Parallel Implementation of Devanagari Text Line and Word Segmentation Approach on GPU

Brijmohan Singh
Department of CSE,
College of Engineering Roorkee,
Roorkee-247667, Uttarakhand,
India

Nitin Gupta, Rashi Tyagi, Ankush Mittal
Department of CSE,
College of Engineering Roorkee, Roorkee-247667,
Uttarakhand, India

Debashish Ghosh
Department of E&C,
IIT Roorkee,
Roorkee-247667, Uttarakhand, India

ABSTRACT

Fast and accurate algorithms are necessary for Optical Character Recognition (OCR) systems to perform operations on document images such as pre-processing, segmentation, feature extraction, training and testing of classifiers and post processing. Text line and word segmentation are two important steps in any OCR system. Wrong segmentation may affect the accuracy rate of OCR systems. The segmentation is very challenging in cases of availability of different types of noises, degradations, and variation in writing and script characteristics. However, existing algorithms suffer from a flawed tradeoff between accuracy and speed. In this research work, Devanagri text line and word segmentation are carried out using modified standard profiling based segmentation approach and parallelized it on Graphics Processing Unit (GPU). The main goal of this research work is to make segmentation faster for processing a large number of document images using parallel implementation of algorithms on GPU. GPUs are emerging as powerful parallel systems at a cheaper cost. Our work employs extensive usage of highly multithreaded architecture and shared memory of multi-cored GPU. An efficient use of shared memory is required to optimize parallel reduction in Compute Unified Device Architecture (CUDA). Experimental results show that our method can achieve a speedup of about 20x-30x over the serial implementation when running on a GPU named GeForce 9500 GT having 32 cores.

General Terms

Document Analysis and Recognition, Pattern Recognition, Parallel Computing.

Keywords

OCR; Segmentation; Profiling; Parallelization; GPU; CUDA

1. INTRODUCTION

Research on Devanagari [1] character [2-10] and word [11-13] recognition is very difficult due to its challenging properties. This area of research is still open for further research due to the extent of variation among writing styles, speed, thickness of character and direction of different writers. Real-world handwriting is a mixture of cursive and non-cursive parts, which makes the problem of recognition and synthesis more difficult. Similar looking characters may give ambiguity, characters segment may touch where they should not or vice versa, variations and noises may get introduced during scanning and continuously increasing demand for accuracy, fast recognition, cheap and more practical approach to implement recognition system.

A well-defined dataset plus well problem oriented distinct algorithms of OCR help to provide a feasible solution to the accuracy and speedup problem. Text line and word segmentation are two important steps of OCR system.

Wrong segmentation can affect the accuracy rate of any script OCR system. The segmentation is very challenging in the cases of availability of different types of noises, degradations, writing and script characteristics. In handwritten document processing, half of the errors are due to segmentation. Segmentation of Devanagari document image is challenging due to touching components, wide variety of handwriting styles, location of line and word boundaries, overlapping of characters, identification of physical gaps between words and characters. The segmentation of Devanagari is difficult due to following problems: There are about 350 basic, modified (matra) and compound character shapes in the script, the characters in a word are topologically connected, shirorekha makes all characters connected in a word , matra comes along with the consonants, occurrence of bindu, conjoined words (yuktashera) and lack of robust handwritten recognition algorithm. The applications [14-16] of OCR systems are automatic form processing, automatic mail sorting, bank checks processing, and office automation for text entry.

Some elaborate studies on text line, word and character segmentation are in [17-22]. The profiling techniques (horizontal profiling, for text line segmentation and vertical profiling for the word segmentation) were proposed by [23-24]. Other techniques such as classifier based segmentation: Character segmentation-by-recognition using log-gabor filters [25] and Recognition of Devanagari characters using a hierarchical binary decision tree classifier [26], Gap clustering techniques [27], Graph based segmentation [28] were also be adopted. Kompalli et al. [29] was designed and compared segmentation driven and recognition driven Devanagari OCR.

In this work, we modified the traditional profiling based segmentation method to extract text lines and words from Devanagari script document images and parallelized it to make faster using CUDA.

In the following sections, we present a detailed description of the proposed methodology as well as experimental results that demonstrate the efficiency of the proposed methodology.

2. INTRODCTION TO nVIDIA CUDA ARCHITECTURE

nVIDIA® CUDA™ is a general purpose parallel computing architecture introduced by nVIDIA. It includes the CUDA Instruction Set Architecture (ISA) and the parallel compute engine in the GPU. C language is used to program to the CUDA architecture. One of the most widely used high-level programming languages, which can then be run with great performance on a CUDA enabled processor [30]. CUDA-enabled GPUs have hundreds of cores that can collectively run thousands of computing threads. Each core has shared resources, including registers and memory. The on-chip shared memory allows parallel tasks running on these cores to share data without sending it over the system memory bus [31].

A fundamental building block of CUDA programs is the CUDA kernel function. When launching a CUDA kernel function, a developer specifies how many copies of it to run. We call each of these copies a task. Because of the hardware support of the GPU, each of these tasks can be small, and the developer can queue hundreds of thousands of them for execution at once. These tasks are organized in a two-level hierarchy, block and grid. Small sets of tightly coupled tasks are grouped into blocks. In a given execution of a CUDA kernel function, all blocks contain the same number of tasks. The tasks in a block run concurrently and can easily communicate with each other, which enables useful optimizations such as those of the section "Shared Memory". GPU's hardware keeps multiple blocks in flight at once, with no guarantees about their relative execution order. As a result, synchronization between blocks is difficult. The set of all blocks run during the execution of a CUDA kernel function is called a grid.

The three key abstractions of CUDA are the thread hierarchy, shared memories and barrier synchronization, which render it as only an extension of C. All the GPU threads run the same code and, are very light weight and have a low creation overhead. A kernel can be executed by a one dimensional or two dimensional grids of multiple equally-shaped thread blocks. A thread block is a 3, 2 or 1-dimensional group of threads as shown in Figure 1. Threads within a block can cooperate among themselves by sharing data through some shared memory and synchronizing their execution to coordinate memory accesses. Threads in different blocks cannot cooperate and each block can execute in any order relative to other blocks. The number of threads per block is therefore restricted by the limited memory resources of a processor core. In current GPUs, a thread block may contain up to 512 threads. The multiprocessor SIMT (Single Instruction Multiple Threads) unit creates, manages, schedules, and executes threads in groups of 32 parallel threads called warps.

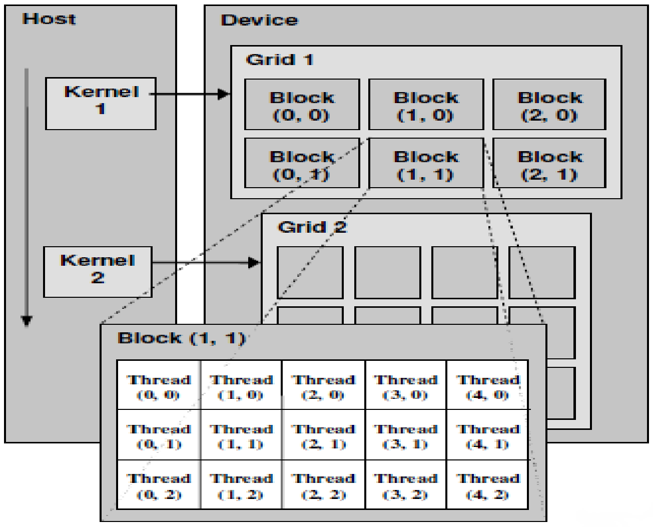


Fig. 1: Thread Hierarchy in CUDA

3. PROPOSED METHODOLOGY

Segmentation process consists of two distinct stages. In the first stage, a preliminary segmentation is performed that executes text line segmentation. This process leads to the isolation of sub-images corresponding to each text line of the complete text. These sub-images contain more than one word. In the second stage, we perform word segmentation which results into the sub-images which consists of each word as a separate image. Now these separate images of words can be further used for recognition in OCR systems.

3.1 Line Segmentation

Segmentation is very important part of the character recognition process which extracts text lines and words out of the document images. The text document as a whole is of no use due to the limitation of feature extraction or classification phases. So we need to extract each word out of the document images for the use of feature extraction phase. In our approach we are using profiling based method which uses the vertical and horizontal density of black pixels along an axis. A vertical plot of pixel density is shown in figure 2

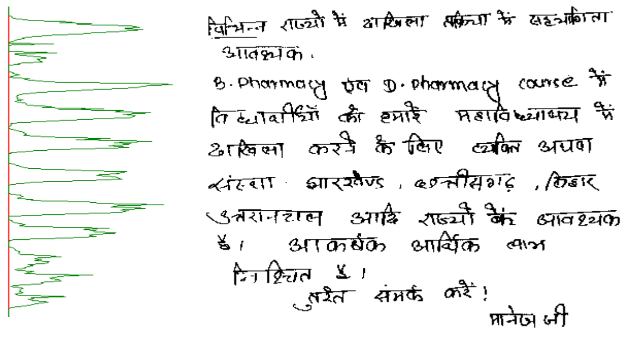

विभिन्न राज्यों में दाखिला प्रक्रिया के सहभागिता
आवश्यक।
B. Pharmacy एवं D. Pharmacy course में
विद्यार्थियों को हमारे महाविद्यालय में
दाखिला करने के लिए व्यक्ति अथवा
संस्था झारखंड, छत्तीसगढ़, बिहार
उत्तरांचल आदि राज्यों के आवश्यक
है। आकर्षक आर्थिक लाभ
निश्चित है।
तुरंत संपर्क करें!
मनोज जी

Fig. 2: Vertical plot of pixel density

The minima's are now identified into the plot with the condition that they are just after or before a maxima, now we use these minima's to crop the document image to get separate text lines.

3.2 Word Segmentation

In the similar way as in text line segmentation, in word segmentation we find the horizontal plot of density of pixels seeking for minima's as inter word space. We crop the image using these minima's and the final output is an isolated words. The horizontal plot of pixel density is shown in figure 3.

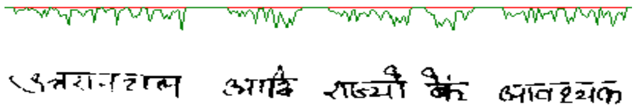


Fig. 3: Horizontal plot of pixel density

4. IMPLEMENTATION

In this research work, the implementation of proposed approach is based on the two set of experiments. In the first set of experiment, proposed algorithm is implemented in C language and in second set; proposed algorithm is parallelized using CUDA. The following sections 4.1 and 4.2 dictate the detailed description of the sequential and parallel implementation of proposed algorithm.

4.1 Sequential Implementation

The following pseudo codes (algorithm 1) outline the structure of proposed algorithm for text line segmentation written in C language:

Algorithm 1: Proposed approach for text line segmentation

```
Input-   2 Dimensional Image
Output- Segmented Text Lines
Subroutine main()
        Calculate Threshold value
        For each row within all row of matrix
            Calculate the density(image)
        End For
        For each row within all row of matrix
            If (Density[j] ==0 && (Density[j - 1] > 0 || Density[j + 1] > 0))
                Then minima[i]=j;
            Check_Minima(minima)
            While(minimaLength !=null)
                Calculate height between two adjacent minima and create a new image of required width.
                Save the image.
            End While
        End For
End Subroutine
Function Density()
        If Intensity is less than threshold
            Then increment the density for each row
End Function
Function Check_Minima()
        Calculate difference b/w two minima
        Then apply Modify()
End Function
Function Modify()
    Calculate average of all the vertical intensities and divide the whole part of this image in to two half. And compare upper part with upper image intensity and lower part with lower image intensity.
    If both parts of image match
        Then embed it into their respective parts
    If one parts of image match
        Then embed matched part with respective part and divide the unmatched part further and repeat above steps till all rows are not checked
End Function
```

The following pseudo codes (algorithm 2) outline the structure of proposed algorithm for word separation from the text line.

Algorithm 2: Proposed approach for word Segmentation

```
Input – Segmented Text Lines (sub-images)
Output – Segmented Words
Subroutine main()
        Calculate Threshold value
        For each column within all column of matrix
            Calculate the density(image)
        End For
        For each column within all column of matrix
            If (Density[j] ==0 && (Density[j - 1] > 0 || Density[j + 1] > 0))
                Then minima[i]=j;
            Check_Minima(minima)
            While(minimaLength !=null)
                Calculate height between two adjacent minima and create a new image of required width.
                Save the image.
            End While
        End For
End Subroutine
Function Density()
        If Intensity is less than threshold
            Then increment the density for each row
End Function
Function Check_Minima()
        Calculate difference b/w two minima
        Then apply Modify()
End Function
Function Modify()
    Calculate average of all the horizontal intensities and divide the whole part of this image in to two half. And compare upper part with upper image intensity and lower
```

```
part with lower image intensity.
    If both parts of image match
        Then embed it into their respective parts
    If one parts of image match
        Then embed matched part with respective part and
divide the unmatched part further and repeat above steps
till all columns are not checked
End Function
```

4.2 Parallel Implementation

In CUDA, it is assumed that both host and device maintain their own DRAM. Host memory is allocated using malloc and device memory is allocated using cudaMalloc. CUDA threads are assigned a unique thread ID that identifies its location within the thread, block and grid. This provides a natural way to invoke computation across the image, by using the thread IDs for addressing. The parallel implementation of proposed algorithm of text line segmentation is shown in the pseudo code (algorithm 3) shown below.

Algorithm 3: Parallel Implementation of proposed algorithm for text line segmentation

```
Input– 2 Dimensional Image
Output – Segmented Text Lines
 Subroutine main()
      Define a block and grid
      For each row in all rows
        Call Density_Kernel(Image)
      End For
      For each horizontal row (j)
           If (Density[j] ==0 && (Density[j - 1] > 0 ||
         Density[j + 1] > 0))
                    Then minima[i]=j;
              Modify_Kernel()
          While(minimaLength !=null)
            Calculate height between two rows
          Call Output_ Kernels(image)
              End While
          End For
 End Subroutine
Density_kernel(image )
    If Intensity is less than threshold
          Then increment the density for each row
End Kernel
Output_kenel(image)
   Use to create an image
End Kernel
Modify_kernel()
Calculate average of all the vertical intensities and divide
the whole part of this image in to two half. And compare
upper part with upper image intensity and lower part with
lower image intensity.
    If both parts of image match
        Then embed it into their respective parts
    If one parts of image match
        Then embed matched part with respective part and
divide the unmatched part further and repeat above steps
till all rows are not checked
End Kernel
```

The following pseudo codes (algorithm 4) outline the structure of proposed parallel algorithm for word extraction from text line.

Algorithm 4: Parallel Implementation of proposed algorithm for text line segmentation

```
Input– Segmented Text Line (2 Dimensional Sub-image)
Output – Segmented words
Subroutine main()
    Define a block and grid
    For each column in all columns
        Call Density_Kernel(Image)
     End For
      For each horizontal row (j)
             If (Density[j] ==0 &&
         (Density[j - 1] > 0 || Density[j + 1]
         > 0))
                Then minima[i]=j;
          While(minimaLength !=null)
              Calculate height between two
         rows
             Call Output_ Kernels(image)
            End While
         End For
End Subroutine
Density_kernel(image )
        If Intensity is less than threshold
             Then increment the density for
         each row
End Kernel
Output_kenel(image)
          Use to create an image
End Kernel
Modify_kernel()
 Calculate average of all the horizontal intensities and
divide the whole part of this image in to two half. And
compare upper part with upper image intensity and lower
part with lower image intensity.
    If both parts of image match
       Then embed it into their respective parts
     If one parts of image match
        Then embed matched part with respective part and
divide the unmatched part further and repeat above steps
till all columns are not checked
End Kernel
```

5. HARDWARE SPECIFICATIONS

All the experiments are carried out using the hardware specifications of GPU: GeForce 9500 GT, 1 MB DDR2, No of Processors = 4, No of core =32, RAM 1 GB, Frequency 1.35 GHz, DDR2 and CPU: Intel Core 2 Duo, 2.66 GHZ, No of cores available =2, No of thread=1, No of thread/core=1, Physical Memory =2 GB, DDR2

6. RESULTS AND DISCUSSION

For testing of proposed approach of text line and word segmentation, we collected a data set of handwritten as well as printed Devanagari documents from newspapers, old books and from different writers. The collected Devanagari documents are scanned using a scanner at 300 dpi and tested on the computer specifications shown in content 5. The results of proposed approach for text line segmentation and comparison with standard profiling based method are shown in fig. 5 that demonstrates the efficiency of our proposed approach. Fig. 6 shows an example of the word segmentation of extracted from segmented text line. The standard profiling based approach fails in the appearance of bindu (dot) at upper side of word, example is shown in fig.5 image number 1 but on the other hand the proposed method takes bindu as a part of word and done correct segmentation. On the basis of visual observation, proposed method of segmentation works better than standard profiling based method. The traditional profiling and proposed method do not work well in the case of overlapping text; both fails in calculating minima and maxima. However, both methods work well in case of straight line segmentation but proposed method is faster than standard profiling method. To make faster the proposed method, we parallelized it on CUDA and achieved a speedup of 20x-30x over the serial implementation when running on a GPU. The speedup is between the 20x to 30x but not more than 30x because this GPU have 30 cores. Table 1 and figure 4 shows the comparison of execution time of proposed algorithm on CPU over GPU. The speedup results heavily depend on the document image size also. Hence presented method proved that it works better than standard profiling based method and run faster on GPU.

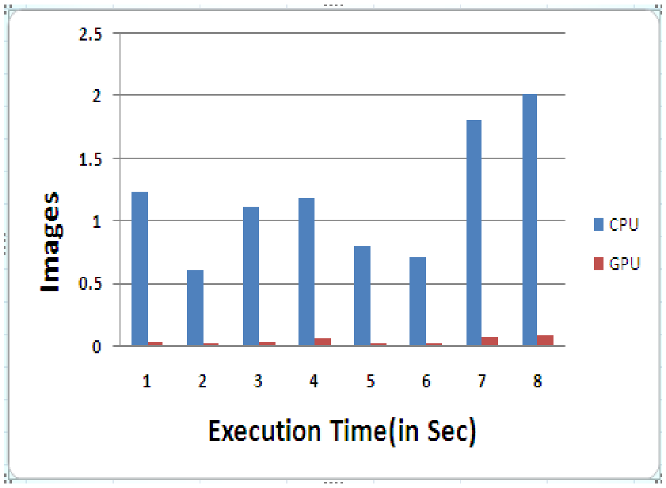
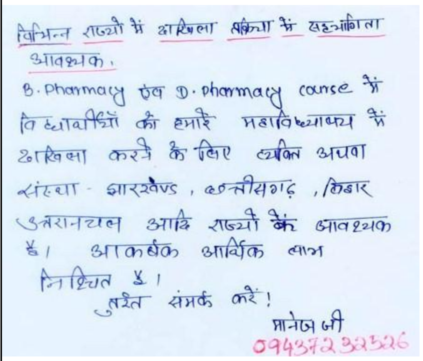
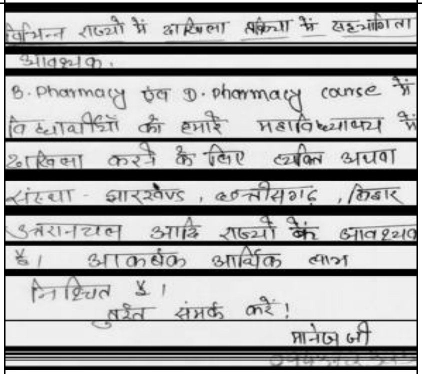
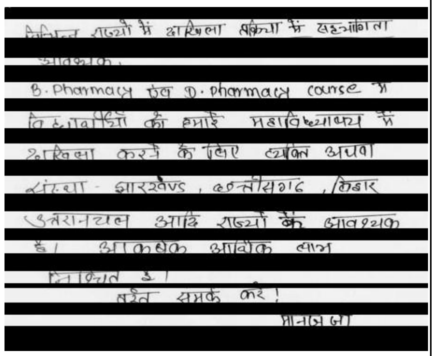
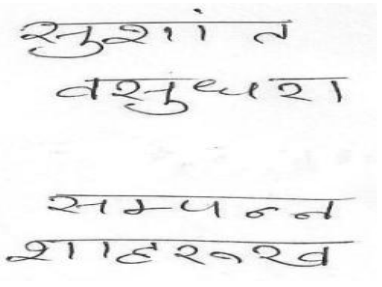
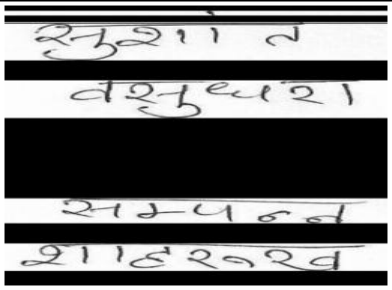
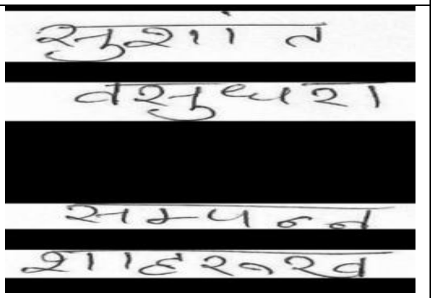
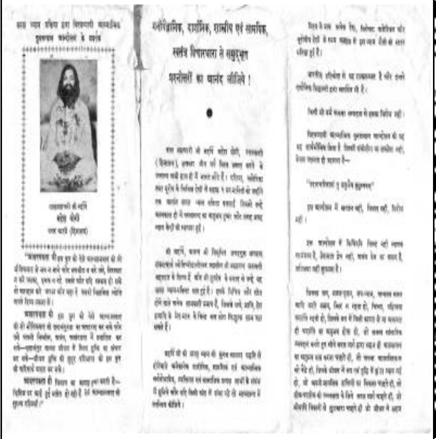
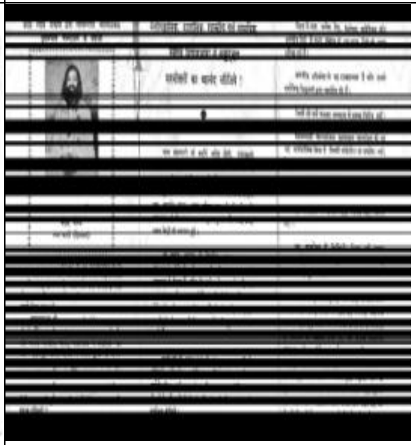
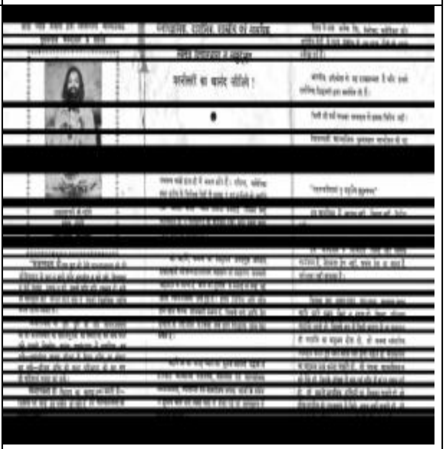
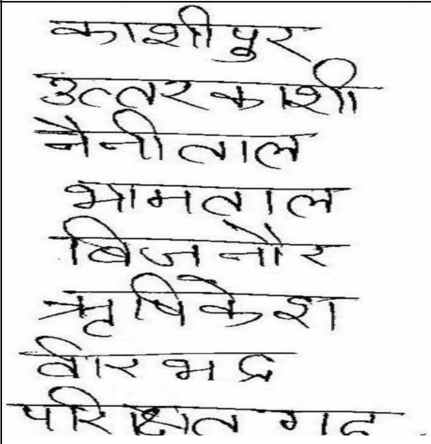
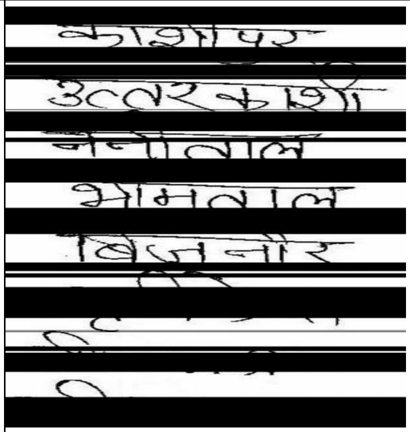
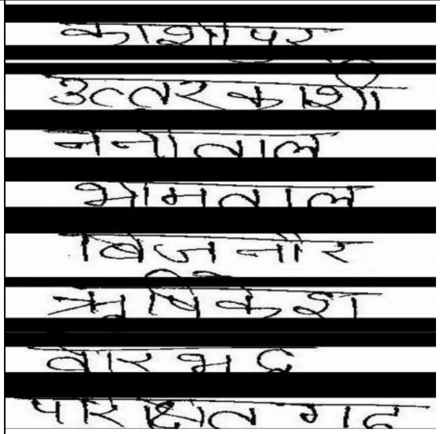


Fig 4: Execution time of all sample images

Table 1: Comparison of execution time of text line segmentation on CPU over GPU

Image	Execution Time on CPU (Sec)	Execution Time on GPU(Sec)	Average (Sec)		Speedup
			CPU	GPU	
1.	1.24	.0412	1.24	.0413	30x
	1.24	.0413			
	1.23	.0413			
2.	.61	.0277	.61	.0277	22x
	.62	.0279			
	.61	.0278			
3.	1.12	.0412	1.12	.0412	28x
	1.13	.0412			
	1.12	.0413			
4.	1.91	.0706	1.91	.0707	27x
	1.93	.0709			
	1.91	.0705			
5.	.81	.0289	.81	.0288	28x
	.82	.0288			
	.81	.0289			
6.	.71	.0269	.72	.0266	27x
	.72	.0266			
	.72	.0267			

S.N.	Input Image	Profiling Method	Proposed Method
1.	 विभिन्न राज्यों में दाखिला प्रक्रिया में सहभागिता आवश्यक। B. Pharmacy एवं D. Pharmacy course में विद्यार्थियों को हमारे महाविद्यालय में दाखिला करने के लिए व्यक्ति अथवा संस्था - झारखंड, छत्तीसगढ़, बिहार उत्तरांचल आदि राज्यों के आवश्यक है। आकर्षक आर्थिक लाभ निश्चित है। तुरंत संपर्क करें! मनोज जी 09437232526		
2.			
3.			
4.	 काशीपुर उत्तरकाशी नैनीताल भीमताल बिजनौर ऋषिकेश वीरभद्र परीक्षितगढ़		

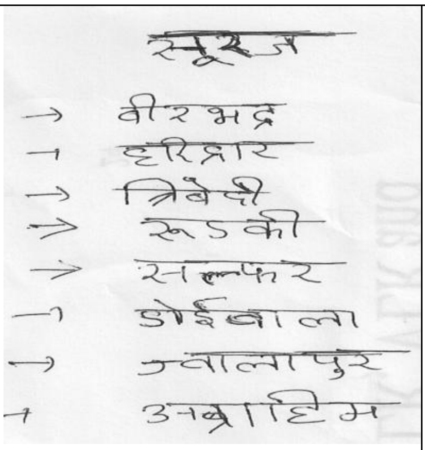
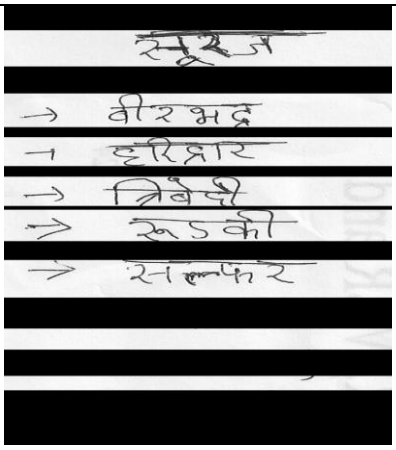
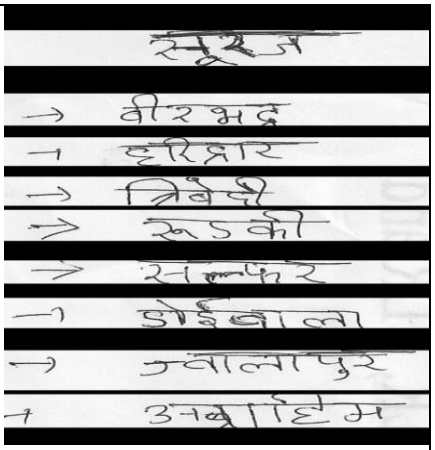
5.			

Fig 5: The text line segmentation of document images

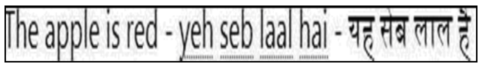
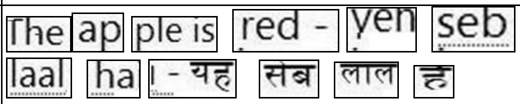
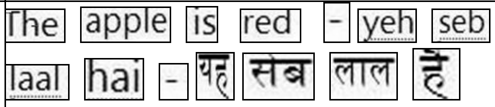
Method Task	Traditional Profiling Method	Proposed Method
Line Segmentation	 The apple is red - yeh seb laal hai - यह सब लाल है	The apple is red - yeh seb laal hai - यह सब लाल है
Word Segmentation	 The ap ple is red - yeh seb laal ha i - यह सब लाल है	 The apple is red - yeh seb laal hai - यह सब लाल है

Fig 6: A sample of line and word segmentation

7. CONCLUSION

Text Line and word segmentation are two of the important steps of any OCR system. In this research work, a modified profiling based fast segmentation algorithm has been presented and analyzed with traditional profiling based approach. The implementation of proposed algorithm on the graphics device is promising, with large two dimensional images (on a relatively low performance GPU) than sequential algorithms. The method of using profiling to segment images and how to accelerate this process using GPUs has been discussed in great detail. This algorithm serves as an excellent framework to solve a diverse array of segmentation problems. The experiments show that proposed segmentation method even works better on other scripts such as Latin scripts documents.

CUDA itself has been shown to be an excellent framework to accelerate computational problems in engineering, and is gaining more features and fewer limitations every few months. The principal disadvantages of CUDA are that it is only effective for very data parallel problems, and that it is not an industry standard. Recently, to counter the latter, it is very likely that it will in fact be replaced by OpenCL (Open Computing Language). The syntax and architecture between CUDA and OpenCL will be very similar, allowing this code to be easily ported to OpenCL. Nonetheless the impressive speedups attained using such low end hardware demonstrate the power of this parallel segmentation algorithm.

8. REFERENCES

[1] Sethi, I. K and Chatterjee, B. 1977. Machine recognition of constrained hand-printed Devanagari. Pattern Recognition 9 (1977) 69-75.

[2] Pal, U. and Chaudhuri, B.B. 2004. Indian script character recognition: a survey. Pattern Recognition 37, 1887 – 1899.

[3] Sharma, N. Pal, Kimura, U. F. and Pal, S. 2005. Recognition of offline handwritten Devanagari characters using quadratic classifier. In Proceeding of ICVGP conference, Springer, LNCS 4338, 805-816.

[4] Kumar, S. and Singh, C. 2005. A study of zernike moments and its use in Devnagari handwritten character recognition. In Proceeding of International Conference on Cognition and Recognition, 514-520.

[5] Hanmandlu,M. Ramana Murthy, O.V. and Madasu, V.K. 2007. Fuzzy model based recognition of handwritten Hindi characters. Digital Image Computing Techniques and

Applications, 9th Biennial Conference of the Australian Pattern Recognition Society, 454-461.

[6] Pal, U. Sharma, N. Wakabayashi, T. and Kimura, F. 2007. Off- line handwritten character recognition of Devanagari script. In Proceeding of 9th International Conference of Document Analysis and Recognition, 496-500.

[7] Arora, S. Bhattacharjee, D., Nasipuri, Basu M. D.K., and Kundu, M. 2008. Combining multiple feature extraction techniques for handwritten Devnagari character recognition. IEEE Region 10 Colloquium and the Third ICIIS, Kharagpur, INDIA, 1-6.

[8] Pal, U. Chanda, Wakabayashi , S. T. and Kimura, F. 2008. Accuracy improvement of Devanagari character recognition combining SVM and MQDF. In Proceeding of 11th International Conference of Frontier of Handwriting Recognition, 367-372.

[9] Pal, U., Wakabayashi T. and, Kimura, F. 2009. Comparative study of Devanagari handwritten character recognition using different feature and classifiers. In Proceeding of 10th International Conference on Document Analysis and Recognition, 1111- 1115.

[10] Plessis, B. Siscu, Menu, A. E. and Moreau, J.W.V. 1992. Isolated handwritten word recognition for contextual address reading. In Proceeding of USPS 51h Advanced Technology Conference, France, 749-750.

[11] Parui, S. K. and Shaw, B. 2007. Offline handwritten Devanagari word recognition: An HMM based approach. In Proceeding of International conference on PReMI 2007, Springer, LNCS 4815, 528–535.

[12] Shaw, B. Parui, S. K. and Shridhar, M. 2008. A segmentation based approach to offline handwritten Devanagari word recognition. In Proceeding of International Conference on Information Technology, IEEE, 256-257.

[13] Marinai, S. 2008. Introduction to document analysis and recognition. Studies in Computational Intelligence (SCI) 90 (2008) 1–20.

[14] Tang, Y.Y., Suen, C.Y., Yan, C.D. and Cheriet, M. 1991. Document analysis and understanding: a brief survey. In Proceeding of First International Conference on Document Analysis and Recognition, Saint-Malo France, 17-31.

[15] Plamondon, R. and Srihari, S. N. 2000. On-line and off-line handwritten recognition: a comprehensive survey. IEEE Trans on PAMI 22 (2000) 62-84.

[16] Lecolinet, E. and Crettez, J. 1991. A grapheme based segmentation technique for cursive script recognition. In Proceeding of First International Conference of Document Analysis and Recognition, 740-748.

[17] Yanikoglu, B. and Sandon, P.A. 1998. Segmentation of off-Line cursive handwriting using linear programming. Pattern Recognition 31, No. 12, (Dec. 1998), 1038-1041.

[18] Pal, U. and Choudhary, B.B. 2001. Machine printed and handwritten text lines identification. Pattern Recognition Letters 22 (2001) 431-441.

[19] Leroux, M., Salome, J.C. and Badard, J. 1991. Recognition of cursive script words in a small lexicon. In Proceeding of First International Conference of Document Analysis and Recognition, 774-782.

[20] LU, Y.I. and Shridhar, M. 1996. Character segmentation in handwritten words. Pattern Recognition 29 (1996) 77- 96.

[21] Casey, R.G. and Lecolinet, E. 1996. A survey of methods and strategies in character segmentation. 199. IEEE Trans. on PAMI 18 (July1996) 156-161.

[22] Garg, N.K. Kaur, L. and Jindal, M.K. 2010. A new method for line segmentation of handwritten Hindi text. In Proceeding of Seventh International Conference on Information Technology: New Generations (ITNG), IEEE, 392 – 397.

[23] Garg, N.K. Kaur, L. and Jindal, M.K. 2010. Segmentation of handwritten Hindi text. International Journal of Computer Applications 1 (2010) 0975 – 8887.

[24] Thillou, C. M. and Gosselin, B. 2006. Character segmentation by recognition using log-gabor filters. In Proceeding of 18th International Conference on Pattern Recognition, Pattern Recognition, 901- 904.

[25] Casey, R. G. and Nagy, G. 1982. Recursive segmentation and classification of composite character patterns. In Proceeding of 6th International Conference Pattern Recognition, Munich, Germany, (1982), 1023–1026.

[26] Kim, S. H., Jeong, S., Lee, G. S. and Suen, C. Y. 2001. Word segmentation in handwritten Korean text lines based on gap clustering techniques. In Proceeding of 6th International Conference of Document Analysis and Recognition, IEEE, 189-193.

[27] Elgammal, A. M., and Ismail, M. A. 2001. A graph-based segmentation and feature extraction framework for Arabic text recognition. In Proceeding of 6th International Conference of Document Analysis and Recognition, IEEE, 622-626.

[28] Kompalli, S., Setlur, S. and Govindaraju, V. 2006. Design and comparison of segmentation driven and recognition driven Devanagari OCR. In Proceeding of Second International conference of Document Image Analysis for libraries, IEEE, 7-102.

[29] NVIDIA CUDA Programming Guide Version 2.0, available at www.nvidia.com/object/cuda_develop.html.

[30] NVIDIA Corporation: NVIDIA CUDA programming guide. Jan 2007, available at http://developer.download.nvidia.com/compute/cuda/2_0/docs/NVIDIA_CUDA_Programming_Guide_2.0.pdf