Collision Avoidance with Deadlock Prevention for Crowd Simulation

Panich Sudkhot

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science at Mahasarakham University

June 2022

การหลีกเลี่ยงการชนที่ป้องกันการติดตายสำหรับการจำลองฝูงชน

พาณิชย์ สุดโคต

เสนอต่อมหาวิทยาลัยมหาสารคาม เพื่อเป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
ปริญญาปรัชญาดุษฎีบัณฑิต สาขาวิทยาการคอมพิวเตอร์
มิถุนายน 2565

Collision Avoidance with Deadlock Prevention for Crowd Simulation

Panich Sudkhot

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science at Mahasarakham University

June 2022

The examining committee has unanimously approved this dissertation, submitted by Mr. Panich Sudkhot, as a partial fulfillment of the requirements for the Doctor of Philosophy in Computer Science at Mahasarakham University.

Examining Committee

.....	Chairman
(Asst. Prof. Chatklaw Jareanpon, Ph.D.)	(Faculty graduate committee)
.....	Committee
(Asst. Prof. Chattrakul Sombattheera, Ph.D.)	(Advisor)
.....	Committee
(Asst. Prof. Manasawee Kaenampornpan, Ph.D.)	(Faculty graduate committee)
.....	Committee
(Asst. Prof. Phatthanaphong Chomphuwiset, Ph.D.)	(Faculty graduate committee)
.....	Committee
(Asst. Prof. Khamron Sunat, Ph.D.)	(External expert)

Mahasarakham University has granted approval to accept this dissertation as a partial fulfillment of the requirements for the Doctor of Philosophy in Computer Science

.....	.....
(Asst. Prof. Sasitorn Kaewman)	(Assoc. Prof. Krit Chaimoon, Ph.D.)
Dean of the Faculty of Informatics	Dean of Graduate School
	Day .... Month ..... Year .....

ACKNOWLEDGEMENTS

I want to express thanks to the dissertation advisor, Assistant Professor Dr. Chattrakul Sombattheera, for his invaluable help and constant encouragement throughout this research. This dissertation would not have been completed without all the support I have always received from him. I am most grateful for his teaching and advice.

Thank you to my committee members, Assistant Professor Dr. Chatklaw Jareanpon, Assistant Professor Dr. Manasawee Kaenampornpan, Assistant Professor Dr. Phatthanaphong Chomphuwiset and Assistant Professor Dr. Khamron Sunat. Your encouraging words and thoughtful, detailed feedback have been very important to me.

I would like to thank Captain. Nanata Tantadsakul, Defense Technology Institute, for assistance and support for Simulation & Training Systems, and Assistant Professor Sasitorn Keawman, Dean of the Faculty of Informatics for facility encourage them to do the dissertation.

This research project was financially supported by Assistant Professor Dr. Chattrakul Sombattheera. I would like to thank you for funding this research.

Thank you to Mahasarakham University for providing opportunities and supporting scholarships for Ph.D. study.

I most gratefully acknowledge my parents and friends for all their support throughout the period of this research. Also, I extend my thanks to all my colleagues at Major of Computer Science and Major of Information Technology, Mahasarakham University, for their continuous encouragement and support.

Finally, I most gratefully acknowledge my family for their encouragement and support throughout my studies.

Panich Sudkhot

TITLE Collision Avoidance with Deadlock Prevention for Crowd Simulation

AUTHOR Mr. Panich Sudkhot

ADVISORS Asst. Prof. Chattrakul Sombattheera, Ph.D.

DEGREE Doctor of Philosophy **MAJOR** Computer Science

UNIVERSITY Mahasarakham University **YEAR** 2022

ABSTRACT

We present a multiagent-based framework for crowd simulation in large space urban area on a standalone PC. We use Belief-Desire-Intention (BDI) for modeling individual agent behavior. We use RVO for handling a large number of agents. We experimented our framework with up to 102,400 agents, navigating them from origins to destinations. We found that we can navigate agents successfully. The execution time increases when the number of agent increase. For visualization, the algorithm can 24 fps for 600 agents. The number of frame per second drop rapidly when the number of agents rises from 19,600 to 102,400 agents. The rendering is very low, 0.4 to 0.04 fps, when the number of agents is larger than or equal to 19,600. In term of memory usage, our algorithm consume slightly more memory than RVO. Both algorithm consume more memory when the number of agent is increased almost linearly. In our experiment we have 1 GB of ram and can carry out simulation up to 720,000 agents.

Keywords : Crowd Simulation, Multiagent Crowd Simulation, Crowd Simulation Framework, BDI Agent, BDI Crowd Simulation

CONTENTS

List of Tables

List of Figures

CHAPTER 1

INTRODUCTION

Crowd simulation is a very challenging and important area of research. It can be used in many areas, including movie industry, video games, crisis training, public safety, urban planning, evacuation training, etc. Among many challenges in these simulations, we consider the most important ones are modeling individual behavior and handling the computation workload incurring from simulating a large number of agents. Each agent representing human being must be modeled appropriately, i.e. as similar as possible but not too much so that it cannot carried out with limited computational power. In addition, there a lot of agents being simulated at the same time. Their behaviors interchangeably affect each other and must be taken into account during the simulation process. These additive workload normally becomes exponential rapidly when the number of agents increase. It is important that the simulation engine carries out the workload wisely in order to cope with larger number of agents.

In general, we create a hybrid simulation framework and test it against multiple environment. Chapter 3 is about agent-based Crowd Simulation where obstacles are bounded and are connected a graph. To support large scale simulation, a couple of scene creation tools were developed as well as algorithms for building graph. Chapter 4 is agent architecture where we embed BDI architecture with RVO agent resulting in a more powerful approach fro navigating agents. Chapter 5 is planning strategy for agent where BDI concept is applied at high-level for re-route agent planning and different individual planning strategies including aggressive, passive and humble. Chapter 6 is about simulations on various combined settings where execution time and simulation time are explored.

1.1 Problem Statement

Here, we propose a multiagent simulation framework for carrying out large number of crowd. We use BDI architecture [45] for modeling individual agent behavior and RVO [2] for handling a large number of agent. There are two levels of navigating

agents: high and low levels. The high level is about generating global plans for moving agents from origins to destinations. In general, a global plan of an agent will be split into several parts. We can consider each of them as a sub-plan. The low level plan is about navigating agents from an origin of a sub-plan to its destination. Because moving agents in low level must avoid collision, the mechanism for moving agents around is very important.

1.2 Objective

1. To create novel crowd simulation framework for large space urban.
2. To demonstrate planning strategy based-on BDI concept [45] from our simulation framework.

1.3 Scope

1. To create crowd simulation framework for large space urban and demonstrate planning strategy (see Chapter 4) based-on individual characteristic agents in our simulation framework.

1.4 Terminology

A wide variety of terms appear in the literature that refer to humans "inhabiting" virtual worlds. *Avatars* are characters that represent and are controlled directly by a real person. The term *avatar* is often confused with characters that take their motivations and behaviors from computer programs and simulators. These computer-driven characters or *virtual humans* may also be called *digital humans*, *autonomous agents*, or *humanoids*. We use these terms interchangeably. Although finer distinctions are plausible, they are not meaningful here: sometimes "autonomous" is equated to "undirected" behaviors, but we prefer to consider all action choices as under some sort of computational control. Different virtual human control mechanisms exist, and we will differentiate their qualities and characteristics shortly.

Existing terminology used to describe multiple beings can be even more confusing. *Artificial life*, "*boids*," and *multiagent systems* simulate more than one

(autonomous) character. Other terms used include crowds, pedestrians, groups, and populations. It is not always clear what is meant by these terms or the functionality of the individuals that these systems represent. Here we will briefly examine some of the terminology associated with crowds.

Table 1.1. WordNet definition

WordNet definitions	
crowd	a large number of things or people considered together
pedestrian	a person who travels by foot
populace	people in general considered as a whole
population	the people who inhabit a territory or state.

Crowd, *populace*, and *population* all inherit from the *group* hypernym. However, *crowd* is also a *gathering* and a *social group*, whereas *populace* and *population* are people. Hyponyms for *crowd* include:

Table 1.2. Extend WordNet definition

WordNet definitions	
army	a large number of people united for some specific purpose
crush,*jam*,*press*	a dense crowd of people
drove,*horde*,*swarm*	a moving crowd
huddle	a disorganized and densely packed crowd
mob,*rabble*,*rout*	a disorderly crowd of people
lynch mob	a mob that kills a person for some presumed offense without legal authority
phalanx	any closely ranked crowd of people
troop,*flock*	an orderly crowd.

Technical definition, we define follow:

Table 1.3. Technical definition

Technical definition	
BDI	Belief Desire Intention (BDI) is a well-known architecture for implementing intelligent agents.
RVO	a simple but powerful concept for simultaneously navigate a large number of agents to their goals in a common environment containing both static and moving obstacles.
simulation step	a step of software in execution loop.
agent	a computational model for simulating the actions and interactions of individuals.
large space	an area of at least 1 km^2 used in crowd simulation.
deadlock	a situation in which agents cannot move from their current positions.

CHAPTER 2

LITERATURE REVIEW

Crowd simulation is defined [55] as "the process of simulating the movement (or dynamics) of a large number of entities or characters." It is widely used to in entertainment industry, such as video games or movies, and serious real world applications, incuding crisis training [10], architecture and urban planning [13], and evacuation simulation [19]. Between these two major streams of crowd simulations, there are differences in focusing of their features, depending on their needs. For example, reducing complexity of the 3D scene and image-based rendering is important for realistic and fast rendering for visualization [52, 54, 32]. For entertainment purposes, these are very important features. However, we require different features for serious real life simulation. Human behavior must be precisely and comprehensively taken into account. Detailed characteristics of involved agents must be integrated. These applications include cars and pedestrians [7, 48], agents interacting with smart objects [24], and more complex physical and social dynamics [40].

In the following, we shall explore related works, starting from history of crowd simlation and recent advances that require more and more details.

2.1 History

We focus on computer-based crowd simulation. The history of crowd simulation is dated back to the 80s. Over the years, advanced techniques have improved the scalability, flexibility, applicability, and realism of simulations In the early days of crowd simulations, behavioral animation was introduced and developed [46]. Flocks of birds alongside schools of fish are studied for group intuition and movement. This basis was improved [53] later with synthetic vision of general view of the environment allowing for a perceptual awareness within their dynamic habitats. More realistic features can be achieved by a relation of the autonomous behavior of the individual within the crowd and the emergent behavior originating from this [35]. However, individualistic navigation must be carefully taken care of. Reynolds [47] believe that

steering behaviors are important for automating agents within a simulation. Low-level locomotion is dependent and reliant on mid-level steering behaviors and higher-level goal states and path finding strategies. Another dimension to improve crowd simualtion is the modeling of real time simulations of these crowds. Levels of autonomy amongst agents is modeled as hierarchical organization for human crowds. This is regarded as an elementary form on humanoid agents [36]. On the other hand, group behavior is also an interesting model for crowd simulation. A hybrid approch combining these two approaches is also introduced [58].

2.2 Crowd Dynamics

One of the major goals in crowd simulation is to Reallistically steering crowds is an important goal of crowd simulation. Collective human dynamic behaviors can be discovered by this. There are many techniques for crowd steering. Artificial intelligence techniques can be combined with crowd simulation, providing extra advantages. An important feature is time scale, i.e. how the objective of the simulation also affects the length of the simulation." Combining these characteristics and then applying classifications can provide better evaluate and organize crowd." Opposite to that, indivdual level of simulation is also presented. The first model of this idea is the entity-based approach. This model consider physical predefined and global law and social/psychological factor as guidelines for modeling. as the guideline for modeling Similar to entity-based approach, agent-based approach considers autonomous agents, equipped with certain degree of intelligence. With that, agents can react to ever changing environment. This may be one of the most realistic approach for crowd simulation, although its computational cost can be very expensive.

2.2.1 Particle System

Another individual simulation is based on particle system [3]. As the name suggests, it add more characteristics, including color, size and velocity, into simulated units, based on autonomouscity of agent-based. With particle system, it is suggested that it is very easy to implement. Another benefit is that computational cost is low. This can be achieved by reducing costly detailed simulation and integrate simuating

for common characteristic when possible. A velocity vector and position vector are used to compute velocity and position of particles. By looking around, each particle can avoid collision with other agents by changing direction. Particles systems have been widely used in films for effects such as explosions, for water effects in the 2000 movie *The Perfect Storm*, and simulated gas in the 1994 film the Mask. However, there are setback of particle systems. It is difficult to diffentialte indepdent particle from associated ones.

2.3 Abstract Review

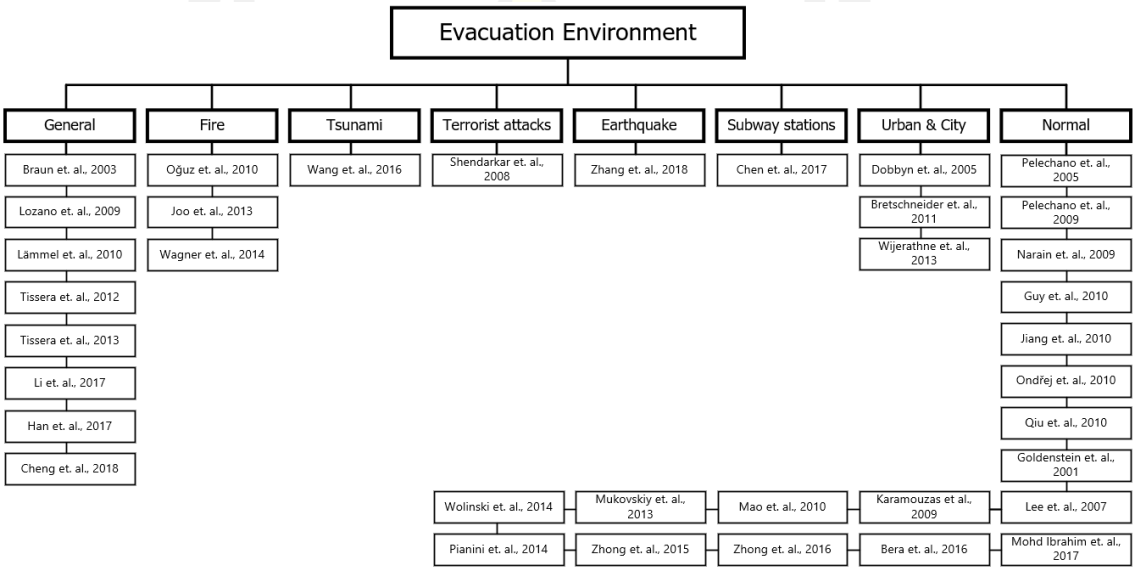


Figure 2.1. Evacuation environment group.

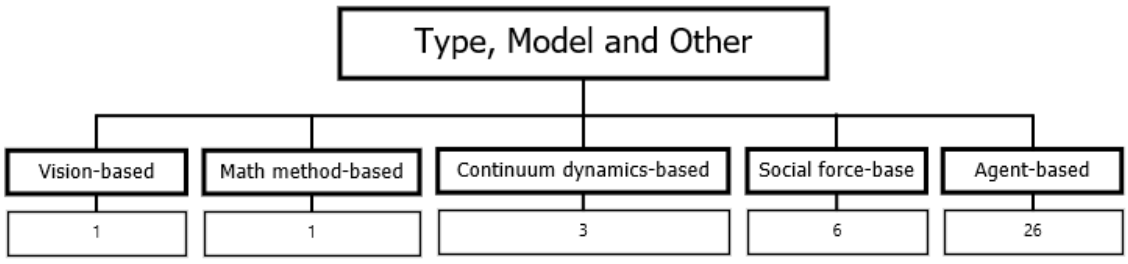


Figure 2.2. Type, Model and Other group.

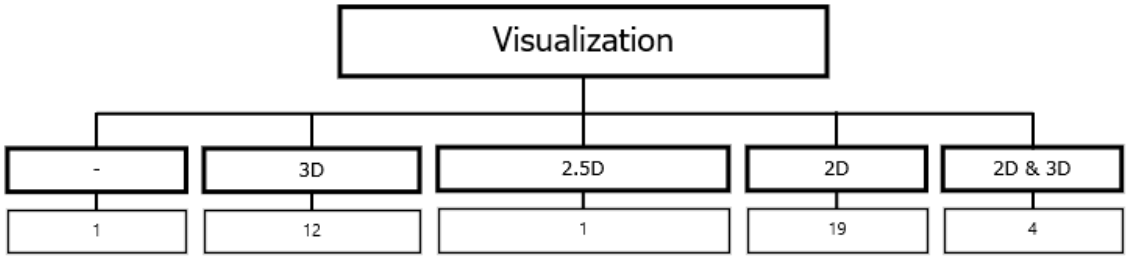


Figure 2.3. Visualization group.

2.3.1 General Evacuation

In this subsection, we review crowd simulation in general evacuation area. Braun et. al. [4] used the physically based model of crowd simulation proposed by Helbing [21] and generalized it in order to deal with different individualities for agent and group behaviors. A variety of simulations with different parameter sets shows significant impact on the evacuation scenario. Lozano et. al. [29] use appropriate computer architectures to support distributed virtual environments. Thousand of autonomous agents can be simulated on a distributed computer architecture efficiently. Based on a hierarchical software architecture, a cluster of interconnected computers help improve flexibility and robustness and efficiently provides consistency. Lämmel et. al. [30] extend multi-agent transportation simulation framework to large-scale pedestrian evacuation simulation. A simple queue model is used to simulate the traffic, taken into account free speed, bottleneck capacities, and space constraints. The most important aspects of evacuations are the congestion effects of bottlenecks and the time needed to evacuate the endangered area. Tissera et. al. [57] address the dynamics of fire and smoke propagation in simulation. The basic model used is Cellular Automata and goal oriented Intelligent Agents. Tissera et. al. [56] extend the characteristics of Intelligent Agents with certain psychological, physiological and social characteristics. In general, when perceive changes around itself, agents react appropriately. There are two submodels: pedestrian and environmental models. Li et. al. [28] use a grouping algorithm for grid density and relationship. Reasonable numbers of grids and weights are selected. The effectiveness of the algorithm is shown through simulation experiments. Han et. al. [20] modify model for collision avoidance strategy and an information transmission model that considers information

loss. The former is used to avoid collision among pedestrians in a simulation, whereas the latter mainly describes how pedestrians obtain and choose directions appropriate to them. Cheng et. al. [9] present cooperative behaviors and evacuation efficiency has been examined in detail by using a cellular automata evacuation model from a game-theoretical perspective.

2.3.2 Fire Evacuation

In this subsection, we review crowd simulation in fire evacuation area. Oğuz et. al. [39] simulate virtual crowds in emergency situations caused by an incident, such as a fire, an explosion, or a terrorist attack. A continuum dynamics-based approach is used to simulate the escaping crowd in he close proximity of the incident region. For rendering, a polygonal mesh is rendered for the agent's skeletal motion. An offline occlusion culling technique is used to speed up the animation and visualization. An affordance-based finite state automata (FSA) modeling, based on the ecological concept of affordance theory is used to simulate human behavior [23]. Wagner et. al. [59] uses agent-based modeling to simulate crowd evacuation in the presence of a fire disaster. The prototype simulate a concert venue setting such as a stadium or auditorium, allowing for any arrangement of seats, pathways, stages, exits, and people as well as the definition of multiple fires with fire and smoke dynamics included.

2.3.3 Urban & City Evacuation

In this subsection, we review crowd simulation in urban & city evacuation area, follow by:

Dobbyn et. al. [12] present a novel hybrid rendering system for crowds that solves the classic problem of degraded quality of image-based representations at close distances by building an impostor rendering system on top of a full, geometry-based, human animation system. This enables almost imperceptible switching between the two representations based on a "pixel to texel" ratio, with minimal popping artifacts. Bretschneider et. al. [5] evacuation model is a dynamic network flow problem with additional variables for the number and direction of used lanes and with additional complicating constraints. Because of the size of the time-expanded network, the

computational effort required by standard software is already very high for tiny instances. To deal with realistic instances they propose a heuristic approach. An evacuation simulation code based on Multi Agent Systems (MAS), with moderately complex agents in 2D grid environments, is developed. The main objective of this code is to estimate the effectiveness of the measures taken to smooth and speedup the evacuation process of a large urban area, in time critical events like tsunami. A vision based autonomous navigation algorithm, which enables the agents to move through an urban environment and reach a far visible destination, is implemented. This simple algorithm enables a visitor agent to navigate through urban area and reach a destination which is several kilometers away. Wijerathne et. al. [61] navigation algorithm is verified comparing the simulated evacuation time and the paths taken by individual agents with those of theoretical. Further, a parallel computing extension is developed for studying mass evacuation of large areas; vision based autonomous navigation is computationally intensive. Several strategies like communication hiding, dynamic load balancing, etc. are implemented to attain high parallel scalability. Preliminary tests on the K-computer attained strong scalability above 94% at least up to 2048 CPU cores, with 2 million agents.

2.3.4 Normal Evacuation

In this subsection, we review crowd simulation in normal evacuation area, follow by:

Pelechano et. al. [41] describe a new architecture to integrate a psychological model into a crowd simulation system in order to obtain believable emergent behaviors. Authors existing crowd simulation system (MACES) performs high level way-finding to explore unknown environments and obtain a cognitive map for navigation purposes, in addition to dealing with low level motion within each room based on social forces. Pelechano et. al. [40] propose HiDAC system (for High-Density Autonomous Crowds) focuses on the problem of simulating the local motion and global way-finding behaviors of crowds moving in a natural manner within dynamically changing virtual environments. Narain et. al. [37] present a novel, scalable approach for simulating such crowds, using a dual representation both as discrete agents and as a single continuous

system. Guy et. al. [17] present a new algorithm for simulating large-scale crowds at interactive rates based on the Principle of Least Effort. Jiang et. al. [22] present a novel approach for crowd simulation in complex environments. Authors method is based on the continuum model proposed by Treuille et. al. Compared to the original method, our solution is well-suited for complex environments. Ondřej et. al. [38] explore a novel vision-based approach of collision avoidance between walkers that fits the requirements of interactive crowd simulation. Qiu et. al. [44] present a unified and well defined framework for modeling the structure aspect of different groups in pedestrian crowds. Both intra-group structure and inter-group relationships are considered and their effects on the crowd behavior are modeled. Goldenstein et. al. [16] present a new methodology for agent modeling that is scalable and efficient. It is based on the integration of nonlinear dynamical systems and kinetic data structures. The method consists of three layers, which together model 3D agent steering, crowds and flocks among moving and static obstacles. Lee et. al. [27] present a data-driven method of simulating a crowd of virtual humans that exhibit behaviors imitating real human crowds. Karamouzas et. al. [25] propose a general method for realistic path planning, the Indicative Route Method (IRM). In the IRM, a so-called indicative route determines a global route for the character, whereas a corridor around this route is used to handle a broad range of other path planning issues, such as avoiding characters and computing smooth paths. Mao et. al. [31] method makes continuum models to be parallelizable while preserving their existing superiority of generating smooth motion. Moreover, for most of large-scale applications, our partitioning method effectively simplifies the complexity of simulation. Mukovskiy et. al. [34] introduce Contraction Theory as a tool to treat the stability properties of such highly nonlinear systems. For a number of scenarios they derive conditions that guarantee the global stability and minimal convergence rates for the formation of coordinated behaviors of crowds. Promising solutions have been suggested for such evaluations (Guy et al. (2012)). Wolinski et. al. [62] address estimating simulation parameters before evaluating: what do evaluation results mean if the assessed model is not performing at its best? Authors propose an optimization-based approach encompassing: reference data, metrics, simulation algorithms and optimization techniques. Pianini et. al. [43] propose a different point

of view on such matter: focussing on situated self-organising systems, i.e. systems in which myriads of nodes deployed in a physical environment locally cooperate in order to obtain a global coherent and robust behaviour. Zhong et. al. [65] proposes an evolutionary framework to automate the crowd model calibration process. Zhong et. al. [64] propose a novel evolutionary algorithm named differential evolution with sensitivity analysis and the Powell's method (DESAP) for model calibration. Bera et. al. [1] present an online parameter learning algorithm for data-driven crowd simulation and crowd content generation. The formulation is based on incrementally learning pedestrian motion models and behaviors from crowd videos. Mohd et. al. [33] model crowd motion subject to exit congestion under uncertainty conditions in a continuous space and compare the proposed model via simulations with the classical social force model.

2.3.5 Other Evacuation

Wang et. al. [60] presents a multi-modal evacuation simulation for a near-field tsunami through an agent-based modeling framework in Netlogo. The goals of this paper are to investigate (1) how the varying decision time impacts the mortality rate, (2) how the choice of different modes of transportation (i.e., walking and automobile), and (3) how existence of vertical evacuation gates impacts the estimation of casualties. Shendarkar et. al. [49] present a novel methodology involving a Virtual Reality (VR)-based Belief, Desire, and Intention (BDI) software agent to construct crowd simulation and demonstrates the use of the same for crowd evacuation management under terrorist bomb attacks in public areas. Studies of crowd behavior with related research on computer simulation provide an effective basis for architectural design and effective crowd management. Based on low-density group organization patterns, a modified two-layer social force model is proposed, Zhang et. al. [63] simulate and reproduce a group gathering process. First, this paper studies evacuation videos from the Luan'xian earthquake in 2012, and extends the study of group organization patterns to a higher density. A multiagent-based model is proposed on the basis of the metamodel proposed by Béhé, which includes the subway station environment abstraction model, three-level pedestrian agent model and interactive rule base [8].

The first term incorporates some notions to satisfy the specific modeling demands of subway stations and uses the object-oriented modeling method to realize a normative, extensible and flexible simulation environment building platform. The second integrates the three levels of NOMAD model and takes staged pedestrian behaviors into account, which works well for simulating real pedestrian behaviors in subway stations.

CHAPTER 3

CROWD SIMULATION FRAMEWORK

3.1 Crowd Simulation Framework

In this section, we describe our approach, using the agent-based modeling technique. The agent-based model approach has been very popular in recent years, more advances in game engine platforms and more efficiency gains computer hardware. This section explores techniques: Our an algorithm and agent-based model approaches which will be described in this section.

3.1.1 The Scenario

To make it simple to understand, let us consider a simple scenario shown in Figure 3.1. A person, waiting under the shiny sun at a bus stop near a small garden, feels thirsty and wants to go to buy a cool soft drink at a convenient store nearby. On the other side of the street lying some shops and buildings which cast shadow on the footpath. The person can cross the street walking under the shadows, crossing another street and get to the store. This seems to be a good plan but there are cars traveling on the streets and the person has to risk his luck for safety. Another plan is to walk along the street, wait for the pedestrian signal, cross the street, walk up a little bit, wait for another pedestrian signal, cross the second street and get to the convenient store. The ultimate goal is to get to the store. There are two alternatives to get there. Which one should the person use and why? Below, we shall explain how our BDI architecture works for the above scenario from a discrete simulation perspective.

3.1.2 Environment

We have created a virtual 3D environment by using Unity3D for experimenting and demonstrating the results. In the scene, there are static objects (see Figure 3.2), representing street blocks in urban area. These blocks are shown as the red rectangles in the figure. They are obstacles that agents are supposed to avoid collision with these objects. We place twenty-nine blocks in our scene. Agents are generated and located

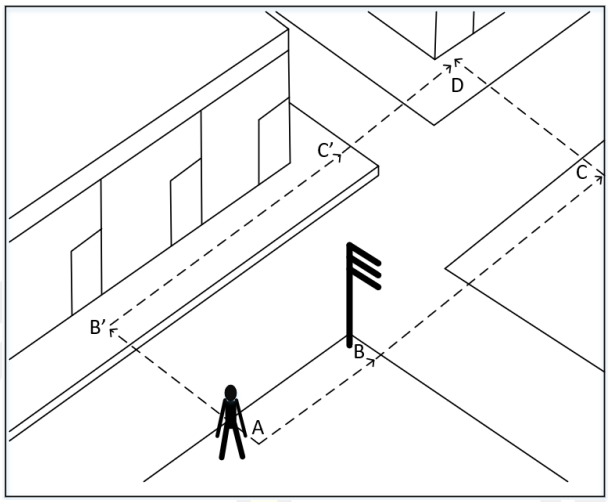


Figure 3.1. A simple world in which an agent wants go to a convenient store at position D.

in the blue rectangle. The target of all agents is the green area. When needed, the blocks can be visualized as real buildings, as shown in the figure.

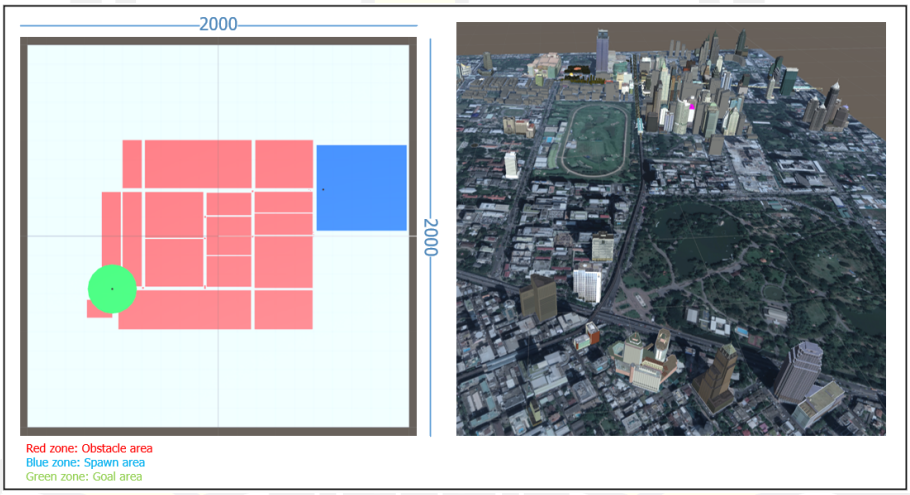


Figure 3.2. Environment: Large Space Urban Scenario.

3.1.3 The Unity Engine

We use Unity (Game Engine) for display virtual agent in this work. Unity gives users the ability to create games in both 2D and 3D, and the engine offers a primary scripting API in C Sharp (programming language), for both the Unity editor in the form of plugins, and games themselves, as well as drag and drop functionality. The engine has support for the following graphics APIs: Direct3D on Windows and Xbox One;

OpenGL on Linux, macOS, and Windows; OpenGL ES on Android and iOS; WebGL on the web; and proprietary APIs on the video game consoles. Additionally, Unity supports the low-level APIs Metal on iOS and macOS and Vulkan on Android, Linux, and Windows, as well as Direct3D 12 on Windows and Xbox One. We demonstrate how to integrate the core engine with the agent.

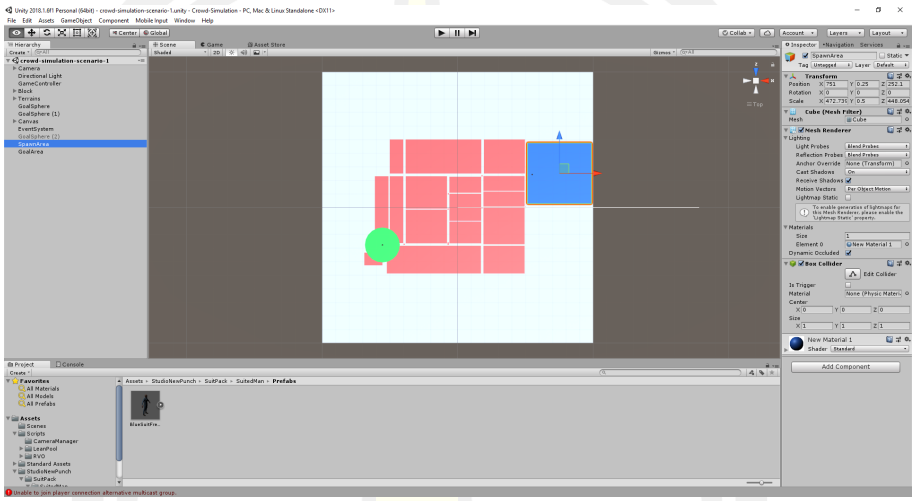


Figure 3.3. The Unity Engine

3.1.4 Controlling Behavior-Based Animation within Unity

To animate the virtual agents, behavior, perception, we have used human model for use as virtual agent. For virtual agent moving realism comply with our simulator, need to create a virtual agent animation. In this work, we have been used the animator controller asset for animate to an virtual agent. The animator controller manages the various animation states and the transitions between them using a so-called *State Machine*, which could be thought of as a kind of flow-chart, or a simple program written in a visual programming language within Unity. However, it can be added more state to an virtual agent. Parameters send-receive between engine and our simulator, for realism to situations in the environment. The position of static obstacles and moving obstacles for compute in our simulator, we used get *Transform* positions (x, y, z) positions from core engine of Unity through *ObstacleCollect* module.

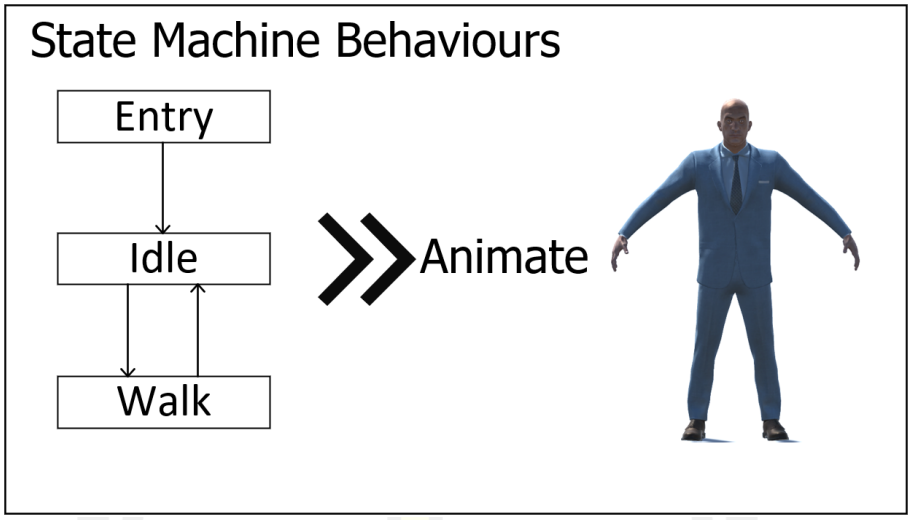


Figure 3.4. State Machine Behaviour.

Figure 3.4 show how control parameters for moving animate of the virtual agent, controlling by *State Machine Behaviors*. Show that defined state, In this work, just wanted to WALK state animate. For example of IDLE and WALK state animation mechanism, when simulator have been started animation state will be defined to IDLE, when happen defined events WALK state will take place.

3.2 Vertex Generator

In this section, we show how generate vertices (or nodes) around obstacle. We use the method of Trigonometric function, calculated from the angle of the obstacle.

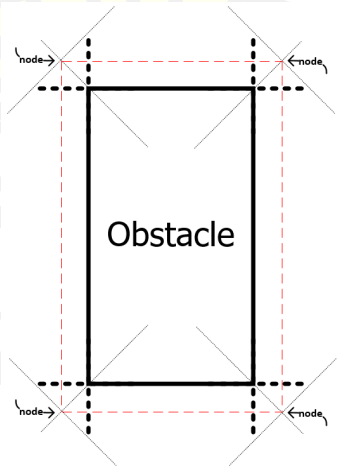


Figure 3.5. How to create nodes based-on obstacle in our framework.

Concave and Convex — A convex polygon has no angles pointing inwards. More

precisely, no internal angle can be more than 180°. If any internal angle is greater than 180° then the polygon is concave. In Figure 3.5 show convex quadrilateral polygon (it's have four side — angle).

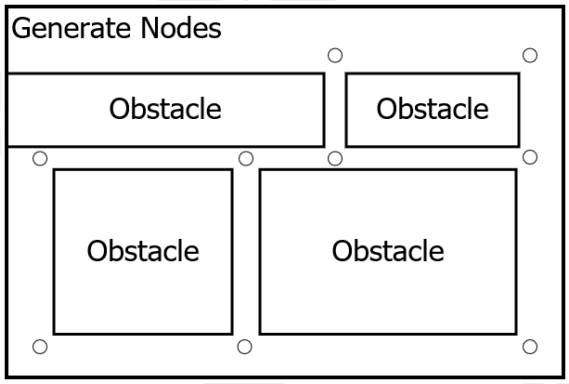


Figure 3.6. How to create multi nodes based-on obstacle in our framework.

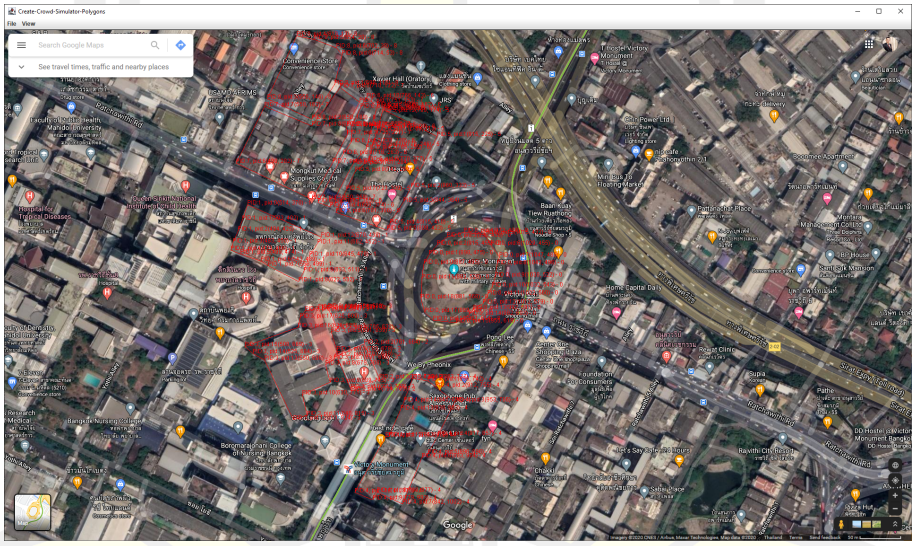


Figure 3.7. Tool for create obstacle in our framework.

In any simulation world consists many polygons inside every scenario (e.g., rocks, house, towers, tree, bridge and ambiance object). In Figure 3.6, we show small scene consists obstacles object in the scenario together with position of vertices based-on obstacles.

Figure 3.8. How to create multi nodes based-on obstacle in our framework - Real Map.

See Figure 3.8, we show how create polygon around object in real world map. We collect the values x, y and z in 3D coordinate from simulator to bring values of each vertices into the algorithm (Algorithm 1 — Vertex Generator Algorithm) for creating it to used as graph in next step.

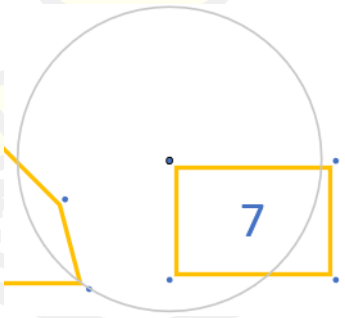


Figure 3.9. Concept for how create graph in our framework.

In Figure 3.9, show that concept of create virtual graph in our framework, we develop vertices match function for add edge to graph. After generate vertices around obstacles by generator algorithm (e.g., Figure 3.10), we will see that there is a vertices at the exterior angles of every obstacle.

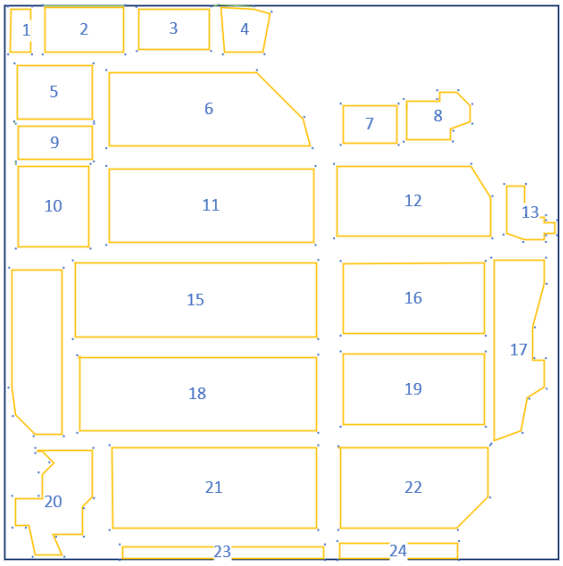


Figure 3.10. How to create multi nodes based-on obstacle in our framework - Real Map polygons.

Algorithm 1: Vertex Generator Algorithm

```
1  Function VertexGenerator(pointList, offset):
2      List of Point nodeList := null
3      numPoints := pointList.size
4      for j := 0, j < numPoints, j++ do
5          i := (j − 1)
6          if i < 0 then
7              i : + = numPoints
8          end
9          k := (j + 1) mod numPoints
10         Vector v1 := Vector(pointList[j].x − pointList[i].x, pointList[j].y − pointList[i].y)
11         v1 := Normalize(v1)
12         v1.x := v1.x ∗ offset
13         v1.y := v1.y ∗ offset
14         Point n1 := Point( − v1.y, v1.x)
15         Point pij1 := Point(pointList[i].x + n1.x, pointList[i].y + n1.y)
16         Point pij2 := Point(pointList[j].x + n1.x, pointList[j].y + n1.y)
17         Vector v2 := Vector(pointList[k].x − pointList[j].x, pointList[k].y − pointList[j].y)
18         v2 := Normalize(v2)
19         v2.x := v2.x ∗ offset
20         v2.y := v2.y ∗ offset
21         Point n2 := Point( − v2.y, v2.x)
22         Point pjk1 := Point(pointList[j].x + n2.x, pointList[j].y + n2.y)
23         Point pjk2 := Point(pointList[k].x + n2.x, pointList[k].y + n2.y)
24         Point poi := FindInterSection(pij1, pij2, pjk1, pjk2)
25         nodeList.Add(poi)
26     end
27     return nodeList
28 End Function
```

3.3 Graph Generator

In a virtual world like computer games or simulations, agents move along a virtual line from point to point, regardless of how the movement is rendered or how the scene is composed. From computer science perspective, the set of points connected by the virtual lines is a graph on which agents travel in the given world. In other

words, the graph is a global map of that world. In graph theory, each point is called a vertex and each virtual line is called an edge.

Once an agent wants to go from its current position to another one, it has to think about how it can travel to the location. There are a lot of points and virtual lines, comprising a graph, on which the agent can travel to the location. It can be thought of as a basis for generating a plan in this project. Once the state of the world is changed, e.g. other mobile agents block the agent's way, or the agent changes its goal or path, the graph may be re-generated.

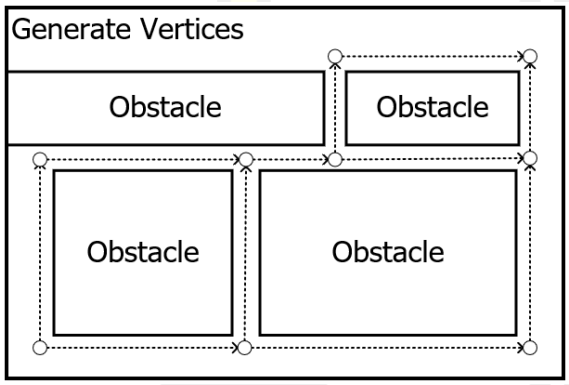


Figure 3.11. How to create vertices in our framework.

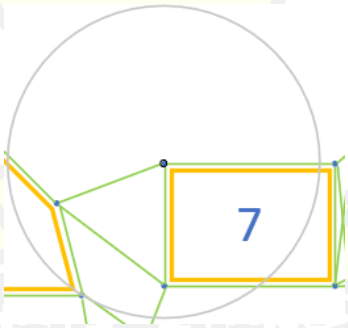


Figure 3.12. Concept for how map edge to graph.

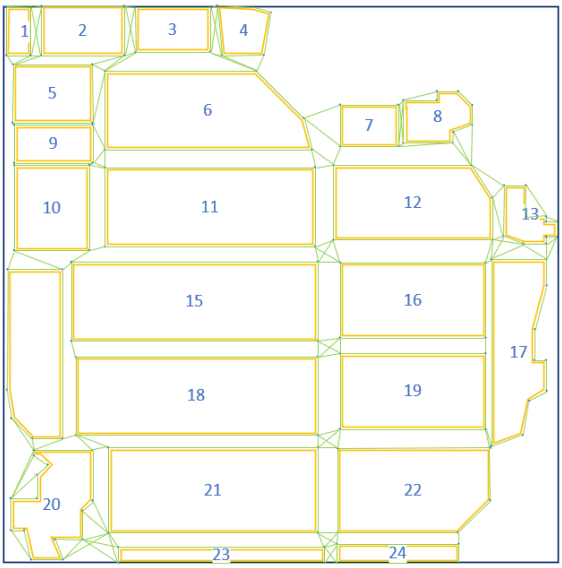


Figure 3.13. How to create vertices in our framework - Real Map.

3.4 Path Generator

Based on the graph generated above, agents can think about possible paths to travel to the destination. Normally, given a global map of a scene (or a graph, in our system), we will think about various ways to travel to the destination because the current status of the world can change arbitrarily. Having a number of choices to select from is always better—we can then choose to travel on the most appropriate one.

One algorithm for finding the shortest path from a starting node to a target node in a weighted graph is Dijkstra's algorithm [11], looking for the optimal route from the origin to the destination. From theoretical perspective, it is commonly set to look for the shortest path connecting both position. However, the "optimal" route of the agents vary—some agents may prefer the shortest one, some agents may prefer the safest one, some agents may prefer the coolest one, etc. The preference of the agents will suggest the value of length of the path. For example, if the agent prefers the safest one, the edges crossing streets will have very high values, while edges going along streets will have low values. This way, the optimal (shortest in terms of danger) path is unlikely to cross or cross fewer streets. We will use one of the Dijkstra's shortest path algorithm implementation for this work.

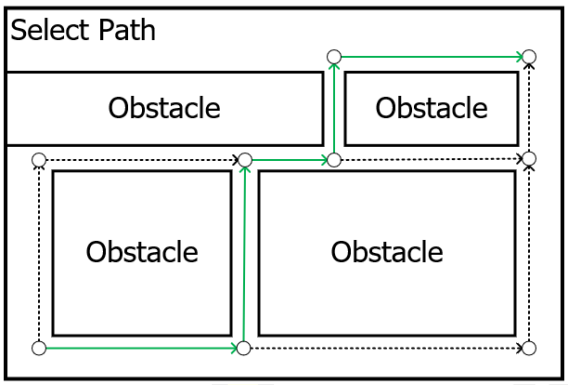


Figure 3.14. How to select path in our framework.

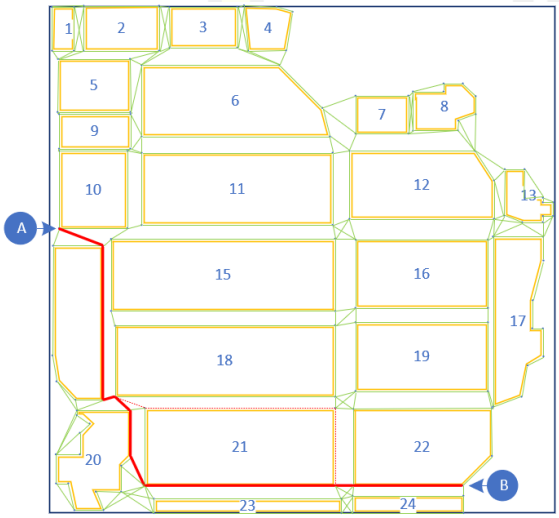


Figure 3.15. How to select path in our framework - Real Map.

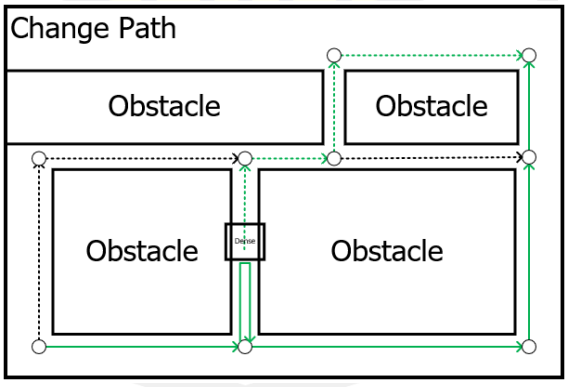


Figure 3.16. How to change path in our framework.

3.4.1 Dijkstra's Shortest Path Algorithm

The most famous algorithm for finding shortest path is Dijkstra's [11]. The reason is not only because it is the first of its kind but also it is based on a simple principle but very powerful. There a lot of direct and indirect applications and modifications of this algorithm. The principle is to find the next nearest neighbor of the present node. The process is repeated until the destination is found. Better paths can be achieved by revising the present solution, i.e. backtrack to the next alternative and proceed until either the better path is found or it is certain that a better one on this branch will never be achieved. A simple pseudo-code of the algorithm is presented in Algorithm 2.

Algorithm 2: Dijkstra's Shortest Path Algorithm

```
1  Function Dijkstra(Graph, source):
2      foreach vertex v in Graph do
3          dist[vertex] := infinity
4          previous[vertex] := undefined
5      end
6      dist[source] := 0
7      Q := the set of all nodes in Graph
8      while Q is not empty do
9          u := node in Q with smallest dist[]
10         remove u from Q
11         foreach neighbor v of u : do
12             alt := dist[u] + dist_between(u, v)
13             if alt < dist[v] then
14                 dist[v] := alt
15                 previous[v] := u
16             else
17                 is not neighbor key
18             end
19         end
20     end
21     return previous[]
22 End Function
```

3.5 Experiment

In this section, we design a scenario for experimenting large and dense crowd simulation in large space urban environment. We implemented high level work for agents, planning and navigating as previously discussed. The low level collision avoidance is taken care by RVO2 mechanism. Our improved algorithm is able to use hardware resource efficiency. Furthermore, we realistically visualize the simulated agents under Unity3d game engine. In our experiments, we focus on navigating agent around. The results and discussions will be shown below.

3.5.1 Simulation Environment

We set up a large space of 2,000 x 2,000 unit^2. As shown in Figure 3.2. Since we simulate a large crowd in urban area, we also create 29 red large blocks, representing street blocks in real world. Their sizes are different. Their overall layout is still rectangular. The blue rectangle is the original area in which agents are located. The green circle, at location (-555.40, 1, -276.60) of radius 10,000, is the area assigned to agents as their destinations. The white narrow space between blocks representing streets in our virtual world. Agents are supposed to move on these streets only. They are not allowed to walk through street blocks.

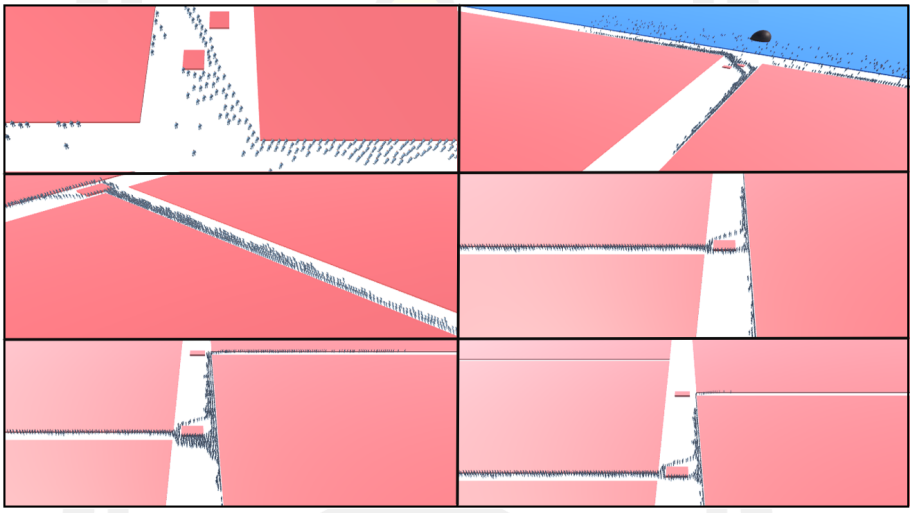


Figure 3.17. The Scenario: 1k of agent in virtual environment.

We have experimented our system with 400, 1,600, 3,600, 6,400, 10,000, 14,400, 19,600 and 25,600 agents. In each setting, individual agent will be generated to an origin and is assigned a location as its goal. Each agent will collect information about this map from the simulation engine and generate a graph, whose nodes are the junctions and intersections, and edges are streets connecting them. Each individual agent then applies Dijkstra's algorithm for their original paths. This path is the global plan, moving from one node along an edge for the the next node is a sub-plan. The agent completes a sub-plan by deploying RVO2 algorithm for avoiding collision with other agents or street blocks.

3.5.2 Computer and Platform

The computer used to run experiments was n PC (LG ISP LED 2K, 27-inch). It features a 3.60 GHz Intel Core i7-4790 processor 4 core 8 thread of CPU, 16 GB 1600 MHz DD3 of RAM, 480 GB SATA3 of SSD and AMD REDEON RX VEGA 64 of GPU. The operating system is Windows 10 x64 build 1803.

3.6 Results and Discussions

The principle of simulation is to simulate real world objects as similar as possible, given a available computational power. Furthermore, it helps decision makers understand the detailed events in the simulation much better with proper visualization. Therefore, we measure our system on these three aspects, i.e. simulation performance, execution time performance and visualization performance.

3.6.1 Simulation Performance

In the first aspect (see Figure 3.18), we are interested in the simulated time it takes, here referred to as *simulation steps*, for agents to move from origins to destinations. In general, this is the most important information retrieved from any simulation system. It gives detailed informations what really happen during the simulation. In our experiments, this tells us how long does it take agents, on average, to move from origins to destinations. Of course, other informations such as the deviated times agents take to reach destinations, how often agents have to change their plans, average paths of groups of agents, individual behavior of agents, etc.

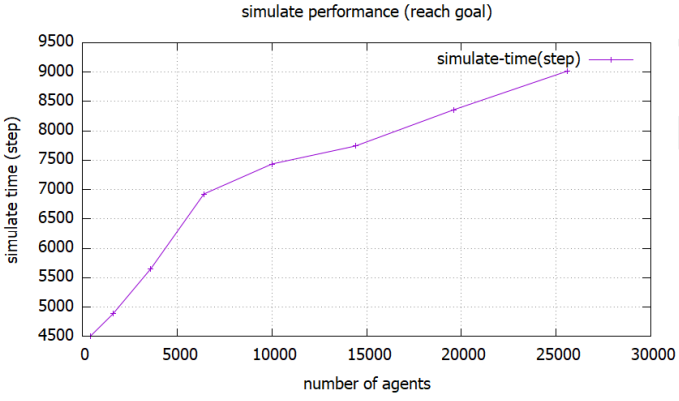


Figure 3.18. Result 1: Crowd size per simulation step.

3.6.2 Execution Time Performance

The second aspect we are interested is the actual time it takes for computer to carry out the simulation. The results are shown in Figure 3.19. This performance behave properly, the higher the number of agents, the longer time it takes. In general, this aspect of performance depends directly on the power of the computer used to carry out the simulation. In our experiment, the increased ratios between agent number and execution time is as the following. From 100 to 500 agents, the increased ratio of number is 5, while that of execution time is about 1.2. From 500 up to 20,000, the approximate ratios are 2:2, 5:2.8, 2:3, 1.5:1.6 and 1.35:2. We can then conclude that when the number of agent increases sharply, the execution time increases even sharper. This is because the overhead in executing increased reciprocal velocity obstacle in dense area. In order to handle larger number of agents, we must reduce the number of nearby agents needed for computing reciprocal velocity obstacle.

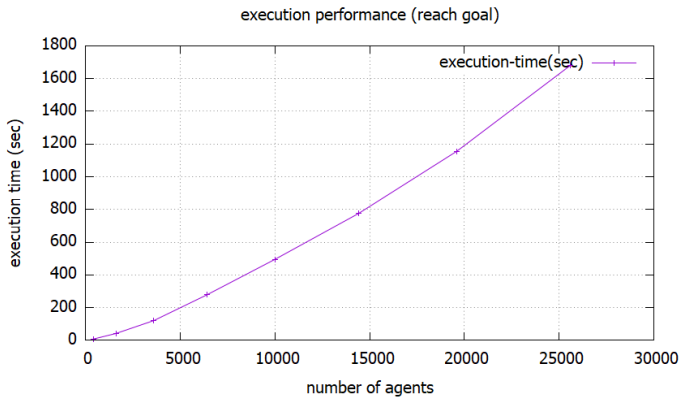


Figure 3.19. Result 2: Crowd size per real time.

3.6.3 Visualization Performance

The third aspect is visualization performance, measured in frame rate per second. Basically, it tells us how quickly and realistically simulation results can be rendered in real time and gives the viewers impressive experience. In typical video files, we have 24 frames per second rate, which gives us smooth motion pictures. Anything below that will make us feel uncomfortable. As shown in Figures 3.20 and 3.21, the frame rate drops to 23.2 when the agent number is 1,000. After that the rate becomes very low. This is down to the communication between RVO2 and Unity3d. The only way

to improve is the minimize the execution time of the simulation.

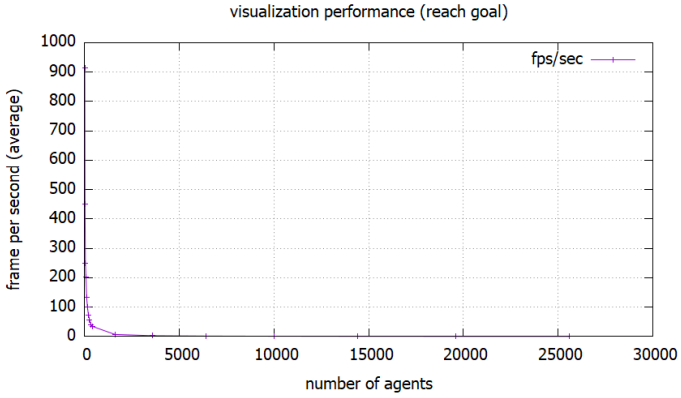


Figure 3.20. Result 3: Crowd size per FPS.

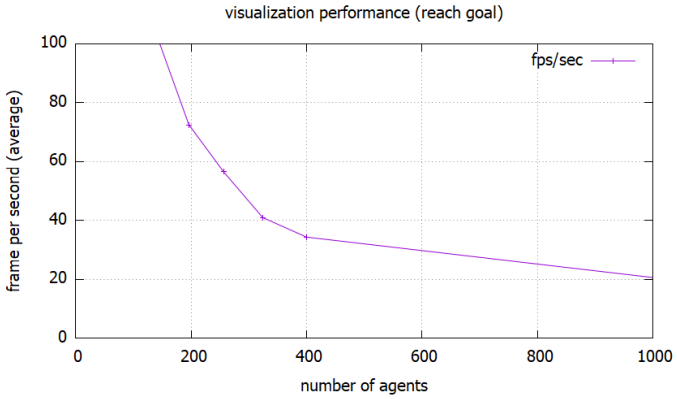


Figure 3.21. Result 3: Crowd size per FPS (Zoom).

3.7 Conclusion

We have invented a BDI architecture for agent-based simulation. We use RVO and Unity3d as our main underpinning mechanism for simulation. We carried out experiments up to 20,000 agents. We investigate three aspects of performance: simulation steps, execution time and visualization. The execution times appear alright. When the number of agent increases, the execution time increases accordingly. The visualization is not good when the number of agents is greater than 1,000. It becomes very slow and needs improvement. The simulation steps behave improperly. When the number of agents is greater than 5,000, the simulation steps do not change. We suspect the coordination between RVO and Unity3d the cause of this problem. In future

work, we aim to simulate to more complex environment, such as virtual Bangkok (see Figure. 3.17). There are a number of improvement we can do, including optimize internal processes in our simulation for better performance.

CHAPTER 4

AGENT ARCHITECTURE

In this chapter, we describe our approach for implementing simulation at agent architecture, we firstly discuss RVO2 in section 4.1. Because RVO2 works like robot agent, we improve this behavior by introducing BDI agent.

4.1 Velocity Obstacles and Multi-agent Navigation

In this section, we briefly review the original concept of Velocity Obstacles [2], derive some of its elementary properties, and show that it generates oscillatory motions when used in navigation among autonomous entities with a symmetric collision-avoidance strategy.

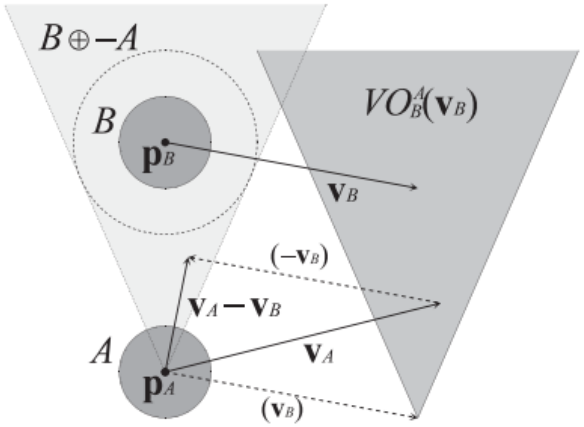


Figure 4.1. The Velocity Obstacle $VO_B^A(v_B)$ of a disc-shaped obstacle B to a disc-shaped agent A [2].

Navigation of agents has always been the first and foremost issue in crowd simulation. An outstanding requirement of simulating crowd is to avoid collision among agents. It is important that each agent remains occupying its own space that other agents must not eclipse into that particular space. Since we are interested in simulating moving agents simultaneously, taking into account moving information, i.e., the position, direction and speed of at least nearby agents, can be helpful. Berg et al. [2] propose RVO that consider other moving agents, or objects, as dynamic

obstacles, and non-moving objects as static obstacles. Let $A = \{a_1, \ldots, a_n\}$ be a set of agents, residing in plane. An agent a_i is said to be at a reference position p_i, moving with a velocity v_i, preferred speed v_i^{pref}, towards the located goal g_i. Let $O = \{o_1, \ldots, o_p\}$ be a set of obstacles, each of which is stationed at position p. In order to move agents simultaneously, a simulation time step Δt must be decide. During each cycle of the simulation, the algorithm choose the most appropriate values for each agent as its new status, i.e. speed, direction, position. The whole simulation process is repeated until each agent reaches its goal. Based on this foundation, more advanced simulation techniques [51, 50, 26] have been further developed. However, they are not considered really high level decision making simulation.

4.2 Agent Architecture

Belief Desire Intention (BDI) [45] is a well-known architecture for implementing intelligent agents. Beleif represents the fact or information about status of the world around an agent. Desire represents the goal or status the agent wants to achieve. Intention represents the plan or sequence of actions for the agent to achieve the goal, taken into account the status of the surrounding world. The most important part of BDI architecture is the planning part. The original high level algorithm is presented in Algorithm. 3. In short, the architecture allows for agents to keep revising its belief, based on the information retrieved from its sensor. The plan is then revised accordingly if needed. The plan can be recursively broken down to sub-plans, each of which has its own sub-goal.

Algorithm 3: BDI-interpreter

```
1  initialize − state();
2  repeat
3    options := option − generator(event − queue);
4    selected − options := deliberate(options);
5    update − intentions(selected − options);
6    execute();
7    get − new − external − events();
8    drop − successful − attitudes();
9    drop − impossible − attitudes();
10 end repeat
```

There are a lot of research papers and implementations available. Most of them are for generic purposes and are theory-oriented. This work intends to, starting from scratch, create a system which implements BDI agents to work in our environment. In the long term, the behavior of the agents should be similar to people. Hence, the

design of BDI agents is rather specific to this purpose. However, we cannot achieve such agents in this very first phase of the long process. We aim at laying out a reasonable foundation for further development towards the objective.

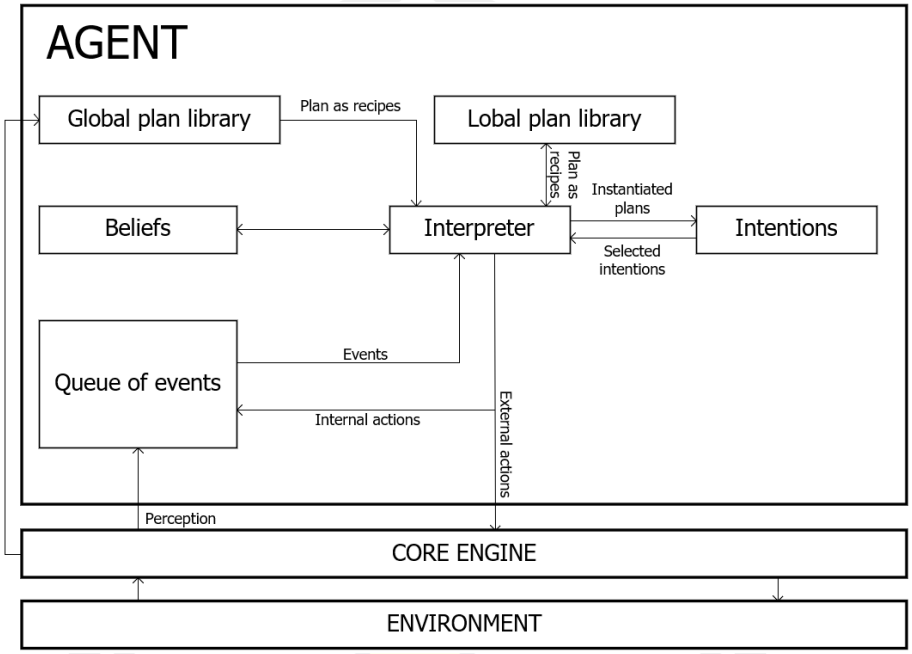


Figure 4.2. Agent design: an agent perceives and actions in the environment.

Figure 4.2 shows a agent block diagrams of agent-based model follow by BDI agent concept, deployed for crowd simulation in this experiment. Agent diagrams shows that architecture of agent, it is prototype to drive perception, navigation, selection, decision and action to animate an virtual agent in the environment.

4.2.1 Core engine

Here, we integrate our approach with the Unity engine in order to transfer information between the environment layer and the algorithm through parameters. The core engine will read and transform data from the environment, which contains the information about the obstacles. The core is supposed to read the information about these obstacles. We have developed a class, namely ObstacleCollect, for collecting information about the position of these obstacles. The class will retrieve the data and pass it on to the core simulator of the agent. The data is in the form Vector3D(x, y, z), representing the polygon point list. The data will be sent to the core simulator of the agent for calculating the unwalkable area for the agent.

4.2.2 Agent: Queue of events

Data transfer between the core engine and event queue is done by agent reading data from parameters. The agent perceives the state of the environment in which it resides via its sensor. The data collected is pass on as parameters to the core engine. The queue of the events will sorted by the proximity of each event in ascending order. The data then will be passed on to the command interpreter in form of position it is detected.

4.2.3 Agent: Belief

Belief is the information about the world and the agent. In agent research, we use the term the "world" to refer to the environment in which the agent resides. From simulation point of view, this component is therefore to keep information about objects in a given scenario. The information can be the number of objects, both static and mobile ones. The locations and shapes of the objects are common information of static objects. The speeds and direction of movement, etc. are additional information for mobile ones. The information about the agent itself includes position, speed, direction, etc. There many more pieces of information which can be kept in the agents' belief. As time goes by, the status about the world and the agent can change (by whatever reason). Pieces of information can comprise larger pieces of information. The information of new status in the world must be inserted into agent's belief.

In the given world above, the belief of agents about static objects include the information about locations and characteristics of the following objects: the streets, the buildings, the sun, the shadows, the set of lights, the convenient store, etc. The belief about mobile objects include the information, in addition to the locations and characteristics, about the speed, direction, and path of the movement of vehicles and agents.

4.2.4 Agent: Desire

Desire is the goal which the agent wants to achieve. Here, we use both terms interchangeably to refer to the same thing. However, the agent cannot achieve the goal without doing anything. The agent has to take actions in order to achieve the goal.

Sometimes, the goal can be achieved by taking a simple course of action. Sometimes, the goal need to be broken into smaller ones, each of which has a set of its own course of actions to be executed. This nested behavior goal and sub-goals can be repeated until atomic actions are derived. Sometimes agents can have multiple goals. The agents may try to achieve them all but may have to prioritize them. Agents may be satisfied by achieving a certain goal which is indifferent to a number of goals. Agents may leave out some goals once they are certainly too hard to achieve.

In the example world, the primary goal of the agent is to be at the convenient store at position D. Being at position A, the agent has to take actions, i.e. walk, to get to the store (see Figure 3.1). It cannot walk straight to the store, however, it has to cross a couple of streets and take a few turns. So, the primary goal can be broken into sub-goals, i.e. either being at a spot from where it will continue to cross or take turn. In this simple world, the agent have two alternatives, either going through B, C and D, or through B', C' and D'. Since these two sets of sub-goals are indifferent in the sense that they will eventually bring about the primary goal to the agent, one of them can be chosen, e.g. going through B and C. (The agent may have some reasons to choose one of these alternatives. This will be discussed in component Planner.) However, the agent may change the (sub-) goals by going to B' and C' because it is too hot if it goes to B and C. So, these goals must be left out.

4.2.5 Agent: Intention

Intention is the plan to achieve a goal. In other words, a plan is a course of actions to be executed to achieve a goal. Here, we use the term plan and intention interchangeably to refer to the same thing. As discussed above in the Desire section, both plan and goal relate closely to each other—the goal will be reached (or achieved) by completely executing a plan. However, there could be several alternative plans to achieve a goal. Since the goals can be broken down into sub-goals, and further down, each of these sub-goals will have, at least, a plan attached to it. Goals can be broken down to the level where actions in the plan are atomic, i.e. they cannot be broken to smaller ones anymore. As discussed above, each atomic action must be completely executed in order to achieve the associated goal. Once the agent realizes

that an atomic action of a current plan cannot be executed, due to whatever reason, that simply means the plan cannot be completed and the associated goal will never be achieved by this plan. The agent has to look for another plan, which can simply be an updated version of the plan, i.e. changing the in-executable actions with the executable ones, or a completely new plan, i.e. new course of actions. The change of plans can happen at all levels and can be reversed back to the top level. However, the process to deliberate what the alternative plans are available and decide which one is the most appropriate is discussed in Planner section.

In our simple world, as discussed above, at the top level, there are two alternative plans, each of which as three sub-plans in the second level,

I Plan–1

(a) Plan-1.1: go to B,

(b) Plan-1.2: go to C,

(c) Plan-1.3: go to D.

II Plan–2

(a) Plan-2.1: go to B',

(b) Plan-2.2: go to C'

(c) Plan-2.3: go to D.

At a coarse-grained level of this simple world, each atomic action is a move by one step. More detailed discussion about plans and actions can be bound in section Planning.

In terms of implementation, Desire contains data about plans and actions. Appropriate data types for actions and plans can be arrays of strings. Boolean functions can be used to evaluate if a plan can be executed, or an action can be completed.

Here we have three level of planning:

- **Agent: Program Interpreter** is the event translation of the agent. This blocks translate queue of events for transfer to Beliefs and Intentions blocks for compute, analyze and decide for an behaviors to the environment.

- **Agent: Global Plan Library** is the planning unit for the agent. Plan will collect data, including obstacles in the scene, from core engine. An agent will receive a plan (or a mission) from the Global Plan Library in the beginning. The plan is global and does not have any dynamic obstacle for its movement. Here, the information will be passed on to the program interpreter for further calculation and analyze.

- **Agent: Local Plan Library** works on planning for agent. The data used for planning is retrieved from the perception of agents via the program interpreter. The program interpreter will examine for dynamic obstacles, such as other agents in close proximity, and return data, direction which will ensure no collision of agents.

4.3 Agent Movement

There for, we bring about the BDI model into our simulation. However, the undesired result of the original RVO is that agents behave more like robots not human beings. BDI allows for repeating making decision. In addition, BDI allows agents to nest their plan in hierarchical manner so that agent can plan more like human beings.

4.3.1 Planning Strategy with Sub-goal

In order to get a better understanding of our BDI agents, consider a simple situation shown in Figure 4.3 where agent X wants to move from position A to B.

Basically, the agent cannot move from A to B immediately. It has to move step by step from A to B. In discrete simulation, the agent will move one step closer to the destination in each iteration. In this example, let assume the time unit is 1 second. The action the agent take in this time unit is the atomic action, i.e. moving one step at a time. In real world, we, human being, do not estimate how many steps we will take to move from one position to another. For the sake of simplicity, let us assume that the long path between A and B is broken into A, $a1$, $a2$, $a3$, $a4$, $a5$ and B, as shown in Figure 4.4 below. So, in this scenario, the agent will move from A, to $a1$, to $a2$, ..to $a5$ and to B. Each of these moves is an action.

At time $t0$, the state of agent X's BDI should be

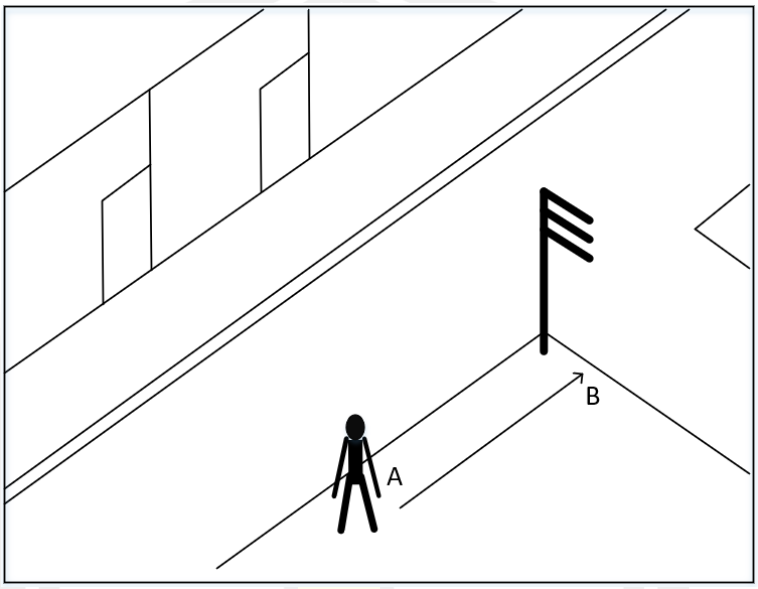


Figure 4.3. A simple world where an agent X wants to move from position A to B.

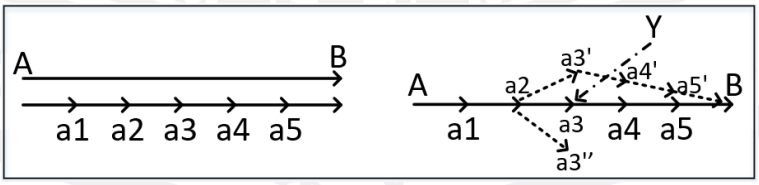


Figure 4.4. An agent changes a course of action due to an approaching agent.

- Belief:
  - There are two positions, A and B.
  - The agent is at A.
  - There are no obstacles between A and B.
  - As an inference, there is a virtual straight line, linking A and B, which the agent will follow.
- Desire:
  - The top level goal is "to be at B".
- Intention:
  - Plan: to move from A to B
  - Action set:
    1. Step from A to $a1$
    2. Step from $a1$ to $a2$
    3. Step from $a2$ to $a3$
    4. Step from $a3$ to $a4$
    5. Step from $a4$ to $a5$
    6. Step from $a5$ to B

Having setup the goal and derived the plan, the agent realizes that the goal cannot be achieve immediately, it generates a set of actions. The state after completing each action is a sub-goal. After generating action set, the Desire is updated accordingly.

- Desire:
  - To be at $a1$
  - To be at $a2$
  - To be at $a3$
  - To be at $a4$

  - To be at $a5$
  - The top level goal is "to be at B".

The agent then takes the first action in the action set. At time $t1$, the state of agent X's BDI should be:

- Belief:
  - There are positions, A, $a1$, $a2$, $a3$, $a4$, $a5$ and B.
  - The agent is at $a1$.
  - There are no obstacles between $a1$ and B.
  - As an inference, there are virtual straight lines, linking $a1$, $a2$, .., and B, which the agent will follow.

The agent then takes the first action in the action set. At time $t1$, the state of agent X's BDI should be:

- Desire:
  - To be at $a2$
  - To be at $a3$
  - To be at $a4$
  - To be at $a5$
  - The top level goal is "to be at B".
- Intention:
  - Plan: to move from A to B
  - Action set:
    1. Step from $a1$ to $a2$
    2. Step from $a2$ to $a3$
    3. Step from $a3$ to $a4$

4. Step from $a4$ to $a5$
5. Step from $a5$ to B

The agent then takes the first action in the action set. At time $t2$, the agent, X, now is at $a2$ and notices that there is an agent, Y, moving into its path. If X maintain its plan, it will collide with Y at $a3$. The belief is updated by removing $a3$, $a4$ and $a5$.

It revises its plan to avoid the collision. There are a number of alternatives to its current plan. I) Agent X may stay at its present position and let the approaching agent to go pass $a3$. Doing this may not be appropriate for the agent because it will delay its goal achievement. II) Agent X may choose to go right to $a3''$. This is risky to collide with agent Y. So, this is not an appropriate choice. III) Agent X may choose to go left to $a3'$. This will avoid collision and keep its time. So, this seems to be the optimal alternative. At $a3'$, there is a, path, a virtual straight line, connecting itself to B. However, the agent cannot move from $a3'$ to B immediately, the path is then broken to $a4'$ and $a5'$. Then the Belief is updated by adding $a3'$, $a4'$ and $a5'$. The Plan is also updated, i.e. the set of actions will be generated. Eventually, the sub-goals will be updated accordingly. Below is the update BDI.

- Belief:
  - There are positions, A, $a1$, $a2$, $a3'$, $a4'$, $a5'$ and B.
  - The agent is at $a2$.
  - As an inference, there are virtual straight lines, linking $a2$, $a3'$, $a4'$, $a5'$ and B, which the agent will follow.
- Desire:
  - To be at $a3'$
  - To be at $a4'$
  - To be at $a5'$
  - The top level goal is "to be at B".

- Intention:
  - Plan: to move from A to B
  - Action set:
    1. Step from $a2$ to $a3'$
    2. Step from $a3$ to $a4'$
    3. Step from $a4$ to $a5'$
    4. Step from $a5$ to B

  Agent X takes the first action from the action set. At time $t3$, the agent is at $a3'$. It updates its belief and senses the world. Since there are no obstacles, it does not need to change its plan. It updates its.

4.3.2 Path Planning Algorithm

In the *Agent: Local Plan Library* works on planning for agent. The data used for planning is retrieved from the perception of agents via the program interpreter. The program interpreter will examine for dynamic obstacles, such as other agents in close proximity, and return data, direction which will ensure no collision of agents.

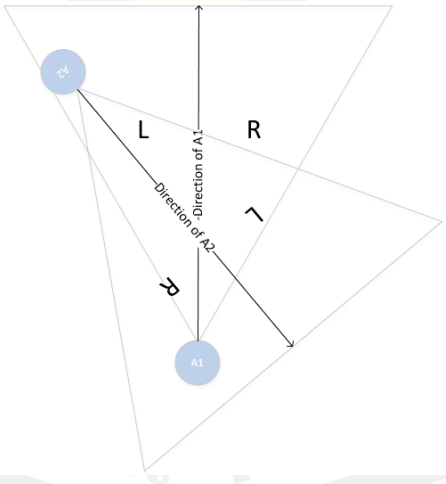


Figure 4.5. Agent decide: an agent censor detect another agents.

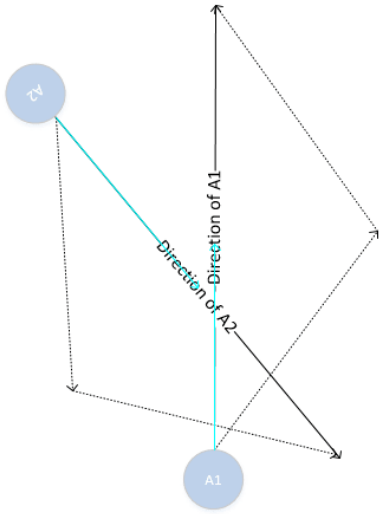


Figure 4.6. Agent decide: an agent compute 3 step forward.

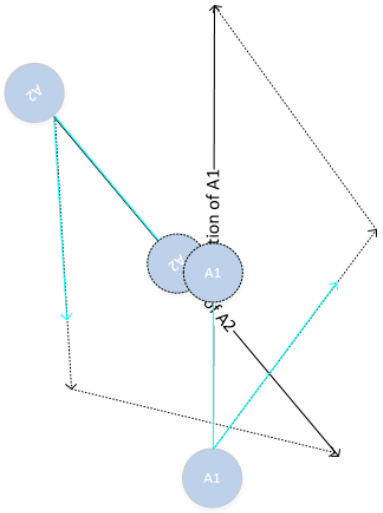


Figure 4.7. Agent decide: an agent go to right follow $A_1 argmin$.

Agent decide for go to left or right follow argument:

$$A_i argmin = \sum_{n=1}^{detect} ; left, right$$

4.3.3 Minimize Scanning Area with Circular Arcs

As human beings, we plan our paths from our knowledge about available data. While we move on, the data we perceive from our eyes may trigger changes in our plans. We see the surrounding environment in limited area where our eyes are looking at.

Typically, this area is a circular arc. In this research, we also take into account this manner, which also helps improve simulation efficiency by reducing scanning surrounding area, compared to RVO which scanning all surrounding agents. As shown in Figure 4.8, given a circle, the area of a sector, bounded by an arc of length L and a pair of radii r separated by angle θ, is $A = \frac{r^2\theta}{2}$. Since the arc is on the angle θ, It is derived that the proportion of area A to the area of the circle and angle θ to the angle of the circle is $\frac{A}{\pi r^2} = \frac{\theta}{2\pi}$. Once canceling π on both sides, we have $\frac{A}{r^2} = \frac{\theta}{2}$. We multiply both sides by r^2, we have the final result $A = \frac{1}{2}\pi r^2$. To make it works with 360 degree circle, we convert the proportion of the angle half θ with regards to 360 and we have $A = \frac{\alpha}{360}\pi r^2$. This area can help determine surrounding agents for making decision whether to change route or not.

Figure 4.8. A circular sector is shaded in green. Its curved boundary of length L is a circular arc.

4.3.4 Deadlock problem

According to "Oxford English Dictionary", deadlock means a situation a progress cannot be made. In the context of crowd simulation, a deadlock is a situation in which agents cannot move away from their positions after encounter an obstacle, either static or mobile. For a static one, an obstacle can be a building, a post, a fence, a table, etc. For a mobile one, an obstacle can be another agent, a car, a motor cycle, a bicycle, etc. Given the aforementioned definition, we carefully design the measurement at deadlock in multiple scales. A most common situation in crowd simulation is when a large number of agents gather in relatively small area. This is commonly referred to as a dense area. There, agents generally can still move but slowly. Some agents may be stuck in a deadlock, i.e., they cannot move at all. For a dense area, we measure the average speed of agents within the area. For a deadlock situation, we measure the lengthiness of agents be stuck in the deadlock.

4.4 Experimental and Results

We have implemented BDI agent within Reciprocal Velocity Obstacles for improve multi-agent navigation and tested our multi-agent navigation approach in three

challenging scenarios:

- **Single agent:** (See Figure. 4.9 for simulate path of agent to see trajectory and dense position.
- **Circle of agent:** (See Figures. 4.11 and 4.14 for circle simulate result. Figures. 4.12 and 4.15 for for circle simulate result.) A variable number of agents are distributed evenly on a circle, and their goal is to navigate to the antipodal position on the circle. In doing so, the agents will form a dense crowd in the middle.
- **Block of agent:** (See Figures. 4.17 and 4.20 for RVO blocks simulate result. Figures. 4.18 and 4.15 for our approach blocks simulate result.) Four groups of 25 agents in each corner of the environment move to the opposite corner of the environment. In the middle, there are four square-shaped static obstacles that form narrow passages. The groups with opposite goal directions meet in the narrow passage.

4.4.1 Single agent

For the first set of experiments, we use mini-scenes to evaluate moving paths and avoiding high-density areas of agents. We set simulation step at 0.25 seconds in the simulator. We set each agent's properties, such as maximum speed, radius, etc., to the same values for all experiments. In this simple scenario, the agent is targeting the opposite position to create a density at the center of the scene. We have six mini scenes. Scenes S_1, S_2, S_3, S_4, S_5, S_6 and S_7 contain 2, 3, 4, 5, 6, 50 and 100 agents. The results are shown in Figure 4.9. In each scene, we compare the result of our approach, appears on the above figure, with that of traditional RVO2, appears on the bottom figure. Since we enhance agent ability by adding BDI principles so that agents can be aware of their surroundings, as well as enhancing their route planning capabilities, the agent performance and overall performance of the simulation improve. In each scenario, our agents can avoid tight congested area (left figures), while traditional RVO tend to lead agents into tight congested area. Note that our approach allow for agents to plan with flexibility how early or far away the agent

wants to avoid the congested area. In Figure 4.9, we show the trajectory of agents paths. Our method allows agents to avoid congestion before reaching it. We choose show scenarios where the decision is to "just" avoid to tight congested area.

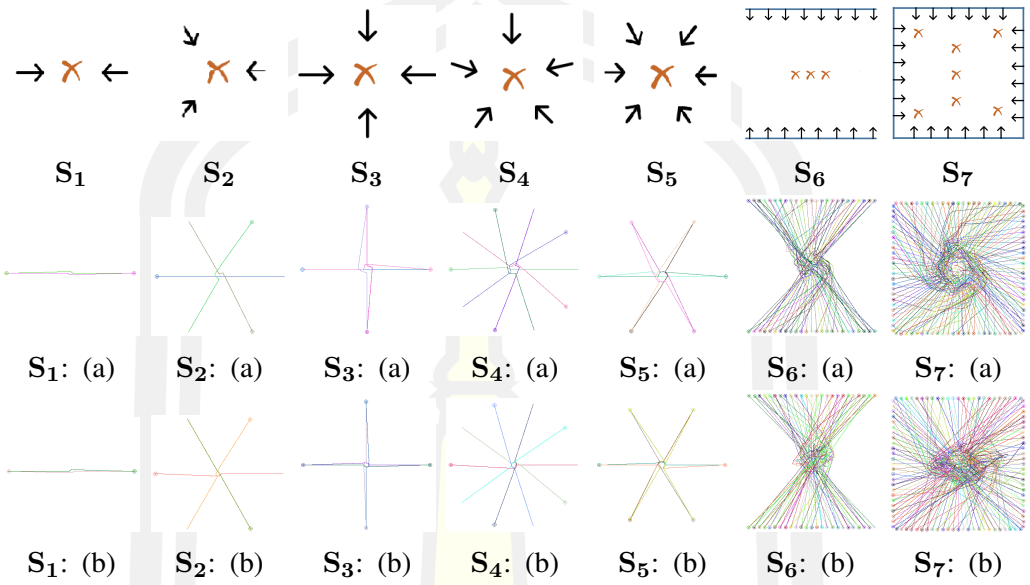


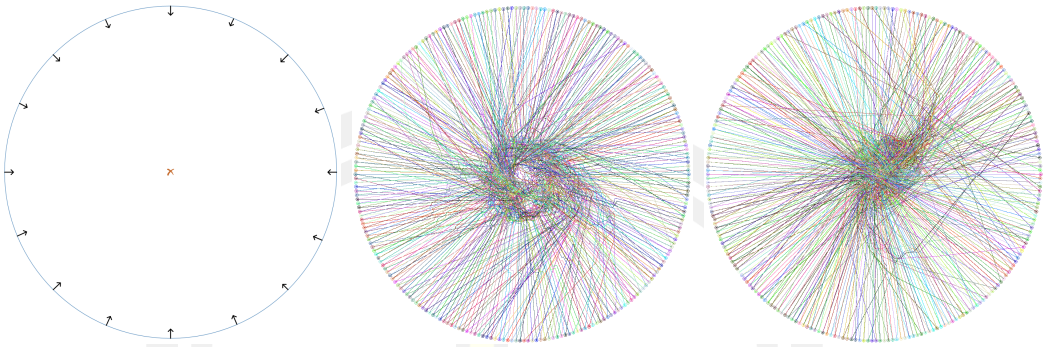
Figure 4.9. Settings: S_1, S_2, S_3, S_4, S_5, S_6 and S_7 scenario to exchange negate position and resulting paths in the scenario using our approach (a) and RVO2 approach (b).

Execution time, our approach was able to reach goal with less time to complete the scene than traditional approach. In the first scene, the agents were able to reach similar targets, while the second, third, fourth, fifth and sixth scenes clearly distinguished themselves. As the number of agents grows, the traditional approach is momentarily deadlocked as the agent moves towards the center of density. It is clear from the sixth scene that the agent group of our approach has a walking path in which the density avoidance occurs. Whereas the traditional approach agents are push, as shown in the simulation time in Table 4.1.

Setting / Method	Execution Time							Simulation Time						
	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_1	S_2	S_3	S_4	S_5	S_6	S_7
RVO2	439	452	1923	4482	516	1894	6443	361.5	362.5	14654.25	37176.5	441.75	816.25	1205.0
Our approach	449	457	474	486	507	1590	3983	361.25	362.5	371.0	377.25	376.0	655.0	710.25

Table 4.1. Simple scenario: execution time (sec) and simulation time (step).

4.4.2 Circle of agent



Circle setting 1. **(a)** **(b)**

Figure 4.10. The resulting paths in the *Circle* scenario using the our approach (a) and the RVO2 approach (b).

For the first benchmark, *Circle*, we started with an experiment containing 250 agents to show the difference between the our approach and our Reciprocal Velocity Obstacle approach. All of the agents are discs with equal radii, have the same preferred and maximum speeds, and do not have constraints on the acceleration. In both experiments, all parameters are equal. The traces of the agents are shown in Figure. 4.10 for both methods. As can be seen clearly from the figures, the our approach best avoid to goal more than RVO2.

Figure 4.11 some snapshots of the simulation's visualization results. For setting *Circle* from simulation time at: 0, 149, 299, 449, 599, 749, 898, 1,048, 1,198, 1,348, 1,498, 1,647, 1,797, 1,947, 2,097 and 2,247.

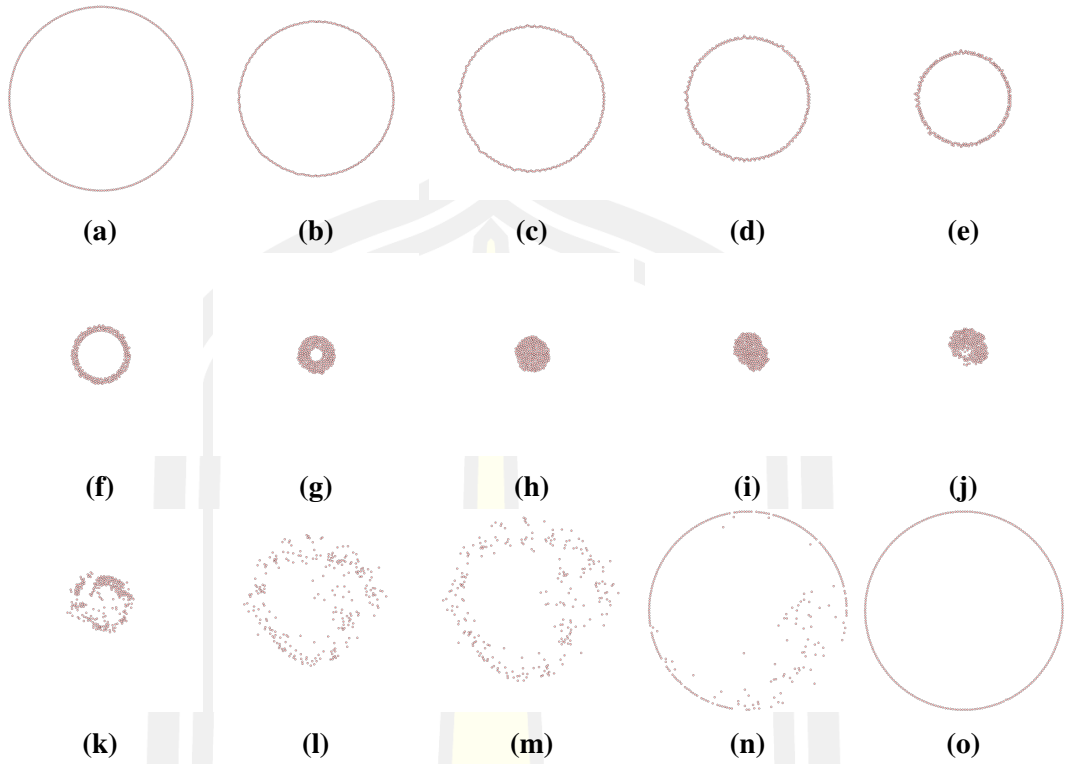


Figure 4.11. The *Circle* scenario for 250 agents(RVO2).

Figure 4.12 some snapshots of the simulation's visualization results. For setting *Circle* from simulation time at: 0, 109, 219, 329, 438, 548, 658, 768, 877, 987, 1,097, 1,207, 1,316, 1,426, 1,536 and 1,464.

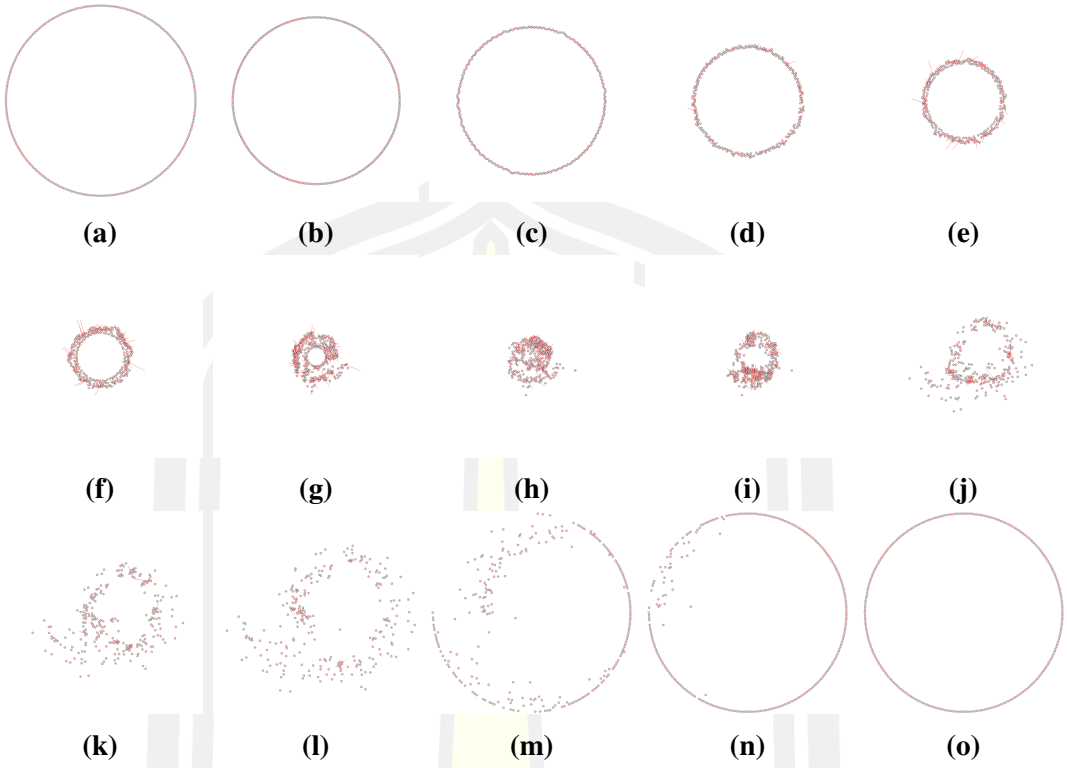
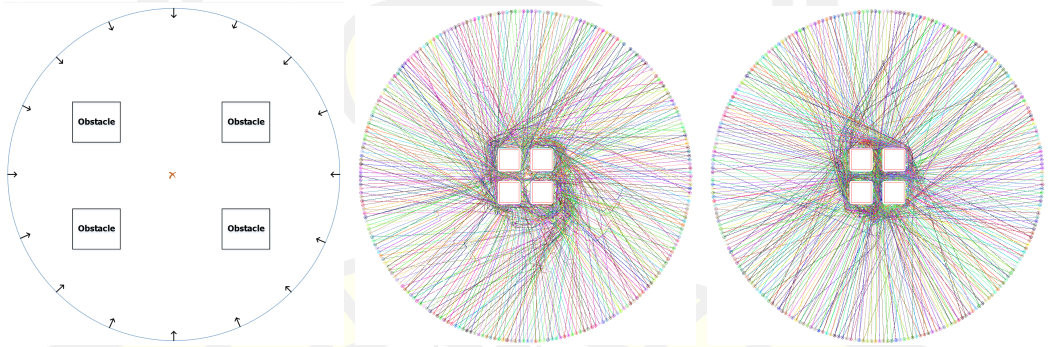


Figure 4.12. The *Circle* scenario for 250 agents(Our approach).



Circle setting 2. **(a)** **(b)**

Figure 4.13. The resulting paths in the *Circle* scenario using the our approach (a) and the RVO2 approach (b).

Figure 4.14 some snapshots of the simulation's visualization results. For setting *Circle* with obstacle from simulation time at: 0, 356, 713, 1,070, 1,427, 1,784, 2,141, 2,498, 2,855, 3,212, 3,569, 3,926, 4,283, 4,640, 4,997 and 5,354.

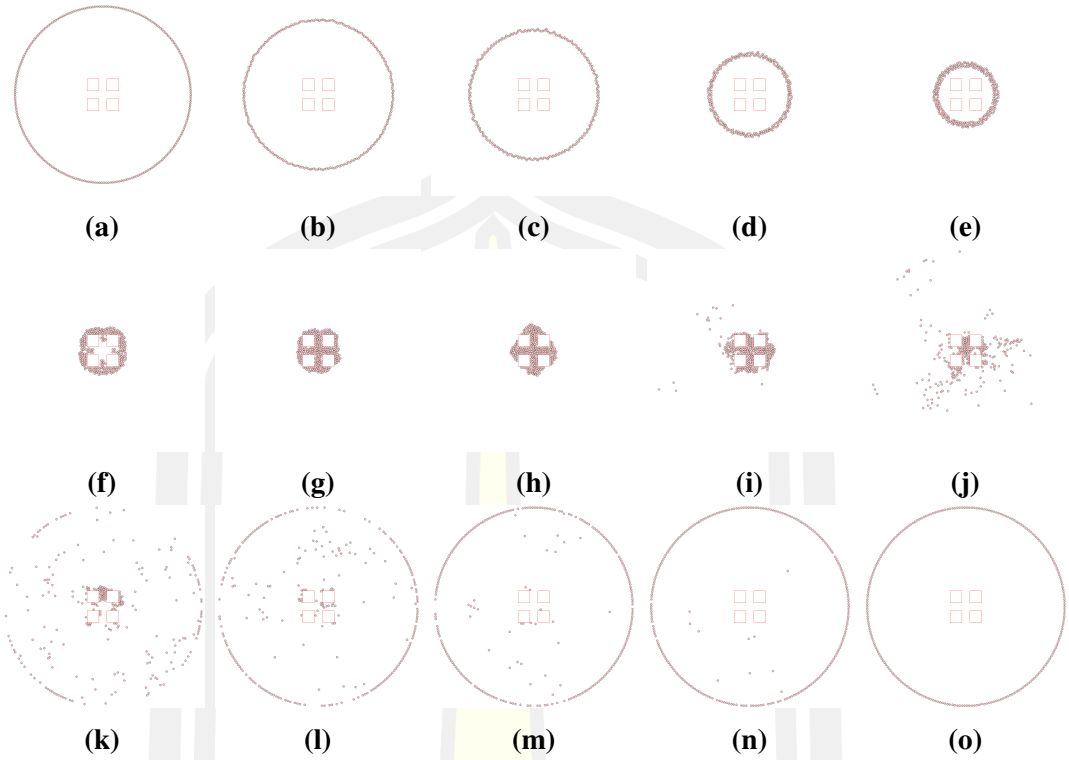


Figure 4.14. The *Circle* scenario for 250 agents (RVO2 with obstacles).

Figure 4.15 some snapshots of the simulation's visualization results. For setting *Circle* with obstacle from simulation time at: 0, 129, 259, 389, 519, 649, 779, 909, 1,039, 1,169, 1,299, 1,428, 1,558, 1,688, 1,818 and 1,948.

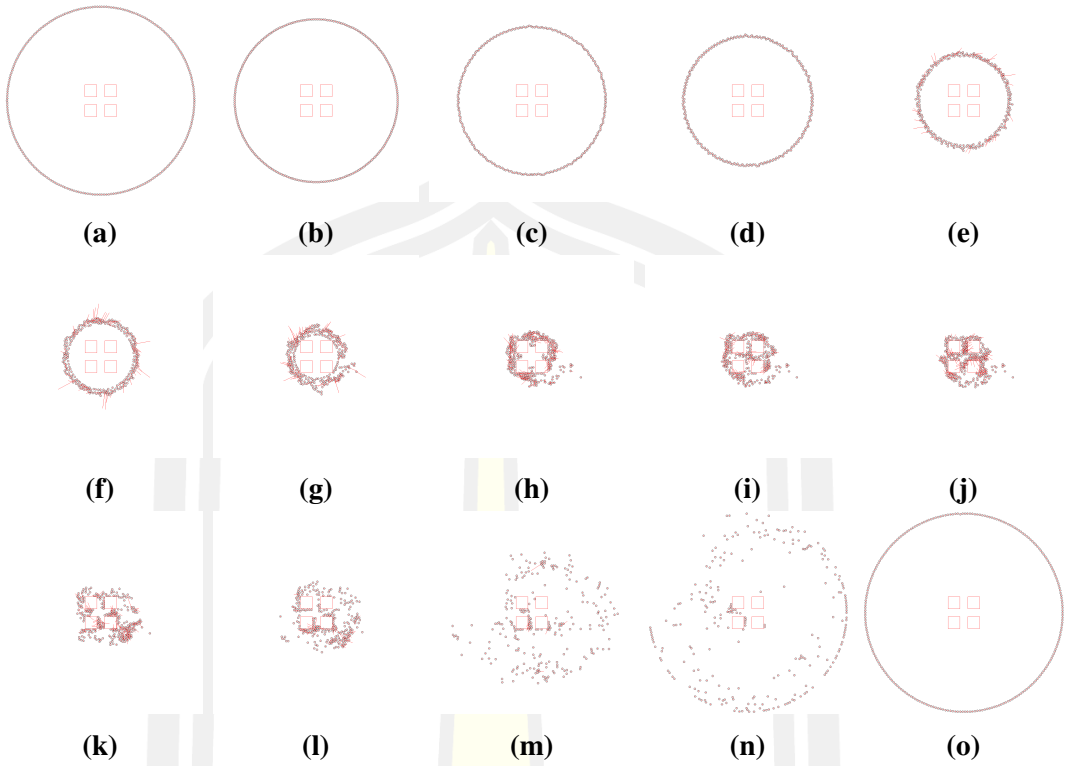


Figure 4.15. The *Circle* scenario for 250 agents(Our approach with obstacles).

Next, we varied the number of agents in the Circle benchmark to see how our approach scales when the number of agents grows (see Figure. 4.12, 4.15 and the accompanying video for the experiment with 250 agents). In this case, we used a circular neighbor region around each agent. We chose the radius of this region such that the navigation of the agents is still safe (which was 8 times the agent radius).

Setting / Method	Circle without obstacles	Circle with obstacles
RVO2	51795	124671
Our approach	36391	44222

Table 4.2. Circle settings scenario: execution time.

Setting / Method	Circle without obstacles	Circle with obstacles
RVO2	2247.25	5354.25
Our approach	1646.0	1948.5

Table 4.3. Circle settings scenario: simulation time.

Clearly, the amount of time needed to generate one frame (i.e., process one cycle in the simulation) scales linearly with the number of agents. Only the neighbor

selecting routine has a quadratic nature, but this step is negligible in the total running time. The graph also shows that even for 1000 agents, we are able to generate more than 10 frames per second. We chose the time step Δt to be 0.25 seconds in this experiment, so these results are obtained in real-time frame rates. We note that the running time of our method scales linearly with the value of parameter N, the number of velocities sampled for each agent in each frame. We fixed this value at 250 in our experiments. We believe that smarter sampling can further improve the performance. Furthermore, since we perform an independent computation for each agent, the approach is fully parallelizable. We took advantage of this feature and used 3.6 GHz AMD Ryzen 5 2600x processor for performing the experiments.

4.4.3 Block of agent

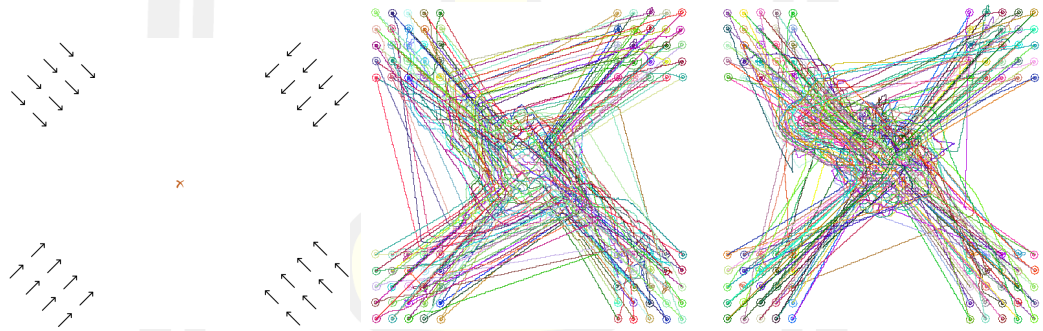


Figure 4.16. The resulting paths in the *Block* scenario using the our approach (a) and the RVO2 approach (b).

Figure 4.17 some snapshots of the simulation's visualization results. For setting *Block* from simulation time at: 0, 96, 193, 289, 386, 483, 579, 676, 773, 869, 966, 1,062, 1,159, 1,256, 1,352 and 1,449.

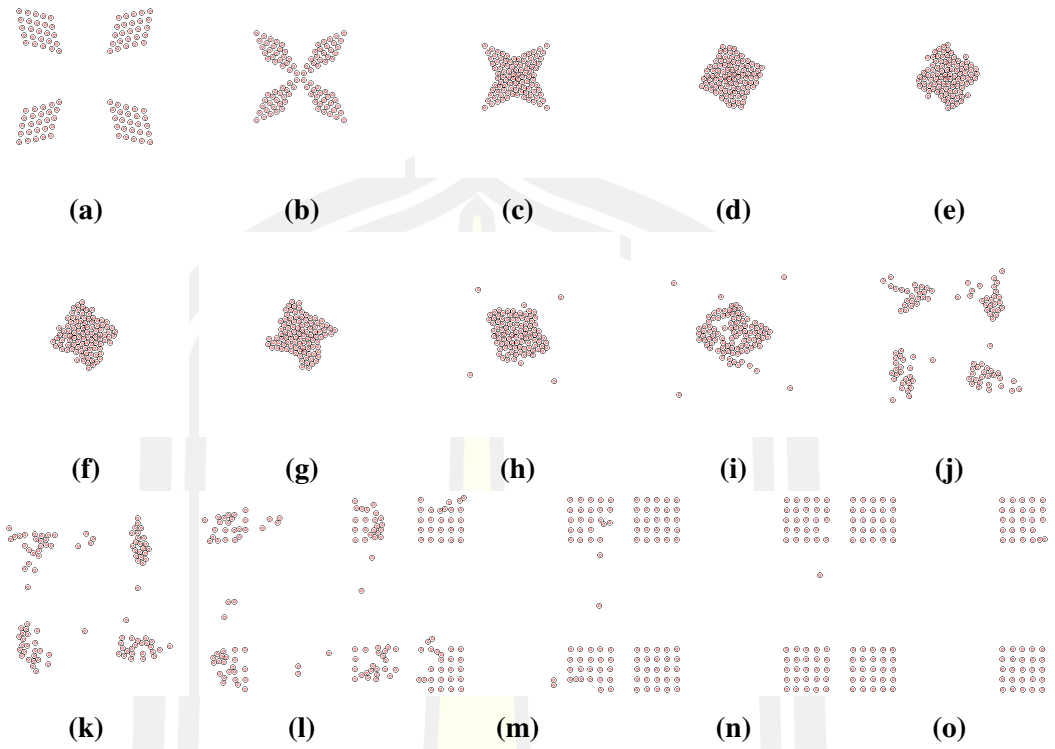


Figure 4.17. Four groups in opposite corners of the environment exchange positions in the *Narrow Passage* scenario (RVO2).

Figure 4.18 some snapshots of the simulation's visualization results. For setting *Block* from simulation time at: 0, 54, 108, 163, 217, 272, 326, 380, 435, 489, 544, 598, 653, 707, 761 and 816.

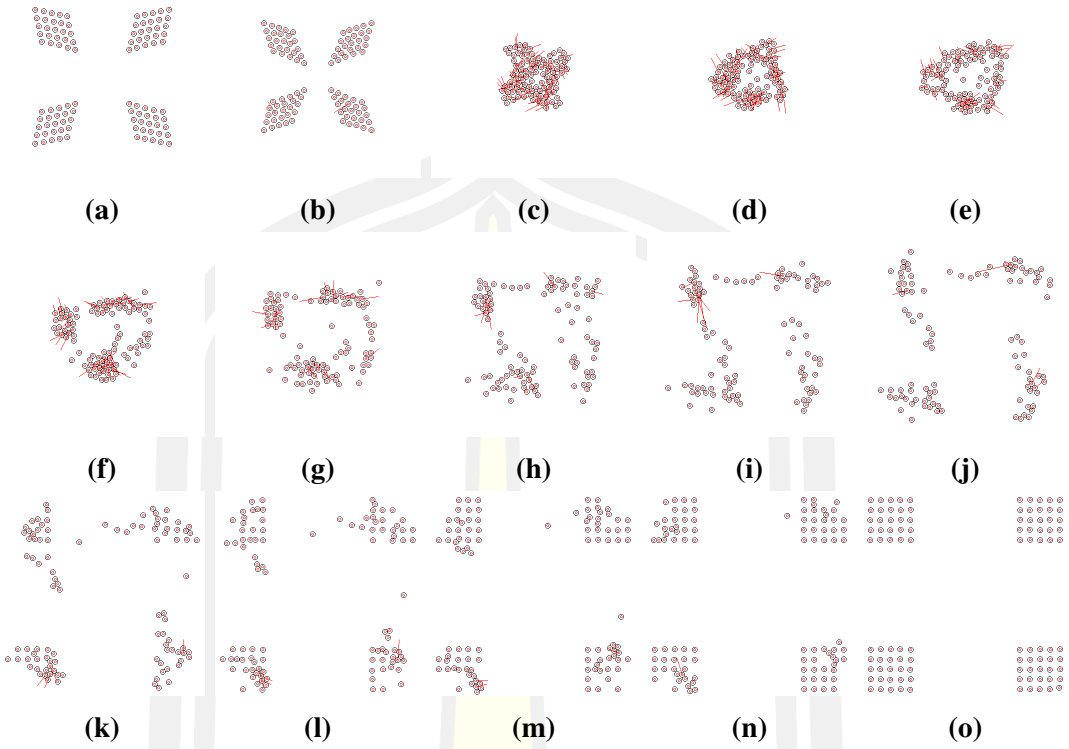
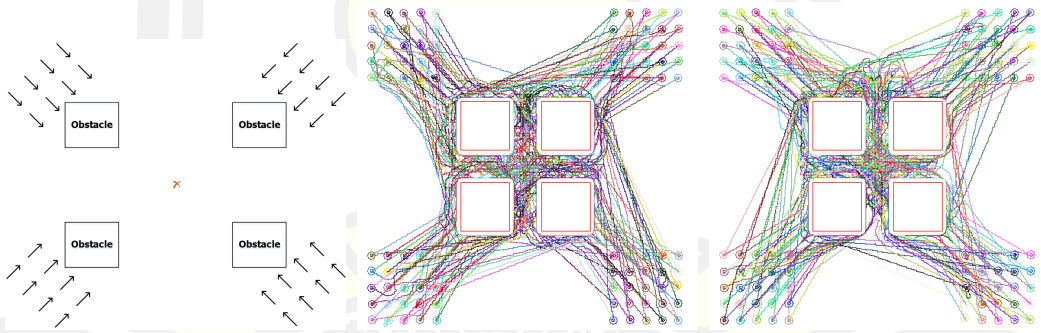


Figure 4.18. Four groups in opposite corners of the environment exchange positions in the *Narrow Passage* scenario (Our approach).



Block setting 2. **(a)** **(b)**

Figure 4.19. The resulting paths in the *Block* scenario using the our approach (a) and the RVO2 approach (b).

Figure 4.20 some snapshots of the simulation's visualization results. For setting *Block* with obstacle from simulation time at: 0, 1541, 3082, 4624, 6165, 7707, 9248, 10790, 12331, 13873, 15414, 16956, 18497, 20039, 21580 and 23122.

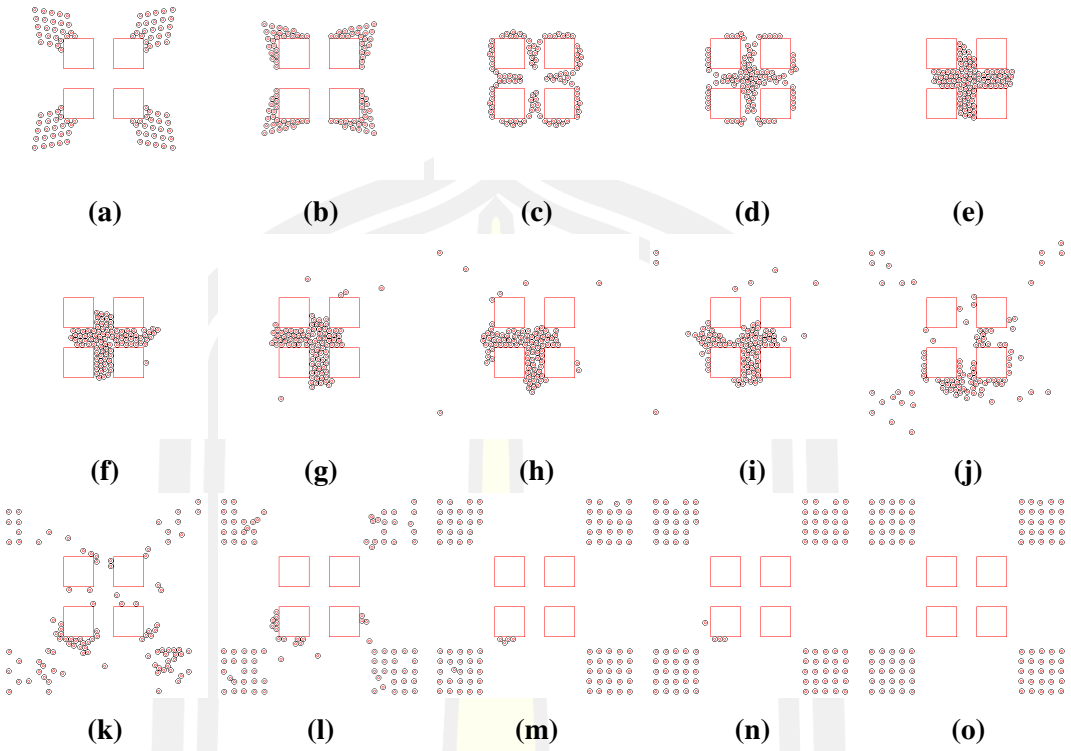


Figure 4.20. Four groups in opposite corners of the environment exchange positions in the *Narrow Passage* scenario (RVO2 with obstacles).

Figure 4.21 some snapshots of the simulation's visualization results. For setting *Block* with obstacle from simulation time at: 0, 111, 223, 335, 447, 559, 670, 782, 894, 1,006, 1,118, 1,229, 1,341, 1,453, 1,565 and 1,677.

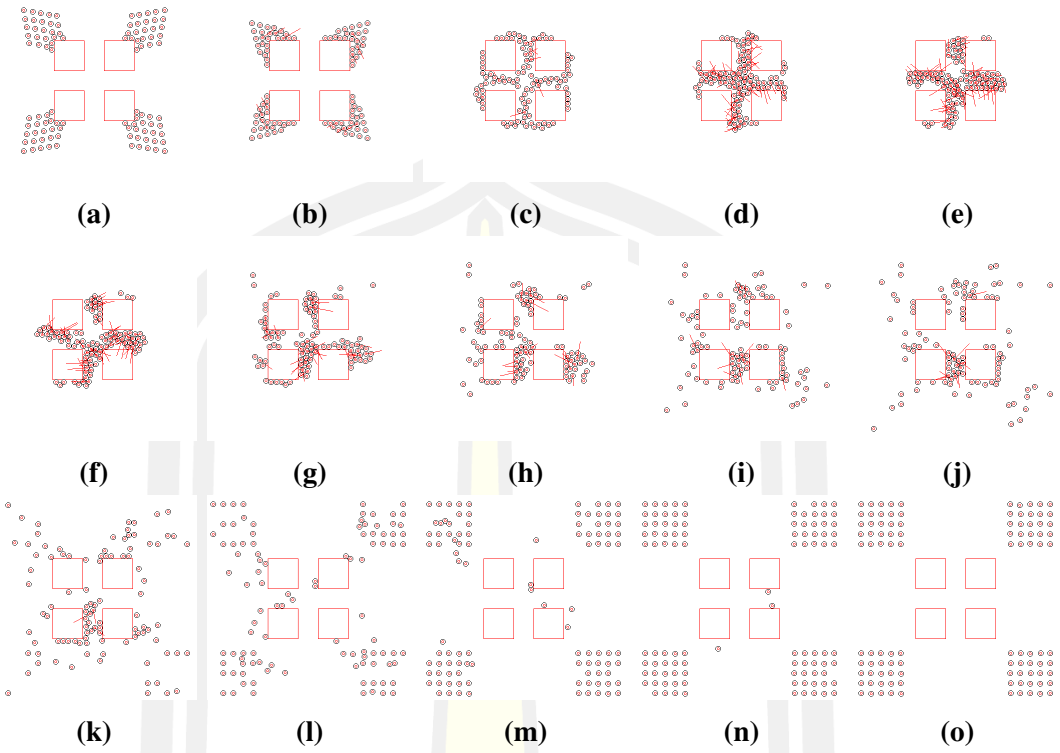


Figure 4.21. Four groups in opposite corners of the environment exchange positions in the *Narrow Passage* scenario (Our approach with obstacles).

Setting / Method	Block without obstacles	Block with obstacles
RVO2	7410	11542
Our approach	4380	9329

Table 4.4. Block settings scenario: execution time.

Setting / Method	Block without obstacles	Block with obstacles
RVO2	1449.5	23122.25
Our approach	816.25	1677.75

Table 4.5. Block settings scenario: simulation time.

4.4.4 Performance and limit of algorithm

To see how our approach performs in presence of narrow passages generated by static obstacles, we performed the experiment in the *Narrow Passage* benchmark. The agents from the different groups will meet inside the narrow passage with opposite goal directions. Even though the passage becomes very crowded (see Figure. 4.18, 4.21), eventually all agents reach their goals safely. We note that if the static obstacles

would form a U-shaped obstacle, the agents may get stuck in there when using our method. To overcome this, the method can be extended with a roadmap that governs the preferred velocities of the agent (instead of the velocity directed towards the goal).

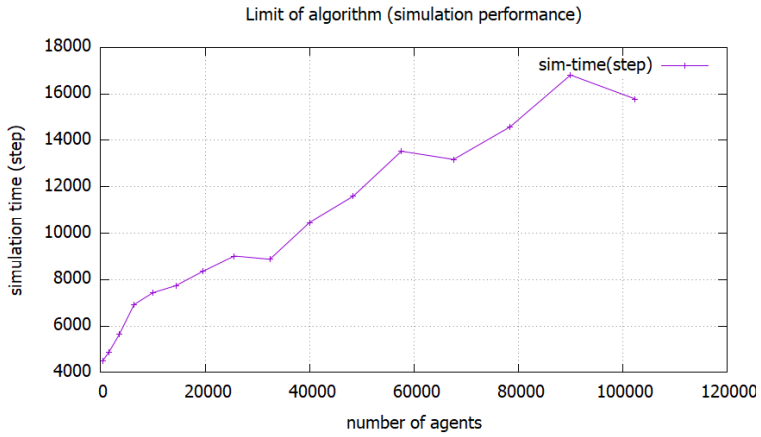


Figure 4.22. Memory used: block scenario without obstacle.

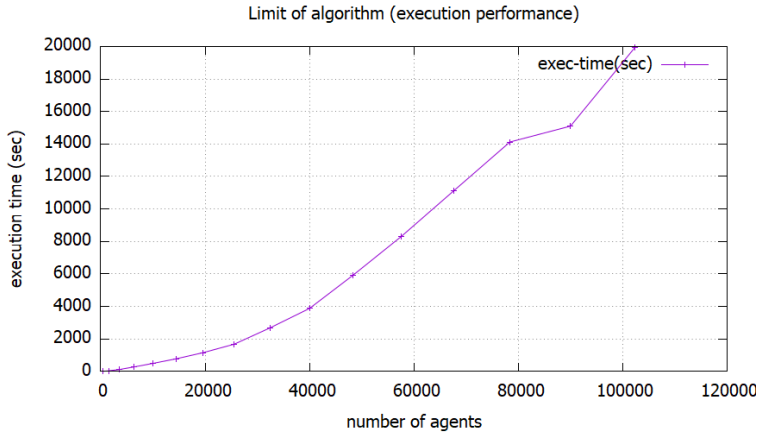


Figure 4.23. Memory used: block scenario without obstacle.

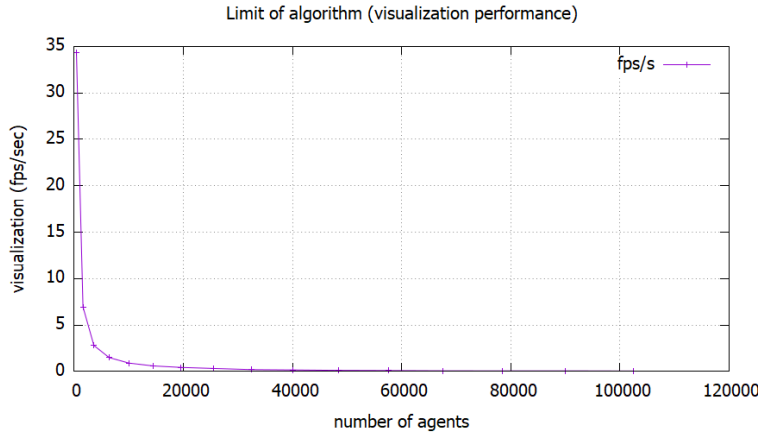


Figure 4.24. Memory used: block scenario without obstacle.

We measure limits of our algorithm by in term the average speed of agents within different setting. If the average speed is slower than the one hour. We can carry out our experiment up to limits shown below Table 4.6

Agent	Execution time(sec)	FPS
400	7.865	34.359822
1600	42.174	6.9504908
3600	121.266	2.7978576
6400	278.232	1.4934659
10000	495.469	0.9005407
14400	774.778	0.5994943
19600	1155.896	0.4339231
25600	1680.735	0.3218949
32400	2676.726	0.1989315
40000	**3877.317**	**0.1618606**
48400	5935.897	0.1171946
57600	8310.373	0.0976779
67600	11108.595	0.071176
78400	14121.182	0.0619336
90000	15099.398	0.0668023
102400	19959.877	0.0474412

Table 4.6. Limit of algorithm

There are this case for considering limits of our algorithm, namely interactive and non-interactive. For the former, the execution time must be less than the simulation time so that the visualization can be done in timely fashion. For the latter, the execution time can be greater than the simulation time as long as it is acceptable. Acceptability for this problem varies depending on urgency of result needs. Since we do not demand immediate results but we cannot wait for too long, we accept delays within matters of hours.

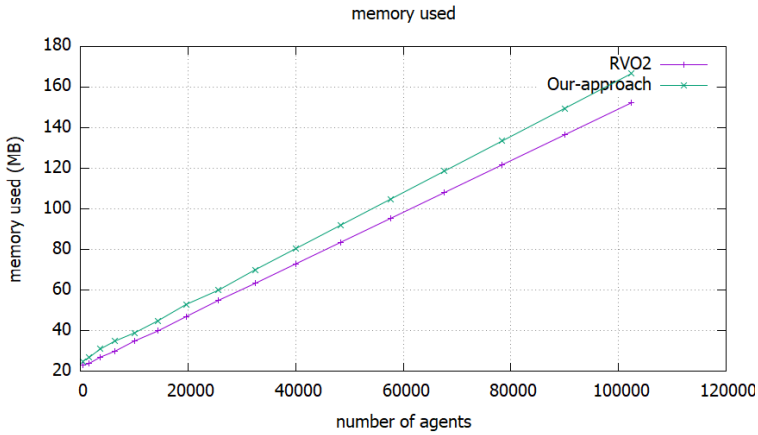


Figure 4.25. Memory used: block scenario without obstacle.

In many cases, the limit of algorithm is the hardware, i.e., the memory, particularity. In our experiments, we also check the limits on both visualization and execution. For visualization, the algorithm can 24 fps for 600 agents. The number of frame per second drop rapidly when the number of agents rises from 19,600 to 102,400 agents. The rendering is very low, 0.4 to 0.04 fps, when the number of agents is larger than or equal to 19,600. In term of memory usage, our algorithm consume slightly more memory than RVO2. Both algorithm consume more memory when the number of agent is increased almost linearly. In our experiment we have 1 GB of ram and can carry out simulation up to 720,000 agents.

4.5 Conclusion

In this chapter, we have improve the concept of Reciprocal Velocity Obstacles for safe and oscillation-free and dead-lock guarantee navigation among autonomous decision-making entities, as well as static and moving obstacles. It has been applied to multi-agent navigation and shown to compute smooth, natural paths at interactive rates for more than 1000 agents moving simultaneously in the same environment.

CHAPTER 5

PLANNING STRATEGY

To strategise plans for agents in crowd simulation, there can be many driving factors. One of the most widely used principles is trait. By definition in the online Cambridge dictionary, trait is "a particular characteristic that can produce a particular type of behaviour." And the meaning of behavior is "a particular way of acting." This means that we act according to our trait, or characteristic. In addition, surrounding environment can affect how we act as well, taking into account our trait. In crowd simulation trait has been used extensively as the underpinning mechanism for simulating how agents behave in crowd simulation.

In this chapter, we base our experiments on [18], where a number of traits are setup. Low-level and high level simulation behavior descriptors are associated. This allows agents to exhibit various degrees of aggressive, active and humble behaviors. In the following, we shall discuss briefly, for the sake of completeness, about trait background, settings in [18] and present our experiment results.

5.1 Personality Models and Trait Theory

Characterizing the spectrum of human personalities has long been a challenging task of psychologists. There have been so many different theories, focusing of various aspects, proposed over several decades or centuries. Among these, an outstanding theories is cross situational consistency, i.e. behavior aspects that are relatively consistent over time and across various situations. Methods to categorize and organize these variations have been proposed by psychologists have proposed among behavior's variety of sources. Guy et.al. [18] builds on Trait Theories of personality, a broad class of theories which categorizes people's behavior based on a small number of personality traits [42].

In [18], personality traits of individuals can be controlled by change parameters, resulting in various behaviors of crowd. The foundation of [18] is the Personality Trait Theory. It is said that a small number of underlying traits comprise complex

variations in behavior. A number of established models in Trait Theory are used to generate different behaviors for each individual, including Eysenck 3-Factor personality model [14], a biologically-based model of three independent factors of personality: Psychoticism, Extraversion, and Neuroticism. The model create a range of personalities.

It is defined [14] that "a personality trait is an habitual pattern of behavior, thought or emotion." It is belief that a small number of these traits are an individual's basic personality center. There are three primary traits that their combinations can describe variations in personality. The result appears and are seen by an individual's personality, depending on a strength or weakness score of each of these primary traits. The Eysenck 3-factor model [14] is among the most well established trait theories. The three factors categorizing personality are Psychoticism, Extraversion, and Neuroticism (commonly referred to as PEN). The combination of PEN factors identifies an individual's personality, dependinging on their strength and weakness. A person's aggression and egocentricity is designated by the Psychoticism factor. The social interest and higher levels of extroversion, associated with more active, assertive and daring behaviors, are designated by the Extraversion factor is a measure of Finally, the emotional instability, corresponding to the shyness and anxiety [15], is designated to the Neuroticism factor. It is interesting to note that the three PEN traits are associated to the levels of hormones, e.g. testosterone, serotonin and dopamine presenting in one's body.

To achieve reliable results, we also take into account factor analysis [6], to help determine "which small number of unobserved latent variables can describe the behavior of a large number of observed variables." Here, we also follow [18] to apply a similar factor analysis technique.

5.2 Behavior Perception User Study

In this section, we briefly review criteria used in our experiments. Since we base our work on [18], we shall present their settings in the following ,for the sake of completeness. The goal is to observe how the behaviors of agents can be affected by changes of parameters. Several commonly used low level parameters, including

preferred speed, effective radius (how far away an agent stays from other agents), maximum number of neighbors affecting the local behavior of an agent, maximum distance of neighbors affecting the agent, and planning horizon (how far ahead the agent plans).

Similarly to [18], a data-driven approach is adopted. A mapping between simulation parameters and perceived agent behaviors is derived. By doing these, there are at least two advantages: i) wider range of parameters provide more personality results, and ii) richer, more complex mappings can be possible. The setting also allows for multiple goals. There can be many adjectives for describing agents in the crowd, including "aggressive", "active" and "humble". There can be multiple psychological characteristics, such as the Eysenck's PEN model. The setting supports a factor analysis, allowing for extracting underlying personality in the crowd.

5.2.1 Method

In this section, we also re-explain the scenarios used in [18], which we follow. There are three scenarios. Simulations were created using Crowd Simulation Framework and Multi-agent simulation navigate (see Chapter 3 and Chapter4). Figure 5.1 shows a snapshot of three scenarios. Scenario 1), namely the Pass-Through scenario, where eight big-circle agents move through a cross-flow of 800 agents. Scenario 2), namely the Hallway scenario, in which 22 big-circle agents move through a hallway past 12 groups. There are 120 other agents in several small groups. Scenario 3), namely, the Narrowing Passage scenario, in which 70 big-circle agents walk alongside 6030 other agents towards a narrowing exit. Within all cases, the non-big-circle agents were given the default parameters from the simulation library, which mostly results in homogeneous behaviors of the agents in the simulation. The big-circle agents all share the same simulation parameters, that are randomly chosen for each question given to the participants.

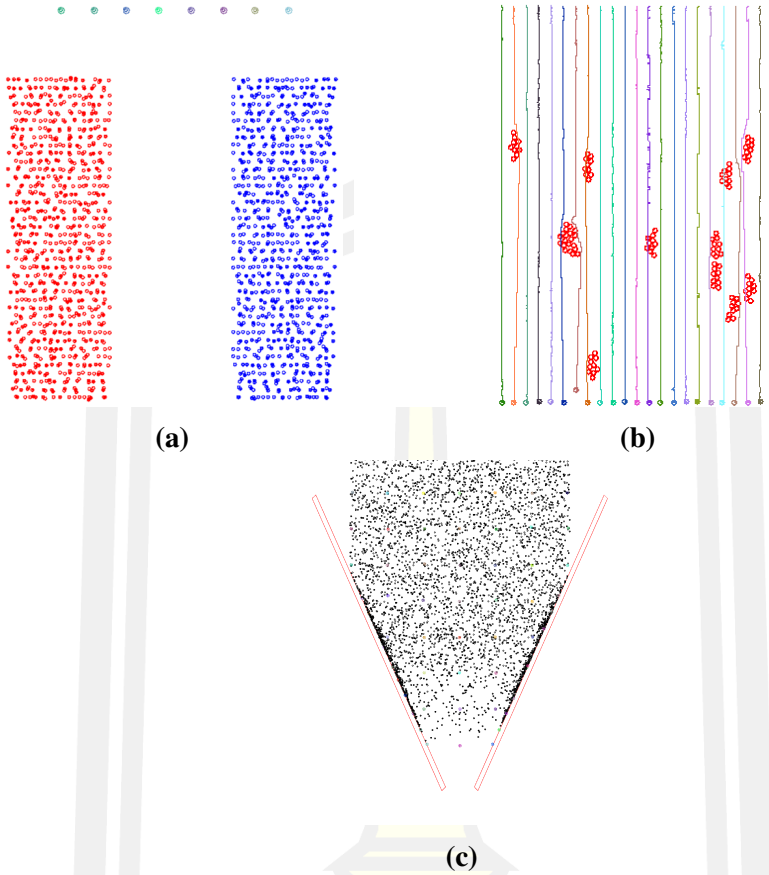


Figure 5.1. Three crowd simulation scenarios. (a) Eight big-circle agents move through crowd. (b) 22 big-circle individuals move through groups of still agents. (c) 70 big-circle individuals compete with others to exit through a narrowing passage.

Furthermore, [18] suggest the following parameter settings, which we follow: i) maximum distance to avoid neighbors, ii) maximum number of neighbors to avoid, iii) planning horizon, iv) agent radius, and v) preferred speed. Random values are assigned to these parameter, whose range is shown in Table 5.1.

Parameter	Min	Max	Unit
Max. neighbors dist.	3	30	m
Max. num. neighbors	1	100	(n/a)
Planning horizon	1	30	s
Agent radius	0.3	2.0	m
Preferred speed	1.2	2.2	m/s

Table 5.1. Range of simulation parameters.

5.3 Data Analysis

In this section, we also present data analysis used by [18], which we follow. The first dimension is *Mapping Perceived Behaviors*, which takes the following form:

$$\begin{pmatrix} Aggressive \\ Active \\ Humble \end{pmatrix} = A_{adj} \begin{pmatrix} \frac{1}{13.5}(Neighbor\ Dist - 15) \\ \frac{1}{49.5}(Max.\ Neighbors - 10) \\ \frac{1}{14.5}(Planning\ Horiz. - 30) \\ \frac{1}{0.85}(Radius - 0.8) \\ \frac{1}{0.5}(Pref.\ Speed - 1.4) \end{pmatrix}$$

Using a linear least-squares approach on the user study data we found the following 3-by-5 matrix A_{adj}:

$$A_{adj} = \begin{pmatrix} -0.02 & 0.32 & 0.13 & -0.41 & 1.02 \\ -0.06 & 0.04 & 0.04 & -0.16 & 1.07 \\ -0.04 & -0.08 & 0.02 & 0.58 & -0.88 \end{pmatrix}$$

It is suggested [18] that A_{adj} is a not a square matrix a mapping from high-level behaviors specified by the adjectives to simulation parameters by taking its pseudoinverse A^{+}_{adj} can be computed.

Another dimension is *Mapping Parameters for the PEN Model*. Guy et.al. [18] suggest that the adjectives from the user study can be mapped to the three PEN factors, corresponding to adjective to PEN factors found in Pervin [42], summarized in Table 5.2.

Trait	Adjectives
Psychoticism	Aggressive
Extraversion	Active
Neuroticism	Humble

Table 5.2. Excerpt from the mapping between adjectives and PEN factors given in [42] and used create A_{pen}.

Like the personality adjectives, we can determine a linear mapping for the PEN model, where:

$$\begin{pmatrix} Psychoticism \\ Extraversion \\ Neuroticism \end{pmatrix} = A_{pen} \begin{pmatrix} \frac{1}{13.5}(NeighborDist - 15) \\ \frac{1}{49.5}(Max.\ Neighbors - 10) \\ \frac{1}{14.5}(Planning\ Horiz. - 30) \\ \frac{1}{0.85}(Radius - 0.8) \\ \frac{1}{0.5}(Pref.\ Speed - 1.4) \end{pmatrix}$$

Based on a linear regression of the study data, A_{pen} was found to be

$$A_{pen} = \begin{pmatrix} 0.00 & 0.08 & 0.08 & -0.32 & 0.63 \\ -0.02 & 0.13 & 0.08 & -0.22 & 1.06 \\ 0.03 & -0.01 & -0.03 & 0.39 & -0.37 \end{pmatrix}$$

This mapping lets us predict expected PEN values from any given simulation parameters. The last dimension is *Factor Analysis*. The strong correlations between different PEN factors can be observed. It is suggested [18] that "Psychoticism and Extraversion show a strong positive correlation with each other and both are negatively correlated with Neuroticism." There exists a correlation between Aggressive and Assertive, which have a Pearson r-squared value of 0.45. The two Principal Component found through factor analysis are:

$$\begin{pmatrix} PC1 \\ PC2 \end{pmatrix} = \begin{pmatrix} 0 & -0.04 & 0.04 & 0.75 & 0.66 \\ 0.14 & 0.5 & 0.8 & 0.15 & -0.19 \end{pmatrix}$$

It is suggested that that PC1 primarily has the effect of increasing an agent's radius and speed. PC2 primarily makes agents plan further ahead and consider more agents for local avoidance. For these reasons, we suggestively refer to PC1 as "Extraversion" and PC2 as "Carefulness".

5.4 Experiments

Based on settings shown above, we present results achieved from our framework. The agents' preferred velocities range is [1.35,1.55] m/s. The parameters that change

behavior by only one “unit” (on the 1-6 scale described in Section 5.2.1). The parameters used are summarized in Table 5.3.

Trait	Neigh. Dist	Num. Neigh.	Plan. Horiz.	Radius	Speed
Psych.	15	40	38	0.4	1.55
Extrav.	15	23	32	0.4	1.55
Neuro.	15	9	29	1.6	1.25
Aggres.	15	20	31	0.6	1.55
Active	13	17	40	0.4	1.55
Humble	15	7	30	1.1	1.25

Table 5.3. Simulation parameters for various personality traits.

5.4.1 Pass-through Scene Results

Pass-through scenario generally allows agents to move freely from one side of the scene to the other side. In the following, we present snapshots of three scenes of running our framework and default RVO2.

As shown in Figure 5.2, agents in (a) move straight toward the other end. On the other hand, agents in (b)’s movement has more variations. This is because our framework allows more flexibility in agent’s movement. With our framework, agents can figure out more appropriately about path planning. While more space is available, our frame work allows for longer more straight movement. On the other hand, default RVO2 works at low level. When agents detect other agents in their path, agents try to figure out how to avoid collision. This results in diverged paths of some agents.

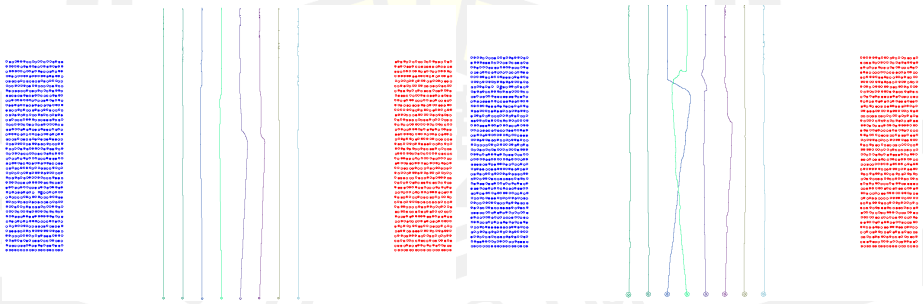
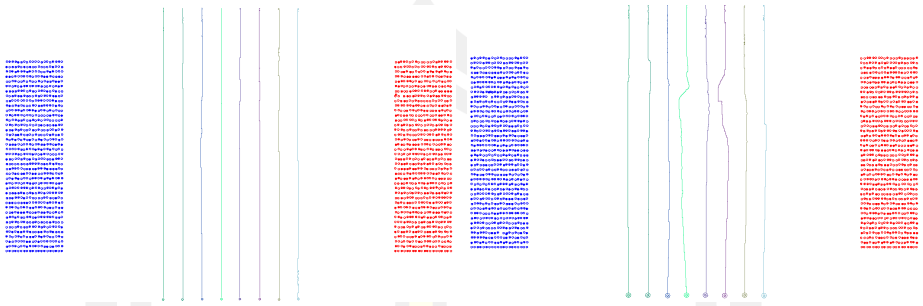


Figure 5.2. Pass-through Scenario-1. Paths of agents using our framework (a) and default RVO2.

Based on diversion of paths in Pass-through Scenario-1, we adjust some parameters in Pass-through Scenario-2 to verify if agents with RVO2 can move more straight

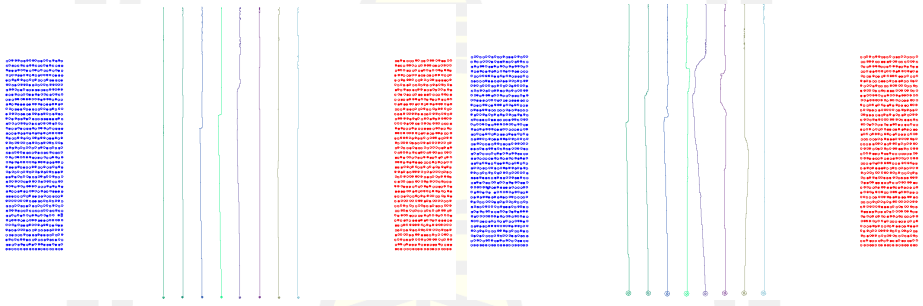
forward. We find that RVO2 agents can move more straight, as shown in Figure 5.3. As in scenario-1, our framework still help agents to move straight toward their targets.



(a) **(b)**

Figure 5.3. Pass-through Scenario-2. Paths of agents using our framework (a) and default RVO2.

We further adjust parameters in Pass-through Scenario-3. We found that RVO2 agents still move not straight, as shownl in Figure 5.4 (b). On the other hand, agents using our framework still move straight toward their targets.



(a) **(b)**

Figure 5.4. Pass-through Scenario-3. Paths of agents using our framework (a) and default RVO2.

Pass-through Scenes provide a lot of space for agents. The computational load is light, as shown in Table 5.4

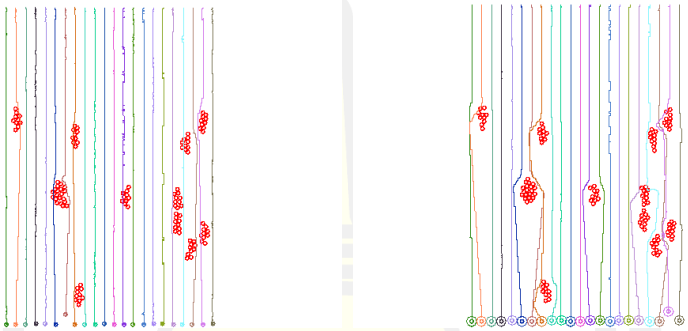
Behavior	Our approach AVG time(ms)	RVO2 AVG time(ms)
Aggressive	16,664	17,273
Active	16,443	17,712
Humble	16,224.25	16,966

Table 5.4. Execution time (AVG): average time of agents (big circles) to reach goal in three Pass-through Scenario.

5.4.2 Hallway Scenario Results

In Hallway Scenarios, agents move from one side of the scene, through static obstacles randomly distributed, to their destinations on the other end. Each obstacle is composed of red circles placed together as shown in Figures 5.5, 5.6 and 5.7. We ran our experiments many times and present here only three snapshots of path ways of agents' movement using both our framework (a) and RVO2 (b).

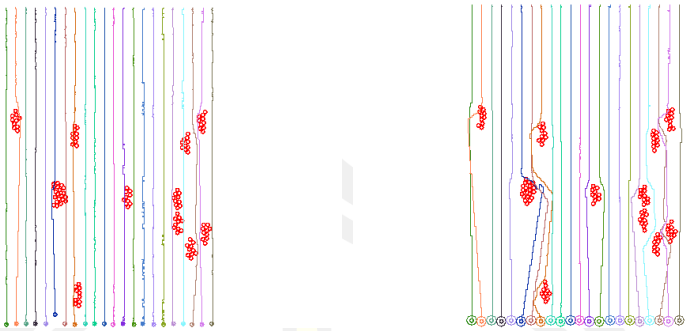
In the first scene, Hallway Scenario-1, agents are characterized by levels of psychoticism. Our framework allows agents to merely avoid obstacles and move straight to their destinations, as shown in Figure 5.5 (a). For agents using RVO2, their paths swerve away a little after avoiding obstacles, as shown in Figure 5.5 (b).



(a) **(b)**

Figure 5.5. Hallway Scenario-1. (a) agents with high levels of "Psychoticism". (b) Default RVO2 agents.

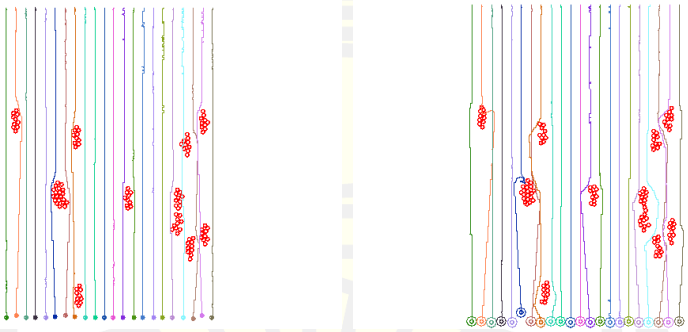
In the second scene, Hallway Scenario-2, agents are characterized by levels of extraversion. Agents with our framework still have similar movement pathways to Hallway Scenario-1. Agents to merely avoid obstacles and move straight to their destinations, as shown in Figure 5.6 (a). For agents using RVO2, their paths swerve away a little after avoiding obstacles, as shown in Figure 5.6 (b).



(a) (b)

Figure 5.6. Hallway Scenario-2. (a) agents with high levels of "Extraversion". (b) Default RVO2 agents.

In the third scene, Hallway Scenario-3, agents are characterized by levels of neuroticism. Agents with our framework still have similar movement pathways to Hallway Scenario-1 and Hallway Scenario-2. Agents to merely avoid obstacles and move straight to their destinations, as shown in Figure 5.7 (a). For agents using RVO2, their paths swerve away a little after avoiding obstacles, as shown in Figure 5.7 (b).



(a) (b)

Figure 5.7. Hallway Scenario. (a) agents with high levels of "Neuroticism". (b) Default RVO2 agents.

The execution times of simulation of agents with our framework and RVO2 for all three characteristics are shown in Table 5.5.

Trait	Our approach AVG time(ms)	RVO2 AVG time(ms)
Psychoticism	891	934
Extraversion	884	895
Neuroticism	838	955

Table 5.5. Execution time (AVG): average time of agents (big circles) to reach goal in three Pass-through Scenario.

5.4.3 Narrow Passage Scene Results

In Narrowing Passage scenes, agents are to move from a wide starting end through a narrowing passage to the exit on the other end. There are three type of agent settings, Aggressive, and neutral (or default), as shown in Figures 5.8, 5.9, and 5.10, respectively.

In the first setting, Aggressive, agents with aggressive characteristic are randomly placed in the crowd. As shown in Figure 5.8, most agents tightly gather along both sides of the narrowing passage. It appears that aggressive agents move pass the exit more quickly than other agents.

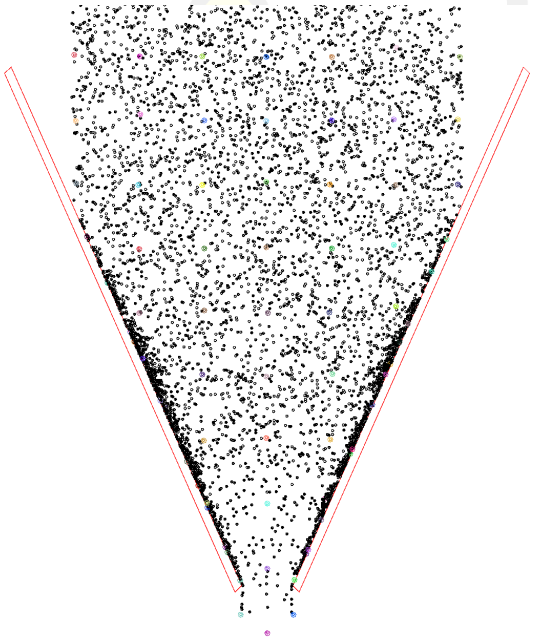


Figure 5.8. Narrowing Passage Scenario-1. A comparison between aggressive agents and other agents.

In the second setting, Humble, agents with humble characteristic are randomly placed in the crowd. As shown in Figure 5.8, most agents tightly gather along both sides of the narrowing passage. It appears that agressive agents move pass the exit more slowly than other agents.

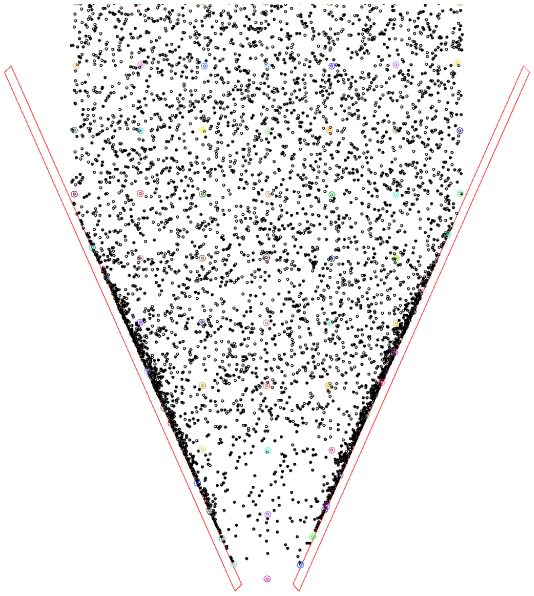


Figure 5.9. Narrowing Passage Scenario2. A comparison between humble agents and other agents.

In the third setting, Neutral, agents with neutral characteristic are randomly placed in the crowd. As shown in Figure 5.8, most agents tightly gather along both sides of the narrowing passage. It appears that agressive agents move pass the exit more quickly than other agents.

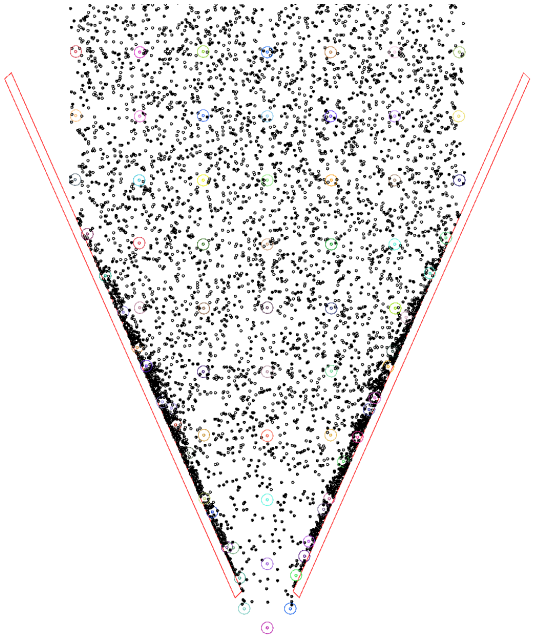


Figure 5.10. Narrowing Passage Scenario. A comparison between neutral agents and other agents.

Behaviors	AVG time(ms)
Aggressive	241,536
Active	243,831
Humble	244,864
Default RVO2	242,427

Table 5.6. Table of comparison four behaviors to reach goal (average time) in the **Narrowing Passage Scenario.**

A comparison of the rate at which the agents of various personalities passed through the exit is shown in Fig. 5.11. Humble agents were the slowest to pass through the exit, as they moved less quickly and packed in less tightly than the Aggressive and Active agents who made it out fastest.

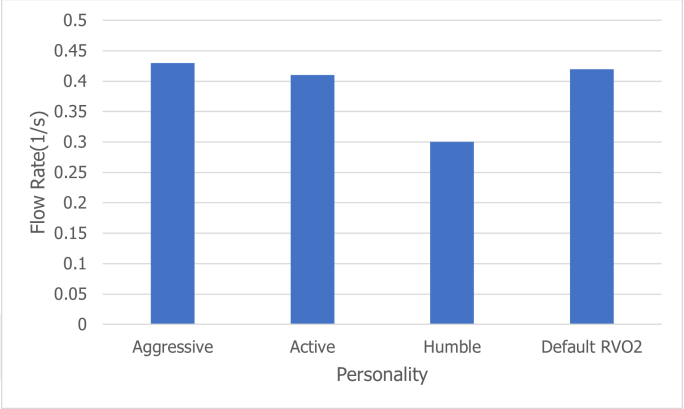


Figure 5.11. Exit Rate. Rate at which agents of various personalities exit in the Narrowing Passage scenario.

5.4.4 Heterogeneous Crowds

Using the mappings derived from experimental study, we can easily generate different simulations that map to different high-level personality specifications. We can use this capability to create interesting variations in complex, heterogeneous crowd simulations.

5.4.5 Timing Results

Because the behavior mapping can be computed as a pre-processing step, our method adds no overhead to the overall simulation run-time. Table 5.7 shows the execution time for simulating agents in several different scenarios, the timings were computed on a 3.6 GHz AMD Ryzen 5 2600x processor. In all cases, the simulation ran at interactive rates.

Scenario	Agents	Obstacles	Time (msec)
Hallway	6100	2	0.4
Narrowing Passage	142	0	1.9
Pass Through	808	0	1.4

Table 5.7. Performance timings per frame.

5.5 Conclusion

We have presented a perceptually driven formulation to model the personality of different agents in a crowd simulations. Our approach can successfully generate crowd simulations in which agents appear to depict specific, user-specified personalities, such

as aggressive, active, and humble. Furthermore, we have shown that our approach can successfully generate simulations where agents appear to have various levels of the established PEN personality traits.

CHAPTER 6

LARGE SPACE SIMULATION

In this chapter, we have combined Crowd Simulation Framework (Chapter 3), Agent Architecture (Chapter 4) and Planning Strategy (Chapter 5) to simulate large space urban, we briefly this follow section 6.1 Combined. We simulate large space urban in 2D space, we shown that in Section 6.2 Large Space Simulate.

6.1 Combined

In chapter 3 — Crowd Simulation Framework, we demonstrate how to create the scenario for simulate in virtual world, animate and control virtual agent in virtual world, vertex generate for define obstacles in the scenario, graph generate for define workable space in the scenario and used in an agent for path planning, path generator and find shortest path used to navigate an agent. In chapter 4 — Agent Architecture, we demonstrate how to integrated Belief Desire Intention (BDI) to an agent: Beleif represents the fact or information about status of the world around an agent. Desire represents the goal or status the agent wants to achieve. Intention represents the plan or sequence of actions for the agent to achieve the goal, taken into account the status of the surrounding world. Furthermore, we improve on original concept of Velocity Obstacles. In chapter 5 — Planning Strategy, we demonstrate how to implement and turning parameter an agent based-on Personality Models and Trait Theory for generate heterogeneous crowd behaviors.

6.2 Large Space Simulate

We set up a large space of 1920 x 1080 pixel. As shown in Figure 6.1. Since we simulate a large crown in urban area, we also create 4080 red large blocks, representing street blocks in real world. Their sizes are different. Their overall layout is still rectangular. The blue rectangle is the original area in which agents are located. The white narrow space between blocks representing streets in our virtual world. Agents are supposed to move on these streets only. They are not allowed to walk

through street blocks.

We have experimented our system with 1,000, 5,000, 10,000, 15,000, 20,000 and 25,000 agents. In each setting, individual agent will be generated to an origin and is assigned a location as its goal. Each agent will collect information about this map from the simulation engine and generate a graph, whose nodes are the junctions and intersections, and edges are streets connecting them. Each individual agent then applies our approach for their original paths. This path is the global plan, moving from one node along an edge for the the next node is a sub-plan. The agent completes a sub-plan by deploying RVO algorithm for avoiding collision with other agents or street blocks.

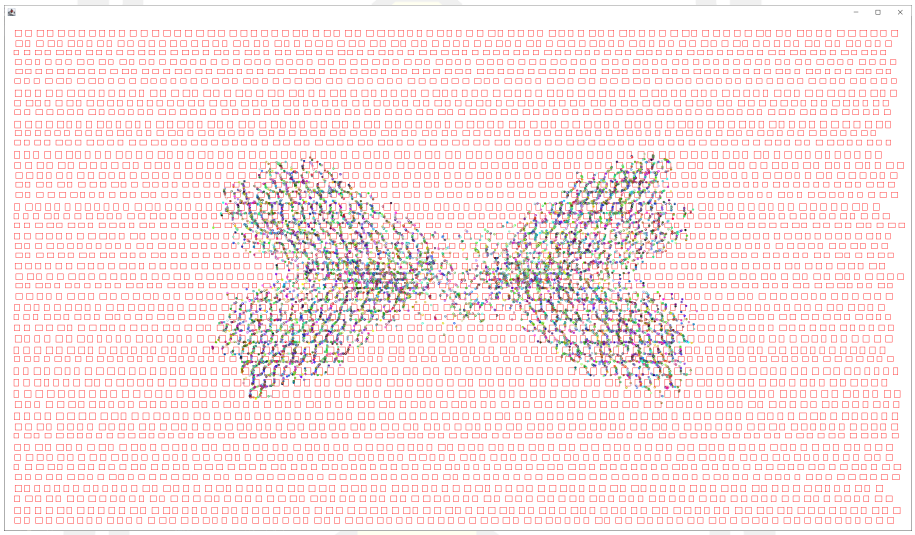


Figure 6.1. The scenario 4080 obstacle 4000 agents.

6.2.1 Behaviors and Planning

Based on our research in Chapter 5, we simulated the behavior of the agent using the PEN model with at least two adjectives for each trait and mapped the argument to the agents into various behavior models to observe behavior occurring in simulation scene. We use the block scenario as shown in Figure 6.2. We modeled them into settings and behaviors: Psychoticism (P), Extraversion (E), Neuroticism (N), Aggressive (A), Active (AT) and Humble (H). We simulate the number of agents as 1,000, 5,000, 10,000, 15,000, 20,000 and 25,000.

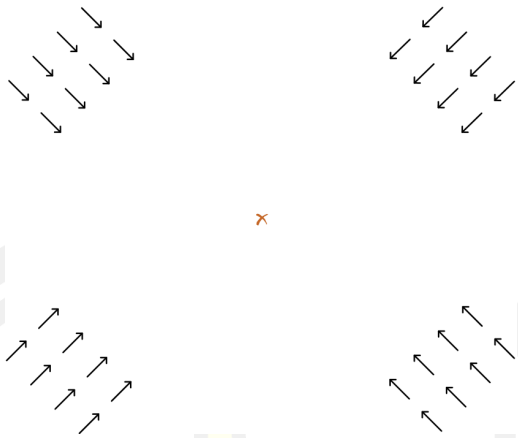
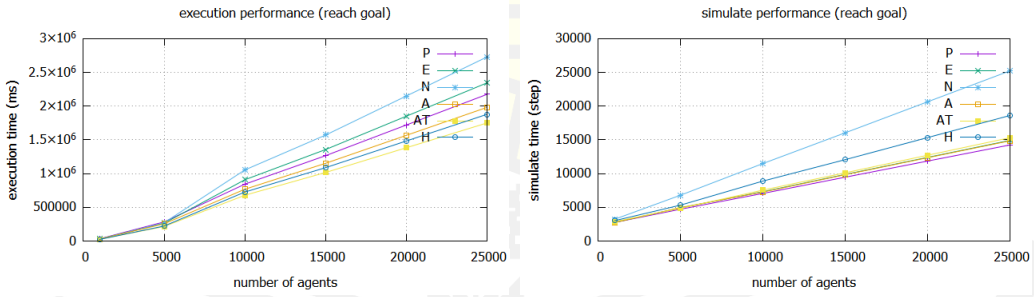


Figure 6.2. The Block Scenario.

The results of the simulation of execution time efficiency until the end of six behaviors as shown in Figure 6.3 (a). The Neuroticism behaviors were found to be most computationally efficient at execution time, Extraversion and Psychoticism behaviors, respectively. In terms of execution time the least were Active, Humble and Aggressive behaviors, respectively. In a real-world environment, most urban crowds were both Aggressive and Active, and both behaviors were found midway between the six behavioral groups. Therefore, the simulation results concluded that both behaviors were most suitable for simulating crowds in urban areas.



(a) **(b)**

Figure 6.3. Execution and Simulation time performance of six behaviors (Psychoticism, Extraversion, Neuroticism, Aggressive, Active and Humble) — (a) and (Psychoticism, Extraversion, Neuroticism, Aggressive, Active and Humble) — (b) setting in block scenario.

In terms of simulation time efficiency for six behaviors, shown Figure 6.3 (b), we found that Neuroticism behaviors took the most simulation time, Humble, and Active, respectively. In terms of simulation time the least were Psychoticism, Extraversion

and Active behaviors, respectively, analogous to the above Figure 6.3 (a).

6.2.2 Mixed Simulation Settings and Results (Behaviors and Planning)

We bring behavior Psychoticism, Active, Extraversion and Aggressive — Figure 6.4 (a) and Extraversion, Active, Neuroticism and Humble — Figure 6.4 (b) are simulated in the same scene, to the diversity of the crowd and the naturalness of the agents on the scene. We use the block scenario as shown in Figure 6.4. In this experiment, we simulated the number of agents 100, 1,600, 3,600, 6,400, 10,000, 14,400, 19,600 and 25,600 divided into groups based on Figure 6.4 (a) and (b).

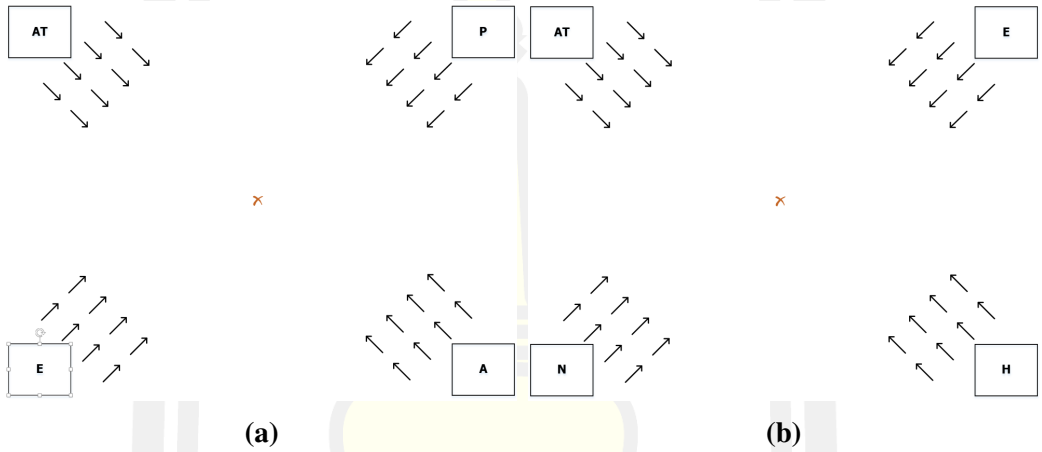
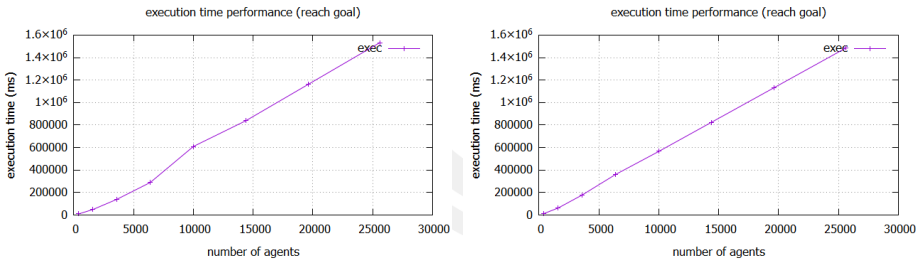


Figure 6.4. The Block Scenario: mixed four behaviors (P, AT, E and A) — (a) and (E, AT, N and H) — (b) setting in block scenario.

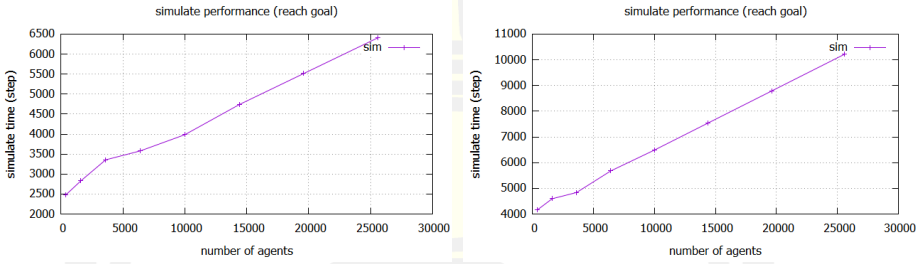
The simulation results of Figure 6.5 showed that group (a) agents spent about the same time moving to targets as those of group (b). However, the both groups of agents differed markedly when the number of agents was small as shown in group (a) compared to group (b). In terms of execution time from the above observations, it can be concluded that there is not much difference in terms of time. Therefore, if a heterogeneous group of behaviors is required to be simultaneous, multiple behaviors can be combined to simulate the event for the purpose of implementation.



(a) **(b)**

Figure 6.5. Execution time performance of four behaviors (P, AT, E and A) — (a) and (E, AT, N and H) — (b) setting in block scenario.

The simulation time results shown in Figure 6.6, agents in group (a) spent less time moving to the target than group (b). We conclude that if simulation versatility is desired, behavior group (a) should be preferred for evacuation, disaster and rush simulations and group (b) should be preferred for general behavior and urban simulations.



(a) **(b)**

Figure 6.6. Simulate time performance of four behaviors (P, AT, E and A) — (a) and (E, AT, N and H) — (b) setting in block scenario.

6.2.3 Simulation Snapshot

Figures 6.7, 6.8 and 6.9 show snapshot of progress of simulation from the beginning to the end. This scene contains 40,000 agents with 621 obstacles. Agent are divided into four groups equally starting from one corner towards the opposite corner. As shown in Figure 6.7 (a) to (d) agents can move directly towards their destinations. Once this large number of agents meet at the center of the scene, the crowded area expand to cover about a third area of the scene, as shown in Figure 6.7 (e) to (j) and Figure 6.8 (a) to (b). Agent slowly move through the crowded area and head towards their destination as shown in Figure 6.8 (c) to (j) and Figure 6.9 (a) to (c).

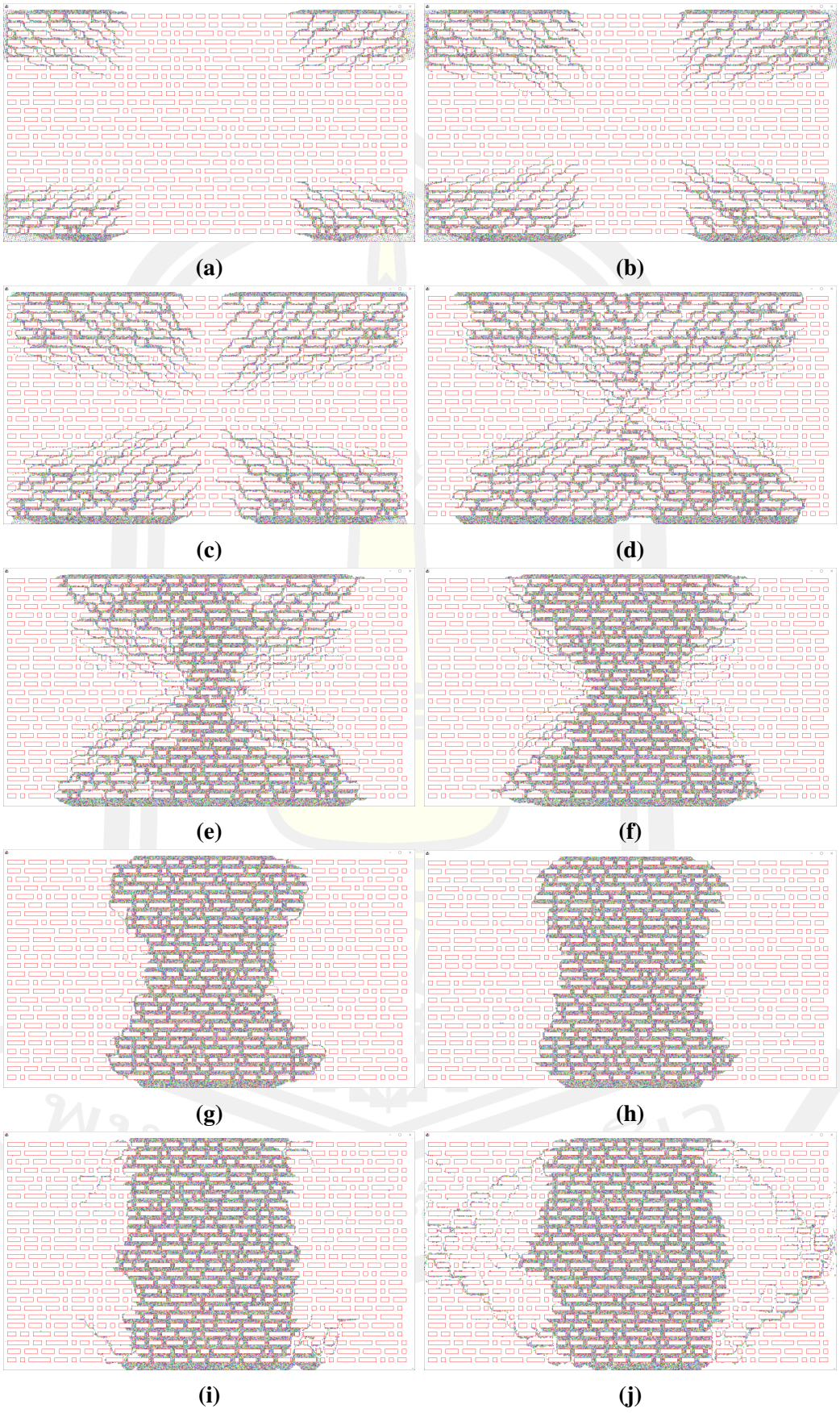


Figure 6.7. Snapshot: 40,000 agents with 621 obstacles(1).

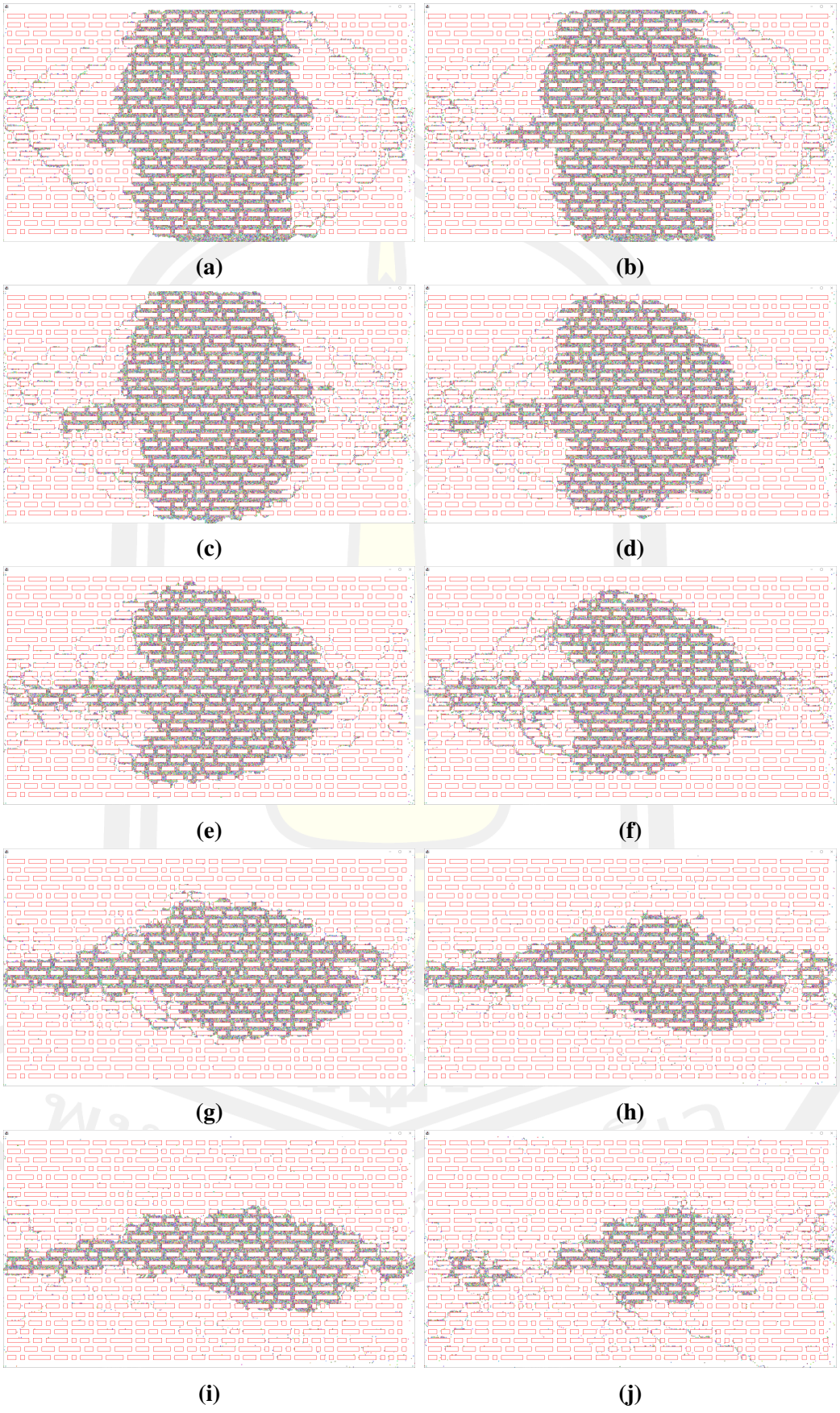


Figure 6.8. Snapshot: 40,000 agents with 621 obstacles(2).

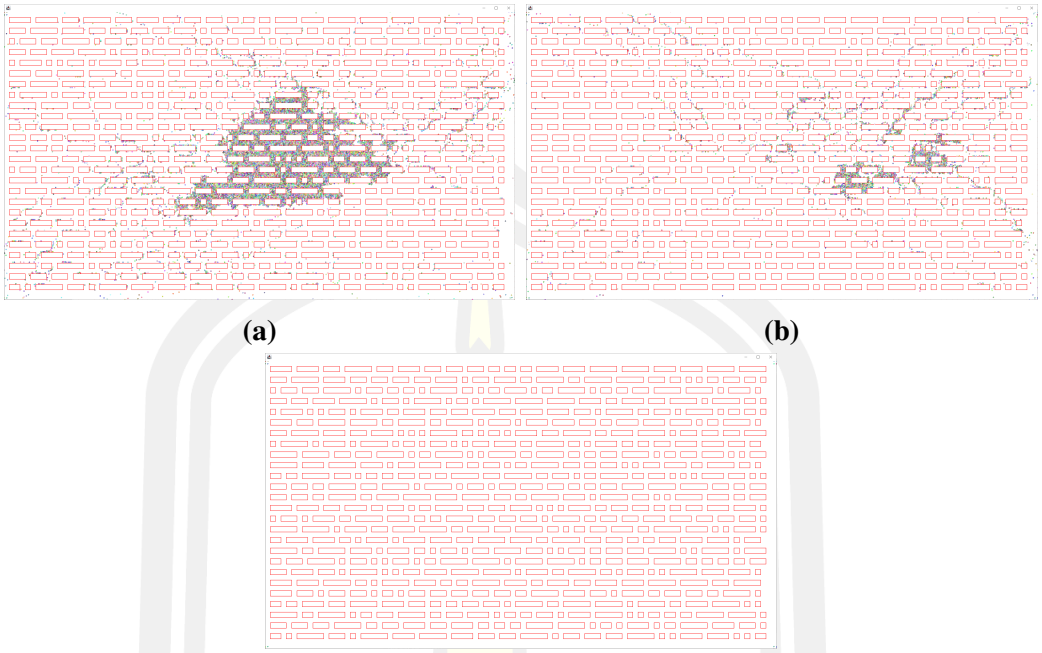


Figure 6.9. Snapshot: 40,000 agents with 621 obstacles(3).

6.3 Conclusion

Finally, we have successful combined Crowd Simulation Framework (Chapter 3), Agent Architecture (Chapter 4) and Planning Strategy (Chapter 5) to simulate large space urban. We can simulate large space urban in 2D space, we can handler our algorithm for individual agent and we mixed heterogeneous crowd behaviors in the scenario. Furthermore, we can simulate more than 100k agents in single scenario.

CHAPTER 7

CONCLUSIONS

In this thesis, we have proosed a hybrid framework for crowd simulation, We have conducted experients under many settings. In Chapter 3, we have invented a BDI architecture for agent-based simulation. We use RVO and Unity3d as our main underpinning mechanism for simulation. We carried out experiments up to 20,000 agents. We investigate three aspects of performance: simulation steps, execution time and visualization. The execution times appear alright. When the number of agent increases, the execution time increases accordingly. The visualization is not good when the number of agents is greater than 1,000. It becomes very slow and needs improvement. The simulation steps behave improperly. When the number of agents is greater than 5,000, the simulation steps do not change. We suspect the coordination between RVO and Unity3d the cause of this problem. In future work, we aim to simulate to more complex environment, such as virtual Bangkok (see Figure. 3.17). There are a number of improvement we can do, including optimize internal processes in our simulation for better performance.

In Chapter 4, we have improve the concept of Reciprocal Velocity Obstacles for safe and oscillation-free and dead-lock guarantee navigation among autonomous decision-making entities, as well as static and moving obstacles. It has been applied to multi-agent navigation and shown to compute smooth, natural paths at interactive rates for more than 1000 agents moving simultaneously in the same environment.

In Chapter 5, we have presented a perceptually driven formulation to model the personality of different agents in a crowd simulations. Our approach can successfully generate crowd simulations in which agents appear to depict specific, user-specified personalities, such as aggressive, active, and humble. Furthermore, we have shown that our approach can successfully generate simulations where agents appear to have various levels of the established PEN personality traits.

In Chapter 6, we have successful combined Crowd Simulation Framework (Chapter 3), Agent Architecture (Chapter 4) and Planning Strategy (Chapter 5) to

simulate large space urban. We can simulate large space urban in 2D space, we can handler our algorithm for individual agent and we mixed heterogeneous crowd behaviors in the scenario. Furthermore, we can simulate more than 100k agents in single scenario.

In conclusion, our framework provide satisfactory results. We are able to carry out simulations for quite a large number of agents. However, we can still improve our framework for larger number of agents and within more realistic scenes.

REFERENCES

REFERENCES

[1] Aniket Bera, Sujeong Kim, and Dinesh Manocha. Online parameter learning for data-driven crowd simulation and content generation. *Computers & Graphics*, 55:68–79, 2016.

[2] J. van den Berg, Lin Ming, and D. Manocha. Reciprocal velocity obstacles for real-time multi-agent navigation. In *2008 IEEE International Conference on Robotics and Automation*, pages 1928–1935.

[3] Eric Bouvier, Eyal Cohen, and Laurent Najman. From crowd simulation to airbag deployment: particle systems, a new paradigm of simulation. volume 6, page 14. SPIE, 1997.

[4] A. Braun, S. R. Musse, L. P. L. de Oliveira, and B. E. J. Bodmann. Modeling individual behaviors in crowd simulation. In *Proceedings 11th IEEE International Workshop on Program Comprehension*, pages 143–148, 2003.

[5] S. Bretschneider and A. Kimms. A basic mathematical model for evacuation problems in urban areas. *Transportation Research Part A: Policy and Practice*, 45(6):523–539, 2011.

[6] Heather E. P. Cattell. *The Sixteen Personality Factor (16PF) Questionnaire*, pages 187–215. Springer US, Boston, MA, 2001.

[7] Qianwen Chao, Zhigang Deng, and Xiaogang Jin. Vehicle–pedestrian interaction for mixed traffic simulation. *Computer Animation and Virtual Worlds*, 26(3-4):405–412, 2015.

[8] Xu Chen, Haiying Li, Jianrui Miao, Shixiong Jiang, and Xi Jiang. A multiagent-based model for pedestrian simulation in subway stations. *Simulation Modelling Practice and Theory*, 71:134–148, 2017.

[9] Yuan Cheng and Xiaoping Zheng. Emergence of cooperation during an emergency evacuation. *Applied Mathematics and Computation*, 320:485–494, 2018.

[10] Jérôme Comptdaer, Emmanuel Chiva, Stéphane Delorme, Henri Morlaye, Jérôme Volpoët, and Olivier Balet. *Multi-scale behavioral models for urban crisis training simulation*. 2007.

[11] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

[12] Simon Dobbyn, John Hamill, Keith O'Conor, and Carol O'Sullivan. Geopostors: a real-time geometry / impostor crowd rendering system, 2005.

[13] George Drettakis, Maria Roussou, Alex Reche, and Nicolas Tsingos. Design and evaluation of a real-world virtual environment for architecture and urban planning. *Presence: Teleoperators and Virtual Environments*, 16(3):318–332, 2007.

[14] M. Eysenck. *Personality and Individual Differences: A Natural Science Approach*. Perspectives on Individual Differences. Springer US, 1985.

[15] S. B. G. Eysenck and H. J. Eysenck. The place of impulsiveness in a dimensional system of personality description. *British Journal of Social and Clinical Psychology*, 16(1):57–68, 1977.

[16] Siome Goldenstein, Menelaos Karavelas, Dimitris Metaxas, Leonidas Guibas, Eric Aaron, and Ambarish Goswami. Scalable nonlinear dynamical systems for agent steering and crowd simulation. *Computers & Graphics*, 25(6):983–998, 2001.

[17] Stephen J. Guy, Jatin Chhugani, Sean Curtis, Pradeep Dubey, Ming Lin, and Dinesh Manocha. Pledestrians: a least-effort approach to crowd simulation, 2010.

[18] Stephen J. Guy, Sujeong Kim, Ming C. Lin, and Dinesh Manocha. Simulating heterogeneous crowd behaviors using personality trait theory. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '11, page 43–52, New York, NY, USA, 2011. Association for Computing Machinery.

[19] S. Gwynne, E. R. Galea, M. Owen, P. J. Lawrence, and L. Filippidis. A review of the methodologies used in the computer simulation of evacuation from the built environment. *Building and Environment*, 34(6):741–749, 1999.

[20] Yanbin Han and Hong Liu. Modified social force model based on information transmission toward crowd evacuation simulation. *Physica A: Statistical Mechanics and its Applications*, 469:499–509, 2017.

[21] Dirk Helbing, Illés Farkas, and Tamás Vicsek. Simulating dynamical features of escape panic. *Nature*, 407:487, 2000.

[22] Hao Jiang, Wenbin Xu, Tianlu Mao, Chunpeng Li, Shihong Xia, and Zhaoqi Wang. Continuum crowd simulation in complex environments. *Computers & Graphics*, 34(5):537–544, 2010.

[23] Jaekoo Joo, Namhun Kim, Richard A. Wysk, Ling Rothrock, Young-Jun Son, Yeong-gwang Oh, and Seungho Lee. Agent-based simulation of affordance-based human behaviors in emergency evacuation. *Simulation Modelling Practice and Theory*, 32:99–115, 2013.

[24] Marcelo Kallmann and Daniel Thalmann. Modeling objects for interaction tasks. Computer Animation and Simulation ’98, pages 73–86. Springer Vienna, 1999.

[25] Ioannis Karamouzas, Roland Geraerts, and Mark Overmars. Indicative routes for path planning and crowd simulation, 2009.

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

[27] Kang Hoon Lee, Myung Geol Choi, Qyoun Hong, and Jehee Lee. Group behavior from video: a data-driven approach to crowd simulation, 2007.

[28] Yan Li, Hong Liu, Guang-peng Liu, Liang Li, Philip Moore, and Bin Hu. A grouping method based on grid density and relationship for crowd evacuation simulation. *Physica A: Statistical Mechanics and its Applications*, 473:319–336, 2017.

[29] M. Lozano, P. Morillo, J. M. Orduña, V. Cavero, and G. Vigueras. A new system architecture for crowd simulation. *Journal of Network and Computer Applications*, 32(2):474–482, 2009.

[30] Gregor Lämmel, Dominik Grether, and Kai Nagel. The representation and implementation of time-dependent inundation in large-scale microscopic evacuation simulations. *Transportation Research Part C: Emerging Technologies*, 18(1):84–98, 2010.

[31] Tianlu Mao, Hao Jiang, Jian Li, Yanfeng Zhang, Shihong Xia, and Zhaoqi Wang. Parallelizing continuum crowds, 2010.

[32] J. Maïm, B. Yersin, and D. Thalmann. Unique character instances for crowds. *IEEE Computer Graphics and Applications*, 29(6):82–90, 2009.

[33] Azhar Mohd Ibrahim, Ibrahim Venkat, and Philippe De Wilde. Uncertainty in a spatial evacuation model. *Physica A: Statistical Mechanics and its Applications*, 479:485–497, 2017.

[34] Albert Mukovskiy, Jean-Jacques E. Slotine, and Martin A. Giese. Dynamically stable control of articulated crowds. *Journal of Computational Science*, 4(4):304–310, 2013.

[35] S. R. Musse and D. Thalmann. A model of human crowd behavior : Group inter-relationship and collision detection analysis. Computer Animation and Simulation ’97, pages 39–51. Springer Vienna, 1997.

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

[37] Rahul Narain, Abhinav Golas, Sean Curtis, and Ming C. Lin. Aggregate dynamics for dense crowd simulation. *ACM Trans. Graph.*, 28(5):1–8, 2009.

[38] Jan Ondřej, Julien Pettré, Anne-Hélène Olivier, and Stéphane Donikian. A synthetic-vision based steering approach for crowd simulation. *ACM Trans. Graph.*, 29(4):1–9, 2010.

[39] Oğuzcan Oğuz, Ateş Akaydın, Türker Yılmaz, and Uğur Güdükbay. Emergency crowd simulation for outdoor environments. *Computers & Graphics*, 34(2):136–144, 2010.

[40] N. Pelechano, J. M. Allbeck, and N. I. Badler. Controlling individual agents in high-density crowd simulation, 2007.

[41] Nuria Pelechano, Kevin D. O'Brien, Barry Silverman, and Norman L. Badler. Crowd simulation incorporating agent psychological models, roles and communication. 2005.

[42] L. A. Pervin. Science of personality. *British Medical Journal*, 2:247 – 247, 1942.

[43] D. Pianini, M. Viroli, F. Zambonelli, and A. Ferscha. Hpc from a self-organisation perspective: The case of crowd steering at the urban scale. In *2014 International Conference on High Performance Computing & Simulation (HPCS)*, pages 460–467, 2014.

[44] Fasheng Qiu and Xiaolin Hu. Modeling group structures in pedestrian crowd simulation. *Simulation Modelling Practice and Theory*, 18(2):190–205, 2010.

[45] Anand S. Rao and Michael P. Georgeff. *BDI Agents: From Theory to Practice*. ICMAS, 1995.

[46] Craig W. Reynolds. Flocks, herds and schools: A distributed behavioral model. *SIGGRAPH Comput. Graph.*, 21(4):25–34, 1987.

[47] Craig W. Reynolds. Steering behaviors for autonomous characters. 1999.

[48] Jason Sewall, David Wilkie, and Ming C. Lin. Interactive hybrid simulation of large-scale traffic. *ACM Trans. Graph.*, 30(6):1–12, 2011.

[49] Ameya Shendarkar, Karthik Vasudevan, Seungho Lee, and Young-Jun Son. Crowd simulation for emergency response using bdi agents based on immersive virtual reality. *Simulation Modelling Practice and Theory*, 16(9):1415–1429, 2008.

[50] Jamie Snape, Stephen J. Guy, Jur van den Berg, and Dinesh Manocha. *Smooth Coordination and Navigation for Multiple Differential-Drive Robots*, pages 601–613. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[51] Jamie Snape and Dinesh Manocha. Navigating multiple simple-airplanes in 3d workspace. In *2010 IEEE International Conference on Robotics and Automation*, pages 3974–3980, 2010.

[52] Franco Tecchia, Celine Loscos, and Yiorgos Chrysanthou. Visualizing crowds in real-time. *Computer Graphics Forum*, 21(4):753–765, 2002.

[53] Demetri Terzopoulos, Xiaoyuan Tu, and Radek Grzeszczuk. Artificial fishes: Autonomous locomotion, perception, behavior, and learning in a simulated physical world. *Artificial Life*, 1(4):327–351, 1994.

[54] D. Thalmann, H. Grillon, J. Maim, and B. Yersin. Challenges in crowd simulation. In *2009 International Conference on CyberWorlds*, pages 1–12.

[55] Daniel Thalmann. *Crowd Simulation*, pages 1–8. Springer International Publishing, Cham, 2016.

[56] Pablo Cristian Tissera, Alicia Castro, A. Marcela Printista, and Emilio Luque. Evacuation simulation supporting high level behaviour-based agents. *Procedia Computer Science*, 18:1495–1504, 2013.

[57] Pablo Cristian Tissera, A. Marcela Printista, and Emilio Luque. A hybrid simulation model to test behaviour designs in an emergency evacuation. *Procedia Computer Science*, 9:266–275, 2012.

[58] Branislav Ulicny, Pablo de Heras Ciechomski, and Daniel Thalmann. Crowdbrush: interactive authoring of real-time crowd scenes, 2004.

[59] Neal Wagner and Vikas Agrawal. An agent-based simulation system for concert venue crowd evacuation modeling in the presence of a fire disaster. *Expert Systems with Applications*, 41(6):2807–2815, 2014.

[60] Haizhong Wang, Alireza Mostafizi, Lori A. Cramer, Dan Cox, and Hyoungsu Park. An agent-based model of a multimodal near-field tsunami evacuation: Decision-making and life safety. *Transportation Research Part C: Emerging Technologies*, 64:86–100, 2016.

[61] M. L. L. Wijerathne, L. A. Melgar, M. Hori, T. Ichimura, and S. Tanaka. Hpc enhanced large urban area evacuation simulations with vision based autonomously navigating multi agents. *Procedia Computer Science*, 18:1515–1524, 2013.

[62] David Wolinski, Stephen J. Guy, Anne-Hélène Olivier, Ming C. Lin, Dinesh Manocha, and Julien Pettré. Optimization-based pedestrian model calibration for evaluation. *Transportation Research Procedia*, 2:228–236, 2014.

[63] Hao Zhang, Hong Liu, Xin Qin, and Baoxi Liu. Modified two-layer social force model for emergency earthquake evacuation. *Physica A: Statistical Mechanics and its Applications*, 492:1107–1119, 2018.

[64] Jinghui Zhong and Wentong Cai. Differential evolution with sensitivity analysis and the powell's method for crowd model calibration. *Journal of Computational Science*, 9:26–32, 2015.

[65] Jinghui Zhong, Nan Hu, Wentong Cai, Michael Lees, and Linbo Luo. Density-based evolutionary framework for crowd model calibration. *Journal of Computational Science*, 6:11–22, 2015.

BIOGRAPHY

BIOGRAPHY

Name	Panich Sudkhot
Date of Birth	February 08, 1989
Place of Birth	Ubon Ratchathani Province, Thailand
Address	33/5 Ban Dong Mafai, Nong Saeng Yai, Khong Chiam, Ubon Ratchathani 34220 Thailand
Institution attended	
2008	Phibun Mangsahan School, Ubon Ratchathani, Thailand
2012	Bachelor of Science in Information Comunication Technology, Faculty of Informatics, Mahasarakham University, Thailand
2014	Master of Science in Information Technology, Faculty of Informatics, Mahasarakham University, Thailand
2020	Doctor of Philosophy in Computer Science, Faculty of Informatics, Mahasarakham University, Thailand
Position and Work Place	
	Managing Director at Zyntelligent Co,. Ltd: 398/20 Kham Riang, Kantarawichai, Maha Sarakham 44150 Thailand
Contact address	
	33/5 Ban Dong Mafai, Nong Saeng Yai, Khong Chiam, Ubon Ratchathani 34220 Thailand
Research grants & awards	
	Research Potential Scholarships, Faculty of Informatics, Mahasarakham University
Research output	

2018 **Panich Sudkhot** and Chattrakul Sombattheera, "A Crowd Simulation in Large Space Urban," 2018 International Conference on Information Technology (InCIT), Khon Kaen, 2018, pp. 1-8. doi: 10.23919/INCIT.2018.8584878

2016 **Panich Sudkhot**, Savanid Vatanasakdakul and Chattrakul Sombattheera, "Task allocation with Optimal Coalition Structure concept", Information Technology Journal, Link: http://www2.it.kmutnb.ac.th/journal/pdf/vol23/ch07.pdf, ฉบับที่ 23(ม.ค. 59 – มิ.ย. 59) 2559. pp. 44-53. (TCI Journal)

2014 **Panich Sudkhot**, Benjawan Intara and Chattrakul Sombattheera, "TASK ALLOCATION IN MULTI-AGENT SYSTEM WITH OPTIMAL COALITION STRUCTURE CONCEPT", In Proceedings of Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology - Conference on Application Research and Development (ECTI - CARD), Chiang Mai, Thailand, 21-23 May 2014. pp. D282-D285.

Benjawan Intara, **Panich Sudkhot** and Chattrakul Sombattheera, "Payoff Distribution of ASEAN Power Grid by Shapley Value", In Proceedings of Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology - Conference on Application Research and Development (ECTI - CARD), Chiang Mai, Thailand, 21-23 May 2014. pp. D103-D106. (in Thai)

Benjawan Intara, **Panich Sudkhot** and Chattrakul Sombattheera, "Payoff Distribution of ASEAN Power Grid in Optimal Coalition Structure by Shapley Value", In The 10th National Conference on Computing and Information Technology, Angsana Laguna Phuket, Thailand, 8-9 May 2014, pp. 667 - 682.

2013 Benjawan Intara, **Panich Sudkhot** and Chattrakul Sombattheera, "A Mobile-based Heart Rate Information Management System", Information Technology Journal, Link: http://www2.it.kmutnb.ac.th/journal/pdf/vol17/ch03.pdf, ฉบับที่ 17(ม.ค. 56 – มิ.ย. 56) 2556. pp. 14-23. (TCI Journal)

Panich Sudkhot, Benjawan Intara and Chattrakul Sombattheera, "Task allocation with Optimal Coalition Structure concept", In The 9th National Conference on Computing and Information Technology, King Mongkut's University of Technology North Bangkok, Bangkok, Thailand, 9-10 May 2013, pp. 392-397.

Benjawan Intara, **Panich Sudkhot** and Chattrakul Sombattheera, "Payoff Distribution of Agents in Optimal Coalition Structure under Non-Superadditive by Shapley Value", In The 9th National Conference on Computing and Information Technology, King Mongkut's University of Technology North Bangkok, Bangkok, Thailand, 9-10 May 2013, pp. 342 - 347.

Panich Sudkhot, Benjawan Intara and Chattrakul Sombattheera, "Optimal Coalition Structure for Simulating Optimization Process", In Proceedings of Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology - Conference on Application Research and Development (ECTI - CARD), Institute of Engineering, SUT, Thailand, 8-10 May 2013. pp. 381-386.

2012 **Panich Sudkhot**, Benjawan Intara and Chattrakul Sombattheera, "A Wireless-Based Pulse Measuring System on Mobile", In Proceedings of Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology - Conference on Application Research and Development (ECTI - CARD), RMUTT, Thailand, 21-22 June 2012. pp. 250-255.