Proceedings of the 2006 Winter Simulation Conference
L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, eds.

A VR-BASED HYPER INTERACTION PLATFORM

Chun-Hong Huang
Hui-Huang Hsu
Timothy K. Shih

Dept. of Computer Science and Information Engineering
Tamkang University, Taipei County, 25137, R.O.C.

Rong-Chi Chang

Department of Information and Design
Asia University, Taichung, Taiwan, 41354, R.O.C.

ABSTRACT

We present a hyper-interaction platform which integrates several newly developed techniques, including motion classification, a motion reaction control mechanism and a high precision 3-D model of the International Space Station. The goal is to build an interaction platform for trainers to navigate in a virtual reality. The platform can be applied to customized mission training. The system integrated visualization environment, underlying 3-D model, and a human body tracking mechanism. The tracking system will be extended to 3-D coordination reconstruction and thus behaviors of users can be precisely identified. These navigation parameters are used in controlling navigation in the 3-D virtual environment. Another important issue of this system is to define a model for motion reaction. The definition of reaction can be applied to avatars or to the virtual environment. These integrated technologies can also be used in other virtual reality environments.

1 INTRODUCTION

Human motion tracking was developed in the past few years. The MARG sensors were used to develop a system which can embed skeleton into a virtual environment (Barchmann et al. 2001). Instead of using sensors, video-based tracking strategy was proposed (Luo et al. 2002, Sigal et al., 2004). Typically, the tracking involves matching objects in consecutive frames using pixels, points, lines, and blobs, based on their motion, shape, and other visual information (Leung et al. 1995, Shio et al. 1991, Polana et al. 1994). 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 commercial applications. We have developed a tracking system that can reconstruct 3-D coordinates of markers on a human skeleton. Since we got the serial 3-D coordinates, the motion types can be classified. We propose an approach for unit motion classification which is based on Neural Networks. As long as the type of unit motion is identified, these types would be the training parameters of Back-propagation Neural Networks. The Back-propagation Neural Networks should understand human motion pattern according to the combination of the unit motions. As long as the human motion is identified, a set of pre-defined reactions is applied to the motion on a 3-D environment, and they are used as the input parameter of a simulated model. The simulative specification can be developed based on particular missions, such as moving objects or performing tasks in the 3-D space. The particular platform that we use is a high precision model of Russia Space Station. We hope that, from motion detection, tracking to understanding, motion reaction can be considered as an interesting research topic.

We built a studio that has a surrounding background (270 degrees) with black color. Light sources are placed on the floor and on the ceiling to stabilize lighting parameters for video tracking. Two cameras are used for calibration of tracking points on a special designed suit. A back-projected screen and a projector are equipped such that the 3-D model can be rendered on the screen without interfering (i.e., shadowing) by the actors in the studio.

2 MOTION TRACKING

A tracking system that can reconstruct 3-D coordinates of markers on a human skeleton. The tracking process involves four steps of computation, which are discussed in turn in the remainder of the paper:

1. Initialization and video camera synchronization.
2. Background and track points separation.
3. Track points identification.
4. Skeleton reconstruction on background video.

2.1 Initialization and video camera synchronization

The background need to be separated from track points in order to identify track points. 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 to update background variation for tracking the skeleton.

2.2 Background and track points separation

Since light sources to track points are quite difficult to stay stable, it is necessary to update the background dynamically in order to obtain a better separation between 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 that, $I_k(x, y)$ is the value of point (x, y) on the coming k_{th} frame, we use:

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

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.

2.3 The identification of track points

The most challenging 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. To get TP property, 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. 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. 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 focuses on the color of track points. Our preliminary experience shows that, with the setup of our environment, the miss detection rate is tolerable.

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 segments of skeleton includes torso, upper arms and legs, and lower arms and legs. The mapping process is performed as soon as all TPs are identified in the first run. Since TPs are tracked with unique IDs, remapping to skeleton is thus not necessary. The mapping strategy takes a simple heuristic rule assuming that the player is not upside down or doing a handspring (i.e., flip):

1. Compute the fulcrum of 12 track points based on the initial posture, set fulcrum to be the pelvis.
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 8 TPs in the skeleton to left and right, according to the 4 track points mapped in the above step. Following the pelvis to find lower arms and legs, and following the neck to find upper arms and legs. Spatial relations of TPs are used as the heuristic.
4. Do the same for the right hand side track points.

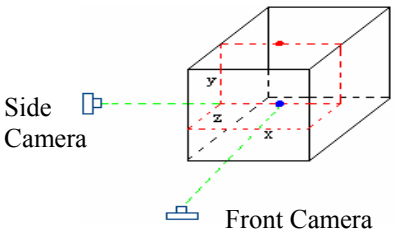


Figure 1: Coordinates and Front and Side Cameras

Only the track points that be 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 1). To restore 3D coordinates, we use the following strategy:

1. For each of the 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 the coordinates to 2D space. One may argue that it is not necessary for 3D coordinate reconstruction and projection, and using 2D coordinates is 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.

3 MOTION CLASSIFICATION

Next work is motion analysis, since we got 3-D coordinates of tracking points that were obtained from 2 or more cameras. In order to identify what motion the actor(s) perform, we use approximation strategy based on Dynamic Programming. A human motion specification can be represented as 9 motion vectors with regards to the 9 tracking points on a skeleton. For each tracking point, multiple vectors are used. We segment the motion of the human body into unit motions, and each unit motion has 200 frames. For example, if a human motion has 820 frames, then it would be divided into 5 unit motions. Suppose that each human unit motion data is a time successive data S consisting n elements, the serial data can be presented as follows:

$$S = S_1, S_2, S_3, \ldots, S_i, \ldots S_n$$

Here, S expresses a motion data of a certain segment of a human body. The difference between S_i and S_{i+1} is considered as the value of vector v'_i. Since human motion data is 3D time serial data, each element S_i can be express as space coordinates $Si=(xi, yi, zi)$. Then the vector v'_i is given as follows:

$$\overrightarrow{v_i} = ((x_{i+1} - x_i), (y_{i+1} - y_i), (z_{i+1} - z_i)) = |S_i - S_{i+1}|$$

The vectors of S can be expressed as follows:

$$\overrightarrow{V_S} = (\overrightarrow{v_{S1}}, \overrightarrow{v_{S2}}, \ldots \overrightarrow{v_{Si}}, \overrightarrow{v_{S(i+1)}}, \ldots \overrightarrow{v_{S(n-1)}})$$

where $\overrightarrow{V_S}$ is the set of vectors, all the feature trajectories with each segment of human body can be represented. Each motion vector is represented by a pair of angles as shown as Figure 2. Angle α is in between the projection of motion vector $\overrightarrow{v_{S1}}$ on the XZ-plane and the X-axis. Angle β is between the motion vector and its projection. An angle has a value between 0° and 360° (inclusive).

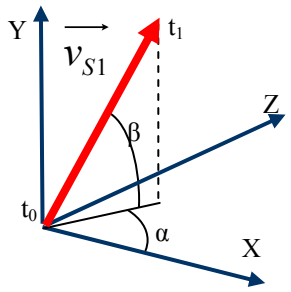


Figure 2: Definition of Motion Vectors

Instead of using precise coordinates in the motion identification computation, we use an approximation approach which only computes the rough location of each motion vector. For each angle (α or β), we define an Angle Code (AC) as:

$$AC = floor(Angle / n)$$

where *Angle* can be α or β and *floor* is the floor function. The parameter n can be set to *45*. As such, Angle Codes are between 0 and 7. It is possible to set $n = 22.5$. Thus, Angle Codes are values between 0 and 15. In our tracking procedure, since the length of a motion vector is short in general, coordinates of tracking points may not reflect the actions precisely. Using an approximation approach, on the other hand, can reduce the computational cost in our work. For the i^{th} motion vector, α and β are represented by a pair of Angle Codes $C_i = (AC_{\alpha j}, AC_{\beta j})$. We can classify the motion types according the serial Angle Codes. We should reconstruct the feature space which stores the serial data of Angle Codes be parsed and translated from the different kinds of standard motions. Assume that each unit motion has its own serial Angle Codes $C_j = (AC_{\alpha j}, AC_{\beta j})$, which stores in the feature space, thus we can compare the serial Angle Codes *Csj* of the new unit motion with the *Csi* via the first level classifier of neural network. After the first level classifier, we can find the similar Unit Motion Types(UMT) for each unit motion as in the following equation:

$$UMT_i = f_1(\text{Angle Codes } Cs),\ (1 \le i \le m)$$

Since the similar types of each unit motions can be recognized via f_1, a motion representation can be *Motion* = { (UMT_1), (UMT_2), ..., (UMT_m) }. m is the amount of unit motions. The representation of motions can be used in the second level of the neural network model. A trained neural network can be used as the mapping function from a motion representation to a *motion classification code* (MCC)

$$MCC_j = f_2(Motion_x)$$

where j is between 1 and the maximum number of motion code to be classified, and x is assumed to be the maximum number of possible motions. The mapping, f_2, decides the MCC via the second level classifier. We propose a three-level classification mechanism using a neural network.

As soon as a motion classification code, MCC_j, is classified by the second level classification mechanism, a series of MCCs may represent a continuous motion. However, similar continuous motions may have different durations. Therefore, a *motion window* is defined as an ordered list of motion control codes, with a variable number of MCCs. Figure 3 illustrates the two-level architecture. Each motion window starts at a motion control code. Thus, the number of motions windows (i.e., n) is equal to the number of motions in an action. A third level classifier, f_3, takes as input a motion window, and returns a motion signature (i.e., MS_k) as in the following equation:

$$MS_k = f_3(MW_k),\ \text{where } 1 \le k \le n$$

Motion signatures are be used to trigger reactions, which we discuss in the next section.

4 MOTION REACTION AND CONTROL

Motion reaction can be defined in two types. The first is changing the camera view of a virtual reality browser, by rendering the 3-D model (represented in VRML or the like) from different viewing angles. This type of reaction depends on the movement of a trainer. Thus, the definition of motions should deal with the movement of a trainer in the studio. Yet, the movement specification should consider the limited space in the studio. Special posture of the trainer is recognized as a particular movement. The second type of reaction includes changing objects in the 3-D model, such as moving an object or assembling an object in the space station. In addition, the second type of reaction may include the reaction of other avatars in the 3-D scene, if the system is implemented on a shared Web-based VR environment. From motion detection, motion tracking, motion understanding, to motion reaction, the pioneer perspective of this research project is to investigate how a virtual environment (including its avatars) should react with respect to a particular motion identified by the system.

For the two types of reactions controls (i.e., changing camera view and changing avatars), we implement them in the Virtools environment as the following:

Control Camera View (MRA_{Camera})

1. Create a target camera
2. Set camera position and viewing angle
3. Attach camera position to an avatar

Control Avatars (MRA_{Avatar})

1. Create a database record by importing the avatar and 3-D model resources.
2. Load the avatar and 3-D model into the Virtools environment.
3. Add animations to the avatar.
4. Initiate keyboard controller (using motion control) and connect character control to animations

After motion signatures are identified by the back-propagation neural network, each motion signature is associated with a list of motion reaction animations (i.e., MRA_{Camera} and MRA_{Avatar}). The animations rely on the above two types of controls in the Virtools. In addition, it may be necessary to have additional object transformations (or to add additional objects) in the 3-D scene, if the mission specifications have such a requirement.

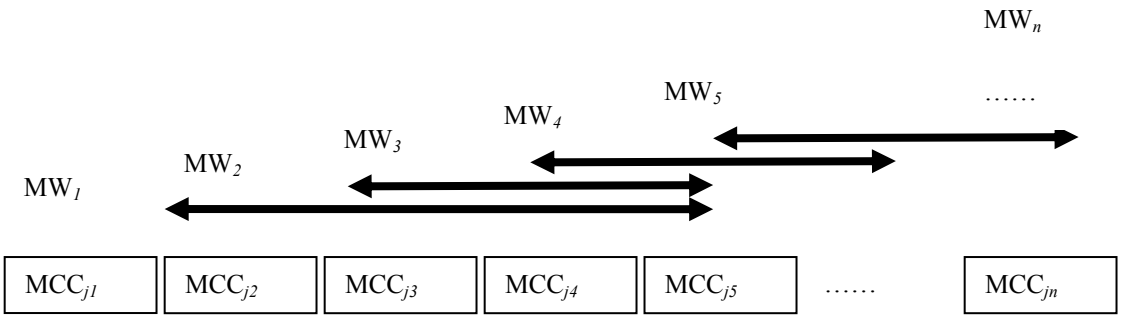


Figure 3: Motion Windows of Variable Time Slots

4.1 Reaction Resolution

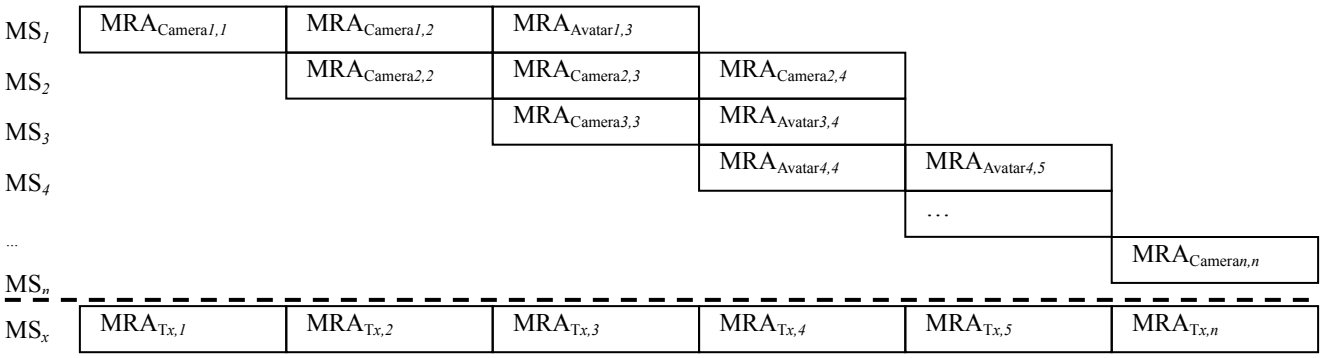
To identify what reactions should be applied to each motion signature, a resolution mechanism is used. Since each motion signature contains a variable number of motion control codes, it is possible to have two or more signatures that share a list of same control codes. In addition, since each motion signature is associated with a list of motion reaction animations, different signatures should share reaction animations in overlapped time slots. Thus, a mechanism to resolve the difference of reaction animations with respect to different motion signatures is necessary. Figure 4 illustrates a list of motion reaction animations for each motion signature. Each MS_x at time slot x has a list of motion reaction animations $MRA_{T\,x,\,t}$. Assuming that a motion reaction animation, $MRA_{T\,x,\,t}$, has a control type T, where T represents Avatar or Camera. The two subscripts, x and t, represents the indices of motion signatures and the index of time slots. For the two types of controls, different strategies are used to deduce the reactions.

The first equation in Figure 4 indicates that for the MRA with type Camera, the deduced motion reaction animations (DMRA) takes the average camera movements. That is, a DMRA refers to all camera motions up to a current time slot t (i.e., $x' \leq x \leq t$), by taking the average camera coordinates in a 3D space. This strategy allows a continuous motion to consider snapshots taken in current time slot, as well as a few slots before. The deduced motion reaction animations results in a smooth camera movement. The second equation is for the Avatar type of animations. It is necessary to consider multiple avatars which response to a motion signature. For new avatar, a new animation associated with the avatar is started. However, the animation of avatars started in previous time slots continues. The computation of DMRA is dynamic. That is, at each time slot t, due to each motion signature, a list of reaction behavior is sent to the reaction controller (to be discussed). Thus, the dynamic reaction results in a smooth motion.

4.2 Rendezvous Communication Control

A rendezvous control is a mechanism which allows two processors to meet at a certain time point, where the two processors complete certain tasks by their own before the rendezvous happen. We use two servers. The tracking server and the VR rendering server are synchronized in real-time. Figure 5 illustrates different entities of the rendezvous communication control.

As soon as a motion signature (MS) is identified, the MS is kept in a *motion signature queue*, which is controlled by the tracking server. An action controller in the tracking server synchronizes with a reaction controller in the VR rendering server, by the rendezvous communication control mechanism. The mechanism decides where the two controllers will meet.



$$DMRA_{Camera\,t} = \left(\sum_{x' \leq x \leq t} MRA_{Camera\,x,t}\right) / (t - x' + 1)$$

$$DMRA_{Avatar\,t} = \begin{cases} MRA_{Avatar\,x'}, & \text{if Avatar } x' \text{ is not complete} \\ MRA_{Avatar\,t}, & \text{if Avatar } t \text{ is new} \end{cases}$$

Figure 4: Resolution of Motion Reaction Animations

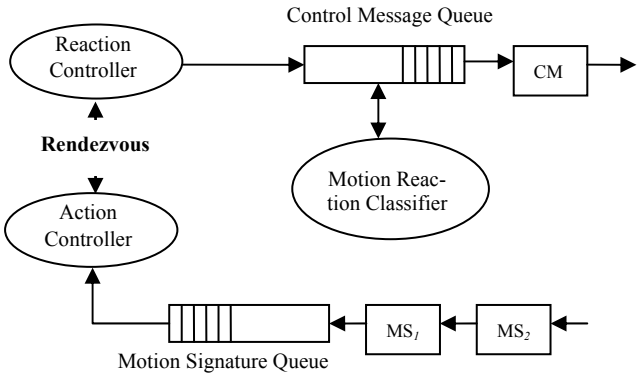


Figure 5: Rendezvous Communication Control

As soon as the controllers meet, the action controller sends a motion message to the reaction controller, which interacts with a *motion reaction classifier* to identify how to trigger avatars in the 3D VR environment via control messages (CMs). The activation of avatars and rendering of 3D scenes consume heavy computation. Thus, the reaction controller needs to wait till the computation is complete in order to move to the next step. Similarly, the action controller needs to wait till the next motion signature is identified. The two controllers rendezvous. Thus, reactions in the 3D environment are synchronized with the motions by the actor.

5 MISSION SPECIFICATION AND CONTROL

A mission specification is defined as a series of actions that a trainer needs to accomplish by using the system. The specification may include some check points, which is used as the base of an assessment model. Mission assessment can be maintained in a database for the analysis of individual training performance. The specification and simulation model are designed by domain experts. A mission specification can be defined as a list of events. Each event is associated with a location in the 3D space and a set of actions:

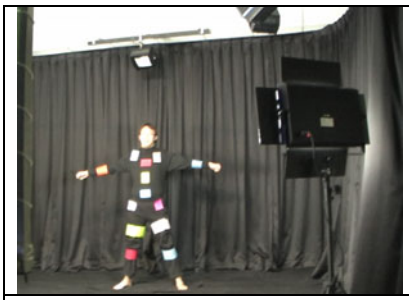
$$Mission_m = (Event_1, Event_2, Event_3, \ldots, Event_x) = ((Location_1, \{A_{11}, A_{12}, \ldots, A_{1i}\}), (Location_2, \{A_{21}, A_{22}, \ldots, A_{2j}\}), \ldots, (Location_x, \{A_{x1}, A_{x2}, \ldots, A_{xk}\})$$

The order of actions within a mission can be defined in any topology. Thus, each action is associated with a predecessor and a successor, except the last and the first action. Actions can be performed in sequential or parallel, as suggested by the topology, which is designed by using a graphical user interface. The assessment of a mission includes an evaluation function which takes two parameters as input: an event in the mission and an event performed by the trainer. Comparison of coordinates can be asserted into the evaluation function as a check point (critical mission-based) or quantitative outcome (overall performance). Comparison of actions in each location can be accomplished by two factors: a detailed movement of the trainer, and the reaction from the avatar. Mission specifications as well as the assessments for each trainer are maintained in a mission database. An analytical model based on the outcome of assessment can be used by a high ranking officer to evaluate individual trainers.

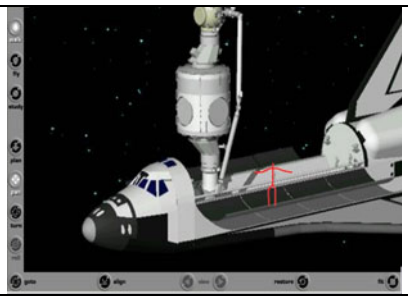
We implement a video studio and its software systems. The software systems run on two computers respectively in a LAN. Figure 6 shows the studio, the interactive module, and example of the 3D model of Russia/International Space Station.

6 CONCLUSION

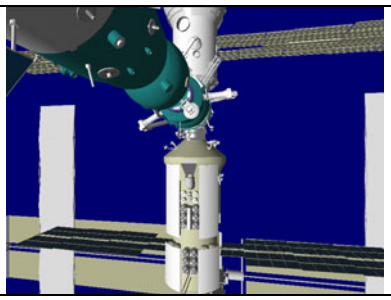
This paper investigates the fundamental techniques and application of building a 3-D virtual environment, as an interaction platform. It proposes the platform and its underlying technology for cosmonaut mission training in the Virtual International Space Station. The system shows research collaboration results that we have accomplished in the past few years.



(a) The Video Studio with an Actor



(b) Interaction with a 3D Stick Figure



(c) Snapshot of Space Station

Figure 6: Implementation of the System and Studio

The system use existing tracking techniques, with a new interesting issue in motion reaction control. The software is implemented on two computers with a rendezvous control mechanism to adjust speed of reactions. Our experience encourages us to further enhance the collaboration to include the interior model of the space station. We hope that, motion reaction will be an important issue in addition to detection, tracking, and understanding in video technology.

ACKNOWLEDGMENTS

This work was supported in part by the National Science Council of the Republic of China under contract NSC 95-2221-E-468-007.

REFERENCES

Barchmann, E. R., R. B. McGhee, Xi. Yun, and M. J. Zyda. 2001. Inertial and magnetic posture tracking for inserting humans into networked virtual environments. In *Proceedings of the ACM Symposium on Virtual reality software and technology*. pp. 9-16.

Luo, Z., Y. Zhuang, F. Liu, and Y. Pan. 2002. Incomplete Motion Feature Tracking Algorithm in Video Sequences. In *Proceedings of the IEEE 2002 International Conference on Image Processing*. pp. III/617-III/620.

Sigal, L., S. Bhatia, S. Roth, M. J. Black, and M. Isard. 2004. Tracking loose-limbed people. In *Proceedings of the 2004 Computer Society Conference on Computer Vision and Pattern Recognition*. pp. I421-I428.

Leung, M. K., and Y. H. Yang. 1995. First sight: a human body outline labeling system. *IEEE Trans. Pattern Anal. Mach. Intell*. 17 (4), pp. 359–377.

Shio, A., and J. Sklansky. 1991. Segmentation of people in motion. *Proceedings of the IEEE Workshop on Visual Motion*. pp. 325–332.

Polana, R., and R. Nelson. 1994. Low level recognition of human motion. In *Proceedings of the IEEE CS Workshop on Motion of Non-Rigid and Articulated Objects*. pp.77–82.