World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:1, No:7, 2007

An Edge-based Text Region Extraction Algorithm for Indoor Mobile Robot Navigation

Jagath Samarabandu, *Member, IEEE,* and Xiaoqing Liu

***Abstract*—**Using bottom-up image processing algorithms to predict human eye fixations and extract the relevant embedded information in images has been widely applied in the design of active machine vision systems. Scene text is an important feature to be extracted, especially in vision-based mobile robot navigation as many potential landmarks such as nameplates and information signs contain text. This paper proposes an edge-based text region extraction algorithm, which is robust with respect to font sizes, styles, color/intensity, orientations, and effects of illumination, reflections, shadows, perspective distortion, and the complexity of image backgrounds. Performance of the proposed algorithm is compared against a number of widely used text localization algorithms and the results show that this method can quickly and effectively localize and extract text regions from real scenes and can be used in mobile robot navigation under an indoor environment to detect text based landmarks.

***Keywords*—**Landmarks, Mobile Robot Navigation, Scene Text, Text Localization and Extraction.

I. Introduction

AUTOMATIC Region of Interest (ROI) detection is an active research area in the design of machine vision systems, and is used in many applications, such as autonomous mobile robot navigation, tourist assistant systems, navigational aids for automobiles, vehicle license plate detection and recognition, etc. Using bottom-up image processing algorithms to predict human eye fixations or mimic human perception and cognition and extract the relevant embedded information in images has been widely applied in automatic ROI detection systems. Psychophysical and neurophysiologic researchers find that two levels of content always draw human visual attention, namely the low level perceptual content, such as color/intensity, texture, edges, geometric shapes, and high level semantic content such as human faces, text, vehicles, etc., [1].

In vision-based mobile robot navigation, many potential landmarks such as nameplates and information signs contain text. By locating those text landmarks, it can provide feature correspondence in a sequence of video frames to reconstruct 3D models. Furthermore, text embedded in images contains large quantities of useful semantic information which can be used to fully understand images. Text recognition is very useful in high level robot navigational tasks, such as path planning and goal-driven navigation. Therefore, scene text is an important feature to be extracted.

We are looking into algorithms which can use low level perceptual features to exploit high level semantic text, i.e., performing text localization/extraction to detect text-based landmarks for mobile robot navigation. However, due to the variety of font sizes, styles, orientations, alignment, effects of uncontrolled illumination, reflections, shadows, the distortion due to perspective projection as well as the complexity of image backgrounds, automatic localizing and extracting scene text is a challenging problem. Wang et al. [2] proposed a connected-component based (CC-based) method which combines color clustering, a black adjacency graph (BAG), an aligning-and-merging-analysis (AMA) scheme and a set of heuristic rules together to detect text in the application of sign recognition such as street indicators and billboards. As the author mentioned, uneven reflections result in incomplete character segmentation which increases the false alarm rate in this method. Lienhart et al. [3] proposed a feed-forward neural network to localize and segment text from complex images. It is designed specifically for horizontal text with at least two characters. Furthermore, the performance of the network heavily depends on the composition of the training set. However, designing a general purpose texture classifier or text detector that can distinguish text with text-like texture in an uncontrolled/changing illumination environment is still an open problem. Zhong et al. [4] proposed a hybrid of CC-based and texture-based method to extract text. Although experimental results show that the combination of these two methods performs better, the monochrome constraint used in CC-based analysis cannot always be fulfilled. Furthermore, it also fails for touching characters. Kim et al. [5] combined a Support Vector Machine (SVM) and continuously adaptive mean shift algorithm (CAMSHIFT) to detect and identify text regions. Chen et al. [6] also used a SVM to identify text lines from candidates. However, experimental results show that both methods above are mainly designed for video captions. Characters in captions normally only have horizontal left-to-right orientation with almost same font size in a text line while scene text can be slanted or distorted by camera view angles with tapering characters. Thus, algorithms designed for

Manuscript received April 10, 2006. Financial support for this research is provided by the Natural Sciences and Engineering Research Council (NSERC), IRIS Precarn and the University of Western Ontario.

J. Samarabandu is with the Department of Electrical and Computer Engineering, The University of Western Ontario, London, ON., N6A 5B9, Canada (e-mail: jagath@uwo.ca)

X. Liu is with the Department of Electrical and Computer Engineering, The University of Western Ontario, London, ON., N6A 5B9, Canada (phone: 519-661-2111 ext. 87615; e-mail: xliu65@uwo.ca).

caption detection can not be directly used for scene text. Gao et al. [7] developed a three layer hierarchical adaptive text detection algorithm for natural scenes. This method has been applied in a prototype Chinese sign translation system which normally only has a horizontal and/or vertical alignment. In order to be useful in mobile robot navigation, text localization/extraction algorithms should comply with the following requirements:

1) Effectiveness: it must be robust with respect to font sizes, styles, orientations, alignment, and effects of lighting, reflections, shadows and perspective distortion.
2) Efficiency: the computation time must be small enough to be used in real-time.

In this paper, we propose an edge-based text extraction algorithm which is robust with respect to font sizes, styles, color/intensity, orientations, effects of illumination, reflections, shadows, perspective distortion, and the complexity of image background, and can quickly and effectively localize and extract text from real scenes.

II. Proposed Method

The proposed method is based on the fact that edges are a reliable feature of text regardless of color/intensity, layout, orientations, etc. Edge strength and density are two distinguishing characteristics of text embedded in images, which can be used as main features for detecting scene text. Characters are made of strokes with different orientations, which can also be interpreted as edges with different orientations. Thus, variance information of the edge orientation is also an important feature of text. If we divide an image into several regions, regions with text in them normally have much higher average values of edge density, strength and orientation variance than those of non-text regions.

The proposed method uses the edge density, strength and orientation variance to extract text from real scenes, which consists of three stages: candidate text region detection, text region localization and character extraction.

A. Candidate Text Region Detection

This stage aims to build a feature map by using three important properties of edges: edge strength, density and variance of orientations. The resulting feature map is a gray-scale image with the same size of the input image where the pixel intensity represents the possibility of text.

1) Directional filtering: In our proposed method, we use magnitude of the second derivative of intensity as a measurement of edge strength as this allows better detection of intensity peaks that normally characterize text in images. The edge density is calculated based on the average edge strength within a window. Considering effectiveness and efficiency, four orientations 0^o, 45^o, 90^o, and 135^o are used to evaluate the variance of orientations, where 0^o denotes horizontal direction, 90^o denotes vertical direction, and 45^o and 135^o are the two diagonal directions, respectively. A convolution operation with a compass operator [8] (as shown in Fig. 1) results in four oriented edge intensity images, $E_{\theta=0,45,90,135}$, which contain all the properties of edges required in our proposed method. Fig. 2 and Fig. 3 demonstrate an original test image and its corresponding results of the compass operator.

-1	-1	-1
2	2	2
-1	-1	-1

0^o kernel

-1	-1	2
-1	2	-1
2	-1	-1

45^o kernel

-1	2	-1
-1	2	-1
-1	2	-1

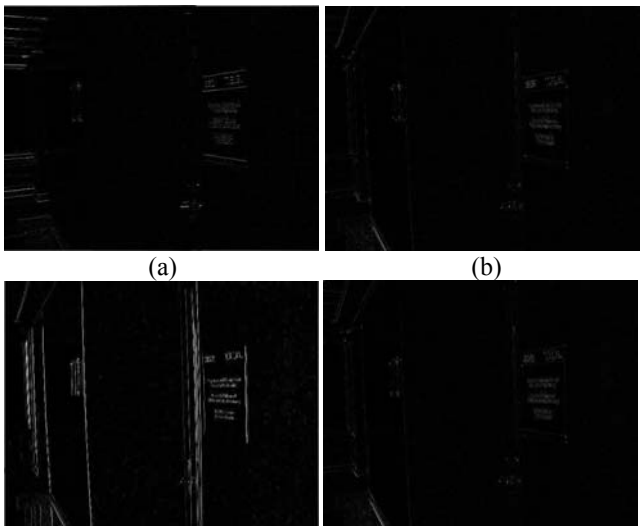
90^o kernel

2	-1	-1
-1	2	-1
-1	-1	2

135^o kernel

Fig. 1 Compass operator

Fig. 2 Test image



(a) (b) (c) (d)

Fig. 3 Edge images obtained by convolution with the compass operator (a) E_0 (b) E_{45} (c) E_{90} (d) E_{135}

2) Edge selection: Vertical edges form the most important strokes of characters and their lengths also reflect the heights of corresponding characters. By extracting and grouping these strokes, we can locate text with different heights (sizes). However, in a real scene under an indoor environment many other objects, such as windows, doors, walls, etc., also produce strong vertical edges. Thus, not all the vertical edges can be used to locate text. However, vertical edges produced by such non-character objects normally have very large lengths. Therefore, by grouping vertical edges into long and

short edges, we can eliminate those vertical edges with extremely long lengths and retain short edges for further processing.

Due to uneven illumination and/or reflections, long vertical edges produced by non-character objects may have a large intensity variance as shown in Fig. 3(c). After thresholding, these long vertical edges may become broken short edges which may cause false alarms (positives). In the meantime, uneven surfaces of characters from various lighting and shadows as well as the nature of the character shape itself (for example, letter "S") also produce broken vertical edges. In order to eliminate the false grouping caused by those broken edges, the proposed method uses a two stage edge generation method. The first stage is used to get strong vertical edges described in (1).

$$Edge_{90bw}^{strong} = |E_{90}|_z \quad (1)$$

where E_{90} is the 90^o intensity edge image which is the 2D convolution result of the original image with the 90^o kernel, $|\bullet|_z$ is a thresholding operator to get a binary result of the vertical edges. Since this stage aims to extract the strong edges, it is not very sensitive to the threshold value. In our implementation, we use Otsu's method [9] to compute a global image threshold. The second stage is used to obtain weak vertical edges described in (2) ~ (4).

$$dilated = Dilation(Edge_{90bw}^{strong})_{1\times 3} \quad (2)$$

$$closed = Closing(dilated)_{m\times 1} \quad (3)$$

$$Edge_{90bw}^{weak} = |E_{90} \times (closed - dilated)|_z \quad (4)$$

where, the morphological dilation with a rectangular structuring element of size 1×3 is used to eliminate the effects of slightly slanted edges and a vertical linear structuring element $m\times 1$ is then employed in a closing operator to force the strong vertical edges clogged. There is a tradeoff on choosing the value of m (i.e., the size of the structuring element). A small value costs less computation time at the expense of false positives. A large value increases the precise rate of detection but it increases computation cost as well. Considering the efficiency and effectiveness, from experiments, we found the value of $m = (1/25) \times width_{image}$ performs desirable detection results with an acceptable computation cost for a real time task. The resultant vertical edges are a combination of strong and weak edges as described in (5).

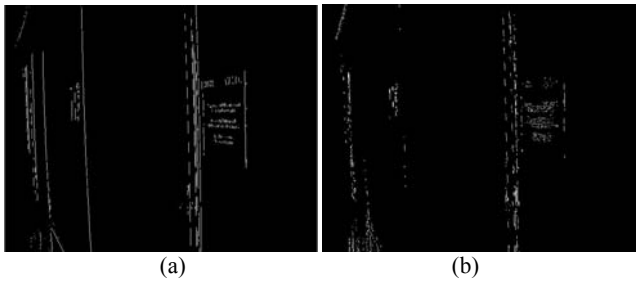
$$Edge_{90bw} = Edge_{90bw}^{strong} + Edge_{90bw}^{weak} \quad (5)$$

The results of the two stage edge generation method and the resultant vertical edge images are shown in Fig. 4 and Fig. 5, respectively. A morphological thinning operator followed by a connected component labeling and analysis algorithms are then applied on the resultant vertical edges as described in (6).

$$thinned = Thinning(Edge_{90bw})$$

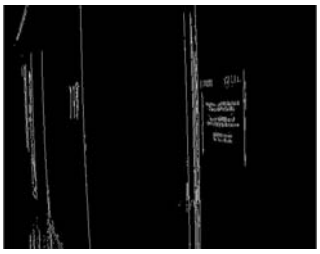
$$labeled = BWlabel(thinned, 4) \quad (6)$$

where, the morphological thinning operator makes the widths of the resultant vertical edges one pixel thick.



(a) (b)

Fig. 4 Strong and weak edges (a) strong edges obtained by thresholding of 90^o edges (b) weak edges obtained by thresholding of 90^o edges between the gaps of strong edges, gaps are obtained by the difference of a closing and dilation operation



$Edge_{90bw}$

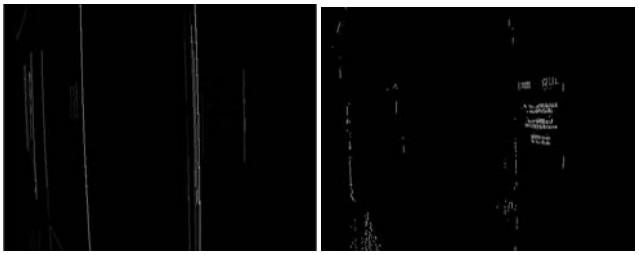
Fig. 5 Resultant vertical edges by combining strong and weak edges

The connected component labeling operator ($BWlabel$) labels the thinned vertical edges. Here, we use 4-neighbor connected component. After the connected component labeling, each edge is uniquely labeled as a single connected component with its unique component number. The labeled edge image is then processed by a length labeling process described in Fig. 6, whose purpose is to let the intensity of edge pixels reflect their corresponding lengths.

```
Algorithm 3.1
lengthlabeledImage E90^lengthlabeled labeledgelength (Image labeled)
Comment: Create an image E90^lengthlabeled with the same size of input
labeled image, while the pixels values are proportional to the areas
of the corresponding connected components
Comment: The input image is an X x Y labeled image, while pixel
values represent the number to which the corresponding connected
components belong
CCnum ← max( labeled )
i ← 1
for i ≤ CCnum
    for all pixels(x,y) ∈ labeled do
        pixnum ← 0
        for all pixels(x,y) ∈ labeled do
            if (0 < x ≤ X) and (0 < y ≤ Y)
                if labeled (x, y) == i
                    pixnum ← pixnum + 1
                endif
            endfor
        if (0 < x ≤ X) and (0 < y ≤ Y)
            if labeled (x, y) == i
            E90^lengthlabeled (x, y) ← pixnum
            endif
        endif
    endfor
    i ← i + 1
endfor
E90^lengthlabeled ← Norm (E90^lengthlabeled)
return (E90^lengthlabeled)
```

Fig. 6 Pseudo code of the edge labeling algorithm

As a result, all the pixels belonging to the same edge are labeled with the same number which is proportional to its length. The result of the length labeling process is shown in Fig. 7(a).



(a) (b)

Fig. 7 Edge labeling and selection (a) length labeled edges, where pixel intensity reflects the length of the corresponding edge (b) short edges obtained by thresholding of (a)

Since a high value in the length labeled image represents a long edge, a simple thresholding described in (7) can be used to separate short edges as shown in Fig. 7(b).

$$short_{90bw} = | E_{90}^{lengthlabeled} |_z \quad (7)$$

where $|\bullet|_z$ is a "Binary-Inv" type of thresholding function described in [10]. Since it is not easy to obtain 100% automatic correct detection, we attempt to minimize false negatives (missed detection) at the expense of false positives in our proposed method. We use a low global threshold value and use edge density and variance of orientation to refine them later on.

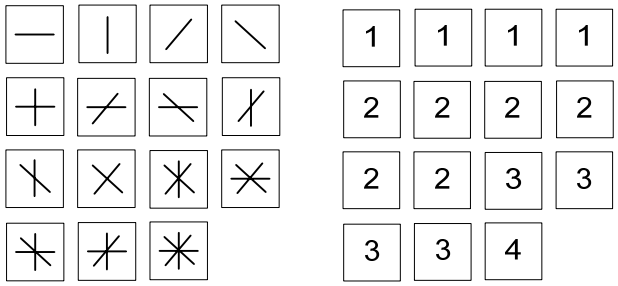
3) Feature map generation: As we mentioned before, regions with text in them will have significantly higher values of average edge density, strength and variance of orientations than those of non-text regions. We exploit these three characteristics to refine the candidate regions by generating a feature map which suppresses the false regions and enhances true candidate text regions. This procedure is described in (8) ~ (10).

$$candidate = Dilation(short_{90bw})_{m\times m} \quad (8)$$

$$refined = candidate \times \sum_{\theta=0,45,90,135} E_\theta \quad (9)$$

$$fmap\ (i,j) = N\left\{ \sum_{m=-c}^{c} \sum_{n=-c}^{c} [refined\ \ (i+m, j+n)] \times weight(i,j) \right\} \quad (10)$$

where, the morphological dilation with a $m \times m$ structuring element employed in the selected short vertical edge image ($short_{90bw}$) is used to get potential candidate text regions and the four orientation edge information ($E_\theta, \theta = 0,45,90,135$) is used to refine the potential candidate regions. In above equations, $fmap$ is the output feature map, and N is a normalization operation that normalizes the intensity of the feature map into a range of [0, 255]. Here, $weight(i,j)$ is a weight function, which determines the weight of pixel (i,j) based on the number of orientations of edges within a window as shown in Fig. 8. Namely, the more orientations the window has the larger weight the center pixel has. For example, if there are 4 orientations in a window, the center pixel of the window has the largest weight of 4. By using the weight function, the proposed method distinguishes the text regions from texture-like regions, such as window frames, wall patterns, etc. Fig. 9(a) shows the feature map of the test image.



(a) (b)

Fig. 8 Weights for windows (a) directional edges in a window (b) weight for the corresponding window

(a) (b)

Fig. 9 Feature map and clustering (a) Feature map obtained by weighted window sum of edges (b) Clustered regions obtained by thresholding of (a) followed by dilation

B. Text Region Localization

This stage localizes text regions through three steps: feature clustering, heuristic filtering and boundary boxes generation.

1) Feature clustering: Since the intensity of the feature map represents the possibility of text, a simple global thresholding can be employed to highlight those with high text possibility regions resulting in a binary image. Normally, text embedded in landmarks, such as nameplates and information signs, appears in clusters, i.e., it is arranged compactly. Thus, characteristics of clustering can be used to localize text regions for indoor mobile robot navigation. A morphological dilation operator can easily connect the very close regions together while leaving those whose positions are far away to each other isolated. In our proposed method, we use a

morphological dilation operator with a 7×7 square structuring element to the binary image obtained from the previous step to get joint areas referred to as text blobs. Fig. 9(b) shows the result of feature clustering.

2) Heuristic filtering: In order to be seen easily by human (i.e., intuitively visible), characters in information signs and nameplates usually are either large enough or small but appear in clusters. As a result, the area as well as the width and height ratio of each text blob can not be too small. In our proposed method, two constraints are used to filter out those blobs which do not contain text. The first constraint is used to filter out all the very small isolated blobs described in (11), where it uses a relative area value instead of the absolute value for each blob. Therefore, there is no specific limitation for the font size, which makes more sense for the real world scenes.

$$Area_{region} \geq (1/20) \times Area_{\max} \tag{11}$$

The second constraint described in (12) filters out those blobs whose widths are much smaller than corresponding heights. A threshold of "0.2" is considered for the worst case, such as single western letter "I" or Arabic digit "1". Experiment results show that this value (0.2) is not sensitive to different situations. The heuristic filtered image is shown in Fig. 10(a).

$$Ratio_{w/h} = \frac{Width_{region}}{Height_{region}} \geq 0.2 \tag{12}$$

3) Boundary boxes generation: The retaining blobs are enclosed in boundary boxes. Four pairs of coordinates of the boundary boxes are determined by the maximum and minimum coordinates of the top, bottom, left and right points of the corresponding blobs. Fig. 10(b) shows the boundary boxes of text regions. In order to avoid missing those character pixels which are lying near or outside of the initial boundary, width and height of the boundary box are padded by a small amounts.

(a) (b)

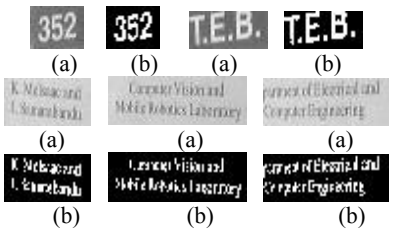
Fig. 10 Heuristic filtered regions and Bounding boxes (a) Heuristic filtered regions (b) Bounding Boxes

C. Character Extraction

Existing OCR (Optical Character Recognition) engines can only accept simple inputs, i.e., it only deals with printed characters against clean backgrounds and can not handle characters embedded in shaded, textured or complex backgrounds. The purpose of this stage is to extract accurate binary characters from the localized text regions so that we can use the existing OCR directly for recognition. In our proposed method, we use uniform white character pixels in a pure black ground by using (13).

$$T = \bigcup_{i=1\ldots} \overline{|SUB_i|}_z \tag{13}$$

In the above equation, T is the text extracted binary output image. U is an union operation. SUB_i are sub-images of the original image, where i indicates the number of sub-images. Sub-images are extracted according to the obtained boundary boxes. $\overline{|\bullet|}_z$ is a thresholding algorithm which segments the text regions into white characters in a pure black background. Since the backgrounds in sub-images are much simpler than the whole image, histograms of sub-images are normally bimodal. Threshold z can be determined by the dips between two peaks in corresponding sub-image histograms. Fig. 11 and Fig. 12 illustrate the segmented sub-images and extracted text image.



(a) (b) (a) (b)

(a) (a) (a)

(b) (b) (b)

Fig. 11 Test image showing both dark and light text (a) Sub-Images (b) Extracted text Strings

(a) (b)

Fig. 12 Test result (a) Original Test Image (b) Text Extracted Image

III. Experimental Results and Discussion

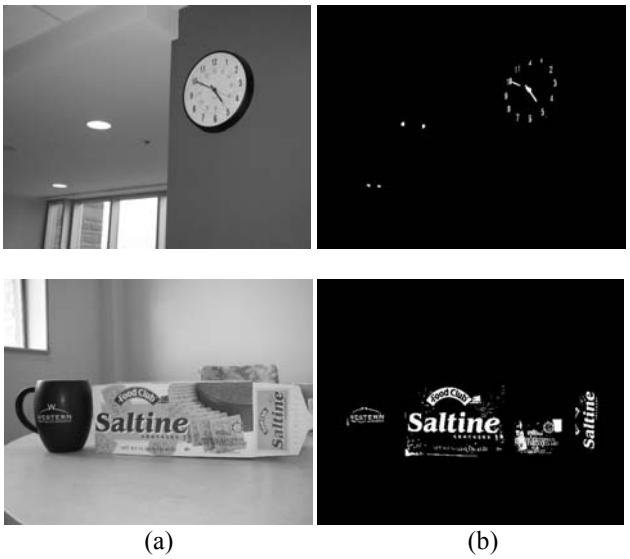
In order to evaluate the performance of the proposed method, we use 25 test images with different font sizes, perspective and alignment under different lighting conditions. Fig. 13~ 16 show some of the results, from which we can see that our proposed method can extract text with different font sizes, perspective, alignment, any number of characters in a text string under different lighting conditions.

(a) (b)

Fig. 13 Results of images with different font sizes (a) Original images (b) Extracted text

(a) (b)

Fig. 14 Results of images with strong reflections and different perspective distortion (a) Original images (b) Extracted text



(a) (b)

Fig. 15 Results of images with different orientation alignments (a) Original images (b) Extracted text

(a) (b)

Fig. 16 Results of images with single character and strong reflections (a) Original images (b) Extracted text

Although there is no generally accepted method that can be used to evaluate the performance of the text localization and extraction method, we use several metrics used by others to facilitate a meaningful comparison. In our proposed method, we compute the accuracy of the algorithm output by manually counting the number of correctly located characters, which are regarded as ground-truth. Precision rate and recall rate are quantified to evaluate the performance given by (14) and (15).

$$\Pr ecision = \frac{CorrectlyL\ ocated}{CorrectlyL\ ocated + FalsePosit\ ive} \times 100\% \quad (14)$$

$$\mathrm{Re}\, call = \frac{CorrectlyL ocated}{CorrectlyL ocated + FalseNegat ive} \times 100\% \quad (15)$$

Table I shows the performance comparison of our proposed method with several existing methods, where our proposed method shows a clear improvement over existing methods. In this table, the performance statistics of other methods are cited from published work. Considering a 95% confidence intervals, it appears that Wang et al. [2], Xi et al. [11] and Gllavata et al. [12] have a similar performance as the proposed method. However, those methods are applied for different types of images. In Table II, we also provide another evaluation using "false positive rate" as defined in (16) for comparison. Although (16) is not a standard definition, it intuitively interprets the false positive rate.

$$FalsePosti\ ve = \frac{FalsePosti\ ve}{CorrectlyL\ ocated + FalseNegat\ ive} \times 100\% \quad (16)$$

The proposed method has been implemented in C++ with the Intel OpenCV (Open Computer Vision) library [10]. The overall average computation time for 25 test images (with 480x 640 resolution) on a PC with a 1.5 GHz Pentium IV CPU and 256MB memory is 2.497s (std dev.=0.697), which includes entire run time including image reading, computation as well as image display. Although around 3 seconds of average processing time is very small, we have not optimized the proposed method, where the edge labeling and length labeling operations take the most time about 1.891s (std dev.=0.694). Our goal is to optimize it less than 1 second, which is based on the initial guess of the real time requirement.

TABLE I
PERFORMANCE COMPARISON

Method	Test Images No.	Text Blocks No.	Correctly Located No./(%)	Precision Rate (%)	Recall Rate (%)
Proposed method	25	208	201	91.8	96.6
Wang et al. [2]	325	3597	3314	≈89.8	≈92.1
Kim et al. [13]	--	839	645	63.7	82.8
Agnihotri et al. [14]	293	434	370	85.8	85.3
Xi et al. [11]	90	244	231	88.5	94.7
Wolf et al. [15]	60	371	--	--	93.5
Gao et al. [7]	--	823	93.3%	≈89.9	--
Gllavata et al. [12]	326	1104	979	83.9	88.7
Messelodi et al. [2]	100	432	394	--	91.2

TABLE II
FALSE POSITIVE COMPARISON

Method	Image Type	False Positive Rate (%)
Proposed method	Indoor scene text	5.0
Wang et al. [2]	Outdoor scene text	10.5
Xi et al. [11]	Text captions	12.3
Gllavata et al. [12]	Overlay text	17.0

IV. CONCLUSION AND FUTURE WORK

In this paper, we propose an edge-based text region extraction algorithm which can localize and extract text from real scenes. Experimental results show that our proposed method is very effective and efficient in localizing and extracting text-based features. It is robust with respect to font sizes, styles, color/intensity, orientations, alignment/layout and perspective of text and can be used in mobile robot navigation under an indoor environment to detect text-based landmarks in real-time.

A. Contributions

Although we did not conduct controlled experiments to study the behavior of the proposed method for each individual parameters, our test image set contains a wide variety of images and the performance was excellent overall. The proposed method is:

1) Not sensitive to image color/intensity. Unlike most of the connected component based methods, it does not need any assumptions for color/intensity within the same character.
2) Robust with respect to aspect to font sizes, and styles, uneven illumination, perspective and reflection effects.
3) Unlike commonly used connected component based methods which analyze every single character, the proposed method only analyzes text blocks. Therefore, it is computationally efficient, which is essential for real-time applications.
4) Distinguishes text regions from texture-like regions, such as window frames, wall patterns, etc., by using the variance of edge orientations.
5) Binary output can be directly be used as an input to an existing OCR engine for character recognition without any further processing.

B. Future Work

In vision-based mobile robot navigation system, landmark recognition is a very important aspect as landmarks are the most significant features in the unknown environment. Furthermore, landmarks can guide a robot and help it localize itself. Our main future work is landmark recognition which involves:

1) Using a suitable existing OCR technique to recognize the extracted text from landmarks.
2) Applying an affine rectification to correct the perspective distortion caused by a camera view angle in order to increase the text recognition rate.
3) Using an image tracking technique, such as Kalman filter, to track the detected ROIs.

REFERENCES

[1] K. Jung, K. I. Kim, and A. K. Jain, "Text information extraction in images and video: a survey," Pattern Recognition, vol. 37, no. 5, pp. 977–997, 2004.
[2] K. Wang and J. A. Kangas, "Character location in scene images from digital camera," Pattern Recognition, vol. 36, no. 10, pp. 2287–2299, 2003.
[3] R. Lienhart and A. Wernicke, "Localizing and segmenting text in images and videos," IEEE Transaction on Circuits and Systems for Video Technology, vol. 12, no. 4, pp. 256–268, 2002.
[4] Y. Zhong, K. Karu, and A. Jain, "Locating text in complex color images," Pattern Recognition, vol. 28, no. 10, pp. 1523–1535, 1995.
[5] K. I. Kim, K. Jung, and J. H. Kim, "Texture-based approach for text detection in images using support vector machines and continuously adaptive mean shift algorithm," IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI), vol. 25, no. 12, pp. 1631–1639, 2003.
[6] D. Chen, H. Bourlard, and J.-P. Thrian, "Text identification in complex background using svm," in computer vision and pattern recognition (CVPR'01), ser. Proceedings of the Int. Conf. on, vol. 2, December 2001, pp. 621–626.
[7] J. Gao and J. Yang, "an adaptive algorithm fot text detection from natural scenes," in Computer Vision and Pattern Recognition, 2001. CVPR 2001, ser. Proceedings of the 2001 IEEE Computer Society Conference on, 2001, pp. II–84–II–89.
[8] A. K. Jain, Fundamentals of Digital Image Processing. Englewood Cliff, NJ: Prentice Hall, 1989, ch. 9, pp. 356–357.
[9] N. Otsu, "A threshold selection method from gray-level histograms," IEEE Transactions on Systems, Man, and Cybernetics, vol. 9, no. 1, pp. 62–66, 1979.
[10] Intel r Open Source Computer Vision Library, Available: http://www.sourceforge.net/projects/opencvlibrary (URL).
[11] J. Xi, X. S. Hua, X. R. Chen, L. Wenyin, and H. J. Zhang, "A video text detection and recognition system," in Multimedia and Expo, 2001. ICME 2001, ser. IEEE International Conference on, 2001, pp. 873–876.
[12] J. Gllavata, R. Ewerth, and B. Freisleben, "A robust algorithm for text detection in images," in Image and Signal Processing and Analysis, 2003. ISPA 2003, ser. Proceedings of the 3rd International Symposium on, 2003, pp. 611–616.
[13] K. C. Kim, H. R. Byun, Y. J. Song, Y. M. Choi, S. Y. Chi, K. K. Kim, and Y. K. Chung, "Scene text extraction in natural scene images using hierarchical feature combining and verification," in Pattern Recognition, 2004, ser. ICPR 2004. Proceedings of the 17th International Conference on, vol. 2, Aug. 2004, pp. 679–682.
[14] L. Agnihotri and N. Dimitrova, "Text detection for video analysis," in Content-Based Access of Image and Video Libraries, 1999. (CBAIVL '99), ser. Proceedings. IEEE Workshop on, 1999, pp. 109–113.

[15] C. Wolf, J. M. Jolion, and F. Chassaing, "Text localization, enhancement and binarization in multimedia documents," in Pattern Recognition, 2002, ser. Proceedings. 16th International Conference on, vol. 2, Aug. 2002, pp. 1037–1040.
[16] S. Messelodi and C. M. Modena, "Automatic identification and skew estimation of text lines in real scene images," Pattern Recognition, vol. 32, no. 5, pp. 791–810, 1999.

Jagath Samarabandu received B. Sc (Eng) in Electronics and Telecommunication with first class honours from the University of Moratuwa, Sri Lanka in 1982. He was awarded the Fulbirght scholarship in 1987 for postgraduate study. He received M.S and Ph.D. degrees in Electrical Engineering from State University of New York at Buffalo in 1990 and 1994 respectively.

He held a post-doctoral position in the Dept. of Biological Sciences at SUNY-Buffalo untill 1997 and joined Life Imaging Systems Inc. as a senior software engineer in 1997. He has been with the Department of Electrical and Computer Engineering at the University of Western Ontario since 2000. His research interests include image analysis, pattern recognition and intelligent systems.

Xiaoqing Liu received the BEng degree in Electrical Engineering from Xi'an Jiaotong University, Xi'an, China, in 1998 and received the M.E.Sc degree in Electrical and Computer Engineering at the University of Western Ontario, London, ON, Canada in June 2005.

She was employed with the SLDI Design Institute as an electrical engineer from 1998 to 2002 and is currently pursuing the Ph.D degree in Electrical and Computer Engineering at the University of Western Ontario, London, ON, Canada. Her research interests include Image Processing, Pattern Recognition and Computer Vision.