Text-Guided Synthesis of Crowd Animation

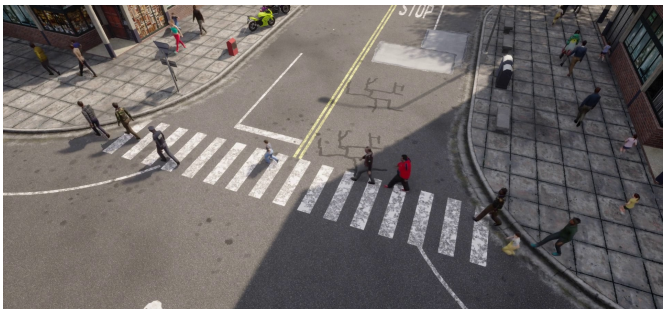
Xuebo Ji
xueboji.cs@gmail.com
The University of Hong Kong
Centre for Transformative Garment Production (TransGP)
Hong Kong, China

Zherong Pan
zrpan@global.tencent.com
LightSpeed Studios
Seattle, WA, USA

Xifeng Gao
xifgao@global.tencent.com
LightSpeed Studios
Seattle, WA, USA

Jia Pan
jpan@cs.hku.hk
The University of Hong Kong
Centre for Transformative Garment Production (TransGP)
Hong Kong, China

Many people are entering from the entrance located at the bottom right of the map, passing through the right access pathway and getting into the top right door. The crowd in another direction that wants to leave the subway passes through the left passage from the top left of the map and leaves at the bottom left.



Some humans are moving from the right of the map and crossing the crosswalk to get to the left. Some people also enter from the right are walking along the left side of the building and finally exit at the top right of the map. Others get into the map from the top left and move along the right of the other building, then leave at the left.

Figure 1: *Given the environment maps, we illustrate two crowd animation scenarios generated using the text prompts below.*

ABSTRACT

Creating vivid crowd animations is core to immersive virtual environments in digital games. This work focuses on tackling the challenges of the crowd behavior generation problem. Existing approaches are labor-intensive, relying on practitioners to manually craft the complex behavior systems. We propose a machine learning approach to synthesize diversified dynamic crowd animation scenarios for a given environment based on a text description input. We first train two conditional diffusion models that generate text-guided agent distribution fields and velocity fields. Assisted by local navigation algorithms, the fields are then used to control multiple groups of agents. We further employ Large-Language Model (LLM) to canonicalize the general script into a structured sentence for more stable training and better scalability. To train our diffusion models, we devise a constructive method to generate random environments and crowd animations. We show that our trained diffusion models can generate crowd animations for both unseen environments and novel scenario descriptions. Our method paves the way towards automatic generating of crowd behaviors for virtual environments. Code and data for this paper are available at: https://github.com/MLZG/Text-Crowd.git.

SIGGRAPH Conference Papers '24, July 27-August 1, 2024, Denver, CO, USA

ACM ISBN 979-8-4007-0525-0/24/07…$15.00
https://doi.org/10.1145/3641519.3657516

CCS CONCEPTS

• **Computing methodologies** → **Procedural animation**.

KEYWORDS

Diffusion Model, Multi-Agent Navigation, Collision Avoidance, Crowd Simulation

ACM Reference Format:
Xuebo Ji, Zherong Pan, Xifeng Gao, and Jia Pan. 2024. Text-Guided Synthesis of Crowd Animation. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers '24 (SIGGRAPH Conference Papers '24), July 27-August 1, 2024, Denver, CO, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3641519.3657516

1 INTRODUCTION

As a major part of the modern game production pipeline, creating vivid virtual content is labor-intensive. Groups of artists take years to figure out the game logic, create visual environments, and design intelligent character behaviors. In response, a large body of engineering and research efforts have been devoted to accelerating the content creation pipeline. These efforts have proven to be successful in many ways including automatic 3D model and environment synthesis [Fisher et al. 2012; Kelly et al. 2018], procedural modeling [Cogo et al. [n. d.]], and character motion synthesis [Zhu et al. 2023]. Comparatively, the collective dynamic behavior of groups of characters has received relatively less attention. Prior works in this area such as [Lemonari et al. 2022] focus on tools for editing crowd animations, where artists still need to manually craft and fine-tune the motion trajectories for each environment, which requires a non-trivial learning curve. Further, these tools ignore the fact that a virtual environment contains rich prior information to generate plausible motion trajectories of the crowd. For example, humans in a city should cross the road at zebra crossings and traverse on sidewalks. Regretfully, such environmental information has never been used in prior crowd animation generators. In this work, we propose a novel approach that can automatically generate crowd animation behaviors from intuitive text descriptions that are compatible with environment settings.

Automatic generation of environment-compatible crowd animations is a significantly challenging task for several reasons. First, the relationship between the environment and agent behaviors is multi-modal and hard to represent. To exactly describe these relationships requires tedious annotations or planning of paths. Further, these relationships are non-deterministic, making the exact description undesirable. For example, a social-rule following agent might always walk along the sidewalks, while a disobedient agent can sometimes walk on the roadway. Second, the agents can exhibit diversified behaviors in different environments and the complexity and dimensionality of agent behaviors vary drastically as well. For example, an empty playground can have relatively few agents playing sports requiring fast motions, while a large crowd in a convention center moves slowly and stops here and there. Such inhomogeneous agent distributions and diversified behaviors can hardly be represented in a unified manner. Existing works use finite state machines, such as behavior trees [Ghzouli et al. 2023], which is again labor-intensive and becomes the bottleneck of the current game design pipeline.

To the best of our knowledge, we introduce the first-ever pipeline that targets at language-guided generation of environment-compatible scenarios involving a large number of agents navigating in real-time. To this end, we use the local navigation algorithm RVO [Van den Berg et al. 2008] as our backend simulator to generate collision-free agent trajectories. We further follow [He et al. 2016; Kim 2004; Luo et al. 2021] and assume agents are divided into multiple groups with similar goals, where each group is controlled by a common velocity field. Based on these assumptions, our method takes as input a map of the environment, and a general script describing the behavior of multiple groups of agents. We then utilize the powerful Large Language Models (LLM) to canonicalize the script into a structured sentence for each group of agents. Each structured sentence is then input to a Latent Diffusion Model (LDM) to predict a velocity field for navigating the agent group. Finally, we devise a constructive method for synthesizing a dataset of randomized environments and agent behaviors. Combined, we show that our method generalizes well to unseen environment maps and agent behaviors.

2 RELATED WORK

We review related work in automatic content creation, crowd animation and simulation, and language-guided generative models.

2.1 Automatic Content Creation

Due to their significant societal impact, the area of content creation represents decades of combined efforts in computer graphics, computer vision, and computational physics. The vast majority of early works on content creation are focused on procedure modeling [Cogo et al. [n. d.]], aiming at generating static 3D models or scene layouts. In parallel, the development of physics-based modeling [Erleben et al. 2005] and character locomotions [Guo et al. 2015] allows for the generation of animated motions for passive physical phenomena or actuated characters. However, a common drawback of these techniques is their case-by-case nature, i.e., a separate set of techniques is needed for generating different types of models, such as buildings, chairs, and planes, or animations, such as balancing, walking, and grasping. Integrating all these techniques into a comprehensive content creation tool could be a formidable task, which largely compromises their domain of applications. Thanks to the maturity of deep generative models, the past five years have witnessed the success of generalized machine learning models that can create contents across multiple domains. For example, the 3D-GAN [Wu et al. 2016] encodes in its latent space 3D models with drastically different topology and geometry. Hu et al. [2020] learns to generate structured layouts of floor plans with a rich variety. Peng et al. [2021] proposes an algorithm to learn general character motions with multiple skills. Our method also falls into the category of diversified content creation of crowd simulation scenarios, where our technique can adapt to arbitrary given environments.

2.2 Crowd Animation

Relatively few works have focused on the creation of collective motions for multiple agents, which is core to a vivid virtual environment. Early works in this area provide efficient tools to create and edit the navigating behavior of crowds. For example, Pettre et al. [2005] represents the environment as a graph and searches for agent paths on the graph. Oshita and Ogiwara [2009] proposes a sketch-based interface for crowd motions. Kim et al. [2014] restrict agent motions to an editable cage. However, these works are still labor-intensive and require artists to craft the approximate motion paths. Further, these methods do not utilize the environmental prior to assisting in the generation of plausible agent motions. In parallel, several data-driven techniques [Bera et al. 2016; Guy et al. 2011; Kim et al. 2016; Li et al. 2015] have been proposed to generate more intelligent agents that exhibit different styles [Bera et al. 2016; Guy et al. 2011] and react to abnormal situations [Kim et al. 2016]. Unfortunately, these techniques are focused on low-level motions, while the high-level navigation trajectories still need to be provided as

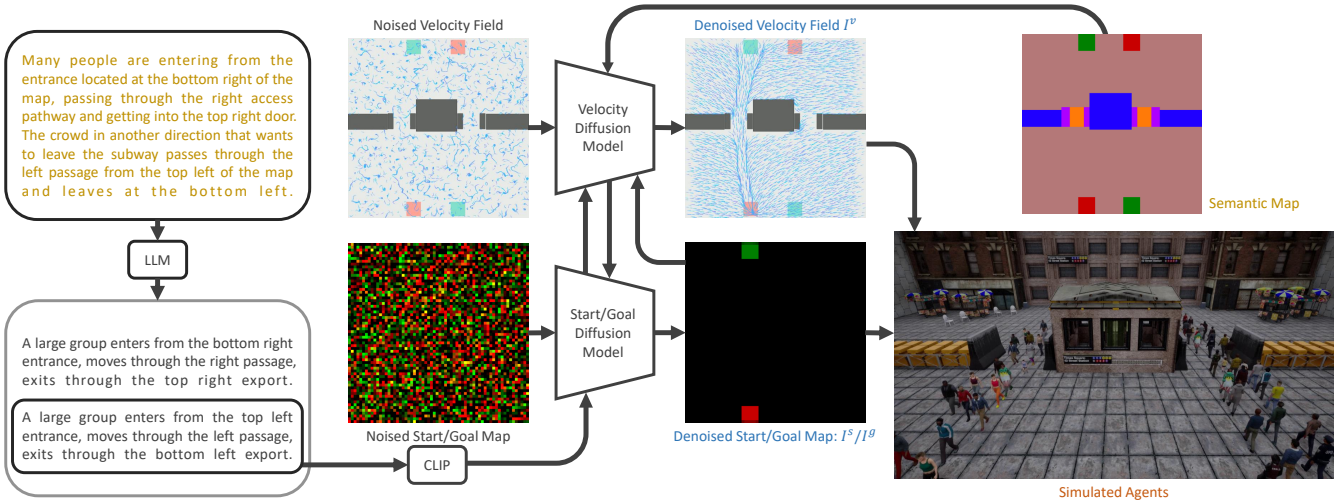


Figure 2: *The overall pipeline of our method. The inputs to our method (yellow) involve a general text describing agent motions and a semantic environment map I_e. We use an LLM to canonical the text, extracting a set of sentences $\{s^j\}$, one for each agent group. Each sentence is embedded by a CLIP model to guide our start/goal diffusion model for start/goal map generation (blue), given the semantic map. Next, guided by the semantic map, the text prompts, and the start/goal distribution map $I^j_{s,g}$, our velocity diffusion model infers the velocity field I^j_v to navigate the agent group (blue). Finally, we can sample agents according to the distribution map and then use the velocity map to guide our RVO agent simulator (red).*

inputs. In addition, although these techniques elevate the individual intelligence of agents, they do not incorporate the connection between the environment and the collective behaviors of agents. Finally, we are aware of a class of crowd-driven environment design algorithms [Aschwanden et al. 2011; Chakraborty et al. 2017; Feng et al. 2016] that bear a strong connection with our method. These methods aim at designing an environment that achieves certain agent-related goals, e.g., reachability and congestion-free. Instead, our method represents the inverse procedure of these techniques. Given a known environment, we predict compatible and plausible agent motions guided by a text description.

2.3 Crowd Simulation

Modern crowd animation systems are divided into two stages. The aforementioned crowd animation system provides the high-level motion trajectories for agent groups, while the low-level crowd simulator ensures the collision-free locomotions of each individual agent. Our method is focused on the high-level crowd animation, while we reuse existing velocity-field-based techniques [Patil et al. 2010; Thalmann et al. 2004; Treuille et al. 2006; Ye et al. 2023] to simulate each group of agents. This method is a well-known solution for controlling an arbitrarily large swarm by having each agent sample their velocity from the field. Several works such as [Treuille et al. 2006] have incorporated various modifications to the velocity field for generating more realistic and congestion-free agent motions.

However, the velocity field alone cannot ensure exact collision-free among agents. Local navigation algorithms [Karamouzas et al. 2017; Van den Berg et al. 2008] are developed to this end. These methods model the multi-agent system as a general dynamics system with collision-free constraints, and cast the crowd simulation as the numerical integration of the dynamics system. Methods [Fiorini and Shiller 1998; Van Den Berg et al. 2011; Van den Berg et al. 2008] based on the Velocity Obstacle (VO), for example, are widely used for local collision avoidance in multi-agent systems. VO [Fiorini and Shiller 1998] calculates the approximate set of velocities that could lead to future collisions. The agent can then navigate safely by choosing velocities outside these regions. Reciprocal Velocity Obstacles (RVO) [Van den Berg et al. 2008] extends VO by considering the velocities of two potentially colliding agents, enabling cooperative collision avoidance. Compared to VO, RVO promotes smoother motions in the multi-agent settings. In this work, we use a machine learning algorithm to predict the guiding velocity field from a given environment. Then, we use RVO as the local planner to generate collision-free navigation trajectories, given initial velocities sampled from the velocity field.

2.4 Language-Guided Generative Models

The most recent development of LLM and LDM unlocks the full ability of learning models in representing data with rich varieties. They quickly find applications in a wide spectrum of areas in computer graphics and computer vision, which includes the generation of 2D images [Rombach et al. 2022], 3D models [Poole et al. 2022], avatars [Wang et al. 2023], textures [Cao et al. 2023], gestures [Ao et al. [n. d.]], and character animations [Rempe et al. 2023]. Their interface with language encoders allows amateur users to create faithful content from intuitive guidance in the form of languages and internet images. Our work extends this ability to the area of crowd animation scenarios. As two closely related works, both Zhong et al. [2023] and Rempe et al. [2023] aim at generating multi-agent animations for vehicles and human characters, respectively, both of which are based on LDM. However, their model predicts a short future trajectory for each agent using a separate, costly model inference, rendering them less efficient for real-time applications.

Instead, our model generates a global guiding velocity field for the entire environment using a single inference.

3 METHOD

Our method takes as input a given environment represented as a semantic map, as well as a text description of the agent behaviors. We then detect a set of agent groups. For each agent group, we predict its start and goal distributions and generate a velocity field to navigate the group. The pipeline of our method is illustrated in Figure 2. In this section, we first introduce the problem setups in Section 3.1. We then describe our method to extract the number of agent groups and generate the canonical sentence for each group (Section 3.2). Next, we describe our model for predicting the start/goal distributions and guiding velocity fields in Section 3.3. Finally, we describe our constructive dataset and training method in Section 3.4.

3.1 Multi-Agent Navigation Scenario Synthesis

We assume that the artists have already designed and annotated the landscape of a virtual environment, which is a rather mild assumption in the game industry. Indeed, artistic assets have been annotated and reused considerably in the modern game production pipeline. Level designers also use these annotations as an essential part of the gameplay logic. Therefore, we assume that the annotated environment is represented using one-hot encoding and rasterized as a $C \times H \times W$ image I_e, where C is the number of different semantic types in the environment and $H \times W$ is the image resolution. Further, we require the user to provide a general text-based description T about how agents should move in the environment and we provide an example in Figure 2. Notably, we allow users to provide descriptions for an arbitrary number of agents, and we do not require users to be precise or follow any structures in crafting these descriptions. Given I_e and T, our goal is to generate a set $\mathcal{A}$ of agent and controller pairs, denoted as:

$$\mathcal{A} \triangleq \{\langle x^0, \pi(x) \mapsto \dot{x} \rangle\},$$

where the position of the agent at the tth timestep is denoted as x^t and the agent's moving direction is calculated as $\dot{x}^t \leftarrow \pi(x^t)$. We require that the generated collective agents' behaviors be compatible with the environment map I_e and conform to the description T. Further, our method is targeted at real-time applications. Therefore, the cost of inferring $\dot{x}^t$ for all agents must be reasonably low.

3.2 Generating Canonicalized Text Prompts

To achieve a low cost of inference, our main assumption is the group-based motion of agents, which has been explored in many prior works on agent simulation and planning [He et al. 2016; Kim 2004; Luo et al. 2021]. Indeed, human crowds tend to spontaneously form groups during navigation, which is beneficial for both congestion resolution and social-rule following. As a result, the user input is an arbitrary text T that contains descriptions of multiple groups.

However, such descriptions can contain several sources of ambiguities. On the high level, the order of descriptions for different groups can be arbitrary or mixed. The tense and order of each sentence can be different. On the low level, the description of entities in the environment might be inaccurate, e.g., referring to tall buildings as skyscrapers, or referring to crosswalks as zebra crossings. Although our method uses pre-trained image-text embedding model CLIP [Radford et al. 2021] to extract visual concepts from simple text prompts, their text encoder is not trained with such complex text scripts as our inputs. Therefore, we propose to remove ambiguity in T and extract canonicalized sentences describing agent motions, which can be well digested by CLIP.

Our idea is inspired by recent works on LLM-assisted robot motion and task planning [Lin et al. 2023; Singh et al. 2023]. Specifically, we provide GPT-4 [Achiam et al. 2023] with an instruction describing our requirement on canonical sentences: First, each sentence should describe the motion of one group of agents. Second, each sentence should use only present tense and take the structure as illustrated in Figure 3, where each comma-separated clause describes a concrete action performed by the group in the temporal order. We further require that the term in each <> bracket must be chosen from a preset dictionary. We find this method very effective in transcribing sentences to unified forms, which significantly improves the performance of our downstream methods for predicting agent distributions and velocity fields. We denote this procedure as a function mapping T into a set of sentences $\{s^j\}$, where we use superscript j to denote the group index. More details can be found in our supplementary documents.

3.3 Diffusion-based Agent Generation & Control

Given each sentence s^j, our next step is to generate a subset of agents $\mathcal{A}^j \subseteq \mathcal{A}$ exhibiting the navigation paths that conform to the sentence. To this end, we use two separate stages for predicting the agent distributions and control policies, respectively. We notice several prior works on traffic trajectory prediction [Alahi et al. 2016; Ma et al. 2019] and human detection [Nguyen et al. 2016] that partially address these problems. However, these methods are not suitable in our work because they solve the problem in an agent-centric manner, i.e., a separate inference or computational procedure is conducted for each agent, which cannot achieve real-time performance when scaling to tens or hundreds of agents. Instead, we propose an image-based representation. For the jth group, we introduce three images I_s^j, I_g^j, and I_v^j. The first two images I_s^j and I_g^j represent the spatial probability field of initial and final agent positions, respectively. Specifically, at an arbitrary location x, $I_s^j(x)$ is the probability that an agent exists at x. Similarly, $I_g^j(x)$ is the probability that the agent at x leaves the scenario and should be deleted. Our third image I_v^j represents the velocity field, i.e. $I_v^j(x)$ represents the velocity along which an agent at x should move.

Our representation has two major benefits. First and foremost, after predicting I_s^j, I_g^j, and I_v^j, our method allows the efficient generation and simulation of $\mathcal{A}^j$. Specifically, given I_s^j, we first sample an agent centered at each pixel x according to $I_s^j(x)$. We assume that all agents have a unit radius r and adopt the Poisson disk sampling [Bridson 2007] to remove all overlapping agents. For all the agents from various groups, we then simulate their collision-free navigation trajectories using the local navigation algorithm RVO [Van den Berg et al. 2008], with I_v^j being the guiding velocity. Finally, during each simulation frame, we remove agent $x^t \in \mathcal{A}^j$ ac-

cording to probability $I_g^j(x^t)$. This simulation procedure can achieve real-time performance and has been adopted in several existing game engines.

As a second benefit, our representation is entirely image-based, lending itself to efficient generative models such as [Goodfellow et al. 2020; Rombach et al. 2022]. Further, the cost of inference is invariant to the number of agents or the length of trajectories. To predict the three images, we adopt the LDM proposed in [Rombach et al. 2022], which is known to achieve superior performance in representing multi-model distributions as required in our application. Our data distribution is denoted as $d_0 \sim q(d_0)$. The LDM is based on the diffusion process that gradually injects noise into the data distribution over T timesteps, denoted as $d_{0:T} \sim q(d_{1:T}|d_0)q(d_0)$. A learnable model is then trained to reverse the diffusion process using a parameterization of form $p_\theta(d_{0:T}) = p(d_T)\prod_{t=1}^{T} p_\theta(d_{t-1}|d_t)$, with θ being learnable parameters. We follow Denoising Diffusion Probabilistic Models (DDPM) scheme [Ho et al. 2020] and re-parameterize the inverse procedure using a predicted noise component $\epsilon_\theta(d^t, t, h)$, which is to be subtracted from d^t to yield:

$$d_{t-1} \sim \mathcal{N}\left(\frac{1}{\sqrt{\alpha_t}}\left(d_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(d_t, t, h)\right), \sigma_t I\right),$$

where α_t is pre-defined noise magnitude, $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$, and σ_t is a pre-defined noise variance. The diffusion model can further be conditioned on a latent signal h, which encodes multi-model user guidance, such as text prompts, semantic maps, sketches, etc. In our work, we use the group-wise action sentence s^j as our guidance signal, which is encoded using CLIP.

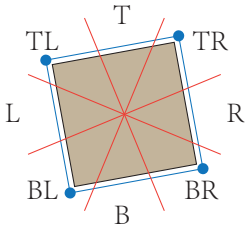
At the early stage of this research, we trained a single LDM to predict all three fields at once. However, the test-time performance of such a model is not satisfactory. In particular, we observe that the velocity fields oftentimes fail to form a path guiding agents from their start to goal positions. To mitigate this problem, we propose two diffusion models. Our first model jointly predicts the start and goal distributions $I_{s,g}^j$, while our second model is conditioned on the start and goal maps to then predict the velocity fields I_v^j. To sum up, our generative model is defined using the following two denoising processes for each group:

$$\begin{aligned} h^j &= \text{CLIP}(s^j) \\ I_{s,g,t-1}^j &= \frac{1}{\sqrt{\alpha_t}}\left(d_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(I_{s,g,t}^j, I_e, t, h^j)\right) \\ I_{v,t-1}^j &= \frac{1}{\sqrt{\alpha_t}}\left(d_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(I_{v,t}^j, I_e, I_s, I_g, t, h^j)\right). \end{aligned} \tag{1}$$

Note that the two diffusion models are trained separated without gradient back-propagation.

3.4 Dataset Preparation and Training

In order to train our generative model, we need a dataset with semantic maps I_e, the canonical group-wise sentence set $\{s^j\}$, as well as the groundtruth maps $I_{s,g,v}^j$ for each group. To the best of our knowledge, we are unaware of any datasets that provide such complete set of annotations. Therefore, we propose a constructive method to generate such a dataset. We assume the environment contains seven types of entities: obstacles in the shape of circle, cube, and triangle, start and goal areas, zebra crossings, and narrow passages. For each type of entity, we further define a set of anchor points as well as edges connecting the anchor points. These points and edges are summarized in Table 1.



To generate an annotated data term, we follow several steps. First, we randomly sample the environment map I_e. To this end, we randomly choose the environment to have $1-3$ start/goal area pairs. We then randomly choose positions on the boundary of I_e to place these start/goal regions. Next, we randomly choose $0-5$ entities in the environment, each entity belongs to one of the first 5 types in Table 1 uniformly at random, with random position, orientation, and scale. We further reject any environment maps with collisions between entities, entities outside the image, or the number of pixels that an entity occupies is beyond the range shown in Table 1. Note that we allow continuous orientation between $0^\circ - 360^\circ$ and we rotate the anchor points along with the entity. As a result, the text description for an anchor point might change when rotated. To determine the description, we evenly divide the space into 8 sectors, each spanning 45°. The anchor description is assigned based on the sector it belongs to, as shown in the inset. At this point, we can generate the groundtruth I_e using a $C = 9$-dimensional one-hot representation, where the first 8 dimensions correspond to each type of entity. Note that the narrow passage entity takes up two dimensions, corresponding to the two obstacles and the traversable region between them, respectively. The last 9th dimension corresponds to the background pixels.

Next, we sample the number of agent groups also in the range of $1-3$. Note that we assume distinct start regions for all agent groups, while their goal regions might be the same, so we reject agent groups with the same start regions, after which we have sampled the groundtruth $I_{s,g}^j$. For each agent group, we sample their navigation path by incrementally sampling anchor points until some goal region is sampled. If the current anchor point is on some entity, then we sample the next anchor on the same entity with a high probability of 0.7. We further reject an anchor point in one of the following cases: 1) The direct line segment connecting the next anchor point is in collision; 2) The path has loops or more than 9 anchor points; 3) the path contains an acute angle of degree less than 90°, indicating agents turning back. This meticulous approach ensures the generation of viable and realistic navigation paths for the agent groups in the simulation.

Finally, we generate the groundtruth canonical sentence s^j for each agent group. To this end, we follow the same canonical structure as in Figure 3 and we generate the shortest sentence that summarizes agent motions using the following rules: 1) the first < action > is always "enter area" and the last < action > is always "leave area"; 2) if an agent visits only one anchor point of an entity, we define the < action > as "pass by"; 3) if an agent visits two anchor points of an obstacle, we define the < action > as "pass by edge"; 4) if an agent visits more than two anchor points, we define the < action > as "circle around"; 5) if agents move through a narrow passage or zebra crossing, we define the < action > as "cross". Note in case 3-5), we use additional < action_adjective > to

Name	Circular Obs.	Cubical Obs.	Triangular Obs.	Zebra Crossing	Narrow Passage	Start	Goal
Figure							
Anchors	TL,TR,BL,BR	TL,TR,BL,BR	T,L,R	T,B,L,R	T,B	C	C
Edges	T,B,L,R	T,B,L,R	B,L,R	V,H	V	N/A	N/A
#Pixel	[70-100]	[70-100]	[70-100]	[170-200]	[170-200]	80	80

Table 1: *From top to bottom, we show name, figure, (comma-separated) anchor points in reference frame, edges, and number of pixels each entity can occupy in I_e after random scaling (the image I_e has a resolution of 1024×1024). In the figure, anchor points and edges are highlighted in blue, and non-traversable obstacles are highlighted with black borders. (T=Top, B=Bottom, L=Left, R=Right, C=Center, H=Horizontal, V=Vertical)*

define the direction of motion. Finally, we use the canonical term to refer to each anchor point as explained in Figure 3. To increase the variety of our dataset, we define a set of synonymous terms for each $< \bullet >$ term and randomly use one of the synonyms. We further randomly drop the optional () block.

Given the sampled path, our final step constructs the groundtruth velocity field I_v^j. Our goal is to create a stable, error-corrective velocity field, such that, when agents deviate from the path, they will be guided back to follow the path. Such velocity field has been studied in prior work [Rezende et al. 2021], where velocities at faraway positions are designed to direct towards the nearest point on the path. However, we find that such a velocity field will push agents to be concentrated on the path, forming a queue, which is not plausible. Instead, we propose a method to modify the velocity field to have agents scattered around the path. To this end, we notice that agents cannot concentrate on the path because they need to be collision-free. As a result, we propose a simulation-assisted velocity adjustment procedure based on RVO [Van den Berg et al. 2008]. Specifically, we initialize and simulate a reference group of 50 agents to follow the path, and we assume the RVO-adjusted, collision-free velocity to be the more plausible velocity, so we blend the adjusted velocity with the original velocity. This simulation and update procedure is performed repeatedly for 15 iterations. Our adjustment procedure is summarized in Algorithm 1, where we denote $\dot{\hat{x}}_t$ as the RVO adjusted velocity.

Algorithm 1 Velocity Field Adjustment

Input: Path following I_v^j and learning rate α
Output: Adjusted I_v^j
for iter=$1, \cdots, 15$ **do**
  Updated $\bar{I}_v^j \leftarrow I_v^j$
  Initialize 50 agents at start region: $\{x_0\}$
  for Each simulation timestep $t = 0, 1, \cdots$ **do**
    for Each agent x_t **do**
      $\dot{x}_t \leftarrow I_v^j(x_t)$
    $\{\langle x_{t+1}, \dot{\hat{x}}_t \rangle\} \leftarrow \text{RVO}(\{\langle x_t, \dot{x}_t \rangle\})$
    for Each agent x_{t+1} **do**
      Update $\bar{I}_v^j(x_t) \leftarrow \text{normalize}(\bar{I}_v^j(x_t) + \alpha \dot{\hat{x}}_t)$
  Apply updated $I_v^j \leftarrow \bar{I}_v^j$
Return I_v^j

Using the aforementioned procedure, we sample a dataset of 57600 tuples of $< I_e, \{s^j\}, \{I_{s,g,v}^j\} >$, in which we retain the same number of tuples with $0-5$ entities in the environment and $1-3$ agent groups for a balanced amount of training data, i.e. 3200 tuples in each case. We use the following standard DDPM loss [Ho et al. 2020] to train our two diffusion models: $\mathbb{E}_{d_0,\epsilon,t}\left[\|\epsilon - \epsilon_\theta\|^2\right]$.

4 EVALUATION

We parameterize the noise predictor ϵ_θ using a down-scaled UNet [Ronneberger et al. 2015]. Specifically for the start/goal diffusion model, the number of channels for the 4 downsampling layers is set to $(80, 160, 320, 320)$ and the number of normalization groups is set to 16. For the velocity diffusion model, the number of channels is set to $(160, 320, 640, 640)$ and the number of normalization groups is set to 32. The resolution of $I_{s,g}^j$ and I_v^j is set to 64×64. We use 75% of the dataset for training and the rest for testing. After experiments, we choose AdamW [Loshchilov and Hutter 2017] as our training algorithm and use the standard parameter settings there. We use 4 RTX 4090 GPU for the training over 50 epochs for the start/goal diffusion model and 200 epochs for the velocity diffusion model. Altogether, it takes 192 hours to train both diffusion models. Runtime agent simulation is performed using CARLA [Dosovitskiy et al. 2017]. Thanks to our efficient representation of agents, both our inference and online simulation are very efficient. Specifically, it takes less than 6 seconds to infer a set of three maps $I_{s,g,v}^j$, and the runtime simulation cost is less than 1ms per timestep, making our method well suited for real-time applications.

Qualitative Analysis. In Figure 1, 5, and 6, we provide details of four navigation scenarios with unseen environments and text descriptions. We refer readers to our supplementary materials for more results. The first two benchmarks Figure 5 and 6 highlight the ability of our method to handle unseen settings that are drastically different from our training dataset. In Figure 5 for example, the environment has 10 entities and user requires a huge agent group, while the passage is too narrow for all agents to pass at once, and some agents would be pushed away from the trajectory. Fortunately, our velocity field computed using [Rezende et al. 2021] is error-corrective, where agents would be guided back to follow the trajectory, however faraway they are. In Figure 6, we have 6 entities and 6 agent groups, each exhibiting complex trajectories and our method can readily generalize to such cases. Our last two examples in Figure 1 are mimicking real-world environments of a subway station and downtown crosswalk.

#Entity		0	1	2	3	4	5
#Agent Group	1	1.519/0.978/0.981	1.275/0.982/0.991	1.411/0.973/0.991	2.870/0.950/0.977	4.283/0.923/0.977	10.828/0.830/0.926
	2	2.116/0.971/0.974	1.276/0.979/0.990	1.868/0.968/0.987	3.603/0.940/0.979	5.555/0.905/0.968	9.401/0.853/0.951
	3	2.394/0.969/0.970	2.035/0.969/0.985	2.474/0.958/0.980	4.600/0.930/0.972	6.016/0.897/0.961	9.956/0.844/0.936

Table 2: *For each combination of agent group numbers and environmental entity numbers (excluding start and goal), we profile the ATD/SSR/RSR. ATD is profiled in pixels, where the image resolution is* 1024×1024.

Quantitative Analysis. We conduct a set of quantitative evaluations on the testing set using three metrics. First, our dataset contains a groundtruth trajectory for each agent group, so we can compare the actual trajectory of each agent with the groundtruth. To this end, we use the Dynamic Time Warping (DTW) algorithm [Senin 2008] to factor out their speed changes due to inter-agent collisions. DTW would compute a set of distances at different time instances for each agent. Further, we notice that our velocity adjustment Algorithm 1 allows agents to be reasonably scattered around the trajectory, so we set a margin of 80 pixels and we consider the agent to be on the trajectory if their distance is within the margin. As our first metric, we profile the average Agent-to-Trajectory Distance (ATD) over an agent group. Our second metric is the Strict Success Rate (SSR) profiled over all testing groups. To define SSR, we assume the agent successfully follows the trajectory, if an agent is within the margin of the trajectory for at least 70% time instances. If at least 80% of the agents in a group are successful, we call the group successful. Finally, we also profile a Relax Success Rate (RSR). RSR is similar to SSR, but we assume an agent is successful if he/she is within the margin for at least 50% time instances, and a group is successful if at least 70% of the agents are successful. In Table 2, we profile all three metrics for each combination of agent group numbers and environmental entity numbers. Evidently, the performance of diffusion model is satisfactory over all metrics. We observe an increase in ATD as more entities are involved, which is due to slight accuracy drop in predicting the velocity field. The ATD also increases slightly as the number of agent groups increases, which is due to collisions between different groups pushing agents away from trajectories. Note that although ATD increases, SSR and RSR do not drop much with number of agent groups or number of entities, again validating the effectiveness of our method.

Comparative User Study. To evaluate the effectiveness of our method, we conduct a comparative user study that leverages scenarios manually crafted in prior studies. Specifically, we choose the Shibuya crossing scenario simulated in GAScrowd [Kim et al. 2012] and use our pipeline to generate animations of a similar theme from designed text descriptions. Subsequently, we engage human participants in a comparative analysis of the two scenarios. Each participant is asked to assign a score for each scenario, scaling from 1-5, where a higher score indicates better realism. A total of 12 participants were recruited. The participants were asked to sign an informed consent form first and would get an equivalent coupon based on the measurement of 38.3 USD per hour after the experiment was completed. The statistical results show that 10 out of the 12 participants prefer the scenario generated by our method, and our method achieves an average score of 4.08 as compared with GAScrowd receiving a score of 2.91. The above results highlight the competitiveness of our method in crowd scenario generation.

Diversity of Agent Behaviors. Due to the random dropout of blocks in the input prompt during training and the stochastic nature of diffusion models, our method can produce varied simulation outcomes from identical text prompts across multiple executions while conforming to the specification of the high-level description. This controllable diversity can be reflected in the navigation trajectory (resulted from the stochastic diffusion model), crowd distribution, as well as crowd density (resulted from the random sampling in the source region). As an illustration, we use a vague description to run our pipeline twice. As shown in Figure 4 (a), the information of "which circular obstacle to interact with" and "which exit to leave through" are masked in our text description. Both of the two scenarios conform to the constraints specified in the text, while they demonstrate the diversity in the unconstrained parts, i.e. the group (left) moves past the lower circle and leaves through the upper exit while the other group (right) follows the opposite route.

Ablation Study. We conduct two ablation studies to explore the influence of the input instructions on our model performance. Our first study examines the impact of perturbing the structured instructions through: V1) expanding the dictionary with unseen words and phrases, without modifying the sentence structure; V2) employing LLM to reformulate the structured prompt into a synonymous, unstructured sentence. Over 60 testing cases, the success rate (strict/relax) for original, V1-perturbed, and V2-perturbed prompts are 0.933/0.966, 0.916/0.950, and 0.833/0.900, respectively. The marginal drop in performance demonstrates the stability of our approach to prompt noises. Our second study assesses the stability of LLM-based text canonicalization. We conduct ten rounds of text canonicalization using LLM on ten scenarios, recording the simulation results for each round. In this setting, we achieve an average (strict) success rate of 0.936 with a variance of 0.041. Such consistently high success rate demonstrates the LLM's reliable performance in text canonicalization.

To evaluate the effectiveness of our velocity field adjustment procedure described in Algorithm 1, we train two diffusion models using velocity fields before and after adjustment respectively and create two crowd animations accordingly. As shown in the comparative results in Figure 4 (b), the velocity field before adjustment (left) pushes agents to be concentrated, forming an unnatural queue, while the adjusted field (right) results in more plausibly scattered agent distributions.

5 CONCLUSION & DISCUSSION

We propose a learning-based, text-guided method for generating multi-agent navigation scenarios from high-level text descriptions.

This problem has potential usage in automatic content creation for the game industry. To the best of our knowledge, our work is the first method addressing this task. Our method has several shortcomings. First, the complexity of our synthetic dataset is not high enough for modeling realistic open-world environments. Future works could harness existing crowd prediction and environment datasets, but annotating these datasets could be rather challenging. Further, our method cannot model more complex agent actions, including inter-action communications and time-dependent motions. It would also be interesting to integrate language-conditioned behaviors for individual agents into the framework. Again, the main challenges in modeling these motions lie in the dataset construction and devising efficient, learnable representation. Finally, our method requires users to describe the behavior of each agent group, which requires a long text script in complex cases.

ACKNOWLEDGMENTS

This research project is partially supported by the Innovation and Technology Commission of the HKSAR Government under the InnoHK initiative, Hong Kong General Research Fund (11202119 and 11208718), Innovation and Technology Commission (GHP/126/21GD), and Guangdong, Hong Kong and Macao Joint Innovation Project (2023A0505010016).

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).

Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. 2016. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 961–971.

Tenglong Ao, Zeyi Zhang, and Libin Liu. [n. d.]. GestureDiffuCLIP: Gesture Diffusion Model with CLIP Latents. *ACM Trans. Graph.* ([n. d.]), 18 pages. https://doi.org/10.1145/3592097

Gideon DPA Aschwanden, Simon Haegler, Frédéric Bosché, Luc Van Gool, and Gerhard Schmitt. 2011. Empiric design evaluation in urban planning. *Automation in construction* 20, 3 (2011), 299–310.

Aniket Bera, Sujeong Kim, and Dinesh Manocha. 2016. Interactive and adaptive data-driven crowd simulation: User study. In *2016 IEEE Virtual Reality (VR)*. 325–325. https://doi.org/10.1109/VR.2016.7504784

Robert Bridson. 2007. Fast Poisson disk sampling in arbitrary dimensions. *SIGGRAPH sketches* 10, 1 (2007), 1.

Tianshi Cao, Karsten Kreis, Sanja Fidler, Nicholas Sharp, and Kangxue Yin. 2023. Texfusion: Synthesizing 3d textures with text-guided image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 4169–4181.

Nilay Chakraborty, Brandon Haworth, Muhammad Usman, Glen Berseth, Petros Faloutsos, and Mubbasir Kapadia. 2017. Crowd sourced co-design of floor plans using simulation guided games. In *Proceedings of the 10th International Conference on Motion in Games*. 1–5.

Emir Cogo, Ehlimana Krupalija, Irfan Prazina, Šeila Bećirović, Vensada Okanović, Selma Rizvić, and Razija Turčinhodžić Mulahasanović. [n. d.]. A Survey of Procedural Modelling Methods for Layout Generation of Virtual Scenes. *Computer Graphics Forum* n/a, n/a ([n. d.]), e14989. https://doi.org/10.1111/cgf.14989 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14989

Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. 2017. CARLA: An open urban driving simulator. In *Conference on robot learning*. PMLR, 1–16.

Kenny Erleben, Jon Sporring, Knud Henriksen, and Henrik Dohlmann. 2005. *Physics-based animation*. Vol. 79. Charles River Media Hingham.

Tian Feng, Lap-Fai Yu, Sai-Kit Yeung, KangKang Yin, and Kun Zhou. 2016. Crowd-driven mid-scale layout design. *ACM Trans. Graph.* 35, 4 (2016), 132–1.

Paolo Fiorini and Zvi Shiller. 1998. Motion planning in dynamic environments using velocity obstacles. *The international journal of robotics research* 17, 7 (1998), 760–772.

Matthew Fisher, Daniel Ritchie, Manolis Savva, Thomas Funkhouser, and Pat Hanrahan. 2012. Example-based synthesis of 3D object arrangements. *ACM Transactions on Graphics (TOG)* 31, 6 (2012), 1–11.

R. Ghzouli, T. Berger, E. Johnsen, A. Wasowski, and S. Dragule. 2023. Behavior Trees and State Machines in Robotics Applications. *IEEE Transactions on Software Engineering* 49, 09 (sep 2023), 4243–4267. https://doi.org/10.1109/TSE.2023.3269081

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2020. Generative adversarial networks. *Commun. ACM* 63, 11 (2020), 139–144.

Shihui Guo, Richard Southern, Jian Chang, David Greer, and Jian Jun Zhang. 2015. Adaptive motion synthesis for virtual characters: a survey. *The Visual Computer* 31 (2015), 497–512.

Stephen J Guy, Sujeong Kim, Ming C Lin, and Dinesh Manocha. 2011. Simulating heterogeneous crowd behaviors using personality trait theory. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics symposium on computer animation*. 43–52.

Liang He, Jia Pan, Wenping Wang, and Dinesh Manocha. 2016. Proxemic group behaviors using reciprocal multi-agent navigation. In *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 292–297.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems* 33 (2020), 6840–6851.

Ruizhen Hu, Zeyu Huang, Yuhan Tang, Oliver Van Kaick, Hao Zhang, and Hui Huang. 2020. Graph2plan: Learning floorplan generation from layout graphs. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 118–1.

Ioannis Karamouzas, Nick Sohre, Rahul Narain, and Stephen J Guy. 2017. Implicit crowds: Optimization integrator for robust crowd simulation. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–13.

Tom Kelly, Paul Guerrero, Anthony Steed, Peter Wonka, and Niloy J. Mitra. 2018. FrankenGAN: Guided Detail Synthesis for Building Mass Models Using Style-Synchonized GANs. *ACM Trans. Graph.* 37, 6, Article 216 (dec 2018), 14 pages. https://doi.org/10.1145/3272127.3275065

Dong-Hun Kim. 2004. Self-organization for multi-agent groups. *International Journal of Control, Automation, and Systems* 2, 3 (2004), 333–342.

Jongmin Kim, Yeongho Seol, Taesoo Kwon, and Jehee Lee. 2014. Interactive manipulation of large-scale crowd animation. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 1–10.

Sujeong Kim, Aniket Bera, Andrew Best, Rohan Chabra, and Dinesh Manocha. 2016. Interactive and adaptive data-driven crowd simulation. In *2016 IEEE Virtual Reality (VR)*. IEEE, 29–38.

Sujeong Kim, Stephen J Guy, Dinesh Manocha, and Ming C Lin. 2012. Interactive simulation of dynamic crowd behaviors using general adaptation syndrome theory. In *Proceedings of the ACM SIGGRAPH symposium on interactive 3D graphics and games*. 55–62.

Marilena Lemonari, Rafael Blanco, Panayiotis Charalambous, Nuria Pelechano, Marios Avraamides, Julien Pettré, and Yiorgos Chrysanthou. 2022. Authoring virtual crowds: A survey. In *Computer Graphics Forum*, Vol. 41. Wiley Online Library, 677–701.

Weizi Li, David Wolinski, Julien Pettré, and Ming C. Lin. 2015. Biologically-inspired visual simulation of insect swarms. In *Computer Graphics Forum*, Vol. 34. Wiley Online Library, 425–434.

Kevin Lin, Christopher Agia, Toki Migimatsu, Marco Pavone, and Jeannette Bohg. 2023. Text2Motion: from natural language instructions to feasible plans. *Autonomous Robots* (14 Nov 2023). https://doi.org/10.1007/s10514-023-10131-7

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101* (2017).

Linbo Luo, Xinyu Wang, Jianfeng Ma, and Yew-Soon Ong. 2021. Grpavoid: Multigroup collision-avoidance control and optimization for UAV swarm. *IEEE Transactions on Cybernetics* (2021).

Yuexin Ma, Xinge Zhu, Sibo Zhang, Ruigang Yang, Wenping Wang, and Dinesh Manocha. 2019. Trafficpredict: Trajectory prediction for heterogeneous traffic-agents. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 6120–6127.

Duc Thanh Nguyen, Wanqing Li, and Philip O Ogunbona. 2016. Human detection from images and videos: A survey. *Pattern Recognition* 51 (2016), 148–175.

Masaki Oshita and Yusuke Ogiwara. 2009. Sketch-based interface for crowd animation. In *Smart Graphics: 10th International Symposium, SG 2009, Salamanca, Spain, May 28-30, 2009. Proceedings 10*. Springer, 253–262.

Sachin Patil, Jur Van Den Berg, Sean Curtis, Ming C Lin, and Dinesh Manocha. 2010. Directing crowd simulations using navigation fields. *IEEE transactions on visualization and computer graphics* 17, 2 (2010), 244–254.

Xue Bin Peng, Ze Ma, Pieter Abbeel, Sergey Levine, and Angjoo Kanazawa. 2021. Amp: Adversarial motion priors for stylized physics-based character control. *ACM Transactions on Graphics (ToG)* 40, 4 (2021), 1–20.

Julien Pettre, Jean-Paul Laumond, and Daniel Thalmann. 2005. A navigation graph for real-time crowd animation on multilayered and uneven terrain. In *First International Workshop on Crowd Simulation*, Vol. 43. Pergamon Press New York, 194.

Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. 2022. DreamFusion: Text-to-3D using 2D Diffusion. *arXiv* (2022).

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*. PMLR, 8748–8763.

Davis Rempe, Zhengyi Luo, Xue Bin Peng, Ye Yuan, Kris Kitani, Karsten Kreis, Sanja Fidler, and Or Litany. 2023. Trace and Pace: Controllable Pedestrian Animation via Guided Trajectory Diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 13756–13766.

Adriano MC Rezende, Vinicius M Goncalves, and Luciano CA Pimenta. 2021. Constructive time-varying vector fields for robot navigation. *IEEE Transactions on Robotics* 38, 2 (2021), 852–867.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 10684–10695.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*. Springer, 234–241.

Pavel Senin. 2008. Dynamic time warping algorithm review. *Information and Computer Science Department University of Hawaii at Manoa Honolulu, USA* 855, 1-23 (2008), 40.

Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. 2023. Progprompt: Generating situated robot task plans using large language models. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 11523–11530.

Daniel Thalmann, Christophe Hery, Seth Lippman, Hiromi Ono, Stephen Regelous, and Douglas Sutton. 2004. Crowd and group animation. In *ACM SIGGRAPH 2004 course notes*. 34–es.

Adrien Treuille, Seth Cooper, and Zoran Popović. 2006. Continuum crowds. *ACM Transactions on Graphics (TOG)* 25, 3 (2006), 1160–1168.

Jur Van Den Berg, Stephen J Guy, Ming Lin, and Dinesh Manocha. 2011. Reciprocal n-body collision avoidance. In *Robotics Research: The 14th International Symposium ISRR*. Springer, 3–19.

Jur Van den Berg, Ming Lin, and Dinesh Manocha. 2008. Reciprocal velocity obstacles for real-time multi-agent navigation. In *2008 IEEE international conference on robotics and automation*. Ieee, 1928–1935.

Tengfei Wang, Bo Zhang, Ting Zhang, Shuyang Gu, Jianmin Bao, Tadas Baltrusaitis, Jingjing Shen, Dong Chen, Fang Wen, Qifeng Chen, et al. 2023. Rodin: A generative model for sculpting 3d digital avatars using diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 4563–4573.

Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. 2016. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. *Advances in neural information processing systems* 29 (2016).

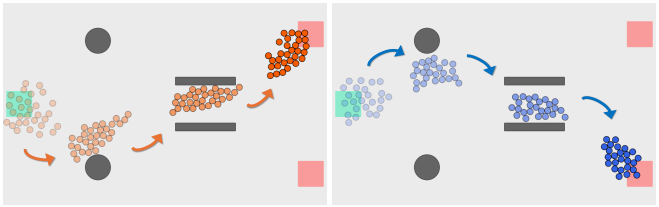
Xiaohan Ye, Zherong Pan, Xifeng Gao, Kui Wu, and Bo Ren. 2023. Differentiable Learning of Scalable Multi-Agent Navigation Policies. *IEEE Robotics and Automation Letters* 8, 4 (2023), 2229–2236.

Ziyuan Zhong, Davis Rempe, Danfei Xu, Yuxiao Chen, Sushant Veer, Tong Che, Baishakhi Ray, and Marco Pavone. 2023. Guided conditional diffusion for controllable traffic simulation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 3560–3566.

Wentao Zhu, Xiaoxuan Ma, Dongwoo Ro, Hai Ci, Jinlu Zhang, Jiaxin Shi, Feng Gao, Qi Tian, and Yizhou Wang. 2023. Human motion generation: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2023).

A <group_size> group
<action> the(<action_location> side of the)(<entity_location>) <entity_name>([action_adjective]),
<action> the(<action_location> side of the)(<entity_location>) <entity_name>([action_adjective]),
...

Figure 3: *The canonical structure of the sentence, where each comma-separated clause describes an action of the agent group, in temporal order. The terms in () bracket are optional and the terms in each <> or [] bracket must be chosen from a preset dictionary provided to GPT-4. Here the < action > indicates the type of agent motion, such as "enter area", "pass by", "circle around", etc. The < action_location > indicates the location of an anchor point relative to an environment entity and the < entity_location > describes the location of an entity in world space, if multiple entities of the same type exist. For example, we can refer to "the top left corner of the upper circular obstacle". Finally, the [action_adjective] describes the direction along which an agent group interacts with the obstacle, e.g., we can say that "an agent group circles around a cubic obstacle anticlockwise".*



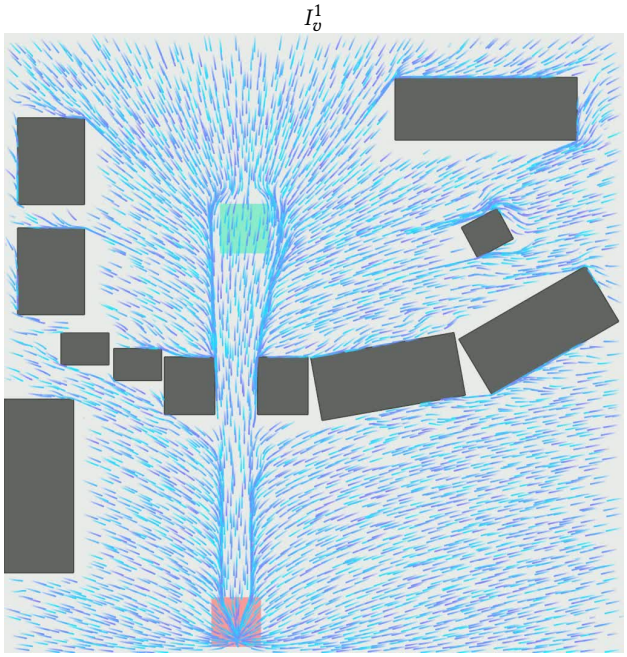
(a) *Two different crowd simulations generated with the same prompt "A big group enters from the left entrance, moves past the circle, walks through the passage, exits through the exit.". The drastically different crowd distributions and behaviors demonstrate the controllable diversity of our model.*

(b) *Compared with the model trained without field adjustment (left), our model (right) generates more scattered and realistic agent motions.*

Figure 4: *Additional results showing the diversity (a) and plausibility (b) of our model.*

Input:
There are hundreds of humans escaping through the passage from the open space above the map to the exit below.

GPT-4 Output:
s^1: A large group enters from the upper entrance, moves through the passage, exits through the bottom exit.



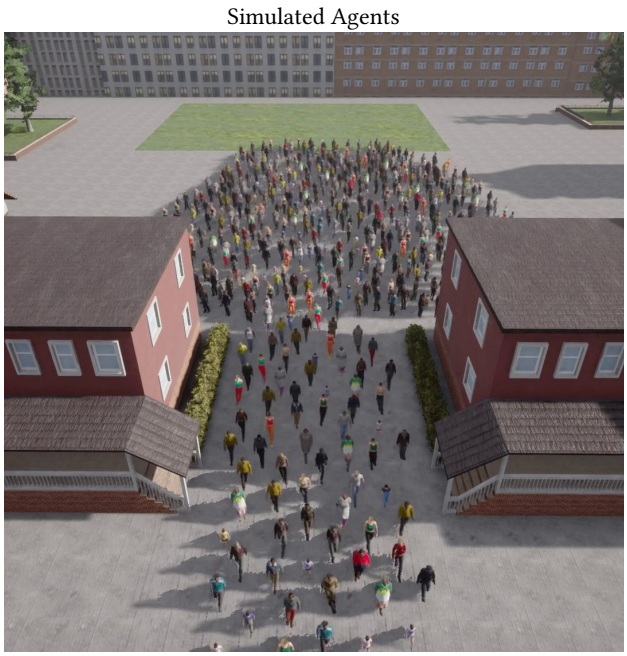


Figure 5: *A scenario with a large swarm of agents. We show the canonicalized $\{s^j\}$ and the predicted $\{I_v^j\}$. The start/goal regions are marked in green and red, respectively.*

Input:
This garden contains a total of 6 small groups of visitors. The first two groups enter from the main entrance in the top right corner and both of them walk through the upper right passage first. Afterward, one group passes the left side of the circular fountain, then gets through the garden passage and leaves at the bottom left exit. Another group follows a different path where they circle around the upper left rectangular groves counterclockwise and visit the fountain, finally exit through the bottom gate. Another two groups get in from the right gate. One group just visits the fountain and leaves through the bottom left export. Another group firstly moves towards the triangular entertainment area and circles around the lower side of it clockwise, then they pass by the fountain and circle around the upper left groves counterclockwise, finally leave through the left gate. The final two groups enter from the top entrance. One of them passes by the right of the upper left groves, circles around the fountain, and leaves the map through the left gate. Another small group visits the lower side of the fountain first. Then, they get around the triangular entertainment area and exit through the bottom export.

GPT-4 Output:
s^1: A small group enters from the top right entrance, walks through the upper right passage, passes by the left side of the circle, moves through the passage, exits through the bottom left exit.
s^2: A small group enters from the top right entrance, walks through the upper right passage, circles around the upper left rectangle counterclockwise, passes by the circle, exits through the bottom exit.
s^3: A small group gets in from the right entrance, passes by the circle, exits through the bottom left exit.
s^4: A small group gets in from the right entrance, circles around the lower side of the triangle clockwise, passes by the circle, circles around the upper left rectangle counterclockwise, exits through the left exit.
s^5: A small group enters from the top entrance, passes by the right side of the upper left rectangle, circles around the circle, exits through the left exit.
s^6: A small group enters from the top entrance, passes by the lower side of the circle, circles around the lower side of the triangle, exits through the bottom exit.

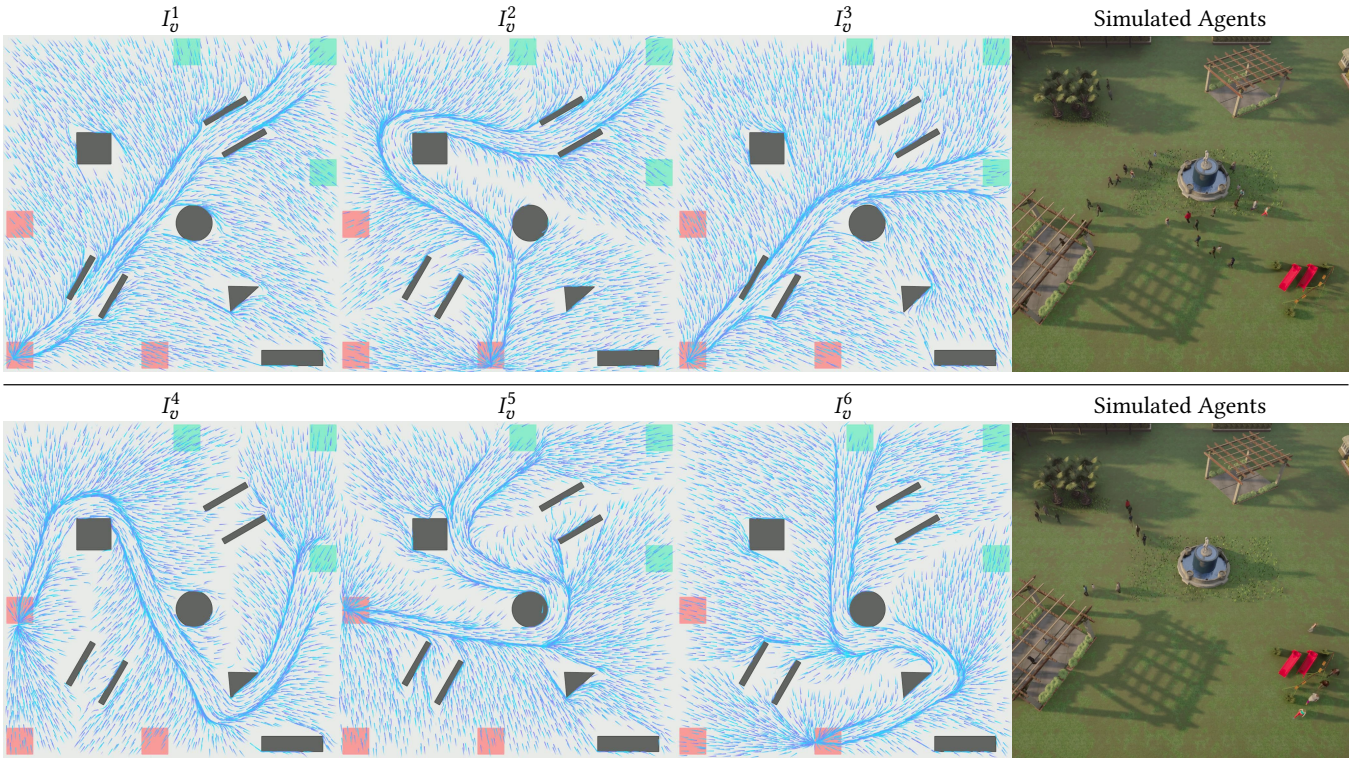


Figure 6: *A complex scenario with 6 agent groups. We show the input T, the canonicalized $\{s^j\}$, and the predicted $\{I_v^j\}$. The start/goal regions are marked in green and red, respectively.*