Interactive Navigation of Multiple Agents in Crowded Environments*

Jur van den Berg Sachin Patil Jason Sewall Dinesh Manocha Ming Lin
University of North Carolina at Chapel Hill, USA
{berg, sachin, sewall, dm, lin}@cs.unc.edu
Project website: http://gamma.cs.unc.edu/RVO/NAVIGATE/

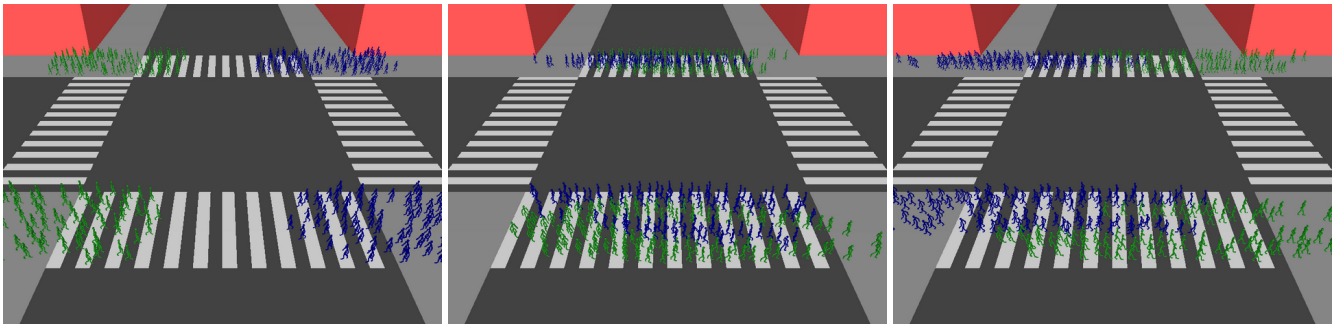

Figure 1: Lane Formation. *Starting from 4 corners of a crossroad, 400 agents cross the steets. As they move in the opposite directions, the agents automatically form lanes.*

Abstract

We present a novel approach for interactive navigation and planning of multiple agents in crowded scenes with moving obstacles. Our formulation uses a precomputed roadmap that provides macroscopic, global connectivity for wayfinding and combines it with fast and localized navigation for each agent. At runtime, each agent senses the environment independently and computes a collision-free path based on an extended "Velocity Obstacles" concept. Furthermore, our algorithm ensures that each agent exhibits no oscillatory behaviors. We have tested the performance of our algorithm in several challenging scenarios with a high density of virtual agents. In practice, the algorithm performance scales almost linearly with the number of agents and can run at interactive rates on multi-core processors.

1 Introduction

The interactions of large numbers of individuals and groups often lead to complex biological, social, and cultural patterns that we observe in nature and society. Modeling of the collective behaviors is an open research issue and is particularly not well understood for groups with *non-uniform spatial distribution* and *heterogeneous behavior characteristics*, such as pedestrian traffic in urban scenes, tourist exploration of an amusement park, and evacuation flows in complex structures with multiple exits.

One of the most challenging problems in modeling the collective behaviors of virtual characters in real-time 3D graphics applications is autonomous navigation and planning of multiple agents in crowded scenes with stationary and moving obstacles. Many of the existing techniques are not known to scale well to handle hundreds and thousands of independent agents of arbitrary distribution density and individual characteristics. The computational complexity of simulating multiple agents in crowded scenes arises from the fact that each moving agent is a dynamic obstacle to other agents and the planning problem becomes significantly harder [LaValle 2006]. In addition, the problem of collision avoidance becomes increasingly difficult, particularly in a highly cluttered environment.

Main Contributions: In his seminal work, Tolman [1948] suggested that animals and men explore and navigate around their environment using "cognitive maps", namely a spatial representation of the environment in their memory. This view, though controversial for animal spatial navigation, has been a popular belief for human navigation in the physical world. Building upon Tolman's hypothesis, we present a fast two-level planning method for real-time navigation of many agents in a crowded virtual environment based on hierarchical spatial behaviors observed in humans and animals. We assume that each indvidual agent has a high-level cognitive map of the environment, which is modeled as a precomputed "roadmap" [Latombe 1991] of the entire scene in our approach. Roadmaps provide macroscopic, global connectivity information and serve as a wayfinding aid to guide the travel of each agent among stationary obstacles. During the run time, each agent will travel toward its goal using the precomputed roadmaps of the static environment and taking into account various constraints (such as speed limit, path clearance, shortest ways, fastest routes), while purposely avoiding collision with nearby obstacles or incoming agents.

Environment adaptivity of each agent is a dynamic process subject to local conditions. Our approach employs a lower-level behavior using a novel approach derived from a recent concept in robotics, called "Velocity Obstacle" [Fiorini and Shiller 1998; Shiller et al. 2001]. However, Velocity Obstacle can exhibit "oscillation" in a crowded workspace. We address this problem by incorporating the fact that each agent can sense its surrounding environment and react to incoming agents or dynamic obstacles. We extend the "Velocity Obstacles" concept by assuming that each agent is a decision making entity capable of selecting the appropriate velocity that responds to the other agent's movement and replanning its path accordingly. The resulting approach alleviates the common, well-known oscil-

*Supported in part by ARO Contracts DAAD19-02-1-0390 and W911NF-04-1-0088, NSF awards 0400134, 0429583, and 0636208, DARPA/RDECOM Contract N61339-04-C-0043, and Intel.

latory behaviors of Velocity Obstacle techniques for many agents in crowded scenes. In addition, the basic algorithmic framework allows easy incorporation of kinodynamic constraints, orientation and visibility regions of each agents, etc.

Our algorithm offers the following characteristics:

- Global precomputation using roadmaps computed directly from static 3D environments;
- Fast, decentralized, and local runtime planning for each agent;
- Reliable collision avoidance exhibiting smooth paths, with no observed oscillatory behavior;
- Scalable performance with empirically nearly linear dependence on the number of agents.

Our approach has been implemented and tested on several challenging scenarios with a high density of virtual agents, as shown in Fig. 1. Our two-level planning technique is simple, robust, and easily parallelizable for implementation on multi- and many-core architectures. It runs at interactive rates for environments consisting of several thousands to ten-thousands of agents and its performance scales well with the number of agents.

Organization: The rest of the paper is structured as follows. Section 2 defines the basic problem formulation and briefly discusses related work. Section 3 presents our approach. We demonstrate and analyze the performance of our algorithm using several interesting scenarios in Section 4 and Section 5 respectively. We conclude with possible future research directions in Section 6.

2 Background

In this section, we first give mathematical characterization of the problem of multi-agent navigation and planning in densely crowded scenes. We also give a brief overview of prior work in this area.

2.1 Problem Formulation

The main objective of this work is to address the following problem. Given an environment containing static and dynamic obstacles, and a large number of agents with specified goal positions, the task is to navigate each of these agents to its goal position without collisions with obstacles or other agents in the environment. We also require that each agent navigates *independently* in a decentralized manner. That is, there should be no global runtime controller that coordinates the motions of all agents.

This is a challenging task, particularly in dense packed, crowded scenarios with several hundreds or thousands of agents. Each agent essentially has to navigate through an unknown dynamic environment; it has no prior knowledge of how other agents or the dynamic obstacles will move in the future. The standard approach to this class of problems is to let the agent run a continuous cycle of sensing and acting. In each cycle, the agent observes its surroundings, thereby acquiring information about the positions and velocities of other agents and obstacles, and based upon this information, the agent decides how to move locally. If the cycle is run at a high frequency, the agent is able to react timely on changes in its surroundings.

This formulation reposes the problem to the question of how each agent should decide to move within each cycle, given its perception of the local environment. It is formally defined as follows. For simplicity, assume that each agent is modeled as a disc. Given n agents $A_1, \ldots, A_n$, let each agent A_i has a radius r_i, a position $\mathbf{p}_i$ (defined by the center of the disc), a velocity $\mathbf{v}_i$, an orientation θ_i, a preferred speed v_i^{pref}, and a goal position $\mathbf{g}_i$. Furthermore, given a set of obstacles $\mathcal{O}$ that are modeled as polygons, each obstacle $O \in \mathcal{O}$ has a position $\mathbf{p}_O$ (defined by the reference point of the polygon) and a velocity $\mathbf{v}_O$. Static obstacles have zero velocity. Let the duration of the sensing-acting cycle be Δt for each agent.

For each agent A_i, we need to perform the following computation during each cycle. Given its current state (i.e., its position $\mathbf{p}_i$, velocity $\mathbf{v}_i$, etc.), we assume that the radii, the positions and the velocities of other agents can be obtained by sensing. The main goal is to compute the new velocity that needs to be adopted until the next cycle. The velocity should be chosen such that the agent eventually reaches its goal position $\mathbf{g}_i$ and avoids collisions with obstacles or other agents.

This problem becomes more challenging when the agent is subject to *kinodynamic constraints*, i.e. kinematic and dynamic constraints that restrict the set of admissible agent movements (e.g. velocities). That is, given a current velocity $\mathbf{v}_i$, there is only a restricted set $K(\mathbf{v}_i, \theta_i, \Delta t)$ of velocities that are feasible in the next cycle. Also, we may restrict the information each agent has regarding the other agents. For instance, the agent may only have information about the agents it can actually see, given its current position and orientation.

2.2 Related Work

Next we briefly discuss some of the prior literature in the areas.

Multiple Moving Entities: The problem of motion planning among multiple agents moving simultaneously in an environment is challenging because of the addition of the number of degrees of freedom which becomes large. The problem was proved to be intractable [LaValle 2006]. It is a specific case of the general time-varying problem. Two key approaches exist to address it: centralized and decoupled planning.

The *centralized* approaches [Latombe 1991; LaValle 2006] consider the sum of all the robots as a single one. For that, configuration spaces of individual robots are combined (using Cartesian product) in a composite one, in which a solution is searched. As the dimension of the composite space grows with the number of degrees of freedom added by each entity, the problem complexity becomes prohibitively high.

Contrarily, the *decoupled* planners proceed in a distributed manner and coordination is often handled by exploring a *coordination space*, which represents the parameters along each specific robot path. Decoupled approaches [Simeon et al. 1999; Warren 1990] are much faster than centralized methods, but may not be able to guarantee completeness.

Dynamic Environments: Evolving elements significantly increase the difficulty of the motion planning problem. In fact, motion planning for a single disc with bounded velocity among rotating obstacles is PSPACE-hard [LaValle 2006]. However, there have been many attempts to provide practical methods to cope with changing environments. For example, Stentz et al. proposed the D* deterministic planning algorithm to repair previous solutions instead of re-planning from scratch [Stentz 1995; Koenig and Likhachev 2002].

There have been two main approaches for adapting randomized planners to dynamic environments [LaValle and Kuffner 2001; Hsu et al. 2002]. The first one includes both PRMs and RRTs that reuse previously computed information to aid in finding a new path [Leven and Hutchinson 2000; Kallmann and Mataric 2004; Jaillet and Simeon 2004; Ferguson et al. 2006]. The second integrates obstacle motion directly into the planning process. Some

variations plan directly in a C-space augmented with a time parameter [Petty and Fraichard 2005].

Rather than changing the roadmap, other work for dynamic environments has focused on adjusting or modifying the path. Potential field planners use gradient descent to move toward a goal, at a potential sink [Khatib 1986]. Building on these ideas, several variation of dynamic changing or elastic roadmaps have been proposed [Quinlan and Khatib 1993; Yang and Brock 2006; Gayle et al. 2007].

Agent Simulation and Crowd Dynamics: Modeling of collective behaviors has been heavily studied in many fields [Helbing et al. 2003; Kamphuis and Overmars 2004; MASSIVE 2006; Schreckkenberg and Sharma 2001; Still 2000]. Simulation of multiple avatars agents and crowds have also been well studied in graphics and VR [Ashida et al. 2001; Shao and Terzopoulos 2005; Thalmann et al. 2006; Reynolds 2006; Treuille et al. 2006]. They differ based on problem decomposition (discrete vs continuous), stochastic vs deterministic, etc.

Numerous approaches have been proposed using variants of agent-based methods, rule-based techniques, and social force models [Reynolds 1987; Tu and Terzopoulos 1994; Musse and Thalmann 1997; Loscos et al. 2003; Sung et al. 2004; Cordeiro et al. 2005; Pelechano et al. 2005; Pelechano et al. 2007]. They focus mainly on the local planning of agents, and are known to exhibit emergent behaviors. Global methods using path planning algorithms have also been suggested [Funge et al. 1999; Bayazit et al. 2002; Lamarche and Donikian 2004; Pettre et al. 2005; Sung et al. 2005; Shao and Terzopoulos 2005] for mostly static environments. More recently algorithms have been proposed to extend the roadmap-based methods to dynamic environments and multiple agents [Garaerts and Overmars 2007; Li and Gupta 2007; Pettre et al. 2005; Gayle et al. 2007; Zucker et al. 2007] in relatively simple environments with only a few entities.

3 Multi-Agent Navigation

In this section, we describe our navigation and planning approach for multiple moving agents. Our approach is decomposed into two levels. The first deals with the global path planning towards the goal, and the second addresses local collision avoidance and navigation. We first describe each of them separately, and then show how these are integrated in a single navigation framework.

In order to simplify the navigation problem, we do not take the orientation of the agents into account. As the agents are modeled as discs, the navigation itself is a geometric problem for which the orientation of the agents is not important. However, for realistic motion of an agent, the orientation is important, and we show here how we deduce the orientation from the resulting motion. Further, we show how to incorporate kinodynamic constraints into our approach and how we deal with dynamic obstacles in the environment. In Fig. 2, a schematic overview of our algorithm is given.

3.1 Global Path Planning

In order to compute a global path to the goal for each agent, we do not take into account the presence of other agents, but only (large) static obstacles, e.g. buildings on a university campus, or walls in an office environment. We extract the global path from a *roadmap* that has been created around these static obstacles in a preprocessing phase. Any roadmap that covers the connectivity of the free space well is suitable. For example, we can compute a roadmap based on random sampling methods [LaValle 2006]. In 2D environments

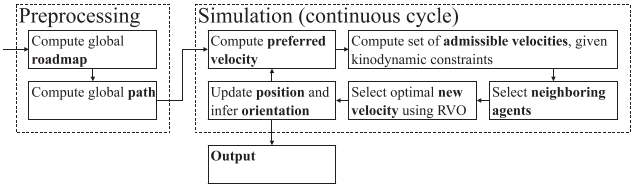

Figure 2: A schematic overview of our algorithm. *As part of a preprocess, we compute a roadmap with static obstacles and represent it as a graph. Different components of the runtime local navigation algorithms are shown in the right hand side box.*

with polygonal obstacles we can use the *visibility graph* [LaValle 2006], as it provides shortest paths for the agents.

Given a roadmap R with a set of nodes N and edges connecting nodes, and the goal positions $\mathbf{g}_i$ of each of the agents, we precompute for each agent the shortest path tree in the roadmap. (Other optimization factors, such as time or the number of turns, can also be incorporated.) First, the goal position of the agent is connected to the roadmap, by adding an edge between the goal position and all nodes in the roadmap that are 'visible' (i.e., the straight-line does not intersect any static obstacle) from the goal position. Then, we use Dijkstra-algorithm to calculate the distances from the goal position to any node in the roadmap.

Given the current position $\mathbf{p}_i$ of an agent A_i during the simulation, the global path is computed by connecting the current position to a node of the roadmap that is visible from the current position. The global path can then be extracted from the shortest path tree computed in the preprocessing, augmented with the segment from the current position to the selected visible node. Among the visible nodes, we select the one for which the sum of the distance between the current position of the agent and the node, and the distance between the node and the goal according to the shortest path tree of agent A_i, is minimal. If the roadmap R is the visibility graph of the static obstacles, this roadmap then gives the shortest path for the agent from its current position to its goal position.

If, during the simulation, an agent loses sight of the next node in its global path, it replans a new path using the procedure above.

3.2 Local Collision Avoidance

In Section 2.1 we describe how the navigation problem can be reduced to selecting a new velocity for each agent in each cycle. Our goal is to choose this velocity such that the agent will not collide with other agents moving in the same environment. This is a challenging problem, as we only know the current velocities of the other agents, and not their future ones. Also, the agents are not able to communicate to coordinate their navigation.

A common solution to this problem is to assume that the other agents are dynamic obstacles whose future motions are predicted as linear extrapolations of their current velocities. The agent, say A_i, then selects a velocity that avoids collisions with these extrapolated trajectories. This is formalized by an elegant geometric formulation called the *Velocity Obstacle* of agent A_i [Fiorini and Shiller 1998; Shiller et al. 2001]. Similar concepts of using velocity information for collision avoidance have also been proposed in [Feurtey 2000; Reynolds 1999]) for steering behavior of virtual agents. It is defined as follows for an agent A_i, and another agent A_j, which is regarded as a moving obstacle maintaining its current velocity $\mathbf{v}_j$:

The velocity obstacle $VO^i_j(\mathbf{v}_j)$ of obstacle (agent) A_j to agent A_i is defined as the set consisting of all those velocities $\mathbf{v}_i$ for A_i

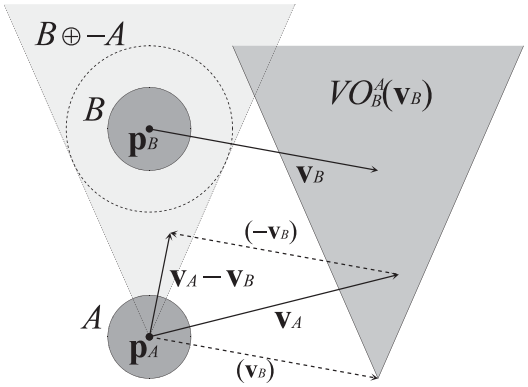

Figure 3: The Velocity Obstacle $VO_B^A(\mathbf{v}_B)$ *of a disc-shaped obstacle* B *to a disc-shaped agent* A*.*

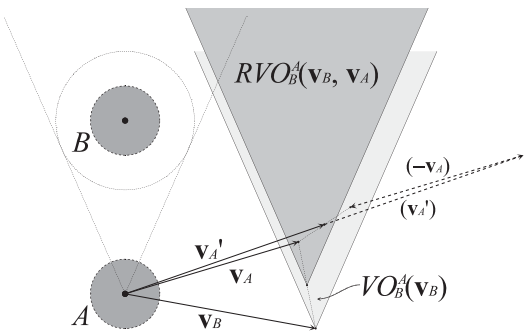

Figure 4: The Reciprocal Velocity Obstacle $RVO_B^A(\mathbf{v}_B, \mathbf{v}_A)$ *of agent* B *to agent* A*. It is used for local collision avoidance. We compute this locally for each agent during each simulation cycle.*

that will result in a collision at some moment in time with obstacle (agent) A_j moving at velocity $\mathbf{v}_j$ (see Figure 3).

Agent A_i can avoid a collision with obstacle (agent) A_j by selecting a velocity outside the velocity obstacle. However, the assumption that agent A_j is a passively moving obstacle overlooks the fact that agent A_j is actually an autonomous decision-making entity as well that reacts on the presence of agent A_i in turn. It has been shown that this causes *oscillatory* motions of the agents as a result [Abe and Matsuo 2001; Feurtey 2000; Kluge and Prassler 2007].

Our formulation builds on the theoretical framework of van den Berg et al. [van den Berg et al. 2008], which provides a principle to select a velocity for agent A_i and implicitly assumes that the other agents A_j use similar collision avoidance reasoning. Informally speaking, this means that agent A_i does only half of the effort to avoid a collision with agent A_j, and assumes that the other agent will take care of the other half. This is formalized into the principle of *Reciprocal Velocity Obstacles*. The Reciprocal Velocity Obstacle $RVO_j^i(\mathbf{v}_j, \mathbf{v}_i)$ of agent A_j to agent A_i is defined as the set consisting of all those velocities $\mathbf{v}_i$ for A_i that will result in a collision at some moment in time with agent A_j, *if* agent A_j chooses a velocity in its Reciprocal Velocity Obstacle as well (see Figure 4).

In other words, if A_i chooses its velocity outside the Reciprocal Velocity Obstacle, A_i is guaranteed to avoid collisions with agent A_j, *provided* that A_j applies the same navigation algorithm, and chooses a new velocity outside its Reciprocal Velocity Obstacle. As there are multiple agents around, each agent computes its Reciprocal Velocity Obstacle as the *union* of the Reciprocal Velocity Obstacles imposed by the individual agents, and selects a velocity outside this union to avoid collisions. Also, the Reciprocal Velocity Obstacle method is guaranteed to avoid oscillatory behavior of the agents. We refer the reader to [van den Berg et al. 2008] for more details and proofs of these properties.

3.3 Kinodynamic Constraints

In addition to geometric constraints, each agent A_i may be subject to kinodynamics constraints, i.e. kinematic and dynamic constraints that restrict the set of admissible new velocities, given its current velocity $\mathbf{v}_i$ and possibly its orientation θ_i. We denote this set $K(\mathbf{v}_i, \theta_i, \Delta t)$. It may have any shape depending on the nature of the agent. For example, if the agent is subject to a maximum speed $v_i^{\max}$ and a maximum acceleration $a_i^{\max}$, the set of admissible velocities is:

$$K(\mathbf{v}_i, \theta_i, \Delta t) = \{\mathbf{v}_i' \mid \|\mathbf{v}_i'\| < v_i^{\max} \wedge \|\mathbf{v}_i' - \mathbf{v}_i\| < a_i^{\max}\Delta t\}.$$

More complicated constraints may restrict the set of admissible velocities depending on the orientation θ_i of the agent, for instance when car-like kinematics are applied.

3.4 Selecting Neighbors

For locally avoiding collisions with other agents, we do not need to take into account the Reciprocal Velocity Obstacles induced by all other agents for two reasons. First, other agents that are far away are not likely to affect the motion of the agent, yet taking it into computation takes time. By only taking into account nearby agents, each agent only has to deal with a small number of agents, which significantly reduces the running time of the algorithm in each cycle. The second reason is that it may not be realistic to assume that the agent knows the position and velocity of all other agents far away. Instead, we can only take into account other agents that the agent can actually see, given its current position and orientation.

3.5 Integration of Global and Local Planning

To integrate the global path planning with local collision avoidance, we proceed as follows. As a first step, we compute for each agent A_i its *preferred velocity* $\mathbf{v}_i^{\text{pref}}$. This is the vector with a magnitude equal to the preferred speed of the agent in the direction of the next node along the agent's global path to the goal. If the agent is close to its goal, we set the preferred velocity to the null vector.

Subsequently, we select for each agent A_i a new velocity $\mathbf{v}_i'$. Ideally, this is the velocity closest to $\mathbf{v}_i^{\text{pref}}$ that is outside the Reciprocal Velocity Obstacle of A_i and inside the set of admissible new velocities. However, the environment may become so crowded that the Reciprocal Velocity Obstacle fills up the entire set of admissible velocities. To address this issue, the algorithm is allowed to select a velocity inside the Reciprocal Velocity Obstacle, but is penalized by this choice. The penalty of a candidate velocity $\mathbf{v}_i'$ depends on its deviation from the preferred velocity and on the expected *time to collision* $tc_i(\mathbf{v}_i')$, this velocity will give:

$$penalty_i(\mathbf{v}_i') = w_i \frac{1}{tc_i(\mathbf{v}_i')} + \|\mathbf{v}_i^{\text{pref}} - \mathbf{v}_i'\|,$$

for some factor w_i, where w_i can vary among the agents to reflect differences in aggressiveness and shyness. The expected time to collision $tc_i(\mathbf{v}_i')$ can easily be calculated, given the definition of the Reciprocal Velocity Obstacles.

We select the velocity with minimal penalty among all velocities in $K(\mathbf{v}_i, \theta_i, \Delta t)$ as the new velocity $\mathbf{v}_i'$ for agent A_i:

$$\mathbf{v}_i' = \arg \min_{\mathbf{v}_i'' \in K(\mathbf{v}_i, \theta_i, \Delta t)} penalty_i(\mathbf{v}_i'').$$

We approximate this minimum by sampling a number N of velocities evenly distributed over the set of admissible velocities. Even though the chosen velocity might not be absolutely safe, empirical results indicate that this is resolved in subsequent time steps of the simulation, where agents can again choose a new velocity.

3.6 Inferring Orientation

The above integrated global path planning and local collision avoidance deals with the geometric problem of collision-free navigation of each of the agents. As the agents are modeled by discs, the orientation of the agents is not important for this shape. However, for the purpose of selecting neighbors and computing the set of admissible velocities, the orientation of the agent is an intrinsic parameter of the agent's state.

Therefore, we update the orientation θ_i of agent A_i in each cycle by inferring it from the motions followed by the agent. In most cases, the orientation of the agent is the same as the direction of the velocity vector. However, this is not always the case, particularly in crowded environments, where, for instance, humans, make steps backwards, sidewards, etc. We use a simple principle to infer the orientation of the agent; we compute it as a weighted average of the preferred velocity and the actual velocity.

$$\theta_i = \arctan(\alpha \mathbf{v}_i + (1-\alpha)\mathbf{v}_i^{\text{pref}}), \text{ for } 0 \leq \alpha \leq 1.$$

3.7 Static and Dynamic Obstacles

We have seen how we can navigate agents among each other, but the environment may be inhabited by some independent moving obstacles as well, that do not perceive their surroundings, and do not react on the presence of agents. Also, even though the global path planning plans around the static obstacles, the local collision avoidance may deviate the agent from this path. Therefore, both the dynamic and the static obstacles have to be taken into account in the local navigation. We perform this step by adding the *velocity obstacles* (not the Reciprocal Velocity Obstacles) induced by the obstacles to the union of the Reciprocal Velocity Obstacles induced by the other agents.

4 Implementation and Performance

We have implemented our algorithm on a PC with multi-core CPUs. Each agent is simulated as an independent entity and we perform localized computation. As observed in our runtime experiments, the total running time to process all agents in each cycle is nearly linear in the number of agents. Since each agent performs an independent computation, our approach is also fully parallelizable. The total running time scales down (almost) linearly with the number of processors used.

4.1 Benchmarks

We investigated the performance our algorithm on three challenging benchmarks:

- **Stadium Scene:** 250 agents are waiting outside a stadium (see Fig. 5). They all have to get in through four narrow entrances, and form the word "I3D 2008" on the field. This scenario is particularly interesting for the parts where the agents form a dense group in front of the entrances, and how they interact on the field when they cross each other's trajectories. In this scenario, we use a small roadmap containing 8 nodes for global path planning. The nodes are located on the inside and outside of the four entrances of the stadium.

- **Office Evacuation:** $1,000$ agents are evacuated from an office floor of a building (see Fig. 6). They all have to escape through two narrow exits. This environment becomes very crowded and it is interesting to observe the flow in the crowd while it is progressing. We use a large roadmap containing 433 nodes for global path planning. Each node is located on a corner of the static environment.

- **Crosswalks in a City:** 100 agents are waiting on each corner of a busy crossroads in an outdoor city environment (see Fig. 7). At the moment their traffic light turns green, they are allowed to cross the street. On either side of the street, two groups of 100 agents move in opposite directions on the crosswalk. This scenario is of particular interest to see how our algorithm deals with these opposite flows. There is no roadmap for global path planning in this scenario (i.e., all agents attempt to move directly to their goals).

4.2 Nearly Linear Function of the Number of Agents

During each simulation cycle, each of the n agents probes a constant number of candidate velocities, and evaluates these velocities against the Reciprocal Velocity Obstacles of the $n-1$ other agents. However, we can reduce this entire computation to a nearly linear running time by only selecting a restricted subset of nearby agents (say the k nearest agents). Therefore, the total running time of our algorithm per cycle of the simulation depends mainly on two steps: the neighbor search to select the group of nearby agents for each agent and the local collision avoidance using RVO. In comparison, the cost of global path planning by connecting to the nearby node is insignificant.

Given a fixed constant number of at most k nearby agents, the RVO computation runs in $O(n)$ time per simulation cycle, given n agents. Obviously one of the key steps in affecting the overall performance of the algorithm is how to select the neighboring agents during each cycle. Our implementation uses a naive neighbor selecting scheme. That is, it iterates over all other agents and only selects the k nearest ones. Hence, our current implementation poses an inherently quadratic running time in the number of agents. A more efficient, linear-time implementation for nearest neighbor search can be employed based on spatial hashing schemes [Overmars 1992] as we described in Section 3.4. However, in practice, the step of selecting nearest k-neighbors is negligible in running time and other memory allocation costs dominate the overall runtime performance of our existing neighbor selection implementation. Thus, the overall neighbor selection step also exhibits an almost linear runtime behavior per simulation step.

We verify the performance of our algorithm empirically with experiments in the Office Evacuation scenario. We select a varying number n of agents and position them randomly across the office building floor. Each agent has to leave the building using the nearest exit. We performed the experiment for up to $20,000$ agents, for which the building becomes really crowded. We ran the computation of all agents on a single Intel Xeon X7350 2.93 GHz with 8 GByte of memory and 16 cores. As stated above, the total running time increases nearly linearly with the number of agents. The results are shown in Fig. 8.

As the figure indicates, the running time of our algorithm increases nearly linearly with the number of agents. Even though there is an inherently quadratic runtime behavior in one step of our current implementation of neighbor selection, this effect is hardly observable in the plots. Even for $20,000$ agents, the running time on 16 cores roughly corresponds to a frame rate of 2 frames per second.

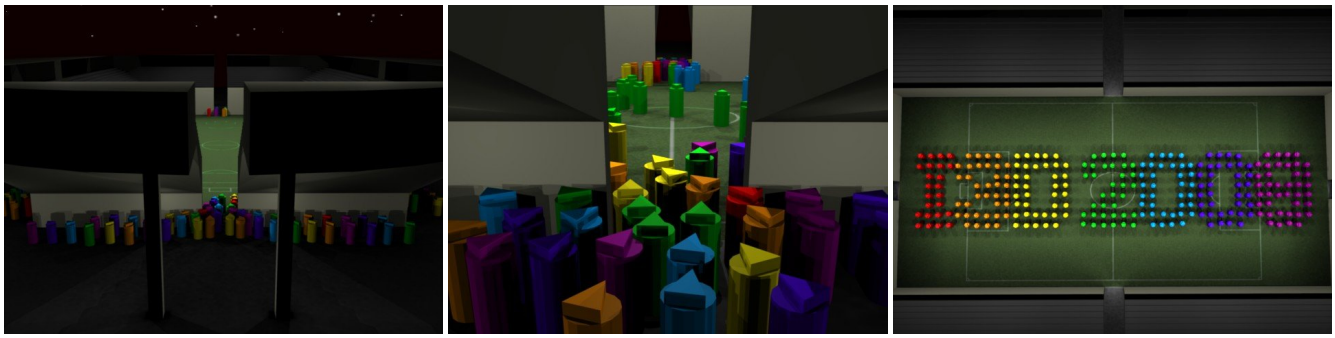

Figure 5: The Stadium Scene. *250 agents form the word "I3D 2008" on the field after entering the stadium. Congestion develops before the entrances of the stadium.*

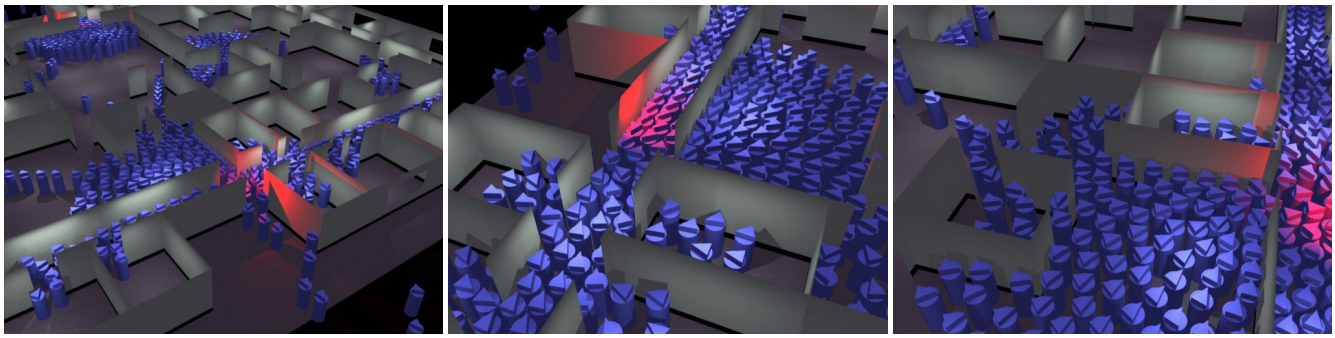

Figure 6: The Office Evacuation Scenario. $1,000$ *agents have to evacuate an office floor building through two narrow exits. The agents get very densely packed in their attempt to leave the building.*

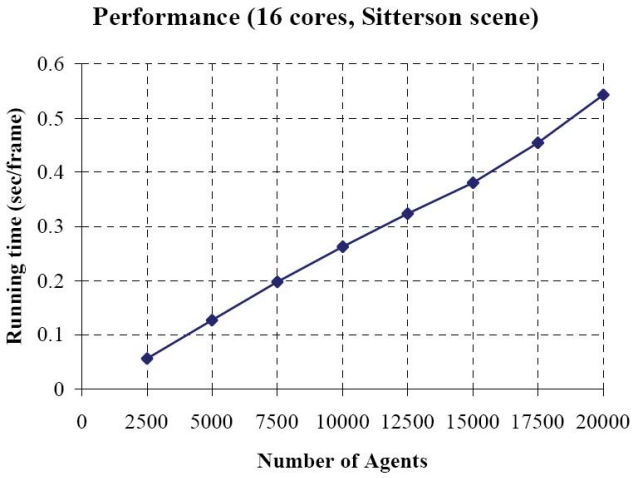

Figure 8: The performance of our algorithm as a function of the number of agents. *The experiment was run with an 16-core PC in the Office Evacuation scenario. The results are simulation-only.*

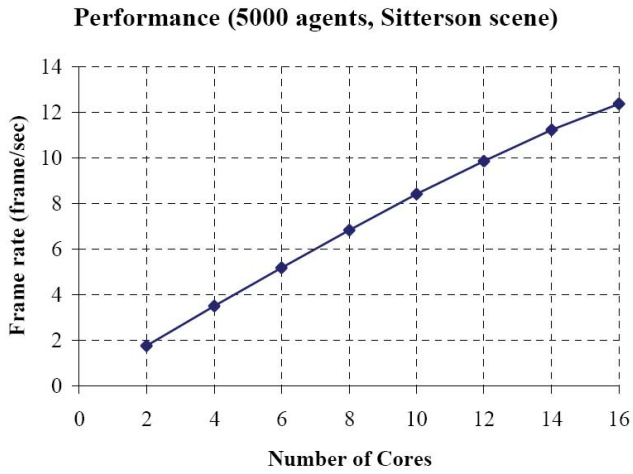

Figure 9: The performance of our algorithm as a function of the number of cores. *The experiment was run for* 5000 *agents in the Office Evacuation scenario. The results are simulation-only.*

4.3 Multi-Core Parallelization

Our algorithm is completely decoupled and localized, i.e. each agent performs its own computations without coordination with other agents. In principle, each agent could carry its own sensing and processing equipment, and run the algorithm as described above. This is analogous to the way individual humans navigate in a shared environment, where each of them makes its own observations and decisions, without explicit communication with others. For this reason, our algorithm is easily parallelizable, as long as each agent is able to observe the same environment and the positions and velocities of other agents in the environment. We prove this assertion by running the office benchmark for a fixed number of agents using a varying number of processing cores. The results are demonstrated in Fig. 9.

As the figure indicates, the running time of our algorithm decreases almost linearly (i.e., the frame rate increases nearly linearly) with the number of processing cores used. We observe that the latter cores add slightly less performance increase than the other cores. This effect may be due memory contention or scheduling overhead of the parallelization.

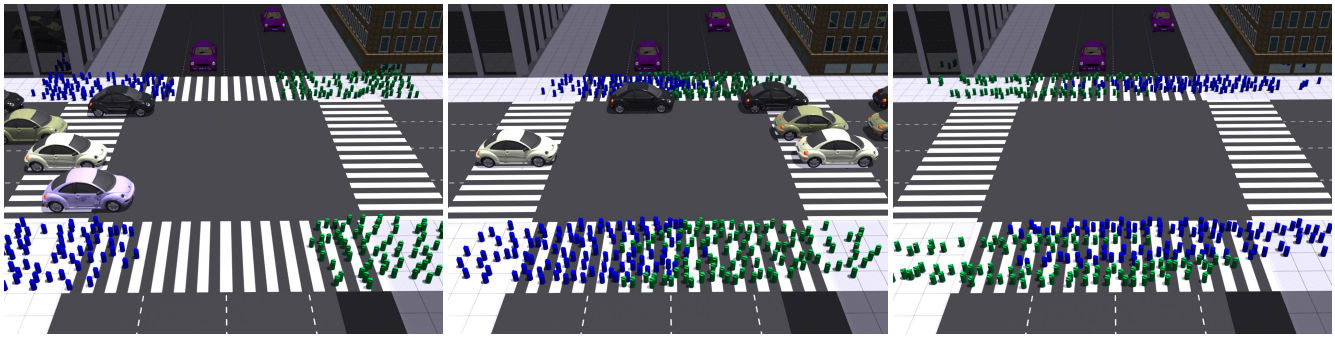

Figure 7: The Pedestrian Crosswalks. *400 agents distributed near the corners of the crossroads move to the other side of the street. The opposite flows of agents automatically form lanes, as they cross each other.*

5 Analysis

We ran our algorithm on the three benchmarks as introduced above. The resulting simulation can be seen in the video accompanying this paper.

In the stadium scenario, we observe that the agents congest in front of the entrances of the stadium, as one would expect in practice. Also, agents pass each other on the field in a natural way. They slightly adapt their trajectories in order to pass each other safely. We see similar behavior in the building office scenario. Even though the environment becomes extremely crowded, the congestion and the flow of the agents look natural. In the crosswalk scenario, we observe the interesting phenomenon of *automatic lane formation* when two groups of agents have opposite goal directions. Even though the number of agents is equal on both sides of the street, we see three lanes forming on one side, and four on the other side of the street. This phenomenon is caused by the random initial positioning of the agents, as expected. The automatic lane formation behavior is observed in reality, and has been studied extensively in the field of social sciences and for crowd simulation (see e.g. [Helbing et al. 2003; Schreckkenberg and Sharma 2001; Treuille et al. 2006]).

The reason that our approach works well in general, and in crowded scenarios in particular, is that agents do not repel each other, as is the case in many approaches based on particle systems and social force models. Each agent just selects a velocity, given the velocities of the other agents, that leads it closer to its goal and avoids collisions with other agents. In none of the benchmark scenarios, we have observed simulation instabilities, or agents getting trapped in local minima, even for relatively large simulation time steps.

A limitation of the approach is that in case two agents have exactly opposite directions, and are in a narrow passage, the agents may do a lengthy "reciprocal dance" before they are able to pass each other. Although this phenomenon is quite natural in reality, humans are always able to resolve the situation quickly. The reciprocal dance is explained by the fact that one of the agents tries to pass the other on the left side, and the other agent tries to pass the first one on the right side. When both agents learn that this is not possible, they both try to do the exact opposite. In slightly less congested environments, such as shown in the crosswalk scenario, this problem rarely occurs.

Another limitation, which at the same time is an advantage, is that we did not put in an exhaustive list of "rules of thumb" in navigation that are observed in reality. As a result, our algorithm can possibly produce some motions that may not be as realistic in a few cases. For instance, in the crosswalk scenario one agent is captured by the flow in its opposite direction, and is only able to escape when the stream of agents thins out. In the Stadium scenario some agents "overshoot" the entrance, and have to turn back to reach it. These may be realistic behaviors in a stampede scenario, but are not often seen on a crosswalk or at a stadium. Other artifacts stem from the fact that in our current implementation agents also adapt their motion to agents behind them. A more advanced neighbor selecting scheme, for instance based on visibility may remedy this effect.

On the other hand, our approach is simple in its formulation and implementation. The driving concept is our Reciprocal Velocity Obstacles concept, and no additional "intelligence" with a complicated set of rule distinctions has been added to influence the results on the case-by-case basis.

6 Conclusion and Future Work

In this paper, we have introduced a multi-agent navigation algorithm using a simple yet effective combination of high-level and low-level methods that model human spatial navigation:

- A pre-computed roadmap for global path planning; and
- Reciprocal Velocity Obstacles for local navigation and collision avoidance.

We have shown that our method scales well (nearly linearly) with an increasing number of agents and that it is especially well suited for parallelization – its performance increases almost linearly with the number of processors used.

For future research, we plan to validate our approach with live video capture of real-world scenarios similar to the examples shown in this paper. Our initial findings using some footage [1] found on the web indicate that our approach is capable of simulating the lane-formation behaviors observed on pedestrian crosswalks in the city.

We would also like to investigate techniques that can automatically integrate human dynamics constraints and other human behaviors with global planning and local navigation to generate more realistic human motion. Although some initial research has been conducted on this topic for a few tens of agents [Lau and Kuffner 2006], little is done for a large number of virtual humans in a crowded scene.

Another interesting addition is to adapt the global path planning based on local information. For instance, a crowd of agents may block a passage between two static obstacles. An agent planning to pass through this passage should detect this and automatically replan its global path, for example using methods from [Stentz 1995; Koenig and Likhachev 2002].

[1] http://www.youtube.com/watch?v=EGKnGthGK4M.

References

ABE, Y., AND MATSUO, Y. 2001. Collision avoidance method for multiple autonomous mobile agents by implicit cooperation. In *Proc. IEEE Int. Conf. on Robotics and Automation*, 1207–1212.

ASHIDA, K., LEE, S. J., ALLBECK, J., SUN, H., BADLER, N., AND METAXAS, D. 2001. Pedestrians: Creating agent behaviors through statistical analysis of observation data. *Proc. Computer Animation*.

BAYAZIT, O. B., LIEN, J.-M., AND AMATO, N. M. 2002. Better group behaviors in complex environments with global roadmaps. *Int. Conf. on the Sim. and Syn. of Living Sys. (Alife)*, 362–370.

CORDEIRO, O. C., BRAUN, A., SILVERIA, C. B., MUSSE, S. R., AND CAVALHEIRO, G. G. 2005. Concurrency on social forces simulation model. *First International Workshop on Crowd Simulation*.

FERGUSON, D., KALRA, N., AND STENTZ, A. 2006. Replanning with RRTs. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)* (May).

FEURTEY, F. 2000. *Simulating the Collision Avoidance Behavior of Pedestrians*. Master's thesis, University of Tokyo.

FIORINI, P., AND SHILLER, Z. 1998. Motion planning in dynamic environments using velocity obstacles. *Int. Journal of Robotics Research 17*, 7, 760–772.

FUNGE, J., TU, X., AND TERZOPOULOS, D. 1999. Cognitive modeling: Knowledge, reasoning and planning for intelligent characters. *Proc. of ACM SIGGRAPH*, 29–38.

GARAERTS, R., AND OVERMARS, M. H. 2007. The corridor map method: Real-time high-quality path planning. In *ICRA*, 1023–1028.

GAYLE, R., SUD, A., LIN, M., AND MANOCHA, D. 2007. Reactive deformation roadmaps: Motion planning of multiple robots in dynamic environments. In *Proc IEEE International Conference on Intelligent Robots and Systems*.

HELBING, D., BUZNA, L., AND WERNER, T. 2003. Self-organized pedestrian crowd dynamics and design solutions. *Traffic Forum 12*.

HSU, D., KINDEL, R., J.-C.LATOMBE, AND ROCK, S. 2002. Randomized kinodynamic motion planning with moving obstacles. *International Journal of Robotics Research*.

JAILLET, L., AND SIMEON, T. 2004. A PRM-based motion planning for dynamically changing environments. *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

KALLMANN, M., AND MATARIC, M. 2004. Motion planning using dynamic roadmaps. *Proceedings of the IEEE Conference on Robotics and Automation (ICRA)* (April).

KAMPHUIS, A., AND OVERMARS, M. 2004. Finding paths for coherent groups using clearance. *Proc. of ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, 19–28.

KHATIB, O. 1986. Real-time obstable avoidance for manipulators and mobile robots. *IJRR 5*, 1, 90–98.

KLUGE, B., AND PRASSLER, E. 2007. Reflective navigation: Individual behaviors and group behaviors. In *Proc. IEEE Int. Conf. on Robotics and Automation*, 4172–4177.

KOENIG, S., AND LIKHACHEV, M. 2002. Improved fast replanning for robot navigation in unknown terrain. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)* (May).

LAMARCHE, F., AND DONIKIAN, S. 2004. Crowd of virtual humans: a new approach for real time navigation in complex and structured environments. *Computer Graphics Forum 23*, 3, 509–518.

LATOMBE, J. 1991. *Robot Motion Planning*. Kluwer Academic Publishers.

LAU, M., AND KUFFNER, J. J. 2006. Precomputed search trees: Planning for interactive goal-driven animation. In *Proc. of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer animation*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 299–308.

LAVALLE, S., AND KUFFNER, J. 2001. Randomized kinodynamic planning. *International Journal of Robotics Research*.

LAVALLE, S. M. 2006. *Planning Algorithms*. Cambridge University Press (also available at http://msl.cs.uiuc.edu/planning/).

LEVEN, P., AND HUTCHINSON, S. 2000. Toward real-time path planning in changing environments. *Proceedings of the fourth International Workshop on the Algorithmic Foundations of Robotics (WAFR)*.

LI, Y., AND GUPTA, K. 2007. Motion planning of multiple agents in virtual environments on parallel architectures. In *ICRA*, 1009–1014.

LOSCOS, C., MARCHAL, D., AND MEYER, A. 2003. Intuitive crowd behaviour in dense urban environments using local laws. In *Theory and Practice of Computer Graphics (TPCG'03)*, 122–129.

MASSIVE, 2006. http://www.massivesoftware.com.

MUSSE, S. R., AND THALMANN, D. 1997. A model of human crowd behavior: Group inter-relationship and collision detection analysis. *Computer Animation and Simulation*, 39–51.

OVERMARS, M. H. 1992. Point location in fat subdivisions. *Inf. Process. Lett. 44*, 261–265.

PELECHANO, N., O'BRIEN, K., SILVERMAN, B., AND BADLER, N. 2005. Crowd simulation incorporating agent psychological models, roles and communication. *First International Workshop on Crowd Simulation*.

PELECHANO, N., ALLBECK, J., AND BADLER, N. 2007. Controlling individual agents in high-density crowd simulation. *Proc. of ACM SIGGRAPH / Eurographics Symposium on Computer Animation (SCA)*.

PETTRE, J., LAUMOND, J.-P., AND THALMANN, D. 2005. A navigation graph for real-time crowd animation on multilayered and uneven terrain. *First International Workshop on Crowd Simulation*.

PETTY, S., AND FRAICHARD, T. 2005. Safe motion planning in dynamic environments. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 3726–3731.

QUINLAN, S., AND KHATIB, O. 1993. Elastic bands: Connecting path planning and control. *Proc. of IEEE Conf. on Robotics and Automation*.

REYNOLDS, C. W. 1987. Flocks, herds, and schools: A distributed behavioral model. In *Computer Graphics (SIGGRAPH '87 Proceedings)*, M. C. Stone, Ed., vol. 21, 25–34.

REYNOLDS, C. 1999. Steering behaviors for autonomous characters. In *Proc. Game Developers Conference*, 763–782.

REYNOLDS, C. 2006. Big fast crowds on ps3. In *sandbox '06: Proceedings of the 2006 ACM SIGGRAPH symposium on Videogames*, ACM Press, New York, NY, USA, 113–121.

SCHRECKKENBERG, M., AND SHARMA, S. D. 2001. *Pedestrian and Evacuation Dynamics*. Springer.

SHAO, W., AND TERZOPOULOS, D. 2005. Autonomous pedestrians. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ACM Press, New York, NY, USA, 19–28.

SHILLER, Z., LARGE, F., AND SEKHAVAT, S. 2001. Motion planning in dynamic environments: obstacles moving along arbitrary trajectories. In *Proc. IEEE Int. Conf. on Robotics and Automation*, 3716–3721.

SIMEON, T., LEROY, S., AND LAUMOND, J. 1999. Path coordination for multiple mobile robots: a geometric algorithm. *Proc. of IJCAI*.

STENTZ, A. 1995. The focussed D* algorithm for real-time replanning. *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.

STILL, G. 2000. *Crowd Dynamics*. PhD thesis, University of Warwik, UK. Ph.D. Thesis.

SUNG, M., GLEICHER, M., AND CHENNEY, S. 2004. Scalable behaviors for crowd simulation. *Computer Graphics Forum 23*, 3 (Sept), 519–528.

SUNG, M., KOVAR, L., AND GLEICHER, M. 2005. Fast and accurate goal-directed motion synthesis for crowds. *Proc. of SCA 2005*, 291–300.

THALMANN, D., O'SULLIVAN, C., CIECHOMSKI, P., AND DOBBYN, S. 2006. *Populating Virtual Environments with Crowds*. Eurographics 2006 Tutorial Notes.

TOLMAN, E. C. 1948. Cognitive maps in rats and men. *The Psychological Review 55*, 4, 189–208.

TREUILLE, A., COOPER, S., AND POPOVIC, Z. 2006. Continuum crowds. *Proc. of ACM SIGGRAPH*, 1160 – 1168.

TU, X., AND TERZOPOULOS, D. 1994. Artificial fishes: Physics, locomotion, perception, behavior. In *Proceedings of SIGGRAPH '94*, A. Glassner, Ed., 43–50.

VAN DEN BERG, J., LIN, M., AND MANOCHA, D. 2008. Reciprocal velocity obstacles for real-time multi-agent navigation. In *Proc. IEEE Int. Conf. on Robotics and Automation*.

WARREN, C. W. 1990. Multiple path coordination using artificial potential fields. *Proc. of IEEE Conf. on Robotics and Automation*, 500–505.

YANG, Y., AND BROCK, O. 2006. Elastic roadmaps: Globally task-consistent motion for autonomous mobile manipulation. *Proceedings of Robotics: Science and Systems* (August).

ZUCKER, M., KUFFNER, J., AND BRANICKY, M. 2007. Multipartite rrts for rapid replanning in dynamic environments. *Proc. IEEE Int. Conf. on Robotics and Automation*.