Augmented Reality behind the Wheel

- Human Interactive Assistance by Mobile Robots

Nuno Costa
IST/UTL & YDreamsRobotics
Portugal
nuno.costa@tecnico.ulisboa.pt

Artur Arsenio
IST-ID, Universidade da Beira Interior & YDreamsRobotics
Portugal
artur.arsenio@tecnico.ulisboa.pt

***Abstract*—Novel approaches have taken Augmented Reality (AR) beyond traditional body-worn or hand-held displays, leading to the creation of a new branch of AR: Spatial Augmented Reality (SAR) providing additional application areas. SAR is a rapidly emerging field that uses digital projectors to render virtual objects onto 3D objects in the real space. When mounting digital projectors on robots, this collaboration paves the way for unique Human-Robot Interactions (HRI) that otherwise would not be possible. Adding to robots the capability of projecting interactive Augmented Reality content enables new forms of interactions between humans, robots, and virtual objects, enabling new applications. In this work it is investigated the use of SAR techniques on mobile robots for better enabling this to interact in the future with elderly or injured people during rehabilitation, or with children in the pediatric ward of a hospital.**

Keywords-Spatial Augmented Reality, Human-Robot Interaction, Augmented Reality, Projective Distortion, Mobile Robots, Robot Localization

I. INTRODUCTION

Robotic applications for health care have sky rocketed over the past decade [1]. Not only have surgical robots seen great improvements, but robotic solutions are also being applied to a greater number of other scenarios, such as human rehabilitation, patient care, assisted living and social therapy. Human–Robot Interaction (HRI) is a field of study dedicated to understanding, designing, and evaluating robotic systems for use by or with humans [2]. Interaction can take several forms, being voice, vision, and gestures the most common ones.

In this work, we will consider two different environments:

- Augmented Human Assistance environment (for work being developed on the scope of a CMU-Portuguese collaborative research program) aiming at patient rehabilitation after injury, allowing the execution of exercises on remote locations without requiring the patient to move into a clinical health provider.
- The pediatric ward on a hospital, the Portuguese Oncology Institute of Lisbon (IPOL) for work being developed in the scope of the MOnarCH project (Multi-Robot Cognitive Systems Operating in Hospitals), focused on introducing a fleet of social robots that will interact with sick children and collaborate with medical staff.

There are several scenarios in which projected AR content is very useful in these two aforementioned environments:

- Teaching, for robots to supporting human teachers by projecting augmented reality content on a wall or on objects.
- Patient rehabilitation exercises support, by projecting augmented reality content during physiotherapeutic activities in which the patient receives in real-time visual feedback and corrective postures
- Information providers, such as projection of AR content informing people that a new activity is about to start, or calling then with visual signs, or even moving along with people to places where action is going to take place (e.g. using projected directional arrows), or informing someone to stop performing an exercise
- People protection, such as projecting augmented reality content (for instance a stop sign) if a person moves into a forbidden zone or door, or performs a forbidden rehabilitation exercise or movement
- People entertainment: robots can play games with children, according to content projected into the floor. In another scenario, a patient rehabilitation can involve game playing (serious games).

We aim to further improve existing HRI by adding to robots a unique capability: SAR, to allow the creation of unique interactive AR scenarios as just mentioned, that otherwise would not be possible. This projection technology is used to turn objects, often irregularly shaped, into a display surface for video projection. Using specialized software, a 2 or 3-dimensional object is spatially mapped on the virtual program, mimicking the real environment, which will be projected upon. The software can interact with a projector to fit any desired image onto the (planar) surface(s) of that object. The SAR solution employs Unity game engine for the correction of the perspective distortion problem (in a virtual world). This problem is solved by applying virtual planar surfaces for projection of AR content onto real world surfaces. An algorithm is applied onto each surface texture so that the perspective distortion caused by the surface's position and pose relative to the projector will be compensated (employing camera-projector calibration), leading to a distort less image.

- This work has been partially funded by CMU-Portuguese program through Fundação para Ciência e Tecnologia, project AHA-Augmented Human Assistance, AHA, CMUP-ERI/HCI/0046/2013.
- Part of this work was also developed in the scope of the Monarch project: Multi-Robot Cognitive Systems Operating in Hospitals, FP7-ICT-2011-9-601033, supported by EU funds.

978-1-4799-6466-6/15/$31.00 ©2015 IEEE

The end goal is to have a robot projecting interactive AR on multiple surfaces, whereas being idle or moving, performing the necessary compensations for each surface being projected. And additionally, reacting to human interactions, or other real objects interactions (e.g. a virtual object reaching a wall corner and bouncing back due to this physical constraint), with the virtual content. This way, projected AR content is undistorted, regardless of the robot's position and orientation.

II. Background

This section reviews SAR work, both on fixed and on mobile platforms, as well as previous work on virtual and real content interactions. Most of the SAR research is inspired by Shader Lamps [3]. The purpose of this work was to augment an existing blank model with computer-generated graphics to make the model exhibit realistic lighting, shadow and even animation effects, thus providing it with characteristics previously inexistent. Dynamic Shader Lamps [4] allows users to interactively change the generated graphics, and to digitally paint onto objects with a stylus.

There are various SAR projects with different applications. iLamp [5] supports SAR by casting the pre-distorted image of virtual objects from the projector, which can capture as well the 3D environment based on structured patterns. An interactive SAR system is described in [6] with multiple depth cameras and a projector. The system detects the information of the surrounding environment along with the users motion through depth cameras, and displays images on the surface objects in a planar, real world table for the user to interact with. LuminAR [7] is a robot introduced for a desktop environment, which has a robotic arm with a projector and a portable-sized camera to interact with the user. Pixelflex [8] adopted an array of multiple projectors to make a huge screen space. Automatically self-configured projectors are able to create a huge projection area.

A projection-based information display system was proposed in [9], which displays virtual images in a room with a series of sensors. The robot tracks the user viewpoint with the sensors, and then the robot can generate anamorphic (i.e., projection format in which a distorted image is *stretched* by an anamorphic projection lens to recreate the original aspect on the viewing screen), properly distorted images for users on flat-surfaces. WARP [10] allows designers to preview materials and finished products by projecting them onto rapid prototype models. The system uses a standard graphical interface, with a keyboard and mouse used for user interface. The projection is not restricted to a fixed area; all feedback is projected onto the tools. Surface Drawing [11] allows a user to sculpt 3D shapes using their hands. Spray modeling [12] uses a mock-up of a physical airbrush to allow the user to sculpt 3D models by *spraying* matter into a base mesh. The system enhances physical tools by projecting status information onto them allowing the overload of a single tool with several functions.

III. Camera-Projector Calibration

It is essential to know the projector's intrinsic and extrinsic properties, so it becomes necessary to go through a camera-projector calibration process, to prevent the naive projection of distorted images of virtual objects. Both camera image capture, as well as projector's image projection, are generally described by a standard pinhole camera model with intrinsic parameters including focal length, principle point, pixel skew factor and pixel size. Furthermore, calibration also estimates the extrinsic parameters (that include rotation and translation) from a world coordinate system to a camera or projector coordinate system.

Camera calibration is just the first step of a longer process that aims at obtaining the projector position and pose (obtained from its extrinsic matrix) with respect to a fiduciary marker placed in a specific position in the real world. This information will then be exported onto Unity, placing its virtual camera on the estimated position of the real world projector. Once the camera's intrinsic is known from calibration, it becomes possible to execute steps 1 and 2 from Fig. 1a, if we know as well the extrinsic matrix of the camera mapping the world referential to the camera's homogeneous referential. For achieving this goal, it is used a modified version of ArUco application, a minimal library for Augmented Reality applications based on OpenCV library. Among other features, this application can determine a marker's position and pose relative to the camera (step 1), represented by a translation vector and a rotation matrix. The use of this application can be seen in Fig. 1b. Step 2 will now follow by merging these results into a single matrix and applying its inverse. In other words, this step allows determining the position of the camera relative to the fiduciary coordinate system.

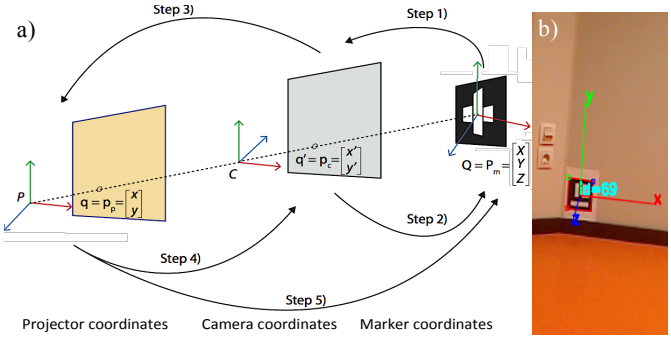

Figure 1: a) Steps taken to find the projector position in relation to a fiduciary marker coordinate system represented by step 5. The point P_m on the fiduciary marker is seen as point p_c on the camera image plane, which in turn is seen as point p_p on the projector image plane. b) Modified version of ArUco: the camera position and pose are determined with respect to a marker on the wall.

Steps 3 and 4 are achieved by projector calibration, for which the software developed makes use of the Matlab's camera calibration toolbox. The core concept is to consider the projector as an inverse camera. By doing so, it is possible to make use of any standard camera calibration procedure in order to calibrate the projector. The main challenge is to find the 3D points of the projected pattern in order to use them together with the 2D points of the image being projected to finally obtain the intrinsic and extrinsic parameters of the projector.

An image containing a chessboard pattern is projected, generating n views. After calibrating the camera, the later captures the chessboard image. A corner extraction algorithm [13] is then applied, generating a set of $n \times k$ 2D points. The correspondent 3D points can now be estimated, using the camera's intrinsic and extrinsic matrix. It is now possible to establish a correspondence between these $n \times k$ 3D points and the 2D coordinates, obtaining the intrinsic projector matrix as

well as all the *n* extrinsic matrices generated [14]. Step 5 can then be achieved by multiplying these two matrices. Hence, the resulting matrix describes projector's position and pose with respect to the marker coordinate system.

IV. Architecture Implementation

The architecture of the system developed, represented in Fig. 2, is comprised of six main categories:

- The hardware includes a projector, a camera, and a Kinect depth camera for human-robot interactions;
- The Kinect SDK interface, which allows direct access to Kinects RGB and depth sensors. One of Kinects' key features we used was skeleton tracking;
- Unity wrapper, which enables the use of Kinect SDK internal functions within Unity;
- A Windows Presentation Foundation application (WPF) was developed in order to simulate positional information retrieved from a robot localization system;
- Unity, the game engine on top of which the main applications were developed;
- The camera-projector calibration application, which makes use of ProCamCalib, developed on Matlab.

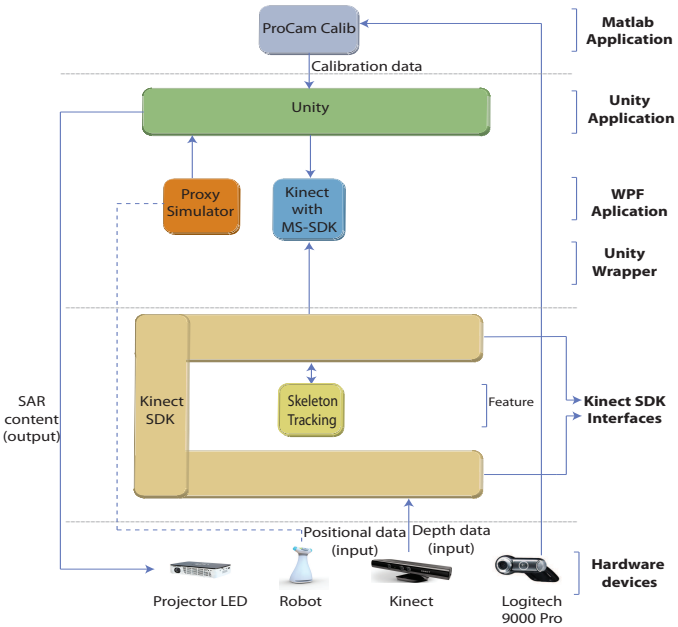

Figure 2: Architecture of the proposed solution. The line in dash represents the connection to the mobile robot localization system.

A. Homography

Shapes are distorted under perspective imaging: when projecting an image through a projector with referential aligned with that of the projection surface, the image appears undistorted. However, when placing the projector in a non-aligned position, the same image will appear distorted.

A homography was used to remove the perspective distortion effect from a perspective (planar) image. It is an invertible mapping of points and lines on the projective plane P^2 into itself [15]. It is applied to one or more image plane(s) in a virtual scene within Unity. It compensates for the distortion that occurs whenever the camera is on a non-parallel position in relation to the image projection plane(s). The application on Unity's camera of the projection matrix makes it assume the parameters and role of a real world projector. So, in reality it becomes possible to compensate for the distortion caused by a projector with a homogeneous referential not aligned with one or more real world projection surface(s).

A distorted image of a plane can be considered as a projective distortion of the original one. By computing the inverse transformation and applying it to the image, it becomes possible to undo this projective transformation. The result will be a new synthesized image in which the objects in the plane are shown with their correct geometric shape [16].

B. Unity Camera Model

There is no use in applying perspective correction to an image while using a virtual camera with its own internal parameters that massively differ from the ones used by a real world lens (that will be used to project that very same image onto the real world), since it will lead to a distorted and out of place image. In order to fix this, it is necessary to apply the results obtained from the camera-projector calibration process.

The extrinsic matrix can be almost directly applied to the Unity's camera since it already has its own transform component containing the position and pose of the camera. Modeling in Unity the intrinsic matrix is not straightforward since it implies adapting Unity's projection matrix model to the calibration intrinsic matrix model. Unity is based on OpenGL in which one can specify the camera shape (which in turn determines the projection matrix). There are two ways to accomplish this, as described hereafter.

Indirectly, by specifying the frustum using unity's default method *gluPerspective(fov, aspect, near, far)*. This method specifies a viewing frustum into the world coordinate system using a projection matrix, but this particular OpenGL method is useless due to the fact that our model for the intrinsic matrix of a real world camera contains five degrees of freedom (represented as two focal values, a principal point and a axis skew) whilst the projection matrix provided by *gluPerspective* only has two (field-of-view and aspect ratio).

OpenGL offers as well an alternative solution consisting of directly specifying the frustum using *glFrustum(left, right, bottom, top, near, far)* method, which can be adapted to allow the use of the intrinsic parameters of a lens. This method defines the clipping space by specifying the coordinates for the horizontal, vertical and distance coordinates. Upon defining this new projection matrix, we can then replace Unity's default by simply replacing it through a *C#* script.

C. Solution Modules

The developed application is composed by a collection of modules (given by a series of scripts in Unity), each designed to perform a specific task:

- Homography matrix update: updates the matrix associated with each projection scene and apply that transformation in OpenGL's vertex pipeline;

- Tracking: Controlling each projection surface position in the virtual scene;
- Intrinsic Setting: Alters Unity's default projection matrix so that the values obtained from the projection calibration process can be properly applied;
- Update Unity's camera position: based on localization (position/pose) information provided by an exterior application (on the robot or elsewhere);
- Save and load: of projection surface positional data from an XML file;
- Human-Robot Virtual Interface (HRVI): Updates the game logic of an interactive AR game based on input received from Kinect's skeleton tracking information.

D. Source and Destination Points

Each virtual projection surface defined in Unity (which in turn, will match a real world projection surface) is represented as a parent-child hierarchy of game objects that can be seen in Fig. 3. The sole purpose of a Projection Surface is to store all the game objects related with each other within one projection surface game object parent. The plane represents the projection surface on which content will be displayed. It corresponds to a real world projection surface (a wall, for instance). Source and Destination points store a set of four elements. Each set will serve as the input for the homography matrix calculation.

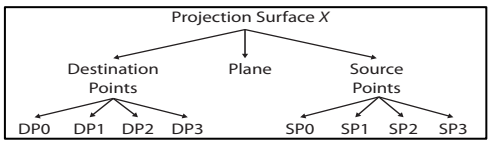

Figure 3: Hierarchical representation (parent-child) of a Projection Surface game object. Each element represents a unique game object in the Unity scene.

The computation of the projective transformation using point-to-point correspondence gives linear equations in the elements of H [15]. So four point correspondences are the minimum required to fulfill the eight linear equations to solve for H elements, constrained to not existing three collinear points. Points correspondence takes place in Unity by creating a set of N source points and a set of N destination points. If using $N > 4$ points, the problem of estimating H corresponds to the problem of determining the pseudo-inverse matrix of a (8 x 2N) matrix. This is equivalent to estimate H from n points in order for minimizing the mean square error. The inverse of the transformation H computed this way is then applied to the whole image to undo the effect of perspective distortion on the selected plane. The estimation process itself is straightforward, after which the Gaussian elimination algorithm (also known as reduction) is applied to solve this linear equation system.

E. Projection Surfaces

A shader is applied to each projection surface (built as a plane Game Object that has its own mesh) meaning each vertex composing the surface mesh will be affected by any operation defined within that very same shader. So, the resulting matrix is applied to the projection surface plane, through a vertex shader [16]. A Unity function is used to transfer homography values, estimated on runtime, directly into the shader. This corresponds to the insertion of the homography transformation matrix directly into the vertex transformation pipeline.

The vertex shader pipeline for each projection surface is shown in Fig. 4. The application was designed so that there can be multiple (up to nine) projection surfaces present in the scene, each with its very own homography matrix. This translates into being able to project onto the real world over nine distinct surfaces at the same time, each with its own position and pose relative to the projector.

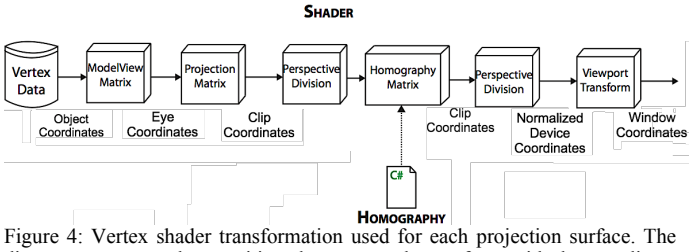

Figure 4: Vertex shader transformation used for each projection surface. The diagram represents the transitions between each transform with the coordinate space used for vertex positions along transforms.

F. Interactive AR Content from Human-Robot Interface

Our goal is to project augmented reality content according to objects physics, or other entities. As such, we introduce the human interactions feedback in order to adapt the augmented reality content, and to achieve virtual-real objects integrated behavior, as represented in Fig. 5. Hence, the application integrates augmented reality information, with projection distortion compensation, and human gestures recognition to enable interactions between people and undistorted augmented reality content. Virtual content is projected onto the real world where humans interact with such content through movements, as detected by Microsoft Kinect's skeleton tracking feature.

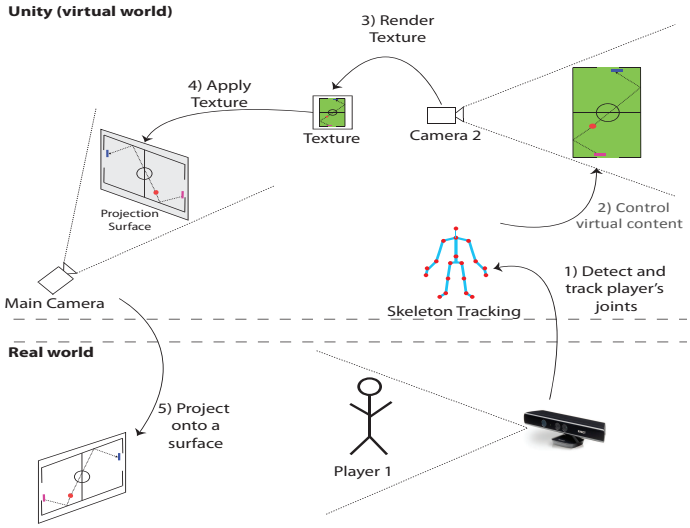

Figure 5: Human interactions. One or two humans are tracked simultaneously using Kinect's skeleton tracking (step 1). The corresponding joint, for each player, controls the respective virtual object (step 2). Camera 2 renders the texture containing the view of the camera (step 3) that is then applied to one of the projection surfaces and updated in runtime (step 4). The final step can now take place: project onto one real world surface (step 5).

G. Matching Virtual Camera position with Projector's

For testing purposes, the application receives location data from an exterior application containing the robot's positional information, developed in WPF (Windows Presentation Foundation) to simulate the update of Unity's virtual camera position based on the output of an exterior application sending

a message containing position and rotation values. The communication process between the application and Unity is done via a network socket, allowing different kind of platforms, besides robots, to provide localization information.

V. Evaluation

This section presents the experimental results of the projection of AR content onto the real world.

A. Scenario Setup

As seen in Fig. 6, each projection surface was placed in a different angle, represented by β, with respect to the wall (and consequently, to the projector as well).

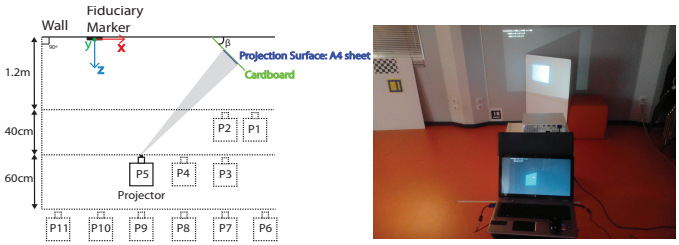

Figure 6: Experimental setup: a projection cardboard is at a β angle relative to the projector. AR content is then projected onto that plane. Measurements then take place for each corner of the plane and its corresponding corner on the projected content. This process is repeated 11 times for each β angle (the chosen angles were 47º and 60º). The world coordinate system is given by a fiduciary marker placed in a known position in the real world.

System performance is evaluated by projecting AR content onto small projection surfaces at two distinct angles. For each angle, the projection will be tested at three distances from the wall, for a total of 11 measurements (4 points each) per angle. The projection offset is measured for each corner of the sheet. This process will then be repeated for each position, represented by $P_{1...11}$ in Fig. 6. The entire experimental procedure workflow can be seen in Fig. 7.

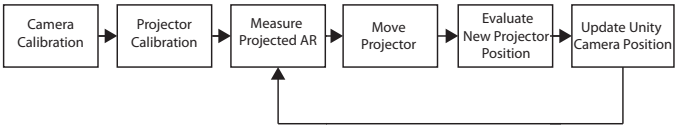

Figure 7: Experimental process workflow.

Localization information is obtained from the modified version of ArUco. Since the camera is placed on top of the projector, both camera and projector have synchronized motion, so by measuring the camera location, it is possible to determine the projector displacement as well.

B. Projector Calibration

Experimental evaluation of the projector calibration process is based on the corner re-projections over the computed images. The error of the projection calibration will obviously depend in the camera calibration errors. The error will also depend not only on positioning accuracy of the experimental setup, but also on the corner extraction algorithm. Additionally, just like with camera calibration, special care has to be taken in this step and exclude outliers from the calibration. The calibration procedure (Fig. 8a) was based on 18 images containing 2 chessboards, a printed one and a projected one.

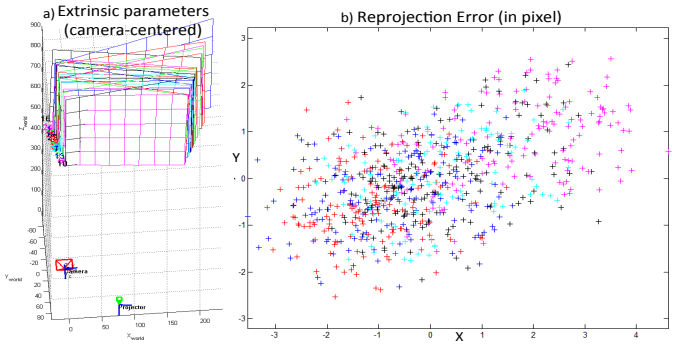

Figure 8: a) the result of camera-projector calibration procedure displaying the projector position in relation with the camera coordinate system; the camera is represented in red, large square, projector in green small square, and each grid represents a chessboard pattern with unique position and pose with respect to both camera and projector; b) Each dot represents the reprojection error (from a pixel point of view) of a feature from the chessboard. The closer they are to origin (center of image), the smaller will be the error.

One of the issues that initially had a significant impact in the corner extraction was the lighting condition, not only in the room, but also for the image being displayed and the camera brightness parameters. Even using average brightness values on both camera and projector, the images obtained were defective: the corners of the chessboard being projected were not clear and as a result the square corners of the chessboard appeared to be apart from each other, which obviously affected the calibration process in quite a negative way.

The best projector calibration resulted in a mean reprojection error of 1.54 pixel, a value that is 50% higher than the expected [14]. It is very likely that this difference is at least partially due to the quality of the camera itself. Since the projector calibration process relies on the image quality obtained from each printed/projected chessboard it inevitably affects the corner extraction during the calibration process. The results can be seen in Fig. 8b. In a standard camera lens, it is expected to have a near-center principal point. In the case of a projector most often that is not the case. Most projectors have a significant vertical displacement so c_y ends up having a higher value than expected. In this case, with a 1024x768 resolution, the obtained c_y value was 696.48, far higher than it would be if it was near-center point (in which case c_y would be close to 384). There is also an unexpected horizontal displacement, as the measured 670.88 differed from the expected 512.

After projector calibration, we tested the AR content projection accuracy onto the real world. The (world) coordinates of the four corners of the projection plane were set as the destination points in the Unity application and its camera was set to the (projector) extrinsic parameters previously calculated. The content was then projected and the offset of the projection for two different β angles was measured and analysed, as described in the following two sections.

C. AR Projection – Scenario 1

In this section the projection surface depicted in Fig. 6 is set at a moderate angle β of 47º. The results for the 44 measured points are shown in Table 1, and visualized in Fig. 9a. There is a significant vertical offset for all projection points. They tend to be in a lower position when compared to the respective target corner. This offset could be caused by any of the two

calibration processes, or even by the mapping between the calibration projector and the unity camera models.

TABLE I. PROJECTION RESULTS FOR ALL SETS OF 4 SURFACE CORNERS.

Experiment 1				
Coordinate point (x,y)	Topleft	Topright	Bottomleft	Bottomright
Physical(measured)	(51.1,73.7)	(71.3,73.7)	(51.1,52.6)	(71.3,52.6)
Projected (median)	(51.6,72)	(71 , 72.7)	(51.4,51.7)	(70.8,51.7)
Projected (std)	(0.53,1)	(0.22,1.1)	(0.53,1.23)	(0.20,1.25)
Error (median)	(0.5,-1.7)	(-0.3,-1)	(0.3,-0.9)	(-0.5,-0.9)
Experiment 2				
Coordinate point (x,y)	Topleft	Topright	Bottomleft	Bottomright
Physical(measured)	(59.3,73.7)	(74.15,73.7)	(59.3,52.6)	(74.15,52.6)
Projected (median)	(61.1,72.3)	(72.65,72.9)	(61.1,51.52)	(72.25,52.2)
Projected (std)	(0.89,1.03)	(0.61,1.18)	(0.99,1.16)	(0.61,1.28)
Error (median)	(1.8,-1.4)	(-1.5,-0.8)	(1.8,-1.08)	(-1.9,-0.4)

Ideally, the colored markers would be all around the respective target corner. However, that is not the case. The dots instead appear, for all distances, in an inwards position (relative to each corner) indicating that there is a horizontal scale offset affecting the projection. This off-set is most likely due to errors on the camera and projector calibration since a scale factor also is taken into account when determining the focal values f_x and f_y, previously through the scale factors present in a camera's intrinsic matrix.

It is interesting to notice that, looking at Fig. 9a, the projection points taken at each distance, all stay within close proximity of each other, meaning the projection results do not vary significantly despite the wide distance range (1.2 to 2.2m) used for extracting measurements.

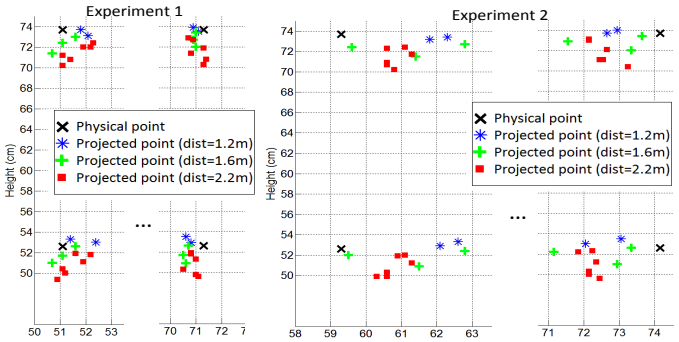

Figure 9: Projection results obtained with the projection surface at an angle of 47º and 60º relatively to the projector, for experiments 1 and 2, respectively. Each X represents a corner of the target projection surface. The remaining markers represent measurements taken at different distances: 1.2m, 1.6m and 2.2m represented by a star, a cross and a square, respectively. Values are measured in cm.

D. AR Projection – Scenario 2

The application was also tested on its limits by introducing a steeper angle: 60º, with the same testing procedure (Fig. 9b). Unlike previous experiments, the results obtained are far more scattered, meaning the systems calibration is not precise enough to deal with this level of projection difficulty. However, the analysis drawn for the last scenario is also applicable, as there is also a significant horizontal scale offset.

VI. HUMAN IN THE LOOP APPLICATION

An interactive AR game, Air Hockey, was developed as a way of showing a possible application for AR from human-robot interactions. In the real world, this game has two competing players trying to score points in the opposing player's goal, using a table having a special low-friction playing surface. So, we projected a virtual table onto the real world where each player controls his own virtual paddle by moving his/her hand up and down (left side player controls his paddle with his/her left hand, same logic goes for the right side player). Each player hand movement is detected by using Microsoft Kinect's skeleton tracking.

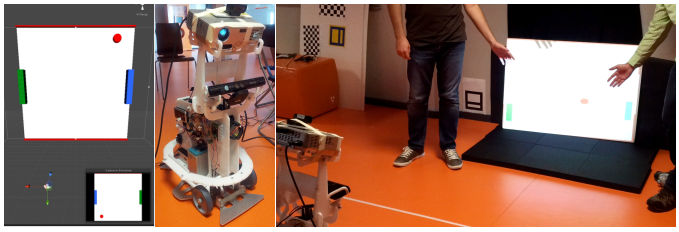

Figure 10: The game, the mobile robot, and people playing the air hockey game as projected by the robot on a non-aligned plane of projection.

Fig. 10 shows a visual representation of Air Hockey's game workflow inside Unity's virtual scene, and the projected game in a wall. The application runs at 70 frames per second.

VII. CONCLUSIONS

This paper proposed a SAR system specifically designed for deployment on mobile platforms, targeting interactions between virtual content and people in augmented human assistance scenarios. The developed system allows the creation of unique interactive AR scenarios that otherwise would not be possible. The perspective distortion caused by robot movement (position and pose), causing non-alignment between the projector and the projection plane, is compensated, leading to a distort-less image. The system is scalable, allowing simultaneous projection over nine distinct real world projection surfaces, each with its own position and orientation. We have shown the application of the system to a game application, involving interaction between virtual content and humans.

Currently, the system can be applied for projection of virtual content up to nine planar surfaces, applying the necessary distortion compensation in order to create a desired 3D effect. This large number of planes allows the application of the solution to various scenarios. However, it is also a system limitation, since it is not currently able to project upon surfaces with a non-planar geometry, such as spherical surfaces. Another aspect that needs improvement is the experimental evaluation with a moving robot at different speeds, and the correspondent determination of the projection error sensitivity to both robots' localization errors as well as to environment map errors (due to errors introduced by the automatic environment reconstruction algorithm or manual human measurements).

It would be also interesting to evaluate an alternative for the usage of a localization system, such as employing the depth sensor of the kinect camera to reconstruct in real-time the planar surfaces (or other higher order manifolds) used for projection.

Future work will address the application of this technology for the projection of feedback information to injured people during recovery therapeutic exercises. In addition, the presented solution sets the framework on top of which it is

possible to design new children-robot, or elderly-robot, interactive games besides Air Hockey.

REFERENCES

[1] G. Xing, G. Tian, H. Sun, W. Liu and H. Liu, "People-following system design for mobile robots using kinect sensor". 25th Chinese Control and Decision Conference (CCDC), 2013, pp. 3190–3194.

[2] A. Michael and A. Schultz, "Human-robot interaction: a survey". Foundations and Trends in Human-Computer Interaction, 2007.

[3] R. Raskara, R. Ziegler and T. Willwacher, "Cartoon dioramas in motion". Proceedings of the 2nd international symposium on non-photorealistic animation and rendering, pp. 7-13, 2002.

[4] D. Bandyopadhyay, R. Raskar and H. Fuchs. "Dynamic Shader Lamps: Painting on Movable Objects". IEEE/Acm international symposium on mixed and augmented reality, 2001.

[5] R. Raskar, J. van Baar, P. Beardsley, T. Willwacher, S. Rao and C. Forlines, "iLamps: Geometrically Aware and Self-Configuring Projectors". ACM Transactions on Graphics (TOG), ISSN: 0730-0301, Vol. 22, No. 3, pp. 809-818, July 2003.

[6] A. Wilson and H. Benko, "Combining Multiple Depth Cameras and Projectors for Interactions On, Above, and Between Surfaces". In Proceedings of ACM UIST '10. pp. 273-282, 2010.

[7] N. Linder and P. Maes. "LuminAR: portable robotic augmented reality interface design and prototype". Proceedings of UIST2010. NY, USA., Oct 2010.

[8] R. Yang, D. Gotz, J. Hensley, H. Towles and M. Brown, "Pixelflex: A reconfigurable multi-projector display system". Proceedings of the conference on Visualization'01, pp. 167-174, 2001.

[9] K. Azuma and S. Miyashita, "Anamorphosis Projection by Ubiquitous Display in Intelligent Space". Proc. int. conf. universal access in human-computer interaction, pp. 209–217, 2009.

[10] C. Harris, "Tracking with rigid models". In Active vision, pp. 59-73 MIT Press Cambridge, MA, USA, 1993.

[11] S. Schkolne, M. Pruett and P. Schröder, "Surface drawing: creating organic 3D shapes with the hand and tangible tools". Proc. of SIGCHI Conf. on Human Factors in Computing Systems, pp. 261–268, 2001.

[12] H. Jung, T. Nam, H. Lee and S. Han, "Spray modeling: Augmented reality based 3D modeling interface for intuitive and evolutionary form development". Proceedings International Conference on Artificial Reality and Telexistence, 2004.

[13] J. Bouguet. "Camera Calibration Toolbox for Matlab", Computational Vision at the California Institute of Technology, 2004.

[14] G Falcao, N Hurtos and J Massich, "Plane based calibration of a projector camera system". VIBOT master, 2008.

[15] R. Hartley and A. Zisserman. "Multiple View Geometry in Computer Vision". Cambridge University Press, 2003.

[16] K. Lammers. "Unity Shaders and Effects Cookbook". Packt Publishing, 2003.