SCENE TEXT DETECTION SUITABLE FOR PARALLELIZING ON MULTI-CORE

Jin Man Park, Heejin Chung and Yeong Kyeong Seong

Software Lab., SAIT, Samsung Electronics

ABSTRACT

Text recognition in natural color images is not successful with the naïve application of general OCR (Optical character recognition) because the numerous visual environments and luminance in the real world scene. This problem can be solved by adopting various brightness/contrast adjustments and texture analysis to create a binary image very similar to printed text. But such a solution can have a shortcoming; increased execution time. We propose an English text detection method that parallelizes processing jobs to utilize a multi-core processor effectively in order to improve text recognition accuracy and minimize execution time overhead.

Index Terms**—** Text recognition, Text detection, Parallel processing, Multi-core

1. INTRODUCTION

Due to the rapid increase of images and videos, efficient methods for managing or extracting useful multimedia information are demanded, resulting in many researches on semantic analysis of media. Among the many semantic concepts, text information is especially favored, as it effectively describes the contents, and is relatively easy to extract. Different text extraction algorithms are applied according to the type of image being handled. Images can be generally classified into document images, images with caption text, and natural images containing scene text. In contrast to document text or caption text, scene text is often affected by variations in scene and camera parameters such as illumination, focus, motion, etc, making text extraction even more difficult.

Most scene text extraction methods can be categorized as texture based, connected component based, or hybrid methods. Texture based methods treat the text region as a special type of texture [5,6,7], while connected component based methods detect text by extracting the connected components of monotonous colors that meet certain size, shape, and spatial alignment constraints [1,2,3,4]. Hybrid approaches combine these methods and other techniques [8,10]. However, the text extraction rate for natural images still has plenty of room for improvement. Computation speed is also an important issue, since higher text extraction rates usually mean much more computation, resulting in slower results.

Nowadays, the majority of processors are multi-core. This trend has started a few years ago, as chip manufacturers faced physical limits in improving microprocessor performance. Most application software is not written to effectively take advantage of multi-core chips, and needs modifications to benefit from them. Some parallel programming issues are; partitioning of the application, load balancing, and minimizing inter-thread communication. Without considering these issues, parallel software cannot achieve satisfying speedup.

In this paper, we propose a method to improve text extraction accuracy while keeping a fast computation speed by utilizing multi-core processors. Our approach is a hybrid method combining a connected component based method and a texture based method. We employ an open source OCR engine OCRopus [11]. The input image is processed several times to achieve higher accuracy. New images are created by applying some basic image processing techniques, such as gray morphology and brightness adjustment, to the input image. Although the total computation is considerably increased, our method can easily be parallelized so the accuracy-speed trade off is not high. In the next section we describe our text extraction method, and in section 3 the parallelization of our algorithm is discussed. Parallelization granularity is experimentally determined and the evaluation results are shown in section 4 with some other experimental results. Section 5 concludes our work and suggests some topics for future work.

2. PROPOSED ALGORITHM

2.1. Connected component method using multiple images

2.1.1. Background

A general OCR recognizes black text on white backgrounds. In contrast to the general OCR, scene text recognition should consider various luminances that the scene text is exposed to, the complex background that has various contrasts, and the existence of various non-text figures. In order to eliminate these characteristics we process the scene text image to get a document-like black and white text image that can be recognized with general OCR algorithms. The image processing sequence is as follows.

1) Apply gray morphology to minimize the distortion caused by light.

2) Adjust the brightness of the image so that when binarized, only the text area will be black.

3) Increase the contrast between text and non-text area to amplify the color difference.

However, the optimal brightness and contrast parameters differ for various images. So using only one parameter set will result in a very low recognition rate. If we use multiple parameter sets for image processing, accordingly executing the recognition processes with the intermediate binary images repeatedly, we will get higher text recognition rate.

This idea has a shortcoming due to the longer execution time cause by the additional image processing. On the bright side, the additional computation is merely caused by the repetition of the original algorithm with variations of the input image that have data independency. Therefore, if our algorithm is running on a multi-core Hardware platform, it can be effectively parallelized, resulting in not only an increased text recognition rate, but also a reduced execution time.

Nowadays, commercial CPUs generally have two or four cores. If the CPU had more cores, say 256 cores, our approach could achieve real time speed with high recognition rates. But that is not the case, so it is necessary to adopt mechanisms to achieve performance enhancement. We select the K most effective parameter sets by evaluating the detection and recognition results for intermediate images created with the full parameter set. We also check the existence of text areas in the image with some heuristics in advance to terminate the processing sequence for non-text images, therefore eliminating unnecessary computation.

2.1.2. Parallel Text Detection

Figure 1 shows the sequence of our coarse grain parallelized text detection algorithm. In the preprocessing stage, the input image is loaded and converted to a gray image and gray morphology is applied to minimize the distortion caused by light. Then K clones of the gray images are created, parameters for each clone are set, and the gray image/parameter pairs are pushed into a queue.

Now connected component based text detection is carried out on the clone images. N threads are created and each thread pops an image/parameter pair to process. The threads use the parameters to execute gray morphology and adjust brightness and contrast of the gray image, resulting in a binary image. Next they search the binary image for connected components and find candidates for character boxes among the connected components. Candidate lines are found from the candidate character boxes, and the threads push the lines and the binary image into a queue. If no character box or line candidates are found, the threads stop further processing and fetch a new gray image and parameter set from the queue to work on.

Finally text recognition is carried out on the document-like images. N threads are created, and each thread fetches lines and binary images from the queue, and inputs the data to Tesseract, a recognition engine. This results in several detected strings.

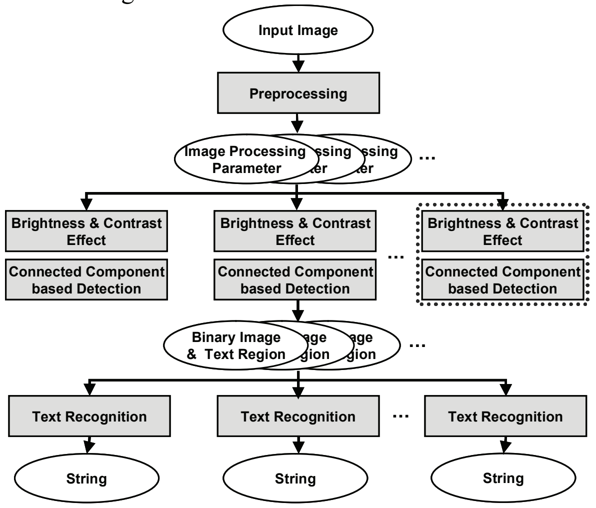
Input Image
Preprocessing
Image Processing Parameter
...
Brightness & Contrast Effect
Connected Component based Detection
Brightness & Contrast Effect
Connected Component based Detection
...
Brightness & Contrast Effect
Connected Component based Detection
Binary Image & Text Region
...
Text Recognition
Text Recognition
...
Text Recognition
String
String
String

Figure 1. Parallel Text Detection

2.2. Hybrid Method: Multiple features with AdaBoost Cascade Classifier

Using multiple images for connected component based detection results in several candidate strings. These strings need to be merged into one result. We can either merge the text regions before text recognition or merge the strings after recognition. Merging the regions is more practical, because image processing can be used, while string merging requires dictionary matching. We perform texture analysis on each region to get the final text region. Therefore our final approach is a hybrid one combining a region based method for extracting candidate text regions and a texture based method for verifying the final text region. Figure 2 shows the overall algorithm flow.

The AdaBoost cascade classifier, which uses several classifiers in a cascade manner for optimal performance improvement, is applied. A set of weak classifiers are combined to make a strong classifier [9]. The optimal weights for combining the weak classifiers are determined with a learning algorithm. Computation speed is improved by placing classifiers with low computation and performance in the front of the cascade sequence and high computation/performance classifiers in the back. . Figure 3 shows an example of a cascade classifier. Every stage of the cascade either rejects the analyzed region or passes it to the next stage. Only the last stage may finally accept the region. Thus, to be accepted, a region must pass through the whole cascade, but rejection may happen at any stage.

We extract feature vectors such as mean, standard deviation, derivative, histogram and edge linking to

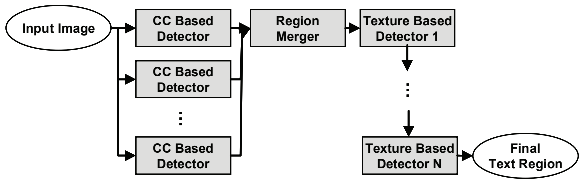

Figure 2. Scene Text Detection with the Hybrid Method

implement the simple classifier. For the simple classifier, k-mean clustering is done on the texture features of text and non-text regions. The classifiers in the front of the sequence contain all the text regions – therefore having high recall and low precision rates – but as the high performance classifiers are added higher precision is reached. The first input to the texture based detector are the candidate regions obtained from our connected component method, and the merged and expanded regions are input into the next texture based detectors.

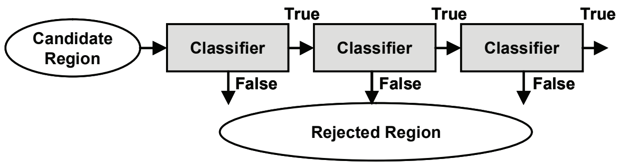

Figure 3. Cascade classifier

3. PARALLELIZATION FOR MULTI-CORE PROCESSOR

Performance of the parallelized algorithm depends on the parallelizable portion, load balancing among the multiple processors, maximizing bus utilization, and minimizing critical sections. The parallelizable portion is limited by the original algorithm, while the other features can be optimized.

The level of granularity applied to parallelization has the most effect on the optimization. Some studies say that coarse granularity show higher levels of speedup due to the low communication requirement [12], while others prefer fine granularity because it allows more precise load balancing [13]. However the optimal granularity depends on the software being parallelized and the underlying hardware. We examine two levels of granularity for the improved OCRopus, coarse grain and fine grain.

As explained in section 2.1, coarse grained parallelization distributes computation by assigning images to each processor, and the text detection sequence is locally carried out. With coarse grained parallelization, almost the entire text detection sequence can be parallelized. When an image is input, memory allocation is done for the additional images, thread parameters are set, and then each processor locally creates the images and executes text extraction on them. The image data and parameters are managed in a queue so the processors can get a new image to work on whenever ready. The text detection results are also saved in a queue, so the text recognition sequence can also be parallelized in a similar manner. Because the sequence is suspended if the image does not contain any text, load is a little imbalanced, but almost no communication is required.

Fine grained parallelization is applied at function level, and each image is split into horizontal or vertical strips (subimages), so each processor can carry out the function on each subimage. The number of subimages equal that of the processors in use. Since applying fine grained parallelization to all functions of the text extraction sequence is impossible, we select a few functions that occupy a high proportion of the total execution time. As shown in Table 1, binarizer, segmenter, and preprocessing are the most time consuming modules, occupying more than 85% of the text detection sequence. The binarize method can be entirely parallelized, but the segmenter requires additional computation, and only a portion of preprocessing can be parallelized. All processes work on data with almost equal size, thus the load is more balanced among them; however, synchronization overhead exists.

Table 1. Time Proportion of Modules

Module	%	Sub Module	%
Detection	86.58	preprocessing	6.94
		binarizer	59.51
		segmenter	17.48
		region extraction	2.64
Recognition	13.42		
Total	100.00		

4. EXPERIMENTAL RESULTS

4.1. Experimental Setup

The experiments were carried out on 47 ICDAR images of outdoor signs, and 13 images taken with a cell phone camera. We ran the tests on a platform board for ARM11 MPCore, which consists of 4 210MHz processors.

The aim of the text detection algorithm is to find the accurate text regions in an image. To measure the accuracy of a text detection algorithm, the detected text region and ground truth region are compared. Text is regarded to be successfully extracted if the total difference between coordinates of the four corners of the regions is less than a certain threshold.

4.2. Comparison of single and multiple images

Table 2. Recognition rate of single/multiple parameter

	47 ICDAR Images			13 Camera Phone Images		
	one parameter	sparse set	optimized set	one parameter	sparse set	optimized set
Success	40.4%	65.9%	74.4%	23.0%	46.1%	69.2%
Partial Success	59.5%	82.9%	87.2%	69.2%	92.3%	92.3%

Success: Detection and recognition is successfully done.
Partial success: Success in detection but only partial success in recognition.
Recognition rate = recognized image / total image

Table 2 shows the recognition results when target images with one parameter giving the highest recognition rate are processed, when evenly distributed parameters in the full parameter set are used, and when N most effective parameters are used. Processing multiple images with an optimized parameter set remarkably improves the recognition rate. For the real use of proposed algoritmn, the the effective parameters in the set should be dynamically updated due the environment's luminance change, and the update should be take place in real-time. So, the algorithm can optinally adopts an early decision scheme that evaluates each parameter's intermediate output such as character and line detection rate, and then decides a parameter's thread to stop or to continue execution

4.3. Comparison of coarse & fine-grained parallelism

We tested 8 images from our test set, and measured the time consumed by the binarize function, so the parallelized portion would be equal for both versions. Table 3 shows that coarse grained parallelism suits our method better than fine grained parallelism. This is because coarse grained parallelism requires no inter thread communication, while fine grained parallelism needs synchronization. And load for both granularities are equally well balanced. Both the speedup and linearity of coarse grained parallelism is superior.

Table 3. Speedup for Coarse & Fine Grained Parallelism

Threads / Test #	Coarse Grained				Fine Grained			
	1	2	3	4	1	2	3	4
1	1.00	1.94	2.54	3.74	1.00	1.82	2.45	2.93
2	1.00	1.91	2.48	3.62	1.00	1.83	2.23	2.80
3	1.00	2.01	2.62	3.89	1.00	1.83	2.40	2.97
4	1.00	1.94	2.52	3.74	1.00	1.76	2.47	2.88
5	1.00	1.92	2.51	3.68	1.00	1.65	2.26	2.76
6	1.00	2.02	2.61	3.88	1.00	1.83	2.51	3.05
7	1.00	1.91	2.48	3.66	1.00	1.78	2.45	2.94
8	1.00	2.01	2.60	3.85	1.00	1.80	2.41	2.88
avg	**1.00**	**1.96**	**2.55**	**3.76**	**1.00**	**1.79**	**2.40**	**2.90**

Speedup = Single thread execution time / Multi thread execution time

4.4. Comparison of CC Based and Hybrid Method

Table 4 compares the accuracy of our connected component based method and hybrid method. As described in section 2.2 the hybrid method improves accuracy.

Table 4. Accuracy of CC Based & Hybrid Method

	Precision (%)	Recall (%)
CC Method	-	81.00
Hybrid Method	52.50	89.81

$$Precision = \frac{\{correct\ extracted\ regions\}}{\{extracted\ regions\}}, Recall = \frac{\{correct\ extracted\ regions\}}{\{text\ regions\}}$$

5. CONCLUSION

This paper proposes a hybrid scene text extraction algorithm that can easily be parallelized. A connected component method is combined with a texture based method, and multiple images are processed to improve accuracy. The increased execution time is reduced by applying data parallelization to the image processing sequence. Coarse grained parallelism shows better performance because there is less synchronization overhead than fine grained (function level) parallelism. With our method, legacy OCR engines can be used for scene text extraction with no overhead as document-like binary images are created by the text extraction process. Also, by tuning the parameter set, the trade off between accuracy and speed can be adjusted, making our approach more suitable for real-time systems.

However, because multiple images are processed, the precision rate is relatively low compared to the recall rate. Degraded image quality due to image capturing and binarizing also lowers the recognition rate. Post processing the binary image and dictionary matching can be applied to further improve accuracy.

6. REFERENCES

[1] B. Gatos, I. Pratikakis, K. Kepene, and S.J. Perantonis, "Text Detection in Indoor/outdoor Scene Images," Proceedings of the First International Workshop on Camera-Based Document Analysis and Recognition, pp. 127-132, 2005

[2] C. Mancas-Thillou and B. Gosselin, "Color text extraction with selective metric-based clustering," Computer Vision and Image Understanding 107, 97-107, 2007

[3] T. Retornaz and B. Marcotegui, "Scene text localization based on the ultimate opening," International Symposium on Mathematical Morphology, 8, vol. 1, pp. 177-188, 2007

[4] K. Wang and J.A. Kangas, "Character location in scene images from digital camera," Pattern Recognition 36, no. 10, 2287-2299, 2003

[5] P. Clark and M. Mirmehdi, "Recognising text in real scenes," International Journal on Document Analysis and Recognition, no. 4, 243-257, 2002

[6] B.K. Sin, S.K. Kim, and B.J. Cho, "Locating characters in scene images using frequency features," International conference on pattern recognition, pp. III: 489-492, 2002

[7] X. Chen, J. Yang, J. Zhang, and A. Waibel, "Automatic detection and recognition of signs from natural scenes," IEEE Transactions on Image Processing 13, no. 1, 87-99, 2004

[8] K. Jung and J.H. Han, "Hybrid approach to efficient text extraction in complex color images," Pattern Recognition Letters 25, no. 6, 679-699, 2004

[9] X. Chen and A.L. Yuille, "Detecting and Reading Text in Natural Scenes," Proceedings of the 2004 IEEE Conference on Computer Vision and Pattern Recognition, 2004

[10] Q. Liu, C. Jung and Y. Moon, "Text segmentation based on stroke filter," Proceedings of the 14th Annual ACM international Conference on Multimedia, 129-132, 2006

[11] T.M. Breuel, "The OCRopus Open Source OCR System," Document Recognition and Retrieval XV. Proceedings of the SPIE, Volume 6815, pp. 68150F-68150F-15, 2008

[12] F. Tischler and A. Uhl, "Granularity Levels in Parallel Block-Matching Motion Compensation," Proceedings of the 9th European PVM/MPI Users' Group Meeting on Recent Advances in Parallel Virtual Machine and Message Passing Interface, pp. 183-190, 2002

[13] Q. Zhang, Y. Chen, J. Li, Y. Zhang and Y. Xu, "Parallelization and Performance Analysis of Video Feature Extractions on Multi-Core Based Systems," Proceedings of the 2007 International Conference on Parallel Processing, 2007