A STUDY ON PEDESTRIAN WALKING BEHAVIOR

by

Anuj Kumar Gupta

A thesis submitted to the Faculty of the University of Delaware in partial fulfillment of the requirements for the degree of Master of Civil Engineering

Summer 2005

UMI Number: 1428195

A STUDY ON PEDESTRIAN WALKING BEHAVIOR

by

Anuj Kumar Gupta

Approved: ____

Shinya Kikuchi, Ph.D.
Professor in charge of thesis

Approved: ____

Michael J. Chajes, Ph.D.
Chair of the Department of Civil and Environmental Engineering

Approved: ____

Eric W. Kaler, Ph.D.
Dean of the College of Engineering

Approved: ____

Conrado M. Gempesaw II, Ph.D.
Vice Provost for Academic and International Programs

ACKNOWLEDGEMENTS

I am extremely grateful to my advisor Prof. Shinya Kikuchi, Ph.D., P.E., for his continuous support and guidance during my studies at the University of Delaware. I am grateful to him for involving me in various research projects with him. I am thankful to him for taking me along with him to conferences. I am also thankful to him for helping me in getting a summer internship, which gave me an opportunity to work in the real world.

I would especially like to thank Dr. Michael J. Markowski for his help during my research work as well as thesis study. I am thankful to him for allowing me to work on his CA and FEET simulation. His guidance has been extremely helpful during my work on this thesis. I would also like to thank him for the numerous meetings I had with him, which I always enjoyed.

I am thankful to Chodai, Co. Ltd., Tokyo, Japan, who sent the pedestrian video recordings and also digitized the video recordings. I appreciate the effort put in by them for sending the video recordings and preparing the digitized data.

I thank Sharat for being such a good lab mate and a friend. I have enjoyed his company throughout my stay at the University of Delaware. I thank Akash for bearing with me as a roommate for two years.

I express my gratitude towards my parents, my grandparents and my brother for giving me unending love.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

In this thesis, a study on pedestrian walking behavior is conducted. A pedestrian's walking behavior is influenced by many factors. A person's age, gender, physical disability, size, the group of which he is part of, the environment and location in which he is walking, whether he is carrying any luggage or if he is with a child, are some of the factors which influence his walking behavior i.e. the way his interactions take place with other pedestrians and his own walking speed. This thesis focuses on the influence of personal characteristics (age, gender, size, ethnicity, social status, attractiveness/ appearance and physical disability) and travel companionship characteristics (whether walking alone or as part of a group e.g. family, friends, couple, co-workers, etc) on the pedestrian's walking behavior. Interaction rules are developed for pedestrians based on their personal and travel companionship characteristics. The interaction rules based on personal characteristics are incorporated in a pedestrian simulation model called FEET.

The approach followed has been divided into two stages. Stage I is to determine the personal and travel companionship characteristics, which affect the pedestrian walking behavior, and determine the interaction rules between pedestrians based on these characteristics. The methods used under Stage I included literature review of existing pedestrian simulation models and the research done in studying pedestrian walking behavior, observing pedestrians from a video recording of people moving in an indoor auto-show obtained from Japan, calculating walking speed of different types of

pedestrians from digitized video recording data obtained from Japan, observing pedestrians in different settings like university cafeteria, shopping mall, main street and university campus, and measuring the walking speed of different types of pedestrians on campus at the University of Delaware.

Based on the literature review it was found that the pedestrian simulation models available at present did not incorporate the personal and travel companionship characteristics of pedestrians while modeling their waking behavior. All pedestrians are treated as identical. There is no consideration given to group movement. All pedestrians are assumed to be walking alone. Also, the pedestrian's scanning area in front of them was limited to only the next walking step. Empirical research has found that the pedestrians scan the area in front of them up to 10 walking steps ahead while choosing their next step.

Stage II of the thesis was to incorporate the pedestrian interaction rules developed in Stage I into the FEET simulation model. The interaction rules based on the personal characteristics were incorporated into the FEET simulation. The interaction rules based on travel companionship can be incorporated into FEET once its programming code has the objects/sub-routines for handling group movement. Also, the FEET simulation was modified so that the pedestrians scanned 3 walking steps ahead in front of them while choosing the next step.

Chapter 1

INTRODUCTION

Every person becomes a pedestrian at some point of his/her daily life. It can be a short trip from the parking lot to the office building, a shopping trip to the nearby grocery store or a long stroll along the beach. Walking is the most basic mode of transportation used by man. Walking forms a part of every trip whether it is made by a car, subway, train, or airplane. Without walking, a person cannot live a normal life. This simplistic activity of walking can result in varied experiences depending on the ambience and the people encountered while walking. This thesis presents a study that elaborates the characteristics of individuals and groups of pedestrians, which result in different walking experiences and formulates the interaction rules between different pedestrians. The results of the study are incorporated in an agent-based pedestrian simulation model called FEET.

At present, none of the pedestrian simulation models that have been developed so far, deal with different individuals or groups of pedestrians. All pedestrians are treated as identical entities, which perform in the simulation as robots instead of human beings. Such models are therefore, unrealistic and limited in their application to different situations. For example, walking in a senior citizen center can result in a much different experience compared to walking in a university student activity center. The walkways in a senior citizen center have to be designed according to the walking characteristics of senior citizens whereas the walkways in a student activity center have to be designed according to the walking characteristics of young students, to provide better mobility.

The study of walking behavior of pedestrians includes and is not limited to the study of pedestrians yielding/ not yielding to opposite moving pedestrians, following and bypassing pedestrians moving in the same direction, formation of temporary lanes of pedestrians moving in the same direction in crowded situations, weaving of pedestrians between two cross streams of people walking, and encountering different types of individuals (male, female, handicap, child, senior citizen, etc.) and groups (couple, family, friends, and co-workers, etc.).

There are a lot of factors that can affect the walking behavior of pedestrians. These factors can be classified under different categories. For example, personal characteristics like age, gender, handicap, and size; characteristics of travel companionship like family, friends, couple etc.; characteristics of the walking trip like trip purpose, and trip length; environmental characteristics; properties of infrastructure like grade, type of floor; and number of people and kind of people in the vicinity. The main focus of this thesis is to study the effects of personal characteristics and the characteristics of travel companionship on the pedestrian's walking behavior.

Chapter two discusses the need to study the pedestrian walking behavior and outlines the problem of this thesis. The shortcomings of the existing pedestrian simulation models are discussed.

Chapter three describes the approach followed in the thesis. The thesis is divided into two stages. Stage I is the study of characteristics of individuals and groups of pedestrians, which affect their walking behavior. A set of interaction rules are formulated that describe how the different individuals and groups of pedestrians interact with each other while walking. Stage II is to incorporate the findings of the Stage I into FEET simulation model to make it more realistic in modeling pedestrian walking behavior.

Chapter four provides an overview of the literature review that was done as part of Stage I of this thesis to examine the past research done in building pedestrian simulation models and the past work done in studying the pedestrian walking behavior. The literature review is divided into two parts. The first part involves studying the currently available models of pedestrian simulation. The second part deals with the past research in the field of pedestrian walking behavior.

Chapter five describes the different methods used under Stage I apart from literature review to study the pedestrian walking behavior. These include observing video recording of pedestrians walking in an indoor auto-show, calculating walking speed from digitized video recording data of pedestrians, observing pedestrians in different settings like shopping mall, university cafeteria and university campus.

Chapter six summarizes the work done under Stage I. Chapter six identifies the various characteristics that affect the pedestrian walking behavior and presents interaction rules between different types of individuals and groups of pedestrians based on the study in Stage I.

Chapter seven presents the Stage II of the thesis. An overview of the FEET simulation model is provided. The FEET simulation consists of 16 files. A brief description of each of these files is presented. Lastly, the modifications made to the FEET simulation programming code to incorporate the findings of Stage I are described. Chapter eight summarizes the work done in the thesis and presents the conclusions.

Chapter 2

PROBLEM STATEMENT

Until recently, not much thought was given to the design of transportation infrastructure (e.g. roadways, public transportation, airport etc.) to facilitate the movement of pedestrians. The main emphasis on research and development in the field of transportation has been vehicular movement, building of highways, better geometric design of roads, and management of traffic with the help of Intelligent Transportation Systems (ITS). However, with increasing population, the existing roadway infrastructure is finding it more and more difficult to cope up with the ever-increasing auto demand.

To ease the demand on the roadways, the current emphasis has been shifting towards multi-modal transportation. Multi-modal transportation involves integrating two or more different modes of transportation together for traveling purposes. Walking forms an important part of multi-modal transportation as a switch between any two modes of transportation requires walking. It can be in the form of walking in subway train station to change subway lines, or walking out from a parking lot to catch a bus. The walking part of a trip should be made as safe and comfortable as possible to entice people to walk. This is crucial for the success of any multi-modal transportation plan.

Also, there has been an increasing trend of building large-scale facilities like football stadiums, movie theatres, amusement parks etc. Such large-scale facilities require careful attention as to the ease of movement of pedestrians and particularly the management of crowd in the event of an emergency evacuation. Again, the safety and comfort of pedestrians is very important. Therefore, pedestrian studies are required for promoting walking, improving safety and comfort of pedestrians and making multi-modal transportation plans successful.

Building pedestrian simulation models has been one of the key approaches to the study of pedestrian movement and better design of facilities to offer greater mobility to the pedestrians. As is also in the case of vehicular simulation models, a plethora of pedestrian simulation models are now becoming available for the analysts. One of the most prevalent approaches to simulating pedestrian movement has been the use of agent-based modeling technique. This approach is also referred to as the "bottom-up" approach by Kikuchi, et al. [14]. Chapter 3 presents the components of an agent-based modeling technique.

An accurate or rational simulation model of pedestrian movement can be built only when all the factors that affect the pedestrian walking behavior are considered. There are a lot of factors that affect the walking behavior of pedestrians. These factors can be classified under various categories: personal characteristics like age, gender, handicap, and size; characteristics of travel companionship like family, friends, couple etc.; characteristics of the walking trip like trip purpose, and trip length; environmental characteristics; properties of infrastructure like grade, type of floor; and number of people and kind of people in the vicinity.

However, none of the pedestrian simulation models that are available at present incorporate the personal and travel companionship characteristics of pedestrians while developing the interaction rules between pedestrians. At present, all individuals are treated as identical entities in the simulation, which function no more than robots, instead of imitating the human behavior. The conflict mitigation rules for avoiding conflicts between pedestrians are based on random probabilities, in the presently available simulation models. For example, if the same target cell is chosen by more than one pedestrian, then with some probability one of the conflicting pedestrians is allowed to occupy the target cell while the rest of the conflicting pedestrians remain at their current position. Also, no consideration is given to group movement. All pedestrians are treated as walking alone. Pedestrians walking as part of a family, group of friends or co-workers are not considered in the available simulation models. As such, the characteristics of pedestrians walking in groups are not incorporated into the pedestrian simulation models.

In light of the above discussion, the problem is to study the effect of different personal and travel companionship characteristics of pedestrians on their walking behavior, and incorporating this study into a pedestrian simulation model (FEET) to make it more realistic in modeling the pedestrian movement. There are two tasks involved in this process:

- Study of pedestrian walking behavior and determining the interactions rules between different individuals and groups of pedestrians based on the personal and travel companionship characteristics of pedestrians.
- Incorporating the interaction rules formulated based on the personal and travel companionship characteristics of pedestrians into FEET, which is an agent based pedestrian simulation model.

Chapter 3

APPROACH TO STUDY PEDESTRIAN WALKING BEHAVIOR

The term pedestrian walking behavior includes pedestrians yielding/ not yielding to opposite moving pedestrians, following and bypassing pedestrians moving in the same direction, formation of temporary lanes of pedestrians moving in the same direction in crowded situations, weaving of pedestrians between two cross streams of people, and encountering different types of individuals (male, female, handicap, child, senior citizen, etc.) and groups (couple, family, friends, and co-workers, etc.).

The pedestrian simulation models available so far do not consider the personal and travel companionship characteristics of pedestrians while developing the pedestrian interaction rules. Personal characteristics of pedestrians include their age, gender, size, physical handicap, ethnicity etc., while the travel companionship characteristics include characteristics of groups e.g. family, friends, couple, co-workers etc. The current pedestrian simulation models treat all pedestrians as identical and no consideration is given to whether they are single or part of a group. This is a shortcoming of the present simulation models as it has been found from past research that pedestrians interact differently with each other based on each other's personal and travel companionship characteristics.

The aim of this thesis is to study the characteristics of individuals and groups of pedestrians, which affect their walking behavior and develop the interaction rules between them, and incorporate these interaction rules into a pedestrian simulation model to make it more realistic and versatile in application. The approach to modeling pedestrian walking behavior is divided into two stages:

- Stage I: Study of pedestrian walking behavior and determining the personal and travel companionship characteristics, which affect walking behavior and also develop the interaction rules between different types of individuals and pedestrian groups.
- Stage II: Incorporating the different characteristics and interaction rules of individuals and groups of pedestrians into a computer simulation model called FEET. This is an agent-based pedestrian simulation model.

The following sections explain the methodology of work done in the two stages of the approach.

3.1 Stage I: Study of pedestrian walking behavior

The following four approaches were adopted to study the pedestrian walking behavior:

1. Literature review: An extensive literature review of the past work in the field of pedestrian simulation models was done. Based on this literature review, it has been concluded that the work done in pedestrian simulation models so far has neglected the area of pedestrian walking behavior related to their personal and travel companionship characteristics. Therefore, a second literature review was conducted which included the study of journals related to engineering as well as various non-engineering fields like communication, sociology and psychology. In these literatures, a considerable amount of attention has been given to how

different people interact with each other while walking, based on their personal characteristics (age, gender, size, handicap etc.) and the characteristics of their walking group (whether single or part of a family, couple, friends, co-workers etc.).

2. Digitizing video recording: Video recordings of people moving in an interior hallway (part of an auto-show exhibition) were obtained from Japan. These video recordings were carefully observed and different time periods were recorded for digitizing the video. The recorded time periods were sent to Japan for digitizing. The digitized data obtained from Japan provided information on the coordinates of people moving with time. The digitized data was used to determine the walking speed of different types of pedestrians (single male/female, same and mixed gender pairs, wheelchair users, etc.) observed in the video.
3. Personal Observations: People walking in different settings were observed and their general walking pattern and interaction with other pedestrians were noticed. The different settings in which people were observed included shopping mall, downtown main street, university campus and cafeteria.
4. Measuring walking speed: The walking speeds of students on campus at the University of Delaware were measured. For this purpose, the length of a particular section of walkway was measured and a stopwatch was used to record the time that different pedestrians took to walk over that particular section of walkway. Measurements made were classified under categories based on gender and number of people in the walking group.

3.2 Stage II: Pedestrian Simulation Model

The characteristics of individuals and groups, which affect their walking behavior and the interaction rules between different individuals and groups of pedestrians as determined in Stage I, are incorporated in a computer simulation model called FEET. FEET is an agent-based pedestrian simulation model developed by Dr. Michael J. Markowski. Chapter 6 presents the various interaction rules as determined in Stage I. The FEET simulation was modified to incorporate the interaction rules developed in Stage I. The pedestrian interaction rules based on their personal characteristics were successfully incorporated into the FEET simulation. The interaction rules based on group characteristics have been clearly laid out in Chapter 6 of this thesis, and can be incorporated into the FEET simulation once the programming code for handling group movement is in place. The following section presents an overview of agent-based modeling approach and discusses why this approach is suitable for the problem at hand.

3.2.1 Agent-Based Modeling

Agent-based modeling approach is also referred to as the bottom-up approach by Kikuchi, et al. [14]. It is different from the traditional modeling approaches, which are generally top-down in nature. In the top-down approach, the objectives of the system as a whole are formulated and the components of the system work together towards achieving the common objectives. In the bottom-up approach, the smallest entities of the system are identified and assigned with certain characteristics and interaction rules. These entities are then allowed to act independently and the overall picture of the system emerges as a result of these small-scale interactions. These entities are referred to as agents in the agent-based modeling approach. Kikuchi, et al. [14] presents an extensive discussion on the agent-based modeling approach. An agent has the following five characteristics:

1. Autonomy: it can act independently on its own, without external intervention,
2. Social ability: it can interact with other agents and cooperate,
3. Responsiveness: it can perceive the environment and respond in a timely fashion to the events occurring,
4. Pro-activeness: it can take initiative whenever the situation demands,
5. Learning ability: it can improve its performance over time.

Further, the interactions between agents are characterized into four categories (see Kikuchi, et al. [14]):

1. Collaborative: autonomy and collaboration with other agents,
2. Interface and learning: changing behavior based on learning from past experience,
3. Cognitive: intention to reach explicit goals leading to choose between possible actions,
4. Reactive: acting according to a stimulus-response process, does not need any memory of experience.

The structure of agent-based modeling is divided into three categories (see Kikuchi, et al. [14]):

1. Sensor: perceives the environment and other agents
2. Cognition: controls and monitors agents' individual, communicative, and co-operative activities; includes knowledge and beliefs about the environment, a set of goals (or desires) and plans (intentions) to direct the agents' action.
3. Actuator: carries out the behavior that have visible effect on the environment.

3.2.2 Feasibility of agent-based approach to simulating pedestrians

An important observation regarding the pedestrian walking behavior is that pedestrians act at individual level or at most the group level (if they are part of a group). The pedestrian is not concerned with what is good from a global level, he/she responds to the stimuli at the local level i.e. the living and non-living things in the immediate vicinity. For example, in case of an emergency evacuation, each pedestrian will try to move ahead by pushing others aside as he is concerned with himself only, whereas from the point of maximum evacuation, it is better if all the pedestrians coordinate and evacuate forming queues.

Also, each pedestrian responds to the stimuli according to his/her personal characteristics. They act individually according to their personal discretion. As a result, a bottom-up approach is most suitable for modeling pedestrian walking behavior. Each individual pedestrian is considered as an agent of the system, with its particular set of characteristics and interaction rules with other pedestrians and the environment. The simulation world is populated with these agents and they are allowed to act autonomously, just as in the real world situation. The overall picture of the system that results from these interactions at the local level is then studied.

3.2.3 Disadvantages of Agent –Based approach

Kikuchi, et al. [14] presents a discussion on the disadvantages of using Agent-based modeling approach. The following two disadvantages of using an agent-based approach are identified:

1. The agent-based approach is not suitable for modeling situation where considerable control is required at the hands of the analyst. The idea of delegating

the tasks to the agents instead of being controlled by the analyst may not be good in some cases e.g. evacuation scenarios.

2. The calibration of parameters is not easy because how individual behavior affects the overall behaviour is not completely known. For example, how to calibrate the individual pedestrian characteristics based on flow data.

Chapter 4

STAGE I: STUDY OF PEDESTRIAN WALKING BEHAVIOR

Stage I consists of determining the characteristics which affect the walking behavior of pedestrians and the interaction rules between different types of individuals and pedestrian groups. Different methods were adopted to study the pedestrian walking behavior under Stage I. These include literature review, digitized video recording of pedestrians, observing pedestrians in different locations, and measuring walking speed of people on campus at the University of Delaware. This chapter presents the first method i.e. the literature review. The next chapter presents the rest of the methods used under Stage I.

The literature review for this thesis has been done in two parts. The first part involved the review of work done in pedestrian simulation models. This literature review helped in deciding the problem statement and the scope of the thesis. In the first part of the literature review, it was found that none of the pedestrian simulation models available so far deal with different types of pedestrians and groups of pedestrians. There is no distinction between pedestrians based on their characteristics like age, gender, disability etc. and no consideration of pedestrians being part of a group, like a couple, family, friends, co-workers etc.

The second part of the literature review was to review the work done in the study of pedestrian walking behavior. More specifically, the second part of literature review was to study literature, which identified characteristics of pedestrians, both as individuals and as in groups that affected the way they interacted with other pedestrians. The two parts of the literature review are presented in the following two sections.

4.1 Literature Review Part I: Pedestrian Simulation Models

The first step in this thesis work was to study the work done in the field of pedestrian simulation models. The findings of this study led to the determination of the problem statement and the scope of this thesis. Following is a brief summary of each literature studied in part I.

(1) Isobe, et al. [12] conducted experiments to study the pedestrian counter flow and simulated the results of the experiment using the lattice gas model proposed by Tajima, et al. [21]. The experiment involved pedestrians moving in two opposite directions starting from left and right sides of a room of width W and length L. The arrival time and mean speed of the pedestrians walking was observed for different densities of pedestrian in the experiment room and varying W and L of the room. Three types of simulation models were run:

1) Simple lattice gas model with no back-stepping (stepping backward if there are people in front blocking the way),
2) Simple lattice gas model with back-stepping,
3) Extended model incorporating back-stepping in lattice gas model proposed by Tajima, et al. [21].

In this simulation, the pedestrians walk on a room, which is a square lattice of dimensions W X L. The front region of pedestrians is divided into three parts as left, right and central, each of length M sites and width N sites (each site is a square of dimensions 0.4 X 0.4 m^2). The important assumption is that the pedestrian is more likely to move into that region which has the most number of pedestrians walking in the same direction as this pedestrian. The simulation model considers only left, right, forward and backward movement of pedestrians. The diagonal movement of pedestrians across the lattice is not modeled. Each pedestrian is assigned with probabilities of moving in different directions. These probabilities are determined based on the pedestrians in the immediate vicinity of the pedestrian under consideration. Following are the findings of the simulation models:

- Travel time increased on increasing width W of the room,
- Travel time increased on increasing length L of the room,
- With increasing N, the filing (temporary lane formation of left and right moving pedestrians) became clearer.

(2) Burstedde, et al. [3] used a two-dimensional cellular automaton model to study the pedestrian dynamics. The main concepts used in their modeling are those of Trail formation and Chemotaxis. In trail formation, the vegetation is destroyed by the walker, leaving a trail as a result. In chemotaxis, the walker leaves a chemical trace behind. In both these concepts, the main essence is that the walker leaves some evidence behind from which other pedestrians can deduce that someone has walked there before them. This idea is used to model the long-range interactions between pedestrians using a virtual trace in the form of floor fields. Two types of floor fields are used in this modeling approach: static and dynamic. The static floor field takes care of the stationary physical features of the surroundings like walls, light poles, and any other obstacles. The dynamic floor field changes with time depending on the movement of the pedestrians. The dynamic floor field also has diffusion and decay characteristics, which is vanishing of the virtual trace with time.

Interactions between pedestrians are considered to be repulsive at short distances (pedestrians want to keep a minimal inter-personal distance) and attractive at long distances. The typical space occupied by a person in dense crowd is 0.4 X 0.4 m^2, which is taken as the size of a cell. Average velocity of pedestrians is taken as 1.3 m/s based on empirical studies. Each pedestrian in the simulation has a 3 X 3 matrix of probabilities of movement in different directions. The model considers pedestrian movement in all directions (straight, sideways and diagonal). Each cell has a matrix of probabilities associated with it for the preferred movement (to handle the physical geometry of the landscape).

For each time-step in the simulation, the target cell of the pedestrian is decided based on its movement probability matrix and the matrix of probabilities of movement associated with the cell that is currently occupied by the pedestrian. If the target cell is occupied, then the pedestrian does not move. If more than one pedestrian have the same target cell, then relative probabilities with which each pedestrian chose the common target cell are considered to decide which one occupies the target cell. In such cases, only one of the conflicting pedestrians is able to move and occupy the target cell while the rest of the conflicting pedestrians do not move. Pedestrians need to move backward and sideways to avoid jamming condition.

The pedestrians are referred to as fermions and the floor field particles are called bosons. Fermions leave a trace by modifying the dynamic floor field of the cell that they just emptied. The oldest boson of each cell is destroyed with probability α. The motion is simply driven by the gradients in the floor field, i.e. gradient in the density of the bosons.

The results show lane formation for the simple case of two opposing flow of pedestrians in a rectangular corridor. This example did not include diagonal movement for simplification. Another example is given to illustrate the application of static floor field to model evacuation from an exit in a closed room. The cells near the exit doorway are given higher values of static floor field to attract pedestrians towards them.

(3) Kirchner and Schadshneider [15] extend the work done by Burstedde, et al. [3]. The approach using the concept of chemotaxis is referred to as a bionics approach. The bionics approach is used to model the evacuation process of pedestrians from a room. The concept of static and dynamic floor field is used to create a virtual trace for modeling long-range interactions for pedestrian movement. The static floor field S is used to specify regions that are more attractive (e.g. area near the exit doors). A gradient of static floor field is used to guide the pedestrians to the exit doorway. The dynamic floor field D represents the virtual trace left behind by the pedestrians and it is subject to diffusion and decay with time. The dynamic floor field is used to model the attraction between pedestrians and results in pedestrians following other pedestrians.

The authors study the significance of the static and dynamic floor field coupling constants (K_S and K_D, respectively). K_S is a measure of the knowledge of the individual regarding the physical geometry of the building, knowledge as to where the exists are in case of an emergency situation. K_D is taken as a measure of the herding behavior. If the static floor field coupling is high, then the pedestrians know the shortest path they need to follow to reach the exit and therefore, the evacuation time is minimal. If the dynamic floor field coupling is high, then it adds noise to the pedestrian movement as they follow each other. The authors suggest that some optimal combination of K_S and K_D leads to the minimal evacuation time. Some dynamic field coupling is favorable to attract the pedestrians to the cells that have been occupied by other pedestrians that are moving. Scenarios of one exit only and more than one exit are also studied.

(4) Weifeng, et al. [24] used a cellular automaton model to simulate the bi-direction pedestrian movement. The cell size is taken as 0.4 X 0.4 m^2, which is the typical space occupied by a pedestrian in a dense crowd situation. The average velocity of pedestrian is taken as 1 m/sec based on empirical studies. One time step in the simulation is approximately 0.4 second.

A simple setting is studied in which two opposing streams of pedestrians are present. The Von Neumann neighborhood is considered for movement of pedestrians (above, below, left & right cells). Different directions of movement are assigned with different probabilities. If the cell in the front is occupied by a pedestrian moving in opposite direction, then the probability of moving to the right side is greater than to the left. This is based on the fact that in some countries, the people prefer to walk on their right side as a custom. Also, the pedestrian can step backwards to avoid deadlock. The route choice and resolving conflict due to same target cell is handled at each time-step. If more than one pedestrian have the same target cell, then one of them is picked randomly with equal probability to occupy the target cell while others stay back their current cells.

Results show that critical density at which a jam occurs, increases with increasing probability of back stepping, while it decreases with increasing system size for constant back-stepping probability. The results of the simulation show some interesting patterns of right and left lane formation for the pedestrians moving in two opposite directions.

(5) Blue and Adler [2] present another cellular automaton microsimulation model for modeling bi-directional pedestrian walkways. Each cell is square shaped with a side of length 0.457 m. The maximum walking speed $v_{max.}$ is 4 cells/timestep. This implies the pedestrian can move at a maximum speed of 1.828 m/timestep. Pedestrians can be fast, standard or slow moving, which determines their aggressiveness to make lane changes. The walking speed is determined by the minimum of v_{max} and the gap ahead. The pedestrians consider lane changing at each step so as to maximize their velocity.

The authors identify three fundamental movements: side stepping, forward movement, and conflict mitigation. Side stepping is not a separate step, i.e., pedestrians change lane and move forward in the same time step, like the vehicular CA model. Two steps are considered in each time step, lane changing and forward movement. Forward movement takes place based on the velocity and the gap available in front. Conflict is avoided using two rules: laterally adjacent walkers moving in opposite directions should not side step into one another, and an empty cell between two walkers is available to one of them with 50% probability. When two opposite moving pedestrians confront each other, then their positions are swapped with a random probability of P_{exchg} (Probability of exchanging places). The swapping helps in increasing the critical density at which jamming situation occurs.

Three kinds of bi-directional flows are considered: separated (formation of left and right lanes), interspersed (no lanes formed) and dynamic multi-lane (DML) (dynamic group formations in crowded situations to avoid oncoming opposite moving pedestrians and follow pedestrians moving in the same direction).

The simulation model collects data on speed, flow, density and number of side steps per pedestrian per minute. Graphs are presented for these data for various values of P_{exchg} and directional split (probability of moving to either left lane or right lane). The results show that interspersed flow and DML flow result in 15% reduction in capacity as compared to separated lane flow. The average number of side steps per person per minute decreases as the density increases. Also, at lower densities lesser directional split (50/50) results in more side steps while at higher densities greater directional split (100/0) results in more side steps. The pedestrians can avoid deadlock and walking behind slow moving persons more often by greater directional split and therefore, move forward. The research suggests that lane changing can be both beneficial and hindering at higher densities to promote higher speeds.

(6) Helbing and Molnar [8] present a social force model for simulating pedestrian dynamics. The social forces are modeled as forces that are acting internally and guide the actions of the pedestrian. The social force model consists of attractive and repulsive forces. The pedestrian is attracted towards the destination and wants to reach there using the shortest route. At the same time, the pedestrian wants to maintain a certain distance from other pedestrians. The pedestrian experiences

repulsive forces if other pedestrians come very close to him/her. A third term is added to the social force model, called fluctuations to take care of the random variations in the pedestrian behavior.

The authors used computer simulation models, which used the social force model to simulate pedestrian behavior. In the simulation model, there are two opposite directions of pedestrian movement. The simulation shows the dynamic lane formations of pedestrians walking in the same direction to avoid the oncoming pedestrians. The number of lanes formed is found to be dependent on the walkway width with a linear relation. The authors believe that the social force model leads to a more effective pedestrian flow because of fewer occurrences of time consuming avoidance maneuvers. The authors also simulate a situation where two opposite moving pedestrian groups try to pass through a narrow door. In this situation, the simulation shows that the direction of motion at the doorway oscillates between the two opposite moving directions of pedestrians.

(7) Hoogendoorn and Bovy [11] use optimal control and differential games to simulate pedestrian flow. The pedestrians are assumed to be "autonomous predictive controllers" that minimize the subjective predictive cost of walking. According to the authors, "Pedestrians predict the behavior of other pedestrians based on their observations of the pedestrians in their direct neighborhood. As such, walking can be represented by a (non-cooperative or cooperative) differential game, where pedestrians may or may not be aware of the walking strategy of the other pedestrians".

The model presented has two components: (1) physical model, which describes the forces acting upon pedestrians when they collide, (2) control model, which describes the control decisions made by pedestrians like acceleration. The activity of walking is modeled in terms of an optimal control problem in which a performance function representing the walking cost is minimized, subject to the kinematics of the pedestrians. The kinematics of pedestrians are represented by differential equations. The walking cost includes proximity discomfort or proximity cost due to walking too close to other pedestrians or obstacles, predicted cost due to applying control (accelerating, braking and sidestepping), and cost due to drifting from planned path.

Three types of differential games are modeled: (1) walking as a non-cooperative differential game in which the pedestrian predicts the walking strategy of opposing pedestrians based on the surroundings, (2) zero-acceleration opponents in which the pedestrian assumes that other pedestrians will not respond to his/her actions, and (3) cooperative games in which the pedestrian considers not only his/her own walking cost but the walking cost of the opposing pedestrian as well.

The simulation results are used to plot the speed – density curves to study the fundamental relations between them. Lane formation in bi-directional opposing flows and the formation of strips in crossing pedestrian flows (pedestrians moving orthogonal to each other) are also studied using the simulation. The simulation results show that the " self-formation of dynamic lanes in pedestrian flow depends critically on the homogeneity of the pedestrian population, in terms of their optimal walking speeds. Moreover, the inclination to make an evasive maneuver (side stepping) rather than to decelerate upon interaction has a negative influence of the lane formation, as well as on the formation of strips in crossing pedestrian flows."

(8) Muramatsu, et al. [17] present a lattice gas model to simulate the pedestrian counter flow. The pedestrian is represented by a gas particle, which performs a biased random walk. Back stepping is not allowed in the simulation. The authors study the counter flow of pedestrians in a subway. There are two type of pedestrians considered based on the two directions of movement (towards left and towards right).

The model consists of a square lattice of dimensions W X L sites, where W is the width of the subway channel and L is the length. The authors study the dependence of critical density at which jamming occurs to the system size, drift strength, boundary density and traffic rules. Traffic rule refers to the situation where the pedestrian is more inclined to walk on his/her right-hand side (as is prevalent in some countries). It is found that critical density does not depend on the system size but it is highly dependent on the drift (biasness for a particular direction of movement) and the traffic rule.

(9) Nagatani [18] presents a mean-field model to simulate the pedestrian flow through a bottleneck. The pedestrian flow is described by a mean-field rate equation on a square lattice. This model is based on the lattice-gas model of biased-random walkers on a square lattice proposed by Muramatsu, et al. [17]. Biased-random walker implies that the walker has a preferred direction of movement. The pedestrian flow analyzed in this paper is unidirectional flow through a bottleneck.

A numerical simulation is done to study the pedestrian flow. The simulation environment involves a corridor of length L and width W_1 at the entrance and W_2 at the exit. The width of the corridor decreases from W_1 to W_2 in the middle of the corridor. Under steady state situation, the densities at the entrance and exit are assumed to be constant (ρ_{in} and ρ_{out}, respectively).

Data on mean flow rate is obtained for different values of ρ_{in} and varying widths at the exit (W_2). The simulation shows the dynamic phase transition from the free-flow phase to choking-flow phase. The author also presents an analysis to approximate the 2-dimensional pedestrian flow through a bottleneck by a one-dimensional flow.

(10) Tsuji [22] presents a numerical simulation of pedestrian flow at high densities. The author analyzes a pedestrian accident that occurred in on July 21, 2001 in Akashi, a city in Japan, west of Kobe. The author regards the pedestrian flow as similar to granular flow. This is because both the granular materials and pedestrian motions are affected by interaction forces from surrounding other granules or pedestrians, respectively. The only difference is that pedestrians have goals and they have intelligence.

The pedestrian motion is described by the Newton's second law. The force term consists of two parts: psychological term and interaction term. The pedestrians are assumed to be spherical in shape. The interaction term consists of contact forces that are assumed to have a normal and a tangential component. The tangential component results in rotational motion due to torque. The contact force is modeled using a model called soft sphere model, which has been used to calculate granular flows based on individual particle motion. The forces acting on individuals are modeled using mechanical elements like spring, dash-pot and friction slider. The psychological term consists of an influence coefficient term and a dissipation coefficient term. The influence coefficient term is assumed to depend on personal nature, contact forces and surrounding situation. The dissipation coefficient is zero if the pedestrian is not in contact with any other pedestrian.

Using this pedestrian flow model based on granular flow, the author analyzed the pedestrian accident that took place in Japan. The setting includes an L shaped pedestrian bridge of width 6 meters. The simulation results showed that contact forces were extremely high at the corner of the L shaped bridge. However, the simulation was not able to show the periodic oscillation of contact forces, which the pedestrians had actually experienced during the incident.

The literature review presented above summarizes the main approaches that have been followed so far by the researchers for modeling pedestrian movement. From this literature review, it is concluded that very little work has been done in studying the characteristics of individuals and characteristics of groups especially, which affect the walking behavior of pedestrians. Individual characteristics include age, gender, size, physical handicap, etc. while the group characteristics include whether the person is walking alone or as part of a group e.g. a family, couple, friends, co-workers etc. There are no interaction rules that have been developed based on these characteristics. The approach towards developing the conflict mitigation rules has been solely based on random probabilities. The pedestrian simulation models so far distinguish pedestrians mainly on the basis of their walking direction. This thesis aims to identify the characteristics of individuals and groups, which affect the walking behavior of pedestrians and determine the interaction rules between different individuals and groups.

4.2 Literature Review Part II: Pedestrian Walking Behavior

The second part of the literature review was to study the work done in the field of pedestrian walking behavior. The goal is to study the literature, which identifies characteristics of individuals and groups that affect their walking behavior and interaction with other pedestrians. Under this section, various studies from engineering as well as non-engineering fields like psychology, sociology and communication, are presented. Following is a brief summary of the literature review conducted in part II.

(1) One of the first literatures studied is the Highway Capacity Manual (HCM) [9]. Chapter 17 of Highway Capacity Manual [9] deals with pedestrians. The HCM specifies that the fundamental relation between speed, density and flow is the same as in the case of vehicular flow, i.e. flow = speed X density. The HCM presents relationships between speed, density and flow based on past observations of pedestrian movement by different researchers. The maximum flow has been observed to be approximately 25 pedestrians per minute per ft. of walkway width, in a subway location. From the flow vs. pedestrian space curves, it is concluded that for a flow of 25 pedestrians/min./ft., an average of 5 to 9 ft^2 of space is required for each moving pedestrian. The HCM presents the Level of Service (LOS) for pedestrian walking areas and pedestrian waiting areas separately. These LOS correspond to the space available per pedestrian in the walking area or waiting area, as the case may be.

The HCM defines clear walkway width as the portion of walkway that can be effectively used for pedestrian movement. This excludes the space occupied by persons standing/window shopping, light poles, mail boxes, parking meters etc. The HCM specifies that two pedestrians crossing should each have atleast 2.5 ft of walkway width. Also, persons knowing each other and walking close together with considerable touching due to body sway will each occupy a space of 2.2 ft.

From the speed vs. density curves, the HCM concludes that daily commuters going for work have higher walking speed as compared to shoppers. The shoppers also reduce the effective walkway width by stopping for window-shopping. Also, old and young people tend to walk slower as compared to other groups. Based on these observations, the HCM suggests that the analysts should study the walking behavior of particular groups which deviate from the regular patterns as represented in the basic speed, volume and density curves, and make provisions for them when designing the pedestrian facilities.

(2) Willis, Gier and Smith [25] observed 1038 displacements involving 3141 persons. They observed people in university restaurant and shopping malls in the greater Kansas City area. The experiments were designed to provide information on spatial displacement of people based on their gender, age, race, group size, and other characteristics like carrying luggage, infant or wearing uniform (janitor, waitress, policeman etc.). The authors considered a displacement to have occurred when one or more of the approaching individuals or groups of pedestrians was judged to have changed his/her path or body angle.

The following present the conclusions of the authors based on their experiment results. The ratio in the parenthesis at the end of each conclusion represents the ratio of number of instances observed for which the conclusion holds true to the number of instances for which it does not hold true.

- In same gender groups of equal size (1,2 or 3), males were more often displaced by females than vice versa. (79:39),
- Single pedestrians are more likely to be displaced by mixed-gender pairs than vice versa (109: 31),
- Mixed-gender pairs were more likely to be displaced by larger groups (19:7),
- Whites are more likely to be displaced by blacks (125: 85),
- Large groups displace small groups more often (424: 132),
- Younger pedestrians are displaced by older ones more often (61: 51),
- Groups comprising of younger members are more likely to be displaced by older groups (57:24),
- Adult with infant is less likely to be displaced (13: 6); similarly for pedestrians with burden (27:17),
- Pedestrians with vehicles (wheel chair or strollers) were less likely to be displaced by others (47: 7),
- Persons with obvious handicap are rarely displaced (11: 0).

The authors conclude that considering maneuverability, it is easier for small groups to move for large groups, and pedestrians to move for those with infants, handicap, and luggage.

(3) Sobel and Lillith [20] conducted experiment on the streets of midtown Manhattan to study the personal space that the individuals maintained for themselves and also the personal space that the individuals provided to others. The pedestrians walking on 34th street between Madison and Fifth Ave. in New York City, on Saturday afternoons in April 1974 were the subjects under study in this experiment. The subjects were classified based on their gender and ethnicity. Of the 116 pedestrians observed, 53% were male and 27% were black.

One male and one female were taken as the experimenters who took turns to walk on the sidewalk. The experimenters choose a particular person as the subject and walked towards him/her along a straight line for a head-on collision. A measurement grid 8 ft. wide and 25 ft. long was drawn in the center of the sidewalk. The observers recorded the distance at which the subject initially deflected from the collision line, and also the distance between the subject and the experimenter at the point when they passed each other.

The results of the experiment indicate that males are given less frontal space as compare to females. Male experimenter was given an average of 37.1 inches of personal space while female was given 55.9 inches. Males gave the least space to one another (31.2 inches) while females gave the most to each other (61. 6 inches).

Old men gave almost 3 times personal space compared to old women but the sample size of this observation was small, only 17 observations.

42 % trials resulted in brushes, indicating that the subjects did not want to give up their right of way unilaterally. This indicates the presence of a strong notion of bilateral accommodation in street behaviour. The critical spacing at which brushing occurred was observed to be 15 inches.

Based on past work, the authors also propose that less personal space is maintained for friends, peers and people with whom attitudes and political and cultural orientations are shared.

(4) Daamen and Hoogendoorn [5] conducted experiments to study the pedestrian flow through bottleneck situations. The authors summarize the different factors that affect the walking speed of pedestrians as follows:

- Personal characteristics: age, gender, size, health
- Characteristics of trip: walking purpose, route familiarity, luggage, trip length
- Properties of infrastructure: type, grade, attractiveness of environment, shelter
- Environmental characteristics: ambient and weather conditions.
- Pedestrian density.

Based on these factors, the authors selected a group of variables, which were controlled during the experiment. For different setting of the variable values, the pedestrian behavior was observed. These variables were: free or desired speed, direction of movement, density, and bottlenecks.

Two types of experiment were conducted: pedestrian flow through narrow bottleneck and four-directional pedestrian crossing flow. For the bottleneck experiment, a rectangular area of 10 X 4 m^2 was chosen, while for the crossing flow experiment, a square area of 8 X 8 m^2 was chosen. The pedestrians were divided into groups and a group leader was assigned to each group. A digital camera was mounted above the experiment location to take video recording of the experiment. The experimenters wore caps of different color based on their direction of movement and their walking speed, so that they could be easily distinguished in the video recording. The bottleneck was created using cardboards and the width was kept as either 1 meter or 2 meters. The pedestrians were walked into the experimental area group by group.

10 walking experiments were done with up to 80 pedestrians in a single experiment. The authors concluded that at low densities, pedestrians walk at the middle of the bottleneck, maximizing the distance between themselves and the walls. At high densities, two trails/lanes are formed and pedestrians tend to walk diagonally behind each other, reducing the headway and maximizing the infrastructure supply. Also, when the pedestrian walking speed was reduced, the pedestrians were less able to walk in a straight line, there is a considerable noise in their movement (zigzag movement).

(5) Cepolina and Tyler [4] present a microscopic model to study the accessibility to pedestrians with regard to the activity they wish to undertake and the abilities required by the environment to enable them to perform those activities. Accessibility is analyzed by considering 3 components: activity, environment and person/individual. In order to accomplish an activity, the environment requires an individual to have certain capability. Whether the activity is accessible by an individual depends on how the individuals' capabilities match up with the capabilities required by the environment to perform the activity.

The authors identify the following characteristics of a microscopic model for analyzing pedestrian accessibility:

- It should be able to incorporate the capabilities of the pedestrian to some detail (sight, hearing, other sensory perception, physical characteristics, use of mobility aids etc.).
- It should be able to incorporate the capability requirements of the environment for a particular activity to be performed.

- It should be able to model the interactions between a pedestrian and another object (can be static or dynamic).
- A wide variety of objects and different capabilities of individuals should be represented.

In this microscopic model, an object is modeled as a mass point, having a stone (equal to the physical boundaries of the object) at the center and a potential buffer surrounding it. Two characteristics of the potential buffer are its dimensions and the rate at which it decreases from the stone boundaries. The potential buffer is an indicator of the personal space required by the pedestrian, which depends on their psychological and behavioral characteristics. Together, the stone and the surrounding buffer constitute the perturbation function which models the objects' presence/ influence on the environment.

Two types of objects are considered in the microscopic model: reactive (e.g. pedestrians) and passive (e.g. lamppost). The reactive elements respond to the action of other elements and can avoid collision by taking action, whereas the passive elements do not respond and can only be avoided by the reactive elements to prevent collision. An awareness area is considered to model the ability of the reactive pedestrians to detect other objects. This accounts for the sight and hearing capability of the pedestrian. For a passive object, its ability to being detected is modeled by a polar buffer, related to the physical characteristics of the object. A lamppost has a larger polar buffer as compared to a trash bin. A reaction occurs when the awareness area meets the polar buffer. The pedestrian area consists of an array of cells with each cell having a capacity of the perturbation. The combined perturbation values of the active and passive objects should be less than the cell capacity at all times to be able to perform the activity.

The main emphasis is that the pedestrian and the environment should be considered together in order to determine the accessibility of the environment to the pedestrians. The authors have presented a conceptual model for determining pedestrian accessibility. This model allows representing pedestrian characteristics like size, agility, behavior, anxiety, and environment characteristics like physical shape, appearance, & visibility, which the authors have identified to affect the accessibility of the environment to pedestrians.

(6) Dabbs and Stokes [6] conducted experiments to study the effect of gender, group size and attractiveness on the personal space that is granted by pedestrians to each other. In the experiment, one male, one female, two males and a mixed-gender pair took turns to stand on a sidewalk and within one foot of the wall. In the case, of single female experimenter, she wore attractive professional clothes in some experiments and normal dress in others. The idea behind this was to determine the effect of attractiveness on the personal space that other people offered to the attractive looking female. The oncoming pedestrians were photographed and their distance from the edge of the sidewalk was measured at several points as they approached and passed by the experimenter. The video was recorded for a 40 ft. long section of sidewalk. Overall, a total of 470 pedestrians were observed during the experiments.

The experiments showed that pedestrians deviated more in their path to stay further from a male standing beside a sidewalk as compared to female (3.8 ft vs. 3.3 ft.). Also, the pedestrians deviated farther from two people as compared to one person, and farther from an attractive female as compared to an unattractive female (3.6 ft vs. 3.3 ft.). This has led the authors to conclude that gender, group size and attractiveness could be thought of as symbols of power, leading to domination of space. Also, the various sources of power are thought to be additive when determining the personal space. For example, a group of handsome men dressed professionally will be granted a lot of space.

(7) Baxter [1] conducted experiments to study the interpersonal spacing between pairs composed of same or mixed gender, different ethnicity and different age group. The ethnic groups included Anglo-, Black-, and Mexican-American people. The age group was divided into three categories: adult, adolescent and child. The authors suggest that people from different cultural background tend to utilize the geographical space differently.

The experiments were conducted in the Houston zoo, Texas. Two indoor and two outdoor settings were selected as observation locations where people coming in pairs were observed for their interpersonal spacing. An observer recorded the age group, ethnicity, and gender of the persons comprising the group. The observer also recorded the distance between the pair to the nearest one-quarter of a foot. Prior to start recording the distances, the observer was given training as to judge distances by just looking. Overall, a total of 859 pairs were observed in the experiments.

The results of the experiments showed that Mexican groups stood closest with an average interpersonal spacing of1.78 ft., Anglos were intermediate with interpersonal spacing of 2.29 ft., and Blacks stood the most distant with an average of 2.66 ft. This order was maintained in all age and gender classifications of groupings. Based on age group classification, it was concluded that children interacted most closely with an average interpersonal spacing of 1.99 ft., adolescents were intermediate with spacing of 2.28 ft., while adults interacted at greatest distances with an average of 2.46 ft. Based on gender classification, the male-female groups interact most proximally (average interpersonal spacing of 2.11 ft.), female-female group intermediate (average interpersonal spacing of 2.23 ft.) and male-male groups interact most distantly (average interpersonal spacing of 2.39 ft.). Also, the experiment results showed that Anglos interacted at almost the same distances both indoors and outdoors, Blacks interacted more closely while indoors while Mexicans interacted more closely when outdoors.

(8) Umemura, et al. [23] conducted experiments to study the path selection behavior of pedestrians in open space and build a simulation model based on the results of the experiment. The subject person in the experiment walked through a virtual room in which a virtual object is moving. The experiments were performed in CAVE, an Electronic Visualization Laboratory (EVL) consisting of a surround-screen, projection-based virtual reality system. Three starting and ending locations (center, right and left corners) were selected on two opposite walls of the room and a virtual object was made to move in the middle of the room. The velocity of the moving object was varied from slow to very fast during different runs of the experiment. The trajectory of the subject, and whether the subject collided with the moving object, was recorded during each experimental run.

From the results of the experiment, it was concluded that when safe, the pedestrians choose the path having the least walking distance. Pedestrians tend to save walking distance instead of time.

Based on the experimental data, the authors suggest a path selection strategy for pedestrians, which is divided into 2 stages:

1. Global path planning stage – In this stage, the pedestrian determines the intermediate transit position, which minimizes walking distance, estimated risk, and fluctuations.
2. Local direction control – Under this, the pedestrian determines the walking direction at every step to avoid a moving obstacle.

The above path selection strategy was used to build a computer simulation model and the parameters of the model were adjusted using trial and error to reproduce the trajectories observed in the experiment. The optimum parameter values thus obtained were used to simulate another experiment to validate the simulation model.

(9) Khisty [13] conducted experiments to study the cross-flow of pedestrians in corridors and hallways. The author notes "When corridor widths are narrow and pedestrian concentrations are high in both streams, one notices pedestrian walking speeds, particularly in the minor flow, coming to a standstill, with queues building up. In the major flow there is evidence of extremely restricted walking speed, shuffling, and frequent conflicts. One of the reasons for this condition is that corridor widths are determined generally by building codes rather than with respect to traffic demand."

For the experiment, a location with crossing corridors was selected in the Washington State University campus. Both the crossing flows were predominantly in one direction and at right angles to each other. The area was marked out so as to get data on walking speed. Video recording of the experiment was also done for further analysis. A total of 375 pedestrians were observed, of which 200 were from the major stream and 175 from the minor stream. The flow, density and speed curves were plotted for the two streams. No significant change in the major flow pattern due to minor stream was observed for pedestrian density of about 0.8 – 1.0 pedestrians/m^2. The minor flow speeds were observed to be slightly higher than the major flow speeds. The reason is that pedestrians in minor flow have to be more aggressive to cross the major stream. The author conducted a "gap analysis" to find the flow in major stream that would make it uncomfortable for the minor stream to cross the major stream.

Based on the results of the analysis the author recommends that the maximum density in cross-flow areas should be limited to 0.8 pedestrians/m^2 for design purposes. Also, the maximum flow should be taken as 75 pedestrians/min. and walking speed of 60 m/min.

(10) Miyazaki, et al. [16] studied the behavior of pedestrians while overtaking a wheelchair person. The aim of the study was to develop methods for evaluating pedestrian evacuation safety for disable persons. In the experimental setup, 30 pedestrians and one wheelchair person were asked to walk in a corridor, which had a doorway at the end. The width of the doorway was varied from 2 m to 3 m. The wheelchair user either used a mechanical wheelchair, electrically driven wheelchair, or pushed by an attendant or stayed at the same place. The wheelchair user moved in the center of the corridor. All the experiments were video recorded. The results of the experiment were used to build a numerical simulation model for a pedestrian group overtaking a wheelchair user.

The experiments showed that pedestrians start altering their path approximately 1 m behind the wheelchair person. They give a space of approximately 0.5 m on the left/right side of the wheelchair user, measured from the center of the wheelchair. Both the pedestrians and the wheelchair user tended to revert back to their initial speed if changed during the course of interaction with each other. The experiments also showed that when the wheelchair person stayed at his/her location, then the pedestrian group overtaking the wheelchair person reduced its speed remarkably. Also, the pedestrian group slowed down more for the case when the wheelchair was manually propelled by the user as compared to the case when the wheelchair was electrically driven.

The walking speed of pedestrians who walked through the doorway together with the wheelchair user or just after him/her decreased to 0.5 m/s in case of 2 m wide doorway and to 1 m/s in case of 3m wide doorway (from an initial speed of 1.5 m/s). The pedestrians recovered to their original speed of 1.5 m/s after overtaking the wheelchair.

The authors formulated 5 rules for the pedestrian walking behavior while overtaking a wheelchair user:

- Rule for dividing walking zone: The pedestrian walking zone is divided into four sections: approach, course change, overtaking and post-overtaking sections.
- Course change rule: Before going into the section of overtaking, the pedestrians change their course. If the pedestrian is on the right-hand side from center of the

wheelchair, then he/she changes course rightward, otherwise the course is changed leftward.

- Speed adjustment rule: While walking in a group, there is a comfortable walking distance between two pedestrians in which the person does not change his/her speed. If the distance between two consecutive persons is less than the comfortable walking distance, then the pedestrians decelerate, and if the distance is greater then they accelerate to catch up with the group in front.
- Avoiding action rule: The speed is adjusted to keep the comfortable distance both forward and backward.
- Cognitive activity cycle rule: Each pedestrian goes through a cognitive cycle while walking. This cognitive cycle has four steps: expect, estimate, modify and execute.

Based on the above rules, a numerical simulation was done and it was found that the result from the simulation described the phenomena observed in the experiments. The authors recommend the use of the simulation model to evaluate the evacuation safety of facilities.

(11) Hayduk and Mainprize [7] did investigations on the personal space preferences of the institutionalized and non-institutionalized blind persons. It was presumed that the visual capacities had an effect on the spatial preferences of the person. Three groups of 25 people each were formed for the experiment. The groups were of institutionalized blind, non-institutionalized blind and sighted persons. In the experiment, a person approached the subject from different directions one by one. Each time while approaching, the person counted out loud so that the subject could hear him and get an idea of how close he is. When the subject felt that the person approaching them had crossed their comfortable personal space, they would ask the person to stop. The permeability of the personal space was also investigated. The subject was asked to tell when he/she was slightly uncomfortable, medium and most uncomfortable, as a person approached them.

The experiments showed that the personal spaces of both the institutionalized and non-institutionalized blind are virtually identical in size, shape and permeability to those of the sighted. Substantial differences in age, income, education, occupational prestige, residence styles, visual abilities, social stigma histories, all failed to produce significant difference between the groups. Mean personal space was found to be 56.3 cm in the front direction, 51.8 cm on right side, 54.1 cm on left side and 50.8 cm from the rear side.

(12) Wolff [26] investigated the behavior of pedestrians moving on 42nd street between Fifth and Sixth Ave, the heart of midtown Manhattan. One male and one female experimenter were used to conduct the experiment. The experimenter selected a subject person and started walking towards him/her from a distance of 25 to 50 ft., in a straight line. All the interactions were filmed.

It was found that the median yielding distance of males/females to the male/female experimenter was 7 feet under low-density conditions (5 – 15 people in the area of encounter) and 5 feet under high-density conditions (16 – 30 people).

While no significant differences were observed at lower density, at higher density same gender yielding (male vs. male or female vs. female) occurred at shorter distances as compared to opposite gender yielding (male vs. female). Male subjects yielded to the female experimenter at greater distances as compared to female subjects yielding to the male experimenter. This indicates that females expect males to yield and males did yield.

In low-density conditions, a complete detour was taken whereas, in high density conditions a step and slide movement was undertaken to avoid the experimenter, which involved brushing. Step and slide movement was initiated when the inter-personal distance was approximately 5 feet. At higher densities, people maintain a head-over-the-shoulder relation with the person in front (who is less than 5 feet ahead). They position themselves such that the head of the following person is behind the shoulder of the leading person. While passing through clusters of people moving in opposite direction, the following person would fall exactly behind the leading person.

When the experimenter did not cooperate in the step and slide movement, then there were stares and angry remarks from the subjects, indicating that a bilateral cooperation is expected from them.

People with bags and packages yielded less and made less effort to coordinate. The author proposes that this explains why walking in shopping mall is tiring even though the pace of moving is much slower. Couples rarely yielded to single pedestrians and would maintain a longer eye contact to indicate that the single pedestrian should yield.

The experiments showed a kind of spread effect in which people walking in the same direction tend to spread out as much as possible along the width of the walkway. In case of opposing flows, the flow direction with greater volume occupies a width larger than the proportionality of the two opposing flows. The reason is that the pedestrians in the smaller flow perceive more people in the opposing flow than there are in reality.

Subjects tried to return to the original path or line-of-walk after detouring around the experimenter, even though all the lanes on the sidewalk were of equal utility for the purpose of reaching the corner. This could be attributed to “position preference” with regard to environmental boundaries. Sometimes, the subjects were prompted to act prematurely (when they were reasonably close to take action) and make a beeline after detouring towards their area of interest.

People treat perceptually different parts of pavements like gratings as obstructions. People constantly monitor the activity of others around themselves. A person walking closely behind or walking on the side for more than a short period of time are held as suspects.

Children of age seven years or less were treated as “baggage” by adult pedestrians, who appeared to drag them through the traffic. Oncoming pedestrians negotiated the right-of-way with the adult and the adult lead the child.

According to the author, “A high degree of cooperation is an intrinsic part of pedestrian behaviour – without which walking would be impossible. ….Most significant factor in affecting and modifying the behavior of pedestrians in a metropolis is the number of people in their immediate environment.”

(13) Pushkarev and Zupan [19] conducted an extensive study of pedestrian dynamics and the design of facilities for pedestrian movement. The graphs showing the relationship between speed, density, flow and pedestrian space are discussed in detail. The authors study the pedestrian behavior related to the available space. For space between 0.2 – 0.5 m^2 per pedestrian, the flow is on the verge of stopping, reverse movement is impossible, physical contact with other pedestrians is unavoidable and it is impossible to overtake other pedestrians. For space between 0.7 – 1.0 m^2 per pedestrian, the flow attains a maximum in relaxed traffic streams, the average speed is about 70% of the free flow speed, crossing and reverse movement is severely restricted with chances of collisions, overtaking is impossible and there may be physical contact with others. For space over 3.7 m^2 per pedestrian, the flow is 20% of maximum capacity, the pedestrians can walk at any speed they choose, crossing and reverse movement is unrestricted, conflicts can be avoided by maneuvering, and overtaking is also free or can be done with some maneuvering.

According to the authors, the pedestrians start taking evasive action anywhere from 2 to 17 ft. (0.6 to 5.2 m) ahead of a stationary or moving obstacle to avoid conflict. This implies that the pedestrians scan the area in front of them up to 10 walking steps ahead.

(14) Hill [10] presents a survey of the work done in social and behavioral sciences related to pedestrian walking, crossing streets and choosing routes. The author divides his study into three parts of walking, crossing streets and choosing routes. This is based on the thought that a person as a child first learns to walk, then he learns to cross the street, and finally he learns to choose his route to reach farther destinations. According to the findings of the author, men walk faster than women, the youth walk faster than the older people, and pedestrians in groups tend to walk slower as compared to unaccompanied pedestrians.

The author notes that the environments for automobile movement and pedestrians movement need to be different, “An environment comfortably simulating from a car becomes monotonously boring on foot while what is interesting on foot becomes chaotic in a car….at high speeds one needs distant views, simplicity and large-scale while at slow speeds one needs small-scale, intricacy and complexity.”

The author quotes an analysis of the “work” of walking by adult persons, which identifies three major components:

- Navigation: ability to walk without bumping into others and walking between members of an oncoming group of pedestrians.
- Recognition: ability to determine whether oncoming pedestrians are alone or part of a group.
- Production: Ability to emit signs to indicate whether one is walking alone or as part of a group.

According to the author’s findings, there are two special moments that occur when two pedestrians encounter each other. These special moments are the emission of a critical sign that tells the other pedestrian that what you intend to do next, and the establishing point or recognition by both individuals that they have exchanged critical signs. The critical sign can be anything from a straight-ahead glance to a slight turning of the shoulder, which tells the oncoming pedestrian of your intentions. Confusion regarding these messages/signs can lead to a sort of reciprocal “dance” between two opposing pedestrians, where they try to figure out who is going in which direction.

The above literature review presented a comprehensive study of the work done by researchers in studying the pedestrian walking behavior. A list of factors, which influence the pedestrian's walking behavior, is identified based on this literature review. These factors are classified under seven different categories. These categories are presented in Chapter 6 of this thesis. Also, based on the findings from this literature review, a set of interaction rules is developed for pedestrians based on their personal and travel companionship characteristics. These interaction rules are also presented in Chapter 6 of the thesis. Finally, these interaction rules are incorporated into a pedestrian simulation model called FEET, which is an agent-based simulation model.

Chapter 5

PERSONAL OBSERVATIONS

For studying the pedestrian walking behavior, people were observed in different situations while they were walking. The observations involved noticing how different persons interact with each other, how the pedestrians walk alone or in groups, and the walking speed of pedestrians. The aim of observing pedestrians was to identify walking behaviors, which are not found in the literature review, and also to observe the behaviors that have been noted in the past literature. For this purpose, the following methods were used:

1. Video recordings of pedestrians walking in an indoor auto-show were obtained from Chodai, Co. Ltd., Tokyo, Japan. The Chodai, Co. Ltd. was very helpful in providing the video recordings and also digitizing the same. These recordings were observed for pedestrian walking behavior. Specific time periods from the video recording were selected and they were sent for digitizing to Japan. The digitized data obtained from Japan provided information on the coordinates of position of selected pedestrians for different times. This information was used to determine the walking speed of selected pedestrians. The digitized data obtained from Japan is given in Appendix A of the thesis report. Appendix A also presents the walking speed data calculated from the digitized data obtained from Japan. Figure 5.1 presents a picture of the digitized video recording from Japan.

Table 5.1 below presents a summary of the walking speed information obtained from the digitized data. There was cross flow of pedestrians where the video recordings were made. Accordingly, Table 5.1 presents the data for both the major and minor streams separately. Overall, movement of 81 pedestrians were digitized. Of these 81 pedestrians, 56 are from the major stream while the rest 25 are from the minor stream.

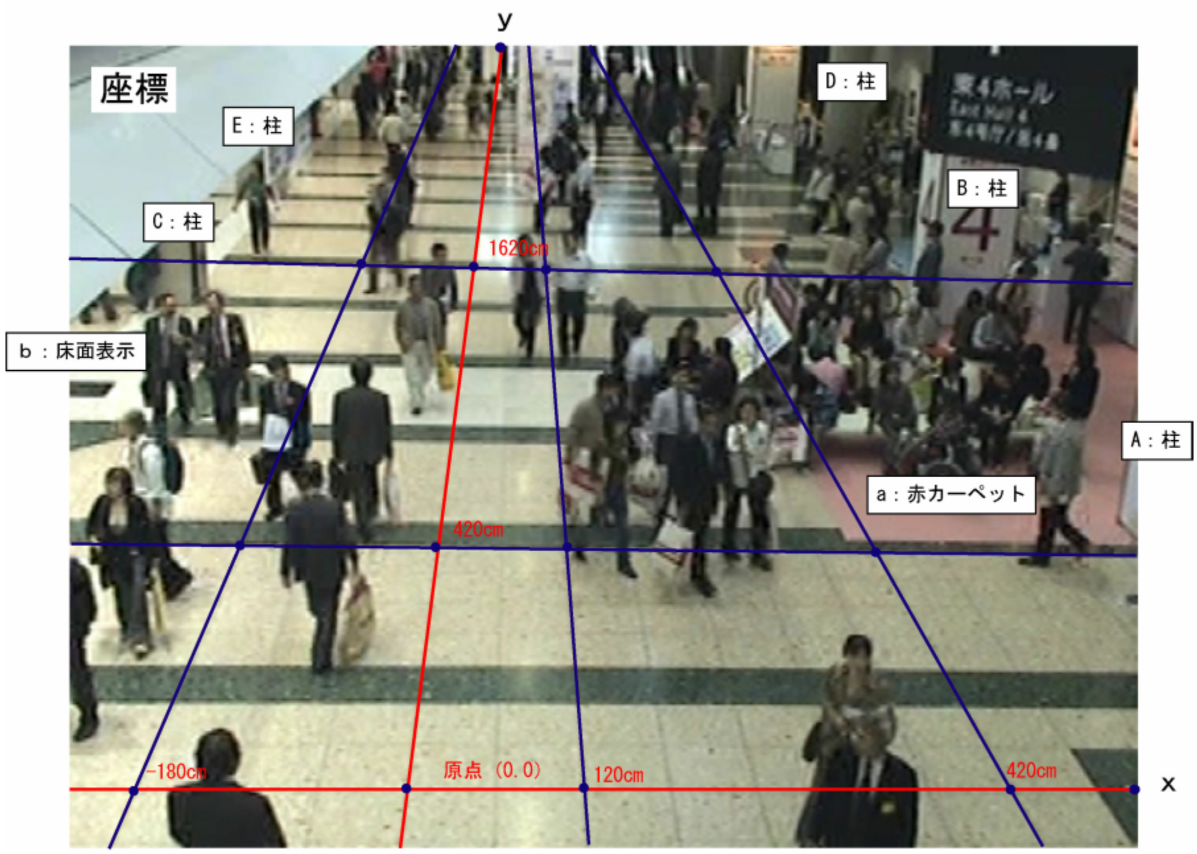


Figure 5.1 Digitized video recording of people moving in an indoor auto-show

Table 5.1 Average walking speed of different pedestrians

	Major Stream		Minor Stream	
	Average Speed (m/s)	No. of Samples	Average Speed (m/s)	No. of Samples
Single pedestrian	1.29	7	0.96	2
2 males	1.16	15	1.07	6
2 females	1.16	10	1.04	13
Couple	0.77	4	0.8	2
Female w/ bags	0.95	2		
Person pushing wheelchair	1.04	16	1.03	2
Motorized wheelchair user	1.6	2		

The following observations were made from the video recordings from Japan:

- Person in a hurry yielded more frequently and walked around the slow moving people in front to overtake,
- Female let the female with child stroller to pass first,
- Baggage carrying people made less effort to yield,
- Single pedestrians yielded to wheelchair users,
- Groups either moved closer to each other when facing opposite movement or they split resulting in a weaving motion,
- People had a tendency to follow people moving in the same direction at a reasonable distance,
- Single pedestrians yielded to groups of 2 persons,
- Speed of motorized wheelchair user was faster,
- While merging/weaving, pedestrians momentarily stopped/slowed down while waiting for gaps,

- Couple yielded to wheelchair person,
- Janitor pushing a trolley tried to be careful and avoid other people's path,
- People pushing wheelchair expected others to give way, by maintaining longer eye contact.

The following are the findings from the digitized data obtained from Japan:

- The average speed of single pedestrian is slightly higher (0.13 m/s in major stream and 0.10 m/s in minor stream) as compared to group of 2 male/female.
- The average speed of 2 male and 2 female groups is the same for the major stream (1.16 m/s). This average speed for 2 male/female groups is slightly higher for major stream as compared to minor stream (higher by 0.12 m/s).
- The average speed of person pushing wheelchair is almost the same (1.04 m/s) for both major stream and minor stream.
- Although there are only two samples, the speed of motorized wheelchair users was found to be quite high (1.6 m/s).
- The average walking speed of couples is the slowest (0.77 m/s for the major stream and 0.80 m/s for the minor stream).

2. Observations were made using the online web camera installed in the Trabant University Center at the University of Delaware. The Trabant University Center is a student activity center and includes a cafeteria, a multipurpose room, offices of various student organizations, a movie theatre and a branch of WSFS bank. The camera is installed in the main corridor of the building. The live video from the camera can be viewed online at this link: http://copland.udel.edu/stu-org/dusc/. There are kiosks along the corridor where different kinds of exhibitions and sales take place.

Frequently, people passing by stop at these kiosks. The people walking in the Trabant University Center are mostly students of the University of Delaware. Figure 5.2a and 5.2b present two snapshots from the online web camera.

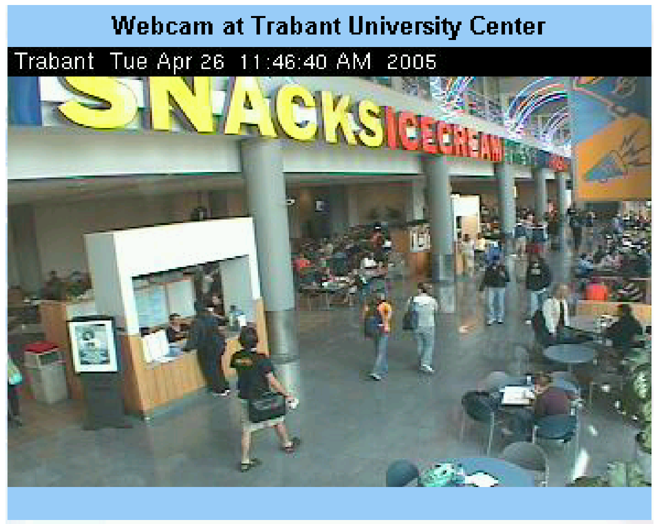


Figure 5.2a Snapshot from Trabant University Center Web Camera

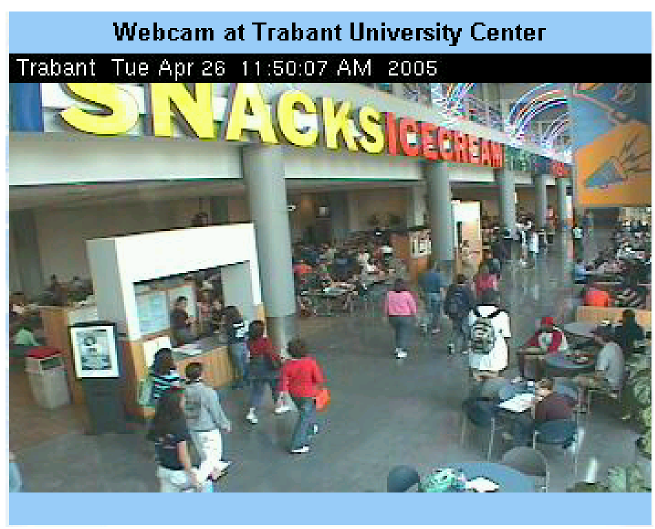


Figure 5.2b Snapshot from Trabant University Center Web Camera

The observations were made during the mid-day rush hours between 12:30 pm to 2:30 pm in the months of March and April 2005. During this time, the students come to the cafeteria for lunch. The observations made are presented below:

- People tend to walk in the middle of the passage if there is no opposing flow of pedestrians,
- Friends formed a loose group (more spacing between them) as compared to couples who walked closer together, holding each other's hand,
- People had a tendency to revert back to the middle of the passage after encountering and passing an opposite moving pedestrian/ or an obstacle (like a person standing in the corridor),
- People talking on cell-phone were less likely to coordinate with opposite moving pedestrians (resulting in other pedestrians to yield for them),
- Platoon formations were observed often, pedestrians came in large groups after almost regular intervals. This could be due to the traffic light on the Delaware Ave. & S. College Ave. intersection, from where the pedestrians are coming, and also because of students coming out from the same class.
- Some-times, a group of students would stand and chat in the middle of the busy corridor, resulting in very congested situations where the oncoming pedestrians had to slow down considerably and meander through the crowd.

3. Observations were made at the Christiana Shopping Mall located in New Castle County, Delaware. Pedestrians, mostly shoppers, were observed during the midday on weekends during the month of March 2005. The pedestrians mostly included groups of families and friends. The observations made are as follows:

- People in groups walked close to each other, almost touching.
- In groups consisting of three people, they either walked in a row or the third person walked in the center behind the other two.
- A maximum of 4 persons walked in a row.
- Kids were dragged along by adults by holding their hands.
- Generally, people moved in their right-hand side walkway.
- Occasionally people turned back to look at the displays, and as a result became an obstruction for other pedestrians.

4. Observations of people walking on campus at the University of Delaware, DE were made. These observations included observing the walking behavior and the walking speed of pedestrians. For the purpose of measuring the walking speed, measurements of different sections of walkways were taken. A concrete paved section of the walkway was selected which clearly stood out in the red colored brick paved walkway. This made it easier to determine when the pedestrian crossed the section. Figure 5.3 shows a general layout of the University of Delaware campus where the observations were made. The time taken by pedestrians to walk on this section was recorded to obtain the walking speed. The walking speed data has been classified on the basis of gender and size of the group. The people observed were adults who were either students or university faculty/staff. The observations were made during the midday, between 2:00 pm to 3:30 pm, in the month of April 2005. The weather was mostly pleasant during the observation periods. Table 5.2 presents a summary of the walking speed data collected. Appendix B presents the data on travel time from this experiment.

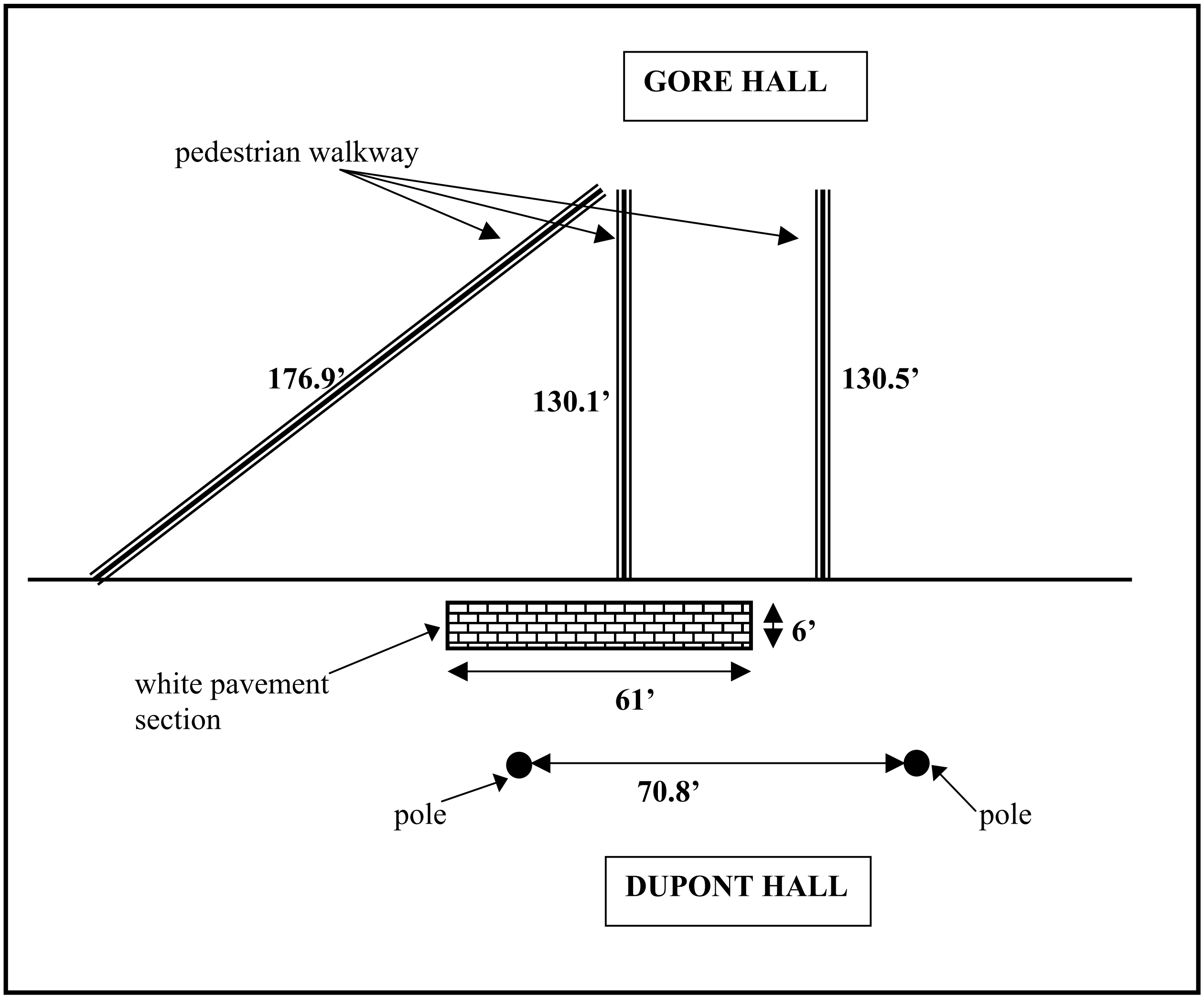


Figure 5.3 General layout of University of Delaware Green

Table 5.2 Walking speed data from on campus observations

Type of pedestrian	Average Walking Speed (m/s)	Number of samples
1 male	1.53	23
1 female	1.45	26
Couple	1.23	11
2/3 female	1.22	14
2/3 male	1.19	4

The findings from the walking speed data and observation of the pedestrians walking on campus are presented as follows:

- Single males and single females walked with almost the same speed (1.53 m/s for males and 1.43 m/s for females).
- Pedestrians in groups walk slower as compared to single pedestrians (0.25 m/s slower on an average). This is consistent with the findings of Hill [10].
- Of the people observed, groups consisting of 2 or 3 females are slowest in walking. Small number of samples for groups consisting of males restrain from making any conclusion about the walking speed of group of males.
- In large groups (more than 10 members), people in the back followed the people in front. Often the group was divided into sub-groups of 2-3 persons who were busy in their own conversations.
- Walking speed of the group was dependant on the walking speed of the group leader. This observation was recorded from observing a number of groups, which were led by university students to give a tour of the campus as part of the orientation program at the University of Delaware.

Comparing the two data sets obtained from the video recordings from Japan and the measurements on campus at the University of Delaware, Newark, DE, the following observations are made:

- The walking speed of pedestrians is generally higher on campus at the University of Delaware as compared to the pedestrians in the auto-show in Japan. One possible reason could be that pedestrians in Japan are walking in an indoor setting while the pedestrians at the University of Delaware are walking in an outdoor setting. Other reasons could be the cultural differences or differences in the personal characteristics like age and size. The pedestrians on campus at the University of Delaware are mostly young students whereas the pedestrians in the auto-show in Japan are middle-aged people. Also, there is a lot of difference between the average size of an American as compared to that of a Japanese.

- The walking speed of single pedestrians is found to be higher than that of groups (couple, 2 males/females) from both the data sets.
- While there was not much difference between the walking speed of couple and group of 2 males/females on campus at the University of Delaware, the average speed of couples was considerably slower than group of 2 males/females in the case of data obtained from Japan.

Chapter 6

PEDESTRIAN INTERACTION RULES

Based on the literature review and personal observations, the following factors that influence the pedestrians walking behavior were identified:

1. Personal Characteristics: Age, gender, size, health, disability, ethnicity, social status, and appearance/attractiveness.
2. Characteristics of trip: Purpose of trip (commuting, shopping, etc.), familiarity of route, luggage, and trip length.
3. Travel companionship: Single or part of a group (couple, family, friends, carrying infant, co-workers, etc.).
4. Properties of infrastructure: Type of pavement, grade, attractiveness of environment, and shelter (protection from weather).
5. Environmental characteristics: Ambient and weather conditions.
6. Function of the location (train station, shopping mall, amusement park etc.)
7. People in the vicinity: Number of people in the surrounding area and type of people.

As clear from above, there are a lot of factors that influence the pedestrian walking behavior and the interactions between them. Each factor forms a separate case of study and experimentation for determining its influence on the walking behavior. The literature review in Chapter 4 of this thesis presented some of these experiments that have been conducted in the past. Also, how these factors come together to influence the pedestrian walking behavior is another area of research. For example, Dabbs and Stokes [6] proposed that gender, group size and attractiveness are additive in their influence on the personal space that is granted to pedestrians. The authors observed that a group of handsome looking gentlemen would be granted a lot of personal space as compared to a single average looking male.

The interaction rules in all the pedestrian simulation models available so far, do not take into account the personal and companionship characteristics of pedestrians. The interaction rules in these simulation models are based only on the dynamics of pedestrian movement. Further, these rules are used to resolve conflicts occurring only in the next immediate time step of the simulation. For example, in case of a head-on or cross-flow collision between two persons in the next time step of the simulation, the slow moving person tries to avoid the fast moving person irrespective of their personal characteristics. In many simulations models that are available at present (Burstedde, et al. [3], Weifeng, et al. [24], Blue and Adler [2]), such conflicts are resolved by choosing one of the conflicting pedestrians to occupy the target cell using a random probability. In these simulations, no consideration is given even to the walking speed of pedestrians.

The following are some of the general walking behavior characteristics of pedestrians based on the findings of the literature review and personal observations:

- There is a strong notion of bilateral cooperation among pedestrians while encountering each other.
- A moving pedestrian on an average occupies 5 to 9 ft^2 of space.
- Two pedestrians crossing each other require atleast 2.5 ft of walkway width for each of them.

- Daily commuters walk faster as compared to shoppers.
- Pedestrians with strollers (any type: wheelchair, child, luggage) are rarely displaced by others in both head-on and cross-flow collisions.
- When walking through narrow corridors (bottlenecks), in case of low density conditions, the pedestrians walk at the middle of the corridor and stay away from the walls as far as possible.
- In case of high density conditions, pedestrians walking through a bottleneck tend to walk diagonally behind each other. The headway (distance between consecutive pedestrians) is reduced so as to maximize the infrastructure supply.
- Pedestrians tend to save walking distance instead of time while choosing their route.
- In case of cross-flow movement involving a major stream and a minor stream, the walking speed of pedestrians from minor streams is slightly higher than those from the major stream because the pedestrians in minor flow have to be more aggressive to cross the major stream.
- Pedestrians tend to take a complete detour when facing conflict in low density conditions, whereas they undertake a step and slide movement (turning the body and sliding past each other) in case of high density conditions.
- At high densities, pedestrians tend to maintain a head-over-the-shoulder relation with the person in front.
- At high densities, when passing through clusters of people moving in opposite direction, the pedestrians tend to fall behind persons moving in the same direction

resulting in the formation of temporary lanes. This is also called Dynamic Lane Formation (DML).

- People walking in same direction tend to spread out as much as possible along the width of the walkway. This is called the Spread effect. In case of opposing flows, the flow direction with greater volume occupies a width larger than the proportionality of the two opposing flows.
- The pedestrians start taking evasive action anywhere from 2 to 17 ft. (0.6 to 5.2 m) ahead of an obstacle to avoid conflict.

The main focus of this thesis has been to study how the personal characteristics and characteristics of travel companionship influence the interaction between different individuals and groups of pedestrians. Based on the literature review and personal observations (both from video recordings as well as directly observing pedestrians while walking), two sets of interaction rules have been developed. The first set of interaction rules called the Individual rules, are based on the personal characteristics of individuals while the second set of rules called the Companionship rules, are based on the characteristics of groups of pedestrians (travel companionship). These rules are presented as follows:

1. **Individual Rules**: The first set of interactions rules is based on the individual characteristics of pedestrians. Figure 6.1 presents a hierarchy of individuals based on their personal characteristics. This hierarchy structure is used to determine which individual has the right of way in case of a conflict. The hierarchy structure in Figure 6.1 holds for both head-on and cross-flow collisions.

Other characteristics like ethnicity, social status and attractiveness/appearance may also influence the interaction between different pedestrians.

The following rules are proposed based on the personal characteristics:

- Handicapped persons and senior citizens are given the highest priority. All other types of pedestrians yield to handicapped persons and senior citizens.
- Pedestrians yield to person carrying bags or luggage. A person carrying luggage has difficulty to yield and therefore, other pedestrians give way to them.
- Males yield to females in case of mixed gender encounters.
- Adolescents yield to adults.

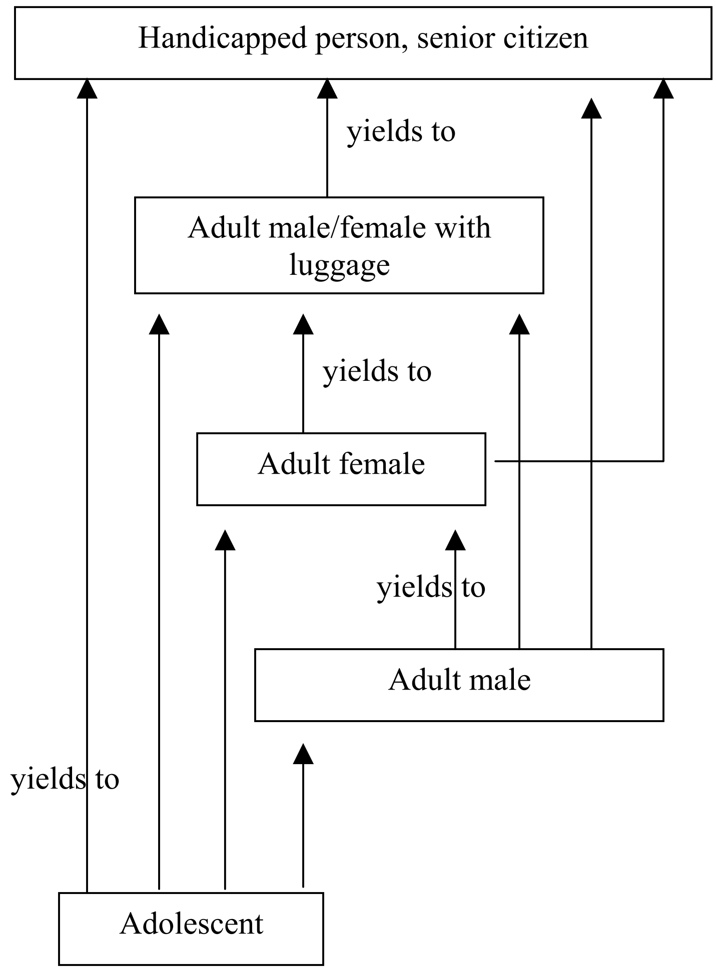


Figure 6.1 Hierarchy structure for interaction between individuals based on their personal characteristics

2. **Companionship Rules**: The second set of interaction rules is based on the travel companionship. Whether the person is alone or part of a group (and what type of group), affects his/her interaction with other pedestrians. The companionship rules are further classified into two sets. The first set presents the general walking behavior of groups, while the second set presents the rules based on the characteristics of members of the groups.

The following present the general characteristics and interaction rules for groups:

- Single pedestrians yield to group of pedestrians.
- Small groups are displaced by larger groups.
- The walking speed of the group members depends on the walking speed of the group leader.
- The walking speed of pedestrians in a group is slower as compared to walking speed of single pedestrians.
- Each group has some kind of an invisible boundary within which the group members reside. The group as a whole tries to maintain this invisible boundary while negotiating the right of way with other individuals or groups.

The following present the second set of companionship rules based on the characteristics of members of groups:

- Mixed-gender pairs are more likely to be displaced by larger groups.
- In same gender groups of equal size (1,2 or 3), males are more likely to be displaced by females.
- Groups comprising of younger members are more likely to be displaced by older groups.
- Groups comprising of senior citizens or handicapped persons are less likely to be displaced by other groups.

According to Pushkarev and Zupan [19], the pedestrians start taking evasive action (change course) anywhere from 2 to 17 ft. (0.6 to 5.2 m) ahead of a stationary or moving obstacle to avoid conflict. This implies that the pedestrians scan the area in front of them up to 10 walking steps ahead. However, all the pedestrian simulation models available so far apply the conflict mitigation rules to avoid conflicts occurring only at the next immediate walking step. These simulation models do not account for the fact that pedestrians scan the area further ahead in deciding their next course of action. For example, if a pedestrian sees an imminent collision 3 steps ahead, then he is likely to alter his path in the next step to avoid the conflict. The individual and companionship rules formulated above, are used to resolve conflicts that occur in up to 3 time steps ahead, in the FEET model. This helps in making the FEET simulation a more realistic pedestrian simulation model.

Chapter 7

STAGE II: FEET

The Stage II of the thesis consists of incorporating the pedestrian interaction rules formulated in Stage I into a pedestrian simulation model. The aim is to make the presently available simulation models more realistic. For this purpose, an agent based pedestrian simulation model called FEET is used. The agent based modeling approach is explained in Chapter 3 of this thesis report. FEET is a pedestrian simulation model that is being developed by Dr. Michael J. Markowski for his Ph.D. dissertation in Transportation Engineering at the University of Delaware. The programming code of the simulation model is written in Java language. Modifications were made to the programming code of FEET to incorporate the research done in Stage I of this thesis.

The following sections discuss the structure of FEET simulation model and the modifications made to FEET to incorporate the interaction rules developed in Stage I.

7.1 Structure of FEET

FEET is an agent based pedestrian simulation model in which the agents are the pedestrians, sources and sinks. Each agent is assigned with a set of attributes and attribute values. The attributes represent the characteristics of the agents. For example, in the case of a pedestrian agent, the attributes can be age, gender, average walking speed, maximum walking speed, physical disability etc. The programming code of FEET is divided into 16 different files. Figure 7.1 presents a flow chart to show the organization and working of the FEET simulation model.

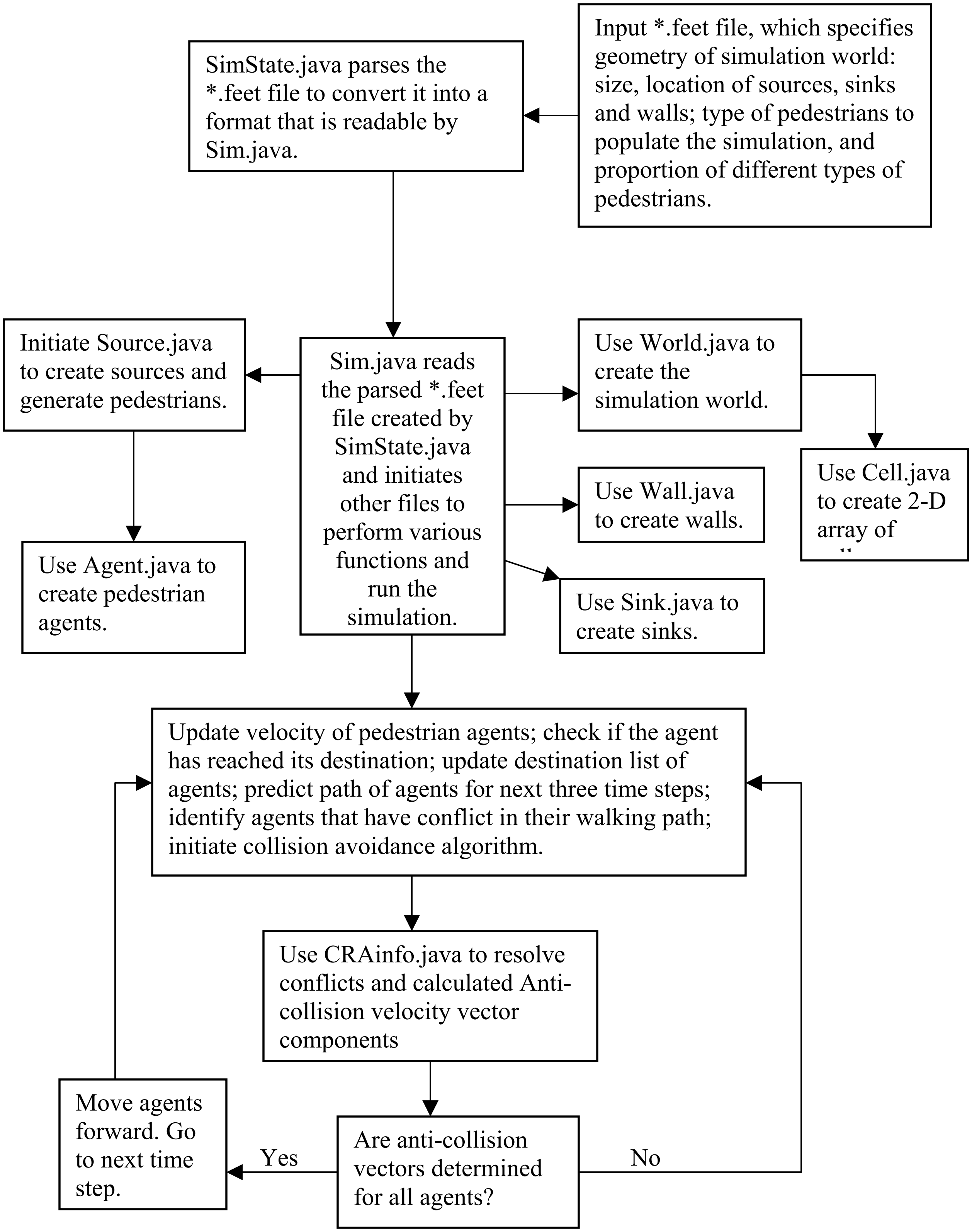


Figure 7.1 Flow chart showing organization and working of FEET simulation model

To provide an overview of FEET, its files are briefly described as follows:

1. **Sim.java**: The Sim.java is the parent file of the simulation code. This file reads in a "*.feet" file to create the simulation world. Sim.java initiates the simulation, creates sources and sinks, generates pedestrians and populates the simulation world. Sim.java makes use of almost all the other files of the simulation code. Each pedestrian agent has a list of temporary and final destinations, which determines their walking route. These destinations are updated in Sim.java to avoid bumping into walls. The walking path of each pedestrian agent is determined for the next time step based on its walking speed and next destination. In case of a conflict, the collision avoidance algorithm is initiated to recalculate the walking paths in order to avoid collision. The velocity of each pedestrian agent is updated in each time step. The velocity has four components:
   - Destination: This takes care of the pedestrian's walking speed characteristic and the direction to reach the next destination
   - Leader: If the agent is part of a group, then its velocity depends on the velocity of the group leader.
   - Collision: If the pedestrian agent is involved in an imminent collision, then the anti-collision velocity component is determined.
   - Noise: Depending on the pedestrian agent's characteristics, noise is added to its velocity vector. For example, children may have a lot of noise component reflecting their random movement.
2. **Agent.java**: The Agent.java file defines the pedestrian agents of the simulation. Each pedestrian agent has a set of attributes associated with it. These attributes

include the size of the pedestrian, the average and maximum walking speed, and the member number if the agent is a member of a group. The Agent.java file contains all the sub-routines for setting the attribute values of the agent, and also the sub-routines for getting/accessing the values of attributes of the agent.

3. **Cell.java**: The Cell.java file defines the cell of the simulation world. The simulation world is formed by a 2-dimensional array of cells. A cell can be part of a source, a sink, a wall, or just a simple cell. A cell can be either empty or occupied by a pedestrian agent. Only one pedestrian agent can occupy a cell at a particular instance of time. Each cell has a collision resolution information file of its own. This file is called the CRAinfo.java in the programming code, and is discussed separately later ahead.
4. **Source.java**: The Source.java file defines the source agents in the simulation. The source cells generate the pedestrian agents and push them into the simulation world. At present, the pedestrians are generated according to a uniform distribution. The pedestrian agents can also be generated according to a Poisson distribution. The source generates pedestrian agents and forms a queue. The pedestrian agents from the source queue are then popped up into the simulation world. When the pedestrian agent is generated, it is assigned with a destination sink and a set of attributes as specified in the "*.feet" file.
5. **Sink.java**: The Sink.java file defines the sink agents of the simulation. The coordinates of the mid-point of the sink are calculated in this file and used as destination coordinates for assigning to the pedestrian agents.

6. **Wall.java**: The Wall.java file defines the walls in the simulation. The walls are used to define the physical boundaries of the simulation world. The openings in the walls are either sources or sinks. The walls can also be used to add obstacles like a pillar or a water fountain to the simulation world. The pedestrian agents cannot pass through the walls. If a pedestrian agent encounters a wall, then its route is altered by adding new temporary destinations to the agent's destination list so that it goes around the wall.
7. **Attributes.java**: The Attributes.java file defines different types of attributes like integer and string. The Attributes.java file contains all the sub-routines for retrieving attribute values. Sub-routines for adding attribute values to an object are also included in this file. Each agent has a set of attributes. Each attribute has access to all these sub-routines defined in the Attributes.java file.
8. **AttributeValue.java**: The AttributeValue.java is a short file, which defines two different types of attribute values: integer and string. The sub-routines for setting the attribute values and obtaining the assigned attribute values are also included in this file.
9. **CRAinfo.java**: Each cell has a CRAinfo.java file or CRAinfo object associated with it. The CRAinfo.java file is updated for each cell in every time-step. It contains the list of pedestrian agents that will occupy a particular cell in the next time-step. If more than one pedestrians are present in the list, then it implies that there is an imminent collision in the cell in the next time-step. To avoid the collision, the CRAinfo.java file has a sub-routine, which calculates the anti-collision vectors for all the colliding pedestrians. The first step to calculate the anti-collision vectors is to

prioritize the pedestrian agents that are involved in the collision. For prioritizing, a fraction is calculated for each pedestrian, which is the ratio of the distance from the starting point to the collision point to the distance from starting point to the ending point (point where the pedestrian would be at the end of the time-step if there were no collision). The pedestrian that has the highest fraction value is given the lowest priority. The assumption here is that the pedestrian agent that collides latest in its trip has the most time to react and slow down or change its path.

The pedestrian with the highest priority is allowed to retain its original course of action, while the other lower priority pedestrian agents alter their respective paths or slow down. For the two highest priority pedestrians, it is then determined whether the collision between them is head-on, bumping from behind, or crossing each other at an angle. Accordingly, their paths are altered to avoid collision. If the collision is head-on, then both the pedestrian agents are moved 0.7 times their body width to their right side to avoid conflict. If the pedestrian is bumping into another person from behind, then the person who is behind is moved 2 times his body width to the side to avoid conflict. Finally, if the collision occurs because of pedestrians crossing each other at an angle, then the pedestrian agent that has the higher priority is given the right of way, while the next lower priority pedestrian walks towards the current position of the higher priority pedestrian. For the rest of the colliding pedestrians, their walking speed is slowed down to avoid conflict.

10. **World.java**: The World.java file defines the simulation world. It includes the dimensions of the 2-D array of cells. It also contains a list of all the sources, sinks and walls that are present in the simulation world.

11. **Vector.java**: The Vector.java file includes all the sub-routines that are needed for vector algebra. This includes adding and subtracting two vectors, multiplying vectors by a scalar quantity, determining the angle between two vectors, calculating a unit vector, determining an orthogonal vector, and forming a vector if the endpoints are provided. The sub-routines defined in the Vector.java file are extensively used in determining the anti-collision vectors in the CRAinfo.java file.
12. **Feet.java**: Feet.java file is used for running the simulation in the Java applet. The animation is started and stopped using the Feet.java file.
13. **FeetMenuBar.java**: The FeetMenuBar.java file is used to draw the editor tool bar on the java applet screen. When the simulation applet is loaded, the editor tool bar is used to read in the *.feet file. It also provides various other options like toggling grid display, toggling velocity and destination vector display, pausing the simulation etc.
14. **SimPane.java**: SimPane.java contains all the graphic details of loading the java applet in which the FEET simulation runs.
15. **SimState.java**: The SimState.java file parses the *.feet file that is input to the FEET simulation. The function of SimState is to convert the *.feet file into a format that is readable by the programming code of FEET simulation.
16. ***.feet**: The *.feet is a file that the simulation has to read in every time it is started. The *.feet file specifies the size of the simulation world, the coordinates of the various sources, sinks and walls, the type of pedestrian agents that the sources will generate, and with what probability these different types of pedestrian agents will be generated. The SimState.java reads in the *.feet file and parses it to convert it into a format that is readable by Sim.java. The analyst can modify the *.feet file to change

the geometry of the simulation world, the number of sources, sinks, walls and any obstacles and the location where they are placed. The analyst can also modify the type of agents that the source will generate and also the probabilities with which the different types of agents are generated. Which sink is to be used by which type of agent and with what probability, can also be specified using the *.feet file. Also, for modeling different scenarios or physical settings, the analyst has to just change the *.feet file.

The above section presented a brief overview of the functioning of the FEET simulation. The above description is based on the state of the programming code before any changes were made to incorporate the pedestrian interaction rules developed in Stage I. The following section elaborates on the changes that were made to the FEET simulation to incorporate the findings of Stage I.

7.2 Modifications to FEET

Changes were made to the programming code of FEET to incorporate the pedestrian interaction rules developed in Stage I. The files that were modified are *.feet, Agent.java, Source.java, Sim.java, and CRAinfo.java. Appendix C of the thesis presents these modified files. The following describes the changes that have been made in each of these five files:

1. ***.feet**: Initially, the *.feet file specified only one type of pedestrian agents i.e. commuters. Also, the only attributes associated with commuters were their size in terms of their length and width and walking speed in terms of average and maximum speed. The size here refers to the space occupied by a commuter in a densely crowded situation. No groups of pedestrians are generated. All pedestrians are considered as

individuals walking alone. At present, this is the state of most of the pedestrian simulation models available. Very few attributes are assigned to the pedestrians and everyone is assumed to be identical.

The *.feet file was modified to include the age, gender and handicap characteristics of pedestrians. The *.feet file now specifies different types of pedestrians like adult male, adult female, adolescent, and handicap/wheelchair user. Also, the *.feet file now specifies families, which can consist of adults, children and handicaps together. The characteristics of a pedestrian agent (speed, size, age, gender, and handicap) are specified according to the type of pedestrian (male, female, child, wheelchair user, and family). At present, for simplicity only family is generated as a group. Other groups like friends, co-workers etc. can also be easily generated using the modified *.feet file.

Different types of pedestrians and family groups have different probabilities of being generated in the simulation. The *.feet file can be easily modified to control the composition of the pedestrian agent population in the simulation world. For example, the file can be modified so that only adults populate the simulation world, or such that there is higher percentage of pedestrians that are part of families. Also, different destinations can be assigned to different types of pedestrian agents. For example, the wheelchair users can only use the exit that has a ramp. So, the probability of the wheelchair users with which they can use the exit with ramp can be set to 1, while the probability of using all other exits is set to 0.

2. **Agent.java**: The Agent.java file initially identified the size and walking speed as the only attributes of the pedestrian agents. The file has been changed to include the age,

gender, and handicap in the list of attributes associated with the pedestrian agents. Sub-routines have been added to the Agent.java file for setting the attribute values of age, gender, and handicap attributes of the pedestrian agents. The corresponding sub-routines for getting/accessing the values of these new attributes have also been included.

A new variable called the Collision Time Step (CTStep) has been defined in the Agent.java file. The CTStep stores the time step in which an agent has an imminent collision. In the modified FEET, at every time step, each agent's walking path is determined for the next 3 time steps. If an agent has a collision in any of the next 3 time steps, then the time step in which the agent has its first collision is stored in the agent's CTStep variable. The CTStep variable tells how far ahead the pedestrian agent will collide in its walking path. This information is useful for formulating the future collision avoidance algorithm.

3. **Source.java**: The Source.java file was modified to read the age, gender and handicap attributes specified in the *.feet file and generate agents accordingly. Initially, the Source.java file was generating pedestrian agents with only size and walking speed attributes. The modified Source.java file generates pedestrian agents with size, speed, age, gender and handicap attributes.
4. **Sim.java**: The Sim.java file initially determined the walking path of pedestrian agents for just the next immediate time step. Based on this walking path, the collision avoidance was performed to resolve all the imminent conflicts that occur in the next time step. This is the state of almost all the pedestrian simulations that are available so far. The pedestrian agents are concerned with only what is going to happen in the

next immediate time step. However, past studies have found that this is not the case in reality. According to Pushkarev and Zupan [19], the pedestrians start taking evasive action (change course) anywhere from 2 to 17 ft. (0.6 to 5.2 m) ahead of a stationary or moving obstacle to avoid conflict. This implies that the pedestrians scan the area in front of them up to 10 walking steps ahead while determining their course of action in the next immediate step.

A new variable called "maxTimestep" is defined which tells the simulation up to how many time steps ahead should the walking path be determined for each pedestrian. At present the value of maxTimestep is set to 3 time steps. The Sim.java file has been changed to calculate the walking path of each pedestrian agent for next 3 time steps ahead. Based on these walking paths, the agents that are facing an imminent collision in the next 3 time steps are identified. The time step in which a particular pedestrian agent is facing its first conflict is recorded (using the variable CTStep defined in Agent.java file). The walking path of the agent beyond the first point of conflict is deleted. It is assumed that a pedestrian agent is only concerned with the next immediate collision it is facing, and will change its path to avoid that collision. Finally, the collision avoidance sub-routine is initiated to make anti-collision velocity vectors. This is done by initiating the mkAntiCVecs( ) sub-routine in CRAinfo.java file. This is explained in detail under the next heading.

5. **CRAinfo.java**: Majority of the sub-routines present in CRAinfo.java file have been modified. As explained previously, each cell has a CRAinfo object associated with it. Initially, the CRAinfo object for each cell contained the list of pedestrian agents that will occupy that cell in just the next time-step. The CRAinfo.java file has been

modified to store the list of agents that are going to occupy a particular cell in the next three time steps. For each of the next three time steps, a separate list of pedestrian agents is created which stores the pedestrians that are going to occupy the cell in those time steps. If more than one pedestrians are present in any of these lists, then it implies that there is an imminent collision in the cell in the time-step corresponding to the list that has more than one pedestrian agents in it.

To avoid collision, the conflicting pedestrians are first prioritized to determine which one has the right of way and which ones have to yield. Earlier, the pedestrians were ranked based on the assumption that the pedestrian agent that collides latest in its trip has the most time to react and slow down or change its path. No consideration was given to the characteristics of pedestrians e.g. age, gender, handicap etc. In the modified CRAinfo.java file, the way in which the pedestrians are ranked has been changed. For collisions that occur in the next immediate time step, the same sub-routine as explained above is used for prioritizing the colliding agents. For collisions that occur in time steps after the next immediate time step, a new algorithm is used for prioritizing. This algorithm considers the characteristics of agents that are colliding e.g. their age, gender, and physical condition (whether handicapped). The algorithm is based on the hierarchy structure presented in Figure 6.1 in Chapter 6 of this thesis. Senior citizens and handicaps are given the highest priority, then comes adult females, which are given priority over adult males, and finally the adolescents are placed at the bottom of the hierarchy structure.

Again, similar to the previously used sub-routine, for the two highest priority pedestrians, it is determined whether the collision between them is head-on, bumping from behind, or crossing each other at an angle. For the case of a head-on collision in any time step following the immediate next time step, the higher priority pedestrian is allowed to retain its course of action while the lower priority pedestrian is moved 2 times the body width on its side (if the head-on collision occurs in the next immediate time step, then both the pedestrians are moved 0.7 time their body width to the side). This is based on the finding that certain type of pedestrians tend to yield to some other type of pedestrians, instead of both yielding to each other (Willis, Gier and Smith [25], Sobel and Lillith [20], and Wolff [26]).

As the CRAinfo object now stores three pedestrian lists for the next three time steps instead of just one list for the next immediate time step, a lot of other sub-routines were modified to account for three pedestrian lists. The modifications have been made such that the analyst can easily change the number of time steps ahead he/she wants to look at while determining the pedestrians walking paths. At present, it has been set to three time steps ahead.

The above discussion presented the changes made to the programming code of FEET simulation model to incorporate the pedestrian interaction rules developed in Stage I of the thesis. The program has been modified so that the pedestrian agents now scan three walking steps ahead while choosing their next walking step. This makes the simulation model more realistic. Also, it is believed that the collision avoidance will become smoother as a result of pedestrians scanning three steps ahead. The pedestrian agents will now start to change their path at an early stage to avoid collision. Therefore, they will not have to make any sudden maneuvers. The pedestrians can detect a crowded area from farther away and can avoid it.

Chapter 8

SUMMARY AND CONCLUSIONS

This thesis presents a study on the pedestrian walking behavior. The various approaches used for developing pedestrian simulation models were studied. It was found that the existing pedestrian simulation models did not incorporate the personal characteristics of pedestrians and the characteristics of groups of pedestrians while modeling the walking behavior. A comprehensive study was done to identify the various factors that influence the pedestrian walking behavior. These factors were classified under the following categories:

1. Personal Characteristics
2. Characteristics of trip
3. Travel companionship
4. Properties of infrastructure
5. Environmental characteristics
6. Function of the location
7. People in the vicinity

Out of this list of categories, the personal characteristics (age, gender, size, handicap, and ethnicity) and travel companionship (single, family, couple, friends, co-workers etc.) formed the main focus of this thesis. Interaction rules between pedestrians were developed based on their personal and travel companionship characteristics. The interaction rules based on personal characteristics were incorporated into a pedestrian simulation model called FEET. This made the FEET simulation model more realistic and versatile in application. The FEET simulation model can be used to analyze different situations e.g. subway train station, senior citizen center, shopping malls etc. For analyzing different situations, the type of pedestrians populating the simulation can be changed to best fit the scenario. For example, for simulating the morning or evening rush hours at a subway train station, the population can be composed of mainly adult males and females.

The approach followed under the thesis was divided into two stages. Stage I included the study of pedestrian walking behavior to determine the personal and travel companionship characteristics, which affect walking behavior and also the developing of interaction rules between different types of individuals and pedestrian groups. Four different methods were used under Stage I. These included literature review, observing video recordings of pedestrians walking and calculating walking speed of different types of pedestrians using digitized data obtained for the same video recording, observing pedestrians in shopping mall, university cafeteria, main street and university campus, and taking walking speed measurements of pedestrians walking on university campus.

Stage II of the thesis consisted of incorporating the different characteristics and interaction rules of individuals and groups of pedestrians into a computer simulation model called FEET. FEET is an agent-based pedestrian simulation model that has been developed by Dr. Michael J. Markowski working at the University of Delaware. The programming code of FEET simulation was modified to incorporate the interaction rules based on the personal characteristics of the pedestrians. The interaction rules based on group characteristics have been clearly laid out in Chapter 6 of the thesis. These rules can be incorporated into FEET after the objects/sub-routines for handling group movements are built into its programming code.

As a result of this thesis, two major shortcomings of the current pedestrian simulation models have been addressed. The first shortcoming is the lack of consideration of pedestrian and group characteristics in modeling the pedestrian walking behavior. Two sets of interaction rules based on personal characteristics and travel companionship characteristics have been developed, which are presented in Chapter 6. The interaction rules based on personal characteristics have been incorporated into the FEET simulation model. The FEET simulation model is more realistic and can be used to simulation different situations (e.g. subway train station, shopping mall, senior citizen center etc.). The composition of the population in the simulation world can be varied to best fit the scenario being analyzed, and accordingly the interaction rules will change.

The second shortcoming is that the current pedestrian simulation models treat the pedestrians as near-sighted people who decide to alter their path only when there is a conflict in the next immediate step. Past research has found that the pedestrians scan the area in front of them up to 10 steps when choosing their course of action in the next immediate step. The FEET simulation model was modified so that the pedestrians scanned the area in front of them 3 steps ahead while deciding their next walking step. As a result of this, the collision avoidance will become smoother. The pedestrian agents will now start to alter their path at an early stage to avoid collision. Therefore, they will not have to make any sudden maneuvers to avoid bumping into one another. Also, the pedestrians can detect obstacles from further away and can more easily avoid it. For example, a pedestrian can detect a crowded area or a wall from farther away and stay away from it. This will also reduce the chances of an agent coming too close and getting stuck in a crowd or keep bumping into a wall again and again.

8.1 Future Study

This thesis conducted a study on the pedestrian walking behavior to determing the interaction rules between pedestrians based on their personal and travel companionship characteristics. However, the question of how different characteristics together influence the interactions between pedestrians is an area of future research.

The factors that influence the pedestrian walking behavior have been classified under seven different categories. This thesis concentrated on two of these categories, namely, personal and travel companionship characteristics. The remaining five categories also form an area of future study. The factors under these five categories should also be included in the interaction rules to accurately model the pedestrian walking behavior and make the pedestrian simulation models more realistic.

REFERENCES

[1] J. C. Baxter. Interpersonal Spacing in Natural Settings. *Sociometry*, Vol 33, No. 4, 1970, pp. 444-456.

[2] V. J. Blue, and J. L. Adler. Cellular Automata Microsimulation for Modeling Bi-directional Pedestrian Walkways. *Transportation Research Part B*, Vol 35, 2001, pp. 293-312.

[3] C.Burstedde, K. Klauck, A. Schadschneider, and J. Zittartz. Simulation of Pedestrian Dynamics Using Two-dimensional Cellular Automaton. *Physica A*, Vol 295, 2001, pp. 507-525.

[4] E. M. Cepolina, and N. Tyler. Microscopic Simulation of Pedestrians in Accessibility Evaluation. *Transpoertation Planning and Technology*, Vol 27, No. 3, June 2004, pp. 145-180.

[5] W. Daamen, and S. Hoogendoorn. Controlled Experiments to Derive Walking Behavior. *European Journal of Transport and Infrastructure Research*, Vol 3 (1), 3002, pp. 39-59.

[6] J. M. Dabbs, Jr., and N. A. Stokes, III. Beauty is Power: The Use of Space on the Sidewalk. *Sociometry*, Vol 38, No. 4, 1975, pp. 551-557.

[7] L. A. Hayduk, and S. Mainprize. Personal Space of the Blind. *Social Psychology Quarterly*, Vol 43, No. 2, 1980, pp. 216-223.

[8] D. Helbing, and P. Molnar. Social Force Model for Pedestrian Dynamics. *Physical Review E*, Vol 51, No. 5, 1995, pp. 4282-4286.

[9] *Highway Capacity Manual Special Report 209*, Transportation Research Board, National Research Council, Washington D.C., 1985.

[10] M. R. Hill. *Walking, Crossing Streets, and Choosing Pedestrian Routes: A Survey of Recent Insights from the Social/Behavioral Sciences*, University of Nebraska Studies: new series no. 66, 1984.

[11] S. Hoogendoorn, and P. H. L. Bovy. Simulation of Pedestrian Flows by Optimal Control and Differential Games. *Optimal Control Applications and Methods*, Vol 24, 2003, pp. 153-172.

[12] M. Isobe, T. Adachi, and T. Nagatani. Experiment and Simulation of Pedestrian Counter Flow. *Physica A*, Vol 336, 2004, pp. 638-650.

[13] C. J. Khisty. Pedestrian Cross Flow Characteristics and Performance. *Environment and Behavior*, Vol 17, No. 6, 1985, pp. 679-695.

[14] S. Kikuchi, J. Rhee, and D. Teodorovic. Applicability of an Agent-based Modeling Concept to Modeling of Transportaion Phenomena. *TRB ID 02-2539*, Washington D.C., 2002.

[15] A. Kirchner, and A. Schadschneider. Simulation of Evacuation Processes Using a Bionics-inspired Cellular Automaton Model for Pedestrian Dynamics. *Physica A*, Vol 312, 2002, pp. 260-276.

[16] K. Miyazaki, H. Matsukura, M. Katuhara, K. Yoshida, S. Ota, N. Kiriya, and O. Miyata. Behavior of Pedestrian Group Overtaking Wheelchair User. *Pedestrian and Evacuation Dynamics 2003*, University of Greenwich, London, 2003.

[17] M. Muramatsu, T. Irie, and T. Nagatani. Jamming Transition in Pedestrian Counter Flow. *Physica A*, Vol 267, 1999, pp. 487-498.

[18] T. Nagatani. Dynamical Transition and Scaling in a Mean-field Model of Pedestrian Flow at a Bottleneck. *Physica A*, Vol 300, 2001, pp. 558-566.

[19] B. Pushkarev, and J. M. Zupan. *Urban Space for Pedestrians*, The MIT Press, Cambridge, Massachusetts, 1975.

[20] R. S. Sobel, and N. Lillith. Determinants of Nonstationary Personal Space Invasion. *The Journal of Social Psychology*, Vol 97, 1975, pp. 39-45.

[21] Y. Tajima, K. Takimoto, and T. Nagatani. Pattern Formation and Jamming Transition in Pedestrian Counter Flow. *Physica A*, Vol 313, 2002, pp. 709-723.

[22] Y. Tsuji. Numerical Simulation of Pedestrian Flow at High Densities. *Pedestrian and Evacuation Dynamics 2003*, University of Greenwich, London, 2003.

[23] H. Umemura, H. Watanabe, and K. Matsuoka. Investigation of the Model for Path Selection in Open Space. *Electronics and Communication in Japan*, Vol 88, No. 1, 2005, pp. 18-26.

[24] F. Weifeng, Y. Lizhong, and F. Weicheng. Simulation of Bi-directional Pedestrian Movement Using a Cellular Automata Model. *Physica A*, Vol 321, 2003, pp. 633-640.

[25] F. N. Willis, Jr., J. A. Gier, and D. E. Smith. Stepping Side: Correlates of Displacement in Pedestrians. *Journal of Communication*, Autumn 1979, pp. 34-39.

[26] M. Wolff. Notes on the Behavior of Pedestrians. *People in Places: The Sociology of the Familiar*, Praeger Publishers, New York, 1973.

Appendix A

DIGITIZED VIDEO RECORDING DATA FROM JAPAN

Appendix A presents the digitized video recording data obtained from Chodai, Co. Ltd., Tokyo, Japan. The video recordings were observed and different time periods were recorded for digitizing. The time periods were selected so as to obtain information on walking speed of different pedestrians and groups. These included single male/female, wheelchair user, person pushing wheelchair, couples, and groups of males/females. The recorded time periods were sent to Japan for digitizing. Tables A1 and A2, and Figures A.1 and A.2 present the digitized data obtained from Japan.

Figures A.1 and A.2 present the scenes from the pedestrian video recording that have been digitized. For each scene, the pedestrian whose movement was digitized is highlighted with a circle.

In Tables A1 and A2, the column SQ gives a sequence number, while T gives the time in the video recording corresponding to the scene being digitized. H, M, S, F represent hour, minute, second, and frame, respectively, corresponding to the time of digitizing. The Scene column in Table A1 presents the scene number for which the digitized data is given. These scene numbers corresponds with the scene numbers given in Figures A.1. In table A2, the time given at the beginning of each data set corresponds to the time given in the digitized scenes given in Figure A.2. The columns IX and IY represent the virtual coordinates of the person as seen in the digitized picture. The columns RX and RY represent the coordinates of the person in the real world, in cm. The coordinates RX and RY are obtained by using a mathematical transformation on IX and IY. The mathematical transformation was developed by Japan.

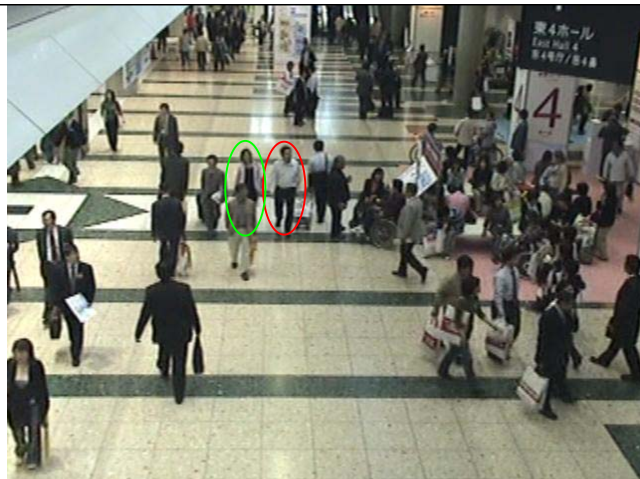
SCENE	Scene of video recording that is digitized	Starting Time	Ending Time	ID of person being digitized
01		0□00□00	0□00□00	□□□0 □□□1
02		0□00□15	0□00□18	□□□0 □□□1

Figure A.1 Scenes from the video recording used for digitizing

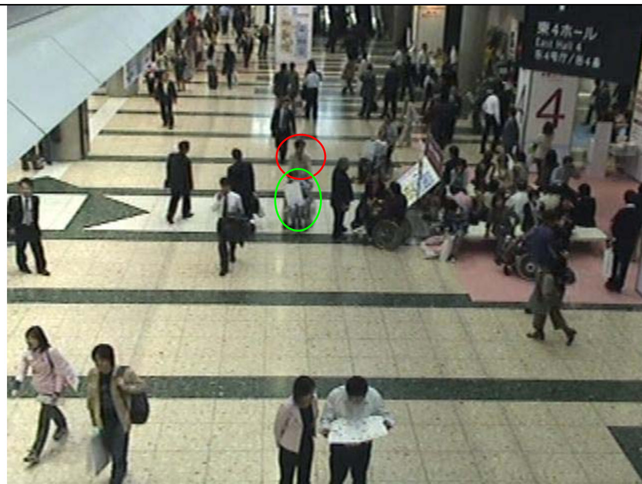
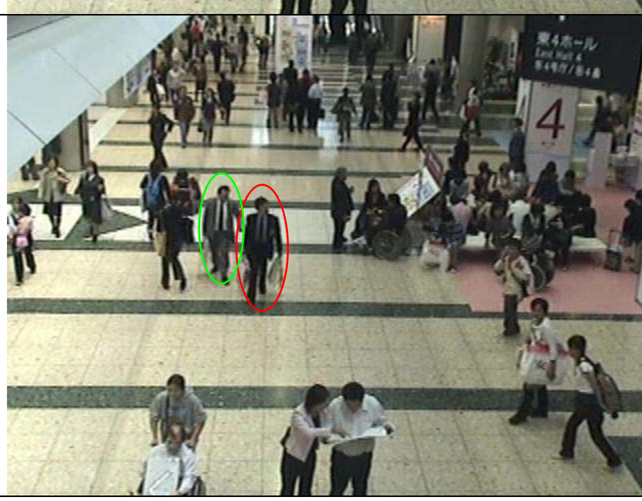
03		0□00□42	0□00□44	□□□□□0 □□□□□□1
04		0□00□52	0□00□53	□□□□□0 □□□□□□1
05		0□00□57	0□00□58	□□□0 □□□1

Figure A.1 Continued

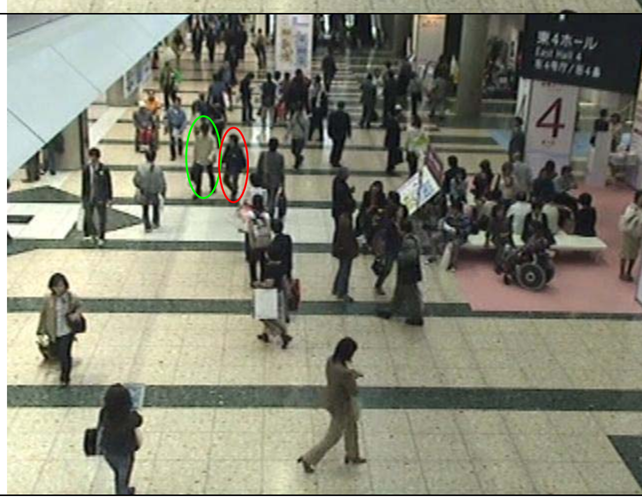
06		0□01□03	0□01□07	□□□□0 □□□1
07		0□01□14	1□01□16	□□□□0 □□□1
08		0□01□20	0□01□21	□□□0 □□□1

Figure A.1 Continued

09		0□01□28	0□01□31	□□□□0 □□□1
10		0□01□42	0□01□50	□□□0 □□□1
11		0□01□50	0□01□50	□□□0 □□□□□1

Figure A.1 Continued

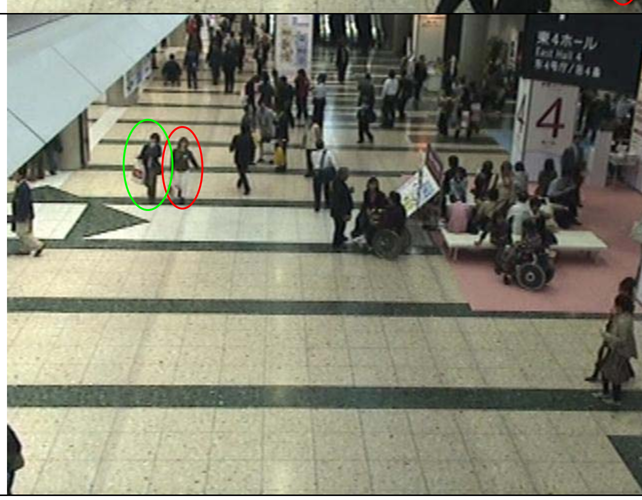
12		0□01□57	0□01□58	□□□0 □□□1
13		0□02□04	0□02□04	□□□□0 □□□□□1
14		0□02□14	0□02□15	□□□0 □□□1

Figure A.1 Continued

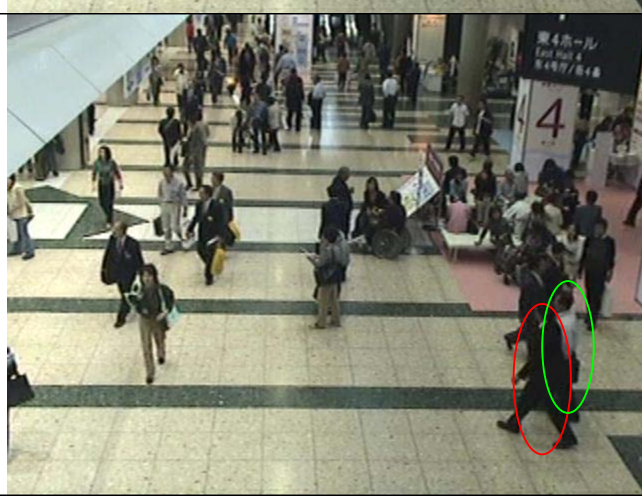
15		0□02□14	0□02□15	□□□□0 □□□1
16		0□02□37	0□02□38	□□□□□0 □□□□1
17		0□02□37	0□02□40	□□□□0 □□□1

Figure A.1 Continued

18		0□02□47	0□02□45	□□□0 □□□1
19		0□02□52	0□02□50	□□□0 □□□1
20		0□03□10	0□03□10	□□□□□0 □□□□□1
□	□	0□03□15	□	

Figure A.1 Continued

21		0□03□32	0□03□35	□□□0 □□□1
22		0□03□40	0□03□39	□□□□□□□0 □□□□□□1
23		0□04□08	0□04□09	□□□0 □□□1

Figure A.1 Continued

24		0□04□16	0□04□18	□□□□□0 □□□□□1
25		0□04□25	0□04□30	□□□0 □□□1
26		0□04□25	0□04□30	□□□0 □□□1 □□□2

Figure A.1 Continued

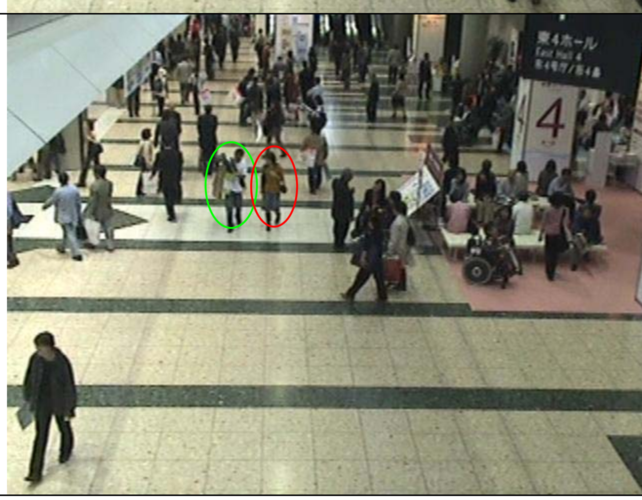
27		0□04□54	0□04□53	□□□□□0 □□□1
28		0□04□57	0□04□56	□□□0 □□□1
29		0□05□02	0□05□02	□□□0 □□□1

Figure A.1 Continued

30		0□05□06	0□05□07	□□□0 □□□1
31		0□05□10	0□05□10	□□□0 □□□1 □□□□□□2 □□□□□□3
32		0□05□24	0□05□25	□□0 □□1

Figure A.1 Continued

33		0□06□47	0□06□52	□□□0 □□□□1 □□□2 □□□□3 □□□4
34		0□07□42	0□07□42	□□□□0 □□□1 □□□□2
35		0□07□48	0□07□47	□□□0 □□□1

Figure A.1 Continued

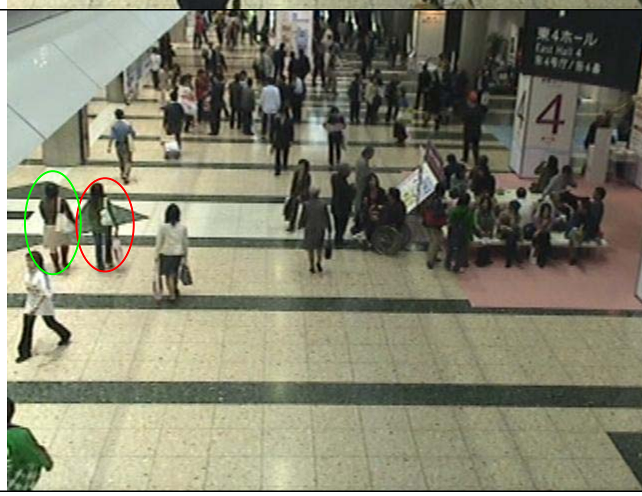
36		0□08□01	0□08□01	□□□0 □□□1
37		0□08□14	0□08□18	□□□0 □□□1

Figure A.1 Continued

Table A1 Digitized data corresponding to scenes in Figure A.1

SQ	T	H	M	S	F	SCENE	ID	TYPE	IX	IY	RX	RY
1	0.03	24	0	0	0	1	0	1	271	183	87.3	1117
16	0.53	24	0	0	15	1	0	1	267	191	78.1	1051
31	1.03	24	0	1	0	1	0	1	261	199	65.2	989
46	1.53	24	0	1	15	1	0	1	256	203	54.9	959
61	2.04	24	0	2	0	1	0	1	247	213	37.4	888
76	2.54	24	0	2	15	1	0	1	243	219	30.1	847
91	3.04	24	0	3	0	1	0	1	246	229	37.1	785
1	0.03	24	0	0	0	1	1	1	301	185	154	1106
16	0.53	24	0	0	15	1	1	1	296	192	141	1048
31	1.03	24	0	1	0	1	1	1	291	202	128	971
46	1.53	24	0	1	15	1	1	1	287	199	120	993
61	2.04	24	0	2	0	1	1	1	279	213	102	892
76	2.54	24	0	2	15	1	1	1	275	222	92.9	832
91	3.04	24	0	3	0	1	1	1	277	232	95.9	771
540	18	24	0	18	0	2	0	1	157	429	-53.4	39.9
555	18.5	24	0	18	15	2	0	1	169	413	-43	77.4
570	19	24	0	19	0	2	0	1	171	389	-46.2	138
585	19.5	24	0	19	15	2	0	1	172	377	-47.8	171
600	20	24	0	20	0	2	0	1	173	354	-52.6	239
615	20.5	24	0	20	15	2	0	1	165	342	-67	277
630	21	24	0	21	0	2	0	1	164	331	-71.9	313
540	18	24	0	18	0	2	1	1	85	436	-133	23.8
555	18.5	24	0	18	15	2	1	1	89	409	-139	86.2
570	19	24	0	19	0	2	1	1	101	389	-133	137
585	19.5	24	0	19	15	2	1	1	107	374	-132	178
600	20	24	0	20	0	2	1	1	112	367	-128	198
615	20.5	24	0	20	15	2	1	1	121	348	-125	256
630	21	24	0	21	0	2	1	1	123	329	-130	319

Table A1 Continued

SQ	T	H	M	S	F	SCENE	ID	TYPE	IX	IY	RX	RY
1320	44	24	0	44	0	3	0	1	286	223	114	827
1335	44.5	24	0	44	15	3	0	1	285	227	112	802
1350	45	24	0	45	0	3	0	1	277	239	95.4	730
1365	45.5	24	0	45	15	3	0	1	271	247	83.9	684
1380	46	24	0	46	0	3	0	1	269	257	79.9	631
1395	46.5	24	0	46	15	3	0	1	265	267	72.8	581
1410	47	24	0	47	0	3	0	1	264	273	71.1	552
1320	44	24	0	44	0	3	1	1	291	209	127	921
1335	44.5	24	0	44	15	3	1	1	289	215	122	880
1350	45	24	0	45	0	3	1	1	286	226	114	808
1365	45.5	24	0	45	15	3	1	1	283	232	107	771
1380	46	24	0	46	0	3	1	1	277	237	95.5	741
1395	46.5	24	0	46	15	3	1	1	285	245	109	697
1410	47	24	0	47	0	3	1	1	283	259	104	622
1590	53.1	24	0	53	0	4	0	1	105	362	-139	213
1605	53.6	24	0	53	15	4	0	1	119	339	-131	285
1620	54.1	24	0	54	0	4	0	1	127	316	-131	365
1635	54.6	24	0	54	15	4	0	1	137	293	-128	455
1650	55.1	24	0	55	0	4	0	1	150	277	-115	525
1665	55.6	24	0	55	15	4	0	1	159	261	-108	601
1680	56.1	24	0	56	0	4	0	1	165	247	-105	674
1590	53.1	24	0	53	0	4	1	1	29	430	-198	36.7
1605	53.6	24	0	53	15	4	1	1	51	393	-192	126
1620	54.1	24	0	54	0	4	1	1	69	365	-185	203
1635	54.6	24	0	54	15	4	1	1	85	343	-176	271
1650	55.1	24	0	55	0	4	1	1	98	317	-172	360
1665	55.6	24	0	55	15	4	1	1	107	294	-173	449
1680	56.1	24	0	56	0	4	1	1	116	279	-168	514

Table A1 Continued

SQ	T	H	M	S	F	SCENE	ID	TYPE	IX	IY	RX	RY
1740	58.1	24	0	58	0	5	0	1	250	278	48	528
1755	58.6	24	0	58	15	5	0	1	244	300	39.9	433
1770	59.1	24	0	59	0	5	0	1	239	313	33.3	382
1785	59.6	24	0	59	15	5	0	1	231	326	22.8	334
1800	60.1	24	1	0	2	5	0	1	216	341	3.5	282
1815	60.6	24	1	0	17	5	0	1	205	357	-8.92	230
1830	61.1	24	1	1	2	5	0	1	176	377	-42.7	171
1740	58.1	24	0	58	0	5	1	1	209	259	-24	616
1755	58.6	24	0	58	15	5	1	1	209	272	-20.7	553
1770	59.1	24	0	59	0	5	1	1	195	283	-40.5	502
1785	59.6	24	0	59	15	5	1	1	187	303	-47.1	418
1800	60.1	24	1	0	2	5	1	1	171	319	-65.9	356
1815	60.6	24	1	0	17	5	1	1	160	335	-76.2	300
1830	61.1	24	1	1	2	5	1	1	139	351	-99.2	247
2010	67.1	24	1	7	2	6	0	1	529	355	443	246
2025	67.6	24	1	7	17	6	0	1	498	343	407	285
2040	68.1	24	1	8	2	6	0	1	459	333	357	318
2055	68.6	24	1	8	17	6	0	1	428	315	322	383
2070	69.1	24	1	9	2	6	0	1	397	313	275	389
2085	69.6	24	1	9	17	6	0	1	369	306	234	416
2100	70.1	24	1	10	2	6	0	1	343	293	198	468
2010	67.1	24	1	7	2	6	1	1	547	336	487	311
2025	67.6	24	1	7	17	6	1	1	513	326	445	346
2040	68.1	24	1	8	2	6	1	1	480	319	400	370
2055	68.6	24	1	8	17	6	1	1	450	306	363	420
2070	69.1	24	1	9	2	6	1	1	424	301	324	439
2085	69.6	24	1	9	17	6	1	1	401	289	294	489
2100	70.1	24	1	10	2	6	1	1	373	277	254	541

Table A1 Continued

SQ	T	H	M	S	F	SCENE	ID	TYPE	IX	IY	RX	RY
2280	76.1	24	1	16	2	7	0	1	601	414	489	79.9
2295	76.6	24	1	16	17	7	0	1	557	404	441	105
2310	77.1	24	1	17	2	7	0	1	530	394	413	131
2325	77.6	24	1	17	17	7	0	1	492	385	369	155
2340	78.1	24	1	18	2	7	0	1	455	379	323	171
2355	78.6	24	1	18	17	7	0	1	421	370	281	197
2370	79.1	24	1	19	2	7	0	1	385	358	237	232
2280	76.1	24	1	16	2	7	1	1	576	362	504	225
2295	76.6	24	1	16	17	7	1	1	537	358	451	237
2310	77.1	24	1	17	2	7	1	1	505	349	413	265
2325	77.6	24	1	17	17	7	1	1	470	341	368	291
2340	78.1	24	1	18	2	7	1	1	439	340	323	293
2355	78.6	24	1	18	17	7	1	1	411	330	287	327
2370	79.1	24	1	19	2	7	1	1	377	328	237	333
2430	81.1	24	1	21	2	8	0	1	222	183	-21.5	1107
2445	81.6	24	1	21	17	8	0	1	213	189	-38.9	1057
2460	82.1	24	1	22	2	8	0	1	203	199	-55.8	979
2475	82.6	24	1	22	17	8	0	1	191	200	-80.1	970
2490	83.1	24	1	23	2	8	0	1	179	212	-97.8	885
2505	83.6	24	1	23	17	8	0	1	172	219	-107	838
2520	84.1	24	1	24	2	8	0	1	163	225	-121	800
2430	81.1	24	1	21	2	8	1	1	197	179	-78.4	1136
2445	81.6	24	1	21	17	8	1	1	187	183	-98.1	1100
2460	82.1	24	1	22	2	8	1	1	175	192	-118	1027
2475	82.6	24	1	22	17	8	1	1	162	199	-140	973
2490	83.1	24	1	23	2	8	1	1	149	206	-161	921
2505	83.6	24	1	23	17	8	1	1	133	217	-183	846
2520	84.1	24	1	24	2	8	1	1	127	225	-188	795

Table A1 Continued

SQ	T	H	M	S	F	SCENE	ID	TYPE	IX	IY	RX	RY
2730	91.1	24	1	31	2	9	0	1	552	327	504	344
2745	91.6	24	1	31	17	9	0	1	517	319	458	372
2760	92.1	24	1	32	2	9	0	1	486	307	420	418
2775	92.6	24	1	32	17	9	0	1	453	297	375	458
2790	93.1	24	1	33	2	9	0	1	425	289	334	490
2805	93.6	24	1	33	17	9	0	1	395	281	289	524
2820	94.1	24	1	34	2	9	0	1	365	275	241	550
2730	91.1	24	1	31	2	9	1	1	583	312	571	403
2745	91.6	24	1	31	17	9	1	1	553	303	533	438
2760	92.1	24	1	32	2	9	1	1	520	297	486	462
2775	92.6	24	1	32	17	9	1	1	491	290	444	490
2790	93.1	24	1	33	2	9	1	1	461	282	401	525
2805	93.6	24	1	33	17	9	1	1	435	271	366	575
2820	94.1	24	1	34	2	9	1	1	409	264	325	608
3330	111	24	1	51	2	10	0	1	292	218	127	861
3345	112	24	1	51	17	10	0	1	285	225	112	815
3360	112	24	1	52	2	10	0	1	280	233	102	765
3375	113	24	1	52	17	10	0	1	281	235	103	753
3390	113	24	1	53	2	10	0	1	278	245	96.7	696
3405	114	24	1	53	17	10	0	1	273	243	87.7	707
3420	114	24	1	54	2	10	0	1	265	241	73.1	717
3330	111	24	1	51	2	10	1	1	261	214	65.4	883
3345	112	24	1	51	17	10	1	1	255	225	54	811
3360	112	24	1	52	2	10	1	1	250	229	44.7	786
3375	113	24	1	52	17	10	1	1	245	235	35.8	749
3390	113	24	1	53	2	10	1	1	244	245	35	692
3405	114	24	1	53	17	10	1	1	246	253	39.3	650
3420	114	24	1	54	2	10	1	1	239	257	27.5	628

Table A1 Continued

SQ	T	H	M	S	F	SCENE	ID	TYPE	IX	IY	RX	RY
3300	110	24	1	50	2	11	0	1	178	445	-26.3	4.82
3315	111	24	1	50	17	11	0	1	182	419	-26.5	63.2
3330	111	24	1	51	2	11	0	1	181	394	-32.6	125
3345	112	24	1	51	17	11	0	1	180	373	-38.5	182
3360	112	24	1	52	2	11	0	1	182	353	-40.7	242
3375	113	24	1	52	17	11	0	1	181	339	-45.7	287
3390	113	24	1	53	2	11	0	1	185	322	-44.6	346
3300	110	24	1	50	2	11	1	1	111	436	-104	24
3315	111	24	1	50	17	11	1	1	117	414	-104	74.4
3330	111	24	1	51	2	11	1	1	121	390	-108	135
3345	112	24	1	51	17	11	1	1	124	368	-112	196
3360	112	24	1	52	2	11	1	1	126	351	-117	247
3375	113	24	1	52	17	11	1	1	131	333	-117	305
3390	113	24	1	53	2	11	1	1	131	317	-125	362
3540	118	24	1	58	2	12	0	1	337	372	167	189
3555	119	24	1	58	17	12	0	1	316	349	143	259
3570	119	24	1	59	2	12	0	1	310	324	139	344
3585	120	24	1	59	17	12	0	1	294	305	117	416
3600	120	24	2	0	4	12	0	1	286	287	107	490
3615	121	24	2	0	19	12	0	1	269	272	79.5	557
3630	121	24	2	1	4	12	0	1	261	255	65.9	641
3540	118	24	1	58	2	12	1	1	289	421	100	59.5
3555	119	24	1	58	17	12	1	1	275	389	84.6	140
3570	119	24	1	59	2	12	1	1	255	362	58.7	217
3585	120	24	1	59	17	12	1	1	246	339	45.4	289
3600	120	24	2	0	4	12	1	1	241	324	37.2	341
3615	121	24	2	0	19	12	1	1	238	303	30.9	420
3630	121	24	2	1	4	12	1	1	235	284	24.1	500

Table A1 Continued

SQ	T	H	M	S	F	SCENE	ID	TYPE	IX	IY	RX	RY
3680	123	24	2	2	24	13	0	1	578	460	426	-25.6
3695	123	24	2	3	9	13	0	1	518	456	359	-17.4
3710	124	24	2	3	24	13	0	1	481	460	314	-25.8
3720	124	24	2	4	4	13	0	1	453	455	284	-15.4
3735	125	24	2	4	19	13	0	1	407	460	230	-25.9
3750	125	24	2	5	4	13	0	1	359	469	174	-44.2
3765	126	24	2	5	19	13	0	1	320	475	131	-56
3680	123	24	2	2	24	13	1	1	525	426	385	49.9
3695	123	24	2	3	9	13	1	1	487	428	337	44.8
3710	124	24	2	3	24	13	1	1	439	431	278	37.4
3720	124	24	2	4	4	13	1	1	410	429	243	41.9
3735	125	24	2	4	19	13	1	1	368	434	192	30.1
3750	125	24	2	5	4	13	1	1	324	440	140	16.4
3765	126	24	2	5	19	13	1	1	283	451	91.9	-7.5
4070	136	24	2	15	24	14	0	1	169	197	-127	988
4085	136	24	2	16	9	14	0	1	169	205	-122	931
4100	137	24	2	16	24	14	0	1	165	215	-123	863
4115	137	24	2	17	9	14	0	1	163	219	-125	837
4130	138	24	2	17	24	14	0	1	158	226	-130	793
4145	138	24	2	18	9	14	0	1	150	238	-137	722
4160	139	24	2	18	24	14	0	1	146	247	-138	672
4070	136	24	2	15	24	14	1	1	135	196	-198	990
4085	136	24	2	16	9	14	1	1	131	201	-201	954
4100	137	24	2	16	24	14	1	1	126	211	-202	884
4115	137	24	2	17	9	14	1	1	120	217	-208	844
4130	138	24	2	17	24	14	1	1	119	226	-202	788
4145	138	24	2	18	9	14	1	1	115	237	-200	724
4160	139	24	2	18	24	14	1	1	115	247	-192	669

Table A1 Continued

SQ	T	H	M	S	F	SCENE	ID	TYPE	IX	IY	RX	RY
4080	136	24	2	16	4	15	0	1	528	369	429	203
4095	137	24	2	16	19	15	0	1	493	366	383	211
4110	137	24	2	17	4	15	0	1	458	357	340	238
4125	138	24	2	17	19	15	0	1	427	346	302	272
4140	138	24	2	18	4	15	0	1	402	337	270	302
4155	139	24	2	18	19	15	0	1	386	324	252	347
4170	139	24	2	19	4	15	0	1	358	317	213	372
4080	136	24	2	16	4	15	1	1	533	353	450	253
4095	137	24	2	16	19	15	1	1	499	345	407	278
4110	137	24	2	17	4	15	1	1	474	334	379	315
4125	138	24	2	17	19	15	1	1	458	331	357	325
4140	138	24	2	18	4	15	1	1	431	323	321	353
4155	139	24	2	18	19	15	1	1	406	315	287	382
4170	139	24	2	19	4	15	1	1	384	307	257	412
4750	158	24	2	38	14	16	0	1	244	221	32.3	835
4765	159	24	2	38	29	16	0	1	235	229	16.2	784
4780	159	24	2	39	14	16	0	1	227	237	2.77	736
4795	160	24	2	39	29	16	0	1	217	242	-14.3	706
4810	160	24	2	40	14	16	0	1	212	253	-20.4	647
4825	161	24	2	40	29	16	0	1	198	259	-42.9	615
4840	161	24	2	41	14	16	0	1	195	273	-43.5	547
4750	158	24	2	38	14	16	1	1	231	207	4.14	926
4765	159	24	2	38	29	16	1	1	231	212	5.23	892
4780	159	24	2	39	14	16	1	1	227	221	-0.66	833
4795	160	24	2	39	29	16	1	1	219	226	-14.8	800
4810	160	24	2	40	14	16	1	1	215	231	-21	769
4825	161	24	2	40	29	16	1	1	207	243	-32.1	700
4840	161	24	2	41	14	16	1	1	203	247	-38	678

Table A1 Continued

SQ	T	H	M	S	F	SCENE	ID	TYPE	IX	IY	RX	RY
4820	161	24	2	40	24	17	0	1	533	423	397	57.1
4835	161	24	2	41	9	17	0	1	491	415	349	76.2
4850	162	24	2	41	24	17	0	1	441	402	293	108
4865	162	24	2	42	9	17	0	1	403	382	252	162
4880	163	24	2	42	24	17	0	1	362	369	202	198
4895	163	24	2	43	9	17	0	1	321	361	148	221
4910	164	24	2	43	24	17	0	1	281	347	94.5	264
4820	161	24	2	40	24	17	1	1	557	399	445	118
4835	161	24	2	41	9	17	1	1	521	391	403	139
4850	162	24	2	41	24	17	1	1	477	382	350	163
4865	162	24	2	42	9	17	1	1	433	371	297	194
4880	163	24	2	42	24	17	1	1	391	360	244	226
4895	163	24	2	43	9	17	1	1	351	344	194	276
4910	164	24	2	43	24	17	1	1	321	335	153	306
4950	165	24	2	45	4	18	0	1	456	351	341	257
4965	166	24	2	45	19	18	0	1	435	362	305	221
4980	166	24	2	46	4	18	0	1	418	367	279	206
5000	167	24	2	46	24	18	0	1	402	385	249	153
5015	167	24	2	47	9	18	0	1	382	397	219	121
5030	168	24	2	47	24	18	0	1	372	406	204	97.2
5045	168	24	2	48	9	18	0	1	353	425	176	50.7
4950	165	24	2	45	4	18	1	1	493	369	381	202
4965	166	24	2	45	19	18	1	1	488	374	370	187
4980	166	24	2	46	4	18	1	1	468	381	339	166
5000	167	24	2	46	24	18	1	1	448	393	306	132
5015	167	24	2	47	9	18	1	1	429	407	275	95.4
5030	168	24	2	47	24	18	1	1	409	420	245	63.1
5045	168	24	2	48	9	18	1	1	392	440	219	16.8

Table A1 Continued

SQ	T	H	M	S	F	SCENE	ID	TYPE	IX	IY	RX	RY
5100	170	24	2	50	4	19	0	1	253	434	57.8	29.3
5115	171	24	2	50	19	19	0	1	260	403	65.7	103
5130	171	24	2	51	4	19	0	1	259	376	64.2	176
5145	172	24	2	51	19	19	0	1	263	356	69.4	235
5160	172	24	2	52	4	19	0	1	266	341	73.6	283
5175	173	24	2	52	19	19	0	1	281	325	95.8	339
5190	173	24	2	53	4	19	0	1	291	307	112	407
5100	170	24	2	50	4	19	1	1	183	430	-23.3	37.8
5115	171	24	2	50	19	19	1	1	192	399	-18.1	112
5130	171	24	2	51	4	19	1	1	190	373	-25.6	183
5145	172	24	2	51	19	19	1	1	197	356	-19.9	233
5160	172	24	2	52	4	19	1	1	223	339	13	289
5175	173	24	2	52	19	19	1	1	239	321	34	352
5190	173	24	2	53	4	19	1	1	246	309	43.6	397
5700	190	24	3	10	6	20	0	1	271	361	80.2	220
5715	191	24	3	10	21	20	0	1	270	353	79	245
5730	191	24	3	11	6	20	0	1	269	340	77.9	287
5745	192	24	3	11	21	20	0	1	270	329	79.6	325
5760	192	24	3	12	6	20	0	1	274	320	85.7	357
5775	193	24	3	12	21	20	0	1	272	311	83.1	391
5790	193	24	3	13	6	20	0	1	271	303	81.8	422
5700	190	24	3	10	6	20	1	1	208	356	-5.06	234
5715	191	24	3	10	21	20	1	1	207	342	-8.86	278
5730	191	24	3	11	6	20	1	1	208	337	-8.37	295
5745	192	24	3	11	21	20	1	1	209	323	-9.55	343
5760	192	24	3	12	6	20	1	1	209	315	-11.1	373
5775	193	24	3	12	21	20	1	1	211	307	-9.7	403
5790	193	24	3	13	6	20	1	1	216	297	-3.99	444

Table A1 Continued

SQ	T	H	M	S	F	SCENE	ID	TYPE	IX	IY	RX	RY
6445	215	24	3	35	1	21	0	1	171	185	-131	1081
6460	216	24	3	35	16	21	0	1	167	198	-131	981
6475	216	24	3	36	1	21	0	1	165	205	-130	930
6490	217	24	3	36	16	21	0	1	165	217	-122	850
6505	217	24	3	37	1	21	0	1	165	231	-114	764
6520	218	24	3	37	16	21	0	1	166	243	-105	696
6535	218	24	3	38	1	21	0	1	165	258	-99.7	617
6445	215	24	3	35	1	21	1	1	130	197	-207	982
6460	216	24	3	35	16	21	1	1	131	209	-194	898
6475	216	24	3	36	1	21	1	1	126	222	-193	813
6490	217	24	3	36	16	21	1	1	126	233	-184	748
6505	217	24	3	37	1	21	1	1	131	247	-165	671
6520	218	24	3	37	16	21	1	1	134	262	-150	594
6535	218	24	3	38	1	21	1	1	140	277	-131	525
6580	220	24	3	39	16	22	0	1	31	419	-202	61.7
6595	220	24	3	40	1	22	0	1	40	399	-202	110
6610	221	24	3	40	16	22	0	1	46	379	-206	163
6625	221	24	3	41	1	22	0	1	61	363	-196	209
6640	222	24	3	41	16	22	0	1	64	345	-203	264
6655	222	24	3	42	1	22	0	1	67	336	-204	293
6670	223	24	3	42	16	22	0	1	72	312	-213	377
6580	220	24	3	39	16	22	1	1	49	390	-196	133
6595	220	24	3	40	1	22	1	1	58	372	-195	183
6610	221	24	3	40	16	22	1	1	61	357	-199	226
6625	221	24	3	41	1	22	1	1	65	343	-203	270
6640	222	24	3	41	16	22	1	1	71	331	-202	310
6655	222	24	3	42	1	22	1	1	71	316	-212	363
6670	223	24	3	42	16	22	1	1	79	305	-208	404

Table A1 Continued

SQ	T	H	M	S	F	SCENE	ID	TYPE	IX	IY	RX	RY
7465	249	24	4	9	3	23	0	1	272	229	86.6	788
7480	250	24	4	9	18	23	0	1	268	235	78.8	752
7495	250	24	4	10	3	23	0	1	263	241	69.4	717
7510	251	24	4	10	18	23	0	1	244	254	35.9	644
7525	251	24	4	11	3	23	0	1	227	259	6.98	617
7540	252	24	4	11	18	23	0	1	219	265	-5.53	587
7555	252	24	4	12	3	23	0	1	212	272	-15.7	553
7465	249	24	4	9	3	23	1	1	225	245	0.77	691
7480	250	24	4	9	18	23	1	1	215	251	-15.7	657
7495	250	24	4	10	3	23	1	1	209	253	-25.7	646
7510	251	24	4	10	18	23	1	1	191	262	-53.8	599
7525	251	24	4	11	3	23	1	1	179	269	-71.3	564
7540	252	24	4	11	18	23	1	1	176	275	-73.9	536
7555	252	24	4	12	3	23	1	1	167	285	-84.4	491
7745	258	24	4	18	13	24	0	1	509	319	445	372
7760	259	24	4	18	28	24	0	1	477	309	403	409
7775	259	24	4	19	13	24	0	1	446	299	362	449
7790	260	24	4	19	28	24	0	1	414	295	312	464
7805	260	24	4	20	13	24	0	1	386	289	270	488
7820	261	24	4	20	28	24	0	1	362	284	232	508
7835	261	24	4	21	13	24	0	1	334	279	187	529
7745	258	24	4	18	13	24	1	1	547	333	490	322
7760	259	24	4	18	28	24	1	1	511	323	444	357
7775	259	24	4	19	13	24	1	1	478	308	406	413
7790	260	24	4	19	28	24	1	1	449	303	363	432
7805	260	24	4	20	13	24	1	1	415	301	310	439
7820	261	24	4	20	28	24	1	1	386	292	268	475
7835	261	24	4	21	13	24	1	1	357	286	223	499

Table A1 Continued

SQ	T	H	M	S	F	SCENE	ID	TYPE	IX	IY	RX	RY
8105	270	24	4	30	13	25	0	1	385	340	244	291
8120	271	24	4	30	28	25	0	1	351	332	197	317
8135	271	24	4	31	13	25	0	1	323	323	158	348
8150	272	24	4	31	28	25	0	1	297	315	121	377
8165	272	24	4	32	13	25	0	1	266	308	74	402
8180	273	24	4	32	28	25	0	1	243	304	38.6	417
8195	273	24	4	33	13	25	0	1	209	301	-14	427
8105	270	24	4	30	13	25	1	1	426	334	307	313
8120	271	24	4	30	28	25	1	1	390	325	258	344
8135	271	24	4	31	13	25	1	1	367	319	226	365
8150	272	24	4	31	28	25	1	1	347	308	199	406
8165	272	24	4	32	13	25	1	1	320	305	158	417
8180	273	24	4	32	28	25	1	1	292	295	115	456
8195	273	24	4	33	13	25	1	1	260	288	64.6	484
8105	270	24	4	30	13	26	0	1	257	191	56.4	1049
8120	271	24	4	30	28	26	0	1	257	198	56.8	996
8135	271	24	4	31	13	26	0	1	255	202	52.8	966
8150	272	24	4	31	28	26	0	1	253	210	49.2	909
8165	272	24	4	32	13	26	0	1	256	220	55.7	843
8180	273	24	4	32	28	26	0	1	257	227	57.9	799
8195	273	24	4	33	13	26	0	1	264	235	71.3	751
8105	270	24	4	30	13	26	1	1	239	175	14.5	1179
8120	271	24	4	30	28	26	1	1	241	186	21	1086
8135	271	24	4	31	13	26	1	1	233	193	5.17	1029
8150	272	24	4	31	28	26	1	1	231	199	2.33	984
8165	272	24	4	32	13	26	1	1	229	209	0.55	912
8180	273	24	4	32	28	26	1	1	228	220	1.05	839
8195	273	24	4	33	13	26	1	1	228	223	1.71	820
8105	270	24	4	30	13	26	2	1	197	187	-74.2	1070
8120	271	24	4	30	28	26	2	1	197	193	-71.1	1023
8135	271	24	4	31	13	26	2	1	199	200	-63.6	971
8150	272	24	4	31	28	26	2	1	202	206	-54.9	929
8165	272	24	4	32	13	26	2	1	195	215	-64.9	867
8180	273	24	4	32	28	26	2	1	193	225	-64.5	803
8195	273	24	4	33	13	26	2	1	188	231	-71.3	766

Table A1 Continued

SQ	T	H	M	S	F	SCENE	ID	TYPE	IX	IY	RX	RY
8785	293	24	4	53	3	27	0	1	231	345	24.9	269
8800	294	24	4	53	18	27	0	1	232	330	24.7	320
8815	294	24	4	54	3	27	0	1	229	313	18.4	381
8830	295	24	4	54	18	27	0	1	225	299	10.3	436
8845	295	24	4	55	3	27	0	1	219	287	-1.15	486
8860	296	24	4	55	18	27	0	1	215	273	-10.5	548
8875	296	24	4	56	3	27	0	1	213	265	-15.7	586
8785	293	24	4	53	3	27	1	1	163	363	-63.3	211
8800	294	24	4	53	18	27	1	1	169	347	-60	261
8815	294	24	4	54	3	27	1	1	169	326	-66.5	331
8830	295	24	4	54	18	27	1	1	168	311	-73	386
8845	295	24	4	55	3	27	1	1	163	294	-87	453
8860	296	24	4	55	18	27	1	1	159	281	-98.8	508
8875	296	24	4	56	3	27	1	1	153	267	-115	572
8875	296	24	4	56	3	28	0	1	365	326	220	339
8890	297	24	4	56	18	28	0	1	327	319	165	363
8910	297	24	4	57	8	28	0	1	301	309	128	400
8925	298	24	4	57	23	28	0	1	279	301	94.3	431
8940	298	24	4	58	8	28	0	1	257	286	59.7	493
8955	299	24	4	58	23	28	0	1	235	283	23.9	504
8970	299	24	4	59	8	28	0	1	217	276	-6.56	535
8875	296	24	4	56	3	28	1	1	328	347	160	266
8890	297	24	4	56	18	28	1	1	294	336	114	301
8910	297	24	4	57	8	28	1	1	245	325	43.1	338
8925	298	24	4	57	23	28	1	1	227	318	16	362
8940	298	24	4	58	8	28	1	1	219	309	2.74	396
8955	299	24	4	58	23	28	1	1	205	294	-21.8	455
8970	299	24	4	59	8	28	1	1	180	285	-63.8	492

Table A1 Continued

SQ	T	H	M	S	F	SCENE	ID	TYPE	IX	IY	RX	RY
9050	302	24	5	2	0	29	0	1	265	208	73.5	924
9065	302	24	5	2	15	29	0	1	260	218	63.5	856
9080	303	24	5	3	0	29	0	1	257	225	57.8	811
9095	303	24	5	3	15	29	0	1	256	232	56.2	768
9110	304	24	5	4	0	29	0	1	250	236	45.2	744
9125	304	24	5	4	15	29	0	1	250	245	45.9	693
9140	305	24	5	5	0	29	0	1	251	251	48.1	661
9050	302	24	5	2	0	29	1	1	227	211	-2.99	898
9065	302	24	5	2	15	29	1	1	225	220	-4.77	839
9080	303	24	5	3	0	29	1	1	222	225	-9.32	807
9095	303	24	5	3	15	29	1	1	227	231	1.52	771
9110	304	24	5	4	0	29	1	1	227	240	3.38	719
9125	304	24	5	4	15	29	1	1	225	245	0.77	691
9140	305	24	5	5	0	29	1	1	217	254	-11.4	642
9215	307	24	5	7	15	30	0	1	351	467	166	-40.2
9230	308	24	5	8	0	30	0	1	407	464	229	-34.1
9245	308	24	5	8	15	30	0	1	455	457	286	-19.6
9260	309	24	5	9	0	30	0	1	507	451	349	-6.73
9275	309	24	5	9	15	30	0	1	547	444	400	8.72
9290	310	24	5	10	0	30	0	1	585	443	446	11.1
9305	310	24	5	10	15	30	0	1	634	447	502	2.49
9215	307	24	5	7	15	30	1	1	246	462	50.7	-30.3
9230	308	24	5	8	0	30	1	1	299	451	110	-7.45
9245	308	24	5	8	15	30	1	1	352	445	171	5.6
9260	309	24	5	9	0	30	1	1	405	439	234	19.1
9275	309	24	5	9	15	30	1	1	455	439	293	19.4
9290	310	24	5	10	0	30	1	1	503	432	354	35.6
9305	310	24	5	10	15	30	1	1	549	432	410	36

Table A1 Continued

SQ	T	H	M	S	F	SCENE	ID	TYPE	IX	IY	RX	RY
9310	311	24	5	10	20	31	0	1	176	433	-30.8	31
9325	311	24	5	11	5	31	0	1	177	412	-33.8	79.9
9340	312	24	5	11	20	31	0	1	185	390	-28.5	136
9355	312	24	5	12	5	31	0	1	193	372	-22	186
9370	313	24	5	12	20	31	0	1	198	353	-19.1	243
9385	313	24	5	13	5	31	0	1	201	337	-18.2	294
9400	314	24	5	13	20	31	0	1	206	324	-13.7	340
9310	311	24	5	10	20	31	1	1	123	461	-83	-28.5
9325	311	24	5	11	5	31	1	1	132	432	-81.1	33
9340	312	24	5	11	20	31	1	1	139	412	-78.8	79.5
9355	312	24	5	12	5	31	1	1	146	388	-77.5	140
9370	313	24	5	12	20	31	1	1	153	371	-73.9	188
9385	313	24	5	13	5	31	1	1	163	349	-67.5	254
9400	314	24	5	13	20	31	1	1	173	336	-57.7	297
9310	311	24	5	10	20	31	2	1	51	449	-165	-4.13
9325	311	24	5	11	5	31	2	1	63	425	-162	48.2
9340	312	24	5	11	20	31	2	1	65	406	-169	93.2
9355	312	24	5	12	5	31	2	1	72	394	-166	123
9370	313	24	5	12	20	31	2	1	82	367	-167	197
9385	313	24	5	13	5	31	2	1	92	351	-162	246
9400	314	24	5	13	20	31	2	1	110	340	-143	281
9310	311	24	5	10	20	31	3	1	65	423	-161	52.8
9325	311	24	5	11	5	31	3	1	76	405	-156	95.8
9340	312	24	5	11	20	31	3	1	80	387	-159	142
9355	312	24	5	12	5	31	3	1	85	369	-162	192
9370	313	24	5	12	20	31	3	1	98	348	-155	255
9385	313	24	5	13	5	31	3	1	110	333	-147	305
9400	314	24	5	13	20	31	3	1	131	320	-123	351
9750	325	24	5	25	10	32	0	1	305	199	159	996
9765	326	24	5	25	25	32	0	1	301	209	147	923
9780	326	24	5	26	10	32	0	1	289	215	122	880
9795	327	24	5	26	25	32	0	1	284	227	110	802
9810	327	24	5	27	10	32	0	1	278	232	97.8	771
9825	328	24	5	27	25	32	0	1	277	245	94.9	696
9840	328	24	5	28	10	32	0	1	275	254	90.7	647
9750	325	24	5	25	10	32	1	1	275	199	94.8	991
9765	326	24	5	25	25	32	1	1	266	205	75.6	946
9780	326	24	5	26	10	32	1	1	259	214	61.4	883
9795	327	24	5	26	25	32	1	1	254	222	51.9	830
9810	327	24	5	27	10	32	1	1	247	235	39.5	750
9825	328	24	5	27	25	32	1	1	241	242	29.2	709
9840	328	24	5	28	10	32	1	1	235	255	20.2	638

Table A1 Continued

SQ	T	H	M	S	F	SCENE	ID	TYPE	IX	IY	RX	RY
12375	413	24	6	52	27	33	0	1	354	255	232	649
12390	413	24	6	53	12	33	0	1	383	258	282	636
12405	414	24	6	53	27	33	0	1	411	259	332	633
12420	414	24	6	54	12	33	0	1	424	259	356	635
12435	415	24	6	54	27	33	0	1	446	254	401	663
12450	415	24	6	55	12	33	0	1	466	255	437	660
12465	416	24	6	55	27	33	0	1	489	252	484	678
12375	413	24	6	52	27	33	1	1	307	241	151	721
12390	413	24	6	53	12	33	1	1	318	244	170	706
12405	414	24	6	53	27	33	1	1	338	250	205	675
12420	414	24	6	54	12	33	1	1	349	251	225	670
12435	415	24	6	54	27	33	1	1	367	252	257	666
12450	415	24	6	55	12	33	1	1	383	256	284	647
12465	416	24	6	55	27	33	1	1	395	251	309	675
12375	413	24	6	52	27	33	2	1	279	245	98.5	696
12390	413	24	6	53	12	33	2	1	294	252	125	660
12405	414	24	6	53	27	33	2	1	314	258	159	630
12420	414	24	6	54	12	33	2	1	340	263	203	607
12435	415	24	6	54	27	33	2	1	367	269	248	579
12450	415	24	6	55	12	33	2	1	389	268	287	586
12465	416	24	6	55	27	33	2	1	415	267	333	593
12375	413	24	6	52	27	33	3	1	245	237	36	738
12390	413	24	6	53	12	33	3	1	255	243	54.8	705
12405	414	24	6	53	27	33	3	1	264	249	71.2	673
12420	414	24	6	54	12	33	3	1	283	253	105	653
12435	415	24	6	54	27	33	3	1	303	257	140	634
12450	415	24	6	55	12	33	3	1	327	259	182	626
12465	416	24	6	55	27	33	3	1	351	258	225	633
12375	413	24	6	52	27	33	4	1	225	237	-0.91	735
12390	413	24	6	53	12	33	4	1	232	243	13	702
12405	414	24	6	53	27	33	4	1	243	248	33.5	676
12420	414	24	6	54	12	33	4	1	259	254	62.4	646
12435	415	24	6	54	27	33	4	1	284	261	106	612
12450	415	24	6	55	12	33	4	1	311	266	152	589
12465	416	24	6	55	27	33	4	1	336	267	195	586

Table A1 Continued

SQ	T	H	M	S	F	SCENE	ID	TYPE	IX	IY	RX	RY
13865	463	24	7	42	19	34	0	1	408	373	262	188
13880	463	24	7	43	4	34	0	1	381	360	231	226
13895	464	24	7	43	19	34	0	1	349	350	189	257
13910	464	24	7	44	4	34	0	1	337	337	175	299
13925	465	24	7	44	19	34	0	1	329	326	166	338
13940	465	24	7	45	4	34	0	1	313	315	145	378
13955	466	24	7	45	19	34	0	1	298	300	124	436
13865	463	24	7	42	19	34	1	1	381	395	219	126
13880	463	24	7	43	4	34	1	1	352	377	186	175
13895	464	24	7	43	19	34	1	1	320	366	146	206
13910	464	24	7	44	4	34	1	1	299	346	120	268
13925	465	24	7	44	19	34	1	1	277	331	89.6	318
13940	465	24	7	45	4	34	1	1	273	317	84.4	368
13955	466	24	7	45	19	34	1	1	262	301	67.9	430
13865	463	24	7	42	19	34	2	1	344	403	170	104
13880	463	24	7	43	4	34	2	1	307	389	125	141
13895	464	24	7	43	19	34	2	1	282	369	94.6	196
13910	464	24	7	44	4	34	2	1	251	356	53.1	235
13925	465	24	7	44	19	34	2	1	233	339	27.1	289
13940	465	24	7	45	4	34	2	1	223	323	10.8	344
13955	466	24	7	45	19	34	2	1	221	306	5.27	408
14000	467	24	7	47	4	35	0	1	185	297	-52	442
14015	468	24	7	47	19	35	0	1	178	319	-55.7	357
14030	468	24	7	48	4	35	0	1	175	340	-53.7	284
14045	469	24	7	48	19	35	0	1	165	361	-61.2	217
14060	469	24	7	49	4	35	0	1	160	388	-60.1	141
14075	470	24	7	49	19	35	0	1	146	413	-70.3	77.1
14090	470	24	7	50	4	35	0	1	124	443	-87	8.85
14000	467	24	7	47	4	35	1	1	144	307	-110	400
14015	468	24	7	47	19	35	1	1	143	326	-104	330
14030	468	24	7	48	4	35	1	1	129	341	-117	279
14045	469	24	7	48	19	35	1	1	121	363	-118	210
14060	469	24	7	49	4	35	1	1	105	394	-126	124
14075	470	24	7	49	19	35	1	1	85	419	-139	62.3
14090	470	24	7	50	4	35	1	1	63	449	-152	-4.08

Table A1 Continued

SQ	T	H	M	S	F	SCENE	ID	TYPE	IX	IY	RX	RY
14420	481	24	8	1	6	36	0	1	156	285	-102	490
14435	482	24	8	1	21	36	0	1	171	279	-80.4	518
14450	482	24	8	2	6	36	0	1	189	263	-56.9	594
14465	483	24	8	2	21	36	0	1	202	250	-38.8	661
14480	483	24	8	3	6	36	0	1	213	239	-22.4	723
14495	484	24	8	3	21	36	0	1	221	229	-10.2	782
14510	484	24	8	4	6	36	0	1	221	219	-12.8	845
14420	481	24	8	1	6	36	1	1	107	271	-188	550
14435	482	24	8	1	21	36	1	1	110	259	-192	607
14450	482	24	8	2	6	36	1	1	117	250	-187	653
14465	483	24	8	2	21	36	1	1	122	236	-189	730
14480	483	24	8	3	6	36	1	1	141	228	-160	779
14495	484	24	8	3	21	36	1	1	149	215	-154	861
14510	484	24	8	4	6	36	1	1	167	208	-124	910
14935	498	24	8	18	11	37	0	1	99	251	-217	647
14950	499	24	8	18	26	37	0	1	103	241	-218	700
14965	499	24	8	19	11	37	0	1	113	233	-207	746
14980	500	24	8	19	26	37	0	1	116	221	-212	818
14995	500	24	8	20	11	37	0	1	119	213	-214	870
15010	501	24	8	20	26	37	0	1	120	199	-225	966
15025	501	24	8	21	11	37	0	1	120	192	-232	1018
14935	498	24	8	18	11	37	1	1	55	253	-289	632
14950	499	24	8	18	26	37	1	1	57	241	-299	696
14965	499	24	8	19	11	37	1	1	63	232	-298	747
14980	500	24	8	19	26	37	1	1	66	222	-304	806
14995	500	24	8	20	11	37	1	1	72	212	-305	870
15010	501	24	8	20	26	37	1	1	76	201	-311	945
15025	501	24	8	21	11	37	1	1	82	195	-306	989

00:51:25 (Time of scene in video recording)

1:27:23

0 : id

1:49:20

3:06:12
□□□2□

3:39:06

4:09:18

Figure A.2 Scenes corresponding to time periods digitized

4:16:28 (Time of scene in video recording)

4:51:00

5:10:00

5:34:05

5:57:15

6:15:07

Figure A.2 Continued

6:30:13 (Time of scene in video recording)

8:22:09

Figure A.2 Continued

Table A2 Digitized data corresponding to scenes in Figure A.2

Time: 0.51.25												
SQ	**T**	**H**	**M**	**S**	**F**	**SCENE**	**ID**	**TYPE**	**IX**	**IY**	**RX**	**RY**
1545	51.6	24	0	51	15	1	0	1	56	466	-153	-38.4
1560	52.1	24	0	52	0	1	0	1	72	428	-151	41.5
1575	52.6	24	0	52	15	1	0	1	90	397	-143	116
1590	53.1	24	0	53	0	1	0	1	105	365	-138	204
1605	53.6	24	0	53	15	1	0	1	117	339	-134	285
1620	54.1	24	0	54	0	1	0	1	128	315	-130	369
1635	54.6	24	0	54	15	1	0	1	139	297	-123	439
1545	51.6	24	0	51	15	1	1	1	220	369	12.8	195
1560	52.1	24	0	52	0	1	1	1	209	386	0.89	147
1575	52.6	24	0	52	15	1	1	1	197	405	-11	97.4
1590	53.1	24	0	53	0	1	1	1	183	427	-23.8	44.6
1605	53.6	24	0	53	15	1	1	1	165	448	-40.2	-1.6
1620	54.1	24	0	54	0	1	1	1	146	471	-55.9	-48.1
1635	54.6	24	0	54	15	1	1	1	129	479	-71.9	-63.4
1545	51.6	24	0	51	15	1	2	1	241	276	33	536
1560	52.1	24	0	52	0	1	2	1	237	294	28.4	457
1575	52.6	24	0	52	15	1	2	1	237	305	29.5	413
1590	53.1	24	0	53	0	1	2	1	238	316	32.1	370
1605	53.6	24	0	53	15	1	2	1	231	336	23.9	299
1620	54.1	24	0	54	0	1	2	1	223	347	14.1	262
1635	54.6	24	0	54	15	1	2	1	214	368	4.76	198

Table A2 Continued

Time: 1.27.23												
SQ	**T**	**H**	**M**	**S**	**F**	**SCENE**	**ID**	**TYPE**	**IX**	**IY**	**RX**	**RY**
2630	87.8	24	1	27	22	1	0	1	139	222	-168	815
2645	88.3	24	1	28	7	1	0	1	147	231	-147	762
2660	88.8	24	1	28	22	1	0	1	155	245	-124	684
2675	89.3	24	1	29	7	1	0	1	161	256	-107	627
2690	89.8	24	1	29	22	1	0	1	169	268	-88.3	568
2705	90.3	24	1	30	7	1	0	1	180	280	-65.6	514
2720	90.8	24	1	30	22	1	0	1	191	292	-44.2	463
2735	91.3	24	1	31	7	1	0	1	204	307	-20.3	403
2750	91.8	24	1	31	22	1	0	1	216	323	0.63	344
2765	92.3	24	1	32	7	1	0	1	227	341	18.9	282
2780	92.8	24	1	32	22	1	0	1	237	359	34.3	225
2795	93.3	24	1	33	7	1	0	1	248	382	50.1	159
2810	93.8	24	1	33	22	1	0	1	263	407	69.4	93.3
2825	94.3	24	1	34	7	1	0	1	277	433	85.7	31.7
2840	94.8	24	1	34	22	1	0	1	296	463	106	-32.2
Time: 1.49.20												
SQ	**T**	**H**	**M**	**S**	**F**	**SCENE**	**ID**	**TYPE**	**IX**	**IY**	**RX**	**RY**
3290	110	24	1	49	22	1	0	1	104	446	-108	2.37
3305	110	24	1	50	7	1	0	1	113	423	-106	53.2
3320	111	24	1	50	22	1	0	1	116	402	-109	104
3335	111	24	1	51	7	1	0	1	121	378	-112	167
3350	112	24	1	51	22	1	0	1	126	359	-113	222
3365	112	24	1	52	7	1	0	1	128	341	-118	279
3380	113	24	1	52	22	1	0	1	130	326	-122	329
3395	113	24	1	53	7	1	0	1	129	311	-131	384
3410	114	24	1	53	22	1	0	1	129	296	-138	443
3425	114	24	1	54	7	1	0	1	131	285	-141	489
3440	115	24	1	54	22	1	0	1	132	271	-148	551
3450	115	24	1	55	2	1	0	1	133	263	-151	589
3465	116	24	1	55	17	1	0	1	134	253	-155	639
3480	116	24	1	56	2	1	0	1	132	244	-165	687
3495	117	24	1	56	17	1	0	1	127	235	-180	736
3510	117	24	1	57	2	1	0	1	122	229	-194	771
3525	118	24	1	57	17	1	0	1	116	223	-210	806
3540	118	24	1	58	2	1	0	1	115	209	-225	896
3555	119	24	1	58	17	1	0	1	119	205	-221	924

Table A2 Continued

Time: 3:06.12												
SQ	**T**	**H**	**M**	**S**	**F**	**SCENE**	**ID**	**TYPE**	**IX**	**IY**	**RX**	**RY**
5590	187	24	3	6	16	1	0	1	196	447	-5.86	0.63
5605	187	24	3	7	1	1	0	1	197	426	-7.76	47
5620	188	24	3	7	16	1	0	1	201	411	-5.28	82.6
5635	188	24	3	8	1	1	0	1	203	394	-5.42	126
5650	189	24	3	8	16	1	0	1	205	381	-4.93	161
5665	189	24	3	9	1	1	0	1	205	367	-7.21	200
5680	190	24	3	9	16	1	0	1	205	356	-9.09	233
5695	190	24	3	10	1	1	0	1	203	344	-14	271
5710	191	24	3	10	16	1	0	1	202	332	-17.8	311
5725	191	24	3	11	1	1	0	1	202	322	-19.9	347
5740	192	24	3	11	16	1	0	1	204	313	-19	380
5755	192	24	3	12	1	1	0	1	207	303	-16.6	419
5770	193	24	3	12	16	1	0	1	211	295	-12.2	451
5785	193	24	3	13	1	1	0	1	214	289	-8.71	477
5800	194	24	3	13	16	1	0	1	217	278	-6.14	525
5815	194	24	3	14	1	1	0	1	216	271	-9.27	558
5860	196	24	3	15	16	1	0	1	221	250	-5.3	663
5875	196	24	3	16	1	1	0	1	225	243	0.36	702
5890	197	24	3	16	16	1	0	1	228	234	4.01	753
5905	197	24	3	17	1	1	0	1	229	231	5.28	771
5590	187	24	3	6	16	1	1	1	279	455	87.2	-15.9
5605	187	24	3	7	1	1	1	1	275	437	83.2	22.8
5620	188	24	3	7	16	1	1	1	278	419	87.3	64.2
5635	188	24	3	8	1	1	1	1	275	404	84.2	101
5650	189	24	3	8	16	1	1	1	275	389	84.6	140
5665	189	24	3	9	1	1	1	1	275	377	85	173
5680	190	24	3	9	16	1	1	1	273	365	82.8	208
5695	190	24	3	10	1	1	1	1	274	353	84.5	245
5710	191	24	3	10	16	1	1	1	275	342	86.3	280
5725	191	24	3	11	1	1	1	1	276	334	88.1	307
5740	192	24	3	11	16	1	1	1	276	320	88.7	357
5755	192	24	3	12	1	1	1	1	277	312	90.6	387
5770	193	24	3	12	16	1	1	1	278	304	92.6	419
5785	193	24	3	13	1	1	1	1	276	293	90.1	464
5800	194	24	3	13	16	1	1	1	275	286	88.8	494
5815	194	24	3	14	1	1	1	1	276	277	91	534
5830	195	24	3	14	16	1	1	1	275	271	89.7	562
5845	195	24	3	15	1	1	1	1	273	263	86.7	601
5860	196	24	3	15	16	1	1	1	273	255	87.1	642
5875	196	24	3	16	1	1	1	1	273	248	87.4	679
5890	197	24	3	16	16	1	1	1	271	242	84.1	712
5905	197	24	3	17	1	1	1	1	271	236	84.4	746

Table A2 Continued

Time: 3.39.06												
SQ	**T**	**H**	**M**	**S**	**F**	**SCENE**	**ID**	**TYPE**	**IX**	**IY**	**RX**	**RY**
6570	219	24	3	39	6	1	0	1	41	405	-197	95.3
6585	220	24	3	39	21	1	0	1	52	383	-196	152
6600	220	24	3	40	6	1	0	1	60	367	-195	197
6615	221	24	3	40	21	1	0	1	63	353	-199	239
6630	221	24	3	41	6	1	0	1	71	337	-198	290
6645	222	24	3	41	21	1	0	1	69	327	-207	324
6660	222	24	3	42	6	1	0	1	73	312	-212	377
6675	223	24	3	42	21	1	0	1	81	299	-209	428
6690	223	24	3	43	6	1	0	1	81	289	-216	469
6705	224	24	3	43	21	1	0	1	87	276	-217	525
6720	224	24	3	44	6	1	0	1	93	265	-216	577
6735	225	24	3	44	21	1	0	1	94	257	-220	616
6750	225	24	3	45	6	1	0	1	99	245	-222	678
6765	226	24	3	45	21	1	0	1	100	231	-233	756
6780	226	24	3	46	6	1	0	1	105	230	-224	763
6795	227	24	3	46	21	1	0	1	111	219	-223	830
6810	227	24	3	47	6	1	0	1	113	213	-225	869
Time: 4.09.18												
SQ	**T**	**H**	**M**	**S**	**F**	**SCENE**	**ID**	**TYPE**	**IX**	**IY**	**RX**	**RY**
7480	250	24	4	9	18	1	0	1	127	209	-202	898
7495	250	24	4	10	3	1	0	1	119	220	-207	825
7510	251	24	4	10	18	1	0	1	110	232	-214	752
7525	251	24	4	11	3	1	0	1	104	245	-213	679
7540	252	24	4	11	18	1	0	1	95	261	-215	596
7555	252	24	4	12	3	1	0	1	84	277	-221	521
7570	253	24	4	12	18	1	0	1	69	294	-231	447
7585	253	24	4	13	3	1	0	1	52	311	-243	380
7600	254	24	4	13	18	1	0	1	29	335	-257	295
7615	254	24	4	14	3	1	0	1	5	355	-274	231

Table A2 Continued

Time: 4.16.28												
SQ	**T**	**H**	**M**	**S**	**F**	**SCENE**	**ID**	**TYPE**	**IX**	**IY**	**RX**	**RY**
7700	257	24	4	16	28	1	0	1	601	364	537	220
7715	257	24	4	17	13	1	0	1	562	349	496	267
7730	258	24	4	17	28	1	0	1	532	343	458	286
7745	258	24	4	18	13	1	0	1	503	334	422	316
7760	259	24	4	18	28	1	0	1	472	314	391	389
7775	259	24	4	19	13	1	0	1	437	314	336	388
7790	260	24	4	19	28	1	0	1	407	309	292	406
7805	260	24	4	20	13	1	0	1	379	301	252	436
7820	261	24	4	20	28	1	0	1	352	297	211	451
7835	261	24	4	21	13	1	0	1	328	291	174	475
7850	262	24	4	21	28	1	0	1	301	286	131	495
7865	262	24	4	22	13	1	0	1	277	281	92.4	516
7880	263	24	4	22	28	1	0	1	252	279	51.3	523
7895	263	24	4	23	13	1	0	1	225	274	6.25	544
7910	264	24	4	23	28	1	0	1	200	268	-36.7	571
7925	264	24	4	24	13	1	0	1	173	265	-83	583
7940	265	24	4	24	28	1	0	1	150	261	-124	601
7955	265	24	4	25	13	1	0	1	133	257	-155	619
7970	266	24	4	25	28	1	0	1	121	256	-176	623
7985	266	24	4	26	13	1	0	1	109	251	-200	648
8000	267	24	4	26	28	1	0	1	99	249	-218	657
8015	267	24	4	27	13	1	0	1	83	245	-250	677
8030	268	24	4	27	28	1	0	1	67	243	-279	686
8045	268	24	4	28	13	1	0	1	49	241	-313	695

Table A2 Continued

Time: 4.51.03												
SQ	**T**	**H**	**M**	**S**	**F**	**SCENE**	**ID**	**TYPE**	**IX**	**IY**	**RX**	**RY**
8695	290	24	4	50	3	1	0	1	265	453	71.6	-11.8
8710	291	24	4	50	18	1	0	1	253	434	57.8	29.3
8725	291	24	4	51	3	1	0	1	245	411	47.6	83.1
8740	292	24	4	51	18	1	0	1	240	390	40.4	137
8755	292	24	4	52	3	1	0	1	235	369	32.5	195
8770	293	24	4	52	18	1	0	1	231	352	25.6	247
8785	293	24	4	53	3	1	0	1	229	336	21.1	299
8800	294	24	4	53	18	1	0	1	228	321	17.9	351
8815	294	24	4	54	3	1	0	1	225	305	11.2	412
8830	295	24	4	54	18	1	0	1	223	292	6.07	465
8845	295	24	4	55	3	1	0	1	218	278	-4.51	526
8860	296	24	4	55	18	1	0	1	215	269	-11.4	567
8875	296	24	4	56	3	1	0	1	211	254	-21.9	641
8890	297	24	4	56	18	1	0	1	207	245	-31.5	689
8905	297	24	4	57	3	1	0	1	203	235	-42	745
8920	298	24	4	57	18	1	0	1	203	221	-47	829

Table A2 Continued

Time: 5.10.00												
SQ	**T**	**H**	**M**	**S**	**F**	**SCENE**	**ID**	**TYPE**	**IX**	**IY**	**RX**	**RY**
9290	310	24	5	10	0	1	0	1	59	459	-152	-24.6
9290	310	24	5	10	0	1	1	1	249	154	34.5	1382
9305	310	24	5	10	15	1	0	1	64	435	-157	25.9
9305	310	24	5	10	15	1	1	1	238	158	8.24	1338
9320	311	24	5	11	0	1	0	1	72	413	-157	76.3
9320	311	24	5	11	0	1	1	1	232	165	-4.28	1268
9335	311	24	5	11	15	1	0	1	73	393	-165	126
9335	311	24	5	11	15	1	1	1	225	171	-18.7	1211
9350	312	24	5	12	0	1	0	1	77	377	-168	169
9350	312	24	5	12	0	1	1	1	225	178	-16.4	1150
9365	312	24	5	12	15	1	0	1	84	358	-169	224
9365	312	24	5	12	15	1	1	1	223	185	-18.6	1091
9380	313	24	5	13	0	1	0	1	104	343	-150	271
9380	313	24	5	13	0	1	1	1	223	189	-17.3	1059
9395	313	24	5	13	15	1	0	1	123	327	-131	326
9395	313	24	5	13	15	1	1	1	219	195	-24	1012
9410	314	24	5	14	0	1	0	1	135	320	-118	351
9410	314	24	5	14	0	1	1	1	219	198	-23	989
9425	314	24	5	14	15	1	0	1	155	304	-95.1	412
9425	314	24	5	14	15	1	1	1	214	203	-31.6	952
9440	315	24	5	15	0	1	0	1	173	293	-71.9	458
9440	315	24	5	15	0	1	1	1	203	209	-51.6	908
9455	315	24	5	15	15	1	0	1	181	283	-62.9	501
9455	315	24	5	15	15	1	1	1	186	217	-81.5	853
9470	316	24	5	16	0	1	0	1	192	273	-48.5	547
9470	316	24	5	16	0	1	1	1	166	223	-117	812
9485	316	24	5	16	15	1	0	1	200	264	-37.9	590
9485	316	24	5	16	15	1	1	1	151	231	-140	762
9500	317	24	5	17	0	1	0	1	211	253	-22.2	646
9500	317	24	5	17	0	1	1	1	141	238	-153	721
9515	317	24	5	17	15	1	0	1	211	237	-26.6	734
9515	317	24	5	17	15	1	1	1	130	245	-168	681
9530	318	24	5	18	0	1	0	1	214	233	-22.3	758
9530	318	24	5	18	0	1	1	1	120	255	-178	628
9545	318	24	5	18	15	1	0	1	215	229	-21.5	781
9545	318	24	5	18	15	1	1	1	107	265	-193	578
9560	319	24	5	19	0	1	0	1	217	219	-20.6	844
9560	319	24	5	19	0	1	1	1	91	274	-212	535
9575	319	24	5	19	15	1	0	1	216	209	-25.6	910
9575	319	24	5	19	15	1	1	1	77	289	-222	468

Table A2 Continued

Time: 5.34.05												
SQ	**T**	**H**	**M**	**S**	**F**	**SCENE**	**ID**	**TYPE**	**IX**	**IY**	**RX**	**RY**
10015	334	24	5	34	5	1	0	1	181	180	-113	1124
10030	335	24	5	34	20	1	0	1	186	183	-100	1100
10045	335	24	5	35	5	1	0	1	192	195	-80.6	1007
10060	336	24	5	35	20	1	0	1	195	200	-71.9	971
10075	336	24	5	36	5	1	0	1	197	206	-65	928
10090	337	24	5	36	20	1	0	1	207	213	-42.2	882
10105	337	24	5	37	5	1	0	1	219	221	-16.1	831
10120	338	24	5	37	20	1	0	1	228	225	2.14	808
10135	338	24	5	38	5	1	0	1	237	229	20	784
10150	339	24	5	38	20	1	0	1	255	243	54.8	705
10165	339	24	5	39	5	1	0	1	268	241	78.6	717
10180	340	24	5	39	20	1	0	1	281	255	101	642
10195	340	24	5	40	5	1	0	1	303	261	139	614
10210	341	24	5	40	20	1	0	1	329	272	181	562
10225	341	24	5	41	5	1	0	1	351	283	214	512
10240	342	24	5	41	20	1	0	1	371	283	248	514
10255	342	24	5	42	5	1	0	1	393	285	284	506
10270	343	24	5	42	20	1	0	1	415	291	316	481
10285	343	24	5	43	5	1	0	1	435	297	345	456
10300	344	24	5	43	20	1	0	1	459	304	379	429
10315	344	24	5	44	5	1	0	1	481	305	413	426
10330	345	24	5	44	20	1	0	1	494	307	433	418

Table A2 Continued

Time: 5.57.15												
SQ	**T**	**H**	**M**	**S**	**F**	**SCENE**	**ID**	**TYPE**	**IX**	**IY**	**RX**	**RY**
10715	358	24	5	57	15	1	0	1	105	200	-254	957
10730	358	24	5	58	0	1	0	1	103	209	-249	894
10745	359	24	5	58	15	1	0	1	95	219	-253	828
10760	359	24	5	59	0	1	0	1	83	231	-263	755
10775	360	24	5	59	15	1	0	1	79	243	-258	687
10790	360	24	6	0	2	1	0	1	71	254	-262	629
10805	361	24	6	0	17	1	0	1	65	269	-258	556
10820	361	24	6	1	2	1	0	1	61	284	-251	489
10835	362	24	6	1	17	1	0	1	61	303	-236	411
10850	362	24	6	2	2	1	0	1	61	317	-225	359
10865	363	24	6	2	17	1	0	1	57	335	-219	296
10880	363	24	6	3	2	1	0	1	55	359	-206	220
10895	364	24	6	3	17	1	0	1	51	381	-198	158
10910	364	24	6	4	2	1	0	1	40	409	-196	85.6
10925	365	24	6	4	17	1	0	1	21	437	-203	21.2
10715	358	24	5	57	15	1	1	1	203	163	-73.6	1281
10730	358	24	5	58	0	1	1	1	209	171	-55.5	1208
10745	359	24	5	58	15	1	1	1	211	174	-49.6	1182
10760	359	24	5	59	0	1	1	1	213	180	-42.5	1130
10775	360	24	5	59	15	1	1	1	208	191	-48.8	1040
10790	360	24	6	0	2	1	1	1	183	189	-103	1052
10805	361	24	6	0	17	1	1	1	155	202	-152	950
10820	361	24	6	1	2	1	1	1	137	210	-181	892
10835	362	24	6	1	17	1	1	1	122	221	-201	819
10850	362	24	6	2	2	1	1	1	111	226	-217	787
10865	363	24	6	2	17	1	1	1	109	239	-209	712
10880	363	24	6	3	2	1	1	1	103	254	-208	632
10895	364	24	6	3	17	1	1	1	95	267	-211	567
10910	364	24	6	4	2	1	1	1	93	281	-203	504
10925	365	24	6	4	17	1	1	1	83	301	-205	420

Table A2 Continued

Time: 6.15.07												
SQ	**T**	**H**	**M**	**S**	**F**	**SCENE**	**ID**	**TYPE**	**IX**	**IY**	**RX**	**RY**
11245	375	24	6	15	7	1	0	1	289	235	118	754
11260	376	24	6	15	22	1	0	1	307	247	149	688
11275	376	24	6	16	7	1	0	1	328	257	184	636
11290	377	24	6	16	22	1	0	1	352	267	223	588
11305	377	24	6	17	7	1	0	1	365	278	240	536
11320	378	24	6	17	22	1	0	1	383	289	265	488
11335	378	24	6	18	7	1	0	1	404	305	290	422
11350	379	24	6	18	22	1	0	1	426	321	315	360
11365	379	24	6	19	7	1	0	1	449	335	341	311
11380	380	24	6	19	22	1	0	1	470	353	359	251
11395	380	24	6	20	7	1	0	1	502	370	392	199
11410	381	24	6	20	22	1	0	1	520	395	399	128
11425	381	24	6	21	7	1	0	1	555	420	426	64.6
11440	382	24	6	21	22	1	0	1	585	443	446	11.1
11455	382	24	6	22	7	1	0	1	623	470	471	-46.1
Time: 6.30.13												
SQ	**T**	**H**	**M**	**S**	**F**	**SCENE**	**ID**	**TYPE**	**IX**	**IY**	**RX**	**RY**
11700	390	24	6	30	12	1	0	1	360	441	181	14.4
11715	391	24	6	30	27	1	0	1	355	421	180	60.2
11730	391	24	6	31	12	1	0	1	349	400	177	112
11745	392	24	6	31	27	1	0	1	348	384	179	155
11760	392	24	6	32	12	1	0	1	339	367	171	204
11775	393	24	6	32	27	1	0	1	334	352	168	250
11790	393	24	6	33	12	1	0	1	328	337	162	299
11805	394	24	6	33	27	1	0	1	325	323	161	348
11820	394	24	6	34	12	1	0	1	318	310	153	397
11835	395	24	6	34	27	1	0	1	317	295	155	458
11850	395	24	6	35	12	1	0	1	309	289	144	483
11865	396	24	6	35	27	1	0	1	302	277	134	536
11880	396	24	6	36	12	1	0	1	297	269	127	574
11895	397	24	6	36	27	1	0	1	291	260	118	618
11910	397	24	6	37	12	1	0	1	285	251	109	664
11925	398	24	6	37	27	1	0	1	284	244	108	702
11940	398	24	6	38	12	1	0	1	280	235	101	753
11955	399	24	6	38	27	1	0	1	281	229	104	789
11970	399	24	6	39	12	1	0	1	278	220	98.9	846
11985	400	24	6	39	27	1	0	1	281	215	105	879
12000	400	24	6	40	12	1	0	1	283	209	110	920

Table A2 Continued

Time: 8.22.09												
SQ	**T**	**H**	**M**	**S**	**F**	**SCENE**	**ID**	**TYPE**	**IX**	**IY**	**RX**	**RY**
15055	502	24	8	22	11	1	0	1	20	417	-215	66.3
15070	503	24	8	22	26	1	0	1	51	381	-198	158
15085	503	24	8	23	11	1	0	1	72	352	-188	242
15110	504	24	8	24	6	1	0	1	86	307	-196	397
15125	505	24	8	24	21	1	0	1	90	284	-206	490
15140	505	24	8	25	6	1	0	1	93	265	-216	577
15155	506	24	8	25	21	1	0	1	97	247	-224	667
15170	506	24	8	26	6	1	0	1	98	229	-238	768
15185	507	24	8	26	21	1	0	1	101	217	-244	842
15200	507	24	8	27	6	1	0	1	105	210	-244	888
15215	508	24	8	27	21	1	0	1	109	199	-247	964
15230	508	24	8	28	6	1	0	1	111	191	-252	1024
15245	509	24	8	28	21	1	0	1	113	183	-257	1086

The walking speed of the pedestrians is calculated from the digitized data using the information from columns T, RX and RY. The following mathematical formulation is used:

Walking speed (m/s) = $[(RX_2 – RX_1)^2 + (RY_2 – RY_1)^2]^{0.5} /[(T_2 – T_1) * 100]$……….(A.1)

Tables A3 and A4 present the walking speed data calculated from the digitized data obtained from Japan. Table A3 presents the walking speed of pedestrians in the major stream, while Table A4 presents the walking speed of pedestrians in minor stream. The average walking speed data from Tables A3 and A4 are presented in Table 1 in Chapter 5.

Table A3 Walking speed of pedestrians in major stream

	Major Stream							
Sample No.	1 male	1 female	person with bags	person pushing wheelchair	motorized wheelchair	couple	2 males	2 females
1	1.13	1.59	0.91	0.9	1.59	0.51	1.22	1.05
2	1.54	1.08	0.98	1.13	1.61	0.85	1.25	1.07
3	1.48			1.27		0.88	1.07	1.54
4	1.21			0.73		0.85	1.19	1.54
5	1			0.72			0.99	0.85
6				0.96			0.84	0.72
7				1.48			1.27	1.18
8				1.13			1.21	1.2
9				0.99			0.99	1.23
10				0.99			1.19	1.18
11				0.84			1.01	
12				1.33			1.44	
13				1.24			1.35	
14				1.24			1.21	
15				0.9			1.21	
16				0.84				

Table A4 Walking speed of pedestrians in minor stream

	Minor Flow					
Sample No.	1 male	1 female	person pushing wheelchair	couple	2 males	2 females
1	0.98	0.95	1	0.83	1.04	1.09
2			1.06	0.77	1.05	1.09
3					1.21	1.11
4					1.15	1.06
5					0.94	0.91
6					1	0.83
7						0.97
8						0.98
9						1.12
10						1.21
11						0.94
12						1.13
13						1.14

Appendix B

TRAVEL TIME DATA FROM UNIVERSITY CAMPUS OBSERVAITONS

Appendix B presents the travel time data that was obtained from measurements taken on campus at the University of Delaware. Table B1 presents the time taken by different pedestrians to walk a 61 ft. long section of walkway. From this data, the walking speed in m/s was calculated. The average walking speed data obtained is presented in Table 5.2 in Chapter 5.

Table B1 Travel time of pedestrians observed on campus at the University of Delaware

	Time in seconds to walk 61 ft.				
Sample No.	couple	1 female	1 male	2/3 female	2/3 male
1	14.99	16.98	12.91	14.97	15.54
2	16.99	13.02	10.91	15.63	14.39
3	14.06	15.27	10.53	14.37	17.26
4	16.6	11.91	12.27	15.68	15
5	14.88	12.17	12.83	15.84	
6	13.44	13.12	12.24	17.24	
7	15.27	13.33	12.81	12.87	
8	13.22	11.11	11.15	18.29	
9	19.54	11.54	12.47	16.76	
10	15.42	14.24	11.54	14.28	
11	12	15.69	12.36	17.03	
12		11.36	9.96	14.25	
13		11.8	9.95	12.01	
14		11.74	12.67	14.84	
15		11.31	13.35		
16		12.72	12.51		
17		9.51	13		
18		10.52	10.66		
19		13.25	11.97		
20		12.77	13.56		
21		13.87	14.02		
22		13.23	13.27		
23		12.43	11.95		
24		11.13			
25		15.24			
26		13.04			

Appendix C

FEET SIMULATION PROGRAMMING CODE

Appendix C presents the five files of the FEET simulation programming code that were modified to incorporate the pedestrian interaction rules developed in Stage I of this thesis study.

C.1 Agent.java

```
import java.awt.*;
import java.awt.geom.*;
import java.util.*;
import javax.swing.*;

public class Agent{

      Point2D.Double justBumped;
      private Agent     parent;
      private Attributes      attrs;
      // Lower left & upper right (x,y) coords of bounding box.
      private int llx, lly, urx, ury; // Bounding box.
      private Color     agtColor;
      private int maxMembers, numMembers; // Member, of child, agents.
      private int lenFront, lenSide; // Size of this agent.
      private int speedAvg, speedCur, speedMax;
      private LinkedList      member, dest, tmpDest;
      private Point2D.Double  ptFrom, ptCur, ptTo;    // Agent from/to
points.
      private String gender, age, handicap; //AG
      private int CTstep; //AG

      Agent() {
            justBumped = null;
            llx = lly = urx = ury = -1;
            lenFront = lenSide = 0;
            member = new LinkedList();
            speedAvg = speedCur = speedMax = 0;
            dest = new LinkedList();
            tmpDest = new LinkedList();
            ptTo = null;
            agtColor = new Color(109, 155, 151);
            gender = age = handicap = null;   //AG
      }

      public void addAgent(Agent agt) {
            member.addLast(agt);
```

```
}

public Agent getAgent(int i) { return (Agent) member.get(i); }

public Attributes getAttributes() { return attrs; }

public Agent getGranddaddy() {
      if (parent == null)
            return this;
      else
            return parent.getGranddaddy();
}

public LinkedList getDest() { return dest; }
public LinkedList getTmpDest() { return tmpDest; }

public Point2D.Double nextDest() {

      if (tmpDest.size() > 0)
            return (Point2D.Double) tmpDest.getFirst();
      else
            return (Point2D.Double) dest.getFirst();
}

public Color getColor() { return agtColor; }
public void setColor(Color color)   { agtColor = color; }
public void setColor(int r, int g, int b) {
      agtColor = new Color(r, g, b);
}

public int getLenFront() { return lenFront; }
public int getLenSide() { return lenSide; }

public String getGender() {return gender;}  //AG
public String getAge() {return age;} //AG
public String getHandicap() {return handicap;}  //AG

public void setCTstep(int i) { CTstep = i; }  //AG
public int getCTstep() { return CTstep;} //AG

public int getLlx() { return llx; }
public int getLly() { return lly; }
public int getUrx() { return urx; }
public int getUry() { return ury; }

public int numAgents() { return numMembers; }

public void setAttributes(Attributes attrs)
            { this.attrs = attrs; }

public Point2D.Double getJustBumped() { return justBumped; }
public void setJustBumped(Point2D.Double pt) { justBumped = pt; }

public int getAvgSpeed() { return speedAvg; }
public int getCurSpeed() { return speedCur; }
public int getMaxSpeed() { return speedMax; }
public void setAvgSpeed(int speed) { speedAvg = speed; }
```

```
public void setCurSpeed(int speed) { speedCur = speed; }
public void setMaxSpeed(int speed) { speedMax = speed; }

public Point2D.Double getFromPt() { return ptFrom; }
public void setFromPt(Point2D.Double pt) { ptFrom = pt; }

public Point2D.Double getCurPt() { return ptCur; }
public void setCurPt(Point2D.Double pt) { ptCur = pt; }

public Point2D.Double getToPt() { return ptTo; }
public void setToPt(Point2D.Double pt) { ptTo = pt; }

public void setBoundingBox(int x0, int y0, int x1, int y1) {
      llx = x0;
      lly = y0;
      urx = x1;
      ury = y1;
}

public int getPosX() { return llx; }
public int getPosY() { return lly; }
public void setPos(int x, int y) {
      llx = x;
      lly = y;
}

public void setParent(Agent parent) {
      this.parent = parent;
}

public void setSize(int front, int side) {
      lenFront = front;
      lenSide = side;
}

public void setGender(String gen)   {
  gender = gen;
}       //AG

public void setAge(String howOld)   {
  age = howOld;
}       //AG

public void setHandicap(String wchair)  {
  handicap = wchair;
}       //AG

public void updateBoundingBox() {
      int i, x0Tmp, y0Tmp, x1Tmp, y1Tmp;
}

int min4(double x0, double x1, double x2, double x3) {
      double min;
      min = Math.min(x0, x1);
      min = Math.min(min, x2);
      min = Math.min(min, x3);
      return (int) Math.floor(min);
```

```
	}

	int max4(double x0, double x1, double x2, double x3) {
		double max;

		max = Math.max(x0, x1);
		max = Math.max(max, x2);
		max = Math.max(max, x3);

		return (int) Math.ceil(max);
	}
}
```

C.2 Source.java

```
import java.awt.*;
import java.awt.geom.*;
import java.util.*;

public class Source{

	private static final double NEARBY = 12.;

	private double	lambda, prob;
	private final int UNIFORM = 0;
	private final int POISSON = 1;

	private Agent	queue[];	// Queue of newly created agents.
	private Attributes	attrs;
	private Cell	cell[][];
	private int arvlProc, posx, posy, qHead, qTail;
	private LinkedList	agentProb; // List of Doubles.
	private LinkedList	agentType; // List of Strings.
	private World	world;

	// From-to coords of line segment on which source resides.
	private int x0, y0, x1, y1;
	private int srcDir; // 1 or -1, dir agents exit source.

	Source() {
		this(0, 0, 0, 0, 1);
	}

	Source(int x0, int y0, int x1, int y1, int dir) {
		arvlProc = -1;
		cell = null;
		this.x0 = x0;
		this.y0 = y0;
		this.x1 = x1;
		this.y1 = y1;
		srcDir = dir;	// Direction agents exit source.
		if ((x0 != x1) && (y0 != y1)) {
			System.out.println("Source (" + x0 + "," + y0 + ")-("
				+ x1 + "," + y1 + ") not horizontal or
vertical!");
			return;
```

```
	}
	setQLen(10);      // Room for 10 agents.
	agentType = new LinkedList();
	agentProb = new LinkedList();
}

/** World grid this source is located in. */

public void setWorld(World world) {
	this.world = world;
	cell = world.getCells();
}

public void setAttrs(Attributes attrs) {
	String ap;

	this.attrs = attrs;
	ap = attrs.getStrVal("process");
	if (ap.compareTo("uniform") == 0) {
		arvlProc = UNIFORM;
		prob = attrs.getNumVal("p");
	} else if (ap.compareTo("poisson") == 0) {
		arvlProc = POISSON;
		lambda = attrs.getNumVal("intensity");
	}

	// Things created by the source.
	// FIXME: must be in lowest to greatest probability order!
	for (attrs.startLoop("agents");
		attrs.loopVarName() != null;
		attrs.nextVar()) {

		agentType.addLast(attrs.loopVarName());
		agentProb.addLast(new Double(attrs.loopVarNumVal()));
	}
}

public void setPos(int x0, int y0, int x1, int y1) {
	this.x0 = x0;
	this.y0 = y0;
	this.x1 = x1;
	this.y1 = y1;
}

public void setQLen(int len) {
	queue = new Agent[len];
	qHead = qTail = 0;
}

/** Add agent(s) to queue based on arrival process properties. */
public void create() {

	double     n;
	int   i;

	if (queue.length == 0)
		return;
```

```
switch (arvlProc) {
case UNIFORM: {
      n = Math.random();
      if (n >= prob)
            break;
            n = Math.random();
      for (i = 0; i < agentProb.size(); i++) {
            if (n <= ((Double) agentProb.get(i)).doubleValue()) {
                  if (qHead == (qTail + 1) % queue.length)
                        break;      // No room for new agent.
            enqueueAgent(queue, (String) agentType.get(i), null);
                  }
            }
      }
      case POISSON: {
      }
      default: {
      }
      }
}

/** Move agent(s) from queue to simulated world. */
public void move() {

      Agent agt;
      boolean     noRoom;
      int         agtFront, agtSide, j, k;
      int         llx, lly, urx, ury;

      if (qHead == qTail)     // No agents enqueued.
            return;

      agt = queue[qHead];
      agtFront = agt.getLenFront();
      agtSide = agt.getLenSide();

      LinkedList agts = world.getAgents();

      if (x0 == x1) {   // Vertically oriented source.

            for (int y = y0; y <= y1-agtFront+1; y++) {

                  // Move along source.
                  // Lower left, upper right of area to search.
                  lly = y;
                  ury = y + agtFront - 1;
                  if (srcDir == 1) {
                        llx = x0;
                        urx = x0 + (agtSide - 1);
                  } else {
                        llx = x0 - (agtSide - 1);
                        urx = x0;
                  }

                  // Find room for a new agent.
                  noRoom = false;
```

```
                  for (j = llx; j <= urx && !noRoom; j++)
                  for (k = lly; k <= ury && !noRoom; k++)
                        if (cell[j][k].getAgent() != null) {
                                    noRoom = true;
                                    break;
                              }
                  if (noRoom)
                        continue;

agt.setPos(llx, lly); // Tell agent where his LL is.
agt.setFromPt(new Point2D.Double(llx, lly));
agt.setCurPt(new Point2D.Double(llx, lly));
queue[qHead] = null; // Remove him from queue.
qHead = (qHead + 1) % queue.length;
agts.add((Object) agt);
LinkedList dest = agt.getTmpDest();
dest.addFirst(new Point2D.Double(llx + NEARBY*srcDir, lly));

      // Place new agent in world.
      for (j = llx; j <= urx && !noRoom; j++)
            for (k = lly; k <= ury && !noRoom; k++)
                  cell[j][k].setAgent(agt);
      if (qHead == qTail)     // No more agents enqueued.
            return;

            }
      } else {
            // Horizontally oriented source.
            for (int x = x0; x <= x1-agtFront+1; x++) {

                  // Lower left, upper right of area to search.
                  llx = x;
                  urx = x + agtFront - 1;
                  if (srcDir == 1) {
                        lly = y0;
                        ury = y0 + (agtSide - 1);
                  } else {
                        lly = y0 - (agtSide - 1);
                        ury = y0;
                  }

      // Find room for a new agent.
      noRoom = false;
      for (j = llx; j <= urx && !noRoom; j++)
            for (k = lly; k <= ury && !noRoom; k++)
                  if (cell[j][k].getAgent() != null) {
                        noRoom = false;
                        break;
                  }
            if (noRoom)
                  continue;

            agt.setPos(llx, lly); // Tell agent where his LL is.
            agt.setFromPt(new Point2D.Double(llx, lly));
            agt.setCurPt(new Point2D.Double(llx, lly));
            queue[qHead] = null; // Remove him from queue.
            qHead = (qHead + 1) % queue.length;
```

```
                agts.add((Object) agt);
        LinkedList dest = agt.getTmpDest();
        dest.addFirst(new Point2D.Double(llx, lly + NEARBY*srcDir));

                        // Place new agent in world.
                        for (j = llx; j <= urx && !noRoom; j++)
                              for (k = lly; k <= ury && !noRoom; k++)
                                    cell[j][k].setAgent(agt);

                  if (qHead == qTail)      // No more agents enqueued.
                              return;

                  }
            }
      }

      private void pickDest(Agent agt, String agentType) {

            Attributes  sinkAttrs;
            double n;
            LinkedList  sinkList;
            Sink  sink;

            n = Math.random();
            sinkList = world.getSinks();
            for (int i = 0; i < sinkList.size(); i++) {
                  // Loop through sinks.
                  sink = (Sink) sinkList.get(i);
                  sinkAttrs = sink.getAttrs();

                  // Probabilities represent fraction of people heading
                  // towards this sink.
      for (sinkAttrs.startLoop("probabilities");
            sinkAttrs.loopVarName() != null; sinkAttrs.nextVar()) {

            String agtType = sinkAttrs.loopVarName();
            double prob = sinkAttrs.loopVarNumVal();
            if ((agtType.compareTo(agentType) == 0) && (n < prob)) {
                  // Add sink's lower left (x, y) to agent's dest list.
                  LinkedList dest = agt.getDest();
                  dest.add(new
                  Point2D.Double(sink.getMidX(), sink.getMidY()));
                              return;
                        }
                  }
            }

      System.out.println("pickDest: ack, shouldn't be here!");
      System.out.println("     sinkList is length " + sinkList.size());
      }


      /** Create agent type <i>agentType</i> and enqueue it. */

  private void enqueueAgent(Agent queue[], String agentType, Agent
  parent) {
```

```
            Agent agt;
            Attributes  agtAttrs, moreAttrs, stillMoreAttrs;

            // Create it.
            agt = new Agent();
            agt.setParent(parent);

            agtAttrs = world.getDefns();
            agtAttrs = agtAttrs.getAttrsVal("agents");
            agtAttrs = agtAttrs.getAttrsVal(agentType);
            moreAttrs = agtAttrs.getAttrsVal("members");

      if (moreAttrs == null) {
            // No "members" => individual walker.
System.out.println("Dr Frankenstein created person type " + agentType +
"!");

                  //   C r e a t e   P e r s o n


                  String gen = agtAttrs.getStrVal("gender");  //AG
            String age= agtAttrs.getStrVal("age");  //AG
            String handicap = agtAttrs.getStrVal("handicap");   //AG

            agt.setGender(gen);  //AG
            agt.setAge(age);  //AG
            agt.setHandicap(handicap); //AG

                  moreAttrs = agtAttrs.getAttrsVal("speed");
                  // Get max & avg speeds.
                  int avg = (int) moreAttrs.getNumVal("avg");
                  agt.setAvgSpeed(avg);
                  int max = (int) moreAttrs.getNumVal("max");
                  agt.setMaxSpeed(max);
                  // Pick this particular agent's initial speed (>= 2).
                  int mySpeed = (int) ((max-2)*Math.random() + 2);
                  agt.setCurSpeed(mySpeed);

            if (parent == null) // Pick dest just for top level groups.
                        pickDest(agt, agentType);

                  // Get agent's length and width.
            if ((moreAttrs = agtAttrs.getAttrsVal("size")) != null)
                        agt.setSize((int) moreAttrs.getNumVal("front"),
                              (int) moreAttrs.getNumVal("side"));

                  // Get agent's color.
            if ((moreAttrs = agtAttrs.getAttrsVal("color")) != null)
                        agt.setColor(
                              (int) moreAttrs.getNumVal("r"),
                              (int) moreAttrs.getNumVal("g"),
                              (int) moreAttrs.getNumVal("b")
                        );

                  if (qHead != (qTail + 1) % queue.length) {
            // Enqueue it, putting an individual in each array cell.
                        queue[qTail] = agt;
```

```
                        qTail = (qTail + 1) % queue.length;
                  }

            } else {

                  //   C r e a t e   G r o u p
System.out.println("Makin' group type " + agentType);

                  //  Random number used to determine which & how many
members to add.
                  double n = Math.random();
                  double p = 0.; // Cumulative probabilty.

                  // Loop through each member type.
                  for (agtAttrs.startLoop("members");
                        agtAttrs.loopVarName() != null;
agtAttrs.nextVar()) {

                        String member = agtAttrs.loopVarName();
                        Attributes mAttrs = agtAttrs.loopVarAttrsVal();
System.out.println("  Can have a member " + member);
System.out.println("  Its attrs: ");
mAttrs.writeAttrs(mAttrs, 0);

                        // Probabilities of given numbers of members.
                        for (mAttrs.startLoop(member);
                  mAttrs.loopVarName() != null; mAttrs.nextVar()) {

                              String probStr = mAttrs.loopVarName(); //
Probability
                              double quantity = mAttrs.loopVarNumVal();
// Quantity
System.out.println("    Prob " + probStr + " of having " + quantity);

                              Double dbl = new Double(probStr);
                              p += dbl.doubleValue();
System.out.println("    Is n < p (" + n + " < " + p + ")?");
                              if (n < p) {
                        // Create given number of member agents.
                        for (int i = 0; i < (int) quantity; i++)
                              enqueueAgent(queue, member, agt);
                                    break;
                              }

                        }
                  }
            }
      }

}
```

C3. Sim.java

```
import java.awt.*;
import java.awt.geom.*;
```

```
import java.io.*;
import java.util.*;

public class Sim {

      private static final double   NEARBY = 12.;
      private static final int      BREAD_CRUMBS = 0;
      private static final int      ERASE = 1;
      private static final int      NO_BUMPS = 2;

      public final int maxTimestep = 3;  //AG

      private     Attributes  attrs, defns, dynamicDefns, staticDefns;
      private     Cell  cell[][];
      private     double      epx0, epy0, epx1, epy1, epDx, epDy,
epPosX, epPosY;
      private     int   frames, xlen, ylen, time, freeCellIndex[][];
      private LinkedList agentList;
      private LinkedList movtPts;
      private LinkedList sinkList;
      private LinkedList srcList;
      private LinkedList wallList;
      private LinkedList wallStack;
      private Vector freeCellList[];
      private World     world;

      Sim(File feetFile) {

            frames = 5; // Assume 5 frames/second frame rate.
            time = 0;

            SimState ss = new SimState(feetFile);
            attrs = ss.init();
            // Parse input (sim description) file.

            xlen = ylen = 50; // Default world size.
            if (attrs != null) {
                  Attributes a = attrs.getAttrsVal("static");
                  if (a != null) {
                        a = a.getAttrsVal("world");
                        if (a != null) {
                              xlen = (int) a.getNumVal("width");
                              ylen = (int) a.getNumVal("height");
                        }
                  }
            }

            world = new World(xlen, ylen);

            cell = world.getCells();
            agentList = world.getAgents();
            sinkList = world.getSinks();
            srcList = world.getSources();
            wallList = world.getWalls();
            wallStack = new LinkedList();
            // Used for obstacle avoidance.
            movtPts = new LinkedList(); // Used for path planning.
```

```
        freeCellList = new Vector[9];
        // Used to find free near cell.
        freeCellList[0] = new Vector(1, 0);
        freeCellList[1] = new Vector(1, 1);
        freeCellList[2] = new Vector(0, 1);
        freeCellList[3] = new Vector(-1, 1);
        freeCellList[4] = new Vector(-1, 0);
        freeCellList[5] = new Vector(-1, -1);
        freeCellList[6] = new Vector(0, -1);
        freeCellList[7] = new Vector(1, -1);
        freeCellList[8] = new Vector(0, 0);
        freeCellIndex = new int[3][3];
        freeCellIndex[0][0] = 5; // (-1, -1)
        freeCellIndex[0][1] = 4; // (-1, 0)
        freeCellIndex[0][2] = 3; // (-1, 1)
        freeCellIndex[1][0] = 6; // (0, -1)
        freeCellIndex[1][1] = 8; // (0, 0)
        freeCellIndex[1][2] = 2; // (0, 1)
        freeCellIndex[2][0] = 7; // (1, -1)
        freeCellIndex[2][1] = 0; // (1, 0)
        freeCellIndex[2][2] = 1; // (1, 1)

        createSim(attrs); // Populates world.
    }

    public int getWorldX()  { return xlen; }
    public int getWorldY()  { return ylen; }

    public int  getFrameRate()               { return frames; }
    public void setFrameRate(int fps)   { frames = fps; }

    public World      getWorld()              { return world; }

    /** Move state of simulation forward by one time step. */
    public void timeStep() {

        Agent agt;
        int   frame, i;

        // Sources create new agents.
        if (srcList != null)
            for (i = 0; i < srcList.size(); i++)
                activateSource((Source) srcList.get(i));

        // Agents update velocity.
        if (agentList != null) {
            for (i = 0; i < agentList.size(); i++) {

                agt = (Agent) agentList.get(i);
                // Remove destinations we've reached.
                //areWeThereYet(agt);
                agt.getTmpDest().clear();
            // Put dest on stack to init updateDestination()
algorithm.

    agt.getTmpDest().addFirst(agt.getDest().getFirst());
            // See if intermediate dests needed to avoid walls.
```

```
                    wallStack.clear();
                updateDestination(agt, agt.getTmpDest(), wallStack);
                    // Remove dest added a few lines above.
                    agt.getTmpDest().removeLast();
                }

                // Perform collision resolution.
                predictedPaths(); // Predict who will crash into who.

        // Move to destination but retain grouping, avoid colliding.
                for (i = 0; i < agentList.size(); i++) {
                    agt = (Agent) agentList.get(i);
                    updateVelocity(agt);
                }
            }

            time++;

        }

        public void activateSource(Source src) {
            src.create();
            // Create agents and add to internal queue.
            src.move();
            // Move 1 or more from queue to sim world.
        }

        /** If a wall exists between current position and destination,
add
          * a new intermediate destination to avoid obstacle.
          * Returns new intermediate destination (Point2D) if an
intermediate
          * destination was found, null otherwise.
          */

// FIXME
// FIXME Add consideration of congestion, not just walls.  From  dir
// FIXME vector <x,y>, use <y,-x> & <-y, x> for 1.5 body widths.  Walk
// FIXME fwd using <x,y> and count density.
// FIXME

        public void updateDestination(Agent agt,
            LinkedList destStack, LinkedList wallStack) {

            boolean     bump, fromTmp;
            Cell  posCell;
            Point2D.Double dest, here, pt[];
            Vector      v;

            dest = (Point2D.Double) destStack.getFirst();
            posCell = doesAgentBumpWall(agt, dest);
            if (posCell == null) // No obstacle, can safely walk
towards it.
                return;

            Wall w = posCell.getWall();
            here = agt.getFromPt();
```

```
            // Unit vector from where pedestrian stands now to next
dest.
            Vector mvt = new Vector(here, dest);

            // Check if bumped wall is already on stack.  Usually means
            // walker is walking nearly parallel and close to wall.
            for (int i = 0; i < wallStack.size(); i++)
                  if (wallStack.get(i) == w) {

                  // We're stuck.  Find nearest open cell towards dest.
                        Vector stuckVec = findFreeCell(agt, dest);
                        stuckVec.mkUnit();
                        stuckVec.multiply(agt.getCurSpeed());
                        stuckVec.add(here);
                        destStack.addFirst(stuckVec.pt2());
                        wallStack.addFirst(w);

                        return;
                  }

            // Consider both endpoints of wall, w0 & w1, and pick best.
            pt = new Point2D.Double[2];
            for (int i = 0; i < 2; i++) {

            // Move a body width or so (random) beyond edge of wall.

            double dist = (1. + 4*Math.random()) * agt.getLenFront();
                  v = w.getU(i); // Unit vector along wall.
                  // FIXME Use MIN(dist, dist-to-nearest-wall)
                  v.multiply(dist); // Dist away.
                  v.add(w.getUstop(i));

// If new dest is offscreen, reflect it back in by same amount.
            //pt[i] = reflectPt(v.pt2());
            pt[i] = v.pt2();
            }

      // At this point, pt[] holds two candidate intermediate dests.
      // We must decide which end of blocking wall to skirt around.

            double d1 = pt[0].distance(dest);
            double d2 = pt[1].distance(dest);
            Point2D.Double newDest;

            // Use intermediate dest which is closest to final dest.
            if (Math.abs(d1 - d2) < NEARBY)
            // Both are about as far from dest, so randomly pick one.
                  newDest = pt[(int) (2. * Math.random())];
            else if (d1 < d2)
                  newDest = pt[0];
            else
                  newDest = pt[1];

            // Don't save intermediate dests close to next dest.
            if (newDest.distance(dest) > NEARBY) {
                  destStack.addFirst(newDest);
                  // Keep track that we almost bumped into this wall.
```

```
            wallStack.addFirst(w);

        // Recurse, see if we'll bump anything enroute to newDest.
            updateDestination(agt, destStack, wallStack);
        }

    }

    private Vector findFreeCell(Agent agt, Point2D.Double there) {

        int indX, indY, lx, ly, x, y;

        Point2D.Double here = agt.getFromPt();
        Point2D.Double emptySpot = new Point2D.Double();
        Vector destFc = new Vector(2); // Free cell destn.
        Vector v = new Vector(here, there);
        double vX = v.getX();
        double vY = v.getY();

        if (Math.abs(vX) < 1e-4)
            indX = 0;
        else if (vX > 1)
            indX = 1;
        else
            indX = -1;
        if (Math.abs(vY) < 1e-4)
            indY = 0;
        else if (vY > 1)
            indY = 1;
        else
            indY = -1;
        int fci = freeCellIndex[indX+1][indY+1];

        lx = agt.getLenFront();
        ly = agt.getLenSide();

        for (int i = 0; i < 5; i++) {

            int j = (fci + i) % 8;
            destFc.set2d(here);
            destFc.add(freeCellList[j]);
            if (roomForAgent(agt, destFc.pt2()))
                return freeCellList[j].copy();

            if (i == 0) // No reflection of vector at 0 degrees.
                continue;
            if (i == 4) // No reflection at 180 degrees.
                break;

            j = fci - i;
            if (j < 0)
                j += 8;
            destFc.set2d(here);
            destFc.add(freeCellList[j]);
            if (roomForAgent(agt, destFc.pt2()))
                return freeCellList[j].copy();
```

```
		}

		fci = freeCellIndex[0][0]; // Can't move!
		return freeCellList[fci].copy();

	}

	public boolean isNextDestTemp(Agent agt) {

		LinkedList tmpDestList = agt.getTmpDest();

		if (tmpDestList.size() > 0)
			return true;
		else
			return false;
	}

	private Cell doesAgentBumpWall(Agent agt, Point2D.Double pTo) {
		return doesAgentBumpWall(agt, agt.getFromPt(), pTo);
	}

	private Cell doesAgentBumpWall(Agent agt,
		Point2D.Double pFrom, Point2D.Double pTo) {

		Cell	posCell;	// Cell where we bump into a wall.
		double	herex, herey, therex, therey;
		int	n, front, side;

		front = agt.getLenFront();
		side  = agt.getLenSide();

		herex = pFrom.getX();	// Current location.
		herey = pFrom.getY();
		therex = pTo.getX();	// Our very next destination.
		therey = pTo.getY();

		for (int x = 0; x < front; x++)
			for (int y = 0; y < side; y++) {

				// Traverse path and mark it.
		setCellEndpoints(herex+x, herey+y, therex+x, therey+y);

				// Walk to pTo & see if we bump anything.
				for (n = 0; n < movtPts.size(); n++) {
					posCell = nextCellInLine(n);
					if (posCell.getWall() != null)
						return posCell;
				}
			}

		return null;
	}

	public void areWeThereYet(Agent agt) {

		Point2D.Double here = agt.getFromPt();
		Point2D.Double there = agt.nextDest();
```

```
            // Here & there are nearby each other.
            if (here.distance(there) < NEARBY) {
                  Cell posCell = doesAgentBumpWall(agt, there);
                  if (posCell == null && isNextDestTemp(agt))
                        agt.getTmpDest().removeFirst();
                        // We've arrived there!
            }
      }

      public void predictedPaths() {

//System.out.println("\n===> TIME t = " + time + "\n");

//System.out.println("=========> predictedPaths() IN");
            // Clear old CRA (collision resolution algorithm)
calculations.
            for (int i = 0; i < xlen; i++)
                  for (int j = 0; j < ylen; j++)
                        cell[i][j].getCra().clear();

            // Clear old CTstep for all agents.
        for (int i = 0; i < agentList.size(); i++) {
            Agent curAgt = (Agent) agentList.get(i);
            curAgt.setCTstep(-1);
            }

            alongPath(BREAD_CRUMBS);
            // Drop bread crumbs where walkers pass.
            // FIXME Not sure cause of bug with ERASE.  Causes some
collisions
            // FIXME to not even be detected.  TOo much being erased??
            //alongPath(ERASE);
            // Erase paths beyond first collision.
            alongPath(NO_BUMPS);
            // Calculate anti-collision vectors.
      }

      private void alongPath(int doWhat) {

            boolean     erase;
            double      x1, y1;
            int   i, j, k, herex, herey, therex, therey, numAgents;
      //    Point2D.Double here, there;

            numAgents = (int) agentList.size();  //AG
            Point2D.Double here[][], there[][];  //AG

            Vector curV[] = new Vector[maxTimestep];  //AG
        here = new Point2D.Double[maxTimestep][numAgents];  //AG
        there = new Point2D.Double[maxTimestep][numAgents];  //AG

      //Make a path for each agent's predicted path for maxTimesteps.
            for (i = 0; i < agentList.size(); i++) {


                  Agent curAgt = (Agent) agentList.get(i);
```

```
            for (int g = 0; g < maxTimestep; g++) {
            if (g == 0) {
                here[g][i] = curAgt.getFromPt();
                there[g][i] = curAgt.nextDest();
                curV[g] = new Vector(here[g][i], there[g][i]);
                  } else {
                        here[g][i] = there[g-1][i];
                        curV[g] = new Vector(here[g][i],
curAgt.nextDest());
                  }
                curV[g].mkUnit();
                curV[g].multiply(curAgt.getCurSpeed());
                curV[g].add(here[g][i]);
                curAgt.setTopt(curV[g].pt2());
                there[g][i] = curAgt.getTopt();

                  if (here[g][i].getX() < there[g][i].getX()) {
                        herex = (int) here[g][i].getX();
                        therex = (int) Math.ceil(there[g][i].getX());
                  } else {
                        therex = (int) there[g][i].getX();
                        herex = (int) Math.ceil(here[g][i].getX());
                  }

                  if (here[g][i].getY() < there[g][i].getY()) {
                        herey = (int) here[g][i].getY();
                        therey = (int) Math.ceil(there[g][i].getY());
                  } else {
                        therey = (int) there[g][i].getY();
                        herey = (int) Math.ceil(here[g][i].getY());
                  }

                  erase = false;
                  for (int y = 0; y < curAgt.getLenSide(); y++)
                        for (int x = 0; x < curAgt.getLenFront(); x++)
{

                              // Traverse path and mark it.
            setCellEndpoints(herex+x, herey+y, therex+x, therey+y);
                  for (k = 0; k < movtPts.size(); k++) {
                        Cell curCell = nextCellInLine(k);
                        CRAinfo cra = curCell.getCra();
                        switch (doWhat) {
                        case BREAD_CRUMBS:
                              cra.addAgent(curAgt, g);
                              break;
                        case ERASE:
                              if (erase || curAgt.getCTstep() > -1)
                                    cra.deleteAgent(curAgt, g);
                              else if (cra.numColliding(g) > 1) {
                                    curAgt.setCTstep(g);
                                    erase = true;
                              }
                              break;
                              Case NO_BUMPS:
                                    cra.mkAntiCVecs(g);
                                    break;
```

```
                              }
                        }
                  }
            }
      }
}

public void updateVelocity(Agent agt) {

      double      curX, curY, newX, newY;
      int   herex, herey, therex, therey;
      Point2D.Double dest, from;

      Point2D.Double here = agt.getFromPt();
      Point2D.Double there = agt.getToPt();
      dest = agt.nextDest(); // Next destination.

      //   S t u c k   i n   C r o w d   ! ! !

      if (agt.getJustBumped() != null) {

            // We're stuck.  Find nearest open cell towards dest.
            Vector stuckVec = findFreeCell(agt, dest);
            stuckVec.mkUnit();
            stuckVec.multiply(agt.getCurSpeed());
            stuckVec.add(here);
            agt.setToPt(stuckVec.pt2());
//System.out.println("Bumped at " + agt.getJustBumped() + ", now to " +
stuckVec.pt2());
            agt.setJustBumped(null);
            return;
      }

      //   D e s t i n a t i o n

      // (x,y) of current lower left corner of agent.
      from = agt.getFromPt();
      curX = from.getX();
      curY = from.getY();

      // Unit vector to dest.
      Vector destV = new Vector(2);
      destV.mkUnit(agt.getFromPt(), dest);

      // Randomly weight dest vec by from 1 to 4.
      // FIXME Should come from agent defn.
      double scaleV = 1. + 3.*Math.random();
      destV.multiply(scaleV);

      //
      //    L e a d e r
      //

      //
      //    C o l l i s i o n
      //
```

```
            double      scaleC = 0.;
            Vector collV = new Vector(0., 0.);

//System.out.println(" " + agt.getFromPt().toString() + " -- " +
agt.getToPt().toString());

            if (here.getX() < there.getX()) {
                  herex = (int) here.getX();
                  therex = (int) Math.ceil(there.getX());
            } else {
                  therex = (int) there.getX();
                  herex = (int) Math.ceil(here.getX());
            }

            if (here.getY() < there.getY()) {
                  herey = (int) here.getY();
                  therey = (int) Math.ceil(there.getY());
            } else {
                  therey = (int) there.getY();
                  herey = (int) Math.ceil(here.getY());
            }

      for (int x = 0; x < agt.getLenFront() && scaleC == 0; x++)
            for (int y = 0; y < agt.getLenSide() && scaleC == 0; y++) {

            // Traverse path and mark it.
            setCellEndpoints(herex+x, herey+y, therex+x, therey+y);

                  for (int j = 0; j < movtPts.size(); j++) {
                        Cell curCell = nextCellInLine(j);
                        CRAinfo cra = curCell.getCra();
                        // Initiate CRA now to avoid bruised walkers.
                        Vector vc = cra.antiC(agt);
                        if (vc != null) {
                              collV = vc;
//System.out.println("**** CollV: " + collV.toString() + "@" +
cra.getCoords());
                        //scaleC = collV.magnitude();
                        scaleC = 1000.;
                        break;
                  }
            }
      }


            //
            //    N o i s e
            //

            // Unit noise vector.
            double      dist, scaleN;
            double      noiseMag = 0.1;
            // FIXME should be from agt attrs.
            Vector noiseV;

            noiseV = new Vector(2);
      noiseV.mkUnit(0., 0., Math.random()-0.5, Math.random()-0.5);
```

```
    noiseV.multiply(noiseMag);
    dist = from.distance(noiseV.pt2());
    if (dist < NEARBY)
        // Closer walker gets to dest, the less noise.
        scaleN = noiseMag * dist/NEARBY;
    else
        scaleN = noiseMag;

    noiseV.multiply(scaleN);

        // If noise makes us bang into wall, zap the noise!
        setCellEndpoints(curX, curY,
            curX + noiseV.getX(), curY + noiseV.getY());
        for (int j = 0; j < movtPts.size(); j++) {
            Cell curCell = nextCellInLine(j);
            if (curCell.getWall() != null) {
                // Noise would make us bang into wall.
                noiseV = new Vector(0, 0);
                scaleN = 0.;
                break;
            }
        }

        //    S c a l e   &   S u m   V e c t o r s

        if (scaleC != 0) {
//System.out.println("collv: " + collV.toString());
            collV.add(agt.getCurPt());
            agt.setToPt(collV.pt2());
//System.out.println("ToPt: " + agt.getToPt().toString());
        } else {

            // Weight above vectors and sum for final velocity.
            Vector vel = new Vector(0., 0.);
            vel.add(destV);
            vel.add(noiseV);
            vel.mkUnit();
            vel.multiply(agt.getCurSpeed());
            // FIXME Speed should vary!!!
            vel.add(from);

//System.out.println("CurPt: (" + curX + "," + curY + ")");
//System.out.println("Noise: (" + noiseV.getX() + "," + noiseV.getY() +
")");
//System.out.println("Vel: (" + vel.getX() + "," + vel.getY() + ")");
    // Point to move to during next time step.
    agt.setToPt(vel.pt2());
// FIXME Sometimes, output below is [Nan, Nan]!!???
//System.out.println("DEST: " + agt.getToPt().toString());
        }

    }

    /** Parses the Attributes tree read in at startup and uses it to
create the initial state of the simulation.  While the Attributes are
var/val pairs where a val may be either a string or more Attributes,
createSims assumes the existence of certain keywords for the <i>var</i>
```

```
name.  If these expectations aren't observed you won't get a working
sim.  So there.  */

      private void createSim(Attributes attrs) {

            Attributes  defns, idefns;

            if (attrs == null || attrs.getVal(0) == null)
                  return;

            //   S t a t i c
            staticDefns = attrs.getAttrsVal("static");
            if (staticDefns != null)
                  world.setDefns(staticDefns);

            //   I n i t i a l
            //   Initial agents are stores, info kiosks, etc.
            defns = attrs.getAttrsVal("initial");
            if (defns != null) {
                  idefns = defns.getAttrsVal("agents");
                  instantiateAgents(idefns);
                  idefns = defns.getAttrsVal("sinks");
                  instantiateSinks(idefns);
                  idefns = defns.getAttrsVal("sources");
                  instantiateSource(idefns);
                  idefns = defns.getAttrsVal("walls");
                  instantiateWalls(idefns);
            }

            //   D y n a m i c
            dynamicDefns = attrs.getAttrsVal("dynamic");
      }

      private void instantiateAgents(Attributes aval) {
            Agent agent;
            Attributes  attrs;
            int   x, y;

            if (aval == null)
                  return;

            for (int i = 0; (attrs = aval.getAttrsVal(i)) != null; i++)
{

            x = y = -1;
            x = (int) getXpos(attrs);
            y = (int) getYpos(attrs);

            if (x == -1 || y == -1) {
            System.out.println("No agent position given, skipping!");
                  return;
                  }

                  agent = new Agent();
                  agent.setPos(x, y);
                  agent.setAttributes(aval);
                  cell[x][y].setAgent(agent);
```

```
            agentList.add(agent);

      }

}

private void instantiateSinks(Attributes aval) {

      Attributes  sinkAttrs, posAttrs;
      int   n, x0, y0, x1, y1;
      Sink  sink;

      if (aval == null)
            return;

for (int i = 0; (sinkAttrs = aval.getAttrsVal(i)) != null; i++) {

            x0 = y0 = x1 = y1 = -1;

            posAttrs = sinkAttrs.getAttrsVal("from");
            x0 = (int) getXpos(posAttrs);
            y0 = (int) getYpos(posAttrs);

            posAttrs = sinkAttrs.getAttrsVal("to");
            x1 = (int) getXpos(posAttrs);
            y1 = (int) getYpos(posAttrs);

            if (x0 == -1 || y0 == -1 || x1 == -1 || y1 == -1) {
      System.out.println("No source endpoint given, skipping!");
                  continue;
            }

            int dir = (int) aval.getNumVal("dir");
            sink = new Sink(x0, y0, x1, y1, dir);
            sink.setAttributes(sinkAttrs);

            if (x0 == x1) {
                  int min = Math.min(y0, y1);
                  int max = Math.max(y0, y1);
                  for (int j = min; j <= max; j++)
                        cell[x0][j].setSink(sink);
            } else {
                  int min = Math.min(x0, x1);
                  int max = Math.max(x0, x1);
                  for (int j = min; j <= max; j++)
                        cell[j][y0].setSink(sink);
            }

            sinkList.add(sink);

      }
}

private void instantiateSource(Attributes aval) {
      Attributes  attrs, posAttrs;
      Cell  posCell;
      int   n, x0, y0, x1, y1;
```

```
		Source	src;

		if (aval == null)
			return;

		for (int i = 0; (attrs = aval.getAttrsVal(i)) != null; i++)
{

			x0 = y0 = x1 = y1 = -1;

			posAttrs = attrs.getAttrsVal("from");
			x0 = (int) getXpos(posAttrs);
			y0 = (int) getYpos(posAttrs);

			posAttrs = attrs.getAttrsVal("to");
			x1 = (int) getXpos(posAttrs);
			y1 = (int) getYpos(posAttrs);

		if (x0 == -1 || y0 == -1 || x1 == -1 || y1 == -1) {
		System.out.println("No source endpoint given, skipping!");
			continue;
			}

			int dir = (int) attrs.getNumVal("dir");
			src = new Source(x0, y0, x1, y1, dir);
			src.setAttrs(attrs);
			src.setQLen((int) attrs.getNumVal("qlen"));
			src.setWorld(world);
			// Source needs agent definitions.
			//cell[x][y].setSource(src);
			srcList.add(src);
		}

	}

	private void instantiateWalls(Attributes aval) {

		Attributes	attrs, posAttrs;
		Cell	posCell;
		double	x0 = -1., y0 = -1., x1 = -1., y1 = -1.;
		int	n;
		Wall	wall;

		if (aval == null)
			return;

		// Loop through all wall descriptions.
		for (int i = 0; (attrs = aval.getAttrsVal(i)) != null; i++)
{

			x0 = y0 = x1 = y1 = -1;

			// Get points from Feet file.
			posAttrs = attrs.getAttrsVal("from");
			x0 = getXpos(posAttrs);
			y0 = getYpos(posAttrs);
			posAttrs = attrs.getAttrsVal("to");
```

```
            x1 = getXpos(posAttrs);
            y1 = getYpos(posAttrs);

      if (x0 == -1 || y0 == -1 || x1 == -1 || y1 == -1) {
      System.out.println("No wall endpoint given, skipping!");
            continue;
            }

            // Create the wall.
            wall = new Wall(x0, y0, x1, y1);

            // Get wall color.
            if ((posAttrs = attrs.getAttrsVal("color")) != null)
                  wall.setColor(
                        (int) posAttrs.getNumVal("r"),
                        (int) posAttrs.getNumVal("g"),
                        (int) posAttrs.getNumVal("b")
                  );

            // Get wall thickness.
            double wallThickness = attrs.getNumVal("thickness");
            if (!(new Double(wallThickness)).isNaN())
                  wall.setWidth((float) wallThickness);

      // Make wall findable by SimPane so it can draw the wall.
            wallList.add(wall);

            // Each cell covered by wall now points to wall data.
            setCellEndpoints(x0, y0, x1, y1);
            n = 0;
            do {
                  posCell = nextCellInLine(n++);
                  posCell.setWall(wall);
            } while (posCell != cell[(int)x1][(int)y1]);

      }
}

private double getXpos(Attributes attrs) {
      Attributes  loc;

      if (attrs == null)
            return -1.;
      loc = attrs.getAttrsVal("pos");
      if (loc == null)
            return -1.;
      return loc.getNumVal("x");
}

private double getYpos(Attributes attrs) {
      Attributes  loc;

      if (attrs == null)
            return -1.;
      loc = attrs.getAttrsVal("pos");
      if (loc == null)
            return -1.;
```

```
		return loc.getNumVal("y");
	}

	public void ageState(Agent agent) {
		agent.setFromPt(agent.getToPt()); // Update position.
	}

public void setCellEndpoints(Point2D.Double p0, Point2D.Double p1) {

	setCellEndpoints((int) p0.getX(), (int) p0.getY(), (int)
p1.getX(), (int) p1.getY());
	}

	public void setCellEndpoints(int x0, int y0, int x1, int y1) {
		setCellEndpoints((double) x0, (double) y0, (double) x1,
(double) y1);
	}

	/** After completing, an internal ordered list of the cells that
	will be traversed is created. */

	public void setCellEndpoints(double dx0, double dy0,
		double dx1, double dy1) {

		double curX, curY;
		int x0, y0, x1, y1;

		if (dx0 <= dx1) {
			x0 = (int) Math.floor(dx0);
			x1 = (int) Math.ceil(dx1);
		} else {
			x0 = (int) Math.ceil(dx0);
			x1 = (int) Math.floor(dx1);
		}

		if (dy0 <= dy1) {
			y0 = (int) Math.floor(dy0);
			y1 = (int) Math.ceil(dy1);
		} else {
			y0 = (int) Math.ceil(dy0);
			y1 = (int) Math.floor(dy1);
		}

		movtPts.clear(); // Zap old calcs.

		if (x0 == x1) {
			// Vert line.
		if (y0 < y1)
			for (int i = y0; i <= y1; i++) {
				Point2D.Double pt = new Point2D.Double(x0, i);
				movtPts.addLast(pt);
			}
		else
			for (int i = y0; i >= y1; i--) {
				Point2D.Double pt = new Point2D.Double(x0, i);
				movtPts.addLast(pt);
				}
```

```
            } else if (y0 == y1) {
            // Horiz line.
      if (x0 < x1)
            for (int i = x0; i <= x1; i++) {
                  Point2D.Double pt = new Point2D.Double(i, y0);
                  movtPts.addLast(pt);
            }
      else
            for (int i = x0; i >= x1; i--) {
                  Point2D.Double pt = new Point2D.Double(i, y0);
                  movtPts.addLast(pt);
            }
      } else {
      double m = (double) (y1 - y0) / (x1 - x0);      // Slope.
      double b = y1 - (m * x1);                 // Y axis intercept.
                  int xDir = -1;
                  if (x0 < x1)
                        xDir = 1;
                  int yDir = -1;
                  if (y0 < y1)
                        yDir = 1;

                  //
                  //   X - a x i s
                  //

            if (xDir == 1) {
                  for (int i = x0; i <= x1; i++) {
                  Point2D.Double pt = new Point2D.Double(i, (m*i)+b);
                  movtPts.addLast(pt);
            }
            } else {
                  for (int i = x0; i >= x1; i--) {
                  Point2D.Double pt = new Point2D.Double(i, (m*i)+b);
                              movtPts.addLast(pt);
                        }
                  }

                  //
                  //   Y - a x i s
                  //

            if (yDir == 1) {

            for (int yCoord = y0; yCoord <= y1; yCoord++) {
            double xCoord = (yCoord - b) / m;
            Point2D.Double pt = new Point2D.Double(xCoord, yCoord);

                              // Insert point keeping list sorted.
                              boolean added = false;
                              for (int j = 0; j < movtPts.size(); j++)
{
                  curX = ((Point2D.Double) (movtPts.get(j))).getX();
                  curY = ((Point2D.Double) (movtPts.get(j))).getY();

                  if (xDir == 1) {
                        if (xCoord < curX) {
```

```
                        movtPts.add(j, pt);
                        added = true;
                        break;
                        } else if (xCoord == curX) {
                        if (yCoord < curY) {
                        movtPts.add(j, pt);
                              added = true;
                              break;
                        } else if (yCoord == curY) {
                              added = true; // Not really!
                              break;
                        }
                  }
                  } else {
                        if (xCoord > curX) {
                        movtPts.add(j, pt);
                        added = true;
                        break;
                  } else if (xCoord == curX) {
                        if (yCoord < curY) {
                              movtPts.add(j, pt);
                              added = true;
                              break;
                        } else if (yCoord == curY) {
                              added = true; // Not really!
                              break;
                              }
                        }
                  }
            }
      if (!added)
            movtPts.addLast(pt);
      }

      } else {

for (int yCoord = y0; yCoord <= y1; yCoord++) {
      double xCoord = (yCoord - b) / m;
      Point2D.Double pt = new Point2D.Double(xCoord, yCoord);

      // Insert point keeping list sorted.
      boolean added = false;
      for (int j = 0; j < movtPts.size(); j++) {
            curX = ((Point2D.Double) (movtPts.get(j))).getX();
            curY = ((Point2D.Double) (movtPts.get(j))).getY();

            if (xDir == 1) {
                  if (xCoord < curX) {
                        movtPts.add(j, pt);
                        added = true;
                        break;
                  } else if (xCoord == curX) {
                        if (yCoord > curY) {
                              movtPts.add(j, pt);
                              added = true;
                              break;
                        } else if (yCoord == curY) {
```

```
                                added = true; // Not really!
                                break;
                            }
                        }
                    } else {
                        if (xCoord > curX) {
                            movtPts.add(j, pt);
                            added = true;
                            break;
                        } else if (xCoord == curX) {
                            if (yCoord > curY) {
                                movtPts.add(j, pt);
                                added = true;
                                break;
                            } else if (yCoord == curY) {
                                added = true; // Not really!
                                break;
                            }
                        }
                    }
                }
                if (!added)
                    movtPts.addLast(pt);
                }
        }

}
}

        public Cell nextCellInLine(int n) {

                int   x, y;

                if (n < 0)
                        n = 0;
                else if (n >= movtPts.size())
                        n = movtPts.size() - 1;

                x = (int) ((Point2D.Double) (movtPts.get(n))).getX();
                y = (int) ((Point2D.Double) (movtPts.get(n))).getY();

                // Make sure a valid cell is chosen.
                // FIXME Reflect or interpolate, not this!
                x = (int) Math.max(x, 0.);
                y = (int) Math.max(y, 0.);
                x = (int) Math.min(x, xlen-1);
                y = (int) Math.min(y, ylen-1);

                return cell[x][y];
        }

        /** If point lies beyond rectangle (0, 0)-(xlen-1, ylen-1)
            then reflect it back by however much it lies beyond. */
        private Point2D.Double reflectPt(Point2D.Double pt) {

                double x, y;
```

```
            x = pt.getX();
            y = pt.getY();

            if (x < 0)
                  x = -x;
            else if (x >= xlen)
                  x -= x - xlen + 1;

            if (y < 0)
                  y = -y;
            else if (y >= ylen)
                  y -= y - ylen + 1;

            pt.setLocation(x, y);
            return pt;
      }


      public void updateDestinationHmmm(Agent agt,
            LinkedList destStack, LinkedList wallStack) {

            boolean     bump, fromTmp;
            Cell  posCell;
            Point2D.Double dest, here, pt[];
            Vector      v;

            dest = (Point2D.Double) destStack.getFirst();
            posCell = doesAgentBumpWall(agt, dest);
            if (posCell == null) // No obstacle, can safely walk
towards it.
                  return;

            Wall w = posCell.getWall();
            here = agt.getFromPt();
            // Unit vector from where pedestrian stands now to next
dest.
            Vector mvt = new Vector(here, dest);

            // Check if bumped wall is already on stack.  Usually means
            // walker is walking nearly parallel and close to wall.
            for (int i = 0; i < wallStack.size(); i++)
                  if (wallStack.get(i) == w) {

                  Vector wv0 = new Vector(here, w.getP0());
                  Vector wv1 = new Vector(here, w.getP1());
                  double wTheta = wv0.cosTheta(wv1);
                  double dist0 = here.distance(w.getP0());
                  double dist1 = here.distance(w.getP1());
                  double distW = w.getP0().distance(w.getP1());
                  double onEdge;
                  if (dist0 < dist1) {
                        onEdge = Math.abs(dist1 - (dist0 + distW));
                  } else
                        onEdge = Math.abs(dist0 - (dist1 + distW));
                        if ((wTheta < 0.1 || wTheta > (3.1416 - 0.1))
                              && (onEdge <= 1.0)) {
//                if (   here.distance(w.getP0()) <= agt.getLenFront()
```

```
//                || here.distance(w.getP1()) <= agt.getLenFront()) {

                  // Walker is looking end-on at wall.  "Else" clause
                  // below won't work in this case.

                  // Try moving orthogonally away from wall edge.
                  wv0 = w.getU0();
                  wv0.orthogonal();
                  wv0.multiply(1.5 * agt.getLenFront());
                  wv0.add(here);
            Cell bumpCell = doesAgentBumpWall(agt, wv0.pt2(), dest);
                  if (bumpCell == null || bumpCell.getWall() == w) {
                        destStack.addFirst(wv0.pt2());
                        wallStack.addFirst(w);
                        return;
                  }

            // Try moving orthogonally away from wall edge
            // in opposite direction from above attempt.
                  wv1 = w.getU0();
                  wv1.orthogonal();
                  wv1.negate(); // Use the other orthogonal.
                  wv1.multiply(1.5 * agt.getLenFront());
                  wv1.add(here);
                  bumpCell = doesAgentBumpWall(agt, wv1.pt2(), dest);
                  if (bumpCell == null || bumpCell.getWall() == w) {
                        destStack.addFirst(wv1.pt2());
                        wallStack.addFirst(w);
                        return;
                  }
            // No clear path, just stumble around a ttle...
            System.out.println("Can't find clear path - randomizing!");
            wv1.mkUnit(0., 0., Math.random()-0.5, Math.random()-0.5);
            wv1.multiply(3);
            wv1.add(here);
            destStack.addFirst(wv1.pt2());
            wallStack.addFirst(w);

            } else {
            // Will bump into nearby wall, so walk parallel to it.
            v = w.getU0(); // Unit vector along wall.
            v.perpProj(mvt); // Component of velocity along wall.
            v.subtract(mvt); // Get perpendicular distance from wall.
            double mag = v.magnitude();
            if (mag < agt.getLenSide()) {
                  v.mkUnit();
                  v.multiply(agt.getLenSide());
            }
            v.add(dest); // Anchor v at dest.
            destStack.addFirst(v.pt2());
            wallStack.addFirst(w);
            }
            return;
}

            // Consider both endpoints of wall, w0 & w1, and pick best.
            pt = new Point2D.Double[2];
```

```
            for (int i = 0; i < 2; i++) {

                  // Move a body width or so (random) beyond edge of
wall.
            double dist = (1. + 4*Math.random()) * agt.getLenFront();
                  v = w.getU(i); // Unit vector along wall.
                  // FIXME Use MIN(dist, dist-to-nearest-wall)
                  v.multiply(dist); // Dist away.
                  v.add(w.getUstop(i));
      // If new dest is offscreen, reflect it back in by same amount.

                  //pt[i] = reflectPt(v.pt2());
                  pt[i] = v.pt2();
            }

            // At this point, pt[] holds two candidate intermediate
dests.
            // We must decide which end of blocking wall to skirt
around.

            double d1 = pt[0].distance(dest);
            double d2 = pt[1].distance(dest);
            Point2D.Double newDest;

            // Use intermediate dest which is closest to final dest.
            if (Math.abs(d1 - d2) < NEARBY)
            // Both are about as far from dest, so randomly pick one.
                  newDest = pt[(int) (2. * Math.random())];
            else if (d1 < d2)
                  newDest = pt[0];
            else
                  newDest = pt[1];

            // Don't save intermediate dests close to next dest.
            if (newDest.distance(dest) > NEARBY) {
                  destStack.addFirst(newDest);
                  // Keep track that we almost bumped into this wall.
                  wallStack.addFirst(w);
            // Recurse, see if we'll bump anything enroute to newDest.
                  updateDestination(agt, destStack, wallStack);
            }

      }

      private boolean roomForAgent(Agent agt, Point2D.Double there) {

            double thereX = there.getX();
            double thereY = there.getY();

            if (   thereX < 0 || thereX >= xlen
                  || thereY < 0 || thereY >= ylen)

                  return false;

            int x0 = (int) thereX;
            int y0 = (int) thereY;
```

```
		for (int i = 0; i < agt.getLenFront(); i++)
			for (int j = 0; j < agt.getLenSide(); j++)
				if (cell[x0+i][y0+j].getWall() != null)
					return false;
				else if (cell[x0+i][y0+j].getAgent() != null &&
					cell[x0+i][y0+j].getAgent() != agt)
					return false;
		return true;
	}

}
```

C.4 CRAinfo.java

```
import java.awt.geom.*;
import java.util.*;

public class CRAinfo {

private boolean		acCalcd[];
// Anti-collision vectors calculated yet?
private LinkedList	whenList;
// Fraction into trip when crash will occur.
private LinkedList	agtLst[]; // Folks involved in crash.
private LinkedList	fracList;
// Fractions of dist-to-crash/dist-to-dest.
private LinkedList	ordered[]; // Priority sorted list of agents.
private LinkedList	preVelList[];
// Predicted "walls" created by velocities.
private LinkedList	acVelList[];
// Future velocities to avoid collisions.
private Point2D.Double	crash;
// Coords of predicted crash (this cell).

	private int maxTimestep;

	CRAinfo(int x, int y, int t) {
		maxTimestep = t;
		acCalcd = new boolean[maxTimestep];
		for (int i = 0; i < maxTimestep; i++)
			acCalcd[i] = false;
		acVelList = new LinkedList[maxTimestep];
		// List of Vectors.
		agtLst = new LinkedList[maxTimestep]; // List of Agents.
		crash = new Point2D.Double(x, y); // Coords of this cell.
		fracList = new LinkedList(); // List of doubles.
		ordered = new LinkedList[maxTimestep]; // List of Agents.
		preVelList = new LinkedList[maxTimestep];
		// List of Vectors.
		whenList = new LinkedList(); // List of doubles.
	}

	CRAinfo(int x, int y) { this(x, y, 3); }
```

```
	public int getMaxTimestep() { return maxTimestep; }
	public void setMaxTimestep(int t) {
		maxTimestep = t;
		acCalcd = new boolean[maxTimestep];
		acVelList = new LinkedList[maxTimestep]; // List of
Vectors.
		agtLst = new LinkedList[maxTimestep]; // List of Agents.
		ordered = new LinkedList[maxTimestep]; // List of Agents.
		preVelList = new LinkedList[maxTimestep]; // List of
Vectors.
	}

	public void clear() {
		for(int k = 0; k < maxTimestep; k++) {
			acCalcd[k] = false;
			acVelList[k].clear();
			agtLst[k].clear();
			ordered[k].clear();
			preVelList[k].clear();
		}
		whenList.clear();
		fracList.clear();
	}

	public int numColliding(int g) {
		return agtLst[g].size();
	}

	public void setCrashCoords(double x, double y) {
		crash.setLocation(x, y);
	}

	public Point2D.Double getCoords() {
		return crash;
	}

	public void addAgent(Agent agent, int j) {

		if (agtLst[j].indexOf(agent) != -1)
			return; // Don't add duplicates.

		agtLst[j].addLast(agent);
		Point2D.Double p0 = agent.getFromPt();
		Point2D.Double p1 = agent.getToPt();
		// From/to endpoints of velocity vector.
		Vector vel = new Vector(p0, p1);
		preVelList[j].addLast(vel);
		if (j == 0) {
			Vector toCrash = new Vector(p0, crash);
			double frac = toCrash.magnitude() / vel.magnitude();
			fracList.addLast(new Double(frac));
		}
//System.out.println("Adding " + frac + " to end of fracList");
//if (agtLst.size() > 1)
//System.out.println("Predicted crash: " + crash);
	}
```

```
	public void deleteAgent(Agent agent, int i) {

		int pos = agtLst[i].indexOf(agent);
		if (pos == -1)
			return; // Can't delete him if he doesn't exist!

		agtLst[i].remove(pos);
		preVelList[i].remove(pos);
		if (i == 0)
			fracList.remove(pos);
	}

	/** Return anti-collision vector allowing agent to probably
	    avoid imminent collision. */
	public Vector antiC(Agent agt) {
		int pos;
		for (int i = 0; i < maxTimestep; i++) {
			pos = ordered[i].indexOf(agt);
			if (pos != -1)
			return (Vector) acVelList[i].get(pos);
		}
			return null;
	}

	// Given two vectors and corresponding points of origin,
	// determine which vector is "first," i.e., represents a
pedestrian
	// who is ahead of another.
	//
	// Front quarter (roughly head on) collisions have no "first" so
	// null is returned.  Roughly side-on collisions are the same
since
	// again each pedestrian has eye contact with the other.  Only
when
	// one has no sight of the other is he said to be first.  Easy
	// concept, messy vectors...

	public Vector whosFirst(Point2D.Double p1, Vector v1,
		Point2D.Double p2, Vector v2) {

		Vector firstGuy1, firstGuy2;

		double ct = v1.cosTheta(v2);
		if (ct < 0) // Obtuse angle => head on crash => no one is
1st.
			return null;

		Vector fromTo = new Vector(p1, p2);

		// Perpendicular projection of v1 onto v2.
		Vector tmp1 = v2.copy();
		tmp1.perpProj(v1);
		tmp1.perpProj(fromTo);
		double ct1 = tmp1.cosTheta(v2);
		if (ct1 >= -1. && ct1 <= -0.99)
			firstGuy1 = v1;
		else
```

```
                firstGuy1 = v2;

          // Perpendicular projection of v2 onto v1.
          Vector tmp2 = v1.copy();
          tmp2.perpProj(v2);
          tmp2.perpProj(fromTo);
          double ct2 = tmp2.cosTheta(v1);
          if (ct2 >= -1. && ct2 <= -0.99)
                firstGuy2 = v1;
          else
                firstGuy2 = v2;

          if (firstGuy1 == firstGuy2)
                return firstGuy1;

          return null;

    }

    /**
     Rank agents from highest to lowest priority.  The idea is that
     walkers who will collide latest in their trip have more time to
     react. Therefore, walkers are prioritized, highest to lowest,
     based on the distances from their starting point to the
potential
     point of impact.
     */

private void orderAgts() {

//System.out.println("agtLst.size() is " + agtLst.size());
//System.out.println("fracList.size() is " + fracList.size());
ordered[0].clear();
for (int i = 0; i < agtLst[0].size(); i++) {

Agent curAgt = (Agent) agtLst[0].get(i);
// How far into trip collision will occur.
double curAgtMag = ((Double) fracList.get(i)).doubleValue();
Vector curAgtVel = new Vector(curAgt.getFromPt(), curAgt.getToPt());

      boolean added = false;
      for (int j = 0; j < ordered[0].size(); j++) {
            Agent ordAgt = (Agent) ordered[0].get(j);
            int k = agtLst[0].indexOf(ordAgt);
            double ordAgtMag = ((Double)
            fracList.get(k)).doubleValue();
            Vector ordAgtVel = new Vector(ordAgt.getFromPt(),
            ordAgt.getToPt());

            Vector v = whosFirst(curAgt.getFromPt(), curAgtVel,
            ordAgt.getFromPt(), ordAgtVel);

            if (v == null) {
            if (curAgtMag < ordAgtMag) {
                  ordered[0].add(j, curAgt);
                  added = true;
                  break;
```

```
            }
        } else if (v == curAgtVel) {
            ordered[0].add(j, curAgt);
            added = true;
            break;
        }
    }

    if (!added)
        ordered[0].addLast(curAgt);
    }
    //System.out.println("Ordered list is:");
    //for (int i = 0; i < ordered.size(); i++) {
    //Agent aaa = (Agent) ordered.get(i);
    //int jjj = agtLst.indexOf(aaa);
    //System.out.println("   " + ((Double)
    fracList.get(jjj)).doubleValue());
    //}
    }

    private void orderAgts( int g) {
           ordered[g].clear();
           for (int i = 0; i < agtLst[g].size(); i++) {

            Agent curAgt = (Agent) agtLst[g].get(i);

    // Get the characteristics of the curAgt.
            String curAgtGender = (String) curAgt.getGender();
            String curAgtAge = (String) curAgt.getAge();
            String curAgtHandicap = (String) curAgt.getHandicap();
            boolean added = false;

            for (int j = 0; j < ordered[g].size(); j++) {
                Agent ordAgt = (Agent) ordered[g].get(j);
                String ordAgtGender = (String) ordAgt.getGender();
                String ordAgtAge = (String) ordAgt.getAge();
                String ordAgtHandicap = (String) ordAgt.getHandicap();

                if (curAgtHandicap.compareTo("Yes") == 0)
                    if (ordAgtHandicap.compareTo("Yes")== 0) {
                        double r = Math.random();
                        if (r < 0.5) {
                            ordered[g].add(j, curAgt);
                            added = true;
                            break;
                            }
                        else {
                            ordered[g].add(j+1, curAgt);
                            added = true;
                            break;
                            }
                        }
                else if (ordAgtHandicap.compareTo("Yes")== 0) {
                        ordered[g].add(j+1, curAgt);
                        added = true;
                        break;
                        }
```

```
            else if (curAgtGender.compareTo("Female") == 0)
                    if (ordAgtGender.compareTo("Female") == 0) {
                        double r = Math.random();
                        if (r < 0.5) {
                                          ordered[g].add(j, curAgt);
                            added = true;
                            break;
                            }
                        else {
                            ordered[g].add(j+1, curAgt);
                            added = true;
                            break;
                            }
                        }
            else if (ordAgtGender.compareTo("Female") == 0) {
                    ordered[g].add(j+1, curAgt);
                    added = true;
                    break;
                    }
            else if (curAgtAge.compareTo("Adult") == 0)
                    if (ordAgtAge.compareTo("Adult") == 0) {
                        double r = Math.random();
                        if (r < 0.5) {
                            ordered[g].add(j, curAgt);
                            added = true;
                            break;
                            }
                        else {
                            ordered[g].add(j+1, curAgt);
                            added = true;
                            break;
                            }
                        }
            if(!added)
                ordered[g].addLast(curAgt);
        }
    }
}


      private Vector orderedVelocity(int timestep, int whichOne) {

            // Given index, find Agent in ordered list.
            int g = timestep;
            Agent orderedAgt = (Agent) ordered[g].get(whichOne);

            // Given index of agent in ordered list, find his index
            // in the original unordered list.
            int unorderedIndex = agtLst[g].indexOf(orderedAgt);

            // Given an Agent in the original unordered list, find his
            // corresponding velocity from the original velocity list.
            Vector v = (Vector) preVelList[g].get(unorderedIndex);

            return v;
      }
```

```
      /*
       *   m k A n t i C V e c s
       *
       *   Go through ordered list.  1st agent keeps going.  2nd must
       *   avoid 1st, 3rd and higher slow down to avoid reaching
       *   impact point.  The assumption is that for most people that's
       *   easier than trying to calculate a safe path.
       */

      public void mkAntiCVecs(int g) {

            if (agtLst[g].size() < 2 || acCalcd[g])
                  return; // No anti-collision needed.

//System.out.println("Anti-C Code! @ " + crash.toString());
            if (g == 0)
                  orderAgts(); // Order agents by fraction-of-trip to
impact.
            else orderAgts(g);
            acVelList[g].clear();

            Agent agt0 = (Agent) ordered[g].get(0);
            Agent agt1 = (Agent) ordered[g].get(1);
            Vector v0 = orderedVelocity(g, 0);
            Vector v1 = orderedVelocity(g, 1);
            double ct = v0.cosTheta(v1);

//System.out.println("v0:"+v0.toString()+", v1:"+v1.toString()+", CT =
" + ct);
            if (ct >= -1.0 && ct <= -.98) {
//System.out.println("HEAD ON!");
                  // Head-on collision.
      Vector v0Tov1 = new Vector(agt0.getFromPt(), agt1.getFromPt());
      Vector v0ToDest = new Vector(agt0.getFromPt(), agt0.getToPt());
      v0Tov1.subtract(v0ToDest);
      Vector v0Orth = v0.orthogonal();
      v0Orth.perpProj(v0Tov1);
      if (v0Orth.magnitude() < 1e-4) // Exactly head on!
            v0Orth = v0.orthogonal();
            v0Orth.mkUnit();
            v0Orth.negate();
      if (g == 0) {
            v0Orth.multiply(0.7 *agt0.getLenFront());
            v0.add(v0Orth);
      } else
            v0Orth.multiply(2 *agt0.getLenFront());
            v1.subtract(v0Orth);
//System.out.println("v0: " + v0.toString() + ", v1: " +
v1.toString());
      acVelList[g].add(0, v0);
      acVelList[g].add(1, v1);
} else {

      // Walker #0's velocity remains unchanged.
      acVelList[g].add(0, v0);

      // Have walker #1 avoid #0.
```

```
      double theta1Before = orderedVelocity(g, 1).theta(v0);
      // Adjust goal to where walker #0 is at this instant.  Since
      // he'll be moving, he should be gone when we get there.
      v1 = new Vector(agt1.getFromPt(), agt0.getFromPt());
      // FIXME Should use max(agt1, agt) lengths really...
      v1.subtract(1+agt1.getLenFront(), 1+agt1.getLenSide());
      double theta1After = v1.theta(v0);

//System.out.println("  Collision");
//System.out.print  ("    agt0:" + v0.toString());
//System.out.println(", at " + agt0.getFromPt().toString());
//System.out.print  ("    agt1 at " + agt1.getFromPt().toString());
//System.out.flush();

// See if we must overtake from behind or avoid head on collision.
      if (Math.abs(theta1After - theta1Before) < 0.5) {
//System.out.println("  INLINE");
// Determine which side we're on of #0.
            Vector vel1 = new Vector(agt1.getFromPt(), agt1.getToPt());
                        double vel1Mag = vel1.magnitude();
//System.out.println("    vel1: " + vel1.toString());
                        Vector v1new = vel1.orthogonal();
//System.out.println("    v1new: " + v1new.toString());
                        v1new.perpProj(v1);
//System.out.println("    v1new (PP v1): " + v1new.toString());
            if (v1new.magnitude() < 1e-4) // Exactly head on.
                  v1new = vel1.orthogonal();
            v1new.negate();
//System.out.println("    v1new negated: " + v1new.toString());
            v1new.mkUnit();
            v1new.multiply(1.5 * agt0.getLenFront());
//System.out.println("    v1new mag 2: " + v1new.toString());
            v1new.add(agt0.getFromPt());
//System.out.println("    v1new @ agt0-FP: " + v1new.toString());
            // Agt1 vector from here to nearest side of agt0.
            v1 = new Vector(agt1.getFromPt(), v1new.pt2());
            double v1Mag = v1.magnitude();
            v1Mag = Math.min(v1Mag, vel1Mag);
            v1.mkUnit();
            v1.multiply(v1Mag);
      }
                  acVelList[g].add(1, v1);
//System.out.println("    Final:" + v1.toString());
//System.out.flush();

            }

      // Walkers #2,... simply slow down to avoid point of impact.
      for (int i = 2; i < ordered[g].size(); i++) {
      // Calculate collision avoidance vector for this pedestrian.
            Agent agtn = (Agent) ordered[g].get(i);
            Vector veln = new Vector(agtn.getFromPt(), crash);
            veln.multiply(0.5);
            acVelList[g].add(i, veln);
      }
      acCalcd[g] = true;
```

```
        }
}
```

C5. *.feet

```
static {
        world {
                width   100
                height  60
        }

        agents {
                male {
                        speed {
                              avg     8
                              max     10
                        }
                        // Geometry of a single male is 3 x 3 cells.
                        size {
                                front    3
                                side     3
                        }
                        gender          male
                        age             adult
                        handicap        no
                }

                female {
                        speed {
                              avg     8
                              max     10
                        }
                        // Geometry of a single female is 3 x 3 cells.
                        size {
                                front    3
                                side     3
                        }
                        gender          female
                        age             adult
                        handicap        no
                }

                child {
                        speed {
                                avg     6
```

```
                max     8
            }
            // Geometry of a commuter is 2 x 2 cells.
            size {
                front   2
                side    2
            }
            gender          neutar
            age             young
            handicap        no
        }

        wheelchair {
            speed {
                avg     7
                max     9
            }
            // Geometry of a wheelchair is 4 x 3 cells.
            size {
                front   4
                side    3
            }
            gender          neutar
            age             adult
            handicap        yes
            }
        }

        family {
            members {
                male            0.8
                female          0.8
                wheelchair      0.3
                child {
                    // probability          number of children
                        0.1                     3
                        0.3                     0
                        0.3                     1
                        0.3                     2
                }
            }
        }
    }
}

initial {
```

```
sinks {

	lowerRight {
		from {
			pos {
				x	99
				y	52
			}
		}
		to {
			pos {
				x	99
				y	58
			}
		}
		probabilities {
			// Everyone uses this sink with probability 1.
			male		1
			female		1
			child		1
			wheelchair	1
			family		1
		}
	}

}

sources {

	upperLeft {
		// Vertical source.
		from {
			pos {
				x	0
				y	1
			}
		}
		to {
			pos {
				x	0
				y	6
			}
		}
		// Source created entities exit to right.
		dir		1
```

```
		// Set internal queue size.
		qlen	20
		// Uniform probability of entity creation/time step.
		processuniform
		p		0.1
		// What to create.
		agents {
			// Create agents with following probabilities
			child		0.1
			wheelchair	0.1
			male		0.25
			female		0.25
			family		0.30
		}
	}

}

walls {

	wall1 {
		from {
			pos {
				x	0
				y	0
			}
		}
		to {
			pos {
				x	93
				y	0
			}
		}
	}

	wall2 {
		from {
			pos {
				x	0
				y	7
			}
		}
		to {
			pos {
				x	0
				y	59
```

```
		}
	}
}

wall3 {
	from {
		pos {
			x	7
			y	59
		}
	}
	to {
		pos {
			x	99
			y	59
		}
	}
}

wall4 {
	from {
		pos {
			x	99
			y	0
		}
	}
	to {
		pos {
			x	99
			y	53
		}
	}
}

diagonal1 {
	from {
		pos {
			x	35
			y	20
		}
	}
	to {
		pos {
			x	55
			y	40
		}
```

```
		}
}

diagonal2 {
	from {
		pos {
			x	45
			y	20
		}
	}
	to {
		pos {
			x	65
			y	40
		}
	}
}

top {
	from {
		pos {
			x	35
			y	20
		}
	}
	to {
		pos {
			x	45
			y	20
		}
	}
}

bottom {
	from {
		pos {
			x	55
			y	40
		}
	}
	to {
		pos {
			x	65
			y	40
		}
	}
```

```
                }

        }
}

dynamic {
}
```