

Robot ARena: Infrastructure for Applications Involving Spatial Augmented Reality and Robots

Daniel Calife Alexandre Tomoyose Diego Spinola João Bernardes Romero Tori

Escola Politécnica da Universidade de São Paulo

Depto de Engenharia de Computação e Sistemas Digitais

Interlab - Interactive Technologies Laboratory

{daniel.calife, alexandre.tomoyose, diego.spinola, joao.bernardes, romero.tori}@poli.usp.br

ABSTRACT

This paper describes an augmented reality infrastructure composed of a camera, a projector, software for tracking, controlling and augmenting reality, and a real robot that can interact with projected virtual content. This spatially augmented environment is intended to be used as a testbed for experiments in interactive augmented reality applications, such as innovative games, educational applications and telepresence. The techniques used to control, communicate and track the robot are discussed as well as the techniques to develop, project and interact with the augmented environment. As a proof of concept, a game developed using this infrastructure is presented.

Keywords

Augmented Reality, Computer Vision, Games, Robots and Tracking.

1. INTRODUCTION

Augmented Reality (AR) [2][3] provides a great potential for innovation to interactive applications through many research fields, such as visualization, interaction, game design, cognition and socialization.

This work presents an infrastructure, called “Robot ARena”, which can be used as a platform for development of innovative applications based on Spatial Augmented Reality (SAR) [4] and a wireless controlled robot. The main goal of the Robot ARena project was to implement and test that infrastructure, exploring new ways of combining low-cost tracking, controlling and spatially augmented reality techniques.

One of the main concerns of this research was to provide a basis to develop applications that further explore the interaction between real and virtual environments, not only with real elements influencing virtual ones, but also the opposite, i.e., virtual elements affecting the real world. The real portion of the Robot ARena infrastructure consists of a robot assembled with Lego Mindstorms components [10] and controlled through a wireless connection. During the construction of this robot, some important questions arose, such as its control and communication, discussed in section 4, and registration and tracking, as shown in section 5.

Another concern, no less important, is the fact that users should be free from uncomfortable devices, such as Head-Mounted Displays and, at the same time, be able to visualize the virtual content directly on the real world. SAR is, therefore, the natural choice, augmenting the real environment with projected images,

correctly registered in 3D. This technique and its implementation are discussed in section 6.

1.1. Robot ARena Setup

The interaction occurs over a common table, in whose center two red markers are positioned. These markers represent the geometric origin of the ARena coordinate system. Additional green markers can be positioned anywhere on the table, representing virtual objects, depending on each application.

Above the table, there is a camera, aligned so that its viewing direction is approximately perpendicular to the table plane. This camera captures images used for the registration of ARena elements.

The projector and the mirror that reflects the projection towards the table rest on a support behind and below the camera.

Figure 1 shows a visual representation of this configuration.

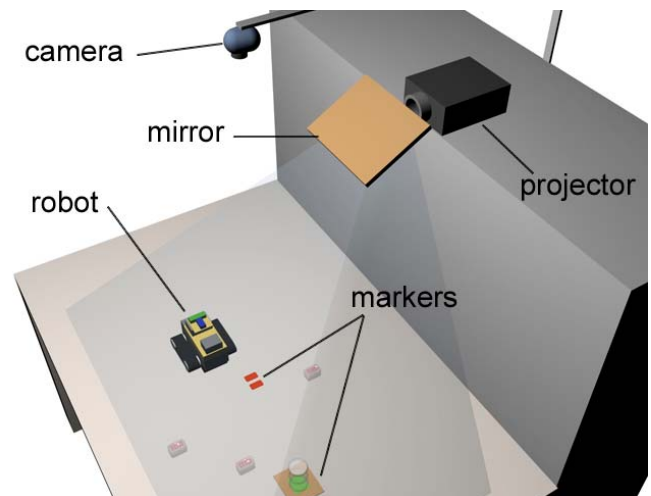


Figure 1. Robot ARena Setup.

2. RELATED WORK

Since the end of the last decade, several applications using augmented reality as an interface have been described in the literature. In the field of robotics, in particular, computer vision and augmented reality (and, of course, robots) have been used and combined even more numerous. This brief review will be limited to more recent projects, and those that are more closely related to the work described in this paper.

First, the concept of Spatial Augmented Reality must be more closely analyzed. Raskar and Low [13] introduce this concept as

the use of projectors to augment the physical environment with images integrated directly in the user's environment, not simply in their visual field. This augmentation can be 2D or 3D, but registered in 3D in relation to the physical environment, as per Azuma's definition of AR [2] [3]. The Robot ARena environment described in this paper is primarily based on this form of interface.

MacWilliams et al. [11] also use projection for the main module of their game, called Herding Sheep, and computer vision to track its elements, as done in the work described here. However, unlike Robot ARena, elements in Herding Sheep are manipulated directly by the user, which is an example of a tangible interface. A ready-made system called DTrack is used for tracking a number of spheres, and their spatial configuration characterizes the tracked object.

While it does not make use of virtual reality, the RoboCup F180 team FU-Fighters [14] also uses computer vision to track robots in real time on a table, and uses an interesting interactive algorithm. Each robot's position is tracked only in a window centered on its last position. The size of the window also varies based on the robot's speed, estimated in the last positions. This greatly reduces tracking time and it is possible that a similar algorithm will be implemented in future versions of this work, if it proves necessary.

An interesting project that makes use of augmented reality and virtual and real objects interacting "physically" is the Kobito project [1]. In this application, a real object, a tea caddy on a table set for a tea break, is moved around by "invisible brownies", virtual agents that can only be seen through a special display, the "Kobito Window", which is nothing but an AR interface for visualization. The caddy is actually moved around by a magnet under the table, which, in turn, is manipulated by the SPIDAR system. The virtual agents not only manage to push the real caddy around, but can also be pushed by it (in fact, that is the only way to interact with the brownies, through a real object) if a user manipulates it. The caddy even offers force feedback, representing resistance from the brownies. The system makes use of physical simulation, but unlike its classical applications, where all simulated elements are virtual, there is a real element that must be tracked precisely (this task is done with a camera) and manipulated (with the SPIDAR system). While in the Robot ARena described here the only physical interactions currently implemented between objects is a collision that does not move objects (virtual or real) around, but simply makes the robot stop, more complex simulations (such as a ball pushed by the robot) are not outside the platform's future scope. Unlike the Kobito project, however, which makes use of a manipulation system that can move the real object freely on the plane, the robot described here can only turn and move back and forward (it cannot be pushed sideways, for instance), so the degree of physical interaction of virtual objects with the robot will be necessarily limited and must be designed to hide this problem as much as possible from the user.

Finally, an application that looks very similar to the one described here is the Augmented Coliseum [9]. Augmented Reality is used to enrich a competition between robots with effects such as rays and explosions and with a virtual environment. Robots can also collide with obstacles in the virtual environment that can block their movement. Unlike the work described in this paper,

however, there is no need for visual tracking. The position of the robots is always known because they actively follow a certain pattern projected on them along with the virtual environment they are in. Each robot has 5 light sensors in known positions that detect variations in this light pattern and can identify where it is moving to and then follow it.

3. USED TOOLS

This section presents the most important tools used to develop the physical as well logical parts of the Robot ARena.

3.1. EnJine

EnJine [7] is an engine for the development of games in Java, developed and supported by Interlab/USP, available under the GNU General Public License.

One of enJine's main purposes is to serve as a teaching support tool in computer graphics courses [16], so it is didactic and relatively simple to learn and use, and has an architecture which facilitates good software engineering practices. Another purpose is to be a framework for implementing and testing new technologies, mainly applied to games.

EnJine was designed to better support action/adventure games, although it is possible to use it for other game styles. Some of enJine's main features are:

- 3D rendering based on Java 3D (which, in turn, currently uses DirectX or OpenGL);
- Stereo sound;
- Skin-and-bones animation;
- Multi-stage collision detection (currently only implementing subspace and bounding volume filters) and ray-casting collision detection;
- An abstract input layer to simplify the use of non-conventional input devices (this is particularly useful for AR games, for instance, to accept inputs from the camera);
- 2D Overlay;
- .X and .OBJ model loaders;
- Core classes constituting a game's basic architecture: the game, its stages, game objects (which are very flexible) etc.;
- A framework package to facilitate the creation of certain games (currently only single-player games);
- Separation between graphical and logical update cycles.

3.2. Lego Mindstorms

Lego Mindstorms [10], or Lego Robotics Invention System (RIS), is a set of blocks, motors, sensors and other components, such as gears and axles, created and marketed by the Lego group. It can be used to build many types of robots and other automated or interactive systems.

The main component of Lego Mindstorms, which controls input (sensors) and output (motors), is called the RCX block. The RCX has a Renesas H8/300 microcontroller that interprets the programs loaded in its RAM through an interface that uses infrared to send and to receive data. It is using the RCX and some other Lego blocks that form the real robot for this game.

3.3. OpenCV

OpenCV [12] (Open Source Computer Vision) is a free library of C/C++ functions mainly aimed at real-time computer vision and image processing.

Common applications that use this library are: Human-Computer Interaction; Object Identification, Segmentation and Recognition; Face Recognition; Gesture Recognition, Motion Tracking; Motion Understanding; Structure From Motion and Mobile Robotics.

As two of the main concerns of Augmented Reality are registration and tracking, and one approach to obtain them is through computer vision, OpenCV is a robust and flexible tool to be used in such applications.

4. THE ROBOT

The definition of robot embraces both autonomous and non-autonomous machines, the behavior of which can be programmed. In the case of the robot used in this project, although some low level programming takes place inside its microcontroller, the computer running the Robot ARena system controls its actions; therefore, it can be classified as a non-autonomous robot.

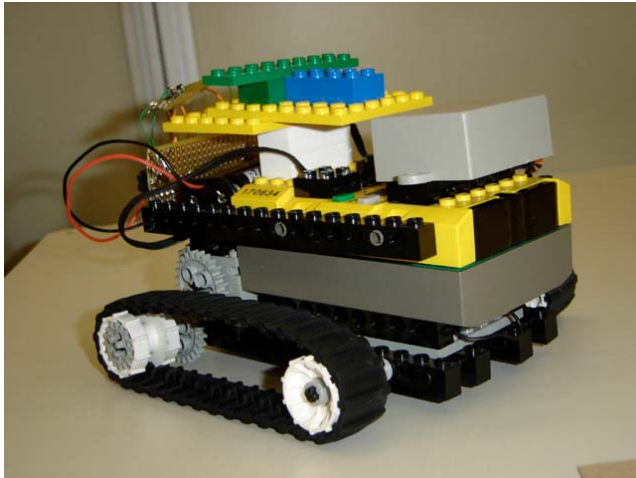


Figure 2. The Robot.

The robot is assembled upon the RCX block, which controls the input sensors and motors. As can be seen in figure 2, it is assembled using a tracked vehicle design, with differential steering, which results in a zero radius turn or a neutral turn. As opposed to using front wheels for steering (like a car steering mechanism) this track (or treaded) locomotion design is able to change the vehicle's orientation without any change in its position. This characteristic is of great importance when simulating the robot's collision response to virtual objects.

4.1. Communication

Originally, Lego Mindstorms uses an infrared (IR) communication protocol to transfer programs from a computer to the microcontroller unit (RCX). Although it is possible to use this interface to directly control the robot's motors, IR communication works best when both elements of the emitter-receiver pair are facing one another. An alternative communication system had to be developed in order to prevent unacceptable glitches and delays in the robot's response.

The Robot ARena uses a pair of wireless 2.4Ghz transceivers to connect the computer to the robot, both modules have a PIC (12F675) microcontroller that initializes the transceiver and works as a sophisticated data repeater.

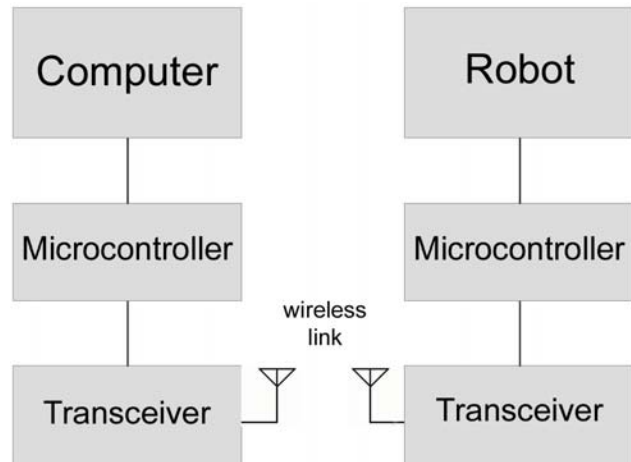


Figure 3. Communication Diagram.

As seen in figure 3, the computer sends data to the microcontroller on the transmitter module, the microcontroller decodes and repeats the information to the receiver module using the transceiver format. In the receiver module, the microcontroller receives data from the transmitter module and decodes it to a 3-bit word that is fed to the RCX through its sensors inputs (which are configured to accept digital pressure sensors). The RCX processes this word of data and activates its motors according to section 4.2.

4.2. Control

The robot is controlled through its three pressure sensors, acting only as a logical input, which can assume two states, 0 or 1. According to the state of each sensor, the direction of the two motors, one for each track, is determined and whether they are on or not.

These states are set by the 3-bit word from the receiver. The first bit sets the first sensor, the second bit sets the second sensor and so on.

The first and last sensors indicate the direction of the two motors. When the sensor is at 1, the motor runs forward and, when 0, backwards. The second sensor turns both motors on (1) or off (0).

With this setup, the robot can have the four necessary movements for interaction: moving forward or backwards and steering left or right, besides remaining still.

For example, when the robot turns left, the sensor controlling the left motor is set to 0, the sensor controlling the right motor is set to 1 and the second sensor controlling both motors is set to 1.

5. TRACKING SYSTEM

The tracking systems is what makes it possible to determine the robot's position and orientation in the real world, and to transform it according to the relative position in the virtual world. Thus the elements of both environments can interact.

The tracking algorithm is implemented in a C++ class that makes use of the OpenCV library [12], which captures and processes the images from a webcam.

Since it is necessary to get the robot's position and orientation in real time, this algorithm must be fast and simple. Therefore, it was decided [6] to use color tracking, which presents these desired features. It is also possible to use the very Lego blocks as fiducial markers. This kind of algorithm is commonly implemented for face detection [5], and also in specific applications to track robots [8][14].

To achieve simplicity, the tracking system only determines the marker's position in the X and Y axes. Due to the camera's position in this system, Z is the height axis and will not be considered, since the robot cannot move along this axis.

The algorithm is divided in two stages: initialization, which determines the referential points for the ARena, and the robot tracking stage.

5.1. Initialization

The initialization stage's main purpose is to determine a referential point in the real world that will be registered as the geometric origin of the augmented environment. Besides finding this origin, the initialization stage calculates the pixels to millimeters ratio, used to determine the positions of the robot and the obstacles.

Yet another purpose of the initialization stage is to find the number of additional elements, green markers, and their positions placed in the ARena.

This stage is also responsible for the camera initialization. And, although in this first phase the performance is not critical, it is necessary to wait about two seconds while the automatic adjustments of the particular model of camera being used are stabilizing. Other cameras may even present much larger times before this stabilization happens.

5.1.1. Pixels to Millimeters Ratio and Origin Registration

After the initialization of the camera, the algorithm searches the captured image for two red markers, fixed in the center of the table, of which the size and distance between centers are known. The red color is segmented, using a threshold with distinct values for each channel of RGB. As the predominance of red is very strong, this color representation system proves quite efficient.

The central point between these markers is found through the calculation of the average of each red pixel's coordinates. The coordinates of this point are stored and serve as the real environment's origin point, which will be the reference for the origin in the augmented environment.

Using this center, the segmented image is vertically divided in two regions, separating each marker, and in each region the same process is applied, thus finding the center of each marker and their distance in pixels.

With this distance in pixels and the known distance in millimeters between the marker's centers, the pixels to millimeters ratio can be calculated. This ratio is then used during the tracking stage to determine the real positions of the obstacles and the robot.

5.1.2. Additional Virtual Elements

In order to determine the additional virtual elements positions, represented in the real world by green markers, a process similar to the one applied to find the origin point is used, with an important difference: the application may have any number of

additional elements, so it is impossible to split the image in two halves.

An approach to solve this problem is to segment all the connected components of interest in the image, in this case, the green markers.

The algorithm used to find the connected components is a variation of the filling algorithm, a simpler variation of many similar algorithms [15], which uses a pixel stack and considers 8-connected pixels.

This stage is divided in two steps of segmentation: - the first one is the segmentation of the green color, using a process similar to the one detailed in section 5.2, but using the HSV color system instead of RGB; the second step segments all the connected components, counting how many were found and determining their position.

5.2. Robot Position and Orientation

The algorithm for tracking the robot's position and orientation is implemented as a method that updates the X and Y position and Orientation attributes of the tracking class.

These attributes represent the center and the direction of a marker, relative to origin point, mounted with two Lego Blocks, one green and one blue, which form a "T". Figure 4 shows this marker.

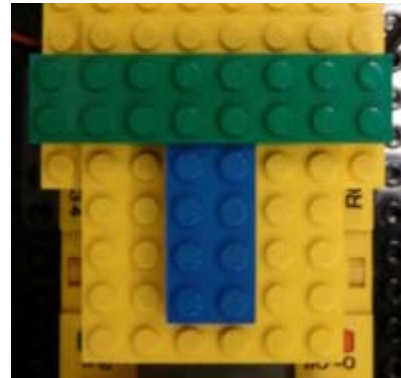


Figure 4. The Colored Marker.

The marker has two colors, so it is possible to separate each block and get a vector using both centers. This vector is used to find the marker's orientation. Since all Lego colored blocks have a strong saturation, the HSV color system was chosen to segment the images.

5.2.1. Position Tracking

The approach to determine the position of the robot is simple: find the center of the "T" marker in the image, transform it in relation to the origin and calculate the real position using the pixels to millimeters ratio found during initialization.

In order to find the marker's center, it is necessary to segment the image colors. The first step of segmentation is to convert the RGB image acquired from webcam to the HSV color system. Then a thresholding is done in the Saturation channel. A representative result of this thresholding is shown in figure 5, keeping the pixel's color of the highest values and setting others to black.

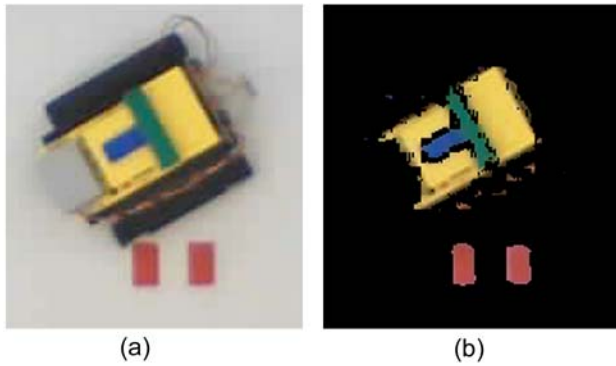


Figure 5. a) Image from camera; b) Image after Saturation thresholding.

At the same time, another thresholding is done in the Value channel. After that, using the values of the Hue channel to filter the green and blue colors of the marker, represented in figure 6, the average of all pixel coordinates for each color is calculated, finding the center of each block.

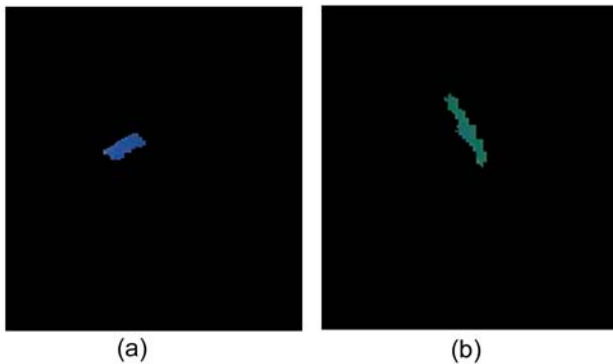


Figure 6. a) Segmented blue block; b) Segmented green block.

The average between these two centers is calculated, determining the marker's center. Then the distance of this center from the origin is calculated and the robot's real position is stored, multiplying this distance by the pixels to millimeters ratio.

5.2.2. Orientation

Once the coordinates of each block center are known, the vector between them can be determined and it is possible to calculate the arc-tangent of this vector, which returns the orientation in radians with values from $-\pi$ to π .

This value is stored and then transformed to the coordinate system used by the Robot ARena by the virtual environment system.

6. AUGMENTED ENVIRONMENT

As shown in the "Robot ARena Setup", the ARena's ground plane consists of a common table surface, over which the virtual content is projected, making it a spatially augmented environment, as shown in figure 7.

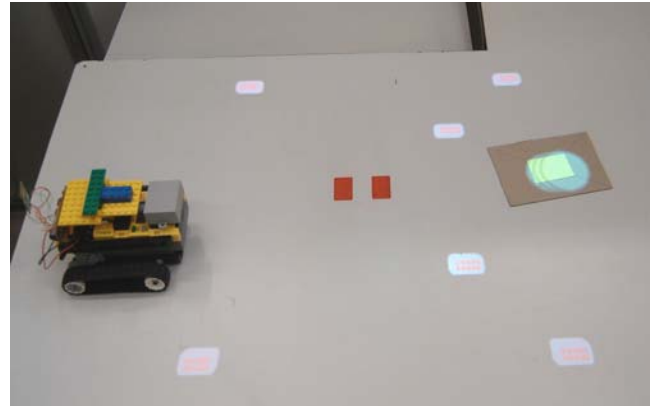


Figure 7. Spatially Augmented Environment.

The virtual elements must be projected in the correct position, based on the information provided by the tracking system. To achieve this, a virtual environment was created using enJine [7], where the virtual objects are rendered following the markers placed on the table. It not only renders all the elements, but also controls all the interaction between real and virtual elements and the interface between the player and the robot.

6.1. Projection

For the visualization of the augmented environment, this work uses the concept of projection-based spatial displays, presented in [4].

A common projector is used, connected to the computer video card and projects exactly the same content presented in the monitor display. To direct this projection, a mirror is used. Normally this mirror is positioned at an angle of 45° in relation to the projector, resulting in a 90° reflection to the projection. With this configuration, the mirror is located precisely at the projection center, and in this case it would totally obstruct the webcam's vision. To solve this problem, the mirror is positioned at an angle of 30° , aiming the reflected projection at an angle of 120° , which means a little bit forward from the mirror's center, but with the addition of some distortion.

Figures 8 and 9 show both configurations.

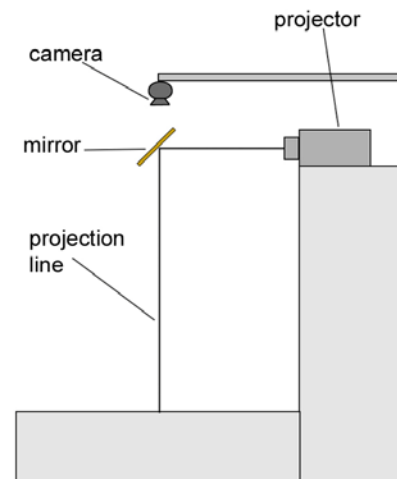


Figure 8. Mirror at 45°

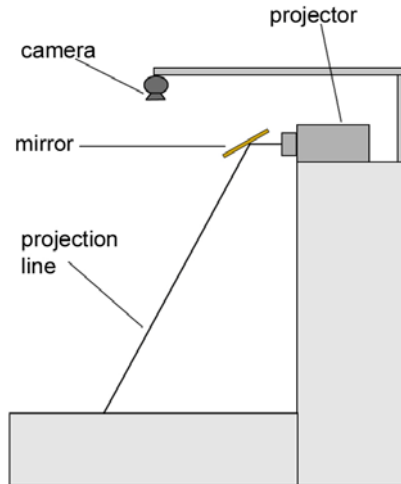


Figure 9. Mirror at 30°.

With these distortions, the registration between the projection and the real environment becomes more complicated. As the environment and the robot model of the Robot ARena have been created using a real scale, it was possible to use visual perception to accomplish the registration adjustments to correct the distortions caused by the mirror's angle. Through the comparison of the real robot with its representation in the virtual environment, the registration was calibrated adjusting projector settings, such as keystone and lens zoom. For a finer adjustment, the video card driver application was used. It allows the manipulation of the distortion for the images presented in the monitor, and also in the projection. The correction of these distortions was accomplished manually.

A top view with a fixed isometric projection was chosen for the augmented environment visualization. Thus, multiple spectators, and also the user, can visualize the augmented content comfortably. To provide representation of the augmented environment with better depth cues, it would be possible to project this content according to the user's viewpoint, but this approach reduces the user's freedom of movement (unless his head position is also tracked, somehow) and the possibility of a comfortable view for spectators.

A problem that occurs during the projection of the augmented environment is the interference of the luminosity level on the tracking system. As the tracking system is based on color segmentation, and especially in its saturation, it is necessary for the ARena to have a good illumination, which affects the projected elements perceived brightness.

Another concern regarding the projection is that the virtual elements cannot have the same colors or similar colors to the ones used for the tracking.

Figure 10 shows a picture of the Robot ARena with the projections.

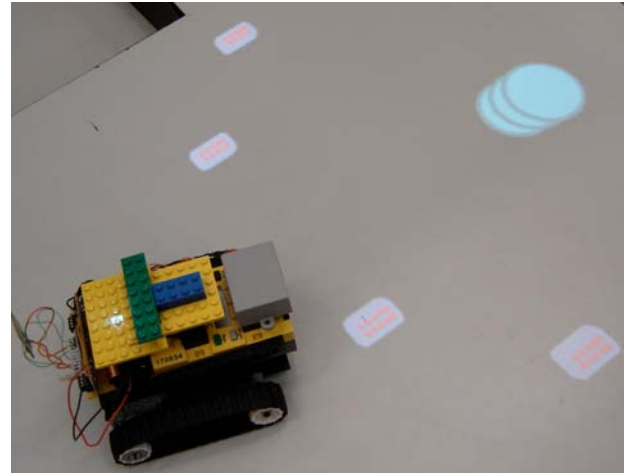


Figure 10. Robot ARena's Projection.

6.2. Integration

The piece of software responsible for managing the communication between the robot, the tracking system and the virtual content is the Robot ARena software developed over enJine's framework.

6.2.1. Tracking System

As the tracking system uses OpenCV, which is implemented in C++, and the system is based on enJine, which is implemented in Java, some interface between these APIs should be created.

To accomplish this task, a tool provided by Java was used that, based on a Java class that encapsulates the calls to the tracking system, creates a C++ header class that can be used to create a DLL. This tool is part of JNI (Java Native Interface). This DLL is loaded in the Java class and then it is possible to directly call its methods.

6.2.2. Robot Communication

The communication with the transmitter module responsible for sending data to the robot occurs through the computer's serial port. A free library called JavaComm is used to initialize the serial port and transfer the data to the communication module.

As discussed in section 4, this data consists of 3-bit words that set the sensor states according to the desired movement, entered by the user through the directional keys on the keyboard.

The Robot ARena software processes these user inputs and sends to the robot a number that represents the 3-bit word: 7 (1 1 1) to move forward; 3 (0 1 1) to turn left; 6 (1 1 0) to turn right, 0 (0 0 0) to stop and 2 (0 1 0) to move backwards.

6.3. Implementation Test

As an application test of the Robot ARena, a prototype game was developed, using all capabilities of the infrastructure that was created.

6.3.1. The Game Prototype

Although currently having simple rules and limited controlling options for the player, the game provides the necessary environment for the interaction between real objects and virtual elements.

The basic game mechanics is to collect the virtual projected batteries, moving the real robot over them. At the same time, the player must evade obstacles, represented by the metal barrels. The movements allowed to the player are going forward, backwards and turning right or left. If the robot hits a virtual barrel, it will not be able to move forward, simulating a real collision, so the robot must go backwards and then move around the barrel.

Any number of virtual obstacles can be located in the game area. These obstacles are represented in the real world by green markers, which are the additional virtual elements, and their positions are recognized in the initialization stage of tracking (section 5.1).

Besides this basic mechanics, three game modes were created: unlimited, time attack and timed.

In the first mode, unlimited, the game simply begins, placing the virtual batteries and barrels in the augmented environment, and the robot's movements are turned on, allowing the player to control it. In this mode, the player can take as long as he wishes to collect all the batteries.

In time attack, a timer is added to the game and stops as soon as all batteries are collected, so players can challenge one another to see who records the best time.

In the last mode, timed, the timer counts down from a preset limit instead of marking elapsed time, so that players have a limited time to collect all the batteries. If the clock reaches zero, the robot stops responding to the player's inputs and a counter shows how many batteries he was able to collect.

While this describes the game's prototype version, future versions promise to be more complex, as discussed in section 7.

6.3.2. *Virtual Content Creation and Interaction*

The 3D models representing the virtual objects were created with the aid of a modeling software. They are simple polygonal models with some colored materials.

The interaction between these models and the real robot is accomplished by using the collision detection system provided by enJine.

Through a bounding sphere that has the same position as the real robot on the augmented environment, it is possible to know when the robot moves over a battery or hits the metal barrel.

When the bounding sphere of the robot collides with the bounding sphere of a metal barrel, the game detects this collision and stops the robot, only allowing the robot to move backwards and move around the barrel.

If the collision is with a battery, the battery is removed from the game, representing its collection by the robot.

Figures 7 and 10 show images of the game.

7. CONCLUSION AND FUTURE WORK

Considering the objectives presented for this infrastructure, the achieved results are satisfactory and promising in all techniques presented here.

The main objective, the development of an infrastructure that allows the interaction in both directions between the real elements and the virtual ones, were explored and attained through the union of all the discussed techniques.

For the robot's assembly, Lego Mindstorms proved to be a simple tool to use, with great capacity and flexibility.

It was possible to develop a fast communication module for a more efficient solution to the interaction between the users and the robot, with an imperceptible response time between the command and the movement of the robot [6].

With the tracking system developed, the positioning and the orientation of the robot can be found in real time during the application, which makes possible the interaction of the virtual elements with the robot. It also allows the representation of the real elements in the virtual environment and the opposite, the representation of virtual elements in a real environment, which makes possible the projection of the virtual elements in the real environment.

OpenCV was an important tool, supplying an abstract layer to the camera's communication and direct and fast image processing functions, such as conversions between color systems.

The use of projections allowed a much more comfortable visualization of the augmented environment, making possible the presence of many spectators during the user's interaction with the application.

Regarding the distortions caused in the projection due to the mirror's positioning, it was possible to correct them, ensuring the correct registration of the environment, another way to solve this problem is using the homograph technique described in [4].

The problem with the influence of the environment illumination in the tracking system, hindering a brighter projection of the augmented environment, will be solved with the implementation of a tracking system using infrared LEDs. The algorithm will be very similar, but with the advantage of the infrared light not suffering direct influence from the surrounding illumination and the projection.

EnJine has proven to be an adequate choice as the system core, facilitating the creation of the virtual environment, the integration between different modules of the system and prototyping game mechanics.

Some performance measurements, application frame rates, were obtained, and these also demonstrated to be satisfactory. For these measurements, a PC with the following configurations was used: Athlon XP 2200+ processor, 512MB of RAM memory and a 3D accelerator video card with 128MB of memory.

The tracking algorithm (position and orientation) running isolated, without graphical rendering, reaches the average frame rate of 52,33 frames per second.

Including all the 3D rendering process, collision detection, interaction and integration of the Robot ARena's modules, all controlled by enJine, an average frame rate of 16.78 frames per second was measured during the prototype game.

Future versions of Robot ARena presented here will explore yet more interaction possibilities between the virtual and real elements, allowing the implementation, for example, of a soccer game with robots, including any combinations of disputes between real or virtual robots, as a game where both can be real or one of them a virtual robot. The ball would also be virtual, adding new tasks for physical simulation.

An interesting possibility brought by the Robot ARena infrastructure is the capacity to have applications involving remote users, in networked games, where each player controls his or her own real robot in a match where the robot of the opponent is presented as a virtual robot.

Another possibility is to use SAR techniques to change the visual appearance of the real robots, changing for example its color to represent a certain team of which the player is part or the robot's state. Stereoscopic projections, assuming that the user is tracked, can also be incorporated to this infrastructure, providing more realism to the augmented environment.

Finally, with Robot ARena, there is now a testbed for developing uncountable experiments involving robots and augmented reality, from educational tools to innovative augmented reality games. New techniques, in areas such as tracking, artificial intelligence, interaction and projector-based augmented reality, can also be researched and tested with the Robot ARena infrastructure.

8. ACKNOWLEDGMENTS

The authors would like to thank all Interlab students, researchers and professors, with a special thank to Ricardo Nakamura, for his suggestions and technical advice, to Mariza Leone, for her kind support and to Epa for his infrastructure support.

The authors also thank CNPq for Daniel Calife's master's research financial support and USP for Diego Spinola's *Bolsa Institucional de Iniciação Científica*.

REFERENCES

- [1] Aoki, T., Ichikawa, H., Asano, K., Mitake, H., Iio, Y., Ayukawa, R., Kuriyama, T., Kawa, T., Matsumura, I., Matsushita, T., Toyama, T., Hasegawa, S. and Sato, M. Kobito - Virtual Brownie - Virtual creatures interact with real objects and real people. SIGGRAPH2005 Emerging Technologies.
- [2] Azuma, R. A Survey Of Augmented Reality. Presence: Teleoperators And Virtual Environments, N. 6, p. 355-385, 1997.
- [3] Azuma, R., Bailiot, Y., Behringer, R., Feiner, S., Julier, S. and Macintyre, B. Recent Advances In Augmented Reality. Ieee Computer Graphics And Applications, V. 21, N. 6, p. 34-47, 2001.
- [4] Bimber, O. and Raskar, R. Spatial Augmented Reality. Siggraph 2005 Course 30 Notes. In: ACM International Symposium on Computer Graphics and Interactive Techniques, 2005.
- [5] Bradski, G. R. Computer Vision Face Tracking for Use in a Perceptual User Interface. Intel Technology Journal, Q2, pp. 1-15, 1998.
- [6] Calife, D., Tomoyose, A., Spinola, D. and Tori, R. Controle e Rastreamento de um Robô Real para um Ambiente de Realidade Misturada. In: II Workshop de Aplicações de Realidade Virtual (WARV 2006), p. 1-4, 2006.
- [7] EnJine: engine for games in Java Homepage. <http://enjinne.incubadora.fapesp.br/porta>, accessed in December/2006.
- [8] Hyams, J., Powell, M. W. and Murphy, R. Cooperative Navigation of Micro-Rovers Using Color Segmentation. Autonomous Robot, V.9, N.1, p. 7-16, 2000.
- [9] Kojima, M., Sugimoto, M., Nakamura, A., Tomita, M., Nii, H. and Inami, M. Augmented Coliseum: an augmented game environment with small vehicles. First IEEE International Workshop on Horizontal Interactive Human-Computer Systems, 2006.
- [10] Lego Mindstorms Homepage. <http://mindstorms.lego.com/>, accessed in December/2006.
- [11] Macwilliams, A., Sandor, C., Wagner, M., Bauer, M., Klinker, G. and Bruegge, B. Herding Sheep: Live System Development for Distributed Augmented Reality. In: IEEE and ACM International Symposium on Mixed and Augmented Reality, 2, 2003.
- [12] Open Source Computer Vision Library Homepage. <http://www.intel.com/technology/computing/opencv/>, accessed in December/2006.
- [13] Raskar, R. and Low, K. Interacting with Spatially Augmented Reality. In: International Conference on Computer Graphics, Virtual Reality and Visualization, 1, 2001.
- [14] Simon, M., Behnke, S. and Rojas, R. Robust Real Time Color Tracking. In International Workshop on RoboCup, Lecture Notes in Computer Science, p. 239-248, 2001.
- [15] Smith, A. R. Tint Fill. Proceedings of the 6th annual conference on Computer graphics and interactive techniques, p. 276-283, 1979.
- [16] Tori, R., Bernardes JR, J. L. and Nakamura, R. Teaching Introductory Computer Graphics Using Java 3D, Games and Customized Software: a Brazilian Experience. In: SIGGRAPH 2006 - The 33rd International Conference and Exhibition on Computer Graphics and Interactive Techniques - Educators Program, 2006.