QUADRILATERAL SIGNBOARD DETECTION AND TEXT EXTRACTION

Angela Tam, Hua Shen, Jianzhuang Liu, and Xiaoou Tang

Department of Information Engineering
The Chinese University of Hong Kong, Hong Kong

ABSTRACT

There are numerous rectangular signboards indoors and outdoors. The information contained in them, such as signs and text, is useful for robot navigation. In this paper, a system for automatically detecting signboards and extracting text from them is presented. First it locates the signboard's position on the image by the Hough transform, edge density checking and quadrilateral finding. Then, with a geometric transformation, the system changes the signboard from an arbitrary quadrilateral to a rectangular shape. Finally, it performs segmentation to extract the characters on the signboard. In our system, the signboards can have large variations such as rotation and scaling and are not limited to be (nearly) rectangular in the images. Experiments show that our system is quite effective.

1. INTRODUCTION

Signboards provide important information in our daily life. This information is also useful to guide the navigation of an autonomous robot indoors and outdoors. In this paper, a system for automatically detecting signboards and extracting text in them is presented. There have been many papers discussing about sign detection. Most of them focus on finding traffic signs and car plates, and use shape matching and color processing for their tasks [1] [2]. In this paper the signboards are not restricted to be traffic signs and car plates. Hough transform [3] and other processing schemes are used for our goal. In the system, the scene images are captured by a digital video camera. With the observation that most of the signboards are in rectangular shape, we focus on the detection of quadrilateral signboards in images. In addition, it is assumed that the whole signboard is present in the image. Too small signboards are not detected because the information contained in them is difficult to recognize. The signboards are allowed to have large variations such as rotation and scaling and are not limited to be rectangular or nearly rectangular in the images.

The rest of this paper is organized as follows. Section 2 presents the sign locating method. Hough transform is employed here to detect the boundaries, and a scheme is proposed to find quadrilaterals. In Section 3, a geometric transformation is given to turn a quadrilateral into the rectangular shape. Section 4 discusses the segmentation of text from found signboards. Finally, the evaluation of our method and the conclusion are presented in Sections 5 and 6.

2. SIGNBOARD LOCATING

2.1. Pre-processing and Hough transform

Because of the large variety of colors on signboards, we don't consider color information in this application. So all the images are turned into grayscale at first. Before applying the Hough transform to finding the edges, noise reduction with smoothing and media filters and Sobel edge detection are performed on an image. The edge map after these pre-processings is shown in Figure 1.

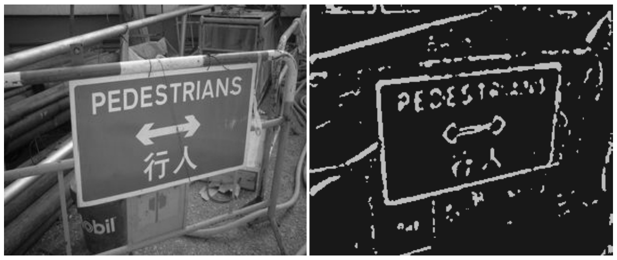


Figure 1. Edge map after pre-processing

Hough transform is an effective way of detecting straight lines in images. Figure 2 shows one result of the Hough transform, from which we can obtain candidate boundary lines of the signboard. The gray dotted lines in Figure 2 denote the lines detected.

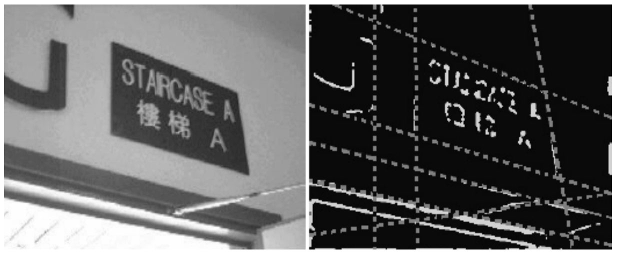


Figure 2. Result of Hough transform

2.2. Segment verification

Even for a straight line that comes from a true edge, only a segment of the line is the edge. Therefore, it is necessary to cut the lines into segments. This is achieved by finding the intersection points between the straight lines obtained and forming segments between two intersection points.

The segments with only one end belonging to an intersection point should be removed, because they cannot be part of a closed boundary. Then, by checking edge points lying on the corresponding segment, we can get the segments most likely to form the boundary of a signboard, as shown in Figure 3.

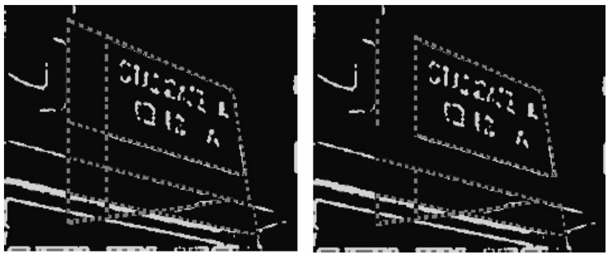


Figure 3. Checking possible segments that form the boundary of a signboard. Left: impossible segments are removed. Right: segments are remained after density checking.

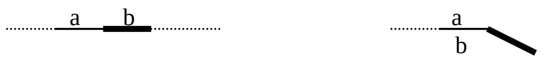
2.3. Quadrilateral finding

Finding quadrilateral is equivalent to finding a close loop with four corners. We define the following types for the possible segments that belong to a quadrilateral: *segment*, *corner*, *alternative* and *unknown*.

Type *segment* means a segment is part of the side of the quadrilateral and its end point is not a corner. Type *corner* means that the end point of a segment is a corner. Type a*lternative* is used when there is more than one segment with the same intersection point as the previous end point. In this case, one of them will be chosen to be *segment* or *corner*, while the others will be *alternative* so that when the searching cannot go further in the later steps, it can come back here and start searching again (see below). Type *unknown* is used when a segment is newly added to the list. For the newly chosen segment, its true type can only be determined after the next segment is found.

At first, we choose the top left most intersection point, and which has at least two segments coming from it. Then searching starts from the longest segment, which is set to be *unknown*, while the other lines are set as *alternative*. The number of corners counted is set to 1.

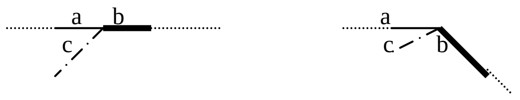
Then continue finding another segment that has the same point as the end point of the previous segment. If the segment found is of the same equation as the previous one, the previous one will be set as *segment*; otherwise set the previous segment *corner*. In this case, increase number of corners by 1. The new segment will be set as *unknown*, as shown in Figure 4.



a: Previous segment.
Set as *segment* in left image,
corner in right image.
b: Newly found segment. Set as *unknown*.

Figure 4. How to set *segment* and *corner*

If there is more than one segment qualified, then the one of the same equation will be chosen, and the others will be set *alternative*, shown in Figure 5. However, if all the segments are not of the same equation as the previous segment, we will chose one according to this criterion: finding the segment that makes an angle the nearest to 90 degree with the previous segment, which then is set to type *corner*. The other segments, except the one chosen, will be set to *alternative*. The number of corners increases by 1. Again, the segment chosen is set to *unknown*. Then continue this searching, regarding the current segment as the previous segment.

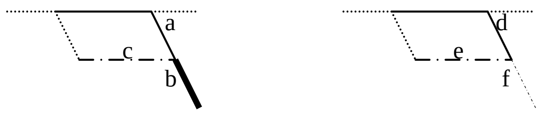


a: Previous segment.
Set as *segment* in left image,
corner in right image.
b: Newly found and chosen segment.
Set as *unknown*.
c: Newly found but not chosen segment.
Set as *alternative.*

Figure 5. Meeting with two qualified segments

In cases that no qualified segment can be found, trace back until a segment labeled as *alternative* is found. Delete appropriate segments chosen. If a corner segment is deleted, the number of corners will be decreased by 1.

When the number of corners exceeds four but still no closed loop found, or quadrilateral found with less than four corners, the deletion of the list will take. Even if a quadrilateral found is with exactly four corners, the length of each side is still checked. If it is shorter than a threshold, it is declared to be invalid and deletion takes place.



Left: a is previous segment, set as *segment*.
b is chosen, set as *unknown*.
c is not chose, set as *alternative.*
Right: Since b in left cannot go further, f in right is deleted.
d is previous segment, changed from *segment* to *corner*.
e is chosen, changed from *alternative* to *unknown*.

Figure 6. Trace back to *alternative* segment

Deletion provides chances to search again when a wrong segment was previously selected. When the list is deleted, the searching goes back to the starting point. If no other alternative ways exist, it will choose another starting point, which is the next left top most intersection point. To prevent the searching from being too long, at most ten tries are allowed. If a valid quadrilateral still cannot be found after ten tries, or there is no other intersection point to start, this image is declared to have no signboard contained. Then the whole process will be terminated.

Unless the image is declared to have no signboard contained, otherwise, the searching process will be repeated until the requirements of a valid quadrilateral are met. The requirements are: it goes back to the starting point, the number of corners is exactly four, and the length of each side is greater than the threshold.

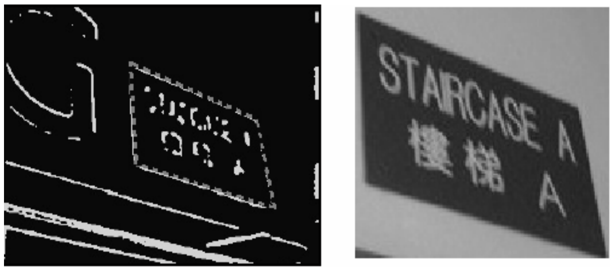


Figure 7. Result of finding a quadrilateral

When the valid quadrilateral is finally found, we cut the smallest rectangle that bounds the quadrilateral, as shown in Figure 7.

3. GEOMETRIC TRANSFORMATION

Because of different viewing angles of images, a rectangular signboard is usually distorted to a quadrilateral shape. It is necessary to transform the signboard back to an undistorted rectangle. In the system, the geometric transformation mentioned in [4] is used for our goal.

Suppose an image *f* with pixel coordinates (*x*,*y*) undergoes geometric distortion to produce a distorted image *g* with pixel coordinates (*x'*,*y'*). To recover the distorted image *g*(*x'*,*y'*) back to *f*(*x*,*y*), we must have information on some tie-points, which are a subset of pixels location in *g* and in *f* . In our system, we have the information of four tie-points. They are the corners' co-ordinates of the signboard from the output of the signboard locating part. Suppose the geometric distortion is modeled by a pair of bilinear equations such that

$$x' = c_1 x + c_2 y + c_3 xy + c_4$$
$$y' = c_5 x + c_6 y + c_7 xy + c_8 .$$

By substituting the four pairs of tie-points into the equations, a total of eight equations are obtained. Thus the eight constants in the bilinear equations can be found. After finding the eight constants, all the co-ordinates of the distorted and the recovered images can be mapped between. Two results are shown in Figure 9.

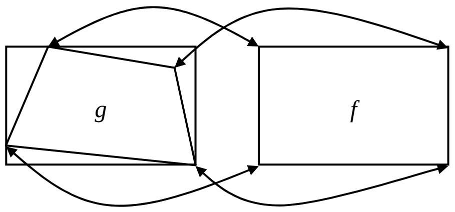


Figure 8. Illustration of geometric transformation

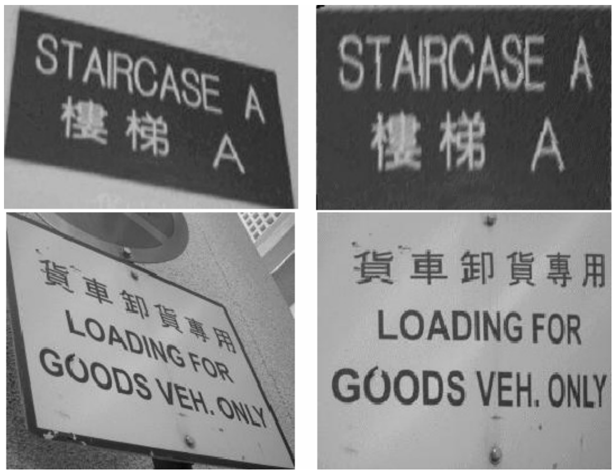


Figure 9. Results of geometric transformation

4. SEGMENTATION

Segmentation is needed in our application to separate the characters from the background of the signboard. First, a suitable threshold is found for a signboard based on the optimal global and adaptive thresholding [4]. Figure 10 gives one result of the thresholding.

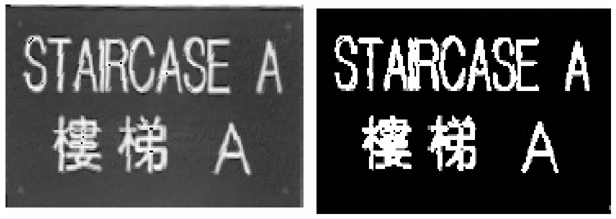


Figure 10. Result after thresholding

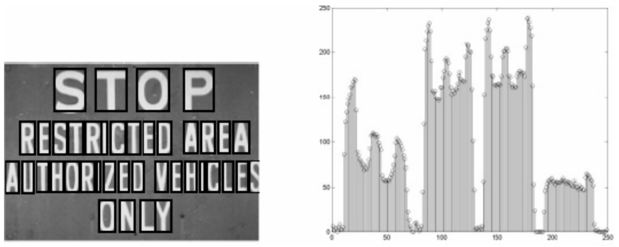


Figure 11. Result of segmentation and horizontal projection

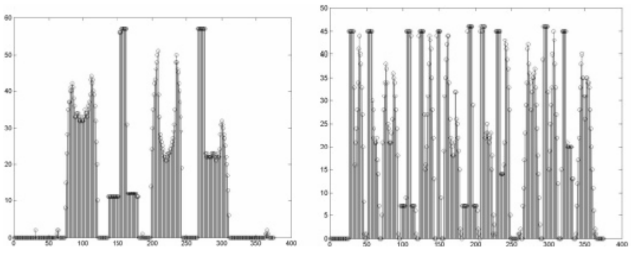


Figure 12. Vertical projection for line 1 and 2

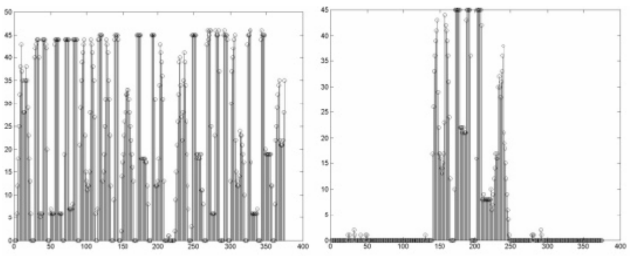


Figure 13. Vertical projection for line3 and 4

The segmentation of characters (letters) is based on horizontal and vertical projections of the thresholded signboard image. The horizontal object strips are found at the positions where the projection values are below a threshold and are local minima. Thus horizontal projection can be used to segment the lines of characters. Similarly, vertical projection of each line gives the information where to separate characters (letters) in the same line [5]. Figures 11, 12 and 13 show the processing.

5. EVALUATION

In the experiment, the images are classified into four sets: clear or slightly distorted images, seriously distorted images, no-signboard images, and out of focus images. The performance is listed below:

From Table 1, we can see that the system is quite good for clear and slightly distorted images. Most of the problems are because of the edges. When the image is blurred or when there are too many strong background edges in it, the performance drops. Therefore, the edge finding stage is very important and sensitive. Correctly detecting edges will improve the system very much.

Table 1. Performance of signboard locating

Clear or slightly distorted images	
Number of signboard	107
Correctly detected signboards	91
Accuracy	85.0467%

Seriously distorted images	
Number of signboard	14
Correctly detected signboards	10
Accuracy	71.4286%

No-signboard images	
Number of no-signboard	7
Correctly detected signboards	4
Accuracy	57.1429%

For segmentation, we tested the images in uniform illumination and non-uniform illuminations. The results are shown in Table 2.

Table 2. Performance of segmentation

Prospective Characters under uniform illumination	
Number of characters	18
Correctly matched characters	15
Accuracy	83.3333%
Prospective Characters under non-uniform illumination	
Number of characters	47
Correctly matched characters	38
Accuracy	80.8511%

For non-uniform illuminated images, the accuracy is a little worse. This is because the histograms are often not bimodal. The presence of pseudo peaks lead to misjudge of background gray levels. But the performance is still good.

6. CONCLUSION

In this paper, we have presented a system that can detect characters (letters) from quadrilateral signboards in images. It can be used to guide the navigation of an autonomous robot. The system mainly consists of three parts: signboards locating, geometric transformation and characters and letters segmentation. The experiments have shown encouraging results. The future work will be in more robust detection of edges and character or word recognitions.

7. ACKNOWLEDGMENT

This work was partly supported by the Research Grants Council of the Hong Kong Special Administrative Region under Grants CUHK4378/99E, CUHK4357/02E, AoE/E-01/99, and a CUHK Direct Grant.

8. REFERENCES

[1] G. Piccioli, E. D. Micheli, P. Parodi, and M. Campani, “Robust road sign detection and recognition from image sequences,” In Proceedings of the Intelligent Vehicles ’94 Symposium, pages 278-283, 1994.

[2] E. Ryung Lee, P. K. Kim, and H. J. Kom, “Automatic recognition of a car license plate using color image processing,” In Proceedings of IEEE International Conference on Image Processing, volume 2, 1994.

[3] L. G. Shapiro and G. C. Stockman, “Computer Vision”, Prentice Hall, 2001.

[4] R. C. Gonzalez and R. E. Woods, “Digital Image Processing,” Prentice Hall, 2002.

[5] X. Gao, and X. Tang, “Automatic News Video Caption Extraction and Recognition,” In Proceedings of Second International Conference on Intelligent Data engineering and Automated Learning, 2001.