Real-time Navigation of Independent Agents Using Adaptive Roadmaps

Avneesh Sud* Russell Gayle* Erik Andersen* Stephen Guy* Ming Lin* Dinesh Manocha*

Dept of Computer Science, University of North Carolina at Chapel Hill

Abstract

We present a novel algorithm for navigating a large number of independent agents in complex and dynamic environments. We compute adaptive roadmaps to perform global path planning for each agent simultaneously. We take into account dynamic obstacles and inter-agents interaction forces to continuously update the roadmap by using a physically-based agent dynamics simulator. We also introduce the notion of 'link bands' for resolving collisions among multiple agents. We present efficient techniques to compute the guiding path forces and perform lazy updates to the roadmap. In practice, our algorithm can perform real-time navigation of hundreds and thousands of human agents in indoor and outdoor scenes.

1 Introduction

Modeling of multiple agents and swarm-like behaviors has been widely studied in virtual reality, robotics, architecture, physics, psychology, social sciences, and civil and traffic engineering. Realistic visual simulation of many avatars requires modeling of group behaviors, pedestrian dynamics, motion synthesis, and graphical rendering. In this paper, we address the problem of real-time motion synthesis for large-scale independent agents in complex, dynamic virtual environments. These agents may correspond to virtual or digital characters that consist of *non-uniform distributions* of many *distinct* entities, each with *independent* behavior, characteristics, and goals. Examples of such environments include virtual humans in large exposition halls, avatars in wide festival arenas, digital actors in busy urban streets, etc.

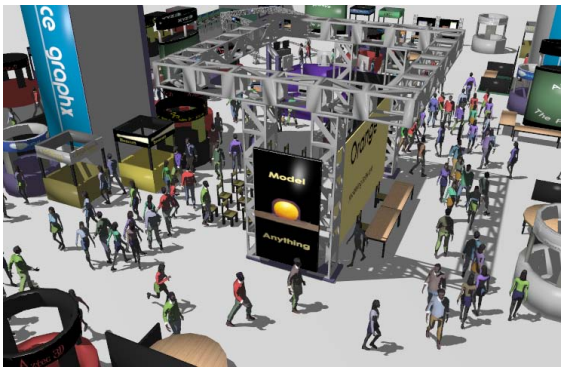

Figure 1: Navigation within an indoor environment: An exhibit hall of a trade show that consists of 511 *booths and* 1,000 *human agents. Each agent has a* distinct *goal (i.e. visiting one of the booths) and behavior characteristic. Our navigation algorithm based on AERO can compute collision-free paths simultaneously for all* 1,000 *agents at* 22 *fps on a PC with* 3*Ghz Pentium D CPU.*

*e-mail: {sud,rgayle,andersen,sjguy,lin,dm}@cs.unc.edu

A major challenge is automatic navigation of each agent through a complex, dynamic environment. More specifically, real-time global path computation for each agent can become a bottleneck, as the number of independent agents in the environment increases. The route planning problem can become intractable, as each individual character moving independently needs to perform collision avoidance with remaining agents. The problem of computing a collision-free path has been extensively studied in robot motion planning, crowd simulation and character animation. Prior global motion planning algorithms are mainly limited to static environments with a few moving robots. Most algorithms for dynamic scenes are based on local collision-avoidance methods, which suffer from convergence and local minima problems. Most of existing work in crowd simulation has been applicable to only a few groups or swarms of agents with the same goal, and not a large number of *independent* agents with *different* intentions [Treuille et al. 2006]. Our work is complementary to these work by addressing the navigation problem simultaneously for many virtual agents with distinct goals and individualized behavior characteristics.

Main Results: In this paper, we present a new algorithm for real-time navigation of large-scale heterogeneous agents in complex dynamic environments. Our approach is based on a novel representation called **"Adaptive Elastic ROadmaps" (AERO)**. AERO is a global connectivity graph that deforms based upon obstacle motion and inter-agent interaction forces.

We use AERO to perform dynamic, global path planning simultaneously for independent agents. We also take into account moving obstacles and the local dynamics among the agents. AERO continuously adapts to dynamic obstacles and deforms according to local force models and global constraints, in order to compute collision-free paths in complex environments. In addition, we introduce the notion of *link bands* to augment the local dynamics and resolve collisions among multiple agents. Due to lazy and incremental updates to the roadmap and efficient computation of guiding-path forces using link bands, our approach can scale to hundreds and thousands of individual agents and perform real-time global navigation of many independent agents in complex, changing environments, with generic obstacles and no restrictions on agent motion.

We demonstrate our approach on complex indoor and outdoor scenarios, including a city scene consisting of 2,000 pedestrians with 50 moving cars and an exhibition hall with 511 stationary booths and 1,000 individual agents on foot and avoiding each other. In order to highlight global navigation, we also place upto 1,000 agents in a dynamic maze environment. Our initial proof-of-concept implementation is able to perform motion simulation of independent agents for these highly challenging scenes in a fraction of a second per frame on a PC with an 3Ghz Pentium D CPU and 2GB memory. As compared to the prior approaches, our algorithm is perhaps among the first to interactively navigate upto thousands of *independent* agents each with *distinct* goals and individualized behavior characteristics, by providing real-time global path planning for all agents *simultaneously* and performing fast local collision avoidance among them.

Organization: The rest of the paper is organized as follows. Section 2 presents related work on multi-agent planning and crowd simulation. We describe "adaptive elastic roadmaps" in Section 3 and use them to navigate multiple virtual agents simultaneously in

Section 4. We describe our implementation and highlight its performance on different benchmarks in Section 5. We analyze the performance of our algorithm and compare it with earlier approaches in Section 6.

2 Related Work

In this section, we give a brief overview of prior work related to multi-agent planning, character animation and crowd simulation.

2.1 Multiple Agent Planning

Extensive literature exists on path planning for multiple agents in robot motion planning and virtual environments [LaValle 2006]. At a broad level, these methods can be classified into *global* (or centralized) and *local* (or distributed) methods. The global paths represent the connectivity of collision-free space in terms of a graph or a roadmap, and require search algorithms to compute a path for each agent [Bayazit et al. 2002; Funge et al. 1999; Kamphuis and Overmars 2004; Lamarche and Donikian 2004; Pettre et al. 2005; Sung et al. 2005; Sud et al. 2007]. Most roadmap based algorithms have been designed for motion planning for a single robot in a static environment and are generated based on random sampling techniques [LaValle 2006]. Recently, some algorithms have been proposed to extend the roadmap-based methods to dynamic environments, multiple agents and deformable models [Gayle et al. 2005; Garaerts and Overmars 2007; Li and Gupta 2007; Pettre et al. 2005; Rodriguez et al. 2006; Gayle et al. 2007; Zucker et al. 2007]. However, they have only been applied to relatively simple environments composed of a few robots and restricted obstacles. These approaches may not scale well to environments with a large number of independent agents.

As compared to global methods, local methods are mostly reactive style planners based on variants of potential fields [Khatib 1986]. They can handle large dynamic environments, but suffer from 'local-minima' problems and may not be able to find a collision-free path, when one exists [LaValle 2006]. Often these methods do not give any kind of guarantees on their behavior. Other route planning algorithms are based on path or roadmap modification, which allow a specified path for an agent to move or deform based upon obstacle motion. These methods include Elastic Bands [Quinlan and Khatib 1993] and Elastic Roadmaps [Yang and Brock 2006]. Our approach bears some close resemblance to these techniques, but AERO is lazily updated to deal with dynamic obstacles and is a significant extension of these algorithms to plan paths for multiple agents simultaneously. We will describe it in detail in Section 4.

2.2 Crowd Dynamics and Human Agents Simulation

Many different approaches have been proposed for modeling movement and simulation of multiple human agents or crowds or individual pedestrians. [Ashida et al. 2001; Schreckkenberg and Sharma 2001; Shao and Terzopoulos 2005; Thalmann et al. 2006; Reynolds 2006]. They can be classified based on specificity, problem decomposition (discrete vs continuous), stochastic vs deterministic, etc.

2.2.1 Discrete methods

Discrete methods rely on a sampling of the environment or of the agents. Some common approaches include:

Agent-based methods: These are based on seminal work of Reynolds [1987] and can generate fast, simple local rules that can create visually plausible flocking behavior. Numerous extensions have been proposed to account for social forces [Cordeiro et al. 2005], psychological models [Pelechano et al. 2005], directional preferences [Sung et al. 2004], sociological factors [MUSSE and Thalmann 1997], etc. Most agent-based techniques use local collision avoidance techniques and cannot give any guarantees on the correctness of global behaviors.

Cellular Automata methods: These methods model the motion of multiple agents by solving a cellular automaton. The evolution of the cellular automata at next time step is governed by static and dynamic fields [Hoogendoorn et al. 2000]. While these algorithms can capture emergent phenomena, they are not physically based. Different techniques for collision avoidance have been developed based on grid-based rules [Loscos et al. 2003] and behavior models [Tu and Terzopoulos].

Particle Dynamics: Computing physical forces on each agent is similar to N-body particle system [Schreckkenberg and Sharma 2001; Helbing et al. 2003]. Sugiyama et al. [2001] presented a 2D optimal velocity (OV) model that generalizes the 1D OV model used for traffic flow. Our formulation is built on some of these ideas and we elaborate them in Section 4.

2.2.2 Continuous Methods

The flow of crowds or multiple agents can be formulated as fluid flows. At low densities crowd flow is like gases, at moderate densities it resembles fluid flow, and at high densities crowd has been compared to granular flow [Helbing et al. 2005]. Most recently, a novel approach for crowd simulation based on continuum dynamics has been proposed by Treuille et al. [2006]. We compare our approach with these methods in Section 7.

3 AERO: Adaptive Elastic ROadmaps

In this section, we describe our representation for navigating heterogeneous agents. We first introduce the notation used in the rest of the paper. Next, we give an overview of our representation and present algorithms to compute it.

3.1 Definitions and Notation

We assume that the multiple agents are contained within a domain D. Each *agent* is denoted as p_i and let the environment consist of k agents, the set of all agents is denoted $\mathscr{A}$. We assume that each agent p_i has a finite radius r_i, a goal position denoted $\mathbf{g_i}$. The dynamics state of each agent at time t consists of its position $\mathbf{x}_i(t)$ and velocity $\mathbf{v}_i(t)$. For ease of notation, we shall not indicate the time dependency of the simulation terms as they are implicitly defined.

In addition to agents, the environment also consists of a set of static and dynamic obstacles. Each obstacle is denoted o_i and set of obstacles is denoted $\mathscr{O}$. The *free space*, is the empty space in the domain, and is given as $D_f = D \setminus (\mathscr{O} \cup \mathscr{A})$. The motion of each agent is restricted to the free space.

The unit normal vector from a point $\mathbf{p} \in D$ to an agent p_i is given by $\mathbf{n}_i(\mathbf{p}) = \frac{\mathbf{p}-\mathbf{x}_i}{\|\mathbf{p}-\mathbf{x}_i\|}$. The agent's *velocity bias field* $\phi_i(\mathbf{p})$ is defined as the angle between the normal to the agent and its velocity, $\phi_i(\mathbf{p}) = \cos^{-1}(\mathbf{n}_i(\mathbf{p}) \cdot \frac{\mathbf{v}_i}{\|\mathbf{v}_i\|})$. Given a pair of agents (p_i, p_j), we define the following: separation distance $d_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|$, separation normal $\mathbf{n}_{ij} = \frac{\mathbf{x}_i-\mathbf{x}_j}{d_{ij}}$. For the sake of simplicity, we assume that all agents have same radius, $r_i = r_j = r_a$ and our algorithm can be easily modified to account for varying radii.

We use the generalized Voronoi diagrams to compute proximity information for the roadmap and link bands. A *site* is a geometric

primitive in $\mathbb{R}^2$. Given a set of sites $\mathscr{S}$, and a domain D, the Voronoi region of a site s_i is denoted $\mathscr{V}(s_i|\mathscr{S})$, and the Voronoi diagram of all sites is VD($\mathscr{S}$). For our work, the sites are the edges on the roadmap, and the domain is the free space.

Multi-agent navigation problem: Given the state of each agent p_i at time t_0, and goal $\mathbf{g_i}$, we compute a sequence of states $\mathbf{q}_i(t_0), \mathbf{q}_i(t_1), \ldots, \mathbf{q}_i(t_f)$, such that $\mathbf{x}_i(t_f) = \mathbf{g_i}, \mathbf{x}_i(t_j) \in D_f(t_j)$ for all i, j. Our goal is to compute a collision-free path for each agent and ensure that its behavior conforms with prior results in pedestrian dynamics. Each agent is also assigned a desired velocity $\mathbf{v}_i^d$, with the magnitude equal to the maximum velocity, v_{max}, of the agent and direction determined by its state and the environment.

3.2 Global Path Planning

Our goal is to use global path planning methods that can help each agent to reach its goal. Prior global approaches are slow in terms of handling complex environments with hundreds of independent agents in real-time. At the same time, local methods can give no guarantees in terms finding a collision-free path and can get stuck in local minima. In order to overcome these problems, we use a novel path planning data structure called Adaptive Elastic ROadmaps (AERO). AERO provides a global roadmap that is updated instantaneously in response to motion of the agents and obstacles in the environment.

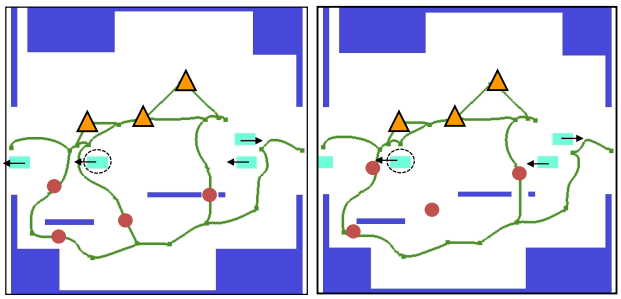

Figure 2: **Adaptive Elastic ROadmap (AERO)***: An example with 4 agents (red circles) and goals (yellow triangles). The static obstacles are dark blue rectangles and dynamic obstacles are cyan rectangles with arrows indicating direction of motion. The green curves represent links of the reacting deforming roadmap. The dynamic obstacles represent cars. As the highlighted car (circled) moves, the affected link in the roadmap is removed.*

AERO is a time-varying roadmap, or a connectivity graph of milestones ($\mathscr{M}$) and links ($\mathscr{L}$), $\mathscr{R} = \{\mathscr{M}, \mathscr{L}\}$, and is used to compute the collision-free guiding path for each agent. Each milestone is a position $\mathbf{x}_i \in D_f$, and each link l_{ab} connects two milestones $\mathbf{x}_a, \mathbf{x}_b$ along a path (see Fig. 2). A link l_{ab} is a closed (including the end points) curve in D_f. In path planning literature, this path is often a straight-line for simplicity. Each agent queries the roadmap to compute a path between two configurations in D_f by using a graph search algorithm (such as A*).

3.3 Particle-based graph representation

As the agents and dynamic obstacles move, we compute and update AERO using a physically-based particle simulator. The main components of the graph, dynamic milestones and adaptive links, are built from particles. Particle i, denoted m_i, is a point-mass in D_f which responds to applied forces. The state of the particle at time t is described by its position $\mathbf{x}_i(t)$ and velocity $\mathbf{v}_i(t)$. In the connectivity graph, the dynamic milestones $\mathscr{M}$ are each represented as a particle. Similarly, the adaptive links are represented using a sequence of particles connected by linear springs. The number of particles in each link can vary as a function of link length. As the obstacles (which may include other agents) move, the repulsive forces cause the milestones to move, and the links to deform toward the open areas of the navigation domain (as shown in Fig. 2).

3.4 Applied Force Computation

We apply forces to move the milestones and links away from obstacles while simultaneously maintaining the connectivity of the roadmap. For each particle i, we consider two forces for each particle, roadmap internal forces and repulsive external forces,

$$\mathbf{F_i} = \mathbf{F_i}^{int} + \mathbf{F_i}^{ext},$$

where $\mathbf{F}_i^{int}$ denotes the internal forces and $\mathbf{F}_i^{ext}$ describes external forces.

The internal forces maintain the length of the links, and are simulated using standard damped Hookean spring. Given two particles m_i and m_j, the force on particle m_i from an incident particle m_j is given as:

$$\mathbf{F_i}^{int} = -(k_s(\|\mathbf{x_{ij}}\| - L) + k_d(\frac{\mathbf{v_{ij}} \cdot \mathbf{x_{ij}}}{\|\mathbf{x_{ij}}\|}))\frac{\mathbf{x_{ij}}}{\|\mathbf{x_{ij}}\|},$$

where $\mathbf{x}_{ij} = \mathbf{x}_i - \mathbf{x}_j$, $\mathbf{v}_{ij} = \mathbf{v}_i - \mathbf{v}_j$, k_s is a spring constant, k_d is the damping constant, and L is the initial distance between the particles. To prevent the entire roadmap from drifting as a result of moving obstacles, additional springs are attached between the dynamic milestones and a particle is fixed at its initial location. Note that the dynamics milestones are not fixed - instead they are allowed to move to avoid dynamics obstacles.

The external force is repulsive potential force from the obstacles. For each obstacle o_j, we apply a force on particle i, m_i, if it is sufficiently close to o_j. This force given is:

$$\mathbf{F_i}^{ext} = \begin{cases} \frac{b}{d(m_i,o_j)}\mathbf{n} & \text{if } d(m_i,o_j) < \delta, \\ \mathbf{0} & \text{otherwise,} \end{cases}$$

where $d(m_i, o_j)$ is the minimum distance between a particle m_i and obstacle o_j, b is a repulsive scaling constant, $\mathbf{n}$ is the normal from the obstacle to the particle, and δ is a repulsive force threshold.

Given the applied forces, we update the state of AERO, the position and velocity of all particles, using numerical integration. In order to prevent undesirable oscillation in the adaptive links, Verlet integration is used [Verlet 1967; Jakobsen 2001]. This method considers the particles to be at rest, $\mathbf{v}(t) = \mathbf{0}$, during integration. Based on forward Euler integration and Newtonian motion, $\mathbf{F_i} = m\mathbf{a}$, the update rule for particle m_i with unit mass is given as:

$$\mathbf{x}_i(t+dt) = \mathbf{x}_i(t) + \frac{1}{2}\mathbf{a}_i(dt)^2 \quad = \mathbf{x_i}(t) + \frac{1}{2}(\mathbf{F_i}^{int} + \mathbf{F_i}^{ext})(dt)^2.$$

Since velocity is necessary to compute $\mathbf{F_i}^{int}$ we can locally approximate it as

$$\mathbf{v}_i(t+dt) = \frac{\mathbf{x}_i(t+dt) - \mathbf{x}_i(t)}{dt}$$

This formulation describes how a roadmap can adapt to the motion of obstacles. In order to compute the initial roadmap, we can use any of the well known methods in the motion planning literature [LaValle 2006]. In our current implementation, the initial roadmap is generated based on edges and vertices of the generalized Voronoi diagram of the free workspace. This provides good initial clearance from the obstacles and captures all the passages in the environment.

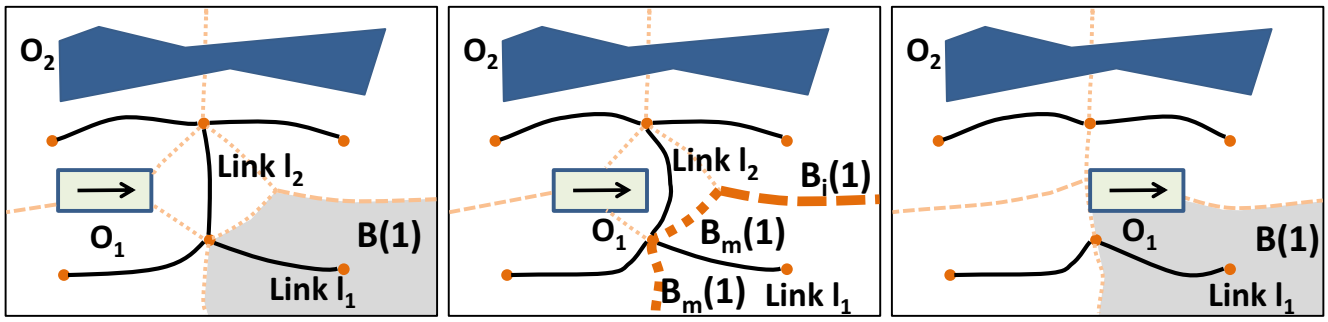

Figure 3: **Roadmap Link Bands***: Link bands are a partition of the freespace based on the links of AERO. (a) Several AERO links, in solid black lines, respond to a static obstacle, O_2 and a dynamic obstacle O_1. The link band, $B(1)$, for link l_1 is shaded, and the link boundaries are shown as dashed lines. (b) As O_1 approaches link l_2 it deforms. Link band $B(1)$'s boundary is highlighted in bold dashed lines and shows the two segments of the milestone boundary, $B_m(1)$, and the intermediate boundary $B_i(1)$. (c) Link l_2 is removed due O_1's motion while the link band $B(1)$ changes to reflect the removal.*

3.5 Roadmap Update

The previous section described the general representation for AERO and how it adjusts to moving obstacles. However, this type of deformation can not guarantee a valid roadmap or that a path can be found at all times, e.g. when a moving obstacle moves directly into a link. These cases require additional link removal or link addition steps to update the roadmap.

3.5.1 Link Removal

In order to maintain a valid roadmap, we remove the links based on both physically-based and geometric criteria. This combination works well since the roadmap computation is a combination of physically-based and geometric approaches.

The physically-based criteria attempts to prune links which have been considerably deformed. A natural measure of deformation for a link is its potential energy. The spring potential energy is a measure of the amount of deformation of a spring. For an adaptive link l_{ab} with n springs $\mathscr{S} = \{s_1, s_2, \ldots, s_n\}$, the average potential is given as

$$P_{ab} = \frac{1}{n} \sum_{s_i \in \mathscr{S}}^{n} \frac{k_s^i}{2} (\|s_i\| - L_i)^2$$

where k_s^i is the spring constant of spring i, $\|s_i\|$ is its current length, and L_i is its rest length. A link is removed when $P_{ab} > \varepsilon_s$, for a spring energy threshold ε_s.

The geometric criteria removes links based on proximity and intersection with the obstacles. Proximity is measured by the nearest distance from an adaptive link l_{ab} to the obstacles,

$$d_{ab} = min_{s_i \in \mathscr{S}, o_j \in \mathscr{O}}(d(s_i, o_j)).$$

Links are removed when this distance is less than the largest radius assigned to an agent, i.e. $d_{ab} < r_a$.

3.5.2 Link Addition

As the links are removed, AERO may no longer capture the connectivity of the free space and maybe unable to find paths. To remedy the situation, it is necessary to repair and add links to the graph. Since our initial roadmap is based on Voronoi diagram, it almost captures the connectivity of the freespace for *static* obstacles. Based on this assumption, we can bias the link addition step to repair links which have been previously removed. When a link is removed, it is placed in a list and re-inserted into AERO when the straight-line path between link's two milestones becomes collision-free. This has the added advantage that our roadmap will try to maintain the connectivity of the freespace of the static environment.

However, in a realistic scenario, it may not be the case that the static and dynamic obstacles are known ahead of time. In this case we need the ability to add milestones to help explore the freespace as the environment changes. In this case, we use random sampling techniques to generate new milestones and additional links [LaValle 2006].

We modify this approach by biasing our search toward areas *behind* obstacle motion. Given the velocity $\mathbf{v_j}$ of an obstacle o_j, we generate a sample outside of the obstacle's axis aligned bounding box in the direction $-\frac{\mathbf{v_j}}{\|\mathbf{v_j}\|}$ from the box's center. This sample is randomly perturbed to help remove uniformity among samples, which can lead to overly regular connectivity graphs. The milestones near to this sample are found and checked to see if an adaptive link can be added. Since a large number of new links can become computationally expensive, link addition is only performed when no path exists.

3.6 Numerical Stability

Since AERO relies upon numerical integration, stability is a concern due to the possibility of applying large spring or repulsive forces. However, when combined with the removal rules, this has not been an issue in practice. In general, using Verlet integration greatly helps in stability by treating the particle to be at rest during integration. Also, our link removal steps also help to ensure stability. Any link which is likely to become unstable will also likely be close to an obstacle or otherwise highly deformed. These types of links will be removed, helping the system remain in a stable state.

4 Navigation using AERO

In this section we describe our approach to compute collision-free paths for independent agents using AERO. In order to allow the agents to occupy the entire free space for navigation, we relax the restriction of constraining the agent's position to the links of the roadmap. Rather, we introduce *link bands* defined by each link of AERO, and use them for path planning as well as local dynamics simulation of each agent (See Figure 5).

4.1 Link Bands

The *link band* associated with a link of the roadmap is the region of free space that is closer to that link than to any other link on the roadmap. Formally, the link band of a link l_i is given by $B(i) = \mathscr{V}(l_i|\mathscr{L}) \cap D_f$. The width of the link band is the minimum clearance from the link to the obstacles in the environment, $B_w(i) = \min_{o_j \in \mathscr{O}}(d(l_i, o_j))$. The link bands form a partitioning of the free space based on proximity to the links. Each link band specifies a collision free zone in a well-defined neighborhood of each link of the roadmap. Additionally, link bands provide the nearest link, which is required for path search (Section 4.2), and distance to the roadmap that is used to compute guiding forces to advance the agent along the path (Section 4.3).

In a dynamic environment, an agent's path might require recomputation. We use link bands to detect such events. In particular we keep track of an agent's motion across a link band boundary. We classify points on a link band boundary into *milestone boundary* and *intermediate boundary* (see Fig. 3). A point on the milestone boundary belongs to two adjacent link bands whose links share a common milestone. On the other hand, the intermediate boundary is all points on the boundary that do not belong to the milestone boundary (see Fig. 3(b)). Formally, the milestone boundary of a link l_i, $B_m(i) = B(i) \cap B(j),\ \forall l_i \cap l_j \neq \emptyset$, and the intermediate boundary of a link l_i, $B_i(i) = \delta B(i) \setminus B_m(i)$. In the next section we show how the link bands are used for global path planning and to detect replanning events.

4.2 Path Planning

We use AERO for global path computation for each agent. Since an agent is not constrained to the roadmap, we initially compute the link band it belongs to. This link is set as the source link. Similarly the link band containing the goal position is computed and the corresponding link is set as the goal link. We assign a weight to each link as a combination of the link length, the reciprocal of the link band width, and the agent density on the link as:

$$w(l_i) = \begin{cases} \infty & \text{if } 2B_w(i) < r_a, \\ \alpha|l_i| + \beta\frac{1}{B_w(i)} + \gamma\frac{n}{|l_i|B_w(i)} & \text{otherwise,} \end{cases}$$

where α, β, γ are constants, $|l_i|$ is the length of link l_i, and n is the number of agents on link $B(i)$. The third term approximates the agent density on the link and causes the agents to plan using less crowded regions. The relative values of the constants are determined by the behavior characteristics of individual agents. A high relative value of α allows for choice of shortest path, a high value of β avoids narrow passages and a high value of γ demonstrates preference of less crowded passages. In our experiements, we used a high value of α for slow agents, whereas aggressive agents are assigned a higher value of γ. Given a weighted roadmap, an A* graph search is performed to compute the minimum weight path from the source to goal link band, which is stored by the agent. Once the agent reaches its goal link band, it proceeds to its goal position within the band.

As the simulation progresses, the nearest link to an agent may change. Based on link boundaries, we determine events that require a path recomputation. Crossing a milestone boundary indicates agent motion along the global path, and does not require a path recomputation. However, it is also possible for an agent to cross over the intermediate boundary. This typically occurs as a result of roadmap modification. In this case, it is possible that an alternative path to the goal exists. However, we allow the agent to move back to its previous path since this should be its path of least cost.

4.3 Local Dynamics Computation

Given a path on AERO, the motion of each agent is computed using a local dynamics model. In this section, we describe the local dynamics model used to guide an agents along the computed path. Our local dynamics model is based on the generalized force model of pedestrian dynamics proposed by Helbing et al [2003]. This force model has been shown to capture emergent crowd behavior of multiple agents at varying densities of crowds. We define the social force model in terms of force fields that are defined over each link band.

We modify the social force model, to include a force $\mathbf{F}^r$ that guides an agent along a link band on the roadmap. In addition, there is a repulsive force $\mathbf{F}^{soc}$ to the nearby agents, an attractive force $\mathbf{F}^{att}$ to simulate the joining behavior of groups, and a repulsive force from dynamic obstacles $\mathbf{F}^{obs}$. Let the agent p_i belong to link band $B(k)$, then the force field at a point $\mathbf{p}$ is given as

$$\mathbf{F}(\mathbf{p}) = \sum_j \left[\mathbf{F}_j^{soc}(\mathbf{p}) + \mathbf{F}_j^{att}(\mathbf{p})\right] + \mathbf{F}_k^r(\mathbf{p}) + \Sigma_o \mathbf{F}_o^{obs}(\mathbf{p}),$$
$$p_j \in \mathscr{A}, j \neq i, o \in \mathscr{O}$$

where,

$$\mathbf{F}_j^{soc}(\mathbf{p}) = A_i \exp^{(2r_a - \|\mathbf{p}-\mathbf{x}_j\|)/B_i}\mathbf{n}_j(\mathbf{p}) \left(\lambda_i + (1-\lambda_i)\frac{1+\cos(\phi_j(\mathbf{p}))}{2}\right),$$
$$\mathbf{F}_j^{att}(\mathbf{p}) = -C_j\mathbf{n}_j(\mathbf{p})$$
$$\mathbf{F}_o^{obs}(\mathbf{p}) = A_i \exp^{(r_a - d(\mathbf{p},o))/B_i}\mathbf{n}_o(\mathbf{p}) \left(\lambda_o + (1-\lambda_o)\frac{1+\cos(\phi_o(\mathbf{p}))}{2}\right)$$
$$\mathbf{F}_k^r(\mathbf{p}) = \frac{\mathbf{v}_k^d(\mathbf{p}) - \mathbf{v}_i}{\tau_i} + D_i d^4(\mathbf{p}, l_k)\mathbf{n}_{l_k}(\mathbf{p})$$

where A_i and B_i denote interaction strength and range of repulsive interactions and C_j strength of attractive interaction, which are culture-dependent and individual parameters. λ_i reflects anisotropic character of pedestrian interaction. The obstacle force field $\mathbf{F}^{obs}$ simulates the repulsion of the agents from other obstacles in the environment. Since the obstacles may be dynamic, we introduce an additional anisotropic term which biases the repulsive forces along the motion of the obstacles. This effect has also been modeled in other approaches by creating a 'discomfort zone' in front of dynamic obstacles [Treuille et al. 2006]. For efficient computation of repulsive force $\mathbf{F}^{soc}$ and obstacle force $\mathbf{F}^{obs}$, we compute forces to agents and obstacles within a radius B_i.

The roadmap force field $\mathbf{F}_k^r$ guides the agent along the link l_k. The link band $B(k)$ is used to define the region which is used to compute the force field for l_k. The first term in $\mathbf{F}_k^r$ makes the agent achieve a desired velocity along the link, whereas the second term attracts the agent within the link band. $\mathbf{n}_{l_k}(\mathbf{p})$ is the unit normal from point $\mathbf{p}$ to the closest point on l_k, $d(\mathbf{p}, l_k)$ is the distance from $\mathbf{p}$ to l_k. The desired velocity $\mathbf{v}_k^d(\mathbf{p}) = v_{max}\mathbf{e}_k(\mathbf{p})$, where $\mathbf{e}_k(\mathbf{p})$ is a unit vector field orthogonal to $\mathbf{n}_{l_k}(\mathbf{p})$. The direction of the normal is chosen such that $\mathbf{e}_k(\mathbf{p})$ points along the roadmap towards the next milestone on an agents path. D_i is a weighting term and the attractive force term keeps an agent inside the link band, reducing toggling across intermediate boundaries.

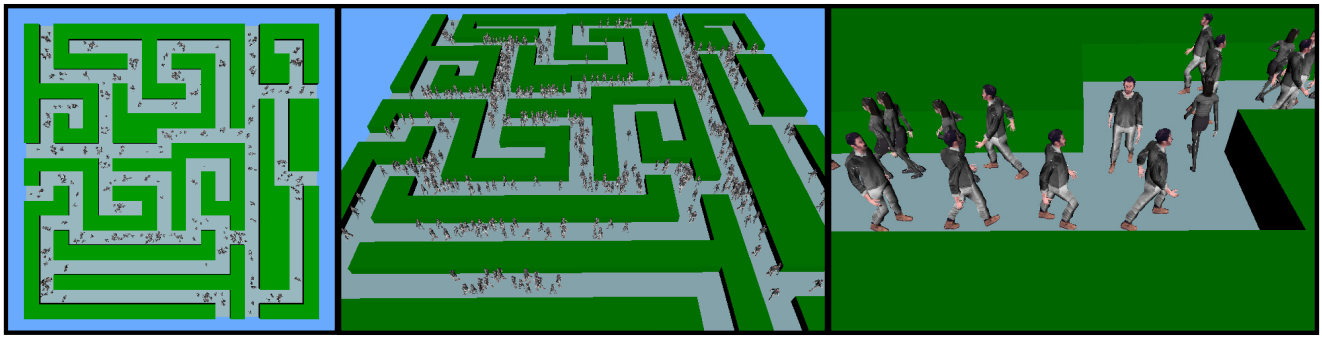

Figure 5: Left: Navigation of 500 *virtual agents in a maze consisting of 8 entrance and 8 exit points. Center: Each agent computes an independent path to the nearest exit using adaptive roadmaps. Right: Our local dynamics simulation framework based on link bands captures emergent behavior of real crowds, such as forming lanes. We perform real-time navigation of* 500 *agents at* 100*fps.*

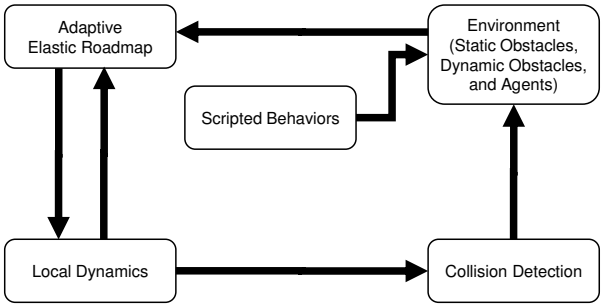

Figure 4: **Navigation System***: Given a description of the environment, an AERO is computed and updatehd. This is used in conjunction with our local dynamics model to simulate the motion of each agent.*

4.4 Behavior Modeling

Once the agent motion has been determined by local dynamics, this motion needs to be animated. Behavioral modeling allows us to translate from this motion to an animated character. To accomplish this task, we use a minimal set of predetermined behaviors; a stop, walk, and jog. A finite state machine is then used to transition and switch between them, as shown in Fig.4.

Transitions between states are determined by an agent's velocity and predefined thresholds. When the velocity is at or very close to zero, the agent moves to a stop state. Similarly, as the agent's speed increases, it transitions to a walk state and then to a jog state a higher speeds. To prevent oscillation between states, the threshold for increasing speeds is different than that of decreasing speeds. This is analogous to the idea that a slow jog can be the same speed as a fast walk.

Depending on the application, some agents are set to be more aggressive by specifying a higher maximum velocity. These agents will be more likely to be in the jogging motion in order to reach their goals.

5 Implementation and Results

In this section we describe the implementation of our multi-agent navigation system and highlight its performance on various environments. We have implemented our algorithm on a PC running Windows XP operating system, with an 3Ghz Pentium D CPU, 2GB memory and an NVIDIA 7900 GPU. We used Visual C++ 7 programming language and OpenGL as the graphics API for rendering the environments. The initial Adaptive Elastic Roadmap (AERO) for an environment is initialized by computing the Voronoi diagram of the static obstacles in the environment. This computation helps initialize the roadmap with links that are optimally clear of obstacles when the simulation begins. To simulate particle dynamics of the agents, we used a semi-implicit verlet integrator [Verlet 1967; Jakobsen 2001].

Proximity computations to dynamic obstacles are accelerated using a spatial hash data structure to identify the nearby objects. We maintain a spatial hash table of all dynamic obstacles, agents and links. Briefly, spatial hashing uses a hash function and table to compress and update a regular spatial decomposition. This step enables efficient lookups and proximity computation. In addition, to accelerate proximity computations to static obstacles, we precompute a discretized Voronoi diagram of the obstacles using the GPU [Sud et al. 2006]. The discrete Voronoi diagram provides proximity information to the nearest obstacle. To get the set of all obstacles within a given radius r, we scan the discrete Voronoi diagram (and distance field) within a window of size $r \times r$ and check if the distance value at the discrete samples is less than r. Thus the proximity computation is reduced to a small number of table lookups.

5.1 Benchmarks

We demonstrate our system on three complex scenarios.

- **Maze**: The maze scenario considers the case of multiple agents navigating a maze. The maze has 8 entry and 8 exit locations, and 1288 polygons. The initial roadmap consists of 113 milestones and 281 links. By using AERO, they have complete knowledge of how to navigate the maze despite its complexity and thus are able to quickly move toward their goals. See Fig. 5.

- **Tradeshow**: The tradeshow scenario is an indoor environment of an exhibit hall in a trade show. The exhibit consists of 511 booths and 110K polygons. The initial roadmap consists of 3996 links and 5996 milestones. Numerous agents walk around and visit multiple booths. The goals for each agent are updated as the agent arrives at a booth. Some agents stop when they reach their goal in order to simulate observation of a particular point of interest. After a certain amount of time, the agents will resume walking towards their next goal. Also, certain booths have fixed agents whose orientation changes according to passing agents. As agents move freely through the floor, they act as dynamic obstacles, and update the AERO. See Fig. 7.

- **City**: The city scenario is an outdoor scene consisting of multiple city blocks. The model consists of 924 buildings and 235K triangles. The initial roadmap for the environment con-

sists of 4K links. The environment also consists of 50 moving cars as dynamic obstacles. As the cars move through the urban setting, links on the path deform around the obstacles and get invalidated. We add a higher potential in front of the cars along their direction of motion, which decreases the probability of the agents from selecting paths in front of moving obstacles. The environment is populated with a non-uniform density of agents moving along the side walks or crossing the streets. Additional behavior characteristics of each agent are assigned at run-time. These individualized behavors includes updating the goals, varying the maximum speed, and changing interaction range of the agents. See Fig. 6.

Demo	Agents	Sim	Path Search	AERO Update	Total Time
Maze	500	9.1	0.005	0.58	9.64
Maze	1000	31.2	0.01	0.58	31.79
Trade Show	500	8.73	3	5.5	17.23
Trade Show	1000	32.95	7	5.5	45.45
City	500	9.75	7.4	15.1	32.25
City	1000	35	13.1	15.1	63.2

Table 1: **Performance on each scenario.** *Timings reported here are the average simulation time per frame (step) broken down into the time for simulating local dynamics (***Sim***), performing path search (***Path Search***), and updating AERO on the fly (***AERO Update***). All timings are in milliseconds.*

5.2 Results

We highlight the performance of our algorithm on the complex benchmarks. Our approach can perform real-time simulation of crowds with up to $1,000$ independent agents at interactive rates – ranging from 16 to 104 frames per second, depending on the scene complexity and the crowd density. Our current implementation is unoptimized and does not make use of all optimized computations on the GPU. The performance of our algorithm in the environments (with different complexity) and varying number of agents is highlighted in Table 1.

6 Analysis and Discussion

In this section we analysis the time complexity of various stages of our algorithm, and provide a qualitative comparison with prior work.

6.1 Analysis

Performance of our approach depends on a number of factors. At each time-step, AERO's complexity is $O(|M|+|E|)$, or linear in the number of particles and edges. The tasks per timestep include force computations, numerical integration, as well as path search and roadmap maintenance. Agent motion also depends linearly in the number of agents, but each agent also performs a path search, thus making the agent portion of computation complexity $O(|N|+|E||N|)$. But, in all of these cases, the performance scales linearly with the number of agents or the complexity of the roadmap. Therefore, this approach should be able to scale well to a large number of agents.

6.2 Comparison and Limitations

We compare some of the features of our approach with prior algorithm and highlight some of its limitations. Our adaptive roadmap based agent navigation algorithm is designed to perform real-time global navigation for a large number (e.g. hundreds or thousands) of independent or *heterogeneous* agents, each with different goals. We also take into account dynamic obstacles and the local dynamics among the agents. AERO continuously adapts to dynamic obstacles and is used to compute collision-free paths in dynamic environments. As compared to local or potential field methods, AERO can compute a global path for each agent. In addition, we use elastic bands and link bands to augment the local dynamics and resolve collisions among multiple agents. Due to lazy and incremental updates to the roadmap and efficient computation of guiding-path forces with link bands, this approach can scale well to hundreds or thousands of agents.

Comparisons: Our work is complementary to several existing works on crowd simulation and multi-agent planning. Continuum Crowds [Treuille et al. 2006] targets navigation for a small umber $(2-5)$ groups of human agents, where each group consists of many (upto thousands) agent with *identical* goals and behavior characteristics. Moreover, this approach uses local collision avoidance and its accuracy is governed by the underlying grid resolution. This approach has not been shown to extend well to a large number of groups or when there are challenging narrow passages in the free space, as shown in our maze and trade show benchmarks.

Graph based approaches[Pettre et al. 2005; Lamarche and Donikian 2004; Li and Gupta 2007] use proximity graphs to capture the connectivity of the navigable space and use it for agent coordination. However, the navigation graphs are precomputed and thus are mainly restricted to static environments. Multi-agent Navigation Graphs [Sud et al. 2007] efficiently compute dynamic navigation graphs for simple agents. However, this approach is limited to a few hundred agents and does not scale with the number of agents. It cannot guarantees coherent and smooth paths, as shown in our video. Corridor Maps [Garaerts and Overmars 2007] use similar proximity ideas as link bands to define navigable space, and can adapt to dynamic obstacles. However, corridor maps cannot easily handle dynamic topology of the roadmap and model emergent behaviors like agents following each other in lanes. Local agent-based and potential-field methods [Reynolds 1987; Shao and Terzopoulos 2005] perform well for a large number of agents and exhibit interesting crowd-like behaviors, but cannot provide same guarantees in path finding as global approaches.

Limitations: Our approach has some limitations. Our current implementation address collision-free navigation of a large number of 3-DoF agents, therefore our work does not produce realistic motion in situations where each human is modeled as a high DoF avatar. Although AERO uses a global roadmap at each given time step for path computation, the local dynamics formulation to update the links can potentially result in an agent getting stuck in a local minimum across space-time. In other words, our work may not be able to provide convergence guarantees or provide completeness on the existence of a collision-free path for each agent in all environments. Furthermore, we currently treat each agent as an individual agent and do not exploit all the behavior-related characteristics of real crowds such as grouping. Finally, the performance of proximity queries is sensitive to choice of hash function parameters and theoretical analysis can be of potential interest for challenging scenarios with many varying parameters.

7 Conclusion

We present a novel approach for real-time navigation of independent agents in complex and dynamic environments. We use adaptive roadmaps and present efficient algorithms to update them. These roadmaps are augmented with link bands to resolve collisions among multiple agents. The algorithm has been applied to complex

indoor and outdoor scenes with hundreds or thousands agents and dynamic obstacles. Our preliminary results are encouraging and the algorithm can compute collision-free paths for each agent towards its goal in real time.

There are many avenues for future work. First of all, we would like to develop multi-resolution techniques to handle a very large number of agents, e.g. 10-20K independent agents at interactive rates. Secondly, we would like to use better models for local dynamics and behavior modeling that can result in more realistic crowd-like behavior. Instead of its current simple model, we would like to use higher DoF articulated models for each agent to generate more realistic motion. However, this would increase the dimensionality of the configuration space and significantly increase the compleixty of the navigation algorithm. Finally, it may be useful to extend these results to generate truly heterogeneous crowd behavior [Bon 1895], using example based models to guide the simulation [Lerner et al. 2007].

Acknowledgments

This work was supported in part by Army Research Office, National Science Foundation, RDECOM, and Intel. We would like to acknowledge members of UNC GAMMA group for useful discussions and feedback. We are also grateful to the anonymous reviewers for their comments.

References

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)*.

BON, G. L. 1895. *The Crowd: A Study of the Popular Mind*. Reprint available from Dover Publications.

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*.

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

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

GAYLE, R., LIN, M., AND MANOCHA, D. 2005. Constraint based motion planning of deformable robots. *IEEE Conf. on Robotics and Automation*.

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*.

HELBING, D., BUZNA, L., JOHANSSON, A., AND WERNER, T. 2005. Self-organized pedestrian crowd dynamics: experiments, simulations and design solutions. *Transportation science*, 1–24.

HOOGENDOORN, S. P., LUDING, S., BOVY, P., SCHRECKLENBERG, M., AND WOLF, D. 2000. *Traffic and Granular Flow*. Springer.

JAKOBSEN, T. 2001. Advanced character physics. In *Game Developer's Conference*.

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

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

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 (Sept).

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

LERNER, A., CHRYSANTHOU, Y., AND LISCHINSKI, D. 2007. Crowds by example. *Computer Graphics Forum (Proceedings of Eurographics) 26*, 3.

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. *Theory and Practice of Computer Graphics (TPCG'03)*.

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

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*.

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*.

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. *Comput. Graph. 21*, 4, 25–34. Proc. SIGGRAPH '87.

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.

RODRIGUEZ, S., LIEN, J.-M., AND AMATO, N. M. 2006. Planning motion in completely deformable environments. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)* (May).

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.

SUD, A., GOVINDARAJU, N., GAYLE, R., AND MANOCHA, D. 2006. Interactive 3d distance field computation using linear factorization. In *Proc. ACM Symposium on Interactive 3D Graphics and Games*, 117–124.

SUD, A., ANDERSEN, E., CURTIS, S., LIN, M., AND MANOCHA, D. 2007. Real-time path planning for virtual agents in dynamic environments. *Proc. of IEEE VR*.

SUGIYAMA, Y., NAKAYAMA, A., AND HASEBE, K. 2001. 2-dimensional optimal velocity models for granular flows. In *Pedestrian and Evacuation Dynamics*, 155–160.

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

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.

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

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

VERLET, L 1967. Computer experiments on classical fluids. I. Thermodynamical properties of Lennard-Jones molecules. *Phys. Rev.*, 159, 98-103.

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*.

Figure 6: Crowd simulation in an urban landscape: A street intersection in a virtual city with 924 *buildings,* 50 *moving cars as dynamic obstacles and* 1,000 *pedestrians. We show a sequence of four snapshots of a car driving through the intersection. As the car approaches a lane of pedestrians (top), the lane breaks (middle two images) and the pedestrians re-route using alternate links on the adaptive roadmap. Once the car leaves the intersection (bottom) the pedestrians reform the lane using the adaptive roadmap. We are able to perform navigation of* 1,000 *pedestrians in this extremely complex environment at* 16*fps on a* 3*Ghz PC.*

Figure 7: Sequence of 4 snapshots from Tradeshow demo. The environment contains 511 booths with 110K polygons. The agents move toward different booths and avoid each other using link bands.