International Journal of Electrical, Electronic and Communication Sciences
ISSN: 2517-9438
Vol:6, No:1, 2012

Augmented Reality Interaction System in 3D Environment

Sunhyoung Lee, AskarAkshabayev, BeisenbekBaisakov, Youngjoon Han, Hernsoo Hahn

***Abstract*—**It is important to give input information without other device in AR system. One solution is using hand for augmented reality application. Many researchers have proposed different solutions for hand interface in augmented reality. Analyze Histogram and connecting factor is can be example for that. Various Direction searching is one of robust way to recognition hand but it takes too much calculating time. And background should be distinguished with skin color.This paper proposes a hand tracking method to control the 3D object in augmented reality using depth device and skin color.

Also in this work discussed relationship between several markers, which is based on relationship between camera and marker.One marker used for displaying virtual object and three markers for detecting hand gesture and manipulating the virtual object.

***Keywords*—**Augmented Reality, depth map, hand recognition, kinect, marker, YCbCr color model

I. Introduction

THIS paper is describes our experiment with the video camera which can generate RGB color output and depth map in the one time. We use the information generated by this camera for drawing and manipulating an augmented cube defined by real marker. First, we found the translation of centre point of the marker in the video frame coordinate. By this information we calculate the depth centre point of the marker, e.g. we found the distance between the camera and marker centre. This distance is used as a third – Z coordinate of the centre point. Also, we found the centre of the real hand in the video frame. Finally, by calculating according Z coordinate to this point, we conclude that it's possible to manipulate the augmented cube by real hand in 3D environment.

To take a skin area, convert RGB input image to YCbCr color model. And we can get candidates of the hand. Calculate the center of gravity. And cut out fingers using the circle which has center at the center of gravity. To initialize hand region we count under blobs. Hand rectangle will have 6 to 7 blobs. At this time we should make corresponding depth map pixels to RGB map. This paper includes KINECT calibration method. We can update hand region by set the threshold in the depth.

II. Virtual Objects Interaction

A. Camera and marker relationship

Sunhyoung Lee, Askar Akshabayev and Beisenbek Baisakov are with the School of Information Technology, Soongsil University, 1-1, Sangdo-5Dong, Dongjak-Gu, Seoul, 156-743, KOREA
(corresponding author to provide phone: 02-821-2050; fax: 02-826-8937; e-mail: showgoon@ssu.ac.kr,askar.akshabayev@hotmail.com , beysenbek@gmail.com).

Youngjoon Han and Hernsoo Hahn are with the School of Information Technology, Soongsil University (e-mail: young, hahn@ssu.ac.kr).

Augmented reality is a displaying virtual object in the real world. One way to display virtual object is using special markers. Currently, there are number of libraries, with specific markers, that can detect markers and put some virtual object relatively to detected marker.

One of such kind of library is ARToolKit library, which use size-known square markers as a base of coordinates [1].

After detecting marker, ARToolKit gives transformation matrix, which defines the position of the marker in the camera coordinate system. Rotation and translation information are stored in transformation matrix. [1]

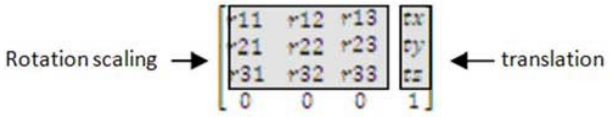

Fig. 1 Rotation and translation matrix.[1]

To get relation between different markers, relation between camera and marker is used. For interaction two different markers we must know the transformation matrix of these markers. Using this information it is possible to find the position of point in one coordinate system in another coordinate system. [1]

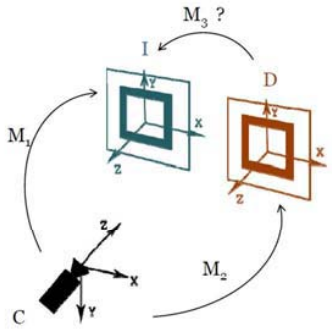

Fig. 2 Relation between markers.[1]

Consider you have three coordinate systems (CS) labeled as*C*,*I* and *D* as shown in figure 1.

$\boldsymbol{M_1}$ - the transformation matrix that transform the representation of a point from CS ***I*** to its representation in CS *C*.

$\boldsymbol{M_2}$ - the transformation matrix that transform the representation of a point from CS *D*to its representation in CS *C*.

$\boldsymbol{M_3}$ -the transformation matrix that transform the representation of a point from CS *I*to its representation in CS *D*.

ARToolKit library give the information about $\boldsymbol{M_1}$and $\boldsymbol{M_2}$ matrices. Using these matrices we must find $\boldsymbol{M_3}$.

$$M_1 = M_{C\leftarrow I} \Rightarrow M_1^{-1} = M_{I\leftarrow C} \quad (1)$$

$$M_2 = M_{C\leftarrow D} \quad (2)$$

$$M_3 = M_1^{-1} M_2 \Rightarrow M_{I\leftarrow D} = M_{I\leftarrow C} M_{C\leftarrow D} \quad (3)$$

Using M_3 we can easily convert the point from one CS to another.

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = M_3 \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (4)$$

B. Hand gesture recognition

Three ARToolKit markers used to hand gesture recognition. One marker is placed at the end of the thumb, second marker at the end of index finger, and third marker in the middle of these two fingers, near the palm as shown in figure 3.

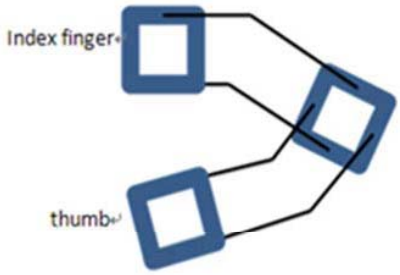

Fig. 3.Location of the markers on the hand

After processing input image, ARToolKit gives information about four vertices of detected marker. Using this information we can easily find line segments of detected marker as shown in figure 4.

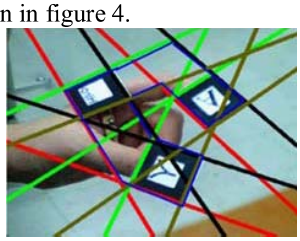

Fig. 4 Line segments of markers.

In the figure 4 you can see lines in that order in which we found them. If we always use this order we can define some lines, and intersection points of these line segments can be used to find hand gesture as shown in figure 5.

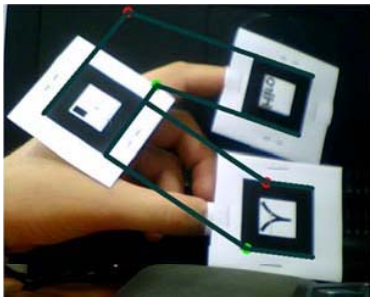

Fig. 5 Hand gesture

C. Algorithm of manipulating virtual object.

This algorithm is one approach of manipulating virtual object using other markers. 4 markers are used for this algorithm, 3 of them to recognize hand gesture and one for displaying virtual object as shown in figure 6. [2] We can denote next variables:

P_1, P_2 – finger point coordinates

P_1', P_2' translated finger points to virtual object coordinates

P_m – middle point of two finger coordinates

P_o – milde point of two finger coordinates in object CS

and make following steps to manipulating virtual objects:

1. Check if P_1' and P_2' are inside the object
2. Continue only if both fingers are inside of object.
3. Store the point in the middle of P_1 and P_2 (in world CS P_m, and in object coordinate system P_o)
4. Store rotation degree between states of fingers line about z-axis. Rotation is calculated as a degree between the previous and current stage of the line, connected by the center point of the finger and the center of the marker placed near the palm.
5. Rotate the object original transformation matrix about P_o (on z-axis)
6. Calculate the current point P_m
7. Translate the object about the difference between current P_m and stored P_m

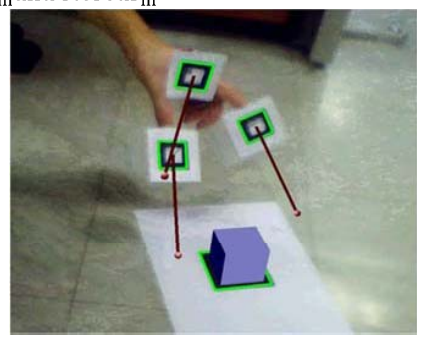

Fig. 6 Virtual object manipulating

III. AR Objects Depth Detection

A. Reading data from sensors device

In our work we used active-sensing depth camera which has structured light, and a color camera. It generates images of 640*480 pixels 30 times a second. For accessing the generated data OpenNI (Open Natural Interaction) framework is used. The term Natural Interaction (NI) refers to a concept where Human-device interaction is based on human senses, mostly focused on hearing and vision. OpenNI is a multi-language, cross-platform framework that defines APIs for writing applications utilizing Natural Interaction. OpenNI APIs are composed of a set of interfaces for writing NI applications. OpenNI defines Production Nodes, which are a set of components that have a productive role in the data creation process required for Natural Interaction based applications. Each production node encapsulates the functionality that relates to the generation of the specific data type [4].

B. Marker detection in the video frame

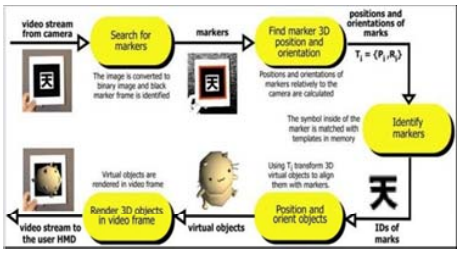

Fig. 7 TheARToolKit tracking steps taken from [1]

ARToolKit is a popular planar marker system for Augmented Reality and Human Computer Interaction (HCI) systems due to its available source code. The bitonal markers consist of a square black border and a pattern in the interior. Color video stream is generated by kinect and transferred to the application layer by the OpenNI Image generator [4]. The image is converted to binary image and black marker frame is identified. After successful identification of marker, the position of the camera relative to the black square is calculated. Area, centre position, lines and vertexes of the detected marker is used to drawing AR object on top of the video of the real world [1].

Fig. 8 Sample marker in frame without virtual object

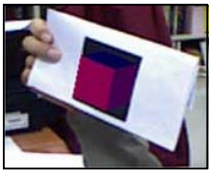

Fig. 9 Sample marker in frame with virtual cube

C. Video frame depth map generating

A depth generator is a production node that is implemented by a sensor, which takes raw sensory data from a depth sensor and outputs a depth map [4]. This data is generated by hardware devices that capture the visual and audio elements of the scene and stored in the 16-bit value. We can consider it as a one dimensional array.

D. Virtual object depth evaluating from the depth map

Let's consider detected marker centre coordinates in the video frame as point P with the coordinates px and py. For finding the correspondence element to P from the depth map we have to evaluate corresponding index. We can use next equation:

$$index = py' * w + px' \quad (5)$$

where w – width of the video frame. Usually, py and px are given by floating numbers. Index is integer value, we can't use *py* and *px* directly, that is why we will use only integer part of this numbers. For reducing error value in the type casting, we have to convert to integer each of number separately, we have:

$$px' = int(px) \quad (6)$$
$$py' = int(py) \quad (7)$$

E. Real hand centre detection

The RGB image information from camera is affected bylight.

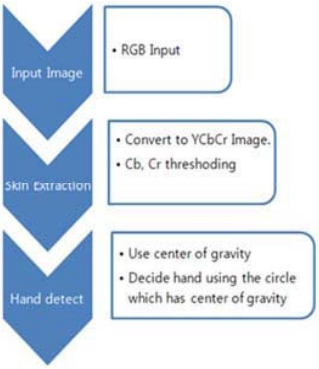

Fig. 10 Flow chart for the hand detection

It is necessary to reduce light effect. We translate this input image to YCbCr color model. Follow equations are transform way when 8bit RGB information use:

$$Y' = 16 + \frac{65.738*R'_D}{256} + \frac{129.057*G'_D}{256} + \frac{25.064*B'_D}{256} \quad (8)$$
$$C_B = 128 + \frac{-37.945*R'_D}{256} + \frac{74.49*G'_D}{256} + \frac{112.43*B'_D}{256} \quad (9)$$
$$C_R = 128 + \frac{112.43*R'_D}{256} + \frac{94.154*G'_D}{256} + \frac{18.285*B'_D}{256} \quad (10)$$

Y is luminance and CbCr is information of color difference. Each range is set77 to 127 and 140 to 170 in this case.

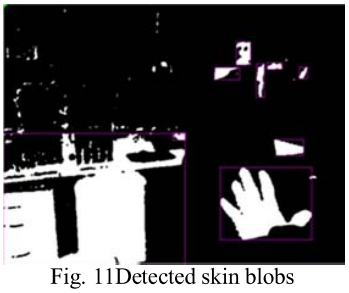

Fig. 11Detected skin blobs

Figure 11 shows the rectangles which include skin area. These rectangles are made by labelling method.

Set every rectangles as a candidate. And make threshold range to the number of pixels and size. Figure 11 shows the rest of rectangles after this process. Now we can find the center of gravity in each region.

Then we will decide the hand rectangle by counting the under blobs.

$$C(x,y) = \frac{\sum_{i=0}^{N} p(x,y)}{N} \quad (11)$$

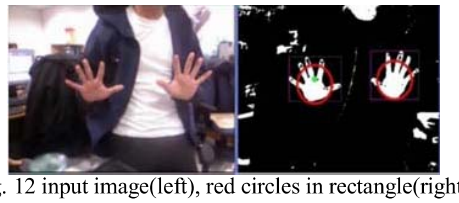

Fig. 12 input image(left), red circles in rectangle(right)

For this operation, we should make changeable circle by every frame. Use the line connected by left top point and centre of gravity. In this case we set 75 percent of that line for radius.

Now we have to get the rectangle which include the hand. At the first step , create same size of mask circle. It will be use to and operation with input image.

Figure 13 shows the result of and operation between mask and input image.

Fig. 13 Input skins binary(Left), Circle mask(centre), and operation result(Right)

We assume if there are 6 to 7 under blobs exist that is rectangle which has the hand.

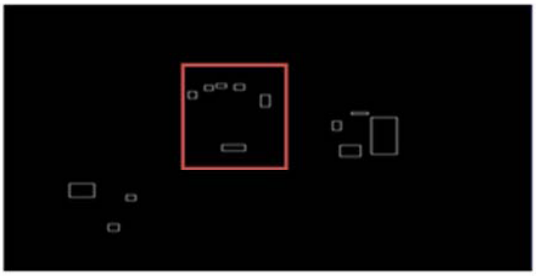

Fig. 14 under blobs for hand decision

As we can see at the figure 14, there is possible that the other object has same condition with hand rectangle. Toreduce that possibility we use the dilation and eroding method to fill the black pepper noise. After this step we use the depth device to track the center point of the hand. “KINECT” is the one of most useful device in computer vision area.

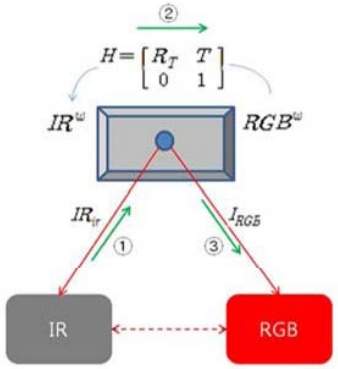

Fig. 15 KINECT calibration

$$IR^w = IR_{ir}^{-1} * IR \quad (12)$$
$$RGB^w = [H^{-1}] * IR^w \quad (13)$$
$$\text{RGB} = [I_{RGB}] * RGB^w \quad (14)$$

Fig. 16 Input image(Left top), Depth map(Right top), without calibration(Left bottom), corresponding two image(Right bottom)

Now we can directly get depth information which correspondsto 2D image plane. We know not only the 2D point which placed at the center of the hand but also 3D information. If we set the depth threshold, hand is the only exist object at that region. In case of that the other object come into this region, we distinguish those objects by color information. Following figure shows the result when we apply the color and depth information in same time.

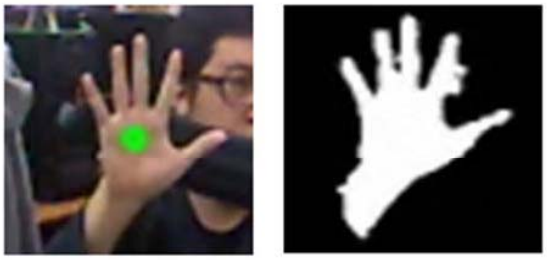

Fig. 17 Maintain center point when another object overlapped(Left), Detect only skin color in threshold depth region(Right)

The "point viewer"[Prime sense open source] example do similar performance with our program but this has better performance when the other object which has not skin color exist at the same depth group. The depth value of the hand centre can be determined with two ways: 1) Using procedure which is described in the detecting virtual cube depth2) directly had taken from the depth generator.

F. Checking the result

For testing algorithm which described above, we use next activity diagram:

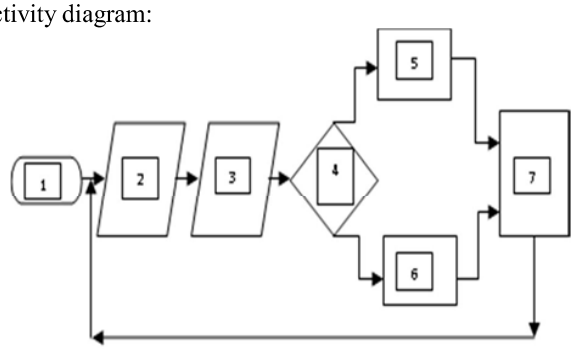

Fig. 18 Basic steps for depth evaluating

1) Hardware and software initializing.
2) Hand centre depth (HCD) evaluating.
3) Virtual object depth (VOD) evaluating.
4) Check if the HCD is equal to the VOD. If it is true then go to 5th step, else go 6th step.
5) Set the virtual object color to blue.
6) Set the virtual object color to red.
7) Draw virtual object on the video frame and return to 2nd step.

Fig. 19 Depth of the cube and hand is not same

Fig. 20 Depth of the cube and hand is same

G. Further work

Depth map errors often lead to noticeable artifacts in 3D video and significantly decrease the resultant quality. Errors often occur in the occlusion areas. In the [5] purposed a method of filtering depth maps provided by Kinect depth camera. Filter uses output of the conventional Kinect camera along with the depth sensor to improve the temporal stability of the depth map and fill occlusion areas. To filter input depth map, the algorithm uses the information about motion and color of objects from the video. The proposed method can be applied as a preprocessing stage before using Kinect output data.

ACKNOWLEDGMENT

This research was supported by the MKE(The Ministry of Knowledge Economy), Korea, under the ITRC(Information Technology Research Center) support program supervised by the NIPA(National IT Industry Promotion Agency)" (NIPA-2011-(C1090-1121-0010))

"This work was supported by the Brain Korea 21 Project in 2011."

REFERENCES

[1] ARTOOLKIT. (2003). Human Interface Technology Laboratory. Retrieved 10 2011, from http://www.hitl.washington.edu/artoolkit/
[2] M.Billinghurst, H. K. (1999). Augmented Reality, 1999. (IWAR '99) Proceedings. 2nd IEEE and ACM International Workshop. Marker tracking and HMD calibration for a video-based augmented reality conferencing system (pp. 85 - 94). San Francisco, CA , USA : ACM .
[3] OpenNI. (n.d.). OpenNI documentation. Retrieved 09 2011, from http://www.openni.org/images/stories/pdf/OpenNI_UserGuide_v4.pdf
[4] Rovelo, G. (n.d.). ARTOOLKIT II. Retrieved 2011, from https://jira.ai2.upv.es/confluence/download/attachments/12222496/WGM18_ARToolKitII.pdf?version=1&modificationDate=1304095263000
[5] S. Matyunin, D. Vatolin, Y. B erdnikov, M. Smirnov. 3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV). Temporal filtering for depth maps generated by Kinect depth came.
[6] V. Buchman, S. Violich and M. Billinghurst and A. Cockburn. (2004). 2nd international conference on Computer graphics and interactive techniques in Australasia and South East Asia. FingARtips – Gesture Based Direct Manipulation in Augmented Reality.
[7] Zhang, Z. (1999). Flexible camera calibration by viewing a plane from unknown orientations. One MicrosoftWay, Redmond,WA 98052-6399, USA : Microsoft Research.