Text segmentation and recognition in unconstrained imagery

Yuko Roodt, Hans Roos and Willem Clarke
HyperVision Research Lab
School of Electrical Engineering
University of Johannesburg
South Africa
Email: yukor@uj.ac.za, hans.jmroos@gmail.com, willemc@uj.ac.za

***Abstract*—In this paper, we present a novel method for recognizing and segmenting symbols and text in complex image sequences. The algorithm is designed to take advantage of the massive computing capability of parallel processing architectures. The additional processing resources will allow for more preprocessing steps reducing the number of simplification assumptions on the orientation, structure, scale and colour of the detected character symbols. The increased algorithmic complexity yields better recognition performance. This optical character recognition framework was designed to run on video sequences of unstructured environments. A robust algorithm will be presented that addresses these underlining vision based issues and will be tested for speed and recognition accuracy.**

I. Introduction

Optical Character Recognition (OCR) is a method for detecting text in images and converting the pixel representation of the letters to an equivalent character form recognized by the computer such as ASCII or Unicode [1].

Traditionally, OCR is used in commercial systems to search and store their large number of paper forms and documents electronically. Searching through these paper documents by hand can be a tedious and time consuming process, which lends itself to automation. Document digitization has become an important and integrated part of modern companies [2].

Traffic monitoring and number plate recognition is another common OCR application. The steps involved include localizing the number plate in the image and then classifying the individual letters on the plate. This can be used to do automatic electronic toll collection, logistic vehicle tracking and traffic surveillance [3]. The primary drawbacks of these systems are that they are designed to function within tight operational constraints and assumptions, unpredictable lighting conditions and out-of-focus distortions can influence the reliability of the results [4].

As camera and mobile computing technologies mature and become widely available, a new range of applications and engineering opportunities emerge. Research fields such as traffic sign recognition [6], automated athlete tracking[7] and the field of machine understanding of text have received attention. Extensive efforts for combining technologies such as OCR and Text-to-Speech have given even the blind or visually impaired access to textual information in their surrounding environments [8].

The trend of using the Graphics Processing Units (GPU) for Image Processing has evolved over recent years from running simple computer vision operators and filters to the development of complex interactive algorithmic solutions. These complex algorithms use advanced functions that work together to solve image processing problems that have high processing requirements. The GPU was developed to create a rich graphical representation from a description of a virtual scene, image processing on the other hand can be considered as the inverse of this process, where information needs to be extracted from an image of a rich environment [9]. If we are able to harness processing potential of this parallel processor, we will be able to process more complex image processing models.

We propose to implement an unconstrained OCR system optimized for parallel processing to deal with the environmental and image processing related issues. This system will enforce only a small number of constraints by harnessing the processing power of the GPU. In Section II we will present the research methodologies required for the implementation of this algorithm. Section III describes our proposed text segmentation and recognition approach, followed by the system testing and analysis of our results.

II. Background

A. Greyscale conversion

The greyscale of an input image can be obtained by calculating a weighted average of the individual colour components to account for human perception. The weights are 21.26% for the Red Component, 71.52% for Green and 7.22% for the Blue Component [10]. This colour space conversion creates a reduction in dimensionality.

$$I = 21.26 * R + 71.52 * G + 7.22 * B \tag{1}$$

where R is the red component, G is the green component and B is the blue component of the colour.

B. Local adaptive thresholding

Thresholding classifies each pixel of an image into a binary representation such as ”true” or ”false”, ”foreground” or ”background”. Traditionally in simple thresholding methods a

single fixed value is used to classify each value into these categories. Unfortunately this simple approach fails under varying illumination conditions across the image. Local adaptive thresholding can be used to improve the binarization results of these complex scenarios[11]. Each value in a greyscale image I can be represented as a value between 0 and 1:

$$I(x, y) \in [0, 1] \tag{2}$$

where x and y are the current location in the image

The mean pixel intensity of a window around the current pixel is used to estimate the local threshold required to binarize the image. This simple comparison is used to classify each value into foreground or background.

$$b(x, y) = \begin{cases} 0 & \text{if} I(x, y) < t(x, y) \\ 1 & \text{else} \end{cases} \tag{3}$$

where x and y are the current location in the image, b is the binarized image, t is the stored local threshold.

C. Optical character recognition

A conventional OCR system consists of 3 processing stages. The steps involved are the detection, segmentation and recognition of text. The detection stage attempts to localize regions in the image that have a high probability of containing text. In controlled environments and setups these regions have a good contrast change between the light and dark regions. There are also a high degree of gradient responses in the horizontal and vertical directions. Many OCR systems use this knowledge to find and extract the textual regions. The next step is the segmentation stage. The image is broken down into more manageable parts. Individual characters or words are extracted from the potential textual regions in the image. The recognition step attempts to classify each extracted region into a valid character or set of characters [12].

III. Unconstrained Text Recognition

The unconstrained text recognition and classification system consists of a number of steps. An overview of the algorithm architecture and the interactions between the different steps can be seen in Figure 1.

The first step is to obtain an image from permanent storage or a video stream such as a digital camera. The next step converts the input data into a more manageable form. Colour invariance is achieved by converting the the RGB colourspace input image to the greyscale representation. This reduction in information will simplify the segmentation process. Errors and digitization artifacts are removed by convolving the image with a small Gaussian kernel. This will remove high frequency information in the input image which can reduce the compression artifacts and smooth the transitions between different image regions.

This greyscale image is then converted to a trinary form, each pixel is classified as being foreground, background or undefined. If a valid classification could not be made or the error associated with making the classification is large, a region will be classified as being undefined. After the initial trinary

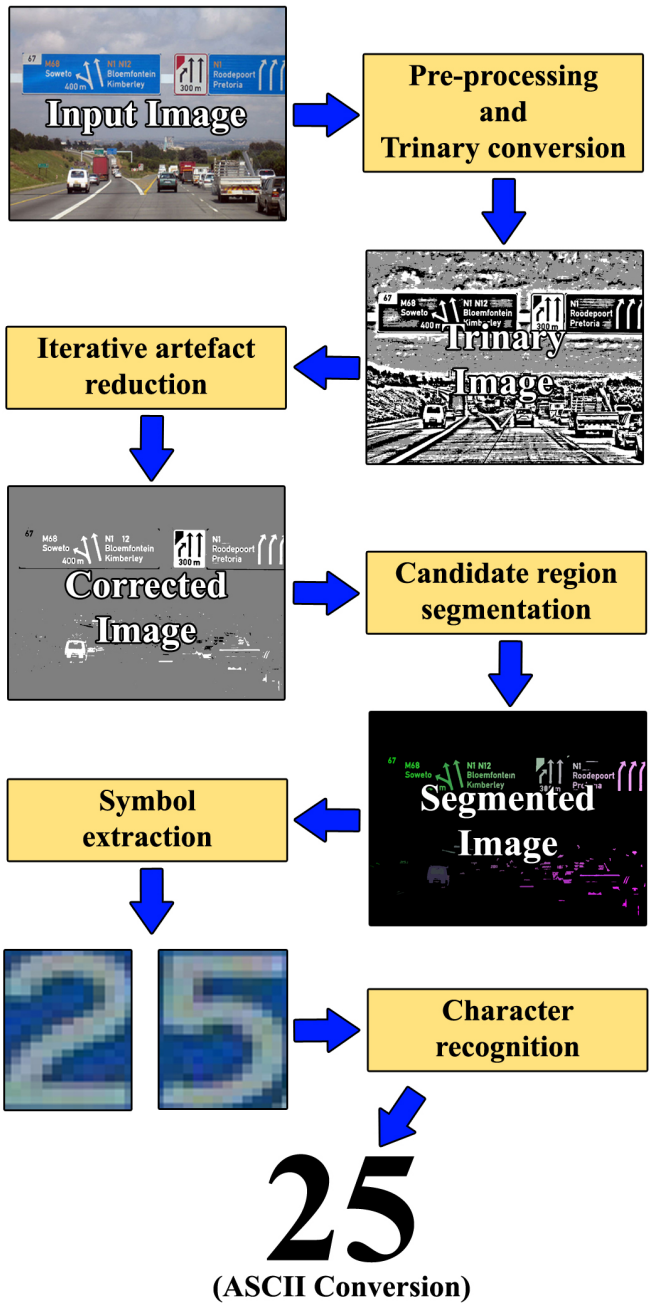

Fig. 1. Architectural overview of the unconstrained text segmentation algorithm

conversion, classification errors should be removed. Classified regions neighboring directly next to undefined or unclassified are prone to trinary classification errors. These areas have a low probability of containing valid textual symbols. An iterative process is used to remove these areas and classify them as undefined.

The different areas of the corrected trinary image then have to be clustered. A 4-connected iterative filter is used to cluster neighboring regions. Each clustered group is assigned a bound ID that can be used to uniquely identify the pixels associated with the group. The bound ID also specifies the region of the

image or bounding box that enclose the group. These clustered groups are then extracted into sub-images as possible symbol candidates. The final stage is to recognize the segmented character candidates. Each candidate image is converted to a histogram form to allow fast rotation invariant classification. The candidates histograms are then recognized and classified by comparing them against a large template database. This is a general overview of the steps involved in the unconstrained segmentation and recognition algorithm, additional depth will be provided to fully understand the proposed method. We will now explain each step in more detail.

A. Preprocessing and trinary image conversion

An overview of the preprocessing and trinary image conversion process can be seen in Figure 2.

1) Greyscale conversion: The input image is converted to greyscale since the segmentation and recognition steps only require the Intensity information to provide reliable results. This will reduce the processing requirement since colour complexity information is discarded. Only a single floating point value is required per-pixel. Conversion to greyscale provide the added benefit of being able to recognize symbols of any colour since no colour information is used, colour invariance in the recognition stage is achieved.

2) Noise Removal: Compression and digitization artifacts can be minimized by convolving the greyscale image with a small Gaussian kernel. This will remove small amounts of high frequency noise. Noise can severely affect the segmentation process and reduce the effectiveness of the unconstrained segmentation and recognition algorithm.

3) Adaptive image trinarization: A simple method for determining if pixels in close proximity belong to the same group is to determine if they are local foreground or background. This is traditionally done by doing local adaptive binarization. It will cluster pixels with similar light intensities and will also provide invariance to lighting changes over the image since an adaptive local neighborhood is considered in the thresholding. This Local adaptive thresholding scheme does not provide the ability to handle and mark classification errors. We extended on local adaptive binarization to include this desired functionality.

A Ternary or trinary numeral system is considered to have a base of 3. Trinary values have 3 possible states and can be either 0,1 or 2 [13]. We made this representation more compliant with the binary image representation. We normalized the traditional trinary states into the ranges of 0 and 1. This gave us 3 possible states [0.0,0.5,1.0] which is "false", "undefined" and "true" respectively. A value can be classified as being "undefined" if its true or false state cannot be accurately determined. It then has the same amount of potential for being either "true" or "false".

The average intensity over the the local region is required to determine if a pixel has a higher or lower intensity than its neighborhood. A large Gaussian blur was performed to obtain the local average image. Varying the size of the Gaussian kernel changes the algorithm's ability to extract smaller or larger textual candidates, larger filters tend to provide better results. Every pixel is then compared to the local average, if it has a larger intensity value it will be marked as "foreground" and as "background" if it was smaller. If the difference between the current intensity value and the local average intensity is to small, a valid classification could not be made and the pixel is flagged as "undefined". Low contrast features will be marked as undefined, these regions will be excluded from the segmentation process.

B. Iterative artifact reduction

In the trinary image, classified pixels which neighbor undefined pixels have a lower probability of being potential textual regions. Even though these values were validly classified as "true" or "false" we want to exclude them from the segmentation process. An iterative 4-connected kernel is used to remove these pixels. Initially we determine and mark all the pixels neighboring "undefined" pixels as being on the edge, this is stored in an edge image. For every iteration of the artifact reduction algorithm, the trinary and edge status of each pixel and their neighbors are extracted from the corresponding images. If a neighbor has the same trinary status as the current pixel and the neighbor was marked as an edge, the current pixel will be marked as a new edge. This process is repeated until all edge pixels have been marked. As a final step the trinary status of edge pixels are classified as "undefined" this will exclude them from further processing. The iterative artifact reduction algorithm can be seen in more detail:

converged=false
while !converged **do**
  for all $pixels$ in $trinary\_Image$ **do**
    $x \leftarrow horizontalPixel\_position$
    $y \leftarrow verticalPixel\_position$
    $c\_Trinary \leftarrow trinary\_Image(x, y)$
    $t\_Trinary \leftarrow trinary\_Image(x, y + 1)$
    $b\_Trinary \leftarrow trinary\_Image(x, y - 1)$
    $r\_Trinary \leftarrow trinary\_Image(x + 1, y)$
    $l\_Trinary \leftarrow trinary\_Image(x - 1, y)$
    $c\_Edge \leftarrow edge\_Image(x, y)$
    $t\_Edge \leftarrow edge\_Image(x, y + 1)$
    $b\_Edge \leftarrow edge\_Image(x, y - 1)$
    $r\_Edge \leftarrow edge\_Image(x + 1, y)$
    $l\_Edge \leftarrow edge\_Image(x - 1, y)$
    if $c\_Trinary! = 0.5$ **then**
      if ($c\_Trinary = t\_Trinary$ **and** $t\_Edge = true$)
      or ($c\_Trinary = b\_Trinary$ **and** $b\_Edge = true$)
      or ($c\_Trinary = r\_Trinary$ **and** $r\_Edge = true$)
      or ($c\_Trinary = l\_Trinary$ **and** $l\_Edge = true$)
      then
        $c\_Edge \leftarrow true$
      end if
    end if
    $edge\_Image(x, y) \leftarrow c\_Edge$
  end for
  {Iteration convergence test}
  iteration_Error=difference(edge_Image,previousEdge_Image)

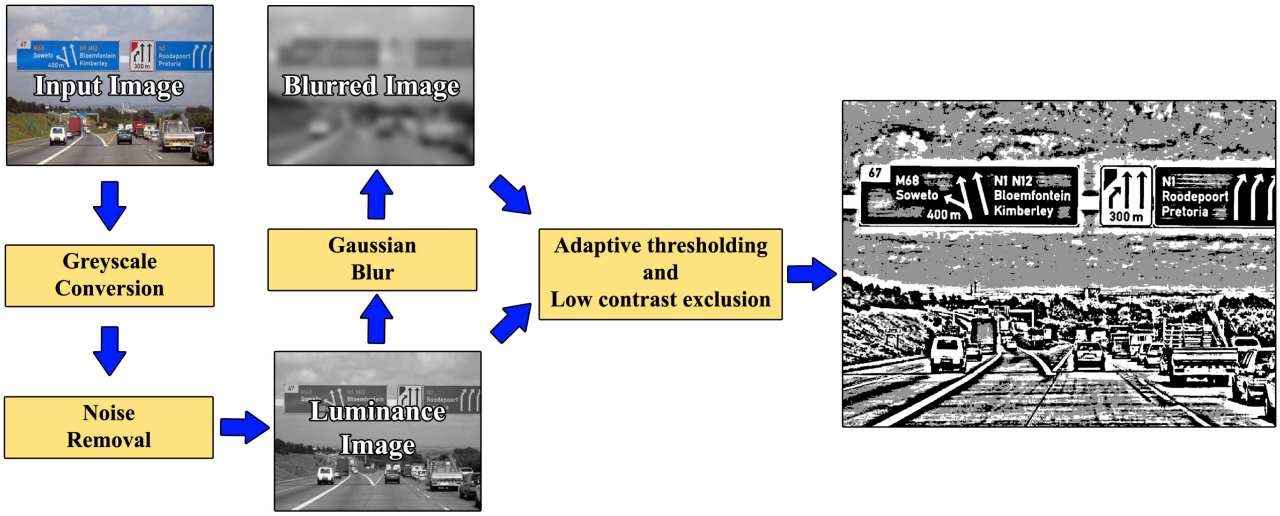

Fig. 2. Image preprocessing and trinarization

if $iteration\_Error = 0$ **then**
$converged \leftarrow true$
else
previousEdge_Image=edge_Image
end if
end while

C. Candidate region segmentation

It is natural to think of image segmentation as clustering of pixels or data points that "belong together". The trinary image provides us with all the information that is needed to group regions with similar properties. Initially every pixel in the trinary image that has a valid classification will be given an unique ID. After some experimentation, we decided to not only propagate a single unique ID inside the clustered regions but to propagate boundary box information. This produces the minimum bounding box which closely approximates the cut out area. This unique ID consists of four values: min_X, min_Y, max_X and max_Y and they are stored respectively in the x, y, z and w channels of the image. They are initialized according to the pixels current position in the image. A simple iterative scheme is used to propagate the minimum and maximum values between connected pixels with similar trinary classifications. It is a 4-connected kernel that can be run in parallel, when a value is read from outside the image the border ID is used. The resultant ID provides the bounding box information that will encapsulate the clustered group. This can be used to extract a sub-image that will contain the candidate symbol. This process is repeated for all pixels with valid trinary states until the system converges. After the segmentation process is done, each pixel in the group will contain the same bound ID. This helps to distinguish between duplicate entries when adding all the potential candidate characters to a list. More detail on the most important parts of the algorithm is provided:

converged=false

while !converged **do**
for all $pixels$ in ID_Image **do**
$x \leftarrow horizontalPixel\_position$
$y \leftarrow verticalPixel\_position$
$c\_Trinary \leftarrow trinary\_Image(x, y)$
$t\_Trinary \leftarrow trinary\_Image(x, y + 1)$
$b\_Trinary \leftarrow trinary\_Image(x, y - 1)$
$r\_Trinary \leftarrow trinary\_Image(x + 1, y)$
$l\_Trinary \leftarrow trinary\_Image(x - 1, y)$
$c\_ID \leftarrow ID\_Image(x, y)$
$t\_ID \leftarrow ID\_Image(x, y + 1)$
$b\_ID \leftarrow ID\_Image(x, y - 1)$
$r\_ID \leftarrow ID\_Image(x + 1, y)$
$l\_ID \leftarrow ID\_Image(x - 1, y)$
if $c\_Trinary! = 0.5$ **then**
if $c\_Trinary$ =t_Trinary **then**
$c\_ID.xy \leftarrow min(c\_ID.xy, t\_ID.xy)$
$c\_ID.zw \leftarrow max(c\_ID.zw, t\_ID.zw)$
end if
if $c\_Trinary$ =b_Trinary **then**
$c\_ID.xy \leftarrow min(c\_ID.xy, b\_ID.xy)$
$c\_ID.zw \leftarrow max(c\_ID.zw, b\_ID.zw)$
end if
if $c\_Trinary$ =r_Trinary **then**
$c\_ID.xy \leftarrow min(c\_ID.xy, r\_ID.xy)$
$c\_ID.zw \leftarrow max(c\_ID.zw, r\_ID.zw)$
end if
if $c\_Trinary$ =l_Trinary **then**
$c\_ID.xy \leftarrow min(c\_ID.xy, l\_ID.xy)$
$c\_ID.zw \leftarrow max(c\_ID.zw, l\_ID.zw)$
end if
end if
$ID\_Image(x, y) \leftarrow c\_ID$
end for
{Segmentation convergence test}
iteration_Error=difference(ID_Image,previousID_Image)

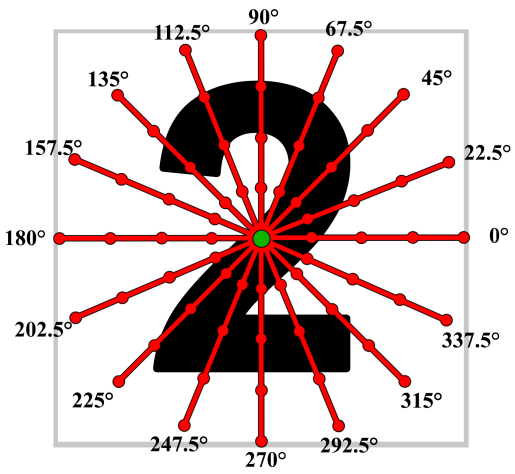

Fig. 3. Calculating the histogram of a character

if $iteration\_Error = 0$ **then**
$converged \leftarrow true$
else
previousID_Image=ID_Image
end if
end while

D. Symbol extraction

After all the pixels are grouped into clusters, the individual clusters need to be segmented or cut out from the rest. The bound IDs are sent back to the host application where the cut out procedure will be initiated. A list with all the unique bounding IDs is created, a large number of duplicate IDs can exist and only if no other such ID can be found in the list is it added. The extracted sub-image is normalized by setting all the pixels that contain the specific group ID to the foreground and every other pixel to the background. This will exclude other potential character regions that could have been included in the extracted sub-image.

E. Character recognition

Text can exist in an image as stand alone characters or words consisting of grouped characters. In a natural image or photo they can exist in any orientation, size and colour as well as being affected by affine transformations. The unconstrained text segmentation and recognition algorithm needs to account for these cases to allow for successful recognition. Every candidate symbol that was extracted from the input image will have to be tested against all templates in the template database. A similarity score is calculated to determine how close a template and a candidate match. The nearest match in the database is considered to be the same character as the candidate. If no match is found that have an adequate matching score, the candidate will be discarded and considered to be a noise artifact. A threshold is used to remove candidates with low matching scores.

To enable the matching of a candidate and a template, the candidate first has to be converted to a histogram representation. Every bucket in the histogram contains the volume discovered on a ray shot at an angle from the center of the extracted character as seen in Figure 3. The ray takes a number of samples in its direction, these samples are accumulated until a total coverage volume is calculated, this is then stored in the histogram bucket. The process is repeated for each bucket in the histogram. A histogram shift represents a rotation of the template. The match percentage is calculated by summing the difference error between each candidate histogram bucket and the corresponding template histogram bucket. The resultant error is then divided by the total number of buckets to produce a matching percentage between the candidate and the template. The candidate is tested against all possible histogram shifts to determine the matching rotation of the template that fits the best.

IV. Experimental setup

The algorithm was tested on a AMD Athlon X2 7750 multi-core CPU with 4GB RAM. The system contained a Nvidia GTX 260 GPU with 216 stream processors. The GPU had 896MB of onboard DDR3 memory and a theoretical peak processing performance of 804.8 Gflops. Our experiments were done on a range of datasets consisting of possible application areas. This showed the flexibility of the proposed algorithm and its ability to solve complex segmentation and recognition problems. A template database of 245 symbols was generated from the most common fonts. The font types include ”Courier New”, ”Times New Roman”, ”Arial” and ”Calibri”. All numerical and alphabetical characters in lower and uppercase were included in the template database. Each candidate symbol is tested against 128 orientations of each template in the template database to obtain rotation invariance. The orientation of a character can be determined up to a resolution of 2.815 degrees.

V. Results and Discussion

A number of experiments were done on different datasets to assess the algorithm abilities to solve difficult segmentation and recognition problems. The different steps of the algorithm were engineered to be applicable in a wide variety of applications. Some of the recognition results obtained can be seen in Table I. There are a number of factors that influence the ability of the algorithm to successfully segment and recognize characters. A large number of symbols occur regularly and are detected in photos of ”man-made” objects and nature scenes. Forms that resemble the letters ”T”, ”I” and ”L” occur regularly and are detected by the system. The assumption that letters should exist in words can be made to reduce the impact of these natural letters but at the cost of missing single symbols. This algorithm does not make this assumption and thus has reduced recognition rates.

Detecting characters at variable rotations is a difficult problem. Characters such as ”n” can be classified as ”u” and vice-versa due to rotation. There are other examples such

TABLE I
Text Recognition Results.

Dataset	Symbol Count	Detected symbol	Recognition Rate
American highway	160	215	41.875%
Soweto road	65	151	40.0%
Worthman sign	30	315	83.334%
Printed symbols	400	126	28.846%

as ”M” and ”W” or ”9” and ”6” that look similar under rotation transformations. These problems reduce the recognition performance drastically. Other factors that influence the recognition rate is the surface area that a symbol occupies in the image. The recognition of larger characters tend to perform better than low quality or far away characters.

TABLE II
Processing time breakdown

Dataset	Segmentation time	Recognition time
American highway	0.09 sec	18.361 sec
Soweto road	0.113 sec	12.517 sec
Worthman sign	0.067 sec	24.4 sec
Printed symbols	0.445 sec	9.694 sec

It can clearly be seen in Table II that the majority of the processing time is spent recognizing the candidate characters. The processing time of the recognition stage can be improved by reducing the orientation calculation resolution. Limiting the number of templates in the template database will also significantly improve performance at the cost of recognition accuracy.

TABLE III
Segmentation processing time

Resolution	cleanup iterations	clustering iterations	Processing time
256 x 192	100	50	0.055 sec
512 x 384	150	100	0.131 sec
1024 x 768	150	175	0.44 sec
2048 x 1536	200	250	1.874 sec

According to the segmentation processing results provided in Table III it can be seen that if the recognition stage could be improved this algorithm has the potential of being processed at interactive rates. The detection and segmentation of the candidate characters can be done in real-time even at high resolutions. I believe the segmentation stage is the strong point of this algorithm, some work will have to be done to improve the recognition stage.

VI. Conclusion

In this paper we presented an unconstrained text recognition system that harnesses the processing power of the GPU architecture to segment and recognize textual regions. We provided detailed explanations and pseudo code of our implementation together with Tables and Graphs depicting our obtained performance and recognition results. The algorithm had difficulty classifying characters and symbols that were composed of separated parts such as ”i”. The base of the character is separated from the top and the segmentation algorithm cluster the single character into two groups. An additional logic step can be integrated into the current system to solve some of the recognition problem that were discovered. Dictionary-based methods for improving recognition performance are widely used in handwriting recognition, incorporating this into the current system will increase the recognition rate.

References

[1] Palkovic, A.J., ”Improving Optical Character Recognition”, CSRS, 2008.

[2] Xiaoqing Ding, Di Wen, Liangrui Peng, Changsong Liu, ”Document digitization technology and its application for digital library in China”, Proc. First International Workshop on Document Image Analysis for Libraries, pp 46-53, 2004.

[3] Parker, J.R., Federl, P., ”An approach to licence plate recognition”, Proceedings of Visual Interface’97, 1997. pp 178.

[4] Emiris, D.M., Koulouriotis, D.E., ”Automated optic recognition of alphanumeric content in car license plates in a semi-structured environment”, Proc. of International Conference on Image Processing, vol 3, pp 50-53, 2001.

[5] Chen, D., Odobez, J.M., Bourlard, H., ”Text detection and recognition in images and video frames”, IDIAP, 2003.

[6] Yanlei, G., Yendo, T., Tehrani, M.P., Fujii, T., Tanimoto, M., ”A new vision system for traffic sign recognition”, Intelligent Vehicles Symposium (IV), IEEE, pp 7-12, 2010.

[7] Huang, Ye Q., Jiang, Q., Liu, S., Gao, W., ”Jersey number detection in sports video for athlete identification”, Proc. of SPIE, Visual Communications and Image Processing, pp 1599-1606, 2005

[8] Zandifar, A., Duraiswami, R., Chahine, A., Davis, L.S., ”A video based interface to textual information for the visually impaired”, Proc. Fourth IEEE International Conference on Multimodal Interfaces, pp 325-330, 2002.

[9] Fung, J., ”Computer Vision on the GPU”, GPU Gems 2, Addison-Wesley, 2005, p 651-652.

[10] Poynton, C.A., ”Digital video and HDTV: algorithms and interfaces”, Morgan Kaufmann Publishers, 2003, p 207-208.

[11] Shahedi, B.K.M., Amirfattahi, R., Azar, F.T., Sadri, S., ”Accurate Breast Region Detection in Digital Mammograms using a Local Adaptive Thresholding Method”, Eighth International Workshop on Image Analysis for Multimedia Interactive Services, WIAMIS ’07, , pp 26-26, 2007.

[12] Shah, P., Karamchandani, S., Nadkar, T., Gulechha, N., Koli, K., Lad, K., ”OCR-based chassis-number recognition using artificial neural networks” IEEE International Conference on Vehicular Electronics and Safety (ICVES), pp 31-34, 2009.

[13] Jun Sun, Naoi, S., Fujii, Y., Takebe, H., Hotta, Y., ”Trinary Image Mosaicing Based Watermark String Detection” 10th International Conference on Document Analysis and Recognition, ICDAR ’09., pp 306-310, 2009.