Enhancing a Telerobotics Java Tool with Augmented Reality

Nancy Rodriguez, Luis Jose Pulido, Jean-Pierre Jessel

IRIT - Institut de Recherche en Informatique de Toulouse
Equipe Synthèse d'Images et Réalité Virtuelle
118 Route de Narbonne
31062 Toulouse, France
{rodri, pulido, jessel}@irit.fr

Abstract. *This paper describes the integration of an Augmented Reality service into our telerobotics system ASSET. ASSET is a teleoperation tool written in Java offering the services of simulation, 3D visualization, devices management and Java3D/VRML2.0 models loading. ASSET allows the definition of behaviour for each simulation object, so it is possible to have entities with different degrees of autonomy sharing the same environment. The Augmented Reality service that we have integrated use the Java binding of the ARToolkit to allow operators and autonomous robots to gather information about the mission. Information points are represented in the real world by visual patterns which, when recognized by the Augmented Reality Service, trigger actions to be executed by the robot or activate virtual objects display.*

1 Introduction

Teleoperation aims at allowing manipulation in dangerous or unreachable work sites, like in toxic substances treatment or spatial exploration. However, as the teleoperated robot is far from the control site, delays in transmission of commands and feedback will appear. This latency could be reduced by using a virtual version of the robot that can be manipulated in real time [13]. The virtual robot is generally implemented using augmented or virtual reality. In Augmented reality environments, real images are overlaying with computer generated images. In Virtual reality, users interact with a virtual world representing the real work site. By using virtual robots it is possible to compensate communication delays because abstract control is less sensitive to latency than direct control [17]. Different projects following this way [14] has shown that virtual reality interfaces can improve mission knowledge by providing tools to analyze and understand the remote environment.

In our system ASSET (Architecture for systems of Simulation and Training in Teleoperation), we have applied virtual reality techniques in the design and implementation of an environment for teleoperation systems development. This tool allows flexible customizing and can be used as a testbed for evaluating interaction techniques, devices, simulation models and autonomous agents behaviours.

Augmented reality mechanisms have been added to ASSET to provide mission information in a different way. The video images from the real robot viewpoint are overlaid with virtual objects in order to guide the user or to signal that an action must be executed. This Augmented Reality service reinforces the overall teleoperation system by allowing users to discover and resolve problems not detected by the simulation module.

This paper is organised as follows: in section 2 we review related work in augmented reality and its application to teleoperation systems. In section 3, we provide a description of the tools used in our work: ASSET, ARToolkit and JARToolkit. Section 4 covers the Augmented Reality Service implementation. Finally, some conclusions and directions for future research are presented.

2 Background

In Augmented Reality (AR) environments, virtual and real elements coexist. AR enriches real images by superimposing virtual elements that the user cannot directly perceive: task instructions, world information (e.g. distance, temperature), etc. AR has been successfully applied in several domains as manufacturing, medicine, training and entertainment. It is widely used to add information about the environment being displayed (Figure 1). In medicine, for example, datasets collected from medical tests could be rendered and combined with a view of the real patient to give access to useful data simultaneously and ease diagnostic [11, 23]. In augmented prototyping, the real product prototype is enriched by adding textual annotations or by "virtually" changing prototype characteristics like material or colour [20, 24]. In touring applications, a user - wearing special equipment- can walk outdoors and visualize graphical objects that provide information about the environment [4]. This idea has been extended to entertainment applications. For instance, the ARQuake system allows users to play Quake game in the real physical world and experience computer-generated graphical monsters and objects [15,16].

As we have stated previously, in telerobotics virtual robots allow to compensate communication delays and to increase efficiency. The system ARGOS was showed that path planning is a more easy and accurate process when augmented reality is used [3,13]. The user can plan the mission and specify the robot's actions by manipulating the local virtual version and having results directly displayed on the real world images. Once the plan is finished and evaluated, the real robot can execute it. Furthermore, as Azuma states: "the virtual versions can also predict the effects of manipulating the environment, thus serving as a planning and previewing tool to aid the user in performing the desired task". Others approaches have used a simulated environment augmented with virtual fixtures to assist programming of teleoperation tasks[22]. In Lloyd's system [12], the operator interacts with a simulated environment, which models each objects as a polyhedron and implements full 3D contact dynamics. This system make easier to place and manipulate objects using input from a simple 2D mouse, allowing robotic programming for untrained users.

Fig. 1. Annotations in an Augmented Reality Application (Image courtesy of W. Piekarski and B. Thomas, Wearable Computer Laboratory, University of South Australia [25])

In Augmented Reality, data from the real world is provided by video cameras and tracking systems. Collected data are then processed to calculate the transformation to be applied to the virtual objects. Finally, the transformed virtual objects are combined with the real image and presented to the user. The most important aspect to be considered in an augmented reality application is the proper overlay of virtual objects onto the real scene. That means a precise calculation of the camera's viewpoint in real time to allow virtual objects to be located at the correct location in the image [7,9].

Several technologies (video see-through, optical see-through and monitor) are available to enable AR applications. A see-through Head Mounted Display (HMD) allows tracking of user's head and combines real and virtual sources using video or optical technologies. In optical see-through HMDs, the user can see the real world through the optical combiners located in front of his eyes. Video see-through HMDs do not allow direct view, the images from the real world are provided by one or two head-mounted video cameras. Video from cameras is combined with the virtual objects and sent to the monitors located in front of the user's eyes.

AR applications can also be monitor-based (Figure 2). In this kind of configuration, the positions of the video cameras are tracked and used to calculate the virtual scene. The video of the real world and the graphic images are then combined and displayed in a monitor. Optionally, the images may be displayed in stereo on the monitor, which then requires the user to wear a pair of stereo glasses [2].

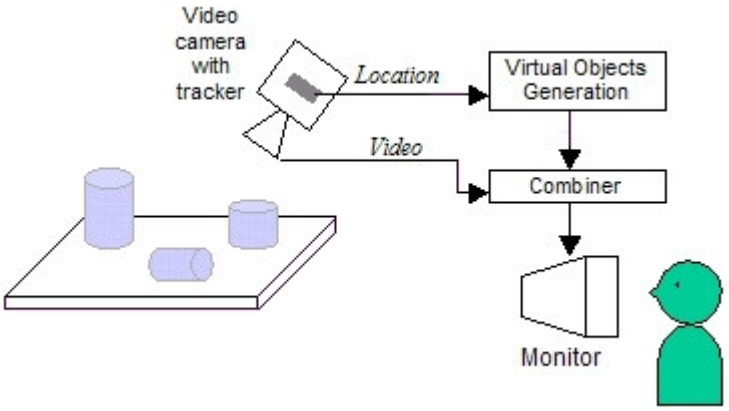

Fig. 2. Monitor based Augmented Reality System

The Augmented Reality Toolkit (ARToolkit) is a C publicly available library that enables the rapid development of new AR applications [1]. ARToolKit uses computer vision techniques to calculate the real camera location by using predefined marked cards, and allows the programmer to overlay virtual objects onto these cards. We have used JARToolkit [8], a Java binding for the ARToolkit, to implement an Augmented Reality service for our telerobotics system ASSET.

3 Tools overview

3.1 ARToolkit

The Augmented Reality Toolkit (ARToolkit) uses visual patterns (tracking markers) and their location in the real world to determine the camera viewpoint. The ARToolkit patterns are black squares with a black and white or colour image in the middle. Markers location is then used to overlay the pattern with its associated virtual object. This process is realized by the ARToolkit in several steps [9]:

1. First the live video image is turned into a binary image based on a lighting threshold value.
2. The binary image is then searched for square regions. ARToolkit finds all the squares in the binary image, many of which are not the tracking markers.
3. For each square, the pattern inside the square is captured and matched against some pre-trained patter templates. If there is a match, then ARToolkit has found one of the AR tracking markers. ARToolkit then uses the known square size and pattern orientation to calculate the position of the real video camera relative to the physical marker.
4. The real video camera location is stored in a transformation matrix that is then used to set the position of the virtual camera coordinates.

5. Since the virtual and real camera coordinates are the same, the virtual objects rendered precisely overlay the real marker.

3.2 JARToolkit

JARToolKit is a tool designed to offer the ARToolkit functionality to Java applications. JARToolkit use Java Native Interface (JNI) to access ARToolkit services and allows the use of different rendering libraries (Java3D and GL4Java) as an alternative to the OpenGL API used in ARToolkit for drawing the virtual objects. By using Java, JARToolkit also provides an object-oriented interface to the ARToolkit [5].

Two classes have been defined in JARToolkit to provide the ARToolKit functionality: JARToolKit and JARFrameGrabber. The JARToolKit class encapsulates all functions needed for tracking and some utility functions (e.g. to access the system time) and the class JARFrameGrabber provides all necessary functions to access video input from a camera. The two classes could be used separately in order to allow the development of different kind of applications.

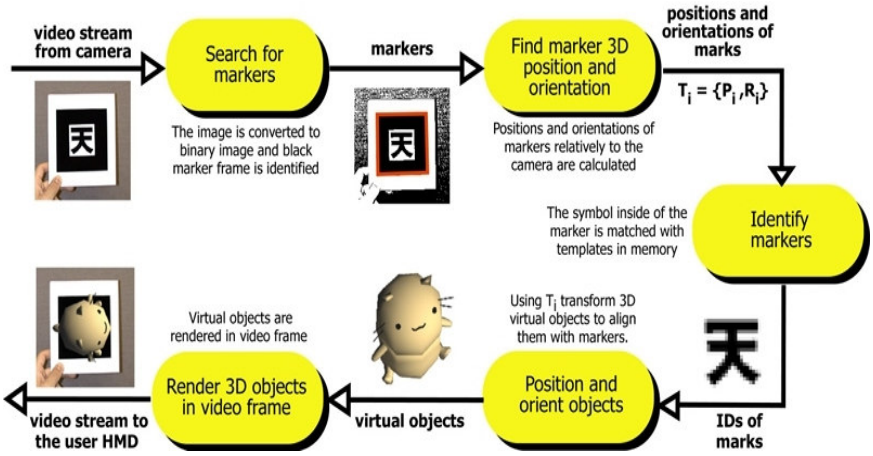

Fig. 3. The ARToolkit tracking process (Image courtesy of M. Billinghurst, HIT Lab, University of Washington [6])

3.3 ASSET

With ASSET we aim at providing a tool for helping development of telerobotics systems, following the philosophy of experimental platforms for virtual reality systems. We have also adopted distributed simulation techniques for minimizing the network bandwidth use and object-oriented development to offer a modular, flexible and easy to use system. Furthermore, because one of major limitations of actual systems is that they are available on just one or on very few platforms, we have

chosen Java and Java3d to develop our system, so that it can run on any platform without further changes. Also, even if we can use high-end displays, we are using a conventional computer monitor to display the virtual world. Our system can thus be regarded as a low cost virtual reality solution that can be used for many applications in telerobotics research [21].

Figure 4 depicts the ASSET architecture. We have two modules representing the local and the remote site and a third module that acts like a coordinator of the other two. In the local site, the operator generates commands to the robot by using the local site interface and the interaction devices. The commands are then executed by the simulation component and feedback is presented to the user. Only valid commands are sent to the remote site to be executed by the robot. The remote module transmits also commands to its simulation component and recovers real world state. If the difference between real world state and simulation state exceeds a user-defined threshold, the simulation state is updated in both, local and remote sites.

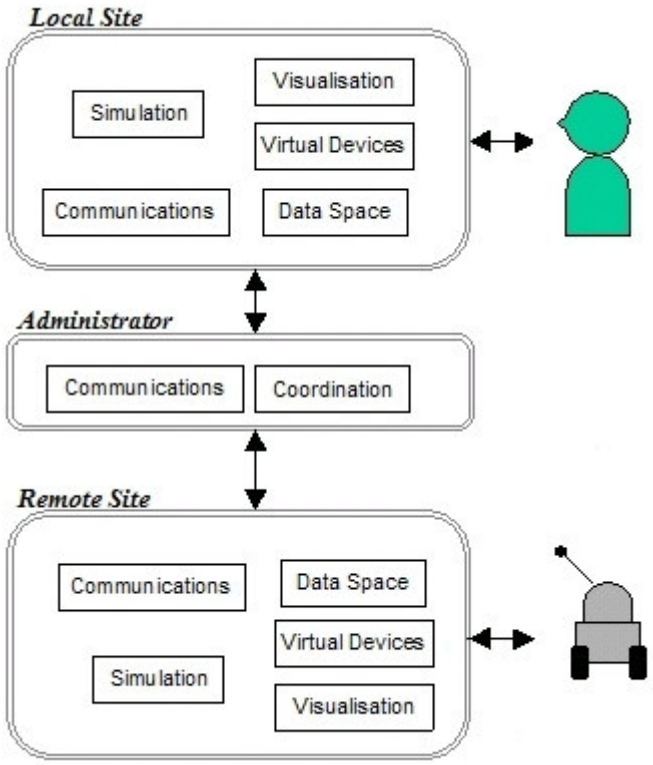

Fig. 4. Architecture of the ASSET System

Communication between ASSET modules is managed by their Communications component. All the interaction between components of a module (simulation, virtual

devices, visualisation and communications) are defined by message passing and synchronised by means of the data space and event handling. Modules and components functionality is described and accessed using interfaces so they can be modified without affecting others. To allow the reuse of ASSET components, specific application information is processed only by the components – such as physical devices controllers or behavioural units- supplied for the application.

The particular application components and models are loaded from a configuration file, read at initialisation. This configuration file allows the customisation of communication protocol, simulation objects (geometry and behaviour), interaction devices, actuators and command validity rules. For analysing different environments configurations and to allow rapid prototyping, the user can change this file, rather than modify the source code and recompile.

Integration of the Augmented Reality service

The Augmented Reality Service (ARService) that we have designed uses JARToolkit to allow operators and autonomous robots to collect information about the mission. Information points are represented in the real world by the tracking markers which, when recognized by the AR Service, trigger actions to be executed by the robot or activate virtual objects that provides information to the operator.

The AR Service has been integrated into the local and remote modules of the ASSET system. The remote AR Service captures the images and sends them to the local site. The local AR Service processes the image to detect patterns and superimpose the virtual objects. The final image, combining the image captured and the virtual scene, is then presented to the user.

In order to implement this AR Service, some modifications has been made to ASSET and JARToolkit [18]:

- ASSET modifications. By default, ASSET modules communicate through TCP/IP sockets using text messages. Therefore, to support images transmission, the communication structure of ASSET needed some changes. In the new version, the sockets are read and written using the Java classes ObjectOutputStream and ObjectInputStream, which allow transmitting any Java object implementing the Serializable[1] Java interface. The class Message, storing before the text message to be sent, has been modified to store any Java object and to implement the Serializable interface.
- JARToolkit modifications. JARToolkit has not been designed to work in a distributed configuration. To allow its integration in ASSET, a new class called

[1] Object Serialization supports the encoding of objects, and the objects reachable from them, into a stream of bytes; and it supports the complementary reconstruction of the object graph from the stream. Serialization is used for lightweight persistence and for communication via sockets or Remote Method Invocation (RMI).

ImageCam has been defined. This class, implementing also the Serializable interface, is used to store all the information about the image captured by the JARFrameGrabber. ImageCam also provides format conversion services to modify the image for network transmission and for processing in the local ARService.

A prototype of the AR Service was developed for Windows platforms. In this prototype configuration, we have a fixed camera viewing the environment and a mobile device being teleoperated. Our mobile device is a Khepera robot [10] with a tracking marker in its top face. This allows us to easily determine its location by using the ARToolkit tracking functions. Several tracking markers are disseminated in the environment, to be used as information points for the operator. The tracking markers are associated to virtual objects in the AR Service. When the robot walks over one tracking marker, the virtual object display is activated. In our current implementation, virtual objects are 3D text labels providing navigation clues (Figure 5). In the future, we will integrate more task related information. For instance, to aid in manipulation tasks, a virtual object (e.g. a virtual robot) could be superimposed in the location where the real robot will successfully pick or drop an object.

Fig. 5. 3D Text Label for Navigation

The AR Service can be activate and deactivate using the local site interface. When the user active the AR Service, a new window is open to display the images coming from the remote site. Therefore, he has two sources of information about the remote site: the 3D environment, which presents in real time actions results, and the AR display, which has an inherent delay due to communications latency. We are not interested in synchronising the two sources because of the effect that this will have in the overall user response time. However, we allow the user to update the 3D environment with the information of the real world anytime. We also planned to implement a function that will allow the user to recover the viewpoint of the captured image to apply it to the virtual camera. We think that this will ease recognition of real (virtual) objects in the virtual (real) world based in their locations.

ASSET is a work in progress and we can then improve our AR Service and the overall system by allowing the trackers markers to trigger actions to be executed by an autonomous robot. To do this, we have to add a vision module in the remote site in order to process captured images for recognize patterns. When a pattern is found (if the camera is on top of the robot) or when a condition is reached (e.g. distance between an environment pattern and the robot pattern in a fixed camera configuration), the vision process will send a message to the behaviour controller of the robot in order to execute a predefined action.

Conclusions and future work

In this work an Augmented Reality Service (AR Service) for teleoperation has been presented. It is based on the Java binding of the ARToolkit, a publicly available library allowing rapid development of Augmented Reality applications. The AR Service has been integrated in our telerobotics tool ASSET to provide additional data about the teleoperation mission being executed by an operator. In our current prototype, the operator can recover useful information by driving the mobile robot to the information points represented by visual patterns. When the AR Service recognizes a marker, a virtual object is superimposed into the real image at the marker location. The AR Service makes possible to combine virtual objects and video images of the real world to indicate, for instance, particular features of the real objects.

Future work includes the improvement of our prototype in order to allow the AR Service to be used by autonomous robots. Additional developments to ease correlation between the 3D synthetic world and the AR display are also planned. We are also interesting in changing our fixed camera configuration with a camera fixed on the top of the robot to take full advantage of the mobile nature of this one.

References

1. ARToolkit, http://www.hitl.washington.edu/artoolkit/

2. Azuma R.T.: A Survey of Augmented Reality. In Presence: Teleoperators and Virtual Environments, Vol. 6, No. 4 (1997)

3. Drascic D., Grodski J.J., Milgram P., Ruffo K., Wong P., Zhai S.: ARGOS: A Display System for Augmenting Reality. In Video Proceedings of INTERCHI '93: Human Factors in Computing Systems. Amsterdam, the Netherlands (1993)

4. Feiner S., MacIntyre B., Hollerer T., Webster A.: A Touring Machine: Prototyping 3D Mobile Augmented Reality for Exploring the Urban Environment. In: IEEE International Symposium on Wearable Computers (1997)

5. Geiger C., Reimann C., Stöcklein J., Paelke V.: JARToolKit – A Java Binding for ARToolKit. In: 1st IEEE International Workshop on the ARToolkit, Darmstadt, Germany (2002)

6. Human Interface Technology Lab, http://www.hitl.washington.edu

7. Ikeuchi K., Sato Y., Nishino K., Sato I.: Photometric Modeling for Mixed Reality. In: Proceedings of International Symposium on Mixed Reality, Yokohama, Japan, (1999)

8. JARToolkit, http://www.c-lab.de/jartoolkit

9. Kato H., Billinghurst M., Blanding R., May R.: ARToolKit Manual, PC version 2.11 (1999)

10. Khepera Robot, http://www.k-team.com/robots/khepera/index.html

11. Lopes P., Calado Lopes A., Salles Dias J.M.: Augmented Reality for Non-Invasive Medical Imaging. In: 1st Ibero-American Symposium on Computer Graphics, Guimarães, Portugal (2002)

12. Lloyd J.E., Beis J.S., Pai D.K., Lowe D.G.: Programming Contact Tasks Using a Reality-based Virtual Environment Integrated with Vision. In: IEEE Transactions on Robotics and Automation, Vol. 15, No. 3 (1999)

13. Milgram P., Zhai S., Drascic D., Grodski J.J.: Applications of Augmented Reality for Human-Robot Communication. In: Proceedings of International Conference on Intelligent Robotics and Systems, Yokohama, Japan, (1993)

14. Nguyen L., Bualat M., Edwards L., Flueckiger L., Neveu C., Schwehr K., Wagner M.D., Zbinden E.: Virtual Reality Interfaces for Visualization and Control of Remote Vehicles, In: Vehicle Teleoperation Interfaces Workshop, IEEE International Conference on Robotics and Automation, USA (2000)

15. Piekarski W., Thomas B.: ARQuake: the Outdoor Augmented Reality Gaming System. In: Communications of the ACM, Vol. 45, No. 1 (2002)

16. Piekarski W., Thomas B.: Interactive Augmented Reality Techniques for Construction at a Distance of 3D Geometry. In: Immersive Projection Technology, Eurographics Virtual Environments, Zurich, Switzerland (2003)

17. Pook P., Ballard D.: Remote Teleassistance. In: IEEE International Conference on Robotics and Automation, Japan (1995)

18. Pulido L.J.: Augmented Reality Environments applied to Teleoperation. DEA dissertation (in French), Paul Sabatier University, Toulouse, France (2003)

19. Regenbrecht H.T., Wagner M.T., Baratoff G.: MagicMeeting: A Collaborative Tangible Augmented Reality System, Virtual Reality, Vol. 6, No. 3 (2002)

20. Rodriguez A.N.: ASSET: A General Architecture for Telerobotics, PhD thesis (in French), Paul Sabatier University, Toulouse, France (2003)

21. Sayers C.R., Paul R.P: An Operator Interface for Teleprogramming Employing Synthetic Fixtures. Technical report, Department of Computer and Information Science, University of Pennsylvania (1994)

22. Seitber F., Hildebrand A.: Stereo based Augmented Reality applied within a Medical Application. In Computer Graphik Topics, Vol. 11, No. 1 (1999)

23. Stork A.: Augmented Prototyping. In: Computer Graphik Topics Vol. 14, No. 1 (2002)

24. Wearable Computer Laboratory http://wearables.unisa.edu.au