Remote Augmented Reality for Multiple Players Over Network

Daniel C. M. Leung, Pak Shing Au, Irwin King, and Edward H. H. Yau
Department of Computer Science and Engineering
The Chinese University of Hong Kong
cmleung4@cse.cuhk.edu.hk, psau4@cse.cuhk.edu.hk,
king@cse.cuhk.edu.hk, edyau@cse.cuhk.edu.hk

ABSTRACT

Augmented Reality (AR) in multimedia gaming is a dynamic and exciting field of research. One of the challenges is to have multiple users interacting in the networked augmented reality environment. In recent years, camera-based body motion capturing console games, which capture the player's body movement to control the objects rendered by the computer, have been developed. In this paper, we present a system called *Tele-Table* which allows multiple users to interact with each other through the network using real objects. Each player's set-up consists of an overhead mounted camera perceiving real objects, a plasma TV placed horizontally to act as a game table, and a computer. One of the most challenging tasks is data synchronization between two terminals. There are mainly three synchronization tasks we need to handle. They are temporal synchronization, spatial synchronization, and game states synchronization between the two terminals. We present these difficulties and present solutions and discussion to solve them. The design goals of the Tele-Table are real-time and realistic user experience. The features of the system are changeable background, using real objects, and real-time networked augmented reality gaming between two terminals. We outline the system, present the detailed description, and discuss current achievements.

Categories and Subject Descriptors: H.5.1 [Information Interfaces and Presentation]: Multimedia Information Systems—*Artificial, augmented, and virtual realities*; C.2.4 [Computer-communication Networks]: Distributed Systems—*Distributed applications*;

General Terms: Performance, Design, Experimentation, Human Factors

Keywords: Networked Augmented Reality, Computer Entertainment, Virtual Gaming

ACE'07, June 13–15, 2007, Salzburg, Austria.

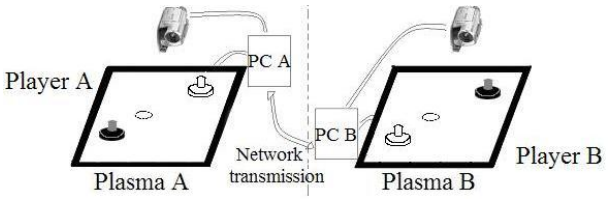


Figure 1: General View

1. INTRODUCTION

Real world games and computer games have their own distinct strengths. Augmented reality allows us to combine strengths from both reality and generated reality by improving existing game styles and producing new ones.

Nowadays, camera-based body motion capturing console games are very popular such as PS2 or XBOX. The players simply hold their hands in the air to control the virtual objects. However, no matter how deep is the involvement of the player, he cannot touch the virtual objects. [9] is one of the examples. The significance of the proposed system is that it can provide the players with real experience to interact with real objects remotely with other players, for example, playing UNO card game using real UNO cards, or playing Chinese chess using half number of real chess pieces on one terminal.

Our project aims at providing a generic platform on which players are allowed to play many types of games such as board games (Chinese chess), card games (UNO) or interactive games (air hockey) using real objects. You can even use it to give personal tutorial or presentation to the others. A general overview is shown in Fig. 1 and Fig. 2.

(a) Terminal A (b) Terminal B

Figure 2: Application: Chinese chess

This paper is organized as follow. In Section 2, we review some recent applications of Augmented Reality techniques in computer games. Section 3 shows our hardware setup

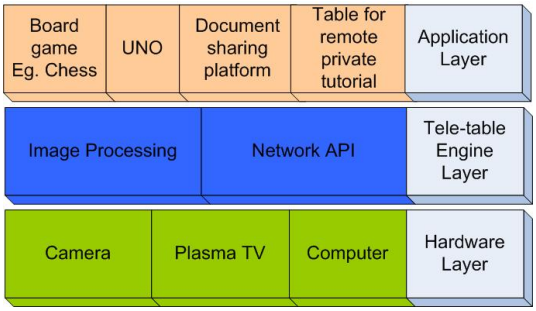


Figure 3: Layer View

and system architecture of Tele-Table system. Section 4 describes the problems we face and the solutions are discussed in Section 5. Finally, Section 6 draws our conclusion.

2. RELATED WORK

Augmented reality is becoming popular in the digital entertainment area [3]. It gives users immersed feeling into the combined virtual and real scenes, which is a very attractive technique for game development. Many recent papers describe some applications of AR in computer games.

In Augmented Reality Checkers [1], a computer generated virtual players playing checkers on a real game board, acting in the same way as real players. In Augmented Reality Mah-Jongg [6], players play and grab virtual tiles from their private panel with a real pen. ARQuake [7] is an outdoor/indoor augmented reality first-person shooting game modified from the desktop game Quake, where players can see and shoot monsters from their personal view through headmounted devices. Another system, the MIND-WARPING System [5], allows two groups of players to fight each other, where the magician group works at a workbench to set monsters to beat the "fighter" group, while "fighter" group see these virtual monsters overlaid with real-world and use gesture to destroy the monsters.

Most of these AR games require tailor-made expensive equipments such as Optical See-through HMD (Head Mounted Display) or sensors. Some well-known examples are ARQuake and AquaGauntlet [8]. Moreover, these AR gaming systems are usually tailored to specific tasks. For example, AR Quake is solely designed for first-person shooting game.

In our system, the Tele-Table, players can use any real objects to play games. One of the examples is AR-Table [2] which allows players to play card games using any cards. In order to support general functionalities, we adopt the image processing approach instead of the computer vision approach or the RFID approach.

3. PROPOSED SYSTEM

The hardware setup of Tele-Table is shown in Fig. 1. The main purpose of the Tele-Table system is to read the video signal from the camera, extract the images of the real objects on your own terminal, and send these images to the remote terminal for further processing. At the same time, it could add any virtual objects on the video stream coming from the remote terminal with a modifiable background before rendering it on the local monitor. The system can be viewed from two perspectives—Layer View (Fig. 3) and Module View (Fig. 4) which are described below.

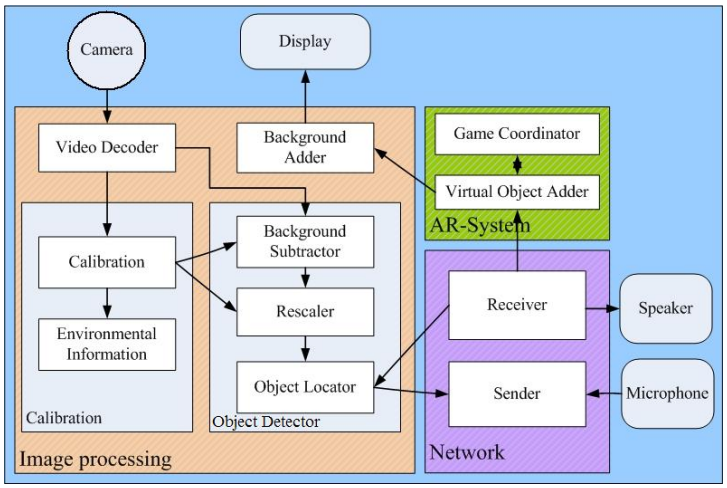


Figure 4: Module View

From the Layer View perspective (Fig. 3), the Tele-Table system consists of three layers including the Hardware Layer, the Tele-Table Engine Layer, and the Application Layer.

The Hardware Layer is the fundamental layer that is responsible for capturing the objects or events from the real world, processing the data and displaying the images on the local monitor.

The Tele-Table Engine Layer includes the Image Processing part and the Network API part. After capturing the real time video from the camera, the Image Processing module is responsible for separating local real objects from the remote virtual objects and the background. It then sends these real object images to the remote terminal through the Network API, which is responsible for data communication between the two terminals.

The Application Layer is responsible for managing the game rules and checking the game rules against the incoming objects. It renders the appropriate virtual objects on the screen by triggering various functions in the Image Processing module.

From the Module View perspective (Fig. 4), the system is divided into three main modules including Image Processing Module, Network Module, and AR-System Module.

Image Processing Module reads the raw video from the camera. During the calibration stage, the Calibration module calculates several coefficients of the environment and updates environment information, which provides essential parameters for Background Subtractor and Rescaler. During the running stage, the Object Detector receives the video from the camera, subtracts the frames from the background, rescales the frames so as to reduce the distortion caused by the camera. After that, the frames will contain the images of the real objects on your own terminal and any virtual objects. As what the receiver receives is the image of real objects from the remote terminal, by comparing the frames from the Rescaler and the Receiver, the Object Locator is able to locate the images of the real objects on the local terminal by comparing the position of these objects (Fig. 6). These frames are then passed to the Sender.

Network Module is responsible for transferring data. The Sender receives the frames from the Object Locator and sends it to the remote terminal. What it sends is the images

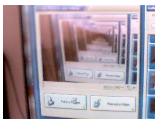
of the real objects on the local terminal. The Receiver receives the frames from the remote terminal and passes it to the Object Locator for comparison and to the AR-System for adding virtual objects. What the receiver receives is the images of the real objects on the remote terminal.

AR-System is responsible for coordinating the game rules and adding any virtual objects on the frames. The Virtual Object Adder receives the frames from the Receiver and adds any virtual objects on it according to the Game Coordinator which coordinates the game rules. It then sends the frames to the Background Adder for adding background and rendering on the local monitor.

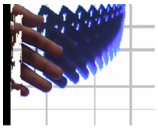
4. INTRINSIC PROBLEMS

For this system, there are mainly three intrinsic problems that should be dealt with. They are Mirror Effect, Barrel Distortion, and Frame Delay. In the following sections, we will outline the difficulties in solving these issues and propose the solutions in Section 5.

Mirror Effect—Barbershop Mirror Effect happens when two plane mirrors are placed parallel to and facing one another. When an object is placed between them, multiple images are seen. Barbershop Mirror Effect can also happen in multimedia devices. This effect happens when using a camera to capture the monitor which displays the video captured by the camera. This scenario is shown in Fig. 5(a), which is a recursive image. In our hardware setup, a camera is used to capture the screen of the monitor on each terminal, assuming no image processing is done, a real object (e.g. a hand) is placed on the monitor of one terminal (let it be terminal A, and the other terminal be terminal B). The camera at terminal A captures the hand and sends the image to terminal B. The virtual hand is rendered on the screen of terminal B and captured by camera in terminal B, then it is sent back to terminal A and displayed on the screen of terminal A. As there exist distortion in the camera, the real hand will not cover the virtual hand exactly. As this process continues, multiple images of the hand can be seen on both terminals. This scenario is shown in Fig. 5(b).



(a) Recursive Image 1



(b) Recursive Image 2

Figure 5: Mirror Effect

Barrel Distortion—Basically, the Barrel Distortion is where horizontal and vertical lines bend outwards towards the edges. By definition, Barrel Distortion is allied with wide angle or minimal zoom lenses. It causes the images to appear spherical or somewhat curved outward. Most digital cameras suffer from this type of distortion especially if it contains a zoom lens [4]. This distortion makes the subtraction in the Object Locator imperfect. This scenario is shown in Fig. 6.

Frame Delay—The Object Locator has two inputs (Fig. 4), one (Input 1) is connected to the Receiver and the other (Input 2) is connected to the Rescaler. After the Receiver receives the video frames from the remote terminal, it delivers the frames to the Object Locator as Input 1 and the AR-System Module simultaneously. The AR-System Module then passes the stream to the Background Adder and Renderer for displaying on the screen. After this, the camera captures this stream which is then passed to the Background Subtractor, the Rescaler and the Object Locator as Input 2. Thus, the frames from the Rescaler lag behind the frames from the Receiver. Therefore, the same frame from Input 2 comes later than Input 1. With this frame delay, the Object Locator is not able to remove the images of objects from the other terminal by comparing its position because these objects may be moving during this small time interval. The remaining images consist of the images of real objects on both terminals. As the frame does not only contain images of real objects on your terminal, this situation is not what we desire.

(a) Rescaler's frame (b) Receiver's frame (c) After subtraction

Figure 6: Left edge after subtraction. The image in black indicates object on my terminal and the image in gray is the image from the other terminal.

5. IMPLEMENTATION

Dealing with Mirror Effect—The perpetual imaging in the Mirror Effect gives great challenges to handle noise since the unremoved noise or images, which are not from the real objects on your terminal, will be sent to the other terminal with problems.

In order to solve this problem, we adopt an algorithm to remove connected components with small area before transferring the frames to the Sender for sending to the other terminal so that the remaining noise will be removed completely.

Dealing with Barrel Distortion (Spatial Synchronization)—To deal with the Barrel Distortion problem, we employ a texture wrapping method. We take advantage of the current commodity graphics accelerators which render millions of texture-mapped triangles in real-time. Upon receiving all video frames, they are used as the texture maps. Using the precomputed triangulated panorama (Fig. 7(a)) and its texture coordinates, the real-time incoming video frames (Fig. 7(b)) are used as texture maps and wrapped onto the triangular meshes in the precomputed panorama and blended together to form a panoramic frame in a real-time fashion.

(a) Precomputed panorama (b) Video frame as texture map

Figure 7: Wrapping

Dealing with Frame Delay (Temporal Synchronization)—In order to solve this problem, Visual Time Code

Scheme is introduced. Visual Time Code (Fig. 8) is added at the bottom of the screen to indicate the frame number of the video stream. In our coding system, the Visual Time Code consists of four bits which are shown on the bottom of Fig. 2.

Figure 8: Visual Time Code in Gray Code

Frames from the Receiver are put into a large buffer first which is of size 16 and stores the last frame with Visual Time Code X in the X-th position of the buffer. After one frame has been captured by the camera, the Object Locator decodes the Visual Time Code and retrieves the frame number, which is used as an index to match the corresponding frame in the large buffer. By comparing the corresponding frame in the large buffer, the Object Locator is able to locate the real objects on your terminal and remove any unrelated images by comparing the position. The Frame Delay problem is then solved.

To add to the above, the Visual Time Code used in our system is Gray code which means that two consecutive numbers differ from one digit only. The reason is that when the Visual Time Code is changing and the camera captures it, what the camera captures may be neither white (0) nor black (1), this leads to decoding error. Suppose the probability of successful decoding of one bit is P when the bit has just been changed. For the binary coding scheme, the four bits may change simultaneously in the worse case. The probability of successful decoding for four bits is P^4. For the Gray code scheme, as there is only one changing bit at a time and the other bits must be decoded correctly. The probability of successful decoding for four bits is P. With this Gray code scheme, the probability of decoding error can be reduced significantly.

Implementing AR-System module (Game Data Synchronization)—The AR-System module is mainly implemented as a Finite State Machine (FSM). Each of the terminals has its own FSM and its own state. When the terminals transfer their images to the other, it tries to encode the FSM state into its outgoing packets. After receiving the packets from the other terminal, it tries to decode the FSM state of the other terminal and uses it as one of the inputs of its own FSM. The developer can decide which objects should be real and which objects should be virtual and implement the game as a FSM. In our system, we have developed Tic-tac-toe and Chinese chess (Fig. 9). Moreover, developers can manage game rules within the FSM and recognize game events or objects by checking the outgoing frames on each terminal because the outgoing frames will only consist of the images of the real objects on your own terminal. The developers can apply some algorithms to detect these events made by the user.

6. CONCLUSION

We have developed a new gaming style, the Tele-Table interactive system, supporting games with real objects and virtual objects simultaneously across the network. This system extends the player's familiar game experience into a "Tele-pursuit" environment. In the current setup, many minigames, board games and card games can be supported with changeable background. Developers can decide the gaming environment and implement with the AR-System module to suit their requirements. Two of the examples are Tic-tac-toe and Chinese Chess (Fig. 9). In the near future, we plan to fine-tune our image processing modules to make the virtual object more accurate. We also plan to make modifications to the code in order to improve system performance. Lastly, we hope to have the System Developer's Kit (SDK) available for developers to extend the system.

(a) Tic-tac-toe (b) Chinese chess

Figure 9: Application examples

7. REFERENCES

[1] S. Balcisoy, R. Torre, M. Ponder, P. Fua, and D. Thalmann. Augmented reality for real and virtual humans. In *Proceedings of Computer Graphics International*, pages 303–307, 2000.

[2] A. H. T. Lam, K. C. H. Chow, E. H. H. Yau, and M. R. Lyu. Art: Augmented reality table for interactive trading card game. In *Proceedings of ACM International Conference on Virtual Reality Continuum and its Applications*, pages 357–360, 2006.

[3] M. R. Lyu, I. King, T. Wong, E. Yau, and P. Chan. Arcade: Augmented reality computing arena for digital entertainment. *In 2005 IEEE Aerospace Conference, Big Sky, Montana, U.S.A.*, 2005.

[4] H. T. Ngo and V. K. Asari. A pipelined architecture for real-time correction of barrel distortion in wide-angle camera images. *IEEE Transactions on Circuits and Systems for Video Technology*, 15(3):436–444, March 2005.

[5] T. Starner, B. Leibe, B. Singletary, and J. Pair. Mind-warping: Towards creating a compelling collaborative augmented reality game. In *5th International Conference on Intelligent User Interfaces*, 2000.

[6] Z. Szalavari, E. Eckstein, and M. Gervautz. Collaborative gaming in augmented reality. In *ACM Symposium on Virtual Reality Software and Technology*, 1998.

[7] B. Thomas, B.Close, J. Donoghue, J. Squires, P. Bondi, M. Morris, and W. Piekarski. Arquake: An outdoor/indoor augmented reality first person application. In *Proceedings of International Symposium on Wearable computers*, pages 139–146, 2000.

[8] T.Oshima. Rv-border guards: A multiplayer entertainment in mixed reality space. In *Poster session of IEEE International Workshop on Augmented Reality*, San Francisco, USA, 1999.

[9] E. Tse, S. Greenberg, C. Shen, and C. Forlines. Multimodal multiplayer tabletop gaming. In *Proceedings of Third International Workshop on Pervasive Gaming Applications*, pages 141–150, 2006.