An Interactive Augmented Reality Chess Game using Bare-Hand Pinch Gestures

Marios Bikos
University of Patras, Electrical and Computer Engineering Department
Patras-Rion 26504, Greece
Email: mariosbikos@computer.org

Yuta Itoh and Gudrun Klinker
Technical University of Munich
Department of Informatics
Garching b. Munchen 85748, Germany
Emails: itoh@in.tum.de, klinker@in.tum.de

Konstantinos Moustakas
University of Patras, Electrical and Computer Engineering Department
Patras-Rion 26504, Greece
Email: moustakas@ece.upatras.gr

***Abstract*—In order to produce realistic simulations and enhance immersion in augmented reality systems, solutions must not only present a realistic visual rendering of virtual objects, but also allow natural hand interactions. Most approaches capable of understanding user interaction with virtual content can often be restrictive or computationally expensive. To cope with these problems, we demonstrate a method which employs user's thumb and forefinger to interact with the virtual content in a natural way, utilizing a single RGB-D camera. Based on this method, we develop and realise an augmented reality chess game, focused on providing an immersive experience to users, so that they are able to manipulate virtual chess pieces seamlessly over a board of markers and play against a chess engine.**

I. Introduction

Realizing high-fidelity AR experiences is a coveted future we seek for. Among many factors that affect the degree of realism and immersiveness in such experiences, interaction with virtual objects plays an important role. A variety of interaction techniques have been suggested to manipulate virtual objects in AR simulations [1][2]. However, most of the approaches require either an excessive amount of computational complexity for tracking the hand posture or visual props attached to user's hand or fingers (e.g markers or special gloves). These factors can break the illusion of direct interaction with virtual objects in the real world.

In the last few years, several low-cost depth sensing devices have offered great opportunities to detect gestures, using computer vision and image processing techniques, allowing natural bare-hand interactions. In this paper, we propose a system which integrates a pinch gesture tracking technique within an augmented reality environment. The system offers a robust and seamless AR interaction by employing a modern, close-range RGB-D sensor, Intel® RealSense™ 3D camera. Our main aim is to optimize the real-time interaction between the user and virtual objects in a first person-view AR environment. The methodology proposed, allows the fast and effective translation of virtual objects in 3d space, selecting and moving them using pinch gestures. Its effectiveness is demonstrated through the development of an interactive augmented reality chess game. The evaluation results indicate that the method proposed, is precise and enjoyable to use.

The main contributions of our work are:

- Integrating a close-range RGB-D Sensor within an augmented reality environment and utilizing a pinch gesture detection technique to allow real-time 3D virtual object manipulation.
- Development of an interactive AR chess game, taking into consideration occlusion problems, so that users can seamlessly manipulate virtual chess pieces using their bare hands.

II. Related Work

An accurate vision-based tracking method for table-top AR environments has been suggested [3], that allows users to manipulate virtual objects in a natural and intuitive manner. Although this approach seems robust, a handheld paddle with a marker on it was considered essential, making the interaction unnatural. AR tracking targets, mounted on a glove have been proposed [4], so that users may directly manipulate virtual objects using hand gestures. Users employed pinching gestures to pick up objects and then positioned them simply by moving their hand. Still, users had to wear special gloves, so the system could only work when the tracking markers were in view of the systems camera. Interaction with virtual objects through pinch gestures has also been attempted using Google Glass [5]. As authors suggest, it is difficult to perform 3D gesture tracking on the device itself since Glass has limited processing power to achieve skeletal tracking in real time, so they opted to use a depth sensor above the area of interaction to generate real-time 3D models of both hands.

Augmented reality chess games have also been implemented for evaluation purposes on previous works. In the first approach [6], a handheld pen prop with a marker cube on top of it has been utilized, in order to interact with the chess pieces. Authors admit, that tracking of interaction props was inaccurate and slow to provide natural use. In the second one [7], optical finger tracking techniques are used to allow gestural interaction with chess pieces, using a hand's 3D model that can determine enough information to robustly track the position, orientation and pose of the user's index finger, without annoying cables. Users can use grab and release gestures to move virtual objects and image processing techniques are utilized to detect hand gestures, using a single camera. However, this solution resorts to using a marked glove with retro-reflective spheres on top of the forefinger's joints, so natural interaction with virtual content is not encouraged. In our system we explore how a simple pinch gesture detection technique can be utilized to support precise 3D hand interaction for wearable AR.

III. Proposed Method

In the next sections, we describe the method proposed and the AR chess application that was developed.

A. Marker Tracking

To augment reality by adding virtual information to images or videos, we need to know where to render the virtual objects. Fiducial markers are still a useful approach, since high precision can be achieved. In the context of the development of an augmented reality chess, users have to be able to see all of the virtual chess pieces at the same time, in order to figure out what their next move is going to be. In real-life chess, during a user's move, hands occlude the chessboard, so using a single marker might fail for different reasons, leading to incorrect camera pose estimation or none at all. To overcome that problem, we utilized a board of markers provided by ArUco library [8]. A markerboard is a marker composed by several markers arranged in a grid. This way, there are more points available for computing the camera pose and the accuracy obtained increases. Since the game of chess is a tabletop game that always uses a chessboard as a base for placing the chess pieces on, we considered that using a markerboard would simulate the use of a chessboard and the occlusion of the markerboard from user's hand would not affect the rendering of virtual chess pieces during a move. So, we utilized a markerboard which corresponds to the size and dimensions of a real chessboard and consists of 64 highly reliable markers in a (8 x 8) grid, as the main markerboard of the system, allowing users to cover part of the tracking markers without loss of camera pose information.

B. Pinch Gesture Detection

The pinch gesture is one of the most intuitive gestures for interaction with digital interfaces. It resembles a grabbing or picking action and offers a natural signal to select or move an object in an interactive system, achieving precision and high performance. While interacting with real objects, it is used in chess games from almost every average player. That is exactly why, in order to create an immersive augmented reality tabletop chess game, a robust pinch gesture detection technique has to be implemented, since user's fingers can't always be tracked from a camera's egocentric viewpoint.

Since part of the hand and fingers are not always visible from the sensor when a user moves a chess piece, we figured out, that using advanced hand tracking techniques would be an exaggeration. In this paper, we utilized a pinch gesture recognition algorithm that triggers a grabbing action to move and release virtual objects in the AR scene, based on A. Wilson's approach [9]. More specifically, we implemented an algorithm which detects pinch gestures as they happen in 3D space, which can be used for correct virtual object selection and manipulation. We emphasize on the blob data and the hole formed when forefinger and thumb of the right hand of the user touch each other. A blob is actually a shape identified within an image, in which the blob's pixels are white and the background pixels are black. Contour lines are the outer and inner border lines that separate the object from its background. Each blob has an external contour line, and optionally one or more internal contour lines, depending on the number of holes

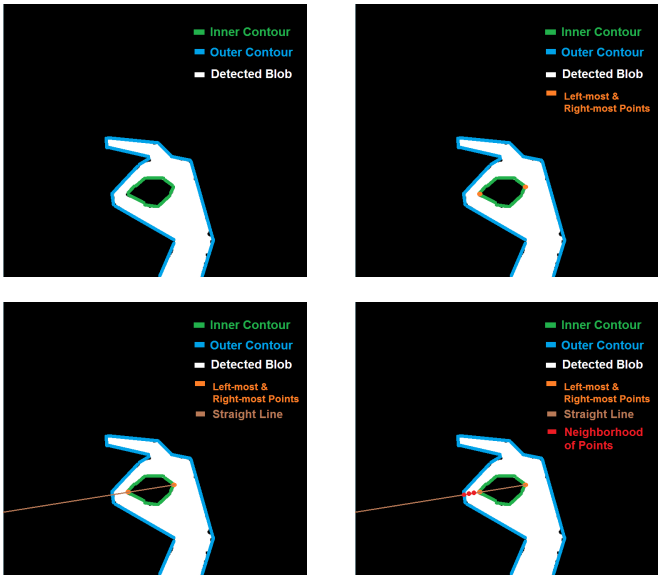

Fig. 1: Pinch Gesture Detection in 3D Space

created, while also each contour line is represented by a series of points.

Based on the above definitions of blob data and contour data, we designed an algorithm which detects the 3D position where the pinch gesture takes place. During each frame, color and depth streams are captured from the 3D camera. After setting specific parameters such as the maximum number of blobs we want to detect and the smoothing of segmentation and contours, the blob detection procedure is activated. We wish to detect the nearest blob of the depth image, since this object usually represents the hand of a player, during a chess game and obtain the number of contours detected. If there aren't more than 2 contours, then there can't be any hole formed in the depth image, so user can't have attempted a pinch gesture. On the other hand, if the number of contours is greater than 2, then there is a probability of a pinch gesture occurrence. To realize whether or not a pinch gesture actually takes place, the points of the largest inner contour are acquired, which also happen to be, the points of the hole formed during pinch gestures. If the number of contour points is below a certain threshold value, then we concede that the current contour data are either depth noise data not related to the hand blob or that user's hand is too far away from the sensor and therefore we should move on to the processing of the next frame. If not, we continue and calculate the left-most and right-most points of the inner contour. Once these points are found, a straight line, defined by these 2 points, can be determined.

Next, we create a "neighborhood" of points, which belong to the straight line that was estimated and the left side of the left-most point of the inner contour (Fig. 1). We get the depth values for each of these points and determine the average depth value of the pixels which gives us a simple measure of depth of the user's hand in 3D space. By mapping each of the "neighborhood" points from depth to color frame, we measure the average x and y values of the color frame's pixel-points and consequently get the x and y values in real

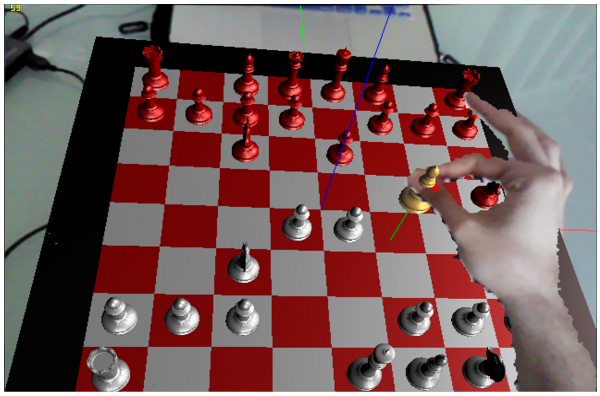

Fig. 2: The AR Chess Application. Notice that occlusion from the users hands is handled correctly.

world units (meters), by projecting these color pixels to the camera coordinate system. Ultimately, a pinch gesture has been correctly detected and the x,y,z values for one specific pinch-point in 3D world coordinates with respect to the camera, which is considered as the point in 3D space where the user decided to pinch has been estimated. To map the pinch gesture to an object manipulation, we first extract the related 3D camera coordinates of the detected pinch, and then project them into the AR markerboard's world coordinate system which is used as a location reference for the virtual object.

C. Chess Engine Integration

The application developed, simulates a real chess game against an opponent. In order for users to be able to play against a computer, a simple way for A.I functionality had to be investigated. That is why, we decided to integrate a chess engine within our application, which analyzes the position of all chess pieces on the board and calculates the best move within a time limit. To communicate with a chess engine, the Universal Chess Interface (UCI) Protocol was used, that enables interaction with our application through the exchange of string commands. Usually, chess engines don't have the ability to know whether or not a move command by the user is a valid one or not, based on the pieces type and the state of the board. However, in this approach, the iCE Engine was utilized, which stores the possible moves for every chess piece and when an invalid move is attempted, a warning is returned. This feature prevents users from executing wrong moves during gameplay. Finally, the engine we used can output the outcome of the game, so that our program can detect if there is a check-mate.

D. Occlusion Handling

For a user to successfully perform tasks in mixed reality environments, occlusion handling between the user's fingers and the virtual objects in the AR space should be taken into consideration, so that virtual content can blend with the natural environment. The main problem when dealing with occlusion is that usually there is no depth information of the real scene. Nonetheless, in our approach, the depth capabilities of the RealSense™ camera can be utilized to acquire a depth map of the environment scene, which is crucial information for realistic occlusion. This way, the depth of every pixel of the depth frame can be estimated. In our approach, we fill the Z-Buffer with the depth values taken from the depth image before rendering any 3D virtual object and after rendering the quad which shows the color video stream in the background. By doing so, chess pieces are correctly occluded by user's hands and other objects. Firstly though, we need to modify the depth data based on the projection that is used (ortho or perspective). In our case, virtual objects are rendered using perspective projection, so the relationship between Z (depth values obtained from sensor) and depth (depth of virtual object in OpenGL scene) is:

$$depth = \frac{Z_{far} Z_{near}}{Z(Z_{far} - Z_{near})} + \frac{Z_{far}}{Z_{far} - Z_{near}} \quad (1)$$

E. Gameplay

When users try a pinch gesture having as a goal to move a chess piece, we must render the chess piece with respect to the 3D pinch-point so that the piece will move according to it (Fig. 2). In order to manipulate a chess piece, users first need to select one. However we have to find out which chess piece is the user trying to select. Hence, we get the 3D position of the pinch-point, based on the procedure we described previously and we calculate all the distances between this pinch-point and each center of all the squares of the board that have a chess piece on them which belongs to user. From all these distances, we estimate the square which has the minimum distance from the pinch-point, as well as the distance itself. To prevent selection of a chess piece when the user attempts a pinch gesture too far away from the chessboard, a condition must be met, i.e the distance must be lower than a threshold value. If there is a valid selection of a chess piece, then during the rendering pipeline, we render all the chess pieces of the board, except for the one that was selected. Instead, we render the selected piece, in the position of the pinch-point and a green line representing the projection of this chess piece to the chessboard, to help users better perceive depth of the virtual scene. If the user decides to make a move, he will translate the chess piece on top of another square of the chessboard and will release the object (forefinger and thumb not touching each other anymore), so that the piece can occupy a new chessboard square. Once a move is completed by the user, our program queries the engine to check if the move is valid based on chess rules. If so, a new best move for the A.I is estimated and the position of chess pieces is updated. Otherwise, user has to attempt a different move.

IV. Experimental Setup

The design and architecture of our application has been built in such a way so that the RealSense™ 3D Camera can be mounted on top of an Oculus Rift device. It has been decided, though, to work in an experimental setup which would simulate the parameters of height and viewpoint. In order to evaluate the developed AR chess application, we created a testing area, where a markerboard is placed on a table, easily reachable by a user sitting in front of it. Users wore a hat with the sensor attached to the back of it, positioned at eye level and in between their eyes, pointing towards the markerboard target, to encompass the area of interaction. Finally, a laptop connected

to the camera was placed in front of the users' view where the augmented video captured by the camera is displayed. Figure 3 shows one of the voluntary participants that took part in the evaluation phase and the setup mentioned.

V. Evaluation

The experiment evaluation design has been chosen to draw a fair conclusion with a limited number of participants and resources. The experiment started with a short briefing to the participants about what this application is about and how it uses pinch gestures to bring the classic game of chess into AR. After this introduction, all participants had to perform a sequence of random moves, trying to beat the chess engine, to test virtual chess piece manipulation. In order to measure the usability and and learnability of the AR chess system developed, the standard System Usability Scale (SUS) questionnaire [10] was used, which provides a global measure of system satisfaction. The average score obtained was 73.25 with a standard deviation of 10.7 and median of 75, collected from 10 participants. Therefore, the application has a tendency towards a good and easy to use system. As additional support to gather information about the user experience, a 7-level likert scale open questionnaire was handed out to the participants. Many said they found the gameplay fun and engaging, and several thought that occlusion handling was really realistic. They also agreed that the green line used, helps them to better understand, on which square the virtual object is going to be dropped.

Pinch tracking may fail some times due to noise, so if the user is holding a chess piece and a pinch is not detected in a number of consecutive frames, the virtual chess piece may occupy a wrong chessboard square at pinch-out. That is why, we decided to measure the number of incorrect moves that took place during each evaluation test. No more than 3 moves out of 30 per user were considered wrong, while the average time required by each user to complete a correct move was measured at 3.72 seconds. Finally, participants had to pinch 1.62 times (on average) in order to correctly grab a virtual chess piece. During blob detection phase, when users decide to select and manipulate a virtual chess piece, their hands might be too close to the table, leading to a detection failure, since the hand and the table are considered as a single blob. However, we solved this problem by simulating the visual design of a real chessboard, rendering a cubic virtual chessboard with a specific height of 3.5cm above the tabletop surface.

VI. Conclusion

Many prototypes implemented with sophisticated computer vision algorithms to robustly recognize hand gestures have demonstrated that gesture recognition is rather complex and computationally expensive. In this paper, we discussed a way in which basic AR applications can be enhanced and interactions between real and virtual objects can be handled. A simple methodology to build an AR chess is presented, which allows simple and fast virtual object manipulation. Based on the results obtained, future work may include several improvements to the current prototype and additional features to manage seamless interaction in AR and create an even more immersive experience for users. Taking everything into consideration, the combination of methods in our approach allows users to play a chess game with virtual objects using their bare hands and pinch gestures without any serious problems.

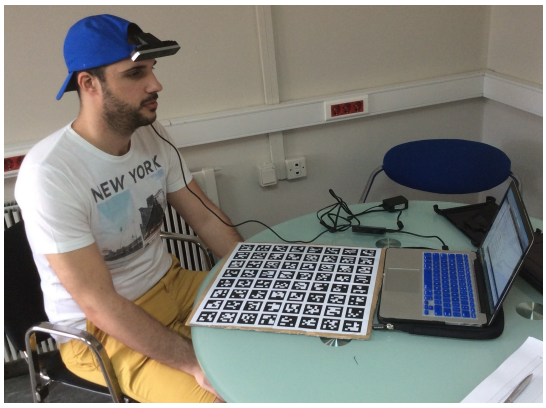

Fig. 3: Experimental Setup

Acknowledgment

This work has been partially supported by the Greek Secretariat for Research and Technology Bilateral Collaboration Project MOMIRAS (ISR_3215).

References

[1] M. Billinghurst, T. Piumsomboon, and H. Bai, "Hands in space: Gesture interaction with augmented-reality interfaces," *IEEE computer graphics and applications*, no. 1, pp. 77–80, 2014.

[2] B. Lee and J. Chun, "Interactive manipulation of augmented objects in marker-less ar using vision-based hand interaction," in *Proceedings of the 2010 Seventh International Conference on Information Technology: New Generations*, ser. ITNG '10. IEEE Computer Society, 2010, pp. 398–403.

[3] H. Kato, M. Billinghurst, I. Poupyrev, K. Imamoto, and K. Tachibana, "Virtual object manipulation on a table-top ar environment," in *Proceedings of the IEEE and ACM International Symposium on Augmented Reality*. IEEE, 2000, pp. 111–119.

[4] V. Buchmann, S. Violich, M. Billinghurst, and A. Cockburn, "Fingartips: Gesture based direct manipulation in augmented reality," in *Proceedings of the 2nd International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia*, ser. GRAPHITE '04. ACM, 2004, pp. 212–221.

[5] H. Bai, G. Lee, and M. Billinghurst, "Using 3d hand gestures and touch input for wearable ar interaction," in *Proceedings of the Extended Abstracts of the 32Nd Annual ACM Conference on Human Factors in Computing Systems*, ser. CHI EA '14. ACM, 2014, pp. 1321–1326.

[6] G. Reitmayr and D. Schmalstieg, "Mobile Collaborative Augmented Reality," in *Proceedings of the IEEE and ACM International Symposium on Augmented Reality*. IEEE, 2001, pp. 114–123.

[7] K. Dorfmuller-Ulhaas and D. Schmalstieg, "Finger tracking for interaction in augmented environments," in *Proceedings of the IEEE and ACM International Symposium on Augmented Reality*. IEEE, 2001, pp. 55–64.

[8] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, 2014.

[9] A. D. Wilson, "Robust Computer Vision-Based Detection of Pinching for One and Two-Handed Gesture Input," in *Proceedings of the 19th annual ACM symposium on User interface software and technology*. ACM Press, 2006.

[10] J. Brooke, "SUS: A quick and dirty usability scale," *Usability evaluation in industry*, vol. 189, no. 194, pp. 189–194, 1996.