# AUTOMATED MODEL DISCOVERY FOR STEERING BEHAVIOR SIMULATION

Hai Le

Department of Computer Science
Georgia State University
25 Park Pl NE
Atlanta, GA, 30303 USA

Xiaolin Hu

Department of Computer Science
Georgia State University
25 Park Pl NE
Atlanta, GA, 30303 USA

## ABSTRACT

Steering behavior is widely used to simulate mobile agents' movement in virtual environments. This work presents a framework to support automated model discovery for steering behavior in simulation. A new category of entity called space entity is developed to allow agents to act on the open space surrounding them. An adaptive space entity generation method is developed to generate space entities based on an agent's nearby open and occupied space segments. The space entity and the developed method are incorporated into a model search framework to support automated discovery of candidate models. Experiment results show that the framework can support discovering multiple steering behavior models for a given scenario. The discovered models are robust, scalable, and can be applied to environments that are different from the ones where the models are discovered.

**Keywords:** agent-based system, steering behaviors, automated model discovery, space entity, adaptive space entity generation method.

## 1 INTRODUCTION

Steering behaviors are behaviors that make autonomous agents move around their world in a realistic and improvisational manner. Steering behaviors have been widely used to simulate mobile agents' movement in a virtual environment (Yang et al. 2020). They support mobile agents' movements from simple behaviors such as seek, flee, and path following (Reynolds 2002) to more complex ones such as combining separation, cohesion, and alignment in the Boids model (Reynolds 1987). Complex scenarios usually need multiple steering behaviors to satisfy a set of desired requirements. Hence, robust steering behaviors are difficult to craft and typically are designed using the traditional approach where models are manually crafted and gradually improved by domain experts. This approach is time consuming and is subjected to potential biases from the modelers.

It is desirable to support automated model discovery for steering behavior simulation. The approach of automated model discovery would bring two advantages. First, since the models are discovered by a search algorithm, this approach can minimize the bias that often exists when developing hand-crafted models. Second, multiple models might be discovered during the automated search process, which is rare in the traditional modeling approach because modelers typically stop after achieving the requirements with one model. In previous work, we have developed a framework that supports an automated search of candidate models for behavior-based mobile agents. The framework includes a 2D space, a set of physical entities including agents and obstacles. Agents have a set of properties such as: position, heading direction, speed, and their values are manipulated by a set of behaviors. The search space is defined by all possible behaviors of agents based on their properties and how they interact with other entities by sensing, filtering, and acting. Once the requirements and evaluations are set, a search algorithm is used to search for the best models.

More details about the framework as well as some discovered models can be found in (Keller and Hu 2019; Le and Hu 2020).
Our previous work has mainly focused on agents' interactions with other physical entities (including other agents), hence an agent can only act on properties (e.g., position, direction, speed) of some physical entities. This limitation of considering only the physical entities, hence makes it difficult for agents to move in directions that are not directly associated with any physical entity. In particular, it is difficult for agents to navigate through open spaces among physical entities, which is crucial for steering behavior simulation. It has been observed that when steering in complex environments, autonomous agents use information not only from nearby physical entities, but also from open space directions surrounding them. Even though an open space appears to have nothing, it actually contains useful information such as how far agents can move along the open space direction, and how much the angle difference is between the open space direction and the agents' current moving directions. Having the capability of processing these types of information would give agents more options for choosing their movement in steering behavior simulation. This would allow agents to move more effectively and naturally in experiment scenarios where utilizing the open space around agents are crucial such as multiple obstacles avoidance, or leader-following.

Motivated by the above discussion, this paper develops a new behavior specification for steering behavior that gives agents the option to interact with the open space surrounding them. Hence, a new category of entity called "space entity" is introduced to add to the physical entities that have been supported in the previous framework. Each agent has a set of space entities corresponding to a set of landmarks that an agent can head to from its current position. At each timestep, an agent's space entities are dynamically generated based on an adaptive space entity generation method, which checks the agent's nearby physical entities and generates a set of possible steering directions adaptively. To work with the existing automated model discovery framework, each space entity has a set of pre-defined properties. For example, the travel distances from the agent to the space entities, or the angle differences between the directions to space entities and the agent' current direction. Agents use the values of these properties to filter which space entity they want to change direction toward. This extension gives agents more flexibility when using steering behaviors. For example, in previous work, when an agent wants to avoid a physical entity, it first needs to get a reference direction from the physical entity. Then a static angle offset is added so that the agent can move away from the entity. This approach is cumbersome and not adaptable (for example, the static angle offset makes the agent always turns left or right with a fixed angle away from the entity). The space entities provide multiple reference directions for agents to evaluate, hence make the steering behaviors flexible and adaptive to many situations.

To evaluate the robustness of this proposed method, we first apply the approach to discover models for two basic steering behaviors including leader-following, and obstacle avoidance. We then apply the approach to discover models for combined steering behaviors in two more complex scenarios: 1) hallway evacuation with obstacle in the middle, and 2) leader-following while avoiding multiple obstacles. The result shows that by providing the space entities in the framework, multiple high quality models can be discovered in an automated way. The discovered models are robust and scalable as they can be applied to environments that are different from the ones where the models are discovered.

## 2 RELATED WORK

Steering behaviors in agent-based systems have been studied extensively in the literature and can be classified into two main approaches: macroscopic and microscopic (Manuele et al. 2013). Our work uses a decentralized framework where agents only receive information by observing nearby objects to study and discover microscopic steering behaviors. Similarly, the famous Boids model is one example where agents rely on their neighbors to perform three steering behaviors: separation, cohesion, and alignment (Reynolds 1987). In Braver and Malikopoulos (2020), and Hasham and Wainer (2021), the authors gather extra information from neighbors (energy and personal space map) to make the models work in complex scenarios. In Song et al. (2021), and Auletta et al. (2022), path planning algorithms are proposed to resolve

herding by caging problems. The main limitation of these works is they focus on surrounding visible objects and ignore the potential of space information. To address this issue, there are several works that concentrate on evaluating the space around agents. The research in Yıldızab and Çağdaş (2020) scans the map layout of the experiments so that the empty and occupied space is known. Agents then use this information as global knowledge when making decisions. In a local scale where agents evaluate space within their field of view (FOV), the work of Boatright et al. (2014) divides FOV into several segments. Each of them is evaluated based on neighbor density, obstacle occupied, and agent's flow within the segment to decide where to steer next. In Rodrigues et al. (2009), a series of nodes are placed in the empty space of the experiment field and presented as landmarks. Each node within an agent's FOV is a steering option that agents can head toward. The work of Kapadia et al. (2009) defines egocentric perception fields where a set of circle nodes is generated around agents. Combined with space time planning, each node is evaluated, and the best path is drawn for the agent to steer. In these works, because the potential landmarks for space are pre-set, in some situations, the best option might not be included in the set. Our work resolves this limitation by providing an adaptive set of space entities depending on surrounding neighbors' positions.

Regardless of whether the end results are based on physical neighbors or the surrounding space, steering behaviors in one model are independent; hence, this might create conflicts between them such as canceling the steering force of each other. To alleviate this limitation, the first approach evaluates all steering forces base on a parameter set and chooses the best one. In Hu, Lees, and Zhou (2013), agents evaluate each decision by observing surrounding neighbors and events that might happen in the future. In Silveira et al. (2010), a homogeneous meshes global environment map is designed including obstacle locations and a goal. A path planner designed algorithm is used to constantly update agents' local maps based on nearby objects and information from global maps to decide which direction to the steer. In Dutra et al. (2017), for each steering option, the authors use gradient color to generate an image that illustrates the risk of collisions between agents and their neighbors. These images are combined, and the steering direction is the path with the lowest risk based on the color. The limitation of this method is the computational costs are usually quite high since all steering forces need to be evaluated carefully. As a result, our work used the second approach where all steering options are contributed to the final decision. In Galvane et al. (2013), to achieve the best angles, the model uses scouting and tracking behaviors to steer the camera. The end results are simply getting the average of all steering forces. This method works when the simulation goal is simple. For more complex scenarios such as evacuation simulation in Trivedi and Rao (2018), the authors consider physical and psychological factors when evaluating several steering behaviors. Similar to Wang et al. (2018), we analyze the conditions of agents themselves or their neighbors to assign a weight for each behavior. The higher the weight, the more dominant the behavior is in the final result. Furthermore, the space behavior specification in our work is designed so that one behavior can fulfill more than one simulation requirement. As a result, it reduces the number of behaviors needed in complex scenarios, and hence also reduces the conflicts between them.

Even though these solutions can resolve the specific problems in the above works, proposed models are manually crafted by modelers. Instead of relying on domain knowledge, several data-driven approaches are used to identify of simulation model automatically. The works of Aness and Kumar (2022) and Solma, Moore, and Shah (2012) extract ground truth data from video clips of human movements. The researchers then analyze the data to identify crowd behaviors (bottlenecks, lanes, blocking, etc.). Similarly, the studies from Venkatramanan et al. (2018), and Amasyali and Gohary (2018) use historic data to predict the outcome. These researches focus on classification and prediction, not the model specifications. Besides collecting data, the works of Tan and Bora (2019), and Beck et al (2015) concentrate on adjusting parameters to make well-calibrated models that work best for the studied settings. Unlike these works, we focus on extending and improving behavior search space to make the framework better by discovering multiple models in challenging scenarios where several simulation requirements need to be accomplished.

# 3 SPACE ENTITY-ENHANCED MODEL DISCOVERY FRAMEWORK

## 3.1 Overview of the Automated Model Discovery Framework

To support an automated model discovery approach for an agent-based system, a formal model specification is provided in our previous framework (Keller and Hu 2019; Le and Hu 2020). Within that specification, a model contains a set of entities (agents, obstacles, and zones), and each entity category has a set of pre-defined properties **P** such as: position, heading direction, or speed. Each agent **a = < P, B>** contains a set of behavior **b = < p, Activation, Action>** in **B**. Each behavior **b** manipulates one property **p** of the agent, and has two main components: Activation and Action. The **Activation** component = **< Criteria, Activation function>** is used to assign a weight for the behavior depends on agents' self or neighbors' criteria and activation function (binary or linear). It is useful when agents need different priorities for their behaviors under different circumstances. The **Action** component = **<F, offset>** has a set of filters **F** to filter entities and extract property's values from them. After the offset is added, the final value replaces current property value **p** of the agent. To support automated discovery of models, a search space is created by combining all possibilities between model components' specifications. Because these components can be modified and exchanged between models, it is suitable to use Genetic Algorithm (GA) (Whitley 1994) as a search method where each possibility is considered as chromosomes for models. After a set of random models is generated as an initial population, GA uses selection, crossover, and mutation to gradually improve it. A set of fitness functions is defined to evaluate how close a model is compared to desired behavior patterns specified by a modeler. After the evaluation, if one of the models passes the threshold, GA finds the desired model and stops. Otherwise, 75% population of the next generation is combined between the best models of selection, crossover, mutation. The remained 25% are randomly generated to increase the diversity of the populations (Keller and Hu 2019). The circle continues until GA finds the best model(s) or the computational limitation is reached.

## 3.2 Space Entity and Adaptive Space Entity Generation Method

This work provides a new component to give agents options to interact with the space surrounding them dynamically. Because the framework has been developed to work with an agent-based system, it is reasonable to make this space component become entities, so that agents can select one as a landmark, and head toward it. Unlike the real physical entities such as agents, or obstacles where their positions are explicitly shown, space entities are not and must be generated depending on the surrounding environment. In theory, the best space entity set evaluates all possible heading directions within an agent's FOV. However, this approach is not practical because of the high computational cost. Therefore, in this work, the adaptive space entity generation method is implemented to provide a set of selected space entities where the headings toward them are worth evaluating.

To fully support an automated discovery model approach, a formal specification of space entity set called **S** is included in entity specification from previous work beside agents, obstacles, and zone entities. **S** is defined as a set of space entities where**: s = <position, P>** $\in$ **S** where **position** is value x and y of s in 2d space**, and P** is a set of pre-defined properties of a space entity.

Figure 1 shows an example of how an agent uses the adaptive space entity generation method to generate a set of space entities **S**. Figure 1a illustrates a current processing red agent A with a 360-degree FOV. Agent **A** has some nearby entities including four agents ($\mathbf{A_1}$, $\mathbf{A_2}$, $\mathbf{A_3}$, $\mathbf{A_4}$), one black circle obstacle ($\mathbf{O_1}$), and two rectangle walls ($\mathbf{W_1}$, $\mathbf{W_2}$). To generate set **S**, at each timestep, agent A first groups entities of the same categories (agents, obstacles, or walls) into clusters. Next, a pair of tangent lines called key heading options are created based on the most outward entities' positions of each cluster. Figure 1b shows how agent A generates blue, red, and green key heading options for agent, obstacle, and wall categories respectively. These key headings are important because they present the boundary between occupied segments (colored by blue, red, and green) and empty space segments (white) within an agents' FOV. Next, all key heading options are combined to create an initial heading set for agent A as figure 1c shows. In many cases, the space between two adjacent key headings is so large that it may leave many good potential headings between the boundaries, such as the gap between $h_2$ and $h_3$ in figure 1c. As a result, staring from FOV's

right bound of the agent, more trajectory options are generated incrementally by a pre-set interval, and follow a counterclockwise direction until it reaches FOV's left bound. While increasing, if an option hits the same value as the key heading value before the next interval, the counting starts again at the matched key value (illustrated by $hs_1$ – $hs_7$ in figure 1d). These heading options are dynamically changed at each time step depending on the positions and heading directions of agents, as well as the positions of nearby neighbors, hence agents have a better spatial set to interact with compared to the works of Kapadia et al. (2009) and Rodrigues et al. (2009). After all key heading options are generated, their intersections with the processing agent's FOV become the positions of space entities **s** (Figure 1e).

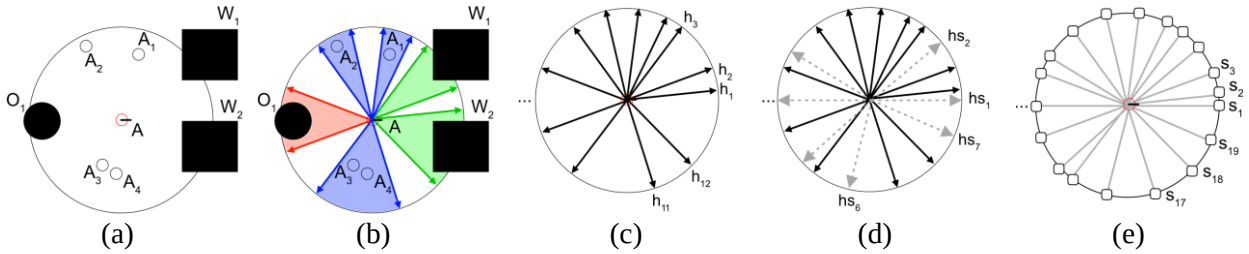



Figure 1: Generating key heading options and space entities.

As mentioned above, besides position property, each space entity **s** also has a set of pre-defined properties from the modelers. Because we focus on obstacle avoidance and leader-following steering behaviors, there are two important properties for each space entity: travel distance and angle distance.

The travel distance property measures various distances on the paths between agents and space entities. In many situations, one heading direction path can be intersected by more than one entity category. Therefore, nearest travel distance values to each and all categories are generated for space entities. Figure 2a illustrates that a path is drawn from agent A to $s_1$ and intersected with obstacle $\mathbf{O_1}$, hence the travel distance to the nearest obstacle is $\mathbf{d_o}$. Since there is no agent or wall along the path, the travel distance to the nearest agent and the nearest wall properties are equal to agent A's FOV distance. The travel distance to all entities is assigned by the distance to the nearest entity (in this example it is $O_1$ with distance = $d_o$). Similarly, figure 2b shows a case where the path from A to $s_2$ is intersected with all entity categories. Hence, $d_a$, $d_o$, $d_w$, and $d_o$ are assigned to the travel distances to the nearest agent, obstacle, wall, and all entities respectively. Figure 2c shows one simple case where the path from A to $s_3$ is not blocked by any entity. Therefore, all travel distances have values equal to agent A's FOV distance. These property values give agents the flexibility to focus on the category they are interested in. For example, if an agent wants to avoid a nearby obstacle, it only needs to check which path gives the shortest travel distance to the obstacle and avoids using it.

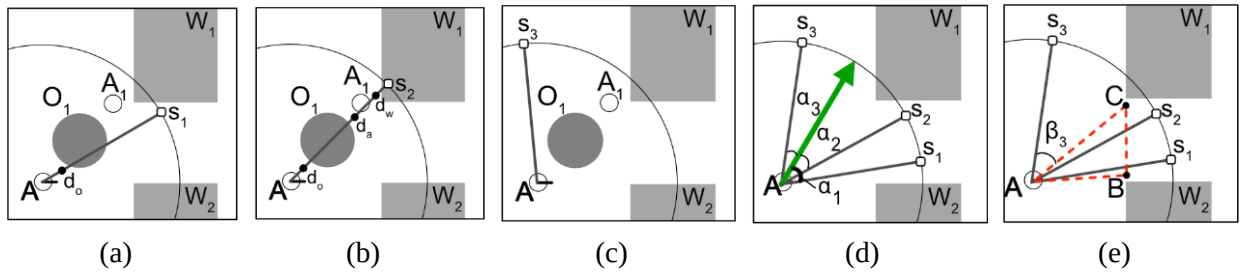



Figure 2: Space entity properties – The travel distance (a)(b)(c) and the angle distance (d)(e).

Two angle distances are used to measure the relative angle differences between a heading option and agents' heading values. First, in the situation where an agent needs to choose between two or more trajectory options that have the same travel distance, the angle distances between the headings and the agent's current heading direction are used to analyze and compare each heading option further. Figure 2d illustrates $\alpha_1$, $\alpha_2$,

$\alpha_3$ are the angle distances from $s_1$, $s_2$, $s_3$ to the current heading direction (green arrow) of agent A respectively. Second, in leader-following (section 4.1.1 & 4.2.2 ) and evacuation application (section 4.2.1), we assume that agents have knowledge of directions heading toward a pre-set area, hence the angle distance from the desired area to a heading option is provided to guild agents to the correct area. Depending on the layout of the testing experiment, this desired landmark is pre-defined by modelers. For example, figure 2e shows points B and C are the bounds of an escape hallway. If an agent is not in the hallway, its position, point B and point C form an area where all heading options inside are considered within desired range direction (in leader-following scenarios (section 4.1.1 & 4.2.2), B and C are the same). In this example, heading options to $s_1$ and $s_2$ have angle distances to the desired area equal to 0. For the options that are outside of the range such as the trajectory between agent A and s3 in figure 2e, $\beta_3$ is the angle distance between it and the nearest desired area bound (AC). If an agent wants to escape, it will select the space entity where the angle distance to the desired area is smallest (either $s_1$ or $s_2$).

These distance properties, when combined with filter components from previous work, create a new set of behaviors for agents to interact with the space surrounding them. Not only are agents able to navigate through physical entities more efficiently, but they also have more heading options on both the left and right sides to evaluate. Most importantly, this extension increases the search space with better options and the behavior sets are diverse enough for GA to discover multiple good strategies where agents' movements are more natural and effective.

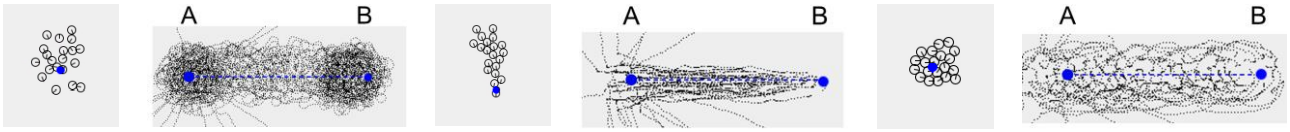
## 4 AUTOMATED MODEL DISCOVERY FOR STEERING BEHAVIOR SIMULATION

As mentioned above, we first use two classic scenarios with a single objective to demonstrate and evaluate the effectiveness of this work including leader-following (4.1.1) and obstacle avoidance (4.1.2). Then we further test the approach with two more complex setups that have multiple requirements for steering behaviors: hallway evacuation with obstacle in the middle (4.2.1), and leader-following while avoiding multiple obstacles (4.2.2). Because each agent presents a robot with a FOV angle = 360 and a FOV distance = 60, a physical constraint is pre-set to prevent agents from overlapping each other. GA uses several fitness functions to evaluate these scenarios. First, the "collision fitness function" is used to count how many times agents collide with other agents or obstacles (used in all scenarios). It is needed for slowing down and speeding up behaviors to be discovered. Second, the "follow leader fitness function" measures the distances between agents and a leader (used in 4.1.2 & 4.2.2). Third, because scenario 4.2.1 is an evacuation application, it needs two fitness functions: the "time fitness function" measures how fast all agents can escape, and the "remained agent fitness function" counts the number of remaining agents after the simulation time is up. It is used to compare between models that cannot evacuate all agents within a limited time. For quality comparisons, model fitness scores of the below tables are averaged between 100 simulations. We also provide more challenging environment setups to test the robustness of two chosen discovered models. The results show that multiple good quality models are discovered for each tested scenario using the automated approach. (All demonstration clips can be viewed using the link: https://sims.cs.gsu.edu/sims/research/Steering_behavior_Videos.html).

### 4.1 Basic Steering Behavior

#### 4.1.1 Leader Following

- Experiment set up: the leader represents a light from a high altitude, hence all agents know the desired direction toward the leader regardless of the distance between them. The leader does not have physical constraints with other entities, and it changes direction/speed randomly after a period.
- Model 1: Agents constantly move around the leader. Table 2a and Figure 3 (a & b).
- Model 2: Agents stop one after another when having reached the leader. Table 2b and Figure 3 (c & d).
- Model 3: Agents warp around the leader. Table 2c and Figure 3 (e & f).




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

Figure 3: snapshots and travel paths of three discovered models for leader-following scenario.

Figure 3b, 3d, and 3f illustrate agents' travel paths between three models on a small scale test where the leader stays still at A, then moves and stops at B. To follow the leader, although all models make agents head to the space entity that has the smallest angle distance to the leader, they have different filter specifications. Table 1a shows model 1 specification in detail and the purpose of each behavior. Table 1a: ID-1 has two filters. Because the first filter removes all paths that have short travel distances to space entities, agents constantly steer away from nearby neighbors (Figure 3b). The second filter selects the space entity that has the nearest angle distance to the leader, hence makes agents move toward it. Because of the first filter, agents also turn away from the leader when nearby (figure 3a), thus it makes the average distance to the leader of all agents high (46.6). The model's speed behavior (Table 1a: ID-2) makes agents always move faster than the leader, hence helping agents reduce their distances to the leader more efficiently. To reduce the repetitions, for the rest of this section, model specifications are described in a shorter version. The weights of activation components, if not specified, are equal to 1. Agents in model 2 simply steer to the leader without any filter (Table 1b: ID-1), hence the congestion happens easily (figure 3c & 3d) because agents can head toward a very nearby neighbor. This is reflected in high average collision violation score compared to other models. Besides two similar filters to table 1a: ID-1 behavior, model 3 further filters out any path that has a long angle distance to desired area, and forces agents to pick the best option in front of them. (Table 1c: ID-1). The result is a group of agents that will try to squeeze to any empty space around the leader (figure 3e) while reducing their movements (Figure 3f).

Table 1: Behavior description of models for leader-following scenario.

| ID | Behavior Specifications | Interpreted Purpose |
|---|---|---|
| | Angel Behavior | |
| 1 | Activation:<br>▪ Always activate with weight = 1.0<br>Action:<br>▪ 1st filter: choose a set of space entities with the travel distances to all entities within [10-60].<br>▪ 2nd filter: choose the space entity with smallest angle distance to desired area.<br>▪ Property to extract: position of the chosen space entity. | Select only space entities with travel distance to all entities > 10, then steer to the desired area. |
| | Speed Behavior | |
| 2 | Activation:<br>▪ Always activate with weight = 1.0<br>Action.<br>▪ 1st filter: choose leader entity category.<br>▪ Property to extract: speed of the chosen leader.<br>▪ Offset: +2.0 | Match speed with the leader, then increase self-speed. |
| Average collision violation = 2.7<br>Average distance to leader = 46.6 | | |

(a)

| ID | **Behavior description** |
|---|---|
| **Behavior Group: angle** | |
| 1 | Steer to the desired area. |
| **Behavior Group: speed** | |
| 2 | If current direction has travel distance > 30, increase self-speed. |
| Average collision violation = 13.1<br>Average distance to leader = 74.8 | |

(b)

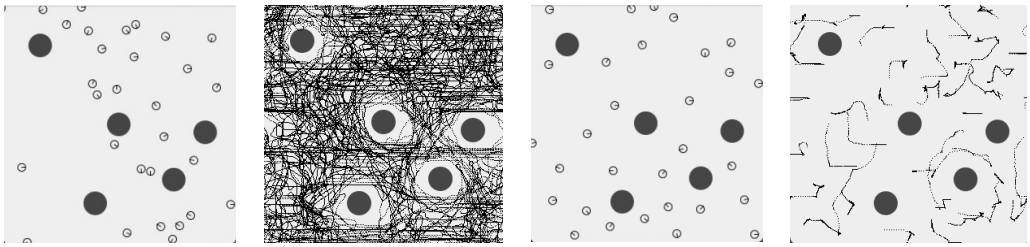
| ID | **Behavior description** |
|---|---|
| **Behavior Group: angle** | |
| 1 | Select only space entity with travel distance > 10 & the angle distance to desired range < 90, then steer to the desired area. |
| **Behavior Group: speed** | |
| 2 | Match speed with the leader, then increase self-speed. |
| Average collision violation = 9.3<br>Average distance to leader = 36.5 | |

(c)

### 4.1.2 Obstacle Avoidance

- Experiment set up: 5 obstacles are placed randomly in a warped world.
- Model 1: Agents constantly move to avoid nearby neighbors. Table 2a and Figure 4 (a & b)
- Model 2: Agents stop if there is no neighbor within their FOV. Table 2b and Figure 4 (c & d)

To avoid nearby objects, both models simply make agents steer to the space entity that has the farthest travel distance. The key difference between these models is that model 1 adjusts the agent's speed based on the agents' self-speed (table 2a: ID-2 & ID-3), hence, agents constantly speed up and slow down, but never stop (agents' travel paths in Figure 4b). Model 2 on other hand, changes speed by checking surrounding neighbors. Due to the weight, slow down behavior (Table 2b: ID-2) always dominates the speed up behavior (Table 2b: ID-3). In other words, agents always want to speed up, but will stop moving if there is a neighbor in front. The travel path of model 2 (Figure 4d) shows that to reduce the collision to a minimum, agents only move when needed. Hence, there is no collision between entities compared to model 1 (collision violation occurs on average 1.2 times each step).



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

Figure 4: snapshots and travel paths of two discovered models for multiple obstacle avoidance scenario.

Table 2: Behavior descriptions of models for multiple obstacle avoidance scenario.

| **ID** | **Behavior description** |
|---|---|
| **Behavior Group: angle** | |
| **1** | Steer to the farthest space entity. |
| **Behavior Group: speed** | |
| **2** | Slow down if agents move too fast. |
| **3** | Speed up if agents move too slow. |
| Average collision violation = 1.2 | |

(a)

| **ID** | **Behavior description** |
|---|---|
| **Behavior Group: angle** | |
| **1** | Steer to the farthest space entity. |
| **Behavior Group: speed** | |
| **2** | Slow down if there is an agent in front. (W = 50) |
| **3** | Speed up if there is an entity nearby. (W = 20) |
| Average collision violation = 0 | |

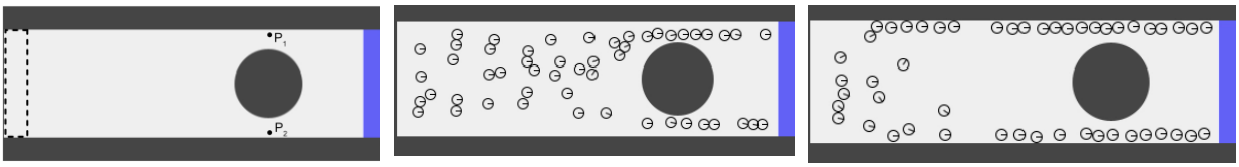
(b)

## 4.2 Combined Steering Behavior

### 4.2.1 Hall Way Evacuation with an Obstacle in the Middle

To show that our work can discover models for more complex scenarios, in this max-throughput application, agents need to avoid an obstacle while traveling through the hallway.

- Experiment set up: Figure 5a illustrates the setting with a hallway and a large obstacle in the middle. All agents have an initial heading direction = 0, are spawned within the dash line rectangle area on the left and are removed when they reach the blue rectangle on the right. Points P1 and P2 are pre-defined locations to form a desired heading area with an agent's position.
- Model 1: Agents avoid the obstacle when they are near. Table 3a and Figure 5b.
- Model 2: Agents move along the wall regardless of the distance to the obstacle. Table 3b and Figure 5c.




(a) (b) (c)

Figure 5: Scenario setup and the snapshots of two discovered models for hallway with obstacle scenario.

Table 3: Behavior descriptions of hallway with obstacle scenario.

| ID | Behavior description |
|---|---|
| **Behavior Group: angle** | |
| **1** | Select only space entity with angle distance to current heading < 30 & distance to obstacles > 30, then steer to desired area. |
| **Behavior Group: speed** | |
| **2** | If there is an agent in front, match the speed of that agent, and speed up. |
| Average collision violation = 0.1<br>Average finish time = 235 | |

(a)

| ID | Behavior description |
|---|---|
| **Behavior Group: angle** | |
| **1** | Select only space entity with the travel distance to wall < 50, then steer to desired direction. |
| **2** | Move along the wall by head toward to farthest travel distance to obstacles. |
| **Behavior Group: speed** | |
| **3** | Match speed with the leader, then increase self-speed. |
| Average collision violation = 0.1<br>Average finish time = 237 | |

(b)

Although model 1 has only one steering behavior (Table 3), it is able to handle both requirements. When approaching the obstacle, agents steer at an angle within 30 degrees from left or right depending on which side gives them the farthest travel distance. By selecting one of the space entities in front of agents (Table 3a: ID-1), they can move forward to the goal while avoiding the obstacle. Model 2 on other hand, shows a different strategy where agents first move toward to the wall, then follow the wall side regardless of their distances to the obstacle by using two angle behaviors. First, table 3b – ID:1 behavior removes all space entities on the opposite side of the wall and forces agents to head toward the wall. When the agents are adjacent to the wall, the second behavior (Table 3b – ID: 2) makes them move along the wall, hence they can escape from the hallway. Both models allow agents to escape smoothly with similar a finish time score and very few collisions.

### 4.2.2 Leader-Following while Avoiding Multiple Obstacles

- Experiment set up: The combination between experiments 4.1.1 and 4.1.2.
- Model 1: Agents constantly move around the leader while avoiding obstacles.
- Model 2: Agents warp around and align with the leader's heading while avoiding obstacles.

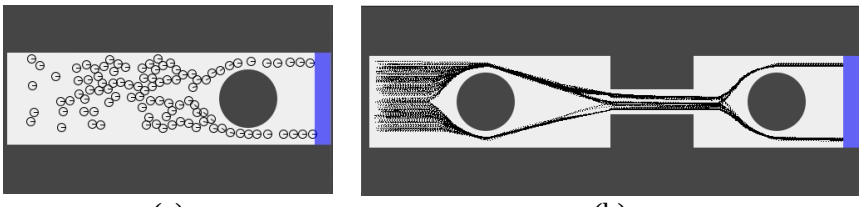
Model 1's behaviors are similar to model 1 in experiment 4.1.1 (Table 1a). Besides that, this model provides one more steering behavior that makes agents' headings match the moving direction of the leader (Table 4: ID - 2). As a result, the collision violation is reduced significantly compared to discovered models in Table 1c. (4.6 vs 9.3) while the average distance to the leader is maintained well.

Table 4. Behavior description of a group of agents follow a leader and avoid obstacles scenario.

| ID | Behavior description |
|---|---|
| **Behavior Group: angle** | |
| **1** | If self-speed > 1.5, select only space entities with the angle distance to desired range < 60, then steer to the farthest one (W = 50) |
| **2** | The nearer to the leader, the more agents want to match its angle. (W= [0-40]) |
| **Behavior Group: speed** | |
| **3** | Match speed with the nearest neighbor, then increase self-speed. |
| Average collision violation = 4.6<br>Average distance to leader = 41.2 | |

### 4.3 Robustness and Scalability of the Discovered Models

Beside the fitness scores, we further test the quality of table 3a and 4 models in two aspects: scalability and adaptation. For the max-through put model (Section 4.2.1 and table 3a), agents are spawned with a faster rate (Figure 6a) (Spawn every 8 timesteps compared to 14 timesteps as before). The layout is also more challenging by adding obstacles along the hallway (Figure 6b), and forcing the steering behaviors process further to make agents navigate through the map's layout. Since the complexity of the experiment increased, both collision and finish time score are increased (compare to table 3a results). However, agents' movements remain smooth even when they travel through the narrow path in the middle (agent's travel paths illustrate in figure 6b).

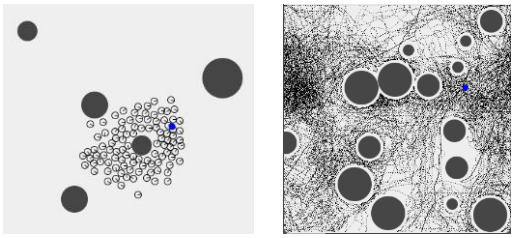


(a) (b)

Figure 6: snapshot of scalability test (a) and travel path of adaptation test (b) of experiment 4.2.1.

For the leader-following while avoiding obstacles scenario (Section 4.2.2 and table 4), the agents' population is increased from 20 to 100. The number of obstacles also increases and has different sizes. Figures 7a and 7b show snapshots of the model that is tested with scalability and adaptation respectively. Because the size of the population increases 5 times, the average distance to the leader increases to 72, and the average collision violation increases to 26. As agents' travel paths in figure 7b illustrate, agents can avoid obstacles of different sizes so well that there is no obstacle overlap violation.

Overall, these tests show that even with an automated approach, several discovered models are good and robust. They can handle different scenarios regardless of the number of agents, or more challenging environment layout.



(a) (b)

Figure 7: snapshot of scalability test (a) and travel path of adaptation test (b) of experiment 4.2.2.

## 5 CONCLUSION

This work provides agents options to interact with space surrounding them using space entities. By applying the adaptive space entity generation method, the evaluations for each of the space entity are meaningful, and the computational cost is kept low. Different distance property information provides a wide range of options for agents to filter, select, and act. With the results above, the automated discovery model approach not only discovers non-bias models, but also provides multiple solutions for each studied scenario. Future works include: First, extending space entity to work with cell spaces where more property information is possible. For example, instead of letting agents know the desired area in advance, agents can keep track of which cell space they already visited. Second, a hybrid approach can be used where modelers use discovered models' specifications as a starting point and further improve discovered models using the traditional approach. Third, more real life applications need to be tested to evaluate this work further such as: evacuation from a group of agents in a closed room with one narrow escape door, or leader-following with mobile obstacles.

## REFERENCES

Anees, M. V., and S. G. Kumar. 2022. "Identification of crowd behaviour patterns using stability analysis". *Journal of Intelligent & Fuzzy Systems* vol. 42(4), pp. 2829-2843.

Amasyali K., and N. M. Gohary. 2018. "A review of data-driven building energy consumption prediction studies". *Renewable and Sustainable Energy Reviews* vol. 81, pp. 1192-1205.

Auletta, F., D. Fiore, M. J. Richardson, and M. D. Bernardo. 2022. "Herding stochastic autonomous agents via local control rules and online target selection strategies". *Autonomous Robots* vol. 46, pp. 469–481.

Boatright, C. D., M. Kapadia, J. M. Shapira, and N. I. Badler. 2014. "Generating a multiplicity of policies for agent steering in crowd simulation". *Computer animation & virtual worlds* vol.25(5), pp. 483-494.

Braver, L. and A. A. Malikopoulos. 2020. "An energy-optimal framework for assignment and trajectory generation in teams of autonomous agents". *Systems & Control Letters* vol. 138.

Dutra, T. B., R. Marques, J. B. Cavalcante-Neto, C. A. Vidal, and J. Pettré. 2017. "Gradient-based steering for vision-based crowd simulation algorithms". *Computer Graphics Forum* vol.36, pp. 337-348

Galvane, Q., M. Christie, R. Ronfard, C. K. Lim, and M. Cani. 2013. "Steering Behaviors for Autonomous Cameras". In *Proceedings of Motion on Games*, edited by S. N. Spencer, pp. 93–102. New York, Association for Computing Machinery.

Hesham, O., and G. Wainer. 2021. "Explicit Modeling of Personal Space for Improved Local Dynamics in Simulated Crowds". *ACM Transactions on Modeling and Computer Simulation* vol. 31(4), pp. 1-29.

Keller, N., and H. Xiaolin, 2019. "Towards Data-Driven Simulation Modeling for Mobile Agent-Based Systems". *ACM Transactions on Modeling and Computer Simulation* 29(1):1-26.

Le. H,. and X. Hu. 2020. "Extended Model Space Specification for Mobile Agent-Based Systems to Support Automated Discovery of Simulation Models". In *Proceedings of the 2020 Winter Simulation*

*Conference*, edited by K.-H. Bae, B. Feng, S. Kim, S. Lazarova-Molnar, Z. Zheng, T. Roeder, and R. Thiesing, pp. 2233-2244. New York, IEEE.

Manuele, B., E. Ferrante, Mauro, B, and M. Dorigo. 2013. "Swarm robotics: a review from the swarm engineering perspective". *Swarm Intelligence* vol.7, pp. 1-41.

Reynolds, C. W., 1987. "Flocks, Herds and Schools: A Distributed Behavioral Model". *ACM SIGGRAPH Computer Graphics* 21(4):25-34

Reynolds, C. "Steering Behaviors For Autonomous Characters". https://www.researchgate.net/publication/2495826. Accessed March. 10, 2022.

Rodrigues R. A., A. L. Bicho, M. Paravisi , C. R. Jung, L. P. Magalhães, and S. R. Musse. 2009. "An interactive model for steering behaviors of groups of characters". *Applied Artificial Intelligence* vol.24, pp. 594-616.

Silveira, R., F. Dapper, E. Prestes, and L. Nedel. 2010. "Natural steering behaviors for virtual pedestrians". *The Visual Computer* vol. 26, pp. 1183–1199.

Yang S., T. Li, X. Gong, B. Peng, and J. Hu. 2020. "A review on crowd simulation and modeling". *Graphical Models* vol. 111.

Solmaz B., B. E. Moore; and M. Shah. 2012. "Identifying Behaviors in Crowd Scenes Using Stability Analysis for Dynamical Systems". *IEEE Transactions on Pattern Analysis and Machine Intelligence* vol. 34(10), pp. 2064-2070.

Song H., A. Varava, O. Kravchenko, D.Kragic, M. Y. Wang, F. T. Pokorny and K. Hang. 2021. "Herding by caging: a formation-based motion planning framework for guiding mobile agents". *Autonomous Robots* vol.45, pp. 613–631.

Tan K. R., and S. Bora. 2019. "Adaptive parameter tuning for agent-based modeling and simulation". *Simulation and Optimization in Engineering* vol 95(9), pp. 771-796.

Trivedi A., and S. Rao. 2018. "Agent-Based Modeling of Emergency Evacuations Considering Human Panic Behavior". *IEEE Transactions on Computational Social Systems* vol. 5, pp. 277-288.

Venkatramanan S., B. Lewis, J. Chen, D. Higdon, A. Vullikanti, and M. Marathe. 2018. "Using data-driven agent-based models for forecasting emerging infectious diseases". *Epidemics* vol. 22, pp. 43-49.

Yıldızab, B., and G. Çağdaş. 2020. "Fuzzy logic in agent-based modeling of user movement in urban space: Definition and application to a case study of a square". *Building and Environment* vol. 169.

Wang Z., R. Zheng, T. Kaizuka, and K. Nakano. 2018. "Driver-Automation Shared Control: Modeling Driver Behavior by Taking Account of Reliance on Haptic Guidance Steering". In *IEEE Intelligent Vehicles Symposium (IV).* pp. 144-149 New York, IEEE.

Whitley, D. 1994. "A Genetic Algorithm Tutorial". *Statistics and Computing* 4:65-85.

## AUTHOR BIOGRAPHIES

**HAI LE** is a PhD student in the Computer science at Georgia State University at Atlanta. His current research focuses on Agent-Based simulation and modeling, particularly on model specifications. His email address is: hle49@student.gsu.edu.

**XIAOLIN HU** is a Professor of the Computer Science Department at Georgia State University, Atlanta, GA. He received his Ph.D. from the University of Arizona, Tucson, in 2004. His research interests include modeling and simulation theory and application, complex systems science, agent and multi-agent systems, and advanced computing in parallel and cloud environments. His work covers both fundamental research and applications of computer modeling and simulation. He was a National Science Foundation (NSF) CAREER Award recipient. His email address is: xhu@gsu.edu.