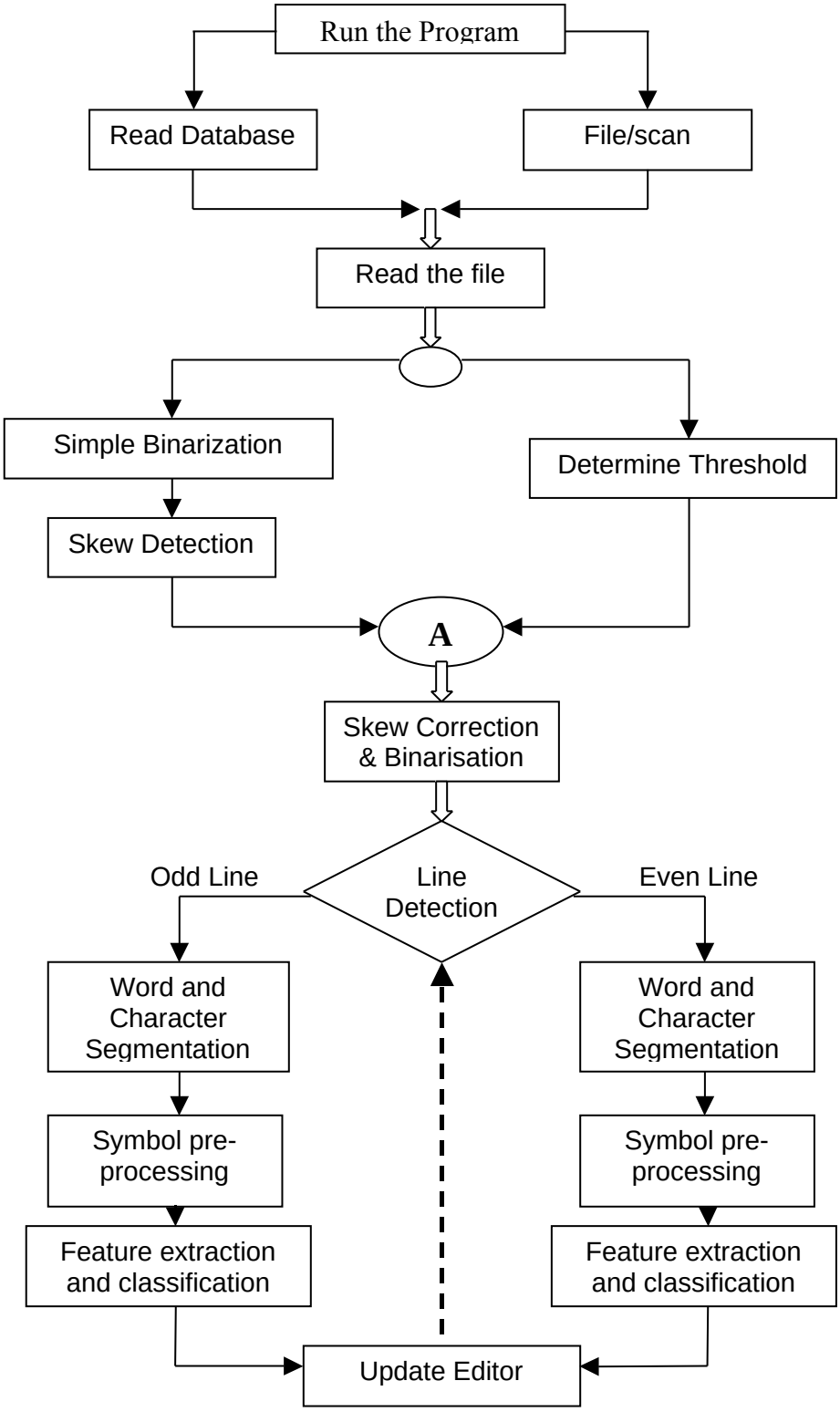

Figure 2: Flow chart of the proposed scheme

4. Reference

[1] Siromoney, G., Chandrashekaran, R., Chandrashekaran, M.: Computer recognition of printed Tamil characters, Vol. 10. Pattern Recognition, (1978) 243-247

[2] Sinha, R.M.K., Mahabala, H.: Computer recognition of printed Devnagari scripts, Vol. 9. IEEE trans. on Systems Man and Cybernetics, (1979) 435-441

[3] Pal, U., Choudhuri, B.B.: A Complete Printed Bangla OCR System, Vol. 31. Pattern Recognition, (1998)

[4] Govindan, V.K., Shivaprasad, A.P.: Character recognition – a review, Pattern Recognition (1990) 671-683

[5] K Mahata and A G Ramakrishnan (2000). A complete OCR for printed Tamil text. *Proc. Tamil Internet 2000, Singapore (July 22-24, 2000),* 165-170.

[6] K G Aparna and A G Ramakrishnan (2001). Tamil Gnaani - An OCR for windows. *Proc. Tamil Internet 2001, Kuala Lumpur (Aug 26-28, 2001)*, 60-63.

[7] V Krishnamoorthy (2001). On Developing OCR software for Tamil. *Proc. Tamil Internet 2001, Kuala Lumpur (Aug 26-28, 2001)*, 69-71.

[8] Jim Beveridge and Robert Weiner, Multithreading Applications in Win32, Addison Wesley,1998.

[9] D Dhanya and A G Ramakrishnan (2001). Simultaneous Recognition of Tamil and Roman Scripts. *Proc. Tamil Internet 2001, Kuala Lumpur (Aug 26-28, 2001)*, 64-68.