Fall 12-14-2010

A Framework for Group Modeling in Agent-Based Pedestrian Crowd Simulations

Fasheng Qiu
Georgia State University

A FRAMEWORK FOR GROUP MODELING IN AGENT-BASED PEDESTRIAN CROWD SIMULATIONS

by

FASHENG QIU

Under the Direction of Dr. Xiaolin Hu

ABSTRACT

Pedestrian crowd simulation explores crowd behaviors in virtual environments. It is extensively studied in many areas, such as safety and civil engineering, transportation, social science, entertainment industry and so on. As a common phenomenon in pedestrian crowds, grouping can play important roles in crowd behaviors. To achieve more realistic simulations, it is important to support group modeling in crowd behaviors. Nevertheless, group modeling is still an open and challenging problem. The influence of groups on the dynamics of crowd movement has not been incorporated into most existing crowd models because of the complexity nature of social groups.

This research develops a framework for group modeling in agent-based pedestrian crowd simulations. The framework includes multiple layers that support a systematic approach for modeling social groups in pedestrian crowd simulations. These layers include a simulation engine layer that provides efficient simulation engines to simulate the crowd model; a behavior-based agent modeling layers that supports developing

agent models using the developed *BehaviorSim* simulation software; a group modeling layer that provides a well-defined way to model inter-group relationships and intra-group connections among pedestrian agents in a crowd; and finally a context modeling layer that allows users to incorporate various social and psychological models into the study of social groups in pedestrian crowd. Each layer utilizes the layer below it to fulfill its functionality, and together these layers provide an integrated framework for supporting group modeling in pedestrian crowd simulations. To our knowledge this work is the first one to focus on a systematic group modeling approach for pedestrian crowd simulations. This systematic modeling approach allows users to create social group simulation models in a well-defined way for studying the effect of social and psychological factors on crowd's grouping behavior. To demonstrate the capability of the group modeling framework, we developed an application of dynamic grouping for pedestrian crowd simulations.

INDEX WORDS: Group modeling, Dynamic grouping, Social groups, Discrete event based simulation, Performance improvement, Behavior-based pedestrian agent, Crowd simulation.

A FRAMEWORK FOR GROUP MODELING IN AGENT-BASED PEDESTRIAN CROWD SIMULATIONS

by

FASHENG QIU

A Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

in the College of Arts and Sciences

Georgia State University

2010

A FRAMEWORK FOR GROUP MODELING IN AGENT-BASED PEDESTRIAN CROWD SIMULATIONS

by

FASHENG QIU

Committee Chair: Dr. Xiaolin Hu

Committee: Dr. Rajshekhar Sunderraman

Dr. Saeid Belkasim

Dr. Jiawei Liu

Electronic Version Approved:

Office of Graduate Studies

College of Arts and Sciences

Georgia State University

December 2010

To my family.

ACKNOWLEDGEMENTS

First, I want to thank my advisor, Dr. Xiaolin Hu, for his generous advice and support for my Ph.D. study at Georgia State University. His board and deep insight in interdisciplinary area inspired and guided me in this research. I really appreciate the research methodology and writing skills learned from this great professor.

Next, I would like to thank my committee members, Dr. Rajshekhar Sunderraman, Dr. Saeid Belkasim, and Dr.Jiawei Liu for their valuable suggestions in improving the quality of this work.

I would also like to thank my colleagues from Dr. Hu's research group: Yi Sun, Feng Gu, and Song Guo for the useful discussion of research ideas.

Last, but not least, I would like to extend my gratitude to my parents for their strong, persistent encouragement, understanding and their continuing support as I have pursued my educational goals. Special gratitude goes to my wife for the encouragement for my research.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

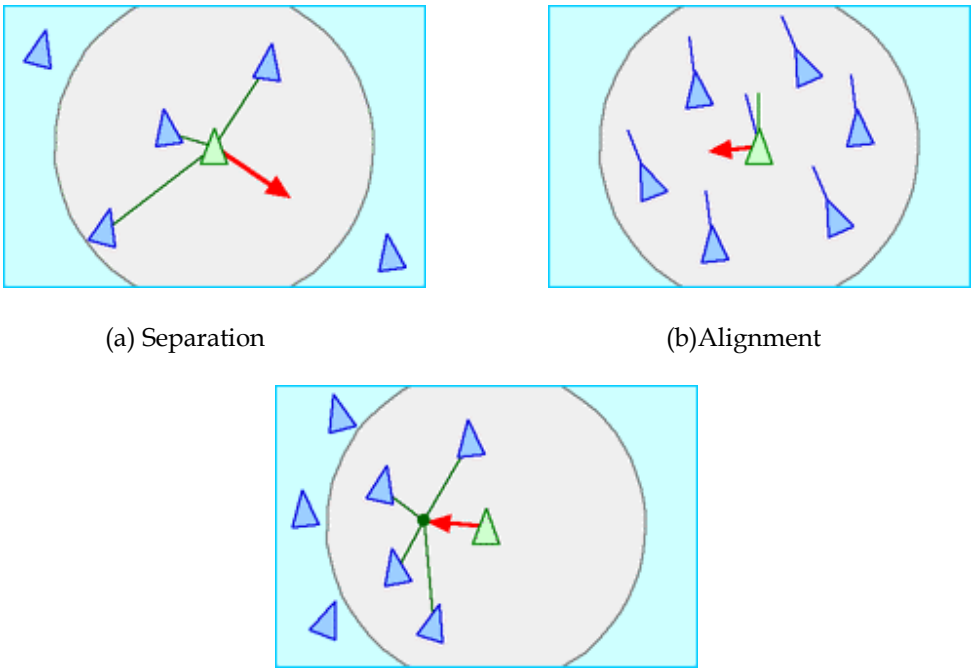
CHAPTER 1
INTRODUCTION

1.1 Agent-based Pedestrian Crowd Simulation

Pedestrian crowd simulation has been studied over two decades. It is particularly useful for people to study crowd behaviors in emergent situations where experiments with humans are impossible to be fulfilled. Because of this, pedestrian crowd simulation is studied in a variety of directions such as civil and safety engineering, urban and city planning, building design, and so on. In civil and safety engineering, people study the flow characteristics of pedestrian crowds in order to ensure safe evacuation under emergent situations. In urban planning and building design, pedestrian crowd simulation is used to test the reliability of public facilities and architectural designs. Pedestrian crowd simulation also finds the application in the entertainment industry such as computer games where people study pedestrian crowd simulations to create realistic look and movement of pedestrians. The essential component which is included in pedestrian crowd simulations is the pedestrian crowd model. Over many years, people have developed many pedestrian crowd models. These models can be roughly categorized into physics inspired model [1-3], cellular automata model [4-6], network based model [7, 8], and agent-based model [9-13]. Physics inspired model is based on the similarity between gas/fluid dynamics and crowd behaviors. Cellular automata model is discrete-space model where pedestrians are located at nodes of a fixed or

adaptive grid. At each step pedestrians move at one or more grids. Network flow model is often used to model emergent evacuation in which buildings and the attributes of the buildings' components are represented as a static network. The evacuation process is then modeled as a dynamic network which is the time expanded version of the static network. Perhaps the most popular modeling paradigm in pedestrian crowd simulations is the agent-based modeling approach where each individual is modeled as an agent with its own characteristic and behavior situated in an environment.

Agent-based modeling originates from complex systems theory [14]. Complex systems theory studies complex systems where independent agents interact with each other [15]. An agent is a basic but interacting entity. Each agent could have simple behaviors. But interactions among agents yield complex collective behaviors which are not pre-programmed. One well-known agent-based model is the Boids model [16], where the flocking behavior is emergent in the movement of a school of birds, each of which is featured with three simple behaviors, cohesion, separation, and alignment. Fig.1.1 shows the three behaviors of the Boids model. Cohesion steers Boids towards the average position of local flocking mates. Separation lets Boids avoid the nearby flocking mates. Alignment steers Boids towards the average heading direction of local flocking mates. The movement of Boids is governed by the addition of the three behaviors.



(a) Separation (b)Alignment

(c) Cohesion

Figure 1.1 Three behaviors of the Boids model [17].

As one of popular pedestrian crowd models, agent-based crowd model is widely used in pedestrian crowd simulations. In agent-based crowd model, each pedestrian is typically modeled as an agent equipped with a set of attributes and other primitives. Unlike particle systems, such as idealized gas-like particles, pedestrian agents are heterogeneous, and dynamic in their attributes and behavioral rules [14]. For example, pedestrian agents could have different social and psychological states [17, 18], personality and emotions [19], and cultures [20]. Each pedestrian agent assesses its situation and makes decisions on a set of behavioral rules [21]. The complexity of pedestrian crowd and the capability of the agent-based modeling approach in simulating collective behavior from the interaction of pedestrians which are equipped

with simple local behavior rules make the approach be well suited for pedestrian crowd simulations. With the agent-based modeling approach, people have studied a set of interesting collective behaviors resulting from interactions among pedestrian agents, such as herding behavior [22], lane formation [1], and leader-follower behavior [23].

Currently, people have proposed many agent-based pedestrian crowd models [9-13] which try to capture the essence of pedestrian behaviors. The work of [10] presented a hierarchical model for real-time simulation of virtual human crowds. The work of [24] presented a computational framework, Multi-Agent Simulation System for Egress analysis (*MASSEgress*), which can be used to study collective behaviors such as competitive behavior, queuing behavior, and herding behavior, during the safe egress. The cognition of pedestrian agents is modeled with social identity and social proof theory. The work of [12] described a HiDAC (High-Density Autonomous Crowds) system for simulating the motion of large, dense crowds of autonomous agents in a dynamically changing virtual environments.

1.2 Grouping And Social Groups

Grouping is a common phenomenon in our daily life. Crowds contain both isolated individuals as well as persons in groups [25]. For example, family members walk beside each other in the shopping mall; in aquarium, a certificated leader guides a group of visitors to different popular exhibits; solders proceed in a line format. In the setting of a city, less than a half of the pedestrians walk alone [26]. Rather than a simple collection

or aggregate of individuals, a social group also has the characteristics which are shared by its members. Such characteristics include e.g. spatial distance (i.e. Euclidian distance), similar goal and interests (i.e. visit the same exhibit), kinship ties (i.e. family members), and social background (i.e. sociality, emotion). There are many definitions of social group in different areas. In the social sciences a group is usually defined as two or more humans who interact with one another, accept expectations and obligations as members of the group, and share a common identity. In computer science, an accepted definition is that of a large group of individuals in the same physical environment, sharing a common goal (e.g. people going to a rock show or a football match) [9]. This work also adopts this definition with the focus on the study of relationships (see details below).

Because of the intriguing and complex nature, social group is extensively studied in socio-psychology. It is also studied in computer science where people develop computer models trying to explore crowds' grouping behavior.

1.3 Related Work

1.3.1 Study of Social Group In Sociology And Psychology

Social group is one of popular research directions in Sociology and Psychology.

In sociology, besides the aggregation nature, members in a social group also develop a set of roles and interpersonal relationships which serve to differentiate from other groups, and represent the "likes" and "dislikes" of members for one another, respectively [27]. A group could be large or small. Large groups include a nation, a

society, an institution, etc. Small groups include family units, friendship groups, etc. There are numerous works on small groups with intimate, kin-based relationships focusing on the inter-group and intra-group relationships (see, e.g., [27]).

Groups are also extensively studied in social psychology. Roughly speaking, there are two traditions of the research of social group in social psychology.

One tradition is the collective tradition. The work of [28] viewed the behavior in social aggregates as guided by a "group mind". The group mind is the force defined by the aggregation and it is not same as individual behaviors or individual minds. The work of [29] emphasized a "collective mind" in groups. According to his work, a collection of individuals does not suffice to form a group. In a group, persons take the same sentiments and ideas, and their conscious personality vanishes. In another word, the individuality of persons is weakened and the heterogeneous individuals become a homogeneous group. Both McDougall's and Le Bon's work viewed that a group is not simply a combination of individuals and groups should be guided by "collective consciousness". Freud [30] extended Le Bon's work on "collective mind" by explaining what makes persons unite together. According to Freud's work, persons gather together because they have something in common with one another. Persons in a group would have a common interest in an object, a similar emotional bias in some situation and consequently some degree of reciprocal influence. Freud's work also emphasizes the interaction between groups, each of which is ruled a single person (leader) and all members follow the leader.

The other tradition is the individualistic tradition. Unlike the collective tradition, the individualistic tradition emphasizes the analysis of group behaviors from the psyche of individuals. One famous work on the individualistic tradition is that of Allport's work [31] which states that : "there is no psychology of groups which is not essentially and entirely a psychology of individuals".

Groups are widely studied within the context of institution or organization to study the group dynamics including the roles of individuals and role conflicts, the effect of group dynamics on personal development, etc. One of important topic in group dynamics is the study of the structural properties of social groups. Here a social group is mainly referred as an organ. Social structure is one of central concepts in sociology which studies the social relationships among members of an organ. One of the well-known social relationships is the leadership where the research focuses on the personality and situational perspectives [32].

1.3.2 Social Group Modeling And Simulation

The grouping relationship can play important roles in crowd behavior [26].

The work of [33] simulates the kin behavior in emergent evacuations and shows that the number of the sub-groups and the members in each sub-group influence the evacuation efficiency a lot. The work of [34] studies the effect of group size on the walking speed through controlled experiments and concludes that the walking speed decreases as the group becomes larger.

Social group is also widely studied in the entertainment industry in order to create realistic animations. In this field, the definition of a group is often given by a list of people who share common goals and movement parameters, for example, the same average speed and destinations. A brief introduction of some related work is listed below. The comprehensive survey of social group model in the entertainment industry is out of range of this dissertation.

In the work of [9], a group is configured with some parameters: the goals (specific positions which the group should reach), group size and the level of dominance (the role of group leader and group members). Besides the shared group parameters, each pedestrian has an emotional state which is a normalized value in [0, 1]. During the simulation, a pedestrian A may join a group B if the relationship between A and B is very high (>0.9) or the relationship is higher than the relationship between A and its own group. The calculation of the relationship is based on the emotional state of A. Once A joins B, some parameters may be changed. For example, the group leader of B may be changed and the goals of A will be changed to those of B. Fig.1.2 shows a simulation scenario of formed groups in a museum. At the beginning of the simulation, the virtual humans are in their randomized initial position. After interactions, the agents are gathered in groups and walk in the museum. Members of the same group look at a same specific work of art.

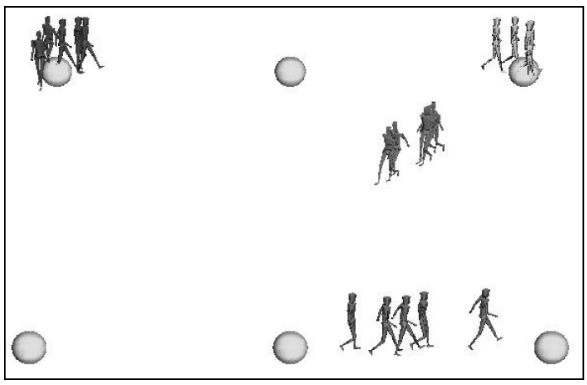


Figure 1.2 Formed groups in a museum of Musse's model [9].

The work of [35] described a model to generate and animate groups which emerge as a function of interactions among virtual agents. In their model, each pedestrian is featured with a set of social parameters and has the capability of movement, perception, interaction, and memory. During the simulation, each pedestrian is able to follow others, to evaluate the quality of interactions, to select the pedestrians with good quality of interactions, and to form groups with those having good quality of interactions. During the simulation, an interaction happens when both pedestrians resides within the perception area of each other. An interaction makes a change to the social states of the participants according to a set of predefined equations. Simulation scenarios of Villamil's model are shown in Fig.1.3. From left to right, the formed groups have high, medium and low cohesion, respectively.

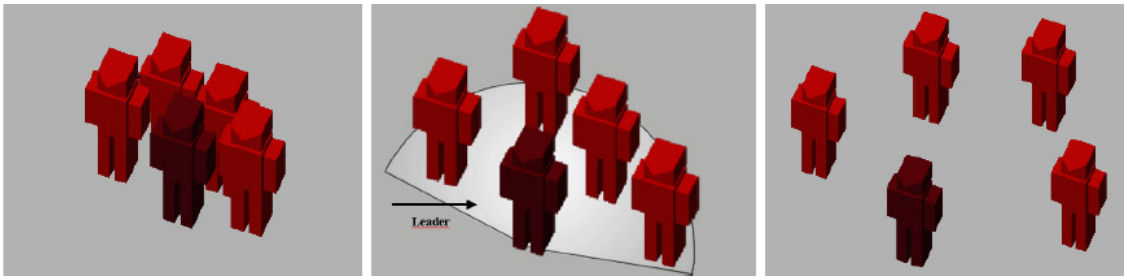


Figure 1.3 Simulation scenarios of Villamil's model [35].

Recently, socio-psychological theories or models, such as Five-Factor personality model and the social comparison theory, are often used to describe the group dynamics and simulate the social groups in pedestrian crowds. The work of [36-38] incorporates the OCEAN personality model into their simulation system to make the simulation more realistic. The work of [37] studied dynamic personality and the impact of the openness personality trait on problem solving ability and cognitive complexity. The work of [38] simulated the Big Five personality factors and associated forces using the Helbing-Molnar-Farjas-Vicsek (HMFV) crowd model.

The work of [39] described a computational framework for pedestrian crowd simulations based on Festinger's social comparison theory [40] which states that when lacking objective means for evaluation of the current situations, people tend to compare their behavior with others that are most like them. In Fridman's framework, each pedestrian is equipped with a set of features, to each of which a real number is assigned to indicate its weight. During the simulation, each pedestrian calculates the similarity with the nearby pedestrians. The pedestrian with the greatest similarity value is selected. The comparing pedestrian then applies actions to minimize the differences

between the target pedestrian and the comparing pedestrian. The similarity is calculated as the weighted sum of the features. Experimental results show that the framework can simulate some behavior scenarios such as the lane formation with the existence of individual or grouped pedestrians. Fig. 1.4 illustrates the simulated grouped movement of pedestrians. The first figure shows the initial position of pedestrians. The second figure shows a simulation scenario after 5000 simulation steps without social comparison. The third figure shows the grouped pedestrian movement after 5000 simulation steps with social comparison: pedestrians with similar color group together.

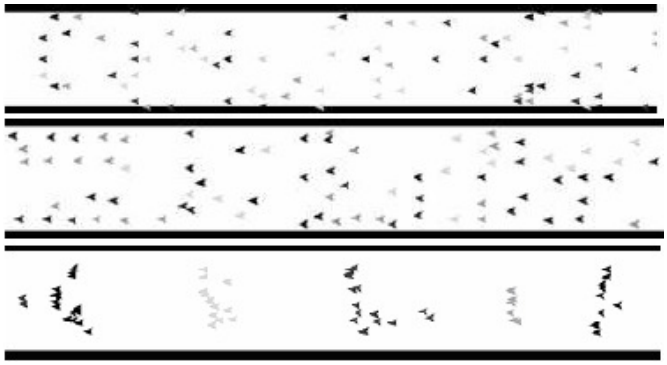


Figure 1.4 Grouped pedestrian movement in Fridman's work [39].

1.4 Motivation And Objective

It is necessary to study different aspects of social groups, such as relationship between group members, the context the groups operate, physical and psychological ability, group size and density, to explore the effect of grouping on crowd behaviors

[41]. For example, it is necessary to test the reliability of the design of public facilities during the emergent situations when the pedestrian crowd consists of different social groups. It is also worthy to explore the emergent evacuation efficiency for pedestrian crowd featuring with social groups. As discussed in [41], different groups affect the flow as well as evacuation efficiency in emergency situations, e.g., a leader-follower group may be more smooth and efficient than a clustered group if a group has a large number of members. In this case, a clustered group can result in slow movement, especially in a constrained area. The study of social groups also serves another purpose of creating realistic simulation scenarios [26].

Because of the complexity nature due to individual differences (e.g age, gender, culture, etc.) and non-linear interaction among individuals, the influence of grouping has not been widely incorporated into pedestrian crowd simulation. Most existing work on group modeling only focuses on the study of one specific social group such as the leader-follower group. Also most social group models are operated within one specific social or psychological context. It is believed that modeling different groups and different social/psychological contexts which the groups are operating can facilitate the understanding of the effect of grouping on crowd behavior. Towards the goal of facilitating the study of different social groups and exploring the effect of various socio-psychological factors on crowd's behaviors, this research develops a well-defined framework for people to systematically study social groups in agent-based pedestrian crowd simulations. Specifically, the framework allows the study of the interaction

within the groups, i.e., the study of grouping behavior from predefined or dynamically changed kinship ties (see Chapter 4 and 5 for details).

Towards such goal, the framework is featured with a multi-layered architecture within which the social context is separated from the specific group model. Here social context refers to the model of agent's socio-psychological states. This separation has the advantage that one can study the effect of different socio-psychological factors on crowd's group behaviors and various social groups independently. That is, one can study the effect of different socio-psychological factors on crowd's group behaviors without the change of underlining group model; and one can study different social groups without the change of the social context either.

1.5 Summary And Contribution

This research develops a framework which includes multiple layers that support a systematic approach for modeling social groups in pedestrian crowd simulations. These layers include a simulation engine layer that provides efficient simulation engines to simulate the crowd model; a behavior-based agent modeling layers that supports developing agent models using the developed *BehaviorSim* simulation software; a group modeling layer that provides a well-defined way to model inter-group relationships and intra-group connections among pedestrian agents in a crowd; and finally a context modeling layer that allows users to incorporate various social/psychological models for study social groups in pedestrian crowd. Each layer utilizes the layer below it to fulfill

its function, and together these layers provide an integrated framework for support group modeling in pedestrian crowd simulations. To our knowledge this is the first work to focus on a systematic group modeling approach for pedestrian crowd simulations. This systematic modeling approach allows users to create social group simulation models in a well-defined way for studying the effect of social and psychological factors on crowd's grouping behavior.

To facilitate group modeling and simulation, a behavior-based agent simulation software *BehaviorSim* is developed. One important effort in developing *BehaviorSim* is to propose a general and well-defined behavior-based model which can capture the essence of agent's behavior. The platform is so intuitive and easy-to-use that people can setup a behavior-based pedestrian crowd simulation without significant programming skills. The flexible design makes it applicable for the simulation of behavior-based agents such as robot, animat, an artificial agent in AI, a character in game design, and so on. *BehaviorSim* is incorporated into a modeling and simulation class to be used by students.

To demonstrate the capability of the framework, we developed an application of dynamic grouping for pedestrian crowd simulations. The dynamic groups are studied using the proposed framework on the basis of utility theory and social comparison theory. Experiment results indicate that groups can be dynamically formed and grouping has significant effect on crowd behaviors.

As the pedestrian crowd simulation systems become more and more pervasive, the scale of pedestrian crowd model also increases. Besides, to create more accurate simulations, people tend to make the decision-making model of pedestrian agents more complex. These affect the simulation performance. Although traditional discrete time based simulation approach is easy to understand, it is inefficient since every agent makes a decision at every time step, regardless their behaviors and moving speeds. To improve the performance of pedestrian crowd simulations, we developed a new method for pedestrian crowd modeling and simulation. This new method uses a discrete event based approach by exploiting the crowd system's heterogeneity resulting from agents' different moving speeds. Experiment results show that, under the condition of fair comparison, the discrete event based approach can lead to more computationally efficient simulations than the discrete time based approach for crowds with different moving speeds.

1.6 Dissertation Organization

The remaining chapters are organized as follows. Chapter 2 proposes the framework for group modeling in agent-based pedestrian crowd simulations. The overview of layered architecture of the framework is introduced, followed by the detailed description of each layer. The relationships between the different layers are also illustrated. Chapter 3 introduces the behavior-based agent modeling layer which is implemented in the behavior-based agent simulation software *BehaviorSim*. The key

concepts underling the platform such as behavior-based control, behavior, and behavior network are described. The behavior-based pedestrian agent model and its working mechanism are also introduced. Chapter 4 presents the group modeling layer. Important group parameters as well as the modeling of both intra-group connections and inter-group relationships are described in detail. As a case study, three social groups are modeled through several simple steps. Several experiments are carried out to study the effect of group structures on crowd behaviors. Chapter 5 describes a specific application of the developed framework. The application studies dynamic grouping behavior in pedestrian crowd simulations on the basis of utility theory and social comparison theory which are modeled in the context layer. Chapter 6 presents our effort in developing an efficient discrete event based simulation engine by exploiting the crowd system's heterogeneity resulting from agents' different moving speeds. The condition of fair comparison between the traditional discrete time based approach and our discrete event based approach is established. Under fair comparison condition, this chapter also compares the two approaches by a set of experiments. The efficiency of our proposed discrete event based approach is analyzed through these experiments. Chapter 7 concludes this dissertation and provides some future research directions.

CHAPTER 2

A FRAMEWORK FOR GROUP MODELING

2.1 Overview

The design of the framework follows the decomposition principle, which is an important principle of Software Engineering in the phase of system design [42]. This principle, i.e. decomposing the system to smaller subsystems, is useful for the design of complex systems such as the pedestrian crowd systems. For complex systems, decomposition can ease the maintenance when some subsystems are changed. Specifically a pedestrian crowd simulation is decomposed to three subsystems: simulation engine, crowd model, and visualization. The crowd model is further decomposed into a set of other models: the agent model, the group model, the environment model and the context model. In the agent model, a pedestrian agent is modeled as an independent component which consists of a set of attributes and models which characterize different features of a pedestrian agent. For example, a pedestrian agent is often equipped with the perception model and the behavior model.

The goal of the design is to achieve the decomposition with high cohesion in each subsystem and loose coupling between subsystems. High cohesion means that the constituted units in the subsystem perform similar tasks and are related to each other. Loose coupling means that changes to one subsystem will not have high impact on other subsystems [42]. Layering techniques are often used to achieve loose coupling. A

layer represents a level of abstraction which only provides services to subsystems of a higher layer. There are many advantages of layering: easy identification of relationships eases the maintenance and the update of systems.

Applied with the layering techniques, the framework consists of four-layer architecture with each layer focusing on some specific functionality, as shown in Fig. 2.1. From bottom to up, they are *simulation engine, behavior-based agent modeling layer, group modeling layer* and *context modeling layer.*

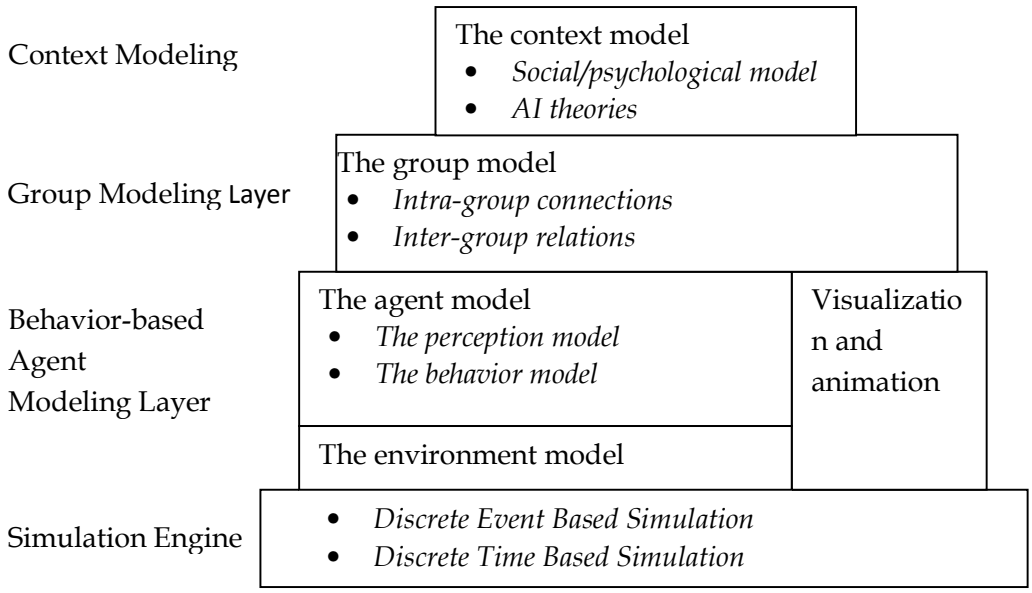


Figure 2.1 The framework of group modeling with four-layer architecture.

2.2 Simulation Engine

Simulation engine drives the simulation of crowd behaviors. The basic simulation protocol follows the discrete time based approach. The discrete time based simulation

consists of a list of continuous time steps, in each of which every pedestrian makes a movement decision. Decision making is an important component in pedestrian crowd model. In each time step, the simulation engine calculates the physical and psychological states for each pedestrian agent according to its decision making component. Typical physical and psychological states include moving direction/speed and emotions such as sad, happy, respectively. Those states are generally fed into the visualization and animation subsystem in order to display or animate the stepwise agent characteristics. The discrete time based approach is the simulation approach adopted in this framework.

Another simulation protocol is the discrete event based approach, where pedestrians' movement decision is only triggered by events, such as the change of internal states or external environment. Although the discrete time based approach is simple and easy-to-understand, it is inefficient in the simulation of pedestrian crowds where agents have different moving speeds. The inefficiency roots in the fact that every agent makes a decision at every time step, regardless the individual differences such as different agent, sex, moving speed, and so on. The inefficiency is even worse for the large-scale complicated pedestrian crowd simulations where each agent is featured with a complicated decision making component and the pedestrian crowd contains a large amount of agents to simulate. Because of this, Chapter 6 proposed an efficient discrete event based simulation engine by exploiting the crowd system's heterogeneity resulting from agents' differences.

2.3 Behavior-based Agent Modeling Layer

Behavior-based Agent Modeling Layer contains the visualization and animation module, the environment model, and the agent model.

The visualization and animation module is used to visualize and animate the simulation results, i.e. agents' physical and psychological states, respectively. In this framework, simulation results are displayed as a series of two-dimensional images. Animation is an important approach to produce stereoscopic simulation scenarios. People have developed a lot of animation techniques for pedestrian crowd simulations. For the sake of simplicity, this framework does not focus on the visualization and animation component. However, it is possible to use any visualization and animation technique in the framework to create stereoscopic and sophisticated simulation scenarios.

The environment model describes the virtual simulation environment which pedestrian agents interact with. The interaction between agents and the simulation environment is achieved by a "perception-decision-movement" process. In this process, perception represents the limited visible range an agent can perceive during the simulation. Agents perceive the environment information by their perception model (which will be described in Chapter 3). Fed with the environment information, agents make movement a decision based on the desired goal. Once the movement decision has been made, agents carry out the movement which is based on the decided moving

direction and speed. This process generally changes the state of the simulation environment such as the position of pedestrian agents.

The agent model specifies agents' characteristics and behaviors. Like most agent-based pedestrian crowd simulations, each agent has the physical, social and psychological attributes such as age, sociality, and emotion. Unlike the existing work, pedestrian agents belong to social groups, i.e. a pedestrian crowd is considered as a set of social groups. Each social group may contain arbitrary number of pedestrian agents. This makes the study of social groups in pedestrian crowd simulation easier (check Chapter 4 for more details). One important aspect of the agent model is the decision making component which is specified in the agent's behavior model.

The agent's behavior model decides agent's behavior at each time step. In this framework, the behavior model adopts a behavior-based control architecture which can be used to study the adaptive and emergent behaviors in agent-based pedestrian crowds [43]. Such architecture is featured with a set of competing or collaborative behaviors and an action selection mechanism. Currently, two action selection mechanisms, namely the *mutual inhibition/excitation* mechanism and the *cooperation* mechanism are supported. For the *mutual inhibition/excitation* mechanism, only one of the behaviors is selected to control the agent's movement. For the *cooperation* mechanism, the behaviors are added together using a vector-sum approach, and the agent's movement is controlled by the result vector. Each pedestrian agent has the behavior of *RandomMove*, *Avoid* and *MaintainGroup*. *RandomMove* allows pedestrian

agents to move around the simulation environment randomly. *Avoid* lets agents avoid collision with nearby obstacles and other agents. *MaintainGroup* lets agents maintain the desired types of social groups (see Chapter 4 for more details). The details of behavior-based control architecture and our pedestrian crowd simulation system will be introduced in Chapter 3.

2.4 Group Modeling Layer

An important component in *Group Modeling Layer* is the group model which can be used to systematically study a variety of social groups and different aspects of social groups. The group model is implemented in the maintaining group behavior as mentioned above. Social group is studied from two aspects, *intra-group connections* and *inter-group relationships*. *Intra-group connections* represent the relationships among members, e.g. likeness and familiarity, in the same group. *Inter-group relationships* represent the relationships among groups, e.g. following, in the pedestrian crowd. Both relationships are represented by a real number which indicates the strength of relationships. There are two types of relationships: static relationship and dynamic relationship. Static relationship represents the relationship which is not changed during the simulation. The static relationship is usually found in stable social groups, e.g. the family units, where the relationship is not changed as the time passes by. While dynamic relationship represents the ever-changing relationship and it is usually found in unstable social groups, e.g. the task-based groups, where the relationship is

temporarily formulated (e.g. before the task begins) and distinguished after some time (e.g. the task is finished). The dynamic relationship is often affected by social or psychological factors, e.g. uncertainty and stress. The effect of these factors on crowd behavior is usually studied through social or psychological theories such as social comparison theory, social proof theory, the five-factor model, and so on. Such effect is achieved by changing the strength of relationships dynamically (see Chapter 5 of dynamic grouping for more details).

By setting different *intra-group connections* and *inter-group relationships*, we can easily create different types of social groups. Chapter 4 develops three social groups by following simple steps.

2.5 Context Modeling Layer

The context is not the physical context of the simulation environment. It represents the social/psychological context which affects agent behaviors. *Context Modeling Layer* clearly separates the social/psychological model from the underlining group model. Unlike most existing work on simulating social groups where the two models are tightly associated with each other, our framework removes such tight associations. This makes the framework be easily used to study the effect of various social/psychological factors on crowd behavior. Moreover, the framework can also study the effect of other models or theories such as artificial intelligence on crowd behavior. AI theories, e.g. the theory of intelligent agent, can be used to create more intelligent pedestrian agents.

However, the effect of artificial intelligence on crowd behavior has not been widely studied in the existing work. An application of the framework – dynamic grouping – presented in Chapter 5 illustrates applying one of AI theories, the utility theory, to the dynamic group model.

2.6 Relationship Between Layers

The framework is featured with loosely coupled layers. Each layer utilizes services from the lower layer through a well-defined set of façade interfaces, as shown in Fig. 2.3.

The context modeling layer feeds the effect of social/psychological context to the group modeling layer by changing the strength of relationships which is represented in two-dimensional matrices (see Chapter 4 for details).

The group modeling layer provides the group information to the behavior-based agent modeling layer by changing the input of the maintaining group behavior so that groups can be properly maintained.

The behavior-based agent modeling layer provides the agent data such as the current position to the simulation engine layer by changing the input channel of the simulation engine.

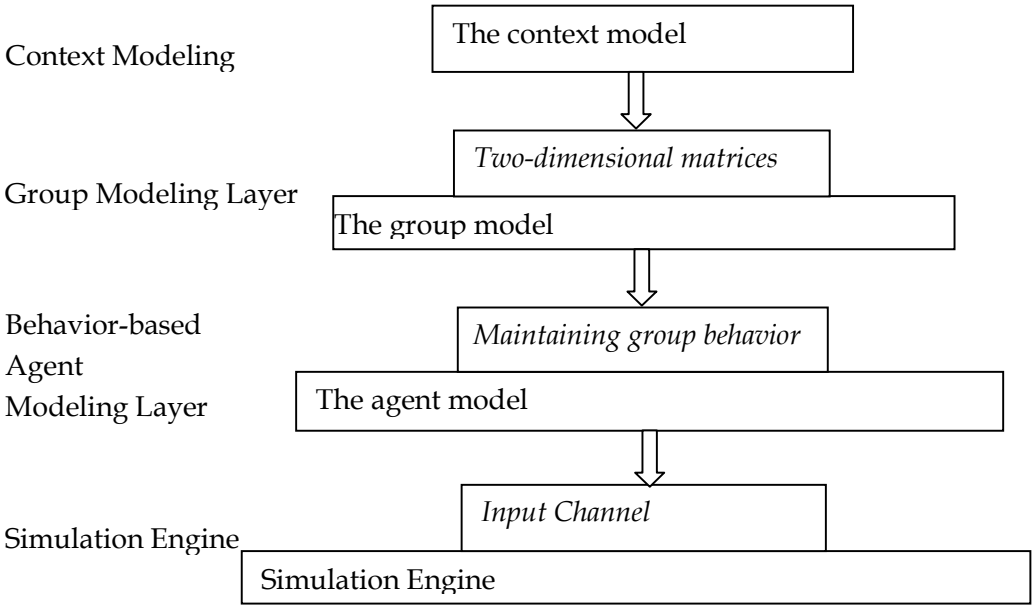


Figure 2.2 Relationship between framework layers.

Note that there are several relationships which are not shown in Fig. 2.2. There is relationship between simulation engine and visualization/animation module in that simulation engine provides the simulation data to the visualization/animation module which displays/animates the simulation scenarios. The relationship between the agent model and the environment is bi-directional. The agent model gets the information from the environment model to make decisions, which in turn changes the state of the environment. The relationship between the environment model and the visualization/animation module is similar to the relationship between simulation engine and visualization/animation module. The environment model provides spatial information to the visualization/animation module to render it.

CHAPTER 3

BEHAVIOR-BASED AGENT MODELING AND PEDESTRIAN CROWD SIMULATION

3.1 Behavior-based Control

Behavior-based control is one of the fundamental control paradigms for autonomous agents to achieve adaptive behavior in a dynamical environment [44], [45]. It finds applications in many different fields such as robotics, artificial intelligence, computer game design, and social crowd simulations. The behavior-based control is featured with a set of behaviors working in parallel, each of which corresponds to a specific behavior of the agent. For example, in the team formation described in the work of [46], each robot is featured with four behaviors, avoid obstacles, follow the predecessor, search for the predecessor, and wait for the followers. This results in a distributed control paradigm as opposed to the centralized deliberative control. The term "behavior-based agent" is adapted from behavior-based robotics [47]. Here an agent is a general term that can be a robot, an animat, an artificial agent in artificial intelligence, a character in game design, or a pedestrian agent in pedestrian crowd simulations.

One critical issue in adopting the behavior-based control paradigm is to find a general behavior-based control architecture which can capture the essence of behavior-based control.

Researchers have proposed several control architectures, such as Subsumption architecture [48], Activation spreading network [49], and Motor schema architecture

[47]. However, behavior-based control has taken many different formats. Each architecture has a different working mechanism and currently there is no "standard" behavior-based architecture (see a discussion in [44]). Some architectures also include concepts, e.g. "goal" in the Activation spreading network [49], that are not explicitly supported by others. Furthermore, there is no common definition of what is a behavior and different architectures may define behaviors at different levels of details. For example, the work of [50] allows high level behavior modules (called options) to be broken down into low-level modules and basic behaviors.

One effort is to propose a general and well-defined behavior-based model which can capture the key components of behavior-based control (see details in the following section). To design the behavior-based model, we proposed an unambiguous view of what is a behavior and how multiple behaviors work together for the control of agent's movement.

3.2 Behavior And Behavior Network

The behavior-based model is inspired from the neurobiological work of mutual inhibition, behavior-based robotics, and the Boids model:

- The neurobiological study, in particular Edwards' work [51], of the mutual inhibition mechanism to account for animals' adaptive behavior in a dynamical environment,
- The work of behavior-based robotics [47], and

- The "Boids" model [16] that demonstrates a variety of steering behaviors in a simulated environment (see Fig.1.1 for details).

Specifically, a behavior is viewed as an independent computation module that can fulfill some particular task for an agent. Each agent can have multiple behaviors that run in parallel. These behaviors form a behavior network that defines how different behaviors compete or cooperate with each other for controlling the agent. This behavior network and its working mechanism represent the behavior-based control architecture and act as the decision making component of the agent. The structures of behavior and behavior network adopted in the general behavior-based model are described below.

In general, a behavior is excited by some (external) sensory stimulus and/or (internal) states of the agent. It defines some actions to fulfill the task associated with this behavior. Based on this, a behavior is characterized by three elements: an *activation* value, an *excitation* module, and an *action* module. The *activation* is a real number and represents the level of strength of this behavior. Its value is computed from two sources: stimulus of the sensory inputs and/or internal states, and inhibitions/excitations from other behaviors. This value is computed in every time step and used in the action selection of the behavior network. The *excitation* module defines how this behavior is excited by the sensory inputs and/or internal states. It returns a value (a real number) called *excitation*. Two things are worthy to mention here.

First, the concepts of sensing and internal states are not explicitly modeled in this general behavior-based model. In this dissertation (see Fig. 2.1 for details), sensing is

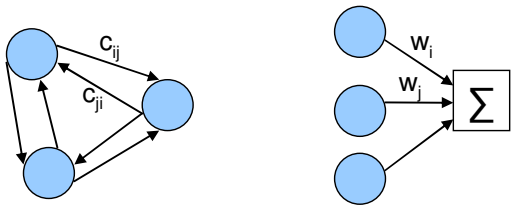
fulfilled in the agent's perception model. The sensing information, such as nearby agents and obstacles, can be forwarded to behaviors in this general behavior-based model as needed. The internal states are modeled through a set of attributes representing agent's various characteristics.

Second, socio-psychological concepts, such as emotion and motivation, are not explicitly modeled in this behavior-based model. Instead, they are presented in the *Context Modeling Layer* (see Fig. 2.1 for details). However, the socio-psychological states can be used by the *excitation* module and they can affect inhibitions/excitations from other behaviors.

The *action* module of a behavior specifies the actions that will be carried out by the behavior if it is selected (a selected behavior is also referred to as *active*). Typically, a behavior's action returns a speed vector that drives the agent to move for one step (if it is active). The action is generally a "one-step" task. However, there are situations where a behavior needs to supports a sequence of tasks in order. For example, considering a "dance" behavior of a mobile robot, if this behavior remains to be active, the robot should dance (move) in a particular order, e.g., moving left first, then right, then…. To support this kind of sequential task, a *task queue* is set up in the *action* module. A task queue defines a list of tasks that will be sequentially executed as long as the behavior continues to be active. When all the tasks in the queue have been executed, the task queue is reset and then executed from the beginning again. A behavior's task queue can be *resumeable* or *non-resumeable*. A resumeable task queue allows a re-activated (from

non-*active*) behavior to resume from the task that was previously interrupted. This is useful to keep track of the progress of a task sequence and to continue the task from where it stops. A non-resumeable task queue is always reset if it is interrupted. Similar to the work [52], the behavior-based model allows a task queue to be set up in a hierarchical manner and supports both sequential tasks and concurrent tasks.

Multiple behaviors form a behavior network that specifies how these behaviors compete or cooperate with each other. A behavior network is characterized by three types of elements: a set of *behaviors*, an *action selection mechanism*, and a set of *coefficients (or weights)* that define the connections among the behaviors. The action selection mechanism defines how the set of behaviors work together. Currently, two action selection mechanisms, namely the *mutual inhibition/excitation* mechanism and the *cooperation* mechanism, are supported. Fig.3.1 illustrates these two mechanisms, where the blue circles are the behaviors.



(a) Mutual inhibition/excitation (b) Cooperation

Figure 3.1 Two action selection mechanisms in *BehaviorSim*.

The *mutual inhibition/excitation* mechanism is a competitive mechanism that allows only one behavior to be selected at any time step based on the winner-take-all principle. In this mechanism, different behaviors asymmetrically inhibit/excite each other through the inhibitory/excitatory *coefficients*, e.g. C_{ij} and C_{ji} in Fig.3.1 (a), which define

the level of inhibition/excitation. A coefficient is a real number ∈ [-1, 1] with a negative value meaning inhibition, a positive value meaning excitation, and 0 meaning no inhibition or excitation. The amount of inhibition/excitation from one behavior to another is computed according to the product of the first behavior's activation and the corresponding coefficient.

$$Activation_j = Excitation_j(S_j, I_j) + \sum_{i}^{i \neq j} c_{ij} * Activation_i \quad (3.1)$$

$$Sv = Sv_{winner} \quad \text{where} \quad winner = Select(Activation_1, ..., Activation_n) \quad (3.2)$$

$$Activation_j = Excitation_j(S_j, I_j) \quad (3.3)$$

$$Sv = \sum_i k_i * w_i * Sv_i \quad \text{where} \quad k_i = \begin{cases} 1 & \text{if } Activition_i \geq T_{threshold} \\ 0 & \text{if } Activition_i < T_{threshold} \end{cases} \quad (3.4)$$

Eq.3.1 shows how a behavior (*behavior_j*)'s activation is calculated. In this formula, c_{ij} is the coefficient from *behavior_i* to *behavior_j*; $Excitation_j(S_j, I_j)$ represents *behavior_j*'s excitation, which is a function of the behavior's internal states S_j and sensory inputs I_j. After all behaviors' activations are calculated, the behavior network selects the behavior with the highest activation level as the winner (*active*) behavior which controls the action of the agent for this time step. In the context of autonomous agent, Eq.3.2 shows that the agent's speed vector *Sv* is defined by the action (speed vector) of the winner (*active*) behavior. Note that the *Select()* method selects the winner (*active*) behavior by comparing the activations of all behaviors.

Different from the *mutual inhibition/excitation* mechanism that selects only one behavior, the *cooperation* mechanism combines the actions of multiple behaviors based on the vector sum principle. The amount of contribution of each behavior is defined by a weight $w \in [0, 1]$, e.g. W_i and W_j in Fig. 3.1 (b). Eq.3.3 shows that in this mechanism, the activation of a behavior only includes the excitation part (since there is no *inhibition/excitation* from other behaviors). Eq.3.4 shows the speed vector *Sv* of the agent is calculated as the vector sum of the actions of those behaviors whose activations are greater than or equal to a pre-defined threshold $T_{threshold}$.

The set of *coefficients* (*weights*) of the behavior network plays important roles in selecting the winner (*active*) behavior or in vector summing the actions of multiple behaviors. The values of these coefficients (weights) can be defined as constants, or be dynamically computed in every time step based on user-defined rules. For example, it makes sense to dynamically adjust the weights of different behaviors used in Fig.3.4 according to the current activations of those behaviors. The work of [53] shows another example where the mutual inhibition coefficients among behaviors are dynamically changed under different conditions.

3.3 Action Selection Mechanism

Action selection is conducted in a step-wise manner as the simulation proceeds. For the discrete time based simulation, in every time step, each agent goes through the following major steps of decision making. While for the discrete event based simulation

(see Chapter 6 for details), only the change of the external environment or the agent's internal states will trigger the following steps.

- Compute the coefficients/weights if they are not constants.
- Calculate excitation of each behavior: $Excitation_j(S_j, I_j)$ in Eq.3.1 or Eq.3.3.
- For the mutual inhibition/excitation mechanism, calculate activation of each behavior using Eq.3.1. Skip this step for the cooperation mechanism.
- For the mutual inhibition/excitation mechanism, select the one behavior (the winner/active behavior) with the highest activation. For the cooperation mechanism, select all the behaviors whose activations are greater than the threshold *Tthreshold*.
- Execute the actions associated with the selected behaviors. For the mutual inhibition/excitation mechanism, only one behavior is selected, thus execute the action for that behavior. For the cooperation mechanism, vector sum all the speed vectors returned from the selected behaviors' action modules. In both cases, if there is a task queue defined for the selected behavior, the task queue is reset if the queue is empty. Otherwise, execute the next task in the queue.
- The executed actions modify the attributes of the corresponding agents which in turn modify the simulation environment.

3.4 *BehaviorSim* Software

A simulation software *BehaviorSim* is developed for users to create different applications based on the behavior-based control paradigm. It allows users to define a

simulation system including behavior-based agents and then run simulations to see the result. This section gives a formal description of a behavior-based agent system developed in *BehaviorSim*, followed by the user interfaces of the environment.

3.4.1 Behavior-based Agent System Specification

In *BehaviorSim*, a behavior-based agent system is described by a tuple <*World, Entities*>, where *World* is a two-dimensional environment and *Entities* is the set of entities.

- The *world* is described by a tuple <*Dimension, Type*>, where *Dimension* specifies the dimension of the two-dimensional map. *Type = {closed | open | rounded},* where *closed* means there exists a "wall" surrounding the field that prohibits agents from moving outside of the field; *open* means the field is open thus agents can move outside of the field and disappear; *rounded* means that an agent moving outside an edge of the map will automatically appear on the other side of the map.

- The *Entities* includes a set of *stationary entities* and *autonomous agents*. A *stationary entity* represents a stationary environmental object such as obstacle, or a source for sensing or a destination point, which is meaningful for the autonomous agents. A stationary entity is described by a tuple <*EntityId, Name, Position, Image, Dimension, Category, {Property}*>, where *EntityId*, *Name, Position, Image, Dimension* are the identification, name, position, image and size of the entity, respectively; *Category* is the entity type, such as "pedestrian", "obstacle". Each category is described by <*CategoryId,*

Name, Image, {Property}>, where *CategoryId, Name,* and *Image* refers to the identification, name, and image of the category, respectively. Note that multiple entities can belong to the same category and thus share common information of the category. *{Property}* is a set of properties that describe domain specific information of the entity. For example, a *food* entity can have a property of *amount*. Each *property* is described by a tuple <*Name, Value, Dynamics*>, where *Name* and *Value* are the name and value of the property, respectively. *Dynamics* is a function specifying how the value of that property may dynamically change in every time step.

An *autonomous agent* represents a behavior-based agent such as a robot, an animat, or a game character. It is described by a tuple <*EntityId, Name, Position, Image, Dimension, Category, SpeedVector, {Property}, Behavior network*>, where *EntityId, Name, Position, Image, Dimension, Category,* and *{Property}* are the same as those of stationary entities. *SpeedVector* represents the current speed vector of the autonomous agent. It specifies both moving speed and direction of the agent at the current time step. *Behavior network* represents the decision making component of an autonomous agent.

The *Behavior network* is described by a tuple <*Behaviors, Action selection mechanism, coefficients/weights*>, where *Behaviors* is a set of behaviors; *Action selection mechanism*= {*mutual inhibition/excitation* | *cooperation*}; and *coefficients/weights* are a list of coefficients/weights of the behavior network. Each *behavior* is described by a tuple <*Name, Activation, Excitation, Action*> (See Section 3.2 for more details).

Both the *Excitation* and *Action* modules of a behavior can invoke method calls. *BehaviorSim* provides a set of primitive system APIs, such as *move (speed, direction)* to move an agent that can be called by users when defining behaviors. Some important system APIs and the descriptions are shown in Table 3.1.

Table 3.1 System APIs of Behaviorsim.

Get current simulation time	
Get the current simulation time	`GetTime()`
Get entities	
Get the identification of current computing entity	`GetMyId()`
Get the identification of specified entity	`GetEntityId(entityName)`
Get the closest entity from the specified category	`GetClosestEntityInCategory(categoryName)`
Get the list of entities that are within the specified distance	`GetListOfEntitiesWithinDistance(distance)`
Get the list of entities that belong to the specified category	`GetListOfEntitiesInCategory(categoryName)`
Get the list of entities of the specified category that are within the specified distance	`getListOfCategoryEntitiesWithinDistance(categoryName, distance)`
Get/Set position/speed/direction/property	
Get position of the specified entity	`GetPosition(entityId)`
Set position of the specified entity	`SetPosition(entityId, X, Y)`
Get relative position from the specified entity to current computing entity	`GetRelativePosition(entityId)`
Get the reverse relative position from the specified entity to current computing entity	`GetReverseRelativePosition(entityId)`

Get moving speed of the specified entity	`GetSpeed(entityId)`
Set moving speed of the specified entity	`SetSpeed(entityId, speed)`
Get moving direction of the specified entity	`GetDirection(entityId)`
Set moving direction of the specified entity	`SetDirection(entityId, direction)`
Get the direction from current computing entity to the specified entity	`GetDirectionToEntity(entityId)`
Get the direction from the specified entity to current computing entity	`GetReverseDirectionToEntity(entityId)`
Get the value of the specified property of the current computing entity	`GetValue(propertyName)`
Set the value of the specified property of the current computing entity	`SetValue(propertyName, propertyValue)`

Get distance

Get the distance between current computing entity and the specified entity	`GetDistanceToEntity(entityId)`
Get the distance between two specified entities	`GetDistanceBetweenTwoEntities(aId, bId)`
Get the distance to the closest entity in the specified category	`GetClosestDistanceToCategory(categoryName)`
Get the IR (infrared) distance of the specified entity in the specified direction	`GetIRDistance(entityId, direction)`
Get the IR distance of the specified entity in all directions	`GetIRDistanceToOthers(entityId)`
Get the IR distance of the specified entity to entities in the specified category	`GetIRDistanceToCategory(entityId, categoryName)`

Get/Set behviornetwork

Set coefficiencies/weights	`SetBehaviorNetworkTable(param)`

Set weight for specified behavior	`SetBehaviorWeight(behaviorName, weight)`
Get behavior activation of current computing entity	`GetBehaviorActivation(behaviorName)`

Move/Turn

Move the current computing entity according to the specified speed and direction	`Move(speed, direction)`
Move the current computing entity for a distance	`MoveForDistance(speed, direction, distance)`
Turn an angle for the current computing entity	`Turn(angularSpeed)`
Turn the current computing entity for an angle	`TurnForAngle(angularSpeed, totalAngle)`

Create/Remove entities

Create a new entity of the specified category	`CreateNewEntityInCategory(categoryName)`
Create a new entity by copying the existing one	`CreateNewEntityByCopy(entityId)`
Remove the specified entity from the simulation	`RemoveEntity(entityId)`

The detailed description of the usage of each system API is out of the range of this dissertation. Readers are recommended to read the user's guide of *BehaviorSim* project for the complete set of system APIs. Besides these system APIs, users can also define their own methods and use them in defining behaviors.

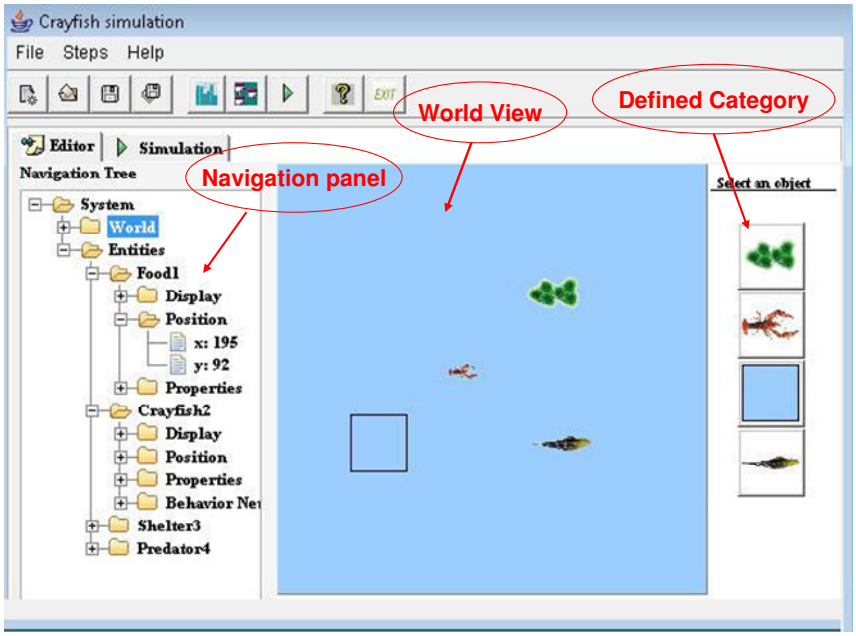
```
if (condition is true) {
        AddToTaskQueue (composite_task(...))
}
else {
        AddToTaskQueue(atomic_task1(...); atomic_task2(...))
        AddToTaskQueue(atomic_task3(...))
}
```

Figure 3.2 A task queue example.

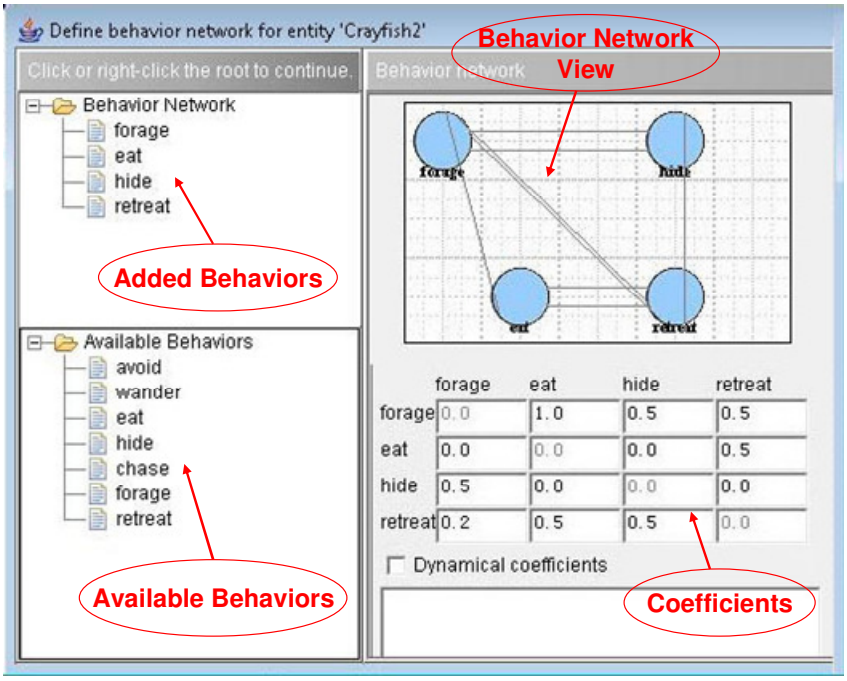
BehaviorSim also allows a user to set up a plan for a set of sequential tasks. This is achieved by a task queue. The pseudo-code in Fig. 3.2 shows an example of defining a task queue. The task queue can be set up in a hierarchical way (see the **composite_task**, which has sub-tasks). Multiple tasks can also be added into the task queue as *concurrent tasks* (executed in one time step), or *sequential tasks* (executed in different time steps). In the code, **atomic_task1** and **atomic_task2** are concurrent tasks.

3.4.2 Application Development Using *BehaviorSim*

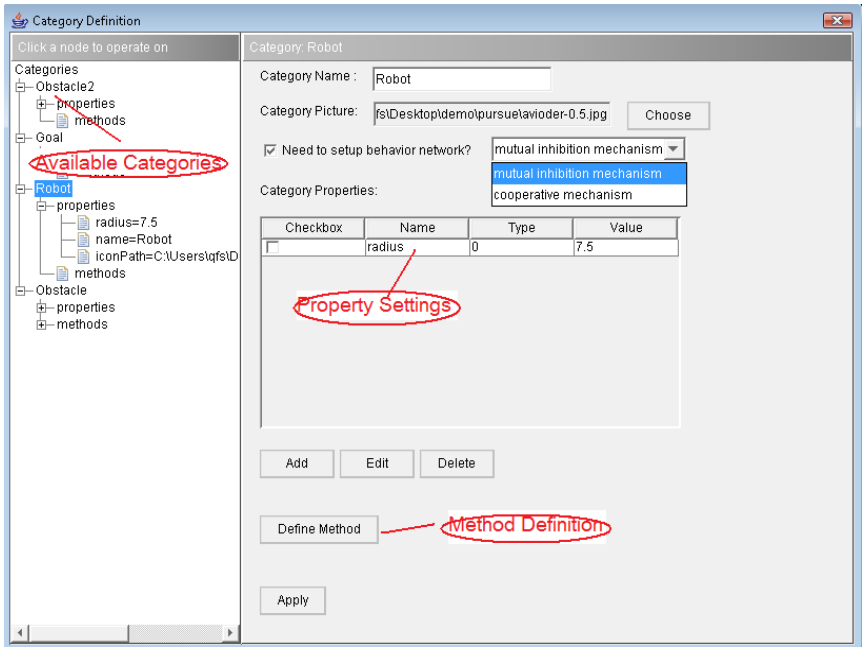
BehaviorSim is an open source project and the software can be downloaded from http://behaviorsim.sourceforge.net. It has four major windows (as shown in Fig. 3.3): a system editor window, a category definition window, a behavior network editor window, and a simulation window. The first three windows allow a user to setup the agent system, to define a category, and to define the behavior network for an agent, respectively. The simulation window allows a user to run simulations and see the results.



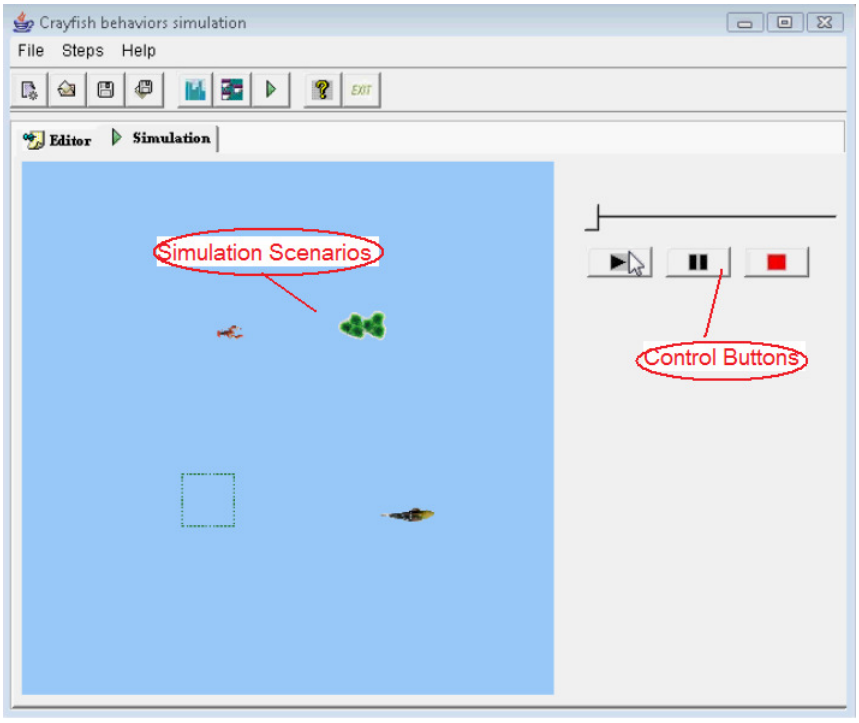
(a) System editor window



(b) Behavior network editor window



(c) Category definition window



(d) Simulation window

Figure 3.3 User Interface of BehaviorSim.

To develop a behavior-based agent system in *BehaviorSim*, the following four steps are generally followed:

- Define agent categories using the category definition window. Set up the common properties and methods for that category (see Fig.3.3 (c)).
- Set up the agents in the agent system using the system editor window. This can be done by dragging and dropping a corresponding category icon into the world (see Fig. 3.3 (a)).
- Define the behavior networks for agents using the behavior network editor window (see Fig. 3.3 (b)), where behaviors and the behavior network can be specified.
- Run simulations to see how agents behave (see Fig. 3.3 (d)). All the configurations are saved as a XML file that can be loaded and modified for the future use.

3.5 Pedestrian Crowd Simulation System

To facilitate group modeling, a pedestrian crowd simulation system is developed on top of *BehaviorSim.* This simulation system implements the framework proposed at Chapter 2 and serves as the basis in the development of group-related applications such as the dynamic grouping (see Chapter 5 for details). This section introduces the specification of the simulation system, followed by pedestrian agents' perception and behavior model.

3.5.1 Pedestrian Crowd Simulation System Specification

A pedestrian crowd simulation system is described as a tuple <*Environment*, *Crowd*>. *Environment* describes the simulation environment where pedestrian agents interact with each other. *Crowd* refers to a specific pedestrian crowd.

The simulation environment includes simulation primitives such as the obstacle information e.g. size, shape, and position. Obstacles are stationary entities which do not move during the simulation. The obstacles serve as the boundary of movements of pedestrian agents. Typical obstacles include columns and walls. Note that the simulation environment also includes the spatial information, such as size and shape, of pedestrian agents.

Besides simulation primitives, the simulation environment also provides a platform for pedestrian agents to communicate with each other. The communication is achieved by a set of functionalities provided by the environment. For example, the environment allows an agent to get a list of agents which are within the perception rage of the agent. Table 3.2 lists the major functionality provided by the environment. The major functionality can be roughly categorized into two categories. Each category contains a list of functions, one by each row.

Table 3.2 Major functionality of pedestrian crowd simulation system.

Get agent(s)	
Get the nearby agents which are	GetNearbyAgents(Agent)

within the perception of the specified agent	
Get members in the specified group	GetGroupMembers(GroupID)
Get the leader of the specified group	GetGroupLeader(GroupID)
Get the agent from other groups which is most similar to the specified agent	GetMostSimilarAgent(AgentID)

Check and generate position

Check whether the specified agent collides with others	LegalPostionToNearbyAgents(AgentID)
Check whether the specified agent collides with obstacles	LegalPositionToNearbyObstacles(AgentID)
Check whether the specified agent is inside the environment	InsideEnvironment(AgentID)
Generate a random position in the environment	GenerateRandomPosition(Randomer)

These functions and the system APIs as shown in Table 3.1 are used by pedestrian agents to get the information of the simulation environment e.g. the position of obstacles or other agents, and the distance to other agents.

In this dissertation, pedestrian agents are arranged into a set of social groups and each agent belongs to one and only one group. The number of agents including in a group is denoted as *group size*. Note that different groups may have different *group size*. And if an agent does not form group with others, the agent is considered to be a group consisting of itself. An important aspect of our framework is to study the relationships between agents within a group, i.e. *intra-group connections* and between different groups, i.e. *inter-group relationships*. Group modeling will be described in detail in Chapter 4. A pedestrian crowd is described as <*{Groups}*, *Inter-group Relationships*>, and a group is

described as <*{Agents}*, *Intra-group Connections*>. Here, {*Agents}* indicates a set of pedestrian agents which belong to the same group.

Each agent consists of a set of attributes and features. The attributes such as moving speed, moving direction, sociality, characterize the agent's physical or psychological states. Two important features of pedestrian agents are behavior model and perception model. An agent is described as <*ID, CurrentPosition, SpeedVector, Radius, PerceptionModel, BehaviorModel,…*>, where

- *ID* is the unique identification of the agent.
- *CurrentPosition* and *SpeedVector* specify the current position and speed vector, respectively. *SpeedVector* specifies both moving speed and direction of the agent.
- Each agent is of a circle shape whose radius is specified by *Radius.*

Note that an agent also has other attributes such as *GroupID, Role, GP,* and *GD* which are used in group modeling (see Chapter 4 for more details).

Besides a set of attributes, each agent is also featured with a perception model and a behavior model.

3.5.2 Agent Perception Model

The agent perception model *PerceptionModel* specifies the region which agents can perceive. It is critical for pedestrian agents to perceive the nearby obstacles and other pedestrian agents. In this framework, agents are equipped with the perception model as shown in Fig. 3.4.

Figure 3.4 Perception model of pedestrian agents.

Fig. 3.4 shows a widely applied perception model which is of elliptical shape. It is believed that this model can achieve more realistic perception since it matches the intuition that humans can perceive further in the front than in the side. In Fig. 3.4, the elliptical area represents the visible area. Only obstacles or agents in this area can be detected by the agent. The solid dot represents a pedestrian agent. The pedestrian's current moving direction is indicated as "*Direction*". *Dist1* and *Dist2* represent the maximum front and side distance for visibility respectively. *Angle* indicates half of the maximum visibility range the pedestrian can detect.

3.5.3 Agent Behavior Model

Each agent is featured with a behavior-based model which contains three behaviors: *RandomMove*, *Avoid* and *MaintainGroup*. These three behaviors compete with each other for the control of agent's movement using the *mutual inhibition* mechanism (Fig.3.1 (b)). Note that social groups are modeled in *MaintainGroup* behavior, the design of which will be described in Chapter 4.

Below follows the description of these behaviors, in particular, the description of the calculation of their excitation and the associated action.

1) Behavior: RandomMove

This behavior is used to simulate the random movement of each agent. The moving path is the shortest path from the agent's current location to the destination, computed through the *Dijkstra* algorithm. When a specific destination is reached, the agent will move to another destination which is generated randomly.

- *Excitation*: Ex = 0.6, which means that this behavior will be moderately excited.
- *Action*: If the agent is not at the destination area, it walks towards the destination according to the shortest path. Otherwise if it arrives at its destination, it will move to a new destination that is randomly generated.

2) Behavior: Avoid

This behavior is used to simulate the obstacle avoidance in the movement. When an agent is within a predefined minimum distance from the nearest neighbor agent or obstacle, it will stay away from it.

- *Excitation*: The excitation of this behavior is calculated through Eq. 3.5.

$$Ex = \begin{cases} \exp(1 - closestDistToObstacle/(1.5*Radius)) & \text{To avoid wall or column} \\ \exp(1 - closestDistToAgent/(2.5*Radius)) & \text{Otherwise} \end{cases} \quad (3.5)$$

In Eq.3.5, *closestDistToObstacle* is the distance from the agent to the surface of the closest obstacle to avoid, and *closestDistToAgent* is the distance from the agent to the center of the closest agent to avoid. Note that the closest obstacle or agent should be within the perception range of the agent. Otherwise, *closestDistToObstacle* and *closestDistToAgent* will be positive infinity. The constant factor 1.5 and 2.5 indicate that once the safety margin between the agent and the nearest wall or agent is less than half of *Radius*, this behavior will be excited. The exponential function indicates that as *closestDistToObstacle* or *closestDistToAgent* decreases, the more likely this behavior will be excited.

- *Action*: The separation and friction forces from the wall and the closest agent are applied on the agent and the acceleration (or deceleration) is calculated, the velocity is updated and the agent moves one step according to the direction indicated by the velocity.

3) Behavior: MaintainGroup

This behavior let agents maintain both the *intra-group connections* and the *inter-group relationships* during the simulation.

- *Excitation*: The excitation of this behavior is calculated through Eq. 3.6.

$$Ex = \begin{cases} \exp((Dist(myPos_i, GP_i) - DesiredDist)/3) & \text{For group members} \\ \exp((Dist(myPos_i, GP_i) - 20 * DesiredDist)/3) & \text{For group leaders} \end{cases} \quad (3.6)$$

In Eq. 3.6, *Dist* is used to calculate the *Euclidian* distance from agent *i* to its group position GP_i. $myPos_i$ is the position of agent *i*. *DesiredDist* is the distance which agent *i* wants to maintain from other members (check Chapter 4 for the description of *GP* and *DesiredDist*). The exponential function indicates that as *Dist* decreases, the more likely this behavior will be excited.

- *Action*: Calculate the vector to maintain both the *intra-group connections* and the *inter-group relationships* and carry out the movement guided by the vector.

Note that, the design of *RandomMove* and *Avoid* is simple. Our focus is the *MaintainGroup* behavior which models social groups (see Chapter 4 for details).

The *inhibitory coefficients* between these three behaviors are listed in Table 3.3.

Table 3.3 Inhibitory coefficients between pedestrian agents' behaviors.

	CasualMove	*Avoid*	*MaintainGroup*
CasualMove	0.0	-0.6	-0.6
Avoid	0.0	0.0	0.0
MaintainGroup	0.0	-0.6	0.0

As shown in Table 3.3, *Avoid* inhibits *CasualMove* and *MaintainGroup*, and *MaintainGroup* inhibits *CasualMove*, with a coefficient 0.6 which indicates an intermediate inhibition.

In each simulation cycle, the *excitation* of each of the three behaviors is calculated, followed by the calculation of *activation* of each behavior. The behavior with highest

activation level will be selected as the one controlling the agent and the corresponding action will be carried out.

CHAPTER 4

GROUP MODELING

4.1 Introduction

This chapter introduces the details of our *group model*. To facilitate group modeling, a pedestrian crowd is considered as a list of social groups, each of which is identified by a unique identification *GroupID* which is assigned automatically when the simulation starts. Each group contains of a list of agents. The number of agents including in a group is denoted as *group size.* Note that different groups may have different *group size*. There are two special cases regarding the *group size:* 1) *group size* is the number of agents in the crowd, i.e. the crowd only consists of one group, and 2) *group size* is 1, i.e., there is no social group in the crowd. As demonstrated later, *group size* has important effect on crowd behaviors.

One important aspect of the group model is to study relationships within the crowd. Each social group is studied from two aspects, i.e. *intra-group connections* and *inter-group relationship*. *Intra-group connections* are the relationships among members, e.g. likeness, familiarity, in the same group. In another word, *intra-group connections* represent the member-to-member influence within a group. *Inter-group relationships* represent the relationships among groups, e.g. following, in the pedestrian crowd. Clearly, *inter-group relationships* represent the group-to-group influence within a crowd. There are two types of relationships: *static* relationship and *dynamic* relationship. *Static* relationship

represents the relationship which is not changed during the simulation. The static relationship is usually found in stable social groups, e.g. the family units, where the relationship is not changed as the time passes by. While *dynamic* relationship represents the ever-changing relationship and it is usually found in unstable social groups, e.g. the task-based groups, where the relationship is temporarily formulated and distinguished after some time. The group model presented in this chapter studies *static* relationships. Chapter 5 presents an application of this group model, i.e. a dynamic group model which studies the *dynamic* relationships.

There are two roles of agents *Role* in a social group, *group leader* and *group member*. Each group has one and only one *group leader*. The rest of the agents in the group are *group members*. By default, the first agent (whose *ID* is *smallest* in its group) is the leader in the group. The *group leader* is considered as a "special" agent in the group because this is the only agent who could be influenced by agents from other groups due to *inter-group relationships* (more details are given later). A *group member* can only be influenced by other members (including the group leader) of the same group due to *intra-group connections*.

The strength of a relationship, e.g. the degree of likeness, is represented by a real number. The strength of both *intra-group connections* and *inter-group relationship* is indicated by a list of real numbers which are represented in two-dimensional matrices (see the following sections for details). *Intra-group connections* are specified by an intra-group matrix. *Inter-group relationships* are specified by an inter-group matrix. These two

influence matrices capture all the information needed to specify the group in a crowd. Note that the purpose of this *group model* is to simulate static social groups, i.e., both the intra-group and inter-group matrices are pre-specified by the user. In other words, this work assumes that social groups are formed when the simulation starts and they will not be changed during the simulation. Possible extensions such as dynamic grouping will be discussed in Section 4.7.

Our group model is implemented in the *MaintainGroup* behavior. The focus of this chapter is thus the *MaintainGroup* behavior and the calculation of its speed vectors. Each agent's maintaining group behavior is composed of two aspects of movements: *Aggregation* and *Following*, which allow an agent to maintain its desired intra-group connections and inter-group relationships. These two aspects of movements are represented by two speed vectors, *aggregation vector* and *following vector* respectively.

- *Aggregation* means an agent moves towards the center of the agents that are in the same group and have non-zero influences (as defined by the intra-group matrix) on this agent. This center is named the group position, denoted as *GP*, of this agent.
- For a group member, *Following* means the member heads towards the average moving direction of other group members who are in the same group and have non-zero influences on this member. This is similar to the *alignment* behavior (see Fig. 1.1 (b)) in Reynolds's work [54]. For a group leader, *Following* means that the leader follows the moving direction of an agent from a different group to maintain the inter-group

relationship. In both cases, the moving direction associated with *Following* is named the group direction, denoted as *GD*, of this agent.

GP and *GD* are two crucial parameters in the calculation of *aggregation* and *following* vector. As will be seen in Section 4.4, both *aggregation* and *following* vector can be calculated directly from these two parameters as well as the intra-group and inter-group matrices. Before introducing how to calculate these two vectors, Section 4.2 and 4.3 present the modeling of intra-group connection and inter-group relationship respectively. These two sections also present the calculation of *GP* and *GD* based on intra-group and inter-group matrices, and the position and speed vector of other group members. Section 4.5 and 4.6 present a case study which uses the group model to create social groups and to explore the effect of grouping on crowd behaviors, respectively. Section 4.7 discusses several possible extensions of the group model to support more features such as clustering, dynamic grouping and so on.

4.2 Modeling Intra-group Connections

4.2.1 Intra-group Matrix

The intra-group matrix is used to represent the member-to-member influence information within a group. In this group model, each group has an intra-group matrix, i.e. members within the group share the influence information. The sharing of this global information makes sense especially when *group size* is small. For the large groups, it may not be practical for all agents in the group to have such global information, i.e.,

the shared intra-group matrix. This is not considered in the group model. However, it is possible to extend the group model to let each member keep its own intra-group matrix.

An intra-group matrix is a two dimensional table where each element is a real number in the range of [0.0, 1.0] which represents the strength of intra-group connections. The number at row with *ID i* and column with *ID j*, denoted as *I(i, j)*, defines how much agent *i*'s movement is influenced by agent *j*. Since each *I(i, j)* has a value, the intra-group matrix specifies not only the network structure of influences among the individuals, but also the influence strength. For example, *I(i, j)* = 0.0 means agent *j* has no influence on agent *i*, and *I(i, j)* = 1.0 means agent *i* is fully influenced by agent *j*. An intra-group matrix with all elements being 0.0 represents the case that individuals of the same group have no influence to each other. Table 4.1 shows a sample intra-group matrix for a group having three pedestrians with *ID* 0, 1 and 2, among which Pedestrian_0 is the group leader.

Table 4.1 A sample intra-group matrix.

ID	Pedestrian_0	Pedestrian_1	Pedestrian_2
Pedestrian_0	N/A	0	0
Pedestrian_1	1	N/A	0
Pedestrian_2	0	1	N/A

As can be seen from Table 4.1, Pedestrian_1 is influenced by Pedestrian_0; Pedestrian_2 is influenced by Pedestrian_1; and Pedestrian_0 (the leader) is not influenced by other pedestrians. Because of these influence relationships, Pedestrian_2 follows Pedestrian_1, which in turn follows the Pedestrian_0 (the leader). As will be discussed later, this table defines a linear group.

Besides the intra-group matrix, three other parameters are used to describe how far pedestrians can stay away from each other within the same group: *SideDist*, *CenterDist*, and *DesiredDist.*

- *SideDist* is the maximum perpendicular distance from *GP* to the line indicated by the agent's current moving direction. Such a perpendicular distance is also called *side distance*.
- *CenterDist* is the maximum Euclidian distance from *GP* to the agent's current position. Such a Euclidian distance is also called *center distance*.
- *DesiredDist* is the desired distance from *GP* to the agent's current position. It is the maximum distance the individual wants to maintain during the movement. This is also called *desired distance*.

At every movement decision, the current *center distance* and *side distance* will be calculated. Only if *center distance* is greater than *desired distance*, the individual's maintaining group behavior will be triggered. During this process, the *center distance* and *side distance* are used to calculate the weight of the aggregation *vector* (see Section 4.4 for more details). They accelerate the aggregation movement, as long as the moving

speed does not exceed a predefined maximum speed. The smaller the *center distance* and *side distance*, the faster an agent will move towards the group center *GP*, and the more compact the group will be.

4.2.2 Calculation Of GP And GD For Group Members

GP is the group position an agent should move towards and *GD* is the average moving direction of other group members that have non-zero influences on the agent. Eq. 4.1 and 4.2 show how *GP* and *GD* are calculated based on the intra-group matrix and the positions and speed vectors of other group members. Suppose the intra-group matrix is labeled with *I(i, j)* where *i, j* is *ID* of agent *i* and *j*. N_i is the total number of group members that are in the perception range (see the perception model later) of agent *i*. Note that only those agents that belong to the same group of agent *i* are counted. $CurrentPosition_j$ and $SpeedVector_j$ are the current position and speed vector of agent *j*, respectively.

$$GP_i = (\sum_{j=1}^{N_i} I(i, j) * CurrentPosition_j) / N_i \qquad (4.1)$$

$$GD_i = (\sum_{j=1}^{N_i} I(i, j) * SpeedVecto\, r_j) / N_i \qquad (4.2)$$

Note that Eq.4.1 and Eq.4.2 only apply for group members. Generally, the greater the perception range, the more neighborhood members can be detected, thus, the more likely the computed *GP* and *GD* will be the global group center and group direction

(where all members of the same group are involved in the computation), and the faster the desired group will be formed. For a group leader *i*, GP_i and GD_i is the position and moving direction of some agent in other groups, respectively. For both group member and group leader *i*, the direction of the *aggregation vector* is the direction from *i* to GP_i, and the direction of *following vector* is the direction indicated by GD_i. The calculation of *GP* and *GD* for a group leader will be based on inter-group relationships, which is described in the next section.

4.3 Modeling Inter-group Relationships

4.3.1 Inter-group Matrix

Besides the member-to-member influence, different groups may also influence each other. For example, a group may follow other nearby groups during an emergency evacuation process because of the lack of objective evaluation of the emergent situations [40]. In this dissertation, this kind of inter-group relationship is specified by an inter-group matrix. Similar to intra-group matrices, the inter-group matrix is a two-dimensional table, which specifies the group-to-group relationships. Note that there is only one inter-group matrix for the whole crowd, while each group may be configured with different intra-group matrices.

Similar to the intra-group matrix, each element in the inter-group matrix is a real number in [0.0, 1.0] which represents the strength of inter-group relationships. The element at row with *GroupID i* and column with *GroupID j*, denoted as *E(i, j)* specifies

how much group *i* (specifically the group leader of that group) is influenced by the agents in group *j*. The number 0.0 means that the row group is not influenced by the column group even when the two groups are close to each other. The number 1.0 means the row group is fully influenced by the column group if the two groups are close to each other. Note that the situation of no inter-group relationships can be represented by setting all elements of the inter-group matrix to be 0.0. Table 4.2 shows a sample inter-group matrix for a crowd including four groups with *GroupID* 0, 1, 2 or 3, all of which fully influence each other (because all elements in the inter-group matrix have value 1.0).

Table 4.2 A sample inter-group matrix.

GroupID	Group_0	Group_1	Group_2	Group_3
Group_0	N/A	1	1	1
Group_1	1	N/A	1	1
Group_2	1	1	N/A	1
Group_3	1	1	1	N/A

Based on the definitions of intra-group matrix and inter-group matrix, one can see there are two ways to specify a crowd that is equivalent to having no group: 1) the whole crowd is one group and the intra-group matrix elements are all zero; 2) each group in the crowd has only one individual and the inter-group matrix elements are all zero.

4.3.2 Calculation Of GP And GD For Group Leader

As mentioned above, only the group leader is influenced by individuals from other groups. The goal of modeling inter-group relationships is to let each group leader follow the individual that has the greatest influences on the leader. To do this each individual's influence weight is calculated. Note that only the individuals, which are from different groups, are considered. In the group model, the calculation of influence weight is based on Festinger's *Social Comparison Theory* (see the work of [40]). The idea is to select the individual that has greatest similarity as the one that has greatest influence on the group leader. This similarity depends on not only the inter-group relationships between the two groups that the leader and the agent belong to but also the Euclidian distance between the leader and the agent.

Specifically, suppose the *inter-group matrix* is *E* and each element of the matrix is *E(G(i), G(j))*, where *G(i), G(j)* is *GroupID* of the groups which agent *i* and *j* belong to respectively. Suppose agent *i* is a group leader. Eq. 4.3 - Eq. 4.5 show the decision of leader *i*. For each agent *j* from another group in the perception range of leader *i*, the similarity between *i* and *j* is calculated using Eq. 4.3. As can be seen, the greater the inter-group relationship between two groups, and the closer the distance from *i* to *j* is, the more likely leader *i* will choose agent *j* to follow. Eq.4.4 and 4.5 show that the agent with the greatest similarity is selected as the one to follow. If no such agent exists, leader *i* will not follow any other agent.

$$Similarity_j = E(G(i), G(j)) * 100 / EuclidianDistBetween(i, j) \quad (4.3)$$

$$MostSimilarId = Maximum(Similarity_1, ..., Similarity_n) \quad (4.4)$$

$$PedestrianToFollow = \begin{cases} Pedestrian \text{ with } MostSimilarId & \text{If } MostSimilarId \text{ exists} \\ \phi & \text{Otherwise} \end{cases} \quad (4.5)$$

The specific procedure for leader i to find an agent (of other groups) with the greatest similarity is shown in Fig. 4.1.

```
procedure Find_Most_Similar_Pedestrian(Leader i)
1   PedestrianToFollow = φ;
2   Similarity = 0.0;
3   Temp = 0.0;
4   PedestrianList = all pedestrians that are in the perception range of leader i;
5   for each pedestrian a in PedestrianList
6     if G(a) ≠ G(i) then
7           Distance = EuclidianDistanceBetween(a, i);
8       Temp = E(G(i), G(a)) * 100 / Distance;
9       if Temp > Similarity then
10          Similarity = Temp;
11          PedestrianToFollow = a;
12  end for;
13  return PedestrianToFollow;
```

end Find_Most_Similar_Pedestrian.

Figure 4.1 Procedure of finding the most similar pedestrian.

The procedure shows that only nearby pedestrians who belong to the different group are considered as candidates. Leader *i* selects the one with greatest similarity value to follow. Once a valid *PedestrianToFollow* ($\neq \phi$) is found, GP_i and GD_i for leader *i* is the position and moving direction of *PedestrianToFollow* respectively. Otherwise, leader *i* will not follow any pedestrian from other groups.

4.4 Calculation Of Agent Motion Parameters

The focus of the group model is to calculate the two vectors, the *following vector* and *aggregation vector*, which guides agents to maintain both intra-group connections and inter-group relationships. This section shows how to calculate the two vectors with the value of *GP* and *GD* (calculated in previous two sections). Fig.4.2 shows a scenario where the two vectors of agent *s* are presented. Without loss of generality, suppose agent *s* is moving horizontally from right to left. The elliptical area represents a perception model of each agent (see section 2.3 for the perception model). Pedestrian *c*, *d*, and *e* are neighbors in the perception range of *s*, where agent *c*, *d*, *e* and *s* belong to the same group. The black solid circle is denoted as the group position GP_s that *s* should move towards. The *side distance* and *center distance* are denoted as *sd* and *cd* respectively.

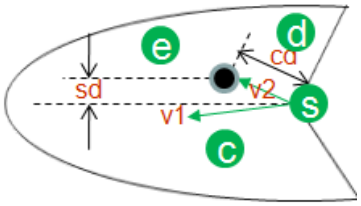


Figure 4.2 Brief description of the following and aggregation vector.

Let v_1 and v_2 be *following vector* and *aggregation vector* respectively. There are two assumptions. One assumption is that neighbors in perception range of s are c, d, and e, which have non-zero influence on s. The other assumption is that the direction of v_1 is the average moving direction of those neighbors.

Eq. 4.6, Eq. 4.7, and Eq. 4.8 show the calculation of v_1. *Speed* is the current moving speed of agent s, and a is the direction indicated by GD_s. For a group leader, it follows the moving direction of *PedestrianToFollow*; while for group members, they follow the average moving direction of nearby neighbors in perception range.

$$v_1 = factor * < Speed * \cos(a), Speed * \sin(a) > \tag{4.6}$$

$$factor = \begin{cases} T & s \text{ is a group leader} \\ 1.0 & s \text{ is a group member} \end{cases} \tag{4.7}$$

$$T = \begin{cases} 0.0 & PedestrianToFollow = \phi \\ 1.0 & \text{Otherwise} \end{cases} \tag{4.8}$$

The calculation of v_2 is shown in Eq. 4.9 - Eq. 4.12. a is the direction from s to GP_s. If s is a group leader, *factor* is calculated through Eq. 4.11. Otherwise, *factor* is calculated through Eq. 4.10. The Euclidian distance from s to *PedestrianToFollow* is denoted as *dist*.

For each agent, it moves towards *GP* and tries to keep the distance from the agent to *GP* within *DesiredDist*. For group members, they also move towards each other to satisfy the predefined *CenterDist* and *SideDist.*

$$v_2 = factor * < Speed * \cos(a), Speed * \sin(a) > \quad (4.9)$$

$$factor = \begin{cases} cd / CenterDist & cd > CenterDist \\ sd / SideDist & cd \le CenterDist \text{ and } sd > SideDist \\ 0.0 & \text{Otherwise} \end{cases} \quad (4.10)$$

$$factor = \begin{cases} E * 20 * DesiredDist / dist & PedestrianToFollow \neq \phi \\ 0.0 & \text{Otherwise} \end{cases} \quad (4.11)$$

$$E = E(G(s), G(\ PedestrianToFollow)) \quad (4.12)$$

$$v = v1 + v2 \quad (4.13)$$

The overall speed vector v to govern the maintaining group behavior is the vector addition of v_1 and v_2 as shown in Eq. 4.13. For an agent s, it will try to move towards GP_s to keep within *DesiredDist* (not shown in Fig. 4.2), and try to follow the direction GD_s. In this way, both the intra-group connections and the inter-group relationship can be maintained.

4.5 Case Study - Simulating Social Groups

This section describes a case study of developing three groups using the developed framework. Three social groups, a linear group, a leader-follower group, and a mixed group, are demonstrated.

In the linear group, members of the same group move in a line formation. Each group member follows another member. The intra-group matrix *I* is defined in Eq. 4.14,

which indicates that member *i* will follow the member *j* with an *ID* which is one less than the *ID* of member *i*.

$$I(i,j)=\begin{cases} c & i!=0 \text{ and } j=i-1 \\ 0.0 & \text{Otherwise} \end{cases} \quad (4.14)$$

In the leader-follower group, members follow the group leader during the movement. The intra-group matrix *I* is defined in Eq. 4.15, which indicates that all members will move close to the leader.

$$I(i,j)=\begin{cases} c & i!=0 \text{ and } j=0 \\ 0.0 & \text{Otherwise} \end{cases} \quad (4.15)$$

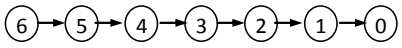
In Eq. 4.14 and Eq. 4.15, *c* is a positive real number defined in (0.0, 1.0], which indicates the member-to-member influence strength. In both groups, the group leader finds the path and moves forward. Fig. 4.3, Fig. 4.4, and Fig. 4.5 show three groups, linear, leader-follower and mixed respectively. Seven agents with *ID* from 0 to 6 are included in each group. The agent with *ID* 0 is the group leader (displayed with red label). For each group, the intra-group matrix and a simulation scenario are given. Blank cells in intra-group matrices indicate the corresponding elements being 0.0. These three figures show how to use an intra-group matrix to present the desired group and

use the proposed framework to simulate it. Groups are created and simulated in three simple steps.

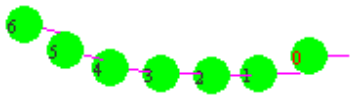
Step 1: Express the group as a network structure (Fig. 4.3-4.5(a)).

Step 2: Express the relationships between network nodes in the intra-group matrix (Fig. 4.3-4.5(b)).

Step 3: Simulate the group (Fig. 4.3-4.5(c)).

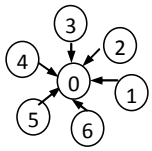


ID	**0**	**1**	**2**	**3**	**4**	**5**	**6**
0							
1	1						
2		1					
3			1				
4				1			
5					1		
6						1	

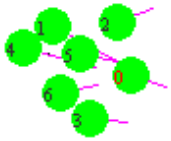


(a) Linear group shape (b) Intra-group matrix (c) Simulation scenario

Figure 4.3 A linear group.



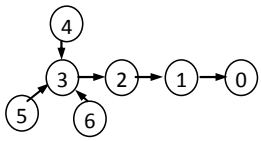
ID	**0**	**1**	**2**	**3**	**4**	**5**	**6**
0							
1	1						
2	1						
3	1						
4	1						
5	1						
6	1						



(a) Leader-follower group shape (b) Intra-group matrix (c) Simulation scenario

Figure 4.4

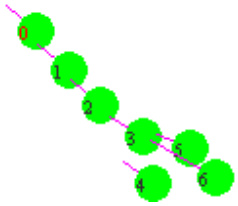
Figure 4.4 A leader-follower group.



(a) Mixed group shape

ID	**0**	**1**	**2**	**3**	**4**	**5**	**6**
0							
1	1						
2		1					
3			1				
4				1			
5				1			
6				1			

(b) Intra-group matrix



(c) Simulation scenario

Figure 4.5 A mixed group.

Note that the mixed group consists of both linear and leader-follower subgroups. In Fig. 4.5, agents with *ID* 0, 1, 2, and 3 form a linear subgroup where agent 3 follows agent 2, agent2 follows agent 1, and agent 1 follows agent 0. While agent 3, 4, 5, and 6 form a leader-follower subgroup where agent 4, 5, and 6 follow agent 3.

4.6 Case Study – Effect of Grouping On Crowd Behaviors

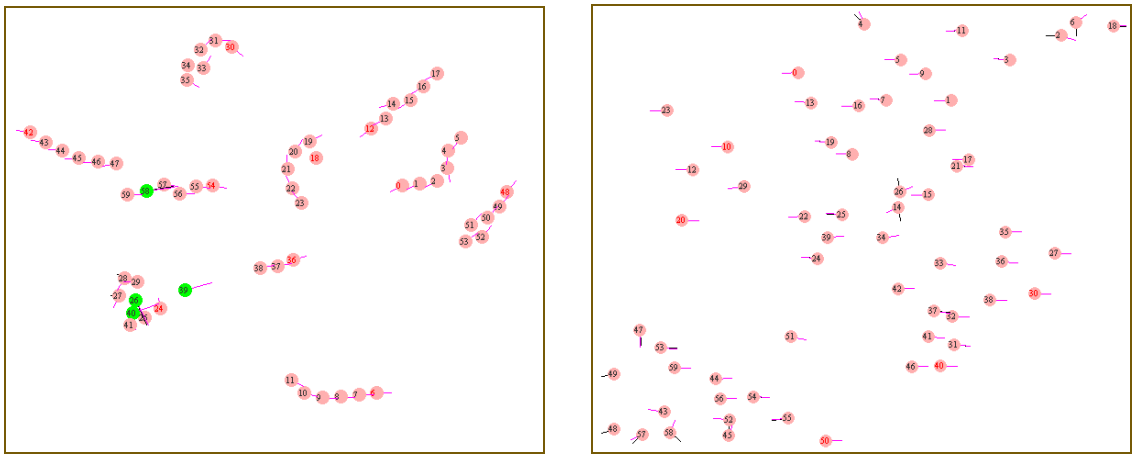
Once we have created social groups, we can study the effect of various factors on crowd's grouping behavior. Specifically this section explores how different parameters, e.g. the strength of intra-group connections, the strength of inter-group relationships, and *group size,* affect the crowd movement. Two typical groups, a linear group and a leader-follower group, are considered (see the previous section for details). Moreover, the effect of the strength of intra-group connections and the strength of inter-group relationships on pedestrian flow will also be explored.

4.6.1 Effect of member-to-member influence strengths on crowd behavior

To show the effect of member-to-member influence strengths, a linear group under four intra-group matrices is considered: one with all elements in the intra-group matrix being 1.0, one with all elements in the intra-group matrix being 0.7, with all elements in the intra-group matrix being 0.4, and the other with elements being 0.1. The effect of intra-group influences is measured by the average distance, from group members of the first group to its group center *GP*. The crowd contains 10 groups, each of which has 6 agents. To only show the effect of intra-group influence strengths, there is no inter-group relationship between groups, i.e. elements of inter-group matrix are set to zero.

Fig. 4.6 (a) shows a simulation scenario of a full member-to-member influence on crowd behavior. The average distance from members of the first group to its group center, for the four intra-group matrices, is shown in Fig. 4.7. For all four intra-group matrices, the average distance is measured over same simulation time interval. In Fig.4.7, "*I=C*" represents the case that all elements in the intra-group matrix are *C*. As can be seen, intra-group influence strength affects the crowd movement. The greater the intra-group influence strengths, the smaller the average distance since the greater the intra-group influence weights, the more compact the group will be, thus the average distance is less than that of smaller intra-group influence strengths. For example, the average distance of the case "*I=0.7*" is less than that of the case "*I=0.4*". For the comparison purpose, a simulation of the crowd without social groups is shown in Fig. 4.6 (b). Correspondingly, the average distance for the crowd without social groups (*I=0.0*) is also shown in the topmost curve in Fig. 4.7. In this case, the average distance is

the mathematical average of the distance from the first 6 members to the center of these members. As can be seen in Fig. 4.7, the crowd without social groups has greater average distance than the crowd with social groups, since without social groups, each individual moves randomly and no specific group will be formed.



(a)A simulation scenario for linear group with I=1.0 (b) Crowd simulation without social groups

Figure 4.6 Simulation scenarios for pedestrian crowds with and without social groups.

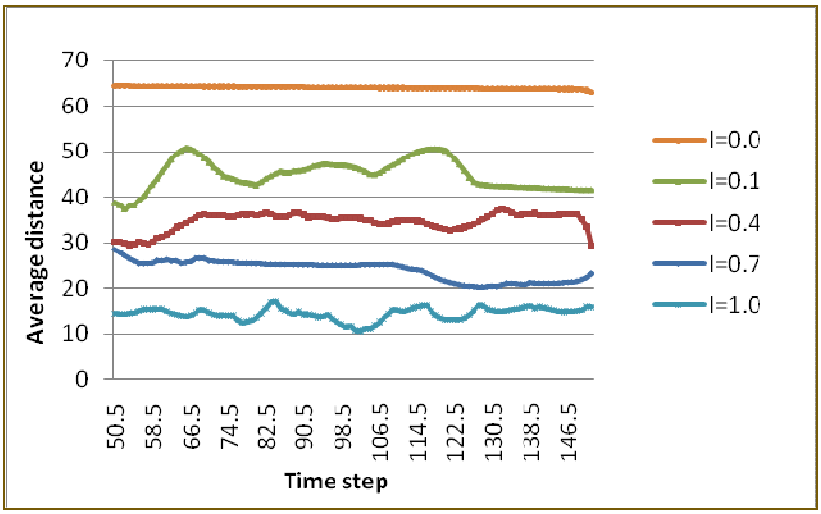


Figure 4.7 Average distance from members of the first group to its group center.

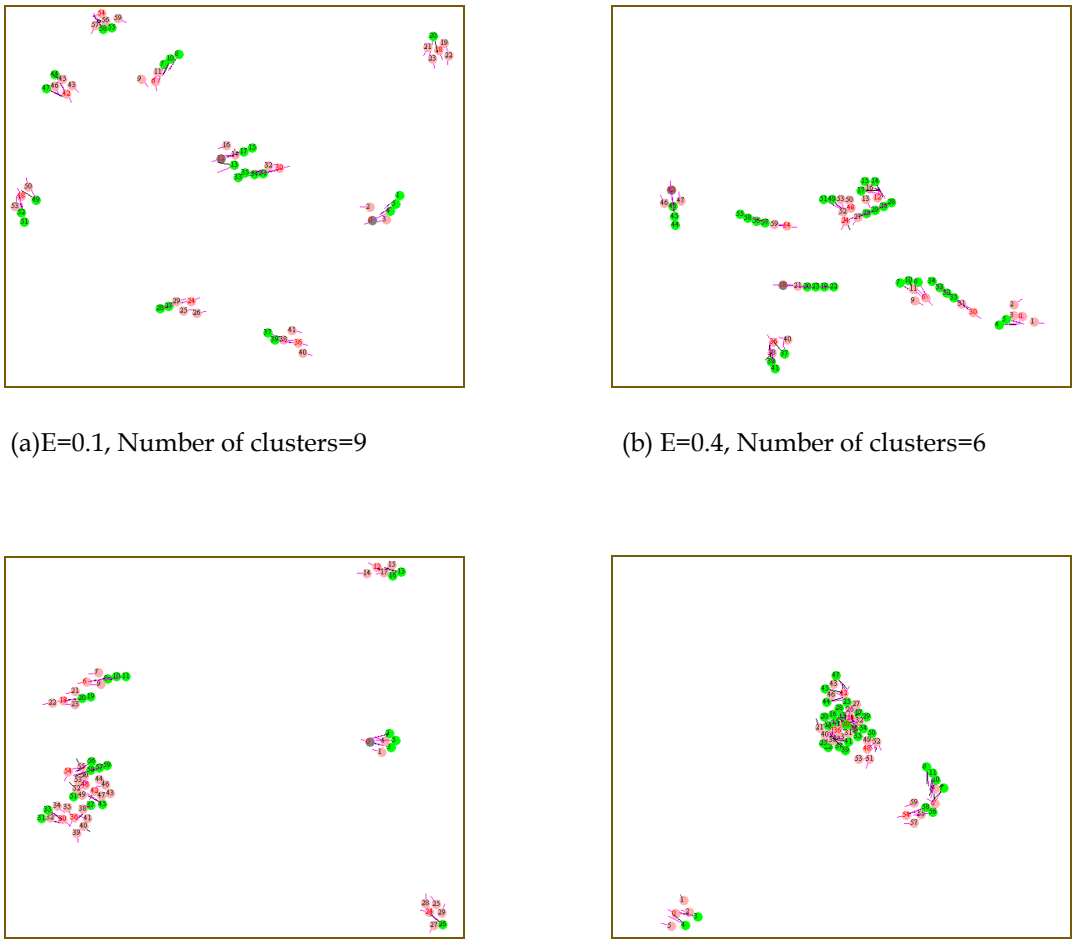
4.6.2 Effect of group-to-group influence strengths on crowd behavior

Besides intra-group connections, inter-group relationships also have effect on crowd behavior. The effect is shown in Fig. 4.9 where four inter-group matrices: one with all elements in the inter-group matrix being 1.0, one with all elements in the inter-group matrix being 0.7, with all elements in the inter-group matrix being 0.4, and the other with elements being 0.1, are studied. The crowd contains 10 groups, each of which has 6 agents. Each group takes a leader-follower group with all elements of the intra-group matrix are 1.0 (c=1.0 in Eq. 4.15).

The effect of inter-group relationships is measured by the number of clusters at a specified simulation time. The rationale is that, the greater the inter-group relationship, the more likely two groups will move together and a larger cluster will be formed. Thus the number of clusters can represent the strength of group-to-group relationships. Each cluster could contain several groups. The calculation of the number of clusters is based on the QT (quality threshold) clustering algorithm presented in the work of [55] where the closest distance between two clusters is no more than the quality threshold, which is 8 times the pedestrian radius R. Procedure of the modified QT clustering algorithm is shown in Fig. 4.8. Once pedestrian j is added into a cluster, the members (including the leader) of the group which agent j belongs to are also added into the same cluster since group members generally stay together during the movement.

```
procedure QT_Modified_Clust(G, d)
1        if |G| ≤ 1 then output G, else do
2          for each i ∈ G
3             set flag = TRUE; set A_i = {i}; /* A_i is the cluster started by i*/
4             while flag= TRUE and A_i ≠ G
5                find j ∈ (G - A_i) such that diameter(A_i ∪ {j}) is minimum;
6                if diameter(A_i ∪ {j}) > d
7                        then set flag = FALSE;
8                        else  /* Also add all members of the group to cluster A_i*/
9                             find all members (including the leader) T of the group
which j belongs to;
10                            for each t ∈ T
11                              if t∉  A_i then set A_i = A_i ∪ {t};
12                            end for;
13            end while;
14         end for;
15         identify set C ∈ {A_1, A_2, …, A_|G|} with maximum cardinality;
16       output C;
17         call QT_Modified_Clust(G - C, d);
end procedure QT_Modified_Clust.
```

Figure 4.8 Algorithm QT_Modified_Clust takes as input the set *G* of pedestrian positions and a diameter threshold *d*, and returns a set of clusters.



(a)E=0.1, Number of clusters=9

(b) E=0.4, Number of clusters=6

(c)E=0.7, Number of clusters=5

(d) E=1.0, Number of clusters=3

Figure 4.9 Effect of inter-group relationships on crowd behavior.

In Fig. 4.9, "*E=C*" represents the case that all elements in the inter-group matrix are *C*. We can see that, as the inter-group relationship becomes greater, the number of clusters is decreasing. Since the greater the inter-group relationship, the more likely a group leader will follow other groups, and the closer the group leader will move

towards other groups, as well as the higher probability that a larger cluster will be formed. For example, when the inter-group relationship is 1.0 (see Fig.4.9 (d)), many groups move together and a large cluster is formed, and the number of clusters is much less than that of other three cases.

4.6.3 Effect of group size on crowd behavior

As indicated by the work of [41], group size may also affect the crowd movement. To only explore the effect of different group sizes, the inter-group relationship E is set as 0.0 and intra-group connections I is set as 1.0. The effect of group sizes is measured by the number of clusters which is calculated in the same way as that in previous section. Fig. 4.10 presents a simulation scenario of a grouped crowd where group size is 10. The number of clusters for different group sizes is shown in Fig. 4.11.

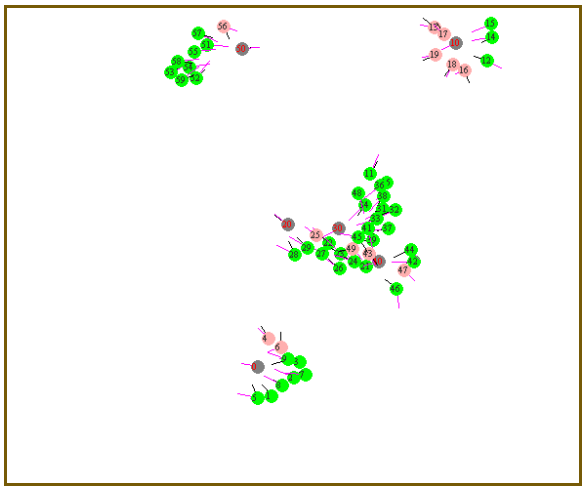


Figure 4.10 Clusters forming for groups of size 10.

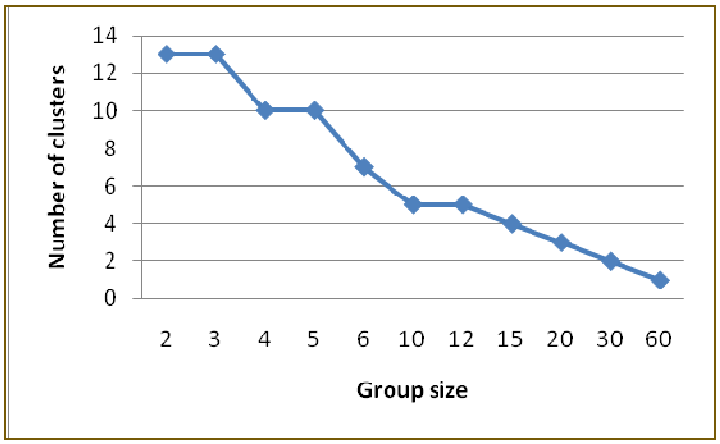


Figure 4.11 Number of clusters for different group sizes.

As can be seen, the larger the group size, the fewer the formed clusters, because it is more likely that pedestrians will follow each other. When the group size is large enough (greater than or equal to 12), the number of clusters is same as the number of groups, since pedestrians in the large group cannot move so freely as that in the small groups, and the probability that pedestrians follow each other is less than the small groups.

4.6.4 Effect of grouping on pedestrian flow

Designing experiments for pedestrian crowd simulation is a challenging task, since many input factors and output responses, and their relationships should be considered. An interesting work related to experimental design, as well as the identification of the process variables of interest, is proposed by Daamen et al. [56]. To simplify the experiments, only the effect of intra-group connections, inter-group relationships and

group sizes, on one of two principal characteristics of pedestrian movement, the flow is explored.

Pedestrian flows are important in the design of pedestrian facilities, such as shopping mall, bus station, museum, and so on. The simulation includes 60 agents which are situated in a circular rectangle-shaped hallway environment with the size 600 in length and 200 in width. A simulation scenario of the environment is shown in Fig.4.12, where the lane width is 50.

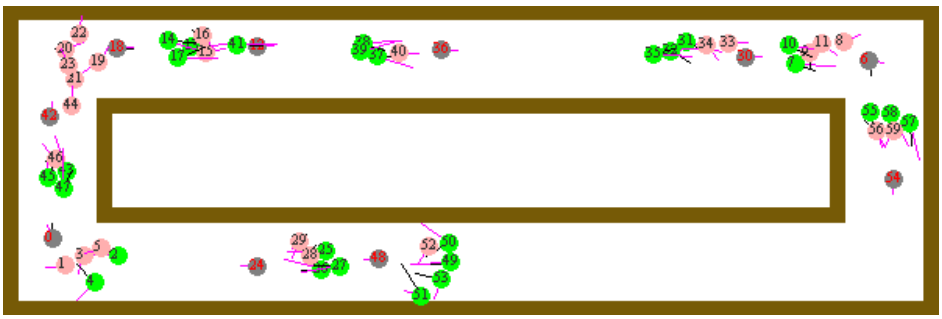


Figure 4.12 A circular rectangle-shaped hallway environment.

To calculate the flow, a virtual "gate" is defined in the hallway (in the middle of top most lane), and monitored agents that move through them during a specified simulation time interval. Similar to the work of [39, 57, 58], the flow is then calculated as the number of agents passed by the "gate" divided by the length of the simulation interval. Two experiments are designed to show the effect of intra-group connections, inter-group relationship, and group sizes on pedestrian flow.

One experiment explores the relationship between intra-group connections/inter-group relationships and pedestrian flow. There are four intra-group or inter-group matrices for the linear group: one with all elements in the inter-group matrix being 1.0, one with all elements in the inter-group matrix being 0.7, with all elements in the inter-group matrix being 0.4, and the other with elements being 0.1. Each group contains 6 agents. To explore the effect of intra-group connections, the inter-group relationships E is set to 0.0 to eliminate the effect of group-to-group relationships on crowd behavior. While in exploring the effect of inter-group relationships, elements of the intra-group matrix I are set as 1.0. Fig.4.13 shows the pedestrian flow under different intra-group or inter-group matrices. The upper curve represents the relationship between pedestrian flow and inter-group relationships. The other curve represents the relationship between pedestrian flow and intra-group influences. The pedestrian flow decreases as the intra-group influence strengths or the inter-group relationships decrease. The smaller influence strengths or relationships, the less desire an agent will follow others, thus the fewer agents passing the "virtual" gate.

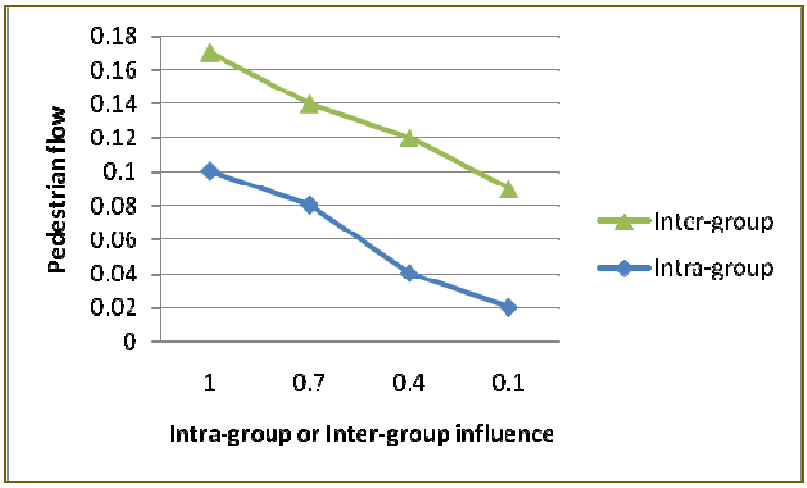


Figure 4.13 Pedestrian flow under different inter-group or intra-group matrices.

The other experiment explores the relationship between group sizes and the pedestrian flow. The inter-group relationship *E* is set as 0.0 and intra-group connections *I* is set as 1.0. Fig. 4.14 shows the effect of group sizes on pedestrian flow under the leader-follower group shape. X and Y axis represent group size and the corresponding pedestrian flow respectively.

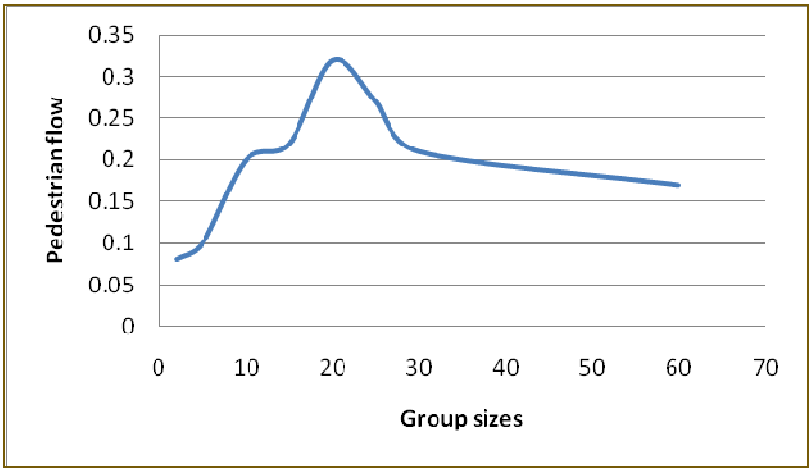


Figure 4.14 Pedestrian flow under different group sizes.

When the group is not so large, the pedestrian flow increases as the group size increases. Otherwise, the pedestrian flow decreases as the group size increases. It is because, when the group is not so large, the larger the group sizes, the more agents will pass the "virtual" gate thus the flow becomes larger. However, when the group becomes very large, more effort will be needed on the group formation and obstacle avoidance, and the agents passing the "virtual" gate are fewer than those of the smaller group sizes.

The two experiments show that the pedestrian flow decreases as the intra-group influence strengths or the inter-group relationships increase. The experiments also show that when the group is not so large, the pedestrian flow increases as the group sizes increase. Otherwise, the pedestrian flow decrease as the group sizes increase.

4.7 Discussions

In the framework, *MaintainGroup* is one of the three behaviors. An agent maintains its group only when the *MaintainGroup* behavior is selected. This may result in situations such as the *MaintainGroup* behavior is not selected thus causing agents do not maintain their groups. For example, in the very crowded environment, a group can be separated by members from other groups. In such situation, collision avoidance will have a higher priority which makes the agent avoid collision with others, rather than maintain its group. One of extensions is thus to let an agent maintains its groups while avoids the approaching obstacles and other agents.

Built on the current work, several other extensions can be developed for more advanced group modeling.

First, the current model assumes a static group (as specified by the intra-group and inter-group matrices) that is maintained through the whole simulation. It also assumes a preselected leader that is not dynamically replaced by others. In other words, the current work does not concern how groups are formed and how they will be dynamically changed. Thus another extension of the current work is to support dynamic group formation and dynamical change of the groups. This extension is illustrated in next chapter which shows an application of the developed group model, i.e. a dynamic group model which studies the dynamic grouping behavior in pedestrian crowd simulations.

Group formation decides who belong to the group and the relationship between members. Leader selection is part of that decision too. In the current model, without loss of generality, the leader is preselected as the first individual of a group. In more advanced simulations the leader might be changed during the simulation. For example, the leader can be replaced by the member who is more familiar with the surroundings when an emergent situation happens.

During the simulation, agents' groups can also be dynamically changed. For example, a "clustered" group can be dynamically changed to a "linear" group when a group approaches a narrow entrance of a building. Besides these factors, groups can also be dynamically formed and/or evolved by the change of group members and

intra-group connections. For example, a group member may leave the group and join another group; groups may be dynamically divided into multiple groups or combined together into a single group. In all these situations, the computation of the intra-group matrix and inter-group matrix depends on the desired relationships among group members that are application specific.

To extend the current work to support these kinds of dynamic grouping features, a dedicated context modeling layer of social or psychological models, such as the Five Factor Model (see [36-38] for more details) can be used to support the decision of group formation and dynamic groups (see Fig.2.1, the work of [59], and Chapter 5 for details). This layer calculates the intra-group and inter-group matrices and updates them in each agent. At the bottom layer, based on these matrices, an agent computes its group position and group direction in the same way as described before.

Another extension of the current work is to extend the three-level structure of crowd, crowd, group, and individual, considered in the current model to include more levels in a hierarchical manner. For example, a *cluster* level can be added on top of the group level: a crowd can include multiple clusters, each of which includes multiple groups. In this case, the model can be extended to have not only inter-group/intra-group matrices, but also inter-cluster/intra-cluster matrices.

Future extension will also include improvement of the performance of large-scale group-based crowd simulations. By adding the group semantics, crowd simulation would take extra memory space to store intra-group and inter-group matrices. Also,

group-based crowd simulations would be slower than those without social groups since more logics are involved. The performance might be possibly improved on the basis of the work of exploiting the spatial-temporal heterogeneity existing in pedestrian crowds [60] which is derived from the work of [61]. The work of this extension is carried out in Chapter 6 which develops an efficient discrete event based simulation engine by exploiting the crowd system's heterogeneity resulting from agents' different moving speeds.

CHAPTER 5

DYNAMIC GROUPING USING THE FRAMEWORK

5.1 Introduction

Chapter 4 introduces our group model which studies static groups, i.e. *intra-group connections* and *inter-group relationship* are not changed during the simulation. In another word, the group model does not concern how groups will be dynamically changed. It is necessary to simulate the dynamic nature of social groups since dynamic groups commonly exist in our daily life. For example, a "clustered" group can be dynamically changed to a "linear" group when a group approaches a narrow entrance of a building. It is a usual case that a group member may leave the group and join another group; groups may be dynamically divided into multiple groups or combined together into a single group. Groups are dynamically changed due to various factors such as the spatial distance (i.e. Euclidian distance), similar goal (i.e. evacuation from emergent situation), social proximity (i.e. family members), and so on.

This chapter studies the dynamic grouping behavior in agent-based pedestrian crowd simulations. A dynamic group model is developed by using our proposed framework. One assumption of this model is that initially each agent belongs to a group which consists of itself only. This model focuses on simulating dynamic group formation, i.e. a group member may leave the current group and join another group; a group member may leave the current group and start with a new group of itself; a

group member may also stay with the current group. The dynamic group formation is achieved through a two-step procedure, agent-to-group interaction and agent-to-agent interaction. In every simulation time step, agent-to-group interaction represents the procedure where an agent decides which group to follow, and agent-to-agent interaction represents the procedure where the agent decides which agent (from the selected group) to follow. Note that, if no group is selected to follow, an agent will start with a new group of itself.

To create a realistic simulation, the dynamic group formation is driven by artificial intelligence and social theory which are included in a two-tiered context model. Artificial intelligence allows pedestrian agents to behave adaptively in the ever changing environment. However, there is little work integrating the effect of artificial intelligence in the study of social groups. This chapter utilizes one of artificial intelligence theory – utility theory to capture agents' preferences in movement decision taken in the group-to-agent procedure based on the assumption that agents will act rationally even in a ever-changing environment (see the following sections for details). This chapter also utilizes one of social theories – social comparison theory to model the agent-to-agent procedure. These two theories are included in the two-tiered context modeling layer which captures the dynamics of both group-to-agent and agent-to-agent interactions. Such dynamics are specified through a set of predefined mathematical formulas which are designed according to the two theories. At every time step, the

formulas are evaluated, the results of which are used to drive the dynamics of the intra-group connections and inter-group relationships.

Experiments show that the developed dynamic group model can be used to simulate dynamic group formation and different types of social groups can be simulated dynamically.

5.2 Context Modeling

In this dynamic group model, content modeling layer incorporates two theories – Utility theory and Social comparison theory which allows pedestrian agents to adapt to the external environment.

Utility theory provides a formal framework for specifying the preferences, or utilities, of agents' potential actions under uncertain worlds. It is an important component in decision theories which assume that a person, even under a dangerous situation, can still make rational decisions [62] (this is also the assumption of the dynamic group model).

The dynamic grouping behavior of pedestrian agents can be modeled with utility theory. Pedestrian crowd simulation is often featured with an ever changing and complex environment due to the non-linear interactions among pedestrians. In such environment, the adaptability is critical for pedestrians to behave rationally to create realistic simulations. In this dynamic group model, the adaptability refers to the capability of performing appropriate group behavior at different time steps. There are

three types of group behaviors including staying with the current group, following another group (i.e., leave the current group), and starting a new group. Staying with the current group indicates that a pedestrian agent does not want to change its group profile. Following another group indicates that a pedestrian agent changes to a different social group. Starting a new group indicates that a pedestrian agent forms a new group which consists of itself only at the current time step. The decision of which group behavior to perform is based on comparing the degree of desirability of the three group behaviors. The desirability is specified through a set of utility functions. At each time step, utility functions are evaluated and the returned values are compared. Pedestrian agents perform the group behavior which has a greater desirability value. If two values are the same, agents stay with their current group.

The other theory used in modeling group behaviors is Festinger's Social comparison theory. The basic idea is that, after the decision among following another group, staying with the current group, and starting a new group, has been made, a pedestrian agent follows a member from the target group. The member followed by the pedestrian is the one which is most similar with the pedestrian. The similarity is calculated according to Eq 4.3 and 4.4. Note that, there are many strategies of selecting which group member to follow. For example, the pedestrian would follow the predecessor of the selected member if the selected pedestrian has a predecessor (the pedestrian followed by the selected member). As demonstrated in Section 5, different strategies lead to different dynamic groups, such as the leader-follower and linear group. Grouping has significant

effect on crowd behaviors (see the work of [13] for details on group modeling and the effect of grouping on crowd behaviors).

Utility theory and social comparison theory form a two-tiered context modeling layer of the dynamic group model as illustrated in Fig. 5.1, where the top and bottom rectangles indicate the top and bottom tier, respectively. The top tier models agent-to-group procedure on the basis of utility theory. The bottom tier models agent-to-agent procedure which is based on social comparison theory.

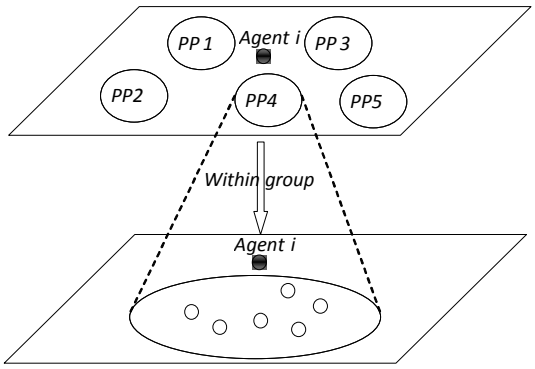


Figure 5.1 A two-tiered context modeling layer.

Fig. 5.1 illustrates the dynamic grouping behavior of agent *i* at some time step *t*. The solid dot represents agent *i*. Circles *PPi* (i=1-5) indicates the nearby groups which agent *i* can perceive (the nearby groups are detected through agent's perception model, see Fig.2.2 for details). Assume at the first step of agent-to-group procedure, agent *i* decides to follow group PP4. As mentioned above, the decision is based on comparing the returned values of utility functions. Agent *i* performs the group behavior whose utility

function has greatest returned value. Then agent *i* goes to next step of agent-to-agent procedure which decides which member (from *PP4*) to follow. The outer circle in the bottom rectangle indicates the selected group *PP4*. The dots inside the outer circle represent members in the group *PP4* which agent *i* can perceive.

In Fig.5.1, the top tier provides the grouping context for the bottom tier to execute the second step. The grouping context includes the information of its members, e.g. the position which provides useful information for agents to decide the most similar counterpart (see details in the following sections). If agent *i* decides to stay with its current group (not shown in Fig. 5.1), the second step is fulfilled between agent *i* and other members in the same group as agent *i*. Otherwise, if agent *i* decides to start a new group, it will not go through the second step of the agent-to-agent procedure.

Note that, it is possible to create the context modeling layer using other theories or models. Fig.5.1 only shows one context model on the basis of utility theory and social comparison theory. The purpose is to demonstrate how to create a context modeling layer which can be used to model agents' dynamic grouping behavior. It is our belief that this context model can produce realistic grouping behaviors.

5.3 Dynamic Group Modeling

In order to model agent-to-group and agent-to-agent interactions, several important parameters are introduced as follows.

- U_f and U_s represent an agent's utility/preference of joining another group and staying with the current group, respectively. Note that if both U_f and U_s are less than a threshold $U_{threshold,}$ the agent will start a new group which consists of itself.
- *Sociality* indicates how social an agent is. The greater the value, the more likely the agent will join another group. *Sociality* is a normalized real number in the range of [0, 1].
- *Similarity* represents agents' similarity value, which is bounded by a predefined minimum value *Smin* and a predefined maximum value *Smax*.

5.3.1 Modeling agent-to-group interactions

The agent-to-group interaction decides which group to follow at the current time step. As discussed above, an agent can join another group, stay with the current group, or start a new group. The decision is made through utility theory. The basic idea is that if both U_f and U_s are less than a threshold $U_{threshold,}$ the agent will start a new group which consists of itself only. Otherwise, the agent will join another group if $U_f > U_s$ and stay with its current group if $U_f <= U_s$.

The calculation of U_f and U_s for agent *i* is described in Eq.5.1-Eq.5.5. *c* and *DesiredDist* are constant numbers. *Duration* is the number of time steps agent *i* has stayed with its current group. It will be reset to 0 once the agent joins another group or starts with a new group. *Threshold* indicates the maximum number of steps within which agent *i* can join other groups. *c* is a constant number. Eq. 5.1 shows that the

distance between agent *i* and the nearby group GP_j is the distance between *i* and the closest agent *j* (belonging to the group GP_j) . U_f is the maximum utility of joining other groups as shown in Eq.5.3. Eq.5.2 and Eq.5.3 show that the closer to the nearby group, the more desirable to follow the group. The longer an agent has stayed with its group, the more desirable it will still stay with its group as illustrated in Fig. 5.3 and 5.4.

$$Dist_{i,GP_j} = Dist(i, j) \tag{5.1}$$

$$t = DesiredDist / Dist_{i,GP_j} \tag{5.2}$$

$$U_{i,f} = \max_{for\ all\ j} (t * e^{Sociality i\ *(1-Duration\ /Threshold\ )}) \tag{5.3}$$

$$U_{i,s} = c * e^{Sociality i\ *Duration\ /Threshold} \tag{5.4}$$

$$GroupToJoin = \begin{cases} The\ group\ with\ U_{if} & U_{i,f} > U_{i,s} > U_{threshold} \\ GP_s & U_{i,s} \geq U_{i,f} > U_{threshold} \\ GP_{new} & U_{i,f} < U_{threshold}\ and\ U_{i,s} < U_{threshold} \end{cases} \tag{5.5}$$

Fig.5.5 shows that when the agent is more desirable to join another group, it will join the group with the maximum utility U_f.

5.3.2 Modeling agent-to-agent interactions

The agent-to-agent interaction decides which member to follow from the selected group. This step is skipped when an agent decides to create a new group consisting of itself. The agent-to-agent interaction is modeled on the basis of social comparison theory [40]. The basic idea is to select a member (from the selected group *GPToJoin*) that

has greatest similarity as the one that has greatest influence on the agent and let the agent follow the selected member. The specific process is described as follows.

Assume *i* and *j* are two agents. v_i and v_j is the velocity of agent *i* and *j* respectively. v_l and v_m is the vector pointing from agent *i*'s current position to the position of the selected member and agent *i*'s destination, respectively. Eq.5.6-Eq.5.8 show the agent-to-agent interaction process for agent *i*. *a* and *b* are two constant numbers. For each agent *j* which belongs to the selected group *GPToJoin* and is in the perception range of pedestrian *i*, the similarity between *i* and *j* is calculated using Eq.5.6. As can be seen, the greater the agent *i*'s sociality, and the closer the distance between *i* and *j*, the more likely agent *i* will follow agent *j*. The moving direction also has effect on the similarity value. Pedestrian agents prefer to follow the pedestrian moving in the same direction. The agent will follow the one with the greatest similarity as calculated in Eq.4.4 and Eq.4.5. If no such agent exists, agent *i* will not follow any agent. Once the most similar agent is decided, one can calculate the possibility that agent *i* will follow the agent, as well as the influence strength of the agent on agent *i*. The calculation is described as follows.

$$Similarity_j = Sociality_i * D_1 * (a * D_2 + b * e^{1-dist(i,j)/Dist1}) \quad (5.6)$$

$$D_1 = \begin{cases} 1 & \text{If } v_l * v_m \geq 0 \\ 0 & \text{Otherwise} \end{cases} \quad (5.7)$$

$$D_2 = \begin{cases} 1 & \text{If } v_i * v_j \geq 0 \\ 0 & \text{Otherwise} \end{cases} \quad (5.8)$$

The possibility that agent *i* will follow agent *j* is calculated according to Eq. 5.9. *Similarity* represents the similarity value between *i* and the selected agent *j*. *Smax* and *Smin* is the predefined maximum and minimum similarity value, respectively. As can be seen, the greater the similarity, the more likely agent *i* will follow the selected agent *j*. Notice that, the possibility is a real number within the range [0, 1].

$$Possibility_i = (Similarity_i - S\min)/(S\max - S\min) \tag{5.9}$$

Eq.5.10 shows the calculation of the weight of influence which the selected agent *j* will enforce on agent *i*. ξ is a small number which represents the noise applied in the calculation of forces. *ran_num* is a random number generated by a random number generator. *t* is proportional to the ratio of the distance between agent *i* and *j*, and a predefined desired distance. As can be seen, the greater the possibility *i*, the larger the distance between *i* and j, the greater the weight will be.

$$Weight_i = \begin{cases} t * e^{possibilityi} + \xi & ran\_num \le possibilityi \\ 0 & \text{Otherwise} \end{cases} \tag{5.10}$$

$$t = Sociality_i * dist(i, j) / desired\_dist \tag{5.11}$$

Note that, if no agent is selected to follow, or the similarity value is out of the specified range [*Smin*, *Smax*], the weight will be 0 and pedestrian *i* will not follow any agents.

5.3.3 Inter-group and intra-group matrix

To make the dynamic group model be simple, the inter-group relationships are not considered. However it is possible to integrate the inter-group relationships by studying the group-to-group relationships in this dynamic group model.

The dynamic group formation is studied through the agent-to-group and agent-to-agent procedures. These two procedures setup the intra-group connections between agents. At each time step, a pedestrian agent computes the similarity between itself and each member in the group *GPToJoin* (Eq. 5.6-Eq.5.8). The possibility and weight of the following behavior will also be calculated (Eq. 5.9-Eq.5.11). The normalized value of the greatest weight (of [0.0, 1.0]) represents the current intra-group connections of the agent. Specifically, the intra-group matrix for agent i at time step t is calculated in Eq. 5.12.

$$I(i, j) = \begin{cases} \textit{Normalized weight } (i, j) & j \text{ is the most similar agent to } i \\ 0 & \text{Otherwise} \end{cases} \qquad (5.12)$$

Note that agent i only has intra-group connection to the most similar agent j. Unlike the intra-group connections presented in Chapter 4, the intra-group connection in this dynamic group model is constantly changed during the simulation.

5.4 Experiments And Result Analysis

This section presents two experiments which explore the developed dynamic group model. The first experiment evaluates whether the developed model can simulate dynamic grouping in pedestrian crowds. The second experiment studies the effect of sociality on crowd behaviors. In the experiments, the crowd contains 60 agents which are situated in a circular rectangle-shaped hallway environment with the size 600 in length and 200 in width. A simulation scenario of the environment is shown in Fig.5.2, where the lane width is 50. The green and gray circles represent agents in and not in maintaining groups at the current simulation cycle, respectively. The number in circles indicates the *ID* of the agents.

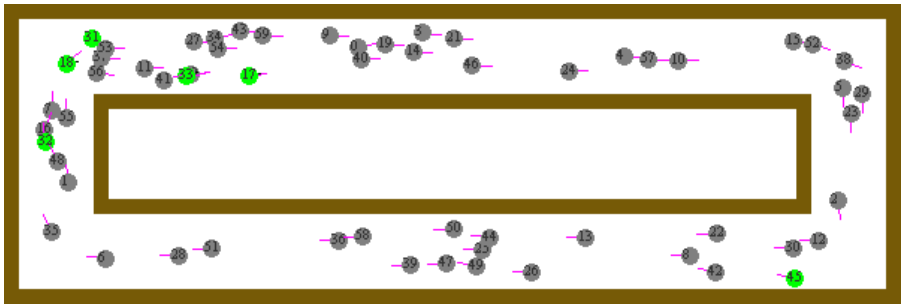


Figure 5.2 A circular rectangle-shaped hallway simulation environment.

5.4.1 Experiment 1 - Simulation of dynamic groups and two groups

This experiment studies the dynamic grouping behavior in pedestrian crowds using the developed dynamic group model.

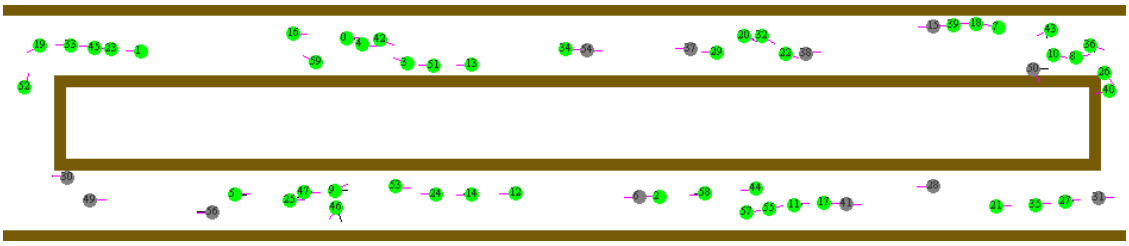


Figure 5.3 A simulation scenario of dynamic grouping in a linear style.

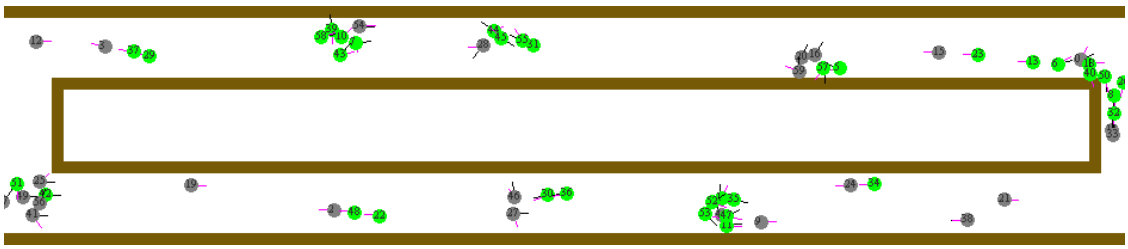


Figure 5.4 A simulation scenario of dynamic grouping in a leader-follower style.

Fig. 5.3 shows a simulation scenario of agents' dynamic grouping in the simulation. Half of agents move clockwise while others move anti-clockwise. All agents have the sociality 1.0. Each agent follows *the member* which is most similar with the agent. The crowd contains linear groups where members follow each other in a line formation. Note that other groups can also be simulated by changing the agent-to-agent procedure. Fig. 5.4 shows another simulation scenario of dynamic grouping in crowds. Different from Fig. 5.3, each agent follows *the predecessor of the member* which is most similar with the agent. The crowd contains leader-follower groups where a single member is followed by other members.

5.4.2 Experiment 2 - The effect of sociality on crowd behaviors

This experiment explores the effect of sociality on dynamic grouping behavior in pedestrian crowds. Intuitively, the greater sociality, the more likely an agent will interact with others and the closer agents will move together. Fig. 5.5 shows the relationship between sociality and the average distance between members in dynamic groups. The average distance is calculated as the average of distances between members in all dynamic groups during 50 cycles starting at the cycle 2850. Five cases are tested. For each case, all agents are set to have same sociality. In this experiment, the desired distance *desired_dist* is set to be 30 (see Eq.5.8 for details). As can be seen, the greater the sociality, the closer agents move together, and the average distance decreases linearly as the sociality of pedestrian agents.

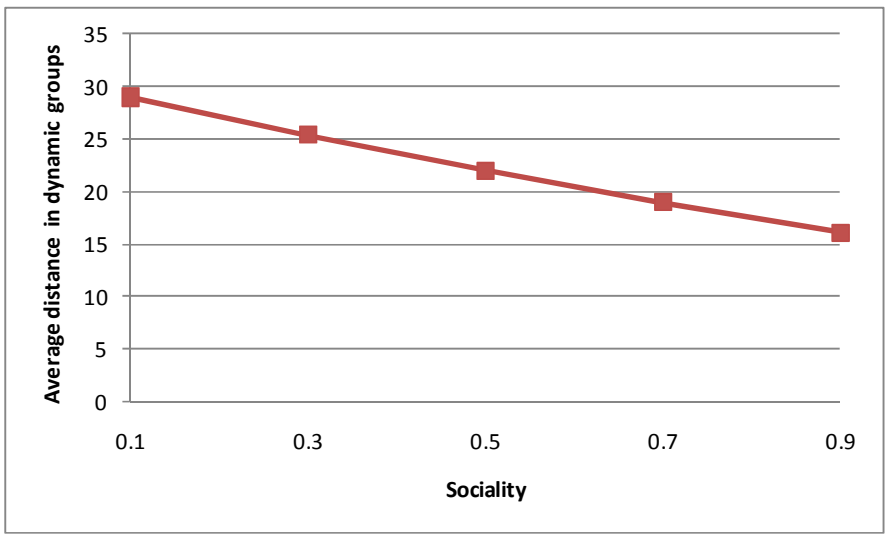


Figure 5.5 The effect of sociality on the average distance between members in dynamic groups.

5.4.3 Analysis of the experimental results

Experiments show that the dynamic group model can be used to simulate dynamic groups and to study the effect of social factors such as sociality on group behaviors. The two-tier context modeling layer allows pedestrian to adapt to external environments by integrating utility theory and social comparison theory. Note that the two tiers are loosely coupled, i.e. the change of one tier does not affect the other tier. For example, one can use other artificial intelligence theories to model the agent-to-group procedure simply by replacing the utility theory without requiring a change on the agent-to-agent procedure. The flexibility is illustrated in experiment 1 where different groups can be formed simply by changing the strategy of selecting which member to follow in the agent-to-agent procedure.

5.5 Discussions

This chapter developed a dynamic group model using the proposed framework. The dynamic group model studies the dynamic group formation through the agent-to-group and agent-to-agent procedures. Several improvements such as integrating the inter-group relationships, removing the assumption that initially each agent belongs to a group consisting of itself, are likely to be made to create a more dedicated group model. Such improvements are not the focus of this chapter.

The purpose of this chapter is to show the applicability of the proposed framework by specifying the different layers according to the specific domains. The two-tier *Context Modeling Layer* illustrates the flexibility in studying the effect of various social,

psychological, and artificial intelligence theories on crowd behaviors. The *Group Modeling Layer* demonstrates the simplicity in developing intra-group connections and inter-group relationships. Different group-related applications can be developed by specifying different *Context Modeling Layer* and *Group Modeling Layer*.

CHAPTER 6

AN EFFICIENT DISCRETE EVENT SIMULATION ENGINE

6.1 Introduction

As pedestrian crowd simulation systems become pervasive, the scale of pedestrian crowd model also increases. For example, New York City has several million people [63]. However, current sequential simulations can support at most a population size of several thousand [64, 65]. Besides, to create accurate simulations, people tend to use sophisticated decision-making model which generally downgrade the simulation efficiency. It is necessary to improve the performance of simulation engine in order to support the simulation of large-scale pedestrian crowds.

Most of the pedestrian crowd models adopt the discrete time based approach where the simulation proceeds in a discrete time manner, where each agent performs a movement decision to decide its next movement in each time step. Although the discrete time based simulation approach is easy to understand, it is inefficient since every agent makes a decision at every time step, regardless of the agent's individual difference such as movement speed, age, sex, and so on. For example, considering two agents situated in a large environment where one agent moves 10 times slower than the other. In a discrete time based approach, both agents make a movement decision in every time step. This is independent of the agent's speed. However, since one agent moves much slower than the other, intuitively one would think that the slower agent

does not need to make movement decisions as frequently as the fast one. In extreme cases where the movement of pedestrian agents is fully blocked, e.g. in the extremely dense pedestrian crowd, there is no movement decision needed to be made. It is difficult for the discrete time based approach to exploit such information, i.e. agents' individual differences. A more computationally efficient way of simulating the two agents is to allow the fast agent to make movement decisions more frequently and the slow agent to make movement decisions less frequently. In pedestrian crowd simulations with realistic human-like behaviors, agents typically have non-uniform movements due to different individual characteristics such as moving speed, personality, the psychological states (e.g., panic, non-panic), and other factors. These non-uniform movements result in spatial and temporal heterogeneity in terms of agents' movement decisions. Thus it is desirable to explore such heterogeneity for more computationally efficient simulations, especially for the large-scale pedestrian crowd simulations. For example, it is desirable to create efficient simulation for the crowd containing a lot of social groups.

In this chapter, we developed a discrete event based simulation approach which is efficient for the simulation of heterogeneous pedestrian crowds. The discrete event based approach is featured with a discrete event model which uses a concept of "*space resolution*", which defines the threshold of an agent's position change in the environment, to decide the frequency of an agent's movement decisions. With the space resolution, an agent's position change less than the space resolution threshold does not

trigger its movement decision. Nor does it trigger the message passing from the agent to others. As a result, agents that move slowly make movement decisions less frequently than the fast agents. This concept of "space resolution" is derived directly from the quantization and activity concepts presented in [66]. The value of the space resolution has significant impacts on the crowd behavior simulation.

On one hand, the larger the space resolution is, the less frequently agents make decisions, and thus the more efficient the simulation is. On the other hand, the space resolution means that an agent does not update its position until its position change bypasses the space resolution threshold. This introduces position errors in the crowd simulation. The larger the space resolution is, the larger the position errors are, and thus the less accuracy the simulation is. Note that similar kind of relationships also exists in a discrete time model, whose efficiency and precision depend on the value of the time step. A main effort of this chapter is to establish a formal "fair-comparison" rule that quantifies the position errors of both the discrete time and discrete event models, and conduct experiments from different aspects to compare the two. Note that both the space resolution in the discrete event model and the time step in the discrete time model are global variables shared by all agents. In our work, the number of decisions taken is used as an indicator of simulation performance. This is based on the observation that an agent's decision making component usually involves complex logics, and thus accounts for the most significant part of computation in a simulation. The work is carried out based on the DEVS [67] modeling and simulation framework, in

particular the DEVSJAVA environment [68]. The DEVS framework was chosen due to its formal formalism and its capability of modeling both the discrete time and discrete event models. Nevertheless, note that the model design and the conclusions drawn in this research are general and do not rely on the DEVS framework.

6.2 Discrete Time and Discrete Event Simulation Model

The developed discrete event model represents a different modeling approach for pedestrian crowd simulations. Discrete time and discrete event simulation model are abbreviated as *DTS* and *DES*, respectively. This section presents the *DES* model and compares it with a *DTS* model. Both the *DES* model and the *DTS* model are implemented based on the DEVS modeling and simulation framework [67], in particular the DEVSJAVA environment [68]. The DEVS framework was chosen due to its formal formalism and its capability of modeling both the discrete time and discrete event models.

6.2.1 DEVS modeling and simulation framework

DEVS (Discrete Event System Specification) is a formal modeling and simulation framework featured with well-defined concepts, such as components, coupling, and hierarchical model composition. These concepts are expressed through a set of mathematical formalism which provides an approximation of dynamic systems using an object-oriented approach. One of the important objects in DEVS is the model. A

model is a set of instructions for generating data comparable to that observable in the real system [69]. In DEVS, the basic model is the atomic model which is described as a structure with several attributes:

$$M = \langle X, S, Y, \delta int, \delta ext, \delta con, \lambda, ta \rangle$$

where,

X : set of external input events such as a nearby agent notifies the source agent of its new position.

S : set of sequential states, such as “make_decision”, “move” and “inform_neighbors” as illustrated later;

Y : set of outputs such as agents’ new position;

$\delta int: S \rightarrow S$: internal transition function which is used to respond to the change of internal states.

$\delta ext : Q \times X^b \rightarrow S$: external transition function which is used to respond to the external event.

$\delta con: Q \times X^b \rightarrow S$: confluent transition function which handles the situation where an internal and external event occur at the same time.

X^b is a set of bags over elements in X,

$\lambda : S \rightarrow Y^b$: output function generating external events at the output;

$ta : S \rightarrow R^+$: time advance function;

$Q = \{ (s,e) \mid s \in S, 0 \leq e \leq ta(s) \}$ is the set of total states where e is the elapsed

time since last state transition.

Different models can be bound hierarchically through DEVS coupling mechanism. The result model is also called a coupled model. The constituted model could either be an atomic model or another coupled model. Models interact with each other through message passing. The source and destination model of the messages are specified by the coupling mechanism. One of the important features in DEVS is supporting the dynamic structure which refers to the capability of a simulation to dynamically change its model structure, such as the coupling information, as the simulation proceeds, through a set of predefined APIs [70].

The capability of DS modeling makes it possible to study interaction intensive system, such as the pedestrian crowd simulation systems and other autonomous systems. This work uses the dynamic structure to model the ever-changing interactions among pedestrian agents. The dynamic interaction is achieved by setting up the coupling between an agent and its neighbors dynamically. That is, every time an agent wants to interact with the neighbors, its old coupling information will be replaced with a new one which allows the interaction with the desired neighbors.

6.2.2 The DTS Model

The *DTS* model for simulating pedestrian crowd is straightforward to understand. At every time step, each agent checks its environment (e.g., if a destination is reached or

if there are other agents in nearby locations), makes a decision to decide its next movement (e.g., move forward, or move sideways to avoid collision), and then carries out the movement action for this time step. The movement action will change the agent's position. Thus at the next time step, the agent goes through the same sequence again to check its environment, make a decision, and carry out the movement.

To implement the *DTS* model in DEVS, each agent is modeled as a DEVS atomic model (see the work of [67] for the formal description of atomic model). Specifically, the model has two states "make_decision" and "update_position". At the "make_decision" state, the agent checks its environment and makes a decision to choose a movement action. After that, the agent transits to the "update_position" state where its position is updated. The above procedure is performed for each time step. Fig. 6.1 shows the procedure which is implemented in the internal transition function *deltint()* of an agent.

```
procedure deltint()
  if state = "make_decision" then
        check the environment;
        perform a movement decision and select a movement action a;
        holdIn("update_position", TimeStep);
  else if state = "update_position" then
        update position based on action a;
        holdIn("make_decision",0.0);
```

```
    end if.
end procedure deltint.
```

Figure 6.1 Internal transition function of the DTS model.

6.2.3 The DES Model

Unlike the *DTS* model, an agent in the *DES* model does not make a decision at every time step. Instead, the movement decision is based on the changes in the environment and/or the changes of the agent's own position (according to the space resolution). Whenever such a change happens, an agent checks its environment and makes a decision to choose a movement action (for example, move forward or avoid neighbor agents). Meanwhile, the agent informs its new position to its neighbors if its position change bypasses the space resolution. After a movement action is selected, the agent carries out the action until the next space resolution is reached or the agent is interrupted by messages from other agents. As a result, each agent performs its movement decision based on its "events" instead of a global time step.

1) Space resolution and message passing

The concept of space resolution is at the center of the *DES* model. It defines the "threshold" of an agent's position update. Whenever an agent's new position reaches this threshold, the agent updates itself to the new position and informs other agents of its new position. A position change within the threshold is not considered as an event, thus does not cause position update and message passing (i.e., an agent always

schedule its position update at the threshold). In a 2D environment, the space resolution forms a threshold circle whose center is the agent's current position.

Fig. 6.2 shows an example which illustrates how the space resolution works in a 2D environment. In Fig.6.2, the solid and dashed circles represent the threshold circles of the agents at two different locations. The radius of the circles is the agent's space resolution *SR*, and PS_i (i=0, 1, 2, 3, 4, 5) are the agent's positions at different time instances. In this example, the agent's initial position is at PS_0, where the agent decides to move in the direction along line PS_0-PS_2. Because of the space resolution, the agent schedules its next destination at PS_2, which is the intersection of the agent's moving direction and the threshold circle. The time for the agent to reach PS_2 is $Dist_{PS0\text{-}PS2}/speed = SR/speed$, where *speed* is the moving speed of the agent. Although the agent schedules to update its position at PS_2, it may be interrupted before it reaches the scheduled destination. In Fig. 6.2, the agent receives a message at position PS_1 from a nearby agent. Such a message indicates that the nearby agent's position has been changed. Since this represents an environmental change, the agent at PS_1 makes a new movement decision to respond to the environmental change. In the example shown in Fig. 6.2, the agent performs a movement decision and changes its moving direction to a new one along line PS_1-PS_4. Similarly, because of the space resolution, the agent schedules its next destination at PS_4, which is the intersection of the agent's moving direction and the threshold circle. In this case, the agent schedules the event (of reaching PS_4) in time $Dist_{PS1\text{-}PS4}/speed$, where *speed* is the current moving speed of the agent. It is important to

note that the threshold circle is always based on the last updated position (PS_0 instead of PS_1), in order to avoid accumulated errors that may occur when the agent receives frequent messages from other agents. Fig. 6.2 shows that the agent receives another message at PS_3 before it reaches its scheduled destination PS_4, and chooses a new moving direction along line PS_3-PS_5 and schedules its new destination at PS_5 on the threshold circle. Finally, the agent reaches PS_5 (on the threshold circle). At PS_5, since the threshold circle is reached, the agent updates its new position to PS_5, and sends a message to its neighbor agents about its new position. The position update also moves the agent's threshold circle to the new position PS_5 (shown by the dashed circle in Fig. 6.2). Meanwhile, the agent makes a decision to decide its next movement. Note that at position PS_1 and PS_3, the threshold circle is not moved and no message is sent to neighboring agents since the agent's spatial change is less than the space resolution.

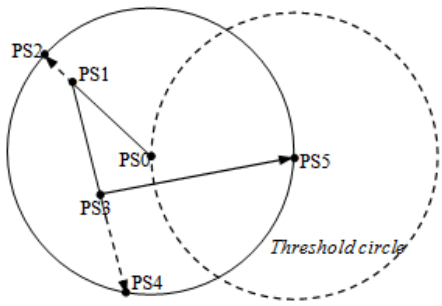


Figure 6.2 Space resolution threshold in 2D environment.

As can be seen, the space resolution defines the threshold of an agent's position update in the space. It decides the frequency of an agent's movement decisions.

Generally speaking, if an agent's moving speed is low, it takes longer time for the agent to perform the next movement decision because it takes longer time for the agent to reach the space resolution. The value of space resolution has significant impacts on crowd behavior simulations. On one hand, the larger the space resolution is, the less frequently agents make decisions, and thus the more efficient the simulation is (In this work one assumption is that the time spent on movement decision accounts for the most significant part of the execution time in a simulation). On the other hand, the space resolution means that an agent does not update its position until it reaches the space resolution threshold. This introduces position errors in the crowd simulation. The larger the space resolution is, the larger the position error is, and thus the less accurate the simulation will be. A similar kind of tradeoffs also exists in the *DTS* model, whose efficiency and accuracy depend on the value of the time step. Time step affects the frequency of agents' movement decisions. The larger the time step is, the less frequent an agent makes movement decision. Time step also introduces position errors in the crowd simulation. An agent does not update its position until the time step is reached. Thus, the larger the time step is, the larger the position errors will be.

The example in Fig. 6.2 also shows the importance of message passing in the *DES* model. An agent informs its position change to other agents through message passing. Such messages are treated as external events by a receiving agent and thus trigger the agent's movement decision to respond to the environmental change. If the agent receives more messages, the agent will need to make more movement decisions. In this

work, to avoid unnecessary message passing, an agent only passes messages to its nearby agents (the agents within a pre-defined distance range *Dist1*. *Dist1*=112.5 in this work. See Fig.2.2 for agent's perception model). An interesting aspect of this work is that the number of messages passed is related to the density of the crowd. In general, when the crowd is very sparse, there is less interaction among agents and thus the number of messages passed is small. As the density increases, agents interact more frequently and thus the number of messages passed also increases. However, it is not the case when the density keeps increasing. In a very dense crowd, agents block each other and thus only a small portion of the agents can move freely. Note that in the *DES* model, when there is no movement, there is no message passing. Thus, in a dense crowd the overall number of messages passed could be very small even if there are a large number of agents. An example of this and performance results are provided in experiments 3 and 4 in the experiment section.

2) DEVS implementation of the DES model

Each agent in the *DES* model is also implemented as an atomic model. Fig.6.3 shows the state transitions of the *DES* model. The black solid line indicates the external transition when a message is received. The dashed lines represent the internal transitions. The red solid line indicates the output (the updated position). In the "make_decision" state, an agent checks its environment and makes a decision. In

"move" state, the agent performs the action and carries out the movement. In the "inform_neighbors" state, the agent informs its updated position to the nearby agents.

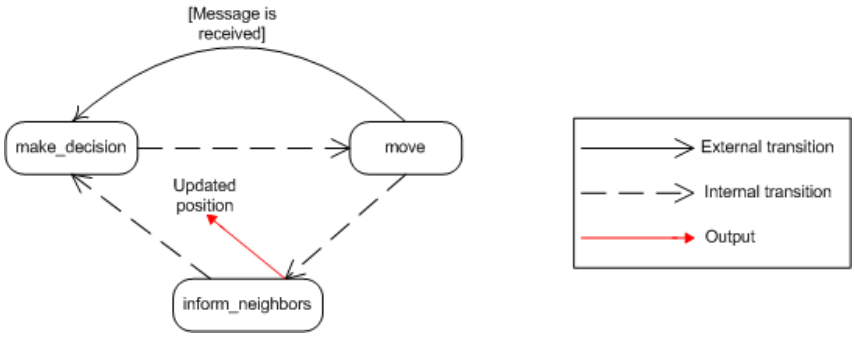


Figure 6.3 Agent state transition diagram of the *DES* model.

After initialization, the agent stays at the "make_decision" state where the agent performs a movement decision through the behavior model to decide the next movement. The result is an action which indicates where the agent should move to. The action is further adjusted such that the movement does not exceed the space resolution. The agent prepares the next movement by changing to the "move" state. After holding in a duration (calculated as *SR/speed*), the agent moves to the new position. Dynamic couplings between the agent and its neighbors are established. The agent goes to the "inform_neighbors" state to send its updated position to the neighbors, which can be referred by nearby agents to avoid possible collisions. After sending the message, the agent transits to the "make_decision" state. After each movement or when a message is received from nearby agents, the agent performs a new movement decision which

continues the procedure as mentioned above. Pseudo-codes of each function of DEVS atomic model are listed as follows.

When the crowd system starts up, each agent is initialized according to the initialization function, as shown in Fig. 6.4. In the function, the moving speed and space resolution are initialized. Besides, the agent's initial state is set as "make_decision". In what follows, holdIn is a function of the DEVS framework which lets the model transit to the specified state with a duration time.

```
procedure initialize()
    speed = Agent's moving speed;
    SR = Predefined space resolution of the agent;
    holdIn("make_decision", 0.0); # Initial state will be "make_decision".
end procedure initialize.
```

Figure 6.4 Initialization function of the *DES* model.

The external transition function is used to handle the messages received from others. When a message is received, the position of the agent is updated to the location where the agent is interrupted, and then the agent transits to the "make_decision" state. The procedure is shown in Fig. 6.5.

```
procedure deltext(e, x)
```

```
    if message received in x then
        update agent's position to the interrupted location based on elapse time e and
the previous position;
        holdin("make_decision", 0.0);
    end if.
end procedure delext.
```

Figure 6.5 External transition function of the *DES* model.

In the internal transition function, the agent performs a movement decision if the current state is "make_decision". Otherwise, it performs the movement, updates its position and sends the new position to nearby agents if the movement distance is equal to its space resolution. The procedure is shown in Fig. 6.6, where *action.distance* represents the movement distance of the current action.

```
procedure deltint()
if state = "make_decision" then
    move the threshold circle to the current position with the radius of SR;
    perform movement decision and select a movement action action;
    if action = NULL then # Special case: No way to go, deactivate the atomic model.
        passivate( );
        return;
```

```
    end if
        holdIn("move", action.distance/speed);
    else if state = "inform_neighbors" then
        holdIn("make_decision", 0.0);
    else # the agent's current state is "move".
        update position based on action action;
        setup couplings with the nearby agents;
        holdIn("inform_neighbors", 0.0);
    end if
end procedure deltint.
```

Figure 6.6 Internal transition function of the *DES* model.

The output function is used to inform nearby agents of the agent's new position if the current state is "inform_neighbors". The procedure is described in Fig. 6.7.

```
procedure output()
if state = "inform_neigbors" then
    inform nearby agents with new position;
end if
end procedure output.
```

Figure 6.7 Output function of the *DES* model.

6.3 Fair Comparison Condition

Pedestrians are autonomous agents, each of which has an *ID* that defines the global unique identification of the agent, and *speed* that is the agent's moving speed. Each pedestrian agent is featured with two behaviors:

1) *CasualMove*: This behavior is used to simulate the casual movement of each agent. An agent moves to pre-specified but randomly generated destinations in a sequential order. The moving path is the shortest path from the current position to the destination. When a destination is reached, the agent moves to the next destination. Note that for the same agent, i.e., agent with same *ID*, the moving path is same in both models for the purpose of comparison.

2) *Avoid*: This behavior is used to simulate the obstacle avoidance during the movement. When an agent is within a predefined minimum distance from the nearest neighbor agent or obstacle, it will stay away from it. In this behavior, if the agent is on the left side of the object to be avoided, it turns right with an angle; otherwise, it turns left. In this process, a basic "collision prediction" subroutine is used to predict if the current computing agent will collide with other agents once the turn is finished. If the subroutine returns true, the agent will try other angles recursively. If the turning left or right is not possible, the agent will stay at its current location.

For the *DES* model, *SR* defines the position change threshold of the agents. For the *DTS* model, *TS* defines the time step of the simulation. Note that each agent has its own space resolution in the *DES* model and the time step in the *DTS* model is a global variable shared by all agents.

Both the *DES* and *DTS* model introduce imprecision, also referred to as *position error*, in modeling agents' position updates. For the *DTS* model, an agent's position will not be updated until the time step is reached. Within a time step, an agent's position is considered as unchanged. For the *DES* model, an agent's position will not be updated until the space resolution threshold is reached. Any position change within the space resolution threshold is not captured. This section analyzes the position error introduced by the *DES* and *DTS* models and studies their relationship. The goal is to build a ground for comparing the *DES* and *DTS* models and showing how the *DES* model can exploit the heterogeneity of the crowd system.

Both the *DTS* and *DES* models are compared with an analytic model for an agent's position update. Assume there are n agents in the crowd and the simulation is running over the time base $[t1, t2]$, where $t1$ is the starting time and $t2$ is the ending time. Assume the position at time $t1$ is known of all agents. This position is also called *initial position*. Using the analytic model, an agent j ($1<=j<=n$)'s position at time t ($t1<j<=t2$) is calculated through Eq.6.1.

$$\vec{p}_{t,j} = \vec{p}_{t1,j} + \int_{t1}^{t} \vec{v}_{t,j} dt \quad (6.1)$$

In both the *DES* and *DTS* models, an agent's position is updated discretely. Eq. 6.2 and Eq. 6.3 represent the position update of the *DTS* and *DES* models respectively, where Δt is the time step of the *DTS* model and *SR* is the space resolution of the *DES* model. In the *DTS* model, Eq. 6.2 shows that before the time step *t* is reached, the agent *j*'s position $\vec{p}_{t,j}$ is not changed. Thus, compared with the analytical model, between the time step *t-1* and *t*, there is an error of agent *j*'s position. Similarly, in the *DES* model, Eq. 6.3 shows that an agent's position will not be updated until the space resolution threshold is reached. Note that in Eq. 6.3, P_{t-1} should be interpreted as the agent's previous position, instead of the position at time *t-1*.

$$\vec{p}_{t,j} = \begin{cases} \vec{p}_{t-1,j} & \text{if } t < t - 1 + \Delta t \\ \vec{p}_{t-1,j} + \int_{t-1}^{t} \vec{v}_{t,j} dt & \text{if } t = t - 1 + \Delta t \end{cases} \quad (6.2)$$

$$\vec{p}_{t,j} = \begin{cases} \vec{p}_{t-1,j} & \text{if } \left| \int_{t-1}^{t} \vec{v}_{t,j} dt \right| < SR \\ \vec{p}_{t-1,j} + \int_{t-1}^{t} \vec{v}_{t,j} dt & \text{if } \left| \int_{t-1}^{t} \vec{v}_{t,j} dt \right| = SR \end{cases} \quad (6.3)$$

In order to make a fair comparison between *DES* and *DTS* models, the following condition should be satisfied: *The maximum position error in both DES and DTS models is*

same. In the *DTS* model, from Eq.6.2 the maximum position error is $\int_{t-1}^{t-1+\Delta t} \vec{v}_{\max} dt$. Here $v_{\max}$ is the maximum moving speed among all agents. While in *DES* model, from Eq.6.3 the maximum error is the space resolution *SR* of agents. Thus, Eq.6.4 holds when the *DES* and *DTS* model are compared.

$$SR = \left| \int_{t-1}^{t-1+\Delta t} \vec{v}_{\max} dt \right| \qquad (6.4)$$

When the moving speed of an agent is constant during a time step, Eq.6.4 can be simplified to Eq.6.5 shown below. In the following, *TS* is used to represent the time step of the *DTS* model.

$$SR = V_{\max} * TS \qquad (6.5)$$

Eq.6.5 is used as a basis in this work for comparing the *DES* and *DTS* models. In the next section, experiments are carried out to compare the two models based on Eq. 6.5.

6.4 Experiments And Result Analysis

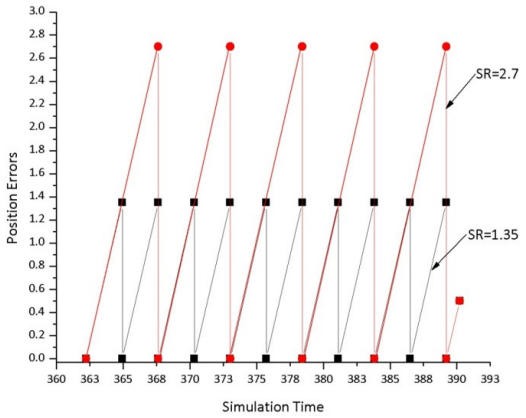
The section presents the verification of the *DES* model through one experiment. This section also presents the performance measurement of both the *DES* model and the *DTS*

model from four experiments. Verification process is used to ensure that in a fair comparison between the *DES* model and the *DTS* model, Eq. 6.5 holds for both models. The performance measurement measures the number of decisions taken and the execution time of both the *DES* model and the *DTS* model. The number of decisions is the total number of decisions which agents have been made during a predefined simulation. Each transition to the "make_decision" state is considered as one movement decision (see Fig. 6.6 for details). The total number of decisions is thus the total number of transitions to the "make_decision" state in the crowd. The execution time represents the duration (in milliseconds) of the predefined simulation. Note that the number of messages passed is not measured separately since each time when an agent receiving a message will make a movement decision, that is, the number of messages passed is included in the number of decisions. Thus, only the number of decisions and execution time will be measured in four experiments (Experiment 1-4). Note that, the ratio of the number of decisions and the execution time will then be calculated. The ratio of the number of decisions is calculated as the number of decisions of the *DTS* model divided by that of the *DES* model. Similarly, the ratio of the execution time is calculated as the execution time of the *DTS* model divided by that of the *DES* model. The ratio of the execution time also represents the speedup of the *DES* model. The greater the ratio of the execution time, the more efficient of the *DES* model. Thus, in what follows, the speedup is used to measure the simulation performance of the *DES* model.

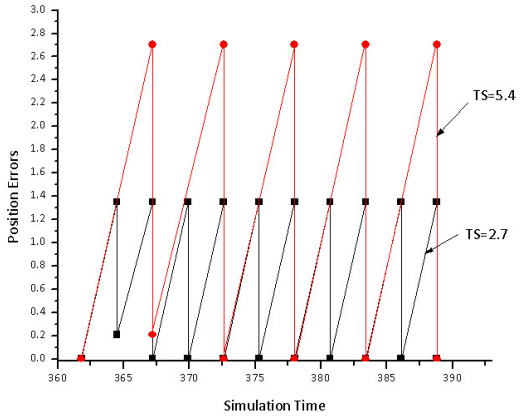
The first experiment (Experiment 1) explores the effect of space resolution (of the *DES* model) / time step (of the *DTS* model) on the simulation performance. The second experiment (Experiment 2) illustrates how the speed heterogeneity affects the simulation performance by varying the number of agents moving at speed 0.02 (the moving speed of an agent is either 0.02 or 2.0). The third experiment (Experiment 3) demonstrates the effect of crowd density on the simulation performance by changing the number of agents in the crowd. The fourth experiment (Experiment 4) shows an emergency evacuation example for both models with a variety of crowd densities. The last two experiments also illustrate the effect of space heterogeneity on the simulation performance due to the different crowd behaviors in different environmental regions. All the four experiments are carried out in a rectangle simulation environment (by default, 900 in length and 360 in width), with four walls of width 20 surrounded the environment. Note that environment size, agent size, wall width, position error, and space resolution have same measurement unit which is meter (m). The agent's moving speed is measured by meter per second (m/s). To make a fair comparison, the same agent, i.e., agent with same *ID*, has the same destination and moving speed (in the range of [0.01, 2.0] by default) in both models. The time step of *DTS* model is defined as the division of space resolution by the maximum moving speed of the crowd. To make the simulation more realistic in that each agent contains complex logic for movement decision, an empty loop with 10,000 runs is included in each movement decision.

6.4.1 Verification Of The DES Model

Before introducing experiments, the *DES* model is verified by comparing with the analytic model. In this experiment, the position errors of both *DES* and *DTS* models are compared. The crowd consists of one agent whose moving speed is 0.5. The agent moves through a series of pre-specified .When all destinations are reached, the simulation stops and the position error for each simulation time are calculated. Here, the position error is the difference of the position between the *DES*/*DTS* model and the analytic model (see Eq. 6.1). Fig. 6.8 presents the position errors of the two models under two space resolutions *SR* and two time steps *TS*. As described in Section 6.3, to ensure fair comparisons between the two models, for a specific space resolution *SR* in a DES simulation, the corresponding time step *TS* in a *DTS* simulation is calculated as *TS* = *SR*/*v*, where *v* is the agent's moving speed (0.5 in this experiment).



(a) The *DES* model



(b) The *DTS* model

Figure 6.8 Position errors in *DES* model and DTS model (SR=1.35 and 2.7, correspondingly TS = 2.7 and 5.4).

Fig. 6.8(a) shows the agent position errors in *DES* model under two space resolutions 1.35 and 2.7. X-axis represents the simulation time. Y-axis indicates the position error at different time. For *SR*=1.35 the agent updates its position at time 2.7*N (N=1, 2, 3…) since the moving speed is 0.5. And the position error is increasing linearly between two position updates. Similarly, for *SR* = 2.7, the agent updates its position at time 5.4*N (N=1, 2, 3…). Fig. 6.8(a) shows that the greater the space resolution, the greater the position error, and the less the simulation accuracy. Fig. 6.8(b) shows the position errors in the *DTS* model under two time steps 2.7 and 5.4. It shows that the position error increases linearly between two time steps. For the time step 2.7, the maximum position error is 1.35. Note that when the agent approaches a destination (i.e. the time step 364.5), there is position error since in our implementation if the agent is near the destination within a specified distance range; the agent is assumed to have

reached that destination. Fig. 6.8 confirms that both the *DES* and *DTS* model introduce position errors in agents' position update. To make a fair comparison, the maximum position error in both models should be the same. The *DES* model with *SR*=2.7 and the *DTS* model with *TS*=5.4 can be fairly compared since the maximum error in both models is the same. Similarly, the *DES* model with *SR*=1.35 and the *DTS* model with *TS*=2.7 can be fairly compared because of the same maximum error.

6.4.2 Experiment 1 – Space Resolution/Time Step vs. Performance

This experiment varies space resolution of the *DES* model (also time step of the *DTS* model calculated according to Eq. 6.5) and explores how space resolution affects the simulation performance. Specifically, one simulation is used to measure each space resolution. Each simulation lasts for 2 hours (*simulate_TN(7200)* in DEVS framework). To let the speed heterogeneity has same effect on simulation performance among different simulations, the speed of an agent is kept same in different simulations. The speed configuration of the crowd is as follows. 10 speeds are randomly generated (in the range of [0.01, 2.0]) and each of which is assigned to 10 percent of the agents. The crowd consists of 100 agents. In the following figures, "DESDM", "DTSDM", "DESTime", "DTSTime" represent the number of decisions of the *DES* model, the number of decisions of the *DTS* model, the execution time of the *DES* model, and the execution time of the *DTS* model, respectively. "DMRatio" represents the ratio between

the total number of decisions made by the *DTS* and *DES* model, and "TimeRatio" is the ratio between the total execution time of the *DTS* and *DES* model.

Fig. 6.9 and Fig. 6.10 show the number of decisions and the execution time under different space resolutions/time steps and the corresponding ratios, respectively. In both figures, x-axis represents space resolution of the *DES* model and time step of the *DTS* model. In Fig. 6.9, the y-axis represents the number of decisions of the crowd (left y-axis) and the execution time (right y-axis, in milliseconds). In Fig. 6.10, the y-axis represents the ratio of the number of decisions and the execution time between the *DTS* model and the *DES* model.

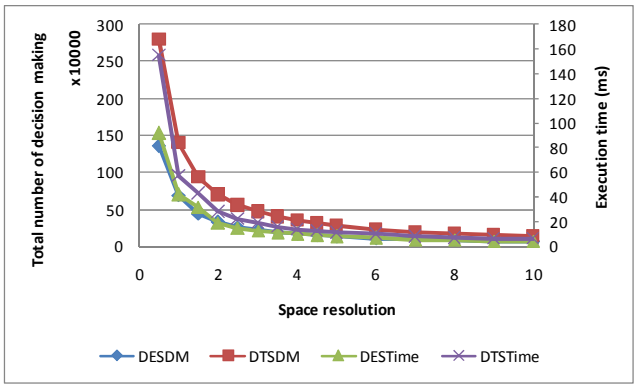


Figure 6.9 The number of decisions and execution time under different space resolutions.

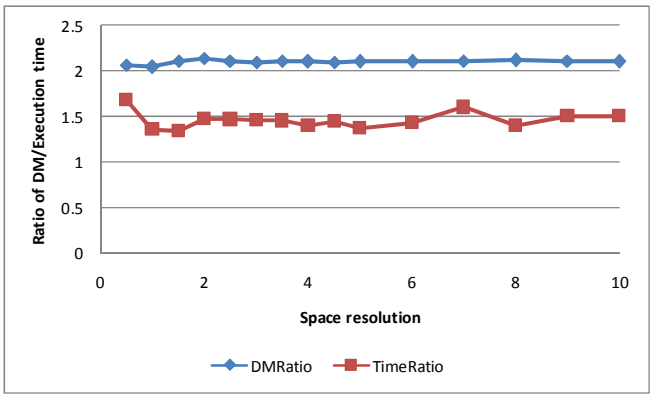


Figure 6.10 Ratio of number of decisions/execution time under different space resolutions.

Fig. 6.9 shows that, as space resolution/time step increases, both the number of decisions and the execution time decrease in both models. This is because with the same speed, the greater the space resolution/time step, the longer the duration between successive decisions. Thus, with a predefined simulation time, the greater the space resolution/time step, the fewer decisions performed by the crowd. Fig. 6.9 also shows that the number of decisions and execution time of the *DTS* model are greater than those of the *DES* model. This is due to the non-uniform moving speeds of the crowd. In the *DES* model, with the same space resolution, the slower an agent moves, the less movement decisions will be made. Fig. 6.10 shows that, the ratio of the number of decisions (DM) or the execution time is not varied so much under different space resolutions. This is because in both models, the number of decisions decreases with a same rate when space resolution increases. Fig. 6.10 also shows that the *DES* model is more efficient than the *DTS* model and about 1.5x speedup can be achieved for crowds of agents with non-uniform moving speeds.

6.4.3 Experiment 2 – Speed Heterogeneity vs. Performance

This experiment explores the effect of speed heterogeneity on the simulation performance. Speed heterogeneity is varied by the change of the number of agents moving at the speed 0.02. The more agents moving at speed 0.02, the greater the speed heterogeneity of the crowd. When all agents move at speed 0.02, there is no heterogeneity in the crowd. The space resolution is predefined as 1.0 and the crowd consists of 100 agents. Each agent moves at speed 0.02 or 2.0. The simulation lasts for 2 hours. Fig. 6.11 presents the number of decisions and the execution time of both the *DTS* model and the *DES* model under different speed distributions. Fig. 6.12 shows the ratio of the number of decisions (DM) and the ratio of the execution time. In both figures, x-axis represents the number of agents moving at speed 0.02. The y-axis in Fig. 6.11 and Fig. 6.12 has same meaning as Fig. 6.9 and Fig. 6.10, respectively.

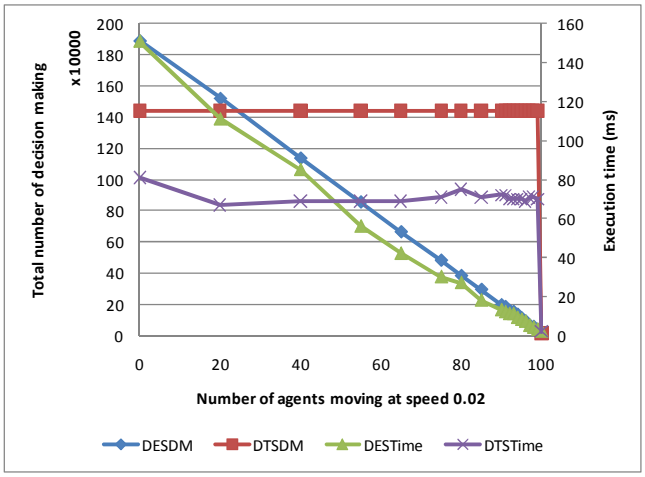


Figure 6.11 The number of decisions/execution time under different speed distribution.

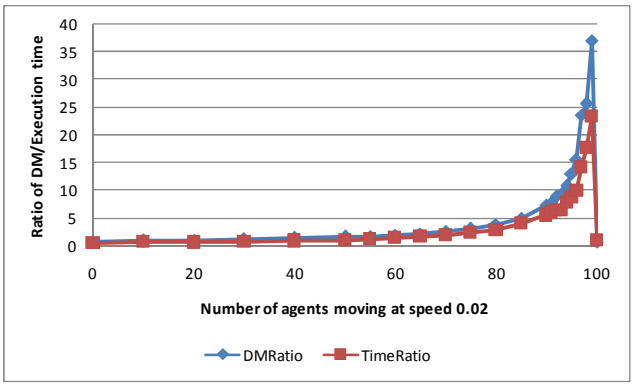


Figure 6.112 Ratio of number of decisions/execution time under different speed distribution.

Fig. 6.11 shows that, as the speed heterogeneity increases, in the *DES* model, both the number of decisions and execution time decrease; while in the *DTS* model, the number of decisions and execution time are not varied so much. This is because in the *DES* model, the greater the heterogeneity, the more agents moving at speed 0.02, the fewer decisions made by the crowd. While in the *DTS* model, the number of decisions and execution time only depends on the time step, which is not changed during the simulation since the space resolution is not changed during simulations. Fig. 6.11 also shows that for the crowd with uniform moving speed (all agents move at the speed 0.02), the number of decisions and the execution time in both models are almost the same. This is because when agents move at the same speed, the number of decisions and execution time of both models are same since each agent performs a decision in each time step. Fig. 6.12 shows that, the ratio of both the number of decisions (DM) and the execution time increases as the speed heterogeneity increases. This is because in

DES model, the greater the speed heterogeneity, the fewer movement decisions of the crowd since more agents are "slower" agents. It can be seen that, the greatest speedup (about 23) can be reached when the crowd has greatest heterogeneity, i.e., only one agent moves at the speed 2.0. For crowds with uniform moving speed 0.02, the *DES* and *DTS* model have same efficiency, and the speedup is 1.0.

6.4.4 Experiment 3- Crowd Density vs. Performance

This experiment demonstrates the effect of crowd density on the simulation performance for both models. Different crowd densities are represented by different number of agents under the same simulation environment. Each density is tested by one simulation. The simulation lasts for half an hour, i.e., *simulate_TN(1800)* in DEVS framework. Similar as Experiment 1, 10 speeds are randomly generated and each of which is assigned to 10 percent of the agents. The speed of each agent is kept same in different simulations to let the speed heterogeneity be the same among different simulations. The space resolution of all agents selected as 1.0. Thus, the time step in the *DTS* model is defined as 1.0 divided by the maximum moving speed in the crowd. Fig. 6.13 presents the number of decisions and the execution time of both the *DTS* model and the *DES* model under different crowd densities. Fig. 6.14 shows the ratio of the number of decisions (DM) and the ratio of the execution time. In both figures, x-axis represents the number of agents in the crowd. The y-axis in Fig. 6.13 and Fig. 6.14 has same meaning as Fig. 6.9 and Fig. 6.10, respectively.

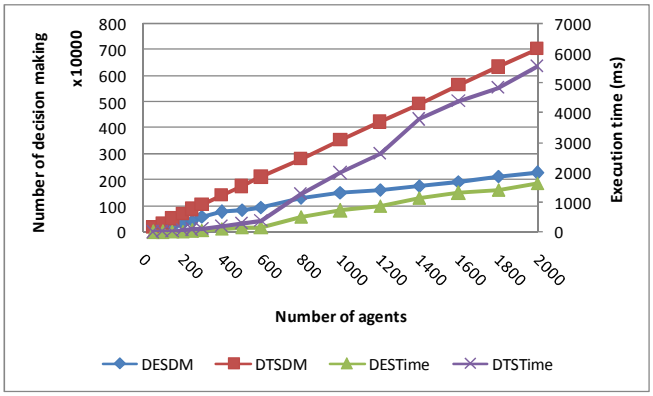


Figure 6.13 The number of decisions and execution time under different number of agents.

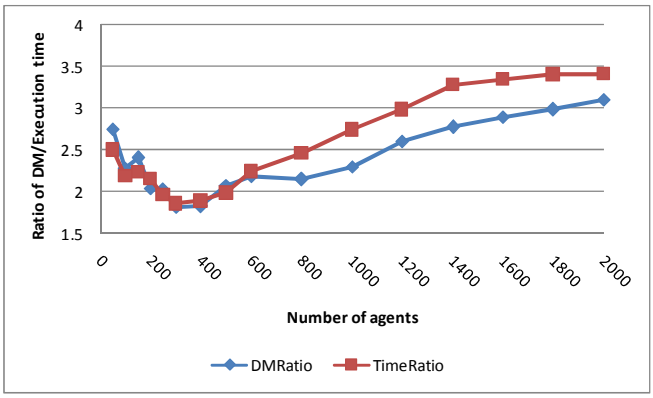


Figure 6.14 Ratio of number of decisions or execution time under different number of agents.

Fig. 6.13 shows that, both the number of decisions and the execution time increase as the number of agents increases in both models. This is because the more agents in the crowd, the more events will occur, i.e., the change of agents' internal status and the environmental status will be more frequent. Also, the number of decisions and the execution time of the *DTS* model are greater than those of the *DES* model due to the non-uniform moving speeds of the crowd. It can also be seen that, the *DES* model is

more computationally efficient than the *DTS* model since in the *DES* model, each agent only makes decisions when a significant external event occurs or its internal state changes.

Fig. 6.14 shows that, when the crowd is sparse (number of agents less than 400), the ratio of both the number of decisions and the execution time decrease as the density increases. This is because the denser the crowd, the more interactions among agents, the more messages passed within the crowd, thus the more decisions performed by the *DES* model. The ratio decreases since the number of movement decisions made in the *DTS* model is not changed (the decision frequency only depends on the global time step which is not changed since the space resolution is not changed). However, the two ratios do not necessarily decrease as the density increases for the crowded situations. Fig. 6.14 shows that when the number of agents is greater than 400, the ratios increase as the density increases. This is because agents block each other; allowing only a small portion of the agents can move freely. Note that in the *DES* model, when there is no movement, there is no change of agents' internal states and the external environment (due to no message passed in the crowd). Therefore, in a dense crowd the overall number of decisions could be very small even there are a large number of agents. The denser the crowd, the less decisions will be made, and the greater the ratios will be. Fig. 6.14 also shows that, for the extremely dense crowd, i.e., number of agents greater than 1400, the increasing of the two ratios will slow down and when the threshold is reached, i.e., number of agents greater than 1800, the ratios keep unchanged. This is because in

the extremely dense crowds, the number of decisions cannot be reduced so much even when the density keeps increasing since the number of agents, which will be blocked, also keeps increasing. When the density reaches the threshold, the number of decisions cannot be reduced further since most agents are blocked. Fig. 6.13 and Fig. 6.14 show that, the *DES* model is more efficient than the *DTS* model and the speedup can be up to 3.5X.

This experiment also demonstrates the effect of space heterogeneity on the simulation performance. The denser the space, the more agents will be blocked, the fewer messages passed in the crowd, thus the fewer decisions are made by the crowd. Experiment 4 demonstrates another example of how space heterogeneity affects the simulation performance.

6.4.5 Experiment 4 – An Emergency Evacuation Example

This experiment demonstrates the effect of crowd density on the simulation performance for both models in emergency evacuations. An exit entry situates at the right side of the environment which is of size 360 in both width and height. The size of the exit entry and the size of an agent is 10 and 7.5, respectively. In the simulation, all agents escape from the exit entry to the outside. Agents' moving speed is randomly generated in the range of [1, 3]. Agents' destinations are also randomly generated. Space resolution of all agents is 1.0. The simulation lasts for half an hour. A scenario of the emergency evacuation near is illustrated in Fig. 6.15.

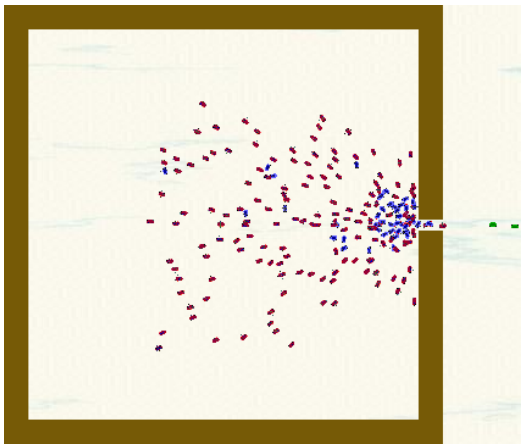


Figure 6.15 A simulation scenario of emergent evacuation.

Fig. 6.16 presents the ratio of the number of decisions and the execution time for different number of agents. X-axis and Y-axis represent the number of agents in the crowd and the ratio of the number of decisions/execution time, respectively. Fig. 6.16 shows that, the denser the crowd, the fewer movement decisions will be made in the *DES* model. This is because, in emergent situations, all agents move towards the exit entry to the outside. Near the exit entry, "clusters" of agents will be formed because of the slow flow in the exit entry (only one agent can enter the exit entry at each time). The denser the crowd, the larger the clusters will be, and the more agents cannot move freely. Thus, the *DES* model will perform fewer movement decisions. This experiment shows the effect of the space heterogeneity on simulation performance.

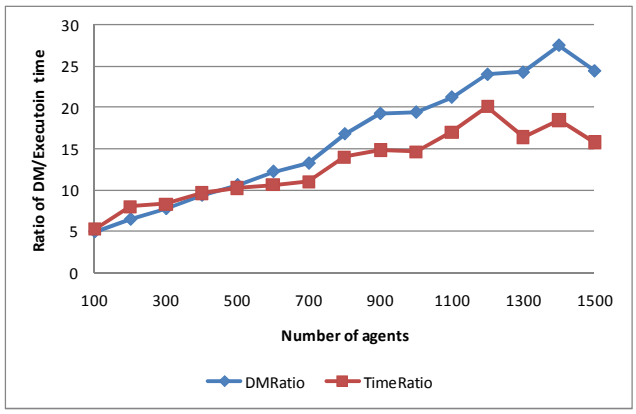


Figure 6.16 Ratio of the number of decisions and execution time in emergent evacuations.

In this experiment, a smaller environment (360x360) is used. It is more likely to have greater space heterogeneity of smaller environments than that of larger environments. The agents cannot move so freely in smaller environments as in larger environments. Thus, the space heterogeneity of the smaller environments will be greater than that of larger environments under same conditions, e.g., agents' destinations are same. Thus, the smaller the environment, the fewer movement decisions will be made, and thus the greater the speedup can be expected. Fig. 6.16 shows that the ratio of the number of decisions and the execution time are greater than the ratios presented in Fig. 6.14.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

7.1 Conclusions

Pedestrian crowd is a common social phenomenon in our daily life. Computer simulation is often used to study the behavior of pedestrian crowds due to the dissimilarity nature of pedestrians and the non-linear interactions in the crowds. Pedestrian crowd simulations have been widely studied in a variety of areas. As commonly existing in pedestrian crowds, group is an important research topic in sociology and psychology. It is widely studied in the context of institution or organization to study the group dynamics including the roles of individuals and role conflicts, the effect of group dynamics on personal development, group structures etc.

It is believed that group modeling can produce more realistic simulations. Although much work has been conducted in pedestrian crowd simulations, the influence of groups on the dynamics of crowd movement has not been incorporated into most existing crowd models because of the complexity of social groups. Group modeling is still a challenge and open problem in pedestrian crowd simulations due to the heterogeneity of pedestrian crowds. A well-defined system is needed to systematically study various social groups and to study the effect of social grouping on crowd behaviors.

This research develops a framework for group modeling in agent-based pedestrian crowd simulations. The framework includes multiple layers that support a systematic approach for modeling social groups in pedestrian crowd simulations. These layers include a simulation engine layer that provides efficient simulation engines to simulate the crowd model; a behavior-based agent modeling layers that supports developing agent models based on the developed *BehaviorSim* simulation toolkit; a group modeling layer that provides a well-defined way to model inter-group and intra-group relationships among pedestrian agents in a crowd; and finally a context modeling layer that allows users to incorporate various social/psychological models for studying social groups in pedestrian crowd. A demonstration of three social group models: leader-follower model, clustered model, and linear model is developed using the framework by following several simple steps. To the best of our knowledge, it is the first framework which can be used to model and simulate a variety of types of social groups.

To facilitate group modeling and simulation, a behavior-based agent simulation environment *BehaviorSim* is developed. One important effort in developing *BehaviorSim* is to propose a general and well-defined behavior-based model which can capture the essence of agent's behavior. The platform is so intuitive and easy-to-use that people can setup a behavior-based pedestrian crowd simulation without significant programming skills. The flexible design makes it even applicable for the simulation of behavior-based agents such as robot, animat, an artificial agent in AI, a character in game design, and so

on. *BehaviorSim* has been incorporated into a modeling and simulation class to be used by students.

To demonstrate the capability of the framework, we developed an application of dynamic grouping for pedestrian crowd simulations. The dynamic groups are studied using the proposed framework on the basis of utility theory and social comparison theory. The application is featured with a two-tiered context modeling layer which allows pedestrian agents to behave adaptively and intelligently in the ever changing environment. In the context modeling layer, dynamic grouping includes two steps, agent-to-group and agent-to-agent interactions. In the agent-to-group interaction, the selection preference of target group is specified by using utility theory. In the agent-to-agent interaction, the most similar member within the target group is selected by using social comparison theory. Experiment results indicate that groups can be dynamically formed and grouping has significant effect on crowd behaviors.

As the pedestrian crowd simulation systems become more and more pervasive, the scale of pedestrian crowd model also increases. Besides, to create more accurate simulations, people tend to use a complicated decision-making model of pedestrian agents. These affect the simulation performance. Although traditional discrete time based simulation approach is easy to understand, it is inefficient since every agent makes a decision at every time step, regardless the difference of their behaviors and moving speeds. To improve the performance of pedestrian crowd simulations, we developed a new method for pedestrian crowd modeling and simulation. This new

method uses a discrete event based approach by exploiting the crowd system's heterogeneity resulting from agents' different moving speeds. Experiment results show that, under fair comparison, the discrete event based approach can lead to more computationally efficient simulations than the discrete time based approach for crowds with different moving speeds.

With the framework for group modeling, the *BehaviorSim* environment, and the efficient discrete event based simulation approach, one can create efficient simulations for group-related applications such as dynamic grouping described in Chapter 5.

7.2 Future Work

In terms of group modeling and simulation, several future directions include validating the proposed framework using the realistic human behavior and movement data, creating a more dedicated group model, and formally analyzing the effect of spatial and temporal heterogeneity on simulation efficiency.

Validation is an important step to ensure the correctness of simulations. It is challenge for pedestrian crowd simulations since the experiments with real humans are difficult, even impossible to design, specifically in certain situations such as the emergent evacuations. Currently, the validation of crowd behavior models is mainly based on mathematical and statistical theories. For example, the work of Malone and Kaup [71] compares data sets generated from the flocking model with different

parameter values and uses statistical verification approach to quantitatively show differences between various simulation scenarios .

The validation of our framework is even more difficult since besides the individual behavior level, the group behavior level should also be validated. However, the framework is possibly validated using the same mathematical approach where the behavior and movement data of real social groups is obtained and then compared with the simulation results.

Another direction of future work is creating a more dedicated group model. In current group model, each social group is assumed to have only one leader and each pedestrian agent belongs to only one group. This assumption can be improved in some situations where each group can have several leaders e.g. in the emergent evacuations several leaders guide the members to exit the building in fire. Each agent can also belong to one or more social groups since an agent can have several social roles which belong to several groups. A separate layer can be added to the framework to extend the current group model to allow the additional functionality.

Another direction is to qualitatively analyze the discrete event based simulation approach on the effect of spatial and temporal heterogeneity on simulation efficiency. The analysis may be used to guide the application of the discrete event based simulation approach on many other domains e.g. robotics, to create efficient simulations.

REFERENCES

[1] D. Helbing, I. J. Farkas, P. Molnar, and T. Vicsek, "Simulation of Pedestrian Crowds in Normal and Evacuation Situations," in *Pedestrian and Evacuation Dynamics*, M. Schreckenberg and S. D. Sharma, Eds. Berlin: Springer, 2002, pp. 21-58.

[2] L. F. Henderson, "On the fluid mechanics of human crowd motion," *Transportation Research,* vol. 8, pp. 509-515, 1974.

[3] R. L. Hughes, "A continuum theory for the flow of pedestrians," *Transportation Research Part B: Methodological,* vol. 36, pp. 507-535, 2002.

[4] V. J. Blue and J. L. Adler, "Cellular automata microsimulation for modeling bi-directional pedestrian walkways," *Transportation Research B: Methodological,* vol. 35, pp. 293-312, 2001.

[5] S. P. Hoogendoorn, P. H. L. Bovy, and W. Daamen, "Microscopic pedestrian wayfinding and dynamics modelling," in *Pedestrian and Evacuation Dynamics*, M. Schreckenberg and S. D. Sharma, Eds. Berlin: Springer-Verlag, 2002, pp. 123-154.

[6] A. Schadschneider, "Cellular automaton approach to pedestrian dynamics - Theory," in *Pedestrian and Evacuation Dynamics*, M. Schreckenberg and S. D. Sharma, Eds. Berlin: Springer, 2002, pp. 75-86.

[7] H. Hamacher and S. Tjandra, "Mathematical Modeling of Evacuation Problems: A State of the Art," *Pedestrian and Evacuation Dynamics,* pp. 227-266, 2002.

[8] S. A. Tjandra, "Dynamic Network Flow Models for Evacuation Problems," in *Department of Mathematics* Ph.D. thesis, Kaiserslautern, Germany: Technische Universität Kaiserslautern, 2001.

[9] S. R. Musse and D. Thalmann, "A Model of Human Crowd Behavior: Group Inter-Relationship and Collision Detection Analysis," in *Computer Animation and Simulations '97*, Budapest, Hungary, 1997, pp. 39-52.

[10] S. R. Musse and D. Thalmann, "Hierarchical Model for Real Time Simulation of Virtual Human Crowds," *IEEE Transactions on Visualization and Computer Graphics,* vol. 7, pp. 152-164, 2001.

[11] H. C. S. Pan X., Dauber K. and Law K. H. , "A Multi-agent Based Framework for Simulating Human and Social Behaviors during Emergency Evacuations," *AI & Society,* vol. 22, pp. 113-132, 2007.

[12] N. Pelechano, J. M. Allbeck, and N. I. Badler, "Controlling Individual Agents in High-Density Crowd Simulation," in *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, San Diego, USA, 2007, pp. 99-108.

[13] F. Qiu and X. Hu, "Modeling Social Group Structures in Pedestrian Crowds," *Simulation Modelling Practice and Theory (SIMPAT),* vol. 18, pp. 190-205, Feb.2010 2009.

[14] M. M. Charles and J. N. Michael, "Tutorial on agent-based modeling and simulation part 2: how to model with agents," in *The 38th conference on Winter simulation*, Monterey, California, 2006, pp. 73-83.

[15] O. James, "Agents and Beyond: A Flock is Not a Bird," in *Distributed Computing*, 1998, pp. 52-54.

[16] C. W. Reynolds, "Flocks, Herds, and Schools: A Distributed Behavioral Model," in *SIGGRAPH '87*, Anaheim, CA, 1987, pp. 25-34.

[17] J. D. Sime, "Crowd psychology and engineering," *Safety Science,* vol. 21, pp. 1-14, November 1995 1995.

[18] M. Dijke and M. Poppe, "Social Comparison of Power: Interpersonal Versus Intergroup Effects," *Group Dynamics: Theory, Research, and Practice,* vol. 8, pp. 13-26, 2004 2004.

[19] J. Allbeck and N. Badler, "Toward Representing Agent Behaviors Modified by Personality and Emotion," in *Autonomous Agents and Multi-Agent Systems*, Bologna, Italy, 2002.

[20] J. H. Barkow, L. Cosmides, and J. Tooby, *The Adapted Mind: Evolutionary Psychology and the Generation of Culture*. New York: Oxford University Press, 1992.

[21] E. Bonabeau, "Agent-based modeling: Methods and techniques for simulating human systems," in *Proceedings of the National Academy of Sciences of the United States of America*, 2002, pp. 7280-7287.

[22] D. J. Low, "Statistical Physics: Following the Crowd," *Nature,* vol. 407, pp. 265-466, 2000 2000.

[23] Q. Ji and C. Gao, "Simulating Crowd Evacuation with a Leader-Follower Model," *International Journal of Computer Sciences and Engineering Systems,* vol. 1, pp. 249-252, 2007 2007.

[24] X. Pan, C. S. Han, K. Dauber, and K. H. Law, "A Multi-agent Based Framework for Simulating Human and Social Behaviors during Emergency Evacuations," *AI & Society,* vol. 22, pp. 113-132, 2007.

[25] F. A. Adrian, "The Not-So-Lonely Crowd: Friendship Groups in Collective Behavior," *Sociometry,* vol. 40, pp. 96-99, 1997.

[26] C. Loscos, D. Marchal, and A. Meyer, "Intuitive Crowd Behaviour in Dense Urban Environments using Local Laws," in *Theory and Practice of Computer Graphics 2003* Birmingham, UK, 2003, pp. 122-129.

[27] A. P. Hare, *Handbook of Small Group Research*, 2nd ed. New York: The Free Press, 1962.

[28] W. McDougall, *The Group Mind: A Sketch of the Principles of Collective Psychology with Some Attempt to Apply Them to the Interpretation of National Life and Character*: Kessinger Publishing, LLC 1920.

[29] G. Le Bon, *The crowd. With a new introduction by Robert A.Nye*. New Brunswick, NJ: Transaction Pub., 1995.

[30] S. Freud, *Group Psychology and the Analysis of the Ego*: Liveright Publishing, 1951.

[31] F. Allport, *Social psychology*. Boston,MA: Houghton Mifflin, 1924.

[32] W. P. Robinson, *Social Groups and Identities: Developing the Legacy of Henri Tajfel (International Series in Social Psychology)* Butterworth-Heinemann, 1996.

[33] L. Z. Yang, D. L. Zhao, J. Li, and T. Y. Fang, "Simulation of the kin behavior in building occupant evacuation based on Cellular Automaton," *Building and Environment,* vol. 40, pp. 411-415, 2005.

[34] H. Klupfel, T. Meyer-Konig, and M. Schreckenberg, "Models for Crowd Movement and Egress Simulation," in *Traffic and Granular Flow*, S. P. Hoogendoorn, Ed. Berlin: Springer, 2004, pp. 357-372.

[35] M. B. Villamil, M. S.R., and L. P. L. d. Oliveira, "A model for generating and animating groups of virtual agents," in *IVA 2003 - Intelligent Virtual Agents*, Irsee, Germany, 2003.

[36] F. Durupinar, J. Allbeck, N. Pelechano, and N. Badler, "Creating Crowd Variation with the OCEAN Personality Model," in *Seventh International Joint Conference on Autonomous Agents and Multi-Agent Systems* Estoril, Portugal, 2008, pp. 11217-1220.

[37] N. Ghasem-Aghaee and T. I. Oren, "Cognitive complexity and dynamic personality in agent simulation," *Computers in Human Behavior,* vol. 23, pp. 2983-2997, 2007 2007.

[38] S. Jaganathan, T. L. Clarke, J. Koshti, D. J. Kaup, and R. Oleson, "Intelligent Agents: Incorporating Personality into Crowd Simulation," in *Interservice/Industry Training, Simulation and Education Conference*, Orlando, Florida, 2007, pp. 1-8.

[39] N. Fridman and G. A. Kaminka, "Towards a Cognitive Model of Crowd Behavior Based on Social Comparison Theory," in *The Twenty-Second National Conference on Artificial Intelligence*, Vancouver, British Columbia, Canada 2007, pp. 731-737.

[40] L. Festinger, "A theory of social comparison processes," *Human Relations,* pp. 117-140, 1954 1954.

[41] G. Santos and B. E. Aguirre, "A Critical Review of Emergency Evacuation Simulation Models," in *NIST Workshop on Building Occupant Movement during Fire Emergencies*, Gaithersburg MD, 2004, pp. 27-52.

[42] B. Bruegge and A. H. Dutoit, *Object-Oriented Software Engineering Using UML, Patterns, and Java*, 2nd ed.: Prentice Hall, 2004.

[43] P. Lakhtanau, X. Hu, and F. Qiu, "BehaviorSim: Towards an Educational Tool for Behavior-based Agent," in *The 45th ACM Southeast Conference*, Winston-Salem, NC, USA, 2007.

[44] J. Bryson, "Cross-Paradigm Analysis of Autonomous Agent Architecture," *Journal of Experimental and Theoretical Artificial Intelligence,* vol. 12, pp. 165-189, April 2000 2000.

[45] M. J. Mataric, "Behavior-Based Control: Main Properties and Implications," in *IEEE International Conference on Robotics and Automation, Workshop on Architectures for Intelligent Control Systems*, 1992, pp. 46-54.

[46] F. Qiu and X. Hu, "BehaviorSim: A Learning Environment for Behavior based Agent," in *The 10th international conference on Simulation of Adaptive Behavior: From Animals to Animats*, Osaka, Japan 2008, pp. 499-508.

[47] R. C. Arkin, *Behavior-based Robotics*: The MIT Press, 1998.

[48] R. A. Brooks, "A Robust Layered Control System for a Mobile Robot," *IEEE Journal of Robotics and Automation,* vol. 2, pp. 14-23, March 1986 1986.

[49] P. Maes, "A Bootom-Up Mechanism for Behavior Selection in an Artificial Creature," in *From Animals to Animats*, 1991.

[50] M. Lotzsch, J. Bach, H. D. Burkhard, and M. Jungel, "Designing Agent Behavior with the Extensible Agent Behavior Specification Language XABSL," in *Lecture Notes in Computer Science* vol. 3020, 2004.

[51] D. H. Edwards, "Mutual inhibition among neural command systems as a possible mechanism for behavioral choice in crayfish," *Neuroscience,* vol. 11, pp. 1210-1223, 1991 1991.

[52] R. Simmons and D. Apfelbaum, "A Task Description Language for Robot Control," in *Intelligent Robotics and Systems*, Victoria BC, Canada, 1998, pp. 1931-1937.

[53] X. Hu, "Context-Dependent Adaptability in Crowd Behavior Simulation," in *The 2006 IEEE International Conference on Information Reuse and Integration*, Waikoloa, Hawaii, 2006, pp. 214-219.

[54] C. W. Reynolds, "Steering Behaviors For Autonomous Characters," in *Game Developers Conference 1999* San Francisco, California, 1999, pp. 763-782.

[55] J. H. Laurie, K. Semyon, and Y. Shibu, "Exploring Expression Data: Identification and Analysis of Coexpressed Genes," *Genome Research* pp. 1106-1115, 1999 1999.

[56] W. Daamen and S. P. Hoogendoorn, "Experimental research of pedestrian walking behavior," *Transportation Research Record,* pp. 20-30, 2003 2003.

[57] H. Klupfel, T. Meyer-Konig, and M. Schreckenberg, "Microscopic modelling of pedestrian motion-Comparison of an Evacuation Exercise in a Primary School to Simulation Results," in *Traffic and Granular Flow*, M. Fukui, Ed. Berlin: Springer, 2003, pp. 549-554.

[58] T. Kretz, A. Grunebohm, and M. Schreckenberg, "Experimental study of pedestrian flow through a bottleneck," *Journal of Statistical Mechanics: Theory and Experiment,* pp. 10014-10034, 2006 2006.

[59] F. Qiu and X. Hu, "Modeling dynamic groups in pedestrian crowd simulations," in *Web Intelligence and Intelligent Agent Technology* Toronto, Canada, 2010.

[60] F. Qiu and X. Hu, "Exploiting Spatial-temporal Heterogeneity for Agent-based Simulation of Pedestrian Crowd Behavior," in *Modeling & Simulation of Evolutionary Agents in Virtual Worlds*, 2009.

[61] B. P. Zeigler, H. S. Sarjoughian, and H. Praehofer, "Theory of Quantized Systems: DEVS Simulation of Perceiving Agents," *Int. Jnl. Cybernetics and Systems,* vol. 31, pp. 611-648, 2000.

[62] A. Mintz, "Non-Adaptive Group Behavior," *Journal of Abnormal and Social Psychology,* vol. 40, pp. 150-159, 1951.

[63] M. R. Bloomberg and A. M. Burden, "New york city population projections by age/sex and borough," 2006.

[64] S. F. Railsback, S. L. Lytinen, and S. K. Jackson, "Agent-based simulation platforms: Review and development recommendations," *Simulation,* vol. 82, pp. 609-623, 2006.

[65] M. Lysenko, R. D'Souza, and K. Rahmani, "A Framework for Megascale Agent Based Model Simulations on the GPU," *Journal of Artificial Societies and Social Simulation (JASSS),* vol. 11, p. 10, 2008.

[66] B. P. Zeigler, R. Jammalamadaka, and S. R. Akerkar, "Continuity and Change (Activity) Are Fundamentally Related in DEVS Simulation of Continuous Systems," in *Artificial Interlligence and Simulation*. vol. 3397: Springer 2005, pp. 1-13.

[67] B. P. Zeigler, T. G. Kim, and H. Praehofer, *Theory of Modeling and Simulation, 2 ed.*: Academic Press, 2000.

[68] ACIMS, "DEVSJAVA Reference Guide," 2004.

[69] X. Hu, "A Simulation-based Software Development Methodology for Distributed Real-time Systems," in *Electrical and Computer Engineering Dept.* vol. PhD: University of Arizona, 2004.

[70] Y. Sun, "High Performance Simulation of DEVS-Based Large Scale Cellular Space Models," in *Computer Science Dept.* vol. PhD: Georgia State University, 2009.

[71] L. C. Malone, D. J. Kaup, R. Oleson, M. Rosa, F. Jentsch, T. L. Clarke, J. Faulkner, and R. Jaggie, "Validation of Crowd Simulations " in *Summer Computer Simulation Conference*, Edinburgh, Scotland, 2008, pp. 363-370.