# Interactive Crowd Simulation for Augmented Reality Environments

Ateş Akaydın
Bilkent University
akaydin@cs.bilkent.edu.tr

Aytek Aman
Bilkent University
aytek.aman@cs.bilkent.edu.tr

Uğur Güdükbay
Bilkent University
gudukbay@cs.bilkent.edu.tr

## Abstract

We propose a system to perform interactive crowd simulation within an Augmented Reality (AR) environment. The camera is localized within the simulation space using vision-based tracking methods. We propose a real-time automatic and occlusion-tolerant registration technique for estimating camera position and orientation using natural markers.

**Keywords:** augmented reality, virtual reality, crowd simulation, virtual worlds

## 1 Introduction

Real-time virtual crowd simulation in dynamically changing environments is a challenging problem that has been studied since 1980s. Studies on virtual crowd simulation have been done primarily for Virtual Reality (VR) environments. AR differs from VR in that it merges a real environment with the synthesized elements.

Compared to VR, AR has many advantages in terms of presenting and reinforcing knowledge within a digital environment. Users are not restricted to work on stationary computers. Up to date, hardware for AR systems are portable and can be carried with ease. It enables users to communicate face to face and collaborate to realize goals of a simulation scenario. In AR, synthetic images are merged with real images. Thus, AR reinforces learning through physical experience in the users' natural environment.

This study proposes a framework for interactive crowd simulation in AR environments. The framework enables multiple clients using physical AR devices to connect to a centralized server that simulates the synthetic crowd. We use a vision-based method for estimating camera position and orientation using natural markers. Our tracking method is also robust in the presence of occlusion and can work even with a very small set of visible natural markers.

## 2 Related Work

A comprehensive survey on AR technologies is carried out by Krevelen and Poelman [1]. In this study, common AR methods and technologies, such as AR display devices, camera calibration techniques, marker tracking methods, user interfaces and interfacing metaphors and rendering techniques are discussed. Carmigniani et al. [2] perform another survey on AR technologies, systems and applications with the challenges that they should address.

AR systems require a way of tracking to register the synthetic objects that are rendered on the display devices. Tracking can be performed using vision-based methods by recognizing and tracking a set of visual markers that reside within the camera images. Rolland et al. [3] provide a comprehensive survey on tracking and camera calibration technologies.

We also consider simulation of computer controlled synthetic crowds in AR environments. Thalmann and Musse [4] carry out an extensive survey on synthesizing and controlling virtual crowds. Zhang et al. [5] propose an online approach for inserting virtual agents into real

Figure 1: An example AR environment. Left: the real environment with simple geometric primitives. Middle: the view the user sees through the AR display. Right: the collision mesh and synthetic agents in virtual environment. The red and blue wireframe hemispheres represent the locations where the agents are created and destroyed.

scenes captured by video. Egges et al. [6] propose a framework for real-time interaction with virtual agents in an AR environment. Papagiannakis et al. [7] describe virtual agents with artificial life behaviors in an AR environment with frescos-paintings. Barakonyi et al. [8] propose MonkeyBridge, which is a multiplayer AR game where users place real and virtual objects in an environment. These objects influence the behavior of virtual agents and they react accordingly.

# 3 Proposed System

## *3.1 Overview*

Figure 1 shows an AR environment constructed by the proposed framework where a synthetic crowd is simulated by the server. The server updates the the synthetic crowd for the connected AR clients. AR clients then register and render the agents on the AR display devices.

In the virtual space, clients are represented as avatars. The system requires the knowledge of positions, orientations and view frusta of these avatars. The AR clients track and estimate the positions of their avatars and transmit this information to the server. The simulation server also carries out path and behavior planning for the synthetic agents within the virtual space.

The proposed approach requires a low-resolution collision mesh of the real environment. Synthetic agents in the virtual space consider this collision mesh to plan collision-free paths. Without this collision mesh, the agents can collide with static entities in the real environment. In this case, the rendered agents on the AR display devices will look as if they are overlaid on real obstacles. We use a hand-modeled collision mesh that accurately represents the important object within the real environment. The same mesh is used for multiple tasks involving camera registration, synthetic object culling and for obstacle avoidance during path planning.

## *3.2 Real-Time Localization*

*Real-time localization* is the process of accurately estimating position, orientation and velocity of an avatar in a virtual space using sensors with different modalities. The modalities may include Inertial Measurement Units (IMUs), positioning technologies (i.e., GPS), and vision based tracking. Figure 2 shows the localization process. Currently, we use only vision-based tracking methods for registration.

The localization and registration problems are actually the same problem. We could devise a transformation function $M$ that would transform the from the *World Space* ($w$) into the *Virtual Space* ($v$) given the extrinsic and intrinsic properties of the AR camera. The intrinsic parameters are determined only once during camera calibration. We seek to estimate the extrinsic camera parameters by fusing measurements obtained from the tracking methodologies. A solution to the localization problem estimates the correct parameters as accurate as possible.

We use a vision-based real-time localization approach that involves two steps. The first step is the calibration step, which is common in many vision based tracking approaches. The second step is camera localization in which we dynamically localize the camera by determining its extrinsic parameters.

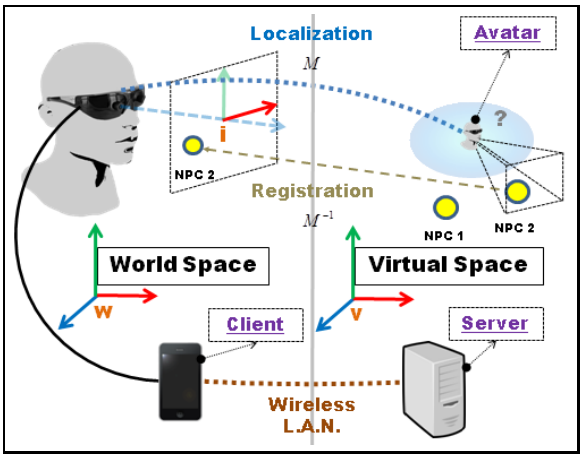


Figure 2: The overview of the localization process. Non-Player Controlled (NPC) entities are the synthetic entities. The world ($w$), image ($i$), virtual ($v$) coordinate spaces are shown.

### 3.2.1 Camera Calibration

A point ($p_w$) in three-dimensional (3D) world space can be projected to a point ($p_i$) on image plane using Equation 1. We use homogeneous coordinates for $p_w$ to simplify transformations involving translations:

$$z_c\, p_i = I\, [\, R\, T\, ]\, p_w \text{ where } p_i = [\, u\ v\ 1\, ]^T \text{ and } p_w = [\, x_w\ y_w\ z_w\ 1\, ]^T \quad (1)$$

In Equation 1, $I$ represents the intrinsic camera parameters. Matrices $R$ and $T$ define rigid-body transformations of rotation and translation, respectively. $R$ and $T$ together form the extrinsic camera parameters.

We only consider the four linear intrinsic parameters that form the $I$ matrix. These are the focal length ($f_x, f_y$) and the principal point ($s_x, s_y$) of the camera. We ignore the nonlinear intrinsic parameters (such as lens distortion) because their effect is negligible.

For camera calibration, we use the technique proposed by Zhang [9]. Zhang uses a checkerboard pattern and identifies point features from camera images. The detected features are matched with the known 3D coordinates on the pattern. The approach uses Levenberg-Marquardt Algorithm to estimate both intrinsic parameters of the camera by solving a nonlinear minimization problem. The method requires at least two dissimilar images of the calibration pattern to work.

One particular problem with calibration is the correctness of the initial feature positions. Temporal noise inherent in the camera images can greatly affect the performance of point feature extraction. An exponential moving average (EMA) is used to filter out most of the noise. This noise-removal method works effectively for stationary cameras, and hence, it is well suited for calibration.

In addition to camera noise, we also identified specular illumination as an important source for incorrect feature detection from camera images. We use adaptive thresholding to reduce the effects of specular illumination.

### 3.2.2 Camera Localization

Localization is the process of finding extrinsic camera parameters. We use a vision-based registration method using natural markers. These markers are coarse meshes that represent the static entities in the simulation environment. Our registration method uses edge based tracking to estimate camera parameters. We first extract the 'important' edges from the collision mesh. We extract line segments from the camera at each frame using Hough transform and discard edges shorter than a threshold. We then define a camera transformation space with six unknowns (for translation and rotation) and search for a correct parameter configuration. At the correct configuration, we expect the image edges to match with the projected 'important' edges from the collision mesh. We search the camera transformation space and calculate a match score for each configuration. A configuration with the best score is selected that corresponds to the registered camera parameters.

To extract important edges from a mesh, we traverse the mesh and find edges that have two neighboring faces. If the angle between the normals of the neighboring faces is greater than a threshold, we mark the edge as important. To extract the edges from an image, we use the Canny edge detector. Probabilistic Hough transformation then locates the line segments in the image. We discard short edges for registration.

To calculate a match score between projected edges and the image edges, we use a method similar to the Lowe's method [10]. We count the matching edges and consider the match ac-

curacy. Two edges ($A$ and $B$) are accepted as matching if the distance between them (Equation 2) is small.

$$d(A, B) = \sum_{p \in A} d(p, B)^2. \quad (2)$$

In Equation 2, $p$ represents the point samples on edge $A$. $d(p, B)$ is the distance of p to the closest point on line $B$. The match score is calculated as $\alpha n - (1 - \alpha)S$, where $n$ is the number of matches and $S$ is the total match score of the matching edges; note that a matching score of 0 means perfect match for a single edge. $\alpha$ is determined experimentally where larger $\alpha$ values yield better localization performance.

### *3.3 Interactive Crowd Simulation*

We use a navigation mesh (NavMesh) based global path planning approach to globally navigate agents. The NavMesh is generated from a rough collision mesh of the static obstacles in real space. We also use this collision mesh for camera localization. Agent paths on this NavMesh are determined by the $A^*$ search algorithm. We use the Reciprocal Velocity Obstacles (RVO) [11] for local collision avoidance.

Occlusion culling of synthetic agents can be performed against the collision mesh. We render both the collision mesh and the synthetic agents on one image frame. The synthetic agents are culled on this image by z-buffer elimination. The resulting image is alpha-channel blended to the real image via a GPU shader program.

## 4 Conclusion

We propose an interactive simulation system for virtual crowd simulation in AR environments. The system provides a cost efficient and practical way of executing simulations in AR environments. The system will be beneficial for applications of serious games: military training, emergency planning, and education.

We are currently using only vision-based methods for tracking and registration operations. As a future work, we are planning to use inertial measurement sensors for improving registration quality. We are planning to use an Extended Kalman Filter to further increase both performance and accuracy of our approach. Multi-modal tracking with an Extended Kalman Filter may significantly reduce the search space of our registration algorithm at each frame.

## 5 Acknowledgments

This work is supported by The Scientific and Technological Research Council of Turkey (TÜBİTAK) with Grant no. 112E110. The second author is supported by TÜBİTAK under BİDEB 2210 Grad. Sch. Programme.

## References

[1] D.W.F. van Krevelen and R. Poelman. A Survey of Augmented Reality Technologies, Applications and Limitations. *Int. J. on Virtual Reality*, 9:1–20, 2010.

[2] J. Carmigniani et al. Augmented Reality Technologies, Systems and Applications. *Multimedia Tools App.*, 51:341–377, 2011.

[3] J.P. Rolland et al. A Survey of Tracking Technologies for Virtual Environments. In *Fund. of Wearable Computers and Augmented Reality*, pages 67–112, 2001.

[4] D. Thalmann and S. R. Musse. *Crowd Simulation*. Springer, London, 2007.

[5] Y. Zhang et al. Online Inserting Virtual Characters into Dynamic Video Scenes. *Comp. Ani. and Virt. Worlds*, 22:499–510, 2011.

[6] A. Egges et al. Presence and Interaction in Mixed Reality Environments. *The Vis. Comp.*, 23:317–333, 2007.

[7] G. Papagiannakis et al. Real-time Virtual Humans in AR Sites. In *Proc. of IEE CVMP '04*, pages 273–276, 2004.

[8] I. Barakonyi et al. MonkeyBridge: Autonomous Agents in Augmented Reality Games. In *Proc. of ACM SIGCHI ACE '05*, pages 172–175, 2005.

[9] Z. Zhang. A Flexible New Technique for Camera Calibration. *IEEE Trans. on PAMI*, 22:1330–1334, 2000.

[10] G. Lowe. Three-Dimensional Object Recognition from Single Two-Dimensional Images. *Artificial Intelligence*, 31:355–395, 1987.

[11] J. van Den Berg et al. Reciprocal Velocity Obstacles for real-time multi-agent navigation. In *Proc. of ICRA '08*, pages 1928–1935, 2008.