2011 International Conference on Document Analysis and Recognition

Recognizing Text Elements for SVG Comic Compression and its Novel Applications

Chung-Yuan Su
Dept. of Engineering Science
National Taiwan University
Taipei, 10617, Taiwan
d98525007@ntu.edu.tw

Ray-I Chang
Dept. of Engineering Science
National Taiwan University
Taipei, 10617, Taiwan
rayichang@ntu.edu.tw

Jen-Chang Liu
Dept. of Computer Science
National Chi Nan University
Nantou, 54561, Taiwan
jcliu.ncnu@gmail.com

***Abstract*—SVG (scalable vector graphics) has become the standard format for 2D graphics in HTML5. Although some image-to-SVG conversion systems had been proposed, the sizes of files they produced are still large. In [1], we proposed a new system to convert raster comic images into vector SVG files. The compression ratio is better than the previous methods. However, these methods do not process text in raster images. In this paper, we improve our system to recognize text elements in the comic and use these text elements to provide better compression and novel applications. The proposed method uses SCW (sliding concentric windows) and SVM (support vector machine) to identify text regions. Then, OCR (optical character recognition) is applied to recognize text elements in those regions. Instead of encoding the text regions as vectors, the text elements are embedded in the SVG file along with their coordinate values. Experimental results show that we can reduce the file sizes to about 52% of the original SVG files. Using these text elements, we can translate comics into other languages to provide multilingual services easily. Text/content-based image search can be supported efficiently. It can also provide a novel application system for story teller.**

Keywords- text detection, SVG, vector compression, SCW segmentation

I. INTRODUCTION

Since Internet grows rapidly in recent years, multimedia transmission has become an important issue on networks. With the mature development of display technology, the media information is delivered not only on computers but also on portable handheld devices. Current raster formats, such as JPEG, BMP, PNG and TIFF, are all pixel-based image formats. In order to rescale the images to be displayed at different resolutions, interpolation of pixel values needs to be applied. However, these operations cause the degradation of image qualities, such as jagged or fuzzy edges, as shown in Fig. 1(c). Furthermore, raster formats cannot represent the meaning of images that can be indexed by image search engines. Vector formats provide an alternative to compensate the mentioned drawbacks.

SWF, PDF and SVG (scalable vector graphics) are popular vector formats nowadays. Among them, SVG [2] is a revolutionary new graphic standard that has been developed by W3C. It is an XML-based language that describes 2D graphic with vector format. Standardization of SVG satisfies the growing demands on a dynamic, scalable, cross-platform and complicated interactive usages of Internet. The current version of SVG is 1.1, and SVG Tiny (SVG-T) 1.2 is suitable for mobile devices. Compared to raster format, SVG format has the following advantages:

- High compression ratio: because these files based on XML contain many repeated fragments of text, it suited to be compressed by gzip.
- Arbitrarily scalable: vector formats record the Bezier curves. When the image needs to be rescaled, only the positions of the control points have to be changed. It maintains a good perceptual quality in different resolutions of display devices, as shown in Fig. 1(b).
- Easily editable: unlike SWF and PDF formats, SVG files can be modified using traditional text editors without using special image editors.
- Embedding of raster images: use the <image> tag to embed raster images.
- Being indexed by search engines: search engines can read XML files, so it can retrieval image based on specific text information.

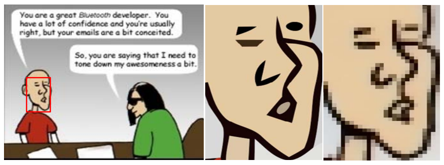

You are a great Bluetooth developer. You have a lot of confidence and you're usually right, but your emails are a bit conceited.
So, you are saying that I need to tone down my awesomeness a bit.
(a)
(b)
(c)

Figure 1. (a) Original image. (b) Vector image × 4. (c) Raster image × 4.

Several raster to vector conversion software have been developed, such as Stanford Vector Magic [3], Adobe Illustrator and Corel Draw. The sizes of files they produced are still large and consume most of time in image rendering. If we limit the file size, image quality degrades accordingly. In our previous work [1], a novel image processing algorithm is proposed to reduce the SVG files size. However, these methods do not consider processing text in raster images. Because vectorization converts a raster image to the coordinates of Bezier curves and lines, it wastes storage to use these elements to represent text, especially for comic images with lots of dialogs, as shown in Fig. 1(a). In this paper, we proposed a system to extract text from raster comic images and then OCR (optical character recognition) task is applied. After erasing text, we convert raster format into SVG vector format. Finally, the OCR results with their coordinate values are embedded using <text> elements in the SVG file. The main contributions of the proposed system are listed as follows. (1) It can further reduce the sizes of SVG files. (2) The OCR results from comic images can be translated into other languages to provide multilingual services easily. (3) It can support text/content-based vector image search efficiently. (4) It can support story teller functionality. Experimental results show that we can reduce the file sizes to about 52% as compared to original SVG files. Novel applications such as SVG image search and story teller are examined.

II. Related Works

The process of raster-to-vector conversion often involves three steps. Firstly, it determines color regions and merges similar color regions. Then it detects their edges. Finally, it fills in colors and constructs a complete vector image. Consequently, the number of colors and the detail of images will affect the converted file sizes. In [1], we adopt Autotrace [4] to vectorize comic images, and then use the vector contour searching algorithms for removing extra spaces, combining the slope of clips, and merging similar color regions to compress the vector images. This method achieves high image quality and compression ratio. Battiato *et al.* [5] used Data Dependent Triangulation (DDT) method to vectorize graphics. Pixel neighborhood is approximated by triangles. In rendering stage, they merge adjacent triangles when they have the similar filling color in order to reduce file size. Zhang *et al.* [6] proposed a new trapped-ball method combined with detected decorative lines to segment cartoon image animation. Temporal coherence is used to extract a unified background or foreground before vectorization to reduce file size and achieve a high quality vectorized result. These approaches often merge or delete as many redundant objects as possible to reduce the file size. One advantage is that it allows perceptually better image to be transmitted within the limited bandwidth, and the other advantage is that it can reduce image rendering time. However, all of them do not process text in the image before vectorization.

Text contained in the images often convey useful message to people. It is important to let readers from different countries to realize text information directly. Canedo-Rodriguez *et al.* [7] focused on the signboard images. Their method extracts text using DCT coefficients and then text is recognized by OCR and translates from English to Spanish. Nakai *et al.* [8] proposed a retrieval method for images of documents in various languages. Local and additional descriptors are used as text features. Their purpose is similar to one of our contributions. Another useful application in [9] is a device with a head-mounted video camera for the blind. Particle filter is applied to track text.

III. Proposed Method

The input of our system is a raster comic image. At first, the system uses SCW (sliding concentric windows) [10] to segment text as our ROI (region of interest). After using morphology and CCL (connected component labeling), the text candidate regions can be determined. Several features, including aspect ratio, orientation, edge density variation and coverage are calculated and then forwarded to SVM (support vector machine) to classify real text regions. Next, OCR task is applied to real text areas and the coordinates of text elements are recorded. We erase text in the comic image and then convert the text-free raster image into SVG vector format. Finally, the recognized text elements and their coordinate values are embedded as readable characters into SVG files. The proposed method can reduce the sizes of SVG files compared with our previous method [1]. Moreover, the embedded text elements can provide opportunities for novel applications. Some applications will be discussed in Section IV. Fig. 2 shows the system flowchart. We now describe each process step in detail.

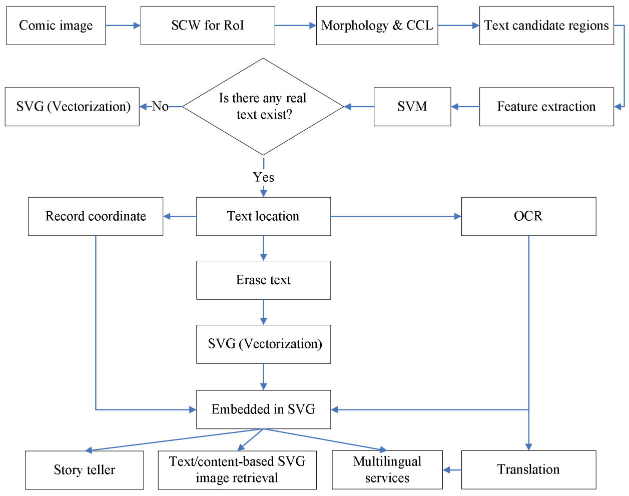


Figure 2. Flowchart of the proposed system.

A. SCW Segmentation Method

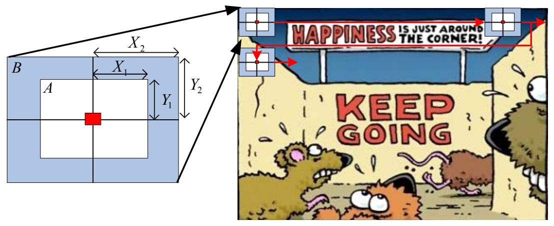
Text regions in the comic image have significant properties, such as irregularities in texture and abrupt changes in local intensity. In this paper, we adopt SCW [10] to segment text from the comic image. The algorithm is composed of the following steps:

- Produce two concentric windows A and B of size $2X_1 \times 2Y_1$ and $2X_2 \times 2Y_2$, respectively. The windows are shown in Fig. 3(a).
- Start to scan the comic image from left to right and from up to bottom, as shown in Fig. 3(b). Then, calculate mean values or standard deviations in windows A and B.

When the ratio of mean values or standard deviations of windows A and B is larger than a threshold, then the central pixel of the windows is considered as a text (set to 1), otherwise, it is taken as a non-text (set to 0). The equation can be expressed as below.

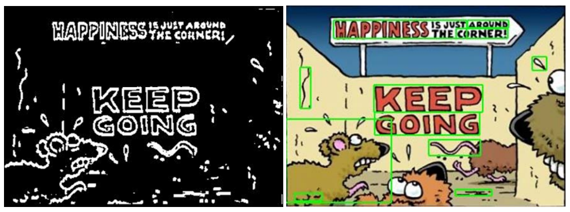
$$I_o(x,y) \Rightarrow \begin{cases} I_s(x,y) = 1, & if\ \dfrac{M_A}{M_B} > T \\ I_s(x,y) = 0, & otherwise \end{cases}, \qquad (1)$$

where I_o is the original comic image, I_s is the binary image after SCW, T is the threshold and M represents the mean value or the standard deviation. In our experiment, we choose to use mean value because it has lower computational complexity, and the performance is similar to that using standard deviation. Then, we apply morphology operation and CCL to I_s. Since we focus on the horizontal alignment of text, the rectangular structure elements of dilation and erosion are chosen to be 1×6 and 2×3, respectively. In the CCL, 4-neighborhood is used to avoid over-connection. Components that have less than 100 pixels are removed because of the limitation of OCR applied in the system. We then obtain the I_c image which contains text candidate regions. Fig. 4(a) shows the SCW result, and Fig. 4(b) shows the text candidate regions with the minimum bounding box (green color).



(a) (b)

Figure 3. (a) Concentric windows A and B. (b) Scanning the raster comic image.



(a) (b)

Figure 4. (a) The result of SCW. (b) Text candidate regions.

B. Connected Components Based Analysis and SVM Classifier

As shown in Fig. 4(b), some text candidate regions are false alarms. Therefore, statistical features on the connected components can be analyzed to distinguish the real text region from the non-text region. The following features are proposed:

1) Aspect ratio of the text candidate regions

The width W and height H can be found from the minimum bounding box of the text candidate region. The aspect ratio is given below.

$$Aspect\ ratio = \frac{W}{H} \tag{2}$$

2) Orientation of the text candidate regions

We use the second moment to calculate the rotation of the text candidate regions. It is defined as follows.

$$\tan(2\theta_i) = 2\times \frac{\sum_{w=1}^{W}\sum_{h=1}^{H} whI_{ci}(w,h)}{\sum_{w=1}^{W}\sum_{h=1}^{H} w^2 I_{ci}(w,h) - \sum_{w=1}^{W}\sum_{h=1}^{H} h^2 I_{ci}(w,h)}\ , \tag{3}$$

where i is the index of text candidate regions.

3) Edge density variation of the text candidate regions

Because text regions have plenty of edges, the EDV (edge density variation) [11] can be calculated to discriminate real text regions from non-text regions. We divide the minimum bounding box into N equal-size blocks, where $N\in$ [4, 8, 10] can be determined by the aspect ratio. The edge density ed_j represents the number of edge pixels in each block for $j = 0, 1, ..., N$ .Then the EDV is defined as follows.

$$EDV = \frac{\sum_{j}^{N}\left|ed_j - ed_M\right|}{ed_M}\ , \tag{4}$$

where ed_M is the average of all the ed_j.

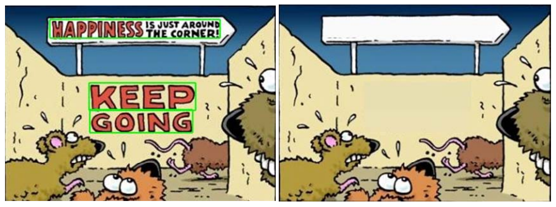
4) Coverage of the text candidate regions

Non-text regions may have similar aspect ratio to the real text regions, but they usually cover less area in the bounding box than text region does.

$$Coverage = \frac{TC_A}{W\times H}\ , \tag{5}$$

where TC_A is the total number of text pixels in the minimum bounding box.

For each text candidate region, a 4-dimension feature is extracted and then SVM is used for classification. Here we use the LIBSVM [12] and choose radial basis function (RBF) as our kernel function. We then merge adjacent minimum bounding boxes to compose text line in horizontal direction in order to reduce the coordinates. Fig. 5(a) shows the result of classified text regions with merging adjacent minimum bounding boxes after SVM. An OCR freeware, JOCR [13] is applied to recognize text elements in each text region. JOCR is adopted because it has high recognition rate for characters. Fig. 5(b) shows the comic image after erasing text elements. OCR recognized characters and their coordinates are recorded for embedding in the SVG file. Note that the art or signature fonts cannot be recognized by JOCR, like the authors signature as shown in Fig. 7(c) and Fig. 7(d). These text regions that cannot be recognized by OCR are encoded as vector formats.



(a) (b)

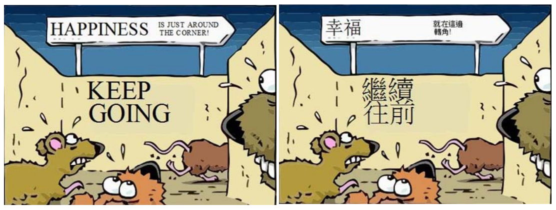
Figure 5. (a) The classified text regions. (b) After erasing text elements.

C. Embedding Characters In SVG Files

The OCR results and coordinate values are embedded in SVG files. In particular, the <text> element is supported for rendering text in SVG. The syntax can be represented as follows.

<text x = “c1” y = “c2” font-size = “c3”>Text</text> (6)

where c1 and c2 are the coordinate values of text position, and c3 is the value of text size to tweak the rendered text to fit the original text region. Fig. 6(a) shows the rendered comic image after decoding the SVG file embedded with <text> tags. These texts are translated to Traditional Chinese in Fig. 6(b). It can provide multilingual services for entertainment and education.

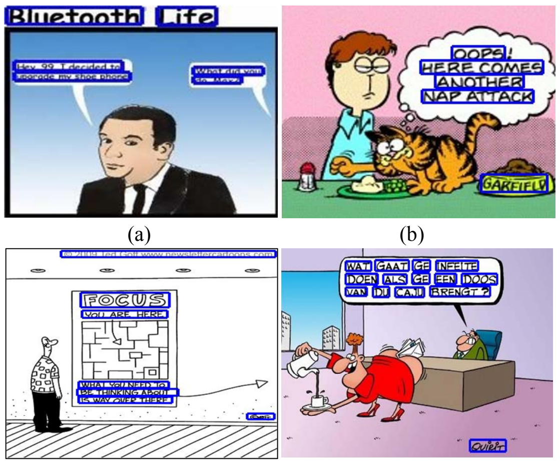


(a) (b)

Figure 6. (a) The OCR results are embedded in SVG file for rendering. (b) Translation to Traditional Chinese.

IV. Experimental Results

We take comic images as our testing data to analyze the compression ratio and image rendering time on portable handheld devices. 200 comic images containing texts or dialogs were collected from the Internet. In these comic images, we use 50 comic images for training and 150 comic images for testing. The parameters of SCW are set as X_1=1, Y_1=2, X_2=2, Y_2=4, T=1.05. Fig. 7 shows some results of text detection.


Bluetooth Life
OOPS! HERE COMES ANOTHER NAP ATTACK
GARFIELD
FOCUS
YOU ARE HERE
WAT GAAT GE INFEITE DOEN ALS GE EEN DOOS VAN DE CAJE BRENGT ?

(a) (b)

(c) (d)

Figure 7. Some experimental results of text detection.

Let *SVG* represents the SVG file without processing text and SVG_t represents the SVG file after processing text. | • | denotes the size of a file. The compression ratio between *SVG* and SVG_t is defined as below.

$$Compression\ ratio = |SVG_t| / |SVG| \times 100\% \qquad (7)$$

Fig. 8 shows the compression ratio between *SVG* and SVG_t for 15 comic images. It can observed that the largest compression ratio is 32.5% in #5 comic image with 214 characters, and the smaller compression ratio is 81.6% in #15 comic image with 37 characters. The average compression ratio for total 150 comic images is about 52%.

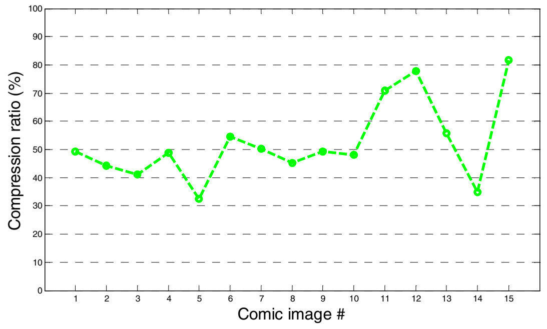


Figure 8. File compression ratio between *SVG* and SVG_t.

In image rendering time, we test 15 comic images above on HTC Magic mobile device which runs Android 1.5 OS on Qualcomm MSM7200A 528 MHz processor with 288 MB RAM. Fig. 9 shows that the comparison of rendering time. SVG_t only requires roughly half of *SVG* rendering time. It proves that our SVG_t is suitable to be applied on mobile devices (such as Ebook or smartphone) with limited CPU speed and memory.

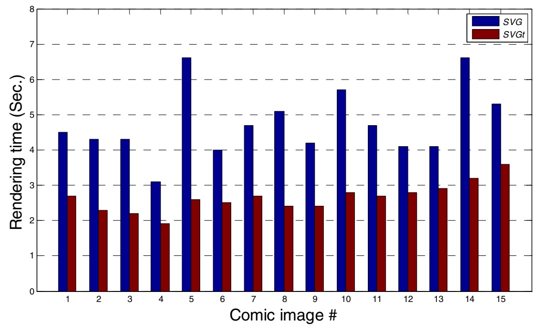


Figure 9. Rendering time comparison between *SVG* and SVG_t.

Since SVG is based on XML file, it is efficiency to search SVG images as explained below. There are two ways to achieve this purpose: text-based, and content-based. For text-based search, we can use text as our input query. As shown in Fig. 10, we input ‘cubicle’ for query and two results are retrieved. There are indeed cubicle scenes existing in these two comic images. For content-based, a vector image is generally composed of several Bezier curves which are connected consecutively to represent object shapes. We can use a stroke as a query, and then these strokes are approximated by Bezier curves. The similarity is calculated by matching features which are extracted from Bezier curves between SVG_t images. Fig. 11 shows the flowchart of the content-based SVG image retrieval. Fig. 12 shows an example that it draws a human head shape as a query, and two results are returned.

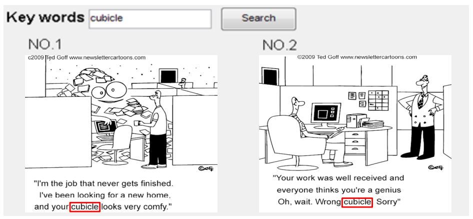

Key words
cubicle
Search
NO.1
c2009 Ted Goff www.newslettercartoons.com
"I'm the job that never gets finished. I've been looking for a new home, and your cubicle looks very comfy."
NO.2
©2009 Ted Goff www.newslettercartoons.com
"Your work was well received and everyone thinks you're a genius Oh, wait. Wrong cubicle. Sorry"

Figure 10. The retrieval results of the text-based SVG image retrieval.

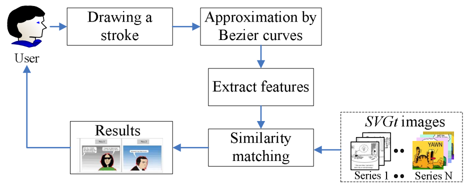


Figure 11. Flowchart of the content-based SVG image retrieval.

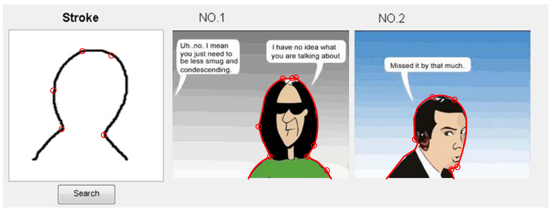

Stroke
NO.1
NO.2
Uh..no. I mean you just need to be less smug and condescending.
I have no idea what you are talking about
Missed it by that much.
Search

Figure 12. The retrieval results of the content-based SVG retrieval.

The flowchart of the story teller system is shown in Fig. 13. In this application, users can type any texts they like as a story. The system will parse keywords in the story and search the related comics in the database. It converts text information into visual image for telling story. For example, we type a short story such as **"After the <u>brainstorm</u>, they got a new <u>idea</u> and then started to implement a product. One month later, they participated the innovation competition and <u>introduced</u> their product to the reviewers. Finally, they got the first <u>award</u> because of their perfect demonstration".** The system will parse the story, extract the keywords and start to find corresponding text in SVG_t images. The results will be arrangement according to text ordering in the story, as shown in Fig. 14.

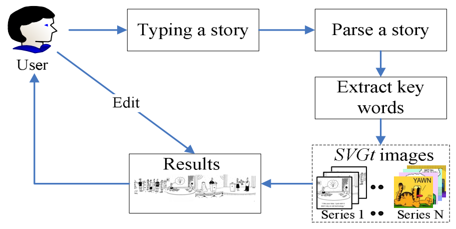
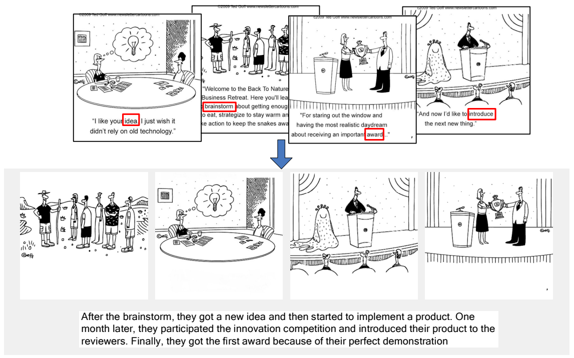


Figure 13. Flowchart of the story teller.


"I like your idea. I just wish it didn't rely on old technology."
"Welcome to the Back To Nature Business Retreat. Here you'll lea brainstorm about getting enoug o eat, strategize to stay warm an e action to keep the snakes awa
"For staring out the window and having the most realistic daydream about receiving an important award..."
"And now I'd like to introduce the next new thing."

After the brainstorm, they got a new idea and then started to implement a product. One month later, they participated the innovation competition and introduced their product to the reviewers. Finally, they got the first award because of their perfect demonstration

Figure 14. The results of the story teller.

V. Conclusion

In this paper, we convert comic raster images to SVG files and recognize/embed text elements in the SVG files. In this way, we can further compress the SVG files to about 52%. Using these text elements, we can translate comics into other languages to provide multilingual services easily. Text/content-based image search can be supported efficiently. A novel application system for story teller is also examined.

Acknowledgment

This paper is supported by NSC Taiwan under grant 98-2410-H-002-060-MY2.

References

[1] Ray-I Chang, Yachik Yen, Ting-Yu Hsu, "An XML-based Comic Image Compression," LECT NOTES COMPUT SC, Vol. 5353, pp. 563–572, 2008.

[2] Scalable Vector Graphics (SVG), http://www.w3.org/Graphics/SVG/.

[3] Stanford Vector Magic, http://vectormagic.com/home.

[4] AutoTrace, http://autotrace.sourceforge.net.

[5] Sebastiano Battiato, Giovanni Gallo, Giuseppe Messina, "SVG Rendering of Real Images Using Data Dependent Triangulation," In Proc. of ACM/SCCG, pp. 191–198, 2004.

[6] Song-Hai Zhang, Tao Chen, Yi-Fei Zhang, Shi-Min Hu, Martin, R.R., "Vectorizing Cartoon Animations," IEEE Transactions on Visualization and Computer Graphics, Vol. 15, pp. 618-629, 2009.

[7] Canedo-Rodriguez, A., Soohyung Kim, Kim, J.H., Blanco-Fernandez, Y., "English to Spanish Translation of Signboard Images from Mobile Phone Camera," In Proc. of IEEE SoutheastCon, pp. 356-361, 2009.

[8] Tomohiro Nakai, Koichi Kise, Masakazu Iwamura, "Real-Time Retrieval for Images of Documents in Various Languages using a Web Camera," International Conference on Document Analysis and Recognition, pp. 146-150, 2009.

[9] Hideaki Goto, Makoto Tanaka, "Text-Tracking Wearable Camera System for the Blind," International Conference on Document Analysis and Recognition, pp. 141-145, 2009.

[10] C.N.E. Anagnostopoulos, I.E. Anagnostopoulos, V. Loumos, E. Kayafas, "A License Plate Recognition Algorithm for Intelligent Transportation System Applications," IEEE Transactions on Intelligent Transportation Systems, Vol. 7, pp. 377-392, 2006.

[11] Wei-Yuan Chen, Shu-Yuan Chen, "Adaptive Page Segmentation for Color Technical Journals' Cover Images," Image and Vision Computing , vol. 16, pp. 855-877, 1998.

[12] LIBSVM, http://www.csie.ntu.edu.tw/~cjlin/libsvm/.

[13] JOCR, http://home.megapass.co.kr/~woosjung/Product_JOCR.html.