Real-time Simulation of Heterogeneous Crowds

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

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

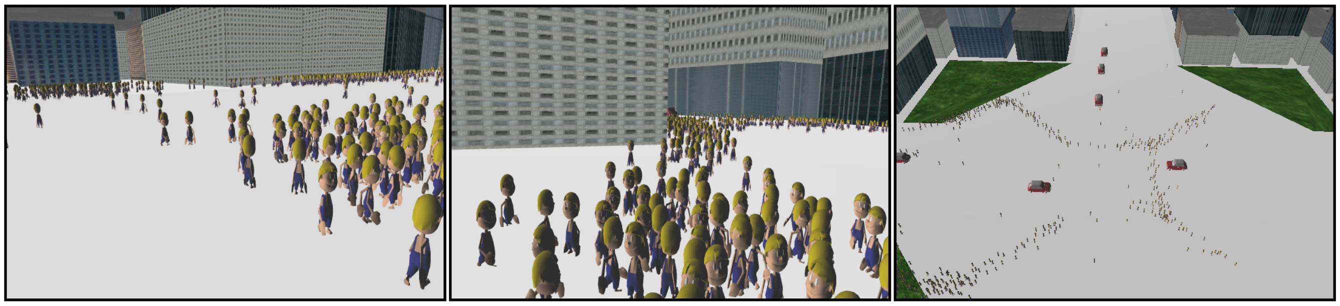

Figure 1: **Crowd Simulation in a City Model:** *Different views from a crowd simulation in a city model with* 924 *buildings and* 200 *moving vehicles (left,middle). We can simulate the motion of* $2,000$ *human agents at interactive rates. Crowds react as dynamics obstacles approach their path (right).*

Abstract

We present a real-time algorithm for simulating heterogeneous crowds with potentially distinct individual behavior characteristics and goals. Our approach combines global motion planning with a generalized pedestrian dynamics model to synthesize motion for numerous groups of people in complex dynamic environments. Inspired by self-organization phenomena in crowds, we introduce "Pedestrian Levels of Detail" to accelerate large-scale simulation of individual agents, while preserving natural collective behaviors observed in real crowds. We highlight the performance of our algorithm on heterogeneous crowds in urban environments.

Keywords: crowd simulation, multi-agent path planning, pedestrian dynamics, interactive system

1 Introduction

Modeling of crowds, multiple agents and swarm-like behaviors has been widely studied in computer graphics, robotics, architecture, physics, psychology, social sciences, and civil and traffic engineering. Realistic visual simulation of human crowds requires modeling of pedestrian dynamics, group behaviors, motion synthesis, and rendering. In this paper, we address the problem of real-time motion synthesis of large-scale "heterogenous crowds" in complex dynamic environments. In his pioneering work, Gustave Le Bon [1895] defined *heterogenous crowd* as consisting of many dissimilar types of groups, each with potentially independent behavior characteristics and goals. Examples include large exposition halls, wide festival arenas, busy urban street scenes, etc. Real-time simulation of heterogenous crowds is highly challenging because pedestrian dynamics exhibits a rich variety of collective effects, such as lane formations, oscillations at bottlenecks, chemotaxis and panic effects [Schreckkenberg and Sharma 2001].

Microscopic models using agent-based methods [Schreckkenberg and Sharma 2001] can capture pedestrian dynamics under varying density and situations for heterogenous crowds. However, these microscopic computations can become a bottleneck in achieving real-time performance and may not be able to capture all aspects of crowd movement. One of the major computational challenges for large-scale crowd simulation is global route planning for each individual agent. The route planning problem can become intractable, as each individual character moving independently is essentially a dynamic obstacle to other agents. Therefore, many existing approaches for crowd simulation only model groups of agents to minimize global navigation computations. However, these approaches are typically restricted to static environments or a small, fixed number of groups. In addition, most of prior algorithms either avoid collision checks to maintain real-time performance or they perform local computations that can result in unnatural collective behavior and fail to sustain interactive performance.

Main Results: In this paper, we present a new algorithm for real-time simulation of large-scale heterogenous crowds in complex dynamic environments. Our approach is based on two novel concepts:

- **'Adaptive Elastic ROadmaps" (AERO)** - It is a connectivity graph structure that is lazily computed using a generalized crowd dynamics model that simultaneously captures macroscopic mutual interaction among multiple moving *groups* and microscopic local forces among *individual* pedestrians or agents. We use AERO to perform dynamic, global path planning based on this force model.
- **"Pedestrian Levels of Detail" (PLODs)** – a new hierarchical data structure selectively computed on the fly and is used to accelerate large-scale simulation of heterogenous crowds. Our formulation is based on the observation and empirical validation in traffic engineering that crowds exhibit self-organization in pedestrian flows [Schreckkenberg and Sharma 2001]. Depending on their dynamic states (e.g. walking speed, heading directions), spatial proximity, and behavior characteristics, heterogeneous crowds are adaptively clustered and subdivided into PLODs. PLODs also help to minimize the global path computations based on dynamically changing factors and accelerate the overall runtime performance without affecting the visual appearance of simulated crowd movement.

We integrate AERO and PLODs to compute a dynamic roadmap for global path planning, in order to correctly predict crowd movement and various emergent phenomena, including dynamic lane formation, crossing flows, bi-directional traffic, and group gathering. We demonstrate our approach on complex indoor and outdoor scenarios, including a city scene consisting of $5,000$ pedestrians with 200

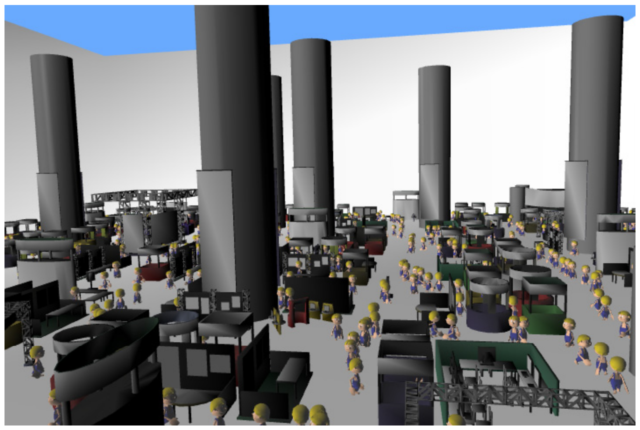

Figure 2: **Simulation in an indoor environment** *Demonstration of our crowd simulator on a tradeshow floor with* 512 *booths, 170K polygons and 1,000 human agents.*

moving cars and an exhibition hall with 511 stationary booths and 5, 000 individual agents on foot and avoiding each other. Our initial proof-of-concept implementation is able to perform crowd simulation for these scenes in less than 1 second per frame on a PC with an 3Ghz Pentium D CPU and 2GB memory.

Organization: The rest of the paper is organized as follows. Section 2 presents related work on crowd simulation and path planning. We give an overview of our approach in Section 3. We introduce "pedestrian levels of detail" to accelerate large-scale crowd simulation in Section 4. Section 4 presents "adaptive elastic roadmap", coupled with a generalized dynamics force model and modeling of agent behavior. We present the results and discuss implementation issues in Section 6. We analyze the performance and highlight a few limitations of our approach.

2 Related Work

In this section, we give a brief overview of prior work related to crowd dynamics, agent planning and hierarchical methods to accelerate simulations.

2.1 Crowd Dynamics

Many different approaches have been proposed for modeling crowd movement and simulation [Schreckkenberg and Sharma 2001; Thalmann et al. 2006]. They can be classified based on specificity, problem decomposition (discrete vs continuous), stochastic vs deterinistic, etc.

2.1.1 Discrete methods

Discrete methods rely on discretization 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.

Cellular Automata methods: These methods model the pedestrian movement 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 1994].

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. Under this model each agents attempts to move at its optimal velocity under constraints from other particles.

2.1.2 Continuous Methods

The flow of crowds has been compared with 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 [Treuille et al. 2006; Helbing et al. 2005]. Most recently, a novel approach for crowd simulation based on continuum dynamics has been proposed by Treuille et al. [2006]. They compute a dynamic potential field that simultaneously integrates global navigation with local obstacle avoidance. The resulting system runs at interactive rates and demonstrates smooth traffic flows for three to four groups of large crowds that are moving with common goals.

2.2 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]. However, these algorithms may not scale to large scale simulation in real time. On the other hand, local methods are mostly reactive style planners based on 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]. 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 include Elastic Bands [Quinlan and Khatib 1993] and Elastic Roadmaps [Yang and Brock 2006]. We extend these algorithms to simultaneously perform route planning computations for multiple pedestrians in dynamic scenes.

2.3 Hierarchical Methods for Simulation

Hierarchical methods have been used to accelerate various aspects of crowd simulation and rendering. These include different levels of autonomy and grouping [MUSSE and Thalmann 1997], and algorithms for accelerating crowd rendering [Dobbyn et al. 2005]. Hierarchical techniques have also been used to generate more realistic animations [Ashida et al. 2001; Sung et al. 2004]. Finally, multiresolution techniques have also been used to accelerate dynamics simulation, multiagent control and particle system simulation [Brogan and Hodgins 2002; Carlson and Hodgins 1997; Multon et al. 1999; O'Brien et al. 2001; Popovic and Witkin 1999; O'Sullivan and et al. 2002].

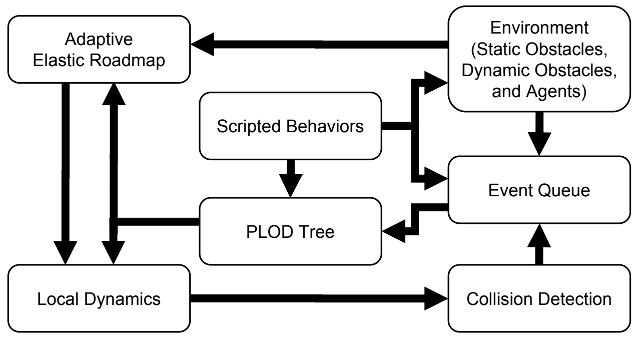

Figure 3: **System Architecture**: *This figure shows the different components of our system.*

3 Overview

In this section, we first describe some salient features of human crowds movement, introduce our notations and finally give an overview of our approach.

Background: Our work builds upon the vast literature on pedestrian dynamics in psychology, transportation science, civil and traffic engineering. One of the key underlying behavorial characterisics in pedestran dynamics is the *principle of least effort* [Zipf 1949]. This implies that among all available options (e.g. accelerating, decelerating, changing direction or doing nothing), a pedestrian tries to choose the option that will yield the smallest predicted *disutility*. An individual will try to adapt to its environment or will try to change the environment to suit its needs, if easier. Under this assumption the pedestrian flow self-evolves into a user-equilibrium state with the emergence of several interesting collective effects at various scales and densities of crowds [Helbing et al. 2005]. Examples of such emergent phenonmena include dynamics lane formation, oscillations at bottlenecks, banding patterns at intersections, trail following and panic effects. In addition, pedestrian dynamics has been compared to gas dynamics at low densities, fluid dynamics at moderate densities and granular flow at high densities [Helbing et al. 2005]. A heterogenous crowd may exhibit more than one such collective behavior. We use these results in formulating our PLOD-based approach so that it can reliably capture the collective behaviors of pedestrians in heterogeneous crowds.

3.1 Definitions and Notation

Crowds are large groups that occupy a common location and share a common focus. Crowds may be classified based on the level of chaos into diferent types of gatherings and mobs [Forsyth 2006]. We assume the crowd is contained within a domain D. We restrict our focus to human crowds, and the domain may be decomposed into a set of 2-manifolds. The smallest individual entity in a crowd is called an *agent*, denoted p_i. The crowd is the set of all agents $\{p_1, p_2, \ldots, p_k\}$, and we treat each human as an agent. Moreover, 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 when implicitly defined. In addition to the dynamics state, the dynamics of an agent is also influenced by environmental and behavior characteristics such as goal position, aggressiveness, etc. The combined dynamics and behavior characteristics define the *pedestrian state*, denoted $\mathbf{q}_i$. 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 (see Section 5.2).

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. Each agent is also associated with certain behavorial characteristics, which influence its dynamics (see Section 5.4).

3.2 Heterogenous Crowd Simulation

Problem definition: 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 $\mathbf{x}_i(t_f) = \mathbf{g_i}$. Our goal is to compute a collision-free path for each agent and ensure that its behavior conforms with prior results in pedestrian dynamics.

Our approach for real-time simulation of heterogenous crowds uses a global planner along with a local dynamics model for updating the state of each agent. The global planning and local dynamics are efficiently computed using an adaptively simplified approximation of pedestrian dynamics for route planning - called *pedestrian levels of detail* (PLOD).

A PLOD is a cluster of agents with similar a pedestrian state - i.e. they have a similar dynamics state (position, velocity) and behavior (e.g. intermediate goals, path selection). We construct a pedestrian state-based subdivision hierarchy of the entire crowd, called the *PLOD-tree*. The PLOD-tree facilitates decoupling between global planning and local dynamics, and scales well for heterogenous crowds. Our local dynamics model computes the dynamics of each agent within a PLOD, and the influence of agents outside the PLOD is approximated by the PLOD-tree. In general constructing a PLOD-tree is an $O(k \log k)$ operation, where k is the size of the input. However, we use a priority queue and lazy updates to amortize this cost over several simulation time-steps.

Our local dynamics model is based on the generalized force model of Helbing et al. [2005]. This force model has been shown to capture various collective phenomenon in crowd simulation and has been demonstrated across varying densities of crowds: from low density gatherings to densely packed panic mobs. We assign each PLOD a maximum velocity based on its behavior characteristics. We compute the desired velocity of each PLOD based on a force term computed using the 2D optimum velocity model [Sugiyama et al. 2001].

Our algorithm computes the path of each agent and PLODs using a global roadmap-based representation. In terms of path planning, a PLOD is treated as a single entity or an agent, and thereby reduces the complexity of route planning in terms of total number of entities. We extend the Elastic Roadmap algorithm [Yang and Brock 2006] and compute an AERO (Adpative Elastic ROadmap) for all moving agents and PLODs in a dynamic scene. The AERO is represented as a graph, where the vertices correspond to milestones and the links represent collision-free paths between the milestones. At each time-step, we deform the links based on the position of other agents and PLODs, while taking into account their local dynamics.

The overall simulation algorithm consists of the following steps: (see Fig. 3):

1. **Update the environment:** This includes updating the state of (non-simulated) dynamic obstacles, the goals and behavior parameters of each agent. Details are presented in Section 5.4.

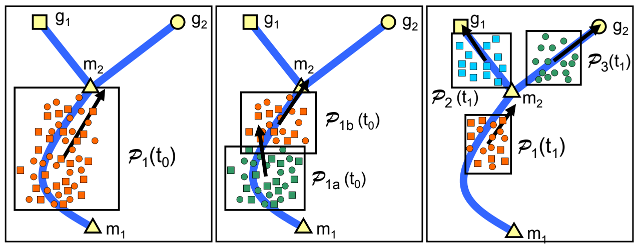

Figure 4: **PLODs**: $\mathbf{g_1}$ *is the goal of the square agents, and* $\mathbf{g_2}$ *is the goal of the circular agents. (left) A coarse resolution PLOD* $\mathcal{P}_1$ *at time* t_0 *of agents with similar pedestrian state. (middle) At finer resolutions, agents may be grouped into multiple PLODs -* $\mathcal{P}_{1a}$ *and* $\mathcal{P}_{1b}$. *(right) At time* t_1, *as agents pass the intermediate milestone* m_2, *they split into separate PLODs* $\mathcal{P}_2$ *and* $\mathcal{P}_3$.

2. **Update PLOD-tree:** An event queue is constructed based on the estimated time to the next critical event when a reclustering is needed. The PLOD-tree is then updated accordingly. This step is described in Section 4.

3. **Compute inter-cluster force fields:** This corresponds to forces exerted by a PLOD on the environment, i.e. the elastic roadmap and other PLODs. We also compute the force field from other dynamic obstacles.

4. **Update AERO:** The roadmap is adaptively recomputed based on the force fields and position of the obstacles. Details are given in Section 5.1.

5. **Path computation:** A cost function is assigned to each link in AERO, and a minimum weight path is computed for each agent or PLOD. This step also updates the intermediate goal for each agent or PLOD. Details are given in Section 5.1.

6. **Apply local dynamics.** This updates the dynamics state for each agent in the cluster using a physically-based microscopic pedestrian dynamics model. At end of this step, the PLOD state of each agent is recomputed to update the PLOD-tree. This invloves the following steps for each cluster (section 5.2):

   - Compute the roadmap force field. This is an attractive force field that guides the agents along the path.
   - Compute scripted and intra-cluster forces, and advance one simulation step.
   - Perform collision detection and contact resolution.
   - Update the priority queue for PLOD hierarchy update.

4 Pedestrian Levels of Detail

Inspired by the work on cluster analysis of dynamic pedestrian grouping [Helbing et al. 2005; Schreckkenberg and Sharma 2001], we introduce PLODs for crowd simulation to capture self-organization of pedestrian flows. Specifically, a PLOD is formed based on the pedestrian state of the agents. The pedestrian state is a 6-DOF vector that consists of the dynamics state, the next intermediate goal and link the agent is traversing, $\mathbf{q}_i = \{\mathbf{x}_i, \mathbf{v}_i, g'_i, l_i\}$. A PLOD $\mathcal{P}_m$ is computed based on the following rules: Two agents p_i, p_j with state vectors $\mathbf{q}_i, \mathbf{q}_j$ belong to the same PLOD $\mathcal{P}_m$ if

1. They share same intermediate goals and links, i.e $g'_i = g'_j$ and $l_i = l_j$.

2. They have similar velocity, i.e. $\|\mathbf{v}_i - \mathbf{v}_j\| \le \delta_1$.

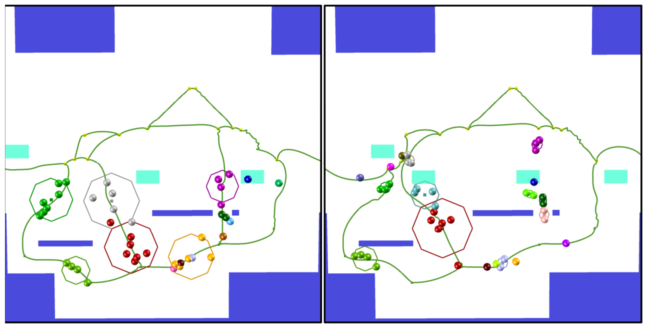

Figure 5: **PLODs coupled with AERO**: *Illustration on a crowd simulation with* 50 *agents. The static obstacles are shown in blue, dynamic obstacles in cyan. The AERO is shown in green curves (links), and they deform as the dynamic obstacles approach in order to maintain valid paths. Each PLOD is circled and shown in a different color. The PLOD represents a cluster of agents with similar physical state and intermediate goal on the roadmap. (a) Two PLODs (in grey and red) moving along the center link. (b) As the grey PLOD reaches the intermediate goals, it splits into 2 PLODs (grey and light blue).*

3. The distance between their positions is sufficiently small, i.e. $\|\mathbf{x}_i - \mathbf{x}_j\| \le \delta_2$.

The first criteria ensures that all the agents in a PLOD would naturally move in the same direction. The second and third criteria ensure that the PLODs are small and similar enough such that the global behavior of each agent can be reliably approximated by a single entity.

4.1 Clustering Data Structure

We create PLODs by grouping the agents in state space using the clustering criteria described above. It is important that we use a clustering method that is quick and scales well to a large number of agents. To fulfill these criteria, we cluster using a 3-dimensional KD-tree, where all the agents lie at the leaf nodes of the tree. Agents that share a leaf are considered together as part of the same PLOD.

The 3-dimensions of the KD-tree represent the state space of position and direction of velocity. This subset of pedestrian space is chosen because it is the minimum subset that allows KD-trees to consistently group nearby agents with similar goals in one cluster. The state space dimensionality is kept small so that we can compute and update the KD-tree quickly.

The splitting plane is chosen using the well known sliding midpoint rule, where we find the longest dimension of the current cell and split at a plane that bisects the cell along that dimension. This splitting rule is fast to compute, and works well with clustered data without leaving any empty cells.

The splitting rule is recursively re-applied to each cell until the above mentioned conditions are met for every leaf node. The result is a fast construction of several PLODs, each containing a group of agents with similar dynamics states and shared goals. In Section 5 we show how PLODs are used for crowd simulation.

4.2 Hierarchy Updates

We perform lazy bottom-up updates of the hierarchy, the PLOD-tree. Instead of updating the PLOD-tree at each time step, we update the hierarchy when an event occurs. We maintain a priority queue of events. Our algorithm relies on the fact that all agents within a PLOD have similar goals and should arrive at an intermediate goal at approximately the same time. As a consequence of the principle of least effort [Zipf 1949], the agents tend to move with the same velocity unless an external change or force is applied. A PLOD-tree update event is performed when a PLOD arrives at an intermediate goal, or its behavior characteristics or the immediate environment change. Based on PLOD velocities, we compute the time to the next update event for each PLOD and maintain them in the priority queue.

The priority queue for updating events is based on finding the minimum time until a reclustering of the KD-tree will be needed. A lower bound on when this situation will happen is computed based on the position and velocity of each PLOD. There are two events for which an update of a PLOD-tree is needed: (1) when a PLOD reaches its current goal, and (2) when two or more PLODs intersect each other. For each PLOD the earliest of these two potential events is stored in a priority queue along with the ID of the PLODs involved in that event. The crowd simulation can then proceed without reclustering until the time stored at the top of the priority queue is reached. At that instance the PLODs in the neighborhood of the ones that caused the update are reclustered locally, affecting only a local region of the KD-tree.

5 Crowd Simulation using PLODs

In this section, we use PLODs for path planning and local dynamics computations on large-scale crowds. First, we present adaptive elastic roadmaps (AERO) for crowd simulation. Next, we describe the force model for crowd dynamics and how it is integrated with PLODs. The dynamics of the agents is computed using intra-PLOD forces, as well as external forces. The intra-PLOD forces, denoted $\mathbf{f}^{int}$, influence the microscopic local dynamics of the agents, whereas the external forces, denoted $\mathbf{f}^{ext}$, influence the links in AERO.

5.1 Adaptive Elastic ROadmap (AERO)

In this section we describe our algorithm to compute AERO and update its links during each time-step. AERO is a roadmap, or a connectivity graph of milestones and links, $\mathcal{R} = \{\mathcal{M}, \mathcal{L}\}$, and is used to compute the collision-free path for each agent or PLOD. Each milestone is a position $\mathbf{x}_i \in D$, and each link l_{ab} connects two milestones $\mathbf{x}_a, \mathbf{x}_b$ along a path. Each entity queries the roadmap to compute a path between two configurations in D by using a graph search algorithm (such as A*) . For agent p_i let the first link on path f_i be l_{ab}. Then, the next milestone, $\mathbf{x}_b$, is the intermediate goal g_i' for agent p_i.

The AERO is computed and updated using a physically-based particle simulation. Specifically, particle $\mathbf{m}_i$ is a point-mass in D which responds to applied forces. The state of the particle at time t is described by its position and velocity. The AERO is represented by dynamic milestones, each represented as a particle. The adaptive links are represented using a sequence of particles connected by linear springs. As the obstacles (which may include other agents and PLODs) move, the repulsive forces cause the milestones to move, and the links to deform towards the open areas of the domain (as shown in Fig. 6).

Force field computation: There are two main forces applied to each particle: roadmap internal forces and repulsive external forces. The internal forces shrink or minimize the length of the links, and are simulated using standard damped Hookean spring. The external force is based on a repulsive potential field due to the obstacles or other entities (including PLODs). For each entity E_j, we apply a force on particle q_i if it is sufficiently close to E_j. This force given is:

$$\mathbf{f}_i^{ext} = \frac{b}{d(q_i, E_j)^2}$$

where $d(q_i, E_j)$ is the minimum distance between particles q_i and entity E_j, and b is a repulsive scaling constant.

Given the applied forces for each particle, we update the AERO using a quasi-static forward Euler integration step. This variation considers the particles to be at rest during integration and prevents undesired oscillation in the links.

We use generalized Voronoi diagram computation to compute the initial position of the milestones in the AERO. The Voronoi diagram provides optimal clearance from the obstacles and can be computed using graphics hardware. The Voronoi edges become adaptive links, and intersections of these edges are the dynamic milestones.

Roadmap modification: At times roadmap deformation by itself cannot always ensure that a path is valid. To maintain the roadmap, we remove the links when they collide with other entities. We perform this computation using spring potential energy and proximity to the other entities. The spring potential energy is a measure of the amount of deformation of a spring. For an adaptive link l_i with n springs $\mathcal{S} = \{s_1, s_2, \ldots, s_n\}$, the average potential is

$$PE_i = \frac{1}{n} \sum_{s_i \in \mathcal{S}}^{n} \frac{k_s^i}{2} (\|s_i\| - L_i)^2,$$

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 $PE_i > \epsilon_s$, for a spring energy threshold ϵ_s. Proximity is measured by the nearest distance from an adaptive link l_i to the entities. Links are removed when this distance is less than the largest radius assigned to an agent.

We add links to the AERO when the entities are unable to find a collision-free path using graph search. Our first approach involves repairing (or recomputing) the links that were removed in the previous time-steps. When the straight-line path between previously connected milestones lies in the collision-free space, an adaptive link is added along this path.

The main advantage of AERO for crowd simulation is that multiple agents or PLODs can use the same roadmap. As a result, we only need to modify AERO to respond to the motion of the other agents or PLODs. Each agent applies a repulsive force to the particles. To ensure that agents do not alter their own path, particles on the agent's path are not affected.

Using a single roadmap like AERO for all the agents and PLODs can result in coordination problems. This can happen when when multiple agents try to use or occupy the same link at the same time while traveling in opposing directions (see Figure 6). To handle this situation, an additional lane is created between the two milestones by adding another adaptive link. Agents or PLODs in lane i then each apply repulsive forces to the other lanes, causing them to move away from lane i.

5.2 Intra-PLOD forces

Our local dynamics model is based on the generalized force model of pedestrian dynamics by Helbing et al [2003]. This force model has been shown to capture emergent crowd behavior at varying densities of crowds and is thus a suitable choice for hetergoenous crowds. This force model is used to compute the dynamics of

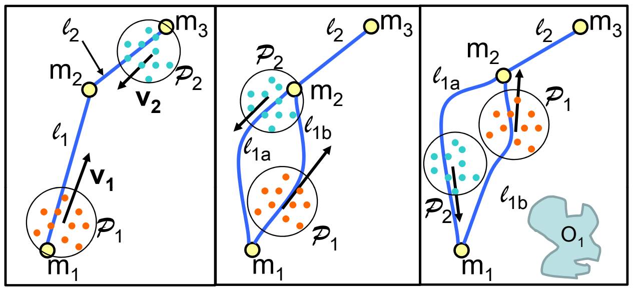

Figure 6: **AERO**: *This figure highlights the milestones and links used in AERO. (Left) $\mathcal{P}_1$ moves towards milestone m_2 and $\mathcal{P}_2$ moves towards the same milestone. (Middle) When two PLODs want to use the same link (l_1) in opposite directions, the algorithm performs lane-splitting to generate two new lanes or links: l_{1a} and l_{1b}. The link l_{1b} physically deforms in presence of obstacle O_1 (rightmost figure).*

agents within a PLOD. For efficient computation on parallel hardware (i.e. GPUs), we define the social force model in terms of force fields that are defined over the spatial extent of a PLOD. We modify the social force model, to include a force $\mathbf{f}^r$ that attracts an agent towards the elastic roadmap. In addition, there is a repulsive force $\mathbf{f}^{soc}$ to nearby agents, an attractive force $\mathbf{f}^{att}$ to simulate joining behavior of groups, and a repulsive force from dynamic obstacles $\mathbf{f}^{obs}$. Let the current link of PLOD $\mathcal{P}_m$ be l, then internal force field at a point $\mathbf{p}$ is given as

$$\mathbf{f}_m^{int}(\mathbf{p}) = \sum_j \left[\mathbf{f}_j^{soc}(\mathbf{p}) + \mathbf{f}_j^{att}(\mathbf{p})\right] + \mathbf{f}_l^r(\mathbf{p}), \quad p_j \in \mathcal{P}_m \tag{1}$$

where,

$$\begin{aligned}
\mathbf{f}_l^r(\mathbf{p}) =& \frac{\mathbf{v}_l^d(\mathbf{p}) - \mathbf{v}_i}{\tau_i}, \\
\mathbf{f}_j^{soc}(\mathbf{p}) =& A_i \exp^{(2r_a - \|\mathbf{p}-\mathbf{x}_j\|)/B_i} \mathbf{n}_j(\mathbf{p}), \times \\
& \left(\lambda_i + (1-\lambda_i)\frac{1+\cos(\phi_j(\mathbf{p}))}{2}\right), \\
\mathbf{f}_j^{att}(\mathbf{p}) =& -C_j \mathbf{n}_j(\mathbf{p}),
\end{aligned}$$

where A_i and B_i denote interaction strength and range of repulsive interactions, which are culture-dependent and individual parameters. λ_i reflects anisotropic character of pedestrian interaction. The roadmap force field $\mathbf{f}_l^r$ guides the agent towards its intermediate goal along the link l. The desired velocity $\mathbf{v}_l^d(\mathbf{p}) = v_{max}\mathbf{e}_l(\mathbf{p})$, where $\mathbf{e}_l(\mathbf{p})$ is a unit vector field pointing in direction of the intermediate goal. For points not lying on the link l, we add the displacement of from the link. Thus $\mathbf{e}_l(\mathbf{p}) = \frac{(g_i' - \mathbf{p}) + \mathbf{n}_l(\mathbf{p})}{\|(g_i' - \mathbf{p}) + \mathbf{n}_l(\mathbf{p})\|}$, where $\mathbf{n}_l(\mathbf{p})$ is the perpendicular vector from $\mathbf{p}$ to link l. The term τ_i is a 'relaxation time' for the agent to achieve the desired velocity, while moving under constant acceleration. This time is influenced by the aggressive behavior. This force field is defined at points corresponding to the position of each agent p_i in $\mathcal{P}_m$. The roadmap force $\mathbf{f}_{il}^r$ is similar to the acceleration term used in the social force model.

The social force field $\mathbf{f}_j^{soc}(\mathbf{p})$ represents the repulsive force from agent p_j at a point $\mathbf{p}$. The velocity bias field $\cos(\phi_j(\mathbf{p}))$ is used to model the anisotropic nature of repulsive forces (the repulsion is higher from agents in front). However, in contrast to the standard social repulsive force presented in [Helbing et al. 2003], we use a modified term to compute the anisotropic nature of the forces. At position of agent p_i, $\mathbf{p} = \mathbf{x}_i$, the velocity bias $\cos(\phi_{ji}) = \cos(\phi_j(\mathbf{x}_i))$, the repulsive force is $\mathbf{f}_{ij}^{soc} = \mathbf{f}_j^{soc}(\mathbf{x}_i)$. The force $\mathbf{f}_{ij}^{soc}$ is biased along the velocity $\mathbf{v}_j$ of agent p_j, instead of the velocity $\mathbf{v}_i$ of agent p_i as described in [Helbing et al. 2003]. This is done for efficiently computing the force fields in parallel using graphics hardware in the near future. However, this simplification does not significantly change the internal social forces. This is due to the fact that, by definition, all agents within a PLOD have similar velocities, thus $\mathbf{v}_i \approx \mathbf{v}_j$ and $\cos(\phi_{ji}) \approx -\cos(\phi_{ij})$.

The attractive force field $\mathbf{f}_j^{att}(\mathbf{p})$ simulates the nature of agents belonging to same behavorial groups (e.g. families, tourist groups) to stay in close proximity and attract each other. This force is applied between two agents p_i and p_j, only if they are apriori defined to be within the same group.

5.3 External Forces

In the previous section, we described the internal forces acting on an agent within a PLOD. We now describe the external forces $\mathbf{f}^{ext}$ interacting on a PLOD. This includes the forces exerted by other PLODs and dynamic obstacles. These forces are computed as a sum of various force fields over the spatial extent of the PLOD. Finally, the PLOD itself exerts forces on the environment, notably the AERO. The external force fields acting on a PLOD $\mathcal{P}_m$ are computed as follows. Let $\mathcal{N}$ denote the set of PLODs in close proximity to $\mathcal{P}_m$, and $\mathcal{O}$ denote the set of dynamic obstacles in close proximity to $\mathcal{P}_m$. Then the external force field on a PLOD $\mathcal{P}_m$ is given by

$$\mathbf{f}_m^{ext}(\mathbf{p}) = \Sigma_{n\in\mathcal{N}} \mathbf{f}_n^{ext}(\mathbf{p}) + \Sigma_{o\in\mathcal{O}} \mathbf{f}_o^{obs}(\mathbf{p}) \tag{2}$$

where

$$\begin{aligned}
\mathbf{f}_n^{ext}(\mathbf{p}) =& \Sigma_j \mathbf{f}_j^{soc}(\mathbf{p}), \ \forall\, p_j \in \mathcal{P}_n \\
\mathbf{f}_o^{obs}(\mathbf{p}) =& A_i \exp^{(r_a - d(\mathbf{p},o))/B_i} \mathbf{n}_o(\mathbf{p}) \times \\
& \left(\lambda_o + (1-\lambda_o)\frac{1+\cos(\phi_b(\mathbf{p}))}{2}\right)
\end{aligned}$$

The obstacle force field $\mathbf{f}^{obs}$ simulates the repulsion of the agents in a PLOD 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]. The set of neighboring PLODs $\mathcal{N}$ may also be treated as dynamic obstacles for the current PLOD $\mathcal{P}_m$. Hence the repulsive force from remaining PLODs is similarly computed. In particular, this repulsive force field is equivalent to the sum of the internal social force fields of neighboring PLODs. The total repulsive force on each agent computed using this approach would be consistent with treating all agents as a single PLOD and using the general social force model for computing dynamics of all agents.

5.4 Behavior Modeling

Our model allows for the addition of specific behaviors in order to simulate real human motion as accurately as possible. Behaviors can be implemented on a crowd, group or individual level. Crowd behaviors reflect the presence of certain features within the environment that might affect the motion of the agents. These features are modeled as points of interest which cause the agents walking in that area to exhibit some behavior, such as slowing down or congregating. The regions of interest are modeled by assigning them as milestones and goals.

# of Agents	Local Dynamics (ms)	Path Search (ms)	AERO Update (ms)	Re-clustering (ms)	Total Time (ms)
500	127.45345	7.3859	207.3626	23.88565	366.0876
1000	270.4083	13.12385	218.8375	60.364	562.73365
2000	519.6833	20.5502	203.411	123.14625	866.79075

Figure 7: **Performance on the big city model:** *Peformance on a city with* 924 *buildings and* 200 *moving cars. All these timings were generated with 2,000 agents. Timings are the average simulation step time broken down by local dynamics, searching for paths, AERO updates, reclustering of the KD-Tree, and total time.*

Group behaviors affect a particular group, as defined by the clustering system. Groups can be made to walk faster or slower, depending on user-specified characteristics for that group. The aggressiveness of an agent is modeled by updating the maximum speed v_{max} and the time to relaxation τ_i. In addition each link in the AERO stores the crowd density across the link. More aggressive groups assign a higher weight to crowded links and commute paths through less dense regions.

6 Implementation and Results

In this section we describe the implementation of our heterogeneous crowd simulation 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. We project the positions of each agent in a PLOD along the link, and the list of agents is sorted by the distance traveled along the link. This enables efficient search of the closest agents within the PLOD. In addition, since all agents within a PLOD have similar velocities, the list is almost sorted and can be resorted in expected linear time. In addition, the sorted list enables efficient collision detection among agents using a sweep and prune technique. The interaction range for social forces in the microscopic model is determined by the average density of the agents within the PLOD. To simulate particle dynamics of the agents, we used a semi-implicit verlet integrator. Proximity computations to the obstacles are accelerated using a spatial hash data structure.

6.1 Benchmarks

We demonstrate our system on two complex scenarios. The first scenario is an outdoor environment consisting of multiple city blocks. The model consists of 924 buildings and 235K triangles. The initial roadmap for the environment consists of 4K links. The environment also consists of 200 cars as dynamic obstacles. As the cars move through the environment, links on the path deform around the obstacles and get invalidated. When links get invalidated, the agents recompute alternate paths. We add a high 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. The heterogeneous crowds in the scene consists of many groups of agents who are assigned independent goals. The behavior characteristics of each agent are assigned at run-time. This includes updating the goals, the maximum speed, and interaction range of the agents. As agents move along the roadmap, we observe several emergent phenomenon of crowd simulation - such as formations of lanes across the same link and banding patterns at crossings.

The second scenarios is an indoor environment of an exhibit hall in a trade show. The exhibit consists of 511 booths and 170K polygons. The initial roadmap consists of 1800 links. Numerous agents walk around and visit multiple booths. The goals for each agent are updated as the agent arrives at a booth. As agents move freely through the floor, they act as dynamic obstacles, and update the AERO.

6.2 Results

We highlight the performance of our algorithm on the complex benchmarks. Our approach can perform real-time simulation of heterogeneous crowds with up to 5000 agents at less than 1 second per frame. Our current implementation is unoptimized and does not make use of processing some of the computations on the GPU. The performance of our algorithm in a two city environments (with different complexity) and varying number of agents is highlighted in Fig. 7 and 8.

6.3 Discussion and Limitations

Our current algorithm uses a combination of force models that globally deform the roadmaps and locally compute the interaction among individual agents. This combination can occasionally lead to instability issue in the dynamics simulation, especially if there are large changes in the scene (e.g. external obstacles). We conjecture that an integration of our approach along with the continuum crowd model [Treuille et al. 2006] will be able to address the issue and extend the continuum model to heterogeneous crowds and further improve its stability.

Our PLOD computation algorithm uses a 3D KD-tree representation for clustering. Each PLOD represents a 6-DOF vector and our algorithm projects it to 3D space corresponding to the (x, y) position in the plane and the goal id. Ideally, we would like to use a 6-dimensional data structure as it would perform more accurate clustering in terms of behavior characteristics. Most of the running time of the algorithm is spent in local dynamics computation and updating the links of AERO. The latter can become a bottleneck if a number of links need to be deleted or added due to the motion of obstacles. Some efficient strategies to handle such cases are presented in [XXX 2007].

# of Agents	Local Dynamics (ms)	Path Search (ms)	AERO Update (ms)	Re-clustering (ms)	Total Time (ms)
500	8.92104	0.31776	10.9428	5.3505	25.5322
1000	51.7944	0.41490	9.6564	1.1218	62.9877
2000	86.0107	0.39252	10.7010	2.5304	99.6347

Figure 8: **Performance on a small city:** *Peformance for a subset of a city with* 10 *static structures and* 4 *moving cars. Real-time simulation of 500 agents. Timings are the average simulation step time broken down by local dynamics, searching for paths, AERO updates, reclustering of the KD-Tree, and total time*

7 Conclusions and Future Work

We present a new crowd simulation algorithm based on PLODs and a dynamic roadmap representation for global navigation of multiple moving agents. Our approach can simulate many groups of people with independent behavior characteristics and goals at nearly interactive rates. Moreover, this formulation can account for macroscopic group interaction as well as microscopic agent behaviors. The simulated crowd movement created using our system exhibits collective effects of pedestrian flows under varying conditions, as observed in real crowds. We have applied the results to simulate crowds in outdoor scenes and initial results are promising.

This work complements existing work by focusing on motion planning for large-scale heterogeneous crowds. However, our current approach makes several simplifying assumptions. First, we do not account for any form of uncertainty in the environment. We also assume perfect visibility of the environment for each agent. In addition, we have not implemented the appropriate force model for capturing panic effects. Our model can be extended to account for these factors and we hope to address them in the future.

There are many other avenues for future work. Moreover, our local dynamics model and route planning algorithm can be significantly accelerated using GPU-based implementation. The local dynamics model can be implemented using fragment programs and we can use distance fields to accelerate the distance computations and link deformations. That can offer considerable speedup and would make it possible for us to handle more complex scenarios with dynamic obstacles. Furthermore, our framework based on PLODs and local dynamics makes it possible to incorporate other macroscopic and microscopic behaviors. Finally, we would like to use our algorithm to model crowd behavior in other environments, including indoor scenes like the airport or train station or more complex outdoor scenes, like a football stadium with tens of thousands of people.

Acknowledgments

Supported in part by ARO Contracts DAAD19-02-1-0390 and W911NF-04-1-0088, NSF awards 0400134, 0429583 and 0404088, DARPA/RDECOM Contract N61339-04-C-0043, Disruptive Technology Office and the DOE High-Performance Computer Science Fellowship administered by he Krell Institute. We would like to thank Sean Curtis for his help with avatar modeling, animation and video editing. We would also like to acknowledge other members of UNC GAMMA group for useful discussions and feedback.

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.

BROGAN, D., AND HODGINS, J. 2002. Simulation level of detail for multiagent control. *Proc. of AAMAS*, 199–206.

CARLSON, D., AND HODGINS, J. 1997. Simulation levels of detail for real-time animation. In *Proc. of Graphics Interface 1997*.

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*.

DOBBYN, S., HAMILL, J., O'CONOR, K., AND O'SULLIVAN, C. 2005. Geopostors: A real-time geometry/impostor crowd rendering system. *ACM Trans. on Graphics 24*, 3.

FORSYTH, D. R. 2006. *Group Dynamics*. Wadsworth Publishing.

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

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.

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/).

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)*.

MULTON, F., VALTON, B., JOUIN, B., AND COZOT, R. 1999. Motion levels of detail for real-time virtual worlds. *Proc. of ASTC-VR'99*.

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

O'BRIEN, D., FISHER, S., AND LIN, M. 2001. Simulation level of detail for automatic simplification of particle system dynamics. *Proc. of Computer Animation*, 210–219.

O'SULLIVAN, C., AND ET AL. 2002. Level of detail for crowds and groups. *Computer graphics Forum 21*, 4, 733–742.

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*.

POPOVIC, Z., AND WITKIN, A. 1999. Physically based motion transformation. In *Proc. of SIGGRAPH 1999*, 11–20.

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.

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

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*. to appear.

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. 1994. Artificial fishes: Physics, locomotion, perception, behavior. In *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, ACM Press, A. Glassner, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH, 43–50. ISBN 0-89791-667-0.

XXX, A. 2007. Reactive deformation roadmaps: Motion planning of multiple robots in dynamic environments. *Technical Report*.

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

ZIPF, G. K. 1949. *Human behavior and the principle of least effort*. Addison-Wesley Press.