Real-Time Path Planning in Dynamic Virtual Environments Using Multiagent Navigation Graphs

Avneesh Sud, Erik Andersen, Sean Curtis, Ming C. Lin, *Member*, *IEEE*, and Dinesh Manocha

Abstract—We present a novel approach for efficient path planning and navigation of multiple virtual agents in complex dynamic scenes. We introduce a new data structure, *Multiagent Navigation Graph* (MaNG), which is constructed using first- and second-order Voronoi diagrams. The MaNG is used to perform route planning and proximity computations for each agent in real time. Moreover, we use the path information and proximity relationships for the local dynamics computation of each agent by extending a social force model [15]. We compute the MaNG using graphics hardware and present culling techniques to accelerate the computation. We also address undersampling issues and present techniques to improve the accuracy of our algorithm. Our algorithm is used for real-time multiagent planning in pursuit-evasion, terrain exploration, and crowd simulation scenarios consisting of hundreds of moving agents, each with a distinct goal.

Index Terms—Crowd simulation, Voronoi diagram, motion planning.

1 Introduction

Crowds, ubiquitous in the real world from groups of humans to schools of fish, are vital features to model in a virtual environment. The realistic simulation of virtual crowds has diverse applications in architecture design, emergency evacuation, urban planning, personnel training, education, and entertainment. Existing work in this area can be broadly classified into *agent-based methods* that focus more on individual behavior and *crowd simulations* that aim to exhibit emergent phenomena of the groups.

In this paper, we address the problem of real-time collision-free navigation of agents moving in a complex virtual environment. Since individuals constantly adjust their behavior according to dynamic factors (for example, another approaching individual) in the environment, agent-based techniques that focus on modeling individual behaviors and intents offer many attractive benefits. They often result in more realistic and detailed simulations. One of the key challenges in a large-scale agent-based simulation is global path planning along with local collision avoidance for each virtual agent. The path planning problem can become very challenging for real-time applications with a large group of moving virtual characters, as each character is a dynamic obstacle for other agents. Many prior techniques are either restricted to static environments or perform only local collision avoidance computations. The latter can result in unnatural behavior or "getting stuck" in local minima. Theseproblems tend to be more prominent in dynamic scenes with multiple moving virtual agents.

Main Results. In this paper, we present a novel real-time algorithm for path planning and navigation of multiple virtual agents in a dynamic environment. We introduce a new data structure called "multiagent navigation graph" (MaNG) and compute it efficiently using GPU-accelerated discrete Voronoi diagrams. Voronoi diagrams have been widely used for path planning computations in static environments [7], [22], and we extend these approaches to dynamic environments. Moreover, we present techniques for local dynamics computation of each agent by extending the model by Helbing et al. [15] and use the proximity relationships computed by MaNG.

Voronoi diagrams capture the connectivity of the space and provide a path of maximal clearance for a robot from other obstacles. In order to use Voronoi diagrams for multiple moving agents in a dynamic scene, prior approaches compute the Voronoi diagram for each agent separately and treat the other agents and the environment as obstacles. This approach can become costly as the number of virtual agents increases. Instead, we compute the *second-order* Voronoi diagram of all the obstacles and agents and show that the second-order Voronoi diagram provides *pairwise* proximity information for all the agents simultaneously. We combine the first- and second-order Voronoi graphs to compute the MaNG for global path planning of multiple virtual agents. Given n dynamic agents, the computational complexity of computing the second-order Voronoi diagram and the MaNG on a discrete grid of resolution $m \times m$ is $O(m^2 + n \log m)$, which is identical to the complexity of computing a first-order discrete Voronoi diagram. Therefore, the computation of global paths using the MaNG is

- *A. Sud is with Microsoft Corporation, One Microsoft Way, Redmond, WA 98052. E-mail: avneesh.sud@microsoft.com.*
- *E. Andersen is with the University of Washington, Computer Science and Engineering Department, 185 Stevens Way AC101, Paul G. Allen Center Box 352350, Seattle, WA 98195-2350. E-mail: eland@cs.washington.edu.*
- *S. Curtis, M.C. Lin, and D. Manocha are with the Department of Computer Science, University of North Carolina, Chapel Hill, NC 27599-3175. E-mail: {seanc, lin, dm}@cs.unc.edu.*

Manuscript received 17 July 2007; revised 1 Nov. 2007; accepted 18 Dec. 2007; published online 17 Jan. 2008.
Recommended for acceptance by B. Sherman and A. Steed.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCGSI-2007-07-0086.
Digital Object Identifier no. 10.1109/TVCG.2008.27.

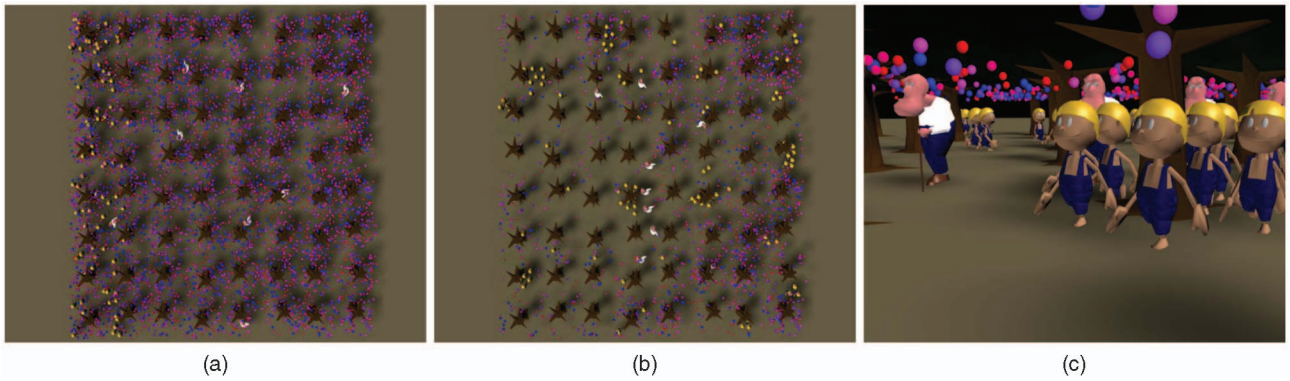


Fig. 1. Fruit stealing simulation. A simulation of 96 fruit pickers (with yellow hair) in an orchard with 64,000 fruits (dark blue and purple) on 64 trees (brown trunks), and four farmers (in white shirts). Each agent maintains an independent goal. (a) Initial top view of the orchard. (b) Top view during the middle of simulation with many fruits collected. (c) Perspective view of the same time step. Our MaNG-based algorithm can perform path planning at 8 fps on a high-end PC.

more efficient than separately computing n first-order Voronoi diagrams.

The MaNG computes paths of maximal clearance for a group of moving agents with different goals *simultaneously* and does not require a separate path planning data structure for each virtual agent. Given the global path for each agent, we also compute the local dynamics for each agent to follow the path generated using MaNG. Our local dynamics model is based on a generalized potential field method and the model by Helbing et al. [15] for capturing emergent phenomenon in real-world crowd motion. Since the MaNG captures pairwise proximity information, we demonstrate that paths computed using the MaNG directly result in collision avoidance among multiple agents. This approach also reduces the number of pairwise proximity tests that need to be performed for local dynamics computation. Furthermore, the MaNG provides paths of maximal clearance, thus resulting in a better coverage of the agents over the environment.

We compute a discrete approximation of the graph structure by using the rasterization hardware and propose an adaptive culling technique to accelerate the computation. We also address the undersampling issues that arise due to discretization and present techniques to improve the accuracy. Some of our key results include

1. a new global data structure, the MaNG for parallel computation of maximal clearance paths among multiple virtual agents moving independently,
2. interactive global path planning and local collision avoidance for multiple virtual agents, each with a possibly different goal, in a complex virtual environment,
3. an improved model for local dynamics computation of each virtual agent and techniques to generate smooth and more natural motion that is similar to real crowds,
4. a fast two-pass algorithm with adaptive culling to compute a discrete MaNG using GPUs, and
5. efficient techniques to handle spatial undersampling issues in the MaNG computed from a discrete grid.

Our overall approach is scalable for global path planning of multiple dynamic agents in a complex virtual world. Although our approach is specifically well suited for simulating multiple virtual agents with distinct intentions, it can also be used in conjunction with crowd simulation. We have demonstrated our algorithm on three challenging scenarios: a pursuit-evasion game of many fruit pickers chased by farmers, a crowd simulation, and terrain exploration by robot rovers. In each of these environments, our algorithm is able to perform real-time global path planning and collision avoidance simultaneously for hundreds of virtual agents with distinct goals.

Organization. The rest of the paper is organized as follows: Section 2 reviews prior literature in related areas. In Section 3, we define our notation and give an overview of our approach. We introduce our data structure, MaNG, and show how it can be used for path planning of multiple agents in Section 4. We describe our algorithm for computing the local dynamics of multiple agents in Section 5. Section 6 describes our algorithm for computing the MaNG in real-time using GPUs. We describe the implementation and highlight three applications of our planning algorithm to complex virtual environments in Section 7 and analyze the algorithm performance in Section 8.

2 Related Work

In this section, we briefly survey related work on multiagent simulation and Voronoi diagrams for path planning.

2.1 Multiple-Agent Simulation

Agent-based methods such as the seminal work of Reynolds [31] generate fast simple local rules that can create visually plausible flocking behavior. Numerous extensions that account for social forces [8], psychological models [29], directional preferences [38], sociological factors [26], etc., have been proposed. Interesting techniques for collision avoidance have also been developed based on grid-based rules [24] and behavior models [42].

Most agent-based techniques perform local collision avoidance. However, global path planning techniques are needed to provide goal-seeking capability. In practice, global planning algorithms typically use graph search techniques for each agent [3], [12], [21], [39]. Pettre et al. [30] proposed a graph structure that decomposes the space into multilayered terrains to support fast graph search for multiple characters. Multiagent path planning has also been investigated extensively in robotics, mostly for performing collaborative tasks [4], [23], [28].

2.2 Voronoi Diagrams and Path Planning

The Voronoi diagram is a fundamental proximity data structure used in computational geometry and related areas [27]. Generalized Voronoi diagrams (GVDs) of polygonal models have been widely used for motion planning [6], [22]. The Voronoi region boundaries in the GVD represent the connectivity of the space. Moreover, they are used to compute paths of maximal clearance between a robot and the obstacles based on potential field approaches [5], [18] or to bias the sample generation for a randomized planner [11], [14], [44]. However, sampling-based methods are limited to static environments, and the potential-field-based planners have been used for 2D environments with very few robots or agents.

A disadvantage of using the GVD is the practical complexity of computing it efficiently and robustly. Hence, several approaches have been proposed to compute an approximation of the GVD. Vleugels and Overmars [43] use adaptive spatial subdivision. Choset and Burdick [6] define a related structure called *hierarchical generalized Voronoi graph*, which is computed using continuation methods. Wilmarth et al. [44] compute points on the GVD without explicitly computing a representation of the entire set. Another set of approaches computes a discrete Voronoi diagram on a uniform grid using graphics hardware [17], [37], [9].

2.3 Crowd Dynamics

There is an extensive amount of literature on crowd simulation and dynamics in computer graphics, as well as architecture, psychology, social sciences, and civil and traffic engineering. Many different approaches have been proposed for modeling crowd movement and simulation [32], [40], [25], [33]. At a broad level, they can be classified based on problem decomposition (discrete versus continuous), stochastic versus deterministic, etc. Discrete methods rely on the discretization of the environment or of the agents. Some common approaches include agent-based methods [31], cellular-automata methods [20], [24], and particle dynamics [32], [16]. In particular, our local dynamics model is based on the *generalized social force* model presented by Helbing et al. [15]. In this approach, physical forces similar to an N-body particle system are computed for each agent. This model by Helbing et al. [15] has previously been applied to simple scenarios and can result in agents getting stuck in a local minimum for more complex environments. We extend that approach to handle complex scenarios and perform global path computations. Recently, a novel approach for crowd simulation based on continuum dynamics has been proposed by Treuille et al. [41]. This work computes 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, where each group has common goals. We provide detailed comparisons with prior approaches in Section 8.

3 Background and Notation

In this section, we introduce the notation used in the paper, give a background on Voronoi-diagram-based motion planning and crowd dynamics, and present an overview of our approach.

3.1 Notation

A geometric primitive or an object (in three dimensions) is called a *site*. In our work, a site refers to a point, an open edge, an open triangle, or a connected polygonal object, and we restrict ourselves to 2D environments. An entity for which a path needs to be computed is called an *agent* (or a robot). All obstacles and agents are represented as sites. The center of mass of a site p_i is denoted as $\pi(p_i)$. The interior and boundary of a set S are denoted $\mathrm{Int}(\mathrm{S})$ and $\partial\mathrm{S}$, respectively.

Given a site p_i, the scalar distance function $d(\mathbf{q}, p_i)$ denotes the distance from the point $\mathbf{q} \in \mathbb{R}^n$ to the closest point on p_i. Given a set of sites P in domain D and a subset T of P, with $|\mathrm{T}| = k$, the kth *order Voronoi region* is the set of points closer to a site in T than to any other site:

$$\mathrm{Vor}^k(\mathrm{T}|\mathrm{P}) = \big\{\mathbf{q} \in \mathrm{D} | d(\mathbf{q}, p_i) \leq d(\mathbf{q}, p_j) \forall p_i \in \mathrm{T}, p_j \in \mathrm{P} \setminus \mathrm{T}\big\}.$$

The kth *order Voronoi diagram* is a partition of the domain D into the kth order Voronoi regions:

$$\mathrm{VD}^k(\mathrm{P}) = \bigcup_{p_i \in \mathrm{P}} \mathrm{Vor}^k(\mathrm{T}, \mathrm{P}), |\mathrm{T}| = k.$$

The standard Voronoi diagram is the same as the first-order Voronoi diagram $\mathrm{VD}^1(\mathrm{P})$. In this paper, we mainly use the first- and second-order Voronoi diagrams, denoted as $\mathrm{VD}^1(\mathrm{P})$ and $\mathrm{VD}^2(\mathrm{P})$, respectively. A first-order Voronoi region $\mathrm{Vor}^1(p_i|\mathrm{P})$ contains points closest to site p_i, and the second-order Voronoi region $\mathrm{Vor}^2(\{p_i, p_j\}|\mathrm{P})$ contains points that are closest to two sites p_i and p_j. For ease of notation, we drop the superscript for the first-order Voronoi diagram $\mathrm{VD}(\mathrm{P})$. The complement of a subdomain X is denoted as X^c and given by $\mathrm{D} \setminus \mathrm{X}$.

The set of closest k-tuples of sites to a point is called the kth order *governor set*. For a point $\mathbf{q} \in \mathrm{D}$, let the set of closest k-tuples of sites be $\mathrm{U} = \{\mathrm{T}_0, \ldots, \mathrm{T}_\mathrm{m}\}$, $|\mathrm{T}_\mathrm{i}| = k$, that is, $\mathbf{q} \in \mathrm{Vor}^k(\mathrm{T}_\mathrm{i}|\mathrm{P})$. Then, the kth order governor set of $\mathbf{q}$ is denoted as $\mathrm{Gov}^k(\mathbf{q}|\mathrm{P}) = \mathrm{U}$. The first-order governor set is the set of closest sites, while the second-order set of a point is the set of closest pairs of sites.

In 2D, the boundaries of Voronoi regions consist of Voronoi edges that are subsets of the bisector between two sites, and Voronoi vertices are equidistant from three or more sites. The arrangement of all Voronoi edges and vertices in the kth order Voronoi diagram is called the kth order *Voronoi graph*, denoted $\mathrm{VG}^k(\mathrm{P})$. Formally, $\mathrm{VG}^k(\mathrm{P}) = (\mathrm{V}, \mathrm{E})$, where

$$\begin{aligned}\mathrm{V} &= \big\{\mathbf{v} \in \mathrm{D} |\ |\mathrm{Gov}^k(\mathbf{v}|\mathrm{P})| \geq 3\big\},\\ \mathrm{E} &= \big\{e | e = (\mathbf{v_1}, \mathbf{v_2}), \mathbf{v_1} \in \mathrm{V}, \mathbf{v_2} \in \mathrm{V}, \exists \text{ connected curve } c, s.t.\\ &\qquad c = \mathrm{Vor}^k(p_i|\mathrm{P}) \cap \mathrm{Vor}^k(p_j|\mathrm{P}), \mathbf{v_1} \in c, \mathbf{v_2} \in c\big\}.\end{aligned}$$

The kth order Voronoi diagram is closely related to the kth *nearest neighbor diagram*. The kth nearest neighbor diagram is the partition of D into kth nearest neighbor regions. The kth nearest neighbor region of site p_i is the set of points for which p_i is the kth nearest site. Similarly, the arrangement of the vertices and edges in the kth nearest neighbor diagram is called the kth nearest neighbor graph, denoted $\mathrm{NG}^k(\mathrm{P})$. Examples of the first- and second-order Voronoi diagrams, Voronoi graphs, and nearest neighbor diagrams are shown in Fig. 2. The first-nearest neighbor

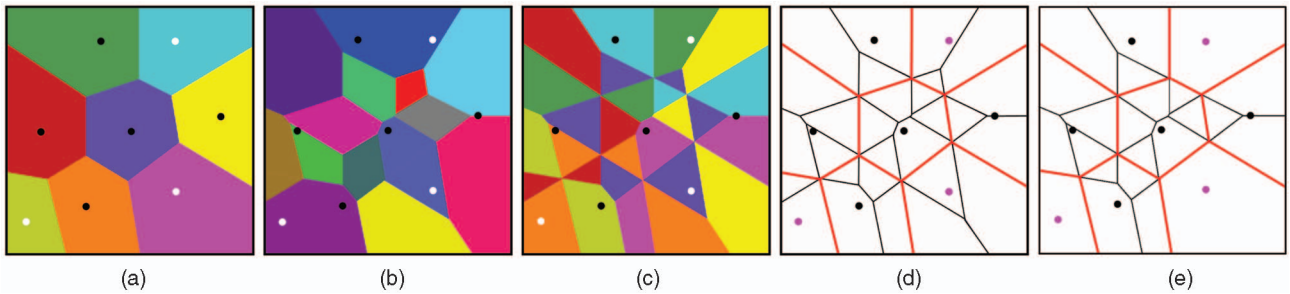


Fig. 2. Voronoi diagrams and Voronoi graphs. Eight point sites consisting of three obstacles (shown in white) and five agents (shown in black). (a) First-order Voronoi diagram. (b) Second-order Voronoi diagram of the eight sites. Each region is closer to a pair of sites than to any other site. (c) Second-nearest neighbor diagram. Each region has the same site as the second-closest site. (d) Second-nearest neighbor graph. Red edges denote edges from the first-order Voronoi graph; black edges are edges from the second-order Voronoi graph. (e) The MaNG for the five agents, which is a subset of the second-nearest neighbor graph.

diagram is identical to the first-order Voronoi diagram. Further properties of higher order Voronoi diagrams are presented in [10] and [27].

3.2 Motion Planning Using Voronoi Diagrams

Voronoi diagrams have been used in motion planning including road-map computation for global approaches and sample generation for randomized approaches or combined with potential field methods for local planners. The set of sites P is the set of obstacles, and the Voronoi diagram of the workspace $\mathrm{VD}(\mathrm{P})$ is computed. The Voronoi graph $\mathrm{VG}(\mathrm{P})$ captures the connectivity of the workspace and provides paths of maximal clearance between the obstacles. The Voronoi vertices closest to the robot and goal are classified as source and destination, and the minimum weight path is then computed.

For complex 3D environments, an approximate Voronoi diagram is computed. The computation of discrete Voronoi diagrams and discrete Voronoi graphs can be accelerated using GPUs and has been used for motion planning in dynamic 2D [19] and 3D environments [37]. The Voronoi vertex closest to the agent is set as an intermediate goal, and the Voronoi diagram is recomputed as the obstacles move.

However, these approaches are inefficient for computing the path of multiple agents in a dynamic environment. For an agent p_i, the remaining agents need to be considered as obstacles, that is, the set of obstacles is $\mathrm{P} \setminus \{p_i\}$. Hence, to compute the path for agent p_i, the modified Voronoi graph $\mathrm{VG}(\mathrm{P} \setminus \{p_i\})$ needs to be computed. Therefore, the cost of computing the path for all agents is $O(nc)$, where n is the number of agents, and $O(c)$ is the cost of computing each modified Voronoi graph $\mathrm{VG}(\mathrm{P} \setminus p_i)$ for $1 \leq i \leq n$.

3.3 Crowd Dynamics

In addition to global path planning, we also compute the local dynamics for each agent in the overall simulation. Our dynamics computation model builds upon the vast literature on pedestrian dynamics in psychology, transportation science, and civil and traffic engineering. One of the key underlying behavioral characteristics in pedestrian dynamics is the *principle of least effort* [45]. This implies that among all available options (for example, accelerating, decelerating, changing direction, or doing nothing), a pedestrian or agent tries to choose the option that will yield the smallest predicted *disutility*. An agent will try to adapt to its environment or will try to change the environment to suit its needs, if easier. Under this assumption, the flow of agents self-evolves into a user-equilibrium state with the emergence of several interesting collective effects at various scales and densities of crowds [15]. Examples of such emergent phenomena include dynamics lane formation, oscillations at bottlenecks, banding patterns at intersections, trail following, and panic effects. However, most of the prior work simulates crowd dynamics in localized environments (for example, corridors and intersections) and does not guarantee global goal seeking in complex dynamic environments. We would like to provide a global collision-free path for each agent in dynamic environments. Using the MaNG, our approach provides a global path in addition to local collision avoidance.

3.4 Overview

Our approach for motion planning of multiple agents uses the first- and second-order Voronoi diagrams to compute a global navigation data structure, the MaNG. The MaNG can be considered as the union of the first- and the second-order Voronoi graphs and is formally presented in Section 4. We treat each agent as a site (in addition to other obstacles in the environment), and the MaNG is computed. The MaNG can be computed in time $O(c)$ and provides a path of maximal clearance for each agent, where $O(c)$ is the cost of computing each modified Voronoi graph (see Section 3.2).

In addition, we compute the proximity information from the second-order Voronoi diagram [35] and apply it within a potential-field-based simulator based on the "generalized social force" model [15], which provides for adding smooth natural motions as observed in human crowds. In this model, different physical interactions among the agents such as obstacle avoidance, grouping, and goal seeking are modeled using potential forces. We integrate this force model with our MaNG-based planner. The global path is computed using the MaNG, and we add a road-map force to guide the agent along its path. We also exploit proximity information from the MaNG to avoid interagent collisions.

4 Multiagent Planning Using Hybrid Voronoi Structures

In this section, we introduce the MaNG and demonstrate its application to multiple-agent planning. We combine them with a local dynamics model for multiagent simulation in Section 5.

4.1 Multiagent Navigation Graph

In multiagent planning, each moving agent represents a dynamic obstacle for the remaining agents. Hence, our goal is

to compute a global navigation data structure that provides the clearance and proximity information for each agent. In particular, for each agent, we want to compute a proximity graph that provides maximal clearance to the obstacles and remaining agents. We partition the set of sites P into two subsets—the set of obstacles P_o and the set of agents P_a. The MaNG, denoted $\mathrm{MG}(\mathrm{P})$, is a union of the first-order Voronoi graph $\mathrm{VG}^1(\mathrm{P})$ and a subset of the second-order Voronoi graph $\mathrm{VG}^2(\mathrm{P})$. The subset of $\mathrm{VG}^2(\mathrm{P})$ is the intersection of $\mathrm{VG}^2(\mathrm{P})$ and the Voronoi regions $\mathrm{Vor}(p_i|\mathrm{P})$ of all agents. Formally

$$\begin{aligned}\mathrm{MG}(\mathrm{P}) &= (\mathrm{V},\mathrm{E}), \quad \text{where}\\ \mathrm{V} &= \left\{v \mid v \in \mathrm{V}^1 \cup \left(\mathrm{V}^2 \cap \mathrm{Vor}(p_i|\mathrm{P})\right) \forall p_i \in \mathrm{P}_a\right\},\\ \mathrm{E} &= \left\{e \mid e \in \mathrm{E}^1 \cup \left(\mathrm{E}^2 \cap \mathrm{Vor}(p_i|\mathrm{P})\right) \forall p_i \in \mathrm{P}_a\right\},\\ \mathrm{VG}^1(\mathrm{P}) &= (\mathrm{V}^1,\mathrm{E}^1) \text{ and } \mathrm{VG}^2(\mathrm{P}) = (\mathrm{V}^2,\mathrm{E}^2).\end{aligned}$$

$\mathrm{MG}(\mathrm{P})$ consists of vertices and edges from the first-and second-order Voronoi graphs $\mathrm{VG}^1(\mathrm{P})$ and $\mathrm{VG}^2(\mathrm{P})$. Some vertices in $\mathrm{MG}(\mathrm{P})$ are common to both $\mathrm{VG}^1(\mathrm{P})$ and $\mathrm{VG}^2(\mathrm{P})$; however, $\mathrm{VG}^1(\mathrm{P})$ and $\mathrm{VG}^2(\mathrm{P})$ do not share any edge [27].

We assign a color to each edge and vertex in $\mathrm{MG}(\mathrm{P})$ based on its membership in $\mathrm{VG}^1(\mathrm{P})$ or $\mathrm{VG}^2(\mathrm{P})$. Edges from $\mathrm{VG}^1(\mathrm{P})$ are colored **red**, and edges from $\mathrm{VG}^2(\mathrm{P})$ are colored **black**. Further, vertices in $\mathrm{VG}^1(\mathrm{P})$ are colored red, and vertices in $\mathrm{VG}^2(\mathrm{P}) \setminus \mathrm{VG}^1(\mathrm{P})$ are colored black. Finally, each edge in the MaNG is assigned a weight based on the cost of traveling that segment. Details on weight computation are presented in Section 6. A 2D example of the MaNG for some point agents and obstacles is shown in Fig. 2.

$\mathrm{MG}(\mathrm{P})$ is closely related to the second-nearest neighbor graph $\mathrm{NG}^2(\mathrm{P})$. In particular, we use the following result about their relationship.

Lemma 1. *Given a set of sites P, the second-nearest neighbor graph $\mathrm{NG}^2(\mathrm{P})$, and the MaNG $\mathrm{MG}(\mathrm{P})$:*

- $\mathrm{MG}(\mathrm{P}) \subseteq \mathrm{NG}^2(\mathrm{P})$.
- *Given an edge $e \in \mathrm{MG}(\mathrm{P})$ incident on two second-nearest neighbor regions of sites p_i and p_j. For any point $\mathbf{x} \in \mathrm{Int}(e)$, $d(\mathbf{x},p_i) = d(\mathbf{x},p_j) = d(\mathbf{x},\mathrm{P}) \Rightarrow e \in \mathrm{VG}^1(\mathrm{P})$. $d(\mathbf{x},p_j) \neq d(\mathbf{x},p_i) \Rightarrow e \in \mathrm{VG}^2(\mathrm{P})$.*

Proof. Let $\mathrm{MG}(\mathrm{P}) = (\mathrm{V},\mathrm{E})$ and $\mathrm{NG}^2(\mathrm{P}) = (\mathrm{E}',\mathrm{V}')$. We shall show that $\mathrm{E} \subseteq \mathrm{E}'$ and $\mathrm{V} \subseteq \mathrm{V}'$. Let $e \in \mathrm{E}$ be any edge in the MaNG. Then, we have two cases:

1. $e \in \mathrm{E}^1$ (e belongs to $\mathrm{VG}^1(\mathrm{P})$). Let $\mathrm{Gov}^1(e) = \{p_i,p_j\}$ be the governors of e. Then, $e \subseteq \partial\mathrm{Vor}^1(p_i|\mathrm{P})$. For any point $\mathbf{x} \in e$, $d(\mathbf{x},p_i) = d(\mathbf{x},p_j) = d(\mathbf{x},\mathrm{P})$ (by the definition of $\mathrm{VG}^1(\mathrm{P})$). Consider the region $\mathrm{X} = \mathrm{Vor}^1(p_i|\mathrm{P}) \cap \mathrm{Vor}^2(\{p_i,p_j\}|\mathrm{P})$. Further, for any point in X, p_j is the second-closest site, and X is a subset of the second-nearest neighbor region of p_j. Since $e \subseteq \partial(\mathrm{X})$, $e \in \mathrm{E}'$. In particular, e is a red edge in MaNG, shown in Figs. 2d and 2e.
2. $e \in \mathrm{E}^2$ (e belongs to $\mathrm{VG}^2(\mathrm{P})$). Each edge in $\mathrm{VG}^2(\mathrm{P})$ is contained entirely in a first-order Voronoi region [27]. Let $e \subset \mathrm{Vor}^1(p_i|\mathrm{P})$. Since e is incident on two second-nearest neighbor regions of sites p_i and p_j, $\{p_i,p_j\}$ forms one governor pair of e. Consider the region $\mathrm{X} = \mathrm{Vor}^1(p_i|\mathrm{P}) \cap \mathrm{Vor}^2(\{p_i,p_j\}|\mathrm{P})$. For any point in X, p_j is the second-closest site, and X is a subset of the second-nearest neighbor region of p_j. Since $e \subseteq \partial(\mathrm{X})$, $e \in \mathrm{E}'$. Further, since $\mathrm{Int}(e) \subset \mathrm{Int}(\mathrm{Vor}^1(p_i))$, for any $\mathbf{x} \in \mathrm{Int}(e)$, $d(\mathbf{x},p_i) = d(\mathbf{x},\mathrm{P})) < d(\mathbf{x},p_j)$. In particular, e is a black edge in MaNG, shown in Fig. 2.

The proof for any vertex $v \in \mathrm{V}$ follows as above. The governors of a vertex from $\mathrm{VG}^1(\mathrm{P})$ are three or more sites, and the governors of a vertex from $\mathrm{VG}^2(\mathrm{P})$ are three or more pairs of sites. □

As a consequence of Lemma 1, both the first- and second-order Voronoi graphs can be extracted from the second-nearest neighbor diagram. We use this result to efficiently compute the MaNG from the second-nearest neighbor diagram in Section 6.

4.2 Multiple-Agent Planning

In this section, we present our approach for efficient path planning of multiple agents using MaNG. The path planning problem for each agent is defined as follows: we are given an agent $p_i \in \mathrm{P}_a$, its current position in the workspace given by its center of mass $\pi(p_i)$, and a goal position of the center of mass $\mathbf{g_i}$. We wish to compute a path for p_i from $\pi(p_i)$ to $\mathbf{g_i}$, which is maximally clear and collision free to the remaining sites $\mathrm{P} \setminus \{p_i\}$. Such a path can be computed using the Voronoi graph $\mathrm{VG}^1(\mathrm{P} \setminus \{p_i\})$. We state a result on the equivalence of the paths computed using the first-order Voronoi graphs and the MaNG.

Lemma 2. *Given an agent p_i and the Voronoi graphs $\mathrm{VG}^1(\mathrm{P} \setminus \{p_i\})$ and $\mathrm{MG}(\mathrm{P})$:*

1. $\mathrm{VG}^1(\mathrm{P} \setminus \{p_i\}) \subseteq \mathrm{MG}(\mathrm{P})$.
2. $\mathrm{VG}^1(\mathrm{P} \setminus \{p_i\}) \cap \mathrm{Vor}(p_i|\mathrm{P}) = \mathrm{MG}(\mathrm{P}) \cap \mathrm{Vor}(p_i|\mathrm{P})$.

Proof. Let $\mathrm{VG}^1(\mathrm{P} \setminus \{p_i\}) = (\mathrm{V}',\mathrm{E}')$, $\mathrm{MG}(\mathrm{P}) = (\mathrm{V},\mathrm{E})$, $\mathrm{VG}^1(\mathrm{P}) = (\mathrm{V}^1,\mathrm{E}^1)$, and $\mathrm{VG}^2(\mathrm{P}) = (\mathrm{V}^2,\mathrm{E}^2)$. We shall show that $\mathrm{E}' \subseteq \mathrm{E}$ and $\mathrm{V}' \subseteq \mathrm{V}$. Let $e \in \mathrm{E}^1$. Then, we have two cases:

1. $e \cap \mathrm{Vor}^1(p_i|\mathrm{P}) = \emptyset$. Then, for any point $\mathbf{x} \in e$, $p_i \notin \mathrm{Gov}^1(\mathbf{x}|\mathrm{P})$, and $e \in \mathrm{E}^1$. Since $\mathrm{VG}^1(\mathrm{P}) \subseteq \mathrm{MG}(\mathrm{P})$, $e \in \mathrm{MG}(\mathrm{P})$. In particular, e is a red edge in Fig. 2.
2. $e \cap \mathrm{Vor}^1(p_i|\mathrm{P}) \neq \emptyset$. Consider the subsets $e_1 = e \cap \mathrm{Vor}^1(p_i|\mathrm{P})$ and $e_2 = e \setminus e_1$. Okabe et al. [28] show that $e_1 \in \mathrm{E}^2$. Also, p_i is an agent; thus, $p_i \in \mathrm{P}_a$, and $e_1 \in \mathrm{MG}(\mathrm{P})$. In particular, e_1 is a black edge inside $\mathrm{Vor}^1(p_i|\mathrm{P})$ in Fig. 2. For any point $\mathbf{x} \in e_2$, $p_i \notin \mathrm{Gov}^1(\mathbf{x}|\mathrm{P})$, and $e \in \mathrm{E}^1$. Thus, $e_2 \in \mathrm{MG}(\mathrm{P})$, and e_2 is a red edge in Fig. 2.

To prove (2), consider all the black edges in $\mathrm{Vor}^1(p_i|\mathrm{P})$. As shown in the second case above, these are exactly the segments from $\mathrm{E}' \cap \mathrm{Vor}^1(p_i|\mathrm{P})$. The proof for the set of vertices follows as above. □

Lemma 2 provides an approach for extracting the Voronoi graph $\mathrm{VG}^1(\mathrm{P} \setminus \{p_i\})$, for each agent p_i, from $\mathrm{MG}(\mathrm{P})$. The complete algorithm for computing a path for an agent p_i is given in Algorithm 1, and an example path is shown in Fig. 3. The function $LocatePoint(\mathbf{g_i})$ returns the

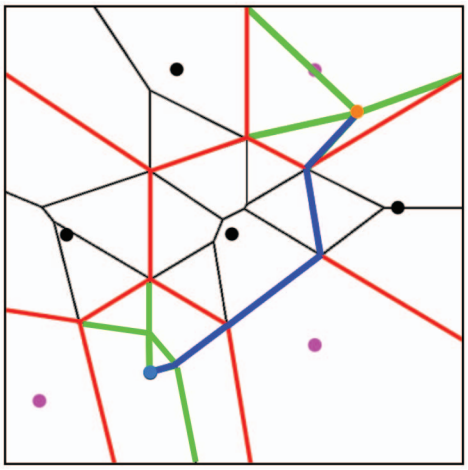


Fig. 3. Multiagent path planning using the MaNG. The MaNG is augmented with green edges connecting the start position (blue dot) to the goal position (orange dot). The computed shortest path for one agent is shown with blue edges.

first-order Voronoi region that contains $\mathbf{g_i}$. The source and goal positions are connected to vertices in the MaNG using green edges. $ShortestPath(p_i, \mathbf{g_i}, \mathrm{MG(P)})$ computes the minimum weight path from $\pi(p_i)$ to $\mathbf{g_i}$ following only the green and red edges in $\mathrm{MG(P)}$. This is equivalent to computing the shortest path by following the second-order Voronoi graph inside the first-order Voronoi region of agent p_i and the first-order Voronoi graph everywhere else (see Fig. 3). The first vertex along this path is chosen as an intermediate goal for agent p_i.

Algorithm 1. $\mathrm{ComputePath}(p_i, \mathbf{g_i}, \mathrm{P}, \mathrm{MG(P)})$: Computes a path for agent p_i to goal $\mathbf{g_i}$ given the set of sites P and the MaNG $\mathrm{MG(P)}$.

Input: Agent p_i, Goal position $\mathbf{g_i}$, Set of sites P, MaNG MG(P)
Output: Path S_i from current position to goal position

$k \leftarrow$ LocatePoint($\mathbf{g_i}$)
if $k = i$ **then**
  $S_i \leftarrow edge(\pi(p_i), \mathbf{g_i})$
  return
Compute $V_i \leftarrow$ set of black vertices in $\mathrm{Vor}(p_i|\mathrm{P})$
Compute $E_i \leftarrow$ set of black edges in $\mathrm{Vor}(p_i|\mathrm{P})$
if $V_i \neq \emptyset$ *and* $\pi(p_i) \notin V_i$ **then**
  Augment MG(P) with green edges,
    $e_j = (\pi(p_i), v_j) \forall v_j \in V_i$
  Assign weight to $e_j, w(e_j) \leftarrow d(\pi(p_i), v_j)$
else
  foreach *edge* $e_j \in E_i$ **do**
    Compute $v_j \leftarrow$ closest point on e_j to $\pi(p_i)$
    Augment MG(P) with green edge $e_j = (\pi(p_i), v_j)$
    Assign weight to $e_j, w(e_j) \leftarrow d(\pi(p_i), v_j)$
  end
Compute $V_k \leftarrow$ set of red vertices in $\mathrm{Vor}(p_k|\mathrm{P})$
Augment MG(P) with green edges $e_j = (\mathbf{g_i}, v_j) \forall v_j \in V_k$
Assign weight to $e_j, w(e_j) \leftarrow d(\mathbf{g_i}, v_j)$
Add green labels to each edge $e_j \in E_i$
$S_i \leftarrow$ ShortestPath (p_i,$\mathbf{g_i}$,MG(P))
Remove green labels from each edge $e_j \in E_i$
Remove all green edges from MG(P)

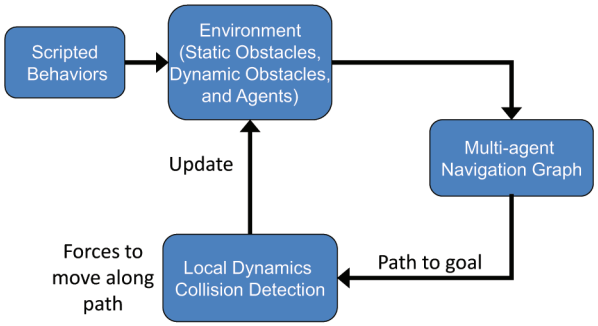


Fig. 4. Multiagent simulation. Given a description of the environment, the MaNG is computed. This is used in conjunction with our local dynamics model to simulate the motion of each agent. The environment is updated using agent motion, as well as behavior specification for each agent.

5 Multiagent Simulation

In this section, we present the multiagent simulation algorithm based on MaNG. First, we present our local dynamics computation model, and after that, we show how MaNG is used in the computation of collision-free paths (Fig. 4).

5.1 Local Dynamics Computation

Given a path on MaNG, the motion of each agent is computed using a local dynamics model. In this section, we describe the local dynamics model used to guide agents along the computed path. Our local dynamics model is based on the generalized force model of pedestrian dynamics proposed by Helbing et al. [16]. 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 Voronoi region.

We modify the social force model to include a force $\mathbf{F}^r$ that guides an agent along the edges of the MaNG. 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 $\mathbf{F}^{obs}$. Given an agent p_i and Voronoi region $\mathrm{Vor}(p_i)$, the force field at a point $\mathbf{p}$ is given as

$$\mathbf{F}(\mathbf{p}) = \sum_j \left[\mathbf{F}_j^{soc}(\mathbf{p}) + \mathbf{F}_j^{att}(\mathbf{p})\right] + \mathbf{F}_i^r(\mathbf{p}) + \sum_o \mathbf{F}_o^{obs}(\mathbf{p}),$$
$$p_j \in \mathrm{P}, j \neq i, o \in \mathrm{O},$$

where

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

where A_i and B_i are constants that denote the strength and range of repulsive interactions, respectively, and C_j is

the strength of attractive interaction. These constants are dependent on the individual behavior characteristics of the agents. λ_i reflects the 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 that 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 [41].

The road-map force field $\mathbf{F}_i^r$ guides the agent p_i along the shortest path to the nearest point on the MaNG. Let v_i be the first vertex on the shortest path S_i computed using Algorithm 1. Clearly, v_i lies on the MaNG, in particular, $v_i \in \mathrm{MG}(\mathrm{P}) \cap \mathrm{Vor}(p_i|\mathrm{P})$. The first term in $\mathbf{F}_k^r$ makes the agent achieve a desired velocity toward its immediate goal v_i. The desired velocity is given as $\mathbf{v}_k^d(\mathbf{p}) = v_{max}\mathbf{e}_k(\mathbf{p})$, where $\mathbf{e}_k(\mathbf{p})$ is a unit vector field in the direction $v_i - \pi(p_i)$. The direction of the unit vector is chosen such that $\mathbf{e}_k(\mathbf{p})$ points along the road map toward the next goal on the agents path. For the efficient computation of repulsive force $\mathbf{F}^{soc}$ and obstacle force $\mathbf{F}^{obs}$, we compute forces to the agents and obstacles within a radius B_i. To accelerate the distance queries, we use the Voronoi diagram $\mathrm{VD}^1(\mathrm{P})$ and compute forces using the Voronoi neighbors of each agent.

5.2 Collision Avoidance Using MaNG

In this section, we describe our local collision avoidance approach used for local navigation. First, we list a property of the shortest paths computed using the MaNG that is used for collision avoidance. Next, we describe a collision avoidance scheme used as part of the local dynamics simulator.

The MaNG is used to compute paths of maximum clearance for each agent, that is, the computed path is maximally clear of the obstacles and other agents. We list a result that guarantees uniqueness of the next immediate goal, as computed by Algorithm 1.

Lemma 3. *Given a set of (at least three) agents P, in general position, and the shortest path S_i for each agent p_i. Let v_i be the first vertex on S_i. Then, $v_i \neq v_j$ for any $j \neq i$, $p_i, p_j \in \mathrm{P}$.*

Proof. Let p_i be any agent. By construction (Algorithm 1), the next intermediate goal v_i lies on the MaNG contained in the Voronoi region $\mathrm{Vor}(p_i|\mathrm{P})$, that is, $v_i \in \mathrm{MG}(\mathrm{P}) \cap \mathrm{Vor}(p_i|\mathrm{P})$. To guarantee that each vertex is unique, we shall prove that v_i belongs to the interior of $\mathrm{Vor}(p_i|\mathrm{P})$. Since the agents are in general position and there are at least three agents, $\partial\mathrm{Vor}(p_i|\mathrm{P})$ consists of at least one Voronoi vertex $\mathbf{x}$. Let $\mathrm{Gov}(\mathbf{x}) = \{p_i, p_j, p_k\}$. Furthermore, any Voronoi region is not degenerate; thus, $\mathrm{Int}(\mathrm{Vor}(p_i|\mathrm{P})) \neq \emptyset$. Thus, there exists at least one point $\mathbf{p} \in \mathrm{Int}(\mathrm{Vor}(p_i|\mathrm{P}))$ such that $d(\mathbf{p}, p_j) = d(\mathbf{p}, p_k)$, that is, $\mathbf{p}$ lies on a Voronoi edge equidistant from p_j and p_k in the Voronoi diagram $\mathrm{VD}^1(\mathrm{P} \setminus p_i)$. Thus, $\mathbf{p} \in \mathrm{VG}^1(\mathrm{P} \setminus \{p_i\}) \cap \mathrm{Int}(\mathrm{Vor}(p_i|\mathrm{P}))$, and by Lemma 2, $\mathbf{p} \in \mathrm{MG}(\mathrm{P}) \cap \mathrm{Int}(\mathrm{Vor}(p_i|\mathrm{P}))$. We shall now show that $v_i = \mathbf{p}$, as selected by Algorithm 1. If the locus of all such points $\mathbf{p}$ contains a Voronoi vertex from $\mathrm{VG}^1(\mathrm{P} \setminus \{p_i\})$ (that is, some $\mathbf{p}$ are black vertices), then the set V_i in Algorithm 1 is nonempty, and $v_i \in V_i$, and $v_i \in \mathrm{Int}(\mathrm{Vor}(p_i|\mathrm{P}))$. If the locus of all such points $\mathbf{p}$ does not contain a Voronoi vertex from $\mathrm{VG}^1(\mathrm{P} \setminus \{p_i\})$ (that is, all $\mathbf{p}$ lie on black edges), then v_i is chosen as a point $\mathbf{p}$ closest to p_i, and $v_i \in \mathrm{Int}(\mathrm{Vor}(p_i|\mathrm{P}))$. Since $\mathrm{Int}(\mathrm{Vor}(p_i|\mathrm{P})) \cap \mathrm{Int}(\mathrm{Vor}(p_j|\mathrm{P})) = \emptyset$, $v_i \neq v_j$. □

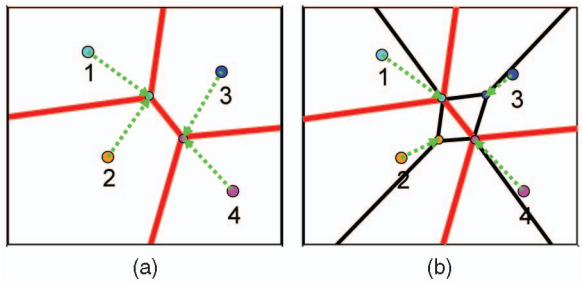


Fig. 5. Comparison of the first-order Voronoi graph and MaNG: four agents, with goals in opposite corners. (a) Intermediate goals computed from the first-order Voronoi graph. Pairs of agents move toward the same goal. (b) Intermediate goals from MaNG. Each agent has a unique intermediate goal.

As a result of Lemma 3, the intermediate goal computed for each agent is unique. Hence, by following the path computed from the MaNG, the agents move toward a unique point, away from the remaining agents. This is shown in Fig. 5 and compared to paths computed using first-order Voronoi graphs.

However, when coupled with a local dynamics simulator, the MaNG alone does not guarantee collision avoidance among multiple agents. This problem is due to the fact that the motion of an agent is not constrained to the MaNG but is governed by a local force model that guides an agent along the MaNG. Furthermore, the MaNG is a geometric data structure, and paths computed from the MaNG do not satisfy the kinodynamic motion constraints of each agent. Hence, neighboring agents may approach each other due to the local dynamics computation. In addition, as explained in Section 6, we compute a discrete approximation to the MaNG. Due to undersampling errors, the intermediate goal of an agent may coincide with the first-order Voronoi graph $\mathrm{VG}^1(\mathrm{P})$, and Lemma 3 may not be valid.

We introduce a constraint-dynamics-based collision response in our local dynamics model. To avoid collisions with the obstacles, we set the agent velocity and applied force to be zero along the normal direction to the obstacle [2]. In addition, to enable agents to move around each other, we add a coordination rule that makes agents move to the right of the nearest approaching agent to avoid collisions with approaching agents [21].

6 MaNG Computation

In this section, we present our approach for efficiently computing the MaNG, which is based on the first- and the second-order Voronoi diagrams. However, the exact computation of GVDs of polygonal models is nontrivial. Rather, we compute the discrete Voronoi diagram along a uniform grid using graphics hardware [17]. The second-nearest neighbor diagram is computed using a second pass with depth peeling, as presented in [10]. We compute the generalized second-nearest diagram of higher order sites (lines and polygons) by rendering the generalized distance function

for each site [37]. We compute the first-order Voronoi diagram in the first pass and compute the second-nearest neighbor diagram in the second pass. Finally, we extract the first- and the second-order Voronoi graphs from the second-nearest neighbor diagram and compute the MaNG.

6.1 Culling Techniques

The distance field is computed by evaluating the distance function to each site at each pixel, and this computation is efficiently performed using the rasterization capabilities of the GPU. However, for a large number of sites, this leads to redundant computation for each pixel, and the computation becomes fill bound. Hence, we use culling techniques to compute conservative bounds on the first- and second-order Voronoi regions. The distance function to each site is computed on the pixels that are contained within its conservative Voronoi region.

Our goal is to efficiently derive a tight upper bound on the first- and second-order Voronoi regions for each site. We compute these bounds by determining the closest site (and two closest sites) along each principle direction $(+X, -X, +Y, -Y)$. We compute the bounds using a quadtree, which subdivides the domain. Each node in the quadtree contains the number of sites contained in the subtree rooted at the node. Using this quadtree, we can efficiently determine the set of nearest neighbors for each site.

The quadtree is constructed as follows: Each leaf node contains the number of sites contained within the node. Let δ be the size of a leaf node. Each intermediate node contains the number of sites contained in all of its four children. Let the function $E(l)$ compute the closest nonempty leaf node to the right of node l in the quadtree. Similarly, $W(l)$, $N(l)$, and $S(l)$, respectively, return closest leaf nodes to the left, top, and bottom of node l. To compute the bound along $+X$ for the first-order Voronoi region of a site p_i, we first identify the leaf node l_i that contains the centroid of the site $\pi(p_i)$. Next, we compute the closest leaf nodes $E(l_i)$, $W(l_i)$, $N(l_i)$, and $S(l_i)$. Let b_1 be the bisector between the centroids of nodes l_i and $E(l_i)$, b_2 be the bisector between the centroids of l_i and $S(l_i)$, and b_3 be the bisector between the centroids of l_i and $N(l_i)$. Let $\mathbf{x}_1 = b_1 \cap b_2$ and $\mathbf{x}_2 = b_1 \cap b_3$. Along $+X$, the bound on the first-order Voronoi region of p_i is given by $+dx = (\pi(p_i) + \Delta X^+ + \delta)$, where $\Delta X^+ = \min((\mathbf{x}_1 - \pi(p_i)) \cdot (1,0), (\mathbf{x}_2 - \pi(p_i)) \cdot (1,0))$. Similarly, the bounds $-dx$, $+dy$, and $-dy$ along the $-X$-, $+Y$-, and $-Y$-axes are computed, and the first-order Voronoi region is bounded by a quad covering the interval $[-dx, +dx] \times [-dy, +dy]$. In addition, for a leaf node l_i, we store the locations of its closest neighbors $E(l_i)$, $W(l_i)$, $N(l_i)$, and $S(l_i)$.

We compute the bounds on all the second-order Voronoi regions of site p_i in the second pass as follows: Along the $+X$-axis, we check the number of sites stored in the closest node $E(l_i)$. If the number of sites in node $E(l_i)$ is two or more, then the bound along $+X$ is $\Delta X^+ = d(l_i, E(l_i)) + 2\delta$. If the number of sites in node $E(l_i)$ is less than two, then we look up the node $E(E(l_i))$ (this has been computed in the first pass), and the bound along $+X$ is $\Delta X^+ = d(l^i, E(E(l_i))) + 2\delta$. Similarly, we compute bounds along the $-X$-, $+Y$-, and $-Y$-axes and compute the distance function of site p_i in a quad that covers these bounds.

To compute the bounds for a higher order site (a line segment or a convex polygon), we store the position of the centroid of the site in the quadtree. We compute the distance bounds for the centroid using the quadtree and add the distance between the centroid and a vertex to compute the distance bounds for the site.

6.2 Undersampling Errors

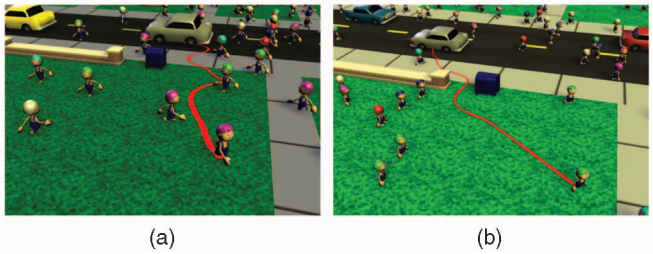
The computation of the Voronoi graph on a uniform grid may result in undersampling errors, which may lead the Voronoi regions to become disconnected [37], and the computed discrete Voronoi graph may have many small disjoint components [18]. As a result, for complex environments with a large number of sites, the combinatorial complexity of the MaNG becomes very high. We address the issue of undersampling for motion planning by reducing the combinatorial complexity of the MaNG without changing its connectivity. We reduce the complexity by appropriately modifying the MaNG near undersampled areas. We rely on the fact that when two Voronoi edges are arbitrarily close, then the agent might follow either edge, as long as the path connectivity does not change. Such edges can be removed from the MaNG provided that their removal does not change the connectivity of the MaNG.

We present the details of our algorithm for reducing the complexity of MaNG. We treat an edge with an adjacent edge less than one pixel away as a candidate for removal. Such edges are exactly those edges that bound a discrete Voronoi region of width 1 pixel. Thus, the test for eliminating such edges is equivalent to removing certain pixels from a discrete Voronoi region, which does not change the connectivity of the Voronoi graph. Hence, our test for the removal of a pixel from a discrete Voronoi region relies on a local 3×3 stencil around a pixel. Let p_a be the governor of a pixel (i, j), and the set α denote the governor set of the four adjacent pixels $(i-1, j)$, $(i+1, j)$, $(i, j-1)$, and $(i, j+1)$. Then, the pixel (i, j) can be removed if either of the following conditions holds (see Fig. 7):

1. $p_a \notin \alpha$. Then, site p_a has an isolated discrete Voronoi region at pixel (i, j), with four Voronoi edges surrounding it. The removal of this Voronoi region does not change the path connectivity in the stencil.
2. $p_a \in \alpha$, and p_a occurs in α exactly once. Then, the pixel (i, j) represents an end point of a discrete Voronoi region of site p_a, and its removal does not change the path connectivity in the stencil.

After a pixel (i, j) satisfies the criteria for removal, we assign it to another discrete Voronoi region to maintain the connectivity of Voronoi edges. The pixel is assigned to a site in $\alpha \setminus \{p_a\}$ with the minimum distance to pixel (i, j). The distance of a site in α to pixel (i, j) can be efficiently computed by relying on the fact that distance vectors are bilinearly interpolated [36]. Thus, distance computation involves a vector summation with a basis vector and vector norm computation.

The operation performed at each pixel is a read followed by a conditional write, and the output of one pixel may affect the connectivity of adjacent pixels. Thus, an efficient parallel algorithm is not feasible, and we perform a sequential scan of the discrete Voronoi diagram to update the Voronoi graph.



(a) (b)

Fig. 6. Path trace of an agent with the same goal in two crowd simulations. The red curve traces the position of an agent over a range of time steps. (a) Path trace of an agent using the approach presented in [34]. (b) Path trace of an agent with the same goal based on our new local dynamics model. Note that due to a different simulation framework, the relative positions of agents is not identical. In comparison to that in [34], our approach results in a more direct and natural-looking path.

6.3 Graph Construction

We now present our algorithm to compute the MaNG. We compute the first-order Voronoi diagram $\mathrm{VD}^1(\mathrm{P})$ and the second-nearest neighbor diagram on the GPU and refine the connectivity information based on the algorithm described in Section 6.2. We then perform sequential tracing of vertices and edges to compute the second-nearest neighbor graph [17].

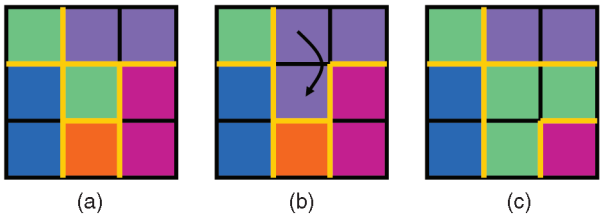
We use the result presented in Lemma 1 to classify the edges in the second-nearest neighbor graph $\mathrm{NG}^2(\mathrm{P})$. An edge is classified as belonging to the first-order Voronoi graph if the distance to the closest site for all pixels on the edge is identical in $\mathrm{VG}^1(\mathrm{P})$ and $\mathrm{NG}^2(\mathrm{P})$. Due to pixel resolution errors, we treat two distance values as identical if they are within one pixel width of each other. Each edge is assigned a weight proportional to its length and inversely proportional to the minimal clearance along the edge. An edge belonging to $\mathrm{VG}^1(\mathrm{P})$ is labeled red, and the remaining edges are labeled black. A vertex is labeled red if it has at least one red edge incident on it; otherwise, it is labeled black. These colors are used by Algorithm 1 to search for an optimal path.

7 Implementation and Results

In this section, we describe the implementation of our multiagent planning algorithm and highlight its application to various multiagent simulations.

7.1 Implementation

We have implemented our algorithm on a PC running the Windows XP operating system with an AMD Opteron 280 CPU, 2 Gbytes of memory, and an Nvidia 7900 GPU. We used OpenGL as the graphics API and Cg language for implementing the fragment programs. The discrete Voronoi diagram and distance field are computed at 32-bit floating-point precision using floating-point buffers. The Voronoi diagram is stored in the red channel, and the distance field, in the depth buffer. We use stencil tests to disable the second-order Voronoi diagram computation in the first-order Voronoi regions of the obstacles. In the first pass, the stencil mask is set for all pixels in the first-order Voronoi regions of the agents. In the second pass, distance functions are evaluated at pixels with a stencil mask set. This optimization speeds up both the discrete Voronoi diagram computation and the MaNG construction. We perform readback of the discrete Voronoi diagrams and construct the MaNG on the CPU. The optimal path is computed using an A^* search with euclidean distance metric to guide the search.



(a) (b) (c)

Fig. 7. Discrete Voronoi region shrinking for undersampling errors: A 3×3 pixel neighborhood of a discrete Voronoi diagram. The discrete MaNG is shown in thick orange lines. (a) The green discrete Voronoi region is disconnected. (b) The center pixel may be assigned to an adjacent Voronoi region reducing the complexity of the MaNG, without changing its connectivity. (c) Reassigning the center pixel will change connectivity of the MaNG.

We use a complete quadtree for Voronoi region culling, described in Section 6.1. The depth of the quadtree is set such that one leaf node corresponds to a block of 32×32 pixels. We need to determine if a node contains up to two sites—hence, the number of sites per node is encoded in 1 byte. By using a complete quadtree, the node addresses can be efficiently computed using bit shifts, avoiding pointer addressing.

7.2 Demos

We describe three multiagent simulations, demonstrating the effectiveness of the MaNG for real-time path planning. The first simulation involves a coverage problem, the second one is of a crowd simulation, and third is of terrain exploration.

Fruit stealing game. The first simulation is of fruit stealing in a dense orchard (see Fig. 1). There are several agents (thieves) that attempt to steal the fruit on the trees. The environment also contains some old farmers who chase the thieves. As the thieves move through the orchard, they steal fruits in close proximity. The goal is for each thief to move toward denser regions of fruits while avoiding the farmers, the trees, and other thieves. The thieves, farmers, and trees are treated as cylindrical sites. The trees are fixed obstacles, the farmers are dynamic obstacles, and the thieves are the agents. A coarse density map is used to track the density of fruits remaining in the orchard. Trees with desirable fruits are assigned a higher density. The agents are initially spread near the boundary of the orchard, and the goal position is set to a distant high-density region. The goal position for each agent is also dynamically updated as the density of the current goal drops below a certain threshold.

The global path of each agent is computed using the approach presented in Algorithm 1. We compute the proximity to the nearest site for each agent from the second-nearest neighbor diagram, which is used in a potential planner for local planning. Finally, we also use the second-order Voronoi diagram to compute the closest agent (thief) for each farmer. This agent is set as the goal for each farmer, and the farmer moves directly toward it. The farmers do not use the MaNG for path planning; however, they use the potential and repulsive forces to stay

Fig. 8. Crowd simulation. Three scenes of a crowd simulation with agents moving between buildings and the sidewalks. The cars represent dynamic obstacles. Our MaNG-based algorithm can perform navigation on 200 agents, each with distinct goals, at 5 fps.

clear of other farmers and trees. A thief is eliminated if caught by a farmer. Hence, it is desirable for each thief to compute the shortest paths of maximal clearance from the farmers (dynamic obstacles) and other thieves (agents) in order to collect the most fruits.

Crowd simulation. We simulate a crowd of people moving in an urban environment with dynamic obstacles (Fig. 8). We simulate only the individual behavior and not the group behavior. The set of sites consists of buildings, cars, and humans. The humans enter the scene from one of the buildings and exit through another building or the sidewalks. Each human is an individual agent with an independent goal. The cars are dynamic obstacles, while the buildings, benches, and fountains are static obstacles. Similar to fruit picking, the proximity information for local planning is computed using the second-order Voronoi diagram. The dynamics of each agent is computed using the local dynamics model presented in Section 5. In combination with the local dynamics model, our navigation system provides smooth natural motions for each agent (see Fig. 6). For goals in the same Voronoi region as the agent or in the adjacent Voronoi region, the shortest path to the goal is used, disregarding the MaNG.

Terrain exploration. The third simulation is one for robot rovers exploring a terrain (see Fig. 9). The environment consists of six rocks and one crater. The rocks and crater are static nonconvex polygonal obstacles. Each rover is an individual agent with an independent goal and motion characteristics. Certain rovers are more "aggressive" and have a higher maximum velocity. The goal for each rover is randomly assigned in the open terrain. Once a rover reaches its goal, it is assigned a new goal. The global path information computed using the MaNG enables the rovers to avoid a local minimum and move around nonconvex obstacles. The dynamics of each agent is computed using the local dynamics model.

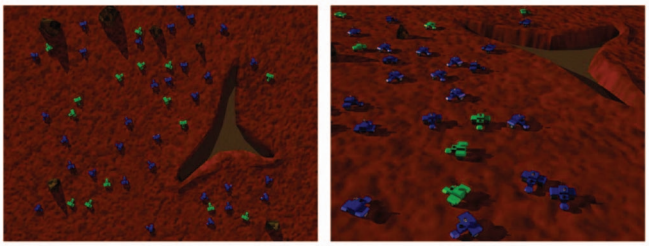


Fig. 9. Terrain exploration. Two scenes of a terrain exploration simulation with agents moving around rocks and craters. The green rovers have a higher maximum velocity and exhibit aggressive motion. Our MaNG-based algorithm can perform navigation on 50 rovers, each with distinct goals, at 13 fps.

7.3 Results

We now highlight the performance of our algorithm in dynamic virtual environments. Our approach can perform real-time path planning for each agent in environments of up to 200 virtual agents with different destinations, at the rates of 5 to 20 fps. The discrete Voronoi diagrams are computed on a grid of resolution $1{,}000 \times 1{,}000$ pixels. The fruit stealing simulation has 64 trees, with a varying number of thieves and farmers. The crowd simulation has 15 static obstacles and between two and five moving cars, with a varying number of humans. The performance of our approach, with a timing breakup, is presented in Table 1.

8 Analysis and Comparison

In this section, we analyze the performance of our algorithm. We highlight its computational complexity and compare it with other approaches for multiagent path planning.

TABLE 1
Performance of the Multiagent Path Planning Algorithm (Average over All Frames)

Demo	Agents	Graph		Time(ms)				
		|V|	|E|	DVD	MaNG	Plan	LD	Total
Crowd	10	206	1051	7	20	0.23	0.18	52
Crowd	25	330	1949	9	22	0.8	0.45	58
Crowd	50	560	3500	10	36	2.0	0.95	73
Crowd	100	946	7058	15	65	5.6	2.23	112
Crowd	200	1927	14669	20	150	18	5.3	217
Fruit	10	565	2282	8	25	1.0	0.21	59
Fruit	100	1378	6099	15	70	20	2.4	133
Mars	10	285	1015	9	21	0.23	0.19	55
Mars	50	790	3580	12	36	2.8	0.98	77

$|V|$ and $|E|$ denote the number of vertices and edges in the MaNG. DVD = time to compute the second-order discrete Voronoi diagram on the GPU and remove undersampled regions. MaNG = time to extract the MaNG from the discrete Voronoi diagram. LD = time for computing the local dynamics of all agents. Plan = time for path planning for all agents. Time for readback of discrete Voronoi diagram and depth buffers at $1{,}000 \times 1{,}000$ resolution $= 25$ ms.

8.1 Analysis

Let the number of sites be n and the size of the grid used to compute the discrete Voronoi diagrams be $m \times m$. We assume that the number of agents $|\mathrm{P}_a| = O(n)$. We now present the time complexity of each stage in our algorithm.

The cost of computing the first- and second-order discrete Voronoi diagrams is given as follows: The size of the quadtree is $O((\frac{m}{32})^2)$, and $\mathrm{depth} = O(\log m)$. Then, the cost of computing the bounds for each site (see Section 6.1) is $O(\log m)$. The cost of rasterizing the distance function for a site p_i is $O(r|\mathrm{Vor}^k(p_i|\mathrm{P})|)$, where $|\mathrm{Vor}^k(p_i|\mathrm{P})|$ is the number of pixels in the Voronoi region of p_i, and r depends on the tightness of the computed Voronoi region bounds, $1 < r < O(n)$. Typically, we have observed that $r = O(1)$. Then, the cost of computing the Voronoi diagram is $O(n \log m + \Sigma_{i=1}^{n}(r|\mathrm{Vor}^k(p_i|\mathrm{P})|)) = O(rm^2 + n \log m)$.

The cost of reading back the frame buffers is $O(m^2)$. The cost of extracting the MaNG is $O(|\mathrm{E}|)$, where $|\mathrm{E}|$ is the number of edges in MaNG. From Lemma 2, the number of edges in MaNG, $|\mathrm{E}| \leq |\mathrm{E}^1| + |\mathrm{E}^2|$, where $|\mathrm{E}^k|$ is the number of edges of $\mathrm{VD}^k(\mathrm{P})$, and $|\mathrm{E}^k| = O(kn)$ [10]. Thus, cost of extracting the MaNG is $O(n)$. The cost of path planning using A^* is typically polynomial in $O(|\mathrm{E}| + |\mathrm{V}|)$. Therefore, the cost of computing all paths is $O(n(|\mathrm{E}| + |\mathrm{V}|)) = O(n^2)$. In practice, as shown in Table 1, the associated constant with path planning is much smaller, and the bottleneck is the discrete Voronoi diagram computation and graph construction.

8.2 Comparisons

Next, we provide qualitative comparisons of our approach with prior methods for multiagent planning.

Comparison with the first-order Voronoi diagram. Our approach provides a global solution for the path planning of each agent using the MaNG. The MaNG computes a road map of maximal-clearance collision-free paths for each agent in $O(1)$ passes, as compared to $O(n)$ passes for computing $O(n)$ Voronoi road maps. In particular, using the second-order Voronoi graph for path planning guarantees that the position selected as the first intermediate goal along the computed path is unique. This approach prevents adjacent agents from moving toward the same intermediate goal and getting stuck in a local minimum of the potential function. An example is presented in Fig. 5. In this example, adjacent agents select the same intermediate goal from the first-order Voronoi diagram, whereas the intermediate goals from the second-order Voronoi diagram are unique. In addition, the path computed has maximal clearance. More specifically, vertices on the Voronoi diagram are used to compute the area of maximum coverage for a new site [1]. Hence, by following the vertices on the MaNG, our planning approach ensures a maximum coverage region for each agent.

The closest related work of Pettre et al. [30] computes an initial road map of a static environment using Voronoi diagrams and constructs a set of homotopic paths for a group of agents. This work implicitly groups agents by their origins and goals. Furthermore, local collision avoidance is not guaranteed. In contrast, our algorithm is able to handle dynamic environments, as the road map is updated in real time, and the use of the second-order Voronoi diagrams provides pairwise proximity information, which is used to guarantee collision avoidance.

The work on continuum crowds [42] computes a dynamic potential field and updates the position of each agent by moving along the gradient of the potential function. The potential field is computed for a small number of groups of agents moving with common goals. However, due to the use of a potential function the agents may get stuck in a local minimum. In contrast, our approach allows for an independent goal for each agent.

In comparison to agent-based methods, our MaNG-based path planning algorithm provides global paths and may be combined with rule-based techniques to simulate more complex and realistic agent behavior.

8.3 Limitations

There are some limitations of our work. We compute the MaNG in the workspace; hence, the approach does not scale well for agents with many degrees of freedom (for example, snakes). We use an A^* graph search algorithm, which may not be optimal. Finally, we compute an optimal path for each time step; however, there is no guarantee on coherence of paths across frames or on convergence over a period of time. In fact, the optimal paths across two time steps may not be coherent (that is, the immediate goal may change considerably), potentially resulting in noisy motions.

9 Conclusions and Future Work

We have presented a novel approach for real-time path planning of multiple virtual agents, based on a new data structure—the MaNG. The MaNG is used to simultaneously compute the paths of maximal clearance for a set of moving agents with independent goals. The MaNG is constructed dynamically using discrete Voronoi diagrams. We also presented culling techniques for accelerating the discrete Voronoi diagram computation and addressed undersampling issues due to discretization. We have demonstrated the application of our approach to real-time simulation involving a large number of independent agents, each with an individual goal.

There are several avenues for future work. One relevant avenue is to constrain the graph search to compute temporally coherent paths that are guaranteed to converge to the final goal. We would like to exploit coherence in graph search when many agents have similar goals and initial positions. Efficient parallel algorithms for simplifying the discrete Voronoi graphs and computing the MaNG would be useful for accelerating the computation. Finally, we would like to extend our approach to handle agents with high degrees of freedom.

References

[1] F. Aurenhammer, "Voronoi Diagrams: A Survey of a Fundamental Geometric Data Structure," *ACM Computing Surveys,* vol. 23, no. 3, pp. 345-405, Sept. 1991.

[2] D. Baraff and A. Witkin, "Physically Based Modeling," *ACM SIGGRAPH Course Notes,* 2001.

[3] O.B. Bayazit, J.-M. Lien, and N.M. Amato, "Better Group Behaviors in Complex Environments with Global Roadmaps," *Proc. Eighth Int'l Conf. Simulation and Synthesis of Living Systems (Alife '02),* pp. 362-370, 2002.

[4] M. Bennewitz, W. Burgard, and S. Thrun, "Finding Solvable Priority Schemes for Decoupled Path Planning Techniques for Teams of Mobile Robots," *Robotics and Autonomous Systems,* vol. 41, nos. 2-3, pp. 89-99, 2002.

[5] J. Champagne and W. Tang, "Real-Time Simulation of Crowds Using Voronoi Diagrams," *EG UK Theory and Practice of Computer Graphics,* pp. 195-201, 2005.

[6] H. Choset and J. Burdick, "Sensor Based Motion Planning: The Hierarchical Generalized Voronoi Graph," *Algorithms for Robot Motion and Manipulation.* A K Peters, pp. 47-61, 1996.

[7] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations.* MIT Press, 2005.

[8] O.C. Cordeiro, A. Braun, C.B. Silveria, S.R. Musse, and G.G. Cavalheiro, "Concurrency on Social Forces Simulation Model," *Proc. First Int'l Workshop Crowd Simulation,* 2005.

[9] M. Denny, "Solving Geometric Optimization Problems Using Graphics Hardware," *Proc. Eurographics '03,* pp. 441-451, 2003.

[10] I. Fischer and C. Gotsman, "Fast Approximation of High Order Voronoi Diagrams and Distance Transforms on the GPU," Technical Report CS TR-07-05, Harvard Univ., 2005.

[11] M. Foskey, M. Garber, M. Lin, and D. Manocha, "A Voronoi-Based Hybrid Planner," *Proc. IEEE/RSJ Int'l Conf. Intelligent Robots and Systems (IROS '01),* vol. 1, pp. 55-60, 2001.

[12] J. Funge, X. Tu, and D. Terzopoulos, "Cognitive Modeling: Knowledge, Reasoning and Planning for Intelligent Characters," *Proc. ACM SIGGRAPH '99,* pp. 29-38, 1999.

[13] P. Glardon, R. Boulic, and D. Thalmann, "Dynamic Obstacle Clearing for Real-Time Character Animation," *Computer Graphics Int'l,* vol. 22, no. 6, pp. 399-414, 2005.

[14] L. Guibas, C. Holleman, and L. Kavraki, "A Probabilistic Roadmap Planner for Flexible Objects with a Workspace Medial-Axis-Based Sampling Approach," *Proc. IEEE/RSJ Int'l Conf. Intelligent Robots and Systems (IROS '99),* pp. 254-259, 1999.

[15] D. Helbing, L. Buzna, A. Johansson, and T. Werner, "Self-Organized Pedestrian Crowd Dynamics: Experiments, Simulations and Design Solutions," *Transportation Science,* pp. 1-24, 2005.

[16] D. Helbing, L. Buzna, and T. Werner, "Self-Organized Pedestrian Crowd Dynamics and Design Solutions," *Traffic Forum 12,* 2003.

[17] K. Hoff, T. Culver, J. Keyser, M. Lin, and D. Manocha, "Fast Computation of Generalized Voronoi Diagrams Using Graphics Hardware," *Proc. ACM SIGGRAPH '99,* pp. 277-286, 1999.

[18] K. Hoff, T. Culver, J. Keyser, M. Lin, and D. Manocha, "Interactive Motion Planning Using Hardware Accelerated Computation of Generalized Voronoi Diagrams," *Proc. IEEE Int'l Conf. Robotics and Automation (ICRA '00),* pp. 2931-2937, 2000.

[19] K. Hoff, A. Zaferakis, M. Lin, and D. Manocha, "Fast and Simple 2D Geometric Proximity Queries Using Graphics Hardware," *Proc. ACM Symp. Interactive 3D Graphics,* pp. 145-148, 2001.

[20] S.P. Hoogendoorn, S. Luding, P. Bovy, M. Schrecklenberg, and D. Wolf, *Traffic and Granular Flow.* Springer, 2000.

[21] F. Lamarche and S. Donikian, "Crowd of Virtual Humans: A New Approach for Real-Time Navigation in Complex and Structured Environments," *Computer Graphics Forum,* vol. 23, no. 3, pp. 509-518, 2004.

[22] J.-C. Latombe, *Robot Motion Planning.* Kluwer Academic Publishers, 1991.

[23] T.-T. Li and H.-C. Chou, "Motion Planning for a Crowd of Robots," *Proc. IEEE Int'l Conf. Robotics and Automation (ICRA '03),* vol. 3, pp. 4215-4221, 2003.

[24] C. Loscos, D. Marchal, and A. Meyer, "Intuitive Crowd Behaviour in Dense Urban Environments Using Local Laws," *Proc. Theory and Practice of Computer Graphics (TPCG '03),* pp. 122-129, 2003.

[25] *MASSIVE,* http://www.massivesoftware.com, 2006.

[26] S.R. Musse and D. Thalmann, "A Model of Human Crowd Behavior: Group Inter-Relationship and Collision Detection Analysis," *Computer Animation and Simulation,* pp. 39-51, 1997.

[27] A. Okabe, B. Boots, and K. Sugihara, *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams.* John Wiley & Sons, 1992.

[28] L.E. Parker, "Designing Control Laws for Cooperative Agent Teams," *Proc. IEEE Int'l Conf. Robotics and Automation (ICRA '93),* pp. 582-587, 1993.

[29] N. Pelechano, K. O'Brien, B. Silverman, and N. Badler, "Crowd Simulation Incorporating Agent Psychological Models, Roles and Communication," *Proc. First Int'l Workshop Crowd Simulation,* 2005.

[30] J. Pettre, J.-P. Laumond, and D. Thalmann, "A Navigation Graph for Real-Time Crowd Animation on Multilayered and Uneven Terrain," *Proc. First Int'l Workshop Crowd Simulation,* 2005.

[31] C.W. Reynolds, "Flocks, Herds, and Schools: A Distributed Behavioral Model," *Computer Graphics (Proc. ACM SIGGRAPH '87),* vol. 21, pp. 25-34, 1987.

[32] M. Schreckkenberg and S.D. Sharma, *Pedestrian and Evacuation Dynamics.* Springer, 2001.

[33] G. Still, "Crowd Dynamics," PhD dissertation, Univ. of Warwick, 2000.

[34] A. Sud, E. Andersen, S. Curtis, M. Lin, and D. Manocha, "Real-Time Path Planning for Virtual Agents in Dynamic Environments," *Proc. IEEE Virtual Reality Conf. (VR '07),* pp. 91-98, 2007.

[35] A. Sud, N. Govindaraju, R. Gayle, I. Kabul, and D. Manocha, "Fast Proximity Computation among Deformable Models Using Discrete Voronoi Diagrams," *ACM Trans. Graphics (Proc. ACM SIGGRAPH '06),* vol. 25, no. 3, pp. 1144-1153, 2006.

[36] A. Sud, N. Govindaraju, R. Gayle, and D. Manocha, "Interactive 3D Distance Field Computation Using Linear Factorization," *Proc. ACM Symp. Interactive 3D Graphics and Games,* pp. 117-124, 2006.

[37] A. Sud, M.A. Otaduy, and D. Manocha, "DiFi: Fast 3D Distance Field Computation Using Graphics Hardware," *Computer Graphics Forum (Proc. Eurographics '04),* vol. 23, no. 3, pp. 557-566, 2004.

[38] M. Sung, M. Gleicher, and S. Chenney, "Scalable Behaviors for Crowd Simulation," *Computer Graphics Forum,* vol. 23, no. 3, pp. 519-528, Sept. 2004.

[39] M. Sung, L. Kovar, and M. Gleicher, "Fast and Accurate Goal-Directed Motion Synthesis for Crowds," *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation (SCA '05),* pp. 291-300, 2005.

[40] D. Thalmann, C. O'Sullivan, P. Ciechomski, and S. Dobbyn, "Populating Virtual Environments with Crowds," *Eurographics '06 Tutorial Notes,* 2006.

[41] A. Treuille, S. Cooper, and Z. Popovic, "Continuum Crowds," *Proc. ACM SIGGRAPH '06,* pp. 1160-1168, 2006.

[42] X. Tu and D. Terzopoulos, "Artificial Fishes: Physics, Locomotion, Perception, Behavior," *Proc. ACM SIGGRAPH '94,* pp. 43-50, 1994.

[43] J. Vleugels and M.H. Overmars, "Approximating Voronoi Diagrams of Convex Sites in Any Dimension," *Int'l J. Computational Geometry and Applications,* vol. 8, pp. 201-222, 1998.

[44] S.A. Wilmarth, N.M. Amato, and P.F. Stiller, "MAPRM: A Probabilistic Roadmap Planner with Sampling on the Medial Axis of the Free Space," *Proc. IEEE Int'l Conf. Robotics and Automation (ICRA '99),* pp. 1024-1031, 1999.

[45] G.K. Zipf, *Human Behavior and the Principle of Least Effort.* Addison-Wesley, 1949.

Avneesh Sud received the BTech degree in computer science and engineering from the Indian Institute of Technology, Delhi, in 2000 and the master's and PhD degrees in computer science from the University of North Carolina, Chapel Hill, in 2003 and 2006. He is currently a program manager in the Many-Core Technology Incubation Group, Microsoft Corp. Before joining Microsoft, he was a postdoctoral researcher in the GAMMA research group in the Department of Computer Science, University of North Carolina, Chapel Hill. His research interests include effective utilization of parallel hardware to solve problems in computer graphics, computational geometry, geometry processing, solid modeling, physically based modeling, and robotics. He received the ACM VRST best paper award in 2007. He has published more than 20 peer-reviewed articles in major graphics and virtual reality journals and conferences such as ACM SIGGRAPH, Eurographics, and IEEE VR. He has also served on the program committee of several conferences and workshops.

Erik Andersen received the BS degree in computer science from the University of North Carolina, Chapel Hill, in 2007. He is currently a first-year graduate student at the University of Washington, where he is also a United States National Science Foundation graduate fellow. His research interests include animation and behavioral modeling.

Sean Curtis received the BA degree in German from Brigham Young University and the BS degree in computer science from the University of Utah. He is currently a PhD student in computer science in the Department of Computer Science, University of North Carolina, Chapel Hill, where he is pursuing research in graphics and simulation.

Ming C. Lin received the PhD degree in electrical engineering and computer science from the University of California, Berkeley. She is currently the Beverly W. Long distinguished professor of computer science in the Department of Computer Science, University of North Carolina (UNC), Chapel Hill. Her research interests include physically based modeling, haptics, real-time 3D graphics for virtual environments, robotics, and geometric computing. She has (co)authored more than 170 refereed publications and coedited/authored three books: *Applied Computation Geometry*, published by Springer, *High-Fidelity Haptic Rendering*, published by Morgan-Claypool, and *Haptic Rendering: Foundations, Algorithms, and Applications*, published by A K Peters. She has served on more than 70 program committees of leading conferences on virtual reality, computer graphics, robotics, haptics, and computational geometry, and cochaired more than 15 international conferences and workshops. She is the associate editor in chief of *IEEE Transactions on Visualization and Computer Graphics*, a member of four editorial boards, and a guest editor for more than 10 special issues of scientific journals and technical magazines. She has also served on four steering committees and advisory boards of international conferences, as well as six technical advisory committees constituted by government organizations and the industry. She has received several honors and awards, including the NSF Young Faculty Career Award in 1995, the Honda Research Initiation Award in 1997, the UNC/IBM Junior Faculty Development Award in 1999, the UNC Hettleman Award for Scholarly Achievements in 2003, and six best paper awards at international conferences on computer graphics and virtual reality. She is a member of the IEEE and the IEEE Computer Society.

Dinesh Manocha received the BTech degree in computer science and engineering from the Indian Institute of Technology, Delhi, in 1987, and the MS and PhD degrees in computer science at the University of California, Berkeley, in 1990 and 1992, respectively. He is currently the Phi Delta Theta/Matthew Mason distinguished professor of computer science in the Department of Computer Science, University of North Carolina, Chapel Hill (UNC Chapel Hill). His research interests include geometric and solid modeling, interactive computer graphics, physically based modeling, virtual environments, robotics, and scientific computation. His research has been sponsored by ARO, DARPA, DOE, Honda, Intel, NSF, the Office of Naval Research (ONR), and the Sloan Foundation. He has published more than 230 papers in leading conferences and journals on computer graphics, geometric and solid modeling, robotics, symbolic and numeric computation, virtual reality, and computational geometry. He has served as a program committee member and chaired for many leading conferences on computer graphics, geometric and solid modeling, virtual reality, animation, and robotics. He has also served as a member of the editorial board and a guest coeditor for leading journals in computer graphics, computer-aided design, applicable algebra, computational geometry, and parallel computing. He was selected as an Alfred P. Sloan research fellow and received the NSF Career Award in 1995, the ONR Young Investigator Award in 1996, the Honda Research Initiation Award in 1997, and the Hettleman Prize for scholarly achievement at UNC Chapel Hill in 1998. He has also received multiple best paper awards at the ACM SuperComputing, ACM Multimedia, ACM Solid Modeling, ACM VRST, IEEE Visualization, IEEE VR, and Eurographics Conferences.