Volume 0 (1981), Number 0 pp. 1–12 COMPUTER GRAPHICS *forum*

Inherent Noise-Aware Insect Swarm Simulation

Xinjie Wang[1] , Xiaogang Jin[1] , Zhigang Deng[2] and Linling Zhou[1]

[1]State Key Lab of CAD&CG, Zhejiang University, China
wangxinjie@zjucadcg.cn, jin@cad.zju.edu.cn, zhoulinling@zjucadcg.cn
[2]Computer Science Department, University of Houston, USA
zdeng@cs.uh.edu

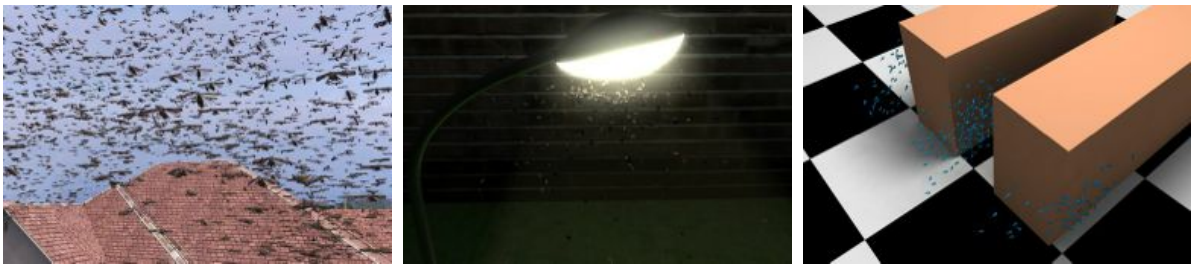


Figure 1: *A variety of insect swarms can be simulated by our approach: (left) a large swarm of migratory locusts invades a hamlet, (middle) a swarm of moths flies around a street lamp, and (right) swarm-like particles fly around two blocks.*

Abstract

Collective behavior of winged insects is a wondrous and familiar phenomenon in the real world. In this paper, we introduce a highly efficient field-based approach to simulate various insect swarms. Its core idea is to construct a smooth yet noise-aware governing velocity field that can be further decomposed into two sub-fields: (i) a divergence-free curl noise field to model noise-induced movements of individual insects in a swarm, and (ii) an enhanced global velocity field to control navigational paths in a complex environment along which all the insects in a swarm fly. Through simulation experiments and comparisons with existing crowd simulation approaches, we demonstrate that our approach is effective to simulate various insect swarm behaviors including aggregation, positive phototaxis, sedation, mass-migrating, and so on. Besides its high efficiency, our approach is very friendly to parallel implementation on GPUs (e.g., the speed-up achieved through GPU acceleration is higher than 50 if the number of simulated insects is more than ten thousands on an off-the-shelf computer). Our approach is the first multi-agent modeling system that introduces curl-noise into agents' velocity field and uses its non-scattering nature to maintain non-colliding movements in 3D crowd simulation.

Keywords: insect swarms, multi-agent systems, crowd simulation, curl-noise, potential field, path planning

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Virtual Reality I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

1. Introduction

Insects are among the most diverse groups of animals in the natural world, with more than a million documented species. Since insects are an essential part of our real world, how to effectively simulate a variety of dynamic and realistic behaviors of insects has become an increasingly important research problem for computer graphics, animation, and virtual reality applications. In fact, besides its graphics and animation applications, accurate computer simulation

of various insect behaviors can find its prominent use in many scientific research fields including but not limited to behavioral biology, zoology, bionics, multi-agent control, and robotics. Among various insect behaviors, insect swarm is considered as one of widely observed biological motions. Generally, an insect swarm is referred to a group of individual insects that exhibit inherently noisy, non-colliding and aggregate motion [BSC*06]. The term *inherent noise* in zoology field is referred to all the random movements made by individuals in an insect group.

Over the years, research advances in the bionics field have yielded plenty of experimental evidence and insights how insect swarms are dynamically formed and evolved [Mor63, RMC95, RMD*03]. For example, researchers revealed that noise-induced movement and increasing the randomness of the movements in response to a loss of group alignment are intrinsic characteristics of certain insect swarms [BSC*06, YEE*09]. These findings imply, besides sharing group-level properties like other animal groups, individual insects in a swarm often make sudden changes in direction spontaneously. On the other hand, existing biological experimental findings (e.g., [STD74]) validated that the compound eye structure of insects makes them react fast to the things surrounded in order to avoid predators or hit obstacles in the environment. This implies that regardless level of the random movements of an insect, it can fly as safely as possible in a dense swarm or in a complex environment. As understanding insect swarms becomes more and more prominent in many scientific fields, developing effective algorithms to simulate realistic large-scale insect swarms is clearly a priority. Indeed, no sound computer modeling approaches for such simulations have been developed and validated to date.

Likewise, existing crowd simulation techniques often fall short of simulating insect swarms effectively. The above characteristics of insect swarms can be summarized as follows:

Noise-induced movements: Since humans and large animals typically move in a trajectory of gradual changes, most of existing crowd techniques focus on generating smooth paths or tend to add small variations [Rey87] to increase naturalness. But in the case of simulation of insect swarms, their paths are dominated by noise. Each insect in a swarm can move somewhat freely. Simple noise fails to create realistic swarms, which implies an appropriate noise function needs to be delicately designed for such simulations.

Non-colliding rapid movements: It is difficult to simulate large-scale random movements while avoiding collisions in previous frameworks. For example, rule-based crowd simulation approaches [Rey87, Rey99], social force models [HM95, PAB07], and continuum-based approaches [TCP06] need to advance for several time steps to find out non-collide velocities/positions. Since rapid movements are difficult to predict in advance, simply adding rapid random movements to these works may fail to avoid most collisions. Meanwhile, without considerable efforts, it is extremely difficult to directly apply example-based (or data-driven) approaches [LCHL07] for insect swarm simulation due to the technical difficulty of accurately tracking rapid moving individual insects in a swarm within various natural environments.

Inspired by the above challenges, in this paper we present a highly efficient field-based approach to realistically simulate behavior patterns of insect swarms under various scenarios. Specifically, we formulate this simulation problem as a mesoscopic multi-agent system that models both macroscopic goal-attained path planning and microscopic behaviors (e.g., individual space, random switching and local collision avoidance). Through simulation experiments and comparisons with existing crowd simulation approaches, we demonstrate that our inherent noise-aware insect swarm simulation approach can be effectively used to simulate various realistic behaviors of insect swarms including aggregation, positive phototaxis, sedation, mass-migrating, and so on.

The contributions of this work are: (i) a simple divergence-free noise field to govern a series of noise-aware yet non-colliding motions, which is highly similar to rapid transitions that result from disordered yet noise-induced movements of individual insects in a swarm, and (ii) an enhanced global velocity field for parallel implementation to control navigational paths especially in a complex environment along which all the insects in a swarm fly.

To the best of our knowledge, our approach is the first multi-agent modeling system that introduces curl-noise into agents' velocity field and uses its non-scattering nature of the resulting field to maintain noise-aware yet non-colliding movements in 3D crowd simulation. The research described in this paper represents some promising first steps toward realistic simulation of various 3D crowds in virtual worlds based on the collective knowledge developed in behavioral biology and other scientific fields.

2. Related Work

A variety of approaches for modeling motion of multiple agents in a crowd have been proposed during the past decades. They can be roughly classified into three categories, broadly termed *microscopic*, *macroscopic* and *mesoscopic* [ZMR*08]. Microscopic models regard agents as discrete individuals; macroscopic models handle a crowd as a whole; mesoscopic models combine features of the first two.

Among the above three types of models, probably microscopic models have been most studied to date. Following the seminal *Boids* model by Reynold [Rey87], various rule-based approaches have been extended [SJ12] and developed by adding behavior-based multi-level group rules [RMT01],

perceptual knowledge of pedestrians [ST05], cognitive modeling [FTT99], or egocentric affordance fields [KSHF09]. Helbing and Molnar [HM95] pioneered the social force model for simulating pedestrian motion such as repulsive interaction, friction forces, dissipation, and fluctuations. Later, it was further applied and generalized to other simulation scenarios such as densely populated crowds [PAB07].

Macroscopic or continuous models are often used when the task is to compute a crowd's path towards its goal such as a navigation graph [PvdBC*11]. Flow tiles and potential field-based models regard the environment as a regular discrete grid, then gradient methods can be applied to find a path from any start position to a destination in an environment [NGCL09]. However, this approach creates incompressible, dense and flow-like crowds. When noise is blended into a compressive flow, the noise component will always be lost during the solving of the incompressible flow (i.e., the UIC projecting process). In addition, interactive path planning and navigation algorithms for multi-agent systems in complex dynamic environments were also proposed [SAC*08, KOOP11, JXW*08].

Mesoscopic models can govern local behaviors of agents while navigating a goal-directed path of the crowd. Researchers have introduced continuum theory [TCP06, SWL11], information theory [GvdBL*12], and synthetic vision [OPOD10] into crowd simulation domain. Goldenstein et al. [GKM*01a, GKM*01b] employ a three-layer hybrid system for agent steering and crowd simulation. Narain and colleagues [NGCL09] further introduced unilateral incompressibility constraints to simulate extremely dense crowds. However, most of these mesoscopic models cannot be directly applied to simulate noise-aware position changes that result from spontaneous direction switching within an insect swarm, due to their strict requirement of local collision avoidances. Note that the method described in this paper can be classified as a mesoscopic model as well.

Example-based (or data-driven) approaches learn crowd behavior features from real-world examples [LCHL07, CSJ13], and then these features can be used to train statistical models for crowd simulation. Researchers also explored novel ways to animate large-scale crowds by cloning crowd motions from existing examples [LCS*12], blending crowd animations [JCP*10], and diversifying the perceived realism of agent motions in a context-aware way [GD11a]. Recently, significant efforts have been attracted to generate group formations in simulated crowds, including: combining heuristic rules with explicit hard constraints to produce and control sophisticated group formations [TYK*09] or introducing a novel sketching interface to specify and control various group formations [GD11b, GD13]. Besides, significant interests have been drawn to study real collective animal groups [BJCC*97, SASBJ11] or learn complex patterns in specific swarms [TB04, BKBS13].

3. Approach Overview

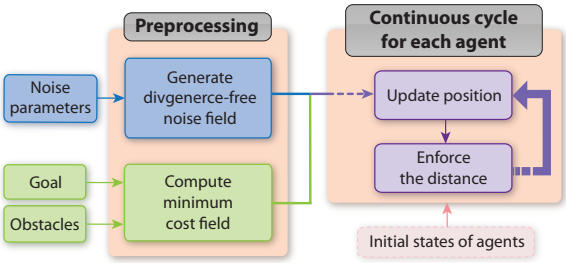


Figure 2: *Schematic view of our introduced inherent noise-aware insect swarm simulation approach.*

As illustrated in Figure 2, our approach can be conceptually regarded as a two-step computing process. At the preprocessing step, we generate a noise field $\vec{N}$ that stores a 3D procedural noise function using the well-known Perlin noise algorithm [Per02], and a minimum cost field C that stores a scalar path-planning cost function, computed based on a given environment with obstacles and specified goal positions. Then, at the simulation step, we continuously update the positions and orientations of all the agents (insects) in a swarm based on the velocities provided via $\vec{N}$ and C, while ensuring collision avoidances with other agents and obstacles in the environment.

Our approach employs potential fields to produce continuous velocity fields spanning free space in the environment. The processing and computation of potential fields requires 3D discretization of the free space in the environment. For simplicity, we use regular grids with values stored in cells for each potential field and assume that the cells form a 6-connected component. It is noteworthy that the two potential fields can be defined with respect to grids with different resolutions; typically, the noise field needs a higher resolution grid for the sake of simulating high-fidelity noise motion.

4. Computing Noise Field

As documented in the literature, insects' random movements exhibit the same patterns with both large and small variations [BHR10]; on the other hand, collision avoidance is one of intrinsic flight behaviors for insects in a swarm [YEE*09]. Thus, the well-known Perlin noise algorithm [Per02] is a suitable scheme for modeling such continuous movements.

Specifically, we first define a randomly varying vector constructed from the Perlin noise as follows:

$$\vec{P} = \left(P_1\left(\frac{\vec{x}}{scale}\right), P_2\left(\frac{\vec{x}}{scale}\right), P_3\left(\frac{\vec{x}}{scale}\right) \right) * gain,$$

where $\vec{x}(x,y,z)$ is a point in 3D space, and P_1, P_2, P_3 are three Perlin noise functions with different random seeds to generate scalar noise values in the range of [-1.0, 1.0]

(defined with respect to a unit of grid), respectively. We use two noise parameters, *scale* (for controlling the grid density indirectly) and *gain* (for adjusting the magnitude of the Perlin noise).

Then, based on the obtained $\vec{P}$, we further construct a divergence-free noise field $\vec{V}_l$, inspired by the work of Bridson et al. [BHN07] that introduced an incompressible flow field modulated by a curl operator $\nabla\times$:

$$\begin{aligned}\vec{V}_l &= \nabla\times(a_1\psi_1 + a_2\psi_2 + \ldots + a_n\psi_n)\\ &= \nabla\times\vec{N}.\end{aligned} \tag{1}$$

Here the noise field $\vec{N}$ can be formed through the addition of a number of potential fields, $\psi_{i=1}^{n}$. Since the divergence of the curl of any vector field is always zero: $\nabla\cdot\nabla\times\equiv 0$, the velocity field $\vec{V}_l$ is divergence-free (i.e., $\nabla\cdot\vec{V}_l = 0$) [BHN07]. The divergence-free noise field can be treated as a collision-free field in a multi-agent model since any two particles in this incompressible field will never intersect. In this work, we treat insects as discrete particles and drive them using $\vec{V}_l$ to obtain collision-free trajectories.

The simplest way to construct $\vec{N}$ would be to define $\vec{N} = \vec{P}$. However, $\vec{P}$ is too random to express a non-colliding velocity field. In fact, Bridson et al. [BHN07] introduced a number of formulas to construct $\vec{N}$ and provided several examples. One of them is well suited to our purpose: Consider there are some static boundaries in the environment, and insects are not allowed to go through the boundaries (that is, the insects' velocities at the boundaries must satisfy the following condition: $\vec{V}_l\cdot\vec{n} = 0$). Its specific construction formula is described as follows:

$$\vec{N} = \alpha\vec{P} + (1-\alpha)(\vec{n}\cdot\vec{P})\vec{n}, \tag{2}$$

where $\vec{n}$ is the normal to the boundary surface, and α=$\alpha(d(\vec{x})/d_0)$ $(0 \le \alpha \le 1)$ is a smooth ramp function. In this ramp function, $d(\vec{x})$ is the shortest distance from a point $\vec{x}(x,y,z)$ to the boundaries, and d_0 is an adjustment factor. In our case, d_0 is set to be equivalent to the side length of the grid. The closer $\vec{x}$ is to the boundaries, the larger value α has. The boundaries can be used not only to keep insects out of obstacles (accurate boundaries) but also to avoid them flying too far away (extended boundaries that are much larger than the swarm).

Eq. 2 implies that the tangential component of $\vec{P}$ is reduced near the boundaries, but the normal component is retained. In this way, the curl of $\vec{N}$ will remove the velocity in the normal direction gradually as approaching to the boundary surface. In this work, a suitable ramp function α is defined as follows (similar to [BHN07]):

$$\alpha(r) = \begin{cases} \frac{3}{8}r^5 - \frac{10}{8}r^3 + \frac{15}{8}r & r \le 1;\\ 1 & \text{otherwise.}\end{cases} \tag{3}$$

In Eq. 3, r implies $d(\vec{x})/d_0$ and should never be negative.

We employ the above $\vec{V}_l$ (Eq. 1) to simulate local behaviors of individual insects in a swarm. Figure 3 shows a

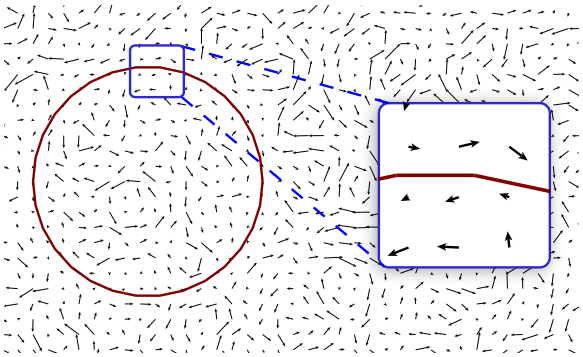


Figure 3: *An example of $\vec{V}_l$ showing how the ramp function removes the velocity component in the normal direction.*

velocity field generated from Eqs. 1 and 2. In addition, we can also use dynamic P that varies over time to make insects who pass a specific spot have different velocities. But it will take extra computational time when updating every frame. In practice, we found it unnoticeable that insects pass one spot are doing the same thing. So a fixed P is enough for our algorithm. From our experiments, we found this scheme is suitable for modeling noise behaviors of insect swarms, compared with observed real-world insect swarms.

5. Computing Minimum Cost Field

Besides noisy local behaviors of individual insects, we also need to script a swarm's path by building migratory urge into our model. Not only this is a common phenomenon existing in real-world insect swarms (e.g., migration, foraging, or attacking), it also allows users to conveniently control goal-directed swarms in simulation.

Given an environment description with specified goal areas and obstacles, computing a goal-directed velocity field means that there should be at least one valid path from each cell in free space to the goal area. For simplicity, we assume that each insect has knowledge of the whole environment, and it typically selects an as-short-as-possible path. Therefore, we choose to use a variant of the Dijkstra's shortest path algorithm to construct a minimum cost field C.

In this work, the total cost of a path from a cell to the goal area is determined by two factors: the path's total *length* and the total *value of repulsion* along the path. A position closer to obstacles typically has a higher value of repulsion, and vice versa. In this way, finding an optimal path is equivalent to minimize the following cost function:

$$Cost = A\underbrace{\int_P 1ds}_{pathlength} + B\underbrace{\int_P gds}_{repulsion}, \tag{4}$$

where A and B are weights of individual terms, g stands for the value of repulsion, and ds denotes that this integral is

taken with respect to path length. In the above equation, the higher the B value is (i.e., the lower the A value is), the more dangerous the obstacles are. In our simulation experiments, we empirically set the values of A and B to 0.6 and 0.4, respectively. In follow-up Section 5.1, we will describe how the obstacles are expressed as a grid of repulsion, and describe how to solve the cost field C in Section 5.2.

5.1. Space Analysis

In an exterior scene, typically there are buildings, facilities, and other landscapes. They are called *obstacles* in a path finding algorithm. When insects approach the boundaries of the obstacles (termed *the area of repulsion*), they need to adjust their speeds and orientations to avoid collisions. Each cell in the area of repulsion is assigned with a value of repulsion. In this way, a swarm can find a smooth and safe path to pass by the obstacles. Without loss of generality, in this writing we assume the obstacles are represented as (or can be converted to) 3D triangular meshes.

The following two conditions can be used to determine whether a 3D grid cell is in the area of repulsion.

- The center of the grid cell lies inside one of the obstacles or just on the obstacles' surface;
- The center of the grid cell lies outside the obstacles, but its shortest distance to one of the obstacles is smaller than a user-specified threshold.

The shortest distance between a 3D grid cell X and the obstacles determines whether X is in the area of repulsion. In our approach, we employ a signed-distance computation algorithm proposed by Baerentzen and Aanaes [BA05] to compute such shortest distances due to its efficiency. Specifically, assuming $d_t(X)$ denotes the shortest distance from X to the obstacles, the signed distance $\phi(X)$ between X and its closest obstacle (mesh) is computed as follows:

$$\phi(X) = \begin{cases} -d_t(X) & \text{if } X \text{ is inside an obstacle;} \\ d_t(X) & \text{otherwise.} \end{cases} \tag{5}$$

To the end, the value of repulsion of the cell X, $g(X)$, can be computed by using a bell-shaped function from [WMW86] as follows:

$$g(X) = h(r) = \begin{cases} 1 & r < 0; \\ 1 - (\frac{4}{9})r^6 + (\frac{17}{9})r^4 - (\frac{22}{9})r^2 & 0 \le r \le 1; \\ 0 & r > 1. \end{cases} \tag{6}$$

In Eq. 6, $r = \frac{\phi(X)}{2a}$, and $a = \sqrt{2}l$ denotes the diagonal length of a regular grid (l is the side length). Figure 4(a) plots the repulsive function (Eq. 6) and illustrates an example of the computed signed distances of a discretized space.

5.2. Solving Minimum Cost Field

On top of the above obtained area of repulsion, we need to further define a minimum cost field C. We can rewrite the above cost function (Eq. 4) as the following discrete form:

$$\| \nabla C(X) \| = \triangle g(X) + l, \tag{7}$$

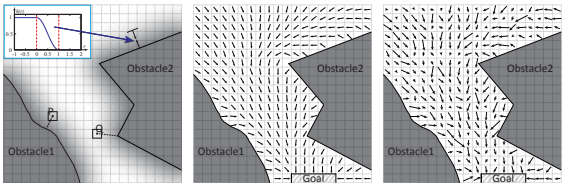
where $g(X)$ denotes the value of repulsion at cell X, and l is the known side length of a cell. Note that the above Eq. 7 is an eikonal equation; its solved optimized paths exactly follow the gradient of C, which gives us a global velocity field $\vec{V}_g$ [TCP06].

Suggested by Treuille et al. [TCP06], we also use the fast marching method [Set96] to solve C in Eq. 7. Its solving process can be conceptually regarded as a variant of the Dijkstra's shortest path algorithm. For more details about the solver, please refer to the work of [Set96, TCP06] . After the minimum cost field C is solved, we can obtain a smooth global velocity field $\vec{V}_g$ by taking the gradient direction of C and scaling it by insect speed s_0, described as follows:

$$\vec{V}_g(X) = \frac{-\nabla C(X)}{\| \nabla C(X) \|} \cdot s_0. \tag{8}$$

Note that in our algorithm s_0 is a user-specified parameter, and its value depends on the species of insects and specific simulation scenarios. Figure 4(b) shows an example of the solved smooth global velocity field. As illustrated in this figure, since every grid cell has at least one neighbor with a minimum cost, there is no local minimum in the solved cost field. In other words, agents (insects) will always find a valid path to the goal area, instead of getting stuck in the middle.

The main differences between the original potential field model in Continuum Crowds [TCP06] and our approach are: First, our approach can handle 3D complex simulation environment which is not provided in the original model, since a compact description of the 3D environment instead of simply adding discomfort values to grids (as in the original model) is very important in insect swarm simulation.



(a) (b) (c)

Figure 4: *An example of processing the global velocity field (2D view only for illustration purpose). (a) Computed signed distances of a discretized space by a repulsive function (top-left corner). The dotted lines of point P and Q are specific examples of signed distances. (b) The corresponding solved global velocity field (indicated by arrows). In this figure, we do not illustrate velocities inside the obstacles since agents cannot get there. (c) The combined velocity field (i.e., global velocity field + curl-noise field).*

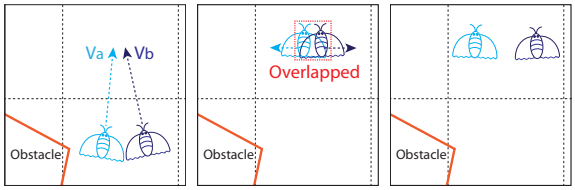
Second, we compute the desired cost field and curl-noise field just once (or when the environment changes) as a preprocessing step, without updating them every simulating frame. Because of this, the simulation rate of our approach only depends on the number of insects. Note that it is extremely slow if we directly use the original potential field model to simulate such a large-scale swarm (more than 10k agents). Third, our approach could be highly paralleled when implemented on GPU. The analysis details are discussed in Section 7.2.

6. Insect Motion Synthesis

To the end, we combine the two velocity fields: the local velocity (noise) field (Eq. 1) and the global velocity field (Eq. 8), to obtain the final governing velocity field $\vec{V}(X)$ for insects in a swarm:

$$\vec{V}(X) = \nabla \times \vec{N}(X) - \frac{\nabla C(X)}{\| \nabla C(X) \|} \cdot s_0. \qquad (9)$$

An example of the combined velocity field is illustrated in Fig. 4(c).

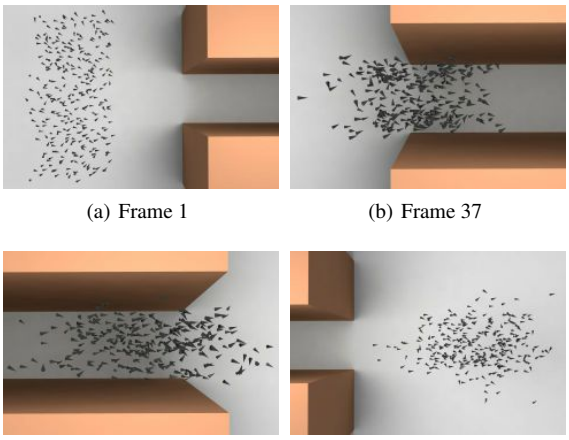


(a) Before update (b) MDE rule applied (c) After update

Figure 5: *Illustration of minimum distance enforcement. (a) Local velocities of insects A and B (marked as colored arrows, V_a and V_b, respectively); (b) when they move to new locations, an overlapping will be anticipated; (c) A and B are effectively separated.*

Minimum distance enforcement: Unfortunately, our global velocity field $\vec{V}_g$ is not as incompressible as $\vec{V}_l$. When the moving of the agents is governed by Eq. 9, artifacts may occur, which is caused by the intersection or overlapping of different agents (insects), especially at corners of obstacles (refer to Figure 5(a)). We address this issue by employing a Minimum Distance Enforcement (MDE) rule after every insect's position has been updated, as suggested by Treuille et al. [TCP06]. This rule can be briefly described as follows: If the distance between two agents are smaller than a threshold (i.e., the minimum distance), we push them apart. To enforce the minimum distance between any insect pair, once the locations of all the agents are updated at each simulation step, we iterate all the pairs of agents in the same cell to identify near-collision pairs (i.e., pairwise distance is smaller than a minimum-distance threshold), and then perform pairwise separations on them. Figure 5 illustrates an example of minimum distance enforcement. In addition, the minimum distance enforcement rule cannot absolutely avoid collisions but it has a reduced linear time complexity with respect to the dimensions of the grid. In this regard, it is a sound computational trade-off.

In sum, given a 3D environment description, the starting area(s), and the goal area(s), our approach can automatically compute smooth yet noise-aware insect trajectories without collisions. Note that the starting area(s) and the goal area(s) can be overlapped in our approach.



(a) Frame 1 (b) Frame 37

(c) Frame 100 (d) Frame 150

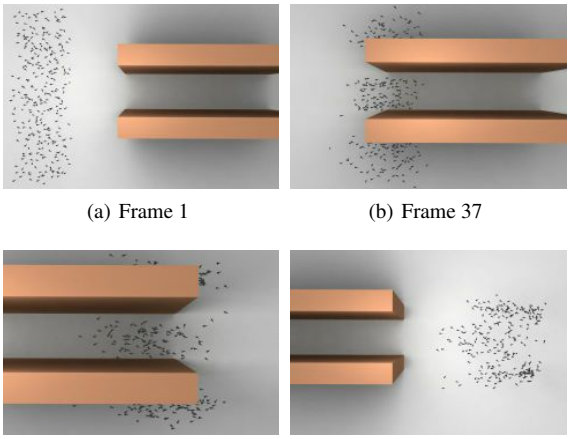
Figure 6: *Simulation of an insect swarm passing through a two-block environment*

7. Experimental Results and Evaluation

We performed simulation experiments to evaluate the effectiveness of our approach. Several selected simulation scenarios are described below. For animation results, please refer to the companion video.

Two blocks: Figure 6 shows an environment consisting of two blocks that form a narrow 3D passage. An insect swarm at one side of the blocks flies to the other side by passing through the passage. In this case, congestion may happen at the entrance of the passage, but the governing velocity field computed by our approach can effectively guide the swarm to fly along a flow-like path. With the same environment, another simulation scenario is to let all the insects in a swarm to fly along their minimum cost paths, regardless passing through the blocks or flying outside the blocks. Figure 7 shows snapshots of its simulation result.

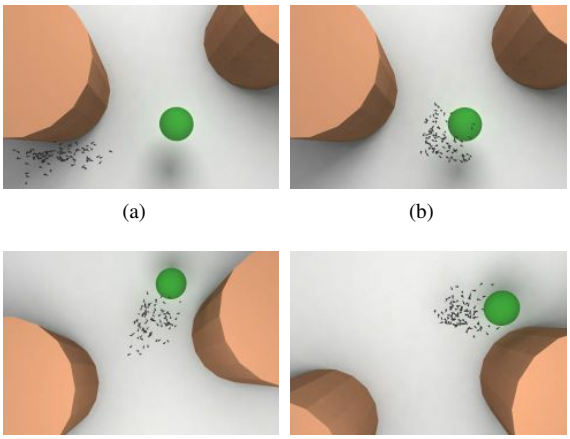
Chasing: In this simulation, an insect swarm continuously chases a target ball that is dynamically manipulated by users. When the position of the target (ball) is changed, the minimum cost field is recomputed by our approach. The ball is also regarded as a boundary condition used in Eq. 2. Snapshots of the simulation result are shown in Figure 8.



(a) Frame 1 (b) Frame 37

(c) Frame 100 (d) Frame 150

Figure 7: *Simulation of an insect swarm moving along the paths with minimal cost*



(a) (b)

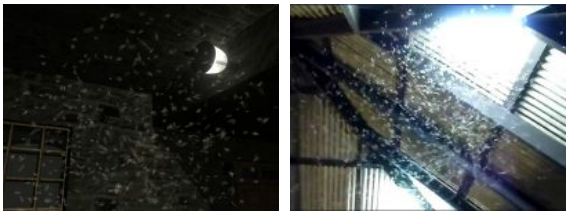
(c) (d)

Figure 8: *In this simulation, an insect swarm is continuously chasing a target green ball that is dynamically manipulated by users.*

Based on the used decay function (Eq. 3), insects near obstacles (i.e., orange balls) will go around it, instead of going through it. This simulation scenario is similar to predatory or aggressive behavior of real-world insect swarms.

Besides generating virtual insect swarms for various scripted scenarios, our approach can be directly used to simulate insect swarms observed in the real world. Through several simulation examples below, we directly compare our simulation results with the recorded real-world insect swarms. It is noteworthy that, due to the technical infeasibility of reconstructing an accurate 3D environment for the exterior scene in the recorded video, such side-by-side comparisons are very challenging for any simulation algorithms.

Aggregation: Aggregation behaviors can often be observed on many gregarious insects. Insect aggregation is typically interpreted as a defense behavior; thus, individuals that stray from swarms have an increased risk from predation [Vul90]. Figure 9(a) shows a simulated aggregation swarm consisting of 800 insects by our approach (Note that the goal area and the starting area are the same). As a comparison, Figure 9(b) shows a real swarm of moths grabbed from a publicly accessible Youtube video clip. As shown in the companion video, vortex-like swarming behavior, induced by inherent noise of insects [SESG08], can be soundly simulated by our approach.



(a) (b)

Figure 9: *Aggregation: (a) simulated insects move erratically in an undisturbed swarm; (b) a real swarm of moths grabbed from a publicly accessible Youtube video clip.*

(a) (b)

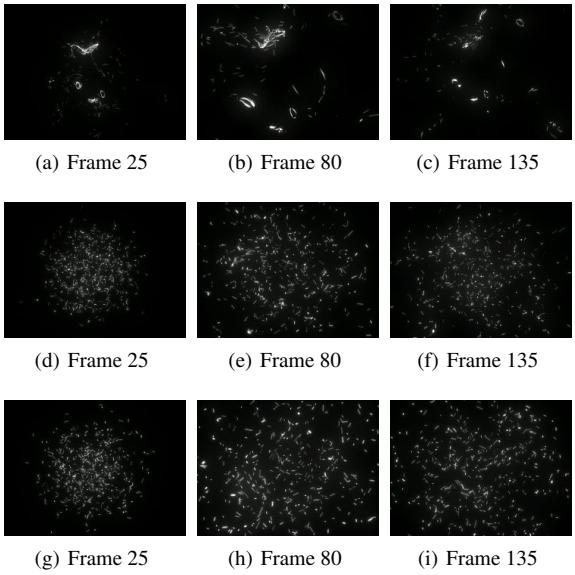
Figure 10: *Positive phototaxis: (a) a snapshot of the simulated phototaxis of 30 moths by our approach; (b) a photo taken at night by an amateur.*

Positive phototaxis: Positive phototaxis is common in organisms. However, besides its various forms of behavior, its underlying mechanisms still remain to be scientifically uncovered [MH10]. To simulate it using our approach, we first initialized the goal area centered on a street lamp and then probabilistically added a small magnitude of speed towards the lamp holder for each insect. Figure 10(a) shows a simulated swarm of 30 moths exhibiting positive phototaxis; as a comparison, Figure 10(b) shows a photo taken at night by an amateur. Figure 1(b) also shows another simulated insect swarm exhibiting positive phototaxis but with a larger number of moths (i.e., 300 moths).

Migratory locusts: One major distinction of our approach is its scalability of efficiently simulating large-scale insect swarms. To demonstrate it, we created a swarm of 5000 locusts on a $120 \times 50 \times 55$ grid (refer to Figure 1(a)). The swarm moves forward in a coordinated way while avoiding obstacles in the environment, as demonstrated in the companion video.

7.1. Comparison with Baseline Models

To validate the curl-noise model is a sound choice for simulating insect swarms, we performed the following baseline comparison experiments: (1) We blend Perlin noise and random noise into Reynold's Boids model [Rey87, Rey99] and the continuum crowds model [TCP06], respectively. Then, we side-by-side compare the results with our curl-noise based simulation results. (2) We also blend Perlin noise and random noise into our insect swarm simulation algorithm (that is, replacing the curl-noise component with Perlin or random noise), respectively. Then, we side-by-side compare the results with original curl-noise based simulation results.



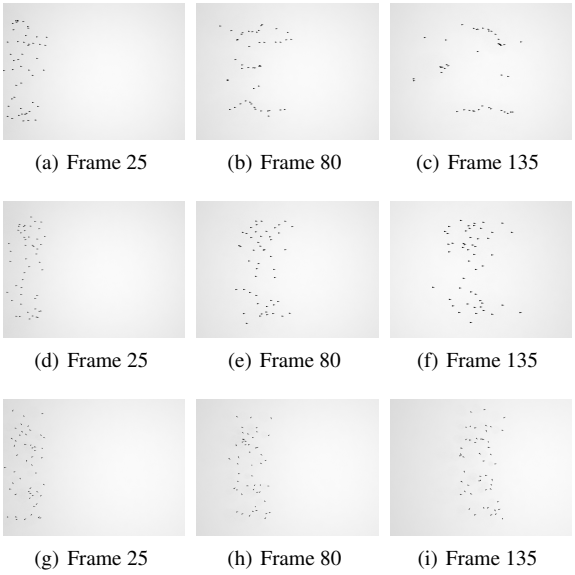
(a) Frame 25 (b) Frame 80 (c) Frame 135

(d) Frame 25 (e) Frame 80 (f) Frame 135

(g) Frame 25 (h) Frame 80 (i) Frame 135

Figure 11: *Comparison with baseline models: (a-c) Boids model with Perlin noise, (d-f) Boids model with random noise; and (g-i) our model with curl-noise.*

Comparison with Boids model + noises. Figure 11 shows side-by-side comparisons among Reynold's Boids model + Perlin noise, the Boids model + random noise, and our simulation model with curl-noise. As shown in this figure as well as the companion video, direct use of Perlin noise would result in a compressible velocity field and make individuals gather and repeat the same motions [BHN07]. Random noise + Boids model generates obvious artificial motion patterns that all the insects are just vibrating in a small area. By contrast, curl-noise in our simulation model show its advantages for simulating insect swarms; the trajectories of the simulated swarms are neither too congested nor moving into slots. Actually, those vortex-like patterns produced by curl-noise is similar to what can be observed in natural insect swarms [McI07].



(a) Frame 25 (b) Frame 80 (c) Frame 135

(d) Frame 25 (e) Frame 80 (f) Frame 135

(g) Frame 25 (h) Frame 80 (i) Frame 135

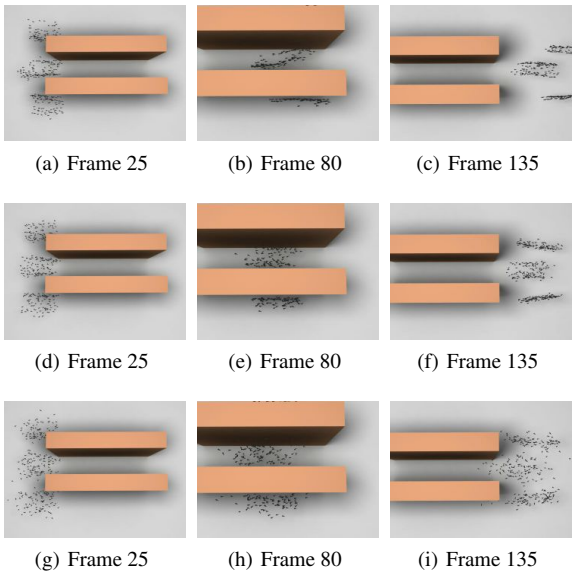
Figure 12: *Comparisons among the continuum crowds model + Perlin noise (a-c), the continuum crowds model + random noise (d-f), and our approach (g-i).*

Comparison with the continuum crowds model + noises. Figure 12 shows comparisons among the continuum crowds model + Perlin noise, the continuum crowds model + random noise, and our approach. Arguably, due to the 2D characteristics of the original continuum crowds model, we can not visually observe the convincing swarm behaviors in the results. Besides, they also simulate unsatisfied random motions (regardless which noise is added). Please refer to the supplemental demo video for the animation result comparison.

Comparison with different noises. Figure 13 shows the side-by-side comparison results by combining different noise functions (Perlin noise, random noise, and curl-noise) with potential field based path finding algorithm (described in Section 5) for a two-blocks insect swarm simulation. Evident from this figure as well as the companion video, our approach (curl-noise based) can produce more realistic simulation results than Perlin noise and random noise functions.

Scenario		Two blocks	Chasing	Aggregation	Phototaxis	Locusts
Number of insects		300	100	1000	30	5000
Noise parameters	scale	0.4	0.4	2.0	0.2	0.6
	gain	0.25	0.25	0.5	0.7	1.0
inherent speed s_0		1.5	1.5	2.5	2.5	8.0
Grid dimensions		60×40×112	80×40×120	20×20×20	100×100×100	120×50×55
Preprocessing time (sec)		0.417	0.729	0.235	2.660	19.405
Simulation FPS	CPU	219.4	587.9	66.2	1988.1	12.2
	GPU	1477.1	2247.2	1550.4	2331.0	659.0
GPU Speedup		6.731	3.822	23.395	1.172	53.891
Average MDE percentage per frame		0.36%	0.54%	0.03%	0.01%	0.68%

Table 1: *Performance statistics and parameter settings of our approach for selected simulation scenarios. More vortex-like patterns are obtained when the parameter scale is increased. The larger gain is, the more intense random motion insects can make. s_0 is usually determined by the real speed of insects.*



(a) Frame 25 (b) Frame 80 (c) Frame 135

(d) Frame 25 (e) Frame 80 (f) Frame 135

(g) Frame 25 (h) Frame 80 (i) Frame 135

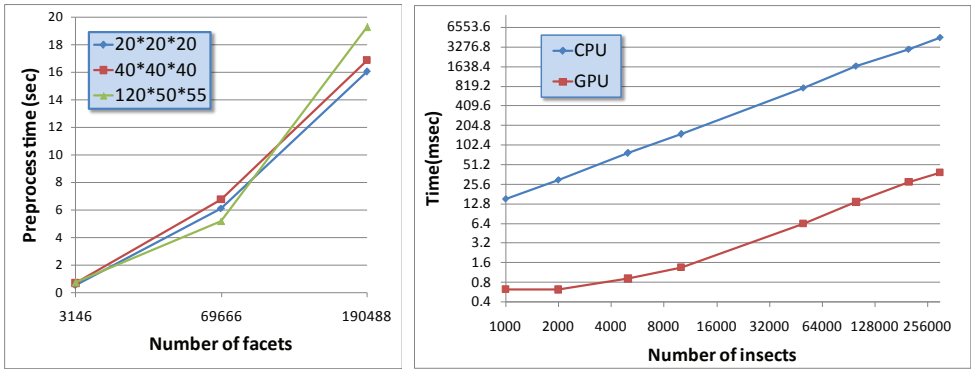
Figure 13: *Noise function comparison in a two-blocks scenario: (a-c) Perlin Noise + potential-field based path finding, (d-f) random noise + potential field based path finding, and (g-i) curl-noise + potential field based path finding.*

7.2. Performance Analysis

We implemented our approach in C++. All our experiments were performed on an off-the-shelf computer with an Intel 2 Duo CPU E7500 and 4GB memory. The performance statistics of the above described scenarios are summarized in Table 1. Its preprocessing time shows the computing time to calculate the two velocity fields (except the chasing case, the average time per user manipulation is indicated), measured in seconds. Its simulation FPS shows the average time to update all the agents' positions per frame using CPU and GPU, respectively. The obtained performance results indicate that besides its high efficiency, our approach is very friendly to parallel implementation on GPUs. In particular, when the number of simulated insects is more than ten thousands, the achieved GPU speedup is typically more than 50.

We take the #5 scenario (i.e., the migratory locusts simulation) in the above Table 1 as a specific example to further analyze how various factors (e.g., the number of insects, the dimensions of the grid, and the complexity of the obstacle meshes) affect the performance of our approach. Figure 14(a) plots the performances of the locusts scenario with varied resolutions of the grid and different numbers of the facets on the obstacle meshes. It provides experimental evidence that the computing time for the preprocessing part in our approach highly depends on the complexity of the obstacle meshes (i.e., the number of facets), but less sensitive to the dimensions of the employed grid. We also plotted simulation time of the locusts scenario with varied numbers of insects, as shown in Figure 14(b). The result shows that the runtime efficiency of our method has an approximately linear relation to the number of simulated insects, but it is independent of the complexity of the obstacle meshes and the dimensions of the used grid. As illustrated in Figure 14(b), compared with the CPU time (implemented in a single core, SIMD way), the achieved GPU speedup is more prominent when the number of simulated insects is increased (In our case, we have 128 threads per block so the speedup is insignificant when the swarm is small). E.g., when the number of simulated insects is 256K, its GPU speedup is higher than 125. Note that such a speedup is impossible to achieve by the Continuum Crowds approach [TCP06] since it has to calculate the potential field per frame.

Statistics of calling the MDE rule. As described in Section 6, the MDE rule is introduced to avoid insect collisions within one cell. In our experiments, we recorded the average MDE percentage per frame in different simulation scenarios (reported in the last row of Table 1). We found that in all our



(a) (b)

Figure 14: *(a) Plotting of the performance of the locusts simulation scenario with varied resolutions of the grid (drawn in different colors) and different numbers of the facets of the obstacles (X axis). (b) Plotting of simulation time (millisecond) of the locusts simulation scenario with varied numbers of insects.*

simulation experiments, the average percentage of calling the MDE rule is generally low, e.g., below 1%.

8. Discussion and Conclusions

In this paper, we introduce a highly efficient approach to simulate various inherent noise-aware insect swarms. Through various simulation experiments, we demonstrate that our approach can be effectively used to simulate a variety of insect swarms existing in the real world such as swarms near an attractive source or towards a moving target, aggregation swarms, positive phototaxis, migratory locusts, and so on. However, due to the technical difficulty of acquiring large-scale ground-truth data, we are not able to rigorously evaluate our approach. We only did qualitative evaluations by visually comparing the simulated results with observed insect swarms in the real world.

Our approach is highly efficient; with GPU implementation it can simulate insect swarms in real-time with more than 256K insects on an off-the-shelf computer. Our experimental results indicate that the runtime efficiency of our method has an approximately linear relation to the number of simulated insects, but it is independent of the complexity of obstacle meshes and dimensions of the grid. Not limited to graphics and animation applications, our approach can find its potential applications in other fields such as artificial intelligence, multi-agent system modeling, robot control, behavioral biology, and so on.

As an early approach to simulate 3D insect swarms, our current approach has the following limitations.

First, our current approach does not take the body size of individual insects into consideration, although the body size of any real insect is non-zero. However, since the speed of an insect is much more larger, compared to its body size; our current approach can practically handle this situation, as demonstrated in our experiments. To simulate certain insects with relatively low speeds and large body sizes (e.g., dragonflies), our current approach needs to be extended by efficiently detecting and handling body collisions between insects as well as modifying the used partial differential velocity solver accordingly. We leave it to the future work.

Figure 15: *A simulation snapshot: insects fly densely around a sharp obstacle while other regions are populated with fewer insects.*

Second, we take computation of the minimum cost field as a preprocessing module, so change of the swarm density is not considered. As shown in Figure 15, insects fly densely around a sharp obstacle while other regions are populated with fewer insects. This is a limitation of our current algorithm, because we assume every agent will move along its optimal path; but the optimal paths of the agents are not evenly distributed within certain 3D environments. A sound solution would be to consider the density as a factor and recompute the minimum cost field at every simulating step, as suggested by Treuille et al. [TCP06]; however, unlike human beings, insects could easily form a large swarm consisting of thousands of individuals. Thus, such real-time field recomputing for insect swarms would impose a greater challenge than the work of [TCP06]. As a future work, we plan to handle this issue by considering additional factors in

the preprocessing module or defining a flow field through the density of swarm on GPUs.

Third, although several goals and swarms can be supported in the domain, they all share one Eikonal Equation. Therefore, every simulated insect will choose the least-cost goal instead of the specified goal. Our current model cannot simulate multi-swarm interaction behaviors. For instance, our current approach cannot effectively simulate two insect swarms that fly through each other in opposite directions.

Fourth, our current approach does not provide sufficient interactive user controls. For example, it does not provide a tool to conveniently edit resultant potential fields; thus, it is non-trivial for users to directly control behaviors of certain insects in a simulated swarm. In the future, we plan to develop novel interactive control interfaces to improve its interactivity.

Lastly, since our current method uses uniform grids, it can only work in a relatively small domain. It cannot simulate swarms in an unlimited wide space. To overcome this limitation, one potential solution would be to apply non-uniform grids to solve both the potential field and the noise field (i.e., using sparse grids for the areas that contain a small number of insects. We plan to explore this as one of our future works.

Our current approach can generate various types of noise-aware insect motions using different parameters. However, gain and scale values are chosen empirically. Considering different insects typically have distinct swarm behaviors, these parameter value should be insect-specific. Also, since various insect species may react to boundaries in different ways, Equations 3 and 6 are expected to be insect-specific as well. This inspires us to look into an interesting inverse problem in the future: If we are able to acquire insect trajectories of various insects in a controlled environment, inversely solving the noise parameters from the acquired insect data would provide unprecedented knowledge and insights to many scientific fields that study biological motion of insects.

Acknowledgments

Xiagang Jin was supported by the China 863 program (Grant no. 2012AA011503), the National Natural Science Foundation of China (Grant no. 61272298), and Zhejiang Provincial Natural Science Foundation of China (Grant no. Z1110154). Zhigang Deng was supported by the Joint Research Fund for Overseas Chinese, Hong Kong and Macao Young Scientists of the National Natural Science Foundation of China (Grant No. 61328204).

References

[BA05] BAERENTZEN J., AANAES H.: Signed distance computation using the angle weighted pseudonormal. *Visualization and Computer Graphics, IEEE Transactions on 11*, 3 (2005), 243–253.

[BHN07] BRIDSON R., HOURIHAM J., NORDENSTAM M.: Curl-noise for procedural fluid flow. *ACM Trans. Graph. 26*, 3 (July 2007).

[BHR10] BORST A., HAAG J., REIFF D. F.: Fly motion vision. *Annual Review of Neuroscience 33*, 1 (2010), 49–70.

[BJCC*97] BEN-JACOB E., COHEN I., CZIRÓK A., VICSEK T., GUTNICK D. L.: Chemomodulation of cellular movement, collective formation of vortices by swarming bacteria, and colonial development. *Physica A: Statistical Mechanics and its Applications 238*, 1-4 (1997), 181–197.

[BKBS13] BRECHT J., KOLOKOLNIKOV T., BERTOZZI A., SUN H.: Swarming on random graphs. *Journal of Statistical Physics 151*, 1-2 (2013), 150–173.

[BSC*06] BUHL J., SUMPTER D. J. T., COUZIN I. D., HALE J. J., DESPLAND E., MILLER E. R., SIMPSON S. J.: From disorder to order in marching locusts. *Science 312*, 5778 (2006), 1402–1406.

[CSJ13] CHAO Q., SHEN J., JIN X.: Video-based personalized traffic learning. *Graphical Models 75*, 6 (2013), 305–317.

[FTT99] FUNGE J., TU X., TERZOPOULOS D.: Cognitive modeling: Knowledge, reasoning and planning for intelligent characters. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques* (New York, USA, 1999), SIGGRAPH '99, ACM Press/Addison-Wesley Publishing Co., pp. 29–38.

[GD11a] GU Q., DENG Z.: Context-aware motion diversification for crowd simulation. *Computer Graphics and Applications, IEEE 31*, 5 (2011), 54–65.

[GD11b] GU Q., DENG Z.: Formation sketching: An approach to stylize groups in crowd simulation. In *Proceedings of Graphics Interface 2011* (Ontario, Canada, 2011), GI '11, Canadian Human-Computer Communications Society, pp. 1–8.

[GD13] GU Q., DENG Z.: Generating freestyle group formations in agent-based crowd simulations. *Computer Graphics and Applications, IEEE 33*, 1 (2013), 20–31.

[GKM*01a] GOLDENSTEIN S., KARAVELAS M., METAXAS D., GUIBAS L., AARON E., GOSWAMI A.: Scalable nonlinear dynamical systems for agent steering and crowd simulation. *Computers & Graphics 25*, 6 (2001), 983–998.

[GKM*01b] GOLDENSTEIN S., KARAVELAS M., METAXAS D., GUIBAS L., GOSWAMI A.: Scalable dynamical systems for multi-agent steering and simulation. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on* (Seoul, Korea, 2001), vol. 4, IEEE, pp. 3973–3980.

[GvdBL*12] GUY S. J., VAN DEN BERG J., LIU W., LAU R., LIN M. C., MANOCHA D.: A statistical similarity measure for aggregate crowd dynamics. *ACM Trans. Graph. 31*, 6 (Nov. 2012), 190:1–190:11.

[HM95] HELBING D., MOLNÁR P.: Social force model for pedestrian dynamics. *Phys. Rev. E 51* (May 1995), 4282–4286.

[JCP*10] JU E., CHOI M. G., PARK M., LEE J., LEE K. H., TAKAHASHI S.: Morphable crowds. *ACM Trans. Graph. 29*, 6 (Dec. 2010), 140:1–140:10.

[JXW*08] JIN X., XU J., WANG C. C. L., HUANG S., ZHANG J.: Interactive control of large-crowd navigation in virtual environments using vector fields. *Computer Graphics and Applications, IEEE 28*, 6 (2008), 37–46.

[KOOP11] KULPA R., OLIVIERXS A.-H., ONDŘEJ J., PETTRÉ J.: Imperceptible relaxation of collision avoidance constraints in

virtual crowds. *ACM Trans. Graph. 30*, 6 (Dec. 2011), 138:1–138:10.

[KSHF09] KAPADIA M., SINGH S., HEWLETT W., FALOUTSOS P.: Egocentric affordance fields in pedestrian steering. In *Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games* (New York, USA, 2009), I3D '09, ACM, pp. 215–223.

[LCHL07] LEE K. H., CHOI M. G., HONG Q., LEE J.: Group behavior from video: A data-driven approach to crowd simulation. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, 2007), SCA '07, Eurographics Association, pp. 109–118.

[LCS*12] LI Y., CHRISTIE M., SIRET O., KULPA R., PETTRÉ J.: Cloning crowd motions. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, 2012), SCA '12, Eurographics Association, pp. 201–210.

[McI07] MCINNES C. R.: Vortex formation in swarms of interacting particles. *Phys. Rev. E 75* (Mar 2007), 032904.

[MH10] MOISE E. R. D., HENRY H. A. L.: Like moths to a street lamp: exaggerated animal densities in plot-level global change field experiments. *Oikos 119*, 5 (2010), 791–795.

[Mor63] MORSE R. A.: Swarm orientation in honeybees. *Science 141*, 3578 (1963), 357–358.

[NGCL09] NARAIN R., GOLAS A., CURTIS S., LIN M. C.: Aggregate dynamics for dense crowd simulation. *ACM Trans. Graph. 28*, 5 (Dec. 2009), 122:1–122:8.

[OPOD10] ONDŘEJ J., PETTRÉ J., OLIVIER A.-H., DONIKIAN S.: A synthetic-vision based steering approach for crowd simulation. *ACM Trans. Graph. 29*, 4 (July 2010), 123:1–123:9.

[PAB07] PELECHANO N., ALLBECK J. M., BADLER N. I.: Controlling individual agents in high-density crowd simulation. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, 2007), SCA '07, Eurographics Association, pp. 99–108.

[Per02] PERLIN K.: Improving noise. *ACM Trans. Graph. 21*, 3 (July 2002), 681–682.

[PvdBC*11] PATIL S., VAN DEN BERG J., CURTIS S., LIN M., MANOCHA D.: Directing crowd simulations using navigation fields. *Visualization and Computer Graphics, IEEE Transactions on 17*, 2 (2011), 244–254.

[Rey87] REYNOLDS C. W.: Flocks, herds and schools: A distributed behavioral model. *SIGGRAPH Comput. Graph. 21*, 4 (Aug. 1987), 25–34.

[Rey99] REYNOLDS C. W.: Steering behaviors for autonomous characters. In *Proceedings of the Game Developers Conference* (1999), pp. 763–782.

[RMC95] RAUCH E. M., MILLONAS M. M., CHIALVO D. R.: Pattern formation and functionality in swarm models. *Physics Letters A 207*, 3-4 (1995), 185–193.

[RMD*03] ROGERS S. M., MATHESON T., DESPLAND E., DODGSON T., BURROWS M., SIMPSON S. J.: Mechanosensory-induced behavioural gregarization in the desert locust schistocerca gregaria. *Journal of Experimental Biology 206*, 22 (2003), 3991–4002.

[RMT01] RAUPP MUSSE S., THALMANN D.: Hierarchical model for real time simulation of virtual human crowds. *Visualization and Computer Graphics, IEEE Transactions on 7*, 2 (2001), 152–164.

[SAC*08] SUD A., ANDERSEN E., CURTIS S., LIN M., MANOCHA D.: Real-time path planning in dynamic virtual environments using multiagent navigation graphs. *Visualization and Computer Graphics, IEEE Transactions on 14*, 3 (2008), 526–538.

[SASBJ11] SHKLARSH A., ARIEL G., SCHNEIDMAN E., BEN-JACOB E.: Smart swarms of bacteria-inspired agents with performance adaptable interactions. *PLoS Comput Biol 7*, 9 (09 2011), e1002177.

[SESG08] STREFLER J., ERDMANN U., SCHIMANSKY-GEIER L.: Swarming in three dimensions. *Phys. Rev. E 78* (Sep 2008), 031927.

[Set96] SETHIAN J. A.: A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences 93*, 4 (1996), 1591–1595.

[SJ12] SHEN J., JIN X.: Detailed traffic animation for urban road networks. *Graphical Models 74*, 5 (2012), 265–282.

[ST05] SHAO W., TERZOPOULOS D.: Autonomous pedestrians. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, USA, 2005), SCA '05, ACM, pp. 19–28.

[STD74] STUMM-TEGETHOFF B., DICKE A.: Surface structure of the compound eye of various drosophila species and eye mutants of drosophila melanogaster. *Theoretical and Applied Genetics 44*, 6 (1974), 262–265.

[SWL11] SEWALL J., WILKIE D., LIN M. C.: Interactive hybrid simulation of large-scale traffic. *ACM Trans. Graph. 30*, 6 (Dec. 2011), 135:1–135:12.

[TB04] TOPAZ C., BERTOZZI A.: Swarming patterns in a two-dimensional kinematic model for biological groups. *SIAM Journal on Applied Mathematics 65*, 1 (2004), 152–174.

[TCP06] TREUILLE A., COOPER S., POPOVIĆ Z.: Continuum crowds. *ACM Trans. Graph. 25*, 3 (July 2006), 1160–1168.

[TYK*09] TAKAHASHI S., YOSHIDA K., KWON T., LEE K. H., LEE J., SHIN S. Y.: Spectral-based group formation control. *Computer Graphics Forum 28*, 2 (2009), 639–648.

[Vul90] VULINEC K.: Collective security: aggregation by insects as a defense. *Insect defenses* (1990), 251–288.

[WMW86] WYVILL G., MCPHEETERS C., WYVILL B.: Data structure for soft objects. *The Visual Computer 2*, 4 (1986), 227–234.

[YEE*09] YATES C. A., ERBAN R., ESCUDERO C., COUZIN I. D., BUHL J., KEVREKIDIS I. G., MAINI P. K., SUMPTER D. J. T.: Inherent noise can facilitate coherence in collective swarm motion. *Proceedings of the National Academy of Sciences 106*, 14 (2009), 5464–5469.

[ZMR*08] ZHAN B., MONEKOSSO D., REMAGNINO P., VELASTIN S., XU L.-Q.: Crowd analysis: a survey. *Machine Vision and Applications 19*, 5-6 (2008), 345–357.