3D Movement Tracking with Asynchronous Digital Cameras for Interactive Systems*

Sehwan Kim and Woontack Woo

K-JIST U-VR Lab.
1 Oryong-dong, Puk-gu, Kwangju, 500-712, Korea
E-mail: {markross, wwoo}@kjist.ac.kr

ABSTRACT

Recent deployments of digital cameras and fast PCs have led real-time interactive systems to adopt vision-based interfaces. However, 2D vision-based interfaces, due to the lack of 3D information, have inherent weakness in tracking 3D objects in real world. In this paper, we propose a vision-based 3D interface for interactive systems that allows an object tracking on the fly in 3D by exploiting multi-view images. Due to the characteristics of interactive systems, requiring real time processing, the main challenge is a simple and robust estimation of 3D information from the images captured by asynchronous digital cameras. The proposed vision-based 3D tracking consists of three steps: (i) dynamic stereo calibration (ii) simple object segmentation (iii) robust 3D movement tracking. To show its effectiveness of the proposed framework, we applied the proposed 3D interface to an interactive 3D Gumdo simulation. Due to the simplicity and robustness of the proposed framework, it can be applied to various real-time interactive applications.

Keywords: Interaction, Vision-based 3D Interface, Asynchronous Digital Camera Calibration, 3D Movement Tracking

1. INTRODUCTION

Nowadays, flourishing Virtual Reality (VR) technologies with the rapid developments of computer hardware and software stir up new challenges for human-computer interface/interaction (HCI). Over the last few years, many researchers have reported a number of promising new ideas for improving the HCI. As a result, traditional HCI methods, such as the keyboard or mouse, will increasingly be replaced by more human-like interface mechanisms. In general, we can bring HCI closer to human-human interaction by providing computers with the ability to understand dynamic gestures, facial expressions, speech, physiological signal and/or emotion. Needless to say, the choice of the HCI techniques strongly influences the acceptability and efficiency for the user within virtual environments. In particular, a vision-based interface will be one of the most effective interfaces toward human-like interaction since it allows to track dynamic gesture without attaching discomfort (wired) sensors [1].

A large amount of research on gesture tracking and analysis has been reported over the last few decades due to its importance in communication between human and computer, as well as between humans. However, many of those systems tend to distract users by requiring conventional interfaces such as keyboard, mouse, gloves, goggles, and/or helmet. Emering et al. described a system for interaction between a real person represented by an avatar and an autonomous actor [2]. The gestures are recognized by the system and the information is transmitted to the virtual actor who is able to react to the gestures and decide which attitude to do. However, problem is that they used magnetic trackers to capture the motion of the real person. Although sensor or marker-based 3D motion tracking provides relatively accurate position and movement information over vision-based tracking, they have several inherent problems. The devices need complicated calibration and conversion procedures to get accurate movement data. In addition, these

* This work was supported in part by Kwangju Institute of Science and Technology (K-JIST) and in part by the Ministry of Education (MOE) through the Brain Korea 21 (BK 21) project.

Visual Communications and Image Processing 2002,
C.-C. Jay Kuo, Editor, Proceedings of SPIE Vol. 4671 (2002) © 2002 SPIE · 0277-786X/02/$15.00

interfaces have limitations in providing immersive interactions because they have to be worn or attached on the body and connected to computers with wires. Therefore, the available 3D movement tracking and analysis schemes, requiring wearing sensors, are far from being a comfortable and/or natural HCI.

In this paper, we propose a vision-based 3D interface for interactive systems that allows an object tracking on the fly in 3D space by exploiting multi-view images. As shown in Figure 1, the movement of an object is traced in 3D space by combining two segmented objects from each camera, with geometry information. However, the main challenge is an efficient estimation of 3D information from the images captured by asynchronous digital cameras. Since interactive systems require real-time responses, simple yet robust real-time movement tracking is essential.

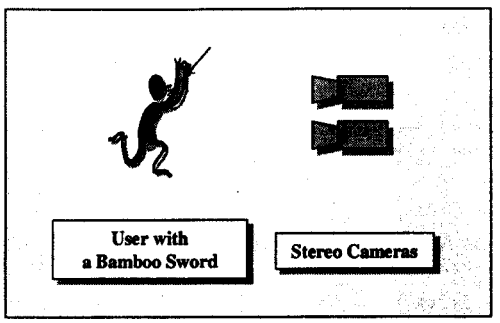


Figure 1. System configuration of 3D vision-based Gumdo game simulation

As shown in Figure 2, the proposed framework consists of three modules: (i) dynamic stereo calibration (ii) simple object segmentation (iii) robust 3D movement tracking. In order to track moving objects using multiple digital cameras, exploiting USB or IEEE 1394, it is necessary to take synchronization issues among cameras into account. As we adopt asynchronous digital cameras, the calibration is a big challenge since the time offset results in an object mismatch problem. Thus, we first propose *a dynamic calibration method* to resolve this problem. After the calibration, we exploit the statistics of the background and color information of objects to segment objects of interest over time-varying background. Finally, we apply the dynamic calibration scheme onto the segmented objects to estimate correct positions of moving objects.

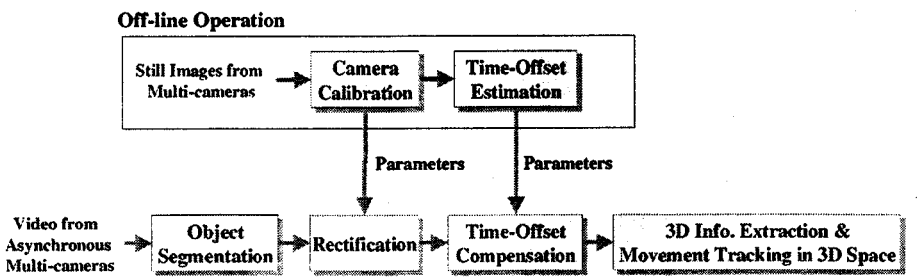


Figure 2. Basic block diagram of 3D movement tracking

The main advantages of the proposed framework are as follows. First of all, it provides users with natural and comfortable interaction by a vision-based 3D interface, unlike conventional methods which distract users by requiring some equipments on the body. In addition, it is simple enough to apply the proposed framework to real time interactive

applications since the proposed framework only tracks the centers of the segmented objects of interest, where the segmented objects of interest are relatively smaller than the moving objects themselves. Furthermore, it is robust even when the proposed framework is applied to a fast movement since we compensate the time offset in the stage of the dynamic calibration. To show its effectiveness of the proposed framework, we applied the proposed 3D interface to an interactive 3D Gumdo simulation. Note however that, due to the simplicity and robustness of the proposed framework, it can be applied to various real-time interactive applications in the areas of entertainment, training, education, engineering, medical operation, teleoperation, etc.

This paper is organized as follows. In chapter 2, we describe the dynamic calibration method in order to make disparity estimation simple and to resolve the time offset problem. Simple object segmentation method using background modeling is explained in chapter 3. In Chapter 4, we explain the vision-based 3D movement tracking based on motion and color information. Finally, experimental results and discussions are followed in chapter 5 and 6, respectively.

2. DYNAMIC CALIBRATION

In general, digital cameras exploiting USB or IEEE1394 are relatively cheaper than analog cameras but the CCD sensors have drawbacks such as color shift or saturation. In order to compensate these effects and keep the quality of the images taken by different cameras, we first perform white balancing. Though auto white balancing is one of the most useful functions in digital cameras, as shown in Figure 6(e), the image may not preserve the original color of the scene. In order to resolve this problem, we capture white image and then estimate the mean and the standard deviation of each color component of the image. After then, we compensate for the difference between the real and captured image. For example, we add $\alpha=(255-m_R-2\sigma_R)$ to the intensity of each pixel, where m_R and σ_R denote the mean and standard deviation of red component of the captured image. The new intensity value of each pixel is limited to (0,255). In the similar way, we can obtain the β and χ for the green and blue components, respectively. Thus, if we move all pixel values by α, β and χ for each component, respectively, we can obtain correctly white balanced images like Figure 6(b) and Figure 6(f).

Without accurate camera calibration, we may fail to estimate accurate 3D information since two cameras have slightly different physical characteristics. Stereo images can be captured using a pair of stereo cameras, where each camera captures a scene from a slightly different perspective. In order to obtain accurate 3D information by exploiting the geometric information of stereo cameras, we need to perform a stereo camera calibration, which determines external camera model as well as internal distortion model. Given a set of corresponding points in 3D space and 2D images, camera parameters can be estimated using a set of equations [16]. In general, standard stereo camera calibration techniques follow 3-step procedures: (i) They establish a list of 3D world coordinates and corresponding 2D image coordinates. (ii) Given this list, camera parameters are estimated for each camera using a set of equations. (iii) The epipolar geometry is constructed from the projection matrices.

We calibrate two IEEE 1394 cameras based on the modified version of Zhang's calibration algorithm [4]. The algorithm estimates intrinsic and extrinsic parameters: The intrinsic camera parameters include the effective focal length, the lens distortion coefficient, the principal point (the center of radial lens). The extrinsic parameters include the rotation matrix R (rotation angles for the transformation between the world and camera coordinate frames) and transformation matrix T (translational components for the transformation between the world and camera coordinate frames).

After performing the calibration, the rectification is accomplished by exploiting the transformation matrix obtained from the relationship between the two sets of extrinsic parameters, $\{R_l, T_l\}$ and $\{R_r, T_r\}$, where the subscript l and r denote left and right, respectively. The calculations associated with stereo algorithms are often considerably simplified when the images of interest have been rectified, i.e., replaced by two projectively equivalent pictures with a common image plane parallel to the baseline joining the two optical centers (Figure 3). The rectification process can be implemented by projecting the original pictures onto the new image plane [5]. After a proper rectification process, that is, with an appropriate choice of coordinate system, (a) the rectified epipolar lines are scan lines of the new images, and

they are also parallel to the baseline and thus (b) corresponding points have identical vertical coordinates. As a result, the matching process of two images can be simplified and efficient [6].

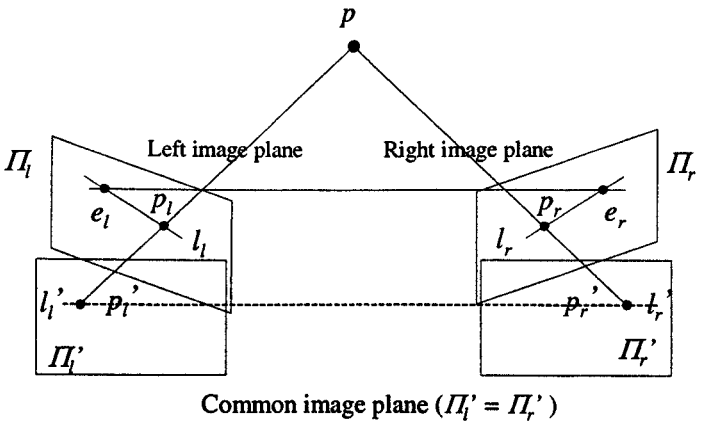


Figure 3. A rectified stereo pair: the two image planes Π_l and Π_r are reprojected onto a common plane $\Pi_l' = \Pi_r'$ parallel to the baseline. The epipolar lines l_l and l_r associated with the points p_l and p_r in the two pictures map onto a common scanline $l_l' = l_r'$ also parallel to the baseline and passing through the reprojected points p_l' and p_r'.

After proper calibration and rectification, we need a time offset estimation to compensate the errors due to the velocity of an object. Tracking objects using asynchronous multi-cameras is a challenging task since we use two asynchronous IEEE 1394 cameras to track the movement of an object in 3D space. If two cameras do not operate synchronously, then we may not find the 3D positions of the moving objects because no corresponding pair of points in the images actually corresponds to the same points in real world. In this scenario, the velocity of objects is an important factor since the video streams from two asynchronous digital cameras have time offsets. If the velocity of the object is negligible over the frame rate, we can ignore the time-offset effects. Otherwise, the problems have to be taken care of. However, the available calibration algorithms assume that the two cameras operate in a synchronous way.

The proposed dynamic calibration method lessens the difficulties in both tracking movement in 3D space, and extracting features of the movement. When we use asynchronous multi-cameras, it is almost impossible to capture two images at the same time on the fly, because we need a little time to save the captured images, which are acquired from right and left camera respectively, into frame memory, resulting in a time offset Δt_i. Nowadays, cameras, such as DragonFly [7], are available for an automatic camera to camera synchronization in real time. However, the camera is too expensive for general purposes. Instead, we exploit two off-the-shelf IEEE 1394 cameras to perform a dynamic camera calibration in estimating 3D information. In this scenario, the velocity of the objects is an essential factor to be considered. If the velocity of the object is negligible over the frame rate, we can ignore the time-offset effects. Otherwise, the problems have to be treated very carefully.

To cope with these problems, we can estimate the effect of asynchronous video capture, as well as the geometric relationship between the two cameras. We first classify the camera calibration for still, slow and fast motion depending on the velocity of an object. In case of still or slow motion, we do not need to consider the dynamic camera calibration, because the time difference between captured two images from right and left cameras, is so small that we can ignore the time offset over the frame rate. However, in case of Gumdo game simulation, brandishing a sword may be included in the fast motion. When the movement of an object is relatively fast compared to the frame rate or time offset, we have to consider the velocity of the object. Figure 4 shows the simple diagram of frame structure according to time. If the time offset is zero, we will get two perfectly matched objects from each camera. However, due to the characteristics of asynchronous cameras, we have to compensate the difference by an appropriate interpolation method. The image frame, which is surrounded by the dotted line, at the time instance of t_i on the right side camera, is a generated frame by the proper interpolation method.

In order to generate interpolated frames, we have some constraints. First, we assume the velocity of an object is uniform. Though the movement or motion of objects is not uniform in general, this assumption is valid in the Gumdo game simulation because we mainly focus on the movement of the sword when the user tries to strike an opponent. Second, we assume the time offset Δt_i, which represents the time difference when two cameras capture and store images into frame buffer, and the processing latency are uniform for all frames. In general, the time offset is very minute because we do some processing only after we store the two frames completely. Thus, we usually do not need to consider the time offset. However, when the velocity of the object is a little fast, problems occur in that the system fails to find the correct sword position. Final assumption is that the movement of the sword is unidirectional. It seems that this assumption is not proper because the motion of the sword is multidirectional. However, we segment the motion mainly into two directions (forward and backward) by analyzing the sword position in each frame.

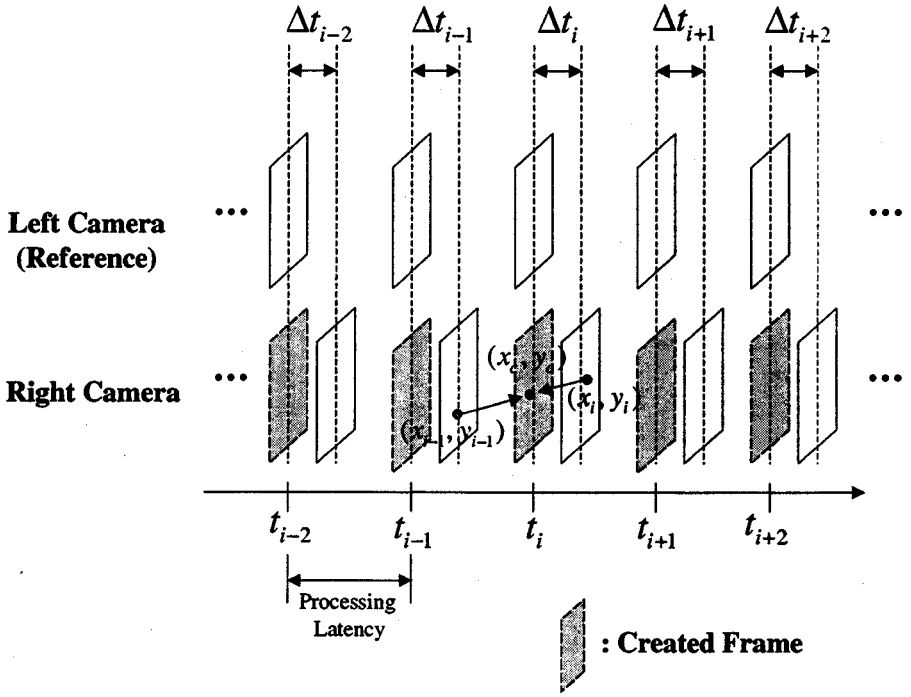


Figure 4. Frame structure according to time

Let the object coordinate of consecutive two frames from right camera at the instance of $t_{i-1}+\Delta t_{i-1}$ and $t_i+\Delta t_i$ as (x_{i-1}, y_{i-1}) and (x_i, y_i), respectively. (x_c,y_c) is the coordinate for a created frame at t_i. And then using linear interpolation between these two frames, we can construct the following equation (1) and (2). Using these simple equations, we can compensate the errors due to the effects of the time offset when the movement of the object is so fast, even though we still have problems when the movement direction of the object changes.

$$x_c = \frac{x_{i-1}\Delta t_i + x_i((t_i - t_{i-1}) - \Delta t_i)}{t_i - t_{i-1}} \tag{1}$$

$$y_c = \frac{y_{i-1}\Delta t_i + y_i((t_i - t_{i-1}) - \Delta t_i)}{t_i - t_{i-1}} \tag{2}$$

3. SIMPLE OBJECT SEGMENTATION

The second step is to segment the user from the natural background scene in order to extract 3D information about the object. There has been a growing interest in object segmentation to provide various applications with object-based functionalities such as interactivity and scalability [8]. The capability of extracting moving objects from video sequences is a fundamental and crucial problem of many vision applications. However, in general object segmentation from natural scenes is still considered to be an open problem [9][10]. The difficulties of automatic segmentation mainly come from defining semantically meaningful areas. Even in cases where we have a clear definition about the areas of interest, an accurate and reliable segmentation is a challenging and demanding task because those areas are not homogeneous with respect to low-level features such as intensity, color, motion, disparity, etc. Up to now, there are lots of methods to separate objects from a static or non-static background [11][12]. Even though the user can be separated from background using bluescreen techniques, in this paper we are not considering such a special environment. Instead, the used scheme segments the user from natural scene by exploiting multiple cues [13][14]. The proposed segmentation algorithm consists of the following three steps: (1) static background modeling (2) shadow removal and (3) moving object segmentation.

The basic idea of separating moving objects from the background scene is to subtract the current image from a reference image that is acquired from a static background during a period of time [8]. The subtraction leaves only non-stationary or new objects. The object separation technique based on static background modeling has been used for years in many vision systems as a preprocessing step for object detection and tracking. However, many of these algorithms are susceptible to both global and local illumination changes which cause the consequent processes to fail. The proposed normalized color space is able to cope with the slight change of illumination conditions. The proposed algorithm for detecting moving objects from a static background scene works fairly well on real image sequences of outdoor scenes, as shown in Figure 7. First, we capture a static background scene and estimate corresponding statistics for each pixel in the image. In order to apply the statistics on each pixel, we exploit $N(m,\sigma)$, where m and σ denote pixelwise mean and standard deviation of the image, respectively. In order to train the static background scene, we gather a number of frames and extract the statistics over those frames. Finally, the statistics $N(m,\sigma)$ are used as threshold values in the initial segmentation process. If we denote the input color image as $I(R,G,B)$, we can calculate the mean and standard deviation images as follows:

$$I_m(R,G,B)_{ij} = \frac{1}{L}\sum_{t=0}^{L-1} I_t(R,G,B) \tag{3}$$

$$I_\sigma(R,G,B)_{ij} = \sqrt{\frac{1}{L}\sum_{t=0}^{L-1}\left(I_t(R,G,B) - I_m(R,G,B)\right)^2} \tag{4}$$

where L denotes the total number of image frames used to estimate statistics. These two images are used for the reference background image and for extracting moving objects as threshold values.

Each pixel can be classified into objects or background by evaluating the difference between the reference background image and the current image in the color space in terms of (R,G,B). To separate pixels in the moving objects from pixels in background, we measure Euclidian distance in RGB color space, and then compare it with the threshold. The threshold is determined in order to obtain a desired detection rate in the subtraction operation. The distance, D_t, and threshold, Th, are defined as follows.

$$D_t = \sqrt{(I_t(R) - I_m(R))^2 + (I_t(G) - I_m(G))^2 + (I_t(B) - I_m(B))^2} \tag{5}$$

$$Th = \alpha\left(I_\sigma(R) + I_\sigma(G) + I_\sigma(B)\right) \tag{6}$$

where variable α is determined according to the changing lighting condition.

To incorporate chromaticity, we introduced normalized color space [8]. Normally, moving objects make shadow and shading effects according to (changing) lighting conditions. However, the (R,G,B) color space is not a proper space to deal with shadow and shading effects. Thus, we used the normalized mean and color images. In the normalized color space, we can separate moving objects from background. The difference between the normalized color image and the normalized mean of the reference image is calculated. If the difference of the pixel is larger than the threshold, then the pixel is classified as a part of background. If the pixel is decided as objects, then we further decompose the color difference into brightness and chromaticity components. Then, based on the observation that shadow has similar chromaticity but slightly different (usually lower) brightness than those of the same pixel in the background image, we can label the pixels into a third category, i.e. shadow. Note that we classify the shadow as a part of the background, since we only need to separate moving objects from the background.

Even after the object segmentation, there could be a few spot-like noises. In this case, we can easily remove these noises using the median filter. After the noise removal, the centroids of the moving objects are estimated. For example, in case of Kendo simulation, the color information of the bamboo sword, which has red and green strips at each end point, is exploited to estimate the centroid of the end points of the sword. Based on the color information and rigidity, we can extract the centroid of the moving objects regardless of the cluttered background while maintaining the simplicity. The object segmentation method is applied to each video sequence from the two cameras. According to our experimental results, the proposed segmentation scheme efficiently separates the user not only from bluescreen but also from real scene, as shown in Figure 7.

4. VISION-BASED 3D MOVEMENT TRACKING

We estimate an appropriate time offset by using the proposed *dynamic calibration method.* In case of Gumdo simulation, we can track the bamboo sword with a proper time offset using color information of red and green strips attached on each end point of the sword. The centroid of the red-colored end point are calculated as follows:

$$x_{rc} = \frac{1}{N_r}\sum_i x_{ri}, \quad y_{rc} = \frac{1}{N_r}\sum_i y_{ri} \tag{7}$$

where N_r is the number of pixels of detected area using the red color. x_{ri} and y_{ri} are the pixel positions in x direction and y direction, respectively. We can also obtain the centroid of the green-colored end point using a similar expression. In the example of Kendo simulation, by tracking the end points of the bamboo sword over a number of frames, we are able to have the corresponding velocity, acceleration as well as the position information.

After stereo camera rectification, we have two cameras so that their optical axes are parallel and separated by a distance b (Figure 5). That is, the baseline is perpendicular to the optical axes. The coordinates of a point $(x,y,z)^T$ in the environment are measured relative to the lens center of the left camera [15]. Let the image coordinates in the left and right images be (x_l',y_l') and (x_r',y_r'), respectively. Then

$$\frac{x_l'}{f} = \frac{x}{z} \text{ and } \frac{x_r'}{f} = \frac{x-b}{z} \tag{8}$$

while

$$\frac{y_l'}{f} = \frac{y_r'}{f} = \frac{y}{z} \tag{9}$$

where f is the distance from the lens center to the image plane in both cameras. These equations can be solved for the three unknowns x, y and z. First, note that

$$\frac{x_l' - x_r'}{f} = \frac{b}{z} \tag{10}$$

If we use the disparity, $x_l'-x_r'$, the difference in image coordinates, we have

$$x = \frac{bx_l'}{x_l' - x_r'}, \quad y = b\frac{(y_l' + y_r')/2}{x_l' - x_r'}, \quad z = b\frac{f}{x_l' - x_r'} \tag{11}$$

Using Eq. (11), we can track the 3D movement of the object in the real space.

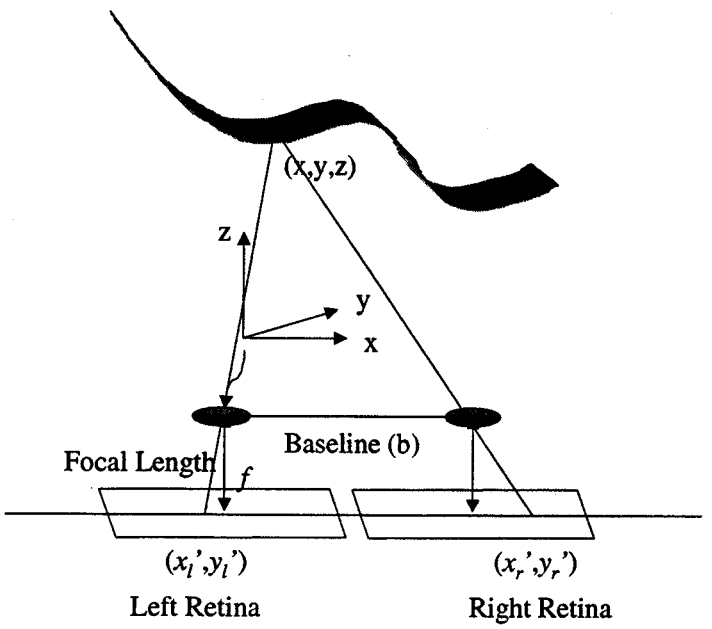


Figure 5. Camera geometry for stereo photography. The optical axes are parallel to one another and perpendicular to the baseline connecting the two cameras

5. SIMULATION RESULTS

We have done experiments in a general indoor environment under a normal fluorescence lighting condition. We employed two off-the-shelf IEEE 1394 cameras to capture a user and its bamboo sword in real time. The speed of the system is limited about 8 Hz mainly due to a background subtraction process. After this process, the system requires little time because we only have to consider the segmented areas, that is, two end points of the bamboo sword. Thus, if we implement the background subtraction process more efficiently, we can increase the speed of the system. Figure 6 demonstrates the white balancing results by the proposed method. Figure 6(a) and 6(b) shows the captured white paper image and its corresponding white balanced image, respectively.

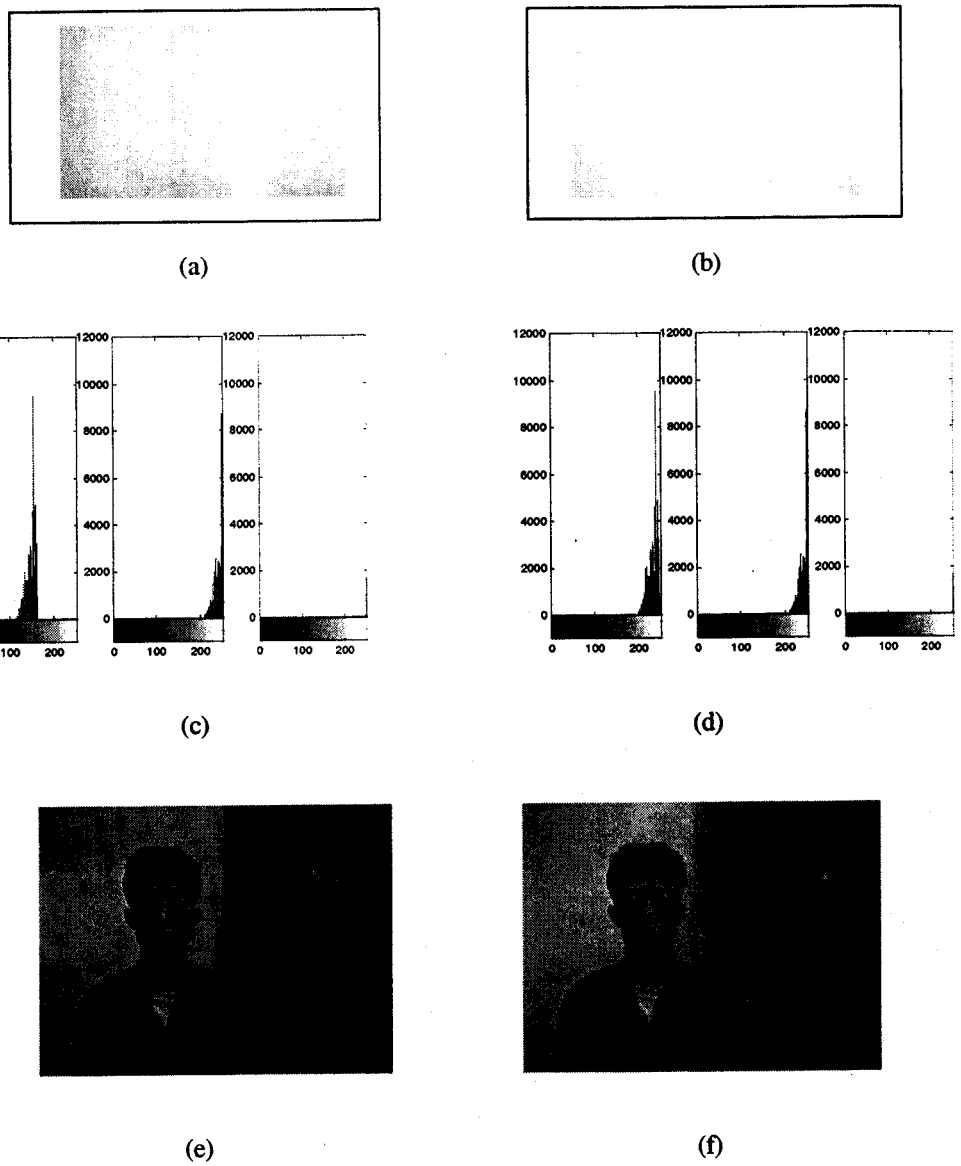
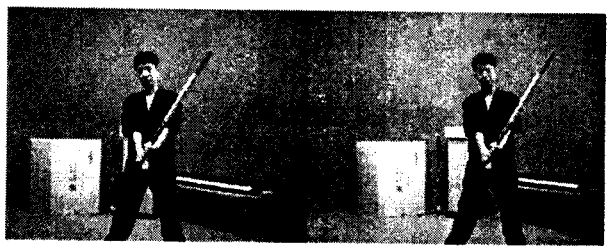
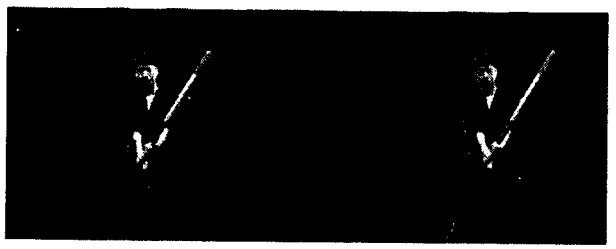


Figure 6. Original image and white balanced image using histogram modification

The original images from the right and left cameras are shown in Figure 7(a) and their corresponding segmentation results in Figure 7(b). On the other hand, Figure 7(c) shows the segmentation results using color information based on the results of Figure 7(b). Because the final segmentation results are obtained by background subtraction results and color information at the same time, we can increase the accuracy. In other words, we can reduce the risk of a cluttered background. We observed that the background subtraction process properly segments the object, irrespective of the background in indoor environments. Through the observation of the experiment, we found that the tracking performance of a sword is sufficient to enjoy a real fighting with a virtual swordsman in virtual environment and tracking is stable with a rapid human motion. In Figure 8, we compare the performance between stereo camera and Digiclops, which is from the Point Grey Research Inc. [7]. The figure shows that the accuracy of the stereo camera is a little lower than that of Digiclops. The reason is that Digiclops employs three lenses to compensate for the errors. Nevertheless, our system works well enough to enjoy the game.



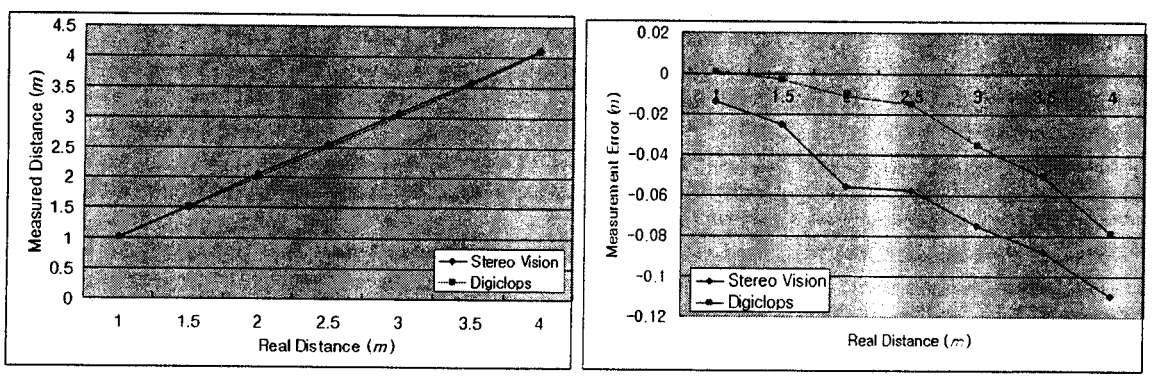
(a)



(b)

(c)

Figure 7. Original images and segmentation results



(a) (b)

Figure 8. Performance Comparison between stereo cameras

6. DISCUSSIONS

Vision-based real-time tracking of an object in 3D space interaction is a challenging task in various interaction systems that basically require simple but robust tracking. However, unlike other interfaces, such as mouse, keyboard, wearable sensor, etc., vision-based interface allows a user to interact with the virtual environment more immersively without distracting the user. As shown in experimental results, the proposed 3D interface using two IEEE 1394 cameras can be applied to an interactive 3D Kendo simulation. When we deal with two or more cameras, the correspondence problem from the different asynchronous real-time video sequences is a big challenge. To lessen the difficulties in 3D information estimation, we used *a dynamic calibration method* by coping with asynchronous video streams. In addition, we segmented objects of interest over time-varying background by adopting normalized color space and statistics. Finally, we tracked the segmented objects by concentrating the centroid of the segmented objects. We showed the effectiveness by applying the proposed 3D interface to an interactive 3D Gumdo simulation. Due to the simplicity and robustness of the proposed framework, it can be applied to various real-time interactive applications.

REFERENCES

1. W. Woo, N. Kim and M. Tadenuma, "Sketch on Dynamic Gesture Tracking and Analysis Exploiting Vision-based 3D Interface," *Proc. of SPIE Intl. Conf. on VCIP*, vol. 4310, Jan. 2001.
2. L. Emering, R. Boulic and D. Thalmann, "Interaction with Virtual Humans though Body Actions", *IEEE Computer Graphics and Applications*, vol. 18, no. 1, pp. 8-11, 1998.
3. S-Y Yoon, R. C. Burke, B. M. Blumberg and G. E. Schneider, "Interactive Training for Synthetic Characters", *Proc. of AAAI 2000*, 2000.
4. Zhengyou Zhang, "Flexible camera calibration by viewing a plane from unknown orientations," *Proc. of the Seventh IEEE International Conference*, vol. 1, pp. 666-673, 1999
5. D. Forsyth and J. Ponce, *Computer Vision - A modern approach*, http://www.cs.berkeley.edu/~daf/book3chaps.html, 2001.
6. W. Woo, N. Kim and Y. Iwadate, "Stereo imaging using a camera with stereoscopic adapter," *Proc. of IEEE - Systems, Man, and Cybernetics (SMC) 2000*, pp. 1512-1517, Oct. 2000.
7. Point Grey Research Inc., http://www.ptgrey.com/, 2001.
8. N. Kim, W. Woo and M. Tadenuma, "Photo-realistic Interactive 3D Virtual Environment Generation Using Multiview Cameras," *Proc. of SPIE PW-EI-VCIP'01*, vol. 4310, pp. 245-254, Jan. 2001.
9. R. Franich, *Disparity Estimation in Stereo Digital Images*, Ph.D. thesis, TUDelft, 1996.
10. H. Jeong, W. Woo, C. Kim, and J. Kim, "A unification theory for early vision," *Proc. of First Korea-Japan Joint Conf. on the Computer*, pp. 298-309, Oct. 1991.
11. T. Horprasert, D. Harwood and L. Davis, "A Statistical Approach for Real-time Robust Background Subtraction and Shadow Detection," *Proc. of IEEE ICCV'99 FRAME-RATE Workshop*, Sept. 1999.
12. A. Elgammal, D. Harwood and L. Davis, "Non-parametric Model for Background Subtraction," *Proc. of IEEE ICCV'99 FRAME-RATE Worksho*, Sept. 1999
13. W. Woo and Y. Iwadate, "Object-oriented hybrid segmentation using stereo images," *Proc. of SPIE IVCP*, pp. 487-495, Jan. 2000.
14. W. Woo, N. Kim, and Y. Iwadate, "Object segmentation for z-keying using stereo images," in *Proc. of WCC*, pp. 1249-1253, Aug. 2000.
15. B. Horn, *Robot Vision*, The MIT Press, 1986.
16. Roger Y. Tsai, "A versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses", *IEEE Journal of Robotics and Automation*, Vol. RA-3, No. 4, pp 323-344, August 1987