Real time motion tracking system for interactive entertainment applications

Article · January 2002

CITATION
1

READS
26

4 authors, including:

Chan-Mo Park
Pohang University of Science and Technology
56 PUBLICATIONS **1,312** CITATIONS

REAL TIME MOTION TRACKING SYSTEM FOR INTERACTIVE ENTERTAINMENT APPLICATIONS

Namgyu Kim, Jaeyong Chung[†], Gerard Jounghyung Kim, and Chan-Mo Park
Virtual Reality Laboratory, Department of Computer Science and Engineering,
Pohang University of Science and Technology (POSTECH)
San 31, Hyoja-dong, Nam-gu, Pohang, Kyung-buk, 790-784, KOREA
[†]VR Research Center, Electrics and Telecommunications Research Institute (ETRI),
161 Gajeong-Dong, Yuseong-Gu, Deaejeon, 305-350, KOREA

ABSTRACT

"Looking human through a camera" is a potentially powerful technique to facilitate human-computer interaction. Many 2D/3D visual human tracking techniques have been introduced in area of computer vision, and applied to smart surveillance, motion analysis, interactive computer graphics and virtual reality applications. For applications emphasizing real time interactivity, the visual tracking techniques need to be fast, robust, and preferably run on an inexpensive hardware system. In this paper, we present a relatively inexpensive (e.g. run on a high-end PC) but reasonably robust real time motion tracking system based on a simple 2D color object segmentation and recognition algorithm. Users grasp color objects for achieving more exact tracking performance on their hands and make predefined motion gestures continuously. Through object segmentation processing based on color information, 2D positions of the objects are computed. And then, a motion gesture corresponding on these 2D position trajectories is found by a simple correlation-based matching algorithm. Also, we demonstrate this system by applying it to a popular interactive entertainment (e.g. TETRIS).

INTRODUCTION

Computer vision technology gives more convenient interaction between human and computer or machine in virtual reality and interactive computer graphics application areas. With the advancement of this vision technology, many 2D/3D visual human tracking algorithms have been introduced with head orientation, hand gesture, gaze tracking, human body tracking, body posture recognition, etc[3]. For communicating with a computer, we no longer want to use cumbersome devices such as mouse, keyboard, glove, and wands etc. In addition, to give immersive illusion to the users, systems should provide more natural and intuitive way for controlling the user's movement and action in virtual environments, which exploit usual legs' motion [1], intended arm motion gestures [6], and blob-based body tracking technique [2].

To interpret users' intention and give interaction illusion to users, motion gesture recognition is necessary [7][9]. Aside from just recognizing the gesture itself, moving gestures create another subproblem that is, detecting the starting and ending points of the intended gesture in the midst of position data that are streaming in. One simple solution is to define a "still" state. For instance, the user is required to be stationary for few seconds to signal the start and the end of a motion command. To overcome such inconvenience, our system finds a meaningful motion pattern from a streaming data contained within a finite data search window [7].

Nowadays, real-time gesture or posture recognition products are commercially available for general PC-based environments [10][11] as well as high cost or high performance virtual environments [4][5][8]. These commercial products make computers easier to use. For example, in computer game applications, rather than pressing button, the player could perform gestures and actions, which the computer would then recognize [10][11][12]. Our system is similar to those in that it is a users' motion gesture based interaction system.

The following is the organization of this paper. First, our color object segmentation scheme and 2D position tracking algorithm are described. Then, motion gesture recognition procedures with 2D position input trajectory and correlation-based matching are described. Finally, we demonstrate this system by applying it to a popular interactive game (e.g. TETRIS).

COLOR-BASED OBJECT TRACKING

There has been a growing interest in object segmentation to be used in various applications with object-based functionalities such as interactivity and scalability. The capability of extracting moving objects from image sequences is a fundamental and crucial problem of many vision applications. The difficulties of automatic segmentation mainly come from defining semantically meaningful areas. To alleviate these difficulties, we use five color props – red, green, blue, magenta, and yellow balls.

(1) Object Segmentation

We first gather color statistics over an image region in a static props-taken frame. Let the color image region for each prop be $I(R,G,B)$.

The mean value $I_m(R,G,B)_{[color]}$, is used for the reference color data and the standard deviation $I_\sigma(R,G,B)_{[color]}$, is used for threshold values on extracting objects. The mean and deviation are calculated as follows:

$$I_m(R,G,B)_{[color]} = \frac{1}{L}\sum_{t=0}^{L-1} I_t(R,G,B)$$

$$I_\sigma(R,G,B)_{[color]} = \sqrt{\frac{1}{L}\cdot\sum_{t=0}^{L-1}(I_t(R,G,B)-I_m(R,G,B))^2}$$

where L denotes the total number of pixels in the image region. The image region for color props are selected by the users. Each pixel can be classified into objects or others by evaluating the difference between the reference color data and the current image in the color space in term of (R,G,B). To separate out pixels in the objects, we measure Euclidian distance in RGB color space and compare with the threshold. The threshold value is determined in order to obtain a desired detection rate in the subtraction operation. The distance, D_t, and threshold, Th, are defined as follow.

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

$$Th = \alpha\cdot(I_\sigma(R)+I_\sigma(G)+I_\sigma(B))$$

where variable α is determined according to the changing lighting condition. The reasonable value is about 3.0 practically. As shown in Figure 1, threshold value determines whether a pixel is a reference object pixel or other pixel.

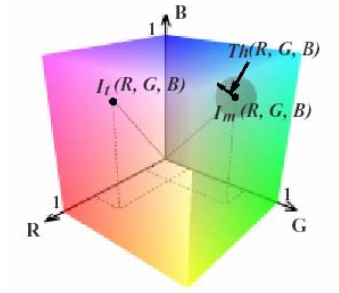


Figure 1 Classification in RGB color space

In addition, we apply morphological filter twice (3x3 close and erode filters) to fill the hole and remove spot noises. Figure 2 shows color props (balls) and segmented object's mask images, and Table 1 shows the color statistics for the object segmentation.

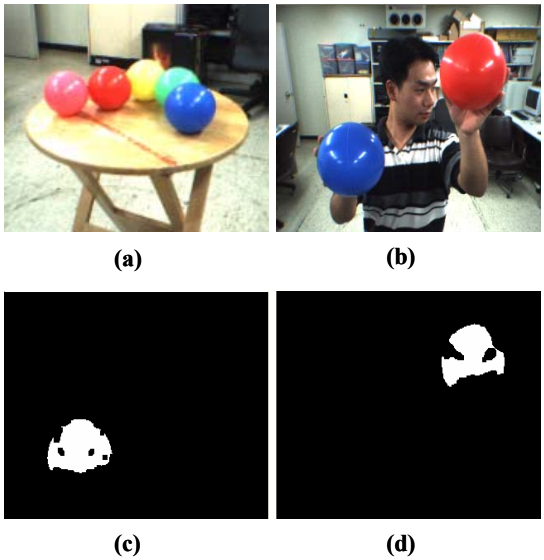


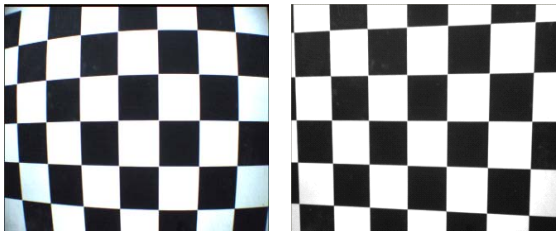
Figure 2 Object segmentation. (a) Five color props (from left to right, magenta, red, yellow, green and blue color balls) (b) Current image. A blue ball (lower-left on the image) on the right hand, and a red ball (upper-right on the image) on the left hand (c) Segmented object for the blue ball (d) Segmented object for the red ball.

	Red	Green	Blue
Blue ball	51.0 (17.62)	109.7 (25.8)	239.1 (23.4)
Red ball	238.6 (14.1)	57.2 (11.2)	50.7 (12.1)

Table 1 Color statistics of blue and red balls. Mean and standard deviation (in parentheses).

(2) 2D Position Tracking

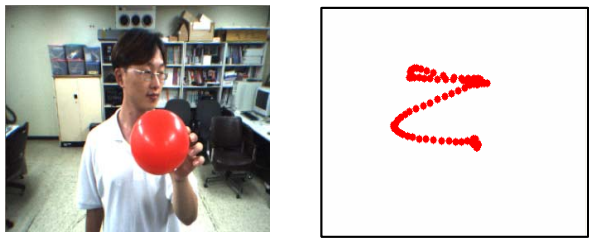
2D positions are computed from their centroids on segmented object mask images for simplicity (See figure 2 (c) and (d)). Although some noise on an image has been noticed, it is not significant in recognizing object motion. For more robust 2D position tracking, we adjust camera lens distortion error – known as rectification process, so that linear movements on 3D real space can be preserved on 2D image space. As shown in Figure 3, linearity of the check board is preserved on the rectified image.



(a) (b)

Figure 3 Rectification. (a) Distorted image (b) Un-distorted (Rectified) image

Figure 4 shows tracked 2D position trajectories in the given interval continuously. Although the red region on the real image is large, the corresponding 2D position is represented by a small dot by using its centroid value.



(a) (b)

Figure 4 2D position tracking: user writes 'z' character with a red ball. (a) Real image (b) 2D position trajectory

MOTION GESTURE RECOGNITION

The user makes continuous motion by moving an arm with the color ball, and the position of the ball tracked by the object segmentation scheme is used as an input to the gesture recognition module. We modeled five basic motion gestures for interactive entertainment environments (e.g. arcade game, TETRIS): go left, go right, rotate left, rotate right and drop. The motion gestures are shown in Figure 5.

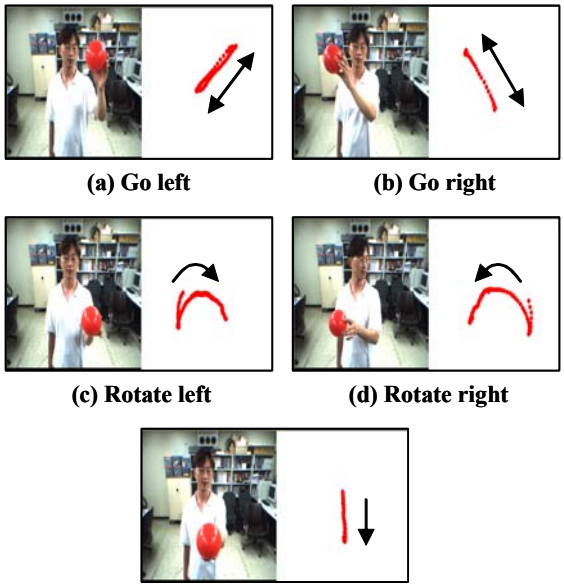


Figure 5 Motion gestures and trajectories of five motion gestures

Aside from just recognizing the gesture itself, moving gestures create another subproblem, that is, detecting the starting and ending points of the intended gesture in the midst of position data that are streaming in. One simple solution is to define a "still" state, and for instance, require the user to be stationary for few seconds to signal the start and the end of a motion command. To overcome such inconvenience, our method looks for a meaningful motion pattern from a stream of data contained within a finite data search window. The data search window starts at a minimum length (e.g. 0.25 seconds, or 5 frames at 20 Hz sampling rate) from the current frame, and grows to a predefined maximum (e.g. 3 seconds, or 60 frames at 20 Hz sample rate). The predefined minimum and maximum lengths of the search window are determined based on heuristics that a given gesture command would require at least that minimum amount of time to be carried out, and must not exceed that maximum amount of time to be completed.

The varying length of the motion command is handled through a normalization process of the sampled data. That is, the motion command template data size is either truncated or elongated to fit the input data size before applying the match algorithm. Thus, as far as the duration of the motion command is kept within a reasonable bound, it will be recognized.

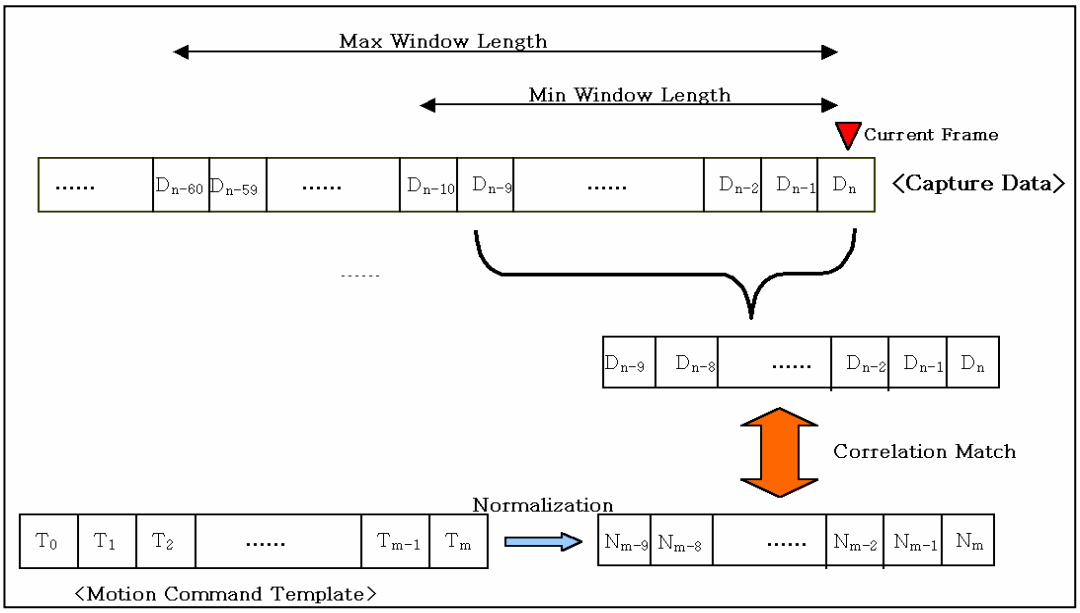


Figure 6 Searching for a match in the data search window

$$Corr_m(u) = \frac{n\left(\sum_{i=0}^{n} T_m(i) I_m(u+i)\right) - \left(\sum_{i=0}^{n} T_m(i)\right)\left(\sum_{i=1}^{n} I_m(u+i)\right)}{\sqrt{\left[n\left(\sum_{i=0}^{n} T_m^2(i)\right) - \left(\sum_{i=0}^{n} T_m(i)\right)^2\right]\left[n\left(\sum_{i=0}^{n} I_m^2(u+i)\right) - \left(\sum_{i=0}^{n} I_m(u+i)\right)^2\right]}}$$

where $m = x,y$

u = current image index $- n$

T = Templete data, I = Input data

Figure 7 Computing for the correlation coefficient

The above search and match process is repeated at every data sampling period (which is about 20 Hz). Once a gesture is recognized the data can be further analyzed for additional input properties such as speed or acceleration. However, being time dependent, this simple algorithm can not handle gestures that are similar in part, for instance, between a "C" and an "O" motion. A "C" gesture would be recognized in the midst of giving an "O" gesture. Figure 6 show a flow diagram for a match in the data search window

Given a data search window, a particular motion gesture is found through a correlation－based match algorithm. The correlation analysis basically analyzes for how well a regression fits the sampled data. The formula shown in Figure 7 computes for a measure of the quality of the fit between the input and the template motion data, and the correlation coefficient is computed in 2D dimensions.

If the correlation coefficient is higher than a predefined threshold value (e.g. 0.85, +1 representing a perfect correlation), it is considered as a mathch. By using the correlation analysis, the ratio of correct classification is made much less sensitive to slight tracking error. In addition, the recognition is made independent from the size or location of the gesture, thus there is no need for data normalization.

APPLICATION

We demonstrate this system by applying it to a popular interactive entertainment game (e.g. TETRIS). For input command, previous five gestures - go left, go right, rotate left, rotate right and drop – are used. A keyboard-like event invoking module is implemented for connecting the game application. For example, "drop"

motion gesture is corresponding to a "DOWN" arrow key in the game application. (See Figure 8 for an overview of the application.). All step performs in real-time, about 20 fps on Pentium4 2.0Ghz PC environment with a IEEE 1394 camera.

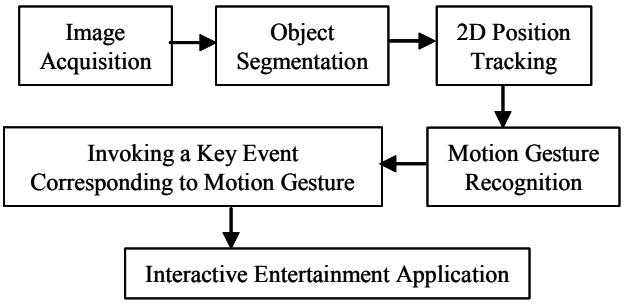


Figure 8 Basic structure of applications

In this interactive environment, a user does not input through a keyboard any more, instead enjoys the application with his body movement actively as shown in Figure 9.

Figure 9 A User enjoys the game without using a keyboard.

This game application with real-time motion gesture recognition is exhibited at the KSF2002 (Korea Science Festival 2002, Pohang, Korea).

CONCLUSION

In this paper, we present a relatively inexpensive (e.g. run on a high-end PC) but reasonably robust real time motion tracking system based on simple 2D color object segmentation and recognition algorithm. A user grasps color props on his one/two hands and performs predefined motion gestures. As illustrated with an interactive entertainment application, users could perform with ease or enjoy the application as compared with previous keyboard interface. The system we developed can open door to create more stimulating interfaces to many existing applications and bring virtual reality to the general public.

REFERENCES

[1] C. Chang, W. Tsai, "Vision based Tracking and Interpretation of Human Leg Movement for Virtual Reality Applications", IEEE Trans. On Circuits and Systems for Video Technology, Vol. 11, No. 1, 2001

[2] C. R. Wren, A. Azarbayejani, T. Darrell, A. Pentland, "Pfinder: Real Time Tracking of Human Body", IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 19, No. 7, pp. 780-785, 1997

[3] D. M. Gavrila, "The visual analysis of human movement: a survey", Computer Vision and Image Understanding, pp. 82-98, 1999.

[4] Immersion Corp., http://www.immersion.com

[5] iReality.com Inc., http://www.genreality.com

[6] J. Chung, N. Kim, G. J. Kim, and C. M. Park., "A Low Cost Real-Time Motion Tracking System for VR Application", International conference on Virtual Systems and MultiMedia, 2001.

[7] J. LaViola, "A Survey of Hand Posture and Gesture Recognition Techniques and Technology", Technical Report CS-99-11, Brown University, Department of Computer Science, Providence RI, June, 1999.

[8] JesterTek Inc., http://www.jestertek.com

[9] R. Watson, "A Survey of Gesture Recognition Techniques", Technical Report TCD-CS-93-11, Department of Computer Science, Trinity College, Dublin 2, 1993.

[10] Reality Fusion Inc., http://www.realityfusion.com

[11] Vivid Group, http://www.vividgroup.com

[12] W. T. Freeman, D. B. Anderson, P. A. Beardsley, C. N. Dodge, M. Roth, C. D. Weissman, W. S. Yerazunis, H. Kage, K. Kyuma, Y. Miyake, and K. Tanaka, "Computer Vision for Interactive Computer Graphics", IEEE Computer Graphics and Applications, Vol. 18, No. 3, pp. 42-53, May-June 1998.