Todo list

SIMOBS Thesis

Djura Smits

September 21, 2011

Abstract

Contents

1 Introduction

In many fields using simulation, the environments are empty except for the absolutely necessary parts of the simulation. However, the environment would look much more realistic if actual civilians would walk around. That is why SIMOBS was developed. SIMOBS is a plugin for the VR-Forces simulation application that makes it easy to quickly generate residential areas inhabited by families, and factories where these civilians go to work. However, at the moment these virtual pedestrians only walk from their houses to their work and back at fixed points in time. The purpose of this thesis is to extend this SIMOBS plugin to create more realistic behavior.

1.1 SIMOBS & VR-Forces

SIMOBS is a plugin for the military simulation toolkit VR-forces [15].

Fix website reference.

It is a tool for generating and executing battlefield scenarios. SIMOBS is a plugin developed to quickly generate inhabitants of an area by drawing residential areas and plants on the map in VR-forces. The behavior of these inhabitants is determined by *Daily Motion Patterns*, which specify where certain types of inhabitants need to go at specified times. The functionality of VR-forces can easily be extended using plugins. B-have is a plugin that is particularly useful when working with pedestrian simulations. Among other things, it adds more sophisticated path planning to the toolkit. This means that our SIMOBS extension does not have to deal with planning the pedestrians' paths.

Tweak research question.

1.1.1 Research Question

The question we are going to base this research around is the following:

> How can a distributed intelligent virtual environment for simulated pedestrians be extended to deal with time-restricted destinations?

Some of these terms may need some clarification:

- *Distributed*:
  We are using a distributed environment to simulate our pedestrians. This means that various tasks such as path planning can be delegated to other services and are not of our concern. Therefore, we can afford to focus more on higher level behavior.

- *Intelligent Virtual Environment*:
  IVE is a broad term, but in this particular case we mean that a large portion of the intelligence needed for the pedestrians to walk around is placed in the environment, instead of in the pedestrians walking in it.
- *Time-restricted destinations*:
  We would like to create a framework that is able to deal with departure hall-like situations. That means that pedestrians will have a destination (e.g. a train, or an airplane, etc.) that will be available for a limited amount of time (until it departs). It can also be used to create simulations with a pattern that is more realistic when run for a long period of time (such as a whole day). Furthermore, in many situations, only the time is known when the person has to arrive at his destination. In those cases it is more intuitive to define the time of arrival, instead of determining at what time someone has to leave his starting point. In the rest of the article, we will refer to these time restrictions as *deadlines*.

We are going to solve this question by trying to answer the following subquestions:

- *To what extent can pedestrians be simulated realisticly?*
  If it were feasible to model the complete human brain, we would probably get the most lifelike behavior. However, since this is not possible, we have to simplify the model somehow. Models can be made in varying levels of complexity. Most often, a higher complexity means slower performance. That is why we have to think about how to get the right balance between realism and performance. We also have to decide how we define realism. Do we take the inner model into account, or do we purely compare the resulting behavior to real people in similar situations?
- *How do we let the pedestrians make decisions based on time left to reach the destination?*
  In situations such as departure halls, people have a destination (e.g. airplane, train) that is only available for a limited amount of time. People may also have other needs, such as eating food, or going to the toilet. How do we let these pedestrians decide between the different options?
- *What kind of freedoms/restrictions do distributed systems give us?*
  As a consequence of using a distributed system we can delegate certain tasks to other units. On the other hand, our model also needs to be compatible with the other units in the distributed system.
- *Is it possible to quickly generate these virtual pedestrians without*

much tweaking for each environment?
We aim at creating pedestrians that can be used in many different environments without much additional scripting. Is it possible to do this and still have varied behavior between environments?

2 Related Work

Indicate tuples for every approach (as much as is possible).

Crowd simulation is a subject that attracts a lot of interest. For many areas of research, such as simulations of panic situations could not do without the modelling of crowds. This field knows many approaches. First of all, the focus can lie on the group as a whole. The different members of the group are then often viewed as particles that influence the other group members near them with attracting and repulsive forces. Other approaches view crowds as a group of individuals, and the members are given some kind of simplified psychological model. The motivation behind this simplified model is that this will lead to complex behavior when many of these simple units are put together in a large crowd. This effect is known as *emergence*.
The aproaches that fall under the previously mentioned categories generally focus on the general movement of crowds of pedestrians. However, when we want truly realistic behavior in a regular public environment, these methods do not suffice, because they miss interactions with objects in the environment. Approaches dealing with this generally tackle the problem in a very different way. Instead of giving the pedestrians all the knowledge there is to know about how to deal with these objects, this information is put in the objects themselves. When a pedestrian approaches such an object, this knowledge is augmented to the basic knowledge of the pedestrian, or the object is even put into control of the pedestrian.
These different approaches will be discussed in the next few sections.

2.1 Group Interaction

The research of interaction in groups started with Reynolds' boids [13] , where members of the group were seen as particles that exert both repulsive and attracting forces on the other members of the flock, depending on the distance to one another, and forces inwards from the outer contour of the flock, in order to remain in a certain shape. However, this flocking behavior is more suitable for modelling behavior of animals such as fish, and will not give a very plausible result when it is used to model humans. Because humans usually act in a way more complicated than a flock. However, many researches have built

upon this idea of modelling large groups by viewing the members as particles.

2.1.1 Global approaches

Many models that have been proposed focus on crowds in panic situations. An example of a model for panic situations is the one proposed by Pelechano et al. [11] who divided people in three categories: *trained leaders*, who have complete knowledge about the building, *untrained leaders*, who handle stress well, help others and will explore the building, and *untrained non-leaders*, who might panic.

Maybe sections have to be named differently, or previous reference has to be moved, the approach uses global effects, but also more individual models

Another example of simulating panic situations, can be found in the article of Helbing, Farkas, and Vicsek [6]. In their model, people exert a repulsive interaction force to stay away from each other, an additional *body force* slowing to counteract body compression, and a *sliding friction force* when a pedestrian comes in contact with another pedestrian or the wall. This model can lead to several effects known to occur in real panic situations.
While these methods give a good insight into the movements in those particular panic situations, they are less suitable for experiments running over a longer period of time. Only in those few moments of panic, or when crowds are very dense, do these models represent a crowd realisticly. What we are looking for, is a framework that gives realistic behavior over longer periods of time. Pelechano, Allbeck and Badler [10] have simulated high-density crowds for normal situations. They base the movement of the crowds on a simple wayfinding algorithm and a number of different psychological (impatience, panic, personality attributes, etc.) and physiological traits(e.g.locomotion and energy level). Furthermore, the agent is given perception and will react to objects and other pedestrians in the nearby space.
Bayazit, Lien and Amato approached the subject of crowd simulation in a very different way [2]. Instead of letting an entity such as one pedestrian or a group do the navigation on the fly, global roadmaps are used. In this method, a map is generated beforehand defining where the pedestrians can walk. During simulation, the pedestrians are given a goal location, and will then explore the paths defined by the map, based on which direction exercises the highest force on the pedestrians. The pedestrians can update this map in real-time indicating if a path is favorable or not when trying to get to the goal. When two paths exert an equal amount of force on the pedestrians, they will split up and both paths will be explored simultaneously. This is a huge differ-

ence with the previous approaches, since in the previous approaches, crowds would generally stay together.

This is not completely true, some panic approaches let the crowds explore, but probably had simpler navigation. Make the previous more nuanced

Find out more about crowd simulation using cellular automata

2.1.2 Smaller Groups and Individual Approaches

The previously mentioned approaches focus on movements of crowds as a whole. This might generate realistic effects when the crowds are very dense, but when the pedestrians are more sparsely scattered in the environment, these models will not suffice. That is why there have been many researches focusing more on crowds as a collection of smaller groups. In an urban environment, most pedestrians move around together with a few other pedestrians

Confirm claim about pedestrians moving in groups

and very few move around on their own. An important step in this direction has been made by Li, Jeng and Chang [16], who proposed a leader-follower model, in which one person in a group gets the role of *leader*, who has the job to decide on the destination and has to plan the path. This leader will exert an attractive force on the *followers*, who will continuously follow this leader around. Hostetler and Kearny chose an approach in which all members of the group cast an equal vote on which direction to head in [7]. Every member of a group casts a vote on which way to turn and how to adjust the speed based on a discretisized action space. The group will then collectively follow the action that has the highest vote.

Mention the ribbon they use for navigation

Peters, Ennis and O'Sullivan decided to have a more direct approach to the formation of groups [12]. They studied a large video corpus of prototypical walking areas and concluded groups always occur in certain formations. Subsequently, they designed a number of formations in a *formation template* that represent discrete formations that the pedestrian groups may adopt, such as walking completely abreast, or in a staggered formation. The distance between the group members is defined by a cohesian matrix, which describes the cohesion between every two members in the group. Another factor contributing to the distance between each member is the minimum frontal aspect the formation can have, which describes the width of the formations. Until now, we have only seen models that deal with crowds in terms

of walking behavior. Interpersonal relationships may have been somewhat defined, but were only expressed through spacial positions. Bcheiraz and Thalmann have attempted to express these interpersonal dynamics through a set of animations that express a persons mood through body language [3].

Clarify paper some more

2.2 Interaction with Objects

As previously mentioned, most researches with the focus on crowds as a group of individuals or viewed globally do not address the problem of how to interact with the environment except for some collision detection. This greatly reduces the realism of the simulation. The obvious solution is to extend the knowledge of the pedestrians with instructions about how to interact with these objects. This has been succesfully done for instance by Shao and Terzopoulos [1]. They used an extensive psychological model to determine the behavior of the individuals. This led to a simulation in which the behavior looks very realistic, even when one invidual is followed for a long time. The downside to this method is that the behavioral model for the pedestrians have to be specifically crafted for the environment, which will be very time consuming.
It would be easier to generate the virtual human agents if the environment would automatically decide for the agents what interactions are possible and appropriate. The first step towards this focus was made by Kallmann and Thalmann who introduced the principle of *smart objects* [9]. Smart objects take complete control of the agent when he approaches the object. The agent has no information at all about these objects, all necessary data such as the animation for the agent when interacting is contained within this object. The object also keeps track about how many agents can interact with it at the same time and if the interaction should be the same for all agents. For instance, an elevator modeled as a smart object will make the first agent interacting with it press the button, but not the next agents that approach this object.

Reword so it is more clear what I mean

By using smart objects, the internal model of the pedestrians can be kept very simple, because they do not need to remember specific information about how to interact with the objects. Furthermore, this means that the pedestrians do not have to be specifically designed for the current simulation environment, because the environment will tell them how to act. In the most basic approach to smart objects, the agents lose all their autonomy when they approach a smart object. A lot of research has been built upon the idea of smart objects. For instance, Kallmann, de Sevin and Thalmann have extended this model

to have agents that have their own motivations and needs [8]. This model uses five main motivation types: eat, drink, rest, work, and go to toilet. These motivations control the action through a hierarchical decision graph. Information about which objects fulfill these different needs are added to the smart objects.
Another take on intelligent environments was given by Sung, Gleicher and Chenny [14]. They create an intelligent environment based on *situations*. A situation is an area in the environment that requires the pedestrians to act in a certain way. The behavior of a pedestrian is described by a finite state machine in which a state is defined as follows:

$$s = \{t, \mathbf{p}, \theta, \mathbf{s}^-\}$$

In which t is the time, $\mathbf{p}$ is the current position in two dimensional space, θ is the orientation, a is an action, and s^- is a list of previous states. By "action" they mean a particular animation clip that has to be played at this state. Situations extend the pedestrians' finite state machine with situation-specific actions. The probability distribution of the actions the pedestrian can take is multiplied with the probability distribution given by the situation. Situations are divided into two categories: *spatial* situations for stationary objects or areas, and *non-spatial* situations to describe concepts such as friendship with another pedestrian.

2.2.1 Mixed Approaches

The distinction between larger and smaller groups is not black and white, as can be seen in the work of Braun et al. [4], who generalized the model of Helbing, and introduced additional features for creating group behaviors, such as family members, dependence level, altruism level, and desired speed of the agent.
Farenc et al. even devised a hierarchical framework that incorporates a number of different models managing the crowd on different levels (crowd behavior, group specification, group behavior, and individual behavior). A framework such as this is very suitable for incorporating multiple crowd behavior techniques. While a single method of the previously mentioned techniques might not generate a satisfactory result, a hierarchic combination of several methods might be able to do the job.
Several other methods also have the potential to be extended to incorporate several other methods. For instance, it is very likely that the "situations" framework can be adapted to deal with higher level states, instead of the low-level animation-oriented states that are used now. It will then probably be possible to let situations incorporate certain effects (eg. a repulsive force) instead of simple animations. These effects could be one (or parts) of the previously mentioned other models.

2.3 General Behavior Modeling

2.3.1 Finite State Automata

Finite-state machines (or finite state automata) are behavioral models that are composed of a number of states associated to transitions. A finite state machine moves from state to state by doing sets of actions associated with certain transitions. They are widely used in a variety of applications such as electronic design automation, but also for parsing.

2.3.2 Petri Nets

Petri nets are a mathematical modeling language used for the description of distributed systems. A petri net is a bipartite graph consisting of two types of nodes: places and transitions. These nodes are connected by directed arcs. An arc can run from either a place to a transition, or from a transistion node to a place, but never from a place to a place, or between two transitions. Activity in a Petri net is expressed by by the movement of tokens from place to place, through transitions. Input arcs (from place to transition) denote which places need to contain tokens in order to enable the transition. When a transition is enabled, it consumes the tokens from the input places, and produces tokens in the place indicated by the output arc. Basic Petri nets can be described by a five-tuple:

$$PN = (P, T, I, O, M_0) \tag{1}$$

which comprises of

- a set of places $P = (p_1, p_2, ..., p_m)$,
- a set of transitions $T = (t_1, t_2, ..., o_m)$,
- a set of input arcs $I \subset P \times T$,
- a set of output arcs $O \subset T \times P$,
- an initial marking $M_0 = (m_{01}.m_{02}, \ldots, m_{0m})$.

Petri nets have been extended in many ways in order to accomodate many different functionalities. The extention that attracts our attention the most is *Stochastic Petri nets*. In this extension, there are two types of transitions: *immediate* and *timed* transitions. The Stochastic Petri net (SPN) model can be described as a six-tuple:

$$SPN = (P, T, I, O, M_0, \Lambda) \tag{2}$$

where (P, T, I, O, M_0) is the marked untimed PN underlying the SPN, and $\Lambda = (\lambda_1, \lambda_2, \ldots, \lambda_n)$ is an array of (possibly marking dependent) firing rates associated with transitions.

Immediate transitions always have priority over timed transitions, and the likelihood of firing a timed transition is dependent on a parameter called the *firing rate* of the transition. This rate indicates the firing delay of the timed transition. This firing rate may be marking-dependent, so it should be written as $\lambda_i(M_j)$. The average firing delay of a transition t_i in marking M_j is $[\lambda_i(M_j)]_{-1}$.

Voeg het volgende stukje toe

Quote from: http://www.ac.tuiasi.ro/pntool/help/ii36generalizedstochasticpetrinets.htm: Generalized Stochastic PNs (GSPNs) have two different classes of transitions: immediate transitions and timed transitions. Once enabled, immediate transitions fire in zero time. Timed transitions fire after a random, exponentially distributed enabling time as in the case of SPNs. For timed transitions, the firing rate (i.e. the inverse of the mean time-duration) is, by default, marking independent, but the user can select a marking-dependent operation (the same way as for SPNs).

2.3.3 Advantages of Petri Nets over Finite State Automata

Petri nets hold several advantages over finite state automata. First of all, Petri nets allow for concurrent behavior. This would enable our pedestrians to do several tasks at once. For instance, it could be possible to model our pedestrians' Petri nets so that a token would represent an arm or a leg. That way, a pedestrian could execute multiple basic tasks independently. It would not be possible to achieve this kind of behavior with a finite state machine, unless a state would be described for every possible combination of activities. However, this is not the only advantage Petri nets have over finite state automata. Petri nets come with a standardized way to incorporate a time aspect. For finite state automata, several techniques have been developed, but there is no general consensus over what methods are most suitable. The problem becomes even larger when we would like to incorporate both time and non-determinism. Both the aspects of time and non-determinism are easily incorporated in Petri nets with only small modifications, and can even be mixed with traditional Petri nets. An essential issue which has to be implemented in the method we are going to choose, is non-determinism. Fortunately, this is one of the basic properties of a Petri net. When multiple transitions are enabled at the same time, any of them may fire.
Lastly, availability will not be an issue either, since there are many packages for many languages freely available on the web.

2.4 Time Planning

The essential extension to the situations framework that is proposed in this thesis adds an element of time to the system. This is needed to enable the system to deal with daily motion patterns. An important element without which the system cannot succeed is knowledge about how long actions are going to take. Only when this information is known to the agent (or system) it can be decided whether taking a certain action will result exceeding the deadline for the goal.

2.5 Which Techniques Can We Use?

We have seen many different techniques for many different purposes. We can immediately eliminate most global approaches to crowds, since for most environments, we do not want the pedestrians to move as one big mass, all pedestrians heading into the same direction. Using purely attracting and repulsive forces will also generally not work when the pedestrians are too far apart, so we have another reason to rule out most global techniques. Of course, the global roadmaps approach does enable the pedestrians to split up and walk in different directions, but using this system on its own will not create very realistic behavior, since it does not have the capability to deal with interpersonal relations, and interaction with objects. The techniques focusing on smaller groups do show some potential. Since most people move in smaller groups, this can lead to realistic effects. Ideally, people should interact with each other based on a variety of social relationships such as "parent" and "friend". However, this will require much configuration beforehand, and may increase processing time for each decision the pedestrians have to make. In this case, configuration may not be the issue, since these relationships can be defined and then used for many different simulations, but processing time is a larger problem, increasing with the number of pedestrians present in the simulation. Of all the possible existing techniques, the "situations" approach of Sung et al. seems the most appealing. However, it is not directly usable for our purposes. One downside to the situations approach (at least how it is described in the paper) is that state transitions are mainly low-level animations. However, it should be possible to adapt the state transitions to correspond to higher-level actions, such as change of goal. Path planning should then be delegated to another module. In this way, it is very likely that the state representation can be reduced from $s = \{t, \mathbf{p}, \theta, \mathbf{s}^-\}$ to $s = \{t, \mathbf{p}, \theta\}$. The s^- was used to remember which actions were taken previously, so that the pedestrian remembers which way it was walking. When the actual walking is delegated to another module, it seems unnecessary to keep s^-. Another advantage of eliminating s^- is that we get to keep the Markov property probabilistic

finite state automata are supposed to have.

3 Method

We decided to base our model around the situations framework. We chose this framework because it allows for probabilistic behavior, and puts emphasis on defining behavior through the environment. This framework will have to be extended to deal with time restrictions. We will do this by replacing the finite state machines with Petri nets. The toolkit we will use for this is *Platform Independent Petri net Editor 2* (PIPE2). We chose this toolkit for a few reasons. First of all, the toolkit is written in Java, which makes it relatively easy to use it together with our own Java code. Secondly, PIPE2 has incorporated a large amount of Petri net extensions, such as *Generalized Stochastic Petri nets*, which add timed transitions to the Petri net framework. Lastly, the PIPE2 toolkit has a clear, simple gui which can be used to create and modify Petri nets.

Create reference to website

Name disadvantages? Most disadvantages only become clear after PIPE2 has been used for a while so choosing PIPE was mostly because there was not enough time to extensively study alternatives, but it shouldn't be put like that.

Extend introduction.

Probably remove from here ↓

3.1 Global Needs Layer

First of all, we will implement a layer on top of the situations framework that will deal with distributing the available needs over the pedestrians. A need is available if there are situations in the environment that can fulfill this need. This global needs layer will check for every situation which need it can fulfill and will add it to the list of available needs. A need includes the following information:

- **Id**
  Of course, a need has to have a unique name by which it can be identified.
- **Intensity function**
  Needs can be dependent on which time of day it is. For instance, people are more likely to be hungry around noon and at the beginning of the evening. That is why a need defines a function which computes the intensity of a need as a function of time. In most cases this function will probably sample from a normal distribution.

- There must be more.

When the different needs have been collected, they will be spread in various intensities accross the pedestrians. For each pedestrian the need intensity function will be called. Since this function is generally probabilistic, this will result in a population that varies in needs.

3.2 Local Needs Layer

The various need intensities will be sent to the additional *local needs layer* of the pedestrians. This layer will combine the information of the global needs with additional information about personal needs. These personal needs are generally dependent on the pedestrian's *personal clock* which keeps track of how much time the pedestrian has left to reach its destination. While the global needs layer and the local needs layer will only communicate occasionally, since the global needs do not change so quickly, the local needs layer will communicate with its pedestrian for every step. The task of the local needs layer is to convert the information about the various needs into behavior (i.e. a timed state automaton).
However, the communication between the local needs layer and the pedestrian layer also goes the other way around. When a pedestrian has executed an action that has fulfilled a certain need, this will be propagated back to the local needs layer so that need can be lowered.

3.2.1 Converting Needs to Finite State Automata

Think about this approach. Do we really want to influence the probability of the fsm of the whole situation or could we better use the needs to influence individual transitions?

An important question one might ask himself is "How do these needs contribute to the actual behavior of the pedestrians?". The answer is that these needs specify how high the probabilities are of the situation-specific transitions. The probabilities of the transitions will be multiplied by the intensity of the need its situation fulfills. Afterwards, the transitions exiting the state are normalized to get the actual probabilities.

↑ to here

3.3 Pedestrian Layer

Some drastic changes will be made in order to adapt the situations framework to our needs. Instead of using regular finite state automata, we use Petri nets so that environments in which time constraints are

important (e.g. train stations, airports, etc.) can be properly dealt with.

3.3.1 Time Planning

The fact that we use Petri nets with timed transitions, instead of finite state automata does not necessarily mean our pedestrians are able to deal with time constraints. However, these Petri nets have helped making our problem representable. In order to use these stochastic Petri nets efficiently for making decisions based on time constraints, we have made a number of assumptions.
First of all, we assume there are certain *base places* from which it is always possible to reach the (time constrained) goal. Then, we will compute for every transition that will not take the pedestrian to its goal, how much time it takes to get back to the base state. Then we can check whether the goal place is still accessible from the base state when a certain transition has been taken. We use this information to modify the timed transition rate, so pedestrians are more likely to choose the actions that leave them more time to reach their goal.

Define function f based on experimenting with different functions. Also, check if it is still correct to refer to transition probabilities

As one may have noticed, this approach does not give a completely watertight solution to the planning problem. However, since we have to be able to model large crowds, we cannot create an overly complex planning system, since we would not be able to run the simulation real-time. Furthermore, we do not aim at finding an optimal solution to the planning problem, but rather the most lifelike behavior. In real life, people make errors in judgement, so creating pedestrians who can look ahead perfectly would not be realistic.

Elaborate on why this is more lifelike

3.4 Assumptions

The method we propose is based on various assumptions which have to be clarified. An important assumption is that the environment the pedestrians have to walk in are designed in such a way that it helps creating realistic behavior. This means that situations have to be defined in such a way that pedestrians will walk into them and act in the appropriate way. In many cases, intuitively designed environments will cause the right behavior. For instance, in most train stations, the eating stands are placed in such a way that pedestrians will have to walk close to them in order to get to their destination.

Find some evidence that train stations are designed like this.

Another assumption we make is that the pedestrians have to be at a certain place at a certain time. This makes the system more suitable for daily routine type situations rather than cases in which pedestrians are walking around without a proper goal. However, most situations can be described as having a deadline (e.g. eventually, most people have to go to bed), so this assumption is not necessarily very restrictive.
Furthermore, we assume that the behavior of the pedestrians can be described as finite state automata, and that these automata incorporate base states to which transitions loop back, and from which it is always possible to reach the goal state (given enough time is left). We need this assumption in order to create an efficient way of determining which actions can be done before the deadline. Otherwise we would have to search through the finite state automaton in order to find the various ways actions can be tied together.

3.5 Preparations

Because the proposed system needs to run real-time, we refrain from using systems that are overly complex.

Elaborate on "overly complex"

However, it is possible to do some computations before running the simulation, and save this for use during the simulation. An important application is the calculation of the distance in time between all the places in a Petri net and the goal place. This distance can easily be computed using the *Dijkstra shortest path algorithm* [5]. Dijkstra's algorithm is a graph search algorithm that can produce a shortest path tree for a single source, for a graph with nonnegative edges. In our system we can use this to compute the time from any place to the goal place (the source). This can be very useful when we would like to compute an estimate of how long a pedestrian will be stuck to the behavior of a certain situation. However, it will never be more than an estimate, since it is possible to design Petri nets with (possibly) infinite loops. But since the Petri nets are probabillistic, it will never be possible to give an exact prediction of the time it takes to execute a certain behavior.

- Write about problems with dijksta and petri nets, such as multiple tokens/slots, and colored tokens etc.

4 Experiments

The following is only a rough draft

It is very difficult to determine whether one has been succesful at modeling lifelike behavior. What we are going to do is study footage of real-life behavior at Rotterdam Airport, and pick out various behavioral patterns. Then, we are going to model these patterns with our system.

4.1 The Dataset

We acquired manually annotated data indicating the tracks of the visitors of Rotterdam airport.

Create reference to Ernst Bovenkamp

.

References

[1] Wei Shao A and Demetri Terzopoulos B. Autonomous pedestrians.

[2] O. Burchan Bayazit, Jyh ming Lien, and Nancy M. Amato. Better group behaviors in complex environments using global roadmaps. In *In Artif. Life*, pages 362–370, 2002.

[3] P. Becheiraz and D. Thalmann. A model of nonverbal communication and interpersonal relationship between virtual actors. In *Proceedings of the Computer Animation*, CA '96, pages 58–, Washington, DC, USA, 1996. IEEE Computer Society.

[4] Adriana Braun, Soraia R. Musse, Luiz P. L. de Oliveira, and Bardo E. J. Bodmann. Modeling individual behaviors in crowd simulation. *Computer Animation and Social Agents, International Conference on*, 0:143, 2003.

[5] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959. 10.1007/BF01386390.

[6] Dirk Helbing, Illes Farkas, and Tamas Vicsek. Simulating Dynamical Features of Escape Panic. *Nature*, 407:487–490, September 2000.

[7] Terry R. Hostetler and Joseph K. Kearney. Strolling down the avenue with a few close friends. In *In Third Irish Workshop on Computer Graphics*, pages 7–14, 2002.

[8] Marcelo Kallmann, Etienne De Sevin, and Daniel Thalmann. Constructing virtual human life simulations. In *Proceedings of the Avatars2000 workshop*, pages 240–247, 2000.

[9] Marcelo Kallmann and Daniel Thalmann. Modeling objects for interaction tasks. In *Proc. Eurographics Workshop on Animation and Simulation*, pages 73–86, 1998.

[10] N. Pelechano, J. M. Allbeck, and N. I. Badler. Controlling individual agents in high-density crowd simulation. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '07, pages 99–108, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.

[11] Nuria Pelechano, Kevin O'brien, Barry Silverman, and Norman Badler. Crowd Simulation Incorporating Agent Psychological Models, Roles and Communication. In *1st Int'l Workshop on Crowd Simulation*, pages 21–30, 2005.

[12] Christopher Peters and Cathy Ennis. Modeling groups of plausible virtual pedestrians. *IEEE Computer Graphics and Applications*, 29:54–63, 2009.

[13] Craig W. Reynolds. Flocks, herds, and schools: A distributed behavioral model. In *Computer Graphics*, pages 25–34, 1987.

[14] Mankyu Sung, Michael Gleicher, and Stephen Chenney. Scalable behaviors for crowd simulation, 2004.

[15] Mak technologies.

[16] Shih-I Chang Tsai-Yen Li, Ying-Jiun Jeng. Simulating virtual human crowds with a leader-follower model. IEEE Computer Society, 2001.