AN INFORMATION THEORETICAL APPROACH TO CROWD SIMULATION

by

ÇAĞATAY TURKAY

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfillment of
the requirements for the degree of
Master of Science

Sabanci University

June 2009

AN INFORMATION THEORETICAL APPROACH TO
CROWD SIMULATION

APPROVED BY

Assist. Prof. Dr. Selim BALCISOY .....
(Thesis Supervisor)

Assoc. Prof. Dr. Mustafa ÜNEL .....

Prof. Dr. Kemal İNAN .....

Assoc. Prof. Dr. Özgür ERÇETİN .....

Assist. Prof. Dr. Yücel SAYGIN .....

DATE OF APPROVAL: .....

AN INFORMATION THEORETICAL APPROACH TO
CROWD SIMULATION

Çağatay Turkay

EECS, M.Sc. Thesis, 2009

Thesis Advisor: Asst. Prof. Selim Balcisoy

Keywords: Crowd Simulation, Information Theory, Automatic Camera Control, Behavioral Modeling

Abstract

Crowd constitutes a critical component in many virtual environment and entertainment applications. In this thesis, we propose methods to solve two distinct problems in crowd simulation domain; automatic camera control and adaptive behavioral modeling. As the basis of our methods, we develop a framework which uses information theoretical concepts to automatically construct analytical maps of crowd's locomotion, which are called behavior maps. The developed framework contains a probabilistic model of the scene to build behavior maps.

In the first part of this thesis, we propose a novel automatic camera control technique which utilizes behavior maps to find interest points which represent either characteristic behaviors of the crowd or novel events occurring in the scene. The camera is updated accordingly to display selected interest points.

In the second part of this thesis, we propose a novel behavioral model which uses behavior maps to control agents' behavior adaptively with agent-crowd interaction formulations. Our model can be integrated into crowd simulators and enhance their behavioral complexity. We made comparative analyses of the presented behavior model with measured crowd data and two agent-based crowd simulators.

KALABALIK SİMÜLASYONLARI İÇİN BİLİŞİM KURAMI TABANLI YAKLAŞIMLAR

Çağatay Turkay

EECS, Yüksek Lisans Tezi, 2009

Tez Danışmanı: Yar. Doç. Selim Balcisoy

Anahtar Kelimeler: Kalabalık Simülasyonu, Bilişim Kuramı, Otomatik kamera kontrolü, Davranış Modellemesi

Özet

Kalabalıklar, pek çok sanal ortam ve eğlence uygulamalarının önemli bir elemanıdır. Bu tezde, kalabalık simülasyonu kapsamında otomatik kamera kontrolü ve uyarlamalı davranışsal modelleme problemleri için çözümler önerilmiştir. Yöntemlerimizin temelinde, bilişim kuramı kavramlarını kullanan ve kalabalığın hareketlerinin analitik haritasını otomatik olarak yaratan bir çatı bulunmaktadır. Üretilen bu haritalara davranış haritaları adı verilmiştir ve bu haritaların üretilmesi için olasılık tabanlı bir model geliştirilmiştir.

Tezin ilk bölümünde, davranış haritalarına göre belirli ilgi noktaları bulan ve bu noktaları göstermek için güncellenen yeni bir otomatik kamera tekniği geliştirilmiştir. Bu ilgi noktaları, ya kalabalığın karakteristik davranışlarını ya da sahnede gerçekleşen orijinal olayları göstermektedir.

Tezin ikinci bölümünde, karakter bazlı kalabalık simülasyonları için kalabalıktaki karakterlerin davranışlarını kalabalık - karakter etkileşim formülasyonları ile tanımlamak için davranış haritalarını kullanan yeni bir davranış modeli önerilmiştir. Bu model, herhangi bir kalabalık simülatörüne eklenerek, bu simülatörün daha karmaşık davranışlar ortaya çıkarmasına imkan sağlamaktadr.

Acknowledgments

I would like to express my deep and sincere gratitude to my supervisor, Selim Balcısoy. His understanding and his encouraging, continuous support have provided a guidance for this thesis. Our long “paper sessions” are very valuable for me and I have learned a lot from these long studies.

I am honored to have Kemal İnan, Mustafa Ünel, Özgür Erçetin and Yücel Saygın as members of my thesis committee. I am grateful for their valuable review and comments on the thesis.

I would like to thank all my lab colleagues in Computer Graphics Laboratory. You made Sabanci University a better place to be and work. I specially thank Emre Koç for his efforts and cooperation in all the studies we made together. I would like to thank Selçuk Sümengen, Mustafa Tolga Eren, Ceren Kayalar, Uraz Türker, Berkay Kaya, İsmail Kasarcı and Kaan Yanç for their great friendship and support. Finally, thanks to Serdar Çakıcı for being the never-ending energy and charm of our laboratory.

I wish to thank my family for always loving and supporting me all the way. And I would also like to thank all my friends, especially from BARAKA, for all the good and bad times.

Finally, I owe my loving thanks to my very near future wife Burcu, she is making all this very beautiful.

Table of Contents

List of Figures

List of Tables

List of Abbreviations & Symbols

$\angle$	Angle between
$\| \quad \vec{x} \quad \|$	Magnitude of a vector
D	Kullback - Leibler Divergence
GHz	Gigahertz
GUI	Graphical User Interface
H	Entropy
KL	Kullback - Leibler
OpenGL	Open Graphics Library
pmf	Probability mass function
RVO	Reciprocal Velocity Obstacles
SLERP	Spherical linear interpolation

Chapter 1

INTRODUCTION

Crowd constitutes a critical component in many virtual environment and entertainment applications. Today it is common to have crowded virtual environments in massive multiplayer online games, crowd simulations and movie pre-visualizations. In order to increase the feeling of presence in a virtual environment, the environment should contain virtual crowds which must be simulated realistically and believably. In this thesis, we propose methods to solve two distinct issues in crowd simulation domain. First of these issues is the automatic camera control methods and second one is the adaptive behavioral modeling for crowd simulations.

The core element of our methods is a framework which uses information theoretical concepts to automatically construct analytical maps of crowd's locomotion. The framework includes a probabilistic model developed in order to use information theory quantitites, and the framework includes structures to produce analytical maps representing crowd's locomotion, which are called *behavior maps*.

Efficient camera control is essential to perform navigation and monitoring tasks in a virtual environment, therefore camera control has always been an interesting problem for the graphics community. A recent survey by Christie and Olivier [5] provides a comprehensive taxonomy of motivations and methods in camera control. Traditional camera control techniques based on user input, character follow-up or scripts do not provide camera control suitable for complex scenes with hundreds of animated characters. Hence, we need a tool which monitors the entire virtual environment, explores interest points and toggles the camera between them to improve user experience while exploring a crowded virtual environment. To aid users through navigational tasks in a crowded scene, an automated camera should build a cognitive model on where the user *would like to look at*. Such

an automated camera should provide sufficient information and insight about the scene being monitored. Our motivation is to find quantitative measures to determine where a user draws her attention in an animated crowded scene.

In order to improve a virtual environment's realism, crowds must be simulated believable in terms of their appearance and behavior. Recent advances in graphics hardware address the issue of photo-realistic rendering of crowds. However, due to the complex nature of human behavior, realistic behavior of agents in crowd simulations is still a challenging problem. Previous approaches either propose i) global solutions with high level formulations [41] - which can simulate large numbers of agents however not suitable for creating complexity in the crowd or ii) low-level scripted, complex agent-based methods - which are computationally expensive and requiring expertise and effort in the production phase [22]. In this study, we are proposing an analytical agent-based behavioral model that integrates global knowledge about crowd formation into local, agent-based behavior control. Principal elements of our behavioral model are;

- Analytical representations of crowd's activities, which are built by using a statistical model based on information theory.
- An agent definition responsive to behavior map values.
- Agent-crowd interaction formulations in order to control agents locally by using analytic crowd representation.

When integrated into an existing crowd simulator, we believe that our model creates a simulation with agents behaving in realistic, variable and complex manners, without the need for low-level scripting.

Our methods and models developed for crowd simulations can be integrated into existing applications which involve virtual crowds and they can provide valuable tools to enhance virtual environment applications. Our methods can make critical contributions in urban visualizations and urban design tools. In addition, they can be integrated into massive multiplayer games to increase the reality of the environment and to enhance user experience by providing automatic navigation tools.

1.1 Outline of the thesis

This thesis propose methods to solve two distinct problems in crowd simulation domain. Methods to produce analytical maps of crowd's activities are presented. These maps are used to develop an automatic camera control technique and adaptive behavioral modeling methods for crowd simulations.

The thesis continues with reviewing the literature in related fields. As two distinct problems are handled in this thesis, related studies are reviewed in two distinct categories. First part of Chapter 2 looks into automatic camera control studies performed in a number of different computer graphics related fields. This chapter finalizes with a detailed analysis of behavioral modeling approaches that have been proposed in the literature.

In Chapter 3, our crowd analysis framework is explained in detail. We begin by introducing information theory quantities that will be used in our methods. Secondly, we present our probabilistic model which uses agents in the crowd as random variables to perform information theory computations. Finally, we introduce the notion of behavior maps and give details on their construction and interpretations.

Automatic camera control technique based on interest points selected from behavior maps to aid navigation in a large crowded environment are covered in Chapter 4. This chapter first introduces the theoretical foundations of our studies on automatic camera control. We then present our camera control algorithm and develop techniques to produce an automatic camera for crowd simulations.

In Chapter 5, we present our behavioral model based on behavior maps for agent-based crowd simulations. We begin by proposing a generic agent representation to access behavior maps. Secondly, a set of agent-crowd interaction formulations are introduced and finally, we define certain analogies used in our behavioral model.

Chapter 6 presents the results obtained from both of the studies. Our automatic camera control technique is examined under certain scenarios and its performance is discussed. Our behavioral model is tested with a number of comparative scenarios concerning real-world data and two different crowd simulation systems.

Finally, Chapter 7 provides conclusive remarks on the studies and results. In this chapter, possible future study directions are discussed.

Chapter 2

RELATED WORK

Both of automatic camera control and behavior modeling for crowds fields involve extensive literatures. Therefore, we will review these fields separately.

2.1 Automatic Camera Control

Several aspects of camera control paradigm have been studied in the literature, we will try to review studies in which the expressiveness of the camera is investigated. There have been notable studies in manipulating the camera with respect to different user preferences. Blinn introduced an algebraic approach [4] to place certain objects at specified locations in the scene. Gleicher et al. proposed *through the lens* camera control [9], in which the user chooses feature points and their desired locations as seen from the lens of the camera. Due to the difficulty of the problem, there were attempts to put some constraints and perform higher level camera control. *The Virtual Cinematographer* by He et al. [10] proposed film idioms, each of which decodes cinematographic expertise and responsible for particular scene organizations. They organize these idioms in finite state machines to compose shots and transitions. All of these techniques require expert users or predefined constraints and not suitable for dynamic and crowded scenes.

A different group of researchers are interested in finding measures to evaluate the visual quality of the view and manipulate camera parameters to provide the best available shot [16, 1, 20]. Most of these algorithms focus on viewing a single object and aim to find the best view on a sphere around this object. Although the best view on a sphere is not directly applicable, the idea of finding a good view is relevant to our problem. In some of these studies, information theory based metrics have proven to be successful. The most

notable metric in this category is called *viewpoint entropy* proposed by Vázquez et al. [44] which expresses the amount of information in a selected view. They define their metric as the ratio of the projected area of each surface to the total area of all the surfaces projected to the view sphere. An extension of this work for time varying volumes is done by Ji et al. [15]. They find best views of a volume data in each frame by enhancing viewpoint entropy measure and do a smooth transition between good views as time evolves. A recent and interesting study by Kwon et al.[19] determines camera parameters for a single animated character. They proposed *motion area* which is the total area swept by the joints of the character projected onto the view plane. By maximizing this motion area, they achieve to display the motion of a single animated character effectively. One application where the camera is manipulated automatically to capture some events is done by Stoev et al. [38]. They developed an automatic camera control mechanism for visualizing historical data where the timing and location of events are pre-defined. They maximize both the projected area and the normalized depth of the scene to select a good view as camera moves between pre-defined locations.

2.2 Behavioral Modeling for crowds

An overall idea of the challenges and improvements in crowd simulation can be obtained in [40]. There are several behavioral models proposed in the literature and a survey by [17] covers most of these studies. There have been many studies on agent-based crowd models to create human-like behaviors. Seminal works of Reynolds used behavioral models considering local rules [28] and create emergent flocking [27] behaviors. There is considerable work on agent-based crowd simulators incorporating psychological models and sociological factors. In [21], they model social group and crowd related behaviors. Their main focus is a layered framework to reflect the natural pattern of human-like decision making process. [29] tried to improve the quality of agent behavior by adding theories from psychology. In their work, they tried to produce more realistic collision avoidance responses. [22] developed virtual human agents with intentions, beliefs, knowledge and perception to create a realistic crowd behavior. In [25], they assigned psychological roles and communication skills to agents to produce diverse and realistic behaviors. In a more recent work, [24] created an improved model by using psychological and geometrical rules with a social and physical forces model. [12] proposed an adaptive crowd behavior

simulation, where he defines a static behavior context layer. When the behavior context is altered with a predefined event, the new context adaptively inhibits certain behavior in agents. However, this scheme is not suitable for dynamic environments. There are studies which model the virtual environment as maps to guide agents' behaviors. [33] modeled the environment with topological, perception and path maps to generate autonomous agents. [8] used adaptive roadmaps, which evolve with the dynamic nature of the environment. In [39], they assign situations and behaviors directly to environment rather than the agents themselves. The concept of behavior maps have been used in robotics and vision field. [7] defined behavior maps as encoding context information of the environment, and use these maps to autonomously navigate a robot on rough terrain. [3] used behavior maps to encode probabilities of moving in a certain direction on a specified location and used these maps to track trajectories of people and to detect anomalies in people's behaviors. In their study, they used expectation maximization algorithms to detect anomalies.

We integrated theories from behavioral modeling and borrowed ideas from studies representing the environment with guidance maps. To compute these maps, we employed quantities from information theory. Information theory have been introduced into computer graphics field by [44] which expresses the amount of information in a selected view. In a recent study, [42] used information theory based formulations to automatically control the virtual camera in a crowded environment.

Chapter 3

INFORMATION THEORY BASED CROWD ANALYSIS

In this section we will introduce the information theory framework which constitutes the core of our automatic camera control and behavioral modeling methods. We will begin by introducing the information theory quantities we have utilized in this framework, we will continue with proposing the probabilistic model developed in order to use information theory quantitites, and finally, we will explain how the proposed structures are used to produce analytical maps representing crowd's activities, which are called *behavior maps*. An overall figure displaying our information theory framework can be seen in Figure 3.1.

3.1 Information Theory Quantities

Information theory deals with quantification of information. It has been used in a wide range of areas such as computer science, physics, biology and natural language processing. The key measure in information theory, *information entropy*, which defines our current understanding of information, is proposed by Shannon [32]. Let X be discrete random variable which takes values from set χ with probability distribution $p(x) = Pr[X = x], x \in \chi$. Entropy, $H(x)$ of random variable X can be defined by:

$$H(x) = -\sum_{x \in \chi} p(x) \log p(x) \tag{3.1}$$

Entropy is a measure of uncertainty of a random variable. It provides us with an insight about how likely a system produces diverse outcomes. Namely, a system with low entropy tends to yield same outcomes in successive tries.

Another critical concept for our measurements is $Kullback-Leibler\ divergence$ (KL) [18]. Take two probability mass functions (*pmf*) $p(x)$ and $q(x)$, divergence between *pmf*'s

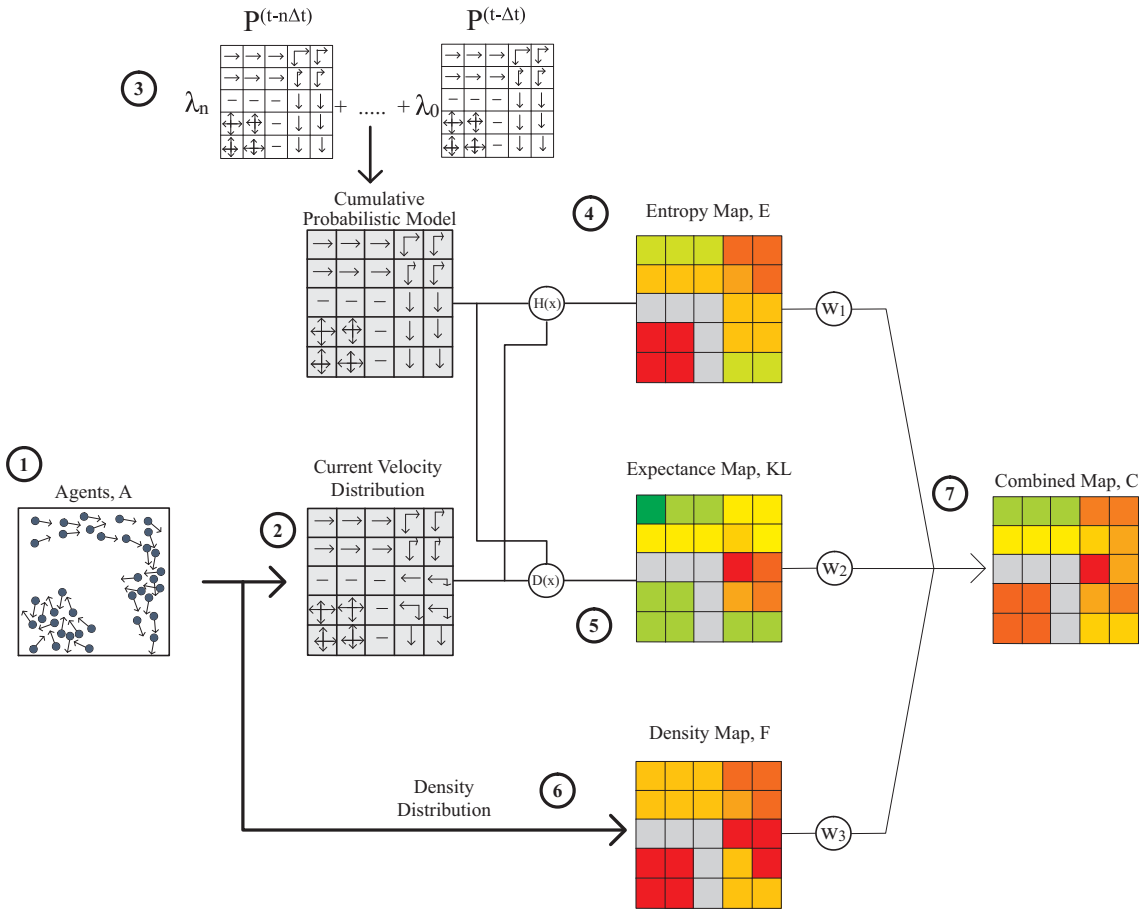

Figure 3.1: Behavior map construction. 1) List of agents is extracted by our model from the crowd simulator. 2) Activities of the crowd are mapped to the underlying grid to form the current distribution function of the activities of the crowd 3) Older distributions are merged with a temporal filter. 4) *Entropy map* of the scene is built by calculations on merged *pmf*'s from ($t - \Delta n$ to t). 5) *Expectance map* is formed by calculating KL divergence between the probabilistic model and the current distribution. 6) *Density map* is formed by calculating the current densities on a specific cell. 7) Behavior maps are blended with user-defined weights to construct combined maps.

$p(x)$ and $q(x)$ is given by:

$$D(p\|q) = -\sum_{x\in\chi} p(x)\log\frac{p(x)}{q(x)} \tag{3.2}$$

which is a non-symmetric metric expressing the difference between two probability distributions. Given the *true distribution* $p(x)$ of data, KL measures the loss of information if we use $q(x)$ instead of $p(x)$ while coding a sample. For further reading on information theory, please refer to [6].

3.2 Probabilistic Model

In this section, we introduce a probabilistic model where both spatial and temporal dimensions of crowd's activities are taken into consideration. Let $A = \{a_1, a_2, ..., a_n\}$ be the set of agents present in a simulation, where a_i represents a single agent. Physical properties of an agent can be described as $a_i = \{u, \vec{v} \ : u, \vec{v} \in \mathbb{R}^2\}$ where u defines the position and $\vec{v}$ defines the velocity of agent a_i. All the agents' movements are projected onto the same plane and the calculations are done on a 2D map, so both of these vectors are in $\mathbb{R}^2$. We classify the activity of an agent by: the location the agent is on, the direction of the agent's movement and the speed the agent is moving with. Different *pmf* structures are used to capture these characteristics. *Pmf*'s for direction and speed values and how this values are mapped into the corresponding *pmf*'s are explained below ;

- $P_{\hat{x}}(x) = \Pr(X = x), x \in \{0, 1, .., n\}$

  Values of random variable X in this *pmf* is found by quantizing the normalized velocity vector $\hat{\vec{v}}$ (belonging to an agent a) into one of n categories. $\hat{\vec{v}}$ is categorized by function;

  $$q_1(\vec{v}) = \{ \left\lfloor \hat{\vec{v}} \angle \langle 1, 0 \rangle / (2\pi/n) \right\rfloor : n \in \mathbb{N},\ 0 < n \leq 2\pi \} \tag{3.3}$$

  which finds the angle between $\hat{\vec{v}}$ and $\langle 1, 0 \rangle$ in a 2D Cartesian coordinate system and finding which interval this angle is in. The value of n effects the quantization resolution.

- $P_{\|\vec{x}\|}(x) = \Pr(X = x), x \in \{0, 1, .., n\}$

  Assuming that $\| \vec{v} \|$ is in the range $[a, b]$, i.e. the agents move with a speed in $[a, b]$, function

  $$q_2(\vec{v}) = \begin{cases} 0 & \text{if } \| \vec{v} \| < a \\ \lfloor \| \vec{v} \| / m \rfloor & \text{if } a \leq \| \vec{v} \| < b \\ n & \text{if } b \leq \| \vec{v} \| \end{cases} \tag{3.4}$$

  calculates which value will the random variable X will take depending on the magnitude of velocity vector. The n value in the above definition is dependent on the values of a, b and m. If the range $[a, b]$ is large, n can be made lower by quantizing this range with m.

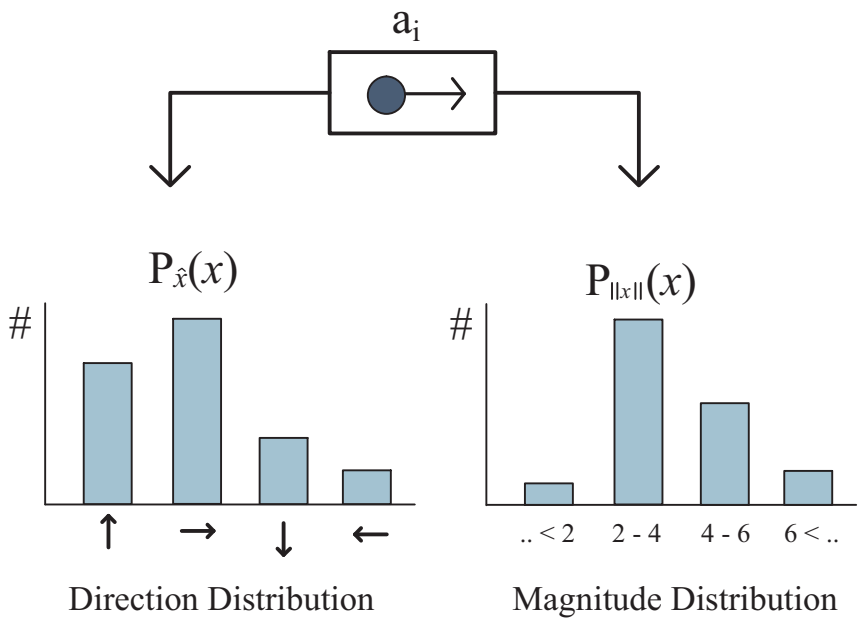

Figure 3.2: Two types of *pmf*s used in our model. Notice that $n = 4$ for both of the distributions.

The above *pmf*s are illustrated in Figure 3.2. We merge these two *pmf*s into a single *pmf*, $P_{\vec{v}}$, with a user defined constant α, which distributes importance to direction or speed distributions, as:

$$P_{\vec{v}} = \alpha P_{\hat{\vec{v}}} + (1 - \alpha) P_{\|\vec{v}\|} \tag{3.5}$$

This combination provides the user with a degree of flexibility to choose which of these distributions to put emphasis on. As $P_{\vec{v}}$ is taking samples over a period of time, a Gaussian shaped filter is applied to control the importance given to temporally cumulated distributions. Let t_1 and t_2 be two time steps where $t_2 - t_1 = n\Delta t$ and $n \in \mathbb{N}^*$, temporal filter is applied as;

$$P_{\vec{v}}^{\,t_1 \to t_2} = \lambda_0 P_{\vec{v}}^{\,t_2} + \lambda_1 P_{\vec{v}}^{\,t_2 - \Delta t} + \ldots + \lambda_n P_{\vec{v}}^{\,t_2 - n\Delta t} \tag{3.6}$$

$$\lambda_n = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(n-\mu)^2}{\sigma^2}},\ \mu = 0 \tag{3.7}$$

where, n is defined as *historical depth* defining the maximum *age* to consider, while *age* meaning the time passed from the moment the distribution have occurred. Δt defines the time interval between two adjacent frames. The λ constants are aging coefficients and they are calculated by using Gaussian distribution function (3.7) with $\mu = 0$. These values can be interpreted as a Gaussian filter applied in temporal domain. By changing

the variance of the distribution function (i.e. by changing σ^2), importance given to older distributions are manipulated. Choosing a lower variance gives less importance to older distribution, making the model highly adaptable to current changes but leaving it more prune to noise. On the other hand, a higher variance creates a model that slowly evolves over time; i.e. only large changes have effect on the model immediately.

Having this temporal probabilistic model in hand, we need to extend our model to cover the spatial characteristics of activities. To accomplish this, a 2D grid G is placed on the scene. G contains w rows and h columns, where each cell is a square with side length l. The grid is adjusted to cover all the extent of the scene, so that every activity on the scene takes place inside this sampling grid. We combine the temporal model we have developed with this grid to end-up with a 2D map carrying temporal dimension. We define the state of the grid G at time t as,

$$G^t = \{g^t_{i,j}\ ;\ 0 \leq i < w,\ 0 \leq j < h\}$$
$$g^t = \{P_{\vec{v}}^{(t-n\Delta t)\rightarrow(t-\Delta t)},\ P_{\vec{v}}^{\ t},\}$$

Every cell, $g^t_{i,j}$ in grid G contains two *pmf*s; one extending back n time steps from time $t-1$, and the other characterizing the distribution at time t. With this definition, we categorize activities depending on their spatial characteristics. The spatial categorization process works by assigning the agent to the corresponding $g_{i,j}$. This spatial categorization finalizes our probabilistic model which takes both the spatial and temporal properties of activities into consideration. At each time step, an agent, a_i is assigned to a cell in grid G and agent's $\vec{v}_{vel}$ is transformed by q_1, q_2 given in equations 3.3 and 3.4, to be included as samples in probability distributions associated with $g_{i,j}$. In this manner, the probabilistic distributions are computed and evolve over time.

3.2.1 Behavior Maps

Behavior maps are analytical representations of crowd's activities which span over the whole virtual environment and monitor agents' locomotion during the simulation. A behavior map, B, is a 2D grid, consisting of w rows and h columns, where each cell is associated with the corresponding cell in G to access to the *pmf*s in this cell.

The information theory quantities, probability distribution functions and the temporal filter mechanism are utilized to construct the behavior maps we called as *entropy* and *expectance map*. In addition to these maps, we also build a density map showing the density of agents and finally, create a combined version of these maps to give user a control over behavior map construction.

3.2.1.1 Entropy Map

Entropy measures the uncertainty of a random variable. If locomotion of an agent is considered as the random variable, entropy values represent the magnitude of predictability of crowd's movements. Entropy values denote whether agents move independently or in a group. Locations with smaller entropy values denote where agents move with similar velocities. Conversely, locations with higher entropy values represent disorder in agents' locomotion. To build an entropy map, E, we begin by considering a random variable, $X_{i,j}$ (i,j indicating location on E), drawn according to *pmf* $(P_{\vec{v}}^{(t-n\Delta t)\rightarrow t})_{i,j}$. Then, E can be defined as;

$$E^t = \{H(X_{i,j}) \ : \ 0 \leq i < w, \ 0 \leq j < h\} \tag{3.8}$$

, where $H(X_{i,j})$ is the entropy of $X_{i,j}$ as defined in Equation 3.1. Figure 3.3 illustrates how agents' locomotion determine entropy map values. Notice that entropy values are lower in zones where crowd has similar locomotion.

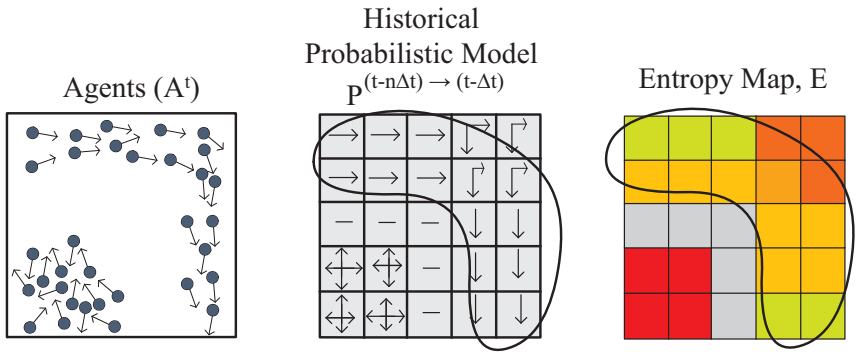

Figure 3.3: Crowd's movement and corresponding entropy map values. Selected zone indicates lower entropy values

3.2.1.2 Expectance Map

Probability distribution of crowd's activities defines the characteristics of locomotion that are likely to occur at specific locations. We define the distribution of crowd's locomotion from time $(t - n\Delta t)$ to $(t - \Delta t)$ by *pmf* $P_{\vec{v}}^{(t-n\Delta t)\rightarrow(t-\Delta t)}$ introduced in Equation 3.6 and the current distribution of crowd's locomotion at time t by $P_{\vec{v}}^{t}$. We use these two *pmf*s in Equation 3.2 to calculate KL divergence values. These values constitute the second type of behavior map called *expectance map*. Expectance map KL is defined as;

$$KL^t = \{(D(P_{\vec{v}}^{(t-n\Delta t)\rightarrow(t-\Delta t)} \| P_{\vec{v}}^{t}))_{i,j} : \ 0 \leq i < w, \ 0 \leq j < h\} \tag{3.9}$$

KL values indicate the difference between the current distribution and the cumulative distribution of crowd's locomotion. Use of KL divergence values to indicate *surprise* is proposed in [13], where they use KL divergence values to discover *surprising* events in video. They employed a principled approach to prove that KL is a powerful measure to represent *surprise*. We use KL values to indicate unexpected, surprising crowd formations. In an expectance map, cells with high KL values denote *surprising* activities taking place at those locations. At cells with lower KL values the state of the crowd remain as *expected*. Figure 3.4 displays that expectance values are high at locations where the current distribution is not "similar" to historical distribution.

3.2.1.3 Density Map

In addition to information theory based maps, a *density map*, F, is also included in our model. This map indicates how crowded a specific location is. In order to produce a measure that is less prune to noise, the temporal filter defined in 3.7 is also applied on F.

$$F^t = \{f_{i,j}^{(t-n\Delta t)\rightarrow(t-\Delta t)} \ : \ 0 \leq i < w, \ 0 \leq j < h\} \tag{3.10}$$

where f is a function giving the number of agents on location i, j between time steps $(t - n\Delta t)$ and $(t - \Delta t)$.

3.2.1.4 Combined Behavior Map

Each behavior map produced so far addresses different aspects in the activities of crowd and as a result, each map has certain effects on an agent's behavior. Therefore, agents should access all the maps and behave in response to all of them. We build a *combined*

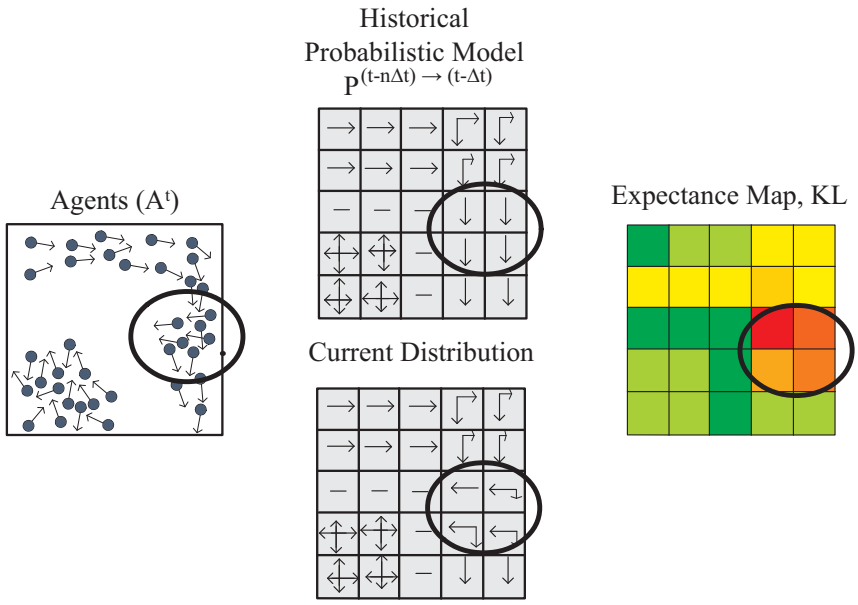

Figure 3.4: Crowd's movement, historical distribution, current distribution and corresponding expectance map values. Selected zone indicates unexpected event, where there is high KL values

behavior map which is a convex combination of entropy, expectance and density maps. This map can be formulated by;

$$C^t = \{w_1 * e^t_{i,j} + w_2 * kl^t_{i,j} + w_3 * f^t_{i,j} \\ : 0 \leq w_n < 1, w_1 + w_2 + w_3 = 1,\ 0 \leq j < h\} \tag{3.11}$$

, where each w_i represents user-defined weight values to determine the contribution of each map in the combined version.

Chapter 4

AUTOMATIC CAMERA CONTROL

In this study, we propose a novel automated camera control technique for large and crowded virtual environments on top of the scene analysis framework introduced in Chapter 3. This framework can be included into game engines or any virtual environment system to automatically aid camera control by using the behavior maps we have developed. These behavior maps give us quantitative answers to questions *"What are the characteristic behaviors of the crowd?"* and *"Where are the novel events happening in the scene?"*. Utilizing the calculated *entropy map*, camera makes a tour over zones which display characteristic behaviors of the crowd. And, in case of a novel event, by analyzing the *expectance map* camera moves to the location of this novel event and capture the moment of surprise.

4.1 Conceptual Foundations

The notion of *interest points* is very suitable for our camera control problem. We borrow the idea of interest point from computer vision domain. It is briefly "..any point in the image for which the signal changes two dimensionally." [30]. Our understanding of an interest point in this work have to be more extensive than this definition. Unlike a static image, a scene full of dynamic objects; or specifically, characters as in crowd simulation, carries both spatial and temporal characteristics. To define interest points in such a multi-dimensional domain, more comprehensive terms come into play, namely; $saliency$, $novelty$ and $surprise$.

Saliency and novelty are essential terms to understand how we perceive information and guide our attention while we are viewing visual images. A salient feature can be

briefly described as a spatial point standing out to be "different" then its surrounding [45]. Salient features have been shown to attract human attention by studies in neurophysiology and vision [14]. In other words, a salient point can be interpreted as, *where you would like to look at* in a visual image. But saliency alone is not adequate to answer this question on a temporally dynamic scene. Novelty complements saliency in temporal dimension and defines an event which have never occurred or occurs seldom as *novel* [36]. Novelty detection works as follows: a model of the system is formed as a basis by examining the behavior of the system over time. Having this base model in hand, current status of the system is evaluated and examined if any novel event is existent. Novelty detection can be interpreted as detecting salient features on temporal domain. Itti et.al combine these two complementary terms and come up with the notion of *surprise* [13]. They define surprise as the change in the observer's belief after the current status is observed. To calculate the surprise of a system modeled with distribution M, Kullback - Leibler divergence (3.2) between prior distribution $P(M)$ and posterior distribution $P(M|D)$ is measured after current data D is presented. They worked on video images to detect surprising points and proved that these points correlate with human viewer's eye movements.

4.2 Camera Control Methods

The entropy and expectance maps are utilized to control the camera. At each time step, an interest point is determined either from entropy or expectance map is chosen and the camera is toggled to display this interest point. The camera control algorithm is described in Appendix B. Figure 4.1 displays how interest points are selected to update camera accordingly.

Capturing unexpected events: In the first phase of the algorithm, τ_{kl}^t threshold value which is an adaptive threshold, is calculated. It is found by storing n last kl_{max} values, where n is the *historical depth* value we have mentioned before. Let μ_{kl}^t be the mean of these kl_{max} values, and σ_{kl}^t be the standard deviation, τ_{kl}^t is calculated by $\tau_{kl}^t = \mu_{kl}^t - \sigma_{kl}^t$. The maximum expectance value, kl_{max} is selected and compared with τ_{kl}^t. If the selected value is larger than this threshold, it is marked as an interest point, which can be interpreted as a *salient* location where there is a *novel* event.

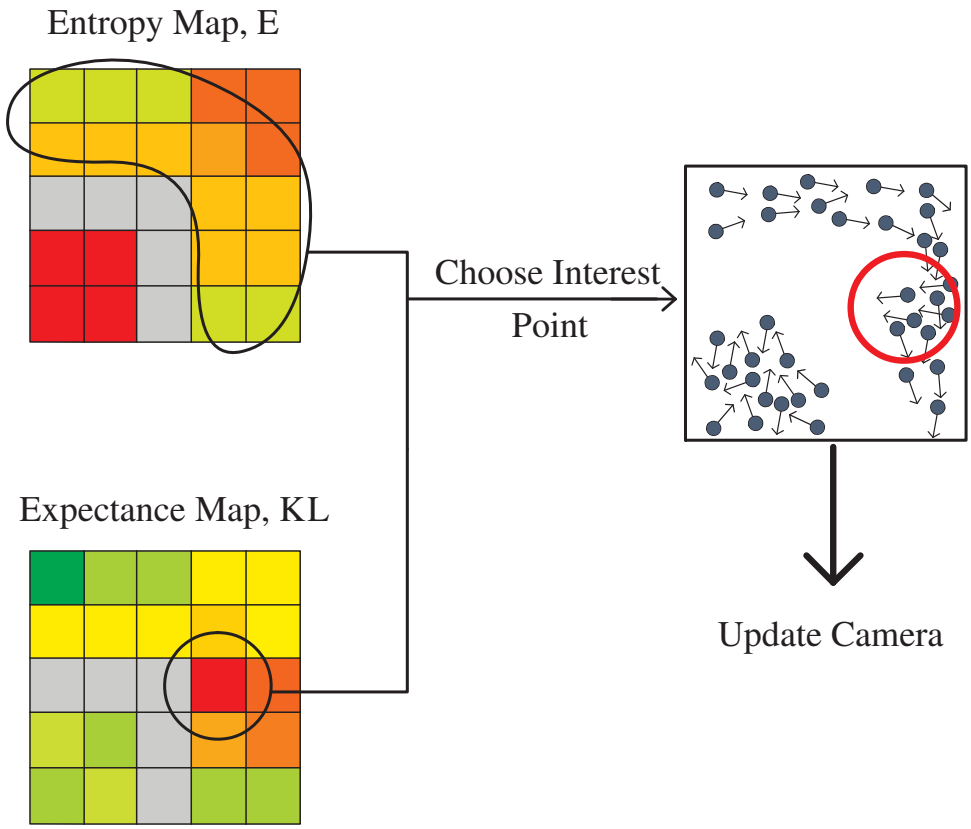

Figure 4.1: Interest point selection for camera control

Displaying characteristic behaviors of crowd: If there is no kl value marked as an *interest point*, attention can be drawn to locations where the characters moves more *together*, i.e. cells with lower entropy values. Under these conditions, camera makes a tour over low entropy zones, until some *novel* event occurs. To have a continuous tour over low entropy points, our method keeps track of the already visited points. At the beginning of the entropy tour, cell with the lowest entropy value is chosen and in each step of the entropy tour, camera starts to search unvisited zones in its neighborhood beginning with the direction of crowd movement. And entropy values are checked against the adaptive threshold value τ_e^t, which is also an adaptive threshold, calculated the same way as μ_{kl}^t, using $e_{i,j}$ values. Visited nodes are kept in a stack, in order to not to visit same zones again. Whenever a point from expectance map is chosen, the visited node stack is cleared to make camera ready for a new tour.

Camera placement: After one point of interest is computed, a good view to this point have to be calculated. We use a three-parameter camera model which represents the

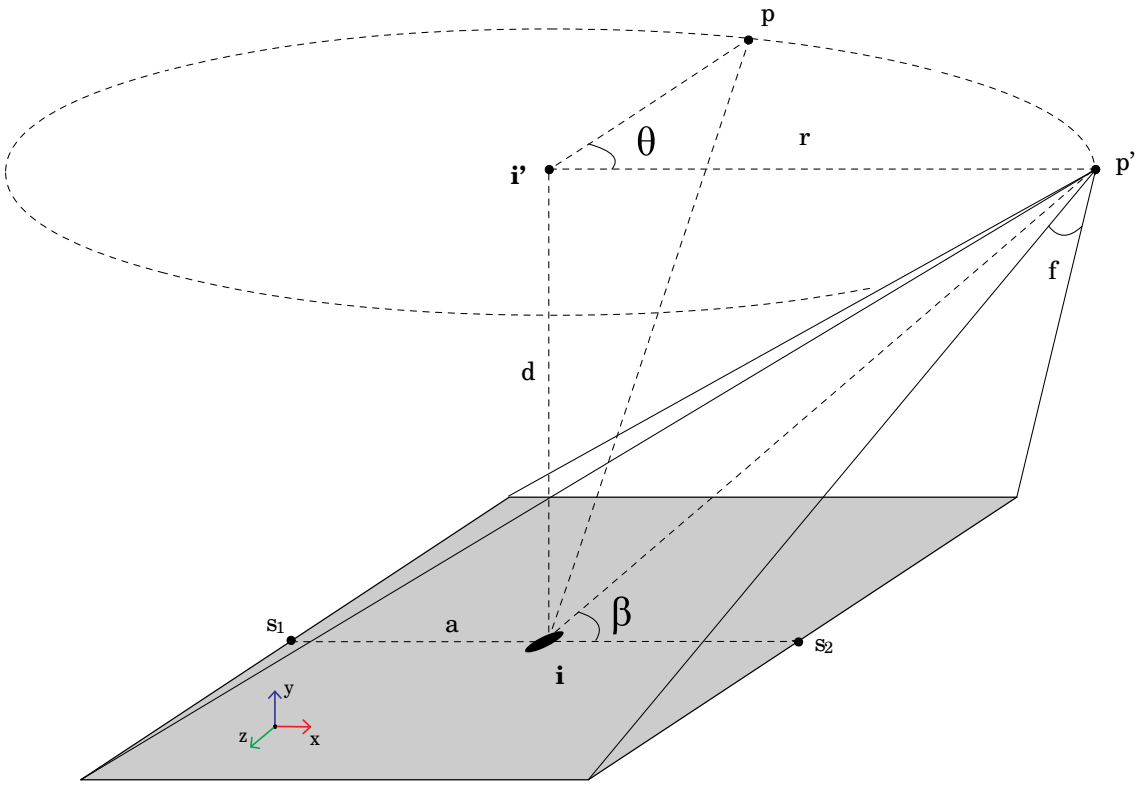

Figure 4.2: Given a fixed field of view f and viewing angle β, the camera should be placed appropriately to cover a square zone with sides $2a$ targeted at point $\vec{i}$. First $\vec{p'}$ is recovered by finding d and r geometrically. Final position $\vec{p}$ is found by incorporating pre-calculated θ angle

camera with its position $\vec{p}$, aim direction $\vec{l}$ and up-vector $\vec{u}$ where $\vec{p}, \vec{l}, \vec{u} \in \mathbb{R}^3$. The camera placement problem is shown in Figure 4.2. After p' is found , θ angle is calculated to make the camera look in the direction which is found to be most frequent direction of crowd movement in the underlying grid. Final position of the camera p is computed by rotating p' with θ degrees on the calculated circle. The second parameter of our camera, $\vec{l}$, is determined by using $\vec{i}$ and P. Finally, camera's up vector, $\vec{u}$, is adjusted properly that the camera never turns upside down through its interpolation. Using $\vec{l}$ and the current aim vector of the camera a quaternion q is built to interpolate the camera rotation using SLERP, proposed by Shoemake in [35]. While the camera is rotating, it moves from its current position to the calculated position p following a quadratic Bezier curve for smooth translation.

Chapter 5

BEHAVIORAL MODEL FOR CROWD SIMULATIONS

Interactions with a crowd are important psychological factors which determine how humans behave [2], however "agent-crowd" interactions are not considered by agent-based crowd simulators. In these simulators, an agent interacts with other agents and with the environment. In order to formulate agent-crowd interactions, an analytic representation which displays both of the spatial and temporal dynamics of crowd is required in our model.

Agent-based behavioral models use rule sets to mimic certain personality properties like aggressiveness, shyness etc. As stated in [34], personality structure can be static but its behavioral output changes greatly under specific circumstances. Therefore, an agent should reflect its personality differently under different conditions. Such a representation should contain intrinsic properties that are altered in response to dynamic and static simulation elements which should also contain a dynamic crowd representation. As agents' intrinsic properties are altered in response to the dynamic conditions, there should be formulations to determine agents' behavior accordingly to these internal changes.

Our proposed behavioral model is founded on *behavior maps* introduced in Chapter 3 which represent activities of the crowd. To utilize behavior maps, we borrow ideas from behavioral mapping techniques used in psychology research. These techniques involve place-centered maps, which keep track of behavior of individuals within a specific space and time. These maps display how and when a place is being populated [37]. The second element of our behavioral model is a generic agent representation which can access behavior maps and modify its intrinsic properties. We finally formulate how agents respond and behave according to their intrinsic properties and behavior maps within the limits of

the crowd simulator's capabilities. Consequently, we achieve agents behaving adaptive to current simulation conditions.

Beneath all this high level structure, we utilize a multi-agent navigation system to solve agent-agent and agent-environment interactions through collision detection and path planning algorithms. Our model can extend any existing agent-based crowd simulator.

Our model provides global knowledge on crowd's activities and enables the crowd simulator to incorporate agent-crowd interactions to modify agents' behavior. Behavior maps constitute the foundation of our model. They record and analytically represent crowd's activities. Second element of our model is a generic agent representation to access behavior maps. The final element in our model is a set of formulations to link the underlying crowd simulator with behavior maps. We customize the agent representation to fit into the current crowd simulator's features before developing these formulations. Prior to performing tests and using our model in crowd simulation scenarios, we define certain analogies between analytical maps, agent representation and agent-crowd interaction formulations. Figure 5.1 illustrates the overall structure of our model.

5.1 Agent Representation

Agent based crowd simulators have access to several motion engines and animation sets which define behavioral output types. These types can range from basic behaviors like changing direction, to complex behaviors like spreading shoulders to clear its path. The feature set of the crowd simulator and the underlying agent model define the complexity of agent behavior. In our behavioral model, we need a generic agent representation to fit into any type of agent based crowd engine. Our agent representation includes two properties, i) *behavior state* which enables interaction between agents and behavior maps and ii) *behavior constants* to determine agents' behaviors in combination with behavior state.

Behavior state, β, is the behavior map cell value assigned to an agent. Agents on the same cell of the map share the same behavior state. As behavior map values are altered temporally and spatially, these values are used in agent-crowd interaction formulations

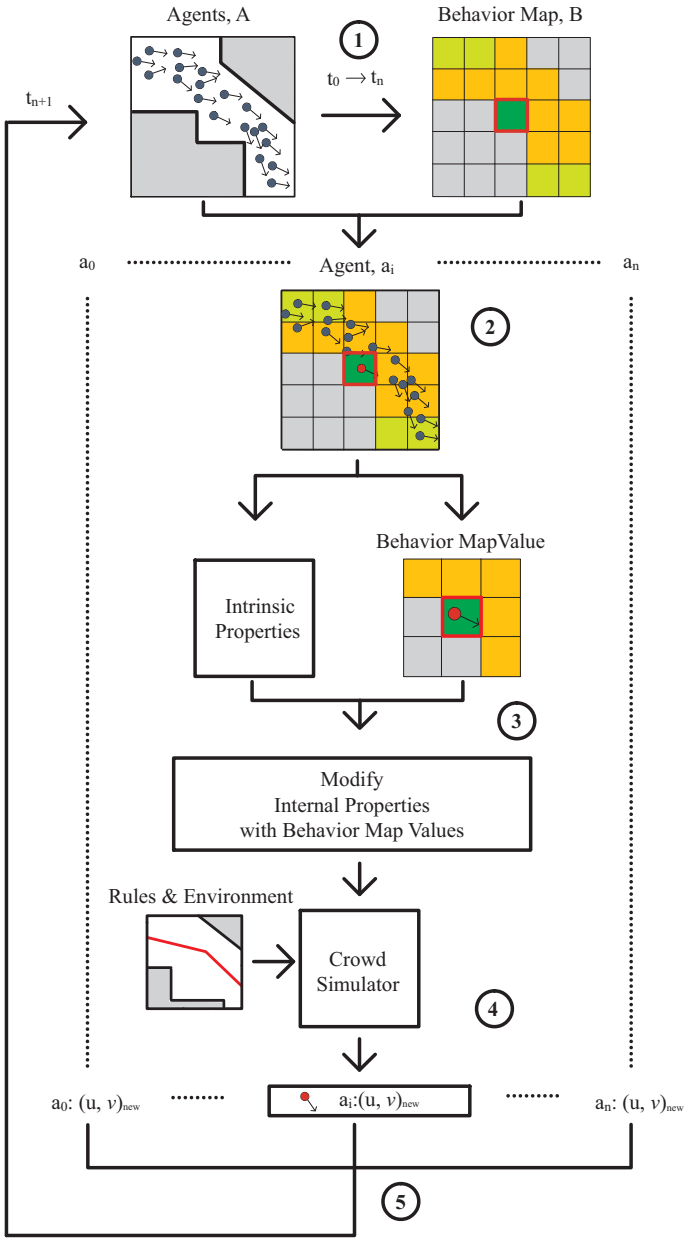

Figure 5.1: Overall structure of our model. 1)Locomotion of agents is extracted from crowd simulator to produce behavior maps. 2) Agents are assigned a specific cell value. 3) Agent's intrinsic properties are modified with behavior map value. 4) Agents are handled by crowd simulator to determine their physical properties. 5) Agent list is updated in the next time step

to adaptively control agents' behavior. Behavior constants, f, are agent specific values which are evaluated as personality attributes. Each feature of an agent which we want to control adaptively is paired with a behavior constant. By assigning an f value, behavioral complexity of an agent is extended and by varying f values, responses of agents to

behavior map values are varied. Behavior constants can be regarded as a mechanism to create complexity and variation in crowd. To wrap up these concepts with an example, assume a crowd simulator where agents have the feature of sweating, which we denote as p_0. In our representation, a behavior constant, f_0, defines how easy an agent sweats. And β values adaptively control when and where an agent will sweat. The agent representation is extended to include these properties, in addition to physical properties, which are position, u, and velocity, v:

$$a_i = \{u, \vec{v}, \beta, \langle f_0, p_0 \rangle, .., \langle f_n, p_n \rangle : \beta, f_n \in [0,1] \forall\ n\} \tag{5.1}$$

p_n is a symbolic representation to indicate a feature associated with a_i. A single $\langle f_n, p_n \rangle$ pair represents p_n is controlled by f_n. Notice that for each $\langle f_n, p_n \rangle$ pair, a formulation should be developed to define how β and f_n values control p_n.

5.2 Agent - Crowd Interactions

Our behavior model introduces agent-crowd interactions into agent based crowd simulators. In order to integrate our model, we first need to customize the agent definition given in Equation 5.1 according to the capabilities of the crowd simulator. This representation is then accompanied with formulations to define how agents handle behavior map values.

In this study, we use Reciprocal Velocity Obstacles (RVO) multi-agent navigation system introduced in [43]. We extended this system by implementing *composite agents* proposed in [11]. A composite agent, a_i, is a special agent equipped with a proxy agent, r_i, to model a number of emergent behaviors realistically. A proxy agent is a virtual agent, which is visible to all agents in the simulation except its parent a_i. r_i moves according to a_i's preferences. For example, if a_i wants to move in a certain direction, r_i is placed in that direction to clear a_i's path. With this mechanism a_i can display particular behaviors. The features, p_n, of the underlying simulation system can be listed as;

- d : Distance between proxy agent's position, $r_i[u]$, and a_i's position u. The longer the distance, the further a_i can proceed with less collisions.
- s : Radius of the circular area r_i occupies. The larger the area, the easier a_i can move.

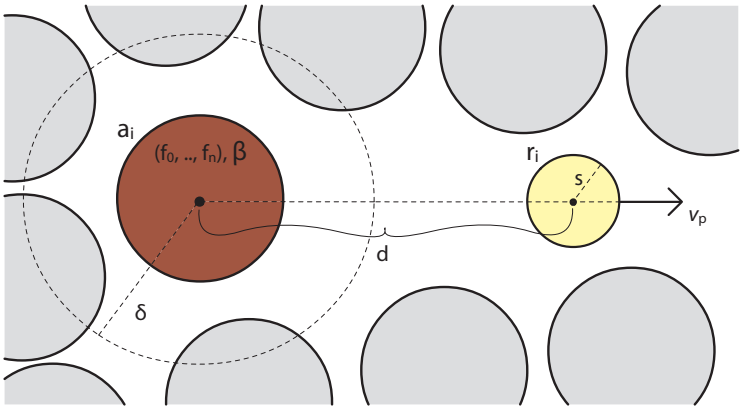

Figure 5.2: A composite agent a_i, its associated proxy agent r_i and certain features of agent representation

- $\vec{v_p}$: This is the *preferred velocity* of an agent a_i. It is the optimal velocity that would bring the agent to its goal. At each time step of the simulation, v_p is calculated with respect to agent's goal and then modified by the navigation system due to collision and path following constraints. We modify v_p's direction with a normalized velocity vector, $\vec{v_b}$, which is calculated with respect to behavior map values. $\vec{v_b}$ is calculated as a vector leading to lower entropy zones found as a result of a local search on behavior map.

- m : Indicates agent speed.

- δ : Indicates safety factor which is the range considered by an agent while calculating possible future collisions. With a high safety factor, an agent considers a higher number of possible collisions and behaves more careful. On the other hand, with a lower safety factor the agent becomes reckless and constitutes a higher possibility of making collisions.

After stating the features of the underlying simulator, we define customized version of the agent representation proposed in Equation 5.1:

$$\begin{aligned} a_i = \{&type, u, \vec{v}, r_i, \beta, \langle f_0, d\rangle, \langle f_1, s\rangle, \langle f_2, \vec{v_p}\rangle, \\ &\langle f_3, m\rangle, \langle f_4, \delta\rangle : \vec{v_p} \in \mathbb{R}^2; f_n, \beta, d, s \in \mathbb{R}\} \end{aligned} \tag{5.2}$$

where $type$ indicates whether the agent is a composite or proxy agent. Each f value with their associated feature is given as pairs. A figure to illustrate the customized agent definition can be seen in Figure 5.2. The next step is developing the formulations to include

behavior state, β, and behavior constants, f, values. The formualations are determined by considering the anologies related to behavior maps and by considering the requirements of the final simulation. We develop formulations to represent agent-crowd interactions for agent a_i as:

$$\begin{aligned} \beta &= kB_{i,j} \\ d &= \sqrt{f_0\beta}\, d_{max} + d_{min} \\ s &= \sqrt{f_1\beta}\, s_{max} + s_{min} \\ \vec{v}_p &= (\vec{v}_p^{\,o} + \widehat{\sqrt{f_2\beta}\, \vec{v}_b})(\sqrt{f_3\beta}\, m_{max} + m_{min}) \\ \delta &= \sqrt{f_4\beta}\, \delta_{max} + \delta_{min} \end{aligned} \tag{5.3}$$

where k is a constant to normalize β values, $B_{i,j}$ is the current behavior map value at cell $\{i,j\}$ and $\vec{v}_p^{\,o}$ is the optimal velocity leading to agent's goal. Each property has a user-defined min and max value to keep the values in a certain range. Certain features of agents with their associated f values and the effect of the formulations are illustrated in Figure 5.3.

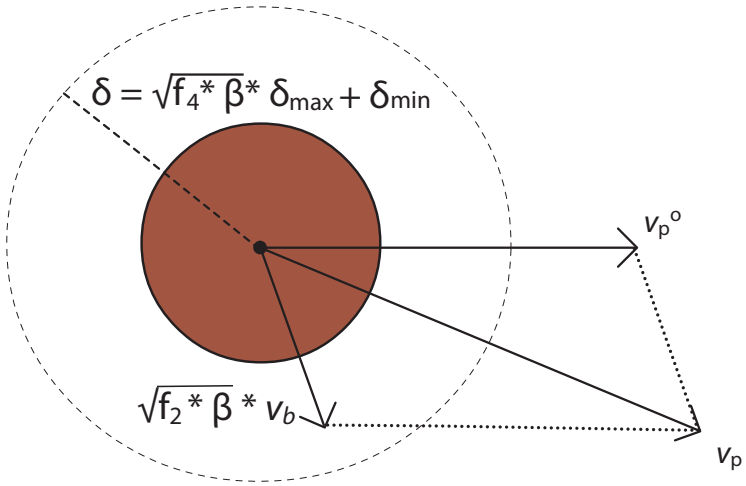

Figure 5.3: Effect of f and β values to their associated agent features

Static Maps In addition to the dynamic behavior maps computed automatically by analysis of activities of agents, our model also allows "temporally static behavior maps". These maps are user-defined maps, which can be utilized to increase the probability of certain behaviors in specific locations of a virtual environment. Designers can create

static maps, convert them into any type of behavior map and feed them into the simulation to effect how agents behave. These static maps can also be used to define certain events in the simulation. To illustrate, a static expectance map consisting of high values (high surprise level) can be toggled in a predefined time to create the effect of heavy rain which can be regarded as “unexpected”.

5.3 Analogies for Crowd Simulations

We define analogies between the interpretations of analytical maps with f values in order to produce realistic crowd simulations. We interpret the analytical maps of our model as seen in Table 5.1.

Table 5.1: Analytical maps and their interpretation

Analytical Map	Behavioral Interpretation
Entropy	Predictability
Expectance	Surprise
Density	Population

In the simulation, our agents can have aggressiveness and/or carefulness properties. To create certain agents which are aggressive and careful, we relate features of agents and formulations with f and β values. In Table 5.2, these behavior types with their related features and f values are listed.

Table 5.2: Behavior types, related features and f values associated with these features

Behavior Type	Feature	f
Carefulness	δ	f_4
Aggressiveness	$d, s, \vec{v}_p, m$	f_0, f_1, f_2, f_3

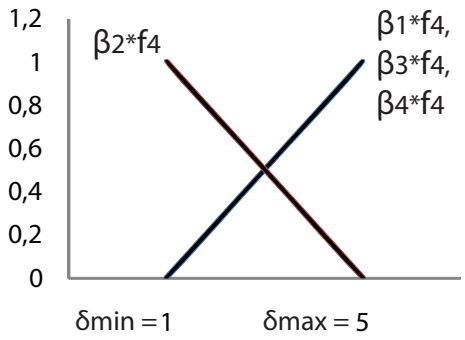

Figure 5.4: Carefulness is determined by δ and this chart shows the relation between β values, f_4 and the resulting δ values. Notice that β_1 (E), β_3 (F) and β_4 (C) values are proportional with δ, however β_2 (*KL*) values are inversely proportional with δ. δ_{min} and δ_{max} are user-defined values

The interpretations of behavior maps are used to define how agents respond to them. In areas with high entropy, where agents' locomotions are diverse, agents become more careful to avoid collisions, and they become more aggressive to get through these regions as quickly as possible. As the expectance map indicates the level of *surprise* in a specific location, aggressive agents do not panic and behave more goal-oriented by preserving their optimal velocity, $\vec{v}_p^o$, and enlarge s, d and m values in order to display their aggressiveness. On the other hand, high *KL* values make an agent less careful. Notice

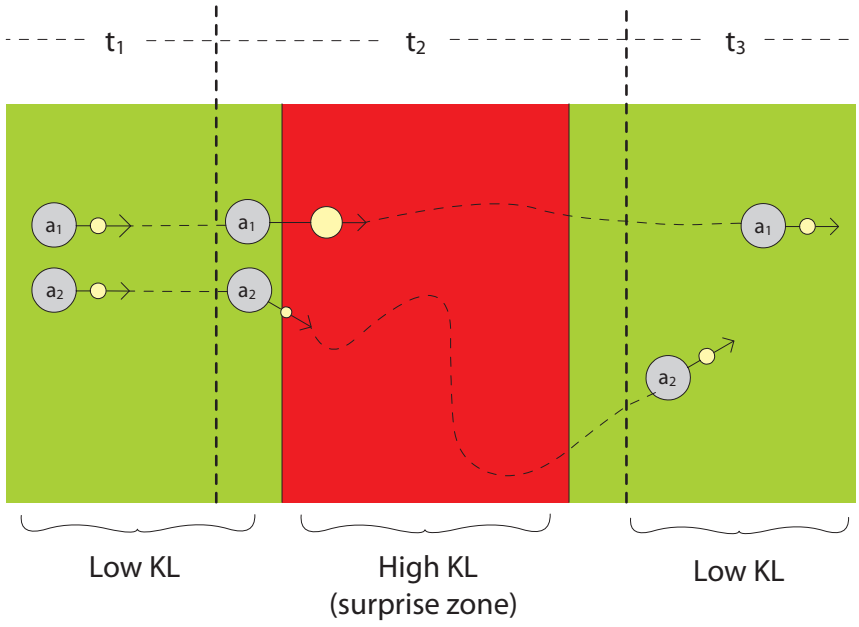

Figure 5.5: Responses of agents to expectance map

that, while carefulness is proportional to entropy values, it is inversely proportional to expectance values. Therefore, f_4 values should be chosen with respect to the behavior map. Figure 5.4 displays the relation between $f_4 * \beta$ values per behavior map type and δ values where δ values determine carefulness. Responses to density maps display how agents react to populated areas. Less aggressive agents avoid crowded places and their $\vec{v}_p$ is modified to lead them to less populated zones.

Figure 5.5 illustrates how agents respond to expectance map at micro level. In this figure, a_1 is an aggressive agent and a_2 is a calm agent. In time interval t_1, a_1 and a_2 behave identical. In t_2, they enter a high KL zone. a_1 responds by enlarging s and d values to keep its $\vec{v}_p$ as close as possible to optimal. However, a_2 mimics a panicking behavior and behaves in an unexpected manner. At t_3, agents return to their initial state. Notice that at the end of t_3, a_1 proceeds further.

Chapter 6

RESULTS

We run a number of tests to demonstrate our model's performance on a system with Intel QuadCore 2.8 GHz and Nvidia GeForce GTX-280. We run two different sets of tests for evaluating the performance of automatic camera control and behavioral model for crowd simulations. We begin with the tests for automatic camera control, followed by the set of tests to evaluate our behavioral modeling system. We implemented two different rendering platforms to visualize our results. One platform works on OpenGL with simple models and environment to provide easily observable results. The other platform works on DirectX and it provides a virtual environment with detailed models and a complex environment. This module enables us to evaluate our methods in a state of the art crowd rendering system.

6.1 Tests for Automatic Camera Control

We tested the effectiveness of the developed automatic camera control techniques on a number of different scenarios. We implemented a real-time crowd simulation environment using a modified version of OpenSteer library [26]. Our tests are grouped into two categories; showing the characteristic properties of the crowd and displaying novel events occurring in the simulation.

Displaying characteristic behaviors of crowd: In our first test scenario, crowd movement forms patterns over time while no unexpected event is occurring. Hence, expectance map contains low values below the adaptive threshold and our method chooses interest points among low values from the entropy map. Storing visited zones in a stack enables the camera to make a complete tour over the low entropy zones. It is seen in Figure.6.2-1

Figure 6.1: A sample screenshot from our test environment. Screenshot shows selected viewing angle.

that camera follows a path over low entropy zones which corresponds to locations where the crowd moves in an apparent pattern. The thresholding mechanism prevents the camera from considering vague patterns in the scene, thus visits to false positive zones are avoided.

Table 6.1: Expectance map values of a cell where a scripted unexpected event occurs at t_1. Value of σ^2 modifies temporal filter

	t_1	t_2	t_3
$\sigma^2 = 0,1$	0,292	0,046	0,021
$\sigma^2 = 1,0$	0,314	0,164	0,06
$\sigma^2 = 5,0$	0,306	0,245	0,167

Capturing unexpected events: As it can be seen in Figure 6.2-2, whenever there is a high value in expectance map, camera moves to that location immediately and retains its position until a new unexpected event occurs or the current interest point loses its importance over time. The duration, attention span, for the same event to remain interesting (to illustrate, duration between t_7–t_8 in Figure.6.2) is dependent on the temporal filter parameters we are applying in our model. If we set the temporal filter to give higher im-

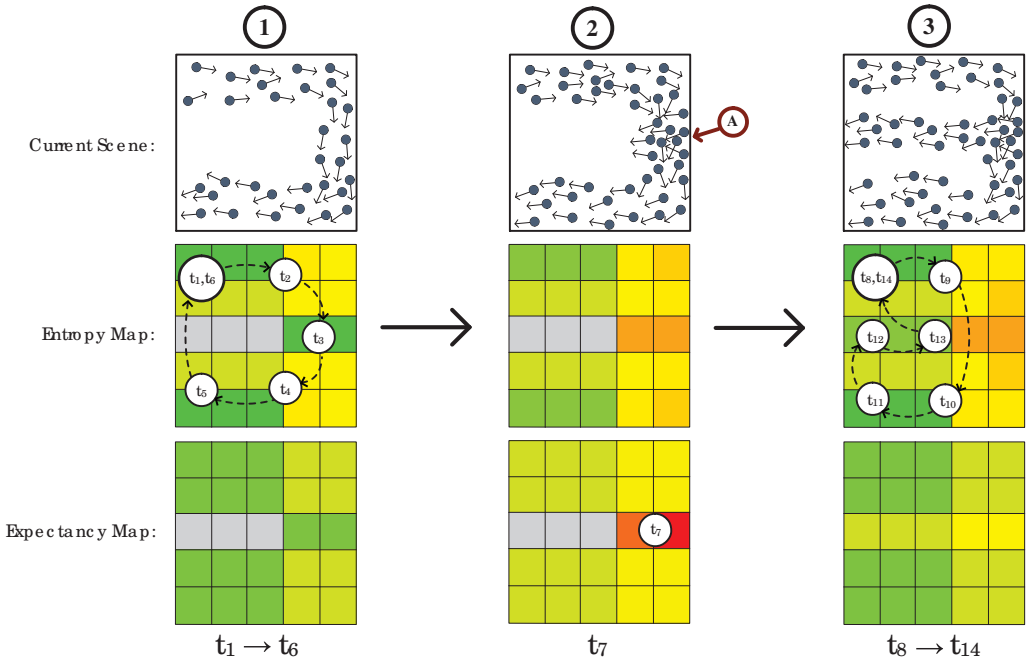

Figure 6.2: Example of moving camera with accompanying analysis maps from time t_1 to t_{14}. The circles represent visited points at the indicated time steps. 1) There is no unexpected event. Camera makes a tour over low entropy zones and after all the low entropy zones are visited, restarts the tour. This tour displays characteristic behaviors of the crowd. 2) At time t_7 number of characters enter the scene from point A and this is interpreted as an unexpected event and the camera immediately goes to the location of the event. 3) Between time steps t_7 and t_8 characters keep entering from A and this activity becomes a pattern in the scene, so the point is not interpreted as a surprising event anymore. The camera continues its tour over low entropy zones with an updated entropy map.

portance to past distributions, the attention span is longer as the unexpected event effects the underlying model slowly. In Table 6.1 we investigate KL values of the same interest point over a period of time for different σ^2 values of the temporal filter. Higher variance values creates a filter which also takes older distributions into account. The results show that with increasing variance, the corresponding KL values decrease more slowly. The variance of the temporal filter can be modified to suit the needs of the application. Figure 6.2-3 displays how the camera behaves after an unexpected event vanishes. As the stack for visited nodes is cleared at this instant, camera moves to the location with lowest entropy value to start a new tour.

Camera placement: To view the computed point properly, camera is placed to cover the entire area of interest. The direction of most dominant crowd motion at the inspected location is chosen as the view angle. In Figure 6.1, the selected viewing setting can be observed. The camera looks in the direction of character movement to give more insight about how the crowd behaves. This view selection mechanism can be accompanied with other metrics which can be user defined entities based on cinematographic concepts.

For different sampling grid resolutions, the behavior of our method varies. While smaller resolutions provide better analyses for capturing micro events, a higher resolution performs better for detecting macro events. If the size of a single cell of the sampling grid is large i.e. the resolution of the grid is low, a large number of activities are stored in a single cell, so micro events have minor effects on the overall distribution of a cell.

6.2 Tests for Behavior Modeling

Formulations in our behavioral model constitute of simple calculations, therefore we observed that integration of our model into a crowd simulator does not bring significant computational overload. The number of agents which can be simulated with our model is limited by the crowd simulator we use in our simulations. In our tests, we use combined maps introduced in Section 3.2.1.4. We observe that equal w_n values for each map performed successfully in most of the scenarios. However, weights of each behavior map can be adjusted according to the effect which a designer wants to create. Our model

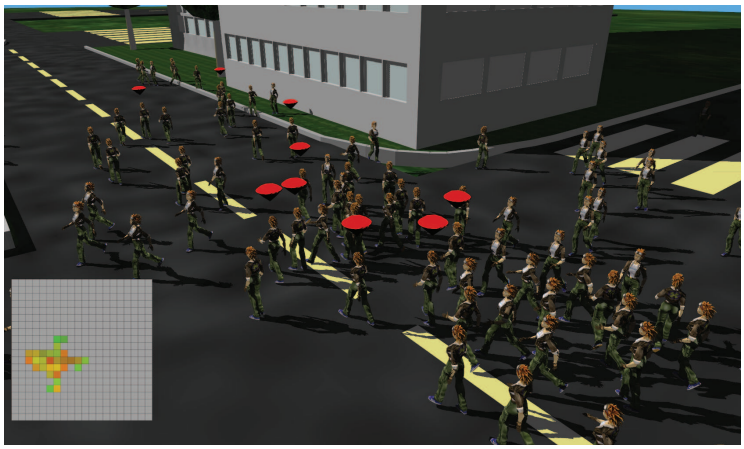

Figure 6.3: In this screenshot, red diamonds indicate aggressive agents

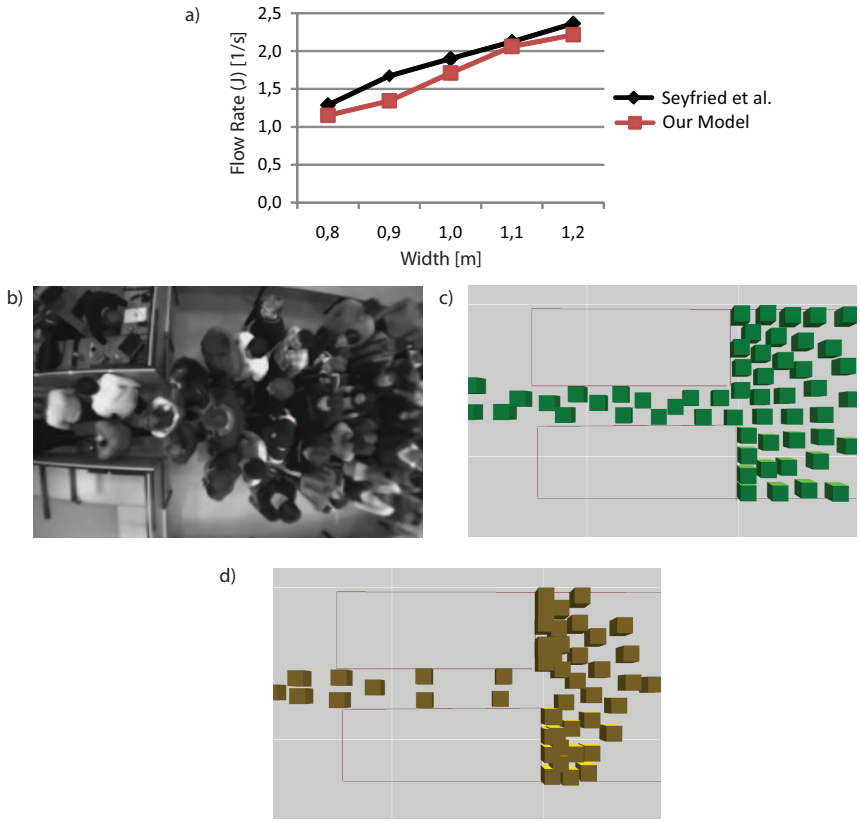

Figure 6.4: a) Chart showing flow vs. width of room exit b) Screenshot of a real-world scenario c) Screenshot from our test environment with less aggressive agents d)Clogging occurs when agents are more aggressive

and the underlying crowd simulator require a number of parameters to be set before performing a test. We build a GUI-based editor to interactively enter behavior constants and crowd simulator parameters. This authoring tool enables the designer to disperse f values over the agents to create variation in crowd interactively. The physical properties of the environment, goals and roadmaps are handled by the crowd simulator. Results of the following tests can be found in the accompanying video. A screenshot from our test environment can be seen in Figure 6.3.

Test - 1 We perform a test to prove the validity of our approach by a comparison with a real world scenario. We used room evacuation videos and data produced by [31] in Research Center Jülich, Germany and made available in [23]. These videos measure the flow of 60 students while evacuating a room with a variable exit width. We measure the flow of our agents with the formula $J = \Delta N/\Delta t$, where N is the number of agents and Δt

is calculated as the difference between the evacuation times of the first and the last agent. As the video incorporates students evacuating the room calmly, we set low aggressiveness to our agents. Screenshots from our test environment, the video and the resulting flow (J) vs. width of the room exit chart can be seen in Figure 6.4. We observe that our results are consistent with the real world case. We made further studies with this scenario setting and instead of adding calm agents, we add aggressive agents into the room. Agents are competing more to get out quickly in this case, as a result clogging occurred through the exit (Figure 6.4).

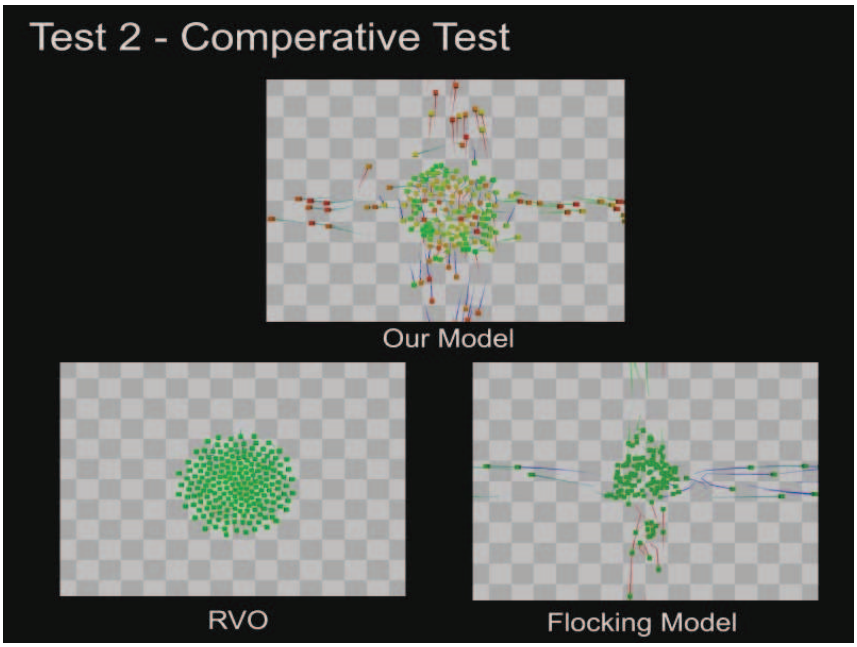

Figure 6.5: A comparative screenshot for RVO, Reynolds and our model.

Test - 2 We made comparison tests with two agent-based crowd simulators. The first one is the flocking model developed by Reynolds [26] and the second is the RVO library, which we also used as the underlying navigation library in this paper [43]. These comparative tests incorporate a scenario where four groups of agents walk through at a piazza. Throughout these test, we create a crowd with varied f values in our crowd simulator and this creates diversity in crowd's behaviors. In other models, agents do not respond to the dynamics of the crowd and behave identically.

Test - 3 We run the same scenario from Test - 2 incorporating a crowd consisting of i) only calm (not aggressive) agents ii) 10% aggressive agents and iii) agents with various

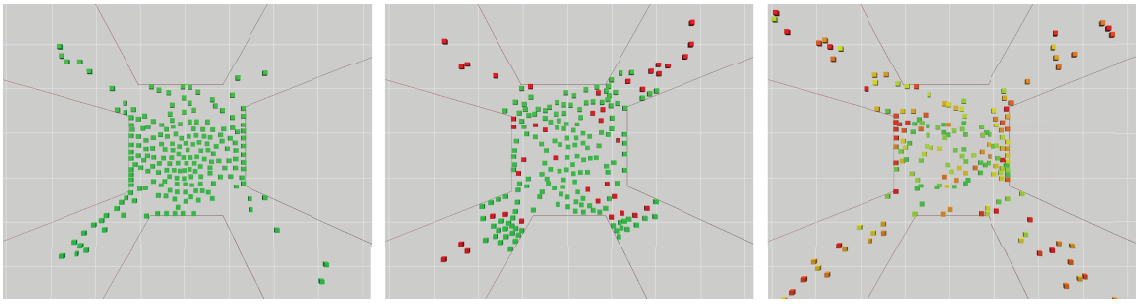

Figure 6.6: Our behavioral model increases agent diversity and complexity of crowd behavior(left to right: calm, few aggressive, diverse agents)

f values. Figure 6.6 displays the results of these tests. We see that only by varying the dispersion of f values, our model is capable of creating diverse and realistic results without requiring any additional scripting or editing effort.

Test - 4 We adopt a scenario where two groups of agents move towards each other. This scenario highlights the function of entropy maps. Before these groups meet, they do not display aggressive behavior as they produce a behavior map zone with low entropies. However, when these groups meet, there is a high level of disorder and entropy values increase. This variance in crowd formation adaptively modifies agents' responses and they start behaving aggressive.

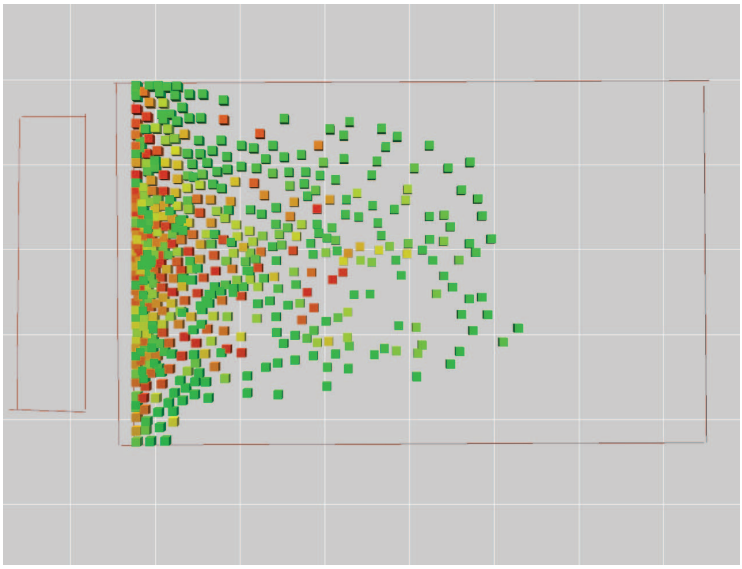

Figure 6.7: A screenshot from the concert scenario. Notice how aggressive agents proceeded to front rows and how calm agents avoided crowded areas.

Test - 5 To present the effects of density maps, a concert scenario is designed where all the agents' destination is the stage. Aggressive agents do not avoid crowded areas and their level of aggressiveness is proportional to density. On the other hand, less aggressive agents avoid crowded zones and stay away from the stage. After a period of time the front rows are packed with aggressive agents. This effect can be seen in Figure 6.7.

Chapter 7

CONCLUSION & FUTURE WORK

In this study, we proposed an automatic camera control approach and an adaptive behavioral modeling method for crowd simulations. As the core of these solutions, we developed a set of analytical maps, called behavior maps, which are produced by monitoring and analyzing the locomotion of agents in a virtual crowd. In order to build these maps, we first developed a probabilistic model to handle agent's locomotion as a random variable and use this random variable to construct analysis maps which keeps track of the crowd temporally and spatially. This probabilistic model is then utilized in the calculations based on information theory quantities namely, information entropy and Kullback - Leibler Divergence. As a result of these calculations, a set of behavior maps are constructed. These maps were then utilized in our methods for automatic camera control and behavioral modeling for crowd simulations.

As the first part of our studies, we have presented a novel automatic camera control technique for crowded scenes which monitors the entire scene and improves user experience. Our automatic camera control approach provides user two different tools: i) A tour over the crowded scene in which the characteristic behaviors of the crowd are displayed. ii) Monitoring of activities in the scene and capturing a location at the moment a novel event occurs. We tested our method in a crowd simulation environment to evaluate its performance under different scenarios.

Our method can easily be integrated into existent camera control modules in computer games, crowd simulations and movie pre-visualization applications. It provides some parameters like the resolution of the grid and the span of the temporal filter; which can be modified to adapt to the needs of the application into which our method is integrated. As

a future work, we will integrate certain cinematographic constraints into our automatic camera control approach to create a camera providing more visually pleasing results. However, due to their subjectiveness, cinematographic constraints are harder to model analytically.

As the second phase of our studies, we presented a novel analytical behavioral model which automatically builds behavior maps to control agents' behavior adaptively with agent-crowd interaction formulations. The presented behavioral model can be integrated into existing agent-based crowd simulators and improve the complexity of resulting crowd behavior. In most of the crowd simulators, low-level scripts are developed to drive complex agent behaviors. The analytical maps produced in our model are utilized to control these behaviors automatically. An important advantage of the proposed model lies in reducing the time spent on creating agents displaying diverse behaviors.

We did a comparative analyses of the presented behavior model with measured crowd data and two agent-based crowd simulators. We also run several well-known test scenarios to demonstrate the performance of our model.

As a future work, we will expand the scope of behavior map construction methods with different quantities from information theory and related fields. These maps can broaden our model with new interpretations and results. In this paper, we only integrated our model into agent-based simulators and used behavior maps to control individual agents. We will try to integrate our model into simulators which solve crowd simulations with global approaches [41]. We believe that our analytical maps will also provide information to control crowds globally.

Bibliography

[1] Tal Arbel and Frank P. Ferrie. Viewpoint selection by navigation through entropy maps. *Computer Vision, IEEE International Conference on*, 1:248, 1999.

[2] P.A. Bell. *Environmental Psychology*. Wadsworth Pub Co, 2001.

[3] Jérôme Berclaz, François Fleuret, and Pascal Fua. Multi-camera tracking and atypical motion detection with behavioral maps. In *ECCV '08: Proceedings of the 10th European Conference on Computer Vision*, pages 112–125, Berlin, Heidelberg, 2008. Springer-Verlag.

[4] Jim Blinn. Where am i? what am i looking at? *IEEE Computer Graphics and Applications*, 8(4):76–81, 1988.

[5] Marc Christie and Patrick Olivier. Camera control in computer graphics. In *Eurographics 2006 State of The Art Report*, 2006.

[6] Thomas M. Cover and Joy A. Thomas. *Elements of information theory*. Wiley-Interscience, New York, NY, USA, 1991.

[7] C. Dornhege and A. Kleiner. Behavior maps for online planning of obstacle negotiation and climbing on rough terrain. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 3005–3011, 2007.

[8] R. Gayle, A. Sud, E. Andersen, S.J. Guy, M.C. Lin, and D. Manocha. Interactive Navigation of Heterogeneous Agents Using Adaptive Roadmaps. *Visualization and Computer Graphics, IEEE Transactions on*, 15(1):34–48, 2009.

[9] Michael Gleicher and Andrew Witkin. Through-the-lens camera control. In *Computer Graphics (Proc. SIGGRAPH '92)*, volume 26, 1992. Proc. Siggraph '92.

[10] Li-Wei He, Michael F. Cohen, and David H. Salesin. The virtual cinematographer: a paradigm for automatic real-time camera control and directing. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 217–224, New York, NY, USA, 1996. ACM.

[11] Yeh Hengchin, Curtis Sean, Patil Sachin, van den Berg Jur, Manocha Dinesh, and Lin Ming. Composite agents. In *Symposium on Computer Animation - SCA'08*, 2008.

[12] X. Hu. Context-Dependent Adaptability in Crowd Behavior Simulation. In *Information Reuse and Integration, 2006 IEEE International Conference on*, pages 214–219, 2006.

[13] Laurent Itti and Pierre Baldi. A principled approach to detecting surprising events in video. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 631–637. IEEE Computer Society, 2005.

[14] Laurent Itti, Christof Koch, and Ernst Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(11):1254–1259, 1998.

[15] Guangfeng Ji and Han-Wei Shen. Dynamic view selection for time-varying volumes. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1109–1116, 2006.

[16] Tomihisa Kamada and Satoru Kawai. A simple method for computing general position in displaying three-dimensional objects. *Comput. Vision Graph. Image Process.*, 41(1):43–56, 1988.

[17] Z. Kasap and N. Magnenat-Thalmann. *Intelligent Virtual Humans with Autonomy and Personality : State-of-the-Art*, pages 43–84. Studies in Computational Intelligence, Springer, 2008.

[18] Solomon Kullback. *Information Theory and Statistics (Dover Books on Mathematics)*. Dover Publications, July 1997.

[19] Kwon, Ji-Yong, Lee, and In-Kwon. Determination of camera parameters for character motions using motion area. *The Visual Computer*, 24(7-9):475–483, July 2008.

[20] Chang H. Lee, Amitabh Varshney, and David W. Jacobs. Mesh saliency. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, pages 659–666, New York, NY, USA, 2005. ACM.

[21] Linbo Luo, Suiping Zhou, Wentong Cai, Malcolm Yoke Hean Low, Feng Tian, Yongwei Wang, Xian Xiao, and Dan Chen. Agent-based human behavior modeling for crowd simulation. *Comput. Animat. Virtual Worlds*, 19(3-4):271–281, 2008.

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

[23] Ped-Net, May 2009. `http://ped-net.org/`.

[24] 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*, pages 99–108. Eurographics Association, 2007.

[25] N. Pelechano, K. OBrien, B. Silverman, and N. Badler. Crowd simulation incorporating agent psychological models, roles and communication. In *First International Workshop on Crowd Simulation*, 2005.

[26] Craig Reynolds. Opensteer, steering behaviors for autonomous characters, 2004. Last Visited: 2008-11-01.

[27] C.W. Reynolds. Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 25–34, 1987.

[28] C.W. Reynolds. Steering behaviors for autonomous characters. In *Game Developers Conference*, 1999.

[29] S.J. Rymill and N.A. Dodgson. A Psychologically-Based Simulation of Human Behaviour. In *Theory and Practice of Computer Graphics*, pages 35–42, 2005.

[30] Cordelia Schmid, Roger Mohr, and Christian Bauckhage. Evaluation of interest point detectors. *Int. J. Comput. Vision*, 37(2):151–172, 2000.

[31] Armin Seyfried, Tobias Rupprecht, Oliver Passon, Bernhard Steffen, Wolfram Klingsch, and Maik Boltes. New insights into pedestrian flow through bottlenecks, 2007.

[32] C. E. Shannon. A mathematical theory of communication. *Bell Systems Technical Journal*, 27:623–656, 1948.

[33] W. Shao and D. Terzopoulos. Autonomous pedestrians. *Graphical Models*, 69(5-6):246–274, 2007.

[34] Yuichi Shoda. Computational modeling of personality as a dynamical system. *Handbook of Research Methods in Personality Psychology*, pages 633 – 652, 2007.

[35] Ken Shoemake. Animating rotation with quaternion curves. *SIGGRAPH Comput. Graph.*, 19(3):245–254, 1985.

[36] Sameer Singh and Markos Markou. An approach to novelty detection applied to the classification of image regions. *IEEE Transactions on Knowledge and Data Engineering*, 16(4):396–407, 2004.

[37] R. Sommer and B. Sommer. *A Practical Guide To Behavioral Research: Tools and Technique*. Oxford University Press, 5 edition, 2002.

[38] Stanislav L. Stoev and Wolfgang Straβer. A case study on automatic camera placement and motion for visualizing historical data. In *VIS '02: Proceedings of the conference on Visualization '02*, Washington, DC, USA, 2002. IEEE Computer Society.

[39] M. Sung, M. Gleicher, and S. Chenney. Scalable behaviors for crowd simulation. In *Computer Graphics Forum*, volume 23, pages 519–528, 2004.

[40] Daniel Thalmann and Soraia Raupp Musse. *Crowd Simulation*. Springer, 2007.

[41] Adrien Treuille, Seth Cooper, and Zoran Popović. Continuum crowds. *ACM Trans. Graph.*, 25(3):1160–1168, 2006.

[42] Cagatay Turkay, Emre Koc, and Selim Balcisoy. An information theoretic approach to camera control for crowded scenes. *Vis. Comput.*, 25(5-7):451–459, 2009.

[43] J. van den Berg, M. Lin, and D. Manocha. Reciprocal Velocity Obstacles for real-time multi-agent navigation. In *Robotics and Automation, 2008.*, pages 1928–1935, 2008.

[44] Pere-Pau Vázquez, Miquel Feixas, Mateu Sbert, and Wolfgang Heidrich. Viewpoint selection using viewpoint entropy. In *VMV '01: Proceedings of the Vision Modeling and Visualization Conference 2001*, pages 273–280. Aka GmbH, 2001.

[45] J. Wolfe. Visual search. In Harold E. Pashler, editor, *Attention*, chapter 1, pages 13–69. University College London Press, London, UK, 1998.

Appendix A

Class Diagram for Information Theory Framework

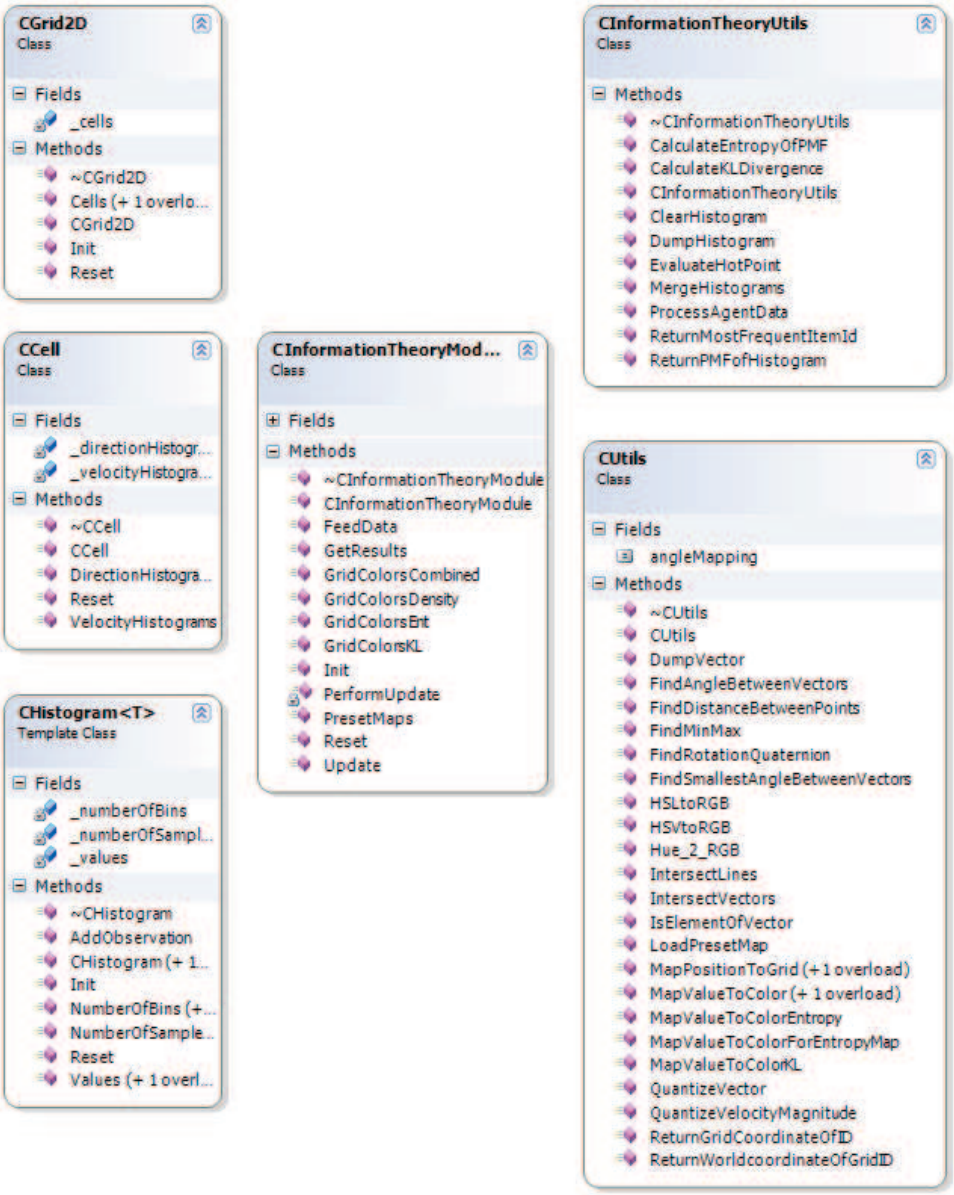

Figure A.1: Class diagram showing the most important classes, members and methods of information theory module.

Appendix B

Automatic Camera Control Algorithm

Algorithm 1 Automatic Camera Control

$t \leftarrow 0$
$inEntropyTour \leftarrow false$ // *to identify if we are in entropy tour*
Clear $tourStack$ // *to keep track of already visited points*
$\vec{ip} \leftarrow \langle 0, 0 \rangle$ // *init interest point*
loop
  for all $a_i \in A^t$ **do**
    Update G accordingly // *update current probabilities*
  end for
  Build KL and E maps
  $kl_{max} \leftarrow \max(kl_{i,j})$
  Calculate τ_{kl}^t
  if $\tau_{kl}^t < kl_{max}$ **then**
    $\vec{ip} \leftarrow \langle i, j \rangle$
    Clear $tourStack$
    $inEntropyTour \leftarrow false$
  else
    Calculate τ_e^t
    if inEntropyTour **then**
      for all $e_{i,j} \in neighborhood(\vec{ip}), e_{i,j} \notin tourStack$ **do**
        if $e_{i,j} < \tau_e^t$ **then**
          $\vec{ip} \leftarrow \langle i, j \rangle$
        end if
      end for
    else
      $\vec{ip} \leftarrow \langle e_{min}[i, j] \rangle$
      $inEntropyTour \leftarrow true$
    end if
  end if
  New camera position $\vec{p'}$ and orientation $\vec{q'}$ are calculated and start interpolation
  $t \leftarrow t + \Delta t$
  Update G accordingly // *add current prob. to history*
end loop

Appendix C

Quaternion Class And C++ Code For Slerp

```
class CQuaternion
{
public:
float X,Y,Z,W;
CQuaternion(void) : X(0),Y(0),Z(0),W(1) { }
CQuaternion(const float NewX,const float NewY,const float NewZ,const float NewW);
CQuaternion(CVector3 Axis,float Angle);
CQuaternion & operator () (const float NewX,const float NewY,const float NewZ);
CQuaternion & operator () (const float NewX,const float NewY,const float NewZ,const
float NewW);
CQuaternion & operator () (CQuaternion & Other);
CQuaternion & operator () (CVector3 Axis,float Angle);
CQuaternion & operator = (CQuaternion & Other);
CQuaternion & operator   ();// Conjuguate
CQuaternion & SetValues(float NewX,float NewY,float NewZ,float NewW);
bool operator == (CQuaternion & Other);
bool operator != (CQuaternion & Other);
CQuaternion operator - ();
CQuaternion operator + (CQuaternion & Other);
CQuaternion operator - (CQuaternion & Other);
CQuaternion operator * (CQuaternion & Other);
CQuaternion & operator += (CQuaternion & Other);
CQuaternion & operator -= (CQuaternion & Other);
CQuaternion & operator *= (CQuaternion & Other);
CQuaternion & operator /= (float & Scalar);
```

```
CQuaternion & operator *= (float & Scalar);
CQuaternion & SetEuler(float Yaw, float Pitch, float Roll);
CQuaternion & Normalize(void);
float GetLength (void);
CMatrix33 GetMatrix33(void);
} ;

CQuaternion Slerp(const CQuaternion & From, const CQuaternion & To, float Interpola-
tion)
{
CQuaternion Temp;
float omega, cosO, sinO;
float scale0, scale1;
cosO = DotProduct(From, To);
if (cosO ¡ 0.0)
{
cosO = -cosO;
Temp = -To;
}
else
{
Temp = CQuaternion(To);
}
if ((1.0 - cosO) > 1e − 6)
{
omega = (float)acos(cosO);
sinO = sinf(omega);
scale0 = sinf((1.0F - Interpolation) * omega) / sinO;
scale1 = sinf(Interpolation * omega) / sinO;
}
else
{
```

```
scale0 = 1.0F - Interpolation;
scale1 = Interpolation;
}
return From*scale0 + Temp*scale1 ;
}
```

Appendix D

Camera Control Implementations

In this piece of code; *position*, *direction* and *upVector* are the three vectors defining the state of our camera in the virtual world. *targetLeft* and *targetRight* parameters are the boundaries of the area that needs to be covered by our camera. *fovAngle* represents the field of view of the camera. *stareAngle* manipulates the height of the camera and *angleOnPositionCircle* determines the desired orientation of the camera.

```
void CCameraUtils::FindCameraVectors
( CVector3 *position, CVector3 *direction, CVector3 *upVector, CVector3 targetLeft,
CVector3 targetRight, float fovAngle, float stareAngle, float angleOnPositionCircle )
{
CVector3 targetPoint = (targetLeft + targetRight) / 2;
CVector3 temp1 = CUtils::IntersectVectors(targetPoint, stareAngle, targetLeft, stareAngle - fovAngle / 2);
CVector3 temp2 = CUtils::IntersectVectors(targetPoint, stareAngle, targetRight, stareAngle + fovAngle / 2);
// Take the higher of intersections
if (temp1.Y > temp2.Y)
{
*position = temp1;
}
else
{
*position = temp2;
}
// Find length of cam-to-target
```

```
temp1 = targetPoint - *position;
float camToTargetLength = temp1.Length();
float targetToUpIntersectLength = camToTargetLength / cosf(stareAngle);
CVector3 upVecPlaneIntersection;
upVecPlaneIntersection.X = targetPoint.X -
targetToUpIntersectLength * cosf(angleOnPositionCircle);
upVecPlaneIntersection.Z = targetPoint.Z -
targetToUpIntersectLength * sinf(angleOnPositionCircle);
upVecPlaneIntersection.Y = 0;
float positionRadius = abs(position->X - targetPoint.X);
position->X = targetPoint.X - positionRadius * cosf(angleOnPositionCircle);
position->Z = targetPoint.Z - positionRadius * sinf(angleOnPositionCircle);
*direction = targetPoint - *position;
*upVector = *position - upVecPlaneIntersection;
}
```