Jurnal
Teknologi

Full Paper

ROBUST HAND-DRAWN SQUARE-ROI CONTOUR DETECTOR BASED ON ADAPTIVE THRESHOLDING

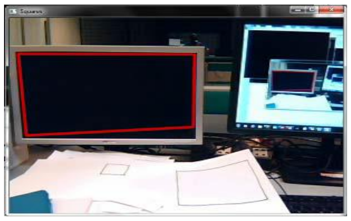
Rechard Lee, Abdullah Bade*, Salina Sulaiman, Siti Hasnah Tanalol

Real Time Graphics and Visualization Research Group (GRAVS), Mathematics with Computer Graphics, School of Science and Technology, Universiti Malaysia Sabah Jalan UMS, 88400 Kota Kinabalu, Sabah, Malaysia

Article history
Received
3 December 2013
Received in revised form
2 July 2014
Accepted
25 November 2014

*Corresponding author
abb@ums.edu.my

Graphical abstract



Abstract

Hand-drawn square-ROI detector was developed as one of the vital components in Real-Time Pre-Placed Markerless Square-ROI (RPMS) recognition technique. It aims to; 1. To verify hand-drawn Square-ROI (Region of Interest) as a square, and 2. To create a robust and flexible square-ROI detector technique which can be applied in uneven lighting condition. In this paper, we aim to detect only the desired ROI and handle the uneven lighting condition which is one of the primary disturbance sources that may generate false results. This may lead to error in registration in Augmented Reality application due to inability to correctly define a marker. As a solution, our technique applies adaptive thresholding in order to address this issue and to create a robust and flexible technique. To verify our proposed technique, two kinds of square is used in the testing and evaluation phase. In this experiment, two influencing factors; viewing distance, and detection accuracy were used to validate our aim. The results of the experiments show that the proposed technique efficiently detects and defines the desired square-ROI and also robust to illumination changes.

Keywords: Augmented reality, contour, feature, ROI, thresholding;

1.0 INTRODUCTION

Milgram [1] defined Augmented Reality (AR) as one part of mixed reality in the continuum of real-to-virtual environments (see Figure 1). The real environment is at one end of the spectrum and the virtual world is at the other.

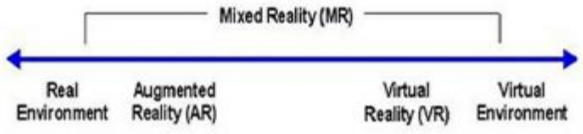


Figure 1 Milgram's Reality-Virtuality Continuum [1]

Whereas, Azuma (1997) [2] defines an AR as systems that have the following characteristics:

- Combines real object with virtual objects,
- Interactive in real-time, and
- Registered in three dimensions.

The ultimate goal of an AR system is to create a mixed digital environment such that the computer–generated objects mixed into the real-world environment and viewed as one of its entities. Thus, the object must be visually registered in every point the user sees. To maintain the user's illusion that the virtual objects are part of the real world requires a consistent registration of the virtual world with the real world, and this stringent requirement of the system is one of the challenges in developing an AR system [2], [3], [4].

As shown in Figure 2, to blend virtual content with the existing reality, a typical AR system need to utilize an element or combined elements in virtual content (i.e. global position, orientation, feature points, and metadata) as the key to perform the registration process so that the enhanced reality called AR could be achieved. Ref. [5] state that, inaccuracies are caused by; incorrect registration, changes in patient's anatomy, improper calibration, and user's stereoscopic perception. By definition, registration referred as a way to properly align the real and virtual object in the mixed reality environment.

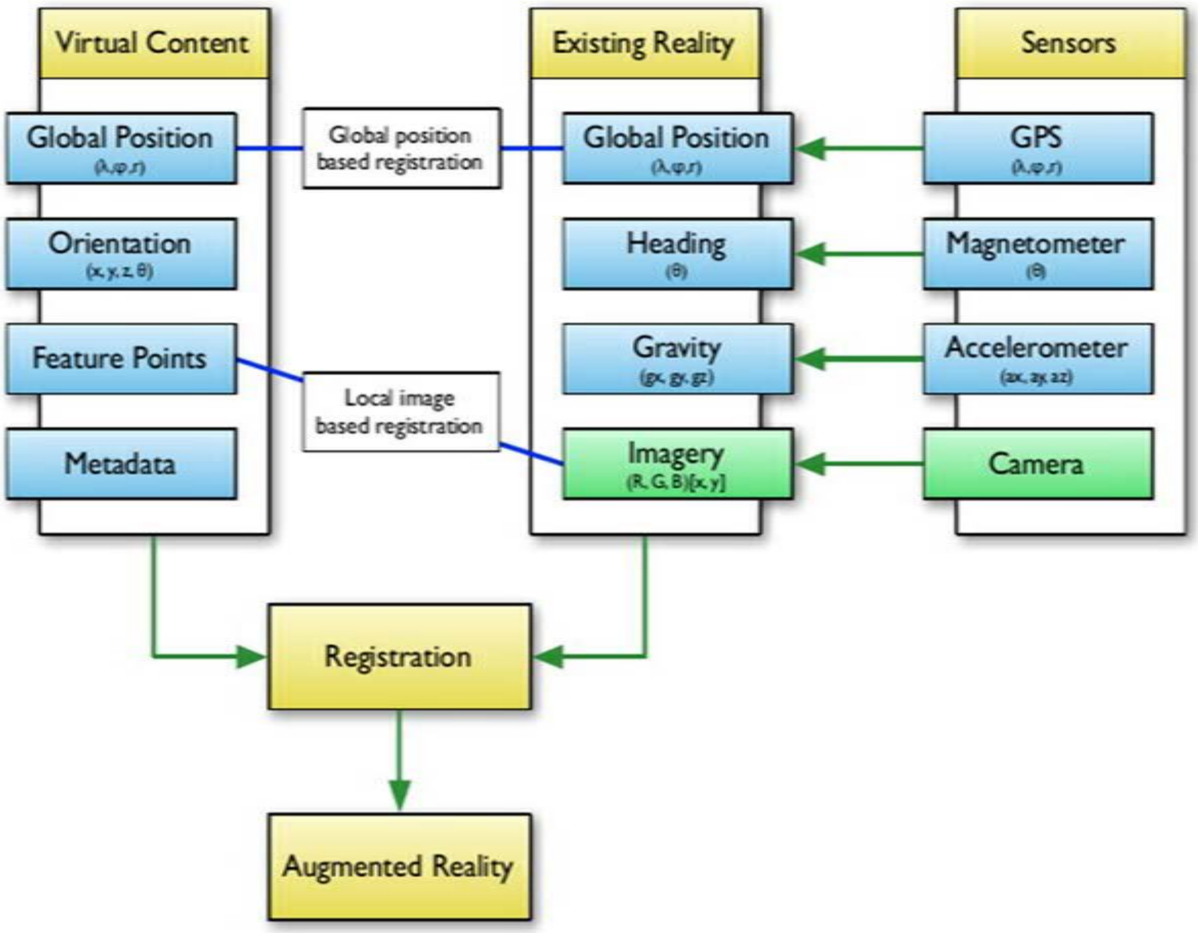


Figure 2 An overview of a typical augmented reality system [6]

Registering an object consists of two stages [7]:

1. Tracking

Make use of feature detection, edge detection or other image processing method to interpret the video feed by detecting fiducial markers or interest point (corner).

2. Reconstructing

Uses the data obtained from the first stage to reconstruct a computer generated object in real world coordinate system. Our proposed technique developed to deal with the first stage by using Canny's operator [8] enhanced with smoothing and adaptive technique in order to correctly detect a hand drawn square-ROI under uneven lighting condition.

2.0 APPROACH TO SQUARE-ROI DETECTION

In this section, we will discuss the approaches and phases involved in the process of identifying a hand-drawn square- ROI in real-time. A square naturally has four equal sides (edges) and four right angles (900) forming a square corner and vertices. Thus to verify the detected ROI as a square, edge and contour detection is needed to ensure the success of our technique. Another reason is that, by performing edge detection, it significantly reduces the amount of data in an image and make it easier and faster to analyse.

2.1 Grey-scale Conversion

The reason for taking up this step is to reduce the amount of data which are very costly to handle. Another reason, it is much easier to operate on greyscale images [9]. A colour image with three channels (red, green, and blue) is returned with eight bits per channel, when we calculate the bits of information, the amount of data per frame equal to 640 x 480 x 3 x 8 = 7, 372, 800, and most typical camera with a width of 640 and 480 in height return between 20 to 30 frames per second. By doing so, the amount of processing required in the subsequent stages is reduced.

The greyscale transformation processes consist of taking the image brightness and change it into new image brightness in real time for displaying stored with 8 bits per sample pixel. The grey value of every pixel position is calculated by adding together 30% of the red value, 59% of the green and 11% of the blue value at the pixel position. The equation is given by: (1)

$$I(x,y) = 0.3 \times R(x,y) + 0.59 \times G(x,y) + 0.11 \times B(x,y) \qquad (1)$$

From the equation, the processor needs to perform three floating point multiplications and two floating point addition per pixel for a total of 1, 536, 000 floating point operations [10], [11], [12].

2.2 Smoothing

Smoothing is a set of local pre-processing method. It aims to suppress image noise and resolution for enhancement details. The new image is based on the averaging of the brightness values of the pixel using a small neighbourhood of the pixel in the grey-scale format. It is done by applying blurring. Without this step, the processed images become pixilated and might pose the problem of blurring or unnatural edges and corners which is a problem for the next step, namely feature detection [10], [12].

2.3 Adaptive Threshold

Another technique which is worth to discuss before we move to feature detection is called thresholding. It is computationally inexpensive, fast, and needed in finding contours.

Thresholding is the simplest segmentation process used to transform grey-scale image f into a binary image g called image binarization as follows:

$$\begin{aligned} g(i,j) &= 1 \; for \; (i,j) \geq T, \\ &= 0 \; for \; (i,j) < T \end{aligned} \qquad (2)$$

Where T is the threshold, $g(i,j) = 1$ for image elements of objects, and $g(i,j) = 0$ for image elements of the background.

The goal is to simplify the information into a meaningful and easy to analyse by determining a value called threshold and comparing each pixel with this value. If the value of the pixel intensity is greater than the threshold, its value becomes 1 (white (255) - object), otherwise it becomes 0 (black - background). There are global and local thresholding due to grey-level variation in object and background. This variation caused by uneven and changing lighting condition and nonuniform input device parameters.

A global threshold is determined from the whole image of

$$f: \quad T = T(f) \qquad (3)$$

It is the easiest and fastest method but greatly depends on lighting conditions and soft intensity changes [9].

Whereas, local threshold is

$$T = T(f, fc) \qquad (4)$$

Where fc is that image in which the threshold is determined [11], [12], [13], [14]. Conceptually, this approach will divide the input image into a set of non-overlapping sub images, and then processing each image separately with a suitable threshold independently [15]. This technique is more preferable when there are strong illumination or reflectance gradients.

2.4 Feature Detection

The concept of feature detection methods is considered to be an essential component and use as the initial step in developing AR techniques and application [16]. It is used to perform a low level feature extraction to find areas of interest from the input [9]. Feature detection algorithms are used in motion detection, image matching, and object recognition, for example. In this thesis, feature detection was considered as a means for detecting corner and contour in order to identify and verified the existence of marker in real-time. Feature point also called an interest point is a small area in an image. In general, a good feature is invariant to changes and can be robustly detected at different time in several frames.

Examples of those methods are; edge detection, corner detection, curvature detection, blobs or region-based detection, and optical flow. Our interest is on edge and contour detection.

In general, the reason to perform contour detection is to significantly reduce the amount of data in an image by detecting edges in a robust manner. Result of edge detection define the boundaries between detected regions in an image [17]. In his paper [17], the selection of an edge detection based on two factors i.e. edge orientation, and noise environment. Our technique is based on

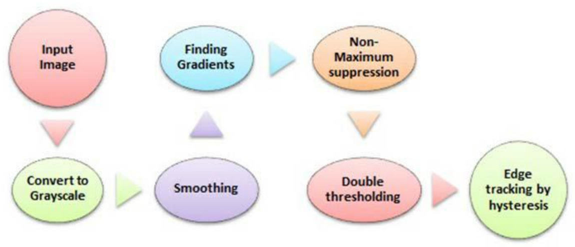


Figure 3 Conventional Canny's Edge detection stages

An algorithm developed by Canny [8]. As depicted in Figure 3, there are six stages in this algorithm.

The ability to detect true weak edges and not susceptible to noise interference [18] was the reason why we opted for Canny's operator. The issue with Canny's algorithm is that it depends heavily on the standard deviation Gaussian filter and the threshold values. Because of these, the computation time is higher. However in [17], compared to Sobel, Prewitt and Robert's operator, the Canny's operator performs better under almost all environments.

According to Canny [8], the followings:-
• Good detection
• Good localization, and
• Only one response to a single edge have been set as a performance criteria in detecting an edge and we will consider those criteria as a testing parameter.

2.5 Proposed Square-ROI Detection Pipeline

To improve the computation time and to add the ability to detect a square, our proposed technique combines Canny's operator with contour detection, image filtering and thresholding. Since our approach using a square figure, it is best to find closed contours and approximate the detected square with polygons of four vertices. The proposed pipeline is depicted in Figure 4. It begins with pre-image processing, where the given input is converted into a grey-scale frame, smoothed and ends with the identification of a square.

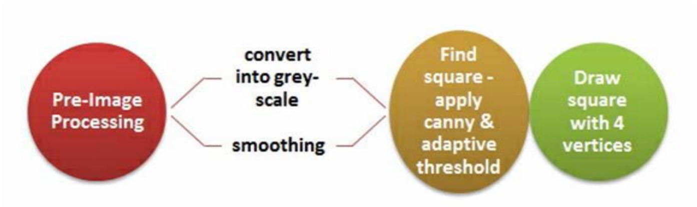

Pre-Image Processing
convert into grey-scale
smoothing
Find square - apply canny & adaptive threshold
Draw square with 4 vertices

Figure 4 A Pipeline for Proposed system

3.0 EXPERIMENTAL SETUP AND RESULTS

We present the results of our proposed method based on the test conducted on two (2) square-ROI (see Figure 5) given as an input in our experiment.

3.1 Experimental Setup

In our approach, we will define a Region of Interest (ROI), called square-ROI, and this square-ROI need to be manually hand-drawn by the user. The rationale to manually handdrawn is to avoid the needs to prepare a printed marker and at the same time to make our proposed technique more flexible with unprepared environment. Thus, we can ensure only desired area is searched for features. Why we use a square? Square naturally produces four (4) possible point and these four (4) points are needed to calculate the pose camera estimation for visualizing objects rendering.

We formulate that number of vertices detected on a contour (square) must be equal to four (4) vertices in order to identify a square and upon successfully identified, a square will be drawn on top of the target area (see Figure 8).

The hardware used to set up the experiment is listed below:

1. 1 x Logitech web camera
2. 1 x A4 paper
3. 1 x hand-drawn Square-ROI (17 cm x 16 cm)
4. 1 x hand-drawn Square-ROI (6 cm x 5 cm)
5. 1 x Dell Precision T1650

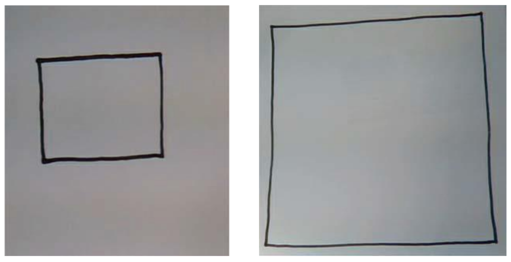

(a) (b)

Figure 5 Hand-drawn Square-ROI (a) 6 cm x 5 cm (b)17 x 16cm. Both with 2 mm line thickness

The software used in the experiment is listed below:
1. Microsoft Visual studio Ultimate 2012 (C++)
2. OpenCV 2.4.2
3. Microsoft Windows 7 Professional

The visual sensor initially positioned at 60 cm from the input (see Figure 6). This will be the initial viewing distance in our experiment setup.

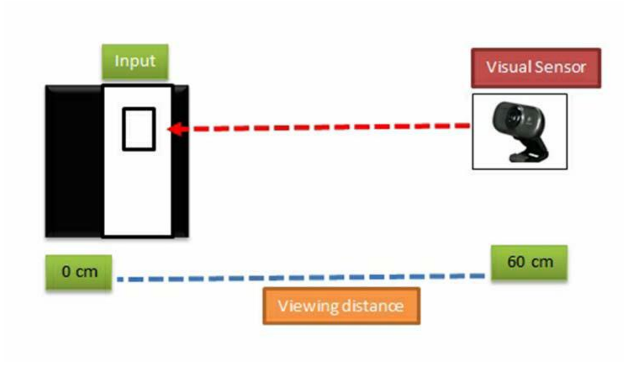


Figure 6 Camera viewing distance

Two test cases will be conducted to show that the proposed technique is able to detect square-ROI contour. Table 1 shows the test cases conducted during the testing process.

Table 1 Test cases

Test case	Description	Input	Expected Result
1	Detect square-ROI contour with {Canny's operator + smooting + threshold}	Hand-drawn Square-ROI	No false square Detected Square-ROI detected
2	Detect square-ROI contour with {Canny's operator + Smoothing + Adaptive thresholding}	Hand-drawn Square-ROI	No false square Detected Square-ROI detected

Both test cases are performed to test whether the developed technique able to correctly and accurately detect only the desired square-ROI with no false square detected. In test case 1, global thresholding is used whereas local or adaptive thresholding is used in test case 2.

3.2 Result

From the experiments, smoothing manage to remove noise and enhance the details needed to discard inconsistency.

Figure 7 (below) showed that, by using global or simple thresholding the technique not able to detect the desired square. Only false square is detected. We can see that, the monitor screen and the window frame have been falsely identified as a square and indicated by the red colour.

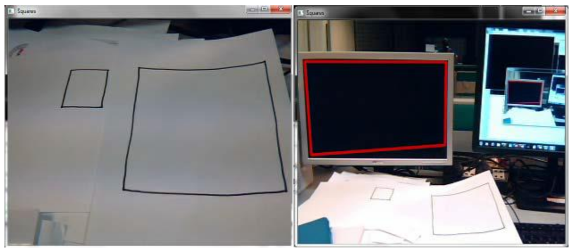


Figure 7 Result from Test Case ID 1

The result for test case ID 2 is depicted in Figure 8.

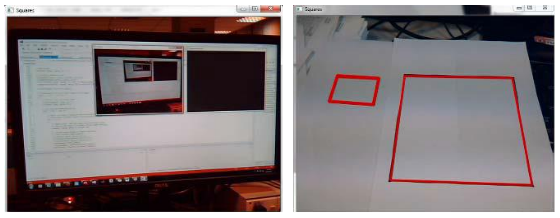


Figure 8 Result from Test Case ID 2

This result can be affected and changed if the square-ROI is positioned outside the viewing distance. We have calculated the square-ROI detection percentage rate for accuracy and consistency based on 299 cycles. Here is the result:-

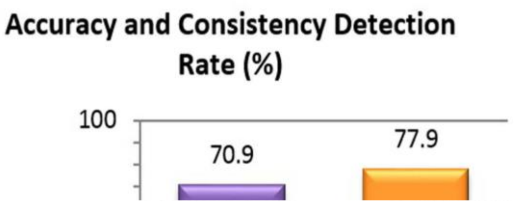


Figure 9 Square-ROI accuracy and consistency detection rate

Figure 9 showed that, in both sizes of hand-drawn square-ROI used i.e. 17 cm x 16 cm and 6 cm x 5 cm, the proposed technique able to detect with 70.9% and 77.9% accuracy and consistency respectively. We can say that both aims mentioned have been reached with promising potential.

4.0 SUMMARY AND DISCUSSION

Based on our experiments in section III, by using adaptive thresholding instead of thresholding, only square-ROI is detected and no more false square has been detected under uneven lighting condition. We also found that the minimum thickness for a hand-drawn square-ROI must be at least 2 mm in diameter and the recommended optimal viewing distance must be at 30 cm - 36 cm for 6 cm x 5 cm square-ROI, and 34 – 38 cm for 17 cm x 16 cm square-ROI. In terms of execution time and frame per seconds (fps), the proposed technique needed between 16 ms to 19 ms with 28.5 fps to find and detect the desired square-ROI. The results also show that, our proposed technique has fulfilled all three testing parameters stated in section II (D). Hence, this technique efficiently detects the desired hand-drawn square-ROI as a square and also robust to illumination changes by applying Canny's detector and adaptive thresholding.

In the future, we would like to extend our method to purely apply natural marker instead of the hand-drawn square-ROI to detect the Region of Interest (ROI) in real-time.

Acknowledgement

The author wish to thank the GRAVLAB research group for their advice and support. This research is supported by a grant (FRGS0295-SG-1/2011) from the Ministry of Higher Education (MOHE), Malaysia.

References

[1] Milgram, P. and F. Kishino. 1994. A Taxonomy of Mixed Reality Visual Displays. *IEICE Transactions on Information and Systems.* E77-D(12): 1-26.

[2] Azuma, R. 1997. A Survey of Augmented Reality. *Presence-Teleoperators and Virtual Environments*. 6(4): 355-385.

[3] Klein, G. 2006. Visual Tracking for Augmented Reality. Ph.D. Thesis. University of Cambridge.

[4] Xu, K., S. J. D. Prince, A. D. Cheok, Y. Qiu and K. G. Kumar. 2003. Visual Registration for Unprepared Augmented Reality Environments. *Personal and Ubiquitous Computing*. 7(5): 287-298.

[5] Blackwell, M., C. Nikou, A. M. DiGioia and T. Kanade. 2000. An Image Overlay System for Medical Data Visualization. *Medical Image Analysis.* 4(1): 67-72.

[6] William, S. G. D., Progress Report: Adaptive Hybrid Tracking and Registration for Mobile Outdoor Augmented Reality. 1-16.

[7] Carmigniani, J., B. Furht, M. Anisetti, P. Ceravolo, E. Damiani and M. Ivkovic. 2010. Augmented Reality Technologies, Systems and Applications. *Multimedia Tools and Applications*. 51(1): 341-377.

[8] Canny, J. 1986. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 8(6): 679-698.

[9] Baggio, D.L., S. Emami, D.M. Escriva, K. Ievgen, N. Mahmood, J. Saragih and R. Shilkrot. 2012. *Mastering OpenCV with Practical Computer Vision Projects*. First Packt Publishing Ltd.

[10] Wyk, C.V. 2011. Markerless Augmented Reality on Ubiquitous Mobile Devices with Integrated Sensors. Master Thesis. Stellenbosch University.

[11] Tresaco, A. 2009. Markers Recognition in A Camera-based Calibration System for Immersive Applications. Master Thesis. Universitat Politècnica de Catalunya.

[12] Sonka, M., V. Hlavac and R. Boyle. 1993. *Image Processing, Analysis and Machine Vision.* Springer US.

[13] Faille, F. 2003. Adapting Interest Point Detection to Illumination Conditions. Proceeding of the VIIth Digital Image Computing: Techniques and Applications. Sydney. 10-12 Dec, 2003. 10-12.

[14] Nixon, M. and A. Aguado. 2002. *Feature Extraction and Image Processing*. Second Edition. Elsevier.

[15] Qureshi, S. 2005. Chapter 5: Edge Detection and Segmentation. from Embedded Image Processing on the TMS320C600 DSP: Examples in Code Composer Studio and MATLAB. US: Springer.

[16] Idris, M. Y. I., H. Arof, E. M. Tamil, N. M. Noor and Z. Razak. 2009. Review of Feature Detection Techniques for Simultaneous Localization and Mapping and System on Chip Approach. *Information Technology Journal.* 8(3): 250-262.

[17] Bansal, B., J. S. Saini, V. Bansal and G. Kaur. 2012. Comparison of Various Edge Detection Techniques. *Journal of Information and Operation Management.* 3(1): 103-106.

[18] Bin L. and M. S. Yeganeh. 2012. Comparison for Image Edge Detection Algorithms. *IOSR Journal of Computer Engineering (IOSRJCE).* 2(6): 1-4.