Stroke-based Semi-automatic Region of Interest Detection Algorithm for In-situ Painting Recognition

Youngkyoon Jang[1] and Woontack Woo[1]

[1] GIST U-VR Lab.
Gwangju 500-712, S. Korea
{yjang, wwoo}@gist.ac.kr

Abstract. In the case of illumination and view direction changes, the ability to accurately detect the Regions of Interest (ROI) is important for robust recognition. In this paper, we propose a stroke-based semi-automatic ROI detection algorithm using adaptive thresholding and a Hough-transform method for in-situ painting recognition. The proposed algorithm handles both simple and complicated texture painting cases by adaptively finding the threshold. It provides dominant edges by using the determined threshold, thereby enabling the Hough-transform method to succeed. Next, the proposed algorithm is easy to learn, as it only requires minimal participation from the user to draw a diagonal line from one end of the ROI to the other. Even though it requires a stroke to specify two vertex searching regions, it detects unspecified vertices by estimating probable vertex positions calculated by selecting appropriate lines comprising the predetected vertices. In this way, it accurately (1.16 error pixels) detects the painting region, even though a user sees the painting from the flank and gives inaccurate (4.53 error pixels) input points. Finally, the proposed algorithm provides for a fast processing time on mobile devices by adopting the Local Binary Pattern (LBP) method and normalizing the size of the detected ROI; the ROI image becomes smaller in terms of general code format for recognition, while preserving a high recognition accuracy (99.51%). As such, it is expected that this work can be used for a mobile gallery viewing system.[1]

Keywords: Semi-automatic ROI Detection, Hough-transform, Planar Object Recognition, Local Binary Pattern.

1 Introduction

Recent applications of mobile augmented reality (AR), such as ‘Layar’ [1] and ‘Wikitude’ [2], were based on points of interest (POI) focusing on ‘where’, when considering 5W1H. However, it does not provide specific information for an object instead of providing information of the area. Thus, recent research interests are now focusing on object-based recognition for mobile AR focusing ‘what’ inside of the ‘where’ as a specific type of context [3-5]. One recent application, ‘Google Goggles’, recognizes different kinds of objects including artwork and books [6]. However, it

[1] This research is supported by MCST and KOCCA, under the CT R&D Program 2011.

does not work properly in the case of altered illumination and severely skewed view directions for a less textured painting. A gallery, where many artworks are displayed, usually has similar challenging situations.

In order to recognize paintings inside of a gallery, Andreatta proposed appearance-based painting recognition for a mobile museum guide [7]. It uses a predefined center region of the captured image for the recognition. The method normalizes the image with the description grid of the region in order to overcome illumination changes of the environment. After the normalization, it detects overexposed areas (specular reflection) and uses the region with no noise. However, it does not support robust recognition when there are view direction changes, because it does not provide an ROI detection method. To overcome the challenge of skewed views and direction changes, the latest mobile guide was based on fast SIFT (Scale-Invariant Feature Transform) proposed by Ruf [8]. This approach utilizes currently available object recognition and network technologies. It captures an image from a mobile phone and sends it to the server. Then, the main procedure - feature extraction and matching - is done on server side. The feature-based approach generally provides accurate detection and recognition. However, it needs a great deal of processing time if it needs to recognize a painting's data due to severely a skewed view direction changes because it requires more training and comparison processes. Furthermore, the feature-based recognition does not provide accurate recognition results in terms of a less textured painting.

In order to overcome all the discussed problems and environment challenges of painting recognition, we propose stroke-based semi-automatic ROI detection algorithm for in-situ painting recognition. The proposed algorithm needs minimal participation by the user for specifying the initial search area of the painting's ROI by dragging a line from one corner of the painting to the other corner. Then, the proposed algorithm detects dominant edges by adaptively finding and using the threshold. Because the proposed algorithm adaptively finds the threshold, it handles both simple and complicated textured paintings and also detects dominant candidate lines of the ROI with Hough-transform method. Next, it estimates and detects unspecified vertices by selecting appropriate lines comprising the predetected vertices, even though it requires a stroke to specify two vertex searching regions (from one end of the ROI and the other diagonally) to begin. By taking the proposed semi-automatic (but simple) algorithm, it accurately detects the painting region, even if there is a severe view direction change and a user gives inaccurate input points. Moreover, the proposed algorithm normalizes the size of the detected ROI as smaller as the pre-determined size in order to make a general code format for recognition. Thus, the proposed algorithm provides robust recognition which is not affected by the various sizes of the paintings. Adopting the LBP [9] method has many advantages such as robust recognition to global illumination changes, fast processing time with binary code matching, and minimal storage size for saving codes. Experimental results of the proposed algorithm show high recognition accuracy (99.51%) with an accurate detection results (1.16 error pixels) for both view direction and illumination changes.

The rest of this paper is organized as follows. Section 2 introduces the general procedure of the proposed semi-automatic ROI detection algorithm, and a detailed description of the algorithm is presented in Section 3. Section 4 describes the

implementation and experimental results of the proposed algorithm, and conclusions and future works are discussed in Section 5.

2 Overview of the Proposed Algorithm

An overview of the proposed algorithm can be seen in Figure 1. In order to initiate the algorithm, a user is required to capture a painting image from a mobile phone and drag a line specifying two initial search regions (red rectangular boxes in Figure 2(a)) on the touch screen of the phone like the yellow line of Figure 2(a). Then, the proposed algorithm determines the adaptive threshold corresponding to the captured painting and detects edges with the determined threshold in the specified regions. Next, it detects two vertices in both specified regions using the Hough transform. In this step, the algorithm gives line parameters in order to comprise the detected vertices. By taking one of two lines comprising one vertex, it finds the other line, which crosses inside an image. Then, it detects the ROI by estimating unspecified vertices (blue circle points in Figure 2(a)) and detecting accurate vertices (black blank circle points in Figure 2(a)) again inside of the estimated vertex regions.

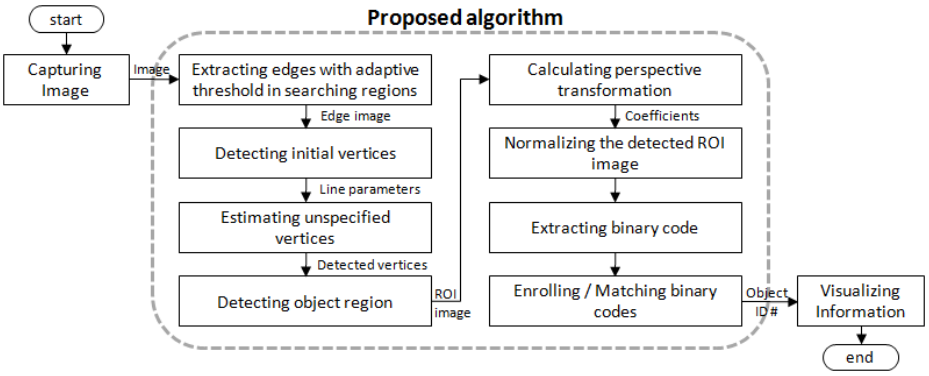


Fig. 1. Proposed algorithm flow

The detected ROI image is normalized by calculating perspective transformations, which affects the resultant general code size irrespective of the real size of the painting as shown in Figure 2(b). Figure 2(b) shows examples of the procedure for the binary code extraction and matching using the LBP method. By using the normalized ROI image, the proposed algorithm then extracts binary code via the LBP method [9], as shown in Figure 6. Next, the proposed algorithm enrolls or matches the extracted LBP code and preloaded LBP codes by calculating the Hamming Distance (HD) value. When the measured HD value is lower than the experimentally determined threshold (0.42), indicating a high level of similarity, the algorithm determines that the detected ROI image is a real painting, and vice versa. This proposed algorithm is suitable for mobile application and it provides fast and robust recognition in a

challenging environment irrespective of view direction and illumination changes, even though the algorithm requires a user's minimal participation.

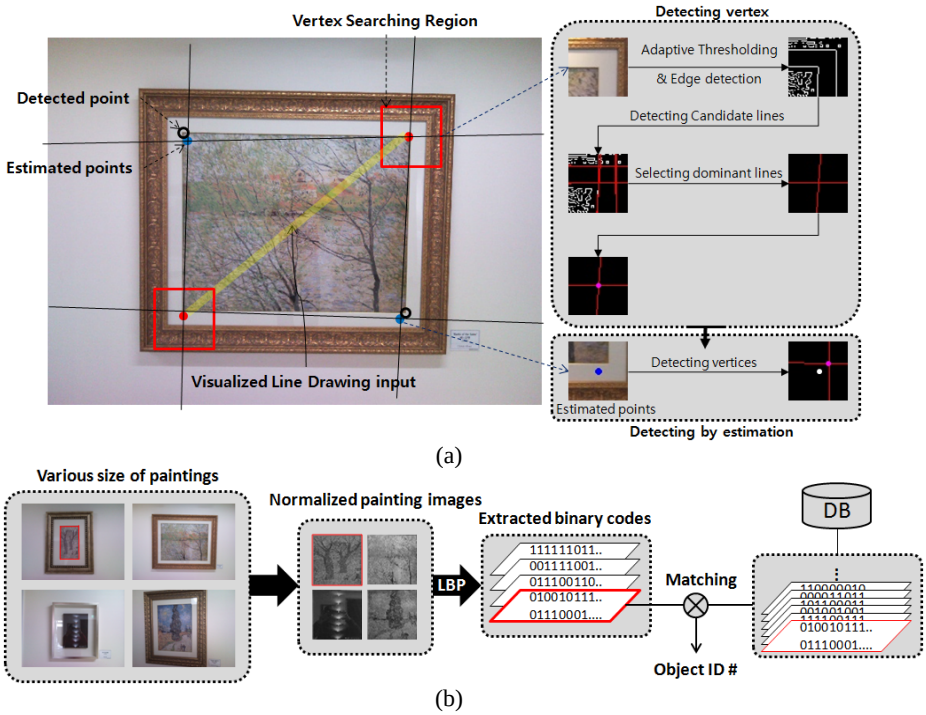


Fig. 2. Examples of the procedure for the proposed algorithm: (a) a stroke-based ROI detection (b) code extraction and matching

3 Proposed ROI Detection Algorithm for Robust Recognition

Because of the different texture amounts among paintings, the proposed algorithm needs to be able to extract dominant edges irrespective of the strength of the edges. Moreover, because the algorithm requires a user's participation (while preserving high detection and recognition accuracies despite view direction changes), it needs to be simple and to reduce the interaction process. To this end, we propose a stroke-based semi-automatic ROI detection algorithm using adaptively extracted edges and a vertex estimation process. Detailed explanations are provided as follows.

3.1 A Stroke-based Semi-automatic ROI Detection with Vertex Estimation

To reduce the noise of the image in paintings with a large amount of texture, the general approach applies the Gaussian kernel to the captured image and extracts

dominant edges. However, the approach does not extract weak edges such as blurred images or indistinct borders of the image, because it removes the weak edges from the borders by only using a predetermined threshold.

In order to overcome the problem, we first apply a global adaptive thresholding method only for the Intensity element of the captured image after converting the pixel format from RGB to HSV. Through the thresholding, we can extract dominant edges for the borders by using Canny edge detector as shown in Figure 3, because the adaptive thresholding method separates two regions. Even though the two regions have similar intensities (the paintings edges are not clearly defined) it automatically finds an appropriate threshold value. Therefore, it properly works for both simple and complicated textured paintings.

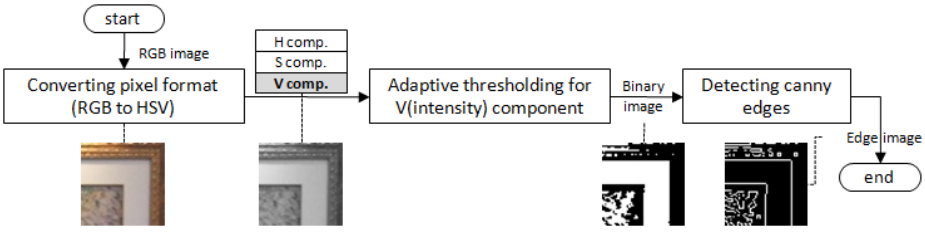


Fig. 3. Procedure of dominant edge detection

By using the detected dominant edges as an input, the Hough-transform detects dominant candidate lines by searching a limited range of parameters. Then, we select the two lines comprising one vertex. We do by examining the angle between two lines and the distance between the initially specified point and the cross point of the two lines. If both conditions are satisfied, as Equation (1), we detect the vertex as a result, as shown in Figure 4.

$$F(\rho_b,\theta_b,\rho_c,\theta_c)=\begin{cases} True, & MIN(D_d = Dist(P_s,P_d = Pt(\rho_b,\theta_b,\rho_c,\theta_c)),D_t)\,\&\,T_a < A = Ang(\rho_b,\theta_b,\rho_c,\theta_c) \\ False, & otherwise \end{cases} \quad \textbf{(1)}$$

, where (ρ_b, θ_b) are the parameters representing a base line, and (ρ_c, θ_c) are for a compared candidate line. We calculate intersection position P_d by using a *Pt*() function with the four parameters. Then we check whether the distance D_d, between the detected point P_d and the specified point P_s is minimal compared with D_t. Shorter distance value is saved as D_t. Then, we also check that the angle, calculated by the function *Ang*() with the parameters representing two lines, is larger than the pre-determined threshold T_a. If the two conditions are satisfied, function *F*() returns true, meaning it was detected, and vice versa.

Like the procedure mentioned above, the algorithm detects two initial vertices inside of the user-specified searching regions. On the other hand, the other two vertices of the painting are estimated by using the two rho and theta values of the lines comprising the two detected vertices. For detecting the initial two vertices, we know the parameter values, rho and theta, of the detected lines through Hough

transform. By taking one of those two lines comprising the detected one of those vertices, we can find the other line which crosses inside of an image, as shown in Figure 5 ①. Therefore, we detect object ROI by estimating the two other intersection points (pink dot in Figure 5 ②) as another two inputs for detecting vertices (white dot in Figure 5 ②). By comprising all the detected four vertices, we accurately detect ROI of a painting.

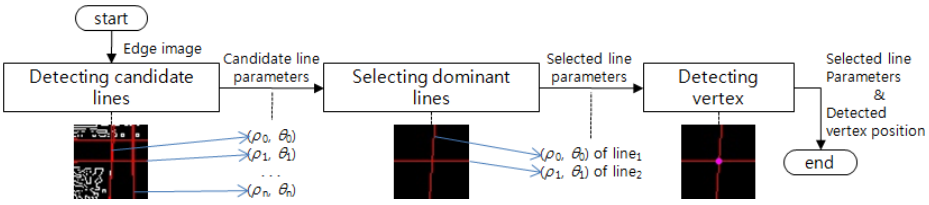


Fig. 4. Procedure of initial vertex detection

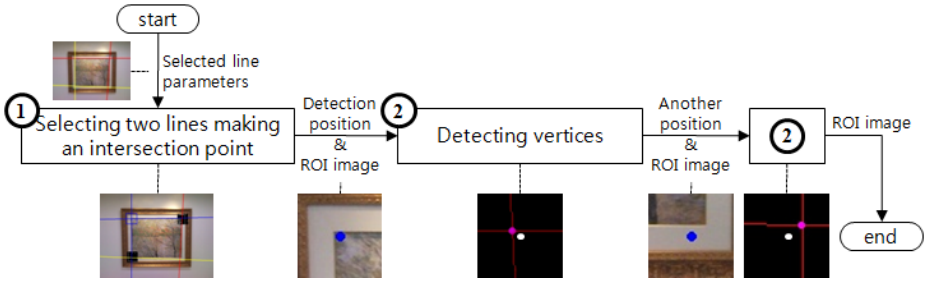


Fig. 5. Procedure of estimation (pink circle points) for detecting vertices at the unspecified position and its detection results (white circle points)

3.2 In-Situ Painting Recognition based on Local Binary Pattern

To recognize both simple and complicated textured paintings in a challenging environment where the global illumination changes, it is important to extract illumination-invariant codes for the painting. For user's satisfaction the algorithm requires fast processing time even on mobile phones; however, the previous feature-based recognition [8] takes too much processing time on mobile phones. Furthermore, with less textured paintings, the previous method decreases the discrimination ability among the several less textured paintings because it does not extract enough numbers of the feature-points for the painting.

In order to overcome the problem, we simplified the LBP method [9] which considers the pattern of image intensity in the localized area. The original LBP method [9] considers rotation-invariant features. However, because the proposed algorithm has already detected the accurate ROI of the painting, the proposed algorithm only extracts the binary codes inside of down-sampled image (15×15) of the detected ROI. This is done by only adopting the idea of LBP code extraction. The LBP can be defined as an ordered set of binary values determined by comparing the gray values of a center pixel and the eight neighborhood pixels around the center, as

shown in Fig. 6. By using this simplified LBP method, we extract 225 bytes (8 * 15 * 15) of LBP codes representing a painting. Even though a user takes a painting image from the flank, the proposed algorithm corrects perspective distortion by applying the perspective transform. Then, it provides same size of the painting code and a simple matching algorithm with an Exclusive OR (XOR) binary operator as shown in Equation (2).

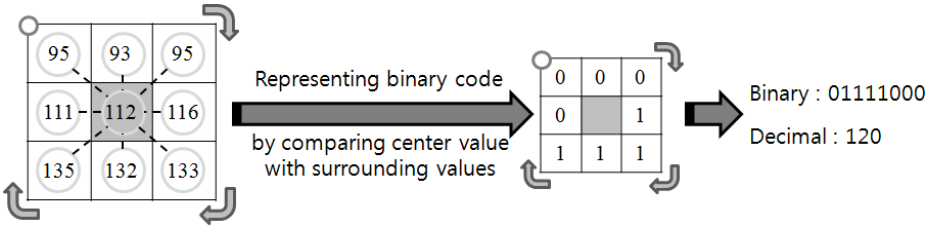


Fig. 6. LBP codes extraction

The currently extracted codes can be practically matched with the filtered set (number of the filtered image: 50) of codes among the code DB, because we assume that we know the direction and location of the user using sensors within the mobile phone. For matching, we use the Hamming distance measuring the dissimilarity between the current extracted code and the filtered set of codes, as represented in Equation (2). A smaller HD (Hamming distance) closer to 0 means the two codes are similar, and vice versa. We determined that the two codes are identical when the HD is less than the pre-determined threshold T_h. According to experimental results, we determined that T_h is 0.42, which shows the high recognition accuracy result.

$$HD = \frac{Num(codeA \otimes codeB)}{CodeBits} \quad (2)$$

, where CodeBits means the number of LBP code bits. In this paper, we extract 1800 (8 * 15 * 15) bits for LBP codes. $\otimes$ represents XOR operator. *Num*() function returns the number of 1's of the codes after the XOR operation. As it only uses the binary pattern for code matching, the proposed system using LBP-based recognition shows a fast processing time. Furthermore, the system could recognize less textured objects even in the case of illumination changes, because the pattern is extracted inside of the accurately detected ROI and it is globally illumination-invariant. Therefore, the proposed algorithm recognizes both less textured and more textured paintings in a challenging environment such as gallery even though the image is captured from the flank.

4 Implementation and Experimental Results

4.1 Implementation

In this work, we implemented a mobile painting recognition system prototype using the proposed a stroke-based ROI detection and LBP-based painting recognition algorithm. We used a DirectShow for capturing images from a camera of a mobile phone and an OpenCV library [10] for implementing basic image processing algorithms. We implemented our proposed algorithm on a mobile phone based on the Windows Mobile 6.3 platform. The phone had a 1 GHz ARMv7 processor and built-in (4.1 inch) touch-screen display.

We implemented the ROI detection algorithm as shown in Figure 7(b). The stroke interaction process is needed to specify the search areas as shown in Figure 7(a). The black line of Figure 7(b) shows an ROI comprised by the initially specified and estimated points. And the green line shows the detected ROI by using the proposed algorithm.

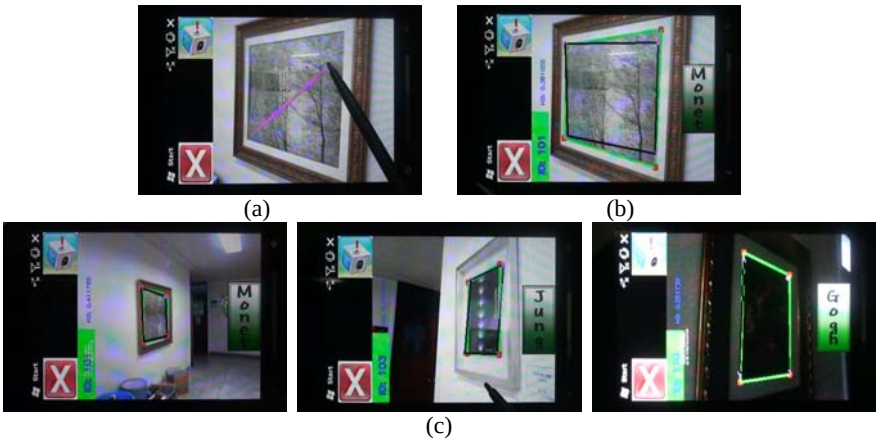


Fig. 7. Examples of the proposed algorithm's output: (a) user's dragging as an initial input; (b) ROI detection (black line: ROI consist of the initially specified and estimated points, green line: ROI comprised by the detected vertices); (c) results in the case of challenging environment (view distance, direction and illumination changes)

After the ROI is detected, it is normalized as shown in Figure 2(b). Then, the proposed algorithm extracts and matches the binary code. The recognized ID number corresponding to the painting and the artist's last name is on the left and right side of the screen.

As shown in Figure 7(b), the proposed algorithm detects an accurate ROI even though there was an inaccurate input. Moreover, the proposed algorithm is robust to the challenging environment (view distance, direction and illumination changes). Figure 7(c) presents the accurate detection and recognition results when the painting is captured from far distance (left), from the flank (middle) and in a dark environment

(right). As such, it is expected that this work can be used for a mobile gallery viewing system.

4.2 Experimental Results

In this paper, we evaluated the performance of our proposed algorithm by using a database of captured gallery painting images. The database is comprised of 250 images acquired using a mobile phone images from 5 different angle under uncontrolled illumination conditions for 50 paintings. Each image in the database has a 24-bit color value and a pixel size of 320×240. This DB set is for experiments. In practice we only store LBP binary codes as a DB set for recognition.

First, we measured the processing time for the proposed algorithm. As a result, our proposed algorithm required 38.97 ms to run in our experimental environment. Specifically, ROI detection required 37.81 ms, the majority of the processing time, while the recognition required 1.16 ms including code extraction and matching, as shown in Table 1. Even without optimization of the source code, this speed is fast enough to get results in real-time on a mobile phone. With optimization, we could therefore expect even faster results during ROI detection.

Table 1. Processing time of the proposed algorithm.

Algorithm	Processing time (ms)		
	Minimum	Maximum	Average
ROI detection	27.15	45.83	**37.81**
LBP code recognition	0.83	1.37	**1.16**
Entire processing time	27.98	47.20	**38.97**

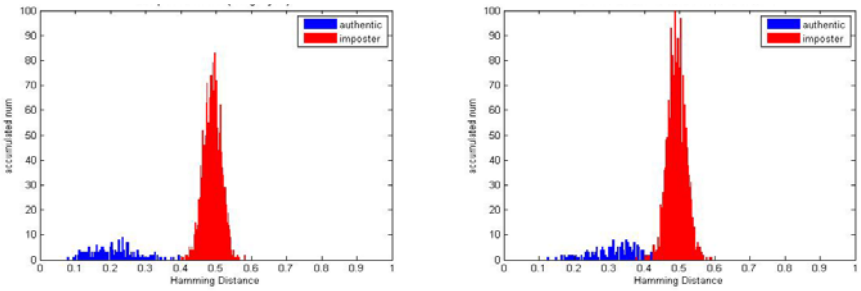
In the second set of tests, we measured the detection accuracies of the corners of the painting ROI, shown in Table 2. The accuracies were measured based on the Root Mean Square (RMS) pixel error between the detected points, and those chosen manually. As can be seen in Table 2, the average RMS pixel error indicates that the proposed algorithm detects accurate (1.16 error pixels) vertices comprising the ROI, even when users give an inaccurate input (4.53 error pixels, on average).

Table 2. Detection accuracies of the corners of painting ROI

Detection method	RMS pixel errors			
	Minimum	Maximum	Average	Standard. deviation
Manual selection	1.00	7.07	4.53	1.68
Proposed algorithm	**0.00**	**3.53**	**1.16**	**0.78**

Finally, we measured the recognition accuracy. For that we measured the Equal Error Rate (EER), which represents the minimum error rate for recognition by taking an exact threshold of the given test set. As experimental results, the proposed detection algorithm supports accurate recognition (EER: 0.48%), whereas the EER was 3.02% without the algorithm (manual selection), as shown in Figure 8.

Thus, we could confirm that the proposed algorithm guarantees robust painting recognition, even though a user may specify inaccurate inputs and in cases in which there are view direction angle and illumination changes.



(a) (b)

Fig. 8. Experimental results for recognition: (a) with the proposed detection algorithm, and (b) without the proposed detection algorithm

5 Conclusions and Future Works

This paper proposed a stroke-based ROI detection algorithm for in-situ painting recognition. We plan to develop an automatic detection algorithm for convenient recognition, while maintaining high recognition rates.

References

1. Layar, http://www.layar.com/
2. Wikitude, http://www.wikitude.org/
3. Lee, Y., Oh, S., Shin, C., Woo, W.: Ubiquitous Virtual Reality and Its Key Dimension. International Workshop on Ubiqiutous Virtual Reality 2009, pp. 5-8 (2009)
4. Shin, C., Kim, H., Kang, C., Jang, Y., Choi, A., Woo, W.: Unified Context-aware Augmented Application Framework for Supporting User-Driven Mobile Tour Guide. 8th International Symposium on Ubiquitous Virtual Reality 2010, pp. 52-55 (2010)
5. Kim, H., Woo, W.: Real and Virtual Worlds Linkage through Cloud-Mobile Convergence. Workshop on Cloud-Mobile Convergence for Virtual Reality, pp. 10-13 (2010)
6. Google Goggles, http://www.google.com/mobile/goggles/
7. Andreatta, C., Leonardi, F.: Appearance Based Paintings Recognition For a Mobile Museum Guide. International Conference on Computer Vision Theory and Applications, VISAPP 2006 (2006)
8. Ruf, B., Kokiopoulou, E., Detyniecki, M.: Mobile Museum Guide Based on Fast SIFT Recognition. 6th International Workshop on Adaptive Multimedia Retrieval (2008)
9. Ojala, T., Pietikainen, M., Maenpaa, T.: Multiresolution Gray Scale and Rotation Invariant Texture Analysis with Local Binary Patterns. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), vol. 24, no.7, pp. 971-987 (2002)
10. Intel Open Source Computer Vision Library, http://sourceforge.net/projects/opencvlibrary/