This is a repository copy of *FASTSWARM: A Data-driven FrAmework for Real-time Flying InSecT SWARM Simulation*.

White Rose Research Online URL for this paper:
https://eprints.whiterose.ac.uk/163467/

Version: Accepted Version

FASTSWARM: A Data-driven **FrA**mework for Real-time Flying In**S**ec**T SWARM** Simulation

Wei Xiang[1]   Xinran Yao[1]   He Wang[2]   Xiaogang Jin[1,3*]

[1] State Key Lab of CAD&CG, Zhejiang University, Hangzhou 310058, China

[2] School of Computing, University of Leeds, Leeds LS2 9JT, United Kingdom

[3] ZJU-Tencent Game and Intelligent Graphics Innovation Technology Joint Lab, Hangzhou 310058, China

Abstract

Insect swarms are common phenomena in nature and therefore have been actively pursued in computer animation. Realistic insect swarm simulation is difficult due to two challenges: high-fidelity behaviors and large scales, which make the simulation practice subject to laborious manual work and excessive trial-and-error processes. To address both challenges, we present a novel data-driven framework, FASTSWARM, to model complex behaviors of flying insects based on real-world data and simulate plausible animations of flying insect swarms. FASTSWARM has a linear time complexity and achieves real-time performance for large swarms. The high-fidelity behavior model of FASTSWARM explicitly takes into consideration the most common behaviors of flying insects, including the interactions among insects such as repulsion and attraction, self-propelled behaviors such as target following and obstacle avoidance, and other characteristics such as random movements. To achieve scalability, an energy minimization problem is formed with different behaviors modelled as energy terms, where the minimizer is the desired behavior. The minimizer is computed from the real-world data, which ensures the plausibility of the simulation results. Extensive simulation results and evaluations show that FASTSWARM is *versatile* in simulating various swarm behaviors, *high fidelity* measured by various metrics, easily *controllable* in inducing user controls and highly *scalable*.

1 Introduction

Insects are ubiquitous in both the real and virtual worlds, and many of them present collective behaviors for efficient and collaborative work. In the real world, flying insect swarms can exhibit a great variety of behaviors such as aggregation, mating, migration and escaping [1], where the individual behaviors are often correlated in various ways, from collaborative to competitive or even adversarial. Simulating realistic insect swarms are in the interest of many areas. In robotics, research of insect swarms has led to new algorithms for robots' collective jobs on information transfer, decision-making, task partitioning or transport [2, 3]. In computer animation, insect swarms have been used to create wondrous natural phenomena and interesting visual effects [4, 5]. However, simulating scalable collective behaviors of insect swarms with high-fidelity remains challenging.

Existing methods for simulating flying insect swarms mainly fall into two categories: empirical and data-driven. Empirical methods aim to abstract swarm behaviors into mathematical models and deterministic systems, such as the rule-based methods [6], the field-based methods [4], or a combination of the field-based with the force-based methods [5]. One limitation of such methods is that the simulated trajectories are of-

---
*Corresponding author. E-mail: jin@cad.zju.edu.cn

ten too regular and lack of visual diversity, due to their determinism nature. In contrast, data-driven methods tend to rely on real-world data, such as using computer vision techniques to capture 3D trajectories of swarms [7, 8, 9] for simulation [10, 11, 12]. However, due to the intrinsic limitations of optical sensors (e.g. occlusions), the motion capture is set up in massively simplified laboratory environments, and there are still excessive tracking errors where only short tracklets can be relatively reliably obtained. This creates tremendous difficulties in simulating flying insect swarms with the desired high-fidelity and scalability. First, the captured trajectories cannot be relied upon to extract all behaviors of flying insects. Second, the generalizability of the model based on simple data is limited by both the environment complexity and the swarm size.

In this paper, we propose a novel data-driven framework (FASTSWARM) to address the challenges for simulating flying insect swarms. Our framework models insects as agents, and the swarm behavior computation as an energy minimization problem. A variety of important behaviors identified in numerical analysis [13] and empirical observations are captured by different energy terms, including the interaction among agents, the self-propulsion of agents and the motion noise of agents, so that the minimizer leads to realistic behaviors. Besides, our framework also model user-defined behaviors by employing user-control energy terms. The total energy function is constructed in the way that it can be optimized quickly to achieve scalability and real-time performance.

During optimization, instead of seeking a minimizer by pure mathematical optimizations which would make the minimizer only ideal in theory, we seek the minimizer by referencing a motion characteristic dataset generated from the real-world data, so that the simulated behaviors mimic the real data. However, this means that the motion characteristics we rely on in the reference dataset has to be reliable. In the real-world data, although excessive noises exist and whole trajectories can rarely be obtained, velocity is much more reliable as it can be estimated from short tracklets [14]. In our simulation framework, both velocity and acceleration are regarded as the motion characteristics for generating the reference dataset, and we there-

fore optimize for the velocity to update the motion states of the agents. In addition, we use an implicit Euler scheme to improve the stability.

Formally, the contributions of the paper include:

- A novel data-driven 3D swarm simulation framework which captures a variety of biologically important behaviors.

- An optimization method that maximally makes use of real-world data to ensure the simulation fidelity for flying insect swarms.

- A scalable model for large swarm simulation with straightforward user control.

The remainder of this paper is organized as follows. After briefly introducing related work in Section 2, we give a pipeline overview of our approach in Section 3. In Section 4, we explain our optimization-based data-driven model. We show simulation results and evaluations of our method in Section 5, and discuss the limitations and future work in Section 6.

2 Related Work

Data-Driven Simulation. In graphics, data-driven methods have been proposed to simulate behaviors of crowds and traffics. Given trajectories (or tracklets) extracted from crowd data, example-based methods can blend them to generate new animations [15], use a "clone and paste" technique to generate larger crowds [16], or cluster them into groups and update the motion of an agent based on the actions of its nearest patch or associated group [17, 18, 19].

In data-driven traffic simulation, Chao et al. [20] present a video-based approach to learn the specific driving characteristics of drivers to reconstruct or simulate traffic flows. By taking the spatio-temporal information of traffic flows as a 2D texture, a texture synthesis technique is developed to populate virtual road networks with realistic traffic flows [21]. In addition, deep learning can also be used to learn the latent patterns of vehicle trajectories for intersectional traffic simulation and editing [22]. Recently, an interactive data-driven optimization approach [14] has been proposed to simulate traffic scenarios with heterogeneous agents.
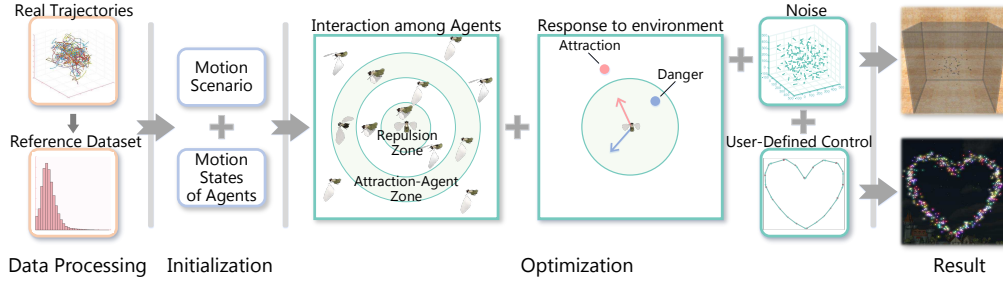
Figure 1: Overview of our data-driven approach for simulating flying insect swarms.

Although these methods can generate plausible crowd or traffic animations, they focus on 2D simulation and cannot be easily extended to 3D flying insect swarms because the motion dynamics of insects are significantly different.

Insect Swarm Simulation. There has been an interdisciplinary effort in the research of collective behaviors of insects. Researchers in agriculture proposed an insect migration trajectory simulation method to accurately predict the destinations of insect migration and achieve effective early warning to reduce the impact of pests on agriculture [23]. Public health researchers proposed an indoor flight behavior model of host-seeking mosquitoes for selecting bed nets that can effectively reduce the spread of the virus from mosquitoes [24]. In swarm robotics, swarm behaviors can be used to control collaborative robots, such as the insect-based biobots for search and rescue [25], unmanned aerial vehicle quadrotors emulating insect swarm behaviors [26], and the task allocation algorithm based on ant colonies [27].

In graphics, methods have been proposed to simulate collective behaviors of ants [28, 29]. However, it is non-trivial to extend these methods to simulate flying insect swarms because of their significant differences in motion dynamics. Data-driven methods have been developed to simulate flying insect swarms. Li et al. [10] present a framework to simulate flying insect swarms by sampling the statistical information from real datasets. However, it is not general enough because a new local steering model is needed whenever the data changes, and it is hard to simulate complex scenarios where agents should interact with dynamic stimulus in the environment by using a predefined density field. Different from the prior work, we present a general framework which is maximally data-driven, able to simulate complex swarm behaviors and automatically adaptive to data, and therefore is not tied to specific data. Recently, data-driven noise models and force-based models are introduced in [11, 12] to generate biologically plausible animations of flying insect swarms. However, they do not generalize well to more complex scenarios because they are prone to numerical errors during generalization.

Except for data-driven methods, a hybrid model combining potential fields and curl-noise [30] is developed in [4] to simulate various behaviors of flying insect swarms. However, such a forced-based and field-based method often generates trajectories that look too regular because all agents share similar motion patterns. For the special effect simulation application, Chen et al. present a flock morphing method of flying insect swarms with pre-defined shape constraints [5]. Instead of pre-defined morphing, our framework can generate plausible animations based on real-world motions to satisfy user-defined constraints.

3 Methodology Overview

As illustrated in Figure 1, FASTSWARM can be conceptually described as a three-phase process: the data processing stage, the initialization stage, and the real-time simulation stage. Our simulation algorithm makes full use of the dataset extracted from the data processing stage to compute the trajectories of the agents in the swarm. At the data processing stage, the raw data are noisy tracklets [7, 9]. For each type of insects, we generate a reference dataset that contains the motion characteristics specific to that type. Each data sample consists of two items: velocity and acceleration, estimated using forward differencing. We organize the data by the

speed, similar to [14], for fast indexing. At the initialization stage, we initialize the motion scenario, the number of insects, and the initial motion state of each agent. The motion state of an agent includes its position, velocity, motion randomness, and control direction. During simulation, we employ a data-driven approach to update the motion state for each agent. For every time step, our model selects a velocity from the reference dataset that minimizes an objective function which models the interactions, self-propulsion, and user-specified controls.

4 Data-Driven Optimization Model

The reference data is denoted as $D = \cup_v d_v$, where $d_v = [\mathbf{v}, \mathbf{a}]$, $\mathbf{v}$ is the velocity, and $\mathbf{a}$ is the acceleration. For a swarm with N agents at time t, the state of agent i ($i = 1, ..., N$) is denoted as $\mathbf{s}_{i,t} = [\mathbf{p}_{i,t}, \mathbf{v}_{i,t}, \hat{\mathbf{v}}_{i,t}^n, \hat{\mathbf{v}}_{i,t}^{cd}]$, $\mathbf{s}_{i,t} \in \mathbb{R}^{12}$, where $\mathbf{p}_{i,t} \in \mathbb{R}^3$ is the current position, $\mathbf{v}_{i,t} \in \mathbb{R}^3$ is the current velocity, $\hat{\mathbf{v}}_{i,t}^n \in \mathbb{R}^3$ is the noise direction, and $\hat{\mathbf{v}}_{i,t}^{cd}$ is the user-defined control direction. We further use $\mathbf{S}_t = \cup_i s_{i,t}$ to represent the motion state of the whole swarm at t. Then, the motion dynamics of agent i is formulated as:

$$
\begin{aligned}
\mathbf{v}_{i,t+1} &= \underset{\mathbf{v}_{i,t+1} \in d_v \in D}{\arg\min} E(i, \mathbf{v}, \mathbf{a}, \mathbf{S}_t), \\
\mathbf{p}_{i,t+1} &= \mathbf{p}_{i,t} + \mathbf{v}_{i,t+1} \Delta t, \\
\hat{\mathbf{v}}_{i,t+1}^n &= f_N(\mathbf{s}_{i,t}), \\
\hat{\mathbf{v}}_{i,t+1}^{cd} &= f_{CD}(\mathbf{s}_{i,t}, ENV_t),
\end{aligned}
\tag{1}
$$

where the velocity $\mathbf{v}_{i,t+1} \in d_v \in D$ minimizes the objective function E, $\mathbf{a}$ is the acceleration in d_v, Δt is a timestep, and $\mathbf{p}_{i,t+1}$ is the position at $t + 1$. After updating the velocity and position of each agent, we update the noise direction $\hat{\mathbf{v}}_{i,t+1}^n$ and the user-defined control direction $\hat{\mathbf{v}}_{i,t+1}^{cd}$ using f_N and f_{CD}, respectively. The objective function E is defined as:

$$
E(i, \mathbf{v}, \mathbf{a}, \mathbf{S}_t) = E_{i,int} + E_{i,sp} + E_{i,n} + E_{i,user}, \tag{2}
$$

where $E_{i,int}$ is an interaction term to model the interactions among agents, $E_{i,sp}$ is an self-propulsion term to control the motion of agents, $E_{i,n}$ is an noise-induced control term to model the randomness of agents' movements, and $E_{i,user}$ is the user-control term to enforce user-defined constraints.

For each agent i, the minimization of E aims to update $\mathbf{v}_{i,t+1}$ by selecting a $\mathbf{v} \in d_v \in D$. For simplicity, we use the selected velocity $\mathbf{v}$ and $\hat{\mathbf{v}}$ to represent $\mathbf{v}_{i,t+1}$ and its direction in the following definitions of the energy terms.

4.1 Interaction among Agents

Insects move in coordination with each other and aggregate without collisions. In FASTSWARM, such coordination is modeled as interactions which include repulsion for collision avoidance and attraction for aggregation. The interaction energy $E_{i,int}$ is therefore defined as:

$$
E_{i,int} = E_{i,rep} + E_{i,att}, \tag{3}
$$

where $E_{i,rep}$ and $E_{i,att}$ represent the repulsion energy and the attraction energy, respectively.

4.1.1 Repulsion

An agent avoids collisions with other agents that are too close to it. Based on the zonal interaction model, which dictates that an insect has a short-range repulsion within a circular distance-based zone [13], we describe the *repulsion zone* using a spherical kernel function, centered at the agent with a predefined cut-off radius d_{rep} (see Figure 1 for the 2D representation), and an agent will avoid collisions with its neighboring agents within the *repulsion zone*. Assuming that agent i will move in a timestep using velocity $\mathbf{v}$ in the data term d_v and it's repulsive neighbor j will hold its current velocity $\mathbf{v}_{j,t}$ in a timestep, then the distance-based repulsion energy $E_{i,rep}$ is calculated as follows:

$$
\begin{aligned}
\mathbf{p}'_{i,t+1} &= \mathbf{p}_{i,t} + \mathbf{v}\Delta t, \\
\mathbf{p}'_{j,t+1} &= \mathbf{p}_{j,t} + \mathbf{v}_{j,t}\Delta t, \\
E_{i,rep} &= w_{rep} \cdot \\
&\quad \frac{1}{|RN|} \sum_{j \in RN} e^{\rho\left(1 - \frac{\|\mathbf{p}'_{i,t+1} - \mathbf{p}'_{j,t+1}\|_2}{d_{rep}}\right)},
\end{aligned}
\tag{4}
$$

where $w_{rep} \geq 0$ is the weight of the repulsion energy, RN is the set of neighbor agents, $|RN|$ is the number of the repulsive neighbors, $\rho > 0$ is a constant used for the scaling of the energy, and $\mathbf{p}'_{i,t+1}$ and $\mathbf{p}'_{j,t+1}$ are the predicted positions of agent i and its neighbor j.

4.1.2 Attraction

Attraction exists between agents, and also between every agent and the center of the swarm

[13]. The attraction energy $E_{i,att}$ therefore consists of both the attraction from the neighbors and the attraction from the swarm center:

$$E_{i,att} = E_{i,oa} + E_{i,cos}, \qquad (5)$$

where $E_{i,oa}$ is the energy of attraction from the neighbors, and $E_{i,cos}$ is the attraction energy from the center of the swarm. As shown in Figure 1, the attraction zone of an agent to its neighbors is also formulated as a spherical kernel function with two cut-off radii, one inner $d_{att,1}$ and one outer $d_{att,2}$: The attraction energy from the other agents $E_{i,oa}$ is defined as:

$$E_{i,oa} = w_{oa} \cdot \frac{1}{|AN|} \sum_{j \in AN} e^{\rho \left(\frac{\|\mathbf{p}'_{i,t+1} - \mathbf{p}_{j,t}\|_2}{d_{att,1}} - 1 \right)}, \qquad (6)$$

where $w_{oa} \geq 0$ is the weight of the energy term, AN is the set of the attraction neighbors of agent i, and $|AN|$ is the number of attraction neighbors. The attraction energy on agent i is normalized by the total number of the neighbor agents.

For the attraction to the swarm center on agent i, we use $\hat{\mathbf{v}}_{i,center}$, the direction that points from the agent's current position to the swarm center, as the desired direction of the chosen velocity $\hat{\mathbf{v}}$. The attraction from swarm center E_i^{cos} is defined as:

$$E_{i,cos} = w_{cos} \cdot e^{\rho \|\hat{\mathbf{v}} - \hat{\mathbf{v}}_{i,center}\|_2}, \qquad (7)$$

where $w_{cos} \geq 0$ is the weight of the energy term.

4.2 Self-Propulsion

Besides being reactive, insects are also self-propelled. We therefore define a self-propulsion energy term, not only to drive the agents to keep moving similar to the reference trajectories, but also to response to external stimuli in environments. For agent i, the self-propulsion energy includes the internal propulsion term $E_{i,vel}$ to drive the agent to generate new trajectories similar to the data, and the reaction term $E_{i,env}$ to drive the agent to react to external stimuli:

$$E_{i,sp} = E_{i,vel} + E_{i,env}. \qquad (8)$$

4.2.1 Internal Drive

We assume that the internal drive ensures the motion smoothness, which is formulated considering the first-order and second-order derivatives of motions:

$$E_{i,vel} = E_{i,dir} + E_{i,acc}, \qquad (9)$$

where $E_{i,dir} = w_{dir} \cdot e^{\rho \|\hat{\mathbf{v}} - \hat{\mathbf{v}}_{i,t}\|_2}$ is the directional continuity energy, and $w_{dir} \geq 0$ is the weight. We also consider the continuity of its acceleration,

$$E_{i,acc} = w_{acc} \cdot (E_{i,adir} + E_{i,amag}), \qquad (10)$$

where $w_{acc} \geq 0$ is the weight. We use $\mathbf{a}' = \mathbf{v} - \mathbf{v}_{i,t}$ to compute the predicted acceleration so that if a d_v is selected, and $\mathbf{a}$ is the corresponding acceleration in d_v. In Equation 10, $E_{i,adir} = e^{\rho \|\hat{\mathbf{a}}' - \hat{\mathbf{a}}\|_2}$ is designed for the continuity of the direction of acceleration, and $E_{i,amag} = |\|\mathbf{a}'\| - \|\mathbf{a}\||$ is designed for the continuity of the magnitude. Overall, $E_{i,dir}$ minimizes the direction changes and $E_{i,acc}$ minimizes the acceleration changes compared to the real data. The regularization on both motion derivatives leads to smooth motions.

4.2.2 Response to Environment

Insects can react quickly to external stimuli, e.g. following a target, or escaping from an approaching predator. To model one stimulus, the energy of response to it is defined as:

$$E_{i,env} = w_{env} \cdot e^{\rho \Psi}, \qquad (11)$$

where $w_{env} \geq 0$ is the weight, and the function Ψ is defined as:

$$\Psi = \begin{cases} \|\hat{\mathbf{v}} - \hat{\mathbf{v}}_{i,sti}\|_2, & attracted, \\ 1 - \dfrac{\|\mathbf{p}'_{i,t+1} - \mathbf{p}_{sti,t}\|_2}{d_{danger}}, & startled, \end{cases} \qquad (12)$$

where $\mathbf{p}_{sti,t}$ is the stimulus' position, $\hat{\mathbf{v}}_{i,sti}$ is the direction pointing from the agent's current position to the stimulus. In addition, as Figure 1 shows, an agent is startled if a potential danger is within a range defined by d_{danger}.

4.3 Noise

As the movements of swarms show strong randomness, we introduce a noise-induced control term to generate plausible trajectories. We introduce the curl noise [30], which has been used in force-based models to generate more accurate simulation results [11, 12], to model the motion randomness. The function f_N denotes the curl noise function for generating a noise direction $\hat{\mathbf{v}}_{i,t}^n$. Similar to the calculation of the energy of direction continuity in Equation 9, the energy of noise-induced direction control is defined as:

$$E_{i,n} = w_n \cdot e^{\rho \|\hat{\mathbf{v}} - \hat{\mathbf{v}}_{i,t}^n\|_2}, \qquad (13)$$

where $w_n \geq 0$ is the weight of the energy term.

4.4 User-Defined Motion Control

Besides the behavioral modeling of swarms, it is necessary to induce user-control for the purpose of animation. We model user-control as a direction control signal and introduce an energy term to constrain an agent to follow predefined user-defined trajectories. For agent i, the user-control energy is calculated as:

$$E_{i,user} = w_{user} \cdot e^{\rho||\hat{\mathbf{v}} - \hat{\mathbf{v}}_{i,t}^{cd}||_2}, \tag{14}$$

where $w_{user} \geq 0$ is the weight and $\hat{\mathbf{v}}_{i,t}^{cd}$ is the control direction of the agent for generating specific trajectories.

5 Results and Evaluations

The implementation is done in C++ and the experiments were run on a PC with an Intel (R) Core (TM) i7 4.00 GHz CPU, 32 GB RAM, and an NVIDIA Geforce GTX 1060 GPU. We provide both qualitative and quantitative evaluations to demonstrate the performance of FASTSWARM. Due to the space limit, we only show representative results and refer the readers to the supplementary materials for more details. In all our experiments, we set $\rho = 2.5$, and the weights of the energy terms for the test scenarios are shown in Table 1.

5.1 Qualitative Results

We first show qualitative results. The results are divided into two parts: natural behaviors and user-controlled behaviors, the former showing the high visual realism and the latter showing controllability.

5.1.1 Collective Behaviors

We use the reference dataset from [7] to generate the *aggregation*, *mating*, and *escaping* behaviors of flying insect swarms. The results show that FASTSWARM is capable of simulating a variety of natural behaviors with good visual quality.

Aggregation. Our approach can generate the *aggregation* behavior of insect swarms in different scales (see Figure 2). Figure 2(a) and Figure 2(b) respectively shows two swarms of different sizes in the aggregation scenario of the reference dataset with an unchanged swarm center.
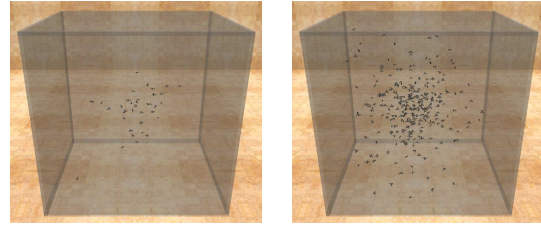
(a) 37 insects   (b) 300 insects

Figure 2: *Aggregation* behavior of insect swarms in different scales.

Mating. Figure 3 shows the *mating* behavior of insects generated by our approach. 100 male agents (the cyan ones) are attracted by a female (the red one) insect and run after the female.

Escaping. Our approach can also generate the *escaping* behavior of insect swarms. In Figure 4, 100 flying insects are startled by a sudden obstacle, and try to escape from the obstacle. During simulation, the obstacle will pass through the swarm, and the insects in the swarm escape from the danger and then aggregate again.

Figure 3: *Mating* behavior.   Figure 4: *Escaping* behavior.

5.1.2 User-Defined Shape Constraint

To evaluate the controllability, we ask the users to draw different 3D shapes which are used as constraints. Figure 5 shows two shape-constrained examples to constrain the agents to move following different 3D curves. We regard the 3D curve as a bidirectional curve with several keypoints (Figure 1: User-Defined Control). Each agent is randomly initialized near a key point with a random velocity selected from the referenced dataset. Then, it is assigned a task to traverse the keypoints. During simulation, we update the control direction $\hat{\mathbf{v}}_{i,t}^{cd}$ of agent i as the direction from the agent's position $\mathbf{p}_{i,t}$ to the po-

sition of the goal keypoint $\mathbf{p}_{i,t}^{goal}$:

$$
\begin{aligned}
\mathbf{V}_{i,goal} &= \mathbf{p}_{i,t}^{goal} - \mathbf{p}_{i,t}, \\
\hat{\mathbf{v}}_{i,t}^{cd} &= \hat{\mathbf{V}}_{i,goal},
\end{aligned}
\tag{15}
$$

where $\hat{\mathbf{V}}_{i,goal}$ is the directional vector of $\mathbf{V}_{i,goal}$. The goal keypoint will be updated to the next keypoint when the agent reaches its current goal.

(a) 500 flying insects form a heart shape.

(b) 500 flying insects form a star shape.

Figure 5: Two shape-constrained flocking examples. There are 11 key points in (a) and 10 key points in (b).

5.2 Quantitative Evaluations

5.2.1 Time Performance

To quantitatively test the scalability, we evaluate FASTSWARM with an increasing number of insects in an *aggregation* scenario, similar to Section 5.1.1. Table 2 shows the time performance of the test scenarios in Section 5.1. Figure 6 shows the time performance of our method against different swarm scales. Theoretically, the time complexity is $O(kN)$ where N is the number of insects and k is data sample size in the reference data. In our experiments, $k = 300$ is enough to generate all behaviors. The linear time complexity guarantees the high-performance of FASTSWARM. We further accelerate the computation in three ways: fast indexing, reduced search space and parallelization. For fast indexing, we discretize the space into a 3D grid and index the neighborhood of every agent for only local search and computation during simulation. For reduced search space, we sort the reference dataset by the speed and divide the dataset into groups, and our algorithm only traverses the corresponding group and the groups with similar speeds for a new velocity for

an agent. Besides, our algorithm is highly parallelizable and can be concurrently computed for multiple agents.
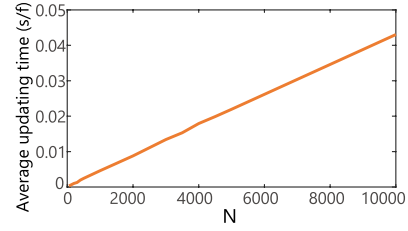

Figure 6: Time performance of our approach. The computation cost is linear w.r.t the number of insects.

5.2.2 Comparisons

We compare FASTSWARM with the two state-of-the-art data-driven methods in [10, 11]. Unlike [10] which calculates waypoints for flying insects by sampling the length of trajectory segment and the turning angle from a reference dataset, our method significantly differs in modelling individual behaviors, such as attraction, repulsion, etc, which gives better generalization on data. On the one hand, the local steering behavior in our method is mainly governed by the repulsion behavior and can work with any data, while a new local steering algorithm for collision avoidance has to be chosen in [10] whenever data changes. On the other hand, our method is able to simulate complex scenarios with different behaviors and still provide easy control to the user, while in [10], the global motion control needs a predefined density field which defines the preference of the choice of the waypoints, making it hard for the swarm to interactively respond to dynamic stimulus in the environment, e.g. *mating* behaviors.

We compare our method with the method in [11] and the ground-truth under different metrics. Specifically, the compared method in [11] is a combination of the data-driven technique and the force-based model and uses different forces to model the key behaviors of flying insects, and the ground-truth are the real tracklets in [9]. In both comparisons, the scenes are the same aggregation scenario in a box as the ground-truth [9], and the swarms are initialized by randomly selecting one frame from the real data. Since directly comparing individual tra-

Scenario		w_{rep}	w_{oa}	w_{cos}	w_{dir}	w_{acc}	w_{env}	w_n	w_{user}
Aggregation (N = 37 / 300)		1	1	0.05	1	1	0	0.2	0
Mating		1	1	0.05	1	1	1	0.5	0
Escaping	Aggregation	1	1	0.2	1	1	2	0.3	0
	In danger		0	0					
Shape Constraint (Heart / Star)		1	0.5	0	1	1	0	1	1

Table 1: The weights for our implementation in Section 5.1.

Scenario	N	Time (s/f)
Aggregation	37	0.0003
	300	0.0025
Mating	100	0.0159
Escaping	100	0.0107
Shape Constraint (Heart / Star)	500	0.0037

Table 2: Time performance of the simulation results shown in Section 5.1.

jectories is not possible, the evaluation metrics include distributions of density, velocity and acceleration, as they capture both the state and the motion dynamics. Similar to previous work [14], we use minimum-distance to describe the density. Results are shown in Figure 7. The distributions of velocity, minimal distance and acceleration of FASTSWARM are much closer to those in the ground-truth than [11]. This is true for both motion dynamics (velocity in Figure 7(a) and acceleration in Figure 7(c-d)) and states (density in Figure 7(b)). The better motion dynamics shows that FASTSWARM is superior in capturing realistic behaviors.

For comparing the density, in [11], the density of the swarm is controlled simply by a distance-based attraction force, and the homogeneous distance control results in the relatively uniform density of the swarm as the agents aggregate around the attraction boundary. Addressing this issue, we control the aggregation of agents by combining the distance-based attraction among agents with a direction control that drives the agents to move towards the swarm center, and Figure 7(b) shows that our method performs better in controlling the density of the swarm to be similar to the reference dataset.

5.2.3 Trajectory Synthesis

Given only trackelets (instead of whole trajectories) are avaiable, FASTSWARM provides a possible avenue to recover them for animation applications. Although it is difficult to exactly recover the trajectories or even compute their numerical accuracy, due to the lack of ground-truth data, our recovered trajectories can achieve global visual similarities. As shown in Figure 8, we extract the trajectory segments of 37 agents from the real-world dataset of [7] (Figure 8(a)). By taking these trajectory segments as the initial trajectories of a swarm with 37 agents, our method can be used to predict the subsequent trajectories (Figure 8(c)).

Our method can also synthesize plausible insect swarm animations by mixing the trajectories from the real-world dataset with the synthetic trajectories. As Figure 8(d) shows, in the real-world trajectories in 8(b), we add 100 agents that are simulated by FASTSWARM.

6 Conclusion

We have presented a general and scalable data-driven optimization framework to simulate flying insect swarms in real time. Our optimization method is capable of generating natural collective behaviors of flying insects by utilizing a motion characteristic dataset extracted from real data. The generated animations are plausible and have high visual realism. We have validated our approach using extensive experiments through qualitative and quantitative analysis. Our method also provides a means to generate interesting shape-constrained swarm behaviors controlled by users, which benefits computer animation. Moreover, our method can be used to predict the missing trajectories in the captured dataset and augment the dataset with
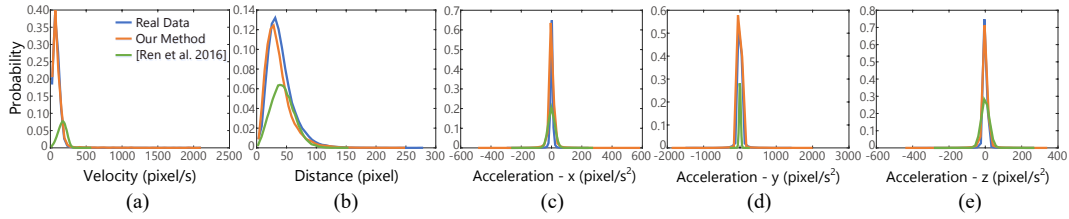
Figure 7: Comparisons of the distributions of the velocity (a), minimum distance (b), and acceleration (c-e for x, y and z direction) between our method, the real data, and [11].
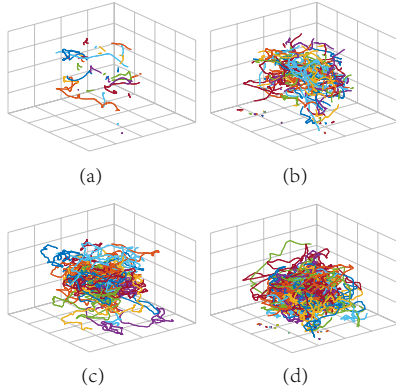

Figure 8: Results of trajectory synthesis. (a) shows the captured intermittent trajectories of 37 agents. (b) shows the real trajectories of the following 500 frames where exist lots of intermittent segments. (c) shows our prediction result of (a) by extending the initial trajectories to 500 frames. (d) shows the blending trajectories of 100 virtual agents with (b).

different number of insects while maintaining the global visual similarity to the real data.

Limitations and Future Work. Our method relies on the quality of the reference dataset. However, as our reference dataset is extracted from imperfect trajectories captured using computer vision techniques, which contain noises and erroneous data, our result may replicate the defects in the input data. In the future, we are interested in exploring a more general framework so that it can simulate any types of motion (not restricted to insects) with limited captured trajectories from real world or created by animators. We are also planning to employ the Long Short-Term Memory (LSTM) networks to explore the latent representations of trajectories to further enhance our results.

References

[1] Michael Sinhuber and Nicholas T Ouellette. Phase coexistence in insect swarms. *Physical review letters*, 119(17):178003, 2017.

[2] Christian Blum and Xiaodong Li. Swarm intelligence in optimization. In *Swarm intelligence*, pages 43–85. Springer, 2008.

[3] José Halloy, Grégory Sempo, Gilles Caprari, Colette Rivault, Masoud Asadpour, Fabien Tâche, Imen Saïd, Virginie Durier, Stephane Canonge, Jean Marc Amé, et al. Social integration of robots into groups of cockroaches to control self-organized choices. *Science*, 318(5853):1155–1158, 2007.

[4] Xinjie Wang, Xiaogang Jin, Zhigang Deng, and Linling Zhou. Inherent noise-aware insect swarm simulation. *Computer Graphics Forum*, 33(6):51–62, 2014.

[5] Qiang Chen, Guoliang Luo, Yang Tong, Xiaogang Jin, and Zhigang Deng. Shape-constrained flying insects animation. *Computer Animation and Virtual Worlds*, 30(3-4):e1902, 2019.

[6] Craig W Reynolds. Flocks, herds and schools: A distributed behavioral model. *ACM SIGGRAPH Computer Graphics*, 21(4):25–34, 1987.

[7] Douglas H. Kelley and Nicholas T. Ouellette. Emergent dynamics of laboratory insect swarms. *Scientific reports*, 3:1073, 2013.

[8] James G Puckett, Douglas H Kelley, and Nicholas T Ouellette. Searching for effective forces in laboratory insect swarms. *Scientific reports*, 4:4766, 2014.

[9] Hai Shan Wu, Qi Zhao, Danping Zou, and Yan Qiu Chen. Automated 3d trajectory measuring of large numbers of moving particles. *Optics express*, 19(8):7646–7663, 2011.

[10] Weizi Li, David Wolinski, Julien Pettré, and Ming C. Lin. Biologically-inspired visual simulation of insect swarms. *Computer Graphics Forum*, 34(2):425–434, 2015.

[11] Xinjie Wang, Jiaping Ren, Xiaogang Jin, and Dinesh Manocha. Bswarm: biologically-plausible dynamics model of insect swarms. In *Proceedings of the 14th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 111–118, 2015.

[12] Jiaping Ren, Xinjie Wang, Xiaogang Jin, and Dinesh Manocha. Simulating flying insects using dynamics and data-driven noise modeling to generate diverse collective behaviors. *PloS one*, 11(5):e0155698, 2016.

[13] Ryan Lukeman, Yue-Xian Li, and Leah Edelstein-Keshet. Inferring individual rules from collective behavior. *Proceedings of the National Academy of Sciences*, 107(28):12576–12580, 2010.

[14] Jiaping Ren, Wei Xiang, Yangxi Xiao, Ruigang Yang, Dinesh Manocha, and Xiaogang Jin. Heter-sim: Heterogeneous multi-agent systems simulation by interactive data-driven optimization. *IEEE Transactions on Visualization and Computer Graphics*, 2019.

[15] Eunjung Ju, Myung Geol Choi, Minji Park, Jehee Lee, Kang Hoon Lee, and Shigeo Takahashi. Morphable crowds. *ACM Transactions on Graphics (TOG)*, 29(6):1–10, 2010.

[16] Yi Li, Marc Christie, Orianne Siret, Richard Kulpa, and Julien Pettré. Cloning crowd motions. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 201–210, 2012.

[17] Kang Hoon Lee, Myung Geol Choi, Qyoun Hong, and Jehee Lee. Group behavior from video: a data-driven approach to crowd simulation. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 109–118, 2007.

[18] Mingbi Zhao, Stephen John Turner, and Wentong Cai. A data-driven crowd simulation model based on clustering and classification. In *2013 IEEE/ACM 17th International Symposium on Distributed Simulation and Real Time Applications*, pages 125–134, 2013.

[19] Panayiotis Charalambous and Yiorgos Chrysanthou. The pag crowd: A graph based approach for efficient data-driven crowd simulation. *Computer Graphics Forum*, 33(8):95–108, 2014.

[20] Qianwen Chao, Jingjing Shen, and Xiaogang Jin. Video-based personalized traffic learning. *Graphical Models*, 75(6):305–317, 2013.

[21] Qianwen Chao, Zhigang Deng, Jiaping Ren, Qianqian Ye, and Xiaogang Jin. Realistic data-driven traffic flow animation using texture synthesis. *IEEE Transactions on Visualization and Computer Graphics*, 24(2):1167–1178, 2018.

[22] Huikun Bi, Tianlu Mao, Zhaoqi Wang, and Zhigang Deng. A deep learning-based framework for intersectional traffic simulation and editing. *IEEE Transactions on Visualization and Computer Graphics*, 2019.

[23] Yixuan Wang, Rui Wang, Kai Cui, Weiming Tian, Dongli Wu, and Shuqing Ma. Insect trajectory simulation method based on radar observation. *The Journal of Engineering*, 2019(21):7365–7369, 2019.

[24] Jeff Jones, Gregory D Murray, and Philip J McCall. A minimal 3d model of mosquito flight behavior around the human baited bed net. *bioRxiv*, 2020.

[25] Alper Bozkurt, Edgar Lobaton, Mihail Sichitiu, Tyson Hedrick, Tahmid Latif, Alireza Dirafzoon, Eric Whitmire, Alexander Verderber, Juan Marin, and Hong Xiong. Biobotic insect swarm based sensor networks for search and rescue. In *Signal Processing, Sensor/Information Fusion, and Target Recognition XXIII*, volume 9091, pages 498–503, 2014.

[26] Argel A Bandala, Elmer P Dadios, Ryan Rhay P Vicerra, and Laurence A Gan Lim. Swarming algorithm for unmanned aerial vehicle (uav) quadrotors–swarm behavior for aggregation, foraging, formation, and tracking. *Journal of Advanced Computational Intelligence and Intelligent Informatics*, 18(5):745–751, 2014.

[27] Yara Khaluf, Seppe Vanhee, and Pieter Simoens. Local ant system for allocating robot swarms to time-constrained tasks. *Journal of Computational Science*, 31:33–44, 2019.

[28] Shihui Guo, Meili Wang, Gabriel Notman, Jian Chang, Jianjun Zhang, and Minghong Liao. Simulating collective transport of virtual ants. *Computer Animation and Virtual Worlds*, 28(3-4):e1779, 2017.

[29] Wei Xiang, Jiaping Ren, Kuan Wang, Zhigang Deng, and Xiaogang Jin. Biologically inspired ant colony simulation. *Computer Animation and Virtual Worlds*, 30(5):e1867, 2019.

[30] Robert Bridson, Jim Houriham, and Marcus Nordenstam. Curl-noise for procedural fluid flow. *ACM Transactions on Graphics (TOG)*, 26(3):46, 2007.
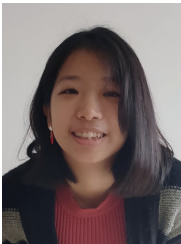
Acknowledgements

Author Biography

Wei Xiang is a Ph.D candidate in the State Key Lab of CAD&CG, Zhejiang University, P. R. China. Prior that, she received her B.Sc. degree in computer science and technology in 2016 from Shandong University, P. R. China. Her main research interests include insect swarm animation, crowd animation and traffic simulation.

Xinran Yao is a master student in the State Key Lab of CAD&CG, Zhejiang University, P. R. China. She graduated from Zhejiang University with a bachelor's degree in 2018, majoring in digital media technology. In recent years, her main researches focus on crowd animation and game AI.

He Wang is an Associate Professor at the School of Computing, the University of Leeds, UK. He leads a research team in computer graphics, vision and machine learning and applications. He serves as the Director of High Performance Graphics and Game Engineering program and an Academic Lead in Visualization & AI at Centre for Immersive Technologies at Leeds University. Before Leeds, he was a Postdoctoral Associate at Disney Research Los Angeles. He received his PhD and did a post-doc afterwards both in the School of Informatics, the University of Edinburgh. Before his PhD, he worked in industry for 4 years after graduating from Zhejiang University, China.

Xiaogang Jin is a Professor of the State Key Lab of CAD&CG, Zhejiang University, P. R. China. He received his BSc degree in computer science in 1989 and his MSc and PhD degrees in applied mathematics in 1992 and 1995, respectively, all from Zhejiang University. His current research interests include traffic simulation, insect swarm simulation, physically based animation, cloth animation, special effects simulation, implicit surface computing, non-photo realistic rendering, computer-generated marbling, and digital geometry processing. He received an ACM Recognition of Service Award in 2015 and the Best Paper Awards from CASA 2017 and CASA 2018. He is a member of the IEEE and the ACM.