Heter-Sim: Heterogeneous Multi-Agent Systems Simulation by Interactive Data-Driven Optimization

Jiaping Ren, Wei Xiang, Yangxi Xiao, Ruigang Yang, *Senior Member, IEEE*, Dinesh Manocha, *Fellow, IEEE*, and Xiaogang Jin, *Member, IEEE*

Abstract—Interactive multi-agent simulation algorithms are used to compute the trajectories and behaviors of different entities in virtual reality scenarios. However, current methods involve considerable parameter tweaking to generate plausible behaviors. We introduce a novel approach (Heter-Sim) that combines physics-based simulation methods with data-driven techniques using an optimization-based formulation. Our approach is general and can simulate heterogeneous agents corresponding to human crowds, traffic, vehicles, or combinations of different agents with varying dynamics. We estimate motion states from real-world datasets that include information about position, velocity, and control direction. Our optimization algorithm considers several constraints, including velocity continuity, collision avoidance, attraction, direction control. Other constraints are implemented by introducing a novel energy function to control the motions of heterogeneous agents. To accelerate the computations, we reduce the search space for both collision avoidance and optimal solution computation. Heter-Sim can simulate tens or hundreds of agents at interactive rates and we compare its accuracy with real-world datasets and prior algorithms. We also perform user studies that evaluate the plausible behaviors generated by our algorithm and a user study that evaluates the plausibility of our algorithm via VR.

Index Terms—Multi-agent model, heterogeneous group, data-driven method, physically driven simulation

1 Introduction

Many virtual reality and training systems need to be able to simulate different types of agents, including human crowds and traffic. Applications include VR therapy for crowd phobias, traffic agents for autonomous driving, urban design and planning, driving simulators for education and entertainment, etc. It is important to simulate the behaviors and trajectories of different types of agents, including pedestrians and vehicles, and the interactions between such heterogeneous agents. Furthermore, it is important to develop general plausible algorithms that are applicable to a wide variety of scenarios.

There are extensive works on interactive multi-agent simulation, including crowd simulation and traffic simulation. These works include techniques based on rule-based methods [1], physics-based simulations [2], [3], vision-based methods [4], energy-based models [5], data-driven methods [6], [7], and combinations of these approaches [8], [9]. These methods are flexible and have been successfully applied to different scenarios. However, they often use many parameters and require a significant amount of effort to achieve good results that are plausible and match the behaviors observed in real-world scenarios. Furthermore, the results of these methods often seem too regular because all the agents have similar locomotion or movement patterns.

With the improvement of data acquisition techniques, more data-driven methods are emerging. Most of these methods are patch-based or use real-world agent trajectories [2], [9], [10], [11]. These methods extract patches or trajectory segments from input datasets and either connect them with some rules or use them to learn some characteristics of an agent's motion. With these methods, users can generate more plausible or more accurate results than with traditional rule-based or physics-based simulation methods. However, the variety of the simulation results depends on that of input data. If the amount of input data is small, the simulation results will be periodic and monotonous.

Most of the existing methods only apply to one kind of agent, e.g., only human pedestrians or only vehicles. In contrast, we want to use a general method to model the behaviors of different kinds of agents in a heterogeneous setting while retaining the motion features of each kind of agent. This is important in many situations like simulating the motion trajectories and interactions between cars and humans at a traffic crossing. Data-driven methods can help us with simulating interactions between heterogeneous agents by preserving the motion features of each kind of

- *J. Ren, W. Xiang, Y. Xiao, and X. Jin are with State Key Lab of CAD&CG, Zhejiang University, Hangzhou 310058, P. R. China. E-mail: {ren_jia_ping, xiang_wei, x_x}@zju.edu.cn, jin@cad.zju.edu.cn.*
- *R. Yang is with the University of Kentucky, Lexington, KY 40506, and with Baidu Research, Beijing 100193, China, and also with National Engineering Laboratory of Deep Learning Technology and Application, Beijing 100193, China. E-mail: ryang@cs.uky.edu.*
- *D. Manocha is with the Department of Computer Science, University of Maryland, College Park, MD 20742. E-mail: dm@cs.umd.edu.*

Manuscript received 27 Jan. 2019; revised 16 Sept. 2019; accepted 28 Sept. 2019. Date of publication 0 . 0000; date of current version 0 . 0000.
(Corresponding author: Xiaogang Jin.)
Recommended for acceptance by P. Wonka.
Digital Object Identifier no. 10.1109/TVCG.2019.2946769

Fig. 1. Our heterogeneous multi-agent simulation algorithm can be used for scenarios with tens or hundreds of different types of agents sharing a physical space. Pedestrians walking on a street (the first). Cars moving on a twisting road (the second). Traffic including cars and pedestrians (the third). Traffic shown through VR (the fourth). Our approach can generate plausible behaviors at interactive rates on a desktop PC and through VR.

agent. However, data-driven methods depend on the input data, and it is difficult to simulate behavior in a scenario that is different from the one that generated the input data.

Main Results. We present a novel, heterogeneous multi-agent simulation algorithm (Heter-Sim) that combines the benefits of prior data-driven and physics-based simulation methods to generate general and plausible simulations. Our interactive approach can simulate not only heterogeneous agents while generating plausible behaviors, but also scenarios different from those included in the input datasets. We convert various datasets captured using different types of sensors into a uniform format and extract the agents' states, including velocity information. We model the decision-making or local navigation process of each agent as an optimization problem and define an energy function that considers collision avoidance, attraction, velocity continuity, and direction control. Our energy function tries to match the results with the characteristics of real-world data. At a given moment, each agent chooses a velocity from a dataset. We align the control directions between simulation agents and real-world agents to diversify agents' possible behaviors and movements where there is relatively less input data available. To accelerate the computation, we utilize spatial continuity to reduce possible collisions and use the velocity continuity to reduce the solution space for energy functions.

Overall, the novel contributions of our work include:

- A general, optimization-based method to simulate heterogeneous multi-agent systems. We use our approach to simulate crowds, traffic, and any combination of those agents.
- A data-driven scheme to improve the plausibility of our simulation. We use two fast search methods based on spatial continuity and velocity continuity to search for possible collision-free solutions.
- A constraint energy function to achieve the heterogeneity of the simulation system. We use different constraint energy functions to model various constraints on dynamics, traffic rules, and interactions for heterogeneous agents.
- A direction adaptation method to simulate more kinds of scenarios. We use direction control, which computes ideal directions, to guide agents in various environments. Our method can simulate agents' behaviors that may differ from those captured by the input data.

We highlight the performance of our approach on different scenarios in Fig. 1. In practice, our approach can generate plausible trajectories and behaviors for tens or hundreds of heterogeneous agents at interactive rates. To demonstrate the benefits of our method, we have conducted two user studies to evaluate the benefits of our method over prior methods while using a top-down view and an agent's view. In both studies, participants exhibit significant preference for our method over a prior crowd simulation method [12] and a traffic simulation method [7]. We also conduct a user study to compare the user experience via VR and via desktop, and VR shows a better user experience (see Section 7).

2 Related Work

There is considerable research in multi-agent simulation, including many algorithms for simulating crowds and traffic. In this section, we give a brief overview of prior methods for parameter estimation and data-driven simulation.

2.1 Parameter Estimation and Real-World Characteristics

Parameter estimation with real-world datasets improves the accuracy of simulation methods. Researchers utilize empirical data to compute the parameters used for rule-based or physically-based multi-agent simulation methods automatically. Wolinski et al. [13] present a method to compute optimal parameters for rule-based or physically-based multi-agent simulation algorithms. Berseth et al. [14] present an approach that computes parameters for steering methods by minimizing any combination of performance metrics. Karamouzas et al. [15] use distortion and longitudinal dispersion of the group to evaluate the results from simulations. Different from these parameter estimation methods, our approach finds the best velocity from real-world datasets to generate realistic motions.

Many techniques have been proposed to learn agent characteristics from empirical data and to then use them for multi-agent simulation. Lee et al. [16] present a crowd simulation method which use an agent model generated from real-world observations. Chao et al. [17] apply characteristics of drivers from an empirical video to an agent-based model. Boatright et al. [18] classify the contexts and learn the characteristics from a dataset. Charalambous et al. [19] present a real-time synthesis method for crowd steering behaviors with the temporal perception pattern. Bi et al. [20] simulate the process of lane-changing in traffic by learning characteristics from features of real vehicle trajectories. Kim et al. [9] compute collision-free trajectories of virtual pedestrians by learning pedestrian dynamics from 2D trajectories. Besides, Ondřej et al. [4] present a vision-based approach of collision avoidance between walkers that fit the requirements of interactive crowd simulation. Our data-driven optimization algorithm is complimentary to these algorithms and can be combined with them.

Reconstruction of certain aspects of real-world scenes has also been used for multi-agent simulation, especially for traffic simulation. Li et al. [21] reconstruct traffic with GPS mobile vehicle data. Wilkie et al. [22] drive an agent-based traffic simulator by using the state of traffic flow estimated from sparse sensor measurements. Yoon et al. [23] propose a refinement method to reconstruct a holistic view of crowd's movements with noisy tracked data. Qiao et al. [24] present a trajectory interpolation method by combining trajectory estimation and global optimization. Our approach is more general than these prior methods.

2.2 Data-Driven Multi-Agent Simulation

Patch-based methods transfer the original trajectories from empirical data into patches and connect these patches with some rules. Yersin et al. [25] extend the concept of motion patches to dense populations in large environments. Li et al. [26] animate large crowds with examples of multi-agent motions by using a copy-and-paste technique. Hyun et al. [27] tile deformable motion patches, which describe episodes of the movements of multiple characters. Jordao et al. [10] propose a crowd sculpting method to guide crowd motion by using intuitive deformation gestures.

As with patch-based methods, researchers replicate trajectory tubes extracted from empirical data to synthesize new agent animations. Lai et al. [28] introduce group motion graphs to animate groups of discrete agents with empirical data. Lerner et al. [29] generate seemingly natural behaviors by copying trajectories from real people and applying them to simulated agents. Ju et al. [11] generate new animations, which can include arbitrary numbers of agents, by blending existing data. Zhao et al. [30] cluster the examples extracted from human motion data and combine similar examples to produce an output. Li et al. [31] propose a general, biologically-inspired framework with a three-level method using statistical information from real datasets. Kielar et al. [32] predict movement behaviors of crowds with a cognitive agent framework. Liu et al. [33] generate crowd movements with neural networks by considering environment layouts. A new data-driven method has been proposed by Chao et al. [7]. They compute the velocity for each agent in each frame from empirical data. However, this method is time-consuming because it tries to minimize the overall traffic texture energy and is therefore not useful for interactive applications. Our approach is complimentary to prior data-driven methods and presents a new method that combines data-driven with physics-based multi-agent methods.

3 Data-Driven Optimization

In this section, we introduce our data-driven optimization approach to simulate heterogeneous multi-agent systems.

3.1 Terminology and Notation

We use *agent* to represent the virtual character in our method. We also use the term *state* to represent the motion characteristics of each agent. Our method is general and applicable for both 2D and 3D motions. State can therefore refer to an agent's movements in either 2D or 3D space. In this paper, we limit our discussions to 2D agents.

We use set $\mathcal{G}$ to specify the set of agents in the scenario. We use the vector $\mathbf{s} = [\mathbf{p}, \mathbf{v}, \mathbf{v}^{\mathrm{d}}]^{\mathrm{T}}$, $\mathbf{s} \in \mathbb{R}^6$ to specify an agent's state, where $\mathbf{p} \in \mathbb{R}^2$ is the agent's position, $\mathbf{v} \in \mathbb{R}^2$ is the velocity, and $\mathbf{v}^{\mathrm{d}} \in \mathbb{R}^2$ is the control direction that guides the motion direction of agents. Then the state of the group becomes $\mathcal{S} = \cup_i \mathbf{s}_i$, where $\mathbf{s}_i$ is the state of agent i. Distinct from the velocity $\mathbf{v}$, the control direction $\mathbf{v}^{\mathrm{d}}$ controls the agent's global direction. We use $\hat{\mathbf{v}} = \frac{\mathbf{v}}{\|\mathbf{v}\|}$ to represent the unit vector of $\mathbf{v}$. We also use $\mathbf{v}_{i,n}$ to represent the velocity of agent i at time t_n. For any state $\mathbf{s} = [\mathbf{p}, \mathbf{v}, \mathbf{v}^d] \in \mathcal{S}$, $\mathbf{p} \in \mathcal{S}_{\mathrm{p}}$, $\mathbf{v} \in \mathcal{S}_{\mathrm{v}}$, $\mathbf{v}^{\mathrm{d}} \in \mathcal{S}_{\mathrm{v}^d}$. We represent our method by $[S(), D(), I(), F()]^{\mathrm{T}}$, where S is the environment evolution function, D is the data processing function, I is the initialization function, and F is the decision making function. S determines the external environment, which consists of the static environment (static obstacles, ground, etc.) and the dynamic environment (moving stimulus). D processes the data set by transferring the trajectories to the estimated states $\mathcal{D} = \cup_n \mathcal{S}_n^* = \cup_n \cup_i \mathbf{s}_{i,n}^*$, where $\mathbf{s}_{i,n}^* = [\mathbf{p}_{i,n}^*, \mathbf{v}_{i,n}^*, \mathbf{v}_{i,n}^{d*}]$ denotes the state of agent i at time t_n of the dataset. The minimal magnitude and the maximal magnitude of $\mathbf{v}_{i,n}^*$ for all i and n are $v_{\min}^*$ and $v_{\max}^*$, respectively. For any $\mathbf{s}^* = [\mathbf{p}^*, \mathbf{v}^*, \mathbf{v}^{\mathrm{d}*}] \in \mathcal{D}$, $\mathbf{p}^* \in \mathcal{D}_{\mathrm{p}}$, $\mathbf{v}^* \in \mathcal{D}_{\mathrm{v}}$, $\mathbf{v}^{\mathrm{d}*} \in \mathcal{D}_{\mathrm{v}^d}$. I initializes each agent's state: position, velocity, and control direction. F is the main routine corresponding to our algorithm and computes a new state for each agent at each timestep.

3.2 Overall Approach

Our model for simulating heterogeneous multi-agent systems references the datasets to control the trajectories and behaviors of the agents (see Fig. 2). The datasets might be videos or other data representations, including trajectories or higher order features. We deal with different types of datasets and transform them into a unified representation, classifying the data by the magnitude of the velocity. The environment may also consist of static and dynamic obstacles. We initialize the position of each agent in the scene randomly and choose an initial velocity for each agent from our datasets. At each step of our simulator, we use an interactive optimization algorithm to make decisions for each agent. In particular, we solve this optimization problem by choosing a velocity from the datasets that tends to minimize our energy function. The energy function is defined based on the locomotion or dynamics rules of heterogeneous agents, including continuity of velocity, collision avoidance, attraction, direction control, and other constraints defined by users. In addition, our approach is general and can deal with different kinds of agents in the same way. We can capture corresponding motion characteristics with different datasets. As a result, we can simulate heterogeneous agents in the same physical space.

3.3 Dynamics Computation

An agent moves according to its surroundings, which include the other agents and the external environment (attractions, obstacles, roads, etc.). In these complex surroundings, the agent makes decisions in relation to all these elements. At each timestep, we calculate the state of each

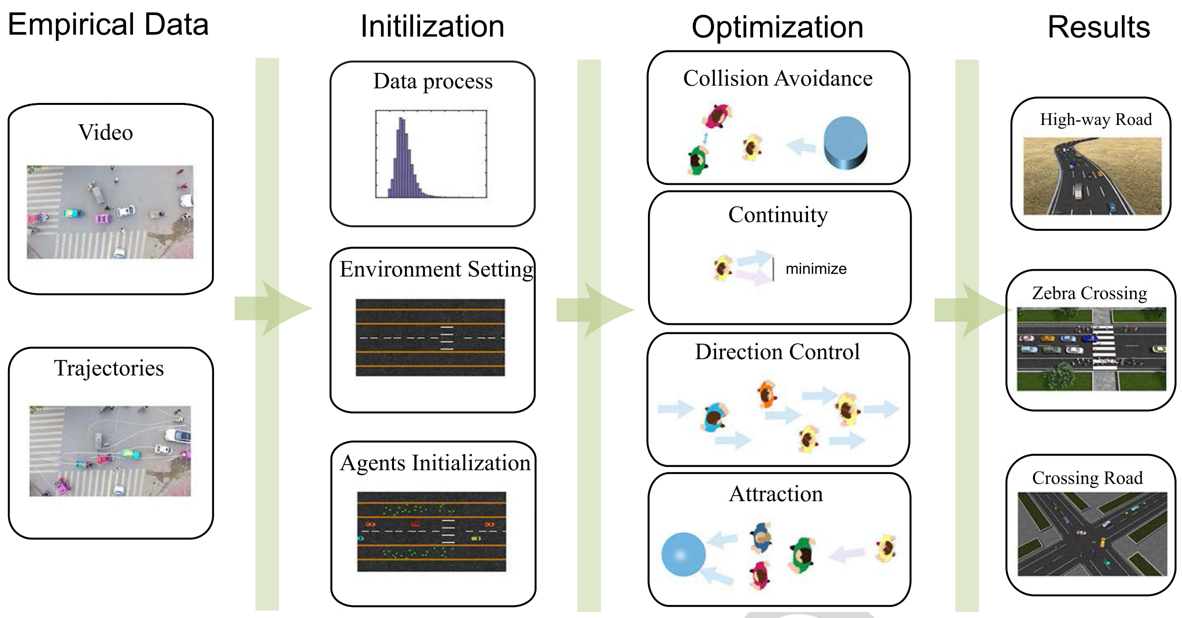


Fig. 2. Overview of our data-driven model for simulating heterogeneous multi-agent systems. We highlight different components of our algorithm. The input empirical data can be videos from a top-down view or trajectories of agents. In the initialization, we first transfer real-world data into a consistent format. With the data and environment information set by the users, we initialize the positions and velocities for agents. We treat the motion decision-making or local navigation process of each agent at every timestep as an optimization problem, and the energy function takes into consideration several factors: velocity continuity, collision avoidance, attraction, direction control, and any other constraints defined by users. Our model can simulate heterogeneous agents in the same scenario, including crowds, traffic, any combination of these agents, etc.

agent according to the prior states of all agents, the environment, and the dataset. Because the external environments may be time-varying, we set the environment evolution function as a function of time. The system of equations for the state of each agent at time t_n is

$$\begin{aligned} \mathbf{p}_{i,n} &= \mathbf{p}_{i,n-1} + \mathbf{v}_{i,n}\Delta t, \\ \mathbf{v}_{i,n} &= \underset{\mathbf{v}\in\mathcal{D}_{\mathrm{v}}}{\operatorname{argmin}}\ E(t_{n-1}, i, \mathbf{v}, \mathcal{S}_{n-1}, S(t_{n-1}, \mathbf{p}_{i,n-1}), \mathbf{v}^{\mathrm{d}}_{i,n}), \\ \mathbf{v}^{\mathrm{d}}_{i,n} &= R(\mathbf{p}_{i,n-1}, S(t_{n-1}, \mathbf{p}_{i,n-1})), \end{aligned} \tag{1}$$

where $E(t_{n-1}, i, \mathbf{v}, \mathcal{S}_{n-1}, S(t_{n-1}, \mathbf{p}_{i,n-1}), \mathbf{v}^{\mathrm{d}}_{i,n})$ is the energy function that chooses the optimal velocity for agent i at time t_n. $R(\mathbf{p}_{i,n-1}, S(t_{n-1}, \mathbf{p}_{i,n-1}))$ is a function that computes the control direction $\mathbf{v}^{\mathrm{d}}$ for each agent at each time. We compute a velocity that minimizes the energy function. If we search the velocity from a continuous-space, our method becomes an energy-based model. To capture the characteristics of different kinds of agents easily, we search for the velocity from the states in the dataset $\mathcal{D}$, which belongs to a discrete space. If the states generated from the dataset are unlimited, the simulation results will approximate those generated from the method with the continuous velocity space.

To simulate heterogeneous agents in the same physical space, we consider the common locomotion rules of multi-agent systems for the energy function $E(t_{n-1}, i, \mathbf{v}, \mathcal{S}_{n-1}, S(t_{n-1}, \mathbf{p}_{i,n-1}), \mathbf{v}^{\mathrm{d}}_{i,n})$ including collision avoidance, attraction, velocity continuity, direction control, and any other constraints.

$$\begin{aligned} &E(t_{n-1}, i, \mathbf{v}, \mathcal{S}_{n-1}, S(t_{n-1}, \mathbf{p}_{i,n-1}), \mathbf{v}^{\mathrm{d}}_{i,n}) \\ &= \sum_{k\in\theta} w_k E_k(t_{n-1}, i, \mathbf{v}, \mathcal{S}_{n-1}, S(t, \mathbf{p}_{i,n-1}), \mathbf{v}^{\mathrm{d}}_{i,n}), \end{aligned} \tag{2}$$

where $\theta = \{\mathrm{m}, c, a, d, s\}$, E_{m} is the energy for velocity continuity, E_{c} is the energy for collision avoidance, E_{a} is the energy for attraction, E_{d} is the energy for direction control, and E_{s} is the energy function for constraints of certain kinds of agents. w_{m}, w_{a}, w_{t}, w_{d}, and w_{s} are the weights of these terms respectively, and each weight represents the importance of the corresponding energy term. Velocity continuity is used to ensure that the agents move smoothly. Collision avoidance is a crucial part of multi-agent simulation. Attraction helps agents remain cohesive with other agents in the same group and has been widely used in multi-agent simulation literature [1]. The direction control represents the direction preference for agents according to the environment. These four elements can describe the basic factors considered by agents when moving. It is possible to add more constraints to control the movements of agents in E_{s}. The definition of E_{s} for each kind of agent is described in Section 5. To achieve the heterogeneity, our method uses different parameters and constraints to implement different dynamics, and use different road constraints and interaction domains to implement different traffic rules and response mechanisms.

3.4 Continuity

Because of the physical limitations, agents cannot change their motion states frequently or abruptly within a Δt time. Thus, the agent i has a tendency to choose a velocity close to $\mathbf{v}_{i,t}$ at a time $t+1$. The continuity energy is used to indicate that the agent tends to keep its velocity unchanged to save its overall energy:

$$E_{\mathrm{m}} = w_{m1} E^{\mathrm{dir}}_{\mathrm{m}} + w_{m2} E^{\mathrm{L}}_{\mathrm{m}}, \tag{3}$$

where $E^{\mathrm{dir}}_{\mathrm{m}} = \left\|\hat{\mathbf{v}}_{i,n-1} - \hat{\mathbf{v}}\right\|_2$ is for direction continuity and $E^{\mathrm{L}}_{\mathrm{m}} = \left\|\|\mathbf{v}_{i,n-1}\| - \|\mathbf{v}\|\right\|_2$ is for continuity of magnitude of velocity. $\mathbf{v}_{i,n}$ is the velocity of agent i at time t_{n-1}.

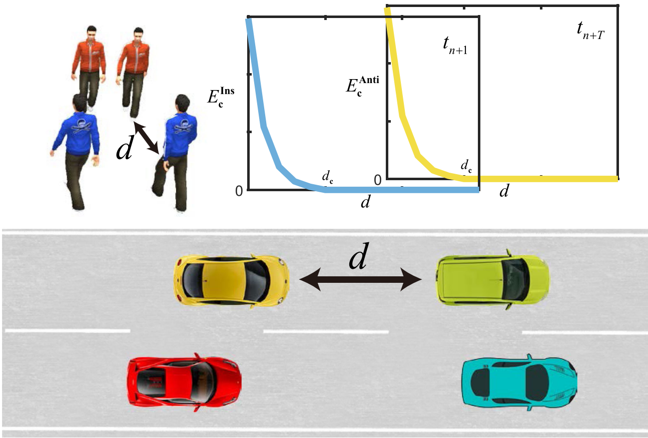


Fig. 3. Collision avoidance. In our method, the energy for collision avoidance E_c consists of two parts: the energy for instantaneous collision avoidance E_c^{Ins} and the energy for anticipation collision avoidance E_c^{Anti}. The blue curve represents E_c^{Ins} changes with the distance d between two agents increases in time $j+1$, and the yellow curve represents E_c^{Anti} changes with d in time $j+T$.

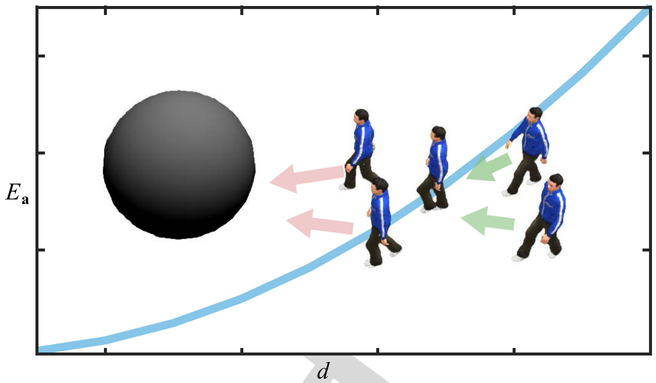


Fig. 4. Attraction. The energy for attraction include the energy for attraction (green arrows) between agents and the energy for attraction (red arrows) from environmental objects.

3.5 Collision Avoidance

Collision avoidance (Fig. 3) is a major issue in multi-agent simulation [3], [34]. To avoid collisions with other agents or the environmental obstacles in the scene, the agent should choose a velocity that will not cause a collision after one of more timesteps by assuming that all objects keep moving with their current velocities. Here, we consider two kinds of collisions to avoid: instantaneous collisions and anticipatory collisions.

$$E_c = w_{c1} E_c^{Ins} + w_{c2} E_c^{Anti}, \tag{4}$$

where instantaneous collision avoidance energy E_c^{Ins} only considers the possible collisions after a timestep, and anticipatory collision energy E_c^{Anti} considers the possible collisions after anticipation time T.

The normalized instantaneous collision avoidance energy E_c^{Ins} is given as

$$E_c^{Ins} = \frac{1}{|\Omega_c(\Delta t, i, t_{n-1})|} \sum_{Q \in \Omega_c(\Delta t, i, t_{n-1})} e^{d_c - d(\Delta t, \mathbf{s}_i, \mathbf{s}_Q, \mathbf{v})}, \tag{5}$$

where $\Omega_c(\Delta t, i, t_{n-1})$ is the predicted neighborhood of agent i after time Δt based on the surrounding information at time t_{n-1}. The neighborhood consists of agents that probably collide with agent i, and $|\Omega_c(\Delta t, i, t_{n-1})|$ represents the number of neighbors. $d(\Delta t, \mathbf{s}_i, \mathbf{s}_Q, \mathbf{v})$ is the predicted distance between agent i and agent Q. For each agent, we only consider collision avoidance within a distance d_c. Similarly, the anticipatory collision avoidance energy E_c^{Anti} can be given as

$$E_c^{Anti} = \frac{1}{|\Omega_c(T\Delta t, i, t_{n-1})|} \cdot \sum_{Q \in \Omega_c(T\Delta t, i, t_{n-1})} e^{d_c - d(T\Delta t, \mathbf{s}_i, \mathbf{s}_Q, \mathbf{v})}, \tag{6}$$

where $\Omega_c(T\Delta t, i, t_{n-1})$ is the predicted collision neighborhood of agent i after time $T\Delta t$ based on the surrounding information at time t_{n-1}. $d(T\Delta t, \mathbf{s}_i, \mathbf{s}_Q, \mathbf{v})$ is the predicted distance between agent i and agent Q after time T. Note that we perform instantaneous collision avoidance in each timestep while the anticipatory collision energy is only used to avoid potential future collisions.

Within the distance d_c, E_c increases when the distance between agent i and agent Q decreases (see Fig. 3). As a result, when we compute the velocity for each agent in each frame, a value making their distance larger will reduce the energy.

3.6 Attraction

If the agents want to move together as a group, we need to account for some attraction forces between them. The agent therefore prefers to choose a velocity that brings it closer to the group, allowing it to become a part of the group over the next few frames. In addition, agents may also be attracted by external stimuli. The attractions in our model include the attraction between the agents and the environment (Fig. 4). The attraction energy is given as

$$E_a = \frac{1}{|\Omega_a(\Delta t, i, t_{n-1})|} \sum_{Q \in \Omega_a(\Delta t, i, t_{n-1})} d^2(\Delta t, \mathbf{s}_i, \mathbf{s}_Q, \mathbf{v}), \tag{7}$$

where $\Omega_a(\Delta t, i, t_{n-1})$ is the predicted attraction neighborhood of agent i after time Δt based on the surrounding information at time t_{n-1}.

When the distance between agent i and agent Q increases, the energy E_a increases (see Fig. 4). Thus, a computed velocity making their distance smaller will reduce the energy.

3.7 Direction Control

We use direction control to imitate agents moving toward their goals. In this case, the agents try to choose velocities that point to their goals or that parallel the path to their goals. We assume that every agent has a goal position to guide its local movement. The goal might change over time. This goal can also be treated as a direction control defined by the users. The energy for direction control is presented as

$$E_d = \left\| \mathbf{v}_{i,n}^{d} - \hat{\mathbf{v}} \right\|_2, \tag{8}$$

where $\mathbf{v}_{i,n}^{d}$ is the control direction for agent i at time t_n.

4 Multi-Agent System Simulation with Data-Driven Optimization

In this section, we present more details about our method, as it is used to simulate heterogeneous agents.

4.1 State Estimation for the Dataset

The dataset of our method consists of trajectories that are time series of positions, $\mathcal{L}: \mathbf{Y}_1, \mathbf{Y}_2, ..., \mathbf{Y}_n....$ We estimate the state $\mathbf{s}_n^* = [\mathbf{p}_n^*, \mathbf{v}_n^*, \mathbf{v}_n^{\mathrm{d}*}]$ in the dataset based on these trajectories, and obtain the estimated position $\mathbf{p}_n^* = \mathbf{Y}_n$ and velocity $\mathbf{v}_n^* = \frac{\mathbf{Y}_n - \mathbf{Y}_{n-1}}{\Delta t}$. Estimating the control direction $\mathbf{v}_n^{\mathrm{d}*}$ is equivalent to estimating the direction to the corresponding agent's goal, according to Section 3.7. Therefore, if the agent only moves one way in the scenario, it is in the same control direction; if the agent changes its direction or goal in the dataset, we estimate its control direction at time t by computing the direction of its displacement, $\mathbf{v}^{\mathrm{d}*} = \frac{\mathbf{Y}_n - \mathbf{Y}_{n-\delta}}{\|\mathbf{Y}_n - \mathbf{Y}_{n-\delta}\|}$, which is computed every $\delta\Delta t$ time. We estimate the control direction by averaging every $\delta\Delta t$ time to reduce the estimation error from local avoidance. The results in Section 5 show that our state estimation for real-world datasets works well.

4.2 Direction Adaptation to Different Scenarios

According to Eq. (1), if we directly search the optimal velocity for each agent from the dataset, the synthesized scenario will be limited in its ability to achieve plausible movements by the scenario of the dataset. To eliminate these constraints, we map the local coordinate of the dataset to that of the scenario in the simulation by align their forward directions. As a result, we can simulate scenarios that may be different from the dataset. We suppose that the simulated scenario and the dataset have the same relative position relationship between the direction of velocity and the control direction; that is, the angle between the velocity direction and the control direction in the simulation is the same with that of the dataset, and

$$\begin{aligned} \hat{\mathbf{v}} \cdot \mathbf{v}^{\mathrm{d}} &= \hat{\mathbf{v}}^* \cdot \mathbf{v}^{\mathrm{d}*}, \\ \hat{\mathbf{v}} \times \mathbf{v}^{\mathrm{d}} &= \hat{\mathbf{v}}^* \times \mathbf{v}^{\mathrm{d}*}. \end{aligned} \tag{9}$$

Therefore, we obtain $\hat{\mathbf{v}}$, and $\mathbf{v} = \|\mathbf{v}^*\|\hat{\mathbf{v}}$.

4.3 Distance and Neighborhood

We hypothesize that the velocity of an agent remains unchanged over a short time t and the shapes of agents or obstacles cannot be ignored. If the agent i moves with the velocity $\mathbf{v}$ chosen from the dataset, the predicted distance between agent i and agent Q after time t becomes

$$d(t, \mathbf{s}_i, \mathbf{s}_Q, \mathbf{v}) = \|\mathbf{p}_i + \mathbf{v}t - (\mathbf{p}_Q + \mathbf{v}_Q t) - (R_i^{dir} + R_Q^{dir})\|_2, \tag{10}$$

where R_i^{dir} is the radius of agent i in the direction toward agent Q ($Q \neq i$). R_Q^{dir} is also a directional radius of agent Q. The shapes of different agents can be different. For example, we use a rectangular object to represent a car and a disc to represent a pedestrian. If Q is an entity in the environment, Eq. (10) becomes a distance function between an agent and the entity in the environment. For a twisting road, we compute the distance between two cars as the distance along the curve of the road.

In contrast to the existing methods [35], the agents in our method try to avoid collisions with not only the homogeneous agents but also the heterogeneous agents. To avoid collisions, each agent tries to keep away from other agents or obstacles when they get too close. In the real world, humans can perceive the environment through both vision and sound [36], and thus we can assume that an agent can avoid collisions in a full field of vision with a limited range. We define the neighborhood for collision avoidance as

$$\Omega_{\mathrm{c}}(t, i, t_n) = \{Q | d(t, \mathbf{s}_i, \mathbf{s}_Q, \mathbf{v}) < d_{\mathrm{c}}, Q \in \mathcal{G} \setminus \{i\} \cup \mathcal{G}_{\mathrm{c}}\}, \tag{11}$$

where d_{c} is the threshold distance for collision avoidance and $\mathcal{G}_{\mathrm{c}}$ is the set of obstacles in the scenario. Each agent considers collision avoidance with the agents or obstacles within a distance d_{c}. Meanwhile, each agent tries to keep close to the agents in its group or to the external attraction stimulus if the distance between the agents is large. We define the neighborhood for attraction as

$$\Omega_{\mathrm{a}}(t, i, t_n) = \{Q | d(t, \mathbf{s}_i, \mathbf{s}_Q, \mathbf{v}) > d_{\mathrm{a}}, Q \in \mathcal{G} \cup \mathcal{G}_{\mathrm{a}}\}, \tag{12}$$

where d_{a} is the threshold distance for attraction and $\mathcal{G}_{\mathrm{a}}$ is the set of attraction in the scenario. An entity that is treated as an attraction can also be an obstacle if the shape of it cannot be ignored, that is, $\mathcal{G}_{\mathrm{c}} \cap \mathcal{G}_{\mathrm{a}} \neq \varnothing$.

4.4 Faster Computation

If we use a brute force method to solve Eq. (1), the computation cost will be large. The underlying time complexity will be $O(n^2 m)$ with n agents in the simulation and m estimate states in the dataset. The most time-consuming parts are searching for the optimal velocity from the dataset and finding the neighborhood for each agent. To achieve interactive performance, we propose two acceleration methods.

4.4.1 *Reduced Solution Space*

To find the optimal velocity for each agent efficiently, we reduce the solution space of Eq. (1). We classify the estimated states of the dataset into groups based on the magnitude of the velocity. Considering the continuity of motion, we search for the velocity for each agent in the current group of velocities and in the adjacent groups,

$$\mathbf{v}_{i,n+1} \in \bigcup_{m=l-z}^{l+z} \{\mathbf{v}^{n*}\}, \tag{13}$$

where $\{\mathbf{v}^l\}$ is the set of velocities of the group l to which $\mathbf{v}_{i,n}$ belongs, z is the scope of the number of groups that are considered for computing optimal velocity, and the group $\{\mathbf{v}^m\}$ with $m \in [l-z, l+z]$ is the neighborhood of $\{\mathbf{v}^l\}$.

4.4.2 *Grid in Space*

To reduce the time consumption for computing the neighborhood for each agent, we introduce the idea of grid in space from fluid simulation [37]. For our simulation, the 2D plane is divided into 2D grids. We suppose that $\mathcal{O}_{x,y}$ denotes the set of all agents in the grid $O_{x,y}$. Then the candidate neighborhood of i in grid $O_{x,y}$ is reduced from $\mathcal{G}$ to $\mathcal{G}'$,

$$\mathcal{G}' = \bigcup_{k_1=x-1}^{x+1} \bigcup_{k_2=y-1}^{y+1} \mathcal{O}_{k_1,k_2}. \tag{14}$$

When we search the neighborhood for collision avoidance, we compare the distances of the agents in the grid $O_{x,y}$ with

TABLE 1
Performance for Different Scenarios

Scenario	Types	Behavior	N	Dataset	Time(s/f)
Crowd-1	human	walking on street	8-148	[Lerner et al. 2007]	0-0.0040
Crowd-2	human	mixture of two crowds	100	[Zhang et al. 2012]	0.0209
Crowd-3	human	avoiding static obstacles	79	[Zhang et al. 2012]	0.0192
Traffic-1	car	movements on a twist road	80	[NGS 2013]	0.0137
Traffic-2	human/car	movements on a crossing road	30/35	[NGS 2013]/[Zhang et al. 2012]	0.0378
Traffic-3	human/bicycle/tricycle/car	mixture of multiple systems	25/15/10/40	video from Shandong, China	0.0342

We summarize the characteristics of the simulation scenarios in this paper. The agents include humans, cars, bicycles, and tricycles. The datasets used for input data vary. We use seconds per frame to measure the time performance of the simulations. Our method can achieve realtime performance using 4 cores on a CPU.

the agents adjacent to this grid instead of comparing them to all the agents in the scenario.

5 Results

In this section, we highlight the performance of our approach in generating simulations of crowds, traffic, and combinations of different types of agents. We have implemented our approach in C++ on a desktop machine with a 3.30 GHz Inter Xeon CPU E3-1230 v3 4-core processor and 32 GB memory. The performances for different scenarios are given in Table 1.

To achieve the heterogeneity of our simulation system, we use different parameters and E_{s} for different kinds of agents. In addition, we employ real-world datasets consisting of pedestrians, bicycles, tricycles and cars captured from real scenarios. We initialize the weights with 1.0, and they can be tuned according to the behaviors of the agents. Table 2 shows the weights of all the benchmarks. We define the user control for each pedestrian with speed control $E_{\mathrm{s}} = E_{\mathrm{sc}} = |\|\mathbf{v}\| - v_i|$, where v_i is the ideal speed for agent i. We define the user control for each car with speed control and position control $E_{\mathrm{s}} = E_{\mathrm{sc}} + E_{\mathrm{p}}$, where E_{s} is the same with that of each pedestrian, $E_{\mathrm{p}} = |\mathbf{v} \cdot (\mathbf{v}^{\mathrm{d}})^{\perp}|$, and $\mathbf{v}^{\mathrm{d}}$ is a tangential vector of the given lane. Cars try to drive in the middle of the lane.

5.1 Data Acquisition

Our method accepts different kinds of input datasets if those datasets contain the velocity information for the agents. Any form of discontinuity or a small amount of abnormal data in the datasets is acceptable.

In our current framework, we have used some widely available datasets from different scenarios and environments. The datasets for crowd simulation include two scenarios: one is from [38] and features two-dimensional bidirectional movements with 304 pedestrians and 1,273 frames; the second is from [29] and features street scenarios with 8-148 pedestrians and 9,014 frames. We set the control directions for the first dataset as the directions that point to the agents' destinations. For the second dataset, the control direction of one agent at a certain time is the direction that points from its current position to the position of its next record.

The traffic dataset is extracted from the Next Generation Simulation (NGSIM) datasets [39], which include detailed, high-quality highway traffic datasets. We extract 300 frames and 161 cars in total. We set the direction of the road as the estimation of the control directions of the cars. The datasets corresponding to the mixed traffic scenarios (including pedestrians, bicycles, tricycles, and cars) are generated from videos. The video was recorded in Shandong, China. We use the optical flow tracking method [40] to trace the agents. The extracted data consists of 435 frames and contains 3 pedestrians, 10 bicycles, 10 tricycles, and 2 cars. The control direction for each agent in every frame is computed by averaging the directions of the agent from 30 frames.

5.2 Human Crowd

We simulate three benchmark scenarios with crowds representing each pedestrian as a disc with a fixed radius.

Crowd-1. We simulate behaviors of pedestrians on a street with the dataset from [29] to show that our method

TABLE 2
The Weights for Simulation

Scenario		$E_{\mathrm{t}}^{\mathrm{dir}}$	$E_{\mathrm{t}}^{\mathrm{L}}$	E_{c}^{Ins}	$E_{\mathrm{c}}^{\mathrm{Anti}}$	E_{a}	E_{d}	E_{p}	E_{sc}
Crowd-1		1.0	1.0	1.0	1.0	0.0	1.0	0.0	0.5
Crowd-2		1.0	1.0	1.0	1.0	0.0	1.0	0.0	1.5
Crowd-3	obstacle zone	1.5	1.0	0.67	0.67	0.0	1.0	0.0	1.0
	before obstacle zone	1.0	1.0	0.67	0.67	0.0	1.0	0.0	1.0
	after obstacle zone	1.0	1.0	0.67	0.67	0.0	1.5	0.0	1.0
Traffic-1		0.5	0.5	1.0	1.0	2.0	3.0	10.0	10.0
Traffic-2	Pedestrian	1.0	1.0	1.0	1.0	0.0	1.5	1.0	10.0
	Car	5.0	1.0	1.0	1.0	2.0	5.0	1.0	10.0
Traffic-3	Type-1	10.0	1.0	1.0	1.0	0.0	5.0	10.0	5.0
	Type-2	0.5	0.5	1.0	1.0	2.0	3.0	1.0	10.0

This table gives the weights for the direction continuity $E_{\mathrm{t}}^{\mathrm{dir}}$, the speed continuity $E_{\mathrm{t}}^{\mathrm{L}}$, instantaneous collision avoidance E_{c}^{Ins}, anticipated collision avoidance E_{c}^{Anti}, attraction E_{a}, direction control E_{d}, position control E_{p}, and speed control E_{sc} in each scenario.

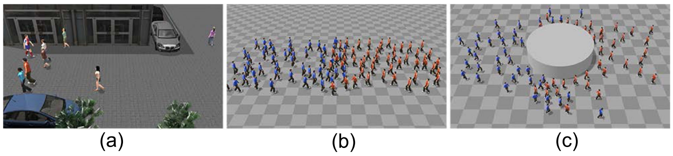


Fig. 5. The mixed crowds with different control directions. (a) Pedestrians with changing control directions walk on a street. (b) Two crowds with inverse control directions. The pedestrians with the same clothes represent individuals in the same crowd. The crowds walk to their own destinations while avoiding collisions with each other. (c) We add an obstacle to the scenario. In addition to avoiding collisions with each other, crowds should also avoid collisions with this obstacle.

can reproduce a scenario from the dataset. In this scenario, we set the number of agents in the initialization and control directions to be the same as those in the dataset. Pedestrian agents, represented as discs, mainly avoid collisions with other pedestrians that are close to them in the scene (see Fig. 5a).

Crowd-2. In this scenario, we simulate two groups (50 pedestrians in each group) with control directions inverse to those from the dataset [38]. We randomly locate the agents in each group at one side of the road and randomly choose a velocity for each agent from the dataset in the initialization. The control direction points from the agent's position to the agent's goal on the other side of the road. The reference speed is the magnitude of the initial velocity. Agents are attracted to those in the same group and avoid collisions with other agents, including pedestrians in other groups (see Fig. 5b).

Crowd-3. Based on the benchmark *Crowd-1*, we add a cylindrical obstacle in the center of the road (see Fig. 5c). We also use the dataset [38] in this benchmark. The initialization method for this benchmark is the same as for the benchmark Crowd-2. In our simulations, we set different control directions for different groups and agents in the same group share the same control direction. Agents avoid the obstacle like they avoid other agents.

Crowd behaviors can be slightly adjusted by setting different parameters. In Crowd-1, the control directions of agents are changing, and thus we decrease the weight of $E_{\rm sc}$ to 0.5 to weaken the speed control so that agents can promptly adjust their directions. In Crowd-2, some agents in high-density areas may stop to avoid potential collisions when two crowds are joining, and we increase the weight of $E_{\rm sc}$ to 1.5 to enhance speed control so that these agents can return back to their desired speeds quickly. For a scenario with obstacle such as the one in Crowd-3, the agent-agent and agent-obstacle collision avoidances will make the collision energies $E_{\rm c}^{\rm Ins}$ and $E_{\rm c}^{\rm Anti}$ much larger than other energy terms. To weaken the influence of collision avoidance, we empirically decrease the weights of $E_{\rm c}^{Ins}$ and $E_{\rm c}^{\rm Anti}$ to 0.67. To adjust the weights of $E_{\rm t}^{dir}$ and E_d, we divide the whole road into three zones for each crowd: (1) *obstacle zone*: the area whose distance to the obstacle is about $2m$ (an empirical value); (2) *before obstacle zone*: the area before a crowd arrives at the *obstacle zone*; (3) *after obstacle zone*: the area after a crowd passes by the *obstacle zone*. In the *obstacle zone*, we increase the weight of $E_{\rm t}^{\rm dir}$ to 1.5 to enhance the direction continuity in order to weaken drastic direction changes for agents in high-density areas. In the *after obstacle zone*, we increase the weight of $E_{\rm d}$ to 1.5 to enhance the direction control so that the agents can quickly return back to their goals after they pass by the obstacle.

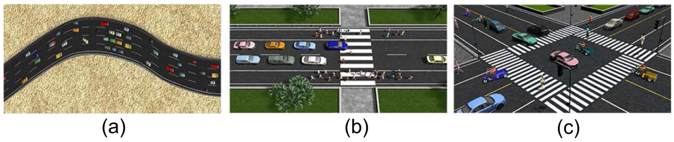


Fig. 6. Traffic simulation. (a) Traffic on a twisting 4-lane highway. (b) A combination of cars and crowds. Some pedestrians are walking on the sidewalk. Cars can be treated as obstacles for crowds and vice versa. (c) Congested traffic in an urban crossroad with a traffic lights.

5.3 Traffic

In traffic simulations, vehicle-agents mainly interact with the cars that are adjacent to them in the same lane, avoiding collisions when they are too close and being attracted by the leader cars when the distance to that car becomes too large. However, cars that are changing lanes also interact with the adjacent cars in the target lanes. The control directions for the cars in traffic are the directions of the lanes to which they currently belong.

Traffic-1. With our method, we can simulate traffic on twisting roads with the straight high way traffic dataset [39] (see Fig. 6a). During the initialization step, 80 cars are distributed on the road. The distance between two adjacent cars is chosen randomly from the dataset. We also randomly select the magnitude of the velocity for each agent from the dataset, and the direction of the velocity is the same as the direction of the road on which the agent is driving. The control direction of each agent is always the direction of the road. In this benchmark, the directions of agents in different positions on the twisting road vary.

Our method is general, so we can mix different kinds of agents in the same scenario. In this section, we show two benchmarks: a zebra striped crosswalk and a crossroad with traffic lights.

Traffic-2. In this benchmark, we simulate a case in which people want to cross the road (see Fig. 6b). We use dataset [38] for the crowd and dataset [39] for the traffic. Each pedestrian has a certain possibility of crossing the road. Once the pedestrian starts to cross road, the control direction becomes perpendicular to the road direction and the pedestrian needs to avoid not only other pedestrians, but also the cars around it. At the same time, the surrounding cars need to stop if the pedestrian is in front of them, and the attractive force from the leading cars disappears for these cars. We implement these interactions by adding corresponding objects to the interaction domain of agents.

Traffic-3. Our model can handle congested scenarios with different or heterogeneous agents. Here we simulate agents (15 pedestrians, 15 bicycles, 10 tricycles, and 40 cars) crossing a congested road with a traffic light (see Fig. 6c). We classify the dataset into groups according to the corresponding type of agent in the original data and choose the velocities of the agents from the corresponding class. Furthermore, we classify the four kinds of agents into two types with different motion constraints. The first type includes pedestrians and bicycles, which can overtake the agents in front of them in the same lane. The second type includes tricycles and cars, which cannot overtake the agent in front in the same lane.

Fig. 7. The avatar in a VR scenario. (a) We provide the user with an immersive VR experience from a first-person perspective with HTC Vive. (b) The avatar drives a car on an high-way road. (c) The avatar drives a car on an urban traffic road. (d) The avatar is walking on the sideroad. (e) The avatar is walking on the crosswalk.

When an agent reaches the crossing, the control direction is the interpolation of the original road direction and the target road direction. The rule for traffic light is not strictly the same as that in the real world. We treat the red traffic light as an obstacle, and agents will gradually stop when they are close to the red traffic light.

5.4 VR Scenarios

Our method can be applied to VR scenarios. We model the user as an avatar in the VR scenario with a first-person perspective (see Fig. 7) (a). The user can sit in a car and observe the movements of other cars around it (see Fig. 7b and 7c). As a walker, the user can also see the traffic flow and other pedestrians at the roadside (see Fig. 7d and 7e).

6 Analysis

6.1 Time Performance

To test the time performance of our method, we simulate a crowd in a scenario with the size of 1,000*1,000. There is no obstacle in the scenario. During the initialization, we randomly locate N agents at random positions. The initial velocities of the agents are randomly copied from the dataset [38]. We set the grid size of the simulation as 10, and the z for Eq. (13) as 2.

In our method, we utilize spatial continuity and velocity continuity to reduce possible collisions among the agents. We use the size of the solution space of the optimization function in Eq. (1) to improve the runtime performance of our simulation. We divide the space into grids and each grid records the agents that belong to it. When we search for the neighbors of each agent, we only need to search the grid to which the agent belongs and the grids that are adjacent to this grid. As a result, our method can reduce the time consumption for multi-agent simulations dramatically (see Fig. 8a).

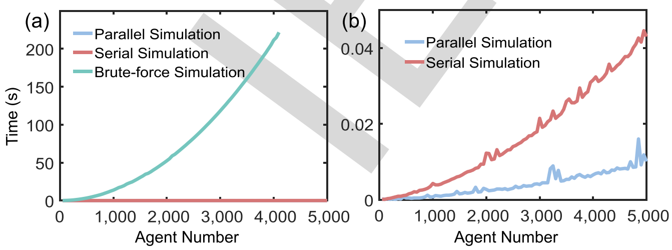


Fig. 8. Time performance. We take a crowd as an example to analyze the time performance of the simulation. (a) We compare the time performance of the brute-force method and our method. With our two search methods, we can improve the performance with 32,298x speedup for 4000 agents. (b) We compare the performance of an 8-threaded parallel implementation with a single-threaded implementation. With parallel computing, as the number of agents increases, the simulation time increases much more slowly. Our method can even simulate 5,000 agents in realtime on a PC machine with a 4.00 GHz Intel i7-6700k CPU processor and 16 GB memory.

Because we can solve the optimization problem for each agent at the same time, we can also easily parallelize our method. Taking the crowd as an example, we compare the time complexity of our simulation using a serial implementation against a parallel implementation (see Fig. 8b). Our parallel implementation can simulate more than 5,000 agents in realtime on a multi-core processor with four cores.

To evaluate the performance of our method further, we compute the running time (seconds per frame) of all the simulation results mentioned in this paper (see Table 1). Our method can achieve real-time performance in various scenarios with multiple kinds of input dataset. The time complexity is not only related to the number of agents in the simulation, but also to the number of classes and the number of data points in each dataset. As a result, similar scenarios with the same number of agents may have different time performances.

6.2 Comparisons

6.2.1 *Statistical Comparisons*

To demonstrate the plausibility of our method, we compare our simulation results (crowds and highway traffic) with given datasets in terms of the distributions of velocities and distances (the distance to the nearest agent). Velocity is a basic factor used to describe the motion, and minimal distance is the factor used to describe density. We use dataset [38] for two-dimensional bidirectional movements to compare our results with [12], which is the state-of-art optimization method for crowd simulation. Meanwhile, we use the dataset [39] on a four-lane highway to compare our results with [7], which is the state-of-art data-driven traffic simulation method.

Comparison for Crowds. We simulate bidirectional movements of pedestrians in a narrow corridor with the method described in [12] and our method. During the initialization, we set the same number, positions, and velocities of agents as in the dataset. For method [12], the minimal and maximal velocities and the minimal distance from neighbors are estimated from the dataset. Other parameters inherit the configuration of the open source code released by the authors. We also tune parameters so that the method can work well for the scenario. For our method, the control direction of each agent is the direction that points from the current position to the agent's destination. The weight $\mathbf{w} = \{0.8, 1.15, 1.2, 0.8, 0, 0.85, 0, 1.2\}$, which corresponds to the items in Table 2. For both methods, the preferred speed of each agent is the average speed of the corresponding agent in the dataset.

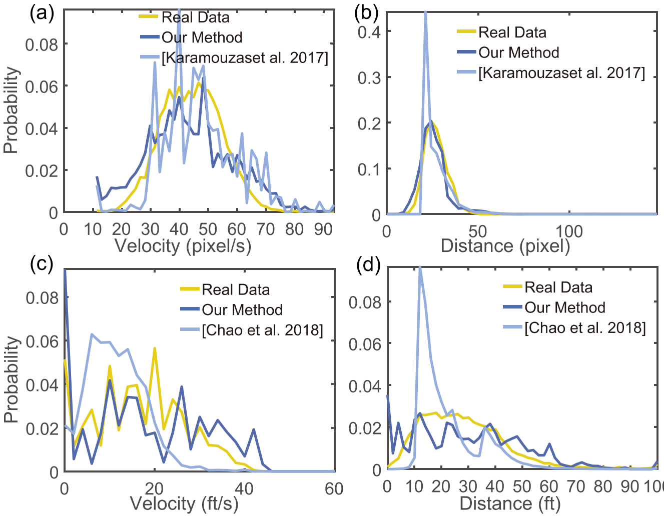


Fig. 9. The distributions of velocity and minimal distance. We compare the probability distributions between our simulation results, existing methods, and input datasets. (a)-(b). The comparison for the crowd simulated. (c)-(d) The comparison for traffic simulated on a straight 4-lane road.

Comparison for Traffic. We simulate traffic in a straight four-lane highway like the dataset [39] using both method [7] and our method. In this comparison, we initialize the number, positions, and velocities of agents in our method to be the same as the dataset. The control direction is the direction of the road. The weight $\mathbf{w} = \{1.0, 5.0, 1.0, 1.0, 20.0, 3.0, 1.0, 0.0\}$, which corresponds to the items in Table 2. For method [7], we set the parameters to be the same as the original parameters. The traffic in method [7] consists of 15 traffic flows. The initialization of each flow is same as in the dataset.

The distributions of velocity and minimal distance for each method are shown in Fig. 9. We compute the difference between simulation results and the dataset as the scores for each method. We divide all the values of each metric into 30 intervals and compute the probability for each interval. The difference between simulation results and the dataset is the sum of the magnitudes of the probability difference in the intervals. The scores of each method are given in Table 3.

The distributions of velocity and minimal distance in our simulations are closer to those in the input data. Although our method selects velocities for agents directly form the dataset, the selected velocities are controlled by Eq. (1). Because our method has velocity distributions that are closer to the input data for crowd simulation, our approach is better at capturing the motion characteristics of a multi-agent system as compared to prior methods ([12], [7]).

In the compared methods ([12] and [7]), the spikes in the distance distribution are not only due to the hard constraints on the separation distance, but also the optimization functions of these methods trying to find similar optimal velocities for different agents. Therefore, the distances of different agents are similar when agents reach the balance of different optimization terms.

6.2.2 Trajectory Comparisons

Several quantitative metrics can be used to compare real data against simulation data [13], [41], [42]. To evaluate the time series in sequence of agents' movements for crowds, we employ the absolute difference metric (ADM) and the path length metric (PLM) proposed by Wolinski et al. [13] as they are straightforward comparing to other quantitative metrics. We simulate the movements of pedestrians on a street using the implicit method [12], the data-driven method (PAG) [19], and our method. We set the same number, positions, and velocities of agents as in the reference dataset [29] when performing the initialization. In addition, we set the control directions to be the same as those in the dataset.

The ADM and PLM for each method are shown in Table 4. Experiments show that our method achieves a lowest score compared to [12] and [19] for crowds. This means that the trajectories generated by our method are more realistic than those generated by methods of [12] and [19]. Compared to the implicit method [12] which is not data-driven, our approach uses real datasets so that it can generate more realistic detailed behaviors. The PAG method [19] searches trajectories only depending on the predicted temporal perception patterns and the distance to the goal, which may produce potential discontinuous velocities. On the contrary, our method can enforce continuous velocity by introducing a velocity continuity energy function.

TABLE 3
Benchmark Scores 1: Used to Measure the Statistical Closeness to the Real-World Datasets

	Velocity			Distance		
	Real	Ours	Others	Real	Ours	Others
Crowd	0.0	0.4132	0.4793	0	0.2691	0.5913
Traffic	0.0	0.2507	0.3766	0	0.2383	0.3475

The scores are the difference between simulation results and the dataset. A lower score for our method versus [12] for crowds and [7] for traffic. This demonstrates that the trajectories and behaviors generated by our method are closer to those generated by prior methods.

6.3 Our Simulation Results with or without Using Dataset

To explore the performance of our data-driven scheme, we compare our simulation results with and without using dataset in terms of the distributions of velocities and minimal distances. We use the dataset [39] on a four-lane highway for our experiments. We use the same initialization method and parameter values as those in Section 6.2. For the method without using dataset, we suppose that the cars move in one direction and compute $\mathbf{v}_{i,n}$ ($\|\mathbf{v}_{i,n}\| \in [v^*_{\min}, v^*_{\max}]$) by minimizing Eq. (2). The underlying assumption is that the minimum and maximum magnitudes of velocities from real-world datasets are reasonable values to restrict the range of the magnitude of velocity.

The distributions of velocity and minimal distance for the comparison are shown in Fig. 10. The velocity difference to

TABLE 4
Benchmark Scores 2: Used to Measure the Trajectory Closeness to the Real-World Datasets

	Real	IMPLICIT	PAG	Ours
ADM	0.0	37.373	65.4278	3.10986
PLM	0.0	20.3423	117.486	3.9529

The scores show the differences between the simulation results and the real-world dataset.

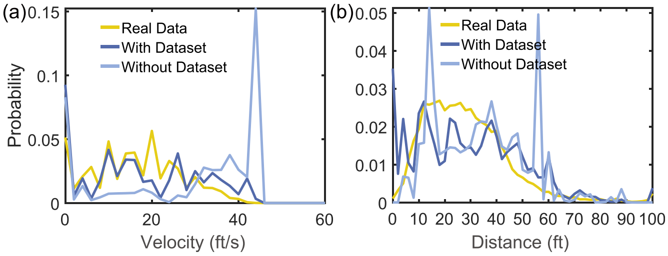


Fig. 10. The distributions of velocity and minimal distance for comparison of the results with and without using dataset. (a) Probability distributions of velocity. (b) Probability distributions of minimal distance.

the dataset of our method using dataset (0.2507) is smaller than that of the method without using dataset (0.6132). The minimal distance difference score of our method using dataset (0.2383) is also smaller than that of the method without using dataset (0.2649). The comparison results indicate that the data-driven scheme can improve the plausibility of simulation results.

7 User Studies and Evaluation

We conduct two user studies to evaluate the plausibility of our method and one user study to show a better user experience through VR. The weights for the user study are shown in Table 5. The eight cases in the first user study are conducted from an overhead view to show the agents' movements. In the second user study, we adopt the agent's view in each case, meaning that the view is closer to that of a participant in his/her daily life. In the third user study, we compare the results as shown in immersive VR and those shown on a desktop in four different scenarios or agents' views.

Experiment Goals & Expectations. For the first user study, we hypothesize that the results simulated by our method will exhibit more plausible movements than prior multi-agent methods. For the second user study, we hypothesize that our method results in a better user experience than the prior methods. Therefore, participants will significantly prefer our method over the prior methods in these evaluations. In the third user study, we hypothesize that the results shown in VR can produce a better user experience that those shown on a desktop.

Comparison Methods. For crowd simulation, we compare our method with the method in [12] which is a state-of-art physical-based method for crowd simulation. We also use the dataset [29] in crowd simulation. For traffic simulation, we compare our method with the method in [7], which is a state-of-art data-driven method on traffic simulation. Here we use the dataset [39]. All 2D trajectories generated from simulation methods or extracted from datasets are assigned to 3D characters. We also compare mixed traffic results shown in VR and those shown on a desktop.

Environments. In the first and second user study, we used three scenarios for crowd simulation. The scenario with the dataset [29] is in a street with 18 agents. The other two scenarios are the one in which two crowds (100 agents in total) encounter each other and the scenario in which 36 agents are located on a circle moving towards the opposite positions. We also use three scenarios for traffic simulation. The scenario with the dataset [39] is on a straight 4-lane road with 156 agents. The other two scenarios are on a twisting 2-lane road with 80 agents and on a twisting 4-lane road with 200 agents. In the third user study, we use one instance for the scenario with 50 cars and a car's view. We also use three instances for the scenario with 35 cars and 30 pedestrians. In each instance, we use different agent views: one from a car's view, one from the view of a pedestrian walking on a zebra crossing, and one from the view of a pedestrian walking on a sidewalk. In the VR scenarios, head turning is controlled by a HTC Vive headset, and the user is allowed to turn his/her head freely with a fixed position in a moving agent.

TABLE 5
The Weights for the User Study

		E_t^{dir}	E_t^{L}	E_c^{Ins}	E_c^{Anti}	E_a	E_d	E_p	E_{sc}
Street		1.0	1.0	1.0	1.0	0.0	1.0	0.0	0.5
Hallway		1.0	1.0	1.0	1.0	0.0	1.2	0.0	1.2
Circle		1.0	1.0	0.5	0.5	0.0	1.0	0.0	1.0
Straight		0.5	0.5	1.0	1.0	1.2	3.0	1.0	0.2
Twist-2Lane		0.5	0.5	1.0	1.0	1.0	3.0	1.0	2.0
Twist-4Lane		0.2	0.6	1.0	1.0	2.0	3.0	10.0	1.0
VR-2Lane		5.0	1.0	1.0	1.0	2.0	5.0	1.0	10.0
VR	Pedestrian	1.0	1.0	1.0	1.0	0.0	1.5	1.0	10.0
	Car	5.0	1.0	1.0	1.0	2.0	5.0	1.0	10.0

This table gives the weights for the direction continuity E_t^{dir}, the speed continuity E_t^{L}, instantaneous collision avoidance E_c^{Ins}, anticipated collision avoidance E_c^{Anti}, attraction E_a, direction control E_d, position control E_p, and speed control E_{sc} in each scenario.

Experimental Design. We conduct the user studies based on a paired-comparison design. For the scenarios with a dataset, we design two comparison pairs: the dataset versus our method, and the dataset versus the prior method. We design one comparison pair for each scenario without a dataset: our method versus the prior method. For each pair, we show two pre-recorded videos in a side-by-side comparison. The order of the scenarios was random. The position (left or right) of each method was also random. For the scenarios for VR versus desktop comparison, we ask the participants to answer the questionnaire after see the scenarios via VR and the scenarios via desktop.

Metrics. In each user study, participants were asked to choose a score using a 7-point Likert scale, in which 1 means that the result presented on the left is strongly plausible, 7 means that the result presented on the right is strongly plausible, and 4 means no preference for either method. To combine the user study results in the same scale, we transfer the score for each method to a certain side when we deal with the scores.

7.1 User Study with an Overhead View

The user studies for crowd simulation and traffic simulation with an overhead view were completed by 26 participants (15 females and 11 males). We performed two-sample t-tests for the scenarios with datasets (one for crowd simulation and another for traffic simulation). We hypothesize that the mean value of our method is bigger than that of the prior method. Meanwhile, we performed one-sample t-tests for the scenarios without datasets (two scenarios for crowd simulation and two for traffic simulation), hypothesizing that the mean value of our method is bigger than 4, which

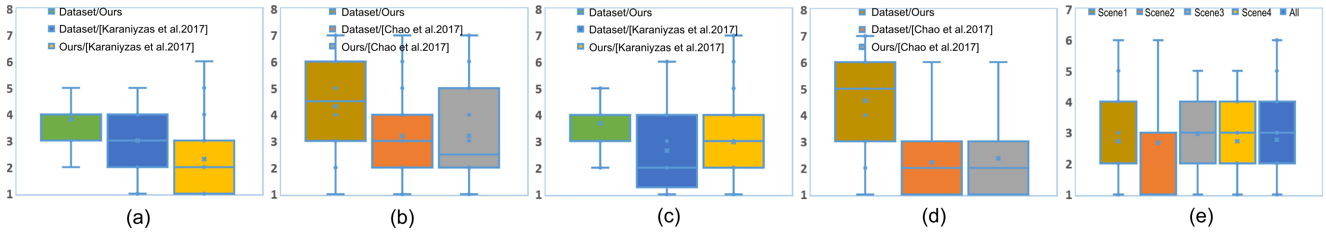


Fig. 11. Plausibility scores of the user study. We use a 7-point Likert scale to measure the plausibility of the methods. The lower the score, the more the participants prefer the method on the left; the higher the score, the more the participants prefer the method on the right. (a)The statistics for crowd simulation with an overhead view. Participants cannot tell the difference between the dataset and our method. Compared to method [12], the participants think the results of our method are more plausible. (b) The statistics for traffic simulation with an overhead view. Our method gets a higher score than method [12] when compared with the dataset. We also get better results in the user study with the dataset. (c) The statistics for crowd simulation from an agent view. Our method is closer to the dataset. The participants believe that the results of our method are more plausible than those of the prior method. (d) The statistics for traffic simulation from an agent's view. Our method has a significantly larger score than method [7] in the user study with the dataset. Our method also shows better performance in the user study without the dataset. (e) The statistics of the user study for the comparison of VR and desktop. The scores are transferred so that VR is supposed on the left. The scenarios shown through VR have better scores.

indicates no difference. Overall, participants believed that our method was more plausible than the compared methods for both crowd simulation and traffic simulation. Fig. 11 (a)-(b) shows details about the scores for each comparison.

User Study for Crowd Simulation. For the scenario with the dataset, our method's mean score is significantly larger than the prior method's mean plausibility score ($t(25) = 2.9111$, $p = 0.0027 < 0.01$). For the scenarios without datasets, our method's mean score shows a significant difference from the hypothetical mean ($t(51) = -8.7555$, $p < 0.001$).

User Study for Traffic Simulation. For the scenarios with datasets, our method's mean of the score is significantly larger than the prior method's mean plausibility score ($t(25) = 2.4422$, $p = 0.0091 < 0.01$). For the scenarios without datasets, our method's mean score shows a significant difference from the hypothetical mean ($t(51) = -3.0169$, $p = 0.002 < 0.01$).

7.2 User Study with an Agent View

The user studies for crowd simulation and traffic simulation from an agent's view were completed by 28 participants (17 females and 11 males). For the user study from an agent view, we also performed two-sample t-tests for the scenarios with datasets hypothesizing that our method has a larger mean score than the prior method. For the scenarios without datasets, we performed one-sample t-tests hypothesizing that the mean value of our method is larger than 4 (no difference). Overall, participants also judged that our method is more plausible than the prior methods. The statistics of the participants' plausibility evaluations can be found in Fig. 11 (c)-(d).

User Study for Crowd Simulation. For the scenario with a dataset, the mean plausibility score of our Heter-Sim is significantly larger ($t(27) = 2.6692$, $p = 0.005 < 0.01$) than the method [12]. The mean score of our method has a significantly superior to the hypothetical mean ($t(55) = -5.0281$, $p < 0.001$) for the scenarios without datasets.

User Study for Traffic Simulation. For the scenario with a dataset, the mean score of our method is significantly larger than the mean score of the prior method ($t(27) = 6.4890$, $p < 0.001$). For the scenarios without datasets, the mean score of our method shows a significant difference from the hypothetical mean with $t(55) = -8.0381$ and $p < 0.001$.

7.3 User Study via VR or Desktop

The user studies for the comparison between VR and desktop were taken by 28 participants (14 females and 14 males). We performed one-sample t-tests for the four instances by hypothesizing that the mean score of VR is bigger than 4 (no difference). Overall, participants believed that the results shown with VR are more plausible than those shown with a desktop. Fig. 11e shows the details about the scores for each comparison. In each scenarios, the score of VR is significantly better than that of desktop. $t(27) = -5.0138$, $p < 0.001$ for the first scenario, $t(27) = -4.16478$, $p < 0.001$ for the second scenario, $t(27) = -3.9890$, $p < 0.001$ for the third scenario, and $t(27) = -5.7564$, $p < 0.001$ for the last scenario. In total, the mean score for VR shows a significant difference from the hypothetical mean ($t(111) = -9.3485$, $p < 0.001$).

8 Conclusion, Limitation and Future Work

We present a novel and general data-driven optimization method that can generate plausible behaviors for heterogeneous agents in different scenarios. We demonstrate our model's generalizability by simulating human crowds, traffic, and mixed traffic in multiple scenarios. To the best of our knowledge, this is the first data-driven multi-agent method that is applicable to such different simulation scenarios and that mixes different kinds of agents (e.g., vehicles and pedestrians).

The simulation results of our method are plausible. We compare our results with prior methods in the same scenarios and by conducting three user studies with various scenarios from different views and analyzing the statistical results of the user studies. Our method can generate results that are closer to the original datasets, than those achieve with the prior methods. In addition, our model is fast and can be used for interactive simulations (Table 1). We also demonstrate that the plausibility of our method can be increased via VR by performing a user study comparing the results via VR or desktop.

Our method can simulate behaviors that are different from those of the input datasets. First, our method can generate larger and denser groups than those in the input datasets (Fig. 5). Second, our method can simulate scenarios that may differ from those of the input datasets (Figs. 5b, 6a). Third, our method can mix different kinds of agents in the same scenario (Fig. 6b and 6c).

Limitations. Although our approach can generate various behaviors even with a simple, sparse input dataset, the actual performance of our approach can vary based on the datasets. For example, if the dataset only has two magnitudes of velocity in it, the velocity of a car attempting to stop and move again after several seconds will not be continuous. Because our method uses a forward Euler integration scheme, the stability of our simulation depends on the size of the timestep. An implicit integration scheme [12] can be introduced to improve the stability. We represent agents as rectangular objects or discs. More precise geometrical shapes should be used to implement better collision avoidance.

As part of future work, our work can be extended in many ways. The input data is not limited to the real datasets and users can also use simulation results to direct certain behaviors. Therefore, the variety or diversity of simulation results can be dramatically increased. We could add traditional context-aware methods to our work to create a variety of behaviors in multiple agents, which would improve the realism of the simulation results. The idea of reducing the solution space according to the continuity of movement can be applied to optimization problems in animation. We can also introduce other additional sensory information such as hearing to increase the realism of interactions among agents [36]. To make our simulation results more realistic, we also plan to use portions of real velocity profiles.

Our model can be extended to other areas. The key idea of our method can be extended to data-driven methods to simulate other particle systems. If we treat the vertex as the agent in our system and the connection between vertices as the relationship, our framework can also be applied to data-driven body animation [43]. Because we model the decision-making process as an energy-based optimization problem, this idea may be applicable to path planning for robotics and unmanned aerial vehicles. Finally, we want to further evaluate the benefits of our simulator in VR and training scenarios.

ACKNOWLEDGMENTS

Xiaogang Jin was supported by the National Key R&D Program of China (Grant No. 2017YFB1002600), Artificial Intelligence Research Foundation of Baidu Inc., the Key Research and Development Program of Zhejiang Province (Grant No. 2018C01090), and the National Natural Science Foundation of China (Grant No. 61972344). Dinesh Manocha is supported in part by ARO Grant No. W911NF-19-1-0069 and Intel. Some of the work was done while the first author was an intern at Baidu Research. The authors thank the reviewers for valuable suggestions.

REFERENCES

[1] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," *ACM SIGGRAPH Comput. Graphics*, vol. 21, no. 4, pp. 25–34, 1987.

[2] X. Wang, J. Ren, X. Jin, and D. Manocha, "Bswarm: Biologically-plausible dynamics model of insect swarms," in *Proc. 14th ACM SIGGRAPH/Eurographics Symp. Comput. Animation*, 2015, pp. 111–118.

[3] I. Karamouzas, B. Skinner, and S. J. Guy, "Universal power law governing pedestrian interactions," *Phys. Rev. Lett.*, vol. 113, no. 23, p. 238701, 2014.

[4] J. Ondřej, J. Pettré, A.-H. Olivier, and S. Donikian, "A synthetic-vision based steering approach for crowd simulation," *ACM Trans. Graphics*, vol. 29, no. 4, pp. 123:1–123:9, 2010.

[5] T. B. Dutra, R. Marques, J. B. Cavalcante-Neto, C. A. Vidal, and J. Pettré, "Gradient-based steering for vision-based crowd simulation algorithms," *Comput. Graphics Forum*, vol. 36, no. 2, pp. 337–348, 2017.

[6] J. Pettré, J. Ondřej, A.-H. Olivier, A. Cretual, and S. Donikian, "Experiment-based modeling, simulation and validation of interactions between virtual walkers," in *Proc. ACM SIGGRAPH/Eurographics . Comput. Animation*, 2009, pp. 189–198.

[7] Q. Chao, Z. Deng, J. Ren, Q. Ye, and X. Jin, "Realistic data-driven traffic flow animation using texture synthesis," *IEEE Trans. Vis. Comput. Graphics*, vol. 24, no. 2, pp. 1167–1178, Feb. 2018.

[8] J. Sewall, D. Wilkie, and M. C. Lin, "Interactive hybrid simulation of large-scale traffic," *ACM Trans. Graphics*, vol. 30, no. 6, pp. 135:1–135:12, 2011.

[9] S. Kim, A. Bera, A. Best, R. Chabra, and D. Manocha, "Interactive and adaptive data-driven crowd simulation," in *Proc. IEEE Virtual Reality Conf.*, 2016, pp. 29–38.

[10] K. Jordao, J. Pettre, M. Christie, and M.-P. Cani, "Crowd sculpting: A space-time sculpting method for populating virtual environments," *Comput. Graphics Forum*, vol. 33, no. 2, pp. 351–360, 2014.

[11] E. Ju, M. G. Choi, M. Park, J. Lee, K. H. Lee, and S. Takahashi, "Morphable crowds," *ACM Trans. Graphics*, vol. 29, no. 6, pp. 140:1–140:10, 2010.

[12] I. Karamouzas, N. Sohre, R. Narain, and S. J. Guy, "Implicit crowds: optimization integrator for robust crowd simulation," *ACM Trans. Graphics*, vol. 36, no. 4, pp. 136:1–136:13, 2017.

[13] D. Wolinski, S. J. Guy, A.-H. Olivier, M. Lin, D. Manocha, and J. Pettre, "Parameter estimation and comparative evaluation of crowd simulations," *Comput. Graphics Forum*, vol. 33, no. 2, pp. 303–312, 2014.

[14] G. Berseth, M. Kapadia, B. Haworth, and P. Faloutsos, "Steerfit: automated parameter fitting for steering algorithms," in *Proc. ACM SIGGRAPH/Eurographics Symp. Computer. Animation*, 2014, pp. 113–122.

[15] I. Karamouzas and M. Overmars, "Simulating and evaluating the local behavior of small pedestrian groups," *IEEE Trans. Vis. Comput. Graphics*, vol. 18, no. 3, pp. 394–406, Mar. 2012.

[16] K. H. Lee, M. G. Choi, Q. Hong, and J. Lee, "Group behavior from video: A data-driven approach to crowd simulation," in *Proc. ACM SIGGRAPH/Eurographics . Comput. Animation*, 2007, pp. 109–118.

[17] Q. Chao, J. Shen, and X. Jin, "Video-based personalized traffic learning," *Graphical Models*, vol. 75, no. 6, pp. 305–317, 2013.

[18] C. D. Boatright, M. Kapadia, J. M. Shapira, and N. I. Badler, "Context-sensitive data-driven crowd simulation," in *Proc. 12th ACM SIGGRAPH Int. Conf. Virtual-Reality Continuum Appl. Ind.*, 2013, pp. 51–56.

[19] P. Charalambous and Y. Chrysanthou, "The pag crowd: A graph based approach for efficient data-driven crowd simulation," *Comput. Graphics Forum*, vol. 33, no. 8, pp. 95–108, 2014.

[20] H. Bi, T. Mao, Z. Wang, and Z. Deng, "A data-driven model for lane-changing in traffic simulation," in *Proc. ACM SIGGRAPH/Eurographics Symp. Comput. Animation*, 2016, pp. 149–158.

[21] W. Li, D. Wolinski, and M. C. Lin, "City-scale traffic animation using statistical learning and metamodel-based optimization," *ACM Trans. Graphics*, vol. 36, no. 6, pp. 200:1–200:12, 2017.

[22] D. Wilkie, J. Sewall, and M. Lin, "Flow reconstruction for data-driven traffic animation," *ACM Trans. Graphics*, vol. 32, no. 4, pp. 89:1–89:10, 2013.

[23] S. Yoon, M. Kapadia, P. Sahu, and V. Pavlovic, "Filling in the blanks: Reconstructing microscopic crowd motion from multiple disparate noisy sensors," in *Proc. IEEE Winter Appl. Comput. Vis. Workshops*, 2016, pp. 1–9.

[24] G. Qiao, S. Yoon, M. Kapadia, and V. Pavlovic, "The role of data-driven priors in multi-agent crowd trajectory estimation," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 4710–4717.

[25] B. Yersin, J. Maïm, J. Pettré, and D. Thalmann, "Crowd patches: populating large-scale virtual environments for real-time applications," in *Proc. Symp. Interactive 3D Graphics Games*, 2009, pp. 207–214.

[26] Y. Li, M. Christie, O. Siret, and R. Kulpa, "Cloning crowd motions," in *Proc. 11th ACM SIGGRAPH / Eurographics Symp. Comput. Animation*, 2012, pp. 201–210.

[27] K. Hyun, M. Kim, Y. Hwang, and J. Lee, "Tiling motion patches," *IEEE Trans. Vis. Comput. Graphics*, vol. 19, no. 11, pp. 1923–1934, Nov. 2013.

[28] Y.-C. Lai, S. Chenney, and S. Fan, "Group motion graphs," in *Proc. ACM SIGGRAPH/Eurographics Symp. Comput. Animation*, 2005, pp. 281–290.

[29] A. Lerner, Y. Chrysanthou, and D. Lischinski, "Crowds by example," *Comput. Graphics Forum*, vol. 26, no. 3, pp. 655–664, 2007.
[30] M. Zhao, S. J. Turner, and W. Cai, "A data-driven crowd simulation model based on clustering and classification," in *Proc. IEEE/ACM 17th Int. Symp. Distrib. Simul. Real Time Appl.*, 2013, pp. 125–134.
[31] W. Li, D. Wolinski, J. Pettre, and M. C. Lin, "Biologically-inspired visual simulation of insect swarms," *Comput. Graphics Forum*, vol. 34, no. 2, pp. 425–434, 2015.
[32] P. M. Kielar and A. Borrmann, "Spice: A cognitive agent framework for computational crowd simulations in complex environments," *Auton. Agents Multi-Agent Syst.*, vol. 32, no. 3, pp. 387–416, 2018.
[33] W. Liu, V. Pavlovic, K. Hu, P. Faloutsos, S. Yoon, and M. Kapadia, "Characterizing the relationship between environment layout and crowd movement using machine learning," in *Proc. 10th Int. Conf. Motion Games*, 2017, pp. 2:1–2:6.
[34] D. Wolinski, M. C. Lin, and J. Pettré, "Warpdriver: Context-aware probabilistic motion prediction for crowd simulation," *ACM Trans. Graphics*, vol. 35, no. 6, pp. 164:1–164:11, 2016.
[35] Z. Ren, P. Charalambous, J. Bruneau, Q. Peng, and J. Pettré, "Group modeling: A unified velocity-based approach," *Comput. Graphics Forum*, vol. 36, no. 8, pp. 45–56, 2017.
[36] Y. Wang, M. Kapadia, P. Huang, L. Kavan, and N. I. Badler, "Sound localization and multi-modal steering for autonomous virtual agents," in *Proc. 18th Meeting ACM SIGGRAPH Symp. Interactive 3D Graphics Games*, 2014, pp. 23–30.
[37] R. Bridson and M. Müller-Fischer, "Fluid simulation: Siggraph 2007 course notes video files associated with this course are available from the citation page," in *Proc. ACM SIGGRAPH Courses*, 2007, pp. 1–81.
[38] J. Zhang, W. Klingsch, A. Schadschneider, and A. Seyfried, "Ordering in bidirectional pedestrian flows and its influence on the fundamental diagram," *J. Statistical Mech.: Theory Exp.*, vol. 2012, no. 2, 2012, Art. no. P02002.
[39] "Next generation simulation," 2013. [Online]. Available: http://ops.fhwa.dot.gov/traffican-alysistools/ngsim.htm
[40] B. K. Horn and B. G. Schunck, "Determining optical flow," *Artif. Intell.*, vol. 17, no. 1–3, pp. 185–203, 1981.
[41] S. J. Guy, J. van den Berg, W. Liu, R. Lau, M. C. Lin, and D. Manocha, "A statistical similarity measure for aggregate crowd dynamics," *ACM Trans. Graphics*, vol. 31, no. 6, pp. 190:1–190:11, 2012.
[42] H. Wang, J. Ondej, and C. OSullivan, "Trending paths: A new semantic-level metric for comparing simulated and real crowd data," *IEEE Trans. Vis. Comput. Graphics*, vol. 23, no. 5, pp. 1454–1464, May 2017.
[43] M. Kim, G. Pons-Moll, S. Pujades, S. Bang, J. Kim, M. J. Black, and S.-H. Lee, "Data-driven physics for human soft tissue animation," *ACM Trans. Graphics*, vol. 36, no. 4, pp. 54:1–54:12, 2017.

Jiaping Ren received the BSc degree from the Department of Mathematics, Zhejiang University of Technology, P. R. China, in 2013. She is working toward the PhD degree in the StateKey Laboratory of CAD&CG, Zhejiang University, P. R. China. Her main research interests include traffic simulation, crowd animation, collective behaviors of bird flocks, and insect swarm animation.

Wei Xiang received the BSc degree in computer science and technology from Shandong University, P. R. China, in 2016. She is working toward the PhD degree in the State Key Laboratory of CAD&CG, Zhejiang University, P. R. China. Her main research interests include crowd animation, insect swarm animation, traffic simulation and autonomous driving.

Yangxi Xiao received the BSc degree in digital media technology from Zhejiang University, in 2014. She is working toward the PhD degree from the State Key Lab of CAD&CG, Zhejiang University, P. R. China. Her main research interests include digital video processing and editing.

Ruigang Yang received the MSc degree from Columbia University and the PhD degree from the University of North Carolina at Chapel Hill. He is the chief scientist of 3D vision with Baidu Research and a full professor of computer science with the University of Kentucky (on leave). His research interests include computer graphics and computer vision, in particular in 3D reconstruction and 3D data analysis. He has published more than 100 papers, which, according to Google Scholar, has received close to 10000 citations with an h-index of 47 (as of 2017). He has received a number of awards, including US NSF Career award in 2004 and the Deans Research Award at the University of Kentucky in 2013. He is a senior member of the IEEE.

Dinesh Manocha is the Paul Chrisman Iribe chair in computer science & electrical and computer engineering with the University of Maryland College Park. He is also the Phi Delta Theta/Matthew Mason distinguished professor emeritus of computer science with the University of North Carolina - Chapel Hill. His research interests include multi-agent simulation, virtual environments, physicallybased modeling, and robotics. He has received many awards, including Alfred P. Sloan Research Fellow, the NSF Career Award, the ONR Young Investigator Award, and the Hettleman Prize for scholarly achievement. His group has developed a number of packages for multi-agent simulation, crowd simulation, and physicsbased simulation that have been used by hundreds of thousands of users and licensed to more than 60 commercial vendors. He has published more than 480 papers and supervised more than 35 PhD dissertations. He is an inventor of nine patents, several of which have been licensed to industry. His work has been covered by the New York Times, NPR, Boston Globe,Washington Post, ZDNet, as well as DARPA Legacy Press Release. He is a fellow of the AAAI, AAAS, ACM, and IEEE and also received the Distinguished Alumni Award from IIT Delhi. See http://www.cs.umd.edu/people/dmanocha.

Xiaogang Jin received the BSc degree in computer science and the MSc and PhD degrees in applied mathematics from Zhejiang University, P. R. China, in 1989, 1992, and 1995, respectively. He is a professor with the State Key Laboratory of CAD&CG, Zhejiang University. His current research interests include digital geometry processing, geometric modeling, 3D printing, virtual try-on, insect swarm simulation, traffic simulation, implicit surface modeling and applications, creative modeling, sketch-based modeling, and image processing. He received an ACM Recognition of Service Award in 2015. He is a member of the IEEE and ACM.

Heter-Sim: Heterogeneous Multi-Agent Systems Simulation by Interactive Data-Driven Optimization

Jiaping Ren, Wei Xiang, Yangxi Xiao, Ruigang Yang, *Senior Member, IEEE*, Dinesh Manocha, *Fellow, IEEE*, and Xiaogang Jin, *Member, IEEE*

Abstract—Interactive multi-agent simulation algorithms are used to compute the trajectories and behaviors of different entities in virtual reality scenarios. However, current methods involve considerable parameter tweaking to generate plausible behaviors. We introduce a novel approach (Heter-Sim) that combines physics-based simulation methods with data-driven techniques using an optimization-based formulation. Our approach is general and can simulate heterogeneous agents corresponding to human crowds, traffic, vehicles, or combinations of different agents with varying dynamics. We estimate motion states from real-world datasets that include information about position, velocity, and control direction. Our optimization algorithm considers several constraints, including velocity continuity, collision avoidance, attraction, direction control. Other constraints are implemented by introducing a novel energy function to control the motions of heterogeneous agents. To accelerate the computations, we reduce the search space for both collision avoidance and optimal solution computation. Heter-Sim can simulate tens or hundreds of agents at interactive rates and we compare its accuracy with real-world datasets and prior algorithms. We also perform user studies that evaluate the plausible behaviors generated by our algorithm and a user study that evaluates the plausibility of our algorithm via VR.

Index Terms—Multi-agent model, heterogeneous group, data-driven method, physically driven simulation

1 Introduction

Many virtual reality and training systems need to be able to simulate different types of agents, including human crowds and traffic. Applications include VR therapy for crowd phobias, traffic agents for autonomous driving, urban design and planning, driving simulators for education and entertainment, etc. It is important to simulate the behaviors and trajectories of different types of agents, including pedestrians and vehicles, and the interactions between such heterogeneous agents. Furthermore, it is important to develop general plausible algorithms that are applicable to a wide variety of scenarios.

There are extensive works on interactive multi-agent simulation, including crowd simulation and traffic simulation. These works include techniques based on rule-based methods [1], physics-based simulations [2], [3], vision-based methods [4], energy-based models [5], data-driven methods [6], [7], and combinations of these approaches [8], [9]. These methods are flexible and have been successfully applied to different scenarios. However, they often use many parameters and require a significant amount of effort to achieve good results that are plausible and match the behaviors observed in real-world scenarios. Furthermore, the results of these methods often seem too regular because all the agents have similar locomotion or movement patterns.

With the improvement of data acquisition techniques, more data-driven methods are emerging. Most of these methods are patch-based or use real-world agent trajectories [2], [9], [10], [11]. These methods extract patches or trajectory segments from input datasets and either connect them with some rules or use them to learn some characteristics of an agent's motion. With these methods, users can generate more plausible or more accurate results than with traditional rule-based or physics-based simulation methods. However, the variety of the simulation results depends on that of input data. If the amount of input data is small, the simulation results will be periodic and monotonous.

Most of the existing methods only apply to one kind of agent, e.g., only human pedestrians or only vehicles. In contrast, we want to use a general method to model the behaviors of different kinds of agents in a heterogeneous setting while retaining the motion features of each kind of agent. This is important in many situations like simulating the motion trajectories and interactions between cars and humans at a traffic crossing. Data-driven methods can help us with simulating interactions between heterogeneous agents by preserving the motion features of each kind of

- *J. Ren, W. Xiang, Y. Xiao, and X. Jin are with State Key Lab of CAD&CG, Zhejiang University, Hangzhou 310058, P. R. China. E-mail: {ren_jia_ping, xiang_wei, x_x}@zju.edu.cn, jin@cad.zju.edu.cn.*
- *R. Yang is with the University of Kentucky, Lexington, KY 40506, and with Baidu Research, Beijing 100193, China, and also with National Engineering Laboratory of Deep Learning Technology and Application, Beijing 100193, China. E-mail: ryang@cs.uky.edu.*
- *D. Manocha is with the Department of Computer Science, University of Maryland, College Park, MD 20742. E-mail: dm@cs.umd.edu.*

Manuscript received 27 Jan. 2019; revised 16 Sept. 2019; accepted 28 Sept. 2019. Date of publication 0 . 0000; date of current version 0 . 0000.
(Corresponding author: Xiaogang Jin.)
Recommended for acceptance by P. Wonka.
Digital Object Identifier no. 10.1109/TVCG.2019.2946769

Fig. 1. Our heterogeneous multi-agent simulation algorithm can be used for scenarios with tens or hundreds of different types of agents sharing a physical space. Pedestrians walking on a street (the first). Cars moving on a twisting road (the second). Traffic including cars and pedestrians (the third). Traffic shown through VR (the fourth). Our approach can generate plausible behaviors at interactive rates on a desktop PC and through VR.

agent. However, data-driven methods depend on the input data, and it is difficult to simulate behavior in a scenario that is different from the one that generated the input data.

Main Results. We present a novel, heterogeneous multi-agent simulation algorithm (Heter-Sim) that combines the benefits of prior data-driven and physics-based simulation methods to generate general and plausible simulations. Our interactive approach can simulate not only heterogeneous agents while generating plausible behaviors, but also scenarios different from those included in the input datasets. We convert various datasets captured using different types of sensors into a uniform format and extract the agents' states, including velocity information. We model the decision-making or local navigation process of each agent as an optimization problem and define an energy function that considers collision avoidance, attraction, velocity continuity, and direction control. Our energy function tries to match the results with the characteristics of real-world data. At a given moment, each agent chooses a velocity from a dataset. We align the control directions between simulation agents and real-world agents to diversify agents' possible behaviors and movements where there is relatively less input data available. To accelerate the computation, we utilize spatial continuity to reduce possible collisions and use the velocity continuity to reduce the solution space for energy functions.

Overall, the novel contributions of our work include:

- A general, optimization-based method to simulate heterogeneous multi-agent systems. We use our approach to simulate crowds, traffic, and any combination of those agents.
- A data-driven scheme to improve the plausibility of our simulation. We use two fast search methods based on spatial continuity and velocity continuity to search for possible collision-free solutions.
- A constraint energy function to achieve the heterogeneity of the simulation system. We use different constraint energy functions to model various constraints on dynamics, traffic rules, and interactions for heterogeneous agents.
- A direction adaptation method to simulate more kinds of scenarios. We use direction control, which computes ideal directions, to guide agents in various environments. Our method can simulate agents' behaviors that may differ from those captured by the input data.

We highlight the performance of our approach on different scenarios in Fig. 1. In practice, our approach can generate plausible trajectories and behaviors for tens or hundreds of heterogeneous agents at interactive rates. To demonstrate the benefits of our method, we have conducted two user studies to evaluate the benefits of our method over prior methods while using a top-down view and an agent's view. In both studies, participants exhibit significant preference for our method over a prior crowd simulation method [12] and a traffic simulation method [7]. We also conduct a user study to compare the user experience via VR and via desktop, and VR shows a better user experience (see Section 7).

2 Related Work

There is considerable research in multi-agent simulation, including many algorithms for simulating crowds and traffic. In this section, we give a brief overview of prior methods for parameter estimation and data-driven simulation.

2.1 Parameter Estimation and Real-World Characteristics

Parameter estimation with real-world datasets improves the accuracy of simulation methods. Researchers utilize empirical data to compute the parameters used for rule-based or physically-based multi-agent simulation methods automatically. Wolinski et al. [13] present a method to compute optimal parameters for rule-based or physically-based multi-agent simulation algorithms. Berseth et al. [14] present an approach that computes parameters for steering methods by minimizing any combination of performance metrics. Karamouzas et al. [15] use distortion and longitudinal dispersion of the group to evaluate the results from simulations. Different from these parameter estimation methods, our approach finds the best velocity from real-world datasets to generate realistic motions.

Many techniques have been proposed to learn agent characteristics from empirical data and to then use them for multi-agent simulation. Lee et al. [16] present a crowd simulation method which use an agent model generated from real-world observations. Chao et al. [17] apply characteristics of drivers from an empirical video to an agent-based model. Boatright et al. [18] classify the contexts and learn the characteristics from a dataset. Charalambous et al. [19] present a real-time synthesis method for crowd steering behaviors with the temporal perception pattern. Bi et al. [20] simulate the process of lane-changing in traffic by learning characteristics from features of real vehicle trajectories. Kim et al. [9] compute collision-free trajectories of virtual pedestrians by learning pedestrian dynamics from 2D trajectories. Besides, Ondřej et al. [4] present a vision-based approach of collision avoidance between walkers that fit the requirements of interactive crowd simulation. Our data-driven optimization algorithm is complimentary to these algorithms and can be combined with them.

Reconstruction of certain aspects of real-world scenes has also been used for multi-agent simulation, especially for traffic simulation. Li et al. [21] reconstruct traffic with GPS mobile vehicle data. Wilkie et al. [22] drive an agent-based traffic simulator by using the state of traffic flow estimated from sparse sensor measurements. Yoon et al. [23] propose a refinement method to reconstruct a holistic view of crowd's movements with noisy tracked data. Qiao et al. [24] present a trajectory interpolation method by combining trajectory estimation and global optimization. Our approach is more general than these prior methods.

2.2 Data-Driven Multi-Agent Simulation

Patch-based methods transfer the original trajectories from empirical data into patches and connect these patches with some rules. Yersin et al. [25] extend the concept of motion patches to dense populations in large environments. Li et al. [26] animate large crowds with examples of multi-agent motions by using a copy-and-paste technique. Hyun et al. [27] tile deformable motion patches, which describe episodes of the movements of multiple characters. Jordao et al. [10] propose a crowd sculpting method to guide crowd motion by using intuitive deformation gestures.

As with patch-based methods, researchers replicate trajectory tubes extracted from empirical data to synthesize new agent animations. Lai et al. [28] introduce group motion graphs to animate groups of discrete agents with empirical data. Lerner et al. [29] generate seemingly natural behaviors by copying trajectories from real people and applying them to simulated agents. Ju et al. [11] generate new animations, which can include arbitrary numbers of agents, by blending existing data. Zhao et al. [30] cluster the examples extracted from human motion data and combine similar examples to produce an output. Li et al. [31] propose a general, biologically-inspired framework with a three-level method using statistical information from real datasets. Kielar et al. [32] predict movement behaviors of crowds with a cognitive agent framework. Liu et al. [33] generate crowd movements with neural networks by considering environment layouts. A new data-driven method has been proposed by Chao et al. [7]. They compute the velocity for each agent in each frame from empirical data. However, this method is time-consuming because it tries to minimize the overall traffic texture energy and is therefore not useful for interactive applications. Our approach is complimentary to prior data-driven methods and presents a new method that combines data-driven with physics-based multi-agent methods.

3 Data-Driven Optimization

In this section, we introduce our data-driven optimization approach to simulate heterogeneous multi-agent systems.

3.1 Terminology and Notation

We use *agent* to represent the virtual character in our method. We also use the term *state* to represent the motion characteristics of each agent. Our method is general and applicable for both 2D and 3D motions. State can therefore refer to an agent's movements in either 2D or 3D space. In this paper, we limit our discussions to 2D agents.

We use set $\mathcal{G}$ to specify the set of agents in the scenario. We use the vector $\mathbf{s} = [\mathbf{p}, \mathbf{v}, \mathbf{v}^{\mathrm{d}}]^{\mathrm{T}}$, $\mathbf{s} \in \mathbb{R}^6$ to specify an agent's state, where $\mathbf{p} \in \mathbb{R}^2$ is the agent's position, $\mathbf{v} \in \mathbb{R}^2$ is the velocity, and $\mathbf{v}^{\mathrm{d}} \in \mathbb{R}^2$ is the control direction that guides the motion direction of agents. Then the state of the group becomes $\mathcal{S} = \cup_i \mathbf{s}_i$, where $\mathbf{s}_i$ is the state of agent i. Distinct from the velocity $\mathbf{v}$, the control direction $\mathbf{v}^{\mathrm{d}}$ controls the agent's global direction. We use $\hat{\mathbf{v}} = \frac{\mathbf{v}}{\|\mathbf{v}\|}$ to represent the unit vector of $\mathbf{v}$. We also use $\mathbf{v}_{i,n}$ to represent the velocity of agent i at time t_n. For any state $\mathbf{s} = [\mathbf{p}, \mathbf{v}, \mathbf{v}^{d}] \in \mathcal{S}$, $\mathbf{p} \in \mathcal{S}_{\mathrm{p}}$, $\mathbf{v} \in \mathcal{S}_{\mathrm{v}}$, $\mathbf{v}^{\mathrm{d}} \in \mathcal{S}_{\mathrm{v}^d}$. We represent our method by $[S(), D(), I(), F()]^{\mathrm{T}}$, where S is the environment evolution function, D is the data processing function, I is the initialization function, and F is the decision making function. S determines the external environment, which consists of the static environment (static obstacles, ground, etc.) and the dynamic environment (moving stimulus). D processes the data set by transferring the trajectories to the estimated states $\mathcal{D} = \cup_n \mathcal{S}^*_n = \cup_n \cup_i \mathbf{s}^*_{i,n}$, where $\mathbf{s}^*_{i,n} = [\mathbf{p}^*_{i,n}, \mathbf{v}^*_{i,n}, \mathbf{v}^{d*}_{i,n}]$ denotes the state of agent i at time t_n of the dataset. The minimal magnitude and the maximal magnitude of $\mathbf{v}^*_{i,n}$ for all i and n are $v^*_{\min}$ and $v^*_{\max}$, respectively. For any $\mathbf{s}^* = [\mathbf{p}^*, \mathbf{v}^*, \mathbf{v}^{\mathrm{d}*}] \in \mathcal{D}$, $\mathbf{p}^* \in \mathcal{D}_{\mathrm{p}}$, $\mathbf{v}^* \in \mathcal{D}_{\mathrm{v}}$, $\mathbf{v}^{\mathrm{d}*} \in \mathcal{D}_{\mathrm{v}^d}$. I initializes each agent's state: position, velocity, and control direction. F is the main routine corresponding to our algorithm and computes a new state for each agent at each timestep.

3.2 Overall Approach

Our model for simulating heterogeneous multi-agent systems references the datasets to control the trajectories and behaviors of the agents (see Fig. 2). The datasets might be videos or other data representations, including trajectories or higher order features. We deal with different types of datasets and transform them into a unified representation, classifying the data by the magnitude of the velocity. The environment may also consist of static and dynamic obstacles. We initialize the position of each agent in the scene randomly and choose an initial velocity for each agent from our datasets. At each step of our simulator, we use an interactive optimization algorithm to make decisions for each agent. In particular, we solve this optimization problem by choosing a velocity from the datasets that tends to minimize our energy function. The energy function is defined based on the locomotion or dynamics rules of heterogeneous agents, including continuity of velocity, collision avoidance, attraction, direction control, and other constraints defined by users. In addition, our approach is general and can deal with different kinds of agents in the same way. We can capture corresponding motion characteristics with different datasets. As a result, we can simulate heterogeneous agents in the same physical space.

3.3 Dynamics Computation

An agent moves according to its surroundings, which include the other agents and the external environment (attractions, obstacles, roads, etc.). In these complex surroundings, the agent makes decisions in relation to all these elements. At each timestep, we calculate the state of each

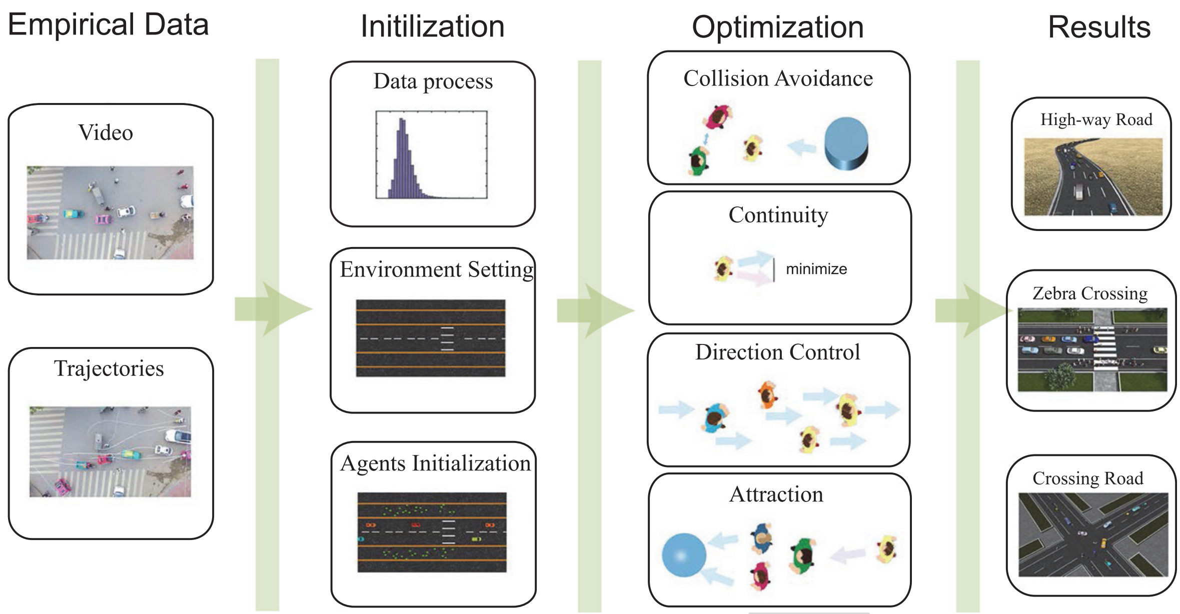


Fig. 2. Overview of our data-driven model for simulating heterogeneous multi-agent systems. We highlight different components of our algorithm. The input empirical data can be videos from a top-down view or trajectories of agents. In the initialization, we first transfer real-world data into a consistent format. With the data and environment information set by the users, we initialize the positions and velocities for agents. We treat the motion decision-making or local navigation process of each agent at every timestep as an optimization problem, and the energy function takes into consideration several factors: velocity continuity, collision avoidance, attraction, direction control, and any other constraints defined by users. Our model can simulate heterogeneous agents in the same scenario, including crowds, traffic, any combination of these agents, etc.

agent according to the prior states of all agents, the environment, and the dataset. Because the external environments may be time-varying, we set the environment evolution function as a function of time. The system of equations for the state of each agent at time t_n is

$$\begin{aligned} \mathbf{p}_{i,n} &= \mathbf{p}_{i,n-1} + \mathbf{v}_{i,n}\Delta t, \\ \mathbf{v}_{i,n} &= \underset{\mathbf{v}\in\mathcal{D}_{\mathrm{v}}}{\operatorname{argmin}}\ E(t_{n-1}, i, \mathbf{v}, \mathcal{S}_{n-1}, S(t_{n-1}, \mathbf{p}_{i,n-1}), \mathbf{v}^{\mathrm{d}}_{i,n}), \\ \mathbf{v}^{\mathrm{d}}_{i,n} &= R(\mathbf{p}_{i,n-1}, S(t_{n-1}, \mathbf{p}_{i,n-1})), \end{aligned} \tag{1}$$

where $E(t_{n-1}, i, \mathbf{v}, \mathcal{S}_{n-1}, S(t_{n-1}, \mathbf{p}_{i,n-1}), \mathbf{v}^{\mathrm{d}}_{i,n})$ is the energy function that chooses the optimal velocity for agent i at time t_n. $R(\mathbf{p}_{i,n-1}, S(t_{n-1}, \mathbf{p}_{i,n-1}))$ is a function that computes the control direction $\mathbf{v}^{\mathrm{d}}$ for each agent at each time. We compute a velocity that minimizes the energy function. If we search the velocity from a continuous-space, our method becomes an energy-based model. To capture the characteristics of different kinds of agents easily, we search for the velocity from the states in the dataset $\mathcal{D}$, which belongs to a discrete space. If the states generated from the dataset are unlimited, the simulation results will approximate those generated from the method with the continuous velocity space.

To simulate heterogeneous agents in the same physical space, we consider the common locomotion rules of multi-agent systems for the energy function $E(t_{n-1}, i, \mathbf{v}, \mathcal{S}_{n-1}, S(t_{n-1}, \mathbf{p}_{i,n-1}), \mathbf{v}^{\mathrm{d}}_{i,n})$ including collision avoidance, attraction, velocity continuity, direction control, and any other constraints.

$$\begin{aligned} &E(t_{n-1}, i, \mathbf{v}, \mathcal{S}_{n-1}, S(t_{n-1}, \mathbf{p}_{i,n-1}), \mathbf{v}^{\mathrm{d}}_{i,n}) \\ &= \sum_{k\in\theta} w_k E_k(t_{n-1}, i, \mathbf{v}, \mathcal{S}_{n-1}, S(t, \mathbf{p}_{i,n-1}), \mathbf{v}^{\mathrm{d}}_{i,n}), \end{aligned} \tag{2}$$

where $\theta = \{\mathrm{m}, c, a, d, s\}$, E_{m} is the energy for velocity continuity, E_{c} is the energy for collision avoidance, E_{a} is the energy for attraction, E_{d} is the energy for direction control, and E_{s} is the energy function for constraints of certain kinds of agents. w_{m}, w_{a}, w_{t}, w_{d}, and w_{s} are the weights of these terms respectively, and each weight represents the importance of the corresponding energy term. Velocity continuity is used to ensure that the agents move smoothly. Collision avoidance is a crucial part of multi-agent simulation. Attraction helps agents remain cohesive with other agents in the same group and has been widely used in multi-agent simulation literature [1]. The direction control represents the direction preference for agents according to the environment. These four elements can describe the basic factors considered by agents when moving. It is possible to add more constraints to control the movements of agents in E_{s}. The definition of E_{s} for each kind of agent is described in Section 5. To achieve the heterogeneity, our method uses different parameters and constraints to implement different dynamics, and use different road constraints and interaction domains to implement different traffic rules and response mechanisms.

3.4 Continuity

Because of the physical limitations, agents cannot change their motion states frequently or abruptly within a Δt time. Thus, the agent i has a tendency to choose a velocity close to $\mathbf{v}_{i,t}$ at a time $t+1$. The continuity energy is used to indicate that the agent tends to keep its velocity unchanged to save its overall energy:

$$E_{\mathrm{m}} = w_{m1} E^{\mathrm{dir}}_{\mathrm{m}} + w_{m2} E^{\mathrm{L}}_{\mathrm{m}}, \tag{3}$$

where $E^{\mathrm{dir}}_{\mathrm{m}} = \left\|\hat{\mathbf{v}}_{i,n-1} - \hat{\mathbf{v}}\right\|_2$ is for direction continuity and $E^{\mathrm{L}}_{\mathrm{m}} = \left\|\|\mathbf{v}_{i,n-1}\| - \|\mathbf{v}\|\right\|_2$ is for continuity of magnitude of velocity. $\mathbf{v}_{i,n}$ is the velocity of agent i at time t_{n-1}.

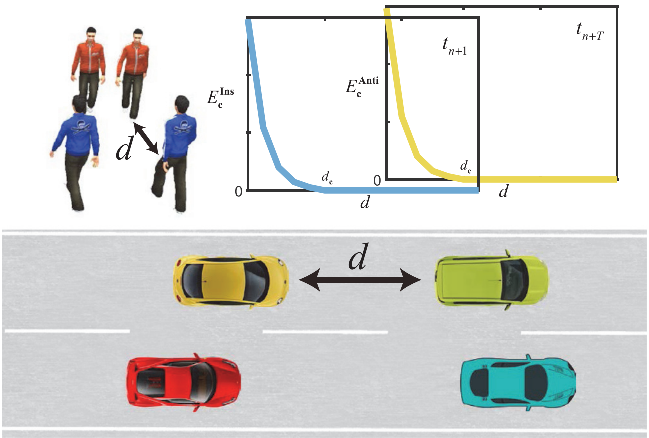


Fig. 3. Collision avoidance. In our method, the energy for collision avoidance E_c consists of two parts: the energy for instantaneous collision avoidance E_c^{Ins} and the energy for anticipation collision avoidance E_c^{Anti}. The blue curve represents E_c^{Ins} changes with the distance d between two agents increases in time $j+1$, and the yellow curve represents E_c^{Anti} changes with d in time $j+T$.

3.5 Collision Avoidance

Collision avoidance (Fig. 3) is a major issue in multi-agent simulation [3], [34]. To avoid collisions with other agents or the environmental obstacles in the scene, the agent should choose a velocity that will not cause a collision after one of more timesteps by assuming that all objects keep moving with their current velocities. Here, we consider two kinds of collisions to avoid: instantaneous collisions and anticipatory collisions.

$$E_c = w_{c1}E_c^{Ins} + w_{c2}E_c^{Anti}, \tag{4}$$

where instantaneous collision avoidance energy E_c^{Ins} only considers the possible collisions after a timestep, and anticipatory collision energy E_c^{Anti} considers the possible collisions after anticipation time T.

The normalized instantaneous collision avoidance energy E_c^{Ins} is given as

$$E_c^{Ins} = \frac{1}{|\Omega_c(\Delta t, i, t_{n-1})|} \sum_{Q \in \Omega_c(\Delta t, i, t_{n-1})} e^{d_c - d(\Delta t, \mathbf{s}_i, \mathbf{s}_Q, \mathbf{v})}, \tag{5}$$

where $\Omega_c(\Delta t, i, t_{n-1})$ is the predicted neighborhood of agent i after time Δt based on the surrounding information at time t_{n-1}. The neighborhood consists of agents that probably collide with agent i, and $|\Omega_c(\Delta t, i, t_{n-1})|$ represents the number of neighbors. $d(\Delta t, \mathbf{s}_i, \mathbf{s}_Q, \mathbf{v})$ is the predicted distance between agent i and agent Q. For each agent, we only consider collision avoidance within a distance d_c. Similarly, the anticipatory collision avoidance energy E_c^{Anti} can be given as

$$E_c^{Anti} = \frac{1}{|\Omega_c(T\Delta t, i, t_{n-1})|} \cdot \sum_{Q \in \Omega_c(T\Delta t, i, t_{n-1})} e^{d_c - d(T\Delta t, \mathbf{s}_i, \mathbf{s}_Q, \mathbf{v})}, \tag{6}$$

where $\Omega_c(T\Delta t, i, t_{n-1})$ is the predicted collision neighborhood of agent i after time $T\Delta t$ based on the surrounding information at time t_{n-1}. $d(T\Delta t, \mathbf{s}_i, \mathbf{s}_Q, \mathbf{v})$ is the predicted distance between agent i and agent Q after time T. Note that we perform instantaneous collision avoidance in each timestep while the anticipatory collision energy is only used to avoid potential future collisions.

Within the distance d_c, E_c increases when the distance between agent i and agent Q decreases (see Fig. 3). As a result, when we compute the velocity for each agent in each frame, a value making their distance larger will reduce the energy.

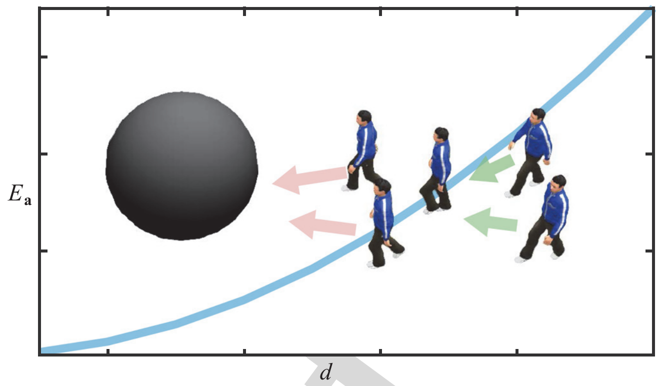


Fig. 4. Attraction. The energy for attraction include the energy for attraction (green arrows) between agents and the energy for attraction (red arrows) from environmental objects.

3.6 Attraction

If the agents want to move together as a group, we need to account for some attraction forces between them. The agent therefore prefers to choose a velocity that brings it closer to the group, allowing it to become a part of the group over the next few frames. In addition, agents may also be attracted by external stimuli. The attractions in our model include the attraction between the agents and the environment (Fig. 4). The attraction energy is given as

$$E_a = \frac{1}{|\Omega_a(\Delta t, i, t_{n-1})|} \sum_{Q \in \Omega_a(\Delta t, i, t_{n-1})} d^2(\Delta t, \mathbf{s}_i, \mathbf{s}_Q, \mathbf{v}), \tag{7}$$

where $\Omega_a(\Delta t, i, t_{n-1})$ is the predicted attraction neighborhood of agent i after time Δt based on the surrounding information at time t_{n-1}.

When the distance between agent i and agent Q increases, the energy E_a increases (see Fig. 4). Thus, a computed velocity making their distance smaller will reduce the energy.

3.7 Direction Control

We use direction control to imitate agents moving toward their goals. In this case, the agents try to choose velocities that point to their goals or that parallel the path to their goals. We assume that every agent has a goal position to guide its local movement. The goal might change over time. This goal can also be treated as a direction control defined by the users. The energy for direction control is presented as

$$E_d = \left\| \mathbf{v}_{i,n}^{d} - \hat{\mathbf{v}} \right\|_2, \tag{8}$$

where $\mathbf{v}_{i,n}^{d}$ is the control direction for agent i at time t_n.

4 Multi-Agent System Simulation with Data-Driven Optimization

In this section, we present more details about our method, as it is used to simulate heterogeneous agents.

4.1 State Estimation for the Dataset

The dataset of our method consists of trajectories that are time series of positions, $\mathcal{L}: \mathbf{Y}_1, \mathbf{Y}_2, ..., \mathbf{Y}_n....$ We estimate the state $\mathbf{s}_n^* = [\mathbf{p}_n^*, \mathbf{v}_n^*, \mathbf{v}_n^{\mathrm{d}*}]$ in the dataset based on these trajectories, and obtain the estimated position $\mathbf{p}_n^* = \mathbf{Y}_n$ and velocity $\mathbf{v}_n^* = \frac{\mathbf{Y}_n - \mathbf{Y}_{n-1}}{\Delta t}$. Estimating the control direction $\mathbf{v}_n^{\mathrm{d}*}$ is equivalent to estimating the direction to the corresponding agent's goal, according to Section 3.7. Therefore, if the agent only moves one way in the scenario, it is in the same control direction; if the agent changes its direction or goal in the dataset, we estimate its control direction at time t by computing the direction of its displacement, $\mathbf{v}^{\mathrm{d}*} = \frac{\mathbf{Y}_n - \mathbf{Y}_{n-\delta}}{\|\mathbf{Y}_n - \mathbf{Y}_{n-\delta}\|}$, which is computed every $\delta\Delta t$ time. We estimate the control direction by averaging every $\delta\Delta t$ time to reduce the estimation error from local avoidance. The results in Section 5 show that our state estimation for real-world datasets works well.

4.2 Direction Adaptation to Different Scenarios

According to Eq. (1), if we directly search the optimal velocity for each agent from the dataset, the synthesized scenario will be limited in its ability to achieve plausible movements by the scenario of the dataset. To eliminate these constraints, we map the local coordinate of the dataset to that of the scenario in the simulation by align their forward directions. As a result, we can simulate scenarios that may be different from the dataset. We suppose that the simulated scenario and the dataset have the same relative position relationship between the direction of velocity and the control direction; that is, the angle between the velocity direction and the control direction in the simulation is the same with that of the dataset, and

$$\begin{aligned}\hat{\mathbf{v}} \cdot \mathbf{v}^{\mathrm{d}} &= \hat{\mathbf{v}}^* \cdot \mathbf{v}^{\mathrm{d}*},\\ \hat{\mathbf{v}} \times \mathbf{v}^{\mathrm{d}} &= \hat{\mathbf{v}}^* \times \mathbf{v}^{\mathrm{d}*}.\end{aligned} \tag{9}$$

Therefore, we obtain $\hat{\mathbf{v}}$, and $\mathbf{v} = \|\mathbf{v}^*\|\hat{\mathbf{v}}$.

4.3 Distance and Neighborhood

We hypothesize that the velocity of an agent remains unchanged over a short time t and the shapes of agents or obstacles cannot be ignored. If the agent i moves with the velocity $\mathbf{v}$ chosen from the dataset, the predicted distance between agent i and agent Q after time t becomes

$$d(t, \mathbf{s}_i, \mathbf{s}_Q, \mathbf{v}) = \|\mathbf{p}_i + \mathbf{v}t - (\mathbf{p}_Q + \mathbf{v}_Q t) - (R_i^{dir} + R_Q^{dir})\|_2, \tag{10}$$

where R_i^{dir} is the radius of agent i in the direction toward agent Q ($Q \neq i$). R_Q^{dir} is also a directional radius of agent Q. The shapes of different agents can be different. For example, we use a rectangular object to represent a car and a disc to represent a pedestrian. If Q is an entity in the environment, Eq. (10) becomes a distance function between an agent and the entity in the environment. For a twisting road, we compute the distance between two cars as the distance along the curve of the road.

In contrast to the existing methods [35], the agents in our method try to avoid collisions with not only the homogeneous agents but also the heterogeneous agents. To avoid collisions, each agent tries to keep away from other agents or obstacles when they get too close. In the real world, humans can perceive the environment through both vision and sound [36], and thus we can assume that an agent can avoid collisions in a full field of vision with a limited range. We define the neighborhood for collision avoidance as

$$\Omega_{\mathrm{c}}(t, i, t_n) = \{Q | d(t, \mathbf{s}_i, \mathbf{s}_Q, \mathbf{v}) < d_{\mathrm{c}}, Q \in \mathcal{G} \setminus \{i\} \cup \mathcal{G}_{\mathrm{c}}\}, \tag{11}$$

where d_{c} is the threshold distance for collision avoidance and $\mathcal{G}_{\mathrm{c}}$ is the set of obstacles in the scenario. Each agent considers collision avoidance with the agents or obstacles within a distance d_{c}. Meanwhile, each agent tries to keep close to the agents in its group or to the external attraction stimulus if the distance between the agents is large. We define the neighborhood for attraction as

$$\Omega_{\mathrm{a}}(t, i, t_n) = \{Q | d(t, \mathbf{s}_i, \mathbf{s}_Q, \mathbf{v}) > d_{\mathrm{a}}, Q \in \mathcal{G} \cup \mathcal{G}_{\mathrm{a}}\}, \tag{12}$$

where d_{a} is the threshold distance for attraction and $\mathcal{G}_{\mathrm{a}}$ is the set of attraction in the scenario. An entity that is treated as an attraction can also be an obstacle if the shape of it cannot be ignored, that is, $\mathcal{G}_{\mathrm{c}} \cap \mathcal{G}_{\mathrm{a}} \neq \varnothing$.

4.4 Faster Computation

If we use a brute force method to solve Eq. (1), the computation cost will be large. The underlying time complexity will be $O(n^2 m)$ with n agents in the simulation and m estimate states in the dataset. The most time-consuming parts are searching for the optimal velocity from the dataset and finding the neighborhood for each agent. To achieve interactive performance, we propose two acceleration methods.

4.4.1 Reduced Solution Space

To find the optimal velocity for each agent efficiently, we reduce the solution space of Eq. (1). We classify the estimated states of the dataset into groups based on the magnitude of the velocity. Considering the continuity of motion, we search for the velocity for each agent in the current group of velocities and in the adjacent groups,

$$\mathbf{v}_{i,n+1} \in \bigcup_{m=l-z}^{l+z} \{\mathbf{v}^{n*}\}, \tag{13}$$

where $\{\mathbf{v}^l\}$ is the set of velocities of the group l to which $\mathbf{v}_{i,n}$ belongs, z is the scope of the number of groups that are considered for computing optimal velocity, and the group $\{\mathbf{v}^m\}$ with $m \in [l-z, l+z]$ is the neighborhood of $\{\mathbf{v}^l\}$.

4.4.2 Grid in Space

To reduce the time consumption for computing the neighborhood for each agent, we introduce the idea of grid in space from fluid simulation [37]. For our simulation, the 2D plane is divided into 2D grids. We suppose that $\mathcal{O}_{x,y}$ denotes the set of all agents in the grid $O_{x,y}$. Then the candidate neighborhood of i in grid $O_{x,y}$ is reduced from $\mathcal{G}$ to $\mathcal{G}'$,

$$\mathcal{G}' = \bigcup_{k_1=x-1}^{x+1} \bigcup_{k_2=y-1}^{y+1} \mathcal{O}_{k_1,k_2}. \tag{14}$$

When we search the neighborhood for collision avoidance, we compare the distances of the agents in the grid $O_{x,y}$ with

TABLE 1
Performance for Different Scenarios

Scenario	Types	Behavior	N	Dataset	Time(s/f)
Crowd-1	human	walking on street	8-148	[Lerner et al. 2007]	0-0.0040
Crowd-2	human	mixture of two crowds	100	[Zhang et al. 2012]	0.0209
Crowd-3	human	avoiding static obstacles	79	[Zhang et al. 2012]	0.0192
Traffic-1	car	movements on a twist road	80	[NGS 2013]	0.0137
Traffic-2	human/car	movements on a crossing road	30/35	[NGS 2013]/[Zhang et al. 2012]	0.0378
Traffic-3	human/bicycle/tricycle/car	mixture of multiple systems	25/15/10/40	video from Shandong, China	0.0342

We summarize the characteristics of the simulation scenarios in this paper. The agents include humans, cars, bicycles, and tricycles. The datasets used for input data vary. We use seconds per frame to measure the time performance of the simulations. Our method can achieve realtime performance using 4 cores on a CPU.

the agents adjacent to this grid instead of comparing them to all the agents in the scenario.

5 Results

In this section, we highlight the performance of our approach in generating simulations of crowds, traffic, and combinations of different types of agents. We have implemented our approach in C++ on a desktop machine with a 3.30 GHz Inter Xeon CPU E3-1230 v3 4-core processor and 32 GB memory. The performances for different scenarios are given in Table 1.

To achieve the heterogeneity of our simulation system, we use different parameters and E_{s} for different kinds of agents. In addition, we employ real-world datasets consisting of pedestrians, bicycles, tricycles and cars captured from real scenarios. We initialize the weights with 1.0, and they can be tuned according to the behaviors of the agents. Table 2 shows the weights of all the benchmarks. We define the user control for each pedestrian with speed control $E_{\mathrm{s}} = E_{\mathrm{sc}} = |\|\mathbf{v}\| - v_i|$, where v_i is the ideal speed for agent i. We define the user control for each car with speed control and position control $E_{\mathrm{s}} = E_{\mathrm{sc}} + E_{\mathrm{p}}$, where E_{s} is the same with that of each pedestrian, $E_{\mathrm{p}} = |\mathbf{v} \cdot (\mathbf{v}^{\mathrm{d}})^{\perp}|$, and $\mathbf{v}^{\mathrm{d}}$ is a tangential vector of the given lane. Cars try to drive in the middle of the lane.

5.1 Data Acquisition

Our method accepts different kinds of input datasets if those datasets contain the velocity information for the agents. Any form of discontinuity or a small amount of abnormal data in the datasets is acceptable.

In our current framework, we have used some widely available datasets from different scenarios and environments. The datasets for crowd simulation include two scenarios: one is from [38] and features two-dimensional bidirectional movements with 304 pedestrians and 1,273 frames; the second is from [29] and features street scenarios with 8-148 pedestrians and 9,014 frames. We set the control directions for the first dataset as the directions that point to the agents' destinations. For the second dataset, the control direction of one agent at a certain time is the direction that points from its current position to the position of its next record.

The traffic dataset is extracted from the Next Generation Simulation (NGSIM) datasets [39], which include detailed, high-quality highway traffic datasets. We extract 300 frames and 161 cars in total. We set the direction of the road as the estimation of the control directions of the cars. The datasets corresponding to the mixed traffic scenarios (including pedestrians, bicycles, tricycles, and cars) are generated from videos. The video was recorded in Shandong, China. We use the optical flow tracking method [40] to trace the agents. The extracted data consists of 435 frames and contains 3 pedestrians, 10 bicycles, 10 tricycles, and 2 cars. The control direction for each agent in every frame is computed by averaging the directions of the agent from 30 frames.

5.2 Human Crowd

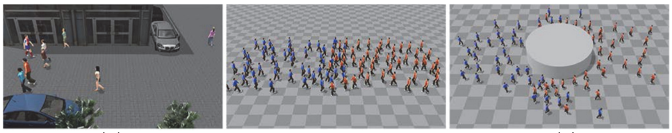
We simulate three benchmark scenarios with crowds representing each pedestrian as a disc with a fixed radius.

Crowd-1. We simulate behaviors of pedestrians on a street with the dataset from [29] to show that our method

TABLE 2
The Weights for Simulation

Scenario		$E_{\mathrm{t}}^{\mathrm{dir}}$	$E_{\mathrm{t}}^{\mathrm{L}}$	E_{c}^{Ins}	$E_{\mathrm{c}}^{\mathrm{Anti}}$	E_{a}	E_{d}	E_{p}	E_{sc}
Crowd-1		1.0	1.0	1.0	1.0	0.0	1.0	0.0	0.5
Crowd-2		1.0	1.0	1.0	1.0	0.0	1.0	0.0	1.5
Crowd-3	obstacle zone	1.5	1.0	0.67	0.67	0.0	1.0	0.0	1.0
	before obstacle zone	1.0	1.0	0.67	0.67	0.0	1.0	0.0	1.0
	after obstacle zone	1.0	1.0	0.67	0.67	0.0	1.5	0.0	1.0
Traffic-1		0.5	0.5	1.0	1.0	2.0	3.0	10.0	10.0
Traffic-2	Pedestrian	1.0	1.0	1.0	1.0	0.0	1.5	1.0	10.0
	Car	5.0	1.0	1.0	1.0	2.0	5.0	1.0	10.0
Traffic-3	Type-1	10.0	1.0	1.0	1.0	0.0	5.0	10.0	5.0
	Type-2	0.5	0.5	1.0	1.0	2.0	3.0	1.0	10.0

This table gives the weights for the direction continuity $E_{\mathrm{t}}^{\mathrm{dir}}$, the speed continuity $E_{\mathrm{t}}^{\mathrm{L}}$, instantaneous collision avoidance E_{c}^{Ins}, anticipated collision avoidance E_{c}^{Anti}, attraction E_{a}, direction control E_{d}, position control E_{p}, and speed control E_{sc} in each scenario.



(a) (b) (c)

Fig. 5. The mixed crowds with different control directions. (a) Pedestrians with changing control directions walk on a street. (b) Two crowds with inverse control directions. The pedestrians with the same clothes represent individuals in the same crowd. The crowds walk to their own destinations while avoiding collisions with each other. (c) We add an obstacle to the scenario. In addition to avoiding collisions with each other, crowds should also avoid collisions with this obstacle.

can reproduce a scenario from the dataset. In this scenario, we set the number of agents in the initialization and control directions to be the same as those in the dataset. Pedestrian agents, represented as discs, mainly avoid collisions with other pedestrians that are close to them in the scene (see Fig. 5a).

Crowd-2. In this scenario, we simulate two groups (50 pedestrians in each group) with control directions inverse to those from the dataset [38]. We randomly locate the agents in each group at one side of the road and randomly choose a velocity for each agent from the dataset in the initialization. The control direction points from the agent's position to the agent's goal on the other side of the road. The reference speed is the magnitude of the initial velocity. Agents are attracted to those in the same group and avoid collisions with other agents, including pedestrians in other groups (see Fig. 5b).

Crowd-3. Based on the benchmark *Crowd-1*, we add a cylindrical obstacle in the center of the road (see Fig. 5c). We also use the dataset [38] in this benchmark. The initialization method for this benchmark is the same as for the benchmark Crowd-2. In our simulations, we set different control directions for different groups and agents in the same group share the same control direction. Agents avoid the obstacle like they avoid other agents.

Crowd behaviors can be slightly adjusted by setting different parameters. In Crowd-1, the control directions of agents are changing, and thus we decrease the weight of $E_{\rm sc}$ to 0.5 to weaken the speed control so that agents can promptly adjust their directions. In Crowd-2, some agents in high-density areas may stop to avoid potential collisions when two crowds are joining, and we increase the weight of $E_{\rm sc}$ to 1.5 to enhance speed control so that these agents can return back to their desired speeds quickly. For a scenario with obstacle such as the one in Crowd-3, the agent-agent and agent-obstacle collision avoidances will make the collision energies $E_{\rm c}^{\rm Ins}$ and $E_{\rm c}^{\rm Anti}$ much larger than other energy terms. To weaken the influence of collision avoidance, we empirically decrease the weights of $E_{\rm c}^{Ins}$ and $E_{\rm c}^{Anti}$ to 0.67. To adjust the weights of $E_{\rm t}^{dir}$ and E_d, we divide the whole road into three zones for each crowd: (1) *obstacle zone*: the area whose distance to the obstacle is about $2m$ (an empirical value); (2) *before obstacle zone*: the area before a crowd arrives at the *obstacle zone*; (3) *after obstacle zone*: the area after a crowd passes by the *obstacle zone*. In the *obstacle zone*, we increase the weight of $E_{\rm t}^{\rm dir}$ to 1.5 to enhance the direction continuity in order to weaken drastic direction changes for agents in high-density areas. In the *after obstacle zone*, we increase the weight of $E_{\rm d}$ to 1.5 to enhance the direction control so that the agents can quickly return back to their goals after they pass by the obstacle.

(a) (b) (c)

Fig. 6. Traffic simulation. (a) Traffic on a twisting 4-lane highway. (b) A combination of cars and crowds. Some pedestrians are walking on the sidewalk. Cars can be treated as obstacles for crowds and vice versa. (c) Congested traffic in an urban crossroad with a traffic lights.

5.3 Traffic

In traffic simulations, vehicle-agents mainly interact with the cars that are adjacent to them in the same lane, avoiding collisions when they are too close and being attracted by the leader cars when the distance to that car becomes too large. However, cars that are changing lanes also interact with the adjacent cars in the target lanes. The control directions for the cars in traffic are the directions of the lanes to which they currently belong.

Traffic-1. With our method, we can simulate traffic on twisting roads with the straight high way traffic dataset [39] (see Fig. 6a). During the initialization step, 80 cars are distributed on the road. The distance between two adjacent cars is chosen randomly from the dataset. We also randomly select the magnitude of the velocity for each agent from the dataset, and the direction of the velocity is the same as the direction of the road on which the agent is driving. The control direction of each agent is always the direction of the road. In this benchmark, the directions of agents in different positions on the twisting road vary.

Our method is general, so we can mix different kinds of agents in the same scenario. In this section, we show two benchmarks: a zebra striped crosswalk and a crossroad with traffic lights.

Traffic-2. In this benchmark, we simulate a case in which people want to cross the road (see Fig. 6b). We use dataset [38] for the crowd and dataset [39] for the traffic. Each pedestrian has a certain possibility of crossing the road. Once the pedestrian starts to cross road, the control direction becomes perpendicular to the road direction and the pedestrian needs to avoid not only other pedestrians, but also the cars around it. At the same time, the surrounding cars need to stop if the pedestrian is in front of them, and the attractive force from the leading cars disappears for these cars. We implement these interactions by adding corresponding objects to the interaction domain of agents.

Traffic-3. Our model can handle congested scenarios with different or heterogeneous agents. Here we simulate agents (15 pedestrians, 15 bicycles, 10 tricycles, and 40 cars) crossing a congested road with a traffic light (see Fig. 6c). We classify the dataset into groups according to the corresponding type of agent in the original data and choose the velocities of the agents from the corresponding class. Furthermore, we classify the four kinds of agents into two types with different motion constraints. The first type includes pedestrians and bicycles, which can overtake the agents in front of them in the same lane. The second type includes tricycles and cars, which cannot overtake the agent in front in the same lane.

(a) (b) (c) (d) (e)

Fig. 7. The avatar in a VR scenario. (a) We provide the user with an immersive VR experience from a first-person perspective with HTC Vive. (b) The avatar drives a car on an high-way road. (c) The avatar drives a car on an urban traffic road. (d) The avatar is walking on the sideroad. (e) The avatar is walking on the crosswalk.

When an agent reaches the crossing, the control direction is the interpolation of the original road direction and the target road direction. The rule for traffic light is not strictly the same as that in the real world. We treat the red traffic light as an obstacle, and agents will gradually stop when they are close to the red traffic light.

5.4 VR Scenarios

Our method can be applied to VR scenarios. We model the user as an avatar in the VR scenario with a first-person perspective (see Fig. 7) (a). The user can sit in a car and observe the movements of other cars around it (see Fig. 7b and 7c). As a walker, the user can also see the traffic flow and other pedestrians at the roadside (see Fig. 7d and 7e).

6 Analysis

6.1 Time Performance

To test the time performance of our method, we simulate a crowd in a scenario with the size of 1,000*1,000. There is no obstacle in the scenario. During the initialization, we randomly locate N agents at random positions. The initial velocities of the agents are randomly copied from the dataset [38]. We set the grid size of the simulation as 10, and the z for Eq. (13) as 2.

In our method, we utilize spatial continuity and velocity continuity to reduce possible collisions among the agents. We use the size of the solution space of the optimization function in Eq. (1) to improve the runtime performance of our simulation. We divide the space into grids and each grid records the agents that belong to it. When we search for the neighbors of each agent, we only need to search the grid to which the agent belongs and the grids that are adjacent to this grid. As a result, our method can reduce the time consumption for multi-agent simulations dramatically (see Fig. 8a).

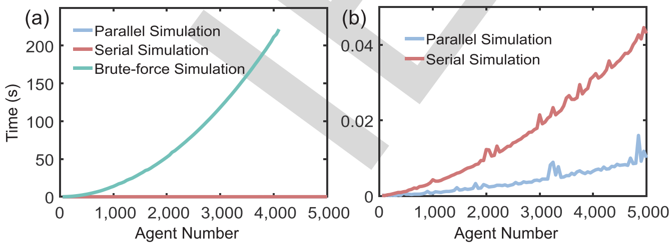


Fig. 8. Time performance. We take a crowd as an example to analyze the time performance of the simulation. (a) We compare the time performance of the brute-force method and our method. With our two search methods, we can improve the performance with 32,298x speedup for 4000 agents. (b) We compare the performance of an 8-threaded parallel implementation with a single-threaded implementation. With parallel computing, as the number of agents increases, the simulation time increases much more slowly. Our method can even simulate 5,000 agents in realtime on a PC machine with a 4.00 GHz Intel i7-6700k CPU processor and 16 GB memory.

Because we can solve the optimization problem for each agent at the same time, we can also easily parallelize our method. Taking the crowd as an example, we compare the time complexity of our simulation using a serial implementation against a parallel implementation (see Fig. 8b). Our parallel implementation can simulate more than 5,000 agents in realtime on a multi-core processor with four cores.

To evaluate the performance of our method further, we compute the running time (seconds per frame) of all the simulation results mentioned in this paper (see Table 1). Our method can achieve real-time performance in various scenarios with multiple kinds of input dataset. The time complexity is not only related to the number of agents in the simulation, but also to the number of classes and the number of data points in each dataset. As a result, similar scenarios with the same number of agents may have different time performances.

6.2 Comparisons

6.2.1 Statistical Comparisons

To demonstrate the plausibility of our method, we compare our simulation results (crowds and highway traffic) with given datasets in terms of the distributions of velocities and distances (the distance to the nearest agent). Velocity is a basic factor used to describe the motion, and minimal distance is the factor used to describe density. We use dataset [38] for two-dimensional bidirectional movements to compare our results with [12], which is the state-of-art optimization method for crowd simulation. Meanwhile, we use the dataset [39] on a four-lane highway to compare our results with [7], which is the state-of-art data-driven traffic simulation method.

Comparison for Crowds. We simulate bidirectional movements of pedestrians in a narrow corridor with the method described in [12] and our method. During the initialization, we set the same number, positions, and velocities of agents as in the dataset. For method [12], the minimal and maximal velocities and the minimal distance from neighbors are estimated from the dataset. Other parameters inherit the configuration of the open source code released by the authors. We also tune parameters so that the method can work well for the scenario. For our method, the control direction of each agent is the direction that points from the current position to the agent's destination. The weight $\mathbf{w} = \{0.8, 1.15, 1.2, 0.8, 0, 0.85, 0, 1.2\}$, which corresponds to the items in Table 2. For both methods, the preferred speed of each agent is the average speed of the corresponding agent in the dataset.

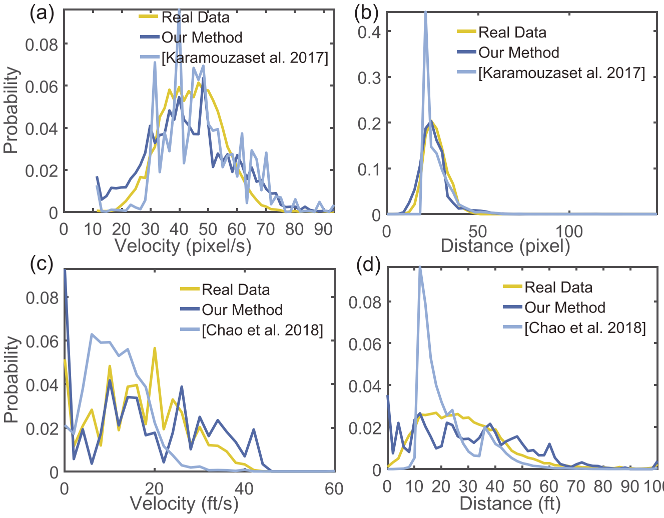


Fig. 9. The distributions of velocity and minimal distance. We compare the probability distributions between our simulation results, existing methods, and input datasets. (a)-(b). The comparison for the crowd simulated. (c)-(d) The comparison for traffic simulated on a straight 4-lane road.

Comparison for Traffic. We simulate traffic in a straight four-lane highway like the dataset [39] using both method [7] and our method. In this comparison, we initialize the number, positions, and velocities of agents in our method to be the same as the dataset. The control direction is the direction of the road. The weight $\mathbf{w} = \{1.0, 5.0, 1.0, 1.0, 20.0, 3.0, 1.0, 0.0\}$, which corresponds to the items in Table 2. For method [7], we set the parameters to be the same as the original parameters. The traffic in method [7] consists of 15 traffic flows. The initialization of each flow is same as in the dataset.

The distributions of velocity and minimal distance for each method are shown in Fig. 9. We compute the difference between simulation results and the dataset as the scores for each method. We divide all the values of each metric into 30 intervals and compute the probability for each interval. The difference between simulation results and the dataset is the sum of the magnitudes of the probability difference in the intervals. The scores of each method are given in Table 3.

The distributions of velocity and minimal distance in our simulations are closer to those in the input data. Although our method selects velocities for agents directly form the dataset, the selected velocities are controlled by Eq. (1). Because our method has velocity distributions that are closer to the input data for crowd simulation, our approach is better at capturing the motion characteristics of a multi-agent system as compared to prior methods ([12], [7]).

In the compared methods ([12] and [7]), the spikes in the distance distribution are not only due to the hard constraints on the separation distance, but also the optimization functions of these methods trying to find similar optimal velocities for different agents. Therefore, the distances of different agents are similar when agents reach the balance of different optimization terms.

6.2.2 Trajectory Comparisons

Several quantitative metrics can be used to compare real data against simulation data [13], [41], [42]. To evaluate the time series in sequence of agents' movements for crowds, we employ the absolute difference metric (ADM) and the path length metric (PLM) proposed by Wolinski et al. [13] as they are straightforward comparing to other quantitative metrics. We simulate the movements of pedestrians on a street using the implicit method [12], the data-driven method (PAG) [19], and our method. We set the same number, positions, and velocities of agents as in the reference dataset [29] when performing the initialization. In addition, we set the control directions to be the same as those in the dataset.

The ADM and PLM for each method are shown in Table 4. Experiments show that our method achieves a lowest score compared to [12] and [19] for crowds. This means that the trajectories generated by our method are more realistic than those generated by methods of [12] and [19]. Compared to the implicit method [12] which is not data-driven, our approach uses real datasets so that it can generate more realistic detailed behaviors. The PAG method [19] searches trajectories only depending on the predicted temporal perception patterns and the distance to the goal, which may produce potential discontinuous velocities. On the contrary, our method can enforce continuous velocity by introducing a velocity continuity energy function.

TABLE 3
Benchmark Scores 1: Used to Measure the Statistical Closeness to the Real-World Datasets

	Velocity			Distance		
	Real	Ours	Others	Real	Ours	Others
Crowd	0.0	0.4132	0.4793	0	0.2691	0.5913
Traffic	0.0	0.2507	0.3766	0	0.2383	0.3475

The scores are the difference between simulation results and the dataset. A lower score for our method versus [12] for crowds and [7] for traffic. This demonstrates that the trajectories and behaviors generated by our method are closer to those generated by prior methods.

6.3 Our Simulation Results with or without Using Dataset

To explore the performance of our data-driven scheme, we compare our simulation results with and without using dataset in terms of the distributions of velocities and minimal distances. We use the dataset [39] on a four-lane highway for our experiments. We use the same initialization method and parameter values as those in Section 6.2. For the method without using dataset, we suppose that the cars move in one direction and compute $\mathbf{v}_{i,n}$ ($\|\mathbf{v}_{i,n}\| \in [v^*_{\min}, v^*_{\max}]$) by minimizing Eq. (2). The underlying assumption is that the minimum and maximum magnitudes of velocities from real-world datasets are reasonable values to restrict the range of the magnitude of velocity.

The distributions of velocity and minimal distance for the comparison are shown in Fig. 10. The velocity difference to

TABLE 4
Benchmark Scores 2: Used to Measure the Trajectory Closeness to the Real-World Datasets

	Real	IMPLICIT	PAG	Ours
ADM	0.0	37.373	65.4278	3.10986
PLM	0.0	20.3423	117.486	3.9529

The scores show the differences between the simulation results and the real-world dataset.

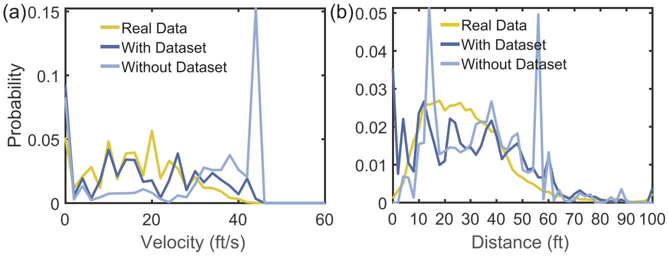


Fig. 10. The distributions of velocity and minimal distance for comparison of the results with and without using dataset. (a) Probability distributions of velocity. (b) Probability distributions of minimal distance.

the dataset of our method using dataset (0.2507) is smaller than that of the method without using dataset (0.6132). The minimal distance difference score of our method using dataset (0.2383) is also smaller than that of the method without using dataset (0.2649). The comparison results indicate that the data-driven scheme can improve the plausibility of simulation results.

7 User Studies and Evaluation

We conduct two user studies to evaluate the plausibility of our method and one user study to show a better user experience through VR. The weights for the user study are shown in Table 5. The eight cases in the first user study are conducted from an overhead view to show the agents' movements. In the second user study, we adopt the agent's view in each case, meaning that the view is closer to that of a participant in his/her daily life. In the third user study, we compare the results as shown in immersive VR and those shown on a desktop in four different scenarios or agents' views.

Experiment Goals & Expectations. For the first user study, we hypothesize that the results simulated by our method will exhibit more plausible movements than prior multi-agent methods. For the second user study, we hypothesize that our method results in a better user experience than the prior methods. Therefore, participants will significantly prefer our method over the prior methods in these evaluations. In the third user study, we hypothesize that the results shown in VR can produce a better user experience that those shown on a desktop.

Comparison Methods. For crowd simulation, we compare our method with the method in [12] which is a state-of-art physical-based method for crowd simulation. We also use the dataset [29] in crowd simulation. For traffic simulation, we compare our method with the method in [7], which is a state-of-art data-driven method on traffic simulation. Here we use the dataset [39]. All 2D trajectories generated from simulation methods or extracted from datasets are assigned to 3D characters. We also compare mixed traffic results shown in VR and those shown on a desktop.

Environments. In the first and second user study, we used three scenarios for crowd simulation. The scenario with the dataset [29] is in a street with 18 agents. The other two scenarios are the one in which two crowds (100 agents in total) encounter each other and the scenario in which 36 agents are located on a circle moving towards the opposite positions. We also use three scenarios for traffic simulation. The scenario with the dataset [39] is on a straight 4-lane road with 156 agents. The other two scenarios are on a twisting 2-lane road with 80 agents and on a twisting 4-lane road with 200 agents. In the third user study, we use one instance for the scenario with 50 cars and a car's view. We also use three instances for the scenario with 35 cars and 30 pedestrians. In each instance, we use different agent views: one from a car's view, one from the view of a pedestrian walking on a zebra crossing, and one from the view of a pedestrian walking on a sidewalk. In the VR scenarios, head turning is controlled by a HTC Vive headset, and the user is allowed to turn his/her head freely with a fixed position in a moving agent.

TABLE 5
The Weights for the User Study

		E_t^{dir}	E_t^{L}	E_c^{Ins}	E_c^{Anti}	E_a	E_d	E_p	E_{sc}
Street		1.0	1.0	1.0	1.0	0.0	1.0	0.0	0.5
Hallway		1.0	1.0	1.0	1.0	0.0	1.2	0.0	1.2
Circle		1.0	1.0	0.5	0.5	0.0	1.0	0.0	1.0
Straight		0.5	0.5	1.0	1.0	1.2	3.0	1.0	0.2
Twist-2Lane		0.5	0.5	1.0	1.0	1.0	3.0	1.0	2.0
Twist-4Lane		0.2	0.6	1.0	1.0	2.0	3.0	10.0	1.0
VR-2Lane		5.0	1.0	1.0	1.0	2.0	5.0	1.0	10.0
VR	Pedestrian	1.0	1.0	1.0	1.0	0.0	1.5	1.0	10.0
	Car	5.0	1.0	1.0	1.0	2.0	5.0	1.0	10.0

This table gives the weights for the direction continuity E_t^{dir}, the speed continuity E_t^{L}, instantaneous collision avoidance E_c^{Ins}, anticipated collision avoidance E_c^{Anti}, attraction E_a, direction control E_d, position control E_p, and speed control E_{sc} in each scenario.

Experimental Design. We conduct the user studies based on a paired-comparison design. For the scenarios with a dataset, we design two comparison pairs: the dataset versus our method, and the dataset versus the prior method. We design one comparison pair for each scenario without a dataset: our method versus the prior method. For each pair, we show two pre-recorded videos in a side-by-side comparison. The order of the scenarios was random. The position (left or right) of each method was also random. For the scenarios for VR versus desktop comparison, we ask the participants to answer the questionnaire after see the scenarios via VR and the scenarios via desktop.

Metrics. In each user study, participants were asked to choose a score using a 7-point Likert scale, in which 1 means that the result presented on the left is strongly plausible, 7 means that the result presented on the right is strongly plausible, and 4 means no preference for either method. To combine the user study results in the same scale, we transfer the score for each method to a certain side when we deal with the scores.

7.1 User Study with an Overhead View

The user studies for crowd simulation and traffic simulation with an overhead view were completed by 26 participants (15 females and 11 males). We performed two-sample t-tests for the scenarios with datasets (one for crowd simulation and another for traffic simulation). We hypothesize that the mean value of our method is bigger than that of the prior method. Meanwhile, we performed one-sample t-tests for the scenarios without datasets (two scenarios for crowd simulation and two for traffic simulation), hypothesizing that the mean value of our method is bigger than 4, which

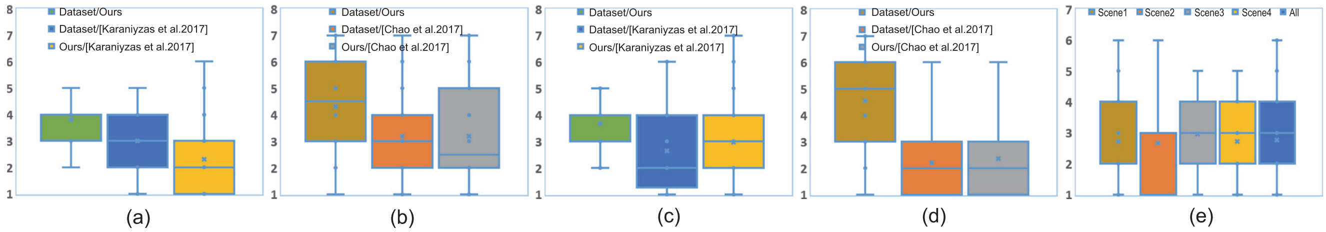


Fig. 11. Plausibility scores of the user study. We use a 7-point Likert scale to measure the plausibility of the methods. The lower the score, the more the participants prefer the method on the left; the higher the score, the more the participants prefer the method on the right. (a)The statistics for crowd simulation with an overhead view. Participants cannot tell the difference between the dataset and our method. Compared to method [12], the participants think the results of our method are more plausible. (b) The statistics for traffic simulation with an overhead view. Our method gets a higher score than method [12] when compared with the dataset. We also get better results in the user study with the dataset. (c) The statistics for crowd simulation from an agent view. Our method is closer to the dataset. The participants believe that the results of our method are more plausible than those of the prior method. (d) The statistics for traffic simulation from an agent's view. Our method has a significantly larger score than method [7] in the user study with the dataset. Our method also shows better performance in the user study without the dataset. (e) The statistics of the user study for the comparison of VR and desktop. The scores are transferred so that VR is supposed on the left. The scenarios shown through VR have better scores.

indicates no difference. Overall, participants believed that our method was more plausible than the compared methods for both crowd simulation and traffic simulation. Fig. 11 (a)-(b) shows details about the scores for each comparison.

User Study for Crowd Simulation. For the scenario with the dataset, our method's mean score is significantly larger than the prior method's mean plausibility score ($t(25) = 2.9111$, $p = 0.0027 < 0.01$). For the scenarios without datasets, our method's mean score shows a significant difference from the hypothetical mean ($t(51) = -8.7555$, $p < 0.001$).

User Study for Traffic Simulation. For the scenarios with datasets, our method's mean of the score is significantly larger than the prior method's mean plausibility score ($t(25) = 2.4422$, $p = 0.0091 < 0.01$). For the scenarios without datasets, our method's mean score shows a significant difference from the hypothetical mean ($t(51) = -3.0169$, $p = 0.002 < 0.01$).

7.2 User Study with an Agent View

The user studies for crowd simulation and traffic simulation from an agent's view were completed by 28 participants (17 females and 11 males). For the user study from an agent view, we also performed two-sample t-tests for the scenarios with datasets hypothesizing that our method has a larger mean score than the prior method. For the scenarios without datasets, we performed one-sample t-tests hypothesizing that the mean value of our method is larger than 4 (no difference). Overall, participants also judged that our method is more plausible than the prior methods. The statistics of the participants' plausibility evaluations can be found in Fig. 11 (c)-(d).

User Study for Crowd Simulation. For the scenario with a dataset, the mean plausibility score of our Heter-Sim is significantly larger ($t(27) = 2.6692$, $p = 0.005 < 0.01$) than the method [12]. The mean score of our method has a significantly superior to the hypothetical mean ($t(55) = -5.0281$, $p < 0.001$) for the scenarios without datasets.

User Study for Traffic Simulation. For the scenario with a dataset, the mean score of our method is significantly larger than the mean score of the prior method ($t(27) = 6.4890$, $p < 0.001$). For the scenarios without datasets, the mean score of our method shows a significant difference from the hypothetical mean with $t(55) = -8.0381$ and $p < 0.001$.

7.3 User Study via VR or Desktop

The user studies for the comparison between VR and desktop were taken by 28 participants (14 females and 14 males). We performed one-sample t-tests for the four instances by hypothesizing that the mean score of VR is bigger than 4 (no difference). Overall, participants believed that the results shown with VR are more plausible than those shown with a desktop. Fig. 11e shows the details about the scores for each comparison. In each scenarios, the score of VR is significantly better than that of desktop. $t(27) = -5.0138$, $p < 0.001$ for the first scenario, $t(27) = -4.16478$, $p < 0.001$ for the second scenario, $t(27) = -3.9890$, $p < 0.001$ for the third scenario, and $t(27) = -5.7564$, $p < 0.001$ for the last scenario. In total, the mean score for VR shows a significant difference from the hypothetical mean ($t(111) = -9.3485$, $p < 0.001$).

8 Conclusion, Limitation and Future Work

We present a novel and general data-driven optimization method that can generate plausible behaviors for heterogeneous agents in different scenarios. We demonstrate our model's generalizability by simulating human crowds, traffic, and mixed traffic in multiple scenarios. To the best of our knowledge, this is the first data-driven multi-agent method that is applicable to such different simulation scenarios and that mixes different kinds of agents (e.g., vehicles and pedestrians).

The simulation results of our method are plausible. We compare our results with prior methods in the same scenarios and by conducting three user studies with various scenarios from different views and analyzing the statistical results of the user studies. Our method can generate results that are closer to the original datasets, than those achieve with the prior methods. In addition, our model is fast and can be used for interactive simulations (Table 1). We also demonstrate that the plausibility of our method can be increased via VR by performing a user study comparing the results via VR or desktop.

Our method can simulate behaviors that are different from those of the input datasets. First, our method can generate larger and denser groups than those in the input datasets (Fig. 5). Second, our method can simulate scenarios that may differ from those of the input datasets (Figs. 5b, 6a). Third, our method can mix different kinds of agents in the same scenario (Fig. 6b and 6c).

Limitations. Although our approach can generate various behaviors even with a simple, sparse input dataset, the actual performance of our approach can vary based on the datasets. For example, if the dataset only has two magnitudes of velocity in it, the velocity of a car attempting to stop and move again after several seconds will not be continuous. Because our method uses a forward Euler integration scheme, the stability of our simulation depends on the size of the timestep. An implicit integration scheme [12] can be introduced to improve the stability. We represent agents as rectangular objects or discs. More precise geometrical shapes should be used to implement better collision avoidance.

As part of future work, our work can be extended in many ways. The input data is not limited to the real datasets and users can also use simulation results to direct certain behaviors. Therefore, the variety or diversity of simulation results can be dramatically increased. We could add traditional context-aware methods to our work to create a variety of behaviors in multiple agents, which would improve the realism of the simulation results. The idea of reducing the solution space according to the continuity of movement can be applied to optimization problems in animation. We can also introduce other additional sensory information such as hearing to increase the realism of interactions among agents [36]. To make our simulation results more realistic, we also plan to use portions of real velocity profiles.

Our model can be extended to other areas. The key idea of our method can be extended to data-driven methods to simulate other particle systems. If we treat the vertex as the agent in our system and the connection between vertices as the relationship, our framework can also be applied to data-driven body animation [43]. Because we model the decision-making process as an energy-based optimization problem, this idea may be applicable to path planning for robotics and unmanned aerial vehicles. Finally, we want to further evaluate the benefits of our simulator in VR and training scenarios.

ACKNOWLEDGMENTS

Xiaogang Jin was supported by the National Key R&D Program of China (Grant No. 2017YFB1002600), Artificial Intelligence Research Foundation of Baidu Inc., the Key Research and Development Program of Zhejiang Province (Grant No. 2018C01090), and the National Natural Science Foundation of China (Grant No. 61972344). Dinesh Manocha is supported in part by ARO Grant No. W911NF-19-1-0069 and Intel. Some of the work was done while the first author was an intern at Baidu Research. The authors thank the reviewers for valuable suggestions.

REFERENCES

[1] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," *ACM SIGGRAPH Comput. Graphics*, vol. 21, no. 4, pp. 25–34, 1987.

[2] X. Wang, J. Ren, X. Jin, and D. Manocha, "Bswarm: Biologically-plausible dynamics model of insect swarms," in *Proc. 14th ACM SIGGRAPH/Eurographics Symp. Comput. Animation*, 2015, pp. 111–118.

[3] I. Karamouzas, B. Skinner, and S. J. Guy, "Universal power law governing pedestrian interactions," *Phys. Rev. Lett.*, vol. 113, no. 23, p. 238701, 2014.

[4] J. Ondřej, J. Pettré, A.-H. Olivier, and S. Donikian, "A synthetic-vision based steering approach for crowd simulation," *ACM Trans. Graphics*, vol. 29, no. 4, pp. 123:1–123:9, 2010.

[5] T. B. Dutra, R. Marques, J. B. Cavalcante-Neto, C. A. Vidal, and J. Pettré, "Gradient-based steering for vision-based crowd simulation algorithms," *Comput. Graphics Forum*, vol. 36, no. 2, pp. 337–348, 2017.

[6] J. Pettré, J. Ondřej, A.-H. Olivier, A. Cretual, and S. Donikian, "Experiment-based modeling, simulation and validation of interactions between virtual walkers," in *Proc. ACM SIGGRAPH/Eurographics . Comput. Animation*, 2009, pp. 189–198.

[7] Q. Chao, Z. Deng, J. Ren, Q. Ye, and X. Jin, "Realistic data-driven traffic flow animation using texture synthesis," *IEEE Trans. Vis. Comput. Graphics*, vol. 24, no. 2, pp. 1167–1178, Feb. 2018.

[8] J. Sewall, D. Wilkie, and M. C. Lin, "Interactive hybrid simulation of large-scale traffic," *ACM Trans. Graphics*, vol. 30, no. 6, pp. 135:1–135:12, 2011.

[9] S. Kim, A. Bera, A. Best, R. Chabra, and D. Manocha, "Interactive and adaptive data-driven crowd simulation," in *Proc. IEEE Virtual Reality Conf.*, 2016, pp. 29–38.

[10] K. Jordao, J. Pettre, M. Christie, and M.-P. Cani, "Crowd sculpting: A space-time sculpting method for populating virtual environments," *Comput. Graphics Forum*, vol. 33, no. 2, pp. 351–360, 2014.

[11] E. Ju, M. G. Choi, M. Park, J. Lee, K. H. Lee, and S. Takahashi, "Morphable crowds," *ACM Trans. Graphics*, vol. 29, no. 6, pp. 140:1–140:10, 2010.

[12] I. Karamouzas, N. Sohre, R. Narain, and S. J. Guy, "Implicit crowds: optimization integrator for robust crowd simulation," *ACM Trans. Graphics*, vol. 36, no. 4, pp. 136:1–136:13, 2017.

[13] D. Wolinski, S. J. Guy, A.-H. Olivier, M. Lin, D. Manocha, and J. Pettre, "Parameter estimation and comparative evaluation of crowd simulations," *Comput. Graphics Forum*, vol. 33, no. 2, pp. 303–312, 2014.

[14] G. Berseth, M. Kapadia, B. Haworth, and P. Faloutsos, "Steerfit: automated parameter fitting for steering algorithms," in *Proc. ACM SIGGRAPH/Eurographics Symp. Computer. Animation*, 2014, pp. 113–122.

[15] I. Karamouzas and M. Overmars, "Simulating and evaluating the local behavior of small pedestrian groups," *IEEE Trans. Vis. Comput. Graphics*, vol. 18, no. 3, pp. 394–406, Mar. 2012.

[16] K. H. Lee, M. G. Choi, Q. Hong, and J. Lee, "Group behavior from video: A data-driven approach to crowd simulation," in *Proc. ACM SIGGRAPH/Eurographics . Comput. Animation*, 2007, pp. 109–118.

[17] Q. Chao, J. Shen, and X. Jin, "Video-based personalized traffic learning," *Graphical Models*, vol. 75, no. 6, pp. 305–317, 2013.

[18] C. D. Boatright, M. Kapadia, J. M. Shapira, and N. I. Badler, "Context-sensitive data-driven crowd simulation," in *Proc. 12th ACM SIGGRAPH Int. Conf. Virtual-Reality Continuum Appl. Ind.*, 2013, pp. 51–56.

[19] P. Charalambous and Y. Chrysanthou, "The pag crowd: A graph based approach for efficient data-driven crowd simulation," *Comput. Graphics Forum*, vol. 33, no. 8, pp. 95–108, 2014.

[20] H. Bi, T. Mao, Z. Wang, and Z. Deng, "A data-driven model for lane-changing in traffic simulation," in *Proc. ACM SIGGRAPH/Eurographics Symp. Comput. Animation*, 2016, pp. 149–158.

[21] W. Li, D. Wolinski, and M. C. Lin, "City-scale traffic animation using statistical learning and metamodel-based optimization," *ACM Trans. Graphics*, vol. 36, no. 6, pp. 200:1–200:12, 2017.

[22] D. Wilkie, J. Sewall, and M. Lin, "Flow reconstruction for data-driven traffic animation," *ACM Trans. Graphics*, vol. 32, no. 4, pp. 89:1–89:10, 2013.

[23] S. Yoon, M. Kapadia, P. Sahu, and V. Pavlovic, "Filling in the blanks: Reconstructing microscopic crowd motion from multiple disparate noisy sensors," in *Proc. IEEE Winter Appl. Comput. Vis. Workshops*, 2016, pp. 1–9.

[24] G. Qiao, S. Yoon, M. Kapadia, and V. Pavlovic, "The role of data-driven priors in multi-agent crowd trajectory estimation," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 4710–4717.

[25] B. Yersin, J. Maïm, J. Pettré, and D. Thalmann, "Crowd patches: populating large-scale virtual environments for real-time applications," in *Proc. Symp. Interactive 3D Graphics Games*, 2009, pp. 207–214.

[26] Y. Li, M. Christie, O. Siret, and R. Kulpa, "Cloning crowd motions," in *Proc. 11th ACM SIGGRAPH / Eurographics Symp. Comput. Animation*, 2012, pp. 201–210.

[27] K. Hyun, M. Kim, Y. Hwang, and J. Lee, "Tiling motion patches," *IEEE Trans. Vis. Comput. Graphics*, vol. 19, no. 11, pp. 1923–1934, Nov. 2013.

[28] Y.-C. Lai, S. Chenney, and S. Fan, "Group motion graphs," in *Proc. ACM SIGGRAPH/Eurographics Symp. Comput. Animation*, 2005, pp. 281–290.

[29] A. Lerner, Y. Chrysanthou, and D. Lischinski, "Crowds by example," *Comput. Graphics Forum*, vol. 26, no. 3, pp. 655–664, 2007.

[30] M. Zhao, S. J. Turner, and W. Cai, "A data-driven crowd simulation model based on clustering and classification," in *Proc. IEEE/ACM 17th Int. Symp. Distrib. Simul. Real Time Appl.*, 2013, pp. 125–134.

[31] W. Li, D. Wolinski, J. Pettre, and M. C. Lin, "Biologically-inspired visual simulation of insect swarms," *Comput. Graphics Forum*, vol. 34, no. 2, pp. 425–434, 2015.

[32] P. M. Kielar and A. Borrmann, "Spice: A cognitive agent framework for computational crowd simulations in complex environments," *Auton. Agents Multi-Agent Syst.*, vol. 32, no. 3, pp. 387–416, 2018.

[33] W. Liu, V. Pavlovic, K. Hu, P. Faloutsos, S. Yoon, and M. Kapadia, "Characterizing the relationship between environment layout and crowd movement using machine learning," in *Proc. 10th Int. Conf. Motion Games*, 2017, pp. 2:1–2:6.

[34] D. Wolinski, M. C. Lin, and J. Pettré, "Warpdriver: Context-aware probabilistic motion prediction for crowd simulation," *ACM Trans. Graphics*, vol. 35, no. 6, pp. 164:1–164:11, 2016.

[35] Z. Ren, P. Charalambous, J. Bruneau, Q. Peng, and J. Pettré, "Group modeling: A unified velocity-based approach," *Comput. Graphics Forum*, vol. 36, no. 8, pp. 45–56, 2017.

[36] Y. Wang, M. Kapadia, P. Huang, L. Kavan, and N. I. Badler, "Sound localization and multi-modal steering for autonomous virtual agents," in *Proc. 18th Meeting ACM SIGGRAPH Symp. Interactive 3D Graphics Games*, 2014, pp. 23–30.

[37] R. Bridson and M. Müller-Fischer, "Fluid simulation: Siggraph 2007 course notes video files associated with this course are available from the citation page," in *Proc. ACM SIGGRAPH Courses*, 2007, pp. 1–81.

[38] J. Zhang, W. Klingsch, A. Schadschneider, and A. Seyfried, "Ordering in bidirectional pedestrian flows and its influence on the fundamental diagram," *J. Statistical Mech.: Theory Exp.*, vol. 2012, no. 2, 2012, Art. no. P02002.

[39] "Next generation simulation," 2013. [Online]. Available: http://ops.fhwa.dot.gov/traffican-alysistools/ngsim.htm

[40] B. K. Horn and B. G. Schunck, "Determining optical flow," *Artif. Intell.*, vol. 17, no. 1–3, pp. 185–203, 1981.

[41] S. J. Guy, J. van den Berg, W. Liu, R. Lau, M. C. Lin, and D. Manocha, "A statistical similarity measure for aggregate crowd dynamics," *ACM Trans. Graphics*, vol. 31, no. 6, pp. 190:1–190:11, 2012.

[42] H. Wang, J. Ondej, and C. OSullivan, "Trending paths: A new semantic-level metric for comparing simulated and real crowd data," *IEEE Trans. Vis. Comput. Graphics*, vol. 23, no. 5, pp. 1454–1464, May 2017.

[43] M. Kim, G. Pons-Moll, S. Pujades, S. Bang, J. Kim, M. J. Black, and S.-H. Lee, "Data-driven physics for human soft tissue animation," *ACM Trans. Graphics*, vol. 36, no. 4, pp. 54:1–54:12, 2017.

Jiaping Ren received the BSc degree from the Department of Mathematics, Zhejiang University of Technology, P. R. China, in 2013. She is working toward the PhD degree in the StateKey Laboratory of CAD&CG, Zhejiang University, P. R. China. Her main research interests include traffic simulation, crowd animation, collective behaviors of bird flocks, and insect swarm animation.

Wei Xiang received the BSc degree in computer science and technology from Shandong University, P. R. China, in 2016. She is working toward the PhD degree in the State Key Laboratory of CAD&CG, Zhejiang University, P. R. China. Her main research interests include crowd animation, insect swarm animation, traffic simulation and autonomous driving.

Yangxi Xiao received the BSc degree in digital media technology from Zhejiang University, in 2014. She is working toward the PhD degree from the State Key Lab of CAD&CG, Zhejiang University, P. R. China. Her main research interests include digital video processing and editing.

Ruigang Yang received the MSc degree from Columbia University and the PhD degree from the University of North Carolina at Chapel Hill. He is the chief scientist of 3D vision with Baidu Research and a full professor of computer science with the University of Kentucky (on leave). His research interests include computer graphics and computer vision, in particular in 3D reconstruction and 3D data analysis. He has published more than 100 papers, which, according to Google Scholar, has received close to 10000 citations with an h-index of 47 (as of 2017). He has received a number of awards, including US NSF Career award in 2004 and the Deans Research Award at the University of Kentucky in 2013. He is a senior member of the IEEE.

Dinesh Manocha is the Paul Chrisman Iribe chair in computer science & electrical and computer engineering with the University of Maryland College Park. He is also the Phi Delta Theta/Matthew Mason distinguished professor emeritus of computer science with the University of North Carolina - Chapel Hill. His research interests include multi-agent simulation, virtual environments, physicallybased modeling, and robotics. He has received many awards, including Alfred P. Sloan Research Fellow, the NSF Career Award, the ONR Young Investigator Award, and the Hettleman Prize for scholarly achievement. His group has developed a number of packages for multi-agent simulation, crowd simulation, and physicsbased simulation that have been used by hundreds of thousands of users and licensed to more than 60 commercial vendors. He has published more than 480 papers and supervised more than 35 PhD dissertations. He is an inventor of nine patents, several of which have been licensed to industry. His work has been covered by the New York Times, NPR, Boston Globe,Washington Post, ZDNet, as well as DARPA Legacy Press Release. He is a fellow of the AAAI, AAAS, ACM, and IEEE and also received the Distinguished Alumni Award from IIT Delhi. See http://www.cs.umd.edu/people/dmanocha.

Xiaogang Jin received the BSc degree in computer science and the MSc and PhD degrees in applied mathematics from Zhejiang University, P. R. China, in 1989, 1992, and 1995, respectively. He is a professor with the State Key Laboratory of CAD&CG, Zhejiang University. His current research interests include digital geometry processing, geometric modeling, 3D printing, virtual try-on, insect swarm simulation, traffic simulation, implicit surface modeling and applications, creative modeling, sketch-based modeling, and image processing. He received an ACM Recognition of Service Award in 2015. He is a member of the IEEE and ACM.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.