Interaction Between Real and Virtual Humans: Playing Checkers

Rémy Torre, Pascal Fua, Selim Balcisoy, Michal Ponder and Daniel Thalmann

Computer Graphics Lab (LIG)
Swiss Federal Institute of Technology
CH 1015 Lausanne (EPFL)
Fax: +41-21-693-5328
FirstName.LastName@epfl.ch

Abstract. For some years, we have been able to integrate virtual humans into virtual environments. As the demand for Augmented Reality systems grows, so will the need for these synthetic humans to coexist and interact with humans who live in the real world. In this paper, we use the example of a checkers game between a real and a virtual human to demonstrate the integration of techniques required to achieve a realistic-looking interaction in real-time. We do not use cumbersome devices such as a magnetic motion capture system. Instead, we rely on purely image-based techniques to address the registration issue, when the camera or the objects move, and to drive the virtual human's behavior.

1 Introduction

Recent developments in Virtual Reality and Human Animation have led to the integration of Virtual Humans into synthetic environments. We can now interact with them and represent ourselves as avatars in the Virtual World [4]. Fast workstations make it feasible to animate them in real-time, which is required for Virtual Reality, Interactive Television, and Video Games applications. Virtual humans can be either guided or autonomous. Guided ones are useful to represent ourselves, the users or participants. Autonomous humans, on the other hand, are able to act on their own. They integrate techniques from Computer Animation, Artificial Intelligence, and Artificial Life. Their behavioral mechanisms drive their actions in response to perceived information. They have an important role to play in virtual environments containing many people such as airports, train stations, or even cities.

Most of these applications, however, have been developed in a Virtual Reality context and these Virtual Humans inhabit purely synthetic worlds. As the demand for Augmented Reality systems grows [1, 7, 12], so will the need to allow them to coexist and interact with real humans who live in the real world [9]. True interaction between virtual and real humans requires two-way communication. Real people are of course easily made aware of the virtual humans' actions. However, it is much more difficult to endow the virtual humans with perceptive capabilities. Direct transfer of data structures is not an option anymore. Recognition methods are required for virtual humans to perceive real humans' behaviors.

For example, in earlier Virtual Reality work, we have developed a fight training simulator [5]. The real user is represented by an avatar that is entirely controlled by a motion capture system while the virtual human is autonomous. He uses the motion capture data to assess the avatar's current posture and to react appropriately.

Our challenge is to achieve the same result in an Augmented Reality context where the user is represented by his own image as opposed to an avatar. This means that he cannot wear cumbersome devices such as a magnetic motion capture system or VR helmet.

In this paper, we use the example of a checkers game between a real and a virtual human to demonstrate the integration of techniques required to achieve a realistic-looking interaction. In this scenario, the real human plays checkers using a real board while the virtual human observes him using a single color camera that can change its viewpoint. As the camera moves, the virtual player updates its position relative to the board and keeps track of the pieces that sit on top of it. He can then respond to the real player's moves and synthetically grasp his own pieces to move them. Using our animation capabilities, we create animations of the virtual player that can be superposed to the real images taken by the camera to produce a realistic-looking augmented-reality interaction.

Recently, image processing has become popular for digital warping [15] to add new real actors into existing films, as was done in the Forrest Gump movie. It has also been shown to become increasingly useful for augmented reality applications [7]. Movie companies have successfully produced films like Jurassic Park, The Mask, or Titanic with digital creatures generated in a non real-time context. Similarly, in earlier work [14], we have developed an Augmented Reality system that allows virtual humans to perform script driven actions. First, color images are acquired with a single static camera. Next, a human operator uses 3–D input devices to supply the required input while a high end graphics workstation renders the synthetic world with virtual actors in correct correspondence with the real world. Finally, this synthetic image is composited with the image of the real scene to generate the output image. Virtual human creation and animation is based on the existing Humanoid 2 Esprit European project [3]. However, in all these cases, the virtual humans are not autonomous at all. There is no interaction between them and the user. It takes a complete real-time, non-invasive vision-based recognition system to create such an interaction. This is the main innovation of this paper.

To achieve our goal, we have augmented our earlier system [14] by developing and integrating many distinct technologies that originate from the fields of Computer Graphics and Computer Vision. In the remainder of the paper we first introduce the model-based tracking algorithm we have developed to follow the checkers board and allow camera motion. We then describe the virtual human's behavioral mechanism. Finally we present our approach to integrating these techniques into our real-time system.

2 Model Based Tracking

To combine real and synthetic scenes, we must be able to project synthetic models at the right location in real images. To do so, we have to calibrate the camera used to film the real scene, that is to compute the internal and external camera parameters that define the projection matrix [13]. Furthermore, as the camera moves, its motion must be tracked.

We propose an approach based on the concept of Model-Based Optimization [6]. It is a paradigm in which an objective function is used to express both geometric and photometric constraints on features of interest. A parametric model of a feature, such as the checkers board, is extracted from one or more images by automatically adjusting the model's state variables until a minimum value of the objective function is obtained. The optimization procedure yields a description that simultaneously satisfies—or nearly satisfies—all constraints, and, as a result, is likely to be a good model of the feature. As in the work reported in [10], the models are defined as sets of 3–D vertices that form a graph structure and obey a set of geometric constraints. Their world positions are a function of a small number of state variables that control the object's shape. The objective function is computed by projecting these vertices into the images, using camera models, and summing image gradient values along the line segments linking the projections. In the case of the checkers board, we use the model of Fig. 1.

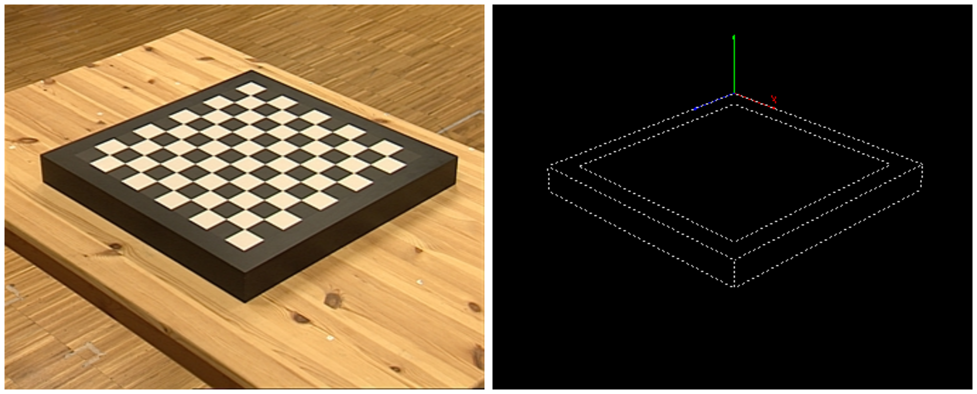

Fig. 1. 3D model of the checkers board. Note that the hidden faces are not displayed and that the hidden edges are not taken into account during the computation.

If the camera is well calibrated, the camera model parameters do not need to be adjusted. However, if these parameters are not known precisely or, if either the camera or board move, the optimization can be performed not only with respect to the model's state variables but also with respect to the unknown camera parameters. In this fashion, camera positions can be recovered both automatically and precisely. In the specific case presented here, we track the checkers board and use it, in effect, as a calibration object. The procedure goes through the steps depicted by Fig. 2.

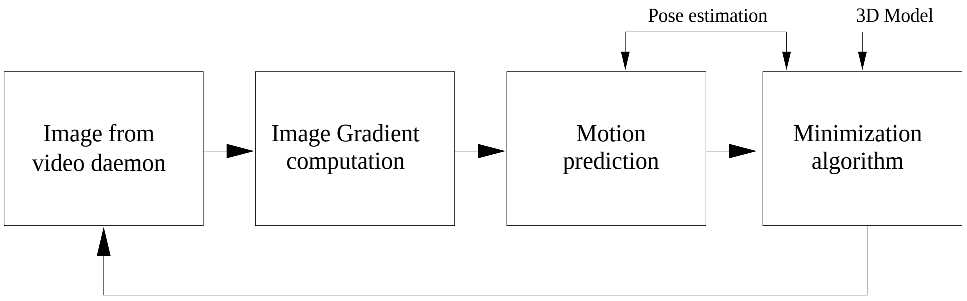

Fig. 2. Overview of the 3–D tracking algorithm.

We acquire a video image, apply a gradient operator, predict the motion based on the previous observation and finally optimize the camera parameters to ensure that the model projects where it should.

An alternative approach would have been to track specific features or targets [8]. However, as shown in Fig. 4, the real player may hide substantial areas of the board. Occlusions can become severe, resulting in the loss of many features. In our approach, because we integrate over the whole wireframe model, as long as a sufficient portion of the board is seen, this does not appear to create major difficulties. It also has the advantage of allowing us to use elements of the environment to perform the registration task, thus avoiding the use of additional ad-hoc targets.

2.1 Motion prediction

To ensure robustness of the tracking process, we use a linear motion predictor. Let $X(t)$ be the 3D position of the model at frame t and $\widetilde{X}(t+1)$ be the estimate 3D position of the model at frame $(t+1)$, then:

$$\widetilde{X}(t+1) = X(t) + V(t).\Delta T \tag{1}$$

Where $V(t)$ is speed between $(t-1)$ and t. We have also tested a second order predictor. In our experience the first order scheme works better than the second order one. The latter takes too long to initialize and, subsequently, does not produce better results.

2.2 Minimization

This the final and the most important step of the tracking process. It fits the object model on the intensity gradient using an iterative minimization of a non-linear energy function with respect to the six 3–D pose parameters. We project the 3–D model into the 2–D image using the previously computed camera parameters.

Let $S_i(t) = [T_x(t), T_y(t), T_z(t), R_x(t), R_y(t), R_z(t)]$ be the vector of the 6 position and orientation parameters of the camera relative to the 3D model and $C_j(t), j \in [1, .., n]$ be the visible projected segments on the image at frame t. We take the objective function along these visible edges to be:

$$E_{S_i(t)} = \sum_{(i,j)\in C(t)} \left[\sum_{(u_i,v_i)}^{(u_j,v_j)} G(u,v) \right] \quad , \tag{2}$$

where $G(u,v)$ is the value of the pixel (u,v) in the gradient image. We compute optimal values of $S_i(t)$ by minimizing $E_{S_i(t)}$ using the Minos software package [11].

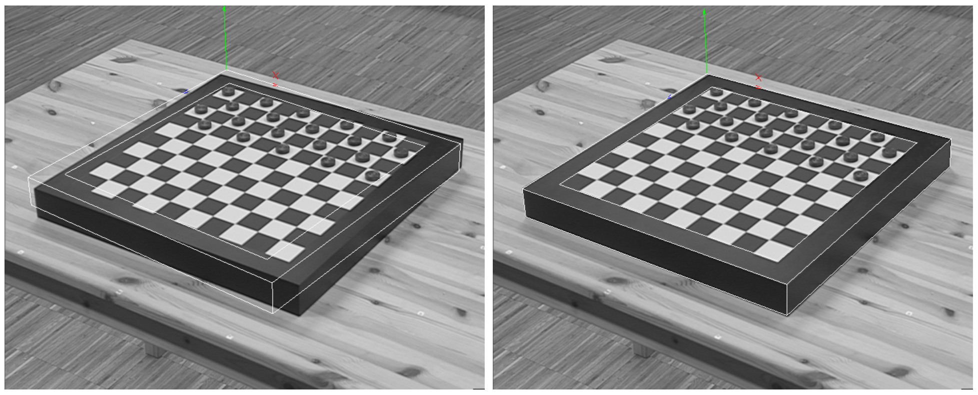

Fig. 3. Finding the exact location and orientation of the board. Left: Interactively supplied initial position. The white wireframe model does not project exactly at the right location. Right: Position after minimization of the objective function. The model's projection is now perfectly superposed with the board.

2.3 Tracking the board

Thanks to this tracking algorithm we can now move either the camera or the board during the game. The tracker is initialized semi-automatically. As shown in Fig. 3, we use the mouse to supply an approximate initial position and refine it by minimizing the objective function of Eq. 2. We then launch the tracking process. To ensure adequate speed and precision, we use a gradient image that is four times smaller than the original image. Using PAL video size (720x576) on an SGI Octane workstation, we track at a rate of 22 Hz. As the camera moves around the board, we obtain the results depicted by Fig. 4. As shown by Fig. 5, even though we do not explicitly model changes in focal length, the tracker can accommodate significant zooming in and out. In effect, it trades changes in focal-length against changes of the estimated distance from the camera to the board.

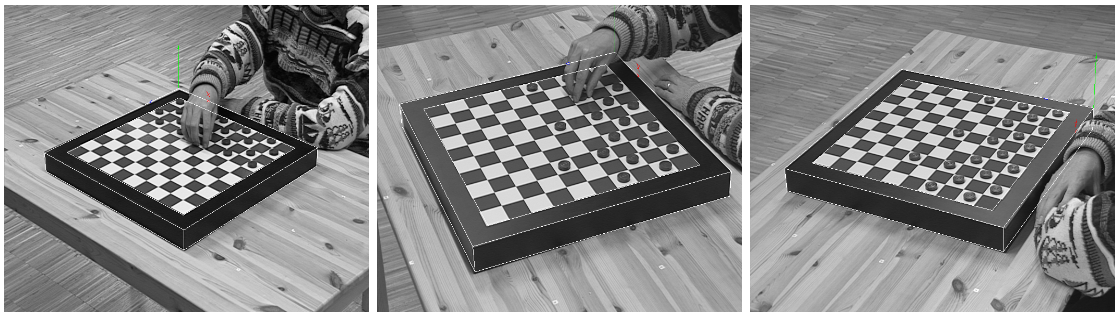

Fig. 4. Tracking results while rotating and translating the camera. The model's projection, drawn in white, remains perfectly superposed with the board. Note that the algorithm is relatively insensitive to occlusions.

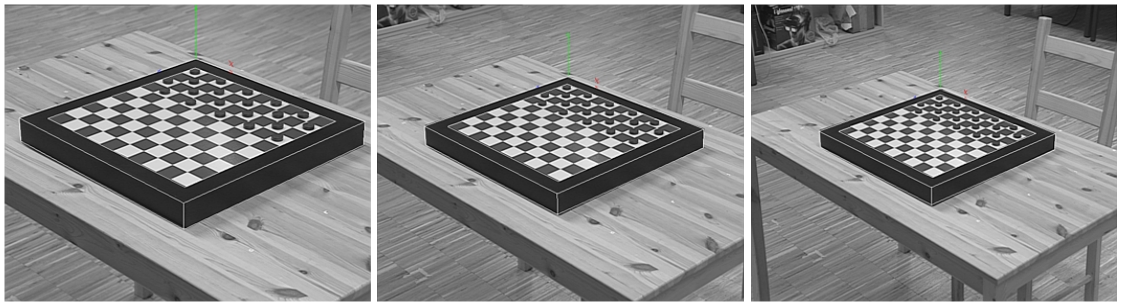

Fig. 5. Tracking results while changing the focal length. Again, the model's projection remains perfectly superposed with the board

As the tracking process uses all the target's visible edges, it is unaffected by partial occlusion. The real player can move his own pieces without disturbing the algorithm, as long as about 50 per cent of the edges remain visible.

3 Playing the game

During the game, the real person plays with real brown pieces and the virtual player plays with virtual white ones. The latter's behavior is driven by a simple algorithm that takes as input the position of all the real pieces and returns the virtual human's next move. This function declares an error if the real player makes an illegal move so that the virtual one can notify him. As the virtual player cannot move pieces, the real player has to remove his own pieces when they are eliminated from the game. We must therefore endow the virtual human with the ability to detect, and keep track of, the position of the real pieces on the board before each action.

3.1 Checkers detection

For each frame, a reliable 3–D estimate of the board's position is maintained by the tracking process. We can therefore work in the board's horizontal plane, depicted in Fig.6 as the XZ plane. The board has 10x10 squares of size D. Let L_i be a "scan segment" parallel to the model's X axis. We write

$$L_i = \left[\left(0, 0, \frac{D}{2} + (i-1).D\right); \left(10.D, 0, \frac{D}{2} + (i-1).D\right)\right] \quad \text{for } i \in [1, .., 10] \ . \tag{3}$$

After projection we have

$$L_i = [(u1, v1); (u2, v2)] \ . \tag{4}$$

As the game is played only on square of the same color, we need only examine the white squares. Let j be the index of a white square on line L_i. We compute a detection value for square j as follows:

$$V_{L_i(j)} = \sum_{\substack{x=u1_j \\ y=v1_j}}^{\substack{x=u2_j \\ y=v2_j}} I(x, y) \qquad \text{for } j \in [1, .., 5] \ , \tag{5}$$

where $I(x, y)$ is the pixel value of the real image. We then assume that, if $V_{L_i(j)} \leq Threshold$, a piece is present on square (j, L_i).

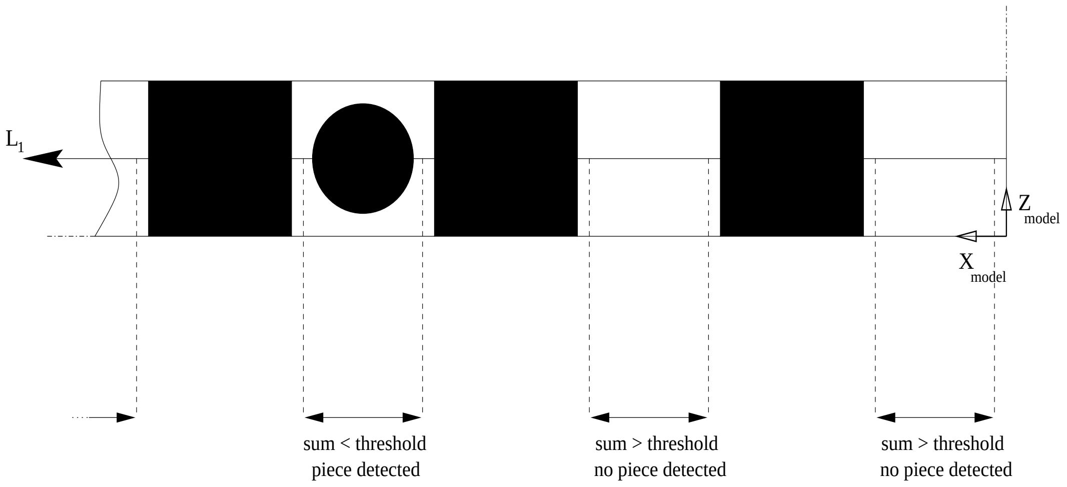

Fig. 6. Checkers detection.

4 Integration into the Real-Time Augmented Reality System

In this section we present our real-time graphics platform, the Virtual Human Director (VHD). It can handle real and virtual elements in a single mixed environment and provides dedicated tools for Augmented Reality. VHD is a system dedicated to real-time production of multimedia products and TV shows [14]. It provides diverse animation tools and a powerful scripting system for recording scenarios. Using these features users can interact with virtual humans and develop complex interactions. In this paper, the checkers game is the scenario.

To allow the virtual human to play against the real one, we have developed some specific tools that take into account the calibration data supplied by the tracker and checkers game engine data. Initially VHD gets the real camera parameters and adjusts the virtual camera parameters. The tracking software is connected through a TCP/IP port. A key issue is to synchronize the rendering of synthetic scenes with the real camera output. The checker game software controls the environment. VHD opens a TCP/IP port to receive virtual human control commands: grasping, playing keyframes, and navigation. According to the information supplied by the checkers game software, VHD animates the virtual player and updates the scene. In our example, the virtual human must grasp and move its pieces. A grasping action is composed of three subtasks: reaching the object's position with the arm, grasping the object with the hand, and animating the grasped object to place it properly into the hand. For the purpose of reaching, we use either inverse kinematics or keyframe animation. Fig. 7 shows the relationship between system components. The system runs on two separate workstations. To produce the results shown in this paper, those were an SGI Onyx2 for VHD and an SGI Octane for the Tracker and Checker Game Controller. VHD can render at the rate of 22 PAL (720x576) frames per second while the tracker currently returns results at the rate of 22 Hertz. The current implementation, however, only uses one processor of a dual processor Octane and a non optimized version of the Minos minimization package [11]. We are working on a multi-threaded implementation that should be noticeably faster.

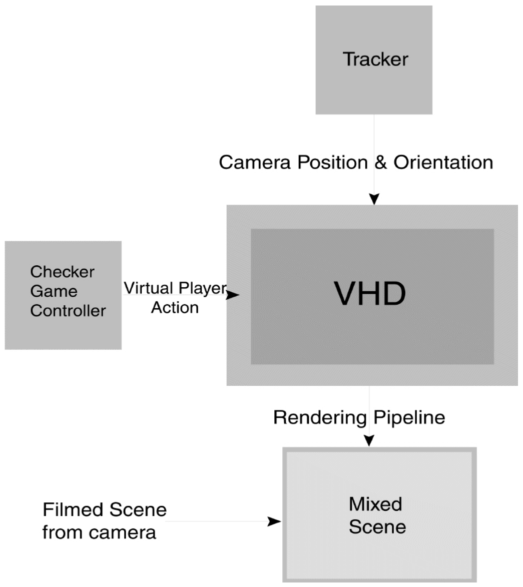

Fig. 7. VHD System Components.

To give the correct illusion of an augmented world, we use chroma-keying, where black is the key color. As shown in Figure 8, the virtual world is rendered with a black background, and all the 3D objects which represent real objects are black. This enables virtual humans to interact with real objects representations and give the illusion that they are interacting with real objects. When a virtual human stands behind a real table, the table's virtual representation masks the virtual human [2]. The real player is getting visual feedback from a large monitor, carefully placed in front of him. It is displaying the mixed image to give a magic mirror illusion.

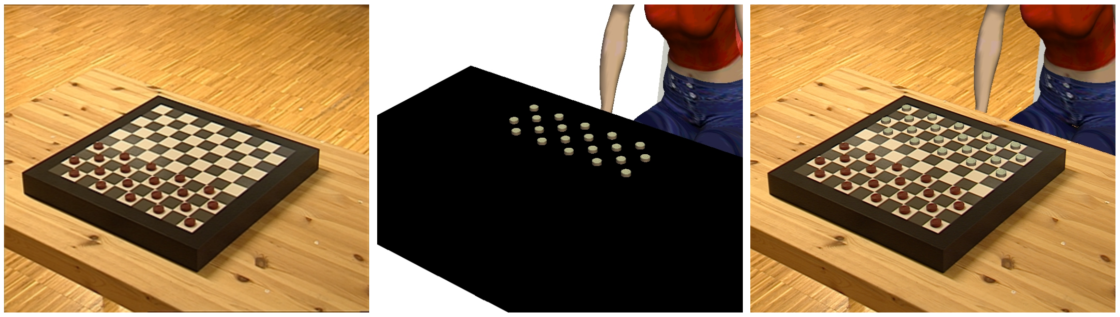

Fig. 8. Image compositing. Left: Real image with overlaid checker board model. Center: The synthetic image with black table for chroma-keying. In the actual system, the whole background is black but, in this image, we painted it white so that the table could be seen, Right: Composited image.

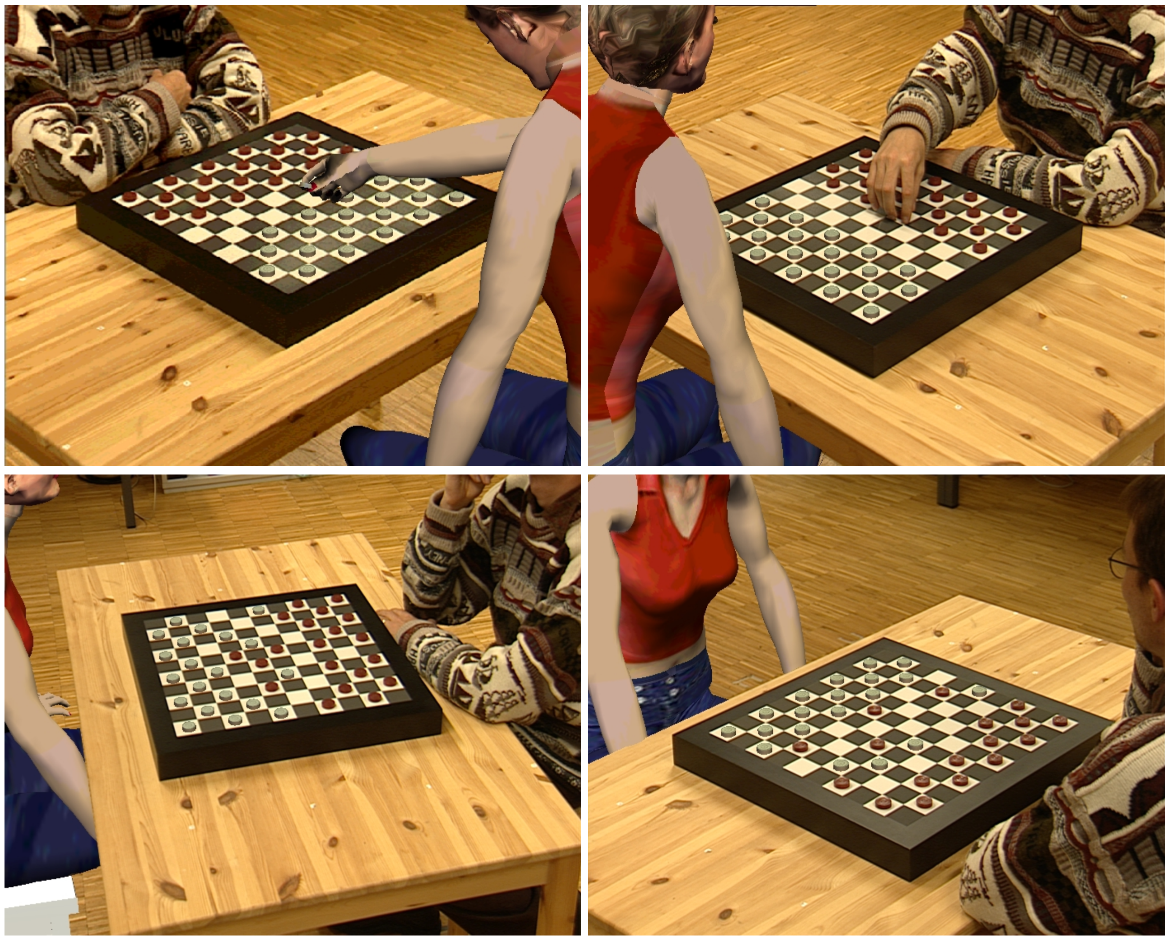

Fig. 9. Checkers game. The real and virtual players appear and move together in the composited images.

The final image sequences are generated in this manner. We use an off the shelf chroma-key blender to combine the real filmed scene, containing the real player, and the synthetic 3D scene containing the virtual one. Figure 9 depicts the results.

5 Conclusion

We have presented an approach to real-time interaction between a real and a virtual person in an augmented reality context. We rely on purely image based techniques to solve the registration problem and to give the virtual human a perception of its surroundings.

We have demonstrated this approach in the context of a checkers game. We do not require sophisticated devices and our model-based approach allows us to use elements of the environment–in this case the board—without requiring specific targets. In future work, we plan to extend this approach to more generic situations so that we can easily generate augmented reality sequences that include real and virtual humans who interact with each other.

In this paper, we used the checker board as a calibration object. We represent it as a generic wireframe model. Thus, this approach can be generalized to any polygonal object such as a table or a chair. We are currently extending our method so that it can simultaneously track many such objects and, therefore, become both more robust and more precise.

References

1. R.T. Azuma. A Survey of Augmented Reality. *Presence, Teleoperators and Virtual Environments*, 6(4):355–385, August 1997.
2. S. Balcisoy and D.Thalmann. Interaction between Real and Virtual Humans in Augmented Reality. In *Computer Animation*, pages 31–38, 1997.
3. R. Boulic, T. Capin, Z. Huang, L. Moccozet, T. Molet, P. Kalra, B. Lintermann, N. Magnenat-Thalmann, I. Pandzic, K. Saar, A. Schmitt, J. Shen, and D. Thalmann. Environment for Interactive Animation of Multiple Deformable Human Characters. In *Eurographics*, pages 337–348, Maastricht, Netherlands, August 1995.
4. T. Capin, I. Pandzic, N. Magnenat-Thalmann, and D.Thalmann. *Avatars in Networked Virtual Environments.* John Wiley and Sons, 1999.
5. L. Emering, R. Boulic, and D. Thalmann. Interacting with Virtual Humans through Body Actions. *IEEE Computer Graphics and Applications*, 18:8–11, 1998.
6. P. Fua. *RADIUS: Image Understanding for Intelligence Imagery*, chapter Model-Based Optimization: An Approach to Fast, Accurate, and Consistent Site Modeling from Imagery. Morgan Kaufmann, 1997. O. Firschein and T.M. Strat, Eds. Available as Tech Note 570, Artificial Intelligence Center, SRI International.
7. G. Klinker, K. Ahlers, D. Breen, P.-Y. Chevalier, C. Crampton, D. Greer, D. Koller, A. Kramer, E. Rose, M. Tuceryan, and R. Whitaker. Confluence of Computer Vision and Interactive Graphics for Augmented Reality. *Presence: Teleoperations and Virtual Environments*, 6(4):433–451, 1997.
8. D. Koller, G. Klinker, E. Rose, D.E. Breen, R.T. Whitaker, and M. Tuceryan. Real-time Vision-Based Camera Tracking for Augmented Reality Applications. In *ACM Symposium on Virtual Reality Software and Technology*, pages 87–94, Lausanne, Switzerland, September 1997.
9. N. Magnenat-Thalmann and D. Thalmann. Animating Virtual Actors in Real Environments. *ACM Multimedia Journal*, 1998.
10. E. Marchand, P. Bouthemy, F. Chaumette, and V. Moreau. Robust real-time Visual Tracking Using a 2D-3D Model-Based Approach. In *International Conference on Computer Vision*, pages 262–268, Corfu, Greece, September 1999.
11. B. Murtagh and M.A. Saunders. Minos 5.4 User's Guide. Technical Report SOL 83-20R, Department of Operations Research, Stanford University, 1983.
12. Z. Szalavari, E. Eckstein, and M. Gervautz. Collaborative Gaming in Augmented Reality. In *ACM Symposium on Virtual Reality Software and Technology*, pages 195–204, Taipei, Taiwan, November 1998.
13. J.P. Tarel and J.M. Vezien. Camcal Manual: A Complete Software Solution for Camera Calibration. Technical Report 0196, INRIA, September 1996.

14. G. Sannier S. Balcisoy N. Magnenat-Thalmann D. Thalmann. VHD: A System for Directing Real-Time Virtual Actors. In *The Visual Computer*. Springer, 1999.
15. G. Wolberg. *Digital Image Warping*. Computer Society Press, 1990.