GPU-ACCELERATED CROWD SIMULATION WITH USER-GUIDANCE AND MULTI-LEVEL UNCERTAINTY

A Thesis

Presented to

the Faculty of the Department of Computer Science

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

By

Xiao Cheng

December 2013

GPU-ACCELERATED CROWD SIMULATION WITH USER-GUIDANCE AND MULTI-LEVEL UNCERTAINTY

Xiao Cheng

APPROVED:

Dr. Zhigang Deng, Chairman
Dept. of Computer Science

Dr. Guoning Chen
Dept. of Computer Science

Dr. Jingmei Qiu
Dept. of Mathematics

Dean, College of Natural Sciences and Mathematics

GPU-ACCELERATED CROWD SIMULATION WITH USER-GUIDANCE AND MULTI-LEVEL UNCERTAINTY

An Abstract of a Thesis

Presented to

the Faculty of the Department of Computer Science

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

By

Xiao Cheng

December 2013

Abstract

A long-standing rule for evaluating the realism and robustness of the crowd simulation is whether the simulations are capable of avoiding the congestion as well as maintaining free agent-wise collision. Although a few methods solved this problem in some specific case, no overall method has been explored. This paper proposes a new approach to simulate large-scale crowd behavior with real-world properties and without obvious artifacts. Specifically, we devised a lightweight user-interaction scheme to combine the navigation field and continuum crowd method. Also, the two-level uncertainty model is imposed onto the above crowd dynamic system to simulate non-goal driven behavior. All of the computations are accelerated by using GPU. Through our experiments and comparisons, we demonstrates the advantages of our approach.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Crowd Simulation

Over the past several decades, crowd simulation has been increasingly used in entertainment industry such as numerous blockbuster movies and video games. Besides, a realistic crowd simulation system either in macroscopic dynamic flow or in microscopic human behavior can find its important applications for urban planning, safety engineering, evacuation simulation, etc. For example, urban planners, as well as building designer, have started to use crowd simulation techniques to evaluate the safety aspect of buildings or communities as certain emergencies or disasters occur.

Agent-based approaches have been the most studied techniques for crowd simulation to date. In such models, each individual in real-world is modeled as an agent in the simulation environment; and each agent is an autonomous decision-maker.

Indeed, agent-based approaches for crowd simulation have achieved significant successes in recent years [30]. However, most of existing agent-based crowd approaches are based on a fundamental assumption that the navigations of all the agents are directed by certain implicit or explicit targets during the simulation. Few existing approaches have been focused on simulating the non-goal directed navigation behaviors of agents. In certain real-world scenarios (e.g., shopping), people's behaviors are not always homogeneous as goal-directed agents.

Furthermore, to the best of our knowledge, a unified framework has not yet been proposed to simultaneously simulate such heterogenous crowd phenomena, that is, seamless mixing of goal-directed agent navigation and non-goal directed navigation behaviors in the simulation. Also, since each agent in agent-based approaches needs to do decision-making and handle collision avoidance based on its local information at each simulation step, the performance of agent-based approaches is widely known for its inefficiency when the number of agents is quickly increased. Any efficient algorithms that can substantially speed up agent-based crowd simulation are potentially useful for many crowd simulation applications.

Quite a few decision-making rules were proposed since the birth of the basic model while most of these rules are based on local information. Due to the lack of global information, agent-based system is usually very performance-expensive and the path of agent is quite un-smooth. This gave the birth of the another type of crowd simulation system, which simulated each agent as a group. Agents are grouped together based on some rules or sharing some common properties: like destination, age or gender. It is the group simulation thought that shed light on the global path

planning, which overcame the deficiency of agent-based system. However, group-based system has its own problems like agent-wise artifacts as collision and unrealistic moving trajectory. Therefore, researchers began to incorporate preprocessed moving trajectory, from user-sketchy or movie footage, into the existing system to manipulate the agent both individually and group-wise.

Agent-wise collision avoidance is another topic along with crowd simulation. Most cases, it is positioned as the last step of the processing pipeline to sweep out the potential pair-wise collision due to the processing granularity of global path planning. Several collision avoidance approaches existed and usually it should be picked as the need of your system. It might cause a dramatic performance drop-down to use a over-complex scheme, which turned out to be incompatible with other part of the pipeline.

Despite a lot of crowd-simulation approaches have been proposed in the past a few years, little, if any, research has been done to model the crowd simulation beyond the above scope. In addition, due to the lack of in-depth study of extra model, simply using the global planning plus local collision avoidance are not robust enough to simulate in all major real-world scenarios like non-goal driven behavior.

1.2 The Contribution

Our approach is motivated by the observation that in a real-world scenario, people's behavior not always looks as homogeneous as goal-oriented agents. Instead, people in a realistic crowd simulation system should present goal-oriented group behavior while

allowing for the flexibility and individual variability of each agent. Besides, absolute non-goal-oriented behavior should be found for certain portion of the agents. Typical non-goal-oriented behaviors include visiting shop and wandering randomly in a mall. In short, heterogeneous crowd behavior is displayed through goal-oriented behavior with agent's variation and purely uncertain agent's behavior with no goal-oriented characteristic.

Inspired by the above research problem, in this paper, we propose a unified approach to simulate various heterogeneous crowd behaviors (e.g., simulating both goal-directed and non-goal directed navigation behaviors in a crowd), while being capable to robustly handle general-purpose crowd simulation problems such as local collision and pedestrian congestion avoidance. Our model has unified global planner and local planner to handle the advection of two agent types: *goal-directed* and *non-goal directed*. Specifically, we implement a GPU-based continuum crowd as the global planner, since it is suitable for medium-density crowds while avoiding macroscopic congestion with a look-ahead planning model. We use a spatial colonization system on GPU as the local planner, which preserve smooth trajectories despite the agents' decisions are made locally. We introduce an Eulerian uncertainty field to add realistic variations into macroscopic group behaviors and a Lagrangian uncertainty model to simulate non-goal-directed agents' behaviors. To resolve congestion in pedestrian traffic areas and simulate informal cultural convention, a user-guided, sub-domain navigation field is also incorporated into our approach.

To speed up the performance of our approach, we fully implement our approach on GPU and thus eliminate the data transfer bottleneck between CPU and GPU

that is needed in many current crowd simulation models. Finally, by integrating our uncertainty models with the flow-based continuum crowd model [43], we demonstrate our approach can simulate a variety of heterogenous crowds in different application scenarios, while providing flexible and intuitive user controls.

The main contributions of our approach are: (1) We introduce a new crowd uncertainty model that seamlessly combines Eulerian and Lagrangian uncertainties for heterogenous crowd simulation. Eulerian uncertainty and a simplified navigation field are computed off-line only once before the simulation starts. Lagrangian uncertainly, on the other hand, is computed at each time step, and it can be controlled by adjusting each agent's attributes. Our uncertainty model can be conveniently plugged into various existing crowd simulation systems and pipelines (e.g., continuum crowd model and navigation field). (2) The high efficiency of our approach, due to its full GPU-accelerated implementation, makes it particularly potentially useful for many real-time graphics and crowd simulation applications.

1.3 Structure of this thesis

The remainder of this paper is organized as follows. Chapter 2 briefly reviews some recent crowd simulation efforts relevant to this work. Chapter 3 details the methodology of our approach, including how to construct the uncertainty models and a sub-domain navigation field, and handle collision avoidance. Chapter 4 review the two global path planner and how our proposed two-level uncertainty model can be

cooperative with them. Chapter 5 describes the non-trivial details of our full GPU-based implementation and also some highlights of 3D character animation. Chapter 6 describes a variety of crowd simulation results by our approach as well as the comparisons between our approach and existing crowd simulation approaches. Lastly, discussion and conclusion remarks are given in Chapter 7.

Chapter 2

Related Works

In this section, we will briefly review recent efforts related to this work. However, comprehensively reviewing all the research efforts in crowd simulation is beyond the scope of this paper; interested readers are referred to the recent survey in this field [30].

Among numerous crowd simulation methods that are based local motion planning, probably, the most well-known approach is the social-force model proposed by Helbing et al. [12, 11]. On top of it, many extensions or variations have also been proposed [10, 20, 5, 2, 3]. However, the agent-based social force model has two limitations: unsmooth agent motion paths and the need of heavy computation. Contrary to the social force model, rule based models [34, 35] can produce more smoother motion paths; nevertheless, it can be effectively used only for small or medium density crowds. And, due to its local decision property, collision is difficult to be detected and processed if happens. One solution to collision avoidance is cellular automata

models [19, 18, 42]. This method discretizes the 2D simulation domain into regular grids, and ensure agents only can occupy free cells at simulation time. Its drawback is that the granularity of the simulation entirely relies on the domain discretization, resulting blocky agent trajectories. Some hybrid approaches such as the HiDAC model [29] consider the pros and cons of the above methods. Furthermore, the HiDAC model also consider other factors including sociological [25] and psychological effects [37, 31].

Another type of crowd simulation methods is based on global planning. Both Sud et al. [41] and Pettré et al. [33] use navigation graphs but they are quite different. The work of [41] introduces *MaNG*, constructed using first and second-order Voronoi diagrams to perform path planning. The work of [33] decomposes the multi-layer domain into navigation corridors through their proposed navigation graph. The work of [38, 21] interprets the simulation environment as 2D maps to obtain sufficient global information for path finding. In addition, approaches such as goal-driven navigation fields to direct crowds [28] is another direction of global planning.

Continuum crowd methods [14, 43] formulate crowd simulation as an optimization problem that considers both local and global planning. These methods are quite effective in solving macroscopic congestion among different groups of people. However, it requires a per-time-step re-computation of potential fields for each group. Hence, as the number of groups is increased, the runtime performance is expected to slow down dramatically. As discussed in the work of [43], the original continuum crowd method is not well suited for dense crowd simulation, since small granularity agent-wise collision is difficult to be resolved within the continuum optimization

framework. Similar to the continuum crowd optimization solution, the work of [28] employs anisotropic cost functions to attain smooth potential fields and thus alleviates resulting blocky motion paths, assuming the 2D grid has a lower resolution support. Another major difference between this method and the continuum crowd is that it only needs to compute the navigation field at the initialization stage.

A number of efforts have been specifically focused on simulating very dense crowds. For example, Narain et al. [26] use the Lagrangian and Eulerian method for each agent and use density-dependent incompressibility to model the dynamics of human crowds. Golas et al. [6] mention that their approach only relies on purely local information and thus cannot plan around congestion at a larger distance.

Local Collision Avoidance (LCA) is a critical step in any crowd simulation systems. Actually, a wide range of LCA algorithms have been developed over the years. Pairwise repulsion force method used in the work of [43, 26] is one of the most intuitive ways to avoid local collision between agents. However, its computational cost can quickly become a performance bottleneck when the number of agents is increased. Currently, one of state-of-the-art methods is to treat possible collisions as obstacles in velocity space [45, 46, 47, 44]. The work of [1] proposes a new approach using Voronoi diagram similar to [13] and further validates its mathematical correctness. One limitation of this approach is that it is not well suited for large-scale simulation due to the performance concern. All the above LCA algorithms somewhat only consider a limited range of local area. Therefore, the work of [6] introduces a new hybrid long-range collision-avoidance method to improve the smoothness of agent trajectories.

In recent years, significant research interests have been drawn to data-driven crowd-simulation models. These approaches automatically diversify agent motions based on a limited motion capture dataset [7] or obtain realistic pedestrian patterns from video footages for crowd simulation [28, 22, 32]. In addition, Lemercier et al. [23] can simulate group of people, following one by one, to walk through a zigzag tunnel. Gu and Deng proposed intuitive sketch-based interfaces to generate various group formations during the crowd simulation process [8, 9].

In this work, we borrow the concept of continuum crowd as the base of global and local planning. As mentioned above, the continuum crowd has two significant shortcomings: performance and LCA. Since our method is full GPU-implementation, the performance has been effectively eliminated to certain extent. Furthermore, inspired by the LCA algorithm in [1], we develop our own GPU-friendly LCA algorithm. Specifically, our approach extends the global planning part of continuum crowd by introducing the Eulerian uncertainty field, and add the Lagrangian uncertainty to form a hybrid local planning scheme. We also add new types of agent groups without goal-directed behavior to enhance both the Eulerian and Lagrangian uncertainties. Our navigation field control is effective to alleviate severe agent congestion and enhance realistic moving trajectories.

Chapter 3

Methods

3.1 Overview

Our approach supports global and local planning on a high-resolution 2D grid for simulation of various density crowds. As illustrated in Figure 3.1, at the initialization step, our approach computes a Eularian uncertainty field and a sub-domain navigation field in an offline manner. At runtime, a simulation loop is executed for every step, described as follows: every goal-directed agent's preferred velocity $\mathbf{v}$ and motion vector $\mathbf{m}$ are computed by the global planner; and both $\mathbf{v}$ and $\mathbf{m}$ are constrained by an Eularian uncertainty field. Every non-goal directed agent's local velocity $\mathbf{v}$ and motion vector $\mathbf{m}$ are computed by the local planner with Lagrangian uncertainty. In addition, in certain pedestrian traffic areas, the agents' $\mathbf{v}$ and $\mathbf{m}$ are solely determined by a user-specified navigation field. Lastly, we adjust $\mathbf{v}$ and $\mathbf{m}$ to avoid local collisions with other agents and the environment.

3.2 Uncertainty Model

Assumption and problem analysis: Our uncertainty model is built based on the following real-world crowd observations in many scenarios (e.g., shopping malls, parks, etc).

- (a) People are likely to stop and then look around in front of their places of interest.
- (b) People towards the same destination are likely to gather in some common waypoints through which their planning routes pass.
- (c) People without specific goals are likely to randomly roam within the area of interest.
- (d) People with varied physical properties (e.g., age and gender) and psychological attributes (e.g., patience and aggressiveness) have different behaviors in a crowd.

We analyze the above observations from the perspective of crowd simulation and model them as follows.

- Macroscopic uncertainty characteristic as demonstrated in (a) and (c) can be approximately modeled as a pre-computed 2D scalar field, combined with a general purpose global planner.
- Microscopic uncertainty characteristic as demonstrated in (b) and (d) can be

modeled as a state machine of Region-of-Interest(ROI) ahead of the local planner in the simulation pipeline.

3.3 Construction of Eulerian Map

We first construct an Eulerian map based on a bird-view scene map. A step-ahead analysis of the scene map is used to tag each building's signature attributes, as illustrated in Fig. 3.2(a). Without loss of generality, we make a high-level simplification on the scene map and only preserve four basic attributes: *entertainment*, *shopping*, *dinning* and *other*. Then, we construct a four-layered Eulerian map based on the four building attributes. Different layers in the Eulerian map can be illustrated in a color-coded way, as shown in Fig. 3.2(b).

3.4 Eulerian Uncertainty Field

Eulerian uncertainty field in our model is to reflect the degree of uncertainty that certain groups of agents overtake during their macroscopic advection in the whole domain. For the sake of discussion, let us assume $\boldsymbol{i}$ is the ith building with attributes A_i. The geometric attribute of $\boldsymbol{i}$ is described by Center C_i and Boundary Set B_i. Empirical observations tell us that the closer people are moving to $\boldsymbol{i}$, the more likely people will consider to stop, wait and see. We use the following mathematical

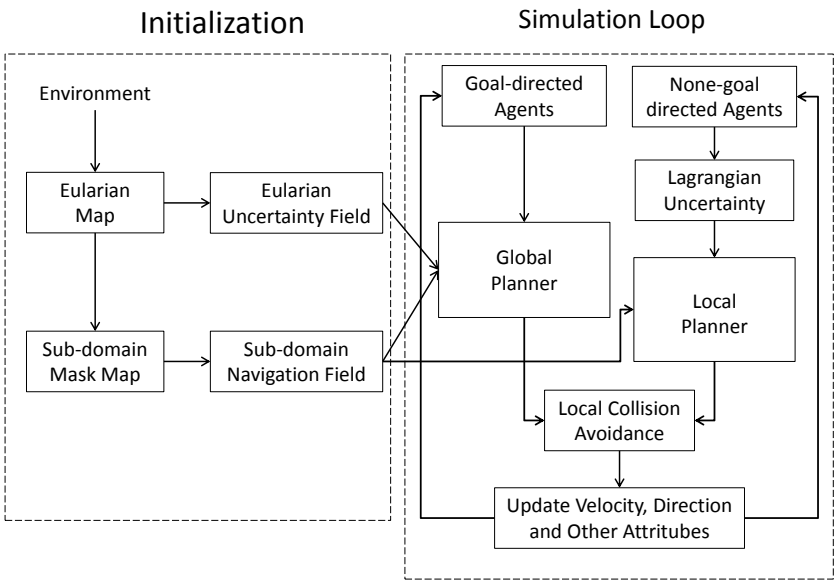
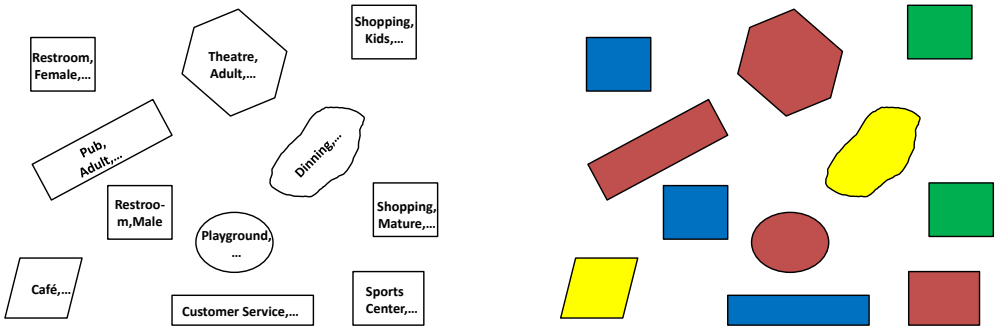


Figure 3.1: A schematic overview of our method



(a) (b)

Figure 3.2: Construction of the Eulerian map from a scene map example: *(a) a 2D scene map from bird-view, and (h) its corresponding color-coded Eulerian map.*

equation to model this uncertainty.

$$\mathbf{U}(X,i) = \begin{cases} e^{-k*dist(X,B_i)} & \text{if } dist(X,B_i) \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

We can further extend the above equation for different layers of the Eularian map, shown in A_i, assuming there are a total of m buildings in the k-th layer.

$$\mathbf{U}_k(X) = \sum_{0<i<m} \mathbf{U}(X,i) \tag{3.2}$$

In this way, the Eulerian map (total K layers) can be described as follows:

$$\mathbf{U}(X) = [U_0(X), U_1(X), \cdots, U_K(X)] \tag{3.3}$$

Agent j is only affected by the relevant layers of Eulerian Map through an individual-specific or group-specific mask, $M(j)$.

$$\mathbf{M}(j) = [M_0, M_1, \cdots, M_i, \cdots, M_K] \tag{3.4}$$

For the agent j, now we have:

$$\mathbf{U}^{'}(X,j) = U(X) * M(j) \tag{3.5}$$

A more smoother model is to use a Gaussian filter in 2D space after comparing the Eulerian uncertainty model with Depth-of-Field Circle-of-Confusion (CoC)[36, 24]. Let G be a Gaussian convolution operator with 3×3 kernel. For the i-th building in the Eulerian map, the propagated Eulerian uncertainty map is computed as follows:

$$\mathbf{U}_g(X,i) = G * E_i = w_G \sum_{P \in \Omega} G(P-X) E_i(X) \tag{3.6}$$

where $G(X)$ is a typical Gaussian weight at offset X, w_G is the sum of Gaussian weights.

Then, by summing all the buildings in a certain layer in the Eulerian map, a Gaussian-based Eulerian uncertainty field can be constructed accordingly as follows.

$$\mathbf{U}_{g,k}(X) = \sum_{0<i<m} \mathbf{U}_g(X, i) \tag{3.7}$$

$$\mathbf{U}_g(X) = [U_{g,0}(X), U_{g,1}(X), \cdots, U_{g,K}(X)] \tag{3.8}$$

$$\mathbf{U}_g^{'}(X, j) = U_g(X) * M(j) \tag{3.9}$$

3.5 Lagrangian Uncertainty Model

The Lagrangian uncertainty model in our approach is build on the following several hypotheses derived from empirical observations on the behavior of non-goal directed agents in a crowd. Unlike goal-directed agents, non-goal directed agents have no clear destination. Each agent has a set of goal candidates denoted by $\mathbf{G}$, which changes over time.

ASSUMPTION #1. *Every agent should take into account the visual occlusion of each building from his/her viewpoint.* – Illuminated by the synthetic-vision based crowd simulation work [27], we assume that each agent does not have infinite field of vision. Instead, the visual condition of each building is rated from worst to excellent based on each agent's field of view.

ASSUMPTION #2. *Each agent has a visit history to minimize the repeated visiting on the same building.* – After visiting a building, people typically exclude their recent visits from their near-future targets. As such, those recently visited buildings should be rated as "less likely to visit in near future".

ASSUMPTION #3. *The shortest distance rule implies that people often visit the comparatively closer building if other external conditions are similar.*

By combining the above assumptions, for each building in G, each agent considers its rating as a linear blending of the following three terms.

- The vision condition, denoted by $V = [V_0, V_1, \cdots, V_n]$.
- The visiting history, denoted by $H = [H_0, H_1, \cdots, H_n]$.
- The distance, denoted by $D = [D_0, D_1, \cdots, D_n]$.

In this way, the rating of the i-th building for the jth agent can be described as follows:

$$\mathbf{R}_j(i) = \alpha V_i + \beta H_i + \gamma D_i \tag{3.10}$$

The ratings of all the goal candidates for the jth agent is denoted as $R_j = [R_j(0), R_j(1), \cdots, R_j(n)]$. After normalization, we have a 1D discrete probability density $R_j^{'}$ for candidates:

$$\mathbf{R}_j^{'} = Normalize(\mathbf{R}_j) \tag{3.11}$$

The discrete cumulative distribution function of $R_j^{'}$ is described as follows:

$$\mathbf{L}_j(i) = \sum_{0<k<i} \mathbf{R}_j^{'}(k) \tag{3.12}$$

Intuitively, we first generate a random value r in $[0, 1]$, and then a *temporal* Lagrangian goal $U_L(j)$ for the jth agent can be determined as follows:

$$\mathbf{U}_L(j) = \begin{cases} 0 & \text{if } L_j(0) \leq \mathrm{r} < L_j(1) \\ 1 & \text{if } L_j(1) \leq \mathrm{r} < L_j(2) \\ \ldots & \\ k & \text{if } L_j(k) \leq \mathrm{r} < L_j(k+1) \\ \ldots & \\ n-1 & \text{if } L_j(n-1) \leq \mathrm{r} \end{cases} \tag{3.13}$$

After the temporal Lagrangian goal is determined (its uncertainty is denoted as r), the j-th agent starts to move towards $U_L(j)$. Before the agent arrives at $U_L(j)$, no new decision needs to be made for an updated Lagrangian goal, $U_L^{'}(j)$. A typical situation can be illustrated in Fig. 3.3.

The agent advection scheme in our approach is a local planner based on the above analysis. Specifically, we extend the spatial colonization scheme [1] by plugging our Lagrangian goal $U_L(j)$ for non-goal directed agents into its framework (see the equation below).

$$\mathbf{g}_t(j) = \mathbf{G}(i) | i = U_L(j) \tag{3.14}$$

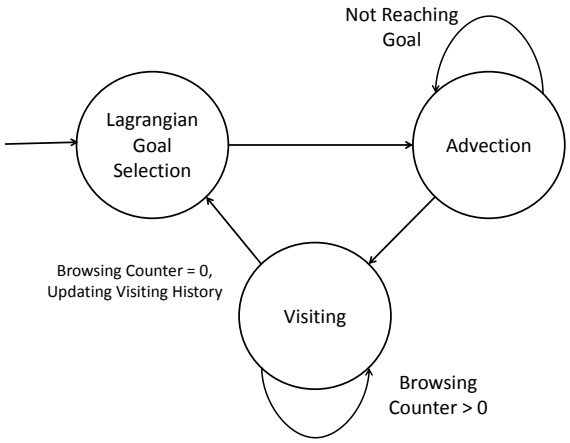


Figure 3.3: The state transition diagram for non-goal directed agents with Lagrangian uncertainty

3.6 Sub-domain Navigation Field

Both global and local planners, or even hybrid planners, are not able to effectively simulate certain informal cultural conventions such as walking on one side of the path. Instead, inspired by the demonstrated effectiveness of user-guided navigation fields (e.g., the work of Patil et al. [28]), we provide a new perspective for this problem and introduce a sub-domain navigation field with the following assumption: *Only a portion of the entire scene needs people to follow cultural conventions and hence resolving congestion, therefore only those places need a navigation field to guide people.* Note that our navigation field is not goal-directed since we have our own global planner for such a task.

To highlight the region with the navigation field out of the entire 2D grid, we

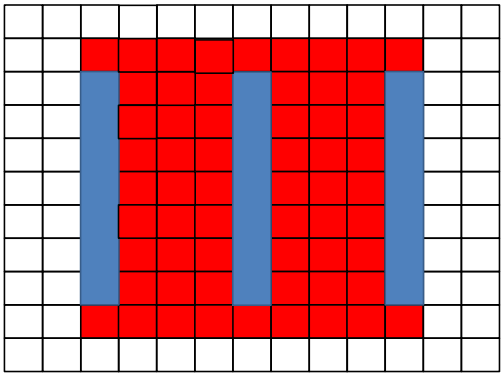
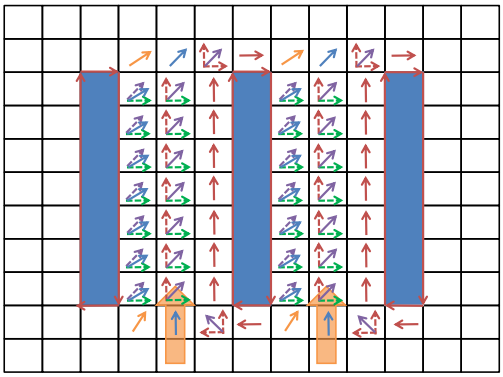


Figure 3.4: An example of the sub-domain mask map in our approach

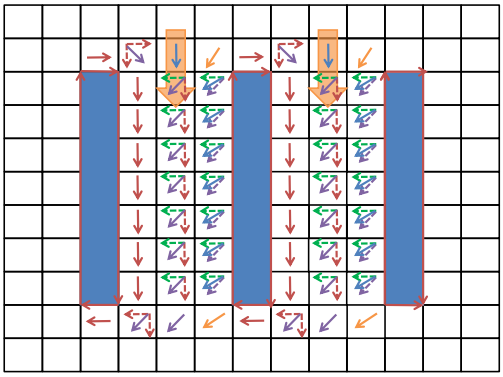
define a *sub-domain mask map* that looks like an axis-aligned bounding box of high-traffic regions with few padding grid cells (refer to Fig. 3.4).

In the highlighted regions in the sub-domain mask map, we employ a navigation field to ensure agents walk on the right side of their path. This would alleviate the potential macroscopic congestion in the scene. We use the following two hypotheses to design the right model for our navigation field: (a) Each agent next to obstacle right-hand is considered to be on the ideal track, and he or she should walk along the boundary of obstacle; and (b) each agent away from obstacle right-hand should adjust his or her heading direction towards the ideal track as soon as possible. Based on these hypotheses, we can intuitively construct a corresponding sub-domain navigation field, as illustrated in Fig. 3.5.

Main computation procedure could be seen as a series of propagation passes like Fast Marching Method [28]) and described as follows.



(a)



(b)

Figure 3.5: Example of sub-domain navigation field: *(a) positive y-axis direction , and (b) negative y-axis direction. Note that solid line with different color indicates the navigating direction calculated by different pass.*

Pass 1, navigating direction of cell adjacent to obstacle (ideal track) highlighted by red color is initialized as Fig. 3.4(a),(b), being consistent with boundary direction of obstacle (counter-clock-wise in right-side advection).

Pass 2, navigating direction of cell next to ideal track highlighted by blue color is computed by different cases.

Case with one neighboring cell visited: we add an offset vector to impel the agents towards the ideal track as shown in following equation.

$$\mathbf{N}(i,j) = \begin{cases} (1,0) + \mathbf{N}(i+1,j) & \text{iff } \mathbf{N}(i+1,j) exists \\ (0,1) + \mathbf{N}(i,j+1) & \text{iff } \mathbf{N}(i,j+1) exists \\ (-1,0) + \mathbf{N}(i-1,j) & \text{iff } \mathbf{N}(i-1,j) exists \\ (0,-1) + \mathbf{N}(i,j-1) & \text{iff } \mathbf{N}(i,j-1) exists \end{cases} \tag{3.15}$$

$$\mathbf{Nav}(i,j) = Normalize(\mathbf{N}(i,j)) \tag{3.16}$$

Case with two neighboring cell visited: we simply average them together since usually it occurred where turning corner existed.

We use the same way to calculate the navigating direction of rest cells pass by pass until all of them are visited.

3.7 Hybrid Crowd Dynamics

As described above, we use a global path planner as well as a local path planner to incorporate the Eulerian uncertainty and the Lagrangian uncertainty to our crowd simulation system, respectively. In addition, we also need to incorporate the sub-domain navigation field into the entire crowd dynamics pipeline. Before the simulation loop starts, the Eulerian uncertainty field and sub-domain navigation field are pre-computed. And, they need to be re-computed only when the environment is changed.

The following procedure summarizes the major steps involved in one simulation loop (also refer to Fig. 3.1). Note that before the simulation starts, we assume the initial positions, directions, and other attributes of all the agents are given or specified by users.

1. The positions, directions and other attributes of all the agents are utilized to compute the discrete Voronoi diagram, density field, cost field, potential field, etc.

2. Global planning is performed on goal-directed agents, while local planning is performed on Lagrangian agents by following the state transition diagram (refer to Fig. 3.3).

3. If there exists an Eulerian uncertainty field where agents stand by, the velocities and directions of these agents will be affected by the Eulerian uncertainty.

4. If agents step into the region with a specified sub-domain navigation field, the

velocities and directions of these agents will be dominated by the navigation field.

5. Finally, local collision avoidance algorithm is performed to update agent positions and make corrections if collision occurs.

3.8 Collision Avoidance

As described in the work of [43], the continuum crowd framework can only simulate dynamics up to the resolution of the used grid; collisions could occur within individual grids. It uses a pair-wise check mechanism to eliminate potential collisions in individual grids. However, this pair-wise check algorithm for collision avoidance is not GPU-friendly. In our approach, we introduce a new, efficient and GPU-friendly collision avoidance algorithm.

Inspired by the work of [1], we compute a discrete Voronoi diagram at each time step and then exploit the geometric properties of the Voronoi diagram and convex sets. Basically, if at each time step every agent's new position is located within its previous Voronoi cell, then no collision will occur. The major performance bottleneck here is to compute the distances from the location of every agent to all the edges of Voronoi cell. Therefore, the original method [1] is severely limited by the large number of agents.

We significantly improve the performance of the above model [1] by borrowing the concept of Poisson disc sampling (refer to Fig. 3.6), commonly seen in GPU

rendering techniques [36, 24]. Tap positions on the filter kernel are determined by using stochastic methods according to a Poisson disc distribution. 13 samples (12 outer samples + the center sample, yellow-colored in Fig. 3.6) are used in image-space post-processing [36, 24]. In our approach, we add four more samples (purple-colored in Fig. 3.6) at both ends of two local orthonormal bases.

In our scenario, the updated position $A^{'}$ at current time step is aligned with the center tap, and the outer taps are sampled from the Voronoi diagram based on their offset and scaled according to the agent's radius R. One intuitive way to detect collisions is to see whether all the outer samples have the same Voronoi cell ID as the center tap. If so, then we can assume that the agent is not penetrating into the neighboring Voronoi cells containing other agents. The example illustrated in Fig. 3.6 shows the opposite situation, where the sample with green color steps into other Voronoi cell. Besides we also want to compute the extent of penetration if occurs. To detect collision and compute penetration in one pass, we re-order all the sample taps along the y-axis (the numbers in Fig. 3.6 denote their orders). Collision detection starts from the 0th sample, then the 1th sample, and so on. Once a collision is detected, the approximate extent of penetration can be quantified as a vector C in Fig. 3.6. Then, the adjusted motion vector $M^{'}$ is offset by Vector C as follows.

$$\mathbf{M}^{'} = M - C \tag{3.17}$$

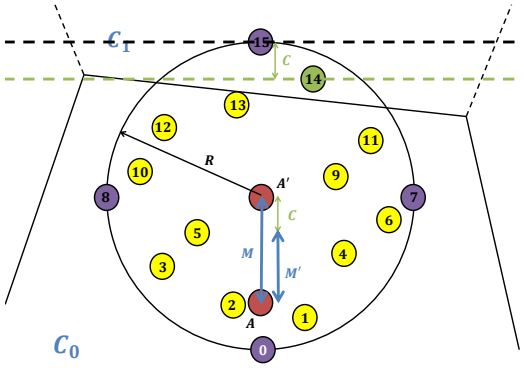
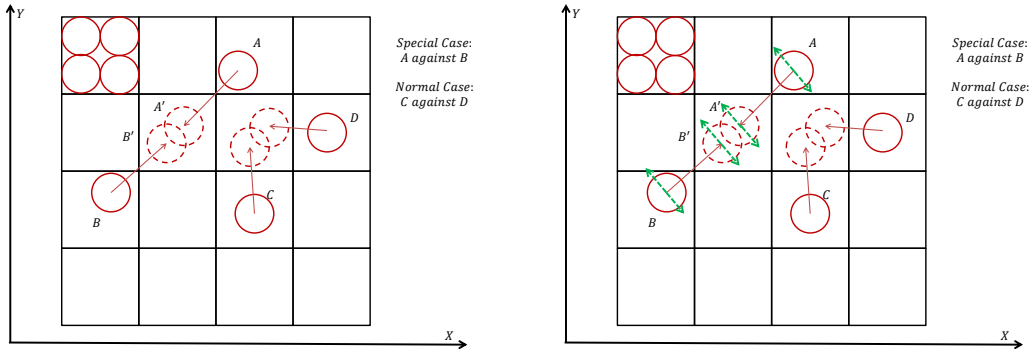


Figure 3.6: Illustration of Poisson disc sampling for collision avoidance in our approach



(a) (b)

Figure 3.7: Predictive Binning with Lagrangian uncertainty : *(a) predictive binning, (b) collision resolving with lagrangian uncertainty.*

3.9 Collision Avoidance with Lagrangian Uncertainty

Collision avoidance in area of crowd simulation usually is composed of two stages: collision detection and thereafter the agents separation. Our contribution is to add Lagrangian uncertainty to separate the agents while handling the predictable collision.

3.9.1 Collision Detection

Regarding the collision detection, our work inherited the same method: Binning in works [40], [39]. Our method differs from those methods in that We splattered the each agent's position few steps ahead, and Binning is operated upon this predictive agent's position as Fig. 3.7(a). There are two possible cased illustrated in the above Figure.

1. Two agents: A and B have the tendency to collide each other face to face. Our Lagrangian uncertainty is specifically handling this situation. We will set a threshold angle close to 180 degree in program.

2. Two agents: C and D are about to collide by the side. We follow the regular way to separate both agents in this scenario.

3.9.2 Resolve Collision

Illustrated in Fig. 3.7(b), face to face collision between two agents A and B, in real-world scenario, might not be avoid right away. In the next step, two people might choose the same offset direction to avoid each other. Regular agent separation method might be enough to simulate this scenario. We designed the Lagrangian uncertainty to determine the offset direction between $[0, \pi]$. Therefore, agent A and B might be or not be collided with each after one step adjustment. This is the break-through out of the regular solution.

Chapter 4

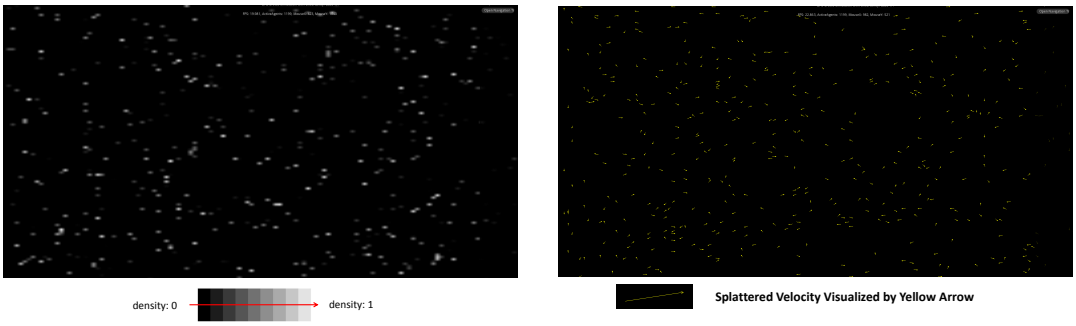
Global Planner

This chapter will review two state-of-art global planner: Continuum Crowd and Navigation Field first. And afterwards details on how to use our Eulerian and Lagrangian uncertainty model upon these two global planners will be explained.

4.1 Continuum Crowd

The main pipeline of continuum crowd is ordered by:

- Construct smooth density and velocity field.
- Calculate anisotropic speed field based on local crowd density.
- Evaluate anisotropic cost field based on uniform cost function.
- Propagate potential field outwards originated from the goal destinations.



(a) (b)

Figure 4.1: Example of density and velocity splatter: *(a) density splatter , and (b) velocity splatter.*

- Deduct changing gradient of above potential field to direct agents.

All these major steps need recalculated every simulation step.

4.1.1 Density and Velocity Splatter

Splattering each agent's attributes: density and velocity, in this thesis, into adjacent grid cells is the first step of the entire pipeline. The major intention of this step is to provide the smooth scalar and vector field over the discrete 2D grids as Fig. 4.1(a),(b). A typical implementation for attributes splatter is by using additive alpha blending with following texture with alpha transparency as Fig. 4.2.

Following by the same rule, we can also easily implement predictable discomfort by splattering density a few motion step ahead.

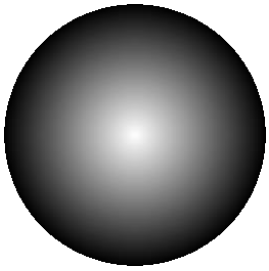


Figure 4.2: Cone texture for additive alpha blending

4.1.2 Anisotropic Speed

Flow Speed is to be utilized to describe if the agent within a certain grid cell can move by its maximal speed or move with the surrounding flow. The speed field is anisotropic along with the four directions: North, West, South and East, corresponding to each cell face. The amount of speed is proportional with the density situation as in following equation. Value of maximal and minimum density as well as maximal speed should go with the specific application. The anisotropic flow speed is calculated by dot product between neighboring velocity with neighboring cell directions.

$$\mathbf{S}peed_a(i,j) = \begin{cases} flowspeed & Density_a(i,j) > DensityMax \\ lerp(flowspeed, maxspeed) & Density_a(i,j) \subseteq [DensityMin, DensityMax] \\ maxspeed & Density_a(i,j) < DensityMin \end{cases} \tag{4.1}$$

$$\mathbf{C}ost(P) = \alpha \int_P 1ds + \beta \int_P 1dt + \gamma \int_P gdt \tag{4.2}$$

4.1.3 Cost Field

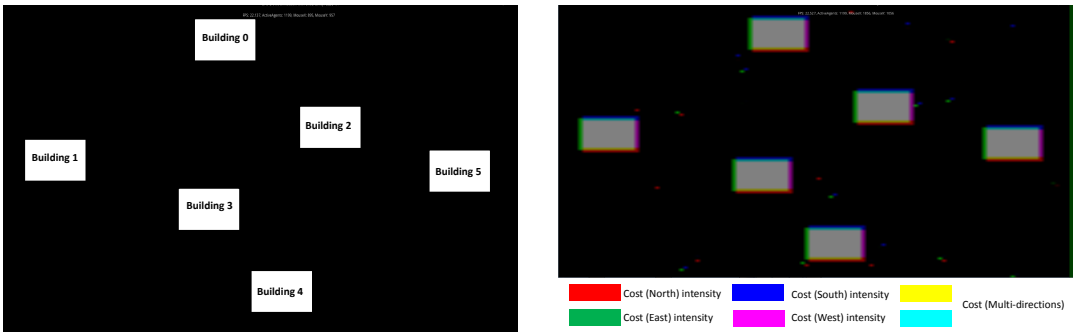
From the viewpoint of continuum crowd method, cost field of each grid cell is defined against its four neighboring cells, quite similar as the previous speed calculation. Three hypothesis based on observation formed up the ultimate cost function, in continuum form as follows. Path length, travelling time and discomfort are the three major terms in the equation, representing the three hypothesis respectively. Weighting coefficient: α, β and γ should be adjusted by the specific applications. The cost function in continuous form can be rewritten into 2D discrete form

$$\mathbf{C}ost_a(i,j) = \frac{Speed_a(i,j) * \alpha + \beta + \gamma * Discomfort_a(i,j)}{Speed_a(i,j)} \tag{4.3}$$

Here, term indicated as $Discomfort_a(i,j)$ is composed by environment map plus predictable discomfort previously introduced as Fig. 4.3(a),(b).

4.1.4 Potential Field

Potential field is a scalar field that evaluate the accumulative cost between each 2D grid cell to goal destination. It has several characteristics: Firstly, the computation is originated in goal destination and propagated outwards. Secondly, potential field needs to be recomputed in each time-step and for each common goal areas. Therefore, potential calculation is the most expensive computation step within the whole



(a) (b)

Figure 4.3: Anisotropic cost field: *(a) simulation environment map , and (b) calculated cost field.*

pipeline. Mathematically, eikonal equation highly abstracted this problem.

$$\|\nabla Potential(P)\| = Cost(P, D) \tag{4.4}$$

Here, P indicates the position at simulation domain. Gradient of potential at P is dependent upon the anisotropic cost at P. The higher the cost attached to specific location P, the faster the potential will get increased. Started from goal cells, potential of which is zero, high cost grid cells would be propagated later than lower-cost grid cells. Original paper computed the potential field by using Fast Match Method (FMM) in CPU. It is very slow if considering the dense grid resolution as well as increased group numbers. This thesis we implemented it in GPU computing pipeline. For more details, you can refer to paper [16], [17] and [15].

As pointed out in [43], each group is sharing a common goal and each goal destination need propagating a entire 2D potential field. Performance of CPU-based

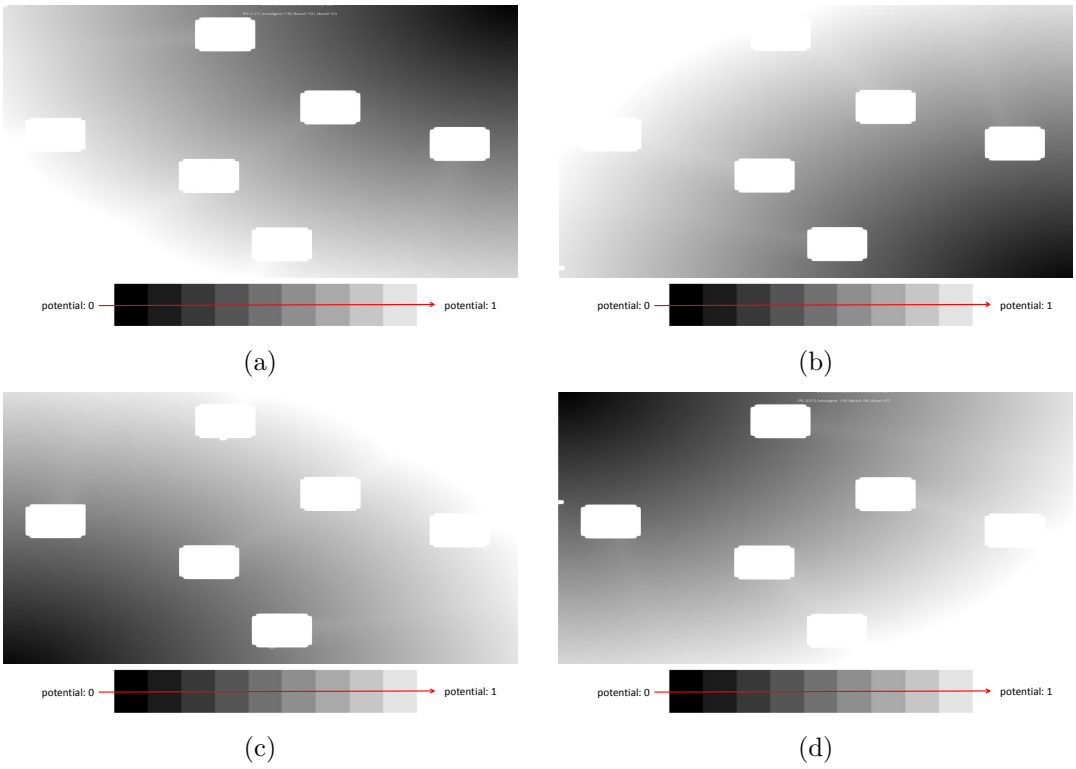


Figure 4.4: Potential fields for four groups: *(a) potential fields propagated from up-left corner (goal area), (b) potential fields propagated from bottom-right corner (goal area), (c) potential fields propagated from bottom-left corner (goal area) and (d) potential fields propagated from up-right corner (goal area)*

implementation is quite impacted by the increasing number of goals. A good GPGPU implementation can dramatically decrease the iteration numbers and moreover make full use of hardware-accelerated 4-tuple vector computing to boost the entire performance. The following four potential fields (Fig. 4.4), goals of which are accordingly located at position of four extreme corners, are well suited for vector packing in GPU computing.

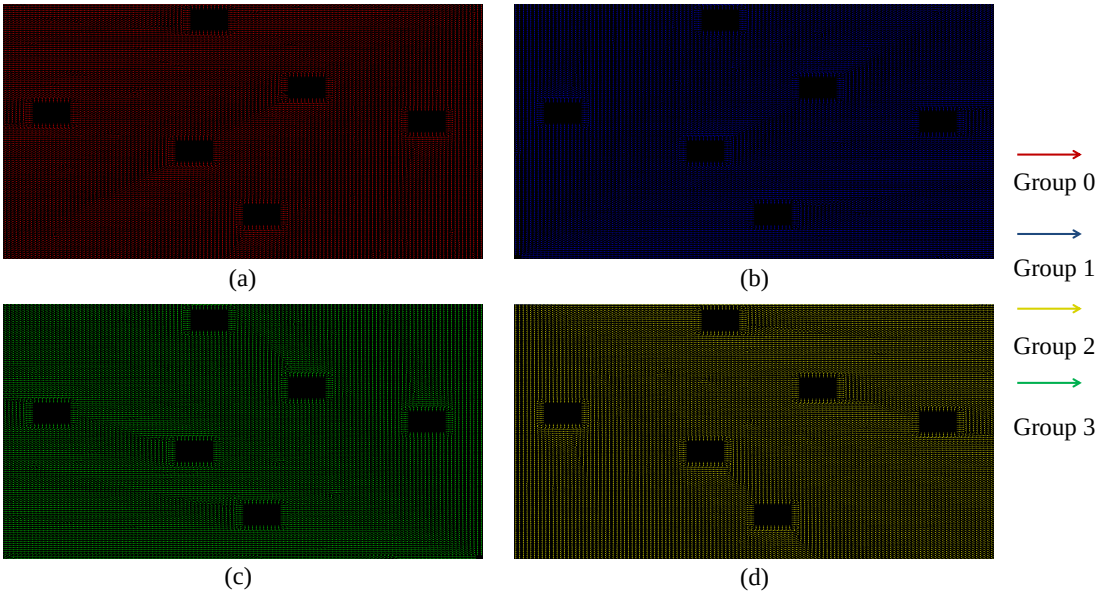


Figure 4.5: Gradient fields for directing four groups: *(a) directing gradient field towards up-left corner (goal area), (b) directing gradient field towards bottom-right corner (goal area), (c) directing gradient field towards bottom-left corner (goal area) and (d) directing gradient field towards up-right corner (goal area)*

4.1.5 Gradient Field

Once we get the potential field in previous step, we can easily calculate the gradient field by taking the potential difference along upwinding directions.

The gradient field will be utilized to navigate the agent's motion. (Fig. 4.5)

$$Direction(P) = -normalize(gradient(P)) \tag{4.5}$$

$$Velocity(P) = Direction(P) * Speed(P) \tag{4.6}$$

$$Motion(P) = Velocity(P) * \triangle(T) \tag{4.7}$$

4.2 Navigation Field

Navigation field differs from Continuum crowds in:

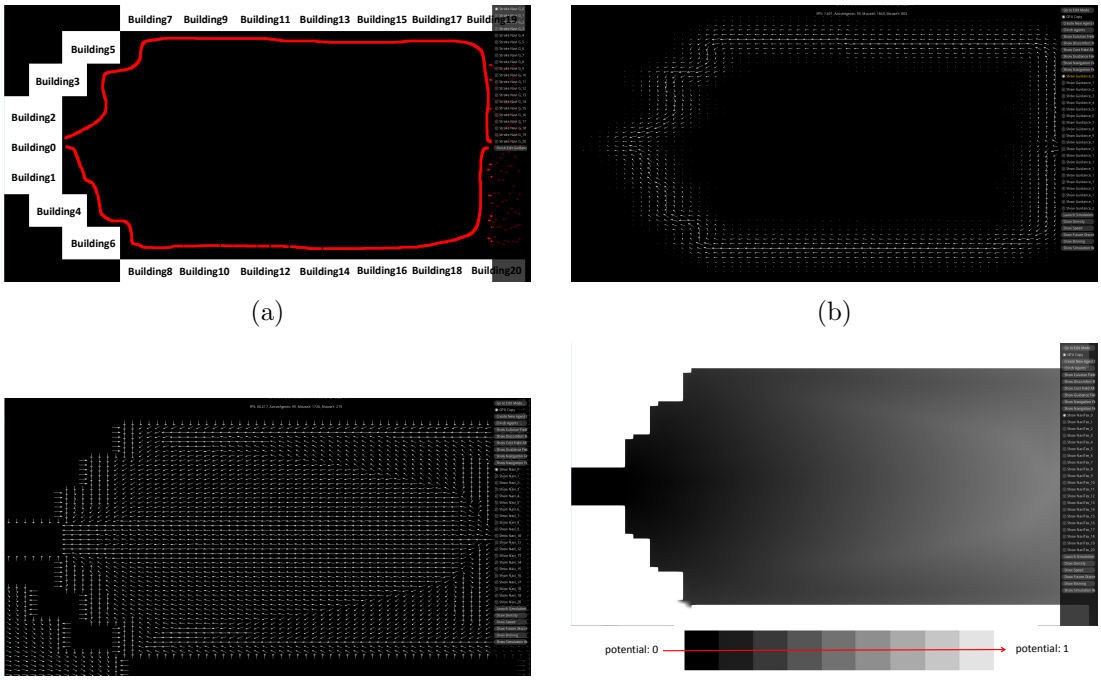
1. Firstly, usually it needs a user-specified guidance field.
2. Secondly, only needs to be recalculated at beginning or guidance has been changed.

4.2.1 User Guidance

A few techniques can provide user guidance: user sketchy interface and movie footage. In this thesis, we only use the user sketchy interface. (Fig. 4.6(a))

4.2.2 Anisotropic Cost Field

The quality of continuum crowd is largely dependent on the resolution of simulation grids. As an extension of continuum crowd, navigation field method can allow agent's move to any places between neighboring cells when it comes to compute the cost field. Therefore, it will be beneficial to overcome the blocky "path" due to the limited set of transition, only four possible directions in continuum crowd. Fig. 4.7 illustrated this possible unlimited set of traversal path in detail.



(a) (b)

(c) (d)

Figure 4.6: Goal-oriented navigation field with user guidance: *(a) user sketchy interface to draw guidance field, (b) propagated guidance field based on (a), (c) goal-oriented navigation field and (d) grey-scale potential field*

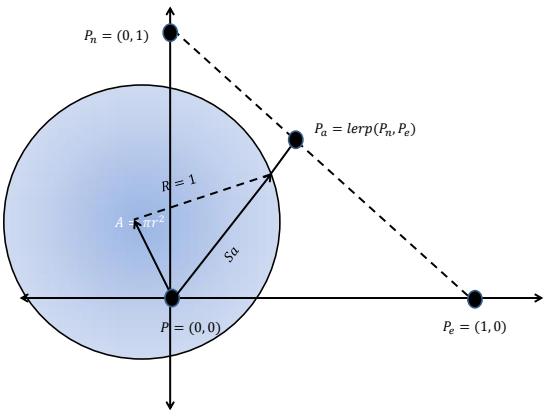


Figure 4.7: Unlimited possible set of traversal path

Mathematically, due to the introducing of speed function correlate to the direction, the previous eikonel equation will be escalated into Static Hamiltion-Jocobi-Bellman equation:

$$max(-\nabla Potential(P) \cdot D)S(P, D) = 1 \tag{4.8}$$

$$\|s\mathbf{a} - \mathbf{P}\| = 1 \tag{4.9}$$

$$s(P, \mathbf{a}) = \mathbf{a} \cdot G(P) + \sqrt{(\mathbf{a} \cdot G(P))^2 - G(P) \cdot G(P) + 1} \tag{4.10}$$

$$C(P, \mathbf{a}) = \alpha C(P_e) + (1 - \alpha)C(P_n) + 1/s(P, \mathbf{a}) \tag{4.11}$$

$$\frac{dC(P, \mathbf{a})}{d\alpha} = 0 \tag{4.12}$$

$$N(P) = \mathbf{a} = (\alpha, 1 - \alpha) \tag{4.13}$$

Compared with normal eikonel equation, it has a new term as $S(P, D)$ and a dot product with direction D. More specifically, D represents the any possible direction out of unlimited set of transition between P to its four neighboring cells. Direction-dependent speed termed as $S(P, D)$ indicated that the cost function right here only consider the time factor. This model proposed in original paper only calculate for once and rule out the factor like density and flow speeds. If we replaced the term $S(P, D)$ with $Cost(P, D)$, then we can need to update the calculation each time-step. But it will be easily incorporated into our current computation pipeline.

4.3 Continuum Crowd with Eulerian Uncertainty

In the previous sections, we introduced our two-level uncertainty model. And after that, we reviewed the two major global planner a little bit. In this section, we will introduce how we can embed the uncertainty model into continuum crowd method.

To use our Eulerian Uncertainty in continuum crowd, we first need to make some adjustment on the original continuum crowd as follows.

1 Extend the goal area from non-object area to unlimited any place in the simulation domain.

2 Arrays of Cost field needs to be computed for every possible goal places. This quite differed from the original method.

3 Potential field and Gradient field need to be accordingly computed as with the costs.

The entire computation task appears to be increased multiple times with the modification of the model here. But actually a well-designed GPU implementation can succeed to avoid that.

1 When calculating the potential field and gradient field, packing the data according to the group and take advantage of the 4-tuple vector processing capability in modern GPU.

2 Only calculate when it is needed, which is according to the specific application.

Eulerian uncertainty can be utilized to change the agent's goal at some specific time, such as there has a emergency event somehow. Eulerian uncertainty can also be utilized to stop agent's advection to simulate "browsing effect". Both of them can be thought of as a "distance-field", which is sourced from some Region-of-Interest (ROI), propagated and decayed with distance away from the source region. Again, add some slight modification on typical splattering can calculate it fast enough in GPU. Remember that the source area is a 2D region rather than a single 2D point. As we introduced our Eulerian model in Section3.4, all the distance-field here should be calculated based on the distance to the boundaries of source region. Fig. 4.8 illustrated it in detail.

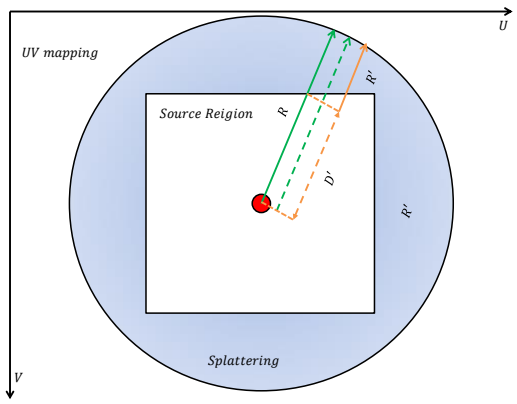


Figure 4.8: Splatter Eulerian uncertainty radiant from source

4.4 Navigation Field with Lagrangian Uncertainty

Navigation field only computed once at initialization stage or external guidance field has been changed. Navigation field is also goal-directed and its traversal cost is propagated outwards from the flagged goal area. In order to affiliate Eulerian uncertainty with the normal navigation field framework, we need to create a group of navigation fields, each of which have different goal destinations, and let the agent to follow them based on time-varying uncertainty. Here is the major steps needed to be done before launching the simulation.

- User manually specified all goal areas, time-varying or non-time-varying.
- For different goal area, global discomfort/obstacles should have little variance as Fig. 4.9(a),(b),(c) and (d).

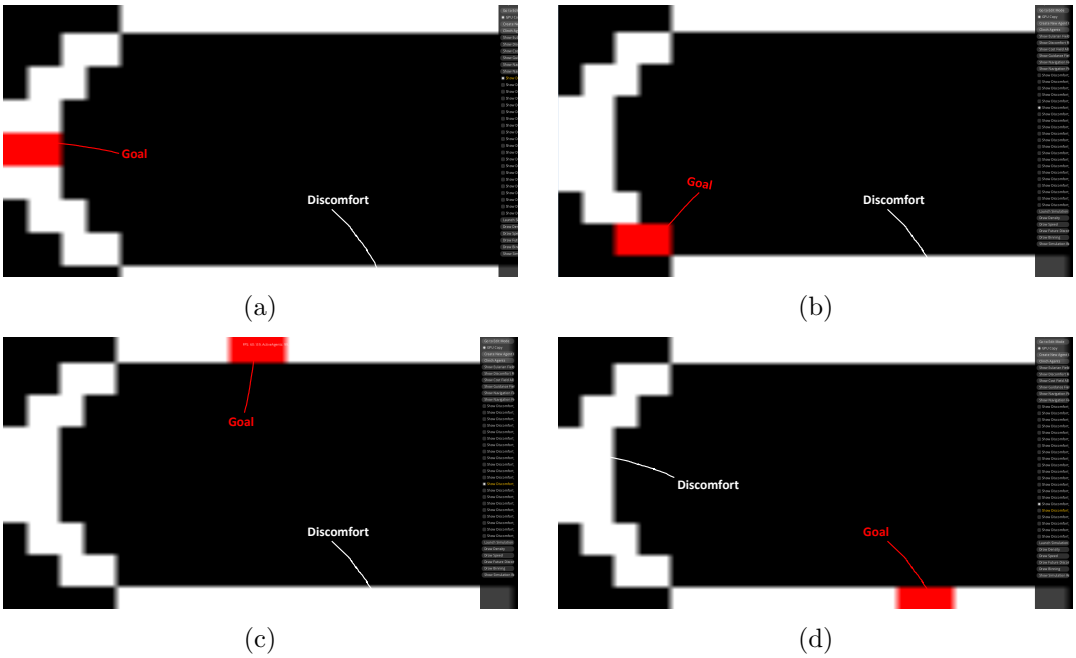


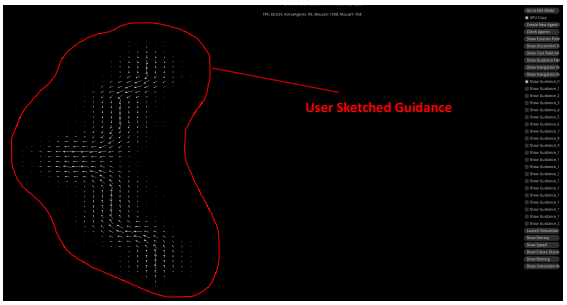
Figure 4.9: Group of four global discomfort for the same environment: *(a) group 0, (b) group 1, (c) group 2 and (d) group 3.*

.

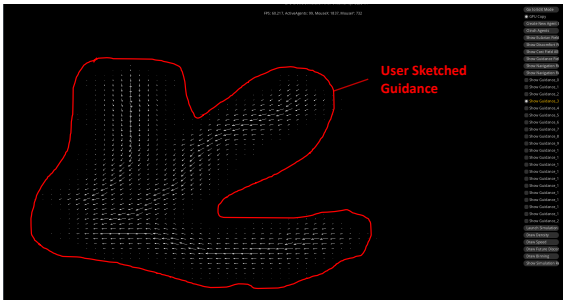
- User can optionally draw preferred guidance field on each navigation field as Fig. 4.10(a),(b),(c) and (d).(Navigation field deteriorated to shortest path navigation if no guidance)

- Generate all the navigation fields in the group as Fig. 4.11(a),(b),(c) and (d).

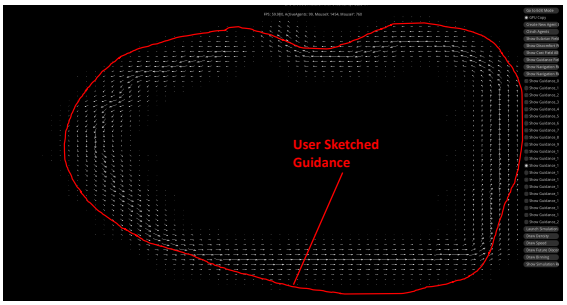
During the simulation runtime, each agents individually evaluate all the possible destinations followed by Section 3.5. Once it picked up one goal based on lagrangian uncertainty, agent move with the corresponding navigation field.



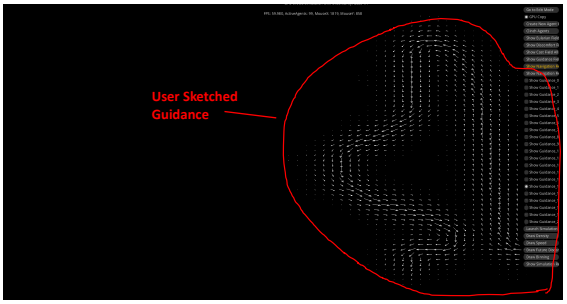
(a)



(b)

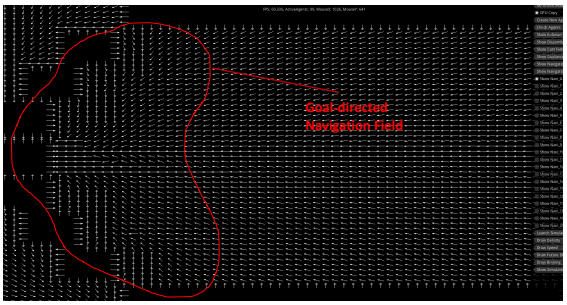


(c)

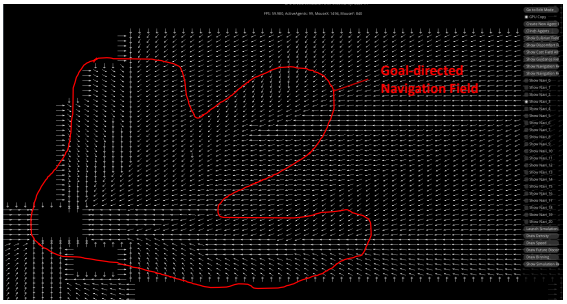


(d)

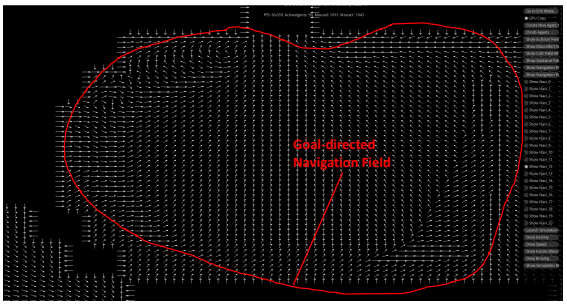
Figure 4.10: Group of four user guidance fields for four major destinations : *(a) group 0, (b) group 1, (c) group 2 and (d) group 3.*



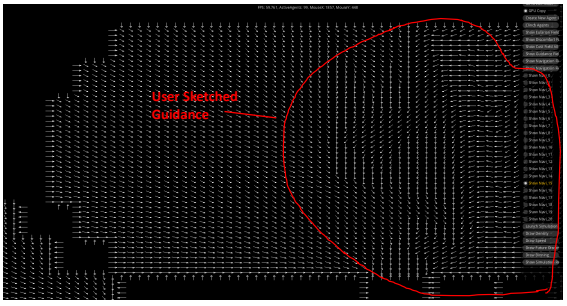
(a)



(b)



(c)



(d)

Figure 4.11: Group of four user computed navigation fields accordingly : *(a) group 0, (b) group 1, (c) group 2 and (d) group 3.*

Chapter 5

Implementation

This chapter will highlight some implementation tricks during this work. There are two major parts: 2D simulation and 3D Rendering. We use both GPGPU and GPU graphics pipeline to carry out the 2D simulation. Most steps we described in previous chapter need recomputing every time-step. As for the 3D Rendering part, we use GPU graphics pipeline to implement the skinning animation with instancing techniques as well.

5.1 GPU Computing Pipeline

Our approach was implemented using OpenGL 4.3 and GLSL on GPUs. The major steps in our GPU-based implementation are illustrated in Fig. 5.1, where the numbers in circles denote the orders of these steps. Before the simulation loop starts, we compute the following items once and recompute them if necessary.

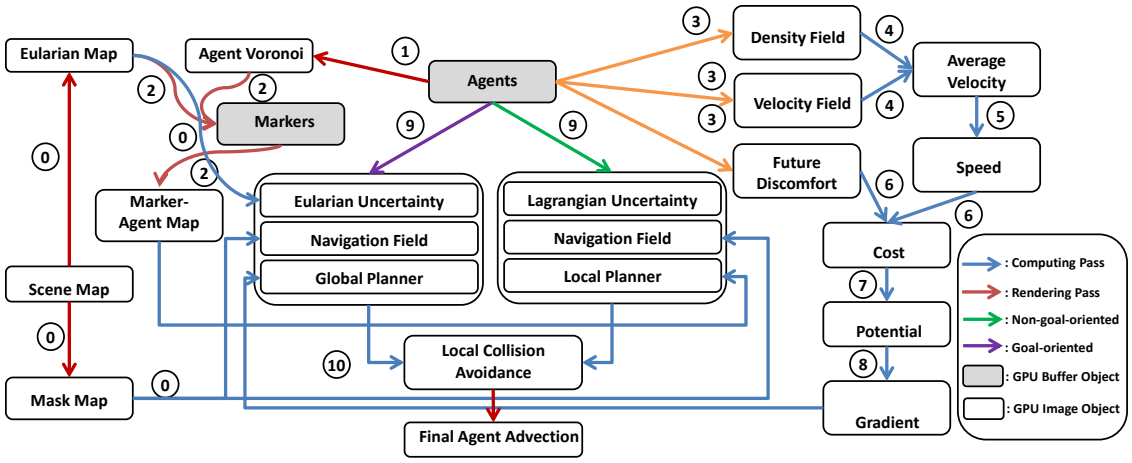


Figure 5.1: The overall pipeline of the full GPU implementation of our approach

- Layout randomized markers across the entire simulation grids using GPU Voronoi diagram and extract the centroid point of each cell;
- Render the scene environment to bake the Eularian map on the fly;
- Render the sub-domain mask map;
- Use GPGPU (Computer Shader) to compute the navigation field; and
- Use computer shader to compute the Eularian uncertainty field.

At step 1, render the GPU Voronoi diagram of all the agents by following the standard graphics rendering pipeline. Each agent's unique ID is encoded as a RGBA8 color.

At step 2, based on the Eularian map and the agent-Voronoi map, a marker-agent map is constructed using simple texel fetch in Fragment Shader. In practice,

we round the markers' positions and turn off hardware linear filtering.

At step 3, we splat all the agents' positions and velocities to compute the smoothed density field and velocity field using additive alpha blending. We also implement a future discomfort model by splatting each agent to its predictive grid cell given a certain amount of time steps. The Multi-Render-Target (MRT) capability provided in modern GPUs is utilized here to simultaneously render multiple textures in one pass.

From step 4 to step 8, a fast and efficient GPU implementation of the Continuum Crowd approach [43] is done using GPGPU technique as follows. At step 4, the average velocity of each grid (computed in computer shader using multiple outputs from step 3) is rendered to texture. At step 5, the velocity field is computed in computer shader using the output from step 4 and rendered to velocity texture. At step 6, based on the optimization function in [43], the cost field is built using the outputs from previous steps (Eularian map, predictive discomfort map, velocity field, density field, etc). At step 7, Eikonel equation is solved in computer shader to compute the scalar potential field. In practice, we optimize the computation using group shared memory resources, tuple packing, and MRT offered in modern GPU hardware. At step 8, a gradient vector field is computed by simply using the potential field.

At step 9, hybrid crowd dynamics are implemented again in computer shader. In this pass, we first select either global planning or local planning based on an agent's attributes. If the agent's type is goal-directed and affiliated with a group, it will be advected based on the gradient vector field from step 8, and corrected if it steps

into hot-spot with the navigation field. On the other hand, if the agent is non-goal directed, it will be advected by the local planner purely based on local available markers. Agents of both types are affected by the Eularian uncertainty field, and a corresponding state-machine will activated when needed. Another state machine for modeling the Lagrangian uncertainty only affects the decisions of non-goal directed agents.

At step 10, a final collision avoidance pass is imposed on the preliminary advection outcome. After collision detection, If a zero-advection goal-directed agent is given a new motion vector based on local marker situation, likely the deducted motion vector will slightly deviate from the result of the original continuum crowd model. However, in our experiments we found that it is a very effective way to solve the *decision deadlock* between two or three agents within the same grid cell.

5.2 3D Rendering

3D Rendering pipeline has two major components: skinning animation for characters and instancing technique for performance boost.

5.2.1 Skinning Animation

We use linear-blending skinning (LBS) for skinning animation in this thesis. Specifically, each vertex of the mesh model will be influenced by few number of bones, such as four bones and eight bones. A usual data structure for vertex attributes object

Figure 5.2: Data structure for VAO layout

(VAO):

Each frame, matrix for all bones will be linear interpolated in CPU between two keyframes and packed as buffer in the form shown in fig. 5.2. To animate the skinning character and make it walk like the same pace as its actual speed, we need to scale up the speed of linear interpolated by different magnitude and reorganized the packing scheme. (fig. 5.4). Each frame, CPU will pass down the packed bone matrix to GPU and also the speed. When executing the LBS in vertex shader, the corresponding group of bone matrix will be fetched from the right offset.

One experience that we obtained from our experiment is not overuse the number of different speed levels especially for the dense bone model. It will copy too much

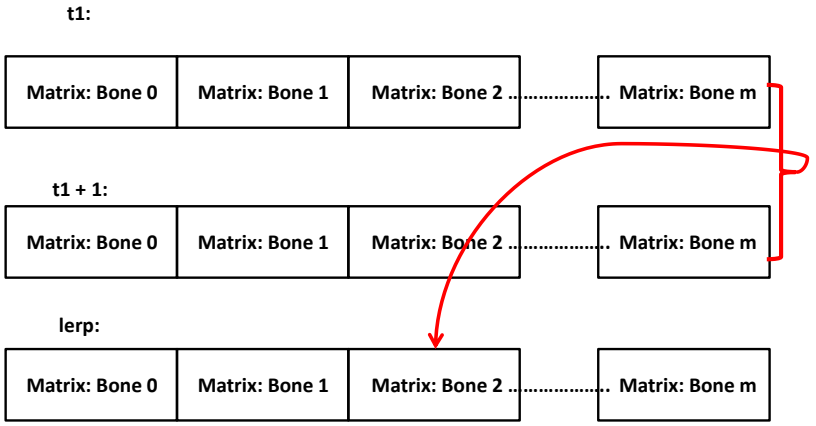


Figure 5.3: Bone matrix interpolation and packing

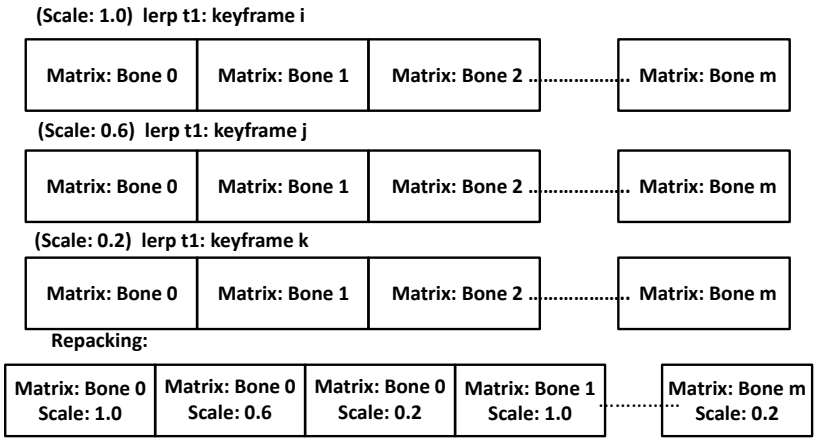


Figure 5.4: Bone matrix interpolation and packing

data from CPU to GPU and drop down the performance.

5.2.2 Instancing

Another technique we use in our 3D animation is to use instancing. Basically, we use the same skinning character for the entire group of agents. This will alleviate the overall computation burden deeply. The only difference between the agents within one group is the transformation matrix. Transformation matrix of all agents are packed into another separate buffer and fetched with the frequency of once a character.

Chapter 6

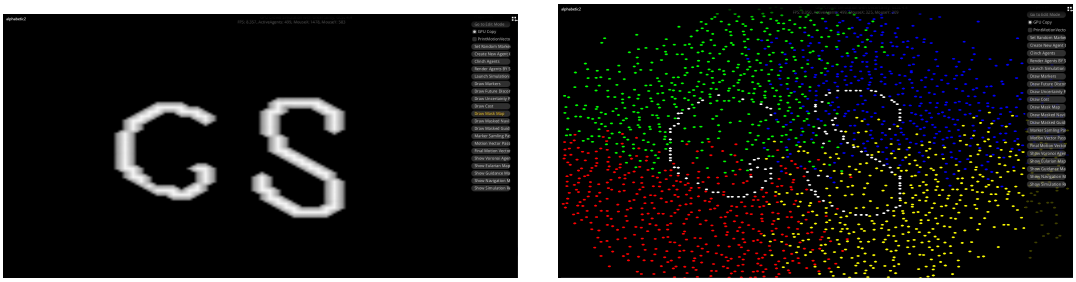
Results

In this section, we report the quality and performance of the proposed Crowd Simulation framework on several different scenarios. We experimented different ways of combination between global planner and uncertainty model.

6.1 Simulations of Separate Component

6.1.1 Eulerian Uncertainty only

A shop browsing effects, Figure.6.1was simulated by adding the Eulerian uncertainty only. We propagated the Eulerian uncertainty field the way as we discussed and people are attracted or distracted depending on their attributes as well as probability.Eulerian fields with user-sketchy was also experimented and it can offer many interesting behavior patterns such as "alphabetical" lingering, "geometrical shape"



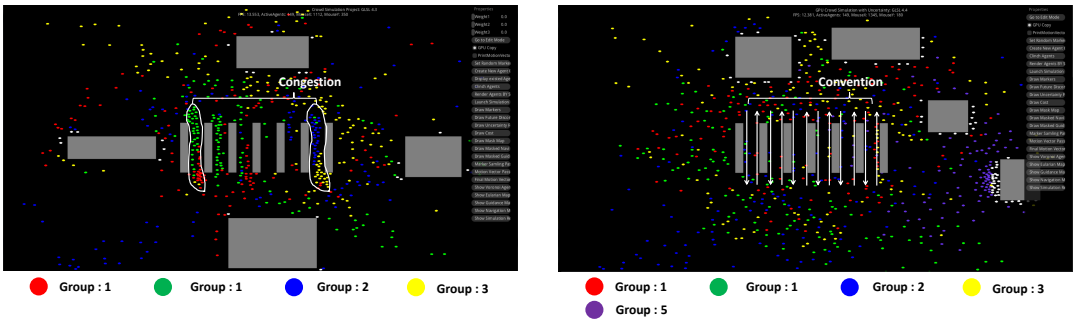
(a) (b)

Figure 6.1: User-sketched Eulerian Field:*(a)English Alphabetical (b) Result*

lingering(Fig.6.1) as an add-on effects to existing simulation pipeline.

6.1.2 Cultural Convention

Tunnel Crossings: Fig.6.2 shows the simulation of combining continuum crowd and sub-domain navigation field. Thousands of agents belonging to four different groups are walking through the narrow tunnel area during their way towards to their goal. Relying on continuum crowd and collision avoidance only cannot solve this problem as Figure.6.2. A much better simulation result was attained by using the hybrid method.



(a) (b)

Figure 6.2: Sub-domain Navigation Field:*(a)Congestion in bad crowd traffic area (b) No congestion and follow cultural convention*

6.1.3 Lagrangian and Eulerian Uncertainty

We simulated a small portion of non-goal-oriented agents behaving using Lagrangian Uncertainty with different combinations of evaluation function for goal candidates and updating frequency. In summary, non-goal-oriented agents are more behaving like moving obstacles by using density-based evaluation function than not. Besides, if the updating frequency increased, agents pattern appears to be more irregular and add a lot of randomness to others. If many agents are arriving the same ROI simultaneously, arc or sphere surrounding effects can be observed.

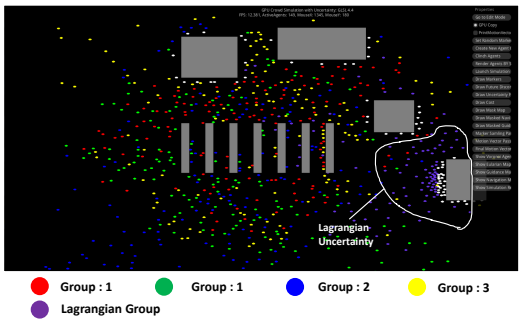


Figure 6.3: Lagrangian Uncertainty simulation result

6.2 Continuum crowd with Eulerian uncertainty

First showcase is to illustrate how Eulerian uncertainty can work with continuum crowd pipeline to simulate a few interesting crowd behaviors.

6.2.1 Continuum crowd

Four corners: four groups of agents are walking crossway inwards started from the four corners of the entire 2D simulation grids. This scenario presented a few emergent effects in crowd as in Fig. 6.4.

Fig. 6.4(a),(b),(c) and (d) display the lane formation and vortex effect in middle of the scene with increasing number of agents.

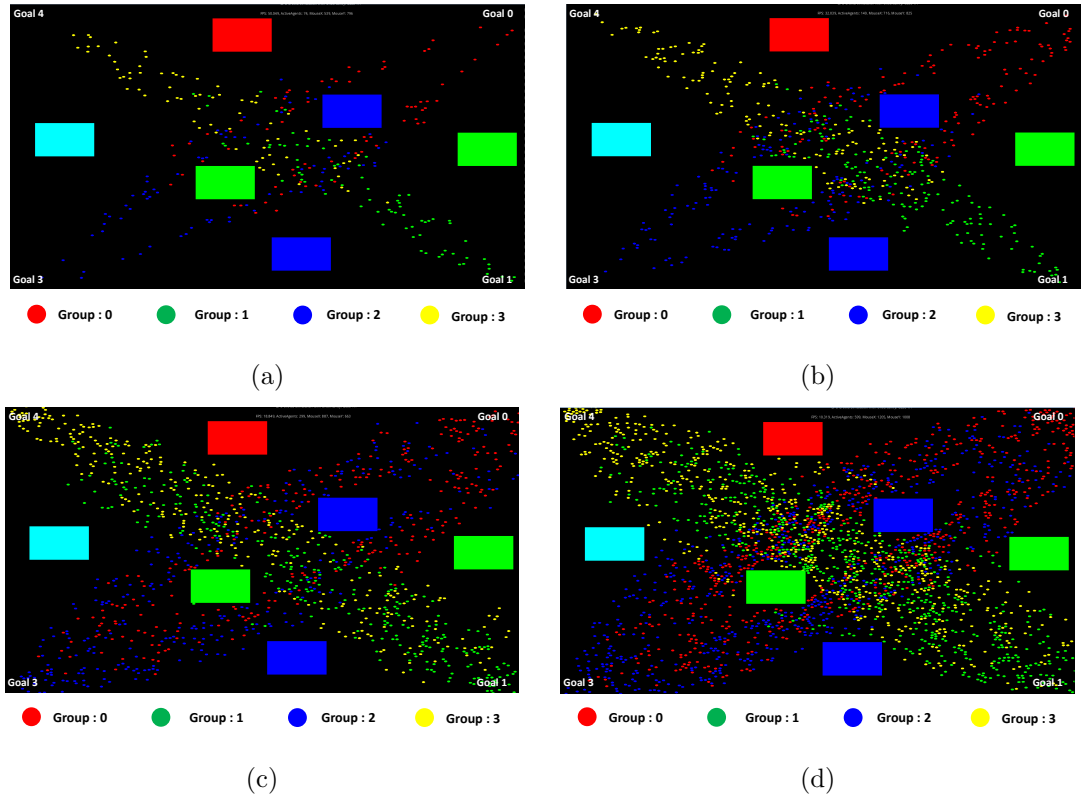


Figure 6.4: Multi-scale four corner continuum crowd with obstacles : *(a) 300 agents, (b) 600 agents, (c) 1200 agents and (d) 2400 agents.*

6.2.2 Two-level Eulerian Uncertainties

Some realistic crowd behavior cannot be simulated simply relying on Continuum crowd model. One of the example is emergent event, which should draw people's attention and thereafter temporarily change their goal-directed behavior. Two Eulerian uncertainties are used in our simulation: (see visualization in Fig. 6.5)

1. Eulerian uncertainty radiant from source of the emergent event to attract agents within a certain scope based on probability.

2. Second Eulerian uncertainty to affect agents stop walking and start watching adjacent to source of the event.

6.3 3D Rendering Result

In this section, a few 3D rendering results with character animations are presented as follows. (Fig. 6.6 and Fig. 6.7)

6.4 Performance of Simulation

We measured the rendering performance of the four demo scenarios in terms of frame rate. All simulations ran on a Intel Core I7-3820 3.6GHz with a SLI NVidia Geforce GTX 770 graphics card. Both the 2D simulation and 3D rendering are implemented purely in GPU, hence you can assume there is no frequent data transfer back from

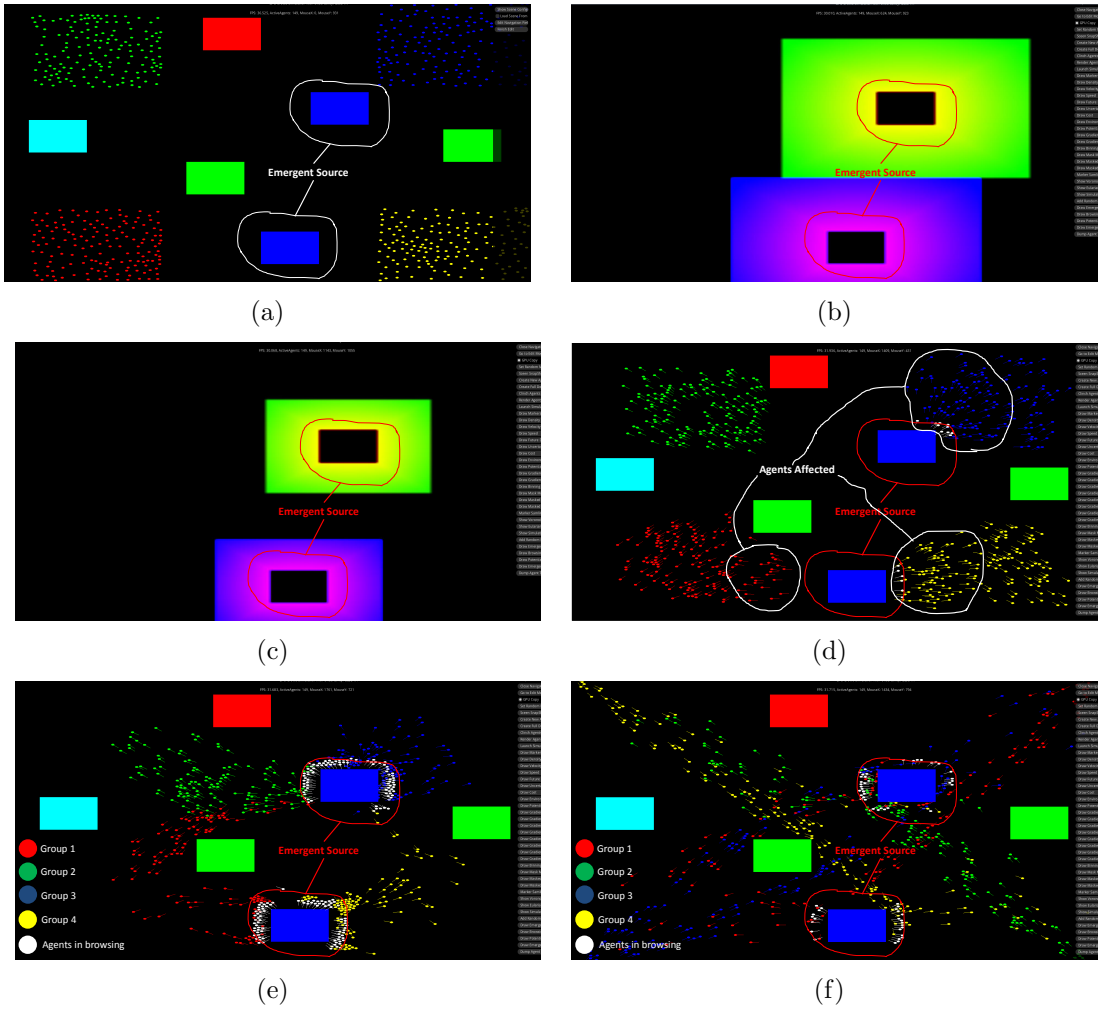
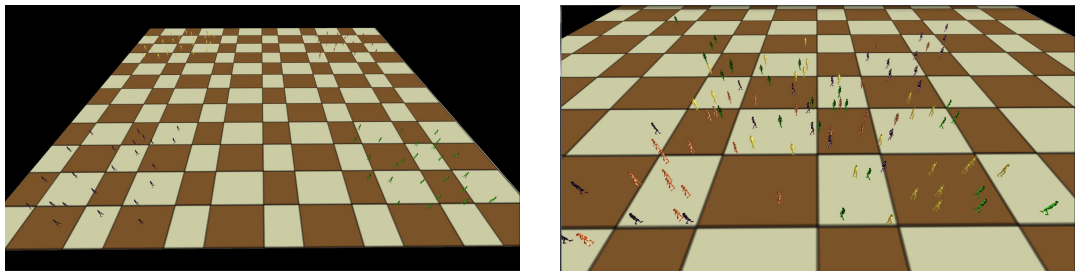
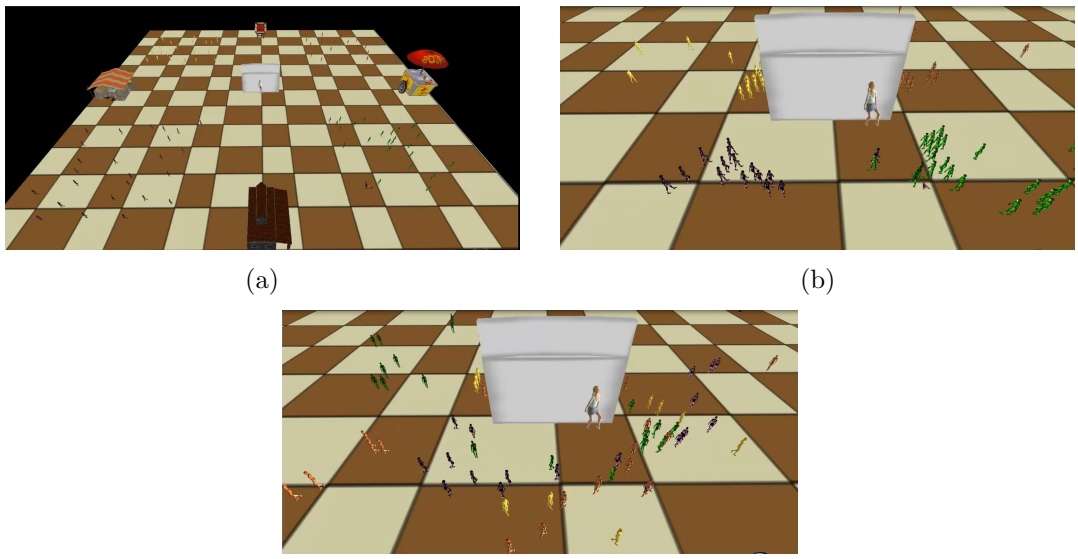


Figure 6.5: Simulation of Eulerian uncertainty with continuum crowd: *(a) Two Sources having emergent event, (b) Goal attraction uncertainty, (c) Stop browsing uncertainty, (d) agents's goal affected if closer to source, (e) uncertainty stop agents and start browsing and (f) agents go back to normal behavior after ending browsing.*



(a) (b)

Figure 6.6: 3D Animation and Rendering for continuum crowd only: *(a) Scene layout before launching the simulation, (b) Simulation in crowd traffic region*



(a) (b)

(c)

Figure 6.7: 3D Animation and Rendering for Eulerian uncertainty: *(a) scene layout with a show event (girl dancing) in middle, (b) two-level uncertainty draw agents closer and stop to browse, (c) agents walk away when finish watching.*

Table 6.1: Performance comparison

Scene	Agents	Potentials	Global Planner	Average FPS
Four Corners (FC)	**100**	**4**	**Continuum crowd**	**52.5**
FC	**100**	**8**	**Continuum crowd**	**49.5**
FC + Eulerian	**100**	**6**	**Continuum crowd**	**48.5**
Four Corners	**400**	**4**	**Continuum crowd**	**40.2**
FC + Eulerian	**400**	**8**	**Continuum crowd**	**36.7**
Four Corners	**800**	**4**	**Continuum crowd**	**25.5**
FC + Eulerian	**800**	**6**	**Continuum crowd**	**21.8**
Four Corners	**1200**	**4**	**Continuum crowd**	**18.9**
Four Corners	**2400**	**4**	**Continuum crowd**	**9.8**
Four Corners	**4000**	**4**	**Continuum crowd**	**5.75**

GPU to CPU. The screen resolution is 1920∗1080.

We illustrate the performance testing result in Table 6.1, differed in number of agents and major computation steps.From this table, we can see performance mainly depends on number of potential field needed recomputing each time-step as well as total number of agents. Since we use GPU vector packing for computing, performance only differed after number of potential fields quadrupled. Eulerian uncertainty required more layer of potential field than regular pipeline. Performance is dropped down a little bit. Overally, our simulation system can support thousands of agents with multiple groups.

Chapter 7

Conclusion

In this paper we presented a novel hybrid crowd-simulation framework to simulate Heterogeneous Crowd Behaviors. More specifically, we develop the Eulerian uncertainty field to overcome the goal-driven limitation from the original continuum crowd model. Adaptive use of local planner in our framework also can compensate for the lack of individual variety of the original model. Local planner played the call as people are tightly packed and therefore get stuck while still respecting the inertia, which is also missing from the original model. We further proposed a simple non-goal-directed navigation field to apply only certain part of the whole domain. We have successfully demonstrated that this special design, despite simple, yet quite effective to resolve congestion. Since navigation field is only applied to sub-domain, our approach avoid the limitation that navigation field is not suited for dense scenario. In our framework, we simulated random walkers through combination of local planner and Lagrangian Uncertainty. Uncertainty of different granularity is unified

into our hybrid crowd simulation framework.

Our paper also provided a improved Voronoi-based collision avoidance method suitable for GPU-acceleration based on paper [13], which we assume it can theoretically guarantee collision-free for all agents under different scale of simulation. Another advantage of our collision avoidance method is to beat the agent-sized grid limitation of effective continuum crowd implementation.

As for the implementation, we use fully GPU implementation(graphics pipeline plus computing pipeline) to handle the computation overhead. Specifically, a well-designed GPU implementation can overcome the performance bottleneck caused by large number of groups. Compared to [43] and [28], we are not decoupling the simulation and visualization.

Our user-guidance along with simple navigation field can be very hard to apply for very complicated shape of narrow geometries. However, it might be possible to combine the following behavior technique [23] with our approach to handle the narrow passage with special shape. Our approach can simulate medium-sized crowds, might not be well suited for simulating hundreds of thousands of agents. This limitation is caused by using the GPU discrete voronoi diagram, but it might be overcame by using super-resolution render targets. Also, the quality of our collision detection scheme will be deteriorated if too less markers are used due to performance concern. Even though our approach can simulate some typical social behaviors, it currently may not be robust enough to have all kinds of real-word crowd behaviors. The performance will be impacted with the increase of group numbers and number of non-goal-orientated agents.

There are several possible improvements for this work in the future. To generate more smooth advection path, we can incorporate anisotropic costs as in [28] and [4]. Then we can decrease the grid resolution for performance gains while still maintain the same quality of path smoothness. More complicated social group behavior like gathering in transitional stop can be simulated by improved Eulerian Uncertainty field. Our model also can be adapted to simulate emergency gathering by extending the state-machine for Lagrangian Uncertainty. Our framework is also open for using other global planner except the continuum crowd as well as other local collision avoidance method if only the method has high potential for being parallelized.

Bibliography

[1] A. d. L. Bicho, R. A. Rodrigues, S. R. Musse, C. R. Jung, M. Paravisi, and L. P. Magalhães. Simulating crowds based on a space colonization algorithm. *Computers & Graphics*, 36(2):70–79, 2012.

[2] A. Braun, S. R. Musse, L. P. L. de Oliveira, and B. E. Bodmann. Modeling individual behaviors in crowd simulation. In *Computer Animation and Social Agents, 2003. 16th International Conference on*, pages 143–148. IEEE, 2003.

[3] O. C. Cordeiro, A. Braun, C. Silveria, S. Musse, and G. Cavalheiro. Concurrency on social forces simulation model. In *First International Workshop on Crowd Simulation*, volume 46, 2005.

[4] D. Ferguson and A. Stentz. Field d*: An interpolation-based path planner and replanner. In *Robotics Research*, pages 239–253. Springer, 2007.

[5] R. Gayle, W. Moss, M. C. Lin, and D. Manocha. Multi-robot coordination using generalized social potential fields. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 106–113. IEEE, 2009.

[6] A. Golas, R. Narain, and M. Lin. Hybrid long-range collision avoidance for crowd simulation. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pages 29–36. ACM, 2013.

[7] Q. Gu and Z. Deng. Context-aware motion diversification for crowd simulation. *Computer Graphics and Applications, IEEE*, 31(5):54–65, 2011.

[8] Q. Gu and Z. Deng. Formation sketching: an approach to stylize groups in crowd simulation. In *Proceedings of Graphics Interface 2011*, pages 1–8. Canadian Human-Computer Communications Society, 2011.

[9] Q. Gu and Z. Deng. Generating freestyle group formations in agent-based crowd simulations. *IEEE Comput. Graph. Appl.*, 33(1):20–31, Jan. 2013.

[10] D. Helbing, L. Buzna, A. Johansson, and T. Werner. Self-organized pedestrian crowd dynamics: Experiments, simulations, and design solutions. *Transportation science*, 39(1):1–24, 2005.

[11] D. Helbing, I. Farkas, and T. Vicsek. Simulating dynamical features of escape panic. *Nature*, 407(6803):487–490, 2000.

[12] D. Helbing and P. Molnar. Social force model for pedestrian dynamics. *Physical review E*, 51(5):4282, 1995.

[13] K. E. Hoff III, J. Keyser, M. Lin, D. Manocha, and T. Culver. Fast computation of generalized voronoi diagrams using graphics hardware. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 277–286. ACM Press/Addison-Wesley Publishing Co., 1999.

[14] R. L. Hughes. The flow of human crowds. *Annual review of fluid mechanics*, 35(1):169–182, 2003.

[15] W.-K. Jeong, P. T. Fletcher, R. Tao, and R. T. Whitaker. Interactive visualization of volumetric white matter connectivity in dt-mri using a parallel-hardware hamilton-jacobi solver. *Visualization and Computer Graphics, IEEE Transactions on*, 13(6):1480–1487, 2007.

[16] W.-k. Jeong and R. Whitaker. A fast eikonal equation solver for parallel systems. In *SIAM conference on Computational Science and Engineering*, 2007.

[17] W.-K. Jeong and R. T. Whitaker. A fast iterative method for eikonal equations. *SIAM Journal on Scientific Computing*, 30(5):2512–2534, 2008.

[18] A. Kirchner, K. Nishinari, and A. Schadschneider. Friction effects and clogging in a cellular automaton model for pedestrian dynamics. *Physical review E*, 67(5):056122, 2003.

[19] A. Kirchner and A. Schadschneider. Simulation of evacuation processes using a bionics-inspired cellular automaton model for pedestrian dynamics. *Physica A: Statistical Mechanics and its Applications*, 312(1):260–276, 2002.

[20] T. I. Lakoba, D. J. Kaup, and N. M. Finkelstein. Modifications of the helbing-molnar-farkas-vicsek social force model for pedestrian evolution. *Simulation*, 81(5):339–352, 2005.

[21] F. Lamarche and S. Donikian. Crowd of virtual humans: a new approach for real time navigation in complex and structured environments. In *Computer Graphics Forum*, volume 23, pages 509–518. Wiley Online Library, 2004.

[22] K. H. Lee, M. G. Choi, Q. Hong, and J. Lee. Group behavior from video: a data-driven approach to crowd simulation. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 109–118. Eurographics Association, 2007.

[23] S. Lemercier, A. Jelic, R. Kulpa, J. Hua, J. Fehrenbach, P. Degond, C. Appert-Rolland, S. Donikian, and J. Pettré. Realistic following behaviors for crowd simulation. In *Computer Graphics Forum*, volume 31, pages 489–498. Wiley Online Library, 2012.

[24] J. L. Mitchell. Poisson shadow blur. *ShaderX3: Advanced Rendering with DirectX and OpenGL*, pages 403–410, 2005.

[25] S. R. Musse and D. Thalmann. A model of human crowd behavior: Group inter-relationship and collision detection analysis. In *Computer Animation and Simulation97*, pages 39–51. Springer, 1997.

[26] R. Narain, A. Golas, S. Curtis, and M. C. Lin. Aggregate dynamics for dense crowd simulation. In *ACM Transactions on Graphics (TOG)*, volume 28, page 122. ACM, 2009.

[27] J. Ondřej, J. Pettré, A.-H. Olivier, and S. Donikian. A synthetic-vision based steering approach for crowd simulation. *ACM Trans. Graph.*, 29(4):123:1–123:9, July 2010.

[28] S. Patil, J. Van Den Berg, S. Curtis, M. C. Lin, and D. Manocha. Directing crowd simulations using navigation fields. *Visualization and Computer Graphics, IEEE Transactions on*, 17(2):244–254, 2011.

[29] 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.

[30] N. Pelechano, J. M. Allbeck, and N. I. Badler. Virtual crowds: Methods, simulation, and control. *Synthesis Lectures on Computer Graphics and Animation*, 3(1):1–176, 2008.

[31] N. Pelechano, K. O'Brien, B. Silverman, and N. Badler. Crowd simulation incorporating agent psychological models, roles and communication. Technical report, DTIC Document, 2005.

[32] J. Pettré, P. d. H. Ciechomski, J. Maïm, B. Yersin, J.-P. Laumond, and D. Thalmann. Real-time navigating crowds: scalable simulation and rendering. *Computer Animation and Virtual Worlds*, 17(3-4):445–455, 2006.

[33] J. Pettre, J.-P. Laumond, and D. Thalmann. A navigation graph for real-time crowd animation on multilayered and uneven terrain. In *First International Workshop on Crowd Simulation*, volume 43, page 194. Citeseer, 2005.

[34] C. W. Reynolds. Flocks, herds and schools: A distributed behavioral model. In *ACM SIGGRAPH Computer Graphics*, volume 21, pages 25–34. ACM, 1987.

[35] C. W. Reynolds. Steering behaviors for autonomous characters. In *Game developers conference*, volume 1999, pages 763–782, 1999.

[36] G. Riguer, N. Tatarchuk, and J. Isidoro. Real-time depth of field simulation. *ShaderX2: Shader Programming Tips and Tricks with DirectX*, 9:529–556, 2003.

[37] T. Sakuma, T. Mukai, and S. Kuriyama. Psychological model for animating crowded pedestrians. *Computer Animation and Virtual Worlds*, 16(3-4):343–351, 2005.

[38] W. Shao and D. Terzopoulos. Autonomous pedestrians. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 19–28. ACM, 2005.

[39] J. Shopf, J. Barczak, C. Oat, and N. Tatarchuk. March of the froblins: simulation and rendering massive crowds of intelligent and detailed creatures on gpu. In *ACM SIGGRAPH 2008 Games*, pages 52–101. ACM, 2008.

[40] J. Shopf, C. Oat, and J. Barczak. Gpu crowd simulation. *ACM Transactions on Graphics, Siggraph Asia*, 27(2008):0, 2008.

[41] A. Sud, R. Gayle, E. Andersen, S. Guy, M. Lin, and D. Manocha. Real-time navigation of independent agents using adaptive roadmaps. In *Proceedings of the 2007 ACM symposium on Virtual reality software and technology*, pages 99–106. ACM, 2007.

[42] F. Tecchia, C. Loscos, R. Conroy-Dalton, and Y. Chrysanthou. Agent behaviour simulator (abs): A platform for urban behaviour development. 2001.

[43] A. Treuille, S. Cooper, and Z. Popović. Continuum crowds. In *ACM Transactions on Graphics (TOG)*, volume 25, pages 1160–1168. ACM, 2006.

[44] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha. Reciprocal n-body collision avoidance. In *Robotics Research*, pages 3–19. Springer, 2011.

[45] J. van den Berg, S. J. Guy, J. Snape, M. C. Lin, and D. Manocha. Rvo2 library: Reciprocal collision avoidance for real-time multi-agent simulation, 2011.

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

[47] J. van den Berg, S. Patil, J. Sewall, D. Manocha, and M. Lin. Interactive navigation of multiple agents in crowded environments. In *Proceedings of the 2008 symposium on Interactive 3D graphics and games*, pages 139–147. ACM, 2008.