The Design and Implementation of Virtual Studio*

ChangWhan Sul, Kwang-Yoen Wohn
(Dept. of Computer Science, KAIST)

Abstract: A virtual reality system using video image is designed and implemented. A participant having $2\frac{1}{2}$DOF can interact with the computer-generated virtual object using her/his full body posture and gesture in the 3D virtual environment. The system extracts the necessary participant-related information by video-based sensing, and simulates the realistic interaction such as collision detection in the virtual environment. The resulting scene obtained by compositing video image of the participant and virtual environment is updated in near real time.

1 INTRODUCTION

1.1 *Reflexive VR*

Using live video images of participants in a VR system is not a new idea. Since the remarkable work of Krueger[3], couple of similar applications have been developed in both entertainment [8], [9] and academic domains[4]. The advancement of image processing and computer graphics technologies enabled these ***Reflexive VR systems*** to shift to more realistic 3 dimensional virtual world and richer interactions. In reflexive VR systems, one or more video cameras replace position / orientation sensors, mouse or keyboard, while a large projection screen replaces HMD's and stereo glasses. A typical setting is depicted in Fig.1. The

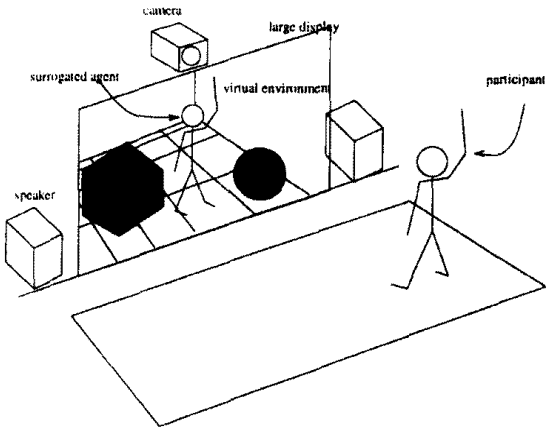

Fig. 1. Configuration for Reflexive VR system

configuration has numbers of advantages over traditional (immersive) VR such as unencumbered interface, full-body interaction and no motion sickness while having disadvantages such as coarse resolution and limited interaction. Due to the characteristics of reflexive VR, application areas such as LBE and computer-generated studios in broadcasting are considered suitable and promising.

*This work was supported partially by Center for AI Research, KAIST

1.2 *Related works*

In Videoplace[3],the first attempt to realize the idea of reflexive VR, Krueger demonstrated the potentials of reflexive VR as a new communication media. The participants were self-represented by their silhouettes, and the underlying virtual environment(VE) was a flat, two dimensional space inhabited by simple flat objects. Mandala[9] from Vivid was an extension of Videoplace in that it represented participants by their video image, while still having flat objects. Major restrictions of these two ancestors of today's reflexive VR was that they only supported two dimensional virtual space and objects. Therefore the interaction between participants and virtual worlds was inherently bound to their 2D-nature.

Recently however, emerging technologies such as IVE(Interactive Video Environment) from MIT[4], [2] and 'Virtual Set'[1] from SGI are extending the virtual space, objects and interactions within the space to near three dimensional ones, broadening application area of reflexive VR. Video images of the participants captured by video camera are composited to images of 3D VE inhabited by 3D objects and agents, and participants are able to interact within the VE using their full body motions and gestures.

With rapidly growing image processing and generation power of computers, reflexive VR technology is opening new opportunities not only to computer engineers, but game designers, broadcasters and film makers.

By combining basic computer vision techniques and virtual reality system we developed[5], we have designed and implemented a prototype of reflexive VR system which is based on analysis of video image of the participant. In the following, the structure of the system, the method used for image analysis and experimental results will be described.

2 DESIGN OF REFLEXIVE VR SYSTEM

In order to interact with participants, any VR system ought to have subsystems for input processing, VE simulation and output generation. Input processing subsystem extracts the intention of the participants by analyzing raw sensor data. VE simulation subsystem simulates the interactions between participants and virtual objects, between virtual objects and implicit rules of VE such as gravity, thereby changes descriptions of VE properly. Output gen-

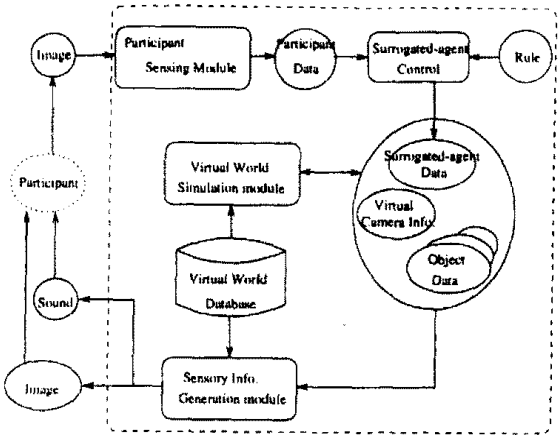

Fig. 2. Flow diagram of the system

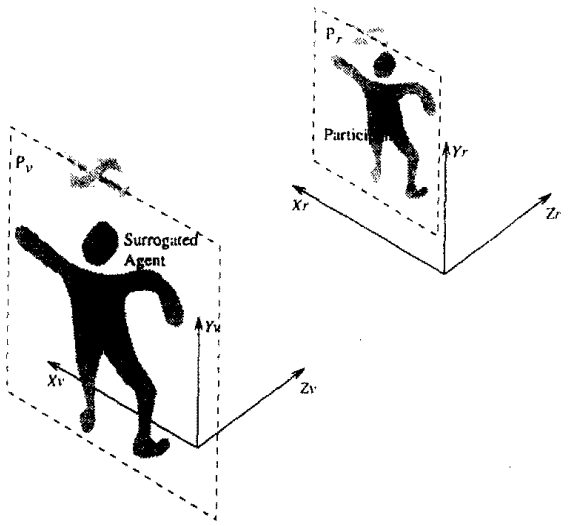

Fig. 3. $2\frac{1}{2}$D model of participant and surrogated agent

eration subsystem renders the status of VE at each frame, generating sensory signals such as images and sounds.

Reflexive VR system differs from its immersive counterpart at input and output subsystems. In the input subsystem, video camera replaces position/orientation trackers, data gloves, wands, joysticks, mouse or keyboard. Most of information on the intention of participant is extracted from the video image. In the output subsystem, in addition to rendering the scene in VE by computer graphics techniques, compositing video image of the participants must be taken into consideration.

Overall flow of the system is shown in Fig.2. In the current design, we assume a single participant. The video image of the participant is fed into the participant sensing module to extract the relevant information from the participant's physical posture and gesture. This information is then transformed into the data on the surrogated agent which represents the participant in virtual world. Along with the surrogated agent data, data of virtual worlds describing virtual objects, camera, lights and behaviors are processed and updated by the virtual world simulation module. The updated data is then transformed into proper visual and auditory information, giving feedback to participant. Each modules will be described later in detail.

2.1 $2\frac{1}{2}D$ Virtual space & interaction model

The necessary information for interactions between surrogated agent and VE comes from two different sources. The information for the surrogated agent can be obtained by analyzing video image. The information for VE is supplied from the virtual world database, application scenario, etc. The two types of information, one being 2D pixels and the other being 3D digital data, are not compatible to each other. Therefore we map the video image into 3D-compatible data by our simple $2\frac{1}{2}$D model. To simplify the computation, we assume following conditions:

- The camera is fixed.
- The participant maintains the pose facing the camera at right angle.
- At any time, at least one body part of the participant touches the floor.
- The participant does not move too quickly.

In our model, the participant is represented by a bitmap on a plane P_r moving in real world coordinate as participant moves. The direction of P_r is fixed, to the direction of the participant moving back and forth about the camera, i.e., P_r always has the form $z_r = depth(t)$. (see Fig.3). Therefore the dimension of the virtual space inhabited by our participant may be regarded as lying at somewhere between 2 and 3. Surrogated agent, self of the participant in VE, is also represented by a bitmap on the plane P_v moving in VE. The movement of P_r and P_v are tightly related.

All objects in VE but the surrogated agent preserve their own dimension, namely the full 3D. The resulting VE is 3D space occupied by 3D objects and $2\frac{1}{2}$D surrogated agent.

Among many levels of interaction,[6] physical interaction level which includes navigation and collision is adopted in our model.

$2\frac{1}{2}$D navigation is obvious from the model. We define collision between $2\frac{1}{2}$D surrogated agent and 3D object as the intersection between bitmap of the surrogated agent and the bounding box of the object as in Fig.4.

2.2 Participant Sensing

The participant sensing module analyzes video image of the participant, updates participant model so that he can interact in VE. As the first step, the region that belongs to the background is removed from the input image. Separation of the foreground from background pixels can be performed by using either special hardware or by software. The first option, using hardware, is called chroma keying and used widely in broadcasting and film industry. The other option varies from simple method based on frame difference to the relatively complicated statistical approach[10]. Despite the cost of setting homogeneous background and the

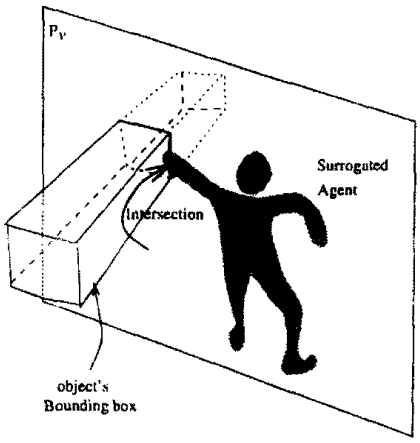

Fig. 4. $2\frac{1}{2}$D collision

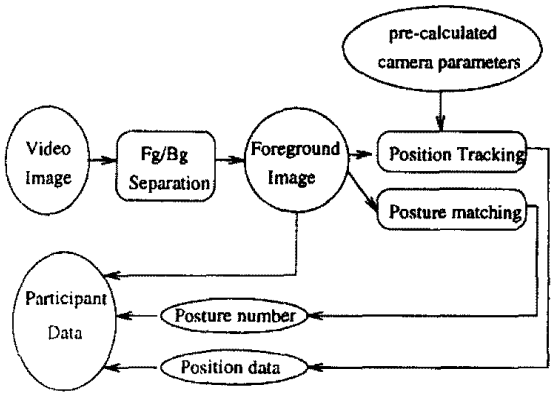

Fig. 5. Flow diagram of participant sensing

need for carefully controlled illumination, the first option is preferred to save the computation.

From the 2D bounding rectangle which encloses the foreground pixels, a bounding rectangle in $2\frac{1}{2}$space in real world is obtained by exploiting the assumptions we have made in section 2.1. Mapping from the 2D image plane to the 3D VE is accomplished by using the pre-calculated camera parameters[7]. More detailed discussion is in order.

Without loosing generality, we set $y_r = 0$ for the floor and by the third assumption in 2.1, the lowest part of the bounding rectangle always satisfies $y_r = 0$. We can determine a unique point among many points satisfying inverse camera projection equation and P_r is determined from that point. Let $ABCD$ be the 2D bounding rectangle for foreground pixels and $A'B'C'D'$ be the $2\frac{1}{2}$D bounding rectangle for participant as in Fig.6. Camera projection is expressed by

$$Prj(x_r, y_r, z_r; T, R, f, \kappa_1, \kappa_2, C_x, C_y) = (x_i, y_i)$$

where $T, R, f, \kappa_1, \kappa_2, C_x, C_y$ are the camera parameters as defined in [7]. Since Prj does not have inverse, we define set Prj^{-1} as

$$Prj^{-1}(x_i, y_i) = \{(x_r, y_r, z_r) \mid Prj(x_r, y_r, z_r) = (x_i, y_i)\}$$

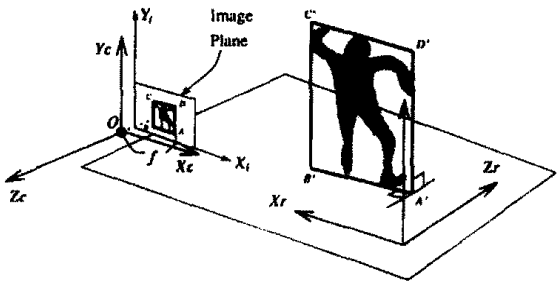

Fig. 6. Position determination

Then, the coordinate of point A' is given by

$$(x_{rA'}, y_{rA'}, z_{rA'}) = Prj^{-1}(x_{iA}, y_{iA}) \cap \{(x_r, y_r, z_r) \mid y_r = 0\}$$

From the z_r value satisfying above, we get the plane P_r. Other points B', C', D' are obtained from intersection of Prj^{-1} and P_r. Position of the participant is defined by mid-point of A' and B'.

Along with the position, the posture of the participant is used for interaction. A posture is defined by a binary template. Postures are tested by matching sub-sampled foreground image with the template. If there are more than one template whose score is higher than threshold value, one with the highest matching score is selected as the current posture. For stability, a posture must be selected in n subsequent frames to be activated (n is programmable).

Participant data, output of the participant sensing module, consists of the foreground bitmap, $2\frac{1}{2}$D bounding rectangle in real world representing global position and posture number.

2.3 Surrogated agent control

Surrogated agent is controlled by the real world participant. By adding one layer of mapping from participant to surrogated agent, many interesting effect such as mirroring, distortion or exaggeration of movement of participant is possible, thereby alleviates restrictions in participant movement which is governed by the physical laws in real world. Surrogated agent control consists of two parts: appearance control and position control. For the appearance control, the way the surrogated agent looks in VE is controlled. Local parameters such as color and transparency can be changed whenever needed and a global disguise function will turn the bitmap of the participant to the bitmap of another agent. As for the position control, the displacement of the participant $\Delta\vec{P} = (\Delta x_r, \Delta y_r, \Delta z_r)$ is mapped to the displacement of the surrogated agent $\Delta\vec{P'} = (\Delta x_v, \Delta y_v, \Delta z_v)$ by

$$\Delta\vec{P'} = (f_x(\Delta x_r), f_y(\Delta y_r), f_z(\Delta z_r)) + \Delta\vec{Q},$$

where $\Delta\vec{Q}$ is the programmable displacement that does not depend on the displacement in real world and f_x, f_y, f_z are the displacement mapping functions. In most cases, $f_x(0) = f_y(0) = f_z(0) = 0$, f_x and f_y are non-decreasing and f_z is non-increasing for the mirroring effect.

2.4 *VE simulation*

We update the status of VE in a frame-based fashion. At each frame, input processing module, VE simulation module and output generation module are executed in sequence. In simulation module, behavior description functions of all objects in VE are called one by one. The behavior description function is a small subroutine associated to one or more active objects. By these functions, the status of the current VE including informations on objects, lights, virtual camera, surrogated agent can be referred to and changed. These functions will respond to the participant by detecting the input event such as collision, position change and posture activation, updating the internal status of the objects under consideration.

2.5 *Output generation*

As the result of VE simulation, the status of VE is updated every frame, and the resulting VE should be rendered into visual and auditory data, giving participants sensory feedback. For visual processing, the traditional keying method of compositing two scenes is not applicable since we're dealing with 3D objects, not two 2D images. In our system, the image of the surrogated agent, which is the bitmap of the participant in most cases, is composited with the image of VE using an algorithm analogous to Z-buffer rendering. At first, the scene of VE is rendered with a conventional rendering method. Z-buffering, either implemented in hardware or software, is a commonly used technique for 3D computer graphics. When rendering of all visible objects in VE is done, the Z-buffer contains the distance to closest object from viewpoint. Using this, the bitmap with Z value determined by the plane P_v is rendered pixel-wisely. The result will be an image of the surrogated agent in VE, properly occluded by objects closer to viewpoint than the agent, and properly occluding objects farther than the agent.

For sound rendering, a MIDI file or sampled sound is played when sound event is triggered by the application program.

3 Implementation

The system is implemented on top of VRAT(Virtual Reality Authoring Toolkit) which is a set of library functions and tools for authoring and execution of multiple user VR application developed in KAIST[5]. A VRAT application can be executed basically on any platform that is capable of running X-Windows system, such as Sun workstations, Linux-running PC's. Besides, it can be executed on SGI to achieve faster and higher quality rendering. Users of VRAT write their application in C using standard API, link with VRAT library to get an executable. VRAT library consist of device drivers, renderers, main simulation loop, network communications, and various query & set functions.

Currently, the system is running only on Linux-based PC platform mainly for availability of low cost frame grabbing hardware. Additional modules were added to VRAT for reflexive VR application. They include $2\frac{1}{2}$D collision detection and participant sensing from video image such as position tracking, posture recognition, etc. The software structure of VRAT is shown is Fig.7 and the shaded part is newly added for the reflexive VR.

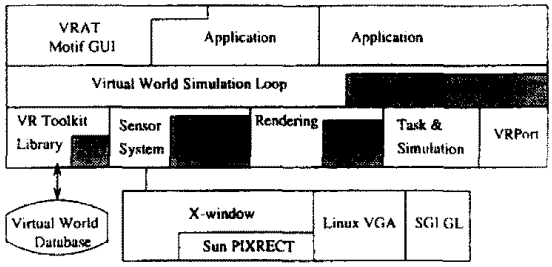

Fig. 7. Software structure of VRAT

Fig. 8. Experiment settings

Application programmers are provided with API's for reflexive VR: they can query the current position and size of the bounding rectangle, move or resize the bounding rectangle to explicitly control surrogated agent, register functions for position control(f_x, f_y, f_z mentioned in section 2.3), and check intersection between the surrogated agent and virtual objects. They can define the posture template of their own and register a callback functions to be called when the posture is activated.

A simple chroma keying box, blue screen and Targa+ frame grabber are used for participant sensing module(Fig.8). Camera parameters are obtained by Tsai's co-planar calibration method[7]. The average performance is about 5 frames per second, for small number of polygons in VE. Examples of reflexive VR application is shown in Fig.9.

4 Summary

We have developed a prototype reflexive VR on top of the existing VR toolkit. Live video image of the participant is analyzed and used for the interaction in virtual environment. Simple techniques from computer vision and pattern recognition are combined to produce basically re-

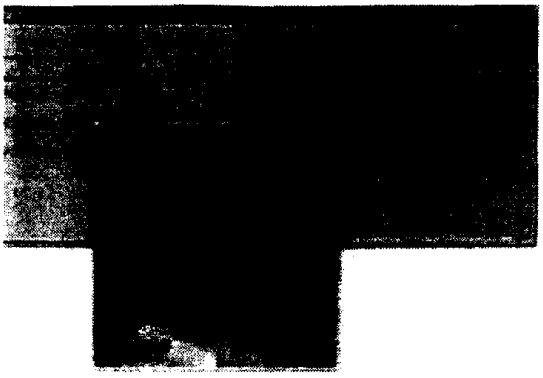

Fig. 9. Reflexive VR application

quired functionalities.

Our future works includes: more intelligent gesture and posture analysis, extension to the dynamic environment in which the camera moves, improvement of video processing both in speed and quality.

REFERENCES

[1] *http://www.studio.sgi.com/ Features/VirtualSets/intro.html.*

[2] Trevor Darrell, Pattie Maes, Bruce Blumberg, and Alex P. Pentland, *A Novel Environment for SItuated Vision and Behavior*, Tech. Report Perceptual Computing TR-261, MIT Media Laboratory, 1994.

[3] Myron W. Krueger, *Artificial Reality II*, Addison-Wesley, 1990.

[4] Pattie Maes, *ALIVE: An Artificial Life Interactive Video Environments*, Computer Graphics Visual Processing, ACM Press, 1993.

[5] Unjae Sung, Sonou Lee, and Kwangyoen Wohn, *The Design and Implementation of Virtual Reality Authoring Tool*, Proceedings of the 21st KISS Fall Conference, KISS, 1994.

[6] Unjae Sung and Kwangyoen Wohn, *A Layered Interaction Model for the Virtual Environment*, Tech. Report CAIR-TR-94-58, CAIR,KAIST, 1994.

[7] Roger Y. Tsai, *An Efficient and Accurate Camera Calibration Technique for 3D Machine Vision*, CVPR, 1986.

[8] Steve Warme, *Mandala sport simulators*, Virtual Reality World 2 (1994), no. 5, 44–49.

[9] Steve Warme, *The Mandala Virtual World System, or, virtual reality-no strings attached*, Virtual Reality World 2 (1994), no. 2, 65–71.

[10] Christopher Wren, Ali Azarbayejani, Trevor Darrell, and Alex Pentland, *Pfinder: Real-Time Tracking of the Human Body*, Tech. Report Perceptual Computing TR-353, MIT Media Laboratory, 1994.