3—6

Robust Tracking Algorithm Based on Color and Edge Distribution for Real-time Video Based Motion Capture Systems

Yoshiaki Akazawa[1], Yoshihiro Okada[1] and Koich Niijima[1]
Graduate School of Information Science and Electrical Engineering, Kyushu University

Abstract

This paper describes a robust tracking algorithm for real-time, video based motion capture systems. Conventional motion capture systems are unable to capture motion data in real time because they use many video cameras and take a long time to deal with many images. To deal with problem, the authors have proposed in their earlier work [1] a real-time motion capture system using one video camera. It takes a video image of the upper part of the body of a person, and generates upper body motion data, e.g., x, y, z position of hands and a face rotation in real time and it employs a very simple tracking algorithm, but it suffers from occlusion problem. In this paper, the authors propose a more robust tracking algorithm that solves the occlusion problem. This paper mainly describes this algorithm, and delineates its usefulness by showing its application and experimental results.

1 Introduction

In this paper, we propose a robust tracking algorithm for real-time, video based motion capture systems. Many researches on the motion capture system have been done so far because there has been a great demand of motion data for CG animation productions and 3D game productions. Conventional video based motion capture systems use many video cameras to obtain accurate, desired motion data so they cannot generate motion data in real time because it takes a long time to deal with many video images. Consequently they cannot be used as real-time input devices. Moreover such systems take much cost and require a huge working space so that they are not suitable as input device for a standard PC. To tackle these problems, we have already proposed a real-time video based motion capture system in our earlier research using one video camera [1] and using two video cameras [2].

Generally video based motion tracking systems track the person's motion by extracting person's image from each frame of video camera images and by computing the difference between two successive person's images. We employed a very simple motion-tracking algorithm based on color and edge distributions. Using this algorithm, our system is capable to track the upper part of the body, i.e., hands and a face, and generates their motion data in real time. In addition, it generates z position data of hands according to the image size of the hands. Furthermore the system recognizes specified shapes of a hand, e.g., a paper or a stone shape. To recognize a hand shape, the system calculates the difference between a current hand shape image and a candidate hand shape image by comparing two histograms of their edge distributions. This hand shape information can be used as an input data like the mouse-device button press or release. Moreover the system also has a network communication facility and then the system works as an input device dedicated for an interactive 3D graphics application running on another computer. In this way, the proposed system is useful as a real-time motion input device for a standard PC. However, the system still has the occlusion problem i.e. the system loses a track area as the system cannot distinguish the track area because of being occluded by other track areas. To solve the occlusion problem, we propose a new algorithm using a chain model. In the new algorithm, each main-track area comes to have its support-track area. The support-track area is successively connected to its main-track area like a chain because each track-area is represented as a ring. Even if two main track areas cross with each other, the system can distinguish the two main track areas while their support-track areas separate from each other. This paper mainly describes this algorithm, and clarifies its usefulness by showing its application and experimental results

Related work

Many works on the video based motion capture system have been made so far [3]. Recently motion capture systems without using any markers have been studied [4]. Their standard method for tracking the human motion is based on construction of 3D shape as voxel data from several silhouette images [5]. However, this process needs a lot of computation time. Some particular techniques and other constraints are necessary to reduce this computation time. Weik and Liedtke proposed a hierarchical method for 3D pose estimation [8]. Luck et al proposed a real-time algorithm with reduced number of joints and their degrees of freedom in a human body [9]. These systems use at least four video cameras and need a huge performance space. Our system uses only one video camera. Some methods that use one video camera have been proposed, but our method is simpler than them.

The remainder of this paper is organized as follows. Section 2 describes a system overview briefly. Section 3 explains the tracking algorithm and how it solves the occlusion problem. Section 4 shows an application example and its performance. Section 5 treats occlusion experiment. Finally Section 6 concludes the paper.

[1] Address: 6-1 Kasuga-koen, Kasuga, Fukuoka 816-8580 Japan. E-mail: {y-aka,okada,niijima}@i.kyushu-u.ac.jp

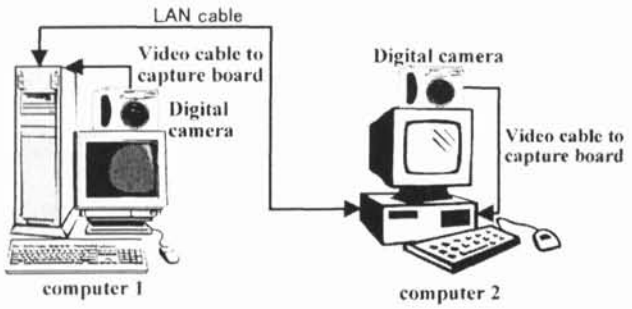

Figure 1: Hardware architecture.

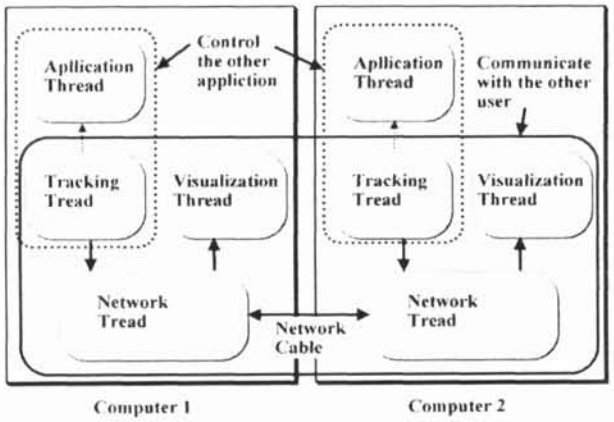

Figure 2: Software architecture.

2 System overview

First of all, as an overview of the system, this section briefly describes its hardware architecture and software architecture.

Hardware architecture

The system hardware consists of a standard PC, a video capturing board, and a video camera. If there are two systems connected with each other through the network as shown in Figure 1, they communicate with each other and work collaboratively. This hardware generates motion data by extracting person's image from each frame of video camera images and by computing the difference between two successive person's images. This motion data is used as an input data for other application. Using the network communication facility (network thread in Figure 2), this motion data is sent to other application running on another computer through the network.

Software architecture

The software architecture has two main threads, i.e., tracking thread and application thread, as shown in Figure 2. Tracking thread tracks the person's motion, generates motion data and sends it to a 3D graphics application, i.e., application thread. Visualization thread displays a virtual person as animation according to the motion data on a display screen. This is used for checking the motion tracking. Finally network thread is a network communication facility itself. Tracking thread sends motion data to other 3D graphics applications running on a different computer through this network thread.

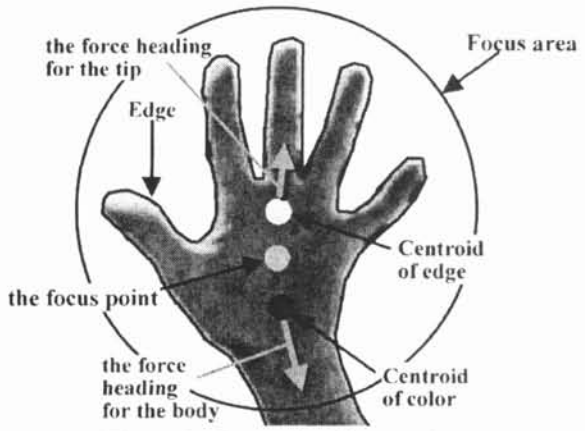

Figure 3: Computing focus point.

3 Tracking Algorithm

The motion tracking is mainly carried out based on the color information of each specific area of the body. Strictly speaking, the median point of the color information is used as the center of the corresponding focus area. It is calculated using equation (1).

$$X_c = \frac{\sum_{i=1}^{m} X_c(i)}{m}, \quad Y_c = \frac{\sum_{i=1}^{m} Y_c(i)}{m} \tag{1}$$

where Xc, Yc are the coordinates of the centroid of the color distribution. $X_c(i)$, $Y_c(i)$ are X, Y coordinates of the i-th color point, and m is the number of color points.

However, practically the color information is insufficient for robust motion tracking. For example, the color of the skin is uniformly distributed over the arm as shown in Figure 3. If the user wants to track his/her hands, its color center is influenced by the arm color and it moves to the center of the arm area gradually. Consequently the system will lose the focus area. To do away with this weakness, we employ new measure involving the edge distribution in addition to the color information. Similar to the color information, the median point of the edges, which are the contour pixels of a focus area, is used as the center of the area. It is calculated using equation (2).

$$X_e = \frac{\sum_{i=1}^{n} X_e(i)}{n}, \quad Y_e = \frac{\sum_{i=1}^{n} Y_e(i)}{n} \tag{2}$$

where X_e, Y_e are the coordinates of the centroid of the edge distribution. $X_e(i)$, $Y_e(i)$ are X, Y coordinates of the i-th edge point and n is the number of edge points.

The edge centroid is always located on the upper part of the hand, so the system does not lose the focus area. However, the edge centroid is strongly influenced by the change in hand shape. Therefore, we use weight values for both the color centroid and the edge centroid. As a result, the focus area becomes stable. The centroid of the focus area is calculated using the equation (3).

$$X_p = \frac{w_c X_c + w_e X_e}{w_c + w_e}, \quad Y_p = \frac{w_c Y_c + w_e Y_e}{w_c + w_e} \tag{3}$$

where X_p,Y_p are the centroid coordinates of the focus area. w_e is the weight of the edge and w_c is the weight of the

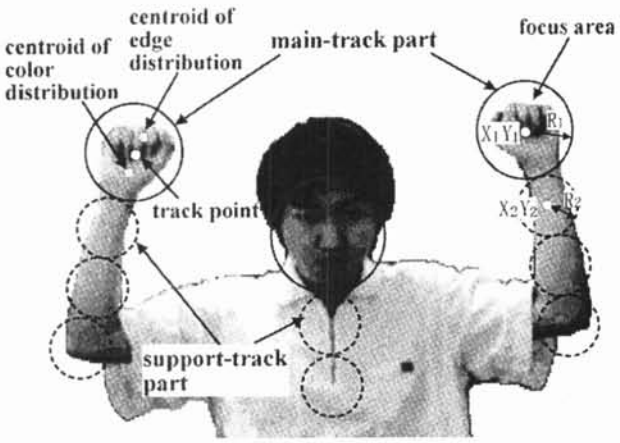

Figure 4: Tracking body parts using support-track parts.

color.

Chain model to solve the occlusion problem

Above method is very simple rather than other already proposed tracking method. Using this method, the system tracks the human motion in real time. However, it does not solve the occlusion problem. To solve the occlusion problem, we decided to introduce support-track parts. The support-track part is additional track area adjoining to each main-track part similar to a chain as shown in Figure 4. The main-track part means the original track area. The first support-track part follows its main-track part, the second support-track part follows the first support-track part and the third support-track part also do so. A parent-child relationship exits between two adjoining track parts. The location of each support-track part is calculated using equation (4). Actually this equation corresponds to the first support-track part adjoining to its main-track part.

$$X_2 = \frac{(R_1+R_2)X_1+(D'-R_1-R_2)X'_2}{D'},\ Y_2 = \frac{(R_1+R_2)Y_1+(D'-R_1-R_2)Y'_2}{D'} \quad (4)$$

where $D' = \sqrt{(X_2'-X_1)^2+(Y_2'-Y_1)^2}$. X_2, Y_2 are the coordinates of the centroid and R_2 is the radius of the first support-track part after adjusting to its main-track part R_2 is a constant value specified by the user. X_1, Y_1 are the coordinates of the centroid and R_1 is the radius of the main-track part. X_1, Y_1 are calculated using the equation (3). R_1 is also determined according to the size of the main-track part. X_2', Y_2' are the coordinates of the centroid of the support-track part before adjusting to its parent-track part. X_2', Y_2' are also calculated using the equation (3).

As shown in Figure 4, by calculating the location of each support-track part using equation (4), each support-track part adjoins to its parent support-track part like chains. Even if the right hand occludes the left hand, i.e., the main-track part of the right hand occludes the main-track part of the left hand, if adjoining support-track parts of the right hand are apart from adjoining support-track parts of the left hand, the system does not lose the main-track area of the right hand. In this way, by introducing support-track parts to each original track area, the tracking algorithm becomes almost free from the occlusion problem and the system becomes robust.

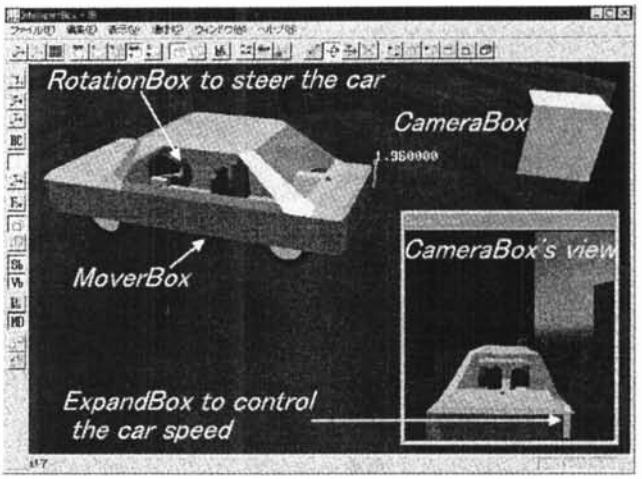

Figure 5: Car driving simulator.

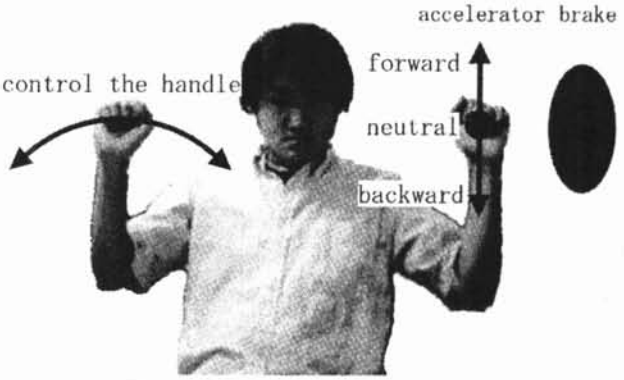

Figure 6: User control example.

4 Application and performance

This section introduces one example of interactive Virtual Reality applications. This application is developed using *IntelligentBox* [6,7], which is a constructive visual 3D software development system. This application provides a 3D virtual space in which a user can drive a car through using our prototype system as an input device. As shown in Figure 5, a camera is attached with the back side of the car. This camera is not a real camera but a software component provided by *IntelligentBox*. This camera is used to display the user's view image. By looking at this view image, the user can drive a car in the 3D virtual space efficiently. The car consists of the same elements as the actual car, e.g., a handle, an accelerator, and a brake. Therefore the car movement is actually controlled by these objects. The user uses his/her hands to control these objects as shown in Figure 6. The user mainly controls the handle to drive the car. When he/she moves his/her right hand like the actual handle operation, the car changes its moving direction. Moreover, he/she moves his/her left hand to control the accelerator and the brake. The car goes forward when he/she raises his/her left hand. And, it goes backward when he/she lowers the hand. In addition, to drive the car smoothly, a neutral position of the accelerator is necessary. When he/she moves the hand into the neutral position, the car comes to keep its speed. Moreover, when he/she wants to make the car stop or to reduce the car speed, all he/she has to do is to move the hand into the brake position as shown in Figure 6. In this way, the car moves freely in the 3D virtual space by the user body motion.

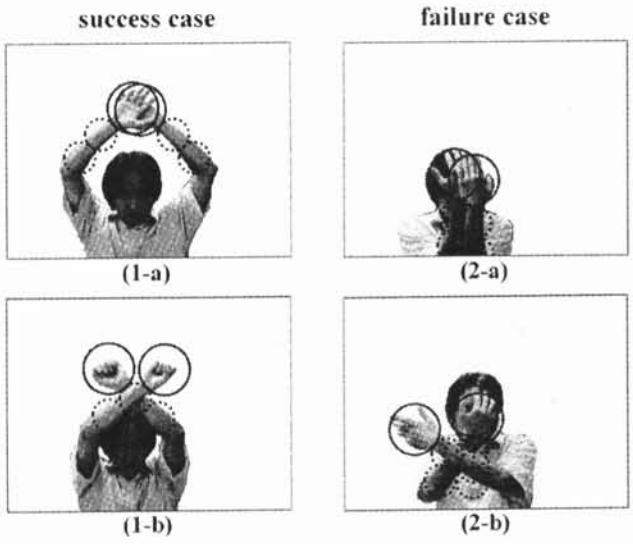

Figure 7: Typical occlusion cases; (1-a) cross hands, (1-b) intersect hands, (2-a) pile up arms, and (2-b) cover face with hands.

Finally, we measured the performance of our system on the standard PC (Pentium IV 2.0 GHz, 1.5GB Memory) with one video camera. Table 1 shows the results. When the resolution of capturing is 160x120 pixels, the sampling rate is at around 15 fps. When its resolution is 320x240 pixels, the sampling rate is also at around 15 fps. When its resolution is 640x480 pixels, the sampling rate becomes at around 7 fps. As Table 1 shows, the tracking latency is proportional to the image resolution. However, the capturing latency is not proportional because it depends on the capture board architecture. Therefore, the frame rate is not proportional to the resolution.

Table 1: Performance

$$Frame\,rate = \frac{1000}{Capturing\,latency + Tracking\,latency}$$

Resolution (pixels)	Frame rate (fps)	Capturing latency (ms)	Tracking latency (ms)
160x120	15.1	62.5	3.9
320x240	15.2	50.3	15.5
640x480	7.4	67.1	67.3

5 Occlusion experiment

We experimented on different occlusion cases. As previously mentioned, our algorithm works correctly in general cases. However, in some cases, the occlusion problem still remains. We checked four typical occlusion cases as shown in Figure 7. (1-a) and (1-b) are success cases. (1-a) is the case where the user crosses his/her hands. (1-b) is the case where the user intersects his/her arms. In these two cases, our proposed method works correctly. On the other hand, (2-a) and (2-b) are failure cases. (2-a) is the case where the user piles up his/her hands and arms. In this case, our system cannot distinguish between both hands. (2-b) is the case where the user covers up his/her face with his/her hands. In this case, it is impossible to calculate the position of support-track parts so that the system cannot track the main-track parts. Actually these cases are very difficult to solve when using only one video camera. We will try to solve these cases in order to improve and to make our algorithm more and more robust.

6 Conclusion

In this paper, we proposed the real-time, video based motion capture system that copes with the occlusion problem. Since conventional video based motion capture systems use many video cameras and take long time to deal with many video images, they cannot generate motion data in real time. Therefore they cannot be used as real-time input devices for a standard PC. On the other hand, our proposed system uses only one video camera and generates motion data in real time since our system employs a very simple tracking algorithm based on color and edge distributions of tracking focus areas. Our system can be used as an input device for a standard PC. Moreover, we solved the occlusion problem and described it in detail. Therefore our system tracks the body more robustly. In addition, we clarified the usefulness of our system by showing the application example and its performance results. As future work, we will develop more application examples and evaluate their performance to improve our algorithm.

References

[1] Y. Akazawa, Y. Okada, and K. Niijima, 2002, "Real-time Motion Capture System Using One Video Camera Based on Color and Edge Distribution," Proc. of CSCC2002 (Recent Advances in Circuits, Systems and Signal Processing), WSEAS Press, 368-373.

[2] Y. Akazawa, Y. Okada, and K. Niijima, 2002 "REAL-TIME VIDEO BASED MOTION CAPTURE SYSTEM BASED ON COLOR AND EDGE DISTRIBUTIONS," Proc. of IEEE Int. Conf. on Multimedia and Expo, Vol. II, 333-336.

[3] Gravrila, D. M. 1999. "The visual analysis of human movement: A survey." CVPR, Vol. 73, 82-98.

[4] Wren, C., Azarbayejani A., Darrel, T. and Pentland, T. 1997. "Pfinder: Real-time tracking of the human body." IEEE Trans. Pattern Anal. and Machine Intel., Vol. 9, No. 7, 780-785.

[5] Snow, D., Viola, P and Zabih, R. 2000 "Exact voxel occupancy with graph cuts." in Proc. IEEE CVPR.

[6] Okada, Y. and Tanaka, Y. 1995. "IntelligentBox: A Constructive Visual Software Development System for Interactive 3D Graphic Applications." Proc. of Computer Animation '95, IEEE Computer Society Press, 114-125.

[7] Okada, Y. and Tanaka, Y. 1998. "Collaborative Environments of IntelligentBox for Distributed 3D Graphics Applications." The Visual Computer, Vol. 14, No. 4, 140-152.

[8] Weik, S. and Liedtke, C.-E. 2001. "Hierarchical 3D pose estimation for articulated human body models from a sequence of volume data." Robot Vision 2001, LNCS 1998, 27-34.

[9] Luck, J., Small, D. and Little, C.-Q. 2001. "Real-time tracking of articulated human models using a 3D shape-from-silhouette method." Robot Vision 2001, LNCS 1998, 19-26.