影片中即時追蹤二維物件軌跡以轉換三維物件資訊
PLAY WITH VIDEO: AN INTEGRATION OF HUMAN MOTION TRACKING AND INTERACTIVE VIDEO

摘要

在這次的計畫中，我們提出了以物件追蹤技術以及其他應用到的影像處理技術來達到「人機互動」的效果。我們利用從攝影機讀入的影像來源，對其加以分析所內含人體動量、關節點座標等資訊。並利用這些資訊讓使用者可以直接控制電腦，與虛擬場景互動。

關鍵詞：人體動量追蹤，影片互動，人機互動

ABSTRACT

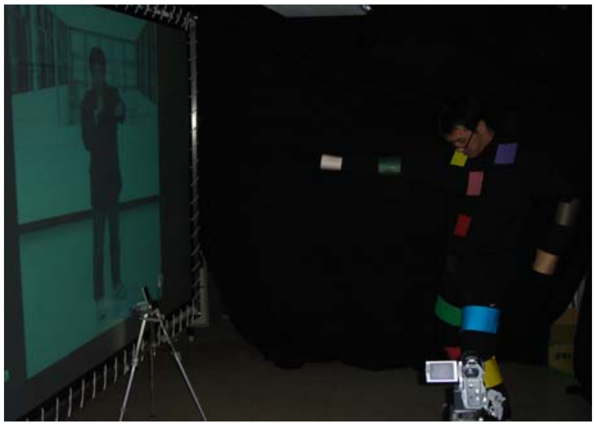
We propose a systematic strategy for human action tracking, with an important extension to integrate tracked skeleton with an interactive video player. The interactive video player can detect and understand a restricted set of human motions and react by jumping to particular video segments pre-recorded. As a consequence, the system allows a player to fight with an avatar or a real person in an interactive video game.

Keywords: Human Motion Tracking, Interactive Video, human computer interaction

1. INTRODUCTION

Human motion tracking was developed in the past few years. The MARG sensors are used to develop a system which can embed skeleton into a virtual environment [1]. Instead of using sensors, video-based tracking strategy was proposed [3, 4]. The difficulty of video-based approach is on how to deal with incomplete motion such as occlusion or bad viewing angle. Another difficulty is due to variations of light sources. Since sensors are expensive in general, video-based approach is usually used in applications. In order to embed avatars into a virtual reality environment, 3D coordinate reconstruction methods need to be used [2, 5, 6].

We have implemented a system which takes a further step to integrate human actions with an interactive video player. Figure 1 shows our system setup. The player wears a black suit, with 16 track points (TPs) placed based on human skeleton. Two video cameras are located in front and on the side of the player to extract 2 sets of 2D coordinate for 3D coordinate reconstruction. The 16 track points are identified and mapped to the skeleton for motion understanding. Finally, the skeleton is mapped onto the interactive video, which contains special trigger points to detect human motions and react on the motions. Application of our system can be used in interactive video games, simulations, training, and others.



2. TRACKING AND INTERACTION

As illustrated in the flowchart in figure 2, the tracking and interaction process involves five steps (illustrated in different zones) of computation:

(1) Initialization and video camera synchronization
(2) Background and track points separation
(3) Track points identification
(4) Skeleton reconstruction on background video
(5) Motion understanding and interaction control

These steps will be discussed in the following sections.

2.1. Initialization and video camera synchronization

Background need to be separated from track points in order to identify track points. In the first step (marked as (1) in figure 2), an initial video background can be obtained from a video segment of 30 to 60 frames, where moving objects are removed and holes are inpainted. We use a fast image inpainting algorithm which relies on a diffusion kernel. In addition, since we use two video cameras for 3D coordinate reconstruction, the two cameras needs to be synchronized. Our strategy requires the player to stand steady for at least 2 seconds. A variation threshold for video changes is set to tell the steady situation. Thus, in the first step, the player need to move around for a few seconds, stand steady for 2 seconds, then our system start the second step (marked as (2)) to track the skeleton.

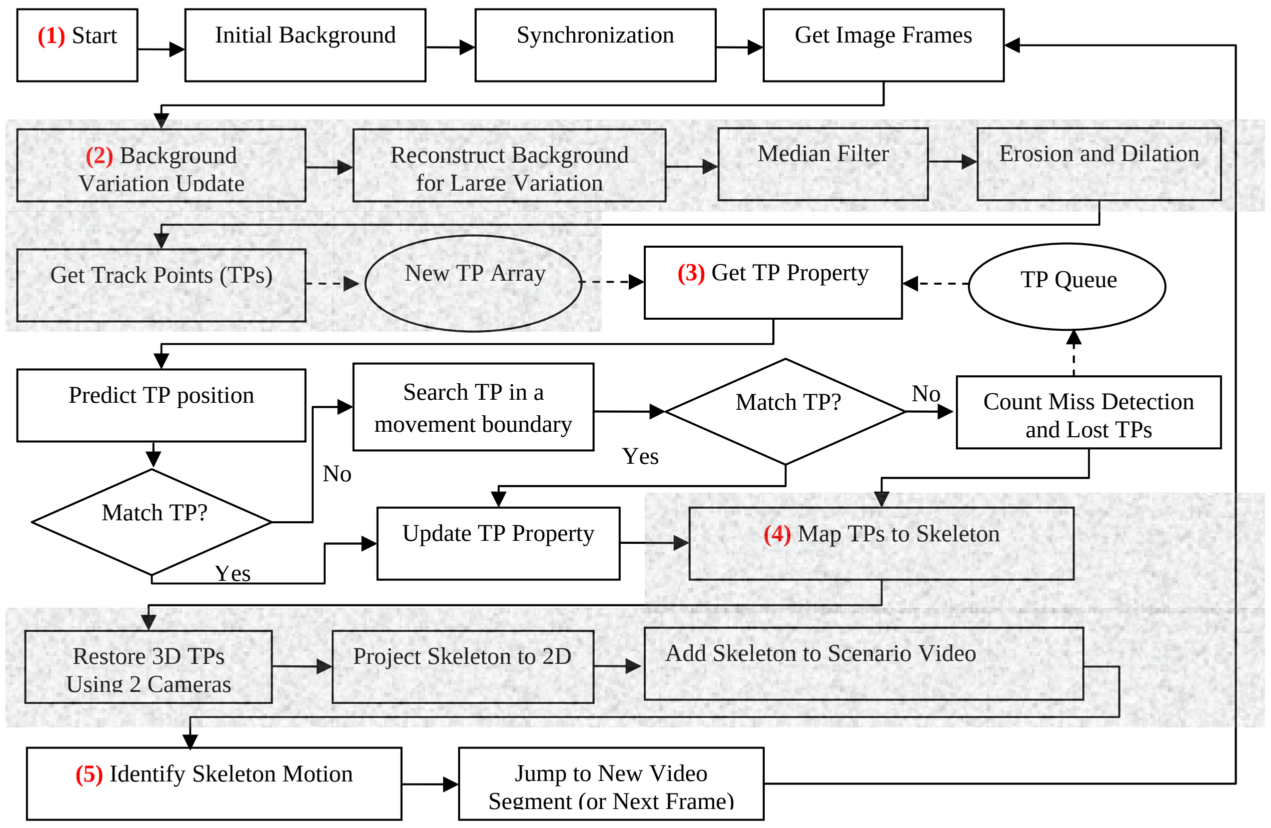

(1) Start
Initial Background
Synchronization
Get Image Frames
(2) Background Variation Update
Reconstruct Background for Large Variation
Median Filter
Erosion and Dilation
Get Track Points (TPs)
New TP Array
(3) Get TP Property
TP Queue
Predict TP position
Search TP in a movement boundary
Match TP?
No
Count Miss Detection and Lost TPs
Yes
Match TP?
No
Yes
Update TP Property
(4) Map TPs to Skeleton
Restore 3D TPs Using 2 Cameras
Project Skeleton to 2D
Add Skeleton to Scenario Video
(5) Identify Skeleton Motion
Jump to New Video Segment (or Next Frame)

Figure 2: Algorithm of Video Tracking and Interaction

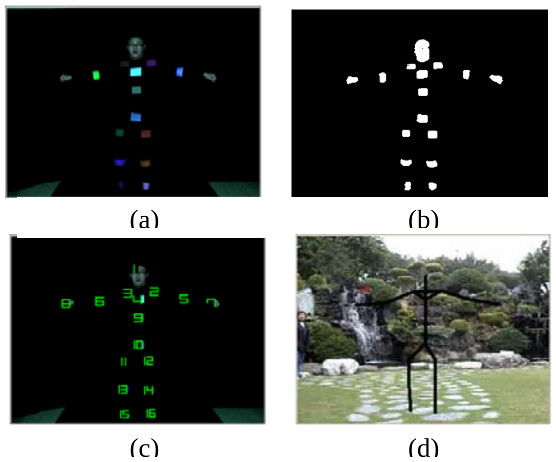


Figure 3: Tracking Process (a) Original Video (b) Track Points (c) Track Points Identification (d) Add Skeleton to Scenario Video

2.2. Background and track points separation

Since light sources to track points are quite difficult to stay stable, it is necessary to update background dynamically in order to obtain a better separation result of background and track points. Assuming that $B_k(x, y)$ and $B_{k+1}(x, y)$ are values of point (x, y) on the k_{th} and the $k+1_{th}$ frames, respectively. And, $I_k(x, y)$ is the value of point (x, y) on the coming k_{th} frame. We use

$$B_{k+1}(x,y) = \begin{cases} B_k(x,y) + \alpha[B_k(x,y) - I_k(x,y)]\ ,\ \alpha = 1 \\ \quad \text{if } I_k(x,y) \in \text{object point} \\ B_k(x,y) + \beta[B_k(x,y) - I_k(x,y)]\ ,\ \beta = 0.01 \\ \quad \text{if } I_k(x,y) \notin \text{object point} \end{cases}$$

This strategy works if small variation of light source occurs on background. However, if the variation is too large, the background should be reconstructed. After that, we use median filter to exclude isolated points. Also, we use two morphology operators (erosion and dilation) to perform the close operation. The close operation is able to eliminate small blocks. Figure 3 shows the original video in 3a and the track points in 3b after the second step.

2.3. Track points identification

The most challenge issue of multiple object tracking is to identify each and all objects. We maintain a track point array to store properties (location and color information) of all track points. In step (3), we use seed filling algorithm on the original video for basic color segmentation. Boundaries of track points are roughly computed, to obtain the center of each track point. Each record in the track point array contains a location of the center and the HSI color information. To precisely track each point, two issues need to be solved. Track points moves in a direction which is hard to predict; and, track points can be cloaked (occlusion). To solve the first problem, prediction and searching mechanisms are used. The second problem can be partially solved with multiple cameras. However, it is possible to solve the second problem using a unique color for each track point and searching in a range of movement. And, this is our approach. A sample predication mechanism which computes the average movement vector of each track point in the latest 30 frames is used. If the prediction of track point matches (within a small threshold of center location), the corresponding track point is updated in the array. Otherwise, the system searches the track points in a small boundary (eight directions to speed up computation). Track point properties are updated if there is a match. Otherwise, we count the track point as miss detection. An evaluation of miss detection is given in section 3. When a miss detection occurs, we keep the miss detected track point in a track point queue. In the next iteration, track points are searched against the queue with a higher priority. The search will focus on the color of track points. Our preliminary experience shows that, with the setup of our environment, the miss detection rate is tolerable. In figure 3c, track points are identified with numbers.

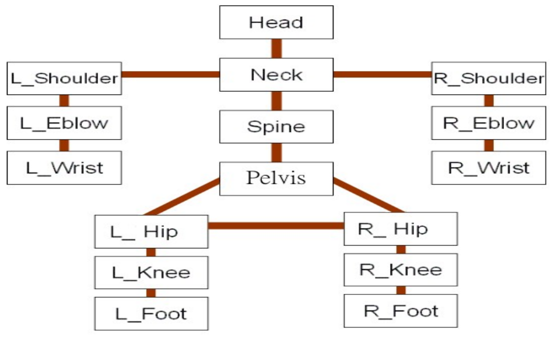


Figure 4: Skeleton with 16 Track Points

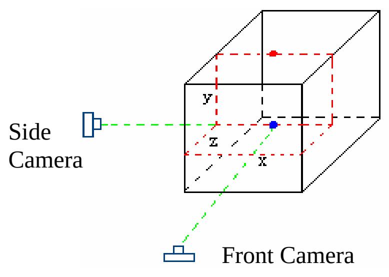


Figure 5: Coordinates and Front and Side Cameras

2.4. Skeleton reconstruction on background video

After the track points are identified, we need to map them to a skeleton which represents a human body. The mapping process is performed as soon as all TPs are identified in the first run. Since TPs will be tracked with unique IDs, remapping to skeleton is thus not necessary. The mapping strategy takes a simple heuristic rule: assuming that the play is not up side down or doing a handspring (i.e., flip). The mapping strategy follows:

1. Compute the fulcrum of 16 track points based on the initial posture (see figure 3), set fulcrum to be the Pelvis (see figure 4)
2. Follow the vertical line up to find Spine, Neck and Head. A horizontal threshold of a few pixels is used in case that the 4 track points of human body are not aligned vertically.
3. Split the rest 12 TPs in the skeleton to left and right, according to the 4 track points mapped in the above step. Following the Pelvis to find L_Hip, L_Knee, and L_Foot and following the Neck to find L_Shoulder, L_Eblow, and L_Wrist. Spatial relations of TPs are used as the heuristic.
4. Do the same for right hand side track points.

Only the track points captured by the front camera are used to identify the track points. However, to reconstruct 3D coordinates for all track points, it is necessary to use the side camera (along the x-axis to the left of the box in figure 5). To restore 3D coordinates, we use the following strategy:

1. For each track points on both cameras, find the differences of coordinates on y-axis.
2. Minimizing the differences to align TPs on y-axis using dynamic programming.
3. Take the x-coordinates from the front camera and the z-coordinates from the side camera.

After the 3D coordinates are computed, it is necessary to perform a projection of these coordinates to 2D space. One may argue that it is not necessary for 3D coordinate reconstruction and projection, and using 2D coordinates will be enough. However, to make the interaction realistic, 3D information is necessary. For instance, the player must fell that a punch is indeed located on an object in the video by moving forward his arm. In addition, if virtual reality avatar is used, 3D coordinate is required. After the projected 2D coordinates are obtained, we map the skeleton onto a scenario video. The scenario video follows MPEG-2, with an important extension allows a hyper jump among video segments. Hyper jump tags are embedded in the user defined data section of standard MPEG-2 video clips. The scenario video can be a pre-recorded video game or a training video for customers. For performance consideration, only a small section of video is stored in memory to speed up accessing time. Figure 3d shows the skeleton on a scenario video.

2.5. Motion understanding and interaction control

The last step is to control interaction. For our prototype system, we use a simple strategy to store skeleton motions. Each of the track points, L_Wrist, R_Wrist, L_Foot, and R_Foot, is recorded in the track point array with a motion vector, in addition to TP coordinate and color information. A motion vector can be one of the eight directions (i.e., East, West, South, North, Southeast, Southwest, Northeast, and Northwest) and the directions are extended to include z-movements. The motion vectors of the 4 track points are analyzed by an interactive controller. With special designed 'hot spot' in the interactive video, the hot spot is able to tell which of the 4 track points is engaged with a certain motion vector. Thus, a video hyper jump can be performed. Figure 1 shows our system with an interactive video made by real person. Note that, each section of the video has a small movement of the avatar to response to a specific trigger from a track point and its motion vector. Our preliminary system contains only hard coded video segments. An authoring tool is underdevelopment to enable other scenario video clips.

3. EXPERIMENT RESULTS AND ANALYSIS

We evaluate the experiment system from two perspectives: the precision rate of tracking and the performance of the system. High tracking precision avoids misunderstanding of motion hence the system reacts as the player expected. Human motion tracking is affected by three important issues: changing light sources, speed of motion, and track point occlusion. We tested 12 video clips to analyze rate of lost track points and rate of miss detection. They are summarized in table 1.

Table 1: Rates of Tract Point Detection

Video	No of Frames	Lost TP Rate	Miss Detection Rate
1	600	0.8 %	1.1 %
2	403	4.5 %	7.2 %
3	350	3.9 %	6.7 %
4	545	1.2 %	1.7 %
5	570	1.2 %	0.2 %
6	330	1.3 %	1.0 %
7	420	2.7 %	3.0 %
8	391	0.3 %	0.1 %
9	818	0.3 %	0.8 %
10	514	0.36 %	1.3 %
11	467	3.7 %	4.3 %
12	687	1.0 %	1.8 %
Average		1.77 %	2.43 %

Video 2 and 3 have a relatively unstable light source and higher occlusion. The miss detection rates are 7.2 and 6.7 percents. Video 7 and 11 have a stable light source. However, track points move faster. The miss detection rates are 3.0 and 4.3 percents. Video 8 and 9 have very low rates since light source is stable. Intuitively speaking, miss detection rates get higher due to changing light source the most. Occlusion and motion speed are secondary factors. Our average miss detection rate is 2.43%, which is tolerable.

The performance of our system is not as we have expected but is reasonable. We use 320 by 240 as the resolution of the two video cameras. The resolution of scenario video is also 320 by 240. We try to lower the resolution of the two cameras. But, precision of tracking is decreased. Step three of the algorithm (i.e., track point identification) is computational expensive, especially in searching miss matched track points. In step four (i.e, skeleton reconstruction on background video), restoration of 3D information takes time. However, we use a simplified approach to use the front camera for skeleton mapping. But, the side camera is used only in the identification of skeleton motion. Thus, project from 3D to 2D is simplified to speed up computation. The system is running on a Pentium 4 2.8GHz computer with 1.5 GB RAM. Preliminary experience shows that, player is able to interact with the video with limited motions to control video reactions.

4. CONCLUSIONS

It is interesting to combine video tracking technique with interactive video to develop a system for video-based games. Our approach involves five steps of schemes to realize the system on a Window based environment. Our experience shows that the approach is feasible on ordinary personal computers, instead of using special hardware devices such as Play Station 2. Our future work includes the design and implementation of an authoring system for the scenario video and motion identification. We hope software on PCs will allow players to design their own video games using PCs and ordinary cameras.

REFERENCES

[1] Eric R. Bachmann, Robert B. McGhee, Xiaoping Yun, and Michael J. Zyda "Inertial and magnetic posture tracking for inserting humans into networked virtual environments," Proceedings of the ACM symposium on Virtual reality software and technology, November 2001.
[2] D.E. Difranco, T.J. Cham, and J.M. Rehg. "Recovery of 3-D figure motion from 2-D correspondences," in CVPR, 2001.
[3] Z. Luo, Y. Zhuang, F. Liu, and Y. Pan, "Incomplete Motion Feature Tracking Algorithm in Video Sequences," in Proceedings of the IEEE 2002 International Conference on Image Processing, New York, USA, September 22-25, 2002.
[4] L. Sigal, S. Bhatia, S. Roth, M.J. Black, and M. Isard. "Tracking loose-limbed people," in CVPR, 2004.
[5] C. Sminchisescu and B. Triggs. "Kinematic jump processes for monocular 3D human tracking," In CVPR, 2003.
[6] KangKang Yin, and Dinesh K. Pai "Intuitive interfaces for animation: FootSee: an interactive animation system," in Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer animation, July 2003.