The Arena: An Indoor Mixed Reality Space

José Miguel Salles Dias
Miguel.Dias@iscte.pt

Rafael Bastos
Rafael.Bastos@iscte.pt

Pedro Santos
Pedro.Santos@iscte.pt

Luis Monteiro
Luis.Monteiro@iscte.pt

Joaquim Canhoto
jgsilva@montepiogeral.pt

ADETTI/ISCTE, Associação para o Desenvolvimento das Telecomunicações e Técnicas de Informática, Edifício ISCTE, 1600-082 Lisboa, Portugal, www.adetti.iscte.pt

Abstract

In this paper we introduce the Arena, an indoor space for mobile mixed reality interaction. The Arena, includes a new user tracking system appropriate for AR/MR applications and a new Toolkit oriented to the augmented and mixed reality applications developer, the MX Toolkit. This toolkit is defined at a somewhat higher abstraction level, by hiding from the programmer low level implementation details and facilitating AR/MR object-oriented programming. The system handles in a uniform way, video input, video output (for head-sets and monitors), sound aurelisation and Multimodal Human-Computer Interaction in AR/MR, including, tangible interfaces, speech recognition and gesture recognition.

Keywords

Augmented Reality (AR), Mixed Reality (MR), Tangible Interfaces, AR Toolkit, User Tracking, Ultra Sound Tracking, Wearable

1. INTRODUCTION

Since the pioneering effort of Mark Billinghurst and Hirokazu Kato that has lead to the development of AR Toolkit and its deployment as an Open Source platform [Kato 2001], a large group of Augmented Reality (AR) and Mixed Reality (MR) application developers has emerged. Although AR Toolkit is based in simple and comprehensive C-based code, some drawbacks do exist for the AR/MR application programmer developer. In fact, the currently available AR Toolkit SDK for the creation of AR/MR applications show a number of deficiencies and don't completely satisfy the demanding AR/MR programmer requirements, namely:

- The lack of user tracking for mixed reality to be used in a higher programming layer, since a complete user tracking system providing 6 DOF for indoor settings for MR platforms and applications is too expensive;
- The lack of object oriented programming framework, easily accessible to the academic and research community, which brings well established programming advantages just like data hiding, polymorphism, operator overloading, re-usability of object classes, enforced modularity, program clarity and flexibility.
- The need to write specific code to handle directly the 2D and 3D Open GL-based graphics output, although some open source libraries do exist to attenuate the problem [openvrml], [openscenegraph].
- The need to handle directly the video capture device or other external devices.
- The need to manage separately fiducial marker representations, as provided by AR Toolkit and associated 3D shapes, although they are logically bound together in the AR Toolkit "philosophy" through the registration process.

The MR Toolkit [Uchiyama 2002], a development by Cannon, has an object-oriented SDK and satisfies most of the requirements enumerated before, but is not available for the academic and research worlds with economic advantage.

Regarding the user tracking system, the problem is largely linked to the cost of a global and complete tracking (position/orientation) system, which is financially impeditive. The high level of accuracy and performance make this kind of system too expensive for even an established research community. The challenge for us is then inevitable: "Can we build a low cost and reliable user tracking system and an object oriented SDK for AR/MR research and application development?"

The objective of this work is thus the creation of a new and exciting high-level mixed reality development environment, as an answer to the identified situation of the lack of availability of such environments for Aug-

lack of availability of such environments for Augmented and Mixed Reality research.

This environment requires:

- A user tracking system, ideally with 6 Degrees Of Freedom (6DOF) in an indoor mobile setting, to introduce the needed level of immersion for the user presence in the mixed reality world.
- A high-level object-oriented Software Development Kit - SDK for the Augmented and Mixed Reality Programmer.
- The uniform handling of various input/output media, such as:
  - Video input;
  - Video output (for head-sets and monitors);
  - Graphics output (for head-sets and monitors);
  - Sound aurelisation;
  - Multimodal Human-Computer Interaction, including, Tangible Interfaces, Speech Recognition, Gesture Recognition.

The high level of technical specialisation on different areas like electronic engineering, computer science and computer graphics, has turned the identified problems into a challenge.

The listed requirements, have led us to create the new concept of the Arena, *an indoor space for mobile mixed reality interaction*. The Arena, includes a new user tracking system appropriate for AR/MR applications and a new Toolkit oriented to the augmented and mixed reality applications developer, the MX Toolkit. This toolkit is defined at a somewhat higher abstraction level than the AR Toolkit software layer, by hiding from the programmer low level implementation details and facilitating AR/MR object-oriented programming.

MX Toolkit is an SDK written in C++, which uses extensively the AR Toolkit library for all matters regarding vision-based and marker-based tracking, but also features higher layer functions such as: Integration of ultra-sound user position tracking (providing two degrees of freedom in translation) and Inertia Management Unit (providing three degrees of freedom in orientation); Simple dialog with Microsoft Foundation Classes and other third party libraries, such as Open VRML and OpenSceneGraph; Integration of Open Audio Library (Open AL) for sound aurelisation; Integration of Open Computer Vision (Open CV) for image processing; Support for Tangible Interfaces, Speech Recognition and Gesture Recognition.

To properly support the user, we have also developed a wearable system, which is linked to a base station computer through a wireless analogue input/output video link, integrating: Autonomous power supply; Orientation user tracking system (Inertia Management Unit); Video see-through head-set.

In synthesis, in this paper we have identified certain problems in the development and deployment of mixed and augmented reality platforms and applications and, as a result, we have proposed some solutions to tackle them.

The paper is organised as follows: in section 2, we provide a background and state-of-the-art in the issues of platforms and environments for AR/MR authoring, interaction and application development. In section 3, we present an overview of the Arena Concept, including relevant issues, such as User Tracking, the Wearable Concept, the Arena Space specific hardware and the global system configuration. The user position tracking system development, namely, studying the best topology for the ultra-sound sensors layout in the Arena Space, and the User Position Tracking Algorithm, are discussed in Section 4. Section 5, covers the issues of the MX Toolkit System Development Kit. Section 6 discusses the User Tracking System evaluation results, and the Tests and Results of MX Toolkit and Arena Usage. Finally, in section 6, conclusions and future directions of research are given.

2. BACKGROUND

The deployment of Augmented Reality and Mixed Reality SDK platforms, interaction spaces and AR/MR authoring tools has emerged as an interesting trend in this research field, since the appearance of AR Toolkit. While not exhaustive, the following text, enumerates some of the more interesting R&D efforts in this domain.

Shared Reality Meeting is a proprietary Collaborative Augmented Reality Environment [Wagner 2002], oriented to workgroup design tasks and supporting several types of Tangible Interaction, although no specific general purpose SDK or platform is reported by the authors.

AMIRE is an European IST project [Grimm] which has released modules to ease AR/MR development tasks, envisaging similar aims as our own research, since its platform relieves the programmer of the detailed knowledge about the underlying AR/MR base technologies. AMIRE resides in the AR Toolkit tracking technology and, as a framework, uses a component-oriented technology, a reusable GEM collection and a visual authoring tool for building AR/MR applications. Our MX Toolkit is oriented specifically to the programmer, providing a simple and minimal set of C++ Classes (although they can grow with the contributions from programmers using the Toolkit), packaged into modules to aid the software development tasks.

The MR Toolkit C++ SDK from Cannon (a proprietary solution), available for a Linux PC environment, is able to manage a parallax-less stereo video see-through HMD. This SDK is composed of a C++ class library for developing runtime MR applications, is able to handle six degree-of-freedom (DOF) input sensors, and uses an alternative marker-based tracking to the AR Toolkit.

Our approach is similar, in the programming framework approach, to the former, but still adheres to the AR Toolkit tracking technology, which, although currently based in a single camera tracking technology, has proven to be a stable development framework and has gained the in-

terest of a large community of users. Our MX toolkit thus profits from the growing installed base of AR Toolkit application developers and from the evolution of this Open Source library, by an enthusiastic research community.

3. THE ARENA CONCEPT

The Arena concept goes beyond its physical space, its hardware structure and its computing architecture (see Figure 1). It provides a mixed reality interaction space,

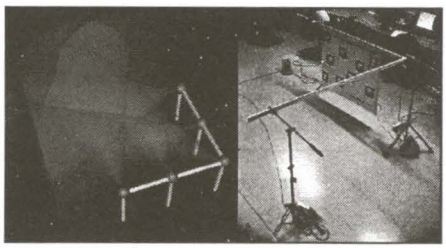

Figure 1. The Arena Space: Left – Virtual Model; Right: Physical realization in the HP New Media Lab

featuring:

- User tracking with five degrees of freedom (2 degrees in translation, *x*, and *y* and 3 degrees in rotation, *yaw*, *pitch* and *roll*);
- 3D virtual objects registration using marker-based approach (supported feature, see Figure 3 left) [Kato 2001] and a novel texture-based technique (on-going development, see Figure 3 right) [Bastos 2004];
- Multimodal HCI, supporting tangible interaction (supported feature) [Dias 2004a], gesture-based interaction (on-going development, see Figure 4) [Dias 2004b] and speech recognition;
- 3D spatial sound;
- A wearable unit, that can place the user in a complete mixed reality environment, by means of a video see-through head-mounted display;
- The MX Toolkit SDK [Dias 2003] appropriate for the mixed and augmented reality programmer.

To deploy our proposed interaction space for mixed reality as well as new paradigm for object-oriented mixed reality programming, we propose the following system architecture, comprising software and hardware layers.

The MX Toolkit, is a software platform oriented to the Augmented Reality/Mixed Reality application developer, aiming at simplifying his/her programming tasks.. The platform utilises extensively the AR Toolkit [Kato 2001], for all matters regarding marker-based tracking, but is defined at a higher abstraction level than the AR Toolkit software layer, by hiding from the programmer low level implementation details and facilitating AR/MR object-oriented programming, using or not the Arena space.

Our final version of the Arena Space covers an area of 9 m², and has a 3x3 side length (Figure 1). In each one of the ends, as well as at the centre and at the middle of

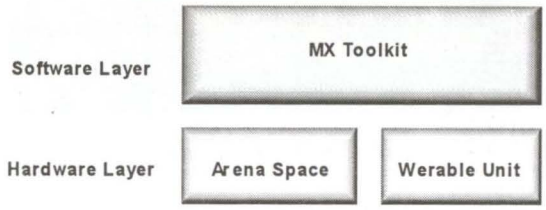

Figure 2. Arena System Architecture

each side, there are 5 ultra-sound sensors (emitter and receiver) installed in order to capture the user motion in its internal movement-sensing areas. In the Arena Space, a user can interact in AR/MR using a Wearable Unit. At the core of the Arena hardware, there is an Object-Oriented Programmable Interface Circuit (OOPIC) [OOPIC] that manages all the sensor responses, which are continuously sent to an external computer. In the Arena boundaries there are two pairs of loudspeakers (one at the front end and the other at the back) that enable the 3D spatial sound perception by the user, within any position of the Arena space. The Arena is handled by the MX Toolkit software package, which manages all the

Figure 3. Augmented Reality view in the Arena space: Left – Marker-based AR; Right – Texture-based AR

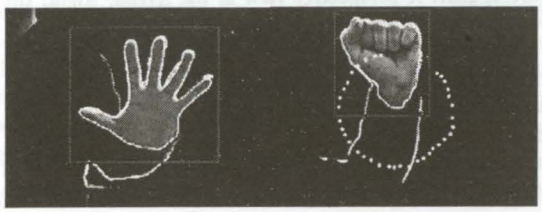

Figure 4 - Gesture-based interaction in the Arena Space: Left – recognizing a static pose; Right – recognizing a circle in a dynamic gesture.

events that will occur within its space, including the user's wearable unit.

The Arena's structure is easily portable. Three steel tripods support the infrastructure, and this setting allows its elements (cabling and electronic) to be easily transported. We can see a 3D model view in Figure 1 Left and the physical model in Figure 1 Right.

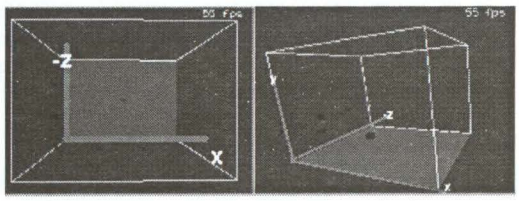

Figure 5. The Arena reference frame

3.1 User Tracking in the Arena Space

In order to create an immersive mixed reality environment perceivable to the user, we must track his/her pose (position and orientation) within the Arena Space, to perform virtual camera calibration, that is, to compute the underlying user's virtual point of view, that matches his/her real pose. Ideally the pose is characterized by three Degrees Of Freedom in position (*x, y, z*) and in orientation (*yaw*, *pitch* and *roll*), for a total of 6 Degrees of Freedom (6DOF). As an initial requirement, our system should be able to determine these 6DOF in real-time in a predefined and well-known coordinate system (the Arena reference frame), although our system supports just 5DOF, as it will be explained. This reference frame is shown in Figure 5 and is consistent with the one used in OpenGL, the graphics output API that has been used to develop our system.

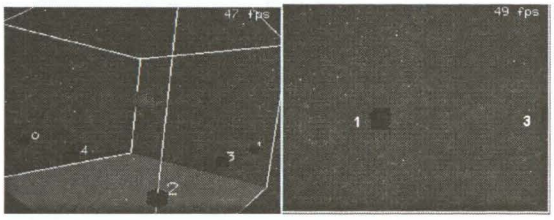

Figure 6. Left: A perspective view of the Arena; Right: the user's own view, within the Arena.

In Figure 6, left, we have a perspective view of the arena, where the small cubes represent the ultra sound sensors (which will be explained later) and the small teapot represents the user's simulated pose. The teapot was positioned and oriented according to the user's tracking information. In Figure 6, right, we depict the view from the user's own pose.

Figure 7. Registration of 3D virtual Objects in the Arena (the cubes match the ultrasound sensors and a helicopter is evolving in the Arena Space).

Once we have determined the virtual camera associated with the user's pose, we can perform the mixing of the reality with the virtuality. The reality is captured by a video camera and the registration is then made after the user tracking is updated (as shown in Figure 7), by means of a video-see through head mounted display. Additionally, and by means of other techniques, virtual objects associated to markers (Figure 3, left) or textures (Figure 3, right) can be also depicted in the arena. In conclusion, as long as we can successfully track the user's pose, we can compute the transformation required to relate the user's coordinate system to the world's (Arena's) coordinate system, apply this transformation to the virtual objects and add those to the user's view of the reality, resulting in a fully mixed reality experience.

3.2 The Wearable Concept

Our Wearable concept comprises small hardware elements that are always in reach and ready for operation. Our wearable system, is currently appropriate for static or mobile usage in the Arena indoor settings and it consists of a battery-powered system worn on the user's body (on a belt, backpack or vest), incorporating a head-mounted video see-through display, audio input/output devices and a small video camera, with analogue video

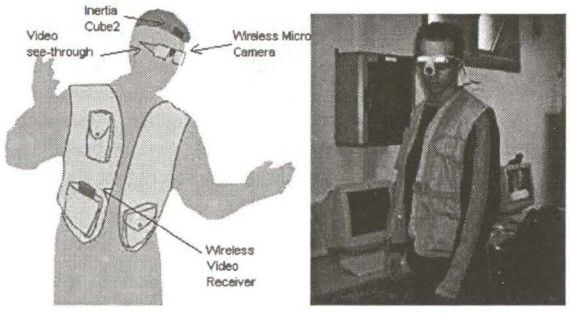

Figure 8. The Wearable Concept

in/out wireless links to a base station computer. The input/output video is processed by a base station computer, running a MX Toolkit based application.

3.3 The Arena Hardware

The Arena's hardware is mainly related to the user's positional tracking. It includes a programmable microcontroller and five ultrasonic sensors. The microcontroller is an OOPic (Object-Oriented Programmable Integrated Circuit) and it features: an object-oriented Basic-style programming language; I/O connectors (using the industry standard I2C bus – 5V power, ground, Serial Data and Serial Clock line) [I2C]; memory socket (for program storage).

The acoustic sensors are the "Devantech SRF08 Ultra-Sonic Ranger" [SRF08] (Figure 11). Its main features are: small dimensions (43mm w x 20mm d x 17mm h);

use ultrasound reflection on obstacles to determine their distance; measure a range from 3 centimeters to 6 meters; also operates on the I2C bus interface. The sensors are connected to the OOPic through a custom-made I2C hub with signal regeneration (a simplified diagram is shown in Figure 9). We have upgraded the microcontroller's default 5 voltage regulator with 100mA to a 1amp type to provide enough power supply to all five sensors. We have also implemented a TTL logic (the signal protocol at the OOPic) to/from a RS232 converter using a MAX203 circuit to provide serial communication through a common DB9 serial port in the PC. Another voltage regulator was made to convert from 12V-30V to the OOPic's supported 6V-15V power supply so that we could use our electrical power supply.

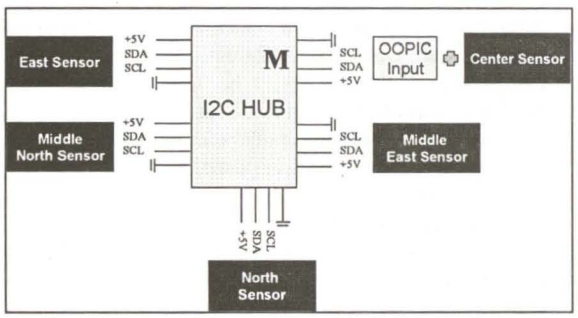

Figure 9. The Arena Hardware.

An appropriate program was developed for the OOPic using the hardware's IDE to control the sensors properly. It was uploaded through the provided parallel port and stored in the microcontroller's memory socket.

With this hardware set-up the OOPic is able to query the five sensor's ranges in runtime (at a determined rate) through the I2C bus and send this information to the PC using the RS232 serial communication.

3.4 System Configuration

The typical hardware and software platforms required by our system, are summarized as follows:

Hardware: Base Station Computer: CPU: Intel Pentium 3, 1GHZ; RAM: 1 Gbyte; Graphics card: NVIDIA GFORCE2 32 Mbyte; Video See-Through Head Mounted Display (Olympus Eye-trek FMD 700, Multimedia Glasses - 800x600 pixel), with a wireless video input link with the Base Station Computer via a Micro Cam wireless ultra miniature video camera from Swann Security. There is a video receiver at the Base Station Computer that captures the live video stream. The Inertia Cube2 mounted on a helmet placed at the user's head, so that his/her head orientation (*roll, pitch, yaw*) can be properly tracked. Arena specific Hardware (see Figure 9). Wearable Unit: Jacket equipped with bidirectional wireless video link and batteries (Figure 8).

Software: MS Visual Studio .NET 2003, ARTK 2.52, DirectX 9.0b, OpenGL and OpenSceneGraph; Indoor Tracking methods supported; Hybrid User Tracking: Ultra-sound (position) and Inertia Cube2 (orientation); Object tracking: Marker-based approach (AR Toolkit) and Texture-based approach.

The head mounted display shows the video images captured by a wireless video receiver working with its correspondent wireless video transmitter. The video transmitter receives images color space converted from VGA (RGB) to YUV, by the Trust Televiewer that is attached to the display adapter PC card situated in the base station computer unit. This type of transmission doesn't need to be in line of sight, since it works by radio frequency.Both the video camera and HMD are wireless based so all the analogue video stream control flow is cable free. The InertiaCube2 orientation tracker however, is not. As a solution, we have extended the Cube original cable from 3 to 9 meters with a custom made cable. This way, we managed to attach it across the ceiling and down the wall connecting it to the PC. The cable is therefore, suspended from the ceiling (at the middle of the arena) to the helmet which is quite acceptable to the user. There is also the power supply and consumption issue. Our objective was to dimension and design a pack of batteries that would be in the wearable so that all these devices could be powered. It would be necessary to check each device input voltage and design the necessary voltage regulators. Finally, we would choose the batteries Ampere-hour consumption according to our desired time length usage of the wearable. At the time of writing of this paper, this wasn't yet done. As an acceptable alternative, all the devices are connected to a power extension in the wearable, and this single cable goes out the vest and connects to a power plug outside the arena. With some care by the user this isn't too concerning. Finally, while using our wearable we have concluded there were very few movement restrictions so the final user's mobility turned out to be quite acceptable.

4. THE USER POSITION TRACKING SYSTEM DEVELOPMENT

During the evolution of the system, there have been several versions of the topology of the Arena, each one with its own advantages and problems, until a final and stable version has been reached. The user's positioning tracking algorithm has been also subsequently changed and improved. This section describes the main phases of arena's evolution. But first, a brief study of the ultra sound sensors is made

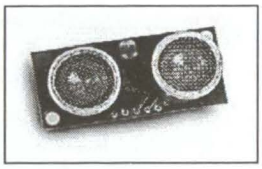

Figure 10. The SRF08 ultra-sound sensor

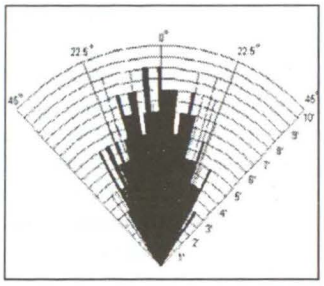

Figure 11. The SRF08 beam pattern

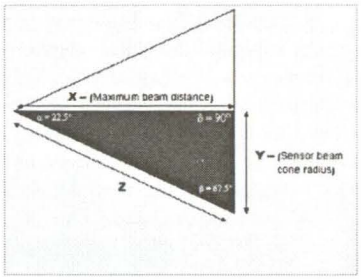

Figure 12. The SRF08 beam parameters

The acoustic sensors are an important element of the user tracking system. As mentioned we have adopted the "Devantech SRF08 UltraSonic Ranger" sensor (Figure 10). The sensor has two main components: the ultrasound emitter (e) and the ultrasound echoes receiver (r). The way these sensors work can be compared to a bat's ultrasound navigational system. The sensor/bat sends an ultra sound and waits for its reflection. Then it measures the time of the reflection and calculates the distance to the object through this reflection time (the round trip time from the ultrasonic wave emission until the arrival of its reflection wave) and the constant speed of sound (340 ms^{-1}):

$$Dist = \frac{Round\ Trip\ Time \times 340\, ms^{-1}}{2}$$

By analyzing the sensor beam pattern (Figure 11), the best performance and accuracy is obtained within a 45° angle (22,5° left plus 22,5° right). This angle is represented in 2D in Figure 12, but physically it has the shape of a 3D cone In Figure 12, Y = 1.16 m for a maximum beam distance of 2.8 m and Y = 414 cm for a maximum beam distance of 1 m.

The sensor determines an obstacle distance, but by its own means, it cannot determine where exactly the object is. With this scheme we can only predict that the object is within a specific radius (provided distance) around the sensor. In a 3D model, a semi sphere represents this distance (Figure 13). The small box represents the sensor and the orange lines represent the radius (provided distance) of the sphere. The detected object is somewhere on the surface of this hemi-sphere. The intersection of 3 different hemi-spheres, provided by 3 different sensors, determines two exact *x, y, z* coordinates in the Arena reference frame. One of the calculated points can be discarded, solving this ambiguity, because it will be outside the Arena's limits (see Figure 14). It should be noted that this method is independent of the number and layout of the sensors. The hemi-spheres used in the 3D models of Figure 14 have the same radius just for visualisation reasons. This radius normally differs for each semi-sphere unless the tracked object is equidistant to the different sensors. In any case, the 3D position can be fully determined.

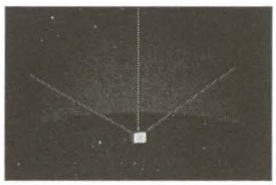

Figure 13. Position detection using one sensor

To cover an area with $9m^2$ with a certain topology of sensors, we have to analyze the sensor's beam patterns, namely, the maximum range in length of each beam and the room spatial conditions. It is also possible that one or more of sensor's generated distance values are invalid (false positives), and this can happen either because the user is outside of the sensors range, or because of errors in the reading procedure. We have to ignore these invalid sensors values and, in case there is no more sensors-generated distance information to correct the problem, determine the user position by software-assisted means. The layout and number of sensors chosen for our architecture must be defined to maximise the Arena's area of interaction and minimise these types of false positives.

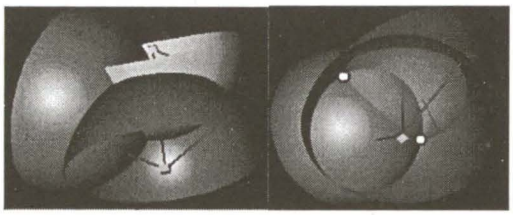

Figure 14. Triangulation of the sensors hemi-spherical ranges to evaluate the user *x, y, z* position

4.1 Sensors Layout

To address the mentioned requirements, two topologies were analysed and tested, a "L" structure and a "X" structure.

4.1.1 "L" and "X "Structures Topological Study

These structures are depicted in Figure 15 (L Structure with 3 sensors)) and Figure 16 (X Structure with 5 sensors). Its topological characteristics are: The green area is the 9 m^2 useful Arena area; The blue area represents the total area of the room (16 m^2); The red squares represent the sensors, separated by a distance of 1.2 m each; The red cones (1.2m radius, opening sensor angle of 45°) represent the beams projected by the sensors;The yellow structure is suspended at a height of 1.6m. (L Structure) or 2.8 m. (maximum value for the X Structure, considering the Lab room).

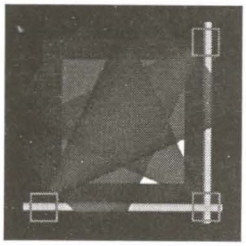

Figure 15. " L" Structure using 3 sensors, Top View

In Figure 17, we can see that only 3 beams intersected the centre of the arena simultaneously. All the green visible spots are regions not supported by the distance measuring tracking module. This means that if this "L" structure is used, we cannot determine the 3D position of the user in all the required area. For the "X" structure case (see Figure 16), the number of sensors was increased by 2 because there was a considerable amount of uncovered area in the "L" Structure. Despite this, three simultaneous beams cover the centre of the Arena, one/two sensors cover the sides but the corners are not covered at all. Still, it is possible to detect and get the complete 3D users' position within a significant part of the arena's area. The use of this kind of structure takes 2 more sensors than the 'L' structure, but the user's tracking is improved in theory.

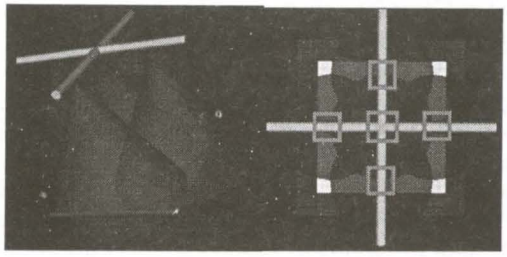

Figure 16. " X" Structure using 5 sensors, Perspective (Left) and Top View (Right)

4.1.2 Overview of both topologies

On the "L" structure, to decrease the possibility of occurring false echoes, we have to use sequential ranging. This type of ranging means that the first sensor performs a measuring, then the second, and finally the third. After all the ranging requests are completed, then we proceed to the distance acquisition in a sequential way. This is a slow process because we are accessing each sensor individually when requesting a ranging operation. Instead, we could use the broadcast ranging property of the I2C bus to increase efficiency, but on the "L" structure this is unsuitable due to the occurrence of errors (echoes). The possibility of occurring false echoes using this property on the 'X' structure is minimum, and this measuring process is much more efficient and robust.

The "L" structure is not able to return the complete 3D position of the user because the sensors are oriented directly towards the user's body, so the height coordinate is undetermined (it isn't possible to determine the user's height).

On the other hand, the "X" structure can be used to determine the height coordinate. The sensors are placed at a height of 2.8 m., so theoretically, the first object they detect is the tallest object on the arena, normally the users head.

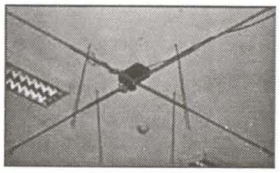

Figure 17. "'X' structure realization - 5 sensors

Based on the presented conclusions, theoretically the 'X' structure seems to be the most suitable solution for our user tracking system.

4.1.3 Pratical Realization

4.1.3.1 'X' structure implementation - 5 sensors

Like it was analyzed before, this kind of structure should be the most suitable to achieve the desired objectives. An inox structure was built to support the 5 sensors with the shape of an X, and all the hardware components described were installed inside a custom made box. This way we tried to give an esthetical and functional aspect to the hardware (Figure 17).

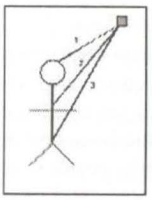

Figure 18. Disparities in the returned distances by a sensor, regarding the user position

On each extremity sensor, an inside inclination of 45° was set up. This way we managed to add more effective coverage to the centre of the Arena while keeping the arena borders.

At this moment, we had two cables connected to the base: a serial cable and the power supply. The structure was fixed in the ceiling, at a maximum height of 2.8 m, and this height could be adjusted through an adjustable iron rail.

Unstable distance values were sensed in our prototype, when used with a 1.7 m tall person. When a sensor "sends" an ultrasound, it sends it along a "circle radius", and it is easy to understand that depending of the reflector position, orientation, material and room conditions, the echo returned can assume different directions and consequently, different distances.

As depicted in Figure 18, a sensor detecting an object (user) that maintains its position along the time, can obtain different distances for each measurement. This happens because a sensor can hit the users head, as well its

waist, or even its feet, in its sequential measuring system. The instability is even more accentuated on this structure because the distances are wider, the sensor is not normal to the user body and the user own anthropomorphic nature is irregular.

If we generalize this single sensor problem the rest of the sensors, we can conclude that for the same object (user), different sensors can provide different and most of the time, incorrect measurements since they reefer to different parts of the user's body. It proved to be extremely difficult to create a position-tracking algorithm when the provided data was so unstable and inconsistent.

To try and solve this problem, a filter at the input was developed. Basically, we tried to interpolate the sensors values so that the disparity between sequential samples wasn't so apparent. It was based in a logarithmic function so that sensors current value tended to remain stable. Even still, it was quite difficult to achieve usable confidence stability. To even increase the difficulty in stabilizing the sensors measurement values, the used structure wouldn't guaranty us the complete detection on all the required area. On the theoretical approach, it was possible to cover a large part of the area, but this was the case if the sensors returned always the maximum distance – only possible at the floor. The problem was that the sensors would or would not return the maximum or minimum distance. The problem was that each sensor would return a distance to the users head, users feet, or even the floor, depending on the different factors referred.

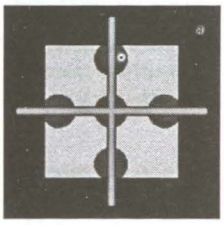

Figure 19. 'X' structure user's head coverage

If we analyze this structure again and if we place on a simulation the users head, the sensors would detect the area depicted in Figure 19. As it is represented, if the sensors would only return distances to an object at the same height of a user head (1.70m), the total covered area is completely unacceptable. By now we have realized that we have incurred on an error in our 'X' structure topological study: the area that we thought that would be covered by the sensors is actually at the user's feet but the area that we are interested is at the user's head (as shown in Figure 19) since the sensors are returning us the distances to the closest object that they find. So, the area covered was, in reality, much smaller than we thought.

We have concluded that with the room height available at our laboratory, even with more sensors, it would be extremely difficult to cover the complete Arena area (at a medium height of 1.70 m). Basically, the whole problem was that that the surface of the obstacle (the user's body) was not perpendicular to the sensor's direction, therefore, the reflected ultrasonic waves are very unreliable. If the sensors were in a higher position, the covered area would increase significantly; the assumption of the sensor's direction being perpendicular to the user's body would be more valid; and the sensors would return more accurate readings. Therefore and because of the number of encountered problems on this kind of structures, we decided to abandon the computing of the complete 3D position. Instead, we decided to only detect the 2D position and therefore, change the X-structure for the L-structure style, assuming a standard user height of 1.75 meters.

4.1.3.2 'L' structure implementation - 5 sensors

A new hardware to support the sensors was built, and placed at a height of 1,15m (see Figure 20). This particular structure has a great advantage relative to the previous "X" one in the fact that, normally, all the objects (users) placed within the arena are perpendicular to each sensor. Like it was explained previously, this particular property stabilises the sensor returned distance values and, consequently, improves the tracking. The structure height was chosen so that the sensors ultrasounds hit a user preferably in his/her waist. This way the user can move his legs and his arms having little influence in the positioning tracking system. A new 3D Model was studied and once more, theoretically, it would be possible to determine the position within the complete arena. We have added two more sensors than in the 'L' initially study, so that we could cover almost the entire Arena Space. Like it is represented in Figure 20, only the completely green areas are undetected, which count for a small part of the space.

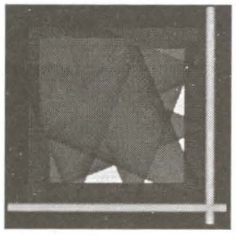

Figure 20. L' structure – 5 sensors - Arena's schematic top view

4.2 User Position Tracking Algorithm

In the "L" topology case, the user position tracking algorithm is based in evaluating analytically the intersection of two circles, whereas each circles radius is the distance to the user that a given sensor is returning. This gives us two 2D points, but we can solve the ambiguity because one of them will always lie outside the arena's margin. In Figure 21 we can see two sensors (the two black cubes) in the margins of the arena and both points of the circles intersection. We can see that we can safely discard the point outside the arena's margin so the remaining point gives the user's position.

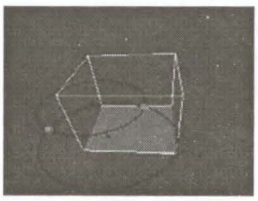

Figure 21. Basic User Position Tracking Algorithm: 2D circles intersection

We have developed two filters at the input and output of the user tracking algorithm. The input filter is at each sensor and starts by storing a history of past sampled sensor distances values.

The average difference between the values is computed and a confidence value is derived from it. The higher this averaged difference is, the lower is the confidence value (it means that there are big disparities between the distances evaluated by the sensors). The calculated estimated radius (distance of the user to the sensor) is the mean value of these samples, only if the confidence value is high enough. Otherwise, the estimated radius is set to zero, meaning that it should not be considered in the position-tracking algorithm.

By using 5 sensors we have 10 possible combinations of pairs of 2 sensors (without repetitions) whose circles intersect. From these 10 combinations we have at most 10 points of intersection (discarding the ones outside the Arena Space), but each of the 5 sensors covers only a part of the whole Arena Space so, we have (in almost all of the Arena), at least 2 sensors tracking the user and in some areas more. To each sensor's retuned distance value we add an offset. This way we are taking into account the width of the user's body (assuming his/her shape to be like a circle). By doing this, the points of intersection should, theoretically, be all coincident, representing the centre of the user's body (this is shown in Figure 22). After calculating these points we simply average them and the obtained user estimated position is then introduced into the output filter.

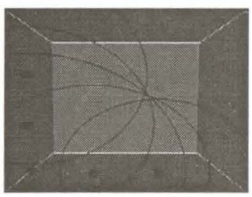

Figure 22. 10 approximately coincident points (circle intersections)

The output filter is applied to the user's computed position, smoothing it through time. The development of this filter started from the premise that we wanted a smooth tracked user movement in our main rendering window. On the other hand, we already knew that the frequency of the sensors radius update algorithm was much lower (since the values from the OOPic were being transmitted only at 3 to 4 times per second) and, consequently, so was the calculated user's position. So, we have realised that we needed a way to interpolate the user's position, at the time of updating the user viewpoint in the main rendering window.

In Figure 23 we see two calculated user positions (red squares 1 and 2) which means that the user is moving across the Arena in direction 1 to 2 (the distance is exaggerated for clarity). The pink dashed line is the interpolated positions that we want to determine. If we take the ratio of the elapsed time between successive window update rendering calls and the elapsed time between successive user position update positions (that has the same frequency than the OOPic transmission frequency), we get the time step that we need. This step, times the vector distance between position 1 and 2, gives us the user increment position that we need across each frame. The filter worked quite well. The user's position is smoothly interpolated and the movement is quite fluid. There is a small delay involved in this algorithm but it is hardly noticeable.

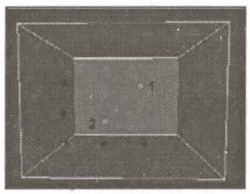

Figure 23. Output filter: position interpolation

As a conclusion, the sequential values returned from the sensors are much more stabilised compared to the previous implemented structure, but occasionally specific and localised errors occur. These errors are minimised by means of software filters. Despite of the limitation in not determining the complete 3D position of a user, this structure provides a great advantage relative to the other ones: The determined position (2D) is reliable, correct, and much more stable, which fulfils one of the main objectives of the Arena, of determining correctly the user position within its space.

5. THE MX TOOLKIT SYSTEM DEVELOPMENT KIT

The MX Toolkit, is an object-oriented programming Toolkit developed by our team and oriented to the mixed reality applications developer. It can manage the Arena or be independent of it. MX Toolkit is defined at a somewhat higher abstraction level than, for example, the AR Toolkit software layer, by hiding from the programmer, low level implementation details and facilitating AR/MR object-oriented programming. This platform comprises a software development kit (SDK) for the Windows environment, consisting of a set of C++ Classes packaged into modules. MX Toolkit which uses extensively the AR Toolkit library, for all matters regarding marker-based tracking, but also has a simple dialog with Microsoft Foundation Classes and with third party file formats, such as Open VRML, 3D Studio and, with scene graphs, such as OpenSceneGraph.

The MX Toolkit comprises eight functionally independent modules (the Speech module is under development), communicating with each other through the main system **Core**. All these modules will interact either directly or indirectly with the Arena Hardware and with the Wearable Unit:

Core System: This module provides an efficient mixed reality programming environment, supporting applications, scenes and sounds plug-in management, that are controlled by a set of three galleries: *Sound Gallery*: supports Sound Engine platforms, such as Open Audio Library and Microsoft Direct Sound and *Sound* File Plug-ins, such as Windows Wave and MPEG Layer 3; *Application Gallery*: MX Toolkit applications, such as Helicopter or Pairs Game [Dias 2003], are developed as DLL plug-ins[1], that can be built and later loaded, started, up-

[1] The MX Toolkit, as a whole, is also available for the AR/MR application programmer as a DLL.

dated and terminated at run-time by the Core Module;
Shape Gallery: Allows the system to load DLL Shape or Scene Graph Plug-ins (3DS and VRML 97);

The Core contains also a set of base class descriptors that can be used by external plug-in applications, such as the Abstract Augmented Reality Object, Augmented Reality Markers, Abstract Shape and Sound Plug-ins, Tangible Interfaces, Magic Books and even specific Hardware Sensors and Hardware Interfaces of the Arena..

Peer-to-Peer Communication: This module can serve applications to stream data or communicate in a higher abstraction level, hiding the application from the TCP/IP layer and communication protocol. This allows the creation of an interactive mixed reality network, where all nodes can share information between each other and use this information to augment its environment. By adding to this model a set of features provided by the Marker Tracking Module, which uses the AR Toolkit subsystem to track fiducial markers in an image, we have created a multi-camera tracking system, solving problems like marker occlusion and object registering precision.

Peripheral Communication: This module allows the system to communicate, in a hardware independent way, with the Wearable System hardware and with other specific Arena hardware, by a serial RS232 communication port. Communication can by synchronous or asynchronous.

Database Communication: This module provides a set of methods to enable database communication. It accesses data through a Data Access Object (DAO) based on the Microsoft Jet database engine. This engine provides the access to data, including through ODBC drivers, via a specific database engine. It also supports Data Definition Language (DDL) operations, such as adding tables and other modifying operations..

Video Capture: This module allows the system to acquire images from any WDM (Windows Device Manager) or VFW (Video for Windows) compatible capturing device, like cameras, VCRs, DVD players, etc. The user can select the input video device, the video input video resolution and the output window resolution, possibility correcting the image rendering, vertically and horizontally, which will be delivered to the Rendering Module.

Marker Tracking and Multi-modal Interfaces: This module includes AR Toolkit and a Texture Tracking object (not available in AR Toolkit). It provides a set of methods related to marker and texture tracking (integration under way), creates associations between 3D shapes and 2D markers or textures and handles Gesture and Tangible Interfaces, placing the programming user at a higher (object-oriented) level of abstraction. The Core system calls a method from this module, every time a frame is being processed, obtaining information and confidence values about the tracked markers or textures and the current camera transformation matrix.

Image Processing: This module provides a set of functionalities to process video streamed images that are delivered by the Video Capture Module. One of its features is the compression of live streamed video. It uses any type of compression compatible with the Windows Video Compression Manager. This functionality may be used in applications that require video streaming in conjunction with the Communication Module, due to the low bit-rates that can be achieved with this module. Another functionality of this module is the ability of applying one of various image filters in a single pass, or in various passes with a given sequence: Flip Image, Brighten Image, Adaptive Contrast, Threshold Image and Adaptive Image Enhance. This module also includes an algorithm that determines the best threshold value to binarise the image, prior to the marker-based tracking technique of AR Toolkit. The algorithm is based in computing the average of the luminance of image regions where markers were found, and from there, estimating a threshold value.

3D Rendering: Objects are rendered by this module independently of their geometry and topology. When creating an object to be rendered, it's geometrical and topological structures must be defined as well as its rendering methods (currently using the OpenGL API). This module also features a set of utilities like Menus, Menu Bars and Text display methods, allowing the user to create, in a simple way, an Augmented Graphical User Interface (AGUI). The AGUI is recognized by the Core and can be controlled by a Tangible Interface just like a simple mouse cursor.

6. RESULTS AND DISCUTION

6.1 User Tracking System Evaluation Results

In order to assess the 2D tracking accuracy in the X and Y-axis we've developed a simple test procedure. By putting known marks on the floor, spaced by 60 cm in both directions, we've simulated user motion and have compared the real position with the resulting system response. A relative error was evaluated, in each dimension, resulting in an error map graphically represented in Figure 24. We can see that, as far as the user moves away from point 0.0 the relative error increases dramatically. This is due to the ranging limitations of the sensors, to the fact that its accuracy is inversely proportional to the range distance and to limitations on the covered area offered by each sensor. In fact, at higher distances from the origin of the Arena reference frame, we obtain the lowest precision and these areas revealed to be the ones with lower coverage by the

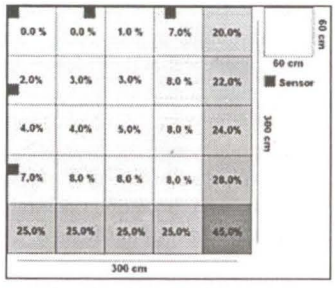

Figure 24. User Position Tracking Algorithm Error Map

sensors. However, we can notice that the Arena has a satisfactory position tracking accuracy, within an area of 180 cm by 180 cm. In what concerns to the Inertia Cube2 (a COTS system), there was no need to make any precision or performance evaluation tests, since its specifications are known [Inertia Cube]. It revealed to be for all due purposes 100% accurate as well as efficient, with an unnoticed latency time, achieving a real-time performance.

6.2 Tests and Results of MX Toolkit and Arena Usage

Several demonstrators and applications have been developed by our team using the Arena Space and/or the MX Toolkit. Here we name just a few: **Helicopter** (Figure 7): A user is placed inside the Arena Space, and a virtual 3D animated helicopter is placed in the real world, whose take off, landing and pose are controlled by a second user. Supports, 3D spatial sound – "mp3" and "3ds"/"wrl" 3D file format; **Mix It and AR Viewer**: Mix It is a tool to author mixed reality worlds using a visual interface, which adopts the metaphor of the Magic Book [Billinghurst 2001]. The Mix It supports multiple Magic Books, which can be created, managed, changed, deleted and printed by the user. The AR Viewer (Figure 3, Left) is the run-time module that enables the user to view and interact within the Arena mixed reality environment; **DIE-HEART Visualisation**: In this application we have applied Tangible Augmented Reality to interactive visualization scenarios in the medical field of cardiology, such as the teaching of medical techniques or the recognition of organs. Moreover the technique can be used for a more detailed and natural interaction with 3D and 4D virtual models of organs, such as the Left Ventricle of the Heart or other physiological structures of the human body.

7. CONCLUSIONS AND FUTURE DIRECTIONS

In this paper we have described the realization of the Arena, an indoor space for mobile mixed reality interaction. The Arena, includes a new user hybrid tracking system appropriate for AR/MR applications and a new object-oriented Toolkit appropriate to the augmented and mixed reality applications developer, the MX Toolkit. The paper discusses in some detail the user tracking technique, the Arena Space topology, the Wearable Concept and the Arena Space specific hardware. It addresses also the issues of the MX Toolkit SDK. We came up with a global solution, whose hardware, including computers, is priced well below existing commercial alternatives. The evaluation results have shown that the Arena has a satisfactory position tracking accuracy, within an area of 180 *cm* by 180 *cm*. Since the availability of the Arena and the MX Toolkit in our HP New Media Lab, more than 10 AR/MR applications have been developed or are under way in this environment, which shows the interest of this infrastructure for in-house AR/MR research and in collaboration with other research groups, namely with the Multi-modal Human-Computer Interfaces group of INESC, Portugal, headed by Prof. Joaquim Jorge. Regarding future developments that are in our R&D roadmap, we can name a few: Tracking the complete 3D user position with more accuracy is a priority. An investigation on other types of sensory devices will be sought (laser sensors, new ultra-sound sensors, infrared cameras); The Arena Hardware, can also be improved. Faster microcontrollers, faster buses or new microprocessors are future options. Future developments in the MX Toolkit SDK are envisaged, in: (A) Multi-modal interaction: Two hand gesture recognition in static and dynamic poses (ongoing); Speech recognition: English is already integrated; Portuguese could be a path to follow in collaboration with INESC; Sketch based interaction in MR (in collaboration with INESC [Fonseca 2000]); Face Recognition; Biometrics/affective computing interaction; New tangible interfaces. (B) Tracking and reconstruction: Texture tracking (on going); Object tracking by vision; Marker less tracking; Vision-based 3D reconstruction.

8. ACKNOWLEDGEMENTS

The authors would like to thank Rui Silvestre and António Lopes, (ADETTI, Portugal), for their contribution to the Arena/MX Toolkit.

9. REFERENCES

[Bastos 2004] Bastos, R., Dias, J. M. S., "Fully Automated Texture Tracking Based on Natural Features Extraction and Template Matching", submitted to ISMAR 04, 3rd IEEE and ACM International Symposium on Mixed and Augmented Reality, in Arlington, VA, USA, Nov. 2-5, 2004

[Billinghurst 2001] Billinghurst, M., Kato, H., Poupyrev, I., "The MagicBook: A Transitional AR Interface", in Augmented Reality: the Interface is Everywhere, SIGGRAPH 2001 Course Notes 27, 2001

[Dias 2003] Dias, J., M., S., Jorge, J., Carvalho, J., Santos, P., Luzio, J., "Developing and Authoring Mixed Reality with MX Toolkit", ART03, The Second IEEE International Augmented Reality Toolkit Workshop, Tokio, Japan, 6th October 2003

[Dias 2004a] Dias, J.M.S. Santos, P., Bastos, R., "Gesturing with Tangible Interfaces for Mixed Reality", in Camurri, A., Volpe, G., ed. *Gesture-Based Communication in Human-Computer Interaction*, ISBN 3-540-21072-5 Springer Verlag, 2004

[Dias 2004b] Dias, J.M.S., Nande, P. Barata, N, Correia, N, "Understanding Portuguese Sign Language by Automatic Hand Gesture Recognition", submitted to ASSETS 2004, 18-20 October, Atlanta, USA.

[Grimm 2002] Grimm, P., et al., "AMIRE, Authoring Mixed Reality", ART02, The First IEEE International Augmented Reality Toolkit Workshop, Darmstadt Germany, 29th September 2002

[Fonseca 2000] Fonseca, M., J., Jorge, J., "CALI : A Software Library for Calligraphic Interfaces", 9º EPCG, 2000

[I2C] www.semiconductors.philips.com/buses/i2c/

[Inertia Cube] www.sp-vsense.com/inertiacube.htm

[Kato 2001] Kato, H., "Developing Applications with ARToolkit", SIGGRAPH 2001 Course Notes 27, 2001

[OOPIC] www.oopic.com

[openvrml] www.openvrml.org

[openscenegraph] www.openscenegraph.org

[SRF08] www.robot electronics.co.uk/shop/Ultrasonic_Ranger_SRF082001.htm

[Uchiyama 2002] Uchiyama, S., et al., "MR Platform: A Basic Body on Which Mixed Reality Applications Are Built", ISMAR 2002, IEEE and ACM International Symposium on Mixed and Augmented Reality, Darmstadt Germany, Sept. 30th - Oct. 1st, 2002