Motion Planning for Human Crowds: from Individuals to Groups of Virtual Characters

Cover design by Ioannis Karamouzas

ISBN/EAN: 978-90-393-5786-6

Copyright © 2012 by Ioannis Karamouzas. All rights reserved.

Motion Planning for Human Crowds: from Individuals to Groups of Virtual Characters

Bewegingsplanning voor Mensenmenigten:
van Individuen tot Groepen van Virtuele Karakters

(met een samenvatting in het Nederlands)

Proefschrift

ter verkrijging van de graad van doctor aan de Universiteit Utrecht op gezag van de rector magnificus, prof. dr. G.J. van der Zwaan, ingevolge het besluit van het college voor promoties in het openbaar te verdedigen op woensdag 29 augustus 2012 des middags te 4.15 uur

door

Ioannis Karamouzas

geboren op 6 november 1982
te Athene, Griekenland

Promotor: Prof. dr. M.H. Overmars
Co-promotor: Dr. ir. A.F. van der Stappen

This research has been supported by the GATE project, funded by the Netherlands Organization for Scientific Research (NWO) and the Netherlands ICT Research and Innovation Authority (ICT Regie).

Contents

1 Introduction **1**
- 1.1 Crowd Simulation . . . 3
  - 1.1.1 Agent-Based Methods . . . 3
  - 1.1.2 Flow-Based Methods . . . 5
- 1.2 Navigation . . . 7
  - 1.2.1 Global Planning . . . 10
  - 1.2.2 Local Collision Avoidance . . . 12
- 1.3 Thesis Outline . . . 14

I Path Planning **17**

2 Adding Variation to Path Planning **19**
- 2.1 Related Work . . . 20
- 2.2 Corridor Map Method . . . 21
  - 2.2.1 Corridor Map . . . 21
  - 2.2.2 Path Planning Using Corridors . . . 23
- 2.3 Overall Problem Description and Approach . . . 24
- 2.4 Variation of Paths Using Perlin Noise . . . 24
  - 2.4.1 Implementation . . . 24
  - 2.4.2 Results . . . 27
- 2.5 Alternative Paths Using Fixed Bias . . . 29
  - 2.5.1 Lane Formation . . . 29
  - 2.5.2 Path Following . . . 30
  - 2.5.3 Results . . . 31
- 2.6 Variation of Speed . . . 32
- 2.7 Conclusions . . . 33

3 Indicative Routes for Path Planning and Crowd Simulation **35**
- 3.1 Limitations of the CMM . . . 35
- 3.2 Limitations of the Corridor Map . . . 37
- 3.3 Overview of the IRM . . . 38
  - 3.3.1 Problem Formulation . . . 38
  - 3.3.2 Overall Approach . . . 39
- 3.4 Computing Corridors . . . 39
  - 3.4.1 Explicit Corridor Map . . . 40
  - 3.4.2 Explicit Corridor . . . 41

3.4.3 Retraction and Corridor Construction . . . 41
3.5 Local Navigation in the IRM . . . 43
3.5.1 The Boundary Force . . . 44
3.5.2 The Steering Force . . . 45
3.5.3 Choosing an Attraction Point . . . 46
3.5.4 Path Variation . . . 47
3.5.5 Time Integration and Final Path . . . 47
3.6 Avoiding Obstacles . . . 48
3.7 Crowd Simulation in the IRM . . . 49
3.8 Experiments . . . 50
3.8.1 Experimental Setup . . . 50
3.8.2 Retraction Efficiency . . . 51
3.8.3 Quality Evaluation . . . 52
3.8.4 Performance . . . 53
3.9 Conclusions . . . 55

4 Density Constraints for Crowd Simulation 57
4.1 Related Work . . . 57
4.2 Overview . . . 58
4.3 Density Adaptation . . . 58
4.3.1 Density Maps . . . 59
4.3.2 Computing a Global Route . . . 59
4.3.3 Local Navigation and Final Motion . . . 60
4.4 Experiments and Discussion . . . 61
4.5 Conclusions . . . 64

5 Space-time Group Motion Planning 67
5.1 Related Work . . . 68
5.2 Preliminaries . . . 70
5.2.1 Duality Theory . . . 70
5.2.2 Reduced Cost . . . 71
5.3 Definitions and Background . . . 71
5.3.1 Multicommodity flow problems . . . 71
5.3.2 Problem formulation and overall solution . . . 72
5.4 Path Planning for Groups . . . 73
5.4.1 Creating a Capacitated Graph . . . 73
5.4.2 ILP Formulation . . . 76
5.4.3 Column Generation and Pathfinding . . . 77
5.4.4 Obtaining an Integral Solution . . . 81
5.5 Agent Motion Planning . . . 81
5.5.1 Constructing Lanes . . . 82
5.5.2 Trajectory Synthesis for an Agent . . . 82
5.6 Results . . . 85
5.6.1 Quality Evaluation . . . 87
5.6.2 Performance . . . 89
5.7 Conclusions . . . 90

II Local Collision Avoidance **93**

6 A Predictive Collision Avoidance Model for Multi-Agent Simulation **95**
6.1 Related Work . . . 96
6.2 Pedestrian Interactions . . . 98
6.2.1 Scanning and Externalization . . . 98
6.2.2 Personal Space . . . 98
6.2.3 Principle of Least Effort . . . 98
6.3 Multi-Agent Simulation Model . . . 99
6.4 Avoiding Collisions . . . 100
6.4.1 Collision Prediction . . . 100
6.4.2 Selecting Colliding Agents . . . 101
6.4.3 Avoidance Maneuvers . . . 101
6.4.4 Computing the Evasive Force . . . 102
6.5 Implementation . . . 103
6.5.1 Efficient Collision Prediction . . . 103
6.5.2 Adding Variation . . . 104
6.5.3 Time Integration . . . 104
6.6 Results . . . 105
6.6.1 Scenarios . . . 106
6.6.2 Quality Evaluation . . . 107
6.6.3 Performance . . . 113
6.7 Conclusions . . . 114

7 A Velocity-Based Approach for Simulating Human Collision Avoidance **115**
7.1 Related Work . . . 116
7.2 Human Locomotion and Collision Avoidance . . . 118
7.3 Experimental Analysis . . . 119
7.4 Multi-Agent Collision Avoidance Model . . . 123
7.4.1 Basic Approach . . . 123
7.4.2 Resolve Threatening Collisions . . . 126
7.4.3 Avoiding Static Obstacles . . . 126
7.5 Results . . . 127
7.5.1 Quality Evaluation . . . 128
7.5.2 Performance Analysis . . . 131
7.6 Conclusions . . . 131

8 Simulating and Evaluating the Local Behaviour of Small Pedestrian Groups **133**
8.1 Related Work . . . 134
8.2 Definitions and Background . . . 135
8.2.1 Problem Formulation . . . 136
8.2.2 Overall Approach . . . 137
8.3 Avoidance Maneuvers for the Group . . . 137
8.4 Avoidance Maneuvers for the Members . . . 142
8.4.1 Computing a Desired Velocity . . . 142
8.4.2 Local Collision Avoidance . . . 145
8.5 Results . . . 146
8.5.1 Scenarios . . . 146
8.5.2 Quality Evaluation . . . 148
8.5.3 Data-driven Validation . . . 149

8.5.4 Performance 152
8.6 Discussion 154
8.6.1 Model Parameters 154
8.6.2 Comparisons 154
8.6.3 Limitations 155
8.7 Conclusions 156

9 Conclusion **157**
9.1 Contributions 157
9.2 Future Work 158

Bibliography **163**

Publications **178**

Samenvatting **180**

Acknowledgments **183**

Curriculum vitae **185**

Chapter 1

Introduction

In 1984, in his classical science-fiction novel *Neuromancer*, William Gibson introduced the concept of *cyberspace*, an interactive world where people can literally "jack in" via cables and virtually inhabit:

The screen bleeped a two-second warning.

The new switch was patched into his Sendai with a thin ribbon of fiber optics.

And one and two and–

Cyberspace slid into existence from the cardinal points. Smooth, he thought, but not smooth enough. Have to work on it. . .

Then he keyed the new switch.

The abrupt jolt into other flesh. Matrix gone, a wave of sound and color. . . She was moving through a crowded street, past stalls vending discount software, prices felt penned on sheets of plastic, fragments of music from countless speakers. Smells of free monomers, perfume, patties of frying krill. For a few frightened seconds he fought helplessly to control her body. Then he willed himself into passivity, became the passenger behind her eyes.

Her body language was disorienting, her style foreign. She seemed continually on the verge of colliding with someone, but people melted out of her way, stepped sideways, made room.

"How you doing, Case"? He heard the words and felt her form them. But the link was one-way. He had no way to reply.

Two blocks later, she was threading the outskirts of Memory Lane. Case kept trying to jerk her eyes toward landmarks he would have used to find his way. He began to find the passivity of the situation irritating.

The transition to cyberspace, when he hit the switch, was instantaneous. He punched himself down a wall of primitive ice belonging to the New York Public Library, automatically counting potential windows.....

In Neuromancer, Gibson describes a rich and detailed virtual world populated with realistic characters. Today, almost thirty years since Gibson's debut novel, interactive virtual environments have become ubiquitous in our everyday lives; computer games, animation films, online communities, as well as serious applications such as training, human factor analysis and education are just a few of the areas where immersive, real-time virtual worlds are commonly used. Such worlds, to become more lively and appealing, are typically populated by large crowds of virtual characters. One of the fundamental tasks that these characters have to perform is, on one hand, to plan their paths between different locations in the world and, on the other hand, to move toward their desired locations in a human-like manner avoiding collisions with each other and with the environment. This will be the main topic of this thesis.

The path planning problem has received considerable attention over the past thirty years and many algorithms have been devised to tackle it. These algorithms were mainly developed in the field of robotics, aiming at creating short and collision-free paths for one or a few robots having many degrees of freedom. In interactive virtual worlds, though, the requirements are different. Paths for hundreds of characters through complex environments should be planned simultaneously and in real-time using only a small percentage of the CPU time. In addition to being collision-free, the paths followed by the characters must also look plausible in order to retain the suspension of disbelief of the viewer. Such paths typically follow smooth curves, are short and keep a certain amount of clearance to obstacles.

Besides exhibiting believable path planning behaviour, the virtual characters should also be able to adapt their motions and handle dynamic interactions with each other and with moving obstacles in a life-like way. This problem is very challenging, since people are accustomed to real-world situations and thus they can easily detect inconsistencies and artifacts in a simulated crowd. In addition, real humans exhibit behaviours of enormous complexity and diversity making their simulation a rather difficult task. Research in the fields of robotics, computer graphics, civil and traffic engineering, social sciences and psychology has led to many interesting models for simulating individuals, groups and crowds of characters. Although current state-of-the-art techniques have significantly improved the way that virtual humans behave and interact, we are still far from the level of realism that Gibson envisioned.

In general, realistic visual simulation of human crowds involves various components, including cognitive modeling, modeling of pedestrian and group dynamics, path planning, collision avoidance, motion synthesis and graphical rendering. In this thesis, we restrict ourselves to real-time path planning approaches for computing the global motions of multiple characters and to algorithms for solving the local interactions between individuals as well as groups of virtual characters. For those readers interested in an comprehensive discussion of virtual crowds, we highly recommend the excellent surveys of Schreckenberg and Sharma [189], Thalmann *et al.* [214], Pelechano *et al.* [166], Pettré *et al.* [171] and Huerre *et al.* [89].

The thesis is divided into two parts. The first part focuses on planning high-quality paths in interactive virtual worlds. Here, we will use the Corridor Map Method (CMM) that was developed at Utrecht University as the algorithmic basis for computing such paths. In an off-line construction phase, the CMM creates a network of collision-free corridors around the medial axis of the environment. In the query phase, paths can be computed inside the corridors providing the required flexibility for the characters to avoid collisions with dynamic obstacles. Based on the CMM, we first present a simple, yet effective, method for creating variants of paths that a character can follow within a given corridor. This not only provides a more challenging and less predictable opponent for the player in a (serious) game, but also enhances the realism of the gaming and/or training experience allowing the virtual worlds to look more lively. We then extend the CMM and introduce the Indicative Route Method (IRM) as a fast and flexible path planning approach in real-time virtual environment applications and games. In the IRM, an indicative route determines the preferred (global) route of the character, whereas a corridor around this route is used to handle a broad range of path planning issues, such as avoiding characters and computing high-quality paths. The method keeps the advantages of the CMM, but provides a much more natural steering mechanism that leads to smoother paths. It also allows more flexibility in the global behaviour of the characters, as the indicative routes no longer have to follow the medial axis of the environment. At the end of the first

part of the thesis, we will discuss two techniques to automatically determine indicative routes for crowd simulation purposes. Notably, we combine the IRM with an A*-based approach that takes crowd density into account in order to compute visually compelling crowd flows. We also present a technique that is based on Integer Linear Programming to efficiently choreograph through space-time the indicative routes of multiple groups of virtual humans.

The second part of the thesis focuses on simulating the local behaviour of individuals and groups of virtual characters. We first present a force-based model for solving interactions between virtual pedestrians having converging trajectories. Similar to real people, each virtual pedestrian predicts possible future collisions with other pedestrians and then makes an efficient move to avoid them. The method is fast, simple to implement and exhibits important emergent phenomena that have been noted in the pedestrian literature. However, like any other method based on (social) forces, it is difficult to control and tune, especially when attempting to steer thousands of characters through the virtual world. To remedy this, we also address the issue of realistic collision avoidance among virtual humans using a velocity-based model. Our velocity formulation is elaborated from experimental interaction data between pairs of pedestrians and is based on the simple hypothesis that pedestrians try to resolve collisions long in advance by slightly adapting their direction and/or speed. We further extend our technique to simulate small groups of virtual humans. Our proposed model is based on empirical observations regarding the spatial organization of real pedestrian groups and is complementary to existing methods for solving interactions between virtual characters. It considers pairs and triples of characters and uses a two-step algorithm to ensure that the groups will safely reach their goals, forming at the same time walking patterns similar to the ones observed in real life.

In the remainder of the introduction, we survey related work in path planning and crowd simulation and provide an overview of the thesis.

1.1 Crowd Simulation

Many models have been developed over the past twenty years for simulating the movements and dynamics of individuals, groups and crowds of virtual characters. We refer the readers to the book of Thalmann and Musse [215] and a recent survey of Pelechano *et al.* [166] for an extensive overview. At a broad level, these models can be classified into *microscopic* or *agent-based* and *macroscopic* or *flow-based* methods. Microscopic models focus on individual characters and study how these characters behave and interact with other characters in a crowd. In contrast, macroscopic models focus on the behaviour of the crowd as a whole. Here the emphasis is on the characteristics of the flow rather than on the individual variability.

1.1.1 Agent-Based Methods

The most common way to model the locomotion of human crowds is with agent-based methods. These approaches can capture the individual variability of real people yielding to the simulation of complex heterogeneous motions. An agent can be seen as a dynamic entity that makes independent decisions. It has distinct characteristics and goals and plans individually its own movements by continuously perceiving its surroundings and (re)acting accordingly.[1] Agent-based methods can be traced back to the seminal work of

[1] Throughout this thesis, the terms agent and character will be used interchangeably.

Reynolds on *flocking* [181]. Reynolds used simple local rules to create visually compelling flocks of birds and schools of fishes. In his model, "boids" (flock members) steer themselves in such a way that they avoid collisions with other boids, while at the same time they try to align themselves with nearby flock mates and to stay close to them. Later, Reynolds extended this technique to incorporate additional steering behaviours for autonomous agents [182].

Since his original work, many approaches in the graphics and animation community have focused on modeling human crowds using *rule-based* techniques. Funge *et al.* [53], for example, combined rule-based behaviours with cognitive models to simulate highly sophisticated agents that are able to learn from the environment. Recently, Shao and Terzopoulos [193] extended this model and integrated motor, perceptual, behavioural, and cognitive components in order to generate an artificial life model of autonomous pedestrians. Musse and Thalmann [150] simulated the collective behaviour of groups of virtual humans in a crowd using concepts from sociology, whereas Pelechano *et al.* [165] combined psychological and geometrical rules with physical forces to control individual agents in dense crowds. Rule-based methods have also been integrated into a number of commercially available systems for crowd simulation. The *Massive Software* [138], for example, which was used in *The Lord Of The Rings* film trilogy, combines simple rules with fuzzy logic and allows the animator to author the behaviour of each individual agent within the crowd. In general, rule-based approaches are able to simulate complex, life-like behaviours and can be used to create realistic crowd motions. However, in most of these approaches, it is often difficult to design rules that consistently generate desired crowd behaviours. This problem becomes more obvious in complex and dynamic environments, as careful parameter tuning is usually required to obtain a satisfactory simulation result.

Cellular Automata (CA) methods are closely related to rule-based methods. Over the past 60 years, CA have been widely studied in many fields with important implications for physics, mathematics, biology, computer science and economics (see the detailed survey of Wolfram [241] and his subsequent book [242] for an extensive overview). In the last decade, CA have also been applied to model the motions of multiple agents for crowd simulation [136, 212] and evacuation planning [17, 106, 205]. In these approaches, the space is discretized into a regular grid of cells, where the state of each cell is uniquely specified by one or more discrete variables. The automaton evolves through a discrete number of time steps. At each time step, each agent occupies a cell and can only move to an adjacent cell that is not occupied by other agents or obstacles. The locations of the agents are updated simultaneously based on a set of rules that take into account the current state of each agent's cell and the states of its neighbouring cells. The main advantage of using CA for simulating virtual crowds is their inherent simplicity. The motions of the agents are governed using a small set of rules. Due to their simplicity, CA are also fast to compute. Furthermore, they are capable of reproducing important emergent phenomena, such as the formation of lanes in counterflows [17]. However, due to the discrete nature of these methods, the spatial movements of the agents are limited to adjacent free cells. This assumption can lead to unrealistic simulations, especially when complex inter-agent interactions must be addressed. In addition, cellular automata cannot adapt well to dense crowds, since the motions of the agents are not continuous and no physical contact is allowed between the agents.

Agent-based modeling has also been studied in civil and traffic engineering. The most influential work in this field is the *social force model* developed by Helbing and his colleagues [74, 77]. Helbing used forces related to physics to describe the social interac-

tions between pedestrians. Based on this work, a number of models have been proposed in transportation research and computer graphics to simulate crowds of pedestrians under normal and emergency situations (e.g. [14, 28, 76, 147, 165, 213]). Social force models are able to capture the emergence of empirically observed crowd phenomena, such as the formation of lanes in opposing flows, appearance of vortices in crossing flows, arching and clogging behaviour at narrow exits, etc. [74, 79, 85]. Closely related to social force models are *particle-based* systems, where physical forces act on the particles and influence their movements. Such systems have also been used to model crowds of virtual characters [13, 72], as well as groups with significant motion dynamics [16]. Nevertheless, all these approaches that are based on (social) forces cannot produce the correct microscopic behaviour of the pedestrians. Due to lack of anticipation and prediction, the virtual pedestrians interact when they get sufficiently close, which results in unrealistic motions and oscillatory behaviour. Furthermore, similar to rule-based techniques, force-based approaches are hard to control and tune, especially when dealing with crowds of high density.

Recently, *example-based* (or data-driven) techniques have also been explored for multi-agent simulation. These approaches use example behaviours from video or motion capture data to drive the simulation of virtual characters. In the work of Lerner *et al.* [128], a database of human trajectories is learnt from video recordings of real pedestrian crowds. During a simulation, each autonomous agent reacts to its perceived state by searching the database and selecting a behaviour (trajectory) that closely matches the situation that the agent is facing. Similarly, Lee *et al.* [126] used a regression-based learning algorithm in order to synthesize realistic group behaviours from crowd videos, whereas Lai *et al.* [117] trained a group behaviour model using the flocking model of Reynolds. More recently, Kyriakou and Chrysanthou [116] presented a novel data-driven approach for generating new crowd behaviour based on texture synthesis principles. Finally, Ju *et al.* [97] have introduced a technique that synthesizes crowds of any size and any length by reconstructing and interpolating spatio-temporal behaviours from simulated and captured crowd data. The main advantage of all these example-based approaches is their capability of reproducing complex and subtle behaviours that would be difficult to model using other types of agent-based solutions. However, their applicability is limited by the size and type of the input crowd data. In addition, these approaches are too computationally expensive for real-time interactive applications and are primarily designed for offline simulations of crowds.

1.1.2 Flow-Based Methods

Flow-based approaches attempt to directly govern the macroscopic (global) behaviour of large crowds. In contrast to agent-based approaches, they favour a broader view of the crowd and do not focus on the behavioural characteristics of the individuals. Consequently, in such approaches, emergent crowd phenomena are more important to preserve than the individuality of the agents.

Macroscopic simulation has paramount importance in civil and traffic engineering for assessing the quality and the design of pedestrian infrastructure facilities (in terms of throughput, walking times, comfort level and safety), as well as for managing pedestrian flows in such facilities. Therefore a number of simulation models have been proposed over the years, for example *regression* [144], *queuing* [137] and *route choice* models [86]. Nevertheless, none of these approaches can take into account the self-organization phenomena that spontaneously appear in pedestrian crowds.

An alternative approach to crowd flow modeling has emerged from fluid mechanics. Henderson [80] hypothesized that the flow of pedestrian crowds is similar to that of gases or fluids and described the motions of pedestrians using partial differential equations. Later, Helbing and his colleagues [78, 79] empirically confirmed this hypothesis. According to their observations, at low densities people in a crowd behave more or less like atoms or molecules in a gas, at moderate densities the crowd flow resembles fluid flow and at high densities the motion of pedestrian crowds can be compared to granular flow.

In the early 1990's, Helbing also improved Henderson's fluid-dynamic approach and proposed a gas-kinetic model that takes into account the avoidance and deceleration maneuvers of the pedestrians due to their particular interactions [73]. Since a numerical solution of the resulting gas-kinetic equations is very difficult to obtain, Hoogendoorn and Bovy [83] used a novel particle discretization approach and devised a discrete gas-kinetic model for simulating pedestrian flows. More recently, Hughes [90, 91] interpreted a human crowd as a flow of "thinking fluids" and introduced a *continuum* theory for the flow of pedestrians. His continuum formulation represents the pedestrians as a continuous density field and uses a pair of non-linear, time-dependent, partial differential equations to guide the pedestrians to their objectives while avoiding collisions. Hughes's method is mainly suited for high density crowds and has been confirmed with real crowd data [82]. It has also been successfully used for medieval battle analysis [25].

Concepts of fluid mechanics have also been adopted for directing simulations of virtual crowds in computer graphics and animation. In this community, perhaps the most intuitive way to guide the motions of virtual agents is using *velocity fields*. A velocity field (also known as vector field or flow field) defines a mapping from a point in the virtual world to a velocity vector that indicates the speed and direction at which an agent needs to move if it finds itself at that point. Such fields are typically defined over the floor plan of the environment and can be either automatically computed or manually designed to encourage certain behaviour. Courty and Corpetti [29] estimated a time series of velocity fields from dense crowd videos and then synthesized a new crowd by advecting virtual characters along this time varying flow. Similarly, Musse *et al.* [149] computed extrapolated velocity fields from filmed video sequences based on computer vision algorithms and combined them with Helbing's social force model to simulate the motion of virtual agents. Chenney [19] designed divergence-free velocity fields on a 2D regular grid in order to simulate crowds and fluid-like phenomena. The divergence-free property ensures that the agents do not require any form of explicit collision avoidance, at the expense of precluding any interaction among them. In addition, the method does not provide an interactive control over the designed velocity field and is not applicable for goal-directed navigation. The work of Jin *et al.* [96] allows the user to sketch velocities on anchor points in the virtual world and then uses an interpolation scheme to compute a continuous vector field that guides the motions of the agents. More recently, Patil *et al.* [164] proposed a novel approach to direct and interactively control virtual crowds. Their method allows the user to specify guidance fields for groups of agents by either drawing paths in the scene or importing flow fields from real crowd footage. Based on these inputs, a smooth navigation field is computed that directs the agents toward their goals. As compared to the aforementioned techniques, this work guarantees that the generated navigation vector field is goal-directed and singularity-free. Consequently, all the agents can reach their destinations without getting trapped in local minima.

Hughes's continuum theory has also been exploited for simulating virtual crowds. Treuile *et al.* [218] transformed Hughes's continuous crowd field into a particle represen-

tation and used an eikonal equation (non-linear partial differential equation) to describe the motion dynamics of the crowd. In their approach, a density-dependent velocity field is dynamically constructed by solving the eikonal equation on a regular grid. The resulting velocity field guides the agents toward their goals without the need of explicit collision avoidance. Although this approach generates a number of visually interesting and empirically-confirmed emergent phenomena, the quality of the simulations is governed by the underlying grid resolution. In addition, since solving the eikonal equation is computationally expensive, the crowd is typically organized into a small number of large homogeneous groups, where each group consists of agents with identical objectives and characteristics. More recently Nurain *et al.* [151] presented a hybrid continuum-based method that treats a crowd as a "unilaterally incompressible" fluid, that is a fluid that is neither purely incompressible, nor purely compressible. Their approach imposes a novel volumetric constraint in the continuous domain, which prevents the crowd density from increasing beyond a maximum value. This results in a flow field that handles inter-agent collisions on a macroscopic level and can be used to simulate at near-interactive rates the motions of very large, dense crowds consisting of up to a hundred thousand agents.

As compared to agent-based methods, all these flow-based approaches do not distinguish individual agents, nor do they try to describe the individual behaviour of the pedestrians. The crowd instead is characterized in aggregate terms of density and flow velocity. Flow based methods, and particularly approaches based on continuum dynamics, are well suited for simulating crowds of thousands of characters in real-time. Typically, such crowds are observed at a distance and thus, their behaviour can be efficiently modeled at a coarser level where no detailed per-agent motion is required.

In this thesis, though, we do not focus on such macroscopic simulations, where the inherent variability of individuals is a priori lacking. Instead, our main emphasis is on agent-based systems, where people (agents) have distinct characteristics and behaviours and make decisions based on personal goals. In particular, we address the problem of *multi-agent navigation and planning*, in which each agent must (independently) navigate toward its specified goal without colliding with other agents and the static part of the environment.

1.2 Navigation

In simple environments, agent-based methods are sufficient to steer the virtual characters toward their goals. The main drawback of all these methods, though, is that the agents act based only on local information. Consequently, in complex and cluttered environments, they can easily get trapped and not be able to reach their destinations. The problem is exacerbated when multiple agents with intersecting paths must simultaneously move in the virtual world. To overcome these challenges, numerous approaches have been proposed over the last ten years for multi-agent navigation and planning in virtual environments. Most of these approaches originate from earlier path planning algorithms developed in the field of robotics.

In its simplest form, the path planning problem can be defined as finding a collision-free path, traversed by a unit, between a given start and goal placement in an environment with obstacles. In robotics, this problem is normally referred to as motion planning. Here, we are given a geometric description of an environment (the workspace $\mathcal{W}$) in which a robot must move or operate among a set of obstacles. Typically, the problem is defined in the configuration space $\mathcal{C}$, which denotes the space of all possible configurations of the robot. A configuration is described using a list of parameters (such as

position, orientation, pose, etc) that correspond to the degrees of freedom of the robot's motion. The $\mathcal{C}$-space is partitioned into a set of forbidden configurations and a set of free configurations. A configuration is called *forbidden*, if it causes a collision between the robot and any of the obstacles in the workspace. The space of all forbidden configurations is called the *forbidden configuration space* $\mathcal{C}_{\text{forb}}$. Besides obstacles, $\mathcal{C}_{\text{forb}}$ also consists of configurations that correspond to robot self-collisions. The union of all collision-free configurations forms the *free configuration space* $\mathcal{C}_{\text{free}}$. The motion planning problem can now be transformed into finding a collision-free path for a point in $\mathcal{C}$-space between a given start configuration $s \in \mathcal{C}$ and a given goal configuration $g \in \mathcal{C}$. During the last three decades this problem and its variants have received considerable attention and many planners have been proposed. We refer the interested reader to the books of Latombe [119], Choset *et al.* [23], and LaValle [121] for an extensive overview.

A common way to plan a path for a robot is using a *roadmap* method. This approach captures the connectivity of the free configuration space in a roadmap (graph) and reduces the path planning problem to a graph searching problem. Given a path planning query, the start and goal configurations are connected to the roadmap and a graph searching algorithm (such as Dijkstra's shortest path algorithm [35]) is then used to find a corresponding path. One of the earliest roadmap methods is the *visibility graph*, which constructs a roadmap by connecting mutually visible obstacle vertices in the configuration space [154]. Another method is the *generalized Voronoi diagram* (or medial axis). This method decomposes the free configuration space into regions, such that all configurations in a region are closer to a particular obstacle than to any other obstacle in the environment. The boundaries of these regions maintain a maximum clearance from the obstacles and define the roadmap that is used for planning. The diagram can either be approximated [58, 81, 153, 233] or computed exactly [130, 155].

Sampling-based methods represent a specific type of roadmap planners, since a roadmap is constructed by randomly generating configurations from the $\mathcal{C}$-space. These methods are usually probabilistically complete, meaning that a solution to a path planning problem is guaranteed to be found, if the method is applied sufficiently long. The most popular one is the *Probabilistic Roadmap Method* (PRM) method [101, 160]. The PRM consists of two phases: a construction phase and a query phase. In the construction phase, a roadmap is built by randomly sampling configurations across $\mathcal{C}_{\text{free}}$. These configurations are added as nodes to the roadmap and are connected to their neighbouring configurations, if a valid path exists between them. This process is repeated until the roadmap is dense enough and satisfies some user-defined criteria. In the query phase, the start and goal configurations are added to the roadmap and the shortest path in the roadmap is obtained by running Dijkstra's algorithm. The PRM method has been successfully applied in many fields, such as robot and human motion planning, manipulation planning, protein folding, computer animation and video games. It is probably the most used motion planning approach and many variations and improvements have been made over the years.

Cell decomposition methods have also been developed for robot path planning. These approaches subdivide $\mathcal{C}_{\text{free}}$ in non-overlapping cells. The adjacency of these cells induces a connectivity graph, which can be searched to answer path planning queries. Both exact [70, 98, 190, 191] and approximate [41, 233, 249] decomposition methods have been proposed. Exact cell decomposition methods compute cells such that their union equals exactly the free configuration space $\mathcal{C}_{\text{free}}$ (e.g. convex polygons, trapezoidal), whereas approximate decomposition methods approximate $\mathcal{C}_{\text{free}}$ by using a collection of predefined cell shapes whose union is strictly included in $\mathcal{C}_{\text{free}}$ (e.g. uniform grids, quadtrees).

Potential field methods [102, 184] offer a more flexible way to solve specific planning queries. A potential field method directs the motion of the robot through an artificial potential field which is cast over the environment. The general idea here is that the robot is attracted toward its goal configuration by a strong attractive force. At the same time, the obstacles exert a repulsive force on the robot to prevent it from colliding with them. The final path can then be found by following the steepest descent of the resulting potential toward the goal. The main disadvantage of this method is that the robot can get trapped in local minima and may not be able to find a collision-free path, even if one exists [110]. To alleviate this problem, a number of approaches try to escape from such minima [7, 8, 194], whereas other solutions use more advanced potential functions with few or no local minima [103, 108]. In general, though, all these approaches cannot give any guarantees on their success.

An extension to the basic path planning problem is planning the robot's motions in dynamic environments, in which both stationary and moving obstacles are present. A straightforward solution is to replan a path when changes are observed due to motion of the dynamic obstacles [93, 131]. Efficient algorithms have been proposed over the years that are able to replan in an anytime, deterministic fashion by reusing information from previous planning episodes instead of (re)planning from scratch [109, 134, 204]. Space-time planning algorithms have also been proposed for path planning in known dynamic environments, that is in environments where the motions of the obstacles are predictable and given in advance. Such approaches plan in a joint *configuration-time space*, in which the time dimension is added to the robot's configuration space [50, 88]. A number of approaches have also tried to reduce the complexity of the planning problem by first constructing a roadmap based on the static elements of the environment, and then creating space-time nodes in a lazy way [229]. However, all these techniques are quite slow and are not suited for online real-time applications.

Multi-robot path planning has also been extensively studied in robotics, mainly for cooperative tasks. In this problem, multiple robots need to simultaneously move in an environment, while mutual collisions and collisions with the obstacles are avoided. The problem was proven to be intractable [121] and is typically handled by *centralized* planners and *decoupled* methods. Centralized planners [133, 192] compute the simultaneous motions of all robots by combining their configuration spaces into a composite one. Although these methods are complete, their running time grows exponentially with the number of robots. To overcome this problem, decoupled techniques have been developed that plan a path for each robot independently and try to coordinate the resulting motions [167, 196]. Recently, *prioritized* planning approaches have also been studied in robotics [10, 24, 39, 228]. In such approaches, a priority is assigned to each robot according to a certain priority scheme. The paths of the robots are then planned sequentially in order of decreasing priority. This reduces the multi-robot motion planning problem to the problem of motion planning for a single robot in a known dynamic environment which can be addressed by roadmap-based space-time planners [88, 229].

The multi-robot path planning problem is closely related to the multi-agent navigation problem in virtual environments. In both problems, multiple units must simultaneously navigate through the world without colliding with each other or with the static part of the environment. The main difference is that, in robotics, the focus is on different aspects of the planning problem, such as obtaining an optimal coordination among the robots. In addition, existing solutions can handle only a few robots and are not suited for real-time applications. In contrast, in interactive virtual worlds, the emphasis is on a realistic and physically correct simulation of large crowds of virtual characters. Multi-agent

navigation has been an active topic of research in the last decade and many path finding approaches have found widespread use in games, training systems and animation applications. In most of these approaches, global path planning and local collision avoidance are often decoupled. Typically, a graph or a roadmap is used to direct the global motion of each agent, whereas collisions with other agents and obstacles are resolved by locally navigating around them. In the remainder of this section, we will provide references to the most important global and local navigation methods developed in the graphics and animation community, as well as in the game industry.

1.2.1 Global Planning

Cell decomposition methods have been widely used in the game development community to plan the global paths of game-controlled entities, such as non playing characters (NPCs) and vehicles. One of the most popular approaches is to divide the world into a grid of cells and search for a free path using an A*-based algorithm on the free cells (see e.g. [33, 185]). This approach guarantees that a path, if one exists, will always be found. However, grid-based searches can become computationally expensive, especially when many paths have to be planned simultaneously in large and complicated environments. Furthermore, the paths returned by A* algorithms tend to be aesthetically unpleasant and thus, extra care should be taken at a post-processing phase to smooth them. Variants of A* algorithms have been proposed in the robotics literature [43] to overcome this limitation and compute smooth paths on a grid at the expense of additional computation time.

An alternative approach that has gained a lot of popularity over the past few years is to use a *navigation mesh* [33, 145, 179] for the global navigation of the game characters. Navigation meshes partition the terrain into convex polygons that represent the walkable area of the environment. The static part of the game world is taken into account upon the construction of the mesh, and thus, the character only has to consider collisions with dynamic obstacles. Similar to a grid-map, a navigation mesh consists of linked cells sharing common edges. Therefore, the character's path can be retrieved by applying an A*-based search algorithm.

In the graphics and animation community, sampling-based planners and probabilistic roadmaps have also been adapted for path planning of virtual characters. Bayazit *et al.* [9] have combined the PRM approach with flocking techniques in order to guide the flock members toward their goals. Sung *et al.* [209] combined probabilistic roadmaps with motion graphs [3, 111, 125] to generate goal-directed collision-free motions for large number of characters. To avoid inter-character collisions, a prioritized planning approach was used by considering characters whose motions have already been planned as moving obstacles. Although their solution produces high-quality motions, the growing complexity of the planning space limits the number of the characters that can be simulated. Similarly, Lau and Kuffner [120] precomputed search trees of animation clips generated by a finite-state machine to synthesize motions for characters navigating through complex environments. They used a grid-planner for real-time path finding and resolved collisions between the characters by planning for each individual character sequentially. Consequently, the same problem as in [209] arises and the approach cannot handle large crowds.

Recently, roadmap and cell decomposition techniques have also been used for interactive navigation and planning of multiple agents in complex and dynamic worlds. Lamarche and Donikian [118] used a constrained Delaunay triangulation to subdivide the walkable space of the environment into a set of convex cells. A hierarchical topological

structure was then built based on the spatial subdivision to enable real-time global path planning for hundreds of virtual humans. Shao and Terzopoulos [193] employed a multi-scale approach to facilitate fast and accurate path planning of crowds of autonomous pedestrians in a virtual train station. They used a topological map to capture the connectivity between different regions of the virtual worlds. For each region, a quadtree map was stored to support global long-range planning and a list of grid maps with different resolutions to support short-range detailed planning. Pettré *et al.* [170, 172] decomposed the navigable space of the environment into a set of interconnected cylinders centered on the Voronoi diagram. The resulting structure captures the environment topology in a *Navigation Graph*. A variant of Dijkstra's shortest path algorithm can then be used to efficiently solve navigation queries for large crowds consisting of thousands of pedestrians.

The *Corridor Map Method* (CMM) [57] uses a similar structure to define the walkable space of a virtual environment. In a preprocessing phase, the medial axis is approximated by exploiting graphics hardware and a high-quality roadmap is constructed. This roadmap, enhanced with clearance information, defines the *Corridor Map*. When a character has to move to a specified goal position, a *backbone path* is extracted from the Corridor Map along with a collision-free *corridor* around it. The backbone path directs the global motion of the character, whereas its local motion is controlled by a potential force-field approach inside the corridor, leading to a smooth path. In addition, the corridor gives the character flexibility to locally deviate from its global path when dynamic obstacles (or other moving characters) have to be avoided. The CMM has been successfully used to steer in real-time thousands of characters through complex virtual worlds, as well as to plan the motions of coherent groups of characters [56, 99, 153]. In the first part of the thesis, we extend the CMM and introduce the *Indicative Route Method* as a fast and flexible path planner for interactive virtual worlds and games.

Adaptive roadmaps have also been proposed for global path planning of multiple virtual agents in complex dynamic scenes. *Multiagent Navigation Graphs* (MaNG) [207] efficiently compute dynamic navigation graphs using the first and second-order Voronoi diagrams of all the obstacles and agents present in the environment. Although this approach generates paths of maximal clearance for each agent, it is limited to a few hundred of agents and cannot guarantee smooth motions. Voronoi diagrams have also been exploited in the *Adaptive Elastic ROadmap* (AERO) system [208]. AERO extends the concept of elastic bands [178] and elastic roadmaps from robotics [54, 245] and constructs a Voronoi diagram that is continuously updated in response to the motion of the agents and the other dynamic obstacles present in the environment. The technique has been combined with the generalized force model of pedestrian dynamics proposed by Helbing *et al.* [75] and can compute collision-free paths for large crowds of characters at interactive rates.

Finally, Treuille *et al.* [218] proposed a novel planning approach that is based on the continuum theory for the flow of human crowds [90, 91]. Their model discretizes the environment into a regular grid and constructs a dynamic potential field that guides the virtual humans toward their goals without colliding with each other or with other dynamic and static obstacles. Although this approach unifies global and local navigation into a single framework, its performance and accuracy depends on the underlying grid resolution. In addition, the method is mainly designed for simulating a small number of large homogeneous groups of characters moving toward common goals, and is not suited for modeling individual characters that have distinct behaviour characteristics and goals. To remedy this, a hybrid model has been recently proposed that combines the continuum crowd formulation with navigation graphs and can simulate in real-time thousands of pedestrians, organized into several groups with a large variety of goals [248].

1.2.2 Local Collision Avoidance

While following its global route, the agent should also be able to avoid collisions with nearby agents, as well as with static and dynamic obstacles that are present in the virtual world. In the last two decades, numerous agent-based methods have been proposed for local collision avoidance, including force-based approaches [72, 74, 77, 165], behavioural and rule-based models [53,136,182,193], vision techniques [156] and geometrically-based algorithms [66, 100, 163, 173, 227]. At a broad level, these methods can be classified into *reactive*, *predictive* and *example-based* methods.

In *reactive steering*, the character adapts its previously computed motion to the dynamic (other characters and moving obstacles) and static obstacles found along its path. Reactive navigation techniques originate from the robotics community and are based on variants of potential field approaches [102]. In the animation community, the concept of reactive planning was introduced by the flocking model of Reynolds [181]. In flocking, repulsive forces are exerted on each flock member by its nearby flock mates in order to prevent it from colliding with them. The distance to these neighbouring members defines the strength of the forces; the closer a flock member comes to another member, the stronger the repulsive force is. In the civil and traffic engineering community, Helbing's social force model [74,77] is probably the most popular reactive approach. It uses a mixture of socio-psychological and physical forces to describe the behaviour of a pedestrian crowd and solves Newton's equations of motion for each individual pedestrian. Interactions between pedestrians are modeled as repulsive and tangential forces that are expressed as a function of their relative distance. Interactions with static obstacles are treated analogously. Later, Pelechano *et al.* [165] extended this model in order to control individual agents in high-density crowds, whereas Heigeas *et al.* [72] proposed a particle-based approach to capture emergent crowd effects. In the latter approach, pedestrians and obstacles are represented as particles and interactions between pairs of individuals and between individuals and obstacles are modeled as a mass-spring-damper system, in which stiffness and viscosity change based on the distance between the interacting particles. Besides force-based techniques, other reactive planners have also been proposed, such as approaches that account for grid-based rules [136] and behavioural models [219]. The main drawback of all these reactive approaches, though, is that virtual agents adapt their motions to resolve collisions with other agents and obstacles when they get sufficiently close, which leads to unrealistic avoidance maneuvers. In addition, due to lack of anticipation, the resulting motions are characterized by oscillatory behaviour.

An alternative way for solving interactions between virtual humans is based on *collision prediction*; assuming that each character maintains a constant velocity, the future motions of the characters are predicted by linearly extrapolating their current velocities and then used to detect and avoid collisions in the near future. Based on this concept, Reynolds has introduced the *unaligned collision avoidance behaviour* [182], whereas Feurtey [44] devised an elegant collision detection algorithm that predicts potential collisions within time and resolves them by adapting the speed and/or the trajectories of the agents. Inspired by Feurtey's work, Paris *et al.* [163] presented an anticipative model to steer virtual pedestrians without colliding with each other. Kapadia *et al.* [100] tackled the collision prediction problem using the concept of affordances from psychology, whereas Pettré *et al.* [173] proposed an egocentric model for solving pair-interactions between virtual pedestrians based on a detailed experimental study. In their model, collisions are resolved using a combination of speed and orientation adaptations depending on the role that each pedestrian has during the interaction (passing first or giving way). More recently, Ondřej *et al.* [156] formulated the collision avoidance behaviour of virtual

humans using a combination of visual stimuli and motor response laws; a virtual pedestrian mainly adapts its orientation to avoid future collisions detected from visual stimuli, whereas a deceleration strategy prevents imminent collisions.

Closely related to the aforementioned techniques is the notion of *Velocity Obstacle* (VO) introduced in robotics by Fiorini and Shiller [45]. For a given entity, the VO represents the set of all velocities that would result in collision at some moment in time with another entity moving at a given velocity. Using the VO formulation, any number of obstacles can be avoided by considering the union of their VOs and selecting a velocity outside the combined VO. Recently, van den Berg *et al.* [227] extended the VO formulation to guarantee oscillation-free behaviour between two interacting characters and introduced the concept of *Reciprocal Velocity Obstacle* (RVO). Later, they combined this technique with a pre-computed roadmap for global path planning in order to simulate large crowds of independent agents moving through complex virtual worlds [230]. Over the past fifteen years, numerous models have been proposed based on the VO and the RVO concepts (e.g. [52, 105, 201, 239]). Among the most popular ones is the work of Guy *et al.* [66] that provides avoidance behaviour similar to the RVO, but is considerably faster as it only guarantees collision avoidance for a discrete time interval and uses a discrete optimization method to compute the maneuvers of each character. Similarly, the *Optimal Reciprocal Collision Avoidance* (ORCA) formulation allows each agent to select a reciprocally optimal collision-free velocity by solving a low dimensional linear program [226].

Recently, *example-based* techniques have also been exploited for resolving collisions between interacting virtual characters in a crowd [97, 126, 128]. In such approaches, a database of example behaviours is built from captured crowd data. During the simulation, each agent selects an example behaviour that closely matches its perceived state, which is typically represented as a set of local spatio-temporal factors, such as features of the environment and the motions of nearby agents. Example-based techniques can realistically simulate interactions between virtual humans and capture highly sophisticated and complex avoidance maneuvers exhibited by real people. However, since the example space cannot be fully covered, situations may arise where none of the matching examples can lead to a collision-free solution. Furthermore, to generate high-fidelity motions, the simulated and example crowds must be similar in nature. Simulating, for example, pedestrians interactions in a high-density crowd based on a sparse observed real crowd may lead to unnatural avoidance behaviours. Finally, most of these approaches are too computational demanding for applications such as interactive virtual worlds and are mainly used for offline crowd simulations.

In the second part of the thesis, we also propose two methods for local collision avoidance. The first approach is force-based and can be used as an alternative to existing reactive techniques, allowing compelling simulations of thousands of agents at real-time frame rates. The second model is velocity-based and derives from experimental data involving interactions between real humans in controlled scenarios. Both of our proposed techniques are predictive in nature and, hence, they bear close resemblance with several of the anticipative models mentioned above. However, they are based on the simple hypothesis that an agent resolves collisions in advance by slightly adapting its motion. Assuming that other agents also exhibit similar behaviour, interactions are solved with minimal effort resulting in a smooth and more optimal flow. We believe that these approaches, albeit not exempt from failure, better capture the avoidance behaviour of real humans.

1.3 Thesis Outline

The thesis is divided into two parts. The first part discusses path planning algorithms for computing the global motions of individuals, groups and crowds of virtual characters in real-time, interactive virtual worlds. It consists of four chapters.

In Chapter 2, we present three simple techniques for creating variants of homotopic paths that characters can follow given a path planning query. Existing path finding approaches generally produce the same path for the same input parameters, i.e., for the same start and goal positions. In virtual environment applications, though, such behaviour can easily break the suspension of disbelief of the user/viewer. An individual does not always take the exact same route when solving a typical path planning problem in real life. In addition, different people may follow slightly different paths, even if their start and goal positions are relatively similar. Consequently, for virtual characters to be believable and acceptable they should also exhibit similar type of behaviour. We show that such path variation can be easily obtained by exploiting the Corridor Map Method (CMM) and locally modifying paths that run along the medial axis of the environment. Three different approaches are proposed. The first uses a coherent noise function to create "controlled" random paths. The second technique creates virtual lanes that characters can follow in order to reach their destinations, whereas the last approach allows a character to cut corners and follow a shorter and more direct path to its goal.

In Chapter 3, we introduce the Indicative Route Method (IRM) as a new path planning technique for interactive virtual worlds and games. In the IRM, an indicative route determines the global (preferred) path of a character, whereas a potential field approach is used to locally guide the character through a collision-free area (corridor) around this route. Similar to the CMM, the IRM exploits the concept of path planning inside corridors providing enough local flexibility for the character when dynamic obstacles (or other moving characters) have to be avoided. However, in the IRM, a much more natural steering potential function is employed that leads to higher quality paths. In addition, as compared to the CMM, as well as to other roadmap-based techniques, the IRM does not limit the global behaviour of the characters. Indicative routes can be computed using any of the well known techniques in the motion planning literature, or even provided manually by the user. The route is then given as an input to our framework and a smooth, C^1-continuous, path is returned. As we will show, the IRM can be easily combined with existing schemes for local collision avoidance, allowing simulations of thousands of characters at real-time frame rates and in complex environments, keeping at the same time the processor resources required quite low.

In Chapter 4, we present a simple technique to improve the indicative routes that virtual characters can follow in a crowd simulation. Our approach places a coarse grid map on top of the environment where every grid cell stores information about the number of characters that traversed this cell in the recent past. Given a path planning query for a virtual character, an indicative route is retrieved from the grid map by running an A* search algorithm that also accounts for the crowd density in the environment. We then apply the IRM combined with a local method for collision avoidance to compute a smooth and a collision-free path for the character. As we will experimentally show, using our proposed approach, characters can intelligently plan around congested areas, preferring to move through less dense regions. Although the resulting paths may be longer, the amount of local interactions among the characters is reduced, which leads to time- and energy-efficient movements.

In Chapter 5, we propose a novel approach for planning and directing large groups of heterogeneous virtual characters by combining techniques from the operations research

theory with the IRM. In particular, our approach uses an Integer Linear Programming model to choreograph through space-time the indicative routes of the groups' members. The computed routes are then given as an input to the IRM in order to determine the final paths of the characters. As compared to previous solutions, our space-time formulation allows the IRM to account for dynamic congestion by spreading the groups' members over alternative routes, as well as to identify and prevent deadlock situations by coordinating the movements of the members and incorporating explicit waiting behaviour. Overall, as we demonstrate through a number of challenging scenarios, our approach is able to solve, in real-time, complex planning problems involving one or multiple groups of characters.

The second part of the thesis addresses the problem of real-time and visually convincing collision avoidance in multi-agent systems. It consists of three chapters.

In Chapter 6, we present a novel approach based on (social) forces for simulating the collision avoidance behaviour of interacting virtual characters. Similar to real people, in our model, each virtual character scans the environment to detect possible future collisions with other characters. It anticipates how these collisions will take place and then makes an efficient move to avoid them. Due to its force-based nature, our predictive model is easy to implement and extremely fast allowing real-time simulations of thousands of characters. Visual and numerical comparisons with previous solutions, namely the approaches proposed in [75,182,227], show that, overall, our method ensures smooth avoidance maneuvers resulting in shorter and less curved paths. It also predicts the emergence of self-organization phenomena on several example scenarios, allowing interactions to be solved more efficiently at a global scale and hence, reducing the amount of effort that the characters need to expend in order to safely reach their targets.

Despite their simplicity and low running times, Newtonian force models, such as the one in Chapter 6, are typically hard to calibrate and require careful parameter tuning to obtain desired and consistent simulation results, especially when attempting to steer thousands of characters through the virtual world. In contrast, velocity-based approaches, at the cost of being more computationally expensive, are much more easier to control and have gained a lot of popularity over the past few years. In these approaches, the velocity space is used to directly plan the avoidance maneuvers of the characters and thus, a more robust behaviour is obtained.

In Chapter 7, we also address the problem of real-time and natural looking avoidance behaviour between virtual characters using a velocity-based model. Our approach, though, is elaborated from interactions of real humans in controlled experiments and is based on the simple hypothesis that individuals take early and effort-efficient actions to avoid collisions by slightly adapting their directions and speeds. Simulations created with our system run at interactive rates, leading to visually compelling motions. As we will show, our model is in good agreement with our experimental interactions data and also captures important emergent phenomena that have been widely noted in the pedestrian literature. Improvements over one of the most popular velocity-based methods, the RVO model [227], are also noticeable. This is due to the fact that the RVO method, as well as the majority of existing velocity-based methods, try at every simulation step to find an optimal collision-free velocity for each character. In contrast, our approach favours small adaptations in the velocity of a character, even though such adaptations may lead to a collision in the (far) future. Assuming that other characters exhibit similar behaviour interactions are solved in advance and with minimal effort.

In Chapter 8, we extend the velocity-based approach introduced in Chapter 7 and

present a novel method to simulate the local behaviour of small groups of characters. The intuition behind our approach is based on observed behaviour of real crowds. In real-life, the majority of the pedestrians do not walk alone, but in small groups consisting of two or three members, such as couples, friends, or family members. Similarly, our model considers pairs and triples of characters and uses a two-step algorithm to ensure that the group members will safely navigate toward their goals, adopting at the same time configurations that closely follow the ones observed in real pedestrian groups. We highlight the potential of our method through a wide range of challenging interaction scenarios. We also compare our technique with existing solutions and evaluate the quality of our simulations using visual and numerical comparisons with video footages of real crowds. In all of the experiments, our algorithm is able to predict the emergence of empirically observed walking patterns generating smooth and convincing group motions.

Part I

Path Planning

Chapter 2

Adding Variation to Path Planning

Path planning in computer games, whether these are serious or entertainment games, plays an important role in the immersion of a player. Immersion enables players to feel as if they are present in the game world and live an experience in which "the dynamic form of successful play becomes beautiful and satisfying" [175].

Recently, the concept of path finding inside *corridors* has been introduced [99] and a general framework, under the name *Corridor Map Method* (CMM), has been proposed by Geraerts and Overmars [57]. The main advantage of the CMM over the traditional path finding approaches is the ability to compute smooth and aesthetically pleasant paths in real-time and in any complicated environment, which makes it ideal for interactive applications like games. However, like most of the existing planners, a fixed path is always returned when the same query is performed. This is highly unnatural behaviour that can break the player's sense of immersion; there is little more disturbing for the player than to see a virtual character performing the same actions and following the same paths over and over again.

An individual does not always take the exact same route, when solving a typical path planning problem in real life. Thus, for a computer game character to be believable, it should follow a (slightly) different path in response to the same query. This not only improves the realism of the game's AI, but also presents a less predictable opponent for the player.

Furthermore, people usually show a preference to a specific route while moving to a desired location. Although the paths might globally be the same, locally they can vary from individual to individual. As an example, consider the paths that people follow when walking down a hallway. One person might have the tendency to stay close to the walls or to turn corners more tightly, whereas another might prefer to walk in the middle of the hallway. Therefore, alternative paths should also be generated to indicate different preferred routes that a virtual character can follow, given a specific query. Such variation enhances the realism of the gaming and/or training experience, allowing the virtual worlds to look more lively.

Another critical issue that can easily destroy the suspension of disbelief and ruin the player's immersion is the speed of the animated characters. In real life, parameters related to the personality, mood, and age of people can strongly influence their walking behaviours. Nevertheless, such details are usually overlooked in discussions of motion planning. Typically, computer games and interactive applications populate their worlds with a large number of characters that all have identical motions and move at the same speed, which results in uncompelling simulations.

In this chapter, we extend the Corridor Map Method by creating high-quality alternative paths that a character can follow within a corridor. We also demonstrate how

variation in the speed of the animated characters can be incorporated into the method, leading to believable characters that navigate through a virtual environment in a natural looking manner.

2.1 Related Work

Over the past few years, a considerable amount of research has emerged aiming to create and evaluate variety for virtual humans in crowd simulation applications. While advancements have been made in the domains of rendering [36,211], modeling [37,141,142] and behavioural simulation [38, 67, 161], little effort has been put to introduce variation in the paths that virtual characters follow.

In the virtual environments community, the most common way to plan a path is to combine a cell decomposition technique (e.g. [33,118,179]) with a graph-based search algorithm like A* [185] or Dijkstra [35]. These algorithms always return the shortest path, if one exists. In addition, different solution paths between a given start and goal position can be obtained by varying the cost function of the search algorithm. For example, Pettré *et al.* [170] created variants of paths by invoking Dijkstra's algorithm on a Navigation Graph [172] and iteratively modifying the cost of the graph edges. Similarly, Paris *et al.* [162] proposed a multi-criteria A*-based search algorithm that takes individual preferences and knowledge of the environment into account upon choosing a path for a character. However, such techniques are primarily designed for generating non-homotopic paths and have a different goal as compared to our approach that aims to create variants of paths belonging to the same homotopic class [1]. In addition, graph-based search algorithms usually produce aesthetically unpleasant paths and hence, extra care should be taken to smooth them.

In the animation community, the most popular approach is the flocking technique of Reynolds [181] that describes the collective behaviour of group members using simple local rules. Later, Reynolds extended the flocking model by incorporating additional algorithmic steering behaviours for autonomous characters [182]. This approach works very well in open environments and leads to visually plausible simulations. We can also combine several of the proposed behaviours, such as the "wander" and "path-following" behaviours, in order to create alternative paths. However, the local nature of the method gives no guarantees on the generated paths. For example, the characters can easily get stuck behind obstacles in cluttered environments. Furthermore, the resulting variation is not very natural, whereas the returned paths may not belong to the same homotopic class. Similar problems are also encountered in the approach of Ulicny and Thalmann [220] that attempts to introduce path variety by randomly choosing waypoints within regions defined around a predetermined sequence of path nodes.

Pedestrian and traffic flow simulation have also received a lot of attention over the past few years. Interesting models have been proposed that can be used to create different paths for computer-controlled characters. The most influential in this field is the social force model of Helbing [75, 76, 77]. Helbing simulated the microscopic behaviour of pedestrians using social analogs of physical forces. The considered forces include attractive, repulsive, frictional and dissipative terms. A fluctuation term is also added that accounts for random variations in the behaviour of the pedestrians, leading to the generation of different paths. However, the same problem as in flocking arises, since in all these models only local information is taken into account.

[1] Two paths, Π_0 and Π_1, with endpoints fixed, are said to be homotopic only if one path can be continuously warped into another without intersecting any obstacles.

Another way of creating alternative paths would be to use rapidly exploring random trees (RRTs). RRTs originate from the robotics community and are particularly suited for solving single-query motion planning problems in high dimensional spaces [113,122, 123]; a tree of valid paths is grown in $\mathcal{C}_{\text{free}}$ from the start configuration using random sampling, until the goal configuration can be connected to the tree. Due to their random nature, RRTs seem ideal for creating different paths given a specific path planning query. Unfortunately, the running time is prohibitively high and the resulting paths can be of low quality, rendering this approach inappropriate for gaming applications.

The Corridor Map Method can overcome the disadvantages of the path finding and steering approaches mentioned above. In the CMM, the global motion of the character is directed by high quality roadmaps, whereas a social potential field [180] is used to guide the local motion of the character inside a corridor. As a result, the method can always guarantee that a path, if one exists, will be found, providing at the same time the flexibility of creating locally different paths by taking additional constraints into account. Bearing also in mind that most of the work is done in a preprocessing phase, the CMM is applicable for creating variation of paths in virtual environment applications.

2.2 Corridor Map Method

In this section, we provide a brief overview of the CMM. We refer the reader to [57,58] for a more thorough explanation. The CMM consists of two phases, an off-line construction phase and an online query phase. In the construction phase, a system of collision-free corridors is created defining the *corridor map* data structure. In the query phase, paths for different types of characters can be planned inside corridors extracted from the corridor map.

2.2.1 Corridor Map

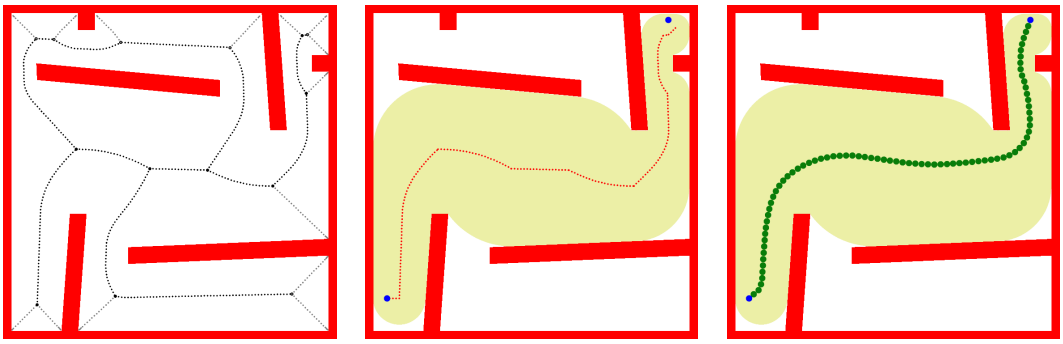
The corridor map was introduced by Gerearts and Overmars [57] as an efficient data structure that captures the (walkable) free space in the environment. As the walkable space is normally 2-dimensional, the corridor map is defined in the plane; the obstacles are the footprints of the original 3-dimensional obstacles of the environment.

The corridor map is represented as a graph whose edges correspond to collision-free *corridors*. Such a corridor consists of a *backbone path* enhanced with clearance information. On every point along the path, the clearance is defined as the radius of the largest disc around this point that does not intersect with the environment. More formally:

Definition 2.1 (Corridor)**.** *A corridor $\mathcal{B}(s) = (B[s], R[s])$ is defined as a sequence of maximum clearance discs with radii $R[s]$ whose center points lie along its backbone path $B[s]$. The parameter s is an index ranging between 0 and 1, and $B[s]$ denotes the coordinates of the center of the disc corresponding to index s.*

Together, the backbone paths form the *skeleton* of the corridor map. See Figure 2.5 for an example of a virtual city, its footprint, and the skeleton defining the corridor map.

The corridors allow a character to handle a broad range of path planning issues, such as avoiding other characters and computing natural paths. To address these issues, the corridor map should satisfy the following criteria. First, if a path exists in the free space, then a corridor must exist in the map that leads the character from its start to its goal position. Second, the map should include all cycles that are present in the environment. These cycles provide short global paths and alternative routes which allow for variation in the



(a) Skeleton of the corridor map (b) Backbone path and its corridor (c) Resulting path

Figure 2.1: The Corridor Map Method.

global characters' routes. Third, corridors extracted from the map must have a maximum clearance. Such a corridor provides maximum local flexibility in the movements of the character.

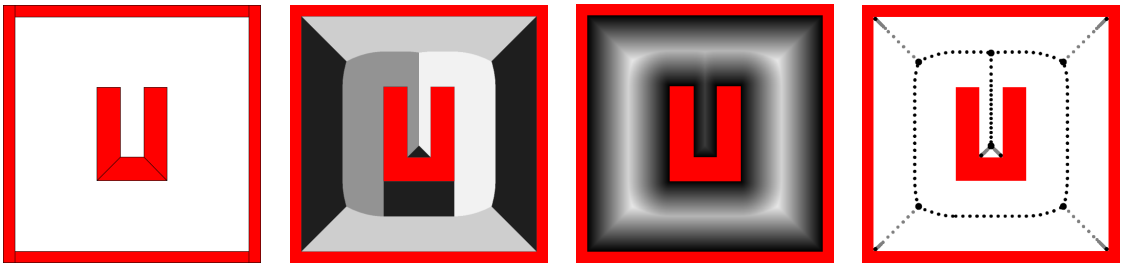
These requirements are met by using the *medial axis* (MA) as skeleton for the corridor map [58]. The MA of a set of (convex) polygonal obstacles is closely related to its *generalized Voronoi diagram* (GVD). In particular, it can be obtained by removing a number of edges and vertices from the GVD; namely only edges and their incident vertices that meet the reflex vertices of the obstacles are not part of the medial axis [22]. Therefore, very often in the literature, the terms MA and GVD are used interchangeably.

Consider a set of convex polygonal obstacles in the plane. Then, the GVD partitions the free space into maximal connected regions (two-dimensional cells) such that all points in a region are closer to a particular obstacle than to any other obstacle of the set [23]. Such a region is called a *Voronoi region*. The boundaries of the Voronoi regions, with certain exceptions (see above), form the underlying graph of the corridor map. We refer to Figure 2.1(a) for an example. Each vertex in this graph is located at a non-convex corner induced by at least two obstacles or at a location at which three or more edges of the graph meet. An edge connects two vertices and traces the boundary between two Voronoi regions. The edges are densely sampled and with each sampled point, the maximum clearance disc centered at this point is stored. A sequence of these discs forms a corridor. Consequently, the corridor map can now be defined as follows:

Definition 2.2 (Corridor Map)**.** *The Corridor Map is a discrete Voronoi graph $G = (V, E)$ enhanced with clearance information. Each edge is densely sampled and each sample point is annotated with the radius of its corresponding maximum clearance disc.*

The GVD can be computed efficiently by exploiting graphics hardware as proposed in [58, 81]. A 3D distance mesh, consisting of polygons, is computed for each geometric obstacle present in the footprint of the environment.[2] Each of the meshes is rendered on the graphics card in a different color. A parallel projection of the upper envelope of the arrangement of these meshes gives the GVD. The diagram can be retrieved from the graphics card's frame buffer and the clearance values (i.e. distance values) can be found in the Z-buffer. These steps are visualized in Figure 2.2. The approach is very fast. For

[2]Upon the construction of the GVD, we require that all obstacles are convex to assure that edges run into concavities. This requirement does not impose any limitation as non-convex obstacles can be converted into convex ones by applying a decomposition scheme, such as triangulation [32].



(a) Environment (b) Frame buffer (c) Z-buffer (d) Corridor map

Figure 2.2: Construction of the Corridor map using graphics hardware [58].

example, the corridor map in Figure 2.5(b) was computed in 0.06 seconds on a modern PC with an ATI Radeon HD 4800 graphics card. Note that the computation of the corridor map happens only once during preprocessing.

2.2.2 Path Planning Using Corridors

In the query phase, to plan a path for a character modeled as a disc with radius r, we first need to extract an appropriate corridor from the corridor map. After connecting the character's start and goal positions to the Voronoi graph, a backbone path is obtained by running an A* shortest path algorithm on the graph. Then, the corresponding corridor is formed by concatenating the corridors of the Voronoi edges of the backbone path. See Figure 2.1(b) for an example of a backbone path and its corridor.

The backbone path guides the global motions of the character, whereas its local motions are controlled by a force field technique. Forces drive the character inside the corridor and the character's movements are updated using Newton's laws of motion and a time integration scheme. To be more specific, at every time step, an *attraction point* moves along the backbone path and attracts the character in such a way that no collisions occur with the environment. Let $\mathbf{x}(t)$ denote the current position of the character at time t defined by the center of the disc (hereafter, for notational convenience, we will not explicitly indicate the time dependence). Then, according to [57]:

Definition 2.3 (Attraction point)**.** *Given the current position* $\mathbf{x}$ *of the character with radius* r*, the attraction point* $\alpha(\mathbf{x})$ *is the point* $B[s]$ *on the backbone path* B *having the largest index* $s : s \in [0, 1]$ *such that the character is still enclosed by the maximum clearance disc of radius* $R[s]$ *centered at* $B[s]$*, that is, for the Euclidean distance* $(\mathbf{x}, B[s])$ *holds:* $(\mathbf{x}, B[s]) < R[s] - r$*.*

A force $\mathbf{F}_{att}$ is applied to the character by the attraction point, making the character move forward and stay inside the corridor. Let d be the Euclidean distance between the character's position $\mathbf{x}$ and the attraction point $\alpha(\mathbf{x})$. Then, $\mathbf{F}_{att}$ is given by:

$$\mathbf{F}_{att} = f\frac{\alpha(\mathbf{x}) - \mathbf{x}}{\|\alpha(\mathbf{x}) - \mathbf{x}\|}\text{, where } f = \frac{1}{R[s] - r - d} - \frac{1}{R[s] - r}. \tag{2.1}$$

The scalar f is chosen such that the force becomes infinite when the character touches the boundary of the clearance disc and zero when the character is positioned on the attraction point. (In practice, though, f will never reach infinity since we require that the radii of the discs are strictly larger than r, that is, during the A* search, we discard all paths that have clearance less than or equal to r.)

Additional behaviour can be incorporated by adding extra forces to $\mathbf{F}_{att}$. For example, repulsive forces can be added so that the character can avoid collisions with dynamic

obstacles or other characters. The final path is obtained by iteratively integrating the resulting force $\mathbf{F}$ over time while updating the velocity, position and attraction point of the character. In [57], it is proved that the generated path is smooth, that is C^1-continuous. An example of such a path is displayed in Figure 2.1(c).

2.3 Overall Problem Description and Approach

We are given a set of (convex) polygonal obstacles in the plane $\mathbb{R}^2$ and a character A modeled as a disc with radius r. The character is only capable of translation and thus, the configuration space is identical to the 2D workspace, that is, $\mathcal{W} = \mathcal{C} = \mathbb{R}^2$. Then, the task is to compute alternative homotopic paths that A can follow from a given start $\mathbf{s} \in \mathcal{C}_{\text{free}}$ to a given goal $\mathbf{g} \in \mathcal{C}_{\text{free}}$ configuration. To define the homotopic class for the different paths, the backbone path from $\mathbf{s}$ to $\mathbf{g}$ computed by the CMM is used.

We describe two approaches for creating such path variation. In the first approach, a random bias is applied in order to retrieve a path that is slightly different than the fixed path returned by the original CMM. In the second approach, a fixed bias is applied to generate alternative routes that a character may prefer to follow inside the corridor. The bias can either be directed to the right (or left) of the backbone path, leading to the formation of different lanes inside the corridor, or it can follow the direction of the backbone path, allowing the character to cut corners and generating more natural looking motions. Both approaches can also be combined.

2.4 Variation of Paths Using Perlin Noise

This section presents our first approach that generates slightly different paths every time the same path planning problem has to be solved. The quality of the returned paths and the effectiveness of the approach are also discussed.

2.4.1 Implementation

An obvious way of generating different paths would be to add at every time step a random force (bias) to the attractive force that guides the character toward its goal. However, the resulting paths will be far from realistic, since the character may change direction almost instantaneously.

A more sophisticated approach is the creation of a "directed" random path, where the steering direction of the character at any moment is related to the previous one. We will show that a coherent *Perlin noise* function [168] can be used to control the direction of the random force and ensure that it will change smoothly at every step of the integration.

Perlin Noise and Random Bias. Perlin noise is a pseudorandom noise function introduced by Perlin [168]. His main goal was to create a wide variety of natural looking textures. Nowadays, it is widely used in the computer graphics and animation community, as well as in the film industry to simulate natural looking effects and create realistic representations of natural elements. The function is simple to implement and provides sufficient spectral control for most applications.

In our problem setting, Perlin noise is implemented as a function $\mu : \mathbb{R}^2 \to [-\pi/2, \pi/2]$. In each cycle of the simulation, the current point of attraction $\alpha(\mathbf{x})$ is given as an input and a *direction* for the bias is returned that varies *pseudorandomly*. To be more specific,

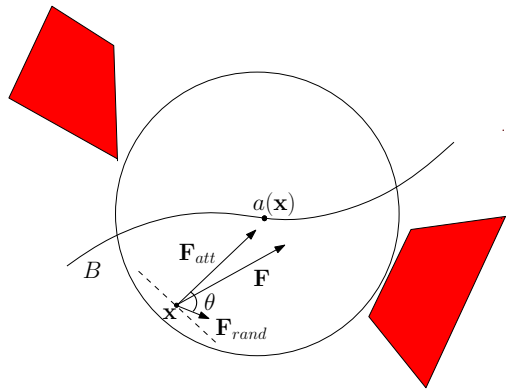


Figure 2.3: Adding a random force.

we define a 2D regular grid where its cells are unit squares and its points are integers. On each grid point, we assign a pseudorandom direction in the range of $[-\pi/2, \pi/2]$. By pseudorandom, we mean that the same direction is always returned for the same grid point. As there are infinite grid points, we do not explicitly store the grid, but rather determine on-the-fly the corresponding directions of the grid points whenever they are needed. Then, given the attraction point of the character (or any other point in $\mathbb{R}^2$), the value of the noise function at $\alpha(\mathbf{x})$ can be easily computed by determining the four points on the grid enclosing $\alpha(\mathbf{x})$ and taking the weighted average of the values corresponding to these points. Note that to smooth out the output of the noise function, a cosine interpolation scheme was preferred over a simple linear interpolation.

As a feature of our Perlin noise function, providing the same attraction point will always result in the same value. Furthermore, a small change in the input point will lead to only a small change in the direction of the bias. Therefore, if the current attraction point lies very close to the attraction point at the previous time step, the new bias will still be heading in almost the same direction as the previous one.

Perlin noise also allows us to control the *frequency* with which the output value is changed. A low frequency results in smooth changes in the direction of the bias, generating aesthetically pleasant paths. On the contrary, a too high frequency noise leads to the retrieval of unrealistic paths, as the character will change its direction at almost every successive time step. Several Perlin noise functions with different frequencies can also be combined at every simulation step. This will create more variety in the resulting paths, at a cost of a small additional computation time. In our implementation, only one function was used, since it was sufficient to obtain smooth random paths.

To initialize our Perlin noise function, a random seed is used. By changing the seed value, the output of the Perlin noise is changed, leading to the generation of a locally different path when the same path planning query must be solved.

Forces and Time Integration. At every step of the numerical integration, two forces are used to generate the character's motion inside the corridor (Figure 2.3). First, the attraction force $\mathbf{F}_{att}$ steers the character toward a point further on the backbone path B as defined in Equation 2.1. Second, a random force $\mathbf{F}_{rand}$ is added to slightly change the direction of the character's movement.

Let θ denote the direction returned by our Perlin noise function. This direction is expressed as an angle of rotation about the current attractive vector. Then, the random

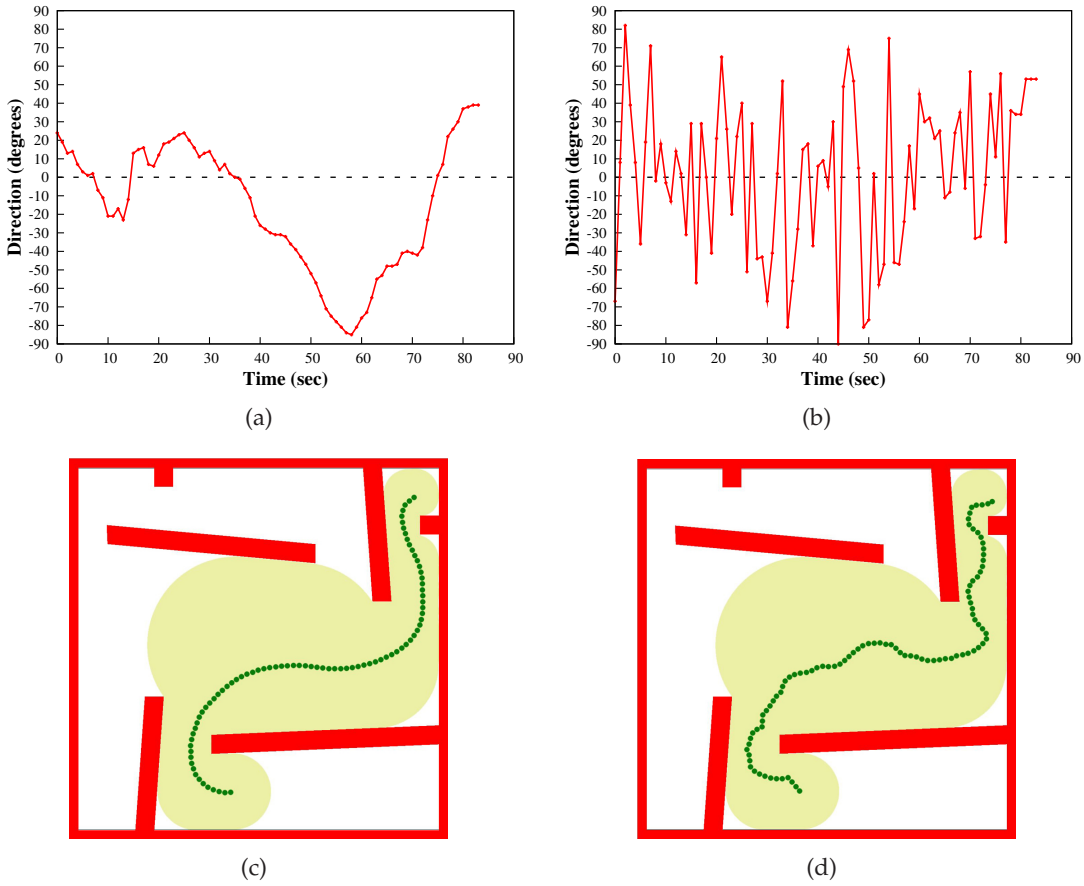


Figure 2.4: The frequency of the Perlin noise function affects the quality of the generated paths. (a), (b) Direction of the random force per simulation time for a low and a high frequency respectively. (c), (d) The corresponding low-frequency and high-frequency paths.

force $\mathbf{F}_{rand}$ can be defined as:

$$\mathbf{F}_{rand} = c_{rand}\, \mathcal{R}(\theta)\, \mathbf{F}_{att} \tag{2.2}$$

where $c_{rand} \in [0, 1]$ is a constant specifying the strength of the force relative to $\|\mathbf{F}_{att}\|$, $\mathcal{R}(\theta)$ represents the 2D rotation matrix and the angle $\theta \in [-\pi/2, +\pi/2]$ as explained above. The latter ensures that the character will slightly deviate from its main route, as the random force will be directed either to the left or to the right of the attraction point. Regarding c_{rand}, we should point out that, due to the way the attraction point is defined, the strength (magnitude) of the attractive force $\mathbf{F}_{att}$ is usually very large (see Section 3.1 for a detailed analysis). Consequently, for relatively small c_{rand} values, $\mathbf{F}_{att}$ tends to cancel out $\mathbf{F}_{rand}$ and, hence, the character will basically follow its backbone path while performing small and hardly noticeable displacements. In contrast, the closer c_{rand} is to 1, the more apparent and diverse will be the resulting variation. A typical value for c_{rand}, also used in our experiments, is 0.7.

Having determined the random force, the final force $\mathbf{F}$ can be calculated by adding the random force to the current attractive force. The resulting force vector is used to compute the new velocity vector and update the position of the character. The attractive force keeps the character inside the corridor, whereas the random force changes the steering direction.

(a) 3D model (courtesy of MetaVR Inc. [143]) (b) Footprint and skeleton

Figure 2.5: The McKenna MOUT training site at Fort Benning, Georgia, USA. (a) 3D representation of the site, (b) The graph of the corridor map used for the experiments. Small buildings and obstacles were not included in the footprint of the test environment.

Frequency Dependence. As mentioned above, the frequency of the noise function affects the quality of the computed random paths. In Figure 2.4, a typical example of varying the frequency is depicted. The first image shows the correlation between the direction of the random force and the motion time, given a low frequency. An aesthetically pleasant path is retrieved as displayed in the third image. The second image demonstrates an example of a high frequency usage that leads to an unrealistic path, displayed in the fourth image. The simulated movement looks like a drunkard's walk and certainly is not a path that someone would have taken in real-life, since no dynamic obstacles are present in the environment.

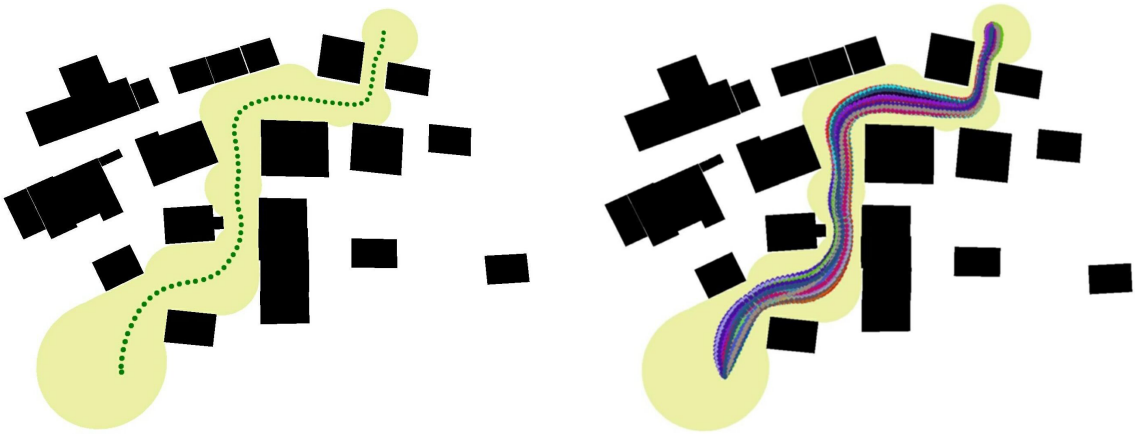
2.4.2 Results

We have implemented the presented approach to test its efficiency and check whether it can produce smooth paths that are similar yet different to the ones returned by the CMM. All the experiments were performed on a Pentium IV 2.4 GHz computer with 1 GB memory.

The experiments were conducted for the environment depicted in Figure 2.5. This is a model of the McKenna MOUT (Military Operations in Urban Terrain) training center, hosted at Fort Benning, Georgia, USA. The center features an urban European village and is used by the United States Armed Forces for military training.

There are many scale differences in the test environment, that is it contains many large open spaces and many narrow passages (see Figure 2.5(b)). This leads to the extraction of different types of corridors, which make this environment ideal for testing our path planning approaches.

Resulting Paths. In our implementation, the character retains its steering direction and at every time step performs smooth random displacements. Thus, a slightly different path is expected to be computed in response to a given query. Figure 2.6(b) shows 100 paths generated by our technique given a start and goal position, whereas the fixed path created by the CMM is displayed in Figure 2.6(a). We compared the length as well as the (average, maximum and minimum) clearance of the paths produced by the two methods.



(a) (b)

Figure 2.6: Using Perlin noise to create slightly different paths. (a) Fixed path returned by the CMM, (b) 100 random paths generated using the noise function.

	Clearance (m)			**Path Length** (m)
	min	avg	max	
Fixed path	3.25	6.21	10.81	144.08
Random path	2.87	5.94	10.01	145.22

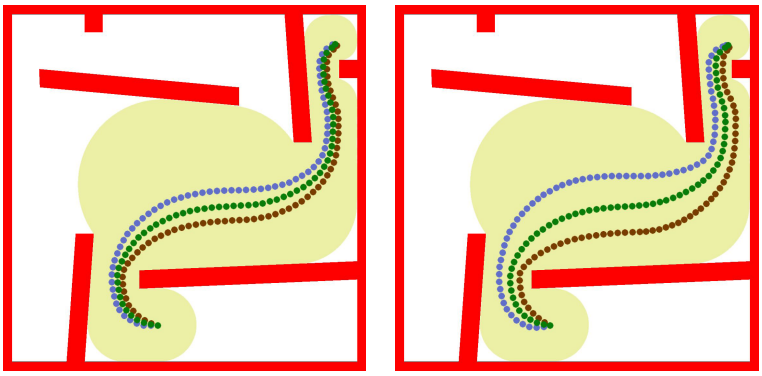
Table 2.1: Comparison between fixed and random paths for the McKenna test environment. The random-path statistics are the averages over 100 paths.

Table 2.1 shows the corresponding statistics. As can be seen, the random paths keep a safe amount of clearance from the edges of the corridor and have almost the same average clearance as the fixed path. In addition, no considerable difference in the length of the paths is observed. As other queries also led to similar results, we can conclude that our proposed method can successfully generate smooth paths that are very similar but still different from the ones returned by the CMM.

Performance. Apart from the quality of the generated paths, we are also interested in the performance of our proposed approach. The paths must be computed in real-time so that the method can be applicable to interactive applications like computer games.

In [57], Geraerts and Overmars have already shown that the CMM produces high-quality paths in real-time. As in our implementation we only extend the force function of the original CMM, we expect that the running time of the method will be marginally influenced.

To test the performance, 100 paths from random start and goal positions were generated. We measured the average CPU time required to compute a path, as well as the average time that the character needs to traverse it. Consistent with the results presented in [57], it took less than a half millisecond CPU time to calculate a second of movement for a character, which implies a CPU load less than 0.05% per character. This clearly demonstrates that our Perlin noise implementation does not add any significant overhead to the CPU usage and is very efficient in producing random smooth paths in real-time.



(a) $k = 0.25$ (b) $k = 0.5$

Figure 2.7: Lane formation inside the corridor for different values of k.

2.5 Alternative Paths Using Fixed Bias

As individuals can have a preference for certain paths over others, a second approach is proposed to generate different preferred paths that a character can follow inside the corridor. Similar to the previous approach, the force function is extended in order to bias the character's motion. Two different implementations are presented and evaluated.

2.5.1 Lane Formation

Our goal is to generate smooth paths that lie either to the left or to the right of the backbone path. This leads to the formation of different *lanes* inside the corridor that the character can follow in order to reach its goal position.

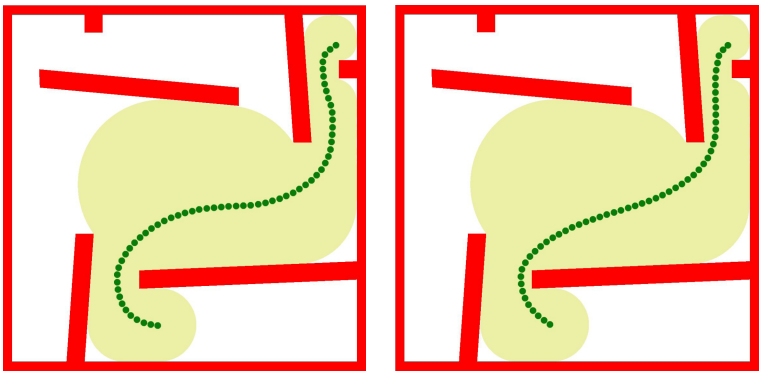
One could achieve this by creating a sub-corridor inside the original corridor. However, a much simpler approach is to update the force that guides the character through the corridor. An additional advantage of this method is that the characters would still be able to use the other side of the corridor, if constraints (like avoiding other entities) would enforce this.

We note that lanes can also be created by placing waypoints along the corridor as proposed in [170] and further discussed in Chapter 5. Here, we present a fast alternative method based on physical forces. Such approach can also be easily combined with Perlin noise to add more variety in the lanes that a character can take.

Forces and Time Integration. At every time step, a force $\mathbf{F}_{perp}$ perpendicular to the main attractive force is exerted in order to change the steering direction of the character. Let d be the Euclidean distance between the character's position $\mathbf{x}$ and the attraction point $\alpha(\mathbf{x})$, $R[s]$ be the clearance corresponding to $\alpha(\mathbf{x})$ and r be the radius of the character. Let also θ denote the angle of rotation about the current attractive vector, $\mathcal{R}(\theta)$ define the corresponding 2D rotation matrix and $k \in [0, 1]$ be a constant. Then the force $\mathbf{F}_{perp}$ can be described by:

$$\mathbf{F}_{perp} = c_{perp}\, \mathcal{R}(\theta)\, \mathbf{F}_{att} \text{ , where } c_{perp} = k\, \frac{R[s] - 2\, r}{d}. \tag{2.3}$$

In contrast to our Perlin noise approach, in the current implementation, the angle θ can only take two values: $\theta \in \{-\pi/2, +\pi/2\}$. A positive angle of rotation steers the character to the left of the attraction point, whereas a negative θ to the right. Therefore,



(a) (b)

Figure 2.8: The "Path Following" method leads to shorter and more convincing paths. (a) Smooth path returned by the original CMM, (b) Shorter path generated by our technique.

by keeping the angle fixed, a new (alternative) path will be generated either to the left or to the right of the backbone path.

The scalar c_{perp} controls the relative strength of the force with respect to $\mathbf{F}_{att}$. The closer the character is to the attraction point (i.e. the smaller the distance d), the stronger the force will be. In addition, the constant k is used to specify how close to the boundary of the corridor the character moves. It must be pointed out that since the attraction point is selected as the furthest advanced point on the backbone path, it always lies in front of the character, and therefore $d > 0$. The distance becomes zero when the character reaches its goal position. We refer the reader to Figure 2.7 for a graphical representation of the described technique. Different right/left paths are returned by varying the value of the constant k.

Our Perlin noise function can also be used to create slightly different lanes in response to a given query. Depending on whether we want to follow a right or a left path, the angle θ will lie within the range of $[-\pi/2, 0]$ or $[\pi/2, 0]$ respectively. The noise function ensures that the character will stay on one side of the corridor, performing at the same time smooth random movements.

2.5.2 Path Following

In this second technique, our goal is to generate paths that either take tight or wide corners. To achieve this, at every time step, we dynamically compute the direction toward which the backbone path is headed. The estimated direction is then used to bias the character's motion. As a result, if the path turns to the right or left, rather than waiting until the very last moment, the character will start the turn in advance. This leads to a more natural and visually interesting movement. The character looks intelligent, in the sense that it anticipates the direction of the path it has to follow and takes a more direct path, minimizing the time needed to reach its goal position.

Direction Estimation and Time Integration. Let $\alpha(\mathbf{x})$ be the attraction point that corresponds to point $B[s]$ on the backbone path having clearance $R[s]$. Then, at every time step, a point that lies ahead of the current attraction point is selected. The point $B^f[s]$ is defined as the furthest point on the backbone path that remains inside the clearance disc

	Clearance (m)			**Path Length** (m)
	min	avg	max	
Fixed path	3.25	6.22	10.81	144.08
Right path	1.36	3.94	8.05	145.56
Left path	1.68	4.48	8.58	147.71
Direct path	1.31	4.89	8.99	131.69

Table 2.2: Clearance and length statistics for the fixed path and the alternative paths in the McKenna test environment. The left/right paths were obtained by setting $k = 0.5$.

of the attraction point, that is

$$B^f[t] = \max_{s' \in (s,1]} \left\{ B[s'] \,\middle|\, dist\left(B[s'], B[s]\right) \leq R[s] \right\}. \tag{2.4}$$

Having extracted the maximum valid $B^f[s]$ point, the resulting force that has to be added to the attractive force equals to

$$\mathbf{F}_{dir} = c_{dir}\ \mathcal{R}(\theta)\ \mathbf{F}_{att}, \tag{2.5}$$

where c_{dir} specifies the strength of the force relative to $\|\mathbf{F}_{att}\|$, $\mathcal{R}(\theta)$ represents the 2D rotation matrix and θ is the angle between the current attractive vector $(\alpha(\mathbf{x}) - \mathbf{x})$ and the estimated direction of the backbone path $\left(B^f[s] - B[s]\right)$. In our experiments, we set $c_{dir} = 0.7$ (see also Section 2.4.1).

Similar to the previous approaches, the direction of the additional force, that is the angle θ, can also be controlled by a Perlin noise function, leading to the generation of different shorter paths. The closer this direction is to the direction of the backbone path, the more tightly the character will turn the corners.

The final force ($\mathbf{F} = \mathbf{F}_{att} + \mathbf{F}_{dir}$) drives the character toward its goal position and at the same time ensures that any unnecessary detour will be avoided. Therefore, a shorter and more believable path is returned, as can be seen in Figure 2.8.

2.5.3 Results

We implemented the two techniques described in this section. Once again the McKenna environment was used to test the performance of the two methods and the quality of the generated paths.

Resulting Paths. In the first part of the experiments, we defined 100 random queries and compared the clearance and the length of the paths produced by our two proposed techniques, as well as by the original CMM. Table 2.2 shows the corresponding statistics for our example query depicted in Figure 2.6(a).

By applying a left and a right perpendicular bias ($k = 0.5$) a 3-lane corridor was obtained. As both the left and right paths are closer to the boundary of the corridor, they have less (minimum, maximum and average) clearance than the fixed path. However, as it can be inferred by the table, no noticeable difference in the length of the paths is observed. In the "Path Following" method the character predicts the direction of the path it has to follow and starts turning in advance. Thus, a more natural looking movement is generated. As it is expected this path is the shortest one.

The results were similar for the other queries. Therefore, we can conclude that the two techniques can successfully generate alternative routes that are aesthetically pleasant to

the observer. The first method forms different lanes that a character can follow inside the extracted corridor, whereas the second method leads to a smooth and shorter path by creating shortcuts through the turns of the backbone path.

Performance. In the second part of our experiments, we tested the performance of the two presented techniques. Similar to the results of our first approach, it took less than a half millisecond per second traversed time to compute a path, either using the "right/left bias" method or the "path following" method. As a result, both implementations are efficient in planning alternative paths inside a corridor.

2.6 Variation of Speed

The approaches described above can be easily implemented to compute alternative paths that a character can follow. This leads to more realistic motions of characters. Another important, but often overlooked, factor that contributes to the realism of virtual environment applications is the speed of the animated character. A character moving at an unnatural speed would immediately break our suspension of disbelief. Therefore, in this section, we report some known facts about human speed and describe how to integrate them in our approach.

According to the U.S. Manual of Uniform Traffic Control Devices (MUTCD), the normal walking speed is 1.2 m/s [42], while the Manual of Uniform Traffic Control Devices for Canada (MUTCDC) indicates that the normal speed can vary from 1.1 to 1.4 m/s [216]. These results are consistent with experimental observations in the field of transportation research (e.g. [107, 213]), as well as with biomechanical studies recommending 1.33 m/s as the typical walking speed for adults in unconstrained environments [238].

These facts can be easily incorporated into the CMM in order to obtain a more believable path planning approach. In particular, at every step of the numerical integration, the speed of the character can be constrained to a maximum value that lies within the range of 1.1 and 1.4. Although, this modification will lead to a more natural motion, it is unrealistic to assume that the character will move at exactly the same speed toward its goal.

In real-life, various factors can influence our walking speed, such as the walking surface, the weather conditions and the crowd density. Fritz and Carver mention that winter conditions can remarkably change the speed, whereas a strong headwind can more than halve the speed of the walker [51]. The morphology, as well as the steepness of the terrain have also a significant effect on the speed. For example, a standard walking pace is unlikely to be achieved on a densely vegetated terrain, a rocky field or an ascent. Furthermore, the presence of many people in an area can also make walking difficult and lead to a decrease in the walker's speed.

An obvious way to simulate these behaviours is to add an extra *drag* force to the final force that steers the character. The drag force is directly related to the speed of the character and is headed in the opposite direction of the character's motion, that is $\mathbf{F}_{drag} = -\kappa \times \mathbf{V}_{t-1}$, where $\kappa > 0$ controls the strength of the force and $\mathbf{V}_{t-1}$ is the velocity of the character at the previous time step.

Emotion can also significantly affect the speed of an individual. Over the past years the relationship between walking and emotion has received a vast amount of attention. In particular, qualitative measures for emotion related gait characteristics have been devised in [146]. For instance, "heavy-footedness" can indicate an angry gait, whereas a sad gait can be distinguished by a small amount of arm swing.

	Anger	Sad	Neutral	Joy	Content
Velocity (m/s)	1.41±0.22	1.10±0.21	1.19±0.13	1.42±0.23	1.29±0.19
Normalised Velocity (1/s)	0.83±0.12	0.66±0.13	0.71±0.08	0.84±0.13	0.76±0.11

Table 2.3: Gait velocities for the basic emotions according to [26].

Furthermore, a number of researchers from the area of psychology and biomechanics have recently managed to quantitatively describe the effects of emotion on gait [26, 63]. Their results indicate that the walking speed of an individual slows significantly with sadness. On the other hand, for emotions with high levels of arousal (i.e. anger, joy), the velocity is faster than the normal walking speed.

These findings are summarized in Table 2.3. The velocities presented in the table can be used to modify the speed of a virtual character and make it consistent with its emotional state and/or mood it is experiencing. This ensures that a more natural motion will be generated for each animated character.

2.7 Conclusions

In this chapter, we extended the CMM in order to obtain a more realistic path planning approach. In particular, we studied how alternative paths can be created within a corridor. Two different approaches were proposed. The first one uses a Perlin noise function to create slightly different paths in response to a given query. This technique is fast, simple to implement and provides tremendous control over the generated paths. In the second approach, a bias toward a fixed direction influences the movements of the character. The bias can either be directed to the left/right of the backbone path, leading to the formation of lanes, or it can follow the direction of the backbone path, resulting in shorter and more direct paths.

Experiments showed that our techniques can compute in real-time alternative paths for a character that are aesthetically pleasant to the viewer. Due to the nature of the CMM (the paths are computed using forces), our framework is also suited for planning alternative paths for crowds of moving characters. In this case, extra forces must be added so that the characters can avoid colliding with each other (see, for example, Chapter 6). Furthermore, to enhance the believability of the characters, some variation in their appearance is also required. Such variation can be easily obtained by varying their colours and/or textures as has been proposed in [141, 142].

Besides planning smooth and convincing paths, virtual characters should also move in a natural looking manner. A natural motion enhances the realism of the virtual environment and results in an interesting gameplay and/or better training transfer. Therefore, we also discussed how variation in the speed of the animated characters can be incorporated into the CMM, leading to characters that exhibit human-like behaviour.

For future work, we would like to evaluate the naturalness of the paths generated using our proposed techniques by conducting a user evaluation study as in [95, 141, 221]. Another interesting avenue for future work would be to measure whether the player's satisfaction in computer games is increased by disguising the repetitive behaviour of NPCs (non playing characters) with our path variation techniques.

Finally, we would like to extend our approach and create variants of global paths that belong to different homotopic classes, similar to the work of Pettré *et al.* [170, 172] and Paris *et al.* [162]. Typically, in games and simulation applications, we assume that virtual

characters take the shortest global path. However, in real-life, many parameters can affect the path choices that we make, leading to different paths (e.g. shortest, simplest, most appealing, safest, less congested). In the past, questionnaires have been used to determine several of such criteria [61]. Nowadays, though, with the existence of immersive and interactive virtual environments, we can accurately assess these criteria and incorporate them to our global planner. For example, we can let the users navigate through a virtual city environment and ask them to perform specific path planning queries. We can then evaluate the criteria most often used in route selection and determine how they influence the paths that users prefer to follow in correlation with e.g. their gender and/or age.

Chapter 3

Indicative Routes for Path Planning and Crowd Simulation

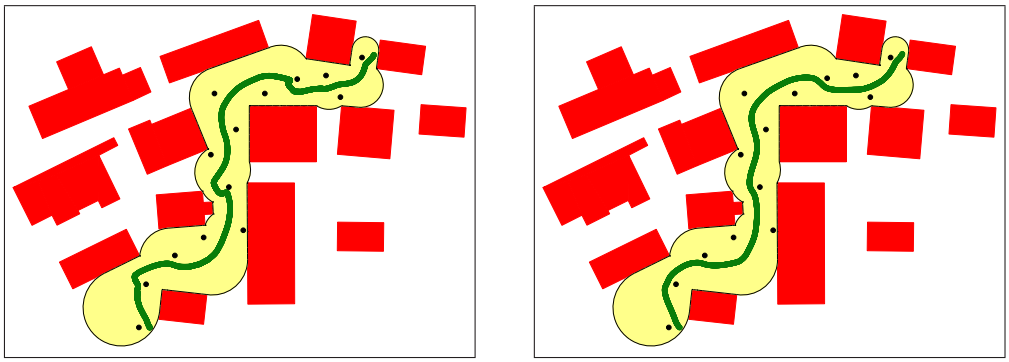
In the previous chapter, we extended the CMM to introduce variation in the paths that characters can follow within a corridor. Although the CMM is fast and flexible, there are a number of problems inherent both to the Corridor Map data structure and to the steering algorithm that guides the characters through the corridors extracted from the Corridor Map. In this chapter, we first highlight these problems and then propose the *Indicative Route Method* (IRM) as a new path planning method in interactive virtual environments and games. The IRM keeps the advantages of the CMM, but provides more control and flexibility in the global and local motions of the characters. As we will show, our method is relatively easy to implement and can be used for real-time navigation of crowds of characters in complex and dynamic virtual worlds generating smooth paths.

3.1 Limitations of the CMM

One of the shortcomings of the CMM is that the corridors' backbone paths are forced to lie on the medial axis of the environment. Such paths are ideal when characters need to traverse narrow passages, as they maintain a maximum clearance from the obstacles. However, these type of paths may lead to unrealistic behaviours in wide open spaces. Consider, for example, how unnatural would it look for a crowd of pedestrians to walk in the middle of a wide square. Note that other Voronoi-based techniques, such as *Navigation Graphs* [170, 172, 207] or the AERO method [208], suffer from similar drawbacks and, like the CMM, limit the global behaviour of the characters.

To alleviate these problems, in the IRM, we introduce the notion of *indicative routes* to indicate the global routes that the characters prefer to take. This allows more freedom in the global planning, as the characters no longer have to follow the medial axis. Indicative routes can be computed using any of the well known motion planning methods [121] and, hence, they do not impose any limitation on the global behaviour of the characters allowing our framework to generate a wide variety of macroscopic behaviours.

Another problem of the CMM is that the resulting paths tend to look unnatural when characters have to avoid collisions with small static obstacles and with each other (see Figure 3.1(a)). As explained in Section 2.2.2, an attractive force is used to simultaneously steer the character toward its goal and keep it inside its corridor. However, due to the way the attraction point is defined (see Definition 2.3), the character lies always close to the boundary of the attraction point's clearance disc, and, thus, an almost infinite force is generated. Although such a force is required when the character is close to the boundary of its corridor (Figure 3.2(a)), it provides limited control over the character's local motions

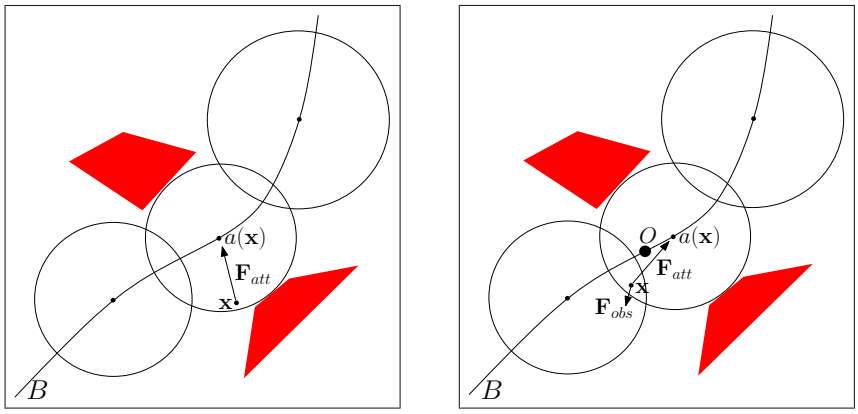


(a) CMM leads to jerky paths (b) IRM leads to smoother paths

Figure 3.1: Obstacle avoidance inside corridors using (a) the CMM and (b) the IRM.

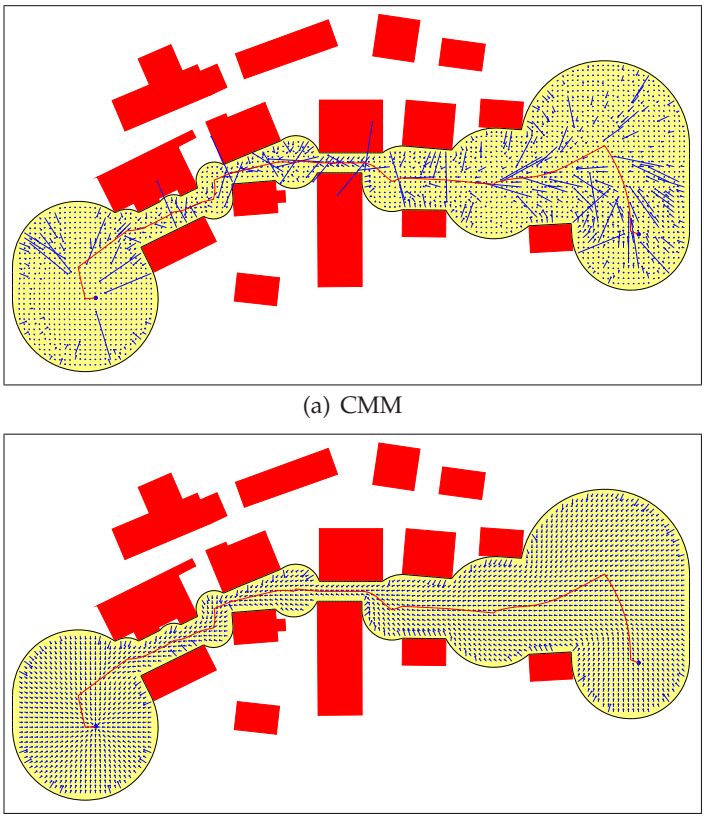
when additional forces need to be exerted (Figure 3.2(b)). Thus, the character has to be very close to an obstacle to avoid it and only at the very last moment adapts its motion, while in real-life people react much earlier.

In the IRM, we address this issue by using a different potential function to direct the character inside the corridor, which results in much more natural steering behaviour (see Figure 3.3 for a comparison between the force fields generated by the CMM and the IRM). In particular, we decouple $\mathbf{F}_{att}$ into two forces; a boundary force $\mathbf{F}_b$ pushes the character away from the boundary of the corridor, whereas a steering force $\mathbf{F}_s$ guides the character forward toward its goal. For the latter, we again use an attraction point. However, such an attraction point does not follow the backbone path of the corridor, as in the CMM, but runs along the indicative route of the character. We also take into account random variations in the character's behaviour to increase the realism of the simulation. Additional forces can also be exerted to steer the character away from other characters, small obstacles and local hazards (see Figure 3.1(b) for an example).



(a) (b)

Figure 3.2: In the CMM, an almost infinite force $\mathbf{F}_{att}$ steers the character during the simulation. (a) Such a force ensures that the character will stay inside its corridor. (b) However, similar large force is exerted when there is no risk of collision with the environment. In this example, $\mathbf{F}_{att}$ annihilates the effect of $\mathbf{F}_{obs}$ and does not allow the character to resolve in advance the collision with the obstacle O.



(a) CMM

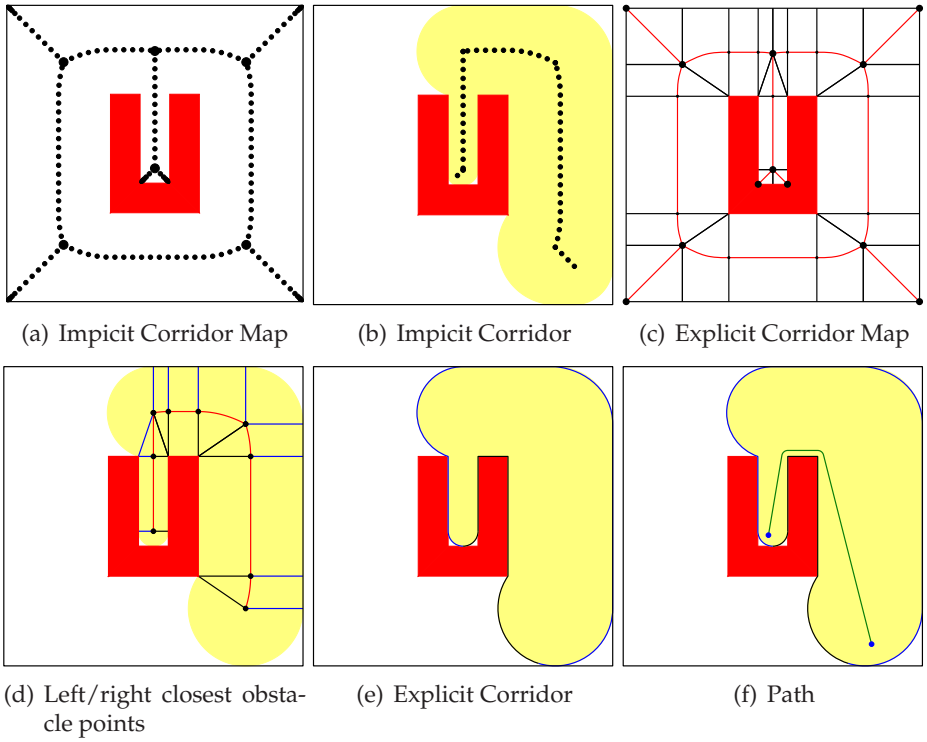
(b) IRM

Figure 3.3: Comparison between vector fields generated by (a) the CMM and (b) the IRM. In the CMM example, the vectors can assume large magnitudes and even point outside of the corridor. In the IRM example, where the medial axis (traced by the red curve) is used as the indicative route of the character, a smooth and singularity-free (except at the goal) vector field is generated.

3.2 Limitations of the Corridor Map

In Section 2.2.1, we densely sampled the edges of the generalized Voronoi diagram (GVD) and referred to the underlying structure as the (implicit) Corridor Map. An example of such a map is given in Figure 3.4(a). For each sample point, the radius of the largest empty disc was assigned. Then, a sequence of adjacent discs was referred to as an (implicit) Corridor, see Figure 3.4(b). However, such a discrete, sequential representation does not provide an explicit description of the corridor's boundaries that is required for computing the boundary force $\mathbf{F}_b$. While one could compute the boundaries of a set of discs, this would still be an approximation of the real boundaries and would take more than linear time [32], which may compromise real-time computations. Another disadvantage of having samples is that they do not fully cover the free space, whereas their memory footprint can be more than linear in the complexity of the free space.

As a result, we need a data structure that allows us to trace the boundaries of the corridors in a fast and accurate way. In addition, we strive for a structure with small storage cost, that is, a structure that is linear in the number of vertices of all obstacles, because modern virtual environments can be very large. Geraerts [55] has recently proposed the *Explicit Corridor Map* (ECM) as a data structure that meets the aforementioned criteria. An example of such a map is given in Figure 3.4(c). In the IRM, we will use the ECM to efficiently form an *Explicit Corridor* around a given indicative route, as shown



(a) Impicit Corridor Map (b) Impicit Corridor (c) Explicit Corridor Map

(d) Left/right closest obstacle points (e) Explicit Corridor (f) Path

Figure 3.4: Construction of the Corridor Map, Implicit and Explicit Corridor (courtesy of Roland Geraerts).

in Figure 3.4(e). As we will see, by exploiting the ECM and our proposed steering algorithm, flexible smooth paths (Figure 3.4(f)) can be obtained inside the corridors within a few milliseconds. Hence, our Indicative Route Method is applicable to a wide range of interactive applications like computer games.

3.3 Overview of the IRM

This section outlines our proposed Indicative Route Method. We first describe the problem setting in more detail and then give an overview of the approach taken to solve it.

3.3.1 Problem Formulation

In our problem setting, we are given the footprint of a virtual environment and a character that has to move from a specified start to a specified goal position without colliding with the environment and with other dynamic obstacles present in the world. The footprint consists of convex obstacles lying in the workspace, the plane $\mathbb{R}^2$. For simplicity, we assume that the dynamic obstacles are discs or polygons translating in this plane. Similarly, we assume that the character moves on the 2D plane and is modeled as a translating disc of radius r. Then, its configuration space is also the 2D Euclidean plane. At a fixed time t, the character is at position $\mathbf{x}(t)$, defined by the center of the disc, and moves with velocity $\mathbf{v}(t)$ (hereafter, for notational convenience, we will not explicitly indicate the time dependence). In addition, the character prefers to reach its goal as comfortable as possible and thus, moves with a certain preferred speed v^{pref}, whereas its actual motion is limited by a maximum speed $v^{\max}$.

3.3.2 Overall Approach

We propose a three-phase method to tackle the aforementioned path planning problem. The first phase consists of computing the *indicative route* of the character. The indicative route $\Pi_{\mathcal{IR}}$ is a path that runs from the start ($\mathbf{x}_S$) to the goal ($\mathbf{x}_G$) position of the character and provides an indication/rough estimation of the character's preferred route. Let $\Pi_{\mathcal{IR}}[s]$ denote each point along the path for some $s \in [0, 1]$. The character should be able to traverse $\Pi_{\mathcal{IR}}$, which means that the clearance $R[s]$ at every point on the path must be strictly larger than the radius r of the enclosing disc of the character. More formally:

Definition 3.1 (Indicative route)**.** *An indicative route* $\Pi_{\mathcal{IR}} : [0, 1] \rightarrow \mathbb{R}^2$ *is a path in the 2D workspace, such that* $\Pi_{\mathcal{IR}}[0] = \mathbf{x}_S$, $\Pi_{\mathcal{IR}}[1] = \mathbf{x}_G$ *and* $\forall\ s \in [0, 1] : R[s] > r$.

An indicative route could be indicated manually by a designer using, for example, waypoints or a sketch-based interface (see, e.g., [158]). Alternatively, any technique in the motion planning literature can be exploited to automatically construct such routes. One, for example, could calculate the medial axis of the environment and extract a path with enough clearance. Another option is to divide the environment into a coarse grid and search for an appropriate path using A*-like approaches (see, e.g., [33]). Note, though, that the paths produced by all these methods are in general not natural. Natural paths typically follow smooth curves, keep a preferred amount of clearance from obstacles, avoid unnecessary detours, allow for variations and can be easily adapted, if additional constraints (like avoiding other characters) impose this.

Consequently, the character should not traverse exactly the indicative route, but rather use it as a guide to plan its final motion. The second phase will allow for this by creating an *explicit corridor* around that route where the character can move without colliding with the environment. The corridor keeps the character's path in the same homotopic class as the indicative route, providing at the same time enough flexibility for the character's local movements. Section 3.4 elaborates on this phase.

In the third phase of our approach the final path Π_f of the character is extracted from the corridor using a potential field approach. In particular, we move an attraction point along the indicative route and apply (social) forces to guide the character's motion through the corridor. This leads to a high-quality path as discussed in Section 3.5.

We should also note that, for the purposes of this chapter, an intuitive authoring interface was developed allowing the user to sketch indicative routes by directly placing waypoints in the virtual world. The waypoints are then connected with line segments to form the character's route. In general, though, the specific choice of the indicative route is application and character dependent and falls outside the scope of this chapter. Hence, for the rest of the chapter, we assume that the indicative route of the character is given and we will mainly focus on the other two phases of our proposed method. However, in Section 3.7, we will give an example of how indicative routes can be obtained for simulating virtual crowds. In the next two chapters, we will also propose two novel approaches for automatically determining indicative routes based on an A* grid search and on Linear Programming techniques respectively.

3.4 Computing Corridors

In this section, we extend the indicative route to form a corridor in which the character can move freely. We first outline the basic aspects of the Explicit Corridor Map (ECM) data structure. Then, we explain how the corresponding corridor of an indicative route can be efficiently computed given the ECM of the environment.

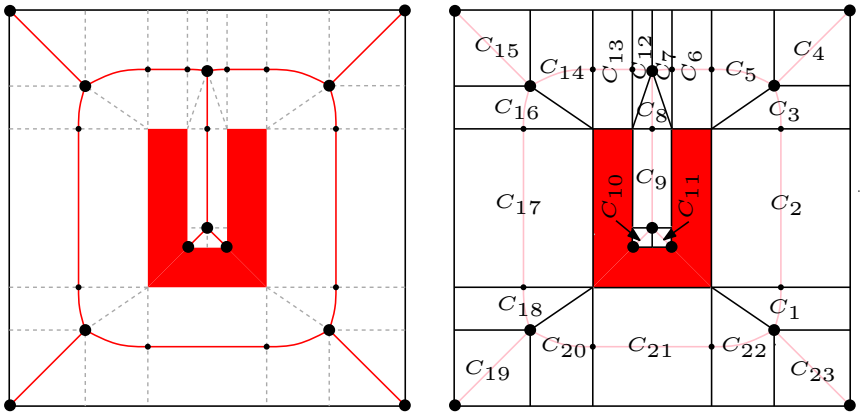


Figure 3.5: (a) An example of an Explicit Corridor Map. The ECM stores the medial axis vertices (large discs) and event points (small discs), together with the normals (indicated by dashed lines) that connect these vertices and points to their corresponding closest obstacle points. The medial axis edges (red curves) are not explicitly stored, as they can be inferred from the aforementioned data. (b) The ECM partitions the environment into a set of two-dimensional cells $C_1 ... C_{23}$ that facilitate proximity computations to nearest obstacles.

3.4.1 Explicit Corridor Map

The Explicit Corridor Map was introduced in [55] as an efficient data structure that captures the 2D walkable (free) space of the virtual environment. To be more specific:

Definition 3.2 (Explicit Corridor Map)**.** *The Explicit Corridor Map (ECM) is an enhanced graph $G = (V, E)$, where V and E represent the vertices and edges, respectively, of the medial axis and each edge is annotated with* event points *together with their* closest obstacle points *on both sides of the edge.*

The medial axis (MA) of the environment is a subset of its generalized Voronoi diagram. Namely, only Voronoi edges running into the reflex vertices of the obstacles are not part of the medial axis [22]. An MA edge captures the locus of points that are equidistant to at least two distinct closest obstacles. In a polygonal environment, such an edge is composed of 1-dimensional curves, that is, line segments and parabolic arcs. Consequently, for efficiency reasons, it is important to determine the minimum number of curves that describe the edge. In other words, we need to find the points along the edge where a curve must end and another one must start. Such points are referred to as *event points* and can be computed using simple linear algebra, as explained in [55].

Having determined the event points along an edge, we can then connect them, using line segments, to their closest points on the two obstacles that formed this edge. The resulting ECM structure partitions the environment into a set of non overlapping regions (two-dimensional *cells*) that fully cover the free space at linear storage cost. We refer the reader to Figure 3.5 for an example. Note that, as compared to the implicit Corridor Map depicted in Figure 3.4(a), the new structure does not require any sampling. It also provides the clearance to the obstacles for any free point and facilitates the computation of the nearest obstacle point for any query point.

The ECM can be efficiently constructed using graphics hardware, as proposed in [55]. We should also point out that the map is created only once during the preprocessing phase. Hence, it does not impose any limitations on the performance of our proposed path planner.

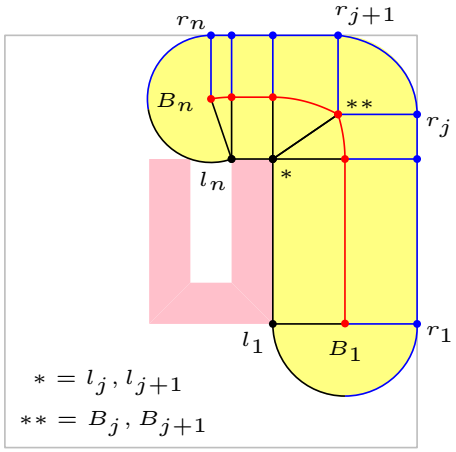


Figure 3.6: Example of an Explicit Corridor. Its left and right boundaries are indicated by black and blue curves, respectively.

3.4.2 Explicit Corridor

For generating the final path of a character, an Explicit Corridor needs first to be extracted from the Explicit Corridor Map. We refer the reader to the following section for more details on this procedure. The elements of the Explicit Corridor can be a part of a single (MA) edge in the graph, or can be formed by concatenating edges along the medial axis.

Definition 3.3 (Explicit Corridor)**.** *An Explicit Corridor is described by the sequence* (B_i, R_i, r_i, l_i), $1 \leq i \leq n$, *where* (B_i, R_i) *denotes a maximum clearance disc centered at an event point or vertex along an ECM edge and having radius* R_i. *For each center* B_i, *its left* (l_i) *and right* (r_i) *closest points to the obstacles' boundaries are also defined.*

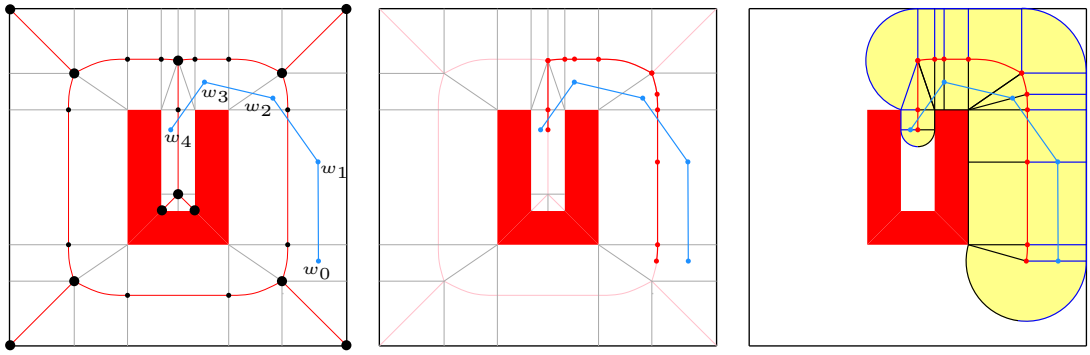
An example of an Explicit Corridor is given in Figure 3.6. By concatenating the left (l_i) and right (r_i) closest obstacle points with line segments and circular arcs, an explicit (continuous) representation of the corridor's boundaries can be computed in $O(n)$ time, where n is the number of the annotated clearance discs of the corridor [55]. These boundaries will play an important role in efficiently computing the boundary force of a character (see Section 3.5.1).

3.4.3 Retraction and Corridor Construction

The ECM provides a system of collision-free corridors that lie along the medial axis. Therefore, to create an Explicit Corridor around a given indicative route, we can retract that route onto the medial axis and then retrieve the corresponding corridor from the corridor map structure.

Let $\Pi_{\mathcal{IR}}$ be the character's indicative route. As mentioned in Section 3.3.2, such a route can be expressed as a series of waypoints $w_0 \ldots w_{n-1}$, where we require the distance $d(w_i, w_{i+1})$ to be at most a predetermined step size. We refer the reader to Figure 3.7(a) for an example. To retract any of these waypoints w_i $(0 \leq i < n)$ onto the medial axis, we need to find its closest obstacle point cp_o and then move w_i toward $\overrightarrow{cp_o w_i}$ until the closest obstacle point changes [155]. Such an operation, though, can be efficiently performed by exploiting the ECM data structure.

The ECM decomposes the walkable space of the environment into a series of two-dimensional cells, as shown in Figure 3.5(b). A cell C encloses the area defined by two



(a) Indicative route (b) Retracted route (c) Explicit corridor

Figure 3.7: Indicative route, medial axis retraction and corridor construction.

adjacent event points (or an event point and a vertex or two vertices) on a MA edge together with their corresponding closest points to obstacles. It consists of two sides (left and right w.r.t. the local 'direction of the edge') separated by the part of the medial axis tracing the event points.

Consequently, the problem of retracting a waypoint w_i can be simplified into determining the cell that embounds w_i. Let B, B' denote the event points of this cell and (l, r) and (l', r') their corresponding left and right closest obstacle points. Then, two distinct cases can be considered:

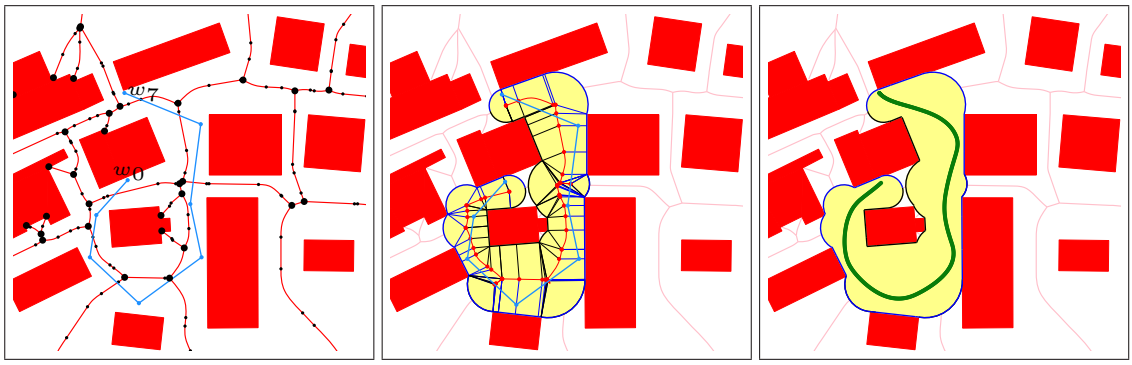
- The right (or left) side of the cell is a quadrilateral, see for example the cells C_1 or C_2 in Figure 3.5(b). This occurs if $r \neq r'$ (or $l \neq l'$). Then, if w_i is inside the quadrilateral, its closest obstacle point can be obtained by projecting w_i on the line segment connecting r and r' (l and l'). The retracted MA point of w_i along with its associated points on the obstacles' boundaries can also be easily obtained using simple linear algebra.

- The right (or left) side of the cell is a triangle, see for example the cells C_1 or C_3 in Figure 3.5(b). This occurs if $r = r'$ (or $l = l'$). Then, if w_i is inside the triangle, its closest obstacle point coincides with r (or l) and its retracted MA point together with its associated points on the obstacles' boundaries can be computed using linear algebra.

As a result, a straightforward approach to retract the entire indicative route of the character would be for each waypoint w_i to first determine its corresponding ECM cell and then retract w_i onto the medial axis using the procedure described above. However, such a naive implementation requires $O(nk)$ time, where n is the number of the waypoints and k the number of cells in the ECM structure. As this may be too expensive for large maps, we can obtain a more efficient solution as follows.

We store the minimum bounding box corresponding to each ECM cell into a regular 2D grid. We can then retrieve the cell enclosing the start position of the indicative route, i.e., w_0, by querying the grid map in $O(m)$ time, where m is the number of reported cells.[1] Let C denote this cell and e its corresponding MA edge in the ECM graph. Once the retracted MA point of w_0 is computed (as explained above), we perform a search over the cells of e, starting from C and iterating over its neighbouring cells until the cell enclosing the next waypoint w_1 is found. Note that if we reach the end of the edge at

[1] An ECM cell may span multiple grid cells.



(a) Indicative route (b) Retraction and corridor (c) Final path

Figure 3.8: Overview of the Indicative Route Method. (a) An indicative route determines the global route of the character. (b) This route is retracted onto the medial axis and an Explicit corridor is formed. (c) A force field method leads to a smooth path inside the corridor.

vertex v, we query the outgoing edges of v and continue the search in the cells of a new ECM edge. We repeat this process for each waypoint $w_i : i \in [1, n)$ storing its retracted MA point along with the visited event points and vertices.

If m is the time required to retrieve from the grid map structure the cell containing the first waypoint and x denotes the total number of ECM cells that are searched in order to determine the cells enclosing the remaining $n - 1$ waypoints of the indicative route, then the total runtime of our retraction procedure is $O(m + x + n)$. Assuming that the distance between adjacent waypoints is relatively small, the runtime complexity becomes $O(m + n)$. This is due to the fact that, given the cell enclosing the current waypoint, we can retrieve the cell enclosing the next waypoint in almost constant time.

An example of a retracted indicative route generated using our approach is shown in Figure 3.7(a). The resulting path is either a part of a single edge in the ECM, or a concatenation of edges forming a chain in the graph (like in the aforementioned figure). Consequently, an Explicit Corridor can be defined around that path, as discussed in Section 3.4.2 and illustrated in Figure 3.7(c). The method is very fast. Retracting, for example, the indicative route depicted in Figure 3.8(a) and constructing its corresponding corridor took 0.104 ms on a 2.4 GHz Core2 Duo CPU.

Finally, we should point out that our approach assumes that the indicative route does not backtrack and is monotonic. If this is not the case (see, e.g., Figure 3.8), we can still apply our retraction procedure. However, in order to properly construct the corridor, extra care should be taken while determining the direction of the closest obstacle points along the retracted route. For every part of the route that does not fully trace a specific edge in the ECM, its monotonicity changes and hence, the local direction of the corridor.[2]

3.5 Local Navigation in the IRM

Once the Explicit Corridor has been constructed, the character plans its final path inside the corridor using a force field approach. Several forces are defined that act on the character and influence its movement. First, the character must stay inside the corridor and hence, a repulsive force pushes the character away from the boundary of the corridor.

[2] An alternative approach is to segment the indicative route into a series of monotonic routes. Then, we can retract these routes onto the medial axis separately, using our proposed technique, and plan separate paths for the character inside the extracted corridors.

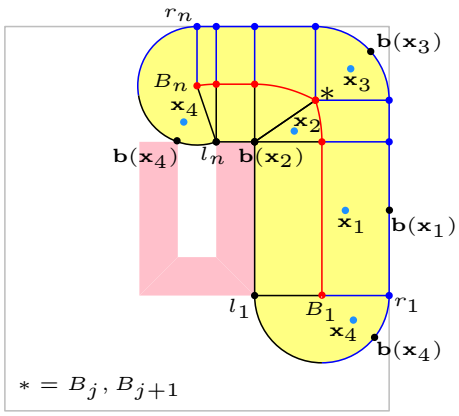


Figure 3.9: Examples of closest points $\mathbf{b}(\mathbf{x})$ on the boundary of an Explicit Corridor, given the current position $\mathbf{x}$ of the character. Four cases can be considered at positions $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ and $\mathbf{x}_4$.

Second, a steering force advances the character toward its goal. Third, a noise force allows for random variations in the character's path. Finally, a repulsive force is exerted on the character to resolve collisions with static and dynamic obstacles, as well as with other characters.

3.5.1 The Boundary Force

Given the Explicit Corridor, the character positioned at $\mathbf{x}$ lies inside one of the cells defined by adjacent (B_i, R_i, r_i, l_i) tuples and the boundary of the corridor, $1 \leq i \leq n$. As depicted in Figure 3.9, four distinct cases arise:

- The character is inside a right (or left) quadrilateral and thus, its closest point $\mathbf{b}(\mathbf{x})$ on the boundary of the corridor can be determined by projecting $\mathbf{x}$ on the line segment between r_i and r_{i+1} (l_i and l_{i+1}), as explained in Section 3.4.3.
- The character lies inside a right (or left) triangle and, as also discussed in our retraction approach, its closest boundary point $\mathbf{b}(\mathbf{x})$ coincides with r_i (or l_i).
- The character is inside the counter-clockwise arc tracing the disc (B_i, R_i) from r_i to r_{i+1} (or from l_{i+1} to l_i). This happens when adjacent discs have the same coordinates, i.e., $B_i = B_{i+1}$, but different right (left) closest points, i.e., $r_i \neq r_{i+1}$ ($l_i \neq l_{i+1}$). The character's closest point $\mathbf{b}(\mathbf{x})$ on the boundary of the circular arc can then be inferred using simple vector calculus.
- The character lies inside the counter-clockwise arc tracing the first clearance disc (B_1, R_1) of the corridor from l_1 to r_1 or the last clearance disc (B_n, R_n) from r_n to l_n. Its boundary point $\mathbf{b}(\mathbf{x})$ can be computed as above.

To efficiently compute the closest boundary point $\mathbf{b}(\mathbf{x})$, we keep track of the cell at which the character was located at the previous time step of the simulation. Starting from this cell, we then perform two searches, one headed toward the ending clearance disc of the corridor and one headed toward the beginning disc of the corridor. Using this approach, it takes on average constant time to determine the cell enclosing the character's position $\mathbf{x}$ and compute its corresponding $\mathbf{b}(\mathbf{x})$.

Having determined $\mathbf{b}(\mathbf{x})$, a repulsive force $\mathbf{F}_b$ from the boundary of the corridor is exerted on the character to ensure that it will remain inside the corridor. Since an individual prefers to keep a safe distance from walls, streets, buildings, etc. [206, 217], such a

force is only exerted if the Euclidean distance d_b between the character at position $\mathbf{x}$ and its corresponding closest point on the boundary of the corridor $\mathbf{b}(\mathbf{x})$ is below a threshold value. Let d_s denote the preferred safe distance. Then, the force is defined as follows:

$$\mathbf{F}_b(\mathbf{x}) = \begin{cases} c_b \dfrac{\mathbf{x} - \mathbf{b}(\mathbf{x})}{\|\mathbf{x} - \mathbf{b}(\mathbf{x})\|}, & \text{if } d_b - r \leq d_s \\ \\ 0 & \text{otherwise.} \end{cases} \tag{3.1}$$

The scalar $c_b = (d_s + r - d_b)/(d_b - r)^\kappa$ is chosen such that the force will become infinite when the character and the boundary point touch[3], where the constant $\kappa \geq 0$ indicates the steepness of the repulsive potential. As κ increases, the potential will increase in steepness toward the boundary of the corridor. In addition, by modifying the safe distance d_s, a wide variety of behaviours can be obtained. For example, a small safe distance allows the character to get close to obstacles, whereas a large one makes the character stay in the middle of the corridor.

However, to guarantee that in each cycle of the simulation, the character will not collide with the environment, a lower bound on the safe distance is set based on the actual distance that the character can traverse given its dynamic constraints. Furthermore, to avoid oscillatory behaviour, an upper bound is also determined by taking into account the minimum clearance $R^{\min}$ of the corridor, that is the minimum radius among all the clearance discs (B_i, R_i) of the corridor.[4] Thus, the set of admissible safe distances $\mathcal{D}$ for the character is defined as:

$$\mathcal{D} = \left\{ d_s \,\middle|\, v^{\max}\Delta t + r < d_s < R^{\min} - r \right\}. \tag{3.2}$$

3.5.2 The Steering Force

While the boundary force keeps the character inside the corridor, an additional force is needed to guide the character forward toward its goal position. For this purpose, we move an *attraction point* along the indicative route of the character. Based on this attraction point, a steering force $\mathbf{F}_s$ is generated that steers the character positioned at $\mathbf{x}$ toward the attraction point $\mathbf{p}_\alpha(\mathbf{x})$. Let c_s be a constant specifying the relative strength of the force. Typically, as the character prefers to move with a certain speed v^{pref}, we set c_s equal to v^{pref}. Then, $\mathbf{F}_s$ is given by:

$$\mathbf{F}_s(\mathbf{x}) = c_s \frac{\mathbf{p}_\alpha(\mathbf{x}) - \mathbf{x}}{\|\mathbf{p}_\alpha(\mathbf{x}) - \mathbf{x}\|}. \tag{3.3}$$

Note that when simulating crowds of virtual humans, the aforementioned steering force can be slightly modified as proposed by Helbing and Molnár [77], that is

$$\mathbf{F}_s(\mathbf{x}) = \frac{1}{\tau}\left(v^{\text{pref}} \frac{\mathbf{p}_\alpha(\mathbf{x}) - \mathbf{x}}{\|\mathbf{p}_\alpha(\mathbf{x}) - \mathbf{x}\|} - \mathbf{v} \right), \tag{3.4}$$

where the parameter $\tau > 0$ allows the character to gradually adapt its current velocity $\mathbf{v}$. Other variants can also be used to encourage certain types of behaviour, such as the steering controller proposed by Boulic [12] that enables a character to reach a given attraction point with a prescribed orientation.

[3] In practice, c_b will never reach infinity, since we require the clearance along the indicative route to be strictly larger than r and the same for the safe distance d_s.

[4] If $d_s \geq R^{\min} - r$ the character will be continuously pushed from the left to the right side of the corridor and vice versa. This is due to the fact that d_b will always be less than $r + d_s$ and hence, $\mathbf{F}_b$ will be exerted on the character at every time step of the simulation.

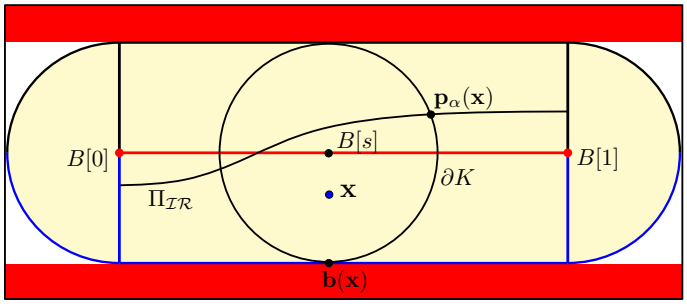


Figure 3.10: The attraction point $\mathbf{p}_\alpha(\mathbf{x})$ for the character positioned at $\mathbf{x}$. It is defined as the furthest intersection point between the character's indicative route $\Pi_{\mathcal{IR}}$ and the maximum clearance disc centered at the retracted point $B[s]$ of the character's position.

3.5.3 Choosing an Attraction Point

To advance the character along its corridor, the chosen point of attraction is of paramount importance. In every cycle of the simulation, the attraction point should steer the character toward its goal, ensuring at the same time that it does not get stuck in local minima.

Up to now we referred to the character's corridor as a discrete sequence of n annotated disks. However, their closest obstacle points embody a continuous sequence, i.e., an infinite number of maximum clearance disks exist (if $n > 1$) whose center points lie along the retracted indicative route of the character. Let us refer to this route as the backbone path of the corridor. Let also $K(B[s], R[s])$ be the clearance disk with radius $R[s]$ centered at $B[s]$, where $B[s]$ denotes the (retracted) point on the backbone path of the corridor corresponding to the current position $\mathbf{x}$ of the character, $s \in [0, 1]$.[5] The attraction point $\mathbf{p}_\alpha(\mathbf{x})$ can then be defined as the furthest point on the character's indicative route $\Pi_{\mathcal{IR}}$ that still intersects the boundary ∂K of the clearance disc K (see Figure 3.10):

$$\mathbf{p}_\alpha(\mathbf{x}) = \max_{s' \in (s,1]} \left\{ \Pi_{\mathcal{IR}}[s'] \mid \Pi_{\mathcal{IR}}[s'] \cap \partial K \neq \emptyset \right\}. \tag{3.5}$$

Such an intersection point always exists, since there is a one-to-one mapping between the character's indicative route and the backbone path. Note that if the character's goal lies inside the intersected clearance disc, it is used as the next attraction point of the character. Otherwise, selecting the intersection point with the largest index drives the character forward toward its goal.

Theorem 3.1. *The attraction point $\mathbf{p}_\alpha(\mathbf{x})$ guarantees that, in the absence of any static and dynamic obstacles inside the corridor, a path to the goal will always be found.*

Proof: The furthest intersection point between the character's indicative route and the clearance disc $K(B[s], R[s])$ lies always ahead of the character, i.e., $\|\mathbf{p}_\alpha(\mathbf{x}) - \mathbf{x}\| > 0$, and hence, the magnitude of the steering force $\mathbf{F}_s$ is always larger than zero (except when the goal position is reached). In addition, the intersection point can never be located on the closest boundary point $\mathbf{b}(\mathbf{x})$ that corresponds to the character's current position, since we require the clearance at every point along the indicative route to be strictly larger than the radius r of the character. Therefore, the boundary force $\mathbf{F}_b$ does not cancel out the steering force $\mathbf{F}_s$ and consequently $\mathbf{F}_s + \mathbf{F}_b \neq \mathbf{0}$. The latter guarantees that the character cannot get trapped in local minima. Therefore, a path to the goal can always be found. ■

[5] As we already know (from Section 3.5.1) the cell enclosing the current position $\mathbf{x}$ of the character along with the character's closest point $\mathbf{b}(\mathbf{x})$ on the boundary of the corridor, we can retract $\mathbf{x}$ onto the backbone path B in constant time using simple geometric calculations.

Note that even when we include the noise force $\mathbf{F}_n$ to the forces exerted on the character, as proposed in Section 3.5.4, Theorem 3.1 still holds, since $\mathbf{F}_n$ differs from the steering direction by at most $\pi/2$. However, when the character has to avoid collisions with static and dynamic obstacles while traversing its corridor, see, e.g., Section 3.6, extra care should be taken so that the total collision response force applied on the character does not cancel out $\mathbf{F}_s + \mathbf{F}_b$. Restricting for example the direction of this force to be perpendicular to the steering direction is sufficient for singularity-free motion.

3.5.4 Path Variation

To take into account random variations in the paths that the characters follow and generate a (slightly) different path every time the same path planning problem has to be solved, a noise force $\mathbf{F}_n$ is also introduced in our model. Like in Chapter 2, we use a Perlin noise [168] function to control the direction of the force, ensuring that it changes smoothly during the simulation. In our problem setting, Perlin noise is implemented as a function over time. The current time step of the simulation is given as an input and a direction for the force is returned that varies pseudo-randomly. Let θ denote this direction, expressed as an angle of rotation around the current steering vector. Then, the noise force is defined as:

$$\mathbf{F}_n(\mathbf{x}) = c_n \, \mathcal{R}(\theta) \, \frac{\mathbf{p}_\alpha(\mathbf{x}) - \mathbf{x}}{\|\mathbf{p}_\alpha(\mathbf{x}) - \mathbf{x}\|}, \tag{3.6}$$

where $\mathcal{R}(\theta)$ represents the 2D rotation matrix.

The constant c_n specifies the relative strength of the force. It must be small with respect to the magnitude of the steering force, whereas $\theta \in [-\pi/2, +\pi/2]$. As a result, the character retains its steering direction while at the same time performs small random displacements. However, care should be taken not to vary the direction of the force, that is, the noise function, too frequently. A low frequency leads to smooth movements, while a relatively high frequency generates erratic and unusable behaviour. We refer the reader to Section 2.4 for more details.

3.5.5 Time Integration and Final Path

Having computed the steering, boundary and noise forces, the final force $\mathbf{F}$ exerting on the character at position $\mathbf{x}$ is defined as:

$$\mathbf{F}(\mathbf{x}) = \mathbf{F}_s(\mathbf{x}) + \mathbf{F}_b(\mathbf{x}) + \mathbf{F}_n(\mathbf{x}). \tag{3.7}$$

The force $\mathbf{F}$ results in an acceleration term as described by Newton's second law of motion, i.e., $\mathbf{F} = m\mathbf{a}$, where m is the mass of the character and $\mathbf{a}$ its acceleration. Without loss of generality, we assume a unit mass. Then the position of the character at time t is computed by integrating the equation of motion, that is

$$\frac{d^2(\mathbf{x}(t))}{dt^2} = \mathbf{F}(\mathbf{x}(t)). \tag{3.8}$$

Any numerical integration scheme, such as Euler's method, can be used to solve this differential equation. However, since we are dealing with forces that may have different scales and can vary quickly, a stable integration scheme like Verlet integration [232] is recommended. To retain stability and avoid stiffness a relatively small time step Δt should also be preferred (see Section 3.8).

The final path Π_f of the character can then be obtained by iteratively integrating $\mathbf{F}$ over time, while updating the character's position, velocity, and attraction point. As an example, consider Figure 3.8(c). It displays a *smooth* (C^1-continuous) path generated using our proposed method. We also refer the reader to Figures 3.4(f) and 3.13(b) for additional examples.

Theorem 3.2. *The IRM generates a path Π_f that is smooth, i.e., C^1-continuous.*

Proof: The final path Π_f is obtained by integrating $\mathbf{F}$ two times, which adds two degrees of continuity to the path that the attraction point follows. Even though this path can be discontinuous, it can be easily shown that double integration leads to C^1 continuity. Therefore, it remains to prove that $\mathbf{F}$ can indeed be integrated, that is, the denominators of all the forces in Equation 3.7 are larger than zero. Regarding the steering $\mathbf{F}_s$ and noise $\mathbf{F}_n$ forces, as already discussed in the proof of Theorem 3.1, the attraction point $\mathbf{p}_\alpha(\mathbf{x})$ lies always ahead of the character, and hence, $\|\mathbf{p}_\alpha(\mathbf{x}) - \mathbf{x}\| > 0$. As far as the boundary force $\mathbf{F}_b$ is concerned, since the clearance along the indicative route is strictly larger than the radius r of the character and the same applies for the character's safe distance d_s, the character can never be located on its closest boundary point, i.e., $\|\mathbf{b}(\mathbf{x}) - \mathbf{x}\| > 0$ and also $d_b - r > 0$. Consequently, as all terms stay positive, $\mathbf{F}$ can be integrated twice, resulting in a C^1-continuous path. ■

3.6 Avoiding Obstacles

Although the Explicit Corridor provides a collision-free area around the indicative route, the character may still have to avoid a number of obstacles while moving inside the corridor. Such obstacles can either be small static obstacles that were not taken into account when the Explicit Corridor Map was created, or dynamic obstacles, such as other characters and moving entities. Thus, the force $\mathbf{F}$ may have to be updated by adding a collision response force toward these obstacles. In the following, we discuss a simple approach to derive such a force.

Let $\mathrm{O} \in \mathbb{R}^2$ denote a set of n obstacles that the character needs to avoid, while planning its motion inside the corridor. Every obstacle $O_i : i \in [1:n]$ exerts a repulsive force, which in its simplest form is monotonically decreasing with the Euclidean distance d_i between the obstacle having radius r_i and the character at position $\mathbf{x}$.[6] Typically, such a force is only applied if the character lies inside the influence region of the obstacle defined by d_{o_i}, that is, if $d_i - r_i - r \leq d_{o_i}$. Let k_r be a parameter used to scale the effect of the repulsive force on the character. Then, the total obstacle avoidance force $\mathbf{F}_o$ that acts on the character can be described by:

$$\mathbf{F}_o(\mathbf{x}) = \sum_{i=1}^{n} c_o \frac{\mathbf{x} - O_i}{\|\mathbf{x} - O_i|}, \text{ where } c_o = \frac{k_r}{d_i - r_i - r}. \tag{3.9}$$

Figure 3.1(b) shows an example path created by the IRM, using the medial axis as the indicative route of the character. The repulsive potential described above has been applied to avoid a number of static obstacles inside the extracted corridor. Clearly, the resulting path is more natural than the path produced by the CMM depicted in Figure 3.1(a).

[6] In case O_i is not (approximated as) a disc, $r_i = 0$.

Although we do not distinguish between static and dynamic obstacles, the problem becomes more challenging when many moving entities populate the virtual world. Therefore, more elaborate avoidance approaches can be used to simulate, for example, human crowds, like the models proposed in Chapters 6 and 7.

3.7 Crowd Simulation in the IRM

Up until now, we have assumed that the indicative route of a character is provided by the level designer. However, if we want to steer crowds of characters, this process would be inefficient and time consuming. Thus, to facilitate the creation and extraction of indicative routes we introduce the notion of *indicative networks*.

Definition 3.4 (Indicative Network)**.** *An indicative network $\mathcal{IN} = (V, E)$ is a graph that can be used to lead the characters toward their goals. Each vertex $\mathrm{v} \in V$ of the network corresponds to a collision-free point in a 2D workspace, whereas each edge $e \in E$ corresponds to an indicative route Π_e that a character can follow.*

Such a network is constructed in a pre-processing phase. In practice, any roadmap graph that captures the connectivity of the free space is suited for guiding the motions of the characters [121]. For example, we can use the medial axis itself to force the characters to stay in the middle of the environment. We can also compute an indicative network based on the Voronoi-Visibility diagram [237], allowing characters to take shortcuts when there is enough clearance.

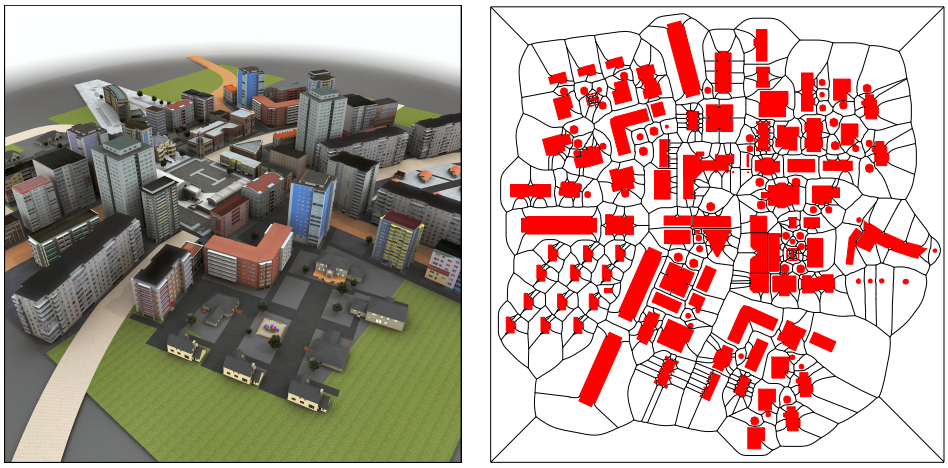
Alternatively, an indicative network can be determined manually by the level designer, to encourage certain behaviour, e.g., characters that prefer to stay on the sidewalks of a road. The network should be flexible enough so that the designer can easily manipulate it by adding, changing, or removing indicative routes. Additional information should also be easily incorporated. For example weights can be added to the graph to indicate appealing locations the characters would like to visit or clever routes they would prefer to follow in case of a tactical game, etc. We refer the reader to Figure 3.12(a) for an example of a typical control network.

Given an indicative network, the actual motion of each character can be computed in three online phases:

a) The indicative route of the character is extracted from the network. To obtain the route, we first connect the start position $\mathbf{x}_S$ of the character to the network. This can be done by adding an edge between the start position and the closest visible point that lies on one of the local routes Π_e of the network. We proceed in the same way for the goal position $\mathbf{x}_G$ of the character. Then, we run an A* shortest path algorithm on the network graph to retrieve the character's indicative route.

b) The computed route is retracted onto the medial axis and a corridor is constructed around that path as described in Section 3.4.3. Alternatively, to increase the performance, we can retract the entire indicative network in a preprocessing phase and store the corresponding (retracted) graph. Then, in the query phase, we only need to retract the two edges that connect the start and goal position of the character to the indicative network.

c) Finally, we (move an attraction point along the indicative route and) use the force field approach discussed in Section 3.5 to guide the character's motion through its corridor.



(a) City Environment (b) Footprint and ECM

Figure 3.11: A 3D model of our test environment, its footprint and Explicit Corridor Map. For clarity, the closest obstacle points in the ECM structure have been left out.

Since we are dealing with many moving characters, we have to choose a more realistic collision avoidance approach than the one described in Section 3.6, so that the characters can evade each other naturally. In general, the IRM can be easily combined with most of the existing agent-based methods for local collision avoidance. In Chapter 4, for example, we combine the IRM with the social force model proposed by Helbing and Molnár [77], whereas in Chapter 5 we integrate the IRM with the predictive avoidance model introduced in Chapter 6. In both of these schemes, collision avoidance is achieved by means of socio-psychological and physical forces acting on each character. Their main difference, though, is that in the predictive model the characters do not repel each other, but rather anticipate future situations avoiding collisions in advance and with minimal effort.

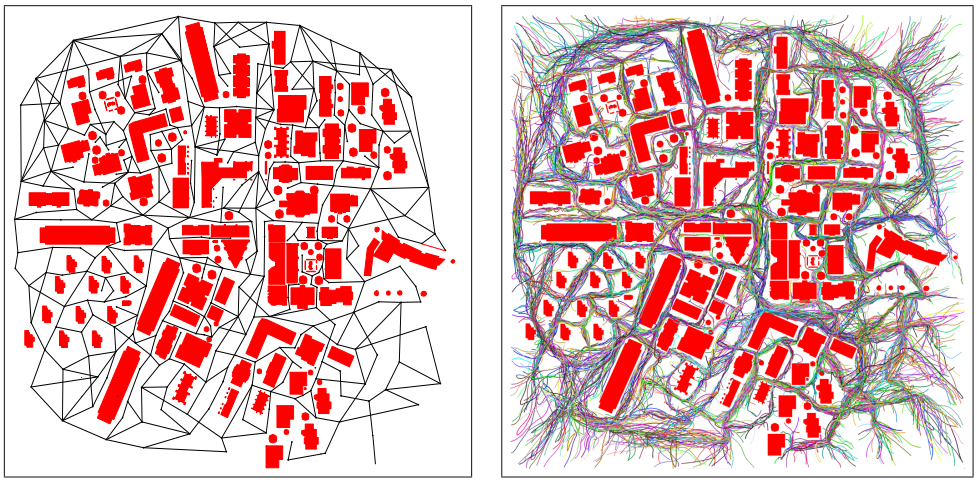
Note that for computing the avoidance forces of the characters, we have to determine for each character its nearest neighbours. Since this operation is performed at every step of the simulation, an efficient data structure for answering nearest neighbour queries has to be implemented. For example, a spatial hash data structure or a 2D grid map that stores and updates at every simulation step the locations of the characters can be employed [159, 183, 193].

3.8 Experiments

We have implemented our proposed IRM to experimentally test its usability in real-time applications and validate the quality of the generated motions. All the experiments were performed on a 2.4 GHz Core2 Duo CPU (on a single thread) with 4 GB memory.

3.8.1 Experimental Setup

The experiments were conducted for the environment depicted in Figure 3.11(a). This is a model of a virtual city. The city measures 500x500 meter. Its geometry is composed of 220K triangles, while its 2D footprint, displayed in Figure 3.11(b), consists of 548 convex polygons that form the input obstacles of the Explicit Corridor Map structure. The ECM was computed in 114 ms and led to 1442 vertices and 1615 edges. On top of the ECM, we



(a) Indicative Network (b) 2000 paths generated by the IRM

Figure 3.12: The Indicative Network used in our experiments and example paths generated by our framework combined with a local method for collision avoidance.

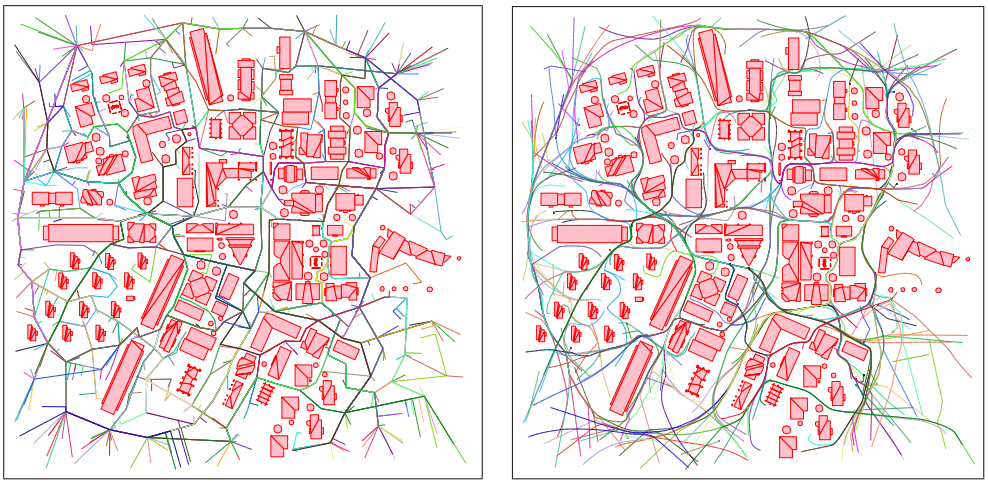
overlaid a rather sparse 2D grid structure consisting of 50x50 cells to facilitate the retraction process, as explained in Section 3.4.3. For obtaining indicative routes, the network shown in Figure 3.12(a) was used that contains 347 vertices and 514 edges.

In our simulations, we used Verlet integration with a small step size $\Delta t = 0.1\,\mathrm{s}$ to keep the errors minimal. The preferred speed of a character was set to $v^{\mathrm{pref}} = 1.4\,\mathrm{m/s}$ and its diameter to $2r = 0.7\,\mathrm{m}$ corresponding to the average walking speed and body diameter of a typical human, respectively. The actual speed of the character was limited to a maximum value of $v^{\max} = 2\,\mathrm{m/s}$.

In the experiments described in Sections 3.8.2 and 3.8.3, we used Equation 3.3 to determine the steering force $\mathbf{F}_s$, whereas Equation 3.4 was preferred for the crowd simulation experiments of Section 3.8.4 ($\tau = 0.5\,\mathrm{s}$). Regarding the boundary force $\mathbf{F}_b$, we set the safe distance from static obstacles to $d_s = 1\,\mathrm{m}$ and the constant $\kappa = 1$. Finally, the relative strength of the noise force $\mathbf{F}_n$ was set to $c_n = 0.5$.

3.8.2 Retraction Efficiency

A key component in our framework is the retraction of the indicative routes onto the medial axis. Overall, using the Explicit Corridor Map structure along with the technique described in Section 3.4.3, we can retract an indicative route and compute its corresponding corridor very efficiently. As an example, we randomly extracted 10,000 (valid) routes from the indicative network shown in Figure 3.12(a). On average, it took $0.097\,\mathrm{ms}$ to connect a query on the graph and retrieve its shortest indicative route using an A* search. Then, the average retraction and corridor construction time was only $0.289\,\mathrm{ms}$. Note that, due to the large size of our test environment, this time corresponds to relatively long indicative routes, having an average path length of $366.252\,\mathrm{m}$. Furthermore, using our technique, only 11.4%, on average, of all the ECM cells (491 out of the 4303) had to be searched in order to perform the retraction of a route. This ensured fast running times, since the runtime complexity of our retraction algorithm scales with the number of inspected ECM cells (see Section 3.4.3 for more details). Finally, we should point out that the step of querying the grid structure to retrieve the first ECM cell that encloses the start point of the indicative route is negligible in running time ($0.002\,\mathrm{ms}$).



(a) Indicative Routes (b) Final Paths

Figure 3.13: Randomly extracted indicative routes from the network depicted in Figure 3.12(a) and their corresponding paths computed by the IRM.

	Curvature $(\mathrm{rad}^2/\mathrm{m}^2)$	Path Length (m)
Indicative Routes	7.75 $(\pm\, 4.49)$	347.66 $(\pm\, 158.94)$
Final Paths	0.59 $(\pm\, 1.01)$	234.71 $(\pm\, 157.34)$

Table 3.1: Curvature and path length statistics for the indicative routes and their corresponding final paths generated by the IRM. The reported statistics are the averages over 500 paths. The curvature is measured as the sum total of square curvature samples along a path. Numbers in parentheses denote standard deviations.

As discussed in the previous section, given an indicative network, we can also use an offline retraction strategy. This means that we retract the entire network upon the construction phase and store its retracted graph. Then, in the query phase, we only have to retract the two edges that connect the start and goal to the indicative network. The rest of the indicative route lies on the indicative network and thus, we can easily retrieve its counterpart on the retracted graph.

To test the offline approach, we first retracted the indicative network of Figure 3.12(a). This took 22.38 ms. Then, we ran the same queries as above. As expected, the time to retract an indicative route (or more accurately, the start and goal edges of the route) and compute its corresponding corridor was insignificant (0.004 ms) and, in practice, the A* search dominated the overall runtime performance.

3.8.3 Quality Evaluation

In the IRM, as soon as an (explicit) corridor has been constructed around an indicative route, a potential function steers the character inside this corridor, as described in Section 3.5. To quantitatively evaluate the quality of the resulting motions, we randomly extracted 500 routes from our indicative network. Then, for each final path, computed in 3.49 ms (average time), we calculated the sum total of squared curvature samples along the path to measure its smoothness [87]. We refer the reader to Table 3.1 for the corresponding results.

As can be inferred from the table, on average, the total path curvature is quite low,

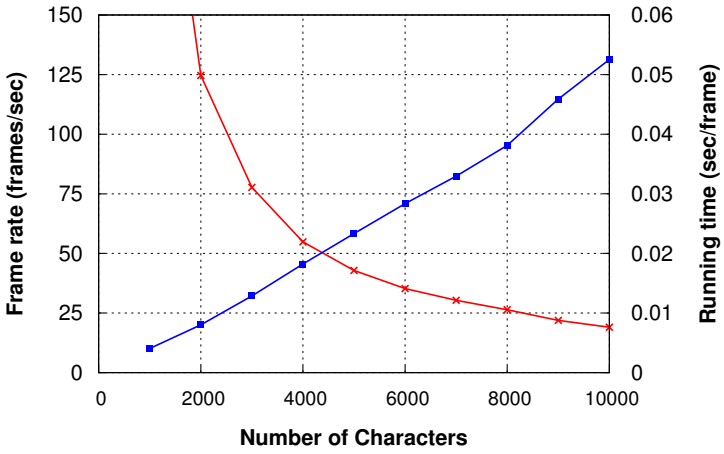


Figure 3.14: The performance of our approach in the virtual city environment as a function of the number of simulated characters. The square-marked graph provides the runtime per simulation step (right-axis), whereas the x-marked graph indicates the corresponding frame rate (left-axis). The reported results are simulation only and were obtained by combining the IRM with the predictive collision avoidance model proposed in Chapter 6.

indicating that characters follow smooth paths inside corridors. This become more obvious when we consider the indicative routes given as input to the IRM. Such routes consist of line segments connecting together and thus, they typically lack smoothness and have high curvatures. As expected, a Student's t-test (5% significance level) showed a significant difference in curvature between the final paths generated by our approach and their corresponding indicative routes, $p < 0.001$. See also Table 3.1 and Figure 3.13 for numerical and visual comparisons, respectively.

Statistical analysis also showed significant differences in the lengths between the final paths and their indicative routes, $p < 0.001$. In general, due to the way the attraction point is defined, i.e. the furthest advanced intersection point between the indicative route and the clearance disc of the character's retraction, the character tends to cut corners and take shortcuts while following its indicative route (see Section 3.5.3 for details). Consequently, as depicted in Table 3.1, the final path is, on average, shorter than the indicative route.

3.8.4 Performance

To test the applicability of the IRM in real-time applications, like computer games, we combined our approach with the predictive model for local collision avoidance proposed in Chapter 6. In our implementation, for performance gain, each character considered interactions with only a fixed, constant number of at most $k = 5$ nearby characters at each step of the simulation. For that reason, a circular neighbour region around each character was defined having a radius of 5 m. We refer the reader to Section 6.6 for all the other default parameter values that we used.

Given N characters, the collision avoidance algorithm runs in $O(N)$ time per simulation cycle.[7] Obviously, as discussed in the previous section, one of the key steps in

[7] Actually the algorithm has a runtime complexity of $O(Nk)$ per simulation cycle, where k denotes the maximum bound of characters within a given neighbour region. By requiring k to be fixed, the runtime is reduced to $O(N)$. See Section 6.5 for details.

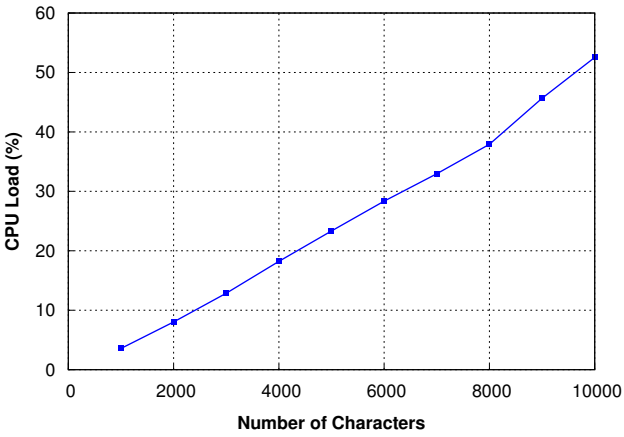


Figure 3.15: The CPU-load in the virtual city environment for a varying number of characters. The reported results were obtained by combining the IRM with the local method for collision avoidance proposed in Chapter 6.

affecting the overall performance of the algorithm is how to select for each character its nearest neighbours. In our implementation, we used an efficient bin-lattice spatial subdivision method, similar to the one described in [183]. Thus, the overall neighbour selection step also exhibits an almost linear runtime behaviour per simulation step.

To verify the overall performance (IRM + local collision avoidance), we selected a varying number of characters and placed them randomly in the city environment. Each character had to advance toward a random goal position by selecting an indicative route from the network depicted in Figure 3.12(a). When it had reached its destination, a new goal was assigned. See Figure 3.12(b) for an example.

We performed experiments for up to $10,000$ characters and set the maximum number of simulation steps to $1,000$. We measured the average time that it takes to compute one cycle of the simulation, as well as the corresponding frame rate evolution. To efficiently compute the corresponding corridors of the characters, we used the offline retraction strategy. Note also that, during our benchmarks, the scene and character rendering was disabled, as our goal was to solely analyze the path planning cost of our combined planner. The results (on a single core) are shown in Figure 3.14.

As the figure indicates, the running time increases nearly linearly with the number of characters.[8] It is also clear that our model can easily handle thousands of virtual characters in real-time. Even in very crowded scenarios, involving $10,000$ characters, the running time roughly corresponds to a frame rate of 19 fps.

Besides the (average) running time, we are also interested in the amount of CPU-load that is required for planning the characters' paths. Typically, in interactive virtual environments and games, only a small fraction of the processor time is scheduled to the path planner, whereas the rest of the CPU resources are allocated to other tasks, such as character animation, rendering, physics and so on. To see the results in the right perspective, we defined the CPU-load as the total amount of CPU (running) time that is required to generate one second of characters' movements, that is, the total CPU time divided by the average time that a character requires to traverse its path.

[8] The jump in the graph at $9,000$ is due to the fact that, in such cluttered environments, the characters tend to get stuck while interacting with each other along low clearance edges. This effect can be alleviated either by replanning the indicative routes of the characters or by employing a different global planning strategy than selecting the shortest indicative route in the network (see, for example, Chapter 4).

As our simulations run for $1,000$ iterations and each iteration (pre)calculates 0.1 seconds of movement for each character ($\Delta t = 0.1$ s), the average traversed time remains fixed to 100 s for all of our experiments. Consequently, the CPU-load also increases almost linearly with the number of the simulated characters.[9] We refer the reader to Figure 3.15 for the corresponding results.

As an example, consider the processor usage that was needed for $6,000$ characters. On average, it took 4.72 ms to compute a character's path, whereas the average time to traverse that path was 100 s, as explained above. Hence, $283.2\ ms$ per second traversed time were required to simultaneously plan the motions of all 6,000 characters, which implies a CPU-load of 28.32%. Note that even for $10,000$ characters, our method can efficiently plan paths at interactive rates requiring less than 52% of the processor power.

3.9 Conclusions

In this chapter, we have presented a new method for high-quality path planning, the Indicative Route Method (IRM). In the IRM, an indicative route directs the global motion of the character, whereas a potential field approach is used to locally guide the character through an (explicit) corridor around this route.

We have experimentally shown that our method can be used for real-time planning of a large number of characters in complex and dynamic environments. Its main advantage over the majority of existing planners is that it does not limit the global behaviour of the characters. An indicative route can be computed using any of the well-known methods in the motion planning literature, or even manually provided by the user. The route is then given as an input to the IRM and a smooth, C^1-continuous, path is returned based on a potential function.

Overall, our proposed steering potential is relatively easy to implement and can be combined with existing schemes for local collision avoidance providing enough local flexibility for the character when dynamic obstacles (or other moving characters) have to be avoided. Bearing also in mind that alternative steering controllers can also be integrated into our method to allow for a more diverse range of behaviours (see, e.g. [12, 77, 99, 182]), the IRM offers a powerful framework for controlling and directing individual agents and crowds of virtual characters.

An obvious avenue for future work would be to extend our approach to 2.5D or 3D space. Preliminary results toward this direction, using the IRM, are quite promising [231]. A future challenge would also be to address changeable environments, in which obstacles may appear, disappear or even change their configurations over time and paths can become blocked or invalid due to the user interaction. It must be pointed out, though, that in such highly dynamic and interactive worlds, where the character may need to replan its indicative route too often, retracting the entire route onto the medial axis may be a waste of time. A better approach would be to perform the retraction on-the-fly, that is, while the character follows its indicative route, we partially retract it and compute its corresponding (partial) corridor. Finally, it would be interesting to investigate ways of automatically creating networks of indicative routes based on how humans behave in real life. To facilitate this, we can, for example, exploit video recordings and motion capture data in order to understand how people solve a typical path planning problem, what parameters affect their path choices and how these parameters influence their preferred indicative routes.

[9] In fact, the corresponding graph is a scaled version of the performance graph depicted in Figure 3.14.

Chapter 4

Density Constraints for Crowd Simulation

In the previous chapter, we have introduced the Indicative Route Method (IRM) as a new approach for path planning and crowd simulation. In the IRM, an indicative route determines the preferred (global) route of the character. Such a route can be indicated manually by the level designer or computed automatically using, for example, an A* algorithm on a coarse grid.

In this chapter, we present a simple technique to improve the indicative routes that characters prefer to follow by taking the crowd density into account. Consequently, in our simulations, characters intelligently plan their paths around congested areas, favouring less dense regions. This not only leads to a smoother and more efficient crowd flow, but also reduces the amount of energy and time consuming avoidance maneuvers that the characters have to perform.

4.1 Related Work

Over the past ten years several approaches have been proposed that attempt to govern the global behaviour of virtual characters and generate visually compelling crowd flows. Several of these approaches are field-based and either automatically compute or let the user manually author velocity fields that the characters can follow.

To that extent, Chenney [19] designed divergence-free flow tiles to guide the motions of the characters through the virtual world without requiring any explicit collision avoidance mechanism. Due to the static nature of the tiles, though, no interactions among the characters can take place. The work of Jin *et al.* [96] allows the user to specify flow fields in the environment and uses a RBF (radial basis function) interpolation scheme to compute a continuous navigation field that directs the movements of virtual crowds. Although this technique can interactively modify the behaviour of a crowd, it cannot guarantee singularity-free navigation.

In contrast, the approach of Patil *et al.* [164] generates a smooth, goal-directed navigation field that is free of local minima and can be combined with existing schemes of local collision avoidance to safely steer the characters toward their destinations. This approach can resolve congestion and generate a wide variety of macroscopic behaviours. However, like in the aforementioned approaches, the generated behaviours are user-specified, as compared to our algorithm that automatically directs character flows based on the crowd density.

An alternative approach has been proposed by Treuille *et al.* [218]. Their model converts a virtual crowd into a density field and computes a dynamic velocity field that simultaneously integrates global navigation with local collision avoidance. Although related to our work, their approach is limited to a small number of homogeneous groups

of characters moving toward common goals. In contrast, we focus on independent characters that have distinct characteristics and goals.

Loscos *et al.* [136] demonstrated that desirable crowd properties, such as congestion avoidance, can be obtained using simple grid-based rules. In their approach, a 2D grid subdivision scheme is used for collision detection and avoidance. To increase the realism of the local interactions, the directions of the characters are also stored into a 2D grid structure. The resulting direction field is dynamically updated during the simulation and is used to provide coordination among the characters, controlling the emergence of pedestrian flows.

Similar to this idea, Jansen and Sturtevant [94] proposed a new approach to cooperative pathfinding by exploiting a data structure that learns about the movements of the characters in the virtual world. Upon planning, a grid-based A* search is used that adds an extra penalty to the A* cost function when characters try to move against the flow. Consequently, the characters are encouraged to follow the directions that have been taken by previous characters, which results in emergent cooperative behaviour. However, as the number of the characters grows, the problem complexity and the running time increases considerably, making this approach insufficient to model large crowds. Furthermore, the paths resulting from the A* search can be of low quality and thus, extra care should be taken to smooth them.

4.2 Overview

Our model is inspired by the work of Loscos *et al.* and the recent approach of Jansen and Sturtevant. We focus, though, on the crowd density rather than on the flow field. Our goal is not to simulate how real pedestrians behave in a crowd, but to provide a crowd model that looks pleasing and convincing to the viewer. Therefore, in our simulations, we aim for characters that prefer to move toward low-density areas. This leads to energy-efficient paths and a better spread of the characters throughout the environment. Our approach allows the users to set the trade-off between path length and preferred density, making it a flexible framework that can simulate a variety of behaviours, even for characters navigating in the same scene.

Globally speaking, our method creates a *density map* that provides information regarding the crowd density in the environment and is also used for the global planning of the characters. This density map consists of a coarse grid in which each cell stores information about the number of characters that traversed this cell in the recent past. The map is dynamic and is continuously updated, while the characters navigate through the virtual world. For the latter, we use an A* search on the density map and compute a path for a given character. Such a path, though, can be rather unrealistic and might not avoid other moving characters. Hence, we do not use this path directly, but as an indicative route. We then apply the Indicative Route Method to guide the character toward its goal position. As a result, a high-quality and collision-free path is generated that avoids crowded areas, keeping at the same time the computational cost of our approach low, since we only need a coarse density map.

4.3 Density Adaptation

In this section, we provide a detailed description of our proposed approach. It is based on a simple idea that characters adapt their path planning choices depending on the den-

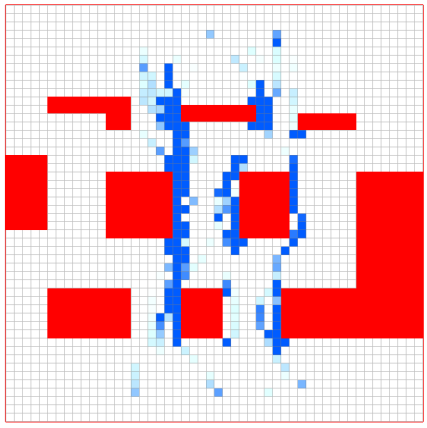


Figure 4.1: An example of a density map used in our simulations. Darker cells represent areas with high crowd density.

sity of the environment. This leads to characters that can intelligently plan their routes, avoiding areas of high congestion.

4.3.1 Density Maps

To incorporate the notion of crowd density into our model, a density map is constructed. This map discretizes the world into a regular grid and for each grid cell a scalar value is assigned indicating the crowd density of this cell. We refer the reader to Figure 4.1 for an example of a density map.

The density map is updated in real-time based on the actual motions of the characters. In the beginning of the simulation, all cells have a zero density value. As soon as a character steps into a grid cell or moves within the cell, its density is updated and a new value ρ_N is stored by taking the current density value ρ_C into account, that is:

$$\rho_N = \rho_C + \rho_I, \tag{4.1}$$

where the parameter ρ_I controls how fast the density value of the cell increases.

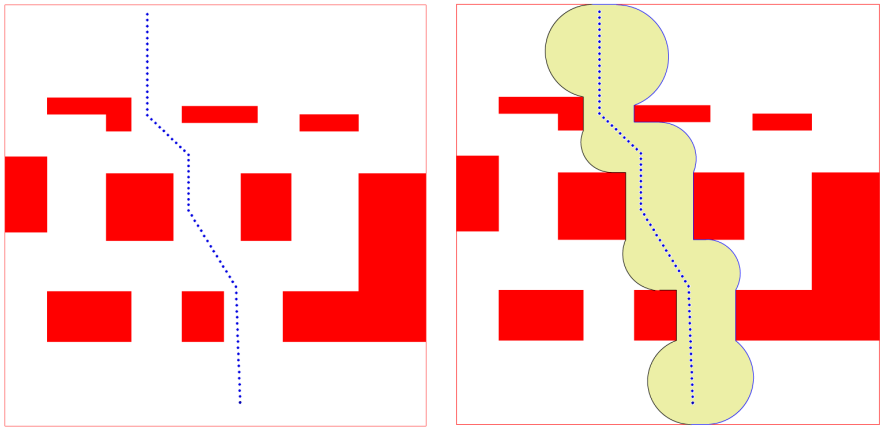
At any time step t of the simulation, the (current) density $\rho(t)$ of a grid cell can be computed as follows:

$$\rho(t) = \rho_O - \Delta\tau * \rho_D, \tag{4.2}$$

where ρ_O is the previously stored density value and $\Delta\tau$ is the amount of time, in seconds, passed since the last update. The parameter ρ_D controls the speed of the density falloff. This means that the density field degrades over time and eventually disappears, allowing for a character to take into account only cells that are recently traversed by other characters (note that the density $\rho(t)$ is clamped to non negative values). The method is very efficient, as computations are only performed when a character visits a cell.

4.3.2 Computing a Global Route

Given the start and the goal position of the character, its route can now be computed by applying an A* based search on the free grid cells. In general, the A* algorithm repeatedly examines the most promising locations on the grid to find the lowest cost path. Therefore,



(a) Indicative route (b) Explicit corridor

Figure 4.2: An indicative route and its corresponding corridor.

a cost function $f(N)$ is associated with each cell N that characterizes the cost of the path up to this cell. The cost function is defined as

$$f(N) = g(N) + h(N), \tag{4.3}$$

where $g(N)$ determines the cost from the start to the current cell, and $h(N)$ is the heuristic function that estimates the minimum path cost between the current location and the goal.

Typically, an A* search algorithm minimizes the distance that the character has to travel and returns the shortest path between the start and the goal position. However, since we strive for paths that avoid crowded areas, the density should also be taken into account upon planning. Therefore, the density map is used to modify the g-cost of a cell as follows:

$$g(N) = g(N-1) + d(N, N-1) + \lambda \cdot \rho_N, \tag{4.4}$$

where $d(N, N-1)$ is the Euclidian distance between the current cell and its predecessor, ρ_N is the density value of the current cell provided in Equation 4.2 and λ is a weighting parameter controlling the balance between density and path length.

Since we cannot predict the density in the remaining part of the path, the heuristic function only estimates the remaining distance to the goal, that is

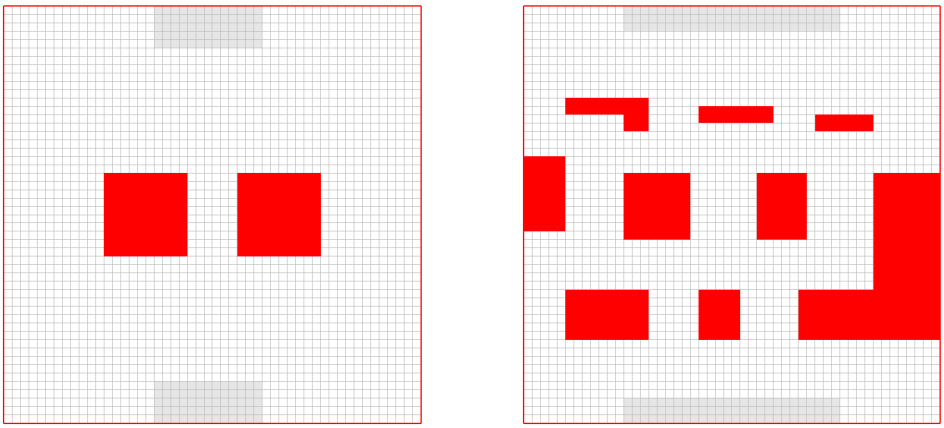
$$h(N) = d(N, goal) \tag{4.5}$$

As this is a lower bound to the actual cost and our cost function is always positive, the heuristic is always admissible. Hence, the method is guaranteed to find the optimal path.

4.3.3 Local Navigation and Final Motion

Having retrieved the global route of the character, we can now plan its final motion using the Indicative Route Method (IRM). One could argue that the character can directly follow the path returned by our modified A* algorithm. However, the resulting motion may not be collision-free. In addition, A* based paths are, in general, not natural. Natural paths typically follow smooth curves, keep a preferred amount of clearance from obstacles and are flexible enough to avoid other characters and moving entities.

The IRM satisfies all these criteria. It can generate high-quality and believable paths. In addition, it is fast and flexible, allowing the real-time simulation of crowds of characters as discussed in Chapter 3.



(a) Simple environment (b) Complex environment

Figure 4.3: The two test environments. Gray areas indicate the start and goal positions of the characters.

We now briefly explain how our proposed approach can be integrated into the IRM. We begin by querying the grid map in order to retrieve a path, as explained in the previous section. This path is expressed as a sequence of grid nodes and defines the indicative route of the character. An example of such an indicative route is shown in Figure 4.2(a). The character uses its indicative route as a guide to safely navigate toward its desired location. For that reason, a corridor is constructed around this route, indicating the area in which the character can move without colliding with the environment (see Figure 4.2(b)).

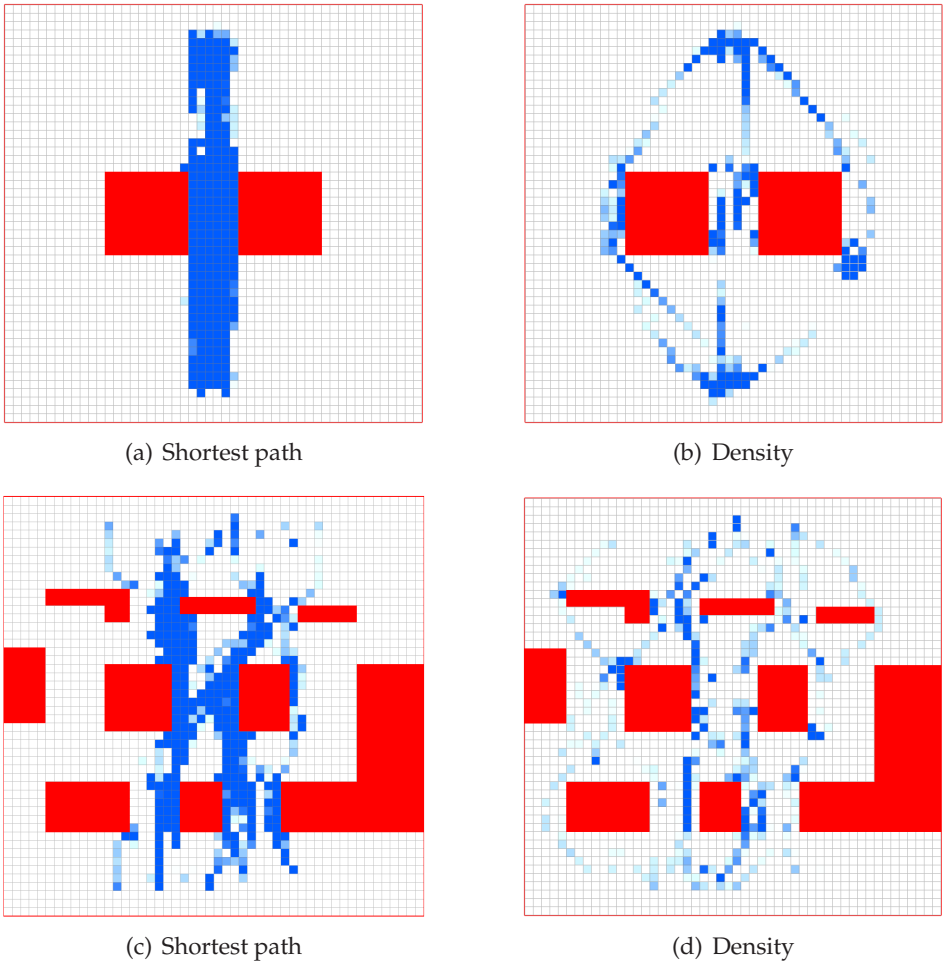
Once the corridor is created, several forces are acted upon the character and drive its motion through the corridor (we refer the reader to Section 3.5 for more details). First, a steering force $\mathbf{F}_s$ advances the character toward its goal. To this end, an attraction point moves along the indicative route and attracts the character. Hence, the character is steered to less dense areas. Second, a boundary force $\mathbf{F}_b$ is applied to ensure that the character remains inside its corridor. To avoid collisions with other characters and moving entities, an additional *collision avoidance force* is also required. Therefore, in our simulations, we have used part of the social force model proposed by Helbing and Molnár in [77], which is known to exhibit important emergent phenomena observed in real crowds.

Given the forces applied on the character, time integration is used to update the character's velocity and position, resulting in a smooth path that stays inside the corridor, moves toward the goal, and avoids other characters.

4.4 Experiments and Discussion

We have implemented our proposed approach to experimentally test its effectiveness and applicability in interactive real-time applications. All the experiments were performed on a PC running Windows XP, with a 2.4 GHz Intel Core2 Duo CPU and 2 GB memory.

The experiments were conducted for the two environments depicted in Figure 4.3. The first environment is very simple. Characters would typically move through the corridor in the center, which will eventually become cluttered. We expect that our new approach will steer the characters around the obstacles. In the second environment, there are many more possible routes that a character can follow to reach its destination. We



(a) Shortest path (b) Density

(c) Shortest path (d) Density

Figure 4.4: Example of density maps obtained by combining an A* shortest-path search algorithm or our density-based A* algorithm with the Indicative Route Method.

expect that our method will allow for variations in the characters' routes, as congestion evolves across the scene.

For both environments a 50 x 50 grid map was used, with each cell covering an area of 2 x 2 meters. In our simulations, characters were represented as discs having radius of $0.6\,\mathrm{m}$ and a preferred speed of $1.4\,\mathrm{m/s}$ (corresponding to the average walking speed of humans [107]). To obtain visually pleasing simulations, the density increase parameter in Equation 4.1 was experimentally set to $\rho_I = 0.2$, the time decay factor in Equation 4.2 to $\rho_D = 0.1$ and the weighting parameter of the A* cost function to $\lambda = 3$ (cf. Equation 4.4).

To determine the quality of the generated paths, we selected a varying number of characters and placed them randomly at the top and the bottom of the two environments. Each character had to advance toward a random goal position on the other side of the environment (see Figure 4.3). New characters were spawn every 3 seconds. Simulations were run for a maximum number of 30,000 iterations and every iteration calculated 0.1 seconds of characters' movements.

Figure 4.4(a) shows an example of a density map obtained using an A* shortest path algorithm (note that the IRM was used to plan the final motion of the character). As can be observed, all characters take the shortest path and move through the narrow corridor between the two obstacles. As the environment becomes more crowded, a congestion

	Time		Avg. Speed		Total Accel		Avg. Accel	
	Mean	Sdev	Mean	Sdev	Mean	Sdev	Mean	Sdev
Shortest + IRM	71.554	0.598	1.259	0.006	766.981	12.274	1.323	0.015
Density + IRM	74.103	0.272	1.349	0.003	570.219	7.749	0.963	0.008

(a) Simple Environment

	Time		Avg. Speed		Total Accel		Avg. Accel	
	Mean	Sdev	Mean	Sdev	Mean	Sdev	Mean	Sdev
Shortest + IRM	81.439	2.088	1.206	0.020	910.432	50.341	1.358	0.037
Density + IRM	78.549	0.516	1.307	0.002	694.650	12.026	1.096	0.017

(b) Complex Environment

Table 4.1: Time (s), speed (m/s) and acceleration statistics (m/s^2) using the shortest path as the indicative route of the character and a path that takes into account both density and path length. For each approach, the reported numbers are the averages over five simulations, each involving 325 characters.

occurs in the center resulting in inefficient movements. Figure 4.4(b) shows an example of a density map obtained using our approach. Clearly, the characters anticipate that a congestion will occur and try to avoid it in advance by moving around the obstacles instead of moving between them, which leads to an effective character flow. Examples of density maps for the other test environment are also depicted in Figure 4.4. Again, it can be observed that the characters spread over the environment and thus, the scene looks more lively.

To quantitatively compare the two approaches a number of metrics have been devised. In particular, we computed the total time that a character needs to reach its goal, as well as the average speed with which the character moves. We are also interested in how energy-efficient are the movements of a character. For that reason, we used the total acceleration as a measure of the movement effort spent by a character [197]. The average acceleration was also reported as a measure of time-independent effort.

Table 4.1 summarizes the results we obtained for the two environments shown in Figure 4.3. Note that, for each approach, five simulation runs were performed. As indicated above, each simulation consists of $30,000$ iterations. However, to ensure that the density values have converged, we first let each simulation run for $10,000$ iterations and then start gathering the detailed statistics of the characters. A two (approaches) x two (environments) x five (simulations) ANOVA test was performed afterwards for each of the proposed evaluation metrics. In each test, the significance level was set to 5%.

According to our analysis, the traveling time of a character is not affected by the approach taken to generate a path, $F(1,4) = 0.1, p = .078$. This can be attributed to the fact that, in our approach, the characters plan early for congestion and select paths that move through less crowded locations. Consequently, they move with a higher speed than characters that follow shortest paths, $F(1,4) = 379.9, p < .001$. In addition, using our approach, the number of avoidance maneuvers that the characters have to perform is reduced. This leads to more energy-efficient paths, as the characters have to spend less

effort to avoid potential collisions. In particular, our density-based algorithm yields to a decrease in the total acceleration of a character, $F(1,4) = 263.95, p < .001$. The same also applies to the average acceleration of the characters, $F(1,4) = 1410.7, p < .001$. In contrast, by only taking the path length into account, the environment becomes easily crowded and hence, the characters have to spend more time and effort to resolve collisions with other characters. Therefore, although they follow more direct routes, their arrival times do not significantly differ from those of characters that also account for crowd density.

We should also note that the simple environment yields to lower traveling times, $F(1,4) = 291.4, p < .001$, and higher walking speeds $F(1,4) = 83.7, p < .001$ as compared to the complex one. In addition, characters in the simple environment exhibit lower total acceleration than characters that navigate through the complex environment, $F(1,4) = 125.4, p < .001$. The average acceleration is affected in a similar way by the type of the environment, $F(1,4) = 40.0, p < .01$. These observations are in accordance with the behaviour of pedestrian crowds in real-life. In a simple environment, individuals can efficiently plan their paths and safely move toward their destinations. They are confident and prefer to walk at higher speeds. Furthermore, they are able to quickly solve interactions with other walkers.

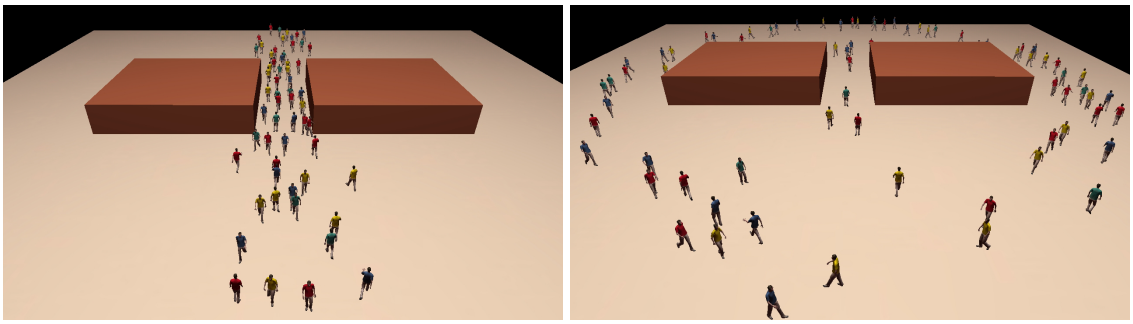
Besides the quantitative analysis, we have also empirically compared the two approaches by observing the resulting simulations. Figure 4.5, for example, shows a simulation frame from the simple environment. In this environment, each character has only three homotopic path choices. It can either move through the narrow corridor or select a path that is at the right or left of the corridor. As can be inferred from the figure, our approach provides a good division of the characters over these routes. Since we used part of the Helbing's social force model for the local interactions of the characters, we observed in both methods the phenomenon of lane formation that is widely noted in pedestrian literature [74]. However, using our approach, the flow of the characters looks smooth and visually pleasing and the paths of the characters are considerably less curved compared to the paths obtained using the IRM combined with an A* shortest path algorithm.

In conclusion, our approach, indeed, leads to a better spread of characters over the environment, less character interactions and more energy-efficient paths, resulting in a more pleasing simulation.

4.5 Conclusions

In this chapter, we presented a simple approach to automatically compute indicative routes for crowd simulation purposes. In our model, characters adapt their planning behaviour based on the crowd density. For that reason, the notion of density map was introduced as a data structure to store density information. Given a density map, an indicative route for the character is retrieved by applying an A* search algorithm that takes into account both density and path length. The final motion of the character is then obtained using the Indicative Route Method. This leads to high-quality and collision-free paths, ensuring at the same time visually convincing motions.

We have experimentally shown that, using our approach, characters can intelligently plan around crowded areas, preferring to move through low-density regions. Although the resulting paths may be longer, the amount of local interactions between characters is reduced, which leads to time and energy efficient movements. During our experiments, an initial phase was used to ensure that the density map has converged to a stable state. In real-time applications, like games, the initial state of the map can be computed in a



(a) Shortest path approach (b) Density-based approach

Figure 4.5: A simulation frame from the simple environment. (a) The shortest path algorithm leads to a congestion in the center. (b) The density-based algorithm spreads the characters over the environment leading to a visually compelling simulation.

preprocessing phase or provided manually by the level designer to encourage certain character behaviour. While the player-controlled character moves through the virtual world and interacts with other characters, the density map is dynamically updated as discussed in Section 4.3.1.

Despite its simplicity, we believe that our proposed approach has a lot of potential and can form the basis for further research. In our current implementation, the characters do not dynamically adapt their trajectories; each virtual character decides about its indicative route only once, which may occasionally lead to unrealistic behaviours, e.g. characters waiting for a narrow passage to be freed whereas the nearby routes are less crowded. To alleviate this problem and increase the realism of the simulation, the characters should automatically replan their indicative routes, using, for example, methods from [109, 134, 204].

Our framework could also be augmented by incorporating the notion of influence regions into our path planner. Such regions can either be dangerous places the characters prefer to avoid or appealing locations they would like to visit. We would also like to simulate characters that prefer to move with the flow, by taking into account the direction in which previous characters have traveled, as proposed in [94, 136]. In all these cases, a similar approach can be applied that places a multi-layer coarse grid over the environment, with each layer storing and maintaining information about different factors that influence a character's path, such as density, danger and/or flow. Alternatively, we can store these influence values at the vertices of a semantically augmented graph [247], or in the cells of a navigation mesh structure, such as the Explicit Corridor Map [55]. Next, an indicative route is determined using an A* search, and, finally, the Indicative Route Method is applied. To preserve fast search times, hierarchical path planning techniques, such as the ones proposed by Paris *et al.* [162] and by Shao and Terzopoulos [193] can be used for the retrieval of the indicative routes.

Chapter 5

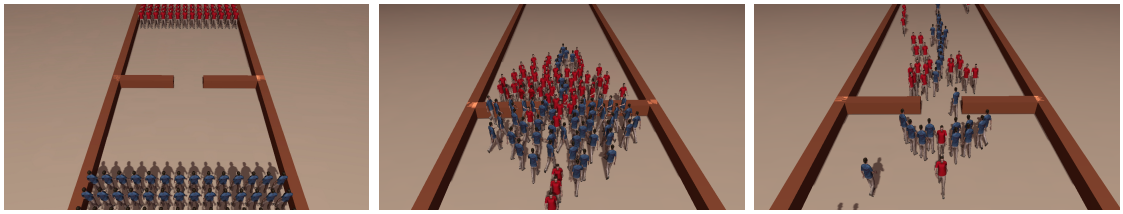
Space-time Group Motion Planning

In this chapter, we address the problem of planning the paths and directing the motions of agents in an environment containing static obstacles. The agents are organized into groups having similar characteristics and intentions, such as an army of soldiers in a real-time strategy game or virtual pedestrians that cross paths at an intersection. Each group has its own start and goal position (or area), and each group member traverses its own path. Our formulation is designed for large groups and the main objective is to steer the group agents toward their destinations as quickly as possible and without collisions with obstacles or other agents in the environment. This task would have been simple, if the paths of the agents were independent of each other. However, due to space restrictions, congestions may appear and waiting or finding alternative paths may be more efficient for some agents. Furthermore, the time dimension introduces additional complexity into the planning process; congestions only appear at certain moments in time, while at other moments the same area may be completely empty allowing the agents to navigate through it without delay.

Existing solutions to the aforementioned problem rely mainly on multi-agent systems. In these solutions, each agent plans separately its own path, whereas the local interactions between the agents are resolved using behavioural rules. Although recent advancements in local methods have significantly improved the collision avoidance behaviour of the agents, it is often difficult to design rules that consistently generate desired crowd behaviour. This problem becomes more obvious in crowded and complex interaction scenarios, as obtaining a satisfactory simulation result usually requires laborious and careful parameter tuning. Moreover, since global path planning and local collision avoidance are typically decoupled, conflicts may arise that can lead to deadlocks. Consider, for example, a large number of agents trying to pass through a narrow bottleneck. The local planner "perceives" the paths provided by the global planner as blocked and consequently, this failure of communication between the planners results in unnatural behaviours (see e.g. Figure 5.1(b)).

To alleviate the aforementioned problems, we present a novel algorithm for planning and directing group motions that is based on *linear programming*. Our solution translates the group planning problem into a *network flow* problem on a graph that represents the free space of the environment. It plans in both space and time and uses the *column generation* technique [48] from the operations research theory to efficiently identify the most promising paths in the graph. The planner computes an optimal distribution of the agents over these paths such that their average traversal time is minimized. The computed space-time plan is then given as an input to an underlying agent-based method in order to handle collisions between the agents and generate their final motions.

As compared to previous solutions, our method offers the following characteristics:



(a) (b) (c)

Figure 5.1: (a) Two opposing groups need to pass through a narrow bottleneck. (b) Agent-based methods lead to congestion and deadlocks. (c) Our technique can successfully handle such challenging scenarios by planning in both space and time.

- Space-time planning of multiple groups of heterogeneous agents based on a linear programming approach.
- Optimal distribution of the agents over alternative paths to resolve congestions.
- Coordination of the movements of the agents to prevent deadlocks that typically occur near bottlenecks (e.g. narrow passages) in the environment.
- Explicit waiting behaviour that leads to convincing simulations of high-density crowds with no observed oscillatory behaviour.
- Seamless integration of global path planning with existing schemes for local collision avoidance.
- Real-time performance in complex and challenging scenarios.

Our overall solution is applicable to a variety of virtual environment applications, such as computer games and crowd simulators. We demonstrate its usefulness against a wide range of scenarios, as can be seen in Figures 5.1, 5.6, 5.8 and 5.7.

5.1 Related Work

The most common approach to simulate groups of virtual characters is the flocking technique introduced by Reynolds [181]. Although flocking leads to natural group behaviour, the group members act based only on local information and thus, they can get stuck in cluttered environments. To remedy this, Bayazit *et al.* [9] combined flocking techniques with probabilistic roadmaps, whereas Kamphuis and Overmars [99] used the concept of path planning inside corridors so that the group members can stay coherent.

In general, two-level planning approaches have been widely used in the graphics and animation community for multi-agent navigation and crowd simulation. In these approaches, a high-quality roadmap or graph governs the global motion of each agent [57, 118, 170, 230], whereas a local planner allows the agent to avoid collisions with other agents and obstacles. Over the past few years, numerous models have been proposed for local collision avoidance, including force-based approaches [76, 77, 165], behavioural and cognitive models [156, 193] and variants of velocity-based methods [173, 226, 230]. These approaches are known to exhibit emergent behaviours. Nevertheless, since they are separated from the global planner, they are susceptible to local-minima problems. In highly dynamic and dense environments, this normally leads to deadlock situations (see Figure 5.1(b))that can only be resolved by rather unnatural motions. In such environments, even

replanning [109, 204] is often insufficient to generate a feasible trajectory. Our algorithm can effectively handle such issues, since the motion of the agents is taken into account during global planning. Consequently, during a simulation, no conflict exists between global planning and local collision avoidance and hence, the agents exhibit smooth motions. Note that, similar to our technique, the recent work of Singh *et al.* [199] resolves deadlocks by planning on a space-time graph. However, their formulation provides a localized space-time plan for each agent and cannot give any guarantees on the overall behavior of the agents, whereas our approach coordinates the movements of the agents at a global scale.

Multi-agent path planning has also been extensively studied in robotics, mainly for cooperative tasks between multiple robots. Centralized planners, such as in [133, 188], compute the motions of all agents simultaneously, whereas decoupled techniques, e.g. [167, 196], plan a path for each agent independently and try to coordinate the resulting motions. However, both solutions are computationally demanding, which makes them unsuitable for real-time interactive applications like games. Prioritized planning approaches have also been proposed in the field of robotics and animation (e.g. [120, 209, 228]). In these approaches, a priority is assigned to each agent according to a certain priority scheme. Paths are then planned sequentially for each agent by considering agents whose motions have already being planned as dynamic obstacles with known trajectories. This reduces the multi-agent path planning problem to the problem of computing a collision-free trajectory for a single agent in a known dynamic environment, which can be addressed by space-time planners [50, 229]. Although prioritized techniques guarantee inter-agent collision avoidance, the growing complexity of the planning space limits the number of characters that they can simulate. In addition, it is unclear how realistic such approaches are.

Prior work in graphics community has also focused on the synthesis of realistic group motions [114, 210] mainly for offline simulations of crowds. Recently, a number of algorithms have been proposed to simulate the local behaviour of small pedestrian groups [148, 169], but they do not address the issue of path planning. Example-based approaches have also been used to construct group behavioural models from motion capture data or from videos of real crowds [97, 126, 128]. The obtained performances, though, are too low for applications such as interactive virtual worlds. In addition, these approaches cannot handle challenging path planning problems.

An alternative approach has been proposed by Treuille *et al.* that attempts to directly guide the global behaviour of large homogeneous groups using continuous density fields [218]. This approach unifies global planning and local collision avoidance into a single framework and can become expensive when simulating a large number of groups. In contrast, our approach plans the global motions of the groups through space-time and uses a per-agent collision avoidance scheme to direct the local motions of the group members. Furthermore, our method can effectively handle challenging narrow bottlenecks and groups with competing goals, as compared to continuum crowd approaches.

More recently, Patil *et al.* [164] proposed a novel approach to steer and interactively control groups of agents using goal-directed navigation fields. Although their approach can effectively handle challenging scenarios, it requires user input to guide the agent flows and successfully resolve congestion, as compared to our model that automatically accounts for time-consuming situations. Guy *et al.* [65] resolve congestion by guiding the agents along a precomputed roadmap and dynamically assigning with each roadmap edge a weight, indicating the amount of biomechanical effort that an agent needs to expend to traverse this edge. However, due to lack of coordination among the agents, such

a method cannot predict and resolve deadlocks, like the one shown in Figure 5.1. Furthermore, as global path planning is decoupled from local collision avoidance, the simulated agents can get trapped in local minima.

Finally, a different solution to the group path finding problem has emerged from the operations research community. Van den Akker *et al.* [225] formulated this problem as a *dynamic multi-commodity flow* problem. Their approach takes as input a graph resembling the free space of the environment and several *homogeneous* groups of units, each having its own start and goal position. A *capacity* is associated with each arc on the graph and represents the maximum number of units that can traverse this arc per unit time. The objective is to find paths that minimize the average arrival times of all units. Since this problem is known to be NP-hard, they proposed a new heuristic that is based on techniques from (integer) linear programming.

Our model is inspired by the work of van den Akker *et al.*, since we use a similar linear program to coordinate the global motions of the groups through space-time. However, we extend their formulation to allow the simulation of *heterogeneous* groups. We also take into account capacity constraints on the *nodes* of the graph, that is on the maximum number of agents that can be on each node per unit time. This provides a more accurate solution to the problem and inhibits the oscillations we observed in [225] when multiple agents have to wait at a certain node. Moreover, van den Akker's model use a directed graph to capture the free workspace. This significantly restricts the movements of the agents; in real-life, for example, pedestrians can walk in either direction through a pavement or simultaneously enter/exit a building through the same door. To resolve this, our method uses *undirected* edges and thus, additional constraints are considered to account for agents sharing the same edge while walking in opposite directions. Finally, van den Akker *et al.* only present a theoretical framework for global planning and do not discuss simulation. We address this by using the Indicative Route Method to guide the agents toward their goals. To that end, a simple, yet effective, algorithm is also introduced to determine the indicative routes of the agents.

5.2 Preliminaries

In this section, we provide a quick overview of concepts from linear programming that will be used throughout this chapter. We refer the reader to the books of Bertsimas and Tsitsiklis [11] and Wolsey [243] for a more detailed explanation.

5.2.1 Duality Theory

Every linear programming problem, referred to as *primal* problem, can be converted into a *dual* problem that provides an upper bound to the optimal value of the primal problem. Consider the following primal problem expressed in its *symmetric* form:

$$\begin{aligned} \min \quad & \mathbf{c}^T\mathbf{x} \\ \text{s.t.} \quad & A\mathbf{x} \geq \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0}, \end{aligned}$$

where the vector $\mathbf{x} \in \mathbb{R}^n$ contains the decision variables that need to be determined, the vector $\mathbf{c} \in \mathbb{R}^n$ contains the (known) coefficients of the objective function, and $\mathbf{b} \in \mathbb{R}^m$, $A \in \mathbb{R}^{m \times n}$ are vector and matrix respectively of (known) coefficients related to the constraints of the problem. Then, its corresponding *symmetric* dual problem is given by:

$$\begin{aligned} \max \quad & \mathbf{b}^T\mathbf{y} \\ \text{s.t.} \quad & A^T\mathbf{y} \leq \mathbf{c} \\ & \mathbf{y} \geq \mathbf{0}, \end{aligned}$$

where $\mathbf{y} \in \mathbb{R}^m$. Note that each constraint in the primal has an associated variable in the dual and to each variable in the primal corresponds a constraint in the dual. Moreover, it can be easily shown that the dual of the dual is the original primal problem.

We say that a solution $\tilde{\mathbf{x}}$ is a *feasible* solution for the primal problem, if it satisfies the constraints of the primal, that is, if $A\tilde{\mathbf{x}} \geq \mathbf{b}$ and $\tilde{\mathbf{x}} \geq \mathbf{0}$. Similarly, $\tilde{\mathbf{y}}$ is a feasible solution for the dual problem, if $A\tilde{\mathbf{y}} \leq \mathbf{c}$ and $\tilde{\mathbf{y}} \geq \mathbf{0}$. The *weak duality* theorem states that the objective value of the primal at any feasible solution is always greater than or equal to the objective value of the dual at any feasible solution. The *strong duality* theorem states that if the primal has an optimal solution, $\mathbf{x}^*$, then the dual has also an optimal solution, $\mathbf{y}^*$, such that $\mathbf{c}^T\mathbf{x}^* = \mathbf{b}^T\mathbf{y}^*$. From the two theorems it follows that if the primal is unbounded, then the dual has no feasible solution and conversely. However, it is possible for both the dual and the primal to be infeasible.

5.2.2 Reduced Cost

Given a linear program in matrix notation, $\{min\ \mathbf{c}^T\mathbf{x} : A\mathbf{x} \geq \mathbf{b}, \mathbf{x} \geq \mathbf{0}, \}$, its *reduced cost* can be computed as $\mathbf{c} - A^T\mathbf{y}$, where $\mathbf{y}$ denotes the variables of the dual of the linear program. Such a cost denotes the amount by which a coefficient of the objective function needs to improve (decrease for our given minimization example), before it would be possible for its corresponding decision variable to obtain a positive value in the optimal solution.

5.3 Definitions and Background

In this section, we provide an overview of multicommodity flow problems and describe our overall solution for planning and simulating groups of virtual agents. We refer the reader to the classic book of Ford and Fulkerson [49] for a detailed discussion of commodity flow problems.

5.3.1 Multicommodity flow problems

Let $G = (N, A)$ be a directed graph in which every arc $a \in A$ has a non-negative real-valued capacity $c_a > 0$ associated with it. Such a graph is also called a *capacitated network*. Let also C denote a set of *commodities*. Every commodity $i \in C$ has an associated source node $s_i \in N$ and a target node $t_i \in N$. A feasible flow is a mapping $f : C \times A \rightarrow \mathbb{R}_+$ satisfying the following two properties:

- Capacity constraints: $\sum_{i \in C} f_{i,a} \leq c_a, \forall a \in A$. This property ensures that the sum of the flows on each arc cannot exceed its capacity.
- Conservation of flow: $\sum_{a \in \delta^+(n)} f_{i,a} - \sum_{a \in \delta^-(n)} f_{i,a} = 0, \forall i \in C$ and $n \in N \setminus \{s_i, t_i\}$. Here $\delta^+(n)$ and $\delta^-(n)$ are the outgoing and the incoming arcs of $n \in N$ respectively. This property guarantees that the net flow to a node is zero, except for the source and target nodes.

The *maximum multicommodity flow* problem aims to maximize the sum of the flow values of the individual commodities; the flow value of a commodity is the net outflow from the source node, or equivalently, the net inflow to the sink node. A variant of this problem is the *minimum cost multicommodity flow* problem. Here, *demands* are added for each commodity and *costs* for every arc of the graph. The goal is to find a minimum cost flow such that for each commodity the net outflow from the source node is at least the demand.

Dynamic flow problems are an extension of static flow problems with *traversal* times added on the arcs. The traversal time is the amount of time that the flow requires to travel from one end of the arc to the other end. Traditionally, dynamic flow problems are solved in *time-expanded* networks [47, 49] that discretize time and create a copy of the original capacitated graph for each discrete time step. This immediately makes all techniques for static flows available to dynamic flows. Unfortunately, time-expanded graphs tend to be very large. Fleischer and Skutella [46] have proposed a *condensed* time-expanded network to solve this issue. Their approach relies on a rougher discretization of time.

While there is substantial work on static multicommodity flow problems as well as on dynamic flow problems for a single commodity, there is hardly any literature regarding the dynamic multicommodity flow problem. However, this problem is known to be NP-hard in the strong sense [68, 104]. Recently, Akker *et al.* [225] devised a new heuristic for the problem that is based on the column generation technique from linear programming. In their approach, groups of units are considered as commodities, while a flow along an arc of the graph represents the amount of units that travel along this arc. Their formulation solves a minimum cost multicommodity flow problem on a time-expanded network, where the objective is to minimize the average travel time of the units. We also refer the reader to [176] for additional details regarding this work.

5.3.2 Problem formulation and overall solution

In our problem setting, we are given k groups of agents that share a 2D environment containing static polygonal obstacles. Each group $C_i, i \in [1, k]$ has its own start s_i and goal position t_i and consists of d_i agents that need to navigate from a specified start to a specified goal area defined around the group's origin and destination point respectively. Furthermore, each group C_i is assigned a desired speed U_i^{des} indicating the speed at which its members prefer to move. Given a group C_i, we denote an agent belonging to C_i as A_{ij}, where $j \in [1, d_i]$. For simplicity, we assume that each group member A_{ij} is modeled as a translating disc in the plane having radius r_{ij} and is subject to a maximum speed $v_{ij}^{\max}$. We further assume that each agent keeps a certain psychophysical distance $\rho_{ij} \geq r_{ij}$ from other agents and static obstacles in order to feel comfortable. This distance defines the *personal space* of A_{ij}, which, for efficiency reasons, we model as a disc centered at the agent. The task is then to steer the group members A_{ij} toward their corresponding goal areas as quickly as possible and without collisions with the environment and with each other.

We propose a two-level framework to solve the aforementioned planning problem. In the first phase, we formulate the group planning problem as a *dynamic multi-commodity flow* problem and employ an Integer Linear Programming (ILP) approach to solve it. The ILP plans in both time and space by taking into account capacity constraints on the edges and the nodes of a graph that is based on the *medial axis* of the environment. Also, restrictions ensuring that all agents will reach their goals are included in the ILP. The variables in the ILP represent the number of agents using a certain path. Such a path is described by the graph nodes that the agent should visit along with the times at which these nodes

should be reached. Furthermore, waiting behaviour can be encoded in the path. Since the number of possible space-time paths (and thus variables) is in general infinite, we use the *column generation* technique [48] to efficiently identify the most promising paths for the agents. In this way, we make the problem tractable, without loosing too much on quality. The search for useful paths in the time-expanded graph is performed efficiently using an A*-based algorithm.

Our ILP formulation is similar to the aforementioned work of van der Akker *et al.* with two significant improvements. Notably, the graph is undirected, which means that an edge can be traversed in either or both directions at the same time, making much harder to model the capacity constraints. In addition, the agents are also subject to capacity constraints on the nodes of the graph.[1] In general, these two additions allow our system to compute a more convincing flow of agents as compared to van der Akker's work.

In the second phase of our framework, the paths computed by the ILP are given as input to an underlying agent-based simulation algorithm in order to generate the final motions of the agents. The idea here is that if multiple agents share the same edge or node at the same time, then there is enough clearance for these agents to walk or stand next to each other. Given the capacity information of the graph, we first create a number of *lanes* across the medial axis edges of the environment. We then compute for each agent an *indicative route* along these lanes that respects the space-time plan provided by the ILP. At every step of the simulation, the agent advances along its indicative route using the Indicative Route Method. A local collision avoidance method is also integrated to guarantee collision-free navigation for the agents.

5.4 Path Planning for Groups

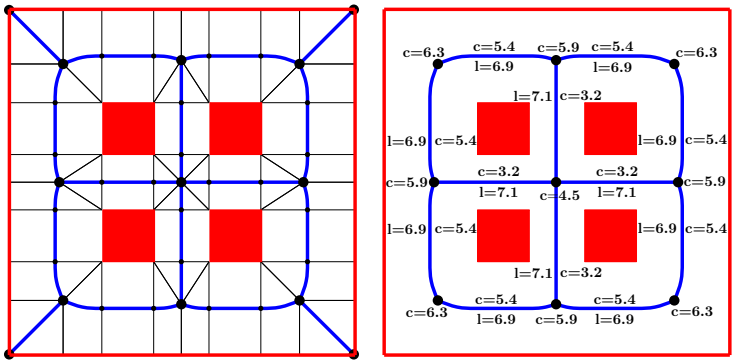
This section outlines the first phase of our framework that formulates the multi-group path planning problem as a dynamic multi-commodity flow problem.

5.4.1 Creating a Capacitated Graph

To solve the problem, we first need a roadmap that resembles the free space of the environment. In order to compute the roadmap, we can use any of the well known methods mentioned in the motion planning literature [121]. Alternatively, such a graph can be manually provided by a level designer using, for example, waypoints. In our approach, the roadmap is generated based on the edges and vertices of the medial axis (MA). This provides maximum clearance from the obstacles and includes all cycles present in the environment.

In particular, we precompute the Explicit Corridor Map (ECM) of the environment, as explained in Section 3.4.1. We first annotate each MA edge with *event points* (nodes of degree 2) that determine the minimum number of curves required to describe the edge. We then add a linear number of line segments connecting the event points and MA vertices to their corresponding closest obstacle points. The resulting ECM structure partitions the environment into a set of non-overlapping regions. Its main advantage over alternative approaches (see, e.g., [170]) is that it fully covers the free space of the environment at linear storage cost. It also provides the clearance to the obstacles for any

[1] In a directed graph, this problem can be reduced to the standard arc capacity problem by a simple devise as proposed by Ford and Fulkerson [49]. In particular, each node is first split into two nodes that are connected with an arc. The capacity of the original node is then imposed as an arc capacity on the new arc. However, this solution is not accurate when the capacitated graph is undirected, as in our problem setting.



(a) (b)

Figure 5.2: (a) Explicit Corridor Map. (b) Its corresponding capacitated graph.

free point and facilitates proximity computations to nearest obstacles. Its underlying MA graph ($MG = (V, E)$) is fine-grained in areas with many obstacles, while the number of event points is dramatically reduced in large open areas. Also, for each event point and MA vertex, its minimum clearance is stored along with its corresponding nearest obstacle points. Figure 5.2(a) shows an example of such an enhanced graph. Note that the graph is undirected.

Based on the ECM, we compute a *capacitated graph* $G = (N, E)$ as follows. We first copy all vertices and edges from MG to G. We then determine all nodes of degree 1 in the graph. Such nodes denote dead-ends in the environment and are removed from the set N in G. Their corresponding edges are also removed from the edge set E of G.

For each remaining edge $e \in E$, we define its traversal time l_e as the time that an agent requires to traverse this edge and compute it as the edge length $\mathcal{D}_e$ divided by the maximum speed $U^{\max}$ among all groups' desired speeds:

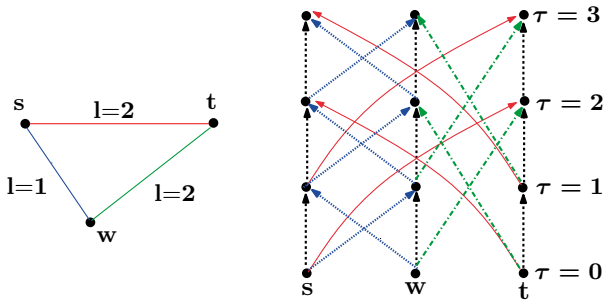
$$l_e = \frac{\mathcal{D}_e}{U^{\max}}, \text{ where } U^{\max} = \max_{i \in [1,k]} \{U_i^{\text{des}}\}. \tag{5.1}$$

We note that l_e is a lower bound and it may take longer for a group to traverse the edge. We will get back to this issue in Section 5.4.3.

A capacity c_e is also assigned to each edge denoting the maximum number of agents that can traverse the edge at the same time while walking next to each other. The edge capacity is computed as the minimum clearance $cl_{\min}$ along the edge divided by the maximum personal space $\rho^{\max}$ among all agents:

$$c_e = \frac{cl_{\min}}{\rho^{\max}}, \text{ where } \rho^{\max} = \max_{i \in [1,k],\, j \in [1,d_i]} \{\rho_{ij}\}. \tag{5.2}$$

Similarly, a capacity c_n is associated with each node n in the graph indicating the maximum number of agents that can simultaneously be on the node. Such capacity is computed by dividing the radius of the maximum clearance disc centered at n by the maximum personal space $\rho^{\max}$ defined above. We refer the reader to Figure 5.2(b) for the capacitated graph corresponding to the Voronoi graph depicted in Figure 5.2(a). Note that if the start s_i or goal t_i position of a group C_i is not included in the graph G, it is added to the set N as an extra node. The created node is also connected to G by determining a suitable point on a nearby edge of the graph. This point introduces a new node in G splitting up its corresponding edge. In case this point is already a node, this addition is not necessary.



(a) (b)

Figure 5.3: (a) A simple capacitated graph with integral traversal times. (b) Its corresponding time-expanded graph with time horizon $T = 4$ and time step $\Delta t = 1s$. Note that for clarity, we omit the capacity information from both graphs.

The generated graph captures different paths that the agents can follow in order to reach their destinations. Nevertheless, to properly avoid congestions, we must not only know which nodes are visited but also at which times these nodes should be reached. In addition, waiting at certain nodes may be more efficient for some of the agents. For that reason a condensed *time-expanded graph* is defined as proposed in [46] that adds the time dimension to the graph G. The basic idea here is to round up the traversal time l_e of each edge e in G to the nearest multiple of an appropriately chosen time step Δt, i.e. $l_e^* = \lceil l_e/\Delta t \rceil$. The capacity of the edge is also multiplied by Δt in order to define the maximum number of agents that can traverse the edge in one time unit, i.e. $c_e^* = \lfloor c_e \cdot \Delta t \rfloor$. In a similar manner, the discrete capacity of each node n is expressed as $c_n^* = \lfloor c_n \cdot \Delta t \rfloor$.

Let $G^T = (N^T, A^T)$ denote the time-expanded graph of G, where T defines the time horizon, that is the total number of discrete time steps. G^T is directed and is constructed from the static graph G based on its discrete capacities and traversal times. The set N^T contains a copy of each node n of the static graph G for each time step $\tau = 0...T-1$. Such a copy is denoted as $n(\tau)$ and has the same discrete capacity as its corresponding node in G. In addition, for every edge $e = \{n, m\}$ in G, two arcs will be created in G^T, one from $n(\tau)$ to $m(\tau + l_e^*)$ and one from $m(\tau)$ to $n(\tau + l_e^*)$. Finally, waiting arcs are also added to A^T from each node $n(\tau)$ to the same node one time-step later $n(\tau+1)$, for all $\tau < T$. Such an arc allows an agent to stay at node n for one time-step period, i.e. $l^* = 1$. Its capacity is defined as the number of agents that can simultaneously wait at the node and is the same as the discrete capacity c_n^* of the static node n.

An illustration of a time-expanded graph for our working example is given in Figure 5.3. Such a graph is not explicitly constructed. Instead, time-expanded nodes and arcs are created on-the-fly as discussed further in Section 5.4.3. In addition, an upper bound on the time horizon T is set by considering that the groups move in turns, that is, group C_{i+1} has to wait until all members of group C_i have arrived to their goal. Such approach leads to a feasible, but not necessarily optimal, solution to our multi-group path planning problem. In particular, let l_i^* denote the length (expressed in discrete time steps) of the shortest path in G between the origin and destination of group C_i consisting of d_i members. Let also c_i^* be the minimum capacity among all edges and nodes in G traversed by the the shortest path. Then, we divide the members of C_i into $h_i = \lceil d_i/c_i^* \rceil$ subgroups. To guarantee that the capacity constraints are met, we assume that the subgroups move sequentially and each subgroup leaves from its origin one time step later than its predecessor. Thus, the first subgroup of C_i arrives at its destination at time l_i^*, the second one needs $l_i^* + 1$ time steps, while it takes $l_i^* + h_i - 1$ steps for the members of the last subgroup

to reach their goal. To avoid any conflicts, as mentioned before, the next group needs to wait until the last member of C_i has arrived at its destination. Therefore, C_{i+1} can start moving no earlier than $l_i^* + h_i$, whereas the arrival times of its members can be computed following an approach similar to the one described above. This process is repeated until all members of the k groups have reached their goals and hence, the upper bound on the time horizon is given by $T = \sum_{i=1}^{k}(l_i^* + h_i) - 1$.

5.4.2 ILP Formulation

In our problem formulation, we are given the origin s_i, destination t_i and the size d_i of each group C_i. Let us assume that we know all valid paths in the time-expanded graph G^T corresponding to each origin-destination pair. A path is valid, if it starts at a node $s_i(0) \in N^T$ for some $i \in [1, k]$ and ends at node $t_i(\tau)$ for the same i. Let $\mathcal{P}$ denote the set of all these valid paths in G^T. For every path $p \in \mathcal{P}$ we introduce a decision variable f_p which denotes the number of agents using the path p. Then, we can model our path planning problem as an Integer Linear Programming (ILP) problem, as follows.

Our goal is to minimize the average arrival time of all agents, or, equivalently, the sum of the travel times of all agents. We use l_p to denote the cost (length) of path p, which equals to the arrival time of p at its destination. This leads to the objective function shown in (5.3). We also formulate demand constraints to guarantee that all agents will reach their targets as expressed in (5.4), where $\mathcal{P}_i$ is the set of paths in G^T starting at s_i and ending at t_i given a group C_i.

Restrictions are also needed to ensure that the arc capacity constraints are obeyed. In our problem setting, these constraints are much harder to model than in a normal multi-commodity flow problem, since the static capacitated graph G is undirected. Consequently, an edge can be traversed in either or both directions at the same point in time. If, for example, the traversal time is l and we want to send x agents through an edge at time t, then this is possible only if the number of agents that started to traverse the edge from the other side at times $t - l + 1, t - l + 2, ..., t + l - 1$ does not exceed the remaining capacity. To avoid having to add this enormous number of constraints, we introduce a non-negative decision variable u_e for every edge e in the static graph G. We call one direction on the edge forward and the other one backward. Let $F(e) \subset A^T$ be the set of forward arcs in the time-expanded graph G^T emerging from e, and $B(e) \subset A^T$ the corresponding set of backward arcs. Then, the capacity c_e of the edge is divided between the two sets. We do not fix the capacity distribution beforehand, but we make it time-independent by putting the capacity of the forward arcs equal to u_e and the capacity of the backward arcs equal to $c_e - u_e$. This adds (5.5) and (5.6) to the ILP, where $\mathcal{P}_a$ denotes the subset of paths using arc $a \in A^T$.

Finally, to account for the throughput limitation of each node v of the time-expanded graph, (5.7) is added to the ILP constraints, where $\mathcal{P}_\mathrm{v}$ denotes the subset of paths in the time-expanded graph using the expanded node v. The above considerations result in the following ILP problem:

$$\min \sum_{p \in \mathcal{P}} l_p f_p \tag{5.3}$$

$$s.t. \sum_{p \in \mathcal{P}_i} f_p \geq d_i \qquad \forall\, i = 1...k \tag{5.4}$$

$$\sum_{p \in \mathcal{P}_a} f_p - u_e \leq 0 \qquad \forall\, e \in E, a \in F(e) \tag{5.5}$$

$$\sum_{p \in \mathcal{P}_a} f_p + u_e \leq c^*_e \qquad \forall\, e \in E, a \in B(e) \tag{5.6}$$

$$\sum_{p \in \mathcal{P}_{\mathrm{v}}} f_p \leq c^*_{\mathrm{v}} \qquad \forall\, \mathrm{v} \in N^T \tag{5.7}$$

$$f_p \geq 0 \text{ and integral} \qquad \forall\, p \in \mathcal{P} \tag{5.8}$$

$$u_e \geq 0 \text{ and integral} \qquad \forall\, e \in E \tag{5.9}$$

Obviously, we do not know the entire set of paths $\mathcal{P}$ and enumerating it would be impractical, particularly for large time-expanded graphs. Therefore, we will make a selection of paths that we consider 'possibly useful', that is, paths that might improve the value of our objective function, and we will solve the ILP for this subset. We determine these paths by considering the LP-relaxation of the ILP, which is obtained by removing the integrality constraints from the decision variables f_p and u_e. We solve the LP-relaxation through the technique of column generation, which was first described by Ford and Fulkerson [48] for the static multi-commodity flow problem. We also refer the interested reader to [174,243] for a more detailed explanation of the column generation algorithm.

5.4.3 Column Generation and Pathfinding

The basic idea of column generation is to solve the LP-relaxation problem for a restricted set of variables (the set of valid paths in G^T, in our case) and then add variables that may improve the objective value until no additional variables can be found anymore. It is well known from the theory of column generation that, in case of a minimization problem, the addition of a variable will only improve the solution if its *reduced cost* is negative.

Recall, from Section 5.2.2, that the reduced costs are expressed in terms of the *dual variables*. In our case, the LP-relaxation yields a dual variable ψ_i for the demand constraint (5.4) corresponding to group C_i, a dual variable μ_a for the capacity constraints (5.5)-(5.6) corresponding to arc $a \in A^T$ and a dual variable ϕ_{v} for the capacity constraint (5.7) corresponding to node $\mathrm{v} \in N^T$. This leads to the following *dual* problem:

$$\max \sum_{i=1}^{k} d_i \psi_i - \sum_{e \in E} \sum_{a \in B(e)} c^*_e \mu_a - \sum_{\mathrm{v} \in N^T} c^*_{\mathrm{v}} \phi_{\mathrm{v}} \tag{5.10}$$

$$s.t. \sum_{i | p \in \mathcal{P}_i} \psi_i - \sum_{a \in \alpha(p)} \mu_a - \sum_{\mathrm{v} \in \lambda(p)} \phi_{\mathrm{v}} \leq l_p \qquad \forall\, p \in \mathcal{P} \tag{5.11}$$

$$\psi_i \geq 0 \qquad \forall i = 1...k \tag{5.12}$$

$$\mu_a \geq 0 \qquad \forall\, a \in A^T \tag{5.13}$$

$$\phi_{\mathrm{v}} \geq 0 \qquad \forall\, \mathrm{v} \in N^T \tag{5.14}$$

where $\alpha(p)$, $\lambda(p)$ denote the arcs and nodes, respectively, in G^T used by path p. From the dual constraints (5.11), it follows that the reduced cost of a path $p \in \mathcal{P}_i$ for a group C_i is equal to:

$$l_p + \sum_{a \in \alpha(p)} \mu_a + \sum_{\mathrm{v} \in \lambda(p)} \phi_{\mathrm{v}} - \psi_i. \tag{5.15}$$

Consequently, given the solution of the current LP, for each group C_i, we need to find a path $p \in \mathcal{P}_i$ such that

$$l_p + \sum_{a \in \alpha(p)} \mu_a + \sum_{\mathrm{v} \in \lambda(p)} \phi_{\mathrm{v}} - \psi_i < 0. \tag{5.16}$$

If such a path exists, we add it to the set of paths currently included in the LP and resolve the problem for the updated set. Otherwise, if all paths have non-negative reduced costs, an optimal solution for the entire problem has been found.

The inequality (5.16) can be actually rewritten to gain more insight into its meaning. The length l_p of a path p is equal to the amount of time that p requires to reach its target and thus, $l_p = \sum_{a \in \alpha(p)} l_a^*$, where l_a^* denotes the discrete length (traversal time) of arc $a \in A^T$ as defined in Section 5.4.1. Note, though, that the traversal time l_a^* is computed based on the maximum desired speed $U^{\max}$ among all groups (see Equation 5.1). However, since each group C_i has its own desired speed U_i^{des}, we define the correct length $l_a^{'*}$ of the arc a belonging to $p \in \mathcal{P}$ as: [2]

$$l_a^{'*} = \lceil l_a^* \cdot \frac{U^{\max}}{U_i^{\text{des}}} \rceil. \tag{5.17}$$

Therefore, the length of the path p can now be given by $\sum_{a \in \alpha(p)} l_a^{'*}$. Regarding the third coefficient of inequality (5.16), it is clear that if a node $\mathrm{v} \in N^T$ is contained in the path p, the path also contains precisely one arc that terminates at this node (with the exception of the origin node). Consequently, the effective contribution of the path's nodes to the reduced cost of p becomes $\sum_{a \in \alpha(p)} \phi_m$, given that $a = (n, m)$. Finally, reorganizing (5.16), we obtain:

$$\sum_{a \in \alpha(p)} (l_a^{'*} + \mu_a + \phi_m) < \psi_i. \tag{5.18}$$

Deciding whether (5.18) holds defines the *pricing problem* and can be efficiently solved for each group C_i as follows. We compute the shortest path for C_i in the time-expanded graph G^T, where each arc $a \in A^T$ has a modified length $l_a^{'*} + \mu_a + \phi_m$. We use an A*-like search to efficiently find such a path from the origin $s_i(0)$ of the group to its destination $t_i(\tau)$, $\tau \geq 0$. In the A* search, explored nodes of the time-expanded graph are created and added to an open set based on a function $f(\mathrm{v})$ that determines how promising each node $\mathrm{v} \in N^T$ is. The function $f(\mathrm{v})$ is defined as

$$f(\mathrm{v}) = g(\mathrm{v}) + h(\mathrm{v}), \tag{5.19}$$

where $g(\mathrm{v})$ is the cost of reaching the node from the origin $s_i(0)$ and is computed based on the modified arc lengths, whereas $h(\mathrm{v})$ is an admissible heuristic that gives a lower bound estimate on the traveling cost between the node v and the destination of C_i. As $h(\mathrm{v})$, we use the distance (expressed in discrete time units) between v and t_i in the static capacitated graph G. This distance is available, if prior to the query phase, a single-source Dijkstra's shortest path algorithm is run on G, with node t_i as its source. Such a heuristic is clearly a lower bound, since the length of a path in a time-expanded graph is equal to the length of its corresponding path in the static graph with possibly additional waiting times. Furthermore, in G^T, the length of the time-expanded arcs may be enlarged by the column generation algorithm based on the dual variables. Since our proposed heuristic is also consistent, our search algorithm retains the optimality of an A* search and can efficiently compute the shortest path for a group C_i. If the reduced cost of this path is negative, that is, (5.18) is satisfied, we can add the path to the LP and iterate. Otherwise, the optimum is found, since adding the path will not decrease the objective value of the LP.

[2] In case the arc a is a waiting arc, its discrete length is always equal to one time unit, i.e. $l_a^* = l_a^{'*} = 1$.

Theorem 5.1. *The search for a shortest path in the time-expanded graph is optimally efficient.*

Proof: The proof of the theorem follows directly from the optimality of the A* algorithm. We refer the reader to [185] for more details. Briefly, it is sufficient to show that whenever the A* selects a node v for expansion, the optimal path from the start to that node has already been found. We can prove by contradiction that this holds. If this was not true, there would have been another frontier node v′ on the optimal path from start to v. Since the f-values along any path are non-decreasing (due to the consistency of our heuristic), the node v′ would have lower f-cost than v, i.e. $f(\mathrm{v}) \geq f(\mathrm{v}')$. This means that v′ has already been expanded prior to v (contradiction). ■

Theorem 5.2. *For a given a group C_i, the shortest path in the time-expanded graph corresponds to the valid path $p \in \mathcal{P}_i$ with minimum reduced cost.*

Proof: This follows from the fact that each arc in the time-expanded graph has a modified length w.r.t. the dual multipliers μ_a, ϕ_{v}. Let p_o be the optimal path returned by the A* search, having length $l_o = \sum_{a \in \alpha(p_o)} (l_a^{'*} + \mu_a + \phi_m)$. By subtracting ψ_i from l_o, the reduced cost of the path is obtained. Since ψ_i is constant over all feasible paths belonging to $\mathcal{P}_i$ and p_o has the shortest length among all these paths, p_o is the path with the minimum reduced cost. ■

We refer the reader to Algorithm 5.1 for an overview of our column generation algorithm. Note that, initially, the LP has no columns (paths) and hence, we need to introduce an initial set of paths, one for each group C_i. We can start with any set, as long as it constitutes a feasible solution. An obvious choice for the initial columns would be to retrieve for each group its shortest path in the time expanded graph. In case the capacity constraints are not violated, this choice is optimal; otherwise the algorithm will fail, since the LP will not have a dual solution. We can easily circumvent the latter problem by letting the groups move in turn, that is group C_{i+1} has to wait until group C_i has reached its destination. Therefore, no conflicts between the groups will arise and a feasible solution can be added to the LP. Now, that we have initial columns (paths), we start iterating. We keep solving the LP for the restricted set of paths and the pricing problem for each group, up until no new path can be added.

Algorithm 5.1 Column Generation

```
1: Find initial paths and add them as columns to the LP
2: repeat
3:    Solve LP for the restricted set of paths
4:    Let μ, φ, ψ be the dual variables corresponding to the dual of the LP
5:    for each group C_i do
6:       Let l'*_a + μ_a + φ_m be the modified length of each arc a = (n, m) ∈ G^T
7:       Find a shortest (s_i(0) − t_i(τ))-path in G^T based on the modified arc lengths
8:       if length of the path is less than ψ_i then
9:          add the path as a new column to the LP
10:      end if
11:   end for
12: until no path has been added
```

When considering the running time of the column generation algorithm, we should take into account how much time does it cost to solve an LP, how much time is required to

solve the pricing problem for a group, and how many iterations the algorithm will take. Grötschel, Lovász and Schrijver [64] showed that, using the ellipsoid method, an LP can be solved in polynomial time with respect to the number of variables and restrictions. Regarding the pricing problem, we need to find for a given group its shortest path in the time-expanded graph. As explained above, such a path can be efficiently computed by applying an A* search algorithm, which is a polynomial time algorithm in the size of the graph.

Hence, the main question is how many iterations and, thus, paths (columns) do we need in the worst case in order to solve the LP-relaxation problem to optimality. Actually, there is a direct relation between the number of columns and the number of demand and capacity constraints in our initial ILP problem. According to (5.4)-(5.7) there are k demand constraints, $|A^T|$ arc capacity constraints and $|N^T|$ node capacity constraints.[3] Consequently, we can show that at most $Q = k + |A^T| + |N^T|$ paths are required to find an optimal solution for the LP-relaxation problem.

Lemma 5.1. *There exists a set of paths $\mathcal{P}' \subseteq \mathcal{P}$ of size Q such that our LP-relaxation problem can be solved to optimality.*

Proof: Every linear problem in its symmetric form can be converted into a *standard form* by transforming each inequality constraint into an equality with the addition of a so-called *slack* variable. The standard form of a linear problem (in matrix notation) is given by $\{\min\ \mathbf{c}^T\mathbf{x} : A\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}$, where $A \in \mathbb{R}^{m\times n}$, $\mathbf{c}, \mathbf{x} \in \mathbb{R}^n$, $\mathbf{b} \in \mathbb{R}^m$. Let B be a submatrix of A such that $B^{-1}\mathbf{b} \geq \mathbf{0}$, N denote a submatrix of A consisting of the columns that are not in B, $\mathbf{c}_B$ and $\mathbf{c}_N$ contain the components of $\mathbf{c}$ corresponding to the columns of B and N respectively and, similarly, $\mathbf{x}_B$ and $\mathbf{x}_N$ denote the decision variables derived from the partition of $\mathbf{x}$. Then, the standard form of the linear program can be rewritten as $\{\min\ {\mathbf{c}_B}^T\mathbf{x}_B + {\mathbf{c}_N}^T\mathbf{x}_N : B\mathbf{x}_B + N\mathbf{x}_N = \mathbf{b}, \mathbf{x}_B \geq \mathbf{0}, \mathbf{x}_N \geq \mathbf{0}\}$. The variables in $\mathbf{x}_B$ and $\mathbf{x}_N$ are known as *basic* and *non-basic* variables, respectively, whereas B is referred to as a *basis* for the linear program. The solution obtained by setting $\mathbf{x}_N = \mathbf{0}$ and $\mathbf{x}_B = B^{-1}\mathbf{b}$ is called a *basic solution*. If all basic variables are non-negative (i.e., $\mathbf{x}_B \geq \mathbf{0}$), such a solution is also feasible and we refer to it as a *basic feasible solution* (see [11] for more details).

Consider now a basic feasible optimal solution to our LP-relaxation problem. Since there are Q many constraints, the size of the basis is Q. Thus, at most Q basic variables (i.e., paths) have a nonzero entry, which means that it is sufficient for $\mathcal{P}'$ to consist of the corresponding paths. See also [174].

■

Based on the aforementioned observations, the following theorem holds:

Theorem 5.3. *Our LP-relaxation problem can be solved in polynomial time by employing the column generation technique.*

Proof: As explained above, if one takes care in the implementation and uses the ellipsoid method, the restricted LPs (line 3 of Algorithm 5.1) are solved in polynomial time. Furthermore, at every iteration of the column generation, the pricing problem for each group can be solved in polynomial time, whereas the algorithm only needs polynomially many iterations to find an optimal solution. Consequently, the column generation can run in polynomial time which leads to the statement of the theorem.

■

[3] The size of the time-expanded graph $G^T = (N^T, A^T)$ can be easily derived based on the size of the capacitated graph G and the maximum time horizon T defined in Section 5.4.1.

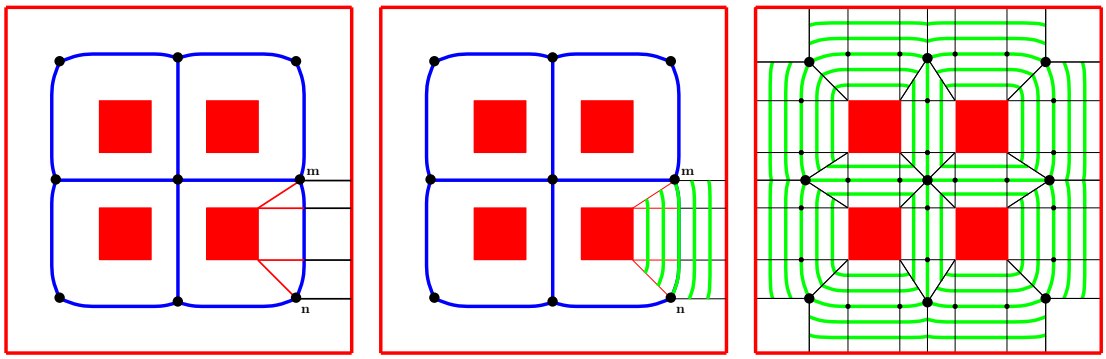


Figure 5.4: (a) A medial axis edge can be expressed as a sequence of portals. (b) Lanes are created along the edge by placing waypoints on the portals. (c) A depiction of all the lanes of the environment. Note that no lanes are formed along dead-end edges.

5.4.4 Obtaining an Integral Solution

At the end of the column generation algorithm, we have an optimal solution for the LP. Most likely, though, some of the variables f_p in our solution will have a fractional value and hence, we cannot follow this solution, since we cannot send fractions of agents along a path. We alleviate the problem as follows. Since we have generated useful paths when we solved the LP, we include all these paths in the ILP. We then round down the decision variables f_p, which leads to an integral solution that satisfies the capacity constraints. Because of the rounding down, some of the agents will be left without paths. For these agents, we construct additional paths using the Cooperative A* algorithm [195]. These paths are added to the ILP, which then can be solved to optimality. An alternative approach, albeit not used in our experiments, is to artificially increase the sizes d_i of the groups while solving the LP-relaxation problem, such that after rounding down the f_p variables all agents will be assigned a path.

5.5 Agent Motion Planning

In this section, we elaborate on the second phase of our framework. The goal here is to generate the final motion of each agent A_{ij} based on its path computed by the ILP. Such a path is defined in the time-expanded graph G^T and can be described by the arcs that it uses. A time-expanded arc is either a waiting arc or a traversal arc that corresponds to an edge in the MA graph MG. Therefore, a straightforward approach for generating the final trajectory of the agent A_{ij} would be to let A_{ij} follow the MA edges, while adjusting its speed in order to adhere to the time-constraints of its ILP path. Note, though, that multiple agents may have the same path in G^T or share the same edge or node at the same time. However, since all paths produced by the ILP satisfy the capacity constraints, there is enough clearance for these agents to walk or stand next to each other. As a result, we can utilize the maximum capacity of each edge and discretize its area into a number of *lanes*. The agents will use these lanes to spread over the environment and reach their goals, respecting at the same time the space-time constraints provided by the ILP paths.

5.5.1 Constructing Lanes

We construct the lanes by exploiting the properties of the ECM data structure and its underlying graph $MG = (V, E)$. Let $e = \{n, m\}$ denote an edge in the set E, where both n and m are not dead-end vertices. Let us also define a pseudo-orientation for the edge, e.g. from n to m. As described in Section 5.4.1, an edge consists of a number b of sample points B_i, $1 \leq i \leq b$, including the two vertices and the event points of the edge. For each sample point its minimum clearance R_i is also stored along with its closest points to the obstacles' boundaries. Given the orientation of the edge, let l_i and r_i denote the left and right closest obstacle points assigned to each point B_i. Then the edge e can be expressed as a sequence of *portals*, where each portal is described by the tuple (B_i, r_i, l_i) (see Figure 5.4(a)).

Lanes can be easily constructed along the edge by placing waypoints on the navigation portals. Let c_e^* be the integral capacity of the edge in the capacitated graph G. This capacity defines the maximum number of agents that can simultaneously traverse the edge and hence, c_e^* lanes will be created. For each lane $L_k, k \in [1, c_e^*]$, a waypoint is generated on each of the b portals of the edge based on a parameter $\omega_k = k/(c_e^* + 1)$, $\omega_k \in (0, 1)$. In case the waypoint is located at the right side of the portal, i.e. $\omega_k \leq 0.5$, its exact position w_i is given by linearly interpolating between r_i and B_i as follows: $w_i = r_i + 2\omega_k(B_i - r_i)$. If the waypoint is at the left side of the portal, its position between the left boundary point l_i and the sample point B_i is computed as: $w_i = l_i + (B_i - l_i)(2 - 2\omega_k)$. Having computed the waypoints, the lane can be formed by a concatenation of one-dimensional curves, i.e. line and parabolic segments. Two successive waypoints, w_i and w_{i+1}, are connected with a parabolic arc[4] if $l_i = l_{i+1} \wedge r_i \neq r_{i+1}$. Similarly, a parabolic arc is defined, if $r_i = r_{i+1} \wedge l_i \neq l_{i+1}$. In any other case, a line segment is used to connect two adjacent waypoints.

We refer the reader to Figure 5.4(b) for the lanes corresponding to the edge e (given that $c_e^* = 5$) and to Figure 5.4(c) for a visual depiction of all the lanes that are formed across the edges of the environment. Since each edge can also be traversed in the opposite direction, for each lane its reverse lane L_k' is also added using the same procedure as above.

5.5.2 Trajectory Synthesis for an Agent

Given the lanes as constructed above and the ILP path p_{ij} corresponding to the agent A_{ij}, we now describe how we can determine a trajectory for A_{ij}. Let p_{ij} consists of q arcs. We first express this path as a sequence of tuples $(a_1, T_1)....(a_q, T_q)$, where for the θ^{th} tuple $a_\theta = (n(\tau), m(\tau + l_{a_\theta}^{'*}))$ denotes an arc in the time-expanded graph G^T and T_θ is the time instant at which the task of the arc (waiting or traversing a lane) should be completed. This time is defined as $T_\theta = (\tau + l_{a_\theta}^{'*})\Delta t + t_{add}$. The parameter $(\tau + l_{a_\theta}^{'*})$ denotes the time in discrete time steps Δt at which the node m should be reached, where the arc length $l_{a_\theta}^{'*}$ is defined as in (5.17). The parameter t_{add} reflects the uncertainty in the agent's decisions due to the presence and movements of other agents. This parameter provides a more relaxed time-plan for the agent and hence, more time to complete its task and resolve challenging interactions. Note that, since the same additional time is added for all arcs

[4]This arc can be expressed as a quadratic Bézier curve and hence, it is defined by three control points: the waypoint w_i, the intersection of the tangents at the waypoints w_i and w_{i+1}, and the waypoint w_{i+1}. The tangent at w_i is given by the line through w_i that is perpendicular to the line connecting l_i and r_i. The tangent at w_{i+1} is computed in a similar manner.

and for all the agents, the constraints of the ILP are not violated. In our prototype, t_{add} was set to $1sec$.

Having determined the times T corresponding to each arc a, we can steer the agent toward its goal as follows. Let $\mathbf{x}_{ij}$ denote the current position of the agent, $\mathbf{v}_{ij}$ its current velocity, and $\mathbf{v}_{ij}^{\text{pref}}$ its preferred velocity. Let also (a_θ, T_θ) denote the next tuple to be processed. If a_θ is a waiting arc, we set the preferred velocity to zero for the next $T_\theta - t$ seconds, where t is the current time of the simulation; note that if $T_\theta \leq t$, we simply ignore the waiting arc and query the next tuple in the list. In case a_θ is a traversal arc, we determine its corresponding edge e in MG along with the edge's lanes $\mathcal{L}$ that have the same orientation as the arc a_θ. The agent needs to select one of these lanes and traverse it within T_θ.

Choosing a Lane

We first determine all lanes in $\mathcal{L}$ that are *free*. A lane L_k is considered as free, if no other agent has entered it at the current time t and if its reverse lane L'_k is unoccupied. Among the free lanes, we select the one that is closest to the agent. Given a lane L_k, the distance $D(L_k, \mathbf{x}_{ij})$ between the agent's current position $\mathbf{x}_{ij}$ and L_k can be computed by projecting $\mathbf{x}_{ij}$ onto the segments defined by the waypoints of the lane.

An agent, due to interactions with other agents, may deviate from the time plan provided by the ILP and arrive at its next arc later than the expected time T_θ. Thus, in practice, there may be no free lane to follow. However, as discussed above, a more relaxed time-plan can be provided for each agent by increasing the parameter t_{add}. Alternatively a heuristic approach can be employed that dynamically assigns with each lane L_k a cost depending on the following criteria:

- The distance $D(L_k, \mathbf{x}_{ij})$ between the agent and the lane, as defined above.
- The energy $E(L_k)$ that the agent needs to spend in order to traverse L_k.

More precisely, we approximate the energy cost for crossing the lane L_k as:

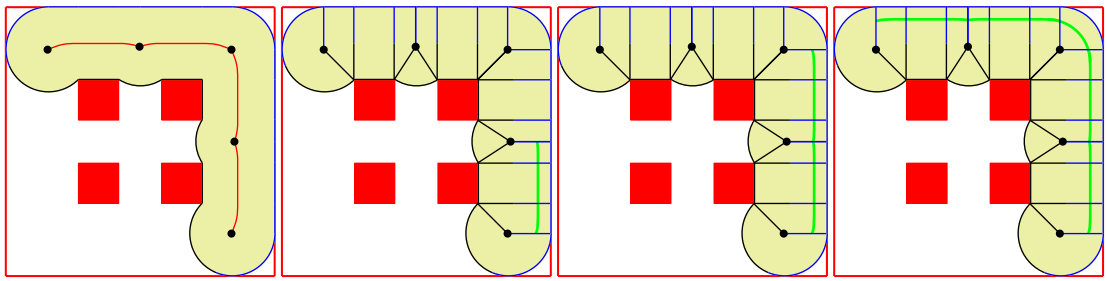
$$E(L_k) = (2.23/\overline{v} + 1.26\,\overline{v})\; d_k, \tag{5.20}$$

where $\overline{v}$ determines the average speed of all agents that are using the lane L_k and its reverse lane L'_k at the current time t, and d_k denotes the length of L_k. In practice, though, we assume that all the lanes of the edge have the same length and thus, without loss of generality, we set $d_k = 1$. The equation quoted above has been recently exploited to generate energy-efficient trajectories for guiding agents in crowd simulations [65]. It is based on a series of experimental studies regarding the relationship between the walking speed and the energy expenditure of adults [238]. As can be observed, if L_k consists of slow moving agents, e.g. due to head-on interactions, a large energy cost will be assigned. The same applies when the lane is traversed by fast moving agents, e.g. in a panicking situation. In contrast, the minimum energy is predicted when $\overline{v} = 1.33\,\text{m/s}$, which corresponds to the normal walking speed of an adult.

Given $E(L_k)$, the total cost of the lane L_k can now be defined as:

$$\text{cost}(L_k) = c_D\, D(L_k, \mathbf{x}_{ij}) + c_E\, E(L_k), \tag{5.21}$$

where c_D, c_E are weights specifying the relative importance of each cost term. Based on the above function, the agent A_{ij} retains the lane $L_s \in \mathcal{L}$ with the lowest cost.



(a) (b) (c) (d)

Figure 5.5: (a) The explicit corridor corresponding to the ILP path of the agent. (b) The initial indicative route. (c) The new indicative route formed by concatenating a lane from the next Voronoi edge to be traversed. (d) The final indicative route.

Moving Along a Lane

The agent A_{ij} uses the selected lane L_s as an *indicative route* in order to traverse the Voronoi edge corresponding to its current arc a_θ. More precisely, at every step of the simulation, an attraction point $\alpha(\mathbf{x}_{ij})$ moves along L_s and attracts the agent as explained in Section 3.5.3. Based on the chosen attraction point, we can estimate the preferred velocity of A_{ij} as: $\mathbf{v}_{ij}^{\text{pref}} = v_{ij}^{\text{pref}} \mathbf{n}_{ij}^{\text{pref}}$, where $\mathbf{n}_{ij}^{\text{pref}}$ is the unit vector pointing from $\mathbf{x}_{ij}$ to $\alpha(\mathbf{x}_{ij})$. The speed v_{ij}^{pref} is determined based on the distance that A_{ij} still has to traverse to reach the endpoint of L_s given the time $T_\theta - t$ that has at its disposal. Note that if $T_\theta < t$, the agent is already behind the schedule provided by its ILP path. In this case, we set $v_{ij}^{\text{pref}} = v^{\max}$ and A_{ij} tries to traverse the lane L_s as fast as possible.

As soon as the end of the lane is reached, the next tuple $(a_{\theta+1}, T_{\theta+1})$ in the list is processed. We should point out that every new lane that the agent has to follow is concatenated to the already existing indicative route. Such a link is obtained by connecting the last waypoint of the current lane to the second waypoint of the next lane using either a straight line or a parabolic segment as explained above.[5] Figure 5.5 shows an example of indicative route formed by concatenating the lanes of the environment. Note that the (explicit) corridor of the resulting indicative route is constructed at once, at the beginning of the simulation, since we already know the Voronoi edges that the agent will visit to arrive at its destination.

Indicative Route Method and Local Collision Avoidance

At each simulation step, the preferred velocity $\mathbf{v}_{ij}^{\text{pref}}$ of the agent A_{ij} is given as an input to the Indicative Route Method (IRM) in order to update the agent's position and compute a smooth path. In particular, a steering force $\mathbf{F}_s$ is exerted on A_{ij}. This force is defined as in Equation 3.4, which enables the agent to gradually adapt its actual velocity $\mathbf{v}_{ij}$ and reach $\mathbf{v}_{ij}^{\text{pref}}$. Consequently, $\mathbf{F}_s$ allows the agent to either advance along its selected lane, or wait at a specific location. In addition, a boundary force $\mathbf{F}_b$ is applied on A_{ij} to guarantee that it will stay inside its corridor and will not collide with the static part of the environment (see Equation 3.1).

[5] We can concatenate lanes using only line segments. The final paths of the agents would still be smooth, i.e. C^1-continuous, due to the properties of the Indicative Route Method (see Chapter 3). However, since quadratic Bézier splines can be easily constructed by exploiting the ECM data structure, parabolic arcs are used to connect successive lanes, whenever such arcs can be defined.

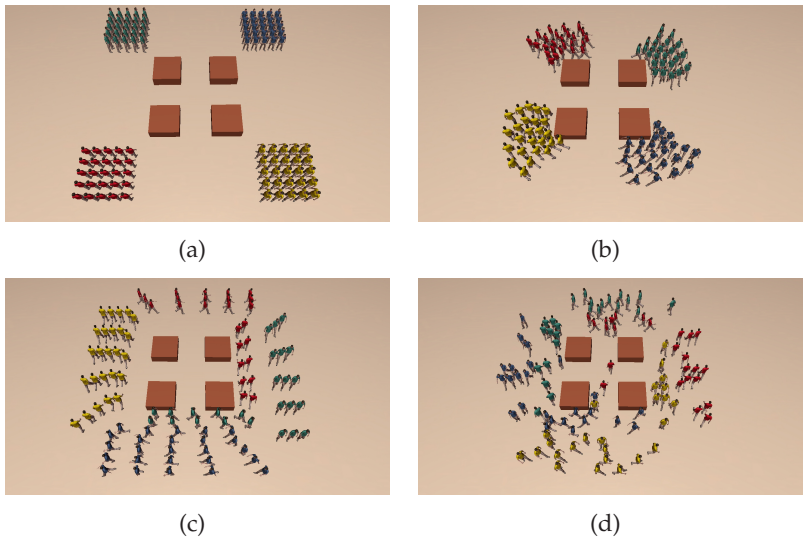


Figure 5.6: Four Blocks scenario. (a) Four groups in opposite corners of the environment exchange positions. (b, c, d) Our approach can automatically generate different macroscopic behaviours through the blocks.

Note that simply using the steering and boundary forces may not be sufficient for a collision-free motion; the agent may still need to resolve collisions with its neighbouring agents while advancing along its selected lane or waiting at a specific location. However, due to the use of lanes, the number of interactions are quite limited. Typically, such interactions take place when agents meet at the vertices of the capacitated graph or exhibit waiting behaviour. To solve these interactions, a local method for collision avoidance should also be used. In general, any method that is based on collision prediction is applicable, see e.g. [173,182,227]. In our implementation, we consider the predictive avoidance model proposed in Chapter 6 that tackles the collision avoidance problem using social forces.

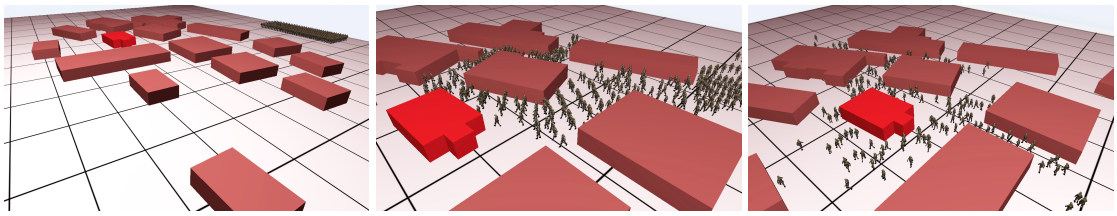
In the predictive model, a number of forces acting on the agent and determine its motion: an evasive force resolves collisions and near collisions with other agents, a repulsive force keeps the agent away from static obstacles, and an attractive force guides the agent toward its goal. We refer the reader to Chapter 6 for additional details. In our case, the evasive force $\mathbf{F}_e$ is computed as given in Equation 6.11, the obstacle force is simply the boundary force of the corridor $\mathbf{F}_b$, whereas the attractive force is replaced by the steering force $\mathbf{F}_s$ of the IRM. The net force $\mathbf{F}$ acting on the agent A_{ij} at a given time-step is then given by:

$$\mathbf{F} = \mathbf{F}_s + \mathbf{F}_b + \mathbf{F}_e \tag{5.22}$$

This force results in an acceleration term as described by Newton's second law of motion and hence, the velocity $\mathbf{v}_{ij}$ and position $\mathbf{x}_{ij}$ of the agent can be updated using numerical integration.

5.6 Results

We have implemented our approach to validate the quality of the generated motions and test its performance. All of our experiments were run on a 2.4 GHz Core2 Duo CPU (on a single thread). We used ILOG CPLEX 11.0 [30] as our LP and ILP solver. We demonstrate



(a) (b) (c)

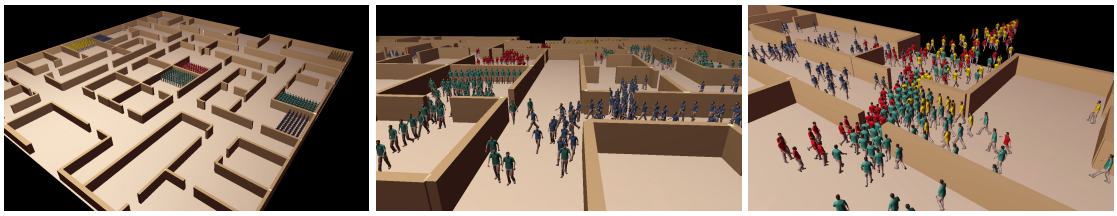
Figure 5.7: Military scenario. (a) An army of 400 soldiers needs to move toward a designated goal area in a village-like environment. (b, c) Intermediate positions of the agents obtained using our approach.

the applicability of our approach in the following challenging scenarios

4-Blocks: Four groups, each consisting of 25 agents, are placed in each corner of an environment and have to move to their diagonally opposite corners. In the middle of the environment, four squared-like obstacles form narrow passages, as shown in Figure 5.6(a). Existing agent-based methods lead to a congestion in the center of the environment. Using our approach, the agents anticipate that a congestion will occur and avoid it by moving around the obstacles (see Figure 5.6(b)). Note that the recent work of Patil *et al.* [164] can also handle such challenging scenarios. However, it requires user intervention to guide the agent flows and resolve congestions, whereas our approach automatically directs the agents over alternative routes based on our ILP formulation. Depending on the parameters of our algorithm, such as the personal space or the capacity information, our method is also able to generate a wide variety of macroscopic behaviours, as illustrated in Figure 5.6. Note that, in this scenario, a small capacitated graph was used consisting of 9 nodes and 12 edges.

Narrow Bottleneck: Two groups, of 56 agents each, travel in opposite directions and have to exchange their positions while passing through a narrow bottleneck, as depicted in Figure 5.1(a). This scenario is challenging and cannot be addressed by local collision avoidance schemes; as the bottleneck becomes crowded, the agents start to exhibit undesirable behaviours and eventually they get stuck in the congested area (see Figure 5.1(b)). Our method is able to prevent such deadlocks from occurring by regulating the starting times of the agents, incorporating explicit waiting behaviour and successfully dealing with opposing flows of agents through the use of indicative routes (see Figure 5.1(c)). Note that some recent agent-based methods attempt to resolve deadlocks by locally coordinating the movements of the agents (e.g. [198,246]). However, such approaches cannot give any guarantees on the resulting behaviour of the agents. In contrast, our algorithm ensures that the agents will reach their destinations providing that they follow the space-time plan computed by the ILP. In this scenario, we use a capacitated graph containing 4 nodes and 3 edges.

Military: An army of 400 soldiers needs to move to a designated goal area in the village-like environment depicted in Figure 5.7(a). This is a model of the McKenna MOUT training center, hosted at Fort Benning, Georgia, USA. Most commercially available games solve the aforementioned problem by letting all the group units follow the same path computed by an A* shortest path algorithm. This leads to undesirable behaviours, as the soldiers squeeze themselves through the narrow passages of the environment, trying



(a) (b) (c)

Figure 5.8: Office scenario. (a) 700 agents need to evacuate an office building through two narrow exits. (b, c) Simulation frames obtained using our approach.

Scene	Max. travel time (s)	Avg. travel time (s)	Optimality error (%)
4-Blocks	32.41	30.01	1.29
Bottleneck	186.90	129.75	6.36
Military	149.78	117.61	4.69
Office	189.10	123.59	8.63

Table 5.1: Results for the four scenarios. The optimality error is the average percentage error between the actual and the estimated arrival times of the agents.

at the same time to avoid collisions with each other and with the static part of the environment. In contrast, our approach identifies areas with low capacity and divides the soldiers over different space-time paths allowing them to avoid congestions and quickly reach their target (Figures 5.7(b) and 5.7(c)). Moreover, using our IRM framework, the agents exploit the maximum capacity of the corridors and spread over the environment while smoothly move toward their destinations. As compared to the previous two scenarios, in this scenario, a large capacitated graph was used containing 42 nodes and 56 edges.

Office: Figure 5.8 shows 700 agents organized into 7 groups that need to evacuate the main floor of an office building. The floor has two exits and each group moves toward its closest exist. It measures 120x120 meter and its capacitated graph consists of 218 nodes and 235 edges. Our approach directs the group flows over alternative paths in both time and space, allowing the agents to escape through the two exits as quickly as possible. Furthermore, by controlling the starting and waiting times of the agents, no oscillatory behaviour is observed when dense crowds of agents thread through the narrow doorways of the environment.

5.6.1 Quality Evaluation

Besides the visual inspection of the generated simulations, we are also interested in a quantitative evaluation of our model. As a result, for each scenario, we computed the average traveling time of the agents and the maximum traveling time among all the agents (see Table 5.1). By planning in both time and space and exploiting the notion of lanes, interactions among agents are efficiently solved, resulting in smoother trajectories and faster traveling times as compared to existing agent-based systems. In the 4-blocks scenario, for example, using the Reciprocal Velocity Obstacle (RVO) method [227] led to an average traveling time of $58.8\,\mathrm{s}$, whereas the latest arrival time was $83.2\,\mathrm{s}$. Even when we extended the RVO with a roadmap graph running along the medial axis [230], the average and the maximum traveling times were $37.06\,\mathrm{s}$ and $52.59\,\mathrm{s}$, respectively. Similarly,

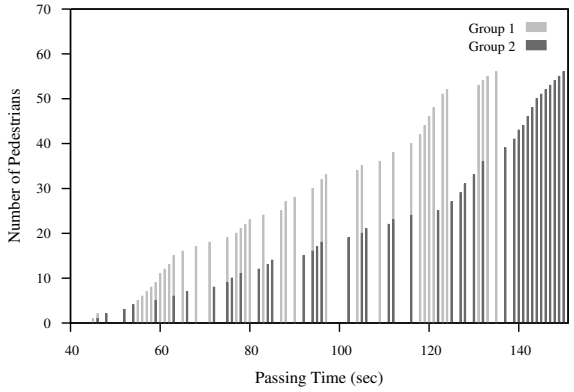


Figure 5.9: Oscillations in the passing direction of the agents at the Narrow Bottleneck scenario. The distance between successive bars of the same color represents the time headway between agents moving in the same direction. The slope of the endpoints of the bars defines the group flow, that is the number of group members that pass through the bottleneck per unit time. The results are in accordance with empirical observations [74].

in the narrow-bottleneck example, it took 219.4 s for the last agent to arrive at its goal and the average travel time was 139.46 s. Analogous results were also obtained when we replaced the RVO method with the ORCA formulation [226]. In the 4-blocks scenario, for example, the average and latest arrival times were 37.67 s and 47.19 s, respectively. The corresponding numbers in the narrow-bottleneck benchmark were 138.03 s and 203.8 s, respectively. Such high traveling times are expected, since the motions of the agents are not taken into account during the global planning and hence, congestions and deadlock situations arise. In contrast, using our approach, such situations are predicted and successfully avoided. Note that, in the 4-blocks scenario, even when we combined the ORCA method with a dynamic roadmap to account for congestion as in [65], the maximum traveling time was once again noticeably high (45.79 s), as the agents were still prone to local minima problems.

To adequately assess the quality of our approach, we also need to determine how efficient is our proposed steering algorithm used in the second phase of our framework. For that reason, we compared the *actual* time that an agent requires in order to reach its destination with its *expected* arrival time. The latter is simply the length of the agent's time-expanded path computed by the ILP (enlarged with the additional times t_{add}). The *percentage error* between the actual and the expected arrival time can then be used as a measure of the optimality of the agent's trajectory. Clearly, since the agent needs to avoid collisions with other agents and is also subject to dynamic constraints, its actual traveling time is higher than the expected one. However, as can be seen in the last column of Table 5.1, in all of our experiments, the optimality error is less than 10% indicating that our steering algorithm generates near time-optimal trajectories.

Emergent Behaviours: Our method is able to generate well-known crowd phenomena which have been noted in the pedestrian literature, such as lane formation (e.g. 4-blocks and military scenarios), queuing and following behaviours (narrow bottleneck and office evacuation), as well as slowing down and waiting behaviours to resolve challenging interactions (narrow bottleneck, military and office evacuation).

Furthermore, in the narrow bottleneck scenario, arch-like blockings are formed at either side of the passage when the two opposite flows meet (Figure 5.1(c)). This phenomenon is in accordance with real-life behaviour [74]. Typically, in such situations,

Scene	#Agents	Graph	t_{LP}	t_{path}	t_{add}	t_{ILP}	Total planning time	Avg. sim. time
4-Blocks	100	(9, 12)	0	15.6	0	0	15.6	0.28
Bottleneck	112	(4, 3)	109.2	46.8	78	124.8	358.8	0.29
Military	400	(42, 56)	343.3	140.4	0	31.2	514.8	0.37
Office	700	(218, 235)	561.6	205.2	62.4	327	1156.2	0.72

Table 5.2: The performance of our approach on the four example scenarios. Reported times are in msec. The average simulation time is expressed in msec/frame. The third column indicates the number of nodes and edges of the underlying capacitated graph.

oscillatory changes of the passing direction are observed, if people do not panic. This allows the pedestrians to efficiently resolve the deadlock and pass through the bottleneck. In our simulation, virtual pedestrians exhibit similar behaviour as highlighted in Figure 5.9. The oscillation frequency is small and normally clusters of pedestrians rather than single individuals try to pass the narrowing from one side to the other.

Regarding the office scenario, our space-time planning approach allows the agents to coordinate their movements and quickly escape from the building through the two narrow exits. However, in real-life evacuations, people tend to exhibit panic behaviour and hence, they move and pass through the bottlenecks in an uncoordinated manner. Typically, people walk considerably faster than normal which results in temporary clogging phenomena that create congestion at the doorways and slow down the evacuation flow. Such behaviour can be obtained by inserting 'aggressive' agents into the simulated crowd, using for example the concept of *composite agents* introduced in [246]. Finally, incorporating waiting behavior in every node of the capacitated graph also led to some visually unpleasant artifacts in the office scenario. As the graph is based on the medial axis of the environment, many nodes are placed in-between the doorways. Of course, waiting at these nodes would be unrealistic, as, typically, people prefer to wait inside the offices. These artifacts can be alleviated by manually indicating the nodes in which waiting behavior can be encoded, or using a different graph than the medial axis that accounts for the behaviour of real people.

5.6.2 Performance

Table 5.2 summarizes the performance of our approach on the four benchmark scenarios. The eighth column of the table indicates the total running time of the first phase of our approach. This time is split into the time used by the column generation part of our algorithm, that is the time for solving the LP-relaxation problem (t_{LP}) and for finding paths using A* in the time-expanded graph (t_{path}), the time needed for finding additional paths using the Cooperative A* due to lack of capacities (t_{add}) and the time to solve the final ILP problem (t_{ILP}). The last column of the table indicates the average computation time of the second phase of our approach, that is the average time taken per simulation step to steer the agents and resolve collisions. As can be inferred from the table, our IRM-based steering algorithm combined with the underlying collision avoidance scheme is very efficient. This is expected, since most of the collisions are taken into account during the ILP planning. Therefore, as explained in Section 5.5.2, only a limited number of interactions need to be resolved at every step of the simulation.

Regarding the total planning time of the first phase, it is clear that the A* algorithm searches paths very efficiently in the time-expanded graph. Furthermore, the step of computing extra paths using the Cooperative A* is negligible in running time and, in practice,

the LP part of the column generation dominates the overall runtime performance. As can be observed, the difficulty of the problem seems to have a high impact on the computational cost of the LP. Consider, for example, the 4-blocks and the narrow bottleneck scenarios. Even though, in both scenarios, the same number of paths needs to be found, in the narrow bottleneck the running time is significantly higher. This is attributed to the limited capacity of the environment, which results in complex linear problems; solving such problems can become very expensive, as the motions of the agents must be coordinated through dense congestion and deadlock situations. The same applies to the office evacuation scenario, where paths need to be simultaneously planned through narrow doorways.

The difficulty of the problem is directly related to the arc and node capacities of the time-expanded graph and can be controlled by the size of the time step Δt used for the discretization of the graph. Note that the chosen Δt allows for a trade-off between accuracy and speed. Shorter time-steps improve the accuracy of the graph at the expense of higher computational times. In contrast, large time-steps result in faster running times; the capacities of the graph are scaled as well, reducing the number of potential bottlenecks in the environment and leading to simpler LP problems. In general, the running time is inversely proportional to the size of the time step, that is as Δt decreases, the running time linearly increases accordingly. This is confirmed by a number of experiments we conducted in different scenarios and for different values of Δt. In our benchmark scenarios, the time-step Δt was set to 1 s with the exception of the office evacuation scenario. Here, to obtain running times that are sufficiently low, we sacrificed the optimality of the LP solution and set $\Delta t = 2$ s.

Besides choosing a rougher discretization of time, we can further reduce the running time of the first phase by quitting the column generation algorithm before we have solved the LP-relaxation to optimality. In addition, if the LP instance is so big that solving the final ILP would require too much time, we can find a feasible solution by rounding the LP solution instead of solving the final ILP. However, since rounding down will reduce the number of agents to which paths are assigned, we need to artificially increase the sizes of the simulated groups. Finally, we can take advantage of the availability of additional CPUs, as well as exploit *thread-level parallelism* to speed up the performance while solving the LP-relaxation and the ILP problems.

5.7 Conclusions

In this chapter, we presented a novel approach for planning and directing large groups of virtual characters by combining techniques from Integer Linear Programming with the Indicative Route Method. We have demonstrated the applicability of our approach through a wide range of challenging scenarios. Simulations created with our system can capture well-known crowd phenomena and natural looking motion patterns that have been observed in real-crowds. Compared to existing techniques, our space-time planning formulation allows our method to successfully resolve congestion, identify and prevent deadlock situations, as well as spread the groups' members over alternative routes. Despite the computational complexity raised by the ILP formulation, our system can run at interactive rates by using the column generation algorithm and choosing an appropriate discretization of time.

Like any other model, our approach makes some simplifying assumptions. Most notably, we assume that the characters choose paths so as to minimize their average traveling time. However, in real-life, other parameters can also affect the path choices that

people make, such as the safety of the area that the path crosses, its attractiveness, etc. Although we could incorporate these factors into our ILP formulation, the resulting problem would be impractical for real-time applications.

We should also point out that our system is specifically designed to model large groups. While ingroup members can have different behaviour characteristics (e.g. size, personal space, maximum speed), during the ILP planning, we assume that they prefer to move with the same desired speed. This enables our method to account for dynamic congestion and other time-consuming situations. We think that this assumption is also justifiable in real-life, where people belonging to the same group tend to exhibit similar walking behaviours.

Our implementation does not require groups to stay coherent. Consequently, group members may split up and take alternative paths trying to reach their targets as quickly as possible. However, group coherence can be obtained either by modifying our ILP program or discarding any isolated path from the solution of the LP-relaxation problem. In addition, although our approach allows the agents to wait at the nodes of the capacitated graph, in certain scenarios, this may lead to unconvincing visual simulations (see, e.g., the Office benchmark in Section 5.6). If our goal is simply to coordinate the movements of the agents so that they can quickly reach their destinations, encoding waiting behaviour in every graph node is not an issue. However, if we strive for realistic simulations, we should either manually annotate the nodes at which waiting is allowed, or use an appropriate input graph as basis for the capacitated graph.

Finally, we must note that the level of heterogeneity between the simulated groups may also have a strong impact on the realism of the resulting simulations. For example, even though we could simulate an army of soldiers interacting with a group of tanks, the generated capacity graph would allow too little space for the soldiers to maneuver, since such a graph is constructed based on the maximum personal space among all entities present in the world.

Despite the aforementioned limitations, we believe that, overall, our solution can be used to efficiently resolve complex planning situations involving one or multiple groups of agents. Our current work focuses on crowds of virtual characters, but it can be easily generalized to address multi-robot planning and coordination problems. Other virtual environment applications can benefit from it as well. An interesting extension, for example, would be to use our formulation for simulating traffic in streets of urban environments and in complex highway scenarios, where congestions and traffic jams are very likely to arise. In this case, refinements to the structure of our capacitated graph and its corresponding ILP will be necessary to account for the kinematic and dynamic constraints of the simulated vehicles and be able to incorporate different types of vehicle motions. Another important direction for future work is to address dynamic and interactive environments, since our current implementation assumes that the virtual world remains static and its dynamics are perfectly known a priori. Note, though, that in dynamically changing environments, the agents may not have the time to plan and replan optimal paths across the time-expanded graphs. To this end, we can trade-off optimality for performance by quitting, for example, the column generation algorithm before we have found an optimal solution to the LP-relaxation problem (see also Section 5.6.2).

Part II

Local Collision Avoidance

Chapter 6

A Predictive Collision Avoidance Model for Multi-Agent Simulation

In the first part of the thesis, we focused on algorithms for computing the global paths of individuals, groups and crowds of virtual characters. While such characters follow their paths to their goals, they should also exhibit human-like behaviour sensing the environment and locally adapting their motions based on the situation at hand. Unfortunately, existing multi-agent systems for real-time crowd and pedestrian simulation lack such natural behaviour leading to a decreased suspense of disbelief. Their main shortcoming lies in the way that the agents interact and avoid collisions with each other.

Several approaches for local collision avoidance have been suggested in the past. Most of them use some sort of reactive planning mechanism, often in the form of a (potential) force field method. In these approaches, though, characters have to be sufficiently close before the reactive forces adapt their routes and resolve the collisions; however, in real-life people react much earlier, sometimes at a distance of more than ten meters. In addition, due to lack of anticipation, the resulting motions are far from natural and are characterized by oscillatory behaviour. The problem becomes more obvious in large and cluttered environments, with characters constantly changing their orientations, pushing each other and moving back and forth.

In this chapter, we address the problem of real-time and visually convincing collision avoidance in multi-agent systems. Although we also employ a force-based approach to drive the agents' motions, our model differs from existing force-based solutions. It is a predictive rather than a reactive approach and is based on the hypothesis that an individual adapts its route as early as possible, trying to minimize the amount of interactions with others and the effort required to avoid the collisions.

Building upon this hypothesis, in our model, each agent computes with which other agents it is in collision course within a certain anticipation time. It calculates how such collisions will take place and then makes an efficient move to avoid them. Consequently, the characters do not repel each other, which is the case in many approaches based on particle systems and (social) force models, but rather anticipate future situations avoiding collisions long in advance and with minimal effort. See Figure 6.1 for an example, showing the difference between the collision avoiding routes our method produces and the routes produced by existing techniques.

We show that the new approach leads to effort-efficient motions and considerably less curved paths for the agents, resulting in a smooth, natural flow. The generated motions are oscillation-free. The method reproduces emergent behaviours, like lane formation, that have been observed in real crowds. The technique is easy to implement and is extremely fast, allowing real-time simulations of crowds of thousands of characters.

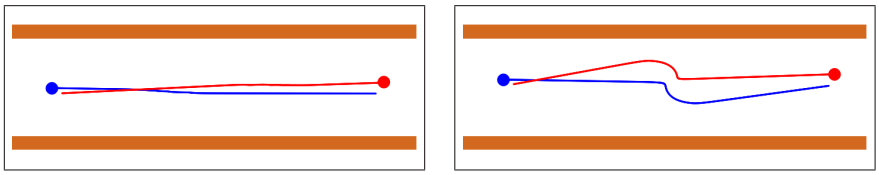


Figure 6.1: Comparison between (a) our predictive avoidance approach and (b) a typical particle-based system.

6.1 Related Work

Over the past twenty years, numerous models have been proposed to simulate individuals, groups and crowds of interacting characters. These include continuous methods that unify global and local navigation into a single framework (e.g. [91, 218]), as well as approaches that decompose global planning from local collision avoidance (e.g. the IRM in Chapter 3 or the approaches in [57,118,170,208]). Since this chapter focuses on local interactions between virtual characters, we highlight here some of the most relevant work in local collision avoidance. We also refer the reader to Section 1.2.2 for a more complete overview.

The most common way to solve interactions between virtual characters is with reactive navigation techniques. Reactive planners originate from the robotics community and are based on variants of potential field methods [102]. In such approaches, the characters avoid colliding with each other by locally adapting their motions. Typically, interactions are modeled as repulsive forces that depend on the relative distances of the characters. In the animation community, the concept of reactive planning was introduced by the work of Reynolds who used simple local rules to describe interactions between autonomous agents [181, 182]. Closely to his work, Musse and Thalmann implemented a multi-resolution algorithm based on simple avoidance laws to handle inter-agent collisions [150], whereas Loscos *et al.* [136] used grid-based rules depending on parameters such as density, speed and direction to resolve collisions between virtual pedestrians. Similarly, Shao and Terzopoulos [193] proposed reactive behavioural routines to determine the avoidance maneuvers of the agents. Although all the aforementioned models are simple to implement, due to their scripted nature, they cannot adapt well to complex and dynamic environments. In addition, despite their flexibility, locally controlled agents do not plan early for collisions and usually react to others when they are already too close leading to unrealistic motions.

Agent interactions have also been extensively studied in the civil and traffic engineering community. The most popular in this field is the work of Helbing. Helbing simulated the behaviour of pedestrians using social forces related to physics [77]. His approach models individuals as velocity-controlled particles whose interactions are described by monotonically decreasing repulsive forces. Helbing's model has been able to reproduce several empirically observed crowd phenomena and has been successfully used in many simulation applications. The approach was later revised by Helbing and his colleagues in [75, 76] and calibrated for specific scenarios [74], whereas in a more recent model the interaction forces between the pedestrians were derived through a series of controlled experiments [147]. Researchers have also exploited social force models and particle systems in the graphics and animation community. Pelechano *et al.*, for example, combined psychological and geometrical rules with social forces to model agent interactions in high-density crowds [165], and Heigeas *et al.* captured the emergence of

collective human phenomena using a mass-spring-damper system [72]. Nevertheless, all these physically based approaches are reactive in nature and do not produce the correct microscopic behaviour. The characters lack anticipation and interact only when they get sufficiently close. Consequently, the resulting motions tend to look unnatural and contain undesirable oscillations.

An alternative way for solving interactions between virtual characters is by anticipating their collisions; assuming that the characters maintain constant velocities, their trajectories are linearly extrapolated and then used to predict and avoid collisions in the near future. Based on this concept, Reynolds proposed the "unaligned collision avoidance" behaviour [182]. Similarly, Feurtey devised an elegant collision detection algorithm that predicts potential collisions and resolves them by adapting the speed and/or the trajectories of the agents [44]. However, his method is not scalable to large crowds. Inspired by Feurtey's work, Paris *et al.* proposed an anticipative collision avoidance method [163]. Although they used motion capture data to calibrate their approach, in their simulations, agents abruptly stop and change their orientations which results in unrealistic flows. A predictive approach was also introduced by Lamarche and Donikian [118] that classifies the types of pairwise collisions between the agents and selects an optimal collision response module for each agent based on a local optimization algorithm.

More recently, van den Berg *et al.* extended the notion of velocity obstacles from robotics [45] and introduced the *Reciprocal Velocity Obstacle* (RVO) method [227, 230]. The idea here is that each agent at every cycle of the simulation selects a new velocity such that no collision will occur with other agents and static obstacles. Nevertheless, in crowded and congested environments, the paths generated by the RVO are far from natural. In addition, as the number of agents grows, the problem complexity and the running time increase considerably. Guy *et al.* significantly improved the performance of the RVO method by using a discrete optimization method to determine the new velocities of the agents [66]. Similarly, the work in [226] allows each agent to efficiently compute an optimal collision-free velocity by solving a low dimensional linear program. Closely related is the egocentric model devised by Pettré *et al.* in [173]. This approach is elaborated from experimental interactions data and uses a combination of speed and orientation adaptations to safely steer virtual agents without colliding with each other. Finally, Ondřej *et al.* have recently introduced a novel vision-based approach that detects future collisions between agents from visual stimuli and employs a reorientation strategy to prevent them, whereas a deceleration strategy resolves imminent collisions [156].

In this chapter, we also propose a model for local collision avoidance that is based on collision prediction. Hence, our work is somewhat similar in nature to the approaches [44, 163, 173, 227] mentioned above. However, it is based on a force field technique and, therefore, it is much easier in its formulation and implementation and considerably faster, allowing the simulation of dense crowds of agents at real-time frame rates. Our approach bears also some resemblance with the avoidance behaviour proposed in [182]. In our model, though, the direction and the magnitude of the collision avoidance forces are computed in a different way, ensuring smooth avoidance behaviour and visually pleasing simulations. We also refer the reader to Section 6.6.2 for a quantitative comparison between our approach and several of the aforementioned state-of-the-art collision avoidance techniques.

6.2 Pedestrian Interactions

In this section we present some key concepts regarding how pedestrians interact with each other in real life. In the next sections, we use these concepts to formulate a model of realistic collision avoidance.

6.2.1 Scanning and Externalization

Pedestrian interactions have been widely studied in the field of sociology. Of particular importance are the studies of Goffman [60] related to how people behave at a microscopic level. According to his observations, two processes govern the avoidance behaviour of pedestrians: *externalization* and *scanning*. In externalization, a pedestrian uses his body language to inform the others about his intentions, that is to indicate his planned course. At the same time, he continuously scans the environment and gathers externalized signals from the nearby pedestrians. Eventually, a voluntary coordination takes place and collisions between interacting pedestrians are resolved. Wolff [240] argues that such a collaboration is essential for successfully solving pedestrian interactions.

6.2.2 Personal Space

Another important concept in social interactions is the *personal space* that surrounds an individual. More formally, the personal space can be defined as the portable territory around an individual which others should not invade [203]. It regulates the safe distance that an individual needs to maintain from others in order to feel comfortable. According to Hall [69], this distance can be used to determine the type of relationship between individuals and decreases as the level of intimacy increases.

In general, the size and the shape of the personal space are constantly changing based on the crowd density and the walking speed of the pedestrians [60, 152]. Other parameters can also significantly affect the personal area, such as the gender and the age of the interacting individuals, their physical dominance, as well as their cultural and ethical background [18, 69, 71, 202]. According to Goffman [60], the personal space can be represented as an oval, narrow at the sides and in the back of the individual and long in front of him/her. Similarly, Navin and Wheeler [152] state that this area has a parabolic shape. More recently Gérin-Lajoie *et al.* [59] have experimentally confirmed the aforementioned observations and shown that pedestrians preserve an elliptical personal space.

6.2.3 Principle of Least Effort

The *principle of least effort* originates from the field of psychology and states that given different possibilities of actions, people select the one that requires the least effort [250]. Consequently, we can assume that, upon interacting with each other, pedestrians try to avoid unnecessary detours and follow effort-efficient trajectories, that is paths that reduce the amount of movement and turning effort that needs to be expended. Based on this principle, Hoogendoorn and Daamen [85] have recently proposed the theory of self-organization in pedestrian flows. According to their model, pedestrians are cost optimizers and thus, they continuously perceive the environment, favouring movements that will minimize their level of discomfort while walking (*pedestrian economicus* [84]); eventually, a Nash-equilibrium state is reached and self-organized behavioural patterns emerge. The principle of least effort has also been exploited by Still to govern the macroscopic behaviour of simulated crowds using cellular automata [205], and by Guy *et al.* to generate effort-efficient trajectories for agents in large-scale crowds [65].

6.3 Multi-Agent Simulation Model

In our problem setting, we are given a virtual environment in which n heterogeneous agents A_i $(1 \leq i \leq n)$ have to navigate toward their specified goal positions $\mathbf{g}_i$ without colliding with the environment and with each other. While our formulation can extend to agents moving in 3D space, for simplicity, we assume that each agent A_i moves on a 2D plane and is modeled as a translating disc with radius r_i. At a fixed time t, the agent A_i is at position $\mathbf{x}_i(t)$, defined by the center of the disc, and moves with velocity $\mathbf{v}_i(t)$. This velocity is limited by a maximum speed $v_i^{\max}$, that is $\|\mathbf{v}_i(t)\| \leq v_i^{\max}$. Hereafter, for notational convenience, we will not explicitly indicate the time dependence.

We propose a social force model for solving the aforementioned planning problem. The behaviour of each agent derives from the following assumptions:

1. At each step of the simulation, each agent A_i is trying to reach its desired goal position $\mathbf{g}_i$. Thus, a goal force $\mathbf{F}_g$ is applied that attracts the agent to its goal.

2. The agent A_i prefers to move toward its destination with a certain speed v_i^{pref}. It tends to reach this speed gradually within a certain time τ [77]. Along with the first assumption, the goal attraction force $\mathbf{F}_g$ can now be defined as in the social model proposed by Helbing and Molnár in [77], that is:

$$\mathbf{F}_g = \frac{1}{\tau}(v_i^{\text{pref}}\mathbf{n}_i - \mathbf{v}_i), \tag{6.1}$$

   where $\mathbf{n}_i = \frac{\mathbf{g}_i - \mathbf{x}_i}{\|\mathbf{g}_i - \mathbf{x}_i\|}$ is the unit vector pointing from agent A_i toward its goal $\mathbf{g}_i$.

3. As the agent A_i advances to its goal, it also has to avoid collisions with the static part of the environment. In our framework, we generally assume that the environment consists of *building walls* modeled as line segments in the 2D plane. Let $\mathcal{W}$ denote the set of walls present in the virtual world. Then, for each wall $w \in \mathcal{W}$ we calculate the shortest distance d_{iw} between the agent A_i and the wall. If this distance is below a threshold value, a repulsive force $\mathbf{F}_w$ is exerted from the wall to the agent. This force can be defined as:

$$\mathbf{F}_w = \begin{cases} \mathbf{n}_w \dfrac{d_s + r_i - d_{iw}}{(d_{iw} - r_i)^{\kappa}}, & \text{if } d_{iw} - r_i \leq d_s \\ \\ 0 & \text{otherwise,} \end{cases} \tag{6.2}$$

   where the constant κ indicates the steepness of the repulsive potential, $\mathbf{n}_w$ is the normal vector of the wall and d_s denotes the safe distance that the agent prefers to keep from the buildings. When κ increases, the potential will increase in steepness toward the boundary of the wall. In addition, by modifying the safe distance d_s, a wide variety of behaviours can be achieved.

4. Each agent A_i keeps a certain psychophysical distance ρ_i from other agents in order to feel comfortable. This distance defines the personal space of the agent. For efficiency reasons, we model this space as a disc $B(\mathbf{x}_i, \rho_i)$ centered at the current position $\mathbf{x}_i$ of the agent and having radius $\rho_i > r_i$. In all of our experiments, this led to compelling avoidance behaviours and hence, a circular personal space was preferred over an elliptical one.

5. An agent A_i perceives the environment to detect imminent and future collisions with other agents. A collision, at some time $t_c \geq 0$, occurs when another agent A_j steps into the personal space of A_i, that is:

$$\exists\, t_c \geq 0 \mid d_{ij} \leq \rho_i + r_j, \tag{6.3}$$

where $d_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|$ denotes the distance between the agents' centers.

6. Given a certain anticipation time t_α, an agent A_i resolves potential collisions within this time by adjusting its trajectory. It reacts to these collisions in advance and safely navigates toward its goal evading other agents with minimal effort. To simulate this behaviour, an evasive force $\mathbf{F}_e$ is applied on the agent. We further elaborate on this in the next section. Note that the anticipation time t_α can vary between agents.

6.4 Avoiding Collisions

The major novelty of our approach is the evasive force $\mathbf{F}_e$ that an agent selects in order to avoid collisions and near collisions with other agents. In most existing approaches based on social forces or particle systems, such a force is exerted when two characters get too close. In our model, though, we devise a predictive avoidance scheme. Given an agent A_i, our collision avoidance algorithm consists of four steps:

6.4.1 Collision Prediction

In the first step of our algorithm, we compute the set $C_i^{t_\alpha}$ of agents that are on collision course with the agent A_i, given a certain anticipation time t_α. To achieve this, we first infer the desired velocity $\mathbf{v}_i^{\text{des}}$ of A_i. We compute $\mathbf{v}_i^{\text{des}}$ as the sum of its actual velocity and the velocity derived by virtually applying only the goal force and the wall repulsive forces:

$$\mathbf{v}_i^{\text{des}} = \mathbf{v}_i + \left(\sum_{w \in \mathcal{W}} \mathbf{F}_w + \mathbf{F}_g \right) \Delta t, \tag{6.4}$$

where Δt is the size of the time step of the simulation.

Then, we estimate the future position $\mathbf{x}_i'$ of A_i based on its current position $\mathbf{x}_i$ and desired velocity $\mathbf{v}_i^{\text{des}}$ as follows:

$$\mathbf{x}_i' = \mathbf{x}_i + t\mathbf{v}_i^{\text{des}},\ t \geq 0. \tag{6.5}$$

Similarly, we predict the future motions of all the other agents that A_i can see by linearly extrapolating their current trajectories. The viewing area of A_i is defined by its desired direction of motion and its field of view. Note that the agent A_i can only estimate the actual velocities of the other agents and not their desired ones, since it does not know their corresponding goal positions. Thus, from the perspective of A_i, the future position of an agent A_j is given by:

$$\mathbf{x}_j' = \mathbf{x}_j + t\mathbf{v}_j,\ t \geq 0. \tag{6.6}$$

We can now determine whether the agent A_i collides with another agent A_j. According to Equation 6.3, a collision occurs at some moment in time $(t \geq 0)$ when A_j lies inside or intersects the personal space $B(\mathbf{x}_i, \rho_i)$ of A_i. By taking the Minkowski difference between the disc corresponding to A_j and the personal space of A_i, the problem is reduced into a ray-disc intersection test (Figure 6.2) that results in the following equation:

$$\|\mathbf{x}_j - (\mathbf{x}_i + \mathbf{v}t)\| = \rho_i + r_j, \tag{6.7}$$

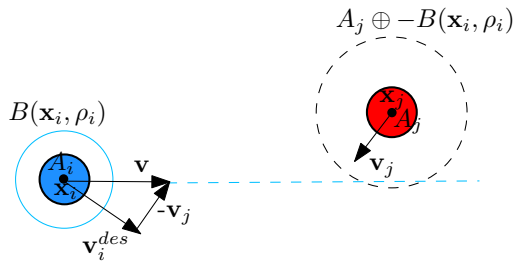


Figure 6.2: Determining whether A_i will collide with A_j. A collision occurs if the ray starting at $\mathbf{x}_i$ and heading in the direction of $\mathbf{v}$ intersects the Minkowski sum of A_j and $-B$, that is the disc corresponding to A_j enlarged with the size ρ_i of A_i's personal space.

where $\mathbf{v} = \mathbf{v}_i^{\text{des}} - \mathbf{v}_j$ denotes the relative velocity between A_i and A_j. Solving the above equation for t, we can estimate the possible collision times tc_{ij} between the personal space of the agent A_i and the agent A_j. If the equation has no solution or a single solution, then no collision takes place. If there are two solutions (t_1 and t_2), three cases can be distinguished:

- $t_1, t_2 \leq 0$: this is a past collision and can be ignored.
- $t_1 < 0 < t_2 \vee t_2 < 0 < t_1$: this is an imminent collision, that is $tc_{ij} = 0$, and hence, the agent A_j is inserted into the set $C_i^{t_\alpha}$.
- $t_1, t_2 \geq 0$: a collision will occur at time $tc_{ij} = min(t_1, t_2)$. If $tc_{ij} \leq t_\alpha$, the agent A_j is inserted into the set $C_i^{t_\alpha}$.

Consequently, the set of agents that are on collision course with A_i is defined as:

$$C_i^{t_\alpha} = \bigcup_{j \neq i,\, 0 \leq tc_{ij} \leq t_\alpha} \{A_j, tc_{ij}\} \tag{6.8}$$

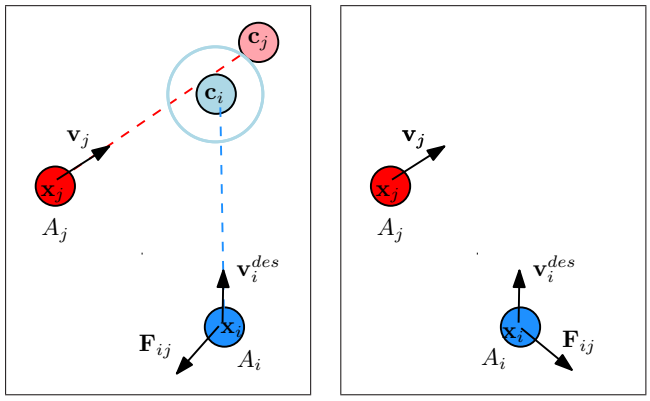
6.4.2 Selecting Colliding Agents

Having computed the set $C_i^{t_\alpha}$, we sort it in order of increasing collision times and keep the first N agents of the set. Preliminary experiments (see Section 6.6) have indicated that this number can be kept small (between 2 to 5 agents), ensuring smooth avoidance behaviour. This not only reduces the running time of our algorithm, but also reflects natural human behaviour. In real-life, an individual tries to avoid a limited number of other pedestrians, usually those that are on collision course with him/her in the coming short time. Similarly our virtual pedestrian A_i tries to evade the N agents with which it will collide first.

6.4.3 Avoidance Maneuvers

We now show how the agent A_i can avoid a potential collision with another agent A_j that belongs to the set $C_i^{t_\alpha}$. Let $\mathbf{c}_i = \mathbf{x}_i + tc_{ij}\mathbf{v}_i^{\text{des}}$ and $\mathbf{c}_j = \mathbf{x}_j + tc_{ij}\mathbf{v}_j$ denote the locations of agents A_i and A_j at time tc_{ij}; $\mathbf{c}_i$ and $\mathbf{c}_j$ derive from Equations 6.5 and 6.6, respectively.

Based on these future locations, we select an evasive force $\mathbf{F}_{ij}$ for the agent A_i, so that it can smoothly avoid the agent A_j. The direction of the force is given by the unit vector



(a) Our approach (b) Particle-based approach

Figure 6.3: Example of the force $\mathbf{F}_{ij}$ used to avoid a future collision (a) in our approach and (b) in a typical particle-based method. Note that, using our technique, the direction of the force depends on the relative positions at the moment of the impact, rather than on the current positions of the agents. The light blue disc represents the personal space of the agent A_i.

$\mathbf{n}_{c_i c_j} = \frac{\mathbf{c}_i - \mathbf{c}_j}{\|\mathbf{c}_i - \mathbf{c}_j\|}$ pointing from $\mathbf{c}_j$ to $\mathbf{c}_i$. As an example, consider Figure 6.3(a). The blue agent A_i will hit the red agent A_j at the back. Therefore, it makes a slight move toward the latter and eventually will pass behind it. In contrast, in a typical particle-based system, the red agent would have forced the blue one to move in the opposite direction, leading to a new collision in the near future (Figure 6.3(b)).

The magnitude of the evasive force $\mathbf{F}_{ij}$ is approximated by a piecewise function $f(D)$ (see Figure 6.4) as follows:

$$\begin{aligned}
&\textit{if} \quad 0 < D < d_{\min} \rightarrow \quad f(D) = \frac{\alpha}{D} + \beta \\
&\textit{if} \quad d_{\min} \leq D < d_{\text{mid}} \rightarrow \quad f(D) = \beta \\
&\textit{if} \quad d_{\text{mid}} \leq D < d_{\max} \rightarrow \quad f(D) = \beta \frac{d_{\max} - D}{d_{\max} - d_{\text{mid}}} \\
&\textit{if} \quad d_{\max} \leq D \rightarrow \quad f(D) = 0
\end{aligned} \tag{6.9}$$

where $D = \|\mathbf{c}_i - \mathbf{x}_i\| + (\|\mathbf{c}_i - \mathbf{c}_j\| - r_i - r_j)$ is the distance between the current position $\mathbf{x}_i$ of agent A_i and its future position $\mathbf{c}_i$ plus the distance between the two agents at their time of collision. The constants α, β and the thresholds $d_{\min}$, d_{mid}, $d_{\max}$ can vary among the agents to simulate different avoidance behaviours.

As can be inferred from Figure 6.4, the function $f(D)$ is defined for four intervals. The threshold $d_{\max}$ determines the start of the avoidance maneuver of the agent, whereas $d_{\min}$ defines the beginning of an impenetrable barrier between the agents. The threshold d_{mid} regulates the start of the constant part of the function. This part is used to eliminate jerky behaviour and smoothly connect the linear and exponential parts of the function.

Having determined the value of $f(D)$, the force $\mathbf{F}_{ij}$ can now be defined as:

$$\mathbf{F}_{ij} = f(D)\mathbf{n}_{c_i c_j} \tag{6.10}$$

6.4.4 Computing the Evasive Force

In the last step of our algorithm, we compute the total evasive force $\mathbf{F}_e$ that is applied on the agent A_i given its set $C_i^{t_\alpha}$. Two approaches can be distinguished. The first is

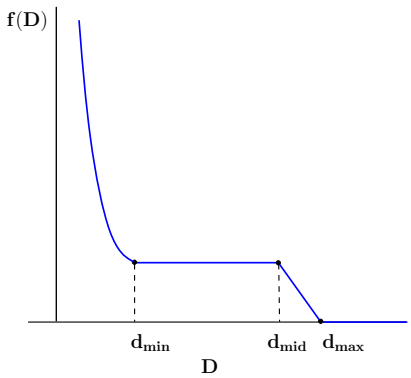


Figure 6.4: The magnitude of the avoidance force $\mathbf{F}_{ij}$ as a function of the distance D.

to take into account each one of the N agents of the set independently and compute a corresponding force $\mathbf{F}_{ij}$ as described in Equation 6.10. Then, the evasive force can be defined as the weighted sum of those N forces:

$$\mathbf{F}_e = \sum_{j=1}^{N} w_{ij}\mathbf{F}_{ij}, \tag{6.11}$$

where the weighting factor w_{ij} gives a higher priority to the most imminent collisions. Note that most models based on social forces simply add the effects of all forces acted on the agent and do not introduce any collision-dependent weights.

The second approach, which also differs from existing solutions, is to apply the forces sequentially (iterative approach). The idea here is as follows. We first consider the top agent in the ordered set $C_i^{t_\alpha}$ and calculate its corresponding evasive force using Equation 6.10. Let us, for notational convenience, name this force $\mathbf{F}_\text{v}$. In the next iteration, we consider the top two agents of the set and determine whether A_i will collide with these agents after (virtually) applying the force $\mathbf{F}_\text{v}$. In other words, we first estimate the desired velocity of A_i by including $\mathbf{F}_\text{v}$ into Equation 6.4. Then, we iterate over the two agents in the set and compute the time of collision tc between A_i and each of the agents (cf. Equation 6.7). If collisions still exist (i.e. $0 \leq tc \leq t_\alpha$), we compute their corresponding avoidance forces as before and determine the new evasive force $\mathbf{F}_\text{v}$ from the weighted average of these forces (Equation 6.11). The resulting force is then used as an input for the next iteration, in which the top three agents of the set $C_i^{t_\alpha}$ are considered.

We repeat this process until all the agents in the set $C_i^{t_\alpha}$ have been accounted. Then, the total evasive force $\mathbf{F}_e$ of A_i is determined as the average of all the forces $\mathbf{F}_\text{v}$. Experiments have confirmed that by applying the forces sequentially a smoother and more realistic avoidance behaviour is achieved, as forces are only added if they are still required to resolve further collisions.

6.5 Implementation

This section provides specific details regarding the implementation of our proposed multi-agent simulation model.

6.5.1 Efficient Collision Prediction

During each simulation step, we have to compute for each agent A_i its set $C_i^{t_\alpha}$ by determining with which other agents it is on collision course given a certain anticipation time

(see Section 6.4.1). A naive implementation would be to iterate over all the remaining $n-1$ agents and check whether a collision will take place, resulting in a quadratic running time per simulation step. However, a more efficient implementation is to prune the search based on some cutoff distance, e.g. the maximum distance that the agent A_i can travel given its anticipation time and maximum speed. Then, A_i only has to consider a limited number of agents for potential collisions. Proximity computations to these agents can be accelerated using a spatial data structure for answering nearest neighbour queries, such as a spatial hashing scheme or a grid map that stores and updates at every simulation step the locations of the agents [159, 193]. In our implementation, we used a bin-lattice spatial subdivision method similar to the one described in [183]. Our technique returns in constant time the nearest neighbours that the agent can actually see, given its current position, orientation and cutoff distance. Note that the number of neighbours within a given search distance is maximally bounded, since the personal space of each agent is modeled as a hard disc with fixed radius and the agents cannot penetrate each other. Let k denote this maximal bound. Then, by selecting a restricted subset of neighbouring agents, the runtime complexity of our algorithm is reduced from $O(n^2)$ to $O(nk)$.[1]

6.5.2 Adding Variation

To increase the realism of the simulation some variation and irregularity is needed among the virtual agents. For that reason, a noise force $\mathbf{F}_n$ is also introduced in our model. This force, on one hand, allows us to take into account random variations in the individual behaviours, as well as incorporate mistakes that individuals make while avoiding each other; after all, despite their intuition and sensors, people bump into others from time to time. On the other hand, such a force is also needed to avoid artifacts that arise from symmetrical patterns, as well as to resolve extreme cases where two agents have exactly opposite directions.

More realism can also be achieved by varying several of the parameters of our model, such as the anticipation time of each agent, the preferred speed, the size of the personal space, etc. Consequently, we can include in our simulations agents with distinct characteristics that exhibit a wide variety of avoidance behaviours.

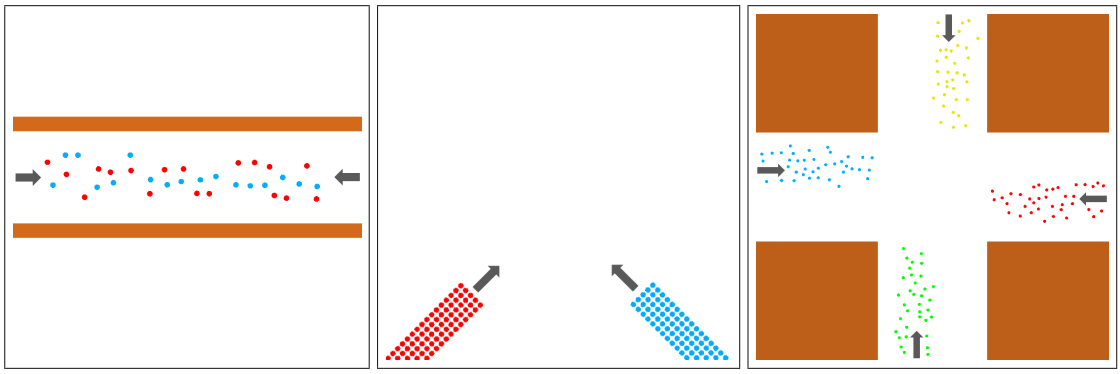
6.5.3 Time Integration

The result from our proposed method is a system of positions, velocities and forces. To simulate this system, we first discretize our model by choosing a time step Δt for our simulation. Then, in each cycle of the simulation, we compute for each agent A_i the goal force $\mathbf{F}_g$ that attracts A_i toward its goal, the total repulsive force $\sum \mathbf{F}_w$ that keeps A_i away from building walls and the evasive force $\mathbf{F}_e$ used to resolve collisions and near collisions with other agents. The noise force $\mathbf{F}_n$ is also taken into account. Given the applied forces, the final force exerting on the agent can be defined as:

$$\mathbf{F} = \mathbf{F}_g + \sum_{w \in \mathcal{W}} \mathbf{F}_w + \mathbf{F}_e + \mathbf{F}_n. \tag{6.12}$$

The force $\mathbf{F}$ results in an acceleration term as described by Newton's second law of motion, i.e. $\mathbf{F} = m\mathbf{a}$, where m is the mass of the agent and $\mathbf{a}$ its acceleration. Assuming

[1] If we require k to be constant, the runtime per simulation cycle is (asymptotically) equivalent to $O(n)$.



(a) Hallway (b) Diagonal crossing (c) 4-way Intersection

Figure 6.5: Benchmark scenarios: a) Two groups of agents heading in opposite directions interact in a hallway. b) Four groups interact at the intersection of two orthogonal corridors. c) Two groups cross paths perpendicularly.

$m = 1$, we update the velocity and position of the agent using numerical integration, that is:

$$\frac{d\mathbf{v}_i}{dt} = \mathbf{F}, \quad \frac{d\mathbf{x}_i}{dt} = \mathbf{v}_i. \tag{6.13}$$

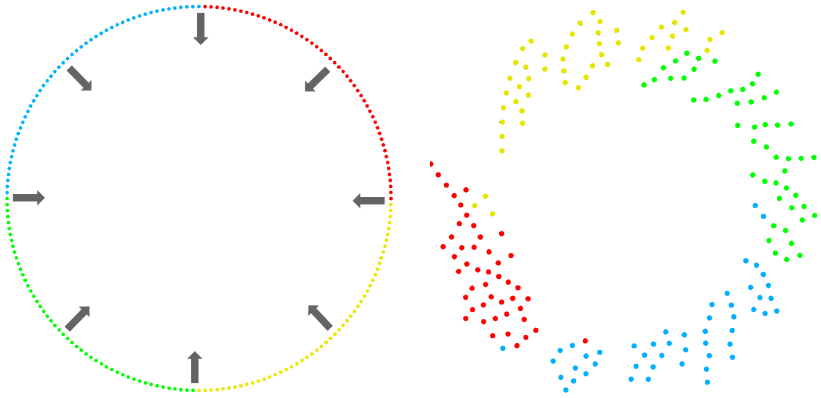
We also infer the agent's orientation θ_i from the weighted average of its previous orientation and the direction defined by the new velocity $\mathbf{v}_i$. To retain stability and avoid stiffness, a stable integration scheme, like Verlet integration [232], is usually recommended. However, in our simulations, we computed an Euler integration of (6.13), since we have not observed any simulation instabilities.

6.6 Results

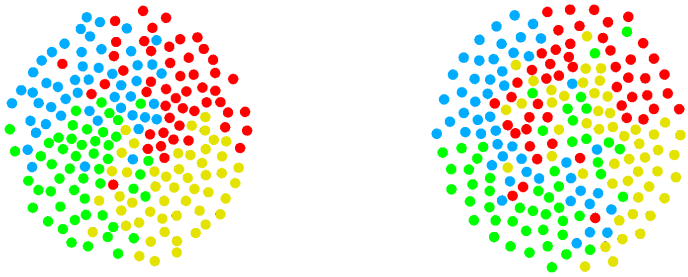
We have implemented our proposed method to experimentally test its applicability in real-time applications and validate the quality of the generated motions. All experiments were performed on a 2.4 GHz Intel Core2 Duo CPU (on a single core) with an ATI Radeon HD 4800 GPU.

We first calibrated the parameters of our model based on a number of test-case scenarios proposed in [197]. These scenarios were designed to capture typical pedestrian behaviours. In all the simulations, the agents' velocities were updated at 10 fps. The agents were modeled as discs with equal radii ($r = 0.4\,\mathrm{m}$) approximating the shoulder widths of typical humans. The preferred speed of each agent was set to $v^{\mathrm{pref}} = 1.4\,\mathrm{m/s}$, which corresponds to the average walking speed of real pedestrians [213]. The actual motions of the agents were limited to a maximum speed of $v^{\max} = 2.0\,\mathrm{m/s}$, whereas the size of the personal space of each agent was set to $\rho = 1.0\,\mathrm{m}$.

During the simulations, we recorded for each time step the position and the direction of every agent. Then, a benchmark tool was implemented to analyze the quality of the generated motions and detect unrealistic behaviours. For that reason, a number of quantitative quality metrics have been devised. In particular, we used the integral of the square of the curvature to measure the smoothness of an agent's path [87]. We also computed the time it takes for an agent to reach its goal, the length of the path it follows, as well as the average speed at which the agent moves. Since we strive for effort-efficient motions, the total acceleration of the agent and the total degrees it turned were also reported to indicate the amount of movement and turning effort spent [197].



(a) Circle scenario (b) Our model



(c) RVO (d) Helbing's model

Figure 6.6: The Circle scenario for 200 agents. Each agent moves to its diametrically opposite position on the circle. Our approach is the only one to capture the emergence of vortex-like patterns.

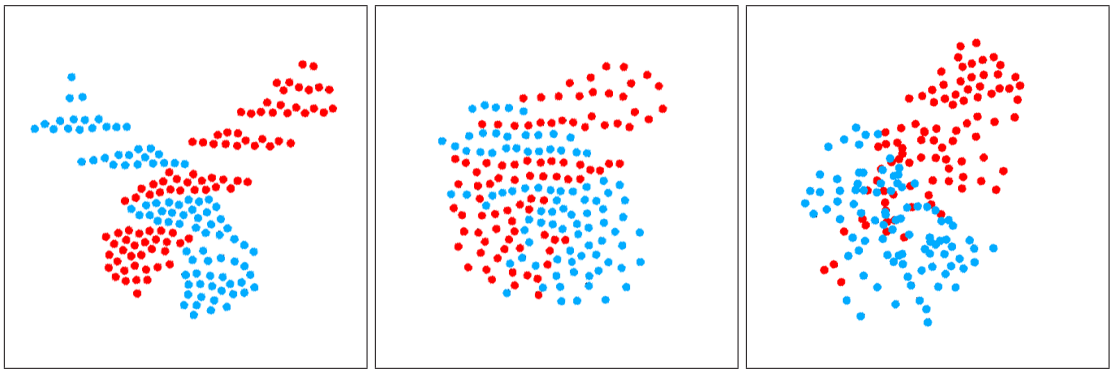
Based on the derived statistics, we experimentally confirmed that a smoother avoidance behaviour is achieved by applying the evasive forces iteratively, as explained in Section 6.4.4. We also determined the optimal parameter settings for our model. Default values are: $N = 5$, $\tau = 0.4\,\mathrm{s}$, $t_\alpha = 8\,\mathrm{s}$, $d_{\mathrm{min}} = 1\,\mathrm{m}$, $d_{\mathrm{mid}} = 8\,\mathrm{m}$, $d_{\mathrm{max}} = 10\,\mathrm{m}$, $\alpha = 1$, $\beta = 3$.

6.6.1 Scenarios

Having performed the initial calibration of our model, we ran a diverse set of interaction scenarios. These include:

Hallway: Agents cross paths while walking in either direction through a hallway (Figure 6.5(a)). The agents should form lanes of uniform walking directions to efficiently resolve collisions [74].

Diagonal crossing: Two groups of agents cross perpendicularly while walking along the diagonals of an obstacle-free domain, as shown in Figure 6.5(b). A main challenge for the agents in this scenario is to quickly reach their goals without significantly deviating from their shortest paths. This can be achieved if clusters of homogeneous agents in terms of desired direction and walking speed emerge within the groups, leading to the formation of diagonal line-shaped patterns [85].



(a) Our model (b) RVO (c) Reynolds' model

Figure 6.7: Diagonal crossing. Our approach leads to the self-formation of stripe-like patterns allowing an efficient overall crowd motion as compared to RVO and Reynolds's model. Note also that the approach of Reynolds can result in many collisions among the agents.

4-way Intersection: Figure 6.5(c) depicts four streams of agents meeting at the intersection of two orthogonal corridors. Here, the agents not only have to avoid collisions with each other but also with the static part of the environment; large walls are present that limit the visibility of each agent and prevent it from perceiving in advance other agents arriving from perpendicular directions. In this setting, a combination of speed and direction adaptations is required to efficiently solve the sum of interactions between the agents.

Circle: 200 agents are evenly distributed along the circumference of a circle and have to move toward their diametrically opposite positions on the circle. This benchmark, initially proposed by van den Berg *et al.* in [227], is very challenging, since all the agents cross nearly at the same time and each one can potentially collide with any of the other agents. Typically, congestion conditions evolve in the center of the circle as the agents try to reach their destinations. Such conditions can be avoided when vortex-like patterns spontaneously emerge leaving the center of the circle void of agents [135]. Corresponding results are shown in Figure 6.6.

6.6.2 Quality Evaluation

Emergent Behaviours

In all of the scenarios, our method was able to capture the emergence of visually interesting and empirically confirmed phenomena that have been observed in real pedestrian crowds. In the first three scenarios, for example, *lanes* are dynamically formed when agents cross in opposite directions. This phenomenon has been widely studied in pedestrian and crowd literature. In our model, though, the characters anticipate future collisions and adapt their motions in advance, evading others in a human-like manner. Even in crowded scenarios, our approach leads to a smooth and natural flow. Agents scan the environment to find a free space to move into. Other individuals follow and a queue (lane) is formed that efficiently resolves the congestion.

During the simulations, we observed that the agents tend to exhibit overtaking behaviour near the edges of the crowd, since the density in these areas is significantly lower than in the center. In the *Diagonal crossing* scenario, *stripe* formations also emerge while

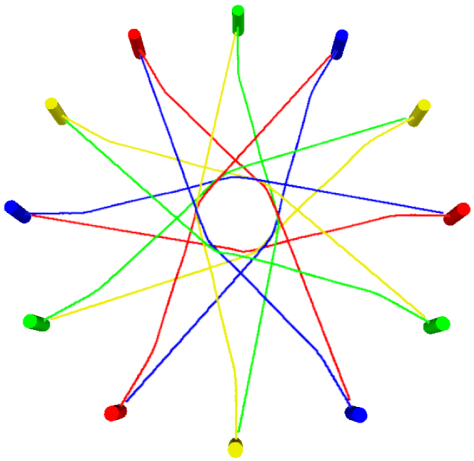
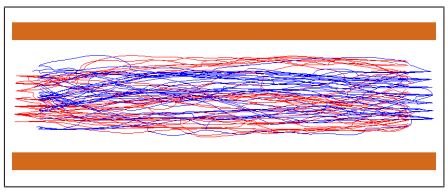
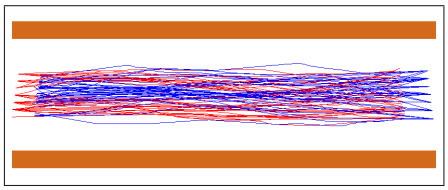


Figure 6.8: The resulting paths for 12 agents in the Circle scenario using our approach. All agents have the same radius, preferred speed and anticipation time.



(a) Helbing's model



(b) Our model

Figure 6.9: Example paths for the Hallway scenario generated by (a) Helbing's model and (b) our predictive avoidance approach.

the agents interact with each other.[2] The agents prefer to slow down and let other agents pass instead of detouring from their planned courses. Therefore, line-shaped clusters of agents are formed whose direction is approximately $45°$ relative to the walking direction, leading to an efficient global motion (see Figure 6.7(a)).

In the *Circle* scenario, the agents anticipate that a congestion will occur in the middle of the environment and a *vortex* is automatically formed that minimizes the number of interactions between the agents and allows them to quickly reach their goals without getting stuck in time-consuming congestion situations. This emergent behaviour has been observed in real life and captured by a number of models (see, e.g., [72]). Our method, though, can successfully reproduce this phenomenon with no oscillatory movements or constant changes in the orientations of the virtual agents. For example, we refer the reader to Figure 6.8 for the trajectories generated by our approach when 12 agents exchange positions in a circle-like scenario.

Finally, the *Intersection* scenario also demonstrates the ability of the agents to slow down and stop in order to resolve imminent collisions due to their limited visibility. Stripe formations and vortex-like patterns temporarily emerge as well, allowing an optimal crowd flow as depicted in Figure 6.10.

Comparisons

We also compared our approach to prior techniques for local collision avoidance using the aforementioned benchmark scenarios. We first ran comparisons with Helbing's social force model (using the implementation proposed in [75]), as this method is representative of typical particle-based reactive planners. Note that, for a fair comparison, we optimized Helbing's model as we did with our predictive approach. In all of our test cases, we observed that agents simulated with Helbing's method tend to adapt their motions rather

[2]Stripes are *dynamic* lanes directed toward the sum of the walking directions of both intersecting groups. Note that the phenomenon of lane formation corresponds to the particular case of stripe formation when the interacting groups have exactly opposite directions.

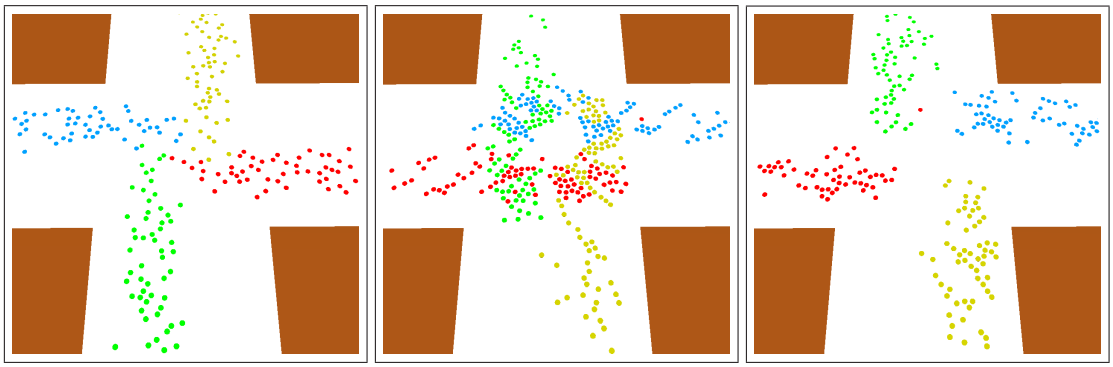


Figure 6.10: Zoomed stills from the Intersection scenario obtained using our approach (evolution in time is shown from left to right). Our technique generates stripe formations and vortex-like patterns similar to the ones observed in real pedestrian crowds.

late in order to avoid potential collisions. This is due to lack of anticipation, since interactions between the agents are formulated as a function of their relative distances. Thus, the resulting paths have a high curvature and the amount of avoidance maneuvers performed by the agents is large. In contrast, with our predictive model, the agents plan early for collisions, avoiding unnecessary detours and minimizing the time needed to reach their goals. In addition, using our approach, the number of interactions between the characters is reduced, which leads to effort-efficient movements and smooth paths. Figure 6.9(a), for example, depicts 100 paths generated by Helbing's method for the *Hallway* scenario, whereas the paths traced by our technique are shown in Figure 6.9(b).

We also compared our approach to the "unaligned collision avoidance" behaviour of Reynolds [182] using the OpenSteer implementation [157]. Similar to our technique, Reynolds's model is force-based and explicitly accounts for collisions between the agents by extrapolating their trajectories in the near future. At every simulation step, each agent detects the most imminent collision and steers laterally to avoid it, adapting at the same time its speed in order to reach the collision site either before or after the predicted time of collision. However, due to the sideways avoidance maneuvers, the resulting simulations differ from the ones generated by our approach. In particular, even though Reynolds's agents plan early for congestion, they tend to spread out following longer paths and taking more time to reach their goals as compared to agents simulated with our technique. We should also point out that, in the *Hallway* scenario, involving head-on interactions, no collisions between the agents were observed using Reynolds's approach. However, in the other three complex benchmarks, the unaligned avoidance model resulted in many agent interpenetrations. For example, the *Diagonal crossing* scenario averaged 14 collisions per frame, whereas the *Circle* scenario led to 22 collisions per frame (see also Figure 6.7(c)).

Finally, we compared our technique against the RVO method proposed by van den Berg *et al.* [227, 230] (using the RVO sampling library implementation [186]). In the RVO, each agent adapts its velocity such that no collision will take place with other agents or the environment. In low-density crowds, the method works well and no significant differences between our predictive avoidance model and the RVO were noted. However, in crowded and congested examples, the RVO produces unrealistic motions as the agents try to guarantee collision-free navigation. Therefore, collective phenomena, such as lane formations, start emerging rather late and the agents seem to lack anticipation moving toward congested areas. In the *Circle* scenario, for example, the agents immediately converge to the center of the environment (see Figure 6.6). Consequently, they have to fre-

quently speed up, slow down or change their orientations to safely reach their goals. On the contrary, using our method, the agents form a vortex-like pattern that decreases considerably the frequency of necessary deceleration, stopping, and avoidance maneuvers, ensuring a smooth flow and more time- and effort-efficient motions.

Quantitative Evaluation

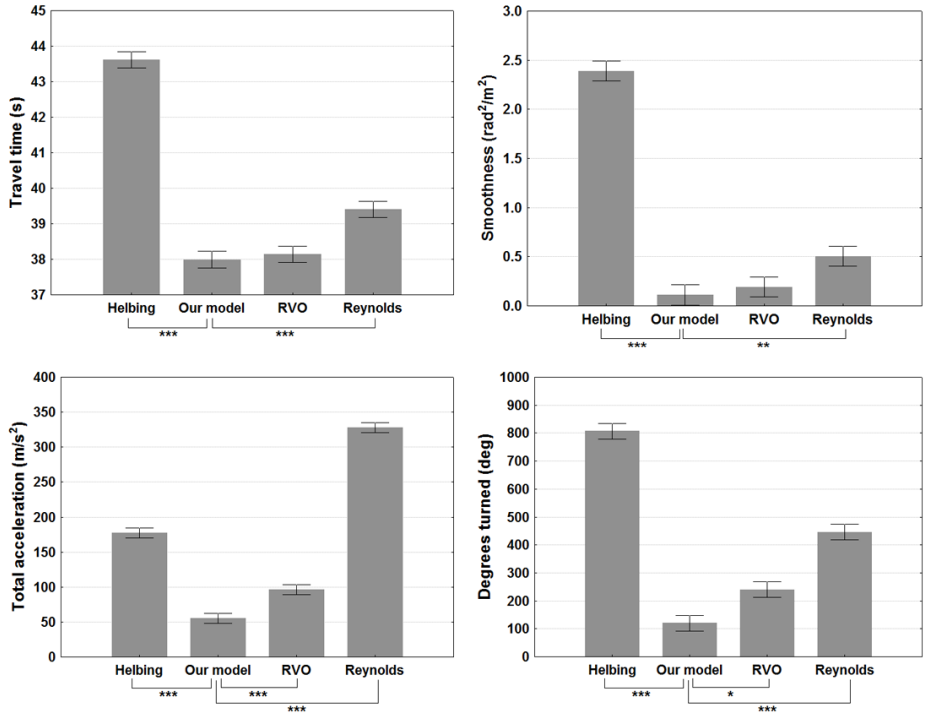
Besides empirically evaluating the quality of the aforementioned local collision avoidance models, we also quantitatively compared them to each other. To be more specific, we performed separate 1-way ANOVA tests for each benchmark scenario using four of our proposed evaluation metrics: the *travel time* that each agent requires to reach its goal, the *smoothness* of the generated paths measured by the sum total of squared curvature samples along these paths, the *total acceleration* of each agent approximated by the sum of its instantaneous accelerations and the *total degrees turned* by the agent while advancing toward its destination.[3]

The corresponding statistics are shown in Figures 6.11 - 6.12 and confirm our empirical observations. Note that regarding Reynolds's model we only report statistics for the *Hallway* scenario. In all the other scenarios, a significant amount of collisions were observed making a fair comparison impossible.

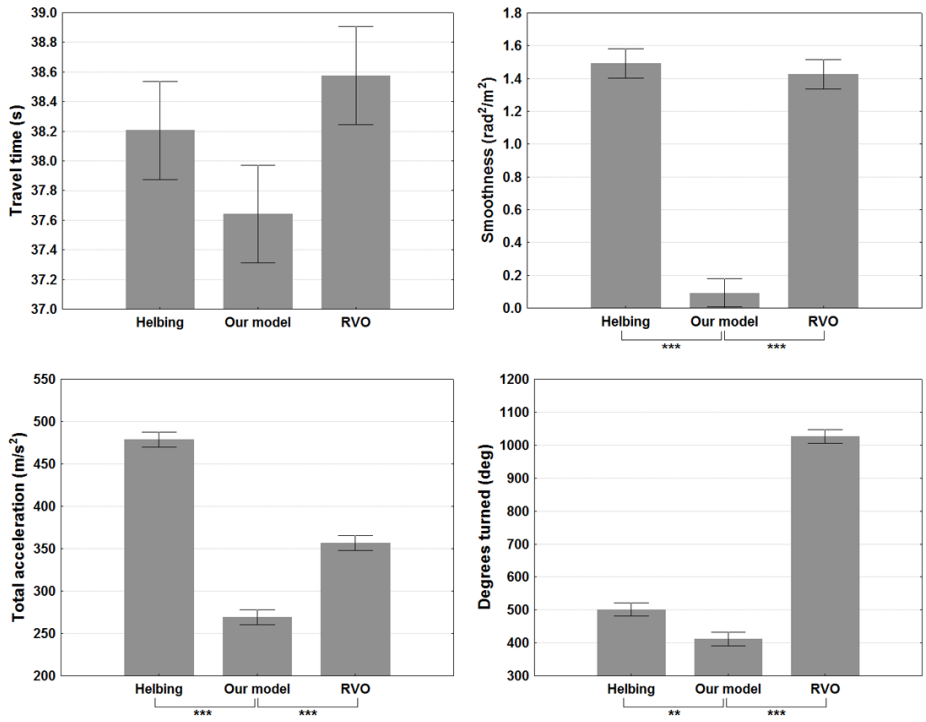
As can be inferred from the two figures, our predictive model results in faster travel times and less curved paths in all scenarios. Pairwise comparisons, based on Tukey's HSD post-hoc test, showed that the differences in travel time and smoothness between our approach and the other techniques were statistically significant. Two exceptions were found. In the *Hallway* scenario, the RVO method exhibited similar behaviour (in terms of travel time and path smoothness) with our model. This was expected, since both approaches are anticipative and their agents can solve such simple interaction scenarios by slightly adapting their motions and forming distinct lanes. The other exception was in the *Diagonal crossing* scenario, where no significant differences in the travel time between our approach, Helbing's model and the RVO method were found. This was due to the fact that all three models were able to predict the emergence of stripe formations, which allowed the agents to efficiently solve interactions without requiring to stop.

Furthermore, Figures 6.11 and 6.12 clearly indicate that the amount of movement effort (total acceleration) is significantly lower using our approach, as compared to the amount expended by the agents in the other models. Similar conclusions can also be drawn for the turning effort (total degrees turned) of the agents, with the exception of the *Circle* scenario. Here, no significant difference was found in the amount of turning effort between agents that employ our model and agents simulated with the RVO method. However, by adapting their orientations, our predictive agents automatically form a vortex that decreases the frequency of necessary deceleration and stopping maneuvers. In contrast, the RVO agents have to frequently speed up, slow down or even stop in order to safely navigate towards their targets, as can be observed by the high acceleration values in Figure 6.12(b).

[3] Although closely related, the curvature measures how fast the direction of the agent's path is changing with respect to the path length, whereas the degrees that the agent has turned measure the amount of orientation changes of the agent and depend on the time step of the simulation. Consider, for example, an agent standing at the same position for a very long time, while its orientation flips at every simulation step. While the curvature is zero, the number of degrees that the agent has turned is very high.

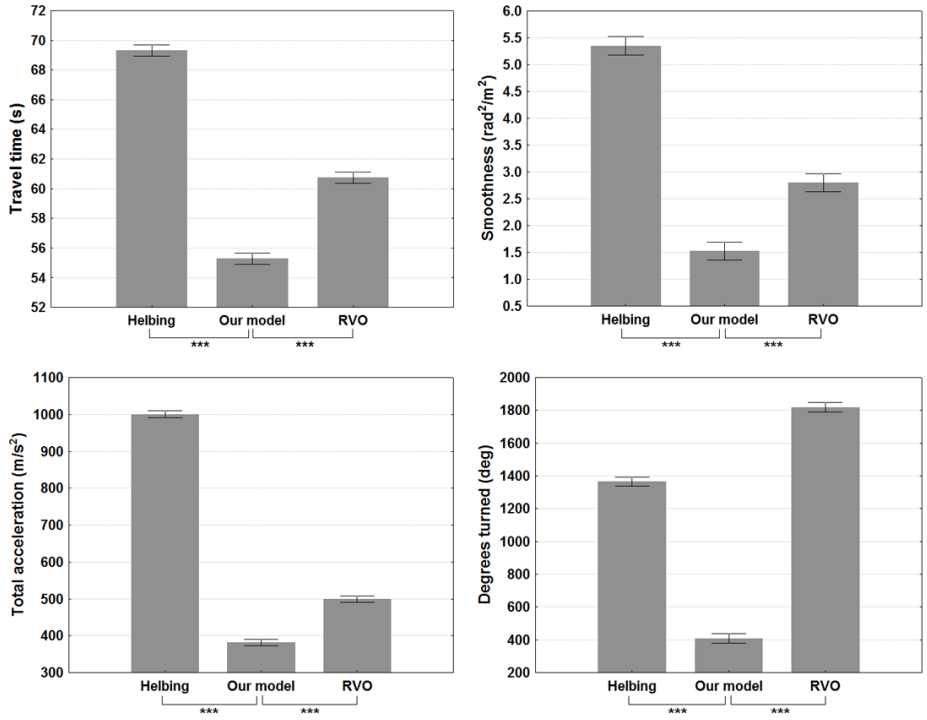


(a) Hallway (75 agents)

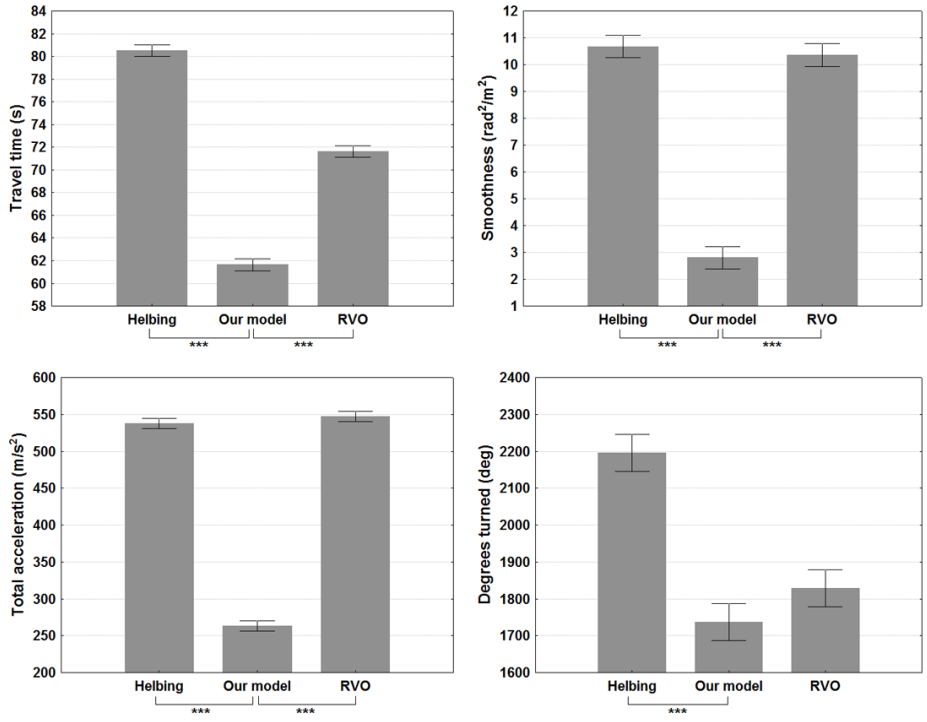


(b) Diagonal crossing (180 agents)

Figure 6.11: Path smoothness, total acceleration, total degrees turned and travel time statistics for the first two benchmark scenarios. Significant pairwise differences between our approach and the other models are illustrated by $^{*}(p < .05)$, $^{**}(p < .01)$ and $^{***}(p < .001)$. Error bars represent standard error.



(a) 4-way Intersection (415 agents)



(b) Circle (200 agents)

Figure 6.12: Corresponding statistics for the Intersection and Circle benchmark scenarios. Significant pairwise differences between our approach and the other models are illustrated by $^{*}(p < .05)$, $^{**}(p < .01)$ and $^{***}(p < .001)$. Error bars represent standard error.

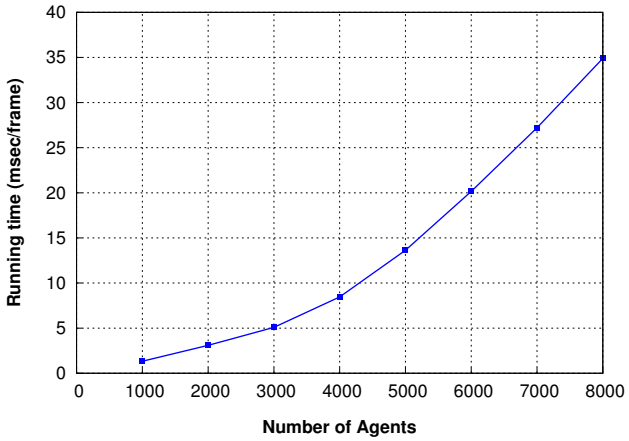


Figure 6.13: The performance of our predictive approach on the obstacle-free environment as a function of the number of agents. The reported numbers are simulation only.

6.6.3 Performance

To test the overall performance of our approach, we selected a varying number of agents and placed them randomly across an obstacle-free environment ($200\,\text{m} \times 200\,\text{m}$). Each agent had to advance toward a random goal position avoiding collisions with the other moving characters; when it had reached its destination, a new goal was chosen. We performed experiments for up to $8,000$ agents and measured the average time it takes to compute one cycle of the simulation. The results are shown in Figure 6.13.

As the figure indicates, our model can easily handle thousands of virtual characters in real-time. Even in very crowded simulations involving $8,000$ agents, the running time roughly corresponds to a frame rate of 29 fps. Such high performance rates are attributed to the force-based nature of our approach, as well as to the fact that only a limited number of neighbouring characters is considered for computing the evasive force of each agent at every step of the simulation (see Section 6.5.1).

To demonstrate the usability of our proposed collision avoidance algorithm in real-time interactive applications, we run two additional benchmark scenarios. In the first, we simulated 500 fully animated characters in a virtual park environment (Figure 6.14(a)). The park consists of 8 main gates/exits. Each character enters the park through a random gate and has to advance toward a randomly selected exit. In the second benchmark, we simulated the interactions of 200 animated pedestrians at the crosswalks of an outdoor virtual city environment (Figure 6.14(b)).

In both scenarios, to increase the realism of the generated simulations, we varied the preferred and maximum speeds among the characters, as well as the anticipation time of each character and the size of its personal space. The virtual characters were animated using a motion capture database consisting of walking and idle animations. Simulations were updated at 10 fps, whereas a parallel thread was used to render the characters and the scene at 30 fps. Our model led to smooth motions and thus, we did not have to decouple the simulation from the rendering. In addition, only a small percentage of the CPU time was dedicated to the planning of the avoidance maneuvers of the characters, yielding to real-time simulations of animated humans.

(a) Virtual park environment (b) Pedestrian interactions at crosswalks.

Figure 6.14: Simulations in virtual environments.

6.7 Conclusions

In this chapter, we presented a novel social force model for simulating the collision avoidance behaviour of interacting virtual characters. Similar to the way people behave and interact in the real world, each virtual character scans the environment to detect future collisions with other characters. It predicts how these collisions will take place and tries to resolve them in the current simulation step by making the most efficient move. As a result, the character adapts its route long in advance (like people do in real life), avoiding collisions with minimal effort.

We have experimentally shown that our system is capable of reproducing emergent phenomena that have been widely noted in the field of pedestrian and crowd simulation. In all of our experiments, the characters exhibit smooth behaviour and evade each other in a natural way. Even in crowded scenarios, the flow of the characters looks realistic and no jerky behaviour has been observed. In addition, our proposed method is very easy to implement and can be used for interactive crowd simulation of thousands of characters.

Our collision avoidance algorithm is easily parallelizable, since each virtual character plans its avoidance maneuvers independently without coordination with the other characters. In our current implementation, only one CPU core was used for motion planning. Therefore, in the future, we would like to distribute the computations of the characters' actions across multiple processors and improve the performance of our algorithm even further.

We also plan to automatically calibrate the parameters of our model and validate our approach by exploiting video and motion capture data of real pedestrian crowds. Finally, it would be interesting to extend our technique to handle panic situations and other types of interactions observed in real life, as well as integrate it into serious games and training simulators. In such applications, though, a more realistic human model is required. Currently, we approximate humans as discs having fixed radius. While this is sufficient for our example scenarios, it ignores the fact that real people maintain an elliptic personal space and may "squeeze" themselves to thread, for example, through narrow bottlenecks or dense crowds. Although such a model is mathematically more complex to handle, it will lead to more realistic character interactions and provide more accurate simulation results.

Chapter 7

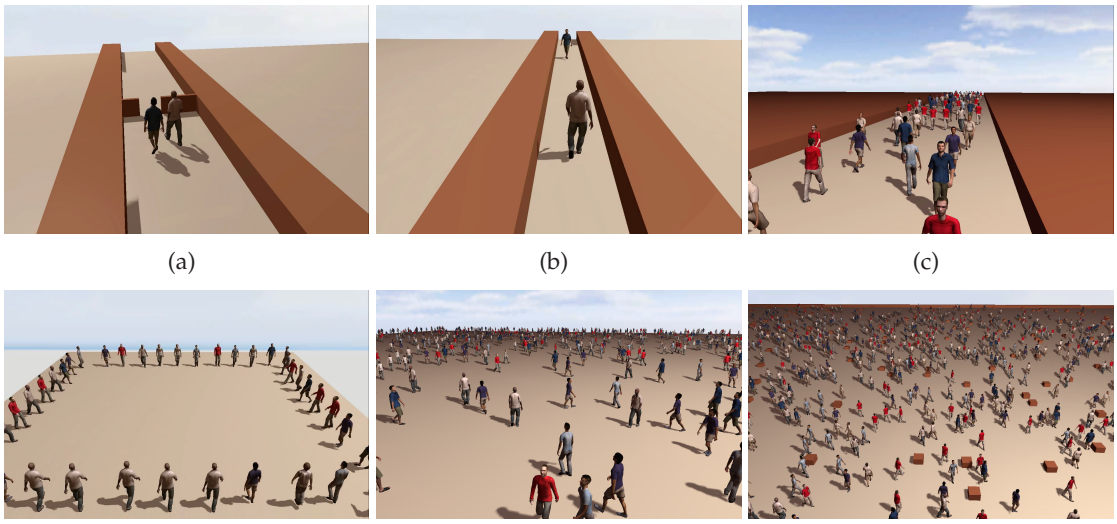
A Velocity-Based Approach for Simulating Human Collision Avoidance

In the previous chapter, we proposed a force-based model for local collision avoidance. Our method is fast, simple to implement and captures the emergence of important macroscopic phenomena allowing an efficient overall crowd motion. In the simulations created with our system, the agents solve interactions by adapting their speeds and directions of motion. Such adaptations, though, are implicitly defined based on the evasive forces exerted on the agents. Consequently, like with any other approach based on (social) forces, it is difficult to generate desired crowd movements and control the output of a simulation, especially when attempting to steer a large number of agents through the virtual world.

On the other hand, *velocity-based* methods are much more easier to regulate and have gained a lot of popularity over the past years. In these approaches, each agent locally avoids collisions with other agents and static obstacles by explicitly selecting at every step of the simulation an optimal velocity among a set of admissible velocities. State-of-the-art velocity models are based on the concept of *Velocity Obstacles* that was introduced in robotics by Fiorini and Shiller [45]. However, even though such models exhibit robust avoidance behaviour, the resulting simulations are far from realistic. Collective phenomena, like the dynamic formation of lanes, start emerging rather late and thus, the agents seem to lack anticipation. In addition, the generated motions are often characterized by oscillatory behaviour. These problems tend to be exacerbated in dense and congested environments, leading to unconvincing crowd flows.

In this chapter, we also address the problem of visually compelling and natural looking avoidance behaviour between virtual characters using a velocity model. Our approach, though, is elaborated from interactions of real humans in controlled experiments, while employing several of the concepts for local collision avoidance introduced in the previous chapter. In our new model, the agents anticipate future collisions and try to resolve them in advance by slightly adapting their orientations and/or speeds. Consequently, they avoid collisions as early as possible and with minimal effort which results in a smooth and optimal crowd flow.

We demonstrate the potential of our approach in several challenging scenarios, as can be seen in Figure 7.1. We also validate our new model based on interactions data of real humans and provide numerical and qualitative comparisons with existing approaches. In all of our simulations, the generated motions are oscillation-free, allowing us to directly infer the orientations of the agents from their underlying velocities without the need of smoothing out abrupt changes in steering directions. In addition, our method naturally exhibits emergent phenomena that have been observed in real crowds. It is relatively easy



(a) (b) (c)

(d) (e) (f)

Figure 7.1: Test-case scenarios: (a)-(b) Interactions in confined environments. (c) Agents crossing paths in a hallway. (d) Square scenario. (e) 1000 agents wandering in an obstacle-free environment. (f) 2000 agents navigating through an obstacle-filled environment.

to implement and can be used to simulate thousands of characters at interactive rates.

7.1 Related Work

Over the past years, many different approaches have been proposed for local collision avoidance in group and crowd simulations. At a broad level, they can be classified into three categories: *reactive*, *predictive* and *example-based*. Since our method is anticipative and derives from the motion analysis of experimental interactions data, this section briefly overviews existing work in the latter two categories. We also refer the reader to Section 1.2.2 and Section 6.1 for more information.

Predictive collision avoidance. In predictive models, the motions of the agents are anticipated by (linearly) extrapolating their current velocities, and then used to detect and avoid collisions in the near future. Based on this concept, Reynolds has introduced the "unaligned collision avoidance" behaviour in [182]. Similarly, Paris *et al.* [163], inspired by Feurtey's work [44], presented a velocity model that predicts potential collisions and resolves them by adapting the velocities of the virtual characters. More recently, Kapadia *et al.* [100] tackled the collision prediction problem from an egocentric perspective and used the concept of affordances from psychology to steer virtual agents without colliding with each other. Pettré *et al.* [173] also proposed an egocentric anticipative model for solving pair-interactions between virtual pedestrians based on a detailed experimental study. In their model, collisions are resolved using a combination of speed and orientation adaptations depending on the role that each pedestrian has during the interaction (passing first or giving way). An alternative approach was introduced by Ondřej *et al.* [156] that formulates the collision avoidance behaviour of virtual humans as a visual-stimuli/motor-response control law; virtual humans adapt their orientations to avoid future collisions detected from visual stimuli, whereas a deceleration strategy prevents imminent collisions.

Closely related to the aforementioned techniques is the notion of *Velocity Obstacle* (VO) introduced in [45]. For a given entity, the VO represents the set of all velocities that would result in a collision with another entity at some moment in time. Using the VO formulation, collisions can be avoided by selecting a velocity outside the VO space. Recently, van den Berg *et al.* [227] extended the VO concept to guarantee oscillation-free behaviour between interacting agents and introduced the *Reciprocal Velocity Obstacle* method (RVO). RVO randomly samples the velocity space and heuristically searches for a velocity that favours collision- and oscillation-free motions, minimizing at the same time the deviation from the agents's desired velocity. Based on the RVO formulation, Guy *et al.* [66] presented a highly-parallel algorithm for collision avoidance. Their proposed *Fast Velocity Obstacle* (FVO) provides avoidance behaviour similar to RVO, but is considerably faster since it only guarantees collision avoidance for a discrete time interval and uses a discrete optimization method to efficiently compute the new velocities of the agents. Similarly, the *Optimal Reciprocal Collision Avoidance* (ORCA) formulation allows each agent to select a reciprocally collision-avoiding velocity by solving a low dimensional linear program [226]. This approach was later extended in [65] to generate biomechanically energy-efficient and collision-free trajectories for each individual in large-scale multi-agent simulations.

Example-based methods. Example-based (or data-driven) techniques have also been explored for simulating interacting virtual characters. These approaches use example behaviours from video or motion captured data to drive the motions of the characters. To that extent, in [128], a database of human trajectories is learnt from real pedestrian crowds. During the simulations, interactions are solved by retrieving the most similar example from the database. A similar approach has been proposed by Lee *et al.* [126] aiming at reproducing realistic group behaviours in simulated crowds. Although both of the aforementioned methods can generate high fidelity motions, their applicability is limited by the size of the example databases. Therefore, situations may arise where none of the example behaviours can lead to a collision-free solution. In addition, these approaches are too computationally expensive for real-time interactive applications and are typically used for offline simulations.

Motion captured data have also been exploited by researchers in the animation and pedestrian simulation community to elaborate and fine tune the parameters of their microscopic simulation models. For example, the predictive model of Paris *et al.* [163] was calibrated from experimental motion capture data, whereas Pettré *et al.* derived and calibrated their velocity model in [173] from experimental data involving pair-interactions in controlled scenarios.

Our approach. Our method is somewhat similar in nature to the VO approaches mentioned above. We also use the velocity space to plan the avoidance maneuvers of each virtual character. However, we significantly reduce the set of admissible velocities that each character can select at every step of the simulation by taking into account how imminent potential collisions are. Note that approaches like FVO also reduce the admissible velocity domain. Our model, though, is based on observed behaviour of real people ensuring convincing avoidance behaviours.

To determine an optimal solution velocity among the admissible ones, we use a cost function similar to the ones proposed in [44, 163, 227]. Nevertheless, we also take into account the amount of *effort* that is required to perform a certain avoidance maneuver. This ensures smooth motions, allowing us to directly infer the character's orientation from the direction of its velocity vector, even for densely packed and congested simula-

tion scenarios. Note that in most velocity-based approaches, the orientation may oscillate over adjacent keyframes and thus, some sort of smoothing is required, which may lead to undesirable effects.

The recent approach of Guy *et al.* [65] also strives for energy-efficient trajectories. Their model computes a collision-free velocity for each agent that minimizes the agent's expected effort to reach to its goal. In contrast, our method focuses on the effort that the agent has to spend in order to adapt its current velocity. As a result, by favouring small velocity adaptations, collisions in our approach are resolved with minimal effort.

Our approach bears also some resemblance with the model proposed by Pettré *et al.* in [163], since we use their experimental study to elaborate our collision avoidance algorithm. Our analysis, though, focuses on the *predicted time-to-collision* between interacting participants and the deviation from their desired velocities, whereas they studied the effect that the *minimum predicted distance* has on the participants' accelerations. Additionally, their approach is mainly suited for pair-interactions. Although it has been extended to simultaneously solve multiple interactions between virtual characters, in the resulting simulations the characters seem to abruptly stop and change their orientations. In contrast, our approach can be used to resolve collisions among multiple characters in real-time, yielding to visually convincing simulations.

Finally, our technique is closely related to the recent synthetic-vision based approach of Ondřej *et al.* [156], as we also explicitly control the angular and tangential velocities of the agents. However, their model only allows the agents to decelerate in case of imminent collisions, whereas our method permits both acceleration and deceleration strategies to resolve agent interactions. Furthermore, the avoidance maneuvers in [156] are based on the notion of *time-to-interaction*, that is, on the remaining time before the distance between the agents becomes minimal. Instead, our approach employs the *time-to-collision* concept to determine the admissible velocities of the agents.

7.2 Human Locomotion and Collision Avoidance

In this section, we present some key concepts about human locomotion and human-human interactions. In the next section, we subsequently analyze publicly available motion capture data to gain a better understanding into how humans solve interactions and avoid collisions with each-other in real life.

Notion of personal space. An important concept in human interactions is the *personal space* that each individual maintains from the others in order to feel comfortable. In general, the size and the shape of the personal space can vary based on the walking speed, the gender and the age of the interacting individuals, their cultural and ethical background, as well as the crowd density [18, 69, 71, 152, 202]. According to Goffman [60], the personal space can be represented as an elongated oval, allowing more space in front of an individual than at his back and at the sides. Gérin-Lajoie *et al.* [59] have experimentally confirmed Goffman's observations and found that this area can be approximated by an elliptic curve. For additional details on the personal space notion, we also refer the reader to Section 6.2.2.

Locomotion, energy expenditure and effort efficiency. An extensive amount of work has been conducted on human and bipedal locomotion. Based on studies related to the energy expenditure in human walking, it is well accepted that individuals favour effort-efficient motions. It has been experimentally confirmed that humans adapt their motions

so that the amount of effort required to perform a step becomes minimal (see e.g. [92]); the motions of body segments coordinate and the energy is transferred back and forth from kinetic to potential, minimizing the total energy expenditure as well as the work of musculature. The criterion of minimum energy has been widely used in recent approaches in the field of robotics and computer animation that are based on optimization theory. Such techniques include the energy consumption in their objective functions aiming to generate believable and efficient motion patterns (see e.g. [187, 244]).

Closely related to the aforementioned concepts is Goffman's *law of minimal change* indicating that pedestrians try to minimize the amount and amplitude of directional changes while advancing to their destinations [60]. Similarly, through the use of experiments in virtual environments, Dalton has shown that there is a tendency for people to reduce their amount of turning effort, that is, the total angle turned [31]. Note, though, that usually there is a trade-off between effort and time efficiency. For example, an individual in a hurry is willing to spend more effort in order to reach his goal as quickly as possible. In contrast, during a leisure walk, individuals prefer to save effort and politely progress toward their goals.

Interactions and the principle of least effort. The *principle of least effort* dates back to 1949 when Zipf stated that given different possibilities for action, people select the one that they expect will require the least amount of effort [250]. Based on the least effort theory, many crowd simulation systems have been proposed over the past years. Crowd Dynamics [205], for example, exploits cellular automata to guide agents through paths of least resistance, while the recent approach of Guy *et al.* [65] simulates large-scale crowds by computing a biomechanically energy-efficient trajectory for each individual agent. However, all these approaches aim to improve the macroscopic (global) behaviour of the virtual agents, whereas our focus is on the local interactions of the individuals. Therefore, based on the principle of least effort, we hypothesize that

"an individual, upon interacting with another individual, tries to resolve potential collisions as early as possible by slightly adapting his motion".

The individual will adjust his trajectory in advance, trying to reduce the amount of interactions with the other walker. By slightly deviating from his desired walking direction[1] or speed[2], he makes his intentions known to the other. Eventually, some sort of coordination takes place and the collision is resolved with minimal effort. In the following section, we test the validity of our hypothesis by analyzing publicly available experimental data.

7.3 Experimental Analysis

The experimental dataset was collected by Pettré and his colleagues as part of the *Locanthrope* project[3] and allows us to gain more insight into the interactions between pairs of pedestrians. Thirty subjects participated in the experiment and in total 429 trials were

[1] Several empirical studies suggest that, in absence of any obstacles, pedestrians prefer to follow the most direct path to their goals (see, for example, [61]).

[2] An individual prefers to reach his goal as comfortable as possible and thus, moves with a certain preferred speed. Typically, this speed equals to $1.33\,\mathrm{m/s}$, which corresponds to the average walking speed for adults in unconstrained environments. It is also important to note that the aforementioned value also predicts the minimum energy consumption per meter walked for adults [238].

[3] `http://locanthrope.inria.fr/`

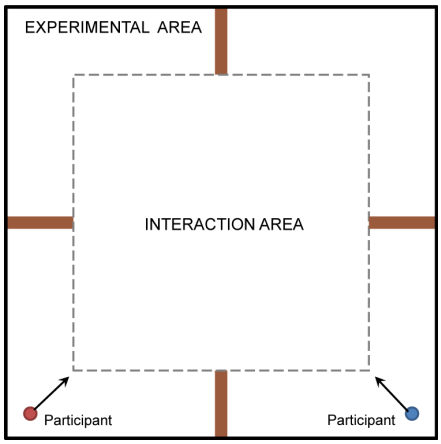


Figure 7.2: The experimental protocol in [173]. Two participants stand still at randomly selected corners of a square experimental area and are simultaneously asked to walk toward the opposite corner. The participants have synchronized trajectories and cross paths perpendicularly. Initially, due to the presence of occluding walls (brown rectangles), they are unable to see each other. Upon entering the interaction area, they can observe one another and are likely to interact in order to avoid colliding. The experimental area is $15\,\text{m}$ long, whereas the interaction square has a length of $10\,\text{m}$.

recorded using a motion capture system. Each trial consists of two participants crossing paths orthogonally while walking toward the opposite corners of a square area (see Figure 7.2). We also refer the reader to [173] for a detailed explanation of the experimental protocol.

Our analysis focuses on the effect that the time-to-collision between two interacting participants has on the *deviation* from the desired motion of each participant. In each trial, similar to [173], the *interaction* starts at time t_s when the two participants can see each other and ends at time t_f when the distance between the participants is minimal, that is

$$t_f = \underset{t \geq t_s}{\operatorname{argmin}} \|\mathbf{x}_2(t) - \mathbf{x}_1(t)\|, \tag{7.1}$$

where $\mathbf{x}_i$ defines the position of the participant's trunk [2], approximated by interpolating the two shoulder markers of the participant. During the interaction period, $t_s < t < t_f$, we estimate the future motions of the two participants by linearly extrapolating their current trajectories:

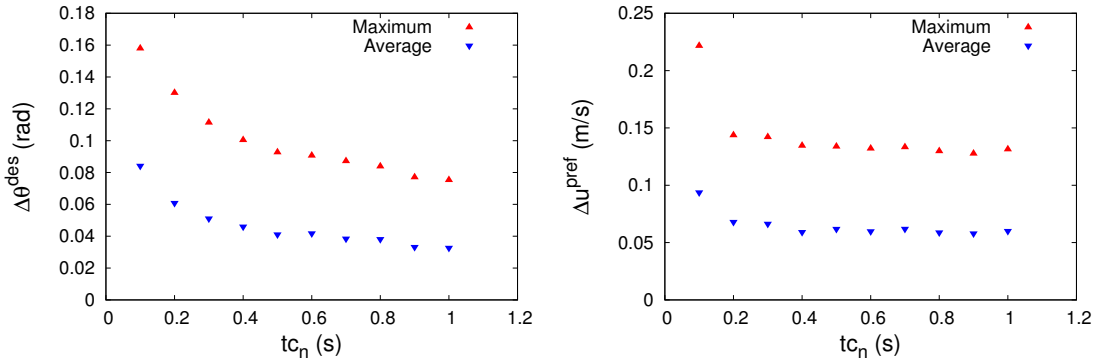
$$\mathbf{x}'_i = \mathbf{x}_i + (u - t)\mathbf{v}_i, \; i \in \{1, 2\}, \tag{7.2}$$

where the participant's velocity $\mathbf{v}_i$ is calculated using backward finite difference and $u \in (t, t_f]$ corresponds to a future time instant. Then, we can determine whether and when the participant P_1 will collide with P_2 as follows:

$$\|\mathbf{x}'_2 - \mathbf{x}'_1\| \leq r_1 + r_2, \tag{7.3}$$

where r_i denotes the radius of a participant derived from the width of the participant's shoulders, that is, from the distance between the two shoulder markers.[4] Solving the above equation, we can predict the time -if any- to collision $tc\,(t)$ between P_1 and P_2 for any time $t \in (t_s, t_f)$ belonging to the interaction period. Finally, to be able to run

[4] In our analysis, we conservatively approximate each participant as a disc of fixed radius moving on the 2D plane.



(a) (b)

Figure 7.3: (a) Maximum and average deviation from the desired orientation as a function of the normalized time-to-collision. (b) Maximum and average deviation from the preferred speed as a function of the normalized time-to-collision.

comparisons over all trials, we normalize the predicted time-to-collisions $tc\,(t)$ as follows:

$$tc_n(t) = \frac{tc\,(t)}{\max\limits_{t' \in (t_s, t_f)} \{tc\,(t')\}}. \tag{7.4}$$

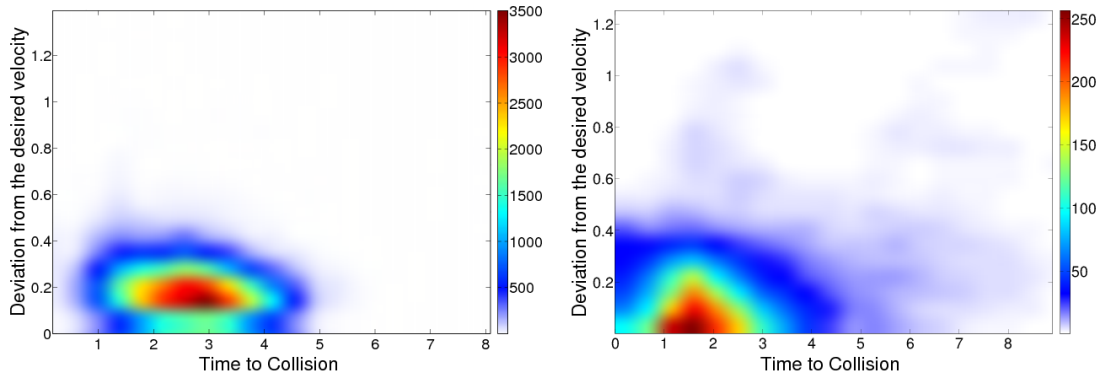
$tc_n(t)$ ranges from 0 to 1, where 0 indicates that the two participants are already colliding and 1 corresponds to the maximum time-to-collision between the two participants for the current trial.

Time-to-collision and desired orientation. For any time $t_s < t < t_f$, we define the participant's deviation from his/her desired orientation as follows:

$$\Delta\theta_i^{\text{des}}(t) = \arccos\left(\frac{\mathbf{v}_i(t)}{\|\mathbf{v}_i(t)\|} \cdot \mathbf{n}_{\theta_i^{\text{des}}}\right), \tag{7.5}$$

where $\mathbf{n}_{\theta^{\text{des}}}$ is the unit vector pointing from the current position of the participant toward his/her goal position. Figure 7.3(a) plots the participants' deviations from their desired orientations as a function of the normalized time-to-collision. We grouped the tc_n into 10 clusters of equal length and then determined the maximum and average deviation angle per cluster. As can be inferred from the figure, the maximum deviation angle is quite small for predicted collisions that will take place in the far future, $tc_n \geq 0.8$. After a small increase, it remains more or less constant for a long period, $0.4 < tc_n < 0.8$. An ANOVA test delivers a p-value of 0.093 thereby accepting the null hypothesis of zero slope coefficient at significance level $\alpha = .05$, with fitted curve $y = -0.0275x + 0.1067$. As soon as potential collisions start to become imminent, that is $tc_n \leq 0.4$, the deviation angle increases exponentially with decreasing time-to-collision reaching to a peak when tc_n tends to 0. The corresponding best fit is $y = 0.122e^{-4.489x} + 0.079$, with a coefficient of determination value of $\mathrm{R}^2 = 0.999$. Note that similar trends are also observed when looking at the evolution of the average deviation angle with respect to tc_n.

Time-to-collision and preferred speed. During the interaction period, we define the preferred speed v^{pref} of a participant by taking the average speed over this period. Then, for any time $t_s < t < t_f$, the participant's deviation from his/her preferred speed can be



(a) (b)

Figure 7.4: Density of the deviation from the desired velocity as a function of the predicted time-to-collision. (a) Participants that cross paths orthogonally. (b) Participants that have to avoid head-on collisions.

described by:

$$\Delta v_i^{\text{pref}}(t) = \Big|\, \|\mathbf{v}_i(t)\| - v_i^{\text{pref}} \,\Big|. \tag{7.6}$$

Figure 7.3(b) shows the effect that the normalized time-to-collision has on the maximum and average speed deviation of the participants. Compared to the orientation deviation, the speed deviation remains constant for a longer period of time, $tc_n \geq 0.2$. Note also the abrupt increase of the maximum deviation at $tc_n < 0.2$, which allows the participants to successfully resolve threatening collisions by refining their speed.

Discussion. In the last two paragraphs, we studied the influence that the normalized time-to-collision has on the speed and the orientation of the participants. To obtain a clear overview of how participants solve interactions, we also cumulated all pairs of predicted collision times and corresponding deviation velocities $\Delta\mathbf{v}^{\text{des}}$ for all trials. For any time $t_s < t < t_f$, the participant's deviation from his/her desired velocity is defined as follows:

$$\Delta\mathbf{v}_i^{\text{des}}(t) = \|\mathbf{v}_i(t) - \mathbf{v}_i^{\text{des}}\|, \tag{7.7}$$

where the desired velocity is determined by the participant's preferred speed and orientation, that is $\mathbf{v}_i^{\text{des}} = v_i^{\text{pref}}\mathbf{n}_{\theta_i^{\text{des}}}$. Figure 7.4(a) shows the density plot of the corresponding bivariate data. As can be seen, in most of the trials the majority of the participants prefer to solve interactions in advance, that is when $2.0 \leq tc \leq 4.0$, favouring small changes in their velocities. Note also that very rarely participants have to adapt their motions at imminent collision times, that is when tc is close to 0, which shows the ability of people to efficiently predict and avoid collisions.

In conclusion, our analysis confirms our hypothesis that individuals resolve collisions long in advance by slightly adjusting their motions. In our current study, we focus on participants that have orthogonally intersecting paths. Another challenging case, though, is when interacting participants have to avoid head-on collisions. For that reason, we have recently conducted an experimental study similar to the one proposed by Pettré *et al.* We refer the reader to [222] for details regarding the experimental setup. Preliminary analysis of the corresponding interactions data seems to support our main hypothesis (see Figure 7.4(b)). Note, though, that due to the fact that the participants have exactly opposite desired directions, some miscommunication on how they pass each other might

be detected. Thus, as can be observed in Figure 7.4(b), there are cases that collisions are resolved at the last moment.

7.4 Multi-Agent Collision Avoidance Model

In this section, we present our velocity-based algorithm for local collision avoidance. Our approach is derived from our experimental observations and can be integrated into existing approaches for crowd simulation.

Problem formulation. In our problem setting, we are given a virtual environment in which n heterogeneous agents A_i $(1 \leq i \leq n)$ have to navigate without colliding with the environment and with each other. For simplicity, we assume that each agent A_i moves on a 2D plane and is modeled as a disc with radius r_i. At a fixed time t, the agent A_i is at position $\mathbf{x}_i$, defined by the center of the disc, and moves with velocity $\mathbf{v}_i$. The motion of A_i is limited by a maximum speed $v_i^{\max}$. Furthermore, at every time step of the simulation, A_i has a desired velocity $\mathbf{v}_i^{\text{des}}$ that is directed toward the agent's goal position $\mathbf{g}_i$ and has a magnitude equal to the agent's preferred speed v_i^{pref}. In order to feel comfortable, the agent also maintains a certain psychophysical distance ρ_i from other agents and static obstacles. This distance defines the personal space of A_i. For efficiency reasons, we model this space as a disc centered at $\mathbf{x}_i$ and having radius $\rho_i > r_i$. This led to avoidance behaviour similar to the one observed in our experimental analysis and hence, the use of an elliptical personal space was not necessary. Note that, to simplify the multi-agent navigation problem, we do not take the orientation of the agents into account. Instead, we assume that each agent can only translate in the planar workspace, and hence, the agent's configuration space is also the 2D Euclidean plane. However, to obtain visually compelling simulations, we deduce the orientation of each agent from the heading of the agent's motion.

7.4.1 Basic Approach

At each simulation cycle, an agent A_i solves interactions with the other agents in three successive steps. First, it computes the set of nearby agents with which it is on collision course. Then, it determines the sets of orientations and speeds that it can select to resolve the collisions and finally, it retrieves a new velocity from these sets. We now describe the three steps in more detail:

Step 1 - Retrieve the set of colliding agents. In the first step of our algorithm, we compute the set CA_i of first N agents that are on collision course with A_i. We first extrapolate the future position of A_i based on its desired velocity $\mathbf{v}_i^{\text{des}}$. Similarly, we predict the future motions of all the nearest agents that A_i can see by linearly extrapolating their current velocities (A_i can only estimate the actual velocities of the other agents and not their desired ones). The viewing area of A_i is modeled as a cone defined by A_i's desired direction of motion and an effective angle of sight of $200°$, corresponding to the field of view of a typical human.

Based on the extrapolated trajectories, we can now determine whether the agent A_i will collide with another agent A_j. We assume that a collision occurs when A_j lies inside or touches the personal space of A_i, resulting in the following equation:

$$\|(\mathbf{x}_j + \mathbf{v}_j t) - (\mathbf{x}_i + \mathbf{v}_i^{\text{des}} t)\| \leq r_j + \rho_i. \tag{7.8}$$

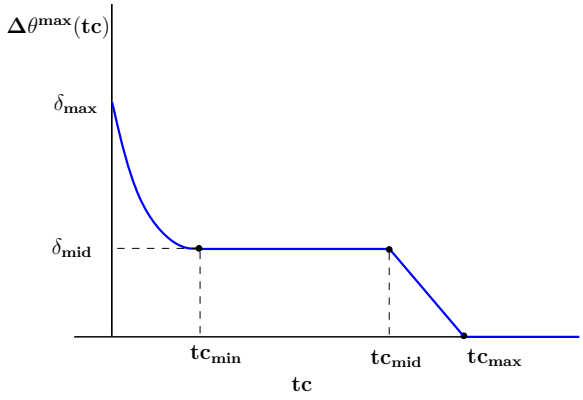


Figure 7.5: Maximum orientation deviation for an agent, as a function of the predicted time to collision.

Solving Equation 7.8 for t, we can deduct the possible collision time tc_{ij} between A_i and A_j. If $tc_{ij} \geq 0$, the agent A_j is inserted into the set of the agents that are on collision course with A_i. Consequently, the set CA_i is defined as:

$$CA_i = \bigcup_{j \neq i, tc_{ij} \geq 0} \{A_j, tc_{ij}\}. \tag{7.9}$$

We sort this set in order of increasing collision time and keep the first N agents. Experiments have indicated that this number can be kept small (default value is $N = 5$). This not only reduces the running time of our algorithm, but also reflects natural human behaviour. In real-life, an individual takes into account a limited number of other walkers, usually those that are on collision course with him/her in the coming short time.

Step 2 - Determine the sets of admissible orientations and speeds. In the second step of the algorithm, we retrieve the sets of candidate orientations $O_i \in \mathbb{R}^2$ and speeds $U_i \in \mathbb{R}$ that the agent A_i can select in order to resolve the collisions with the agents belonging to the set CA_i. First, we retrieve from CA_i the collision time tc with the most threatening agent, that is the front agent in the (ordered) set CA_i. Then, based on our analysis in Section 7.3, we compute the maximum angle that the agent A_i can deviate from its desired velocity. The maximum deviation angle $\Delta\theta_i^{\max}$ is approximated by a piecewise function (see Figure 7.5) as follows:

$$\Delta\theta_i^{\max}(tc) = \begin{cases} \frac{\delta_{\max} - \delta_{\mathrm{mid}}}{e^{tc}} + \delta_{\mathrm{mid}}, & \text{if } 0 \leq tc < tc_{\min} \\ \delta_{\mathrm{mid}}, & \text{if } tc_{\min} \leq tc < tc_{\mathrm{mid}} \\ \delta_{\mathrm{mid}} \frac{tc_{\max} - tc}{tc_{\max} - tc_{\mathrm{mid}}}, & \text{if } tc_{\mathrm{mid}} \leq tc < tc_{\max} \\ 0, & \text{if } tc_{\max} \leq tc. \end{cases} \tag{7.10}$$

The parameter $tc_{\max}$ defines the maximum time that A_i anticipates a collision. Note that, in the experimental analysis, the maximum collision time averaged over all experiments is 4.2 s. However, in our simulations, each agent has to simultaneously solve interactions with multiple agents. Thus, a higher anticipation time is used to ensure smooth avoidance behaviour (its default value is set to $tc_{\max} = 8$ s). The threshold tc_{mid} regulates the start of the constant part of the function, whereas $tc_{\min}$ defines collisions that are imminent to A_i leading to higher deviation angles (default values used in our experiments are

$tc_{\text{mid}} = 6\,\text{s}$ and $tc_{\text{min}} = 2.5\,\text{s}$, respectively). The parameter δ_{max} determines the maximum admissible angle that A_i can deviate from its desired direction of motion. In our simulations, we assume that virtual characters cannot backtrack and thus, $\delta_{\text{max}} = \pi/2$. Finally, the parameter δ_{mid} defines the deviation angle during the constant interval of the function (we set the default value to $\delta_{\text{mid}} = \pi/6$).

Having retrieved the maximum deviation angle $\Delta\theta_i^{\text{max}}$, we determine the agent's admissible orientation domain O_i as follows:

$$O_i = \{\ \mathbf{n}_\theta \mid \theta \in [\theta_i^{\text{des}} - \Delta\theta_i^{\text{max}},\ \theta_i^{\text{des}} + \Delta\theta_i^{\text{max}}]\ \}, \tag{7.11}$$

where $\mathbf{n}_\theta = [\cos\theta, \sin\theta]^T$ is the unit vector pointing in direction θ and θ_i^{des} is the direction derived from A_i's desired velocity.

A similar approach is also used to determine the admissible speed domain U_i of the agent A_i. Based on the speed deviation plot, shown in Figure 7.3(b), U_i is approximated as follows:

$$U_i = \begin{cases} v \mid v \in [0, v_i^{\text{max}}], & \text{if } 0 \leq tc \leq tc_{\text{min}} \\ v \mid v \in [v_i^{\text{pref}} - \Delta v_i^{\text{max}},\ v_i^{\text{pref}} + \Delta v_i^{\text{max}}], & \text{if } tc_{\text{min}} < tc \leq tc_{\text{max}} \\ v_i^{\text{pref}}, & \text{if } tc_{\text{max}} < tc, \end{cases} \tag{7.12}$$

where for the maximum speed deviation Δv_i^{max} holds that $\Delta v_i^{\text{max}} \leq \min(v_i^{\text{max}} - v_i^{\text{pref}}, v_i^{\text{pref}})$. In our simulations, we set the default value of Δv_i^{max} to $0.4\,\text{m/s}$, whereas the default value of v_i^{max} is set to $2.4\,\text{m/s}$.

Finally, we should point out that a piecewise constant function is used for the admissible speed domain as compared to the C^0-continuous function employed for the admissible orientations (see Equation 7.10). The main reason is that real humans are capable of high accelerations and decelerations and thus, we try to capture this type of behaviour. In addition, the low weight of the speed cost term in our objective function is sufficient to provide smooth speed transitions (see the next step).

Step 3 - Select an optimal solution velocity. In the final step of our collision avoidance algorithm, we compute an optimal solution velocity for the agent A_i. First, we deduce the set of feasible avoidance velocities FAV_i from the agent's admissible orientation and speed domains as follows:

$$FAV_i = \{v\,\mathbf{n}_\theta \mid v \in U_i \wedge\ \mathbf{n}_\theta \in O_i\}. \tag{7.13}$$

In practice, an infinite number of feasible velocities exist. Thus, we restrict the O_i domain to a discrete set of orientation samples (the default size of the discretization step is set to 0.078 radians). Similarly, we discretize the U_i domain into a set of adjacent speed samples (the default distance between adjacent samples is set to 0.1).

Next, we select the agent's new velocity $\mathbf{v}_i^{\text{new}}$ from the set of feasible velocities. Among the candidate velocities $\mathbf{v}^{\text{cand}} \in FAV_i$, we retain the solution minimizing the amount of effort that the agent needs to adapt its current velocity to the new candidate velocity, the risk of collisions with the other agents and the deviation from the agent's desired velocity:

$$\mathbf{v}_i^{\text{new}} = \underset{\mathbf{v}^{\text{cand}} \in FAV_i}{\arg\min}\ \{\ \overbrace{\alpha(1 - \frac{cos\,(\Delta\phi)}{2})\ +\ \beta\frac{\big|\ \|\mathbf{v}^{\text{cand}}\| - \|\mathbf{v}_i\|\ \big|}{v^{\text{max}}}}^{\text{Effort}} \underbrace{+\gamma\,\frac{\|\mathbf{v}^{\text{cand}} - \mathbf{v}_i^{\text{des}}\|}{2v_i^{\text{max}}}}_{\text{Deviation}}\ +\ \underbrace{\delta\,\frac{tc_{\text{max}} - tc}{tc_{\text{max}}}}_{\text{Collisions}}\ \} \tag{7.14}$$

where $\Delta\phi$ defines the angle between the agent's current velocity vector and $\mathbf{v}^{\text{cand}}$. Consequently, in our cost function, the effort expended by the agent is approximated by taking into account changes both in the speed and the direction of the agent. Regarding the collision cost, tc denotes the minimum predicted collision time between the agent A_i and the agents in the set CA_i, assuming that A_i selects a velocity $\mathbf{v}^{\text{cand}}$; note that tc is upper bounded by $tc_{\max}$. The constants $\alpha, \beta, \gamma, \delta$ define the weights of the specific cost terms and can vary among the agents to simulate a wide variety of avoidance behaviours. We set the default values to: $\alpha = 1$, $\beta = 0.05$, $\gamma = 1$, $\delta = 1$. These parameters generated the best agreement with our experimental observations, leading at the same time to visually convincing simulations.[5]

Having retrieved the new velocity $\mathbf{v}_i^{\text{new}}$, the agent advances to its new position $\mathbf{x}^{new}$ as follows:

$$\mathbf{x}_i^{new} = \mathbf{x}_i + \mathbf{v}_i^{\text{new}} \Delta t, \tag{7.15}$$

where Δt is the time step of the simulation. Note that if the agent is also subject to dynamic constraints (e.g. maximum acceleration), we can either take them intrinsically into account upon constructing the set of feasible velocities, or restrict the newly derived velocity based on the given constraints. During each simulation cycle, we also update the orientation of the agent. The effort term in our cost function does not allow the agent to abruptly change its direction, ensuring smooth avoidance motions. Thus, the new orientation θ_i^{new} of the agent is directly inferred from the solution velocity, that is:

$$\theta_i^{new} = \arctan(\mathbf{v}_i^{\text{new}}) \,. \tag{7.16}$$

7.4.2 Resolve Threatening Collisions

In most cases, the agent A_i solves interactions in advance, smoothly evading the other agents. It is possible, though, for A_i to collide with another agent A_j, especially in very crowded environments. A collision occurs when A_j invades the personal space of A_i, which does not necessarily mean that A_j penetrates A_i. As a result, the agent A_i must select a new velocity that will resolve the collision as fast as possible. For that reason, we drop the effort cost term from our optimization function and modify Equation 7.14 as follows:

$$\mathbf{v}_i^{\text{new}} = \underset{\mathbf{v}^{\text{cand}} \in FAV_i}{\arg\min} \{\gamma \frac{\|\mathbf{v}^{\text{cand}}\|}{v_i^{\max}} + \delta \frac{tc}{tc_{\max}}\}, \tag{7.17}$$

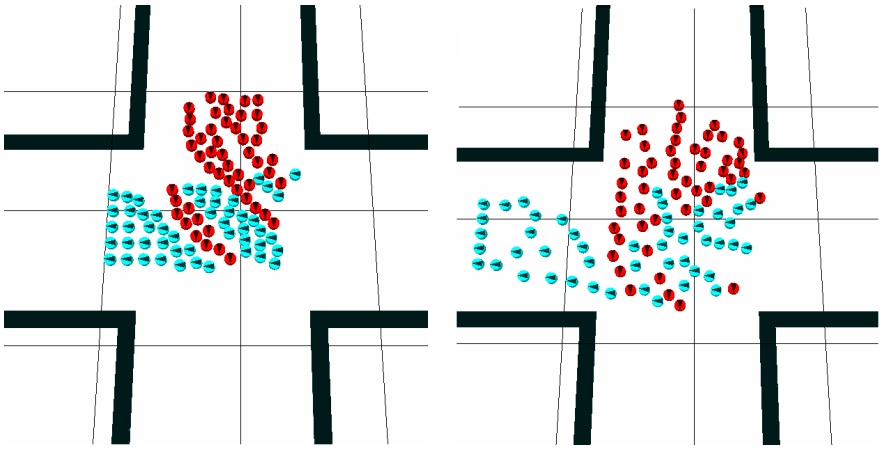
where now tc denotes the time that is needed to escape the collision, that is, the time that A_i needs to exit from the disc $B(\mathbf{x}_j, r_j + r_i + \mu_i)$ centered at agent A_j and having radius equals to the sum of the radius of A_j and the personal space of A_i. Note also that the desired velocity in the deviation cost term is set to zero.

7.4.3 Avoiding Static Obstacles

An agent A_i has also to avoid collisions with the static obstacles of the environment. In our framework, these obstacles are modeled as a collection of line segments. Then, collisions are resolved by following an approach similar to the one described above.

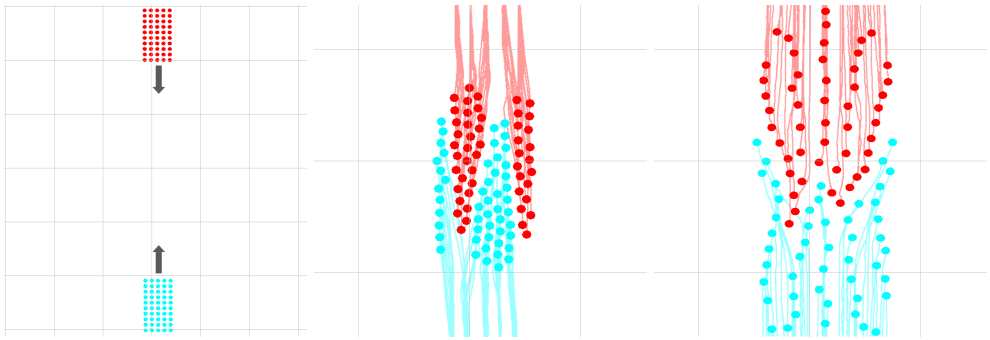
We first retrieve the nearest obstacle neighbours of A_i. We then define the set of admissible speeds and orientations for the agent, based on the minimal time-to-collision between the agent and its obstacle neighbours. These sets are merged to the admissible

[5]The default values were estimated from the experimental data by minimizing the difference between observed and simulated trajectories (based on the Manhattan distance).



(a) Our model (b) RVO

Figure 7.6: Interactions at a crossing. Using our approach, stripe formations emerge that result in an efficient and smooth flow compared to RVO.



(a) Group-swap (b) Our model (c) RVO

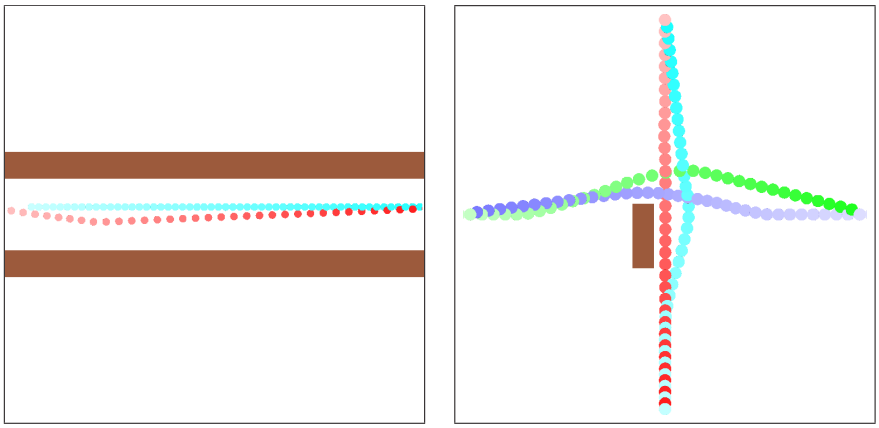
Figure 7.7: (a) Two groups of agents heading in opposite directions swap their positions. (b) In our velocity-based method, distinct lanes emerge allowing an efficient overall motion. (c) In RVO, the lanes start forming rather late leading to time-consuming and inefficient avoidance strategies.

speed and orientation domains defined in the second step of Section 7.4.1. In our simulations, we assume that the time-to-collision has the same effect on agent-agent interactions and on static obstacle avoidance. Thus, in practice, the admissible speed and orientation domains of A_i are defined by the agent's most threatening neighbour, whether this is an obstacle or an agent. Finally, a new velocity $\mathbf{v}_i^{\text{new}}$ is computed as described in the third step of Section 7.4.1.

We note that our algorithm allows the agent to avoid collisions with the obstacles but not to move around them. Therefore, some global path planning approach should be used to ensure that the agent cannot get stuck in local minima and can intelligently plan its path around the obstacles (see, for example, the IRM method introduced in Chapter 3 or the global navigation techniques proposed in [57, 118, 172]).

7.5 Results

We have implemented our collision avoidance algorithm to experimentally test its applicability in real-time applications and validate the quality of the generated motions. In all



(a) Overtake (b) Confusion-obstacle

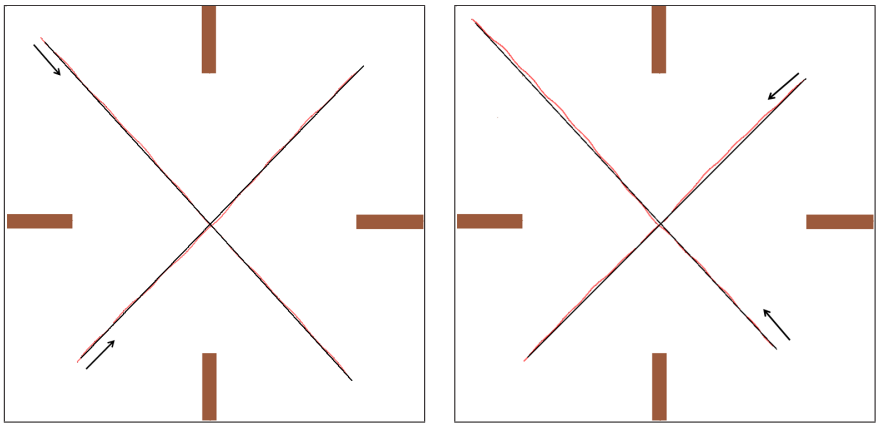
Figure 7.8: Time-lapse agent positions in two interaction scenarios. Each agent is represented as a coloured disc and its trajectory progresses from translucent to opaque colours. (a) A faster moving agent overtakes a slower moving agent in a hallway. (b) Four agents cross paths while avoiding a static obstacle in their way.

of our experiments, we set the time step of the simulation to $\Delta t = 0.2\,\mathrm{s}$, the radius of each agent to $r = 0.5\,\mathrm{m}$ and the size of its personal space to $\rho = 1\,\mathrm{m}$. To approximate the behaviour of a typical human, the preferred speed of each agent was set to $v^{\mathrm{pref}} = 1.3\,\mathrm{m/s}$ and its maximum speed to $v^{\mathrm{max}} = 2.4\,\mathrm{m/s}$. We refer the reader to Section 7.4 for all the other parameter values used in our experiments. To efficiently compute for each agent its set of colliding agents, a spatial data structure for answering nearest neighbour queries was also implemented, as the one described in Section 6.5. During our simulations, we set the cut-off distance for the nearest neighbour queries to $10\,\mathrm{m}$.

7.5.1 Quality Evaluation

We evaluated our approach against a wide range of test-case scenarios, as proposed in the *Steerbench* framework of Singh *et al.* [197]. These scenarios range from simple interactions between pairs of agents to more challenging and large-scale cases:

- *Doorway*: Two agents, travelling in the same direction, have to pass through a narrow door to reach the other side of a hallway (Figure 7.1(a)).
- *Squeeze*: Two agents have to avoid a head-on collision while walking through a narrow corridor (Figure 7.1(b)).
- *Overtake*: An agent moves down a hallway and encounters a slower moving agent in its front (Figure 7.8(a)).
- *Confusion-obstacle*: Four agents, travelling in opposite directions, cross paths, avoiding at the same time collisions with a static obstacle (Figure 7.8(b)).
- *Hallway*: 150 agents cross paths in a hallway while walking from opposite directions (Figure 7.1(c)).
- *Crossing*: Two groups of 50 agents, having perpendicular trajectories, meet at the intersection of a crossing (Figure 7.6).
- *Group-swap*: Two groups of 50 agents travel in opposite directions and have to exchange their positions (Figure 7.7(a)).



(a) (b)

Figure 7.9: Comparison between real (light lines) and simulated (dark lines) trajectories.

- *Square*: 40 agents are placed along the perimeter of a square and have to walk toward their antipodal positions (Figure 7.1(d)).
- *Random*: 1000 agents with random goals navigating through an environment void of obstacles (Figure 7.1(e)).
- *Forest*: 2000 agents wandering through an environment filled with small-sized obstacles (Figure 7.1(f)).

Visualization. The virtual characters in our simulations were animated with a simple motion capture database consisting of walking and idle animations. The speed of the animation was scaled to the speed of the characters, whereas animation blending was used to transition from one animation to another. Note that motions artifacts, such as foot sliding, were due to our simplified animation technique and not because of our steering algorithm.

Realism. In all of our scenarios, the agents smoothly evade collisions with other agents and static obstacles, whereas the generated motions in the resulting simulations are oscillation-free. In addition, our approach exhibits emergent phenomena that have been observed in real crowds, such as the dynamic formation of lanes (see, e.g., the *Hallway* scenario), queuing behaviour (*Oncoming-groups* and *Square* scenarios), formation of diagonal stripes in crossing flows (*Crossing* scenario), as well as the emergence of slowing down and stopping behaviour to efficiently resolve imminent collisions (see, e.g., the *Random* and *Forest* scenarios). Natural, human-like, behaviour is also captured by our system. In the *Doorway* scenario, for example, one of the two agents slows down and gently allows the other one to pass through the door first. After the passing, though, the two agents will keep moving behind each other instead of reforming to walk side-by-side. To alleviate this, we have extended our method to take into account group formations (see Chapter 8 for more details).

Comparisons. Using our model, we reproduced several interaction examples from the experimental dataset of Pettré *et al.* In all of these examples, the simulated agents were able to follow the trajectories of real humans very closely. Figure 7.9(a) shows a comparison between a real and a simulated interaction. In this case, at any time step, the

Scenarios		Time		Smoothness		Total Accel.		Total Deg. Turned	
		Mean	Stdev	Mean	Stdev	Mean	Stdev	Mean	Stdev
Crossing	Our Model	39.41	1.06	0.06	0.16	135.71	61.05	82.25	64.59
	RVO Model	48.28	6.25	3.56	2.3	464.56	130.65	311.84	142.59
Group-swap	Our Model	39.87	0.84	0.08	0.11	190.64	46.86	44.90	16.10
	RVO Model	47.81	5.06	4.07	2.62	528.55	68.86	348.24	104.12

Table 7.1: Statistics for the Crossing and Group-swap scenarios. The reported time, smoothness, total acceleration and total degrees turned are measured in s, $(rad/m)^2$, m/s^2 and deg respectively.

simulated trajectories deviate from the real ones by an average of only 0.111 m. Another comparison is depicted in Figure 7.9(b). Although the initial conditions vary from the previous example, the paths taken by the real and virtual humans are still very similar (the average difference is 0.209 m).

We have also run comparisons with van den Berg's RVO method [227, 230] (using the RVO sampling library implementation [186]). We chose the RVO because of its increased popularity among the methods that are based on the VO formulation, and its many existing variants. The main difference between our approach and VO methods is that in the latter, at every simulation step, each agent tries to find an optimal collision-free velocity. Consequently, in rather confined and crowded environments, the agent may not be able to find such a velocity and thus, the only solution would be to abruptly adjust its direction or change its speed and stop. In contrast, our approach (by reducing the set of admissible velocities) favours small changes in the velocity of each agent, even though such changes may lead to a collision in the (far) future. Assuming that the other agents will also exhibit similar behaviour, collisions are resolved in advance with minimal effort resulting in a smoother, more optimal flow. We believe that such an approach, albeit not exempt from failure, better captures the avoidance behaviour of real humans.

We also quantitatively compared our technique with RVO. For that reason, a number of quantitative quality metrics have been devised to objectively evaluate the steering behaviours of the virtual agents. In particular, we computed the total time that an agent needs in order to reach its goal. Furthermore, we used the sum total of squared curvature samples along an agent's path as a measure of the smoothness of such a path [87]. Since we strive for effort-efficient movements, we also computed the total acceleration of each agent [197], approximated as the sum of its instantaneous accelerations. To explicitly indicate the amount of turning effort spent by the agent, we also measured the total degrees that an agent had to turn while advancing toward its destination.

Table 7.1 reports the corresponding statistics for the crossing (Figure 7.6) and the group-swap (Figure 7.7) scenarios. Separate Student's t-tests at a significance level of 5% were performed afterwards to determine how significant the results were. The analysis showed that our approach leads to time-efficient motions, allowing the agents to follow much more smoother paths than the agents in RVO (the differences in time and smoothness are statistically significant in both scenarios, $p < 0.001$). In addition, using our approach, the agents have to spend less effort to avoid potential collisions and thus, the total acceleration is significantly lower compared to RVO ($p < 0.001$). The same applies to the turning effort of the agents ($p < 0.001$).

Consequently, using our approach, interactions are solved more efficiently allowing the agents to spend considerably less time and effort in order to reach their goals and follow less curved paths as compared to the RVO method. In the future, we would also like to quantitatively compare our technique with the new release of the RVO library that

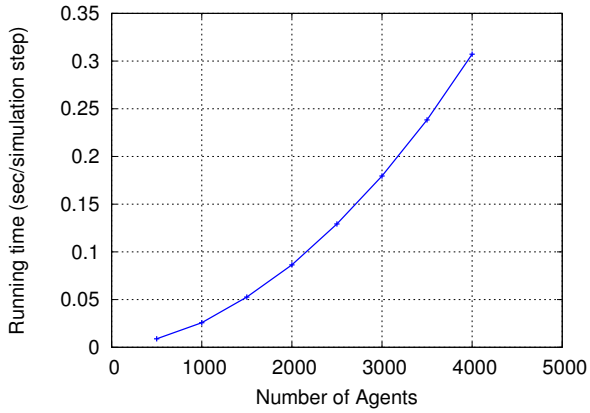


Figure 7.10: The performance of our algorithm on the forest scenario as a function of the number of agents. The reported numbers are simulation only.

is based on the optimal reciprocal collision avoidance (ORCA) formulation [226], as well as with the recent approach of Ondřej *et al.* [156] that formulates the collision avoidance behaviour of virtual humans using a combination of visual stimuli and motor response laws.

7.5.2 Performance Analysis

Besides the quality, we are also interested in the performance of our proposed approach. To test its usability in real-time applications, we selected a varying number of agents and placed them randomly across the *Forest* environment that measures $200\,\mathrm{m} \times 200\,\mathrm{m}$ (see Figure 7.1(f)). Each agent had to advance toward a random goal position avoiding collisions with the obstacles and the other moving agents; when it had reached its destination, a new goal was chosen.

Figure 7.10 highlights the performance of our approach for up to $4,000$ agents on a 2.4 GHz Core2 Duo CPU. Note that our current implementation is unoptimized and uses only one core for planning the avoidance maneuvers of the agents. As the figure indicates, even for $3,000$ agents, it takes $179\,\mathrm{ms}$ per simulation step to compute the three steps of our collision avoidance algorithm. Since, in our system, the velocities of the agents were updated at 5 fps, it is clear that our approach can simulate thousands of virtual characters at real-time rates.

7.6 Conclusions

In this chapter, we proposed a velocity-based method for solving interactions among virtual agents. Based on observed behaviour of real people, we hypothesize that each agent tries to resolve collisions in advance by slightly adapting its desired walking direction and/or speed. Consequently, in our simulations, the virtual characters exhibit smooth and convincing motions, avoiding collisions with minimal effort.

We have experimentally confirmed our hypothesis on pairs of interacting participants by studying the effect that the time-to-collision has on the desired velocities of the participants. A future objective is to analyze motion captured data and focus on the lateral displacements of the participants. This will allow us to gain more insight into the avoidance behaviour of individuals and extend our model accordingly, since, in pedestrian

interactions and particularly during head-on encounters, real humans also resolve collisions by sidestepping. We would also like to conduct an experimental study in order to investigate the behaviour of individuals when they have to avoid collisions with static obstacles. In our model, each agent avoids in a similar way both static obstacles and other agents. However, we believe that in real life people react to obstacles much earlier and resolve the potential collisions by mainly adapting their orientations.

Another important direction for future work is to determine the parameters that affect an individual's desired velocity. In our current work, we assume that the desired speed of an agent remains constant during the entire simulation. Nevertheless, parameters such as the crowd density can have a significant effect on the velocity leading to different avoidance behaviours. We would also like to adaptively sample the agent's admissible velocity domain with smaller bins near the current orientation/speed of the agent and larger ones for bigger deviations, as compared to the uniform sampling scheme employed in our current framework. Finally, we aim to combine our approach with a more sophisticated animation technique [115, 223] in order to obtain higher quality animations.

Chapter 8

Simulating and Evaluating the Local Behaviour of Small Pedestrian Groups

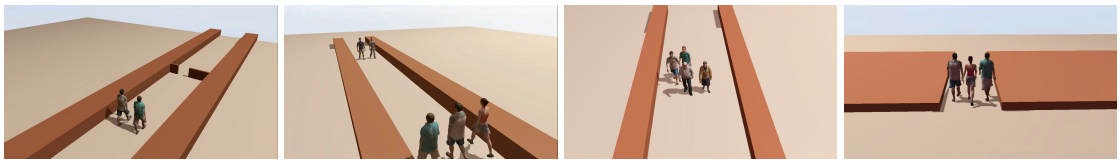
While recent advancements in local methods have significantly improved the collision avoidance behaviour of virtual characters (see, e.g., Chapters 6 and 7), the majority of existing studies treat crowds as a collection of individual agents. However, in real-life, most of the pedestrians do not walk alone, but in small groups consisting mainly of two to three members, such as couples and friends going for shopping or families walking together [27, 148].

Over the past few decades, a number of approaches have been proposed for simulating group motions. Nevertheless, most of these methods focus on large groups of virtual characters or on how such groups are formed and not on the dynamics of small groups and on how group members interact and behave within a crowd. In addition, approaches based on flocking rules, as well as leader-follower models are mainly applicable to simulate the collective behaviour of large herds or flocks. The requirements, though, for guiding the motion of small pedestrian groups are different; friends or couples prefer to walk next to each other, rather than following a leader.

More recently, several approaches have tried to address the issue of realistic behaviour of small groups of virtual humans based on empirical observations of real crowds. These methods are able to capture the macroscopic behaviour of small groups, generating formations similar to the ones observed in real pedestrian groups. The problem, though, is that, in the resulting simulations, the group members lack anticipation and prediction, which sometimes leads to unrealistic microscopic behaviours, such as oscillations or backward motions. Additionally, due to the fact that such methods are mainly based on rules or social forces, they often require careful parameter tuning to generate desired group movements.

Therefore, in this chapter, we extend the velocity-based approach introduced in the previous chapter and present a novel approach to simulate the walking behaviour of small groups of characters. The focus of our work is on the local behaviour of such groups, that is, on how group members interact with each other, with other groups and individual agents. Our proposed model is elaborated from recent empirical studies regarding the spatial organization of pedestrian groups [148] and is complementary to existing methods for solving interactions between virtual characters.

In contrast to prior approaches, we use the velocity space to plan the avoidance maneuvers of each group, striving to maintain a configuration that facilitates the social interactions between the group members. The final motion of each individual member is then computed by an underlying agent-based algorithm. We demonstrate the potential and flexibility of our approach against a wide range of test case scenarios, as can be seen in



(a) (b) (c) (d)

Figure 8.1: Example test-case scenarios: (a) A group has to adapt its formation to pass through a doorway, (b) Group interactions in a confined environment, (c) A faster group overtakes a slower moving group, (d) A group of three agents walking through a narrow corridor.

Figure 8.1. In all of our simulations, the groups exhibited convincing behaviour, smoothly avoiding collisions with other groups and individuals. We show that even in challenging scenarios, the groups safely navigate toward their goals by dynamically adapting their formations, just like groups of pedestrians do in real-life. A quantitative evaluation of our model is also presented that allows us to objectively assess the steering quality of the simulated groups and verify the emergence of empirically observed walking patterns.

8.1 Related Work

The most common way to model the locomotion of human crowds is with agent-based methods, in which each agent plans individually its own actions. In such approaches, global path planning and local collision avoidance are typically decoupled. We refer the reader to [121] for an extensive literature on global navigation techniques. Regarding the microscopic (local) behaviour of individual agents, numerous models have been proposed, including force-based approaches [76,147,165], behavioural models [53,193], synthetic vision techniques [156] and variants of velocity-based methods [66,163,173,227].

Agent-based modeling has also been used in the virtual environments community to simulate the behaviour of groups of virtual entities. The work of Reynolds on boids has been influential in this field [181]. Reynolds used simple local rules to create visually compelling flocks of birds and schools of fishes. Later he extended his model to include additional steering behaviours for autonomous agents [182]. Since his original work, many interesting models have been introduced for controlling group motions. Loscos *et al.* [136] presented a leader-follower model in which the leader decides about the motion of the entire group and the rest of the group members follow. Musse and Thalmann [150] defined a rule-based model that allows virtual humans to switch groups based on sociological factors. Brogan and Hodgins [16] accounted for motion dynamics while simulating groups of humanoid characters. Braun *et al.* [14] expanded Helbing's social force model and used attractive forces to form groups of pedestrians, whereas Qiu and Hu [177] used behavioural approaches to model different group structures in pedestrian crowds. More recently, Peters and Ennis [169] proposed a rule-based model to simulate plausible behaviours of small groups consisting of up to four individuals based on observations from real crowds. Similarly, Moussaïd *et al.* [148] have conducted a series of studies to gain more insight into the organization of pedestrian groups in urban environments and introduced a force model that accounts for social interactions among people walking in groups.

Agent-based group models have also been extensively studied in robotics. In [4], for example, Arkin used a schema-based approach to control a team of mobile robots that do not communicate with each other and have no explicit leaders. Similarly, Mataric

(a) Line-Abreast (b) V-like (c) River-like

Figure 8.2: Group formations according to [148].

[139, 140] showed that simple behaviours can be combined to produce emerging flocking behaviour in a group of wheeled robots without explicit communication. Maintaining and coordinating group formations of multiple robots has also received a lot of attention and several local methods have been proposed, including behavioural-based approaches [5, 124], leader-follower models [34, 234], virtual structure techniques [127, 132] and potential field methods [6].

At the global level, Kamphuis and Overmars [99] developed a method for planning the motions of coherent groups inside corridors, while Bayazit *et al.* [9] combined flocking techniques with probabilistic roadmaps to guide the flock members toward their goals. In the robotics community, centralized planners have also been exploited to compute the simultaneous motion of multiple units [133]. Nevertheless, the running time of such approaches grows exponentially with the number of robots. An alternative approach is based on continuum dynamics which attempts to directly guide the global behaviour of large homogeneous groups using continuous density fields [218].

Prior work in graphics and animation community has also focused on the synthesis of realistic group motions. Kwon *et al.* [114] proposed a technique that allows the user to interactively edit existing group motions, whereas Takahashi *et al.* [210] used a spectral-based approach to control group formations in applications like mass performances and tactical sports. Recently, example-based approaches have also been used to construct group behaviour models from motion captured data or from videos of real-crowds [97, 126, 128]. However, these approaches are too computationally expensive for real-time interactive applications and are commonly used for offline crowd simulations.

8.2 Definitions and Background

Our approach is directly inspired by the recent work of Moussaïd *et al.* [148] and focuses on the local behaviour of small pedestrian groups. In [148], empirical data of pedestrian crowds were collected using video recordings of urban areas. The analysis of the corresponding data has shown that the majority of the pedestrians walk in small groups consisting of up to three members. In addition, regarding the spatial organization of pedestrian groups, three distinct *formations* can be observed, as shown in Figure 8.2. Note that similar walking patterns were also observed by Peters and Ennis [169].

Typically, group members tend to walk next to each other forming a line perpendicular to the walking direction (*line-abreast* formation). Such a formation allows the pedestrians to easily communicate with each other while advancing toward their goal. At moderate crowd densities, the group space is reduced and a "*V-like*" formation emerges, facilitating the social communication between the group members. Finally, at high densities, safety prevails over social interactions and group members choose to walk behind

each other, which results in a "*river-like*" formation (leader-follower model). Note that for groups of two pedestrians, the V-shape formation is replaced by a more compact abreast formation, in which the security distance between the group members is significantly reduced.

8.2.1 Problem Formulation

In our problem setting, we are given a geometric description of the virtual environment in which small groups of agents must move. For simplicity, we assume that the agents are represented as discs translating in the plane amidst static polygonal obstacles and hence, the configuration space of each agent is identical to the 2D workspace. Then, based on the aforementioned empirical observations, we simulate groups of up to three agents; we assume that agents in larger groups tend to form smaller subgroups consisting of pairs or triples of individuals as also noted in pedestrian literature [27]. Given a group G_i having size $1 < N \leq 3$, we denote the position, radius and the velocity of an agent A_{ij} belonging to G_i as $\mathbf{x}_{ij}$, r_{ij} and $\mathbf{v}_{ij}$ respectively, where $j \in [1, N]$. During each simulation cycle, we also define the centroid $\mathbf{C}_i$ of the group, i.e. the average position of all the group members. Similarly, we determine the current velocity $\mathbf{V}_i$ of the group as the average velocity of its members. Finally, each group has a desired velocity $\mathbf{V}_i^{\text{des}}$ indicating the desired speed and direction of motion of its members.

Based on the empirically observed walking patterns, we further assume that each group G_i has k prioritized formations F_k, $k = 2 + N$, which in order of decreasing preference are the abreast, V-like and N river-like formations (see Figure 8.2). The group has as many river formations as the number N of its members, since any of the members can decide about the group's actions and become the group leader. In contrast, in the abreast and V-like formations, all members collectively decide for the group and thus, only two formations are taken into account. Each formation F_k is characterized by the tuple $(\mathbf{p}_i^{F_k}, \mathbf{o}_{ij}^{F_k})$, where $\mathbf{p}_i^{F_k}$ represents the reference point of the formation and $\mathbf{o}_{ij}^{F_k}$ describes the relative position of each group member with respect to the reference point. In case of the abreast or the V-shape formation, the reference point is represented by the centroid of the group, as all of its members are equally responsible for the decision-making process, whereas in the river-like formation, the reference point is designated by the position of the leader agent. For more information on the aforementioned reference techniques, we refer the reader to [5].

To be able to express each formation F_k in a coordinate-invariant way, we express the relative positions $\mathbf{o}_{ij}^{F_k}$ in a local coordinate system centered at the formation's reference point and oriented toward the group's desired direction of motion $\mathbf{n} = \frac{\mathbf{V}_i^{\text{des}}}{\|\mathbf{V}_i^{\text{des}}\|}$. Consequently, at any time step of the simulation, we can determine the desired location of the agent that is currently positioned at $\mathbf{x}_{ij}$ as:

$$\mathbf{x}_{ij}^{F_k} = \mathbf{p}_i^{F_k} + \mathbf{o}_{ij}^{F_k}[0]\mathbf{n} + \mathbf{o}_{ij}^{F_k}[1]\mathbf{n}^{\perp} \tag{8.1}$$

The formations of the groups are part of the problem description and are given in advance (e.g. by the level designer). In our simulations, some noise was also introduced to allow for variation in the three template formations. For convenience, we also assume that initially the group members are placed at their abreast formation. The problem can then be characterized as follows. The group agents A_{ij}, $j \in [1, N]$, need to reach a specified goal area without colliding with the environment and with each other, as well as with other individuals or groups that may be present in the virtual world. In addition,

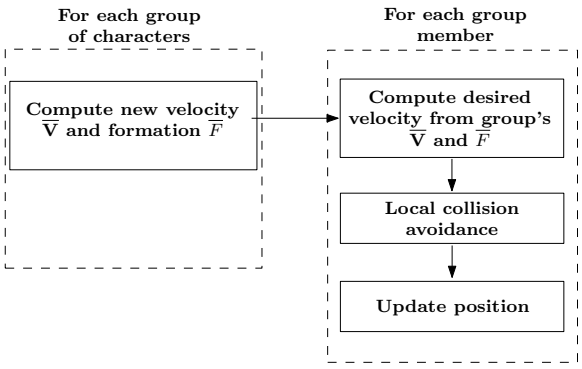


Figure 8.3: Schematic overview of our proposed framework.

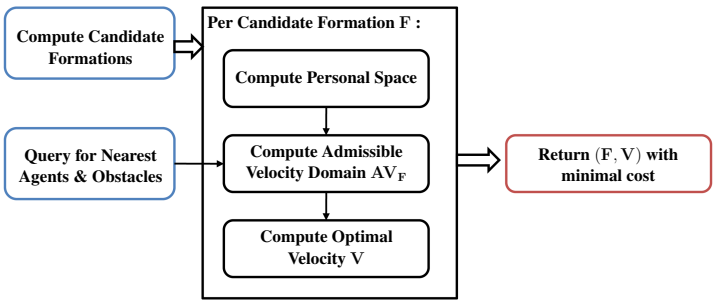


Figure 8.4: The first phase of our algorithm. Each group performs these computations at each time step. The candidate formations are computed using Eq. (8.2), whereas the cost function in Eq. (8.9) is used for the optimal velocity computation.

the agents must walk close together as a coherent group, striving to maintain a formation that supports the social communication among them. We assume that the goal is reached when all the agents of the group are within the goal area.

8.2.2 Overall Approach

We propose a two-phase approach to solve the aforementioned planning problem (Figure 8.3). In our setting, we assume that at every cycle of the simulation the desired velocity of each group is provided by some higher level path planning approach. Then, in the first step of our algorithm, we determine an avoidance maneuver for each group of agents. We formulate this as a discrete optimization problem of finding an optimal new formation and velocity for the entire group (Section 8.3). In the second step of our approach, we use the computed solution velocity and formation to determine the desired velocity of each group member. This velocity is then given as an input to a local collision avoidance model which returns the new velocity for the group agent (Section 8.4).

8.3 Avoidance Maneuvers for the Group

This section elaborates on the first phase of our approach. The goal here is to determine at each simulation step the new velocity $\overline{\mathbf{V}}_i$ and formation $\overline{F}_i$ of each group entity G_i, so that the group can safely navigate toward its target. At the same time, we aim for a formation $\overline{F}_i$ that facilitates the social interactions among the group members.

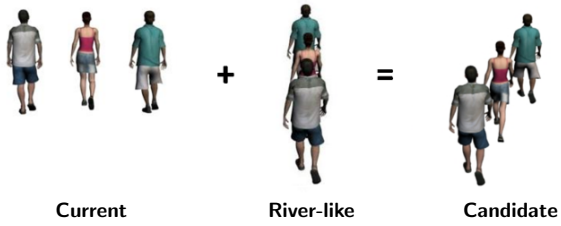


Figure 8.5: Example of interpolation between a current and a river-like formation to derive a candidate formation.

Our solution extends the velocity-based model proposed in Chapter 7 to account for different configurations that the group members can adopt. An optimal velocity (according to a certain cost function) is computed for each of these formations and the one with the lowest cost is retained. Figure 8.4 summarizes our extended velocity-based model. We now describe in detail the successive steps of one model iteration.

Step 1 - Determine the set of admissible formations AF for the group. Ideally, the group members prefer to walk next to each other in an abreast formation. However, in crowded or rather confined environments this can lead to deadlocks. Imagine for example two pairs of agents walking down a narrow corridor from opposite directions. If there is not enough room for both groups to fit through the corridor, the agents will get stuck. To resolve these challenging scenarios, the group should be able to dynamically adapt its formation, just like groups of pedestrians do in real-life. As a result, we consider a number of alternative formations by linearly interpolating between the current formation of the group and its k template formations.

Since Equation 8.1 provides the desired position for each group agent A_{ij} in the template formation F_k, our linear interpolation scheme over the interval $s \in [0, 1]$ can be formulated as follows:

$$\begin{aligned} \mathbf{x}_{ij}(s) &= (1-s)\mathbf{x}_{ij} + s\mathbf{x}_{ij}^{F_k}, \\ \mathbf{p}_i(s) &= (1-s)\mathbf{C}_i + s\mathbf{p}_i^{F_k}. \end{aligned} \tag{8.2}$$

An example of such an interpolation is depicted in Figure 8.5, where the current formation of the group combined with a river-like formation ($s = 0.5$) results in a diagonal stripe formation. Note that when the target formation is either the line-abreast or the V-shape, both the source and target formations share the same barycenter and thus, the reference point of the interpolated formation remains fixed to the centroid of the group. However, if the target formation is the river-like, the new reference point is determined from the weighted average between the source and target reference points.

Based on Equation 8.2, several candidate formations are formed (see Section 8.5.4 for more details). The generated formations are then inserted into the set AF in order to be evaluated in the next steps of our algorithm.

Step 2 - Define the personal space of each candidate formation $F \in AF$. Just like an individual is protected from unwanted social and physical contact by its personal space, a portable territory is also formed around a group formation indicating the area that others should not invade [1]. In Chapter 7, we approximate the personal space of an agent as a disc. Such a circular representation for a group, though, is an over-approximation of the actual space that the group members occupy, and cannot produce the correct microscopic

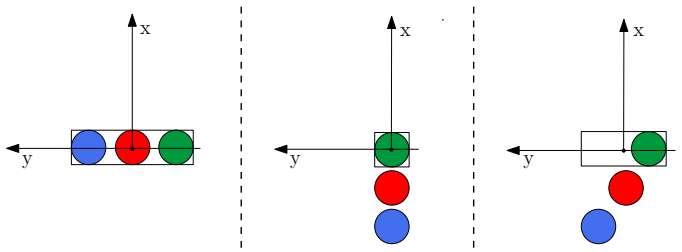


Figure 8.6: Personal space of group formations. Each colour represents a different group member. The reference point of the formation is located at the origin.

behaviour. Consider for example the minimum discs enclosing the line-abreast and the river-like formations in Figures 8.2(a) and 8.2(c), respectively. Both discs cover similar areas and thus, similar results will be obtained for two completely different configurations.

To resolve these issues, in our model, the personal space of each candidate formation is conservatively approximated as an axis-aligned bounding box (aabb). The aabb is defined in the local coordinate system centered at the reference point of the candidate formation and oriented toward the group's desired velocity. We refer the reader to Figure 8.6 for an example. As can be observed, the personal space of the formation is not determined by its minimum bounding box. Instead, our representation focuses on the lateral space that the formation occupies, which allows us to determine how soon the front of the formation will collide with static and dynamic obstacles. Then, using a min-max representation, the personal space of the candidate formation F can be computed from its local coordinates $\mathbf{o}_{ij}^F$ as follows:

$$\begin{aligned}&\left[\max_{j=1}^{N}\{\mathbf{o}_{ij}^F[0]-r_{ij}\},\ \min_{j=1}^{N}\{\mathbf{o}_{ij}^F[1]-r_{ij}\}\right]\times\\&\left[\max_{j=1}^{N}\{\mathbf{o}_{ij}^F[0]+r_{ij}\},\ \max_{j=1}^{N}\{\mathbf{o}_{ij}^F[1]+r_{ij}\}\right].\end{aligned}\tag{8.3}$$

Step 3 - Determine the admissible velocity domain AV_F of each candidate formation. In particular, we compute the first N agents that are on collision course with F given a certain anticipation time t_α. We assume that a collision at some time $ttc \geq 0$ occurs when an agent steps into or touches the personal space of the formation. Consequently, collision prediction can be achieved by performing a swept test [40] between an aabb (formation's personal space) and a disc (agent A), where the motion of the box is determined by the group's desired velocity and the motion of the disc by the current velocity of the agent. Note that, for efficiency reasons, we only consider agents that are visible and in close proximity to G_i. This also leads to a more natural (human) behaviour; in real life, group members typically pay attention to a limited number of other people, those that they can see and usually those that are nearby. In a similar way, we determine the set of threatening static obstacle neighbours of F. Then, the set of admissible velocities AV_F varies according to the minimal time-to-collision ttc between the candidate formation and its neighbouring agents/obstacles; collisions in the far future lead to a rather limited domain, whereas shorter collision times allow larger variation in the admissible velocities to resolve imminent collisions.

The relationship between the minimal time-to-collision and the set of admissible velocities is mathematically defined based on the motion analysis of human interactions in

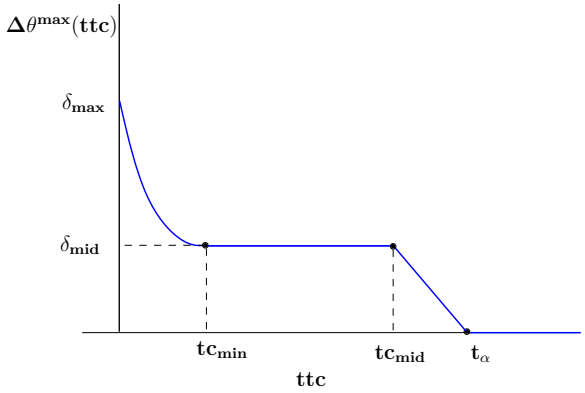


Figure 8.7: Maximum orientation deviation as a function of the minimal predicted time to collision.

controlled scenarios (see Chapter 7). Figure 8.7 plots the maximum angle $\Delta\theta^{\max}$ that the formation can deviate from its desired direction of motion as a function of the minimal ttc for some user-defined $tc_{\min}$, tc_{mid} and d_{mid} and $d_{\max}$ parameters. We detail the role of these parameters in Section 8.6.1.

Having retrieved the maximum deviation angle $\Delta\theta^{\max}$, we determine the admissible orientation domain O_F of the candidate formation as follows:

$$O_F = \{\ \mathbf{n}_\theta \mid \theta \in [\theta^{\mathrm{des}} - \Delta\theta^{\max}, \theta^{\mathrm{des}} + \Delta\theta^{\max}]\ \}, \tag{8.4}$$

where $\mathbf{n}_\theta = [\cos\theta, \sin\theta]^T$ and θ^{des} is the orientation angle derived from the group's desired velocity $\mathbf{V}^{\mathrm{des}}$. Similarly, the admissible speed domain U_F is approximated by the following piecewise function:

$$U_F = \begin{cases} U^{\mathrm{des}}, & \text{if } tc_{\min} < ttc \\ v \mid v \in [U_-, U_+]\,, & \text{otherwise,} \end{cases} \tag{8.5}$$

where

$$U_- = U^{\mathrm{des}}(1 - e^{-ttc}), \tag{8.6}$$

$$U_+ = U^{\mathrm{des}} + (U^{\max} - U^{\mathrm{des}})\, e^{-ttc}. \tag{8.7}$$

$U^{\mathrm{des}} = \left\|\mathbf{V}^{\mathrm{des}}\right\|$ denotes the preferred speed of the group and $U^{\max} \geq U^{\mathrm{des}}$ defines the maximum speed at which the group members can move. Note that, compared to the velocity model of Chapter 7 (cf. Equation 7.12), the admissible speed domain is only adapted in case of an imminent collision. Thus, U_F is set to the preferred speed U^{des} by default, whereas a negative exponential function controls the admissible speeds when ttc is lower than the $tc_{\min}$ threshold.

Finally, given the admissible orientation and speed domains, we can deduce the set of feasible velocities AV_F as:

$$AV_F = \{v\,\mathbf{n}_\theta \mid v \in U_F \wedge \mathbf{n}_\theta \in O_F\}. \tag{8.8}$$

We refer the reader to Section 7.4.1 for more details regarding this step.

Step 4 - For each candidate formation F, determine an optimal solution velocity. In particular, we discretize the set AV_F into a number of candidate velocities and retain the velocity

that minimizes a specific cost function. Similar to [227], Similar to Chapter 7 (cf. Equation 7.14), our cost function depends on the deviation from the group's desired velocity $\mathbf{V}^{\text{des}}$ and the minimal predicted time-to-collision ttc between the formation's personal space and the neighbouring agents/obstacles determined in the previous step of our model. An additional cost term $g(F)$ is also included to indicate the penalty of selecting the formation F. The cost for a candidate velocity $\mathbf{V}$ can now be computed as:

$$cost(\mathbf{V}, F) = \kappa_1 \frac{\|\mathbf{V} - \mathbf{V}^{\text{des}}\|}{2\,U^{\max}} + \kappa_2 \frac{t_\alpha - ttc}{t_\alpha} + \kappa_3\, g(F), \tag{8.9}$$

where the constants $\kappa_1, \kappa_2, \kappa_3$ define the weights of the specific cost terms and can vary to simulate different avoidance (see Section 8.6.1 for details).[1]

The term $g(F)$ indicates the energy that is required to deform from the abreast formation into the candidate formation and is approximated as:

$$g(F) = \frac{\left|W^{Abreast} - W^{F}\right|}{W^{Abreast} - W^{River}}, \tag{8.10}$$

where W^i indicates the width of the personal space of formation i, that is the width of its corresponding aabb. As can be observed, the deformation cost becomes zero when the candidate formation is the line-abreast formation. This is in accordance with the empirical observations mentioned in Section 8.2 indicating that group members prefer to walk side by side, since they can easily communicate with each other. In contrast, the deformation cost is maximal when the river-like formation is selected, as there is no social communication between the members.

Step 5 - Select the new velocity $\overline{\mathbf{V}}_i$ and formation $\overline{F}_i$ of the group. Having retrieved the optimal solution velocity for each candidate formation, we retain the velocity $\overline{\mathbf{V}}_i$ that has the minimal cost among all the solutions velocities. Its corresponding formation is also selected as the new group formation $\overline{F}_i$ for the current step of the simulation:

$$(\overline{\mathbf{V}}_i, \overline{F}_i) = \underset{\substack{F \in AF \\ \mathbf{V} \in AV_F}}{\operatorname{argmin}} \{cost\,(\mathbf{V}, F)\}. \tag{8.11}$$

Pruning Candidate Formations. For each candidate formation we need to solve Equation 8.9 in order to determine its optimal avoidance velocity. This may lead to performance bottlenecks, especially when solving the equation for too many candidate formations. However, many of these discrete formations are obsolete and thus, it is important to prune them without evaluating them. Let us assume that a number of candidate formations have been evaluated and the optimal formation and its corresponding velocity have been computed. When a new candidate formation needs to be evaluated, we first determine its weighted deformation cost. If this cost is higher than the current optimal one, then, based on Equation 8.9, the minimal cost of the new formation will also be higher. Hence, the candidate formation is pruned.

[1] Note that the effort term of Equation 7.14 was dropped from our cost function. The group members will still perform smooth avoidance maneuvers since, in the second phase of our algorithm, we use Equation 7.14 to plan separately for each group member (see Section 8.4.2).

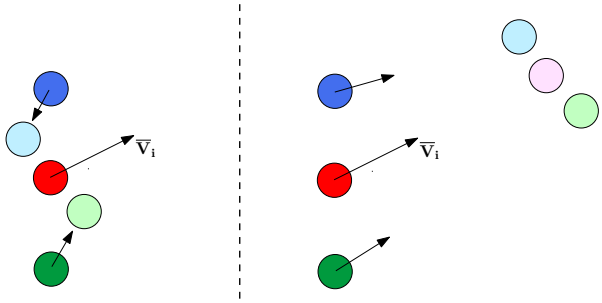


Figure 8.8: Determining the desired direction of motion for each group member. **Left:** Using the new formation, one of the members has to move backwards. **Right:** Extrapolating the new formation in the future results in smooth motions. Current and future positions are represented with dark and light colours respectively. The red disc also denotes the reference point of the new formation.

Optimizations. Recall that in the third step we need to perform collision tests between the candidate formation of the group G_i and each out-group agent. This task is computationally expensive, especially when the total number of simulated agents is large. In practice, though, as discussed above, we only consider agents that are visible and in close proximity to G_i. To accelerate the search of nearest visible agents, a spatial-hash data structure for answering nearest neighbour queries is used similar to the one described in Section 6.5. We also assume that physical dominance plays an important role in the avoidance behaviour of the group and hence, we only take into account agents that belong to larger or equally-sized groups than G_i. This not only reduces the running time of our algorithm, but also captures the behaviour of real pedestrian groups. Consider, for example, a group of three members interacting with a single individual. We expect the individual, rather than the group, to take an evasive action and avoid the collision. Note that employing the latter technique is sufficient for a collision-free navigation of the group members; imminent collisions that are not handled in the first phase of our algorithm will be resolved during the second phase, as explained in Section 8.4.2.

8.4 Avoidance Maneuvers for the Members

In this section, we elaborate on the second phase of our global approach. The goal here is to plan the motion of each group member A_{ij} based on the new velocity $\overline{\mathbf{V}}_i$ and formation $\overline{F}_i$ of its group G_i. Our proposed model consists of two steps. The first step determines the desired velocity of the agent A_{ij}. This is followed by a local collision avoidance step, which computes a collision-free velocity for the agent.

8.4.1 Computing a Desired Velocity

In the first step, we retrieve the new desired velocity $\mathbf{v}_{ij}^{\text{des}}$ of A_{ij}, that is its desired direction of motion $\mathbf{n}_{ij}^{\text{des}}$ and speed v_{ij}^{des}. Ideally, at every cycle of the simulation, the agent A_{ij} should move toward its corresponding position in the new formation $\overline{F}_i$. Note, though, that this can introduce oscillations, e.g., if the agent is already ahead of its desired position or too close (see Figure 8.8, left). To alleviate this, we extrapolate the reference point of the new formation into the near future based on the new velocity $\overline{\mathbf{V}}_i$, that is:

$$\mathbf{p}_i' = \mathbf{p}_i^{\overline{F}_i} + \overline{\mathbf{V}}_i * t_{\text{extrapolate}} \tag{8.12}$$

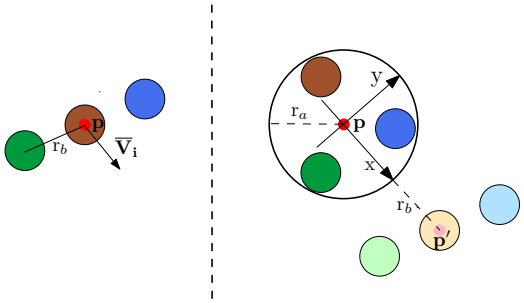


Figure 8.9: Left: The new formation $\overline{F}_i$ of the group. **Right:** Extrapolating $\overline{F}_i$ into the near future based on the new velocity $\overline{\mathbf{V}}_i$ and the $t_{\text{extrapolate}}$ parameter. Current and future group member positions are represented with dark and light colours respectively. The small red disc denotes both the reference point $\mathbf{p}^{\overline{F}_i}$ of the new formation and the centroid of the current group formation.

To ensure smooth behaviour and avoid the aforementioned oscillations, the parameter $t_{\text{extrapolate}}$ is clamped to a minimum value $t_{\min}$ determined as:

$$t_{\min} = \frac{\max\limits_{j=1}^{N}(\|\mathbf{x}_{ij} - \mathbf{p}_i^{\overline{F}_i}\| + r_{ij}) + \max\limits_{j=1}^{N}\|\mathbf{o}_{ij}^{\overline{F}_i}\|}{\|\overline{\mathbf{V}}_i\|}. \tag{8.13}$$

The first term in the numerator of Equation 8.13 denotes the radius of the smallest disc centered at $\mathbf{p}_i^{\overline{F}_i}$ that encloses all the members of the group. The second term defines the radius of the new formation, that is the maximum distance between the reference point of the new formation and the other points in the formation. We refer the reader to Figure 8.9 for a visual depiction.

Lemma 8.1. *Choosing $t_{\text{extrapolate}} \geq t_{\min}$ guarantees that the desired position of each group member A_{ij} lies ahead of its current location $\mathbf{x}_{ij}$.*

Proof: Let us first for notational convenience denote

$$\begin{aligned} r_a &= \max_{j=1}^{N}(\|\mathbf{x}_{ij} - \mathbf{p}_i^{\overline{F}_i}\| + r_{ij}), \\ r_b &= \max_{j=1}^{N}\|\mathbf{o}_{ij}^{\overline{F}_i}\|. \end{aligned} \tag{8.14}$$

Then, according to Equation 8.13, $t_{\min} = \dfrac{r_a + r_b}{\|\overline{\mathbf{V}}_i\|}$.

We prove by contradiction that the lemma holds. Let $t_{\text{extrapolate}} \geq t_{\min}$ and let at least one of the group members A_{ij} have its desired future position $\mathbf{x}'_{ij} = \mathbf{x}_{ij}^{\overline{F}_i}$ lying behind its current position $\mathbf{x}_{ij}$. Consider also the local coordinate system centered on $\mathbf{p}^{\overline{F}_i}$ and oriented towards the new velocity $\overline{\mathbf{V}}_i$ of the group (hereafter, for notational convenience, we will drop the subscripts i, j). Then, according to Figure 8.9, it holds:

$$\|\mathbf{x} - \mathbf{p}^{\overline{F}}\| < r_a \Rightarrow \mathbf{x}_L[0] < r_a, \tag{8.15}$$

where the subscript L denotes that the coordinates are expressed in the local system. Regarding the member's desired position $\mathbf{x}'$ and the extrapolated position $\mathbf{p}'$ of the reference point, it also holds:

$$\|\mathbf{x}' - \mathbf{p}'\| \leq r_b \Rightarrow \mathbf{p}'_L[0] - \mathbf{x}'_L[0] \leq r_b. \tag{8.16}$$

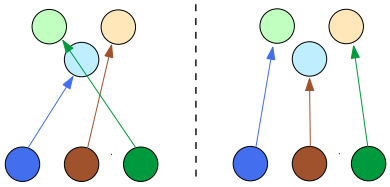


Figure 8.10: Determining the desired direction of motion by **left:** moving towards the desired future positions, **right:** finding the best matching between the current and the desired positions of the group members.

Since we assume that the desired position lies behind the current position, its x local coordinate will always be smaller than the one of the current position:

$$\begin{array}{ccccc} \mathbf{x}_L[0] & > & \mathbf{x}'_L[0] & \Longrightarrow & \\ -\mathbf{x}_L[0] & < & -\mathbf{x}'_L[0] & \Longrightarrow & \\ \mathbf{p}'_L[0] - \mathbf{x}_L[0] & < & \mathbf{p}'_L[0] - \mathbf{x}'_L[0] & \underset{\mathbf{p}'_L[0]=r_a+r_b}{\overset{(8.16)}{\Longrightarrow}} & \\ r_a + r_b - \mathbf{x}_L[0] & \leq & r_b & \Longrightarrow & \\ \mathbf{x}_L[0] & \geq & r_a & & \end{array} \tag{8.17}$$

Thus, according to Equation 8.15, we arrive at a contradiction. Therefore, if $t_{\text{extrapolate}} \geq t_{\min}$, the desired position of A_{ij} is guaranteed to always lie ahead of its current location $\mathbf{x}_{ij}$.

■

Based on the extrapolated position $\mathbf{p}_i{}'$ of the reference point, we can now deduce the future formation $\overline{F}_i'$ of the group around that position and also determine the corresponding desired future location of the agent A_{ij} from Equation 8.1, where the group's desired direction of motion $\mathbf{n}$ is estimated from $\overline{\mathbf{V}}_i$, that is, $\mathbf{n} = \frac{\overline{\mathbf{V}}_i}{\|\overline{\mathbf{V}}_i\|}$. Then, the desired direction of motion of A_{ij} (see Figure 8.8, right) can be computed as:

$$\mathbf{n}_{ij}^{\text{des}} = \frac{\mathbf{x}_{ij}^{\overline{F}_i'} - \mathbf{x}_{ij}}{\|\mathbf{x}_{ij}^{\overline{F}_i'} - \mathbf{x}_{ij}\|}. \tag{8.18}$$

It must be pointed out that the technique described so far assumes that the agents respect their relative formation positions $\mathbf{o}_{ij}^{\overline{F}_i'}$ while selecting their new direction of motion. However, this may not always lead to a desired behaviour, as the group members may need to put a lot of effort and perform a number of extra avoidance maneuvers to safely reach their desired (future) locations without colliding with each other (Figure 8.10, left). Thus, an alternative approach, is to find a one-one correspondence between the current positions of the group members and the positions in the extrapolated future formation $\overline{F}_i'$ while minimizing the total distance that the members have to travel (Figure 8.10, right). In all of our simulations, this approach was used to derive the new desired direction of motion $\mathbf{n}_{ij}^{\text{des}}$ of each agent A_{ij} resulting in a smooth transitioning of the group formation.

Using the $\mathbf{n}_{ij}^{\text{des}}$, the agent A_{ij} is now able to find its place in the desired new formation. However, it should also adapt its desired speed v_{ij}^{des} accordingly, so that the entire group G_i can smoothly progress from its current formation to the new one. In other words, the agent A_{ij} has to be able to slow down and wait for the other members or speed up if it lags behind. Thus, we first determine for both the reference point $\mathbf{p}_i^{\overline{F}_i}$ and the A_{ij}

the distance that they have to traverse from their current position to their corresponding extrapolated future position. We then define the difference d_{diff} between these distances as:

$$d_{\text{diff}} = \|\mathbf{x}_{ij}^{\overline{F}_i'} - \mathbf{x}_{ij}\| - \|\mathbf{p}_i' - \mathbf{p}_i^{\overline{F}_i}\|. \tag{8.19}$$

The new desired speed of A_{ij} is now computed as:

$$v_{ij}^{\text{des}} = \|\overline{\mathbf{V}}_i\| + d_{\text{diff}}/t_{\text{extrapolate}}, \tag{8.20}$$

where v_{ij}^{des} is clamped to be nonnegative. A special case exists when the reference point $\mathbf{p}_i^{\overline{F}_i}$ is located on the current position of the member A_{ij}, that is, A_{ij} is the leader of the group. Then, the leader's desired speed is computed in a similar fashion using Equation 8.20. However, d_{diff} is now determined by the difference between the distance that the reference point has to traverse and the average traverse distance of the other group members. This allows the leader to adapt its speed and wait for the slower moving members.

Finally, having retrieved both the desired direction $\mathbf{n}_{ij}^{\text{des}}$ and speed v_{ij}^{des} of the agent A_{ij}, its desired velocity $\mathbf{v}_{ij}^{\text{des}}$ is given by $\mathbf{v}_{ij}^{\text{des}} = v_{ij}^{\text{des}}\mathbf{n}_{ij}^{\text{des}}$.

8.4.2 Local Collision Avoidance

In the second step of the agent planning phase, the desired velocity $\mathbf{v}_{ij}^{\text{des}}$ of the group agent A_{ij} is used as an input to a local collision avoidance method in order to retrieve a collision-free velocity for the agent and update its position. The agent still needs to avoid collisions with the other members of its group, as well as with nearby individuals and groups, due to the fact that in the first phase of our algorithm we plan for the personal space of each candidate group formation and not for the actual formation of the group. As a result, collisions may occur that need to be resolved.

In general, any local method that is based on collision prediction can be used, such as force-based approaches (see, e.g., the predictive model introduced in Chapter 6 or the work of Reynolds in [182]) and velocity-based models [173, 227]; force-field approaches yield to higher performances, whereas velocity methods are much more easier to control and provide more robust avoidance behaviour. In our implementation, we used the velocity-based approach proposed in Chapter 7, since it is elaborated from experimental interactions data resulting in smooth and convincing motions. The approach takes as input the desired velocity of the agent and returns an optimal new velocity among a set of admissible velocities by minimizing the risk of collisions with other agents, the deviation from the desired velocity and the amount of effort that the agent requires to adapt its motion. We refer the reader to the aforementioned chapter for more details.

It is important to note that, in our case, the agent A_{ij} perceives and avoids other groups as single entities, just like individuals do in real-life. Consequently, when solving interactions with out-group members, the agent takes into account the space that their group occupies, instead of trying to avoid each group member individually. Only when the distance between the group members becomes too large, the group is not considered anymore as one entity but as a collection of individual pedestrians [21].

Since we strive for coherent groups, we would also like the group members to avoid the static and dynamic obstacles without splitting up.[2] This can be achieved by restricting the admissible velocity domain AV_{ij} of each group agent A_{ij}, so that A_{ij} will either select

[2] In case the force-based approach of Chapter 6 is used for local collision avoidance, group coherence can be achieved by introducing an additional attractive force for each group member A_{ij} that points from A_{ij} toward the new reference point $\mathbf{p}_i'$ of the group.

(a) Crowd simulation in a shopping mall (b) Another view of the simulation (c) A still from a video of the mall

Figure 8.11: Group interactions at the Hoog Catharijne shopping mall in Utrecht, Netherlands (c). Our method is able to predict the emergence of empirically observed walking patterns generating convincing motions (a, b).

its desired velocity or a velocity that will steer the agent close to the new reference point $\mathbf{p}_i'$ of the group. In particular, we decompose AV_{ij} into the domains AV_{ij}^+ and AV_{ij}^-, which respectively correspond to the half-plane of velocities on the right and the half-plane of velocities on the left of the agent's desired velocity $\mathbf{v}_{ij}^{\text{des}}$:

$$AV_{ij}^+ = \{\mathbf{v} \in AV_{ij} \,|\, \mathbf{v} \times \mathbf{v}_{ij}^{\text{des}} \geq 0\}, \tag{8.21}$$

$$AV_{ij}^- = \{\mathbf{v} \in AV_{ij} \,|\, \mathbf{v} \times \mathbf{v}_{ij}^{\text{des}} \leq 0\}. \tag{8.22}$$

We then determine the new set of admissible velocities AV_{ij}' as follows:

$$AV_{ij}' = \begin{cases} AV_{ij}^+, & \text{if } \lambda(\mathbf{x}_{ij}^{\overline{F}_i'}, \mathbf{p}_i^{\overline{F}_i}, \mathbf{p}_i') > 0 \\ AV_{ij}^-, & \text{if } \lambda(\mathbf{x}_{ij}^{\overline{F}_i'}, \mathbf{p}_i^{\overline{F}_i}, \mathbf{p}_i') < 0 \\ AV_{ij}^+ \cup AV_{ij}^-, & \text{otherwise,} \end{cases} \tag{8.23}$$

where $\lambda(\mathbf{c}, \mathbf{a}, \mathbf{b})$ denotes the signed distance from the point $\mathbf{c}$ to the line $\overrightarrow{\mathbf{ab}}$. Consequently, if the future position $\mathbf{x}_{ij}^{\overline{F}_i'}$ of the agent lies to the right (left) of the line between the reference point $\mathbf{p}_i^{\overline{F}_i}$ and its extrapolated position $\mathbf{p}_i'$, only the left (right) admissible domain is considered. Note that if $\mathbf{x}_{ij}^{\overline{F}_i'}$ lies on the line, both velocity domains are taken into account.

8.5 Results

We have implemented our approach to validate the quality of the generated motions and test its applicability in real-time applications. All experiments were performed on a 2.4 GHz Core2 Duo CPU (on a single thread) with an ATI Radeon HD 4800 GPU. At runtime, the desired velocity of each group was provided by the IRM using the medial axis as indicative route (see Chapter 3).

8.5.1 Scenarios

We demonstrate the capability of our approach in different scenarios:

Deformable Formations. One of the key concepts of our approach is the ability of the groups to dynamically adapt their formations in confined and dynamic environments in order to safely navigate towards their goals. At the same time, the groups favour formations that facilitate the communication between their members, exhibiting behaviour

similar to the one observed in real pedestrian groups. For example, we simulate a group of three agents that have to navigate through a narrow corridor (Figure 8.1(d)). Using our approach, the group adapts to a V-shape formation, allowing its members to avoid collisions and communicate with each other. In contrast, using only a local collision avoidance method results in a wedge (inverse V-like) formation. Although such formation is more efficient than the V-shape, it is not suited for social interactions between the group members, since the leader has to turn his back on the other two members. Another example is shown in Figure 8.1(a), where a group of two agents has to pass through a doorway. Since only one agent can fit through the door, the group gradually adapts to a river-like formation ensuring a collision-free motion.

Interaction Scenarios: Figure 8.1(b) shows a group-group interaction in a narrow corridor, where both groups have to change their formations in order to safely reach their goals. Note, though, that when a group has enough space to maneuver, it prefers to slightly deviate from its desired direction and maintain its abreast formation, rather than adapt its configuration. Another interaction example is depicted in Figure 8.1(c), in which a fast moving group encounters a slower moving group heading into the same direction. Using our approach, the faster group tries to stay as coherent as possible by adapting its formation for a short time period before getting back to a line formation. In contrast, planning separately for each group member results in the faster moving group to split in order to avoid the slower moving agents.

Shopping Mall: Figures 8.11(a) and 8.11(b) show a crowd of 300 agents entering/exiting a shopping mall area. The crowd consists of pairs, triples and individual agents. We visually compared the corresponding simulation with the motion of a real pedestrian crowd observed by means of video recordings. The video data was collected in the Hoog Catharijne shopping mall in Utrecht, the Netherlands (see Figure 8.11(c)). Although a perfect reproduction of the real crowd is almost impossible, overall, the simulated groups exhibit similar behaviour with the real ones. Furthermore, our model is able to predict the emergence of walking patterns and collective phenomena, such as the formation of lanes, noted in the real video.

Busy Crosswalk: We also simulated pedestrians interactions at a crosswalk of a virtual city environment. Here, small groups of agents cross paths with other groups and individual agents while travelling in either direction through a crowded street that is $200\,\mathrm{m}$ long and $20\,\mathrm{m}$ wide and is characterized by an average density of $0.15\,\mathrm{peds}/\mathrm{m}^2$. To reflect a real pedestrian crowd, the sizes of the pedestrian groups (including individuals) were distributed using a Poisson distribution [27]. Our method was able to generate well-know crowd phenomena which have been noted in the pedestrian literature, such as the dynamic formation of lanes, overtaking behaviour near the edges of the crowd and the emergence of slowing down and stopping behaviour to successfully resolve imminent collisions. In addition, the simulated groups exhibit coherent behaviour and form spatial patterns similar to the ones observed in real crowds. Note that, in dense areas, the group members might limit their social interactions and temporarily employ wedge-like formations to safely move within the crowd. Such formations are not explicitly modeled, but they derive from our formation interpolation scheme. As still some communication may take place, they have a higher preference (lower deformation cost) over the river-like formations.

8.5.2 Quality Evaluation

Besides the visual inspection of the generated simulations, we are also interested in a quantitative evaluation of our model. As a result, we have devised a number of simple quantitative metrics that can capture the overall steering behaviour of the simulated groups. In particular, we compute the *percentage* of the time that a group is not in its line-abreast or V-shape formation. Additionally, since we aim for groups of agents that remain as coherent as possible, we propose two additional metrics to determine the coherent behaviour of a group. The first metric measures the *distortion* of the group, that is how much the actual formation of a group G_i deviates from its desired abreast formation F_0. For a given time step of the simulation, the distortion D_i of G_i is defined as:

$$D_i = \min_{k \in [1..N!]} \sum_{j=1}^{N} \|\mathbf{x}_{ij} - \mathbf{x}_{ij}^{P_k(F_0)}\|, \tag{8.24}$$

where $P_k(F_0)$ denotes the set of all possible position permutations of the formation F_0. Consequently, to determine the group's distortion, we search for the best matching between the current and the line-abreast formation based on the sum of the Euclidean distances between the corresponding formation positions. The minimum of these sums defines the group's distortion from its ideal abreast formation.

The second metric is the *longitudinal dispersion* of the group, measured by the distance between the front s_i^f and the back s_i^b of the group G_i. The front of the group is determined by the member that is leading the group, that is, the agent whose position in the group's local space has the maximum x coordinate. Similarly, the back of the group is determined by the minimum x local coordinate among the group members. Then, the longitudinal dispersion L_i of the group G_i can be defined as $L_i = \|s^f - s^b\|$.

Comparison with Empirical Data

To determine how well our approach captures the behaviour of real pedestrian groups, we compared our numerical simulations with empirical data of pedestrian groups using our three evaluation metrics. In particular, we exploited two publicly available pedestrian datasets that were collected by means of video recordings. The first one consists of 360 manually annotated trajectories of a relatively sparse pedestrian crowd [235]. The second dataset was obtained from a 3.5 minute long video of a dense crowd and contains 434 pedestrians annotated with splines [236]. Since our focus is on pedestrian groups, a simple visualization front-end was developed to identify groups in the video sequences. The tool allows the user to easily navigate through the video frames and indicate pedestrians belonging to the same group. It also corrects the perspective distortion of the tracked trajectories by letting the user specify in the video scene a rectangle whose dimensions in the real world are known. Based on the known coordinates, a homography matrix is estimated and used to transform the trajectories of the pedestrians from image to real-world coordinates.

In total, we identified 49 groups composed of two to three members in the sparse video and 100 groups in the dense video. We then used the projectively corrected trajectories of the tracked pedestrians to gather the corresponding evaluation statistics of the groups. Table 8.1 compares the group metrics of the sparse crowd with the ones obtained from the shopping mall simulation. Note that for the distortion metric, we define the desired formation of a real pedestrian group as the average over all configurations that the group adopts during its lifespan.

Metrics	Mall simulation		ETH dataset		p-values
	Mean	Stdev	Mean	Stdev	
Dispersion (m)	0.45	0.12	0.40	0.57	0.617
Distortion (m)	0.64	0.85	0.55	0.57	0.540
Out of formation (%)	2.3	0.20	3.7	0.14	0.311

Table 8.1: Comparison between observed and simulated groups using our proposed evaluation metrics.

The results show that, on average, only $2.3\% \pm 0.2$ of the time a group in our simulation maintains a formation different than the line-abreast or the V-like formation. This demonstrates the ability of the group members to find comfortable walking positions supporting the communication with each other. Additionally, the average dispersion and distortion of the simulated groups are quite low, $0.45 \pm 0.12, 0.64 \pm 0.85$ respectively, indicating that the groups remain as gathered as possible during their navigation and deform only if it is necessary.

As can be inferred from the table, a series of Student's t-tests revealed no significant differences between the simulated groups and the real ones for our given metrics ($p > 0.1$ for all three metrics). Similar results were also obtained when evaluating the crosswalk simulation against the dense crowd dataset, indicating that, overall, the simulated groups match the observed ones very well.

8.5.3 Data-driven Validation

A simple comparison using our proposed evaluation metrics is not sufficient to fully and objectively assess the quality of a simulation. We also need to examine each group in detail, so that, at every simulation step, we can determine whether characteristics such as the group's distortion or dispersion exhibit abnormal behaviour that may hinder the overall quality of the simulation. Therefore, we also employed a *data-driven* evaluation approach similar to the one proposed by Lerner *et al.* [129].

Approach

In [129], videos of real crowds are analyzed and then used to create a database of examples. Each example represents a unique pedestrian behaviour and is described by a *state-action* pair. At a specific point in time and space, a state S is defined as a vector of important attributes that may influence the pedestrian's behaviour, such as the crowd density or the distance to nearby individuals. Based on this state, an action A is assigned to the pedestrian. Such an action is typically expressed by a segment of the pedestrian's trajectory. Given a simulation, a similar analysis is also applied to the simulated trajectories of the agents and state-action pairs are also defined. These pairs are then used as queries to search the database and evaluate the quality of the simulation. For each query, the most similar example in the database is retrieved based on a similarity measure and an evaluation score from 0 to 100 is returned providing a detailed assessment of the agent's behaviour at that specific moment in time.

We modified the aforementioned data-driven approach so that we can evaluate how groups, rather than individual agents, interact and behave within a simulated crowd. As explained in Section 8.2, the spatial organization of pedestrian groups is mainly influenced by the crowd density. Consequently, in our case, the state attributes are density-based. Similar to [129], to evaluate a group's decision at time T, the density state is

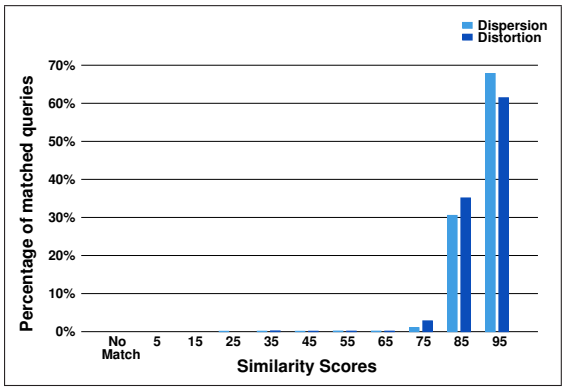


Figure 8.12: Distribution of similarity scores when evaluating the shopping mall simulation against the sparse crowd database.

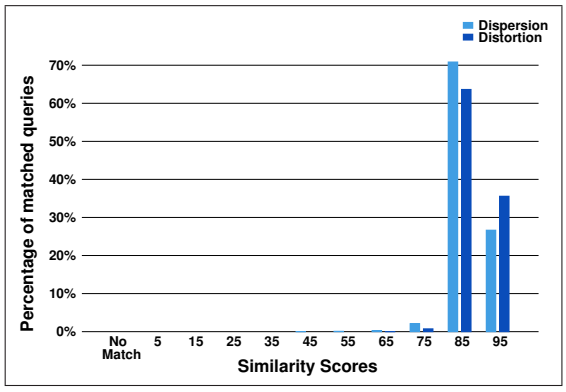


Figure 8.13: Distribution of similarity scores when evaluating the crosswalk simulation against the dense crowd database.

sampled over five time steps, at times $T-1, T-\frac{1}{2}, T, T+\frac{1}{2}, T+1$. Each sample stores the number of individuals in sixteen regions surrounding the pedestrian group. To be more specific, we consider a 2D grid defined in the local coordinate system that has its origin at the centroid of the group and is oriented towards the group's desired direction of motion. The front of the grid is extended 6 meters from the group's centroid and its back 2 meters. Laterally, the grid is extended 4 meters to the left and 4 meters to the right of the centroid. In total, it covers an area of 8 by 8 meters and is divided into sixteen square cells. For simplicity, we clamp the maximal density of each cell to five people and normalize the densities accordingly. Sampling the density state over the five time steps, produces a total vector of 80 normalized density values.

To assess the behaviour of a group, we use two action measures that are based on our proposed distortion and dispersion metrics. For both actions, we consider three samples over a window of two seconds, one second before and one second after the current time T. Each sample in the distortion measure stores the distance between the current formation of the group and its desired one according to Equation 8.24. Regarding the dispersion measure, we store at each sample time the group's longitudinal dispersion L_i. Lastly, we define the similarity function $C(Q, E)$ between a query state-action pair Q and an example pair E as in [129]:

$$C(Q, E) = (1 - D_S(S_Q, S_E)) \times (1 - D_A(A_Q, A_E)) \times 100. \tag{8.25}$$

The distance between the density states $D_S(S_Q, S_E)$ is defined as a weighted Manhattan distance of their attribute vectors with more importance placed around the current time step T. Similarly, the distance between the actions $D_A(A_Q, A_E)$ is computed as a weighted Manhattan distance between their samples. Each sample distance is clamped by an upper value (proportional to the maximum dispersion/distortion value observed in the example database) and normalized to $[0, 1]$.

Results

To validate our simulations, we built two example databases. The first one was created from the sparse crowd dataset and the second from the dense crowd dataset. In our first experiment, we used the sparse crowd database as input and evaluated the shopping mall simulation. It must be pointed out that, during the evaluation, we distinguished between groups of two and groups of three members. Therefore, for each query, we

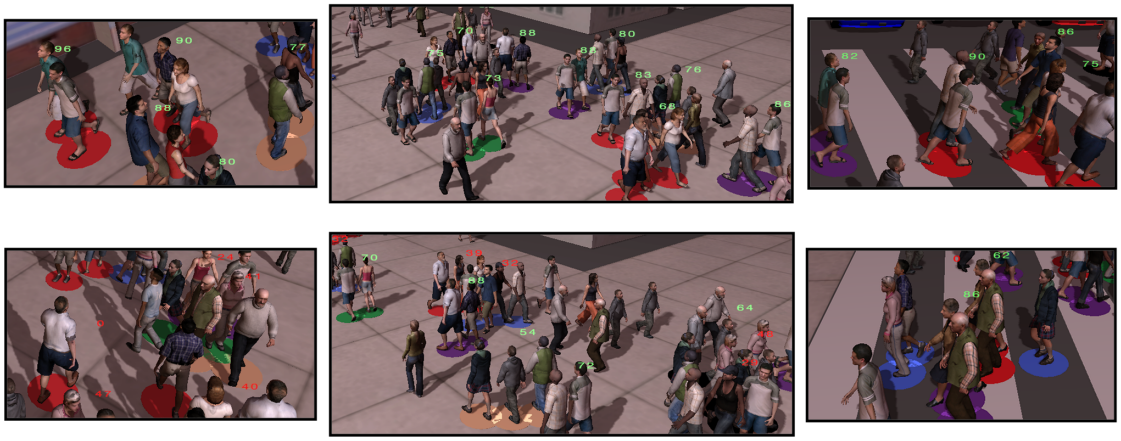


Figure 8.14: Examples from the evaluation of the crosswalk simulation using the formation measure. **Top:** With our approach, high-quality matches are obtained. **Bottom:** Using the approach of Moussaïd *et al.* [148], the groups exhibit atypical behaviours and receive low similarity values.

searched either the "2-member" or the "3-member" group examples, depending on the actual size of the evaluated group. Figure 8.12 shows the results we obtained for both the dispersion and the distortion measures. The columns of the histograms represent ranges of evaluation scores and the height of each column the relative number of simulation examples that fall within that range.

As can be inferred from the figure, almost 99% of the queries were classified as high-quality, receiving a similarity score higher than 75. This can be observed for either measure and shows that the groups simulated with our approach exhibit behaviours similar to the ones observed in real pedestrian groups. However, a small number of simulated examples also received low evaluation scores, that is, below 50. Further analysis revealed that the low-quality matches do not correspond to atypical behaviours, but are attributed to the fact that the input video does not contain enough examples to cover these behaviours. The number of such false positive examples, though, was not significant (21 out of 6708 queries for the distortion measure and 24 out of 6708 for the dispersion).

Similar conclusions can also be drawn when evaluating the crosswalk simulation against the dense crowd database. Figure 8.13 shows the corresponding distribution of the similarity scores for both the distortion and the dispersion measures. It is interesting to note that, this time, the input video contains enough unique group behaviours. Therefore, the example space is better covered and no simulation query received a low-quality score.

Besides the distortion and the dispersion of the simulated groups, we are also interested in evaluating the actual formations that the group members employ during the simulation. Therefore, we also propose a *formation* measure. Similar to the distortion and the dispersion measures, we sample its action at times $T-1, T, T+1$. Each sample stores the configuration of the group expressed in its local coordinate system. We define the distance $D_A(A_Q, A_E)$ between the actions as the weighted sum of the distances between their samples. Note that the distance between a query and an example formation sample is computed as the sum of the Euclidean distances between the formation positions of the corresponding group members, clamped by an upper value of 10 meters.

We ran the crosswalk scenario using our two phase algorithm and then compared the behaviour of the simulated groups against the dense crowd database based on the formation measure (see Figures 8.14 and 8.15). We also simulated the same scenario,

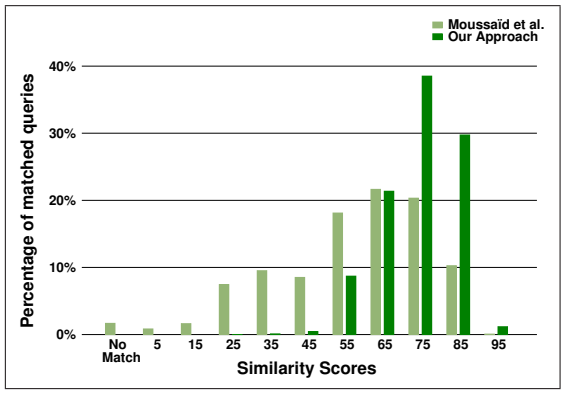


Figure 8.15: Comparison between our approach and the approach of Moussaïd *et al.* [148] when evaluating the crosswalk scenario using the formation measure.

using this time the method proposed by Moussaïd *et al.* [148]. Figure 8.15 shows the corresponding evaluation of the group formations. When compared to our approach, it is clear that the overall quality of the matches is lower. In particular, the distribution of the similarity scores is spread over the entire range of values, whereas a non-trivial amount of queries cannot be explained by the example database. A close inspection of the simulation shows that most of the low-quality matches correspond to problematic group behaviours, such as collisions or near collisions, congestion, abrupt avoidance maneuvers and lack of coherence. These behaviours significantly affect the spatial organization of the groups, which is reflected on the overall quality of the simulation. For example in Figure 8.14, bottom-right, the group that is highlighted in blue discs did not find any suitable match in the example database and received a 0 similarity score. The reason is that one of the group members, the young girl, got stuck behind two other pedestrian groups.

Finally, it must be noted that a number of false negative examples were also identified upon evaluating the crosswalk simulations. Such false negatives appear when groups have to avoid head-on collisions and are attributed to the fact that the example database cannot distinguish whether the configuration that a group adopts is due to an avoidance maneuver or due to social interactions (e.g. friends meeting and start chatting).

8.5.4 Performance

Our algorithm consists of two phases. In the first phase, an optimal new formation and velocity is determined for each group of agents, whereas in the second phase a collision-free velocity is obtained for every group member. Obviously, the running time of the first phase is significantly affected by the number of candidate formations F that we evaluate during each cycle of the simulation. As mentioned in Section 8.3, these formations are derived by linearly blending between the current formation of a group and its $k = N + 2$ template formations. Assuming μ intermediate formations between the current and each template formation, the total number of candidate formations that need to be evaluated per group is $\mu * k + k + 1$.

In general, increasing the number of intermediate formations, and consequently the total number of candidate formations, incurs an almost linear increase in the computation time of the first phase of our algorithm. Figure 8.16 shows the corresponding effect that μ has on the running time of 50 groups wandering through an environment ($100\,\mathrm{m} \times 100\,\mathrm{m}$) void of obstacles. As can be observed, the lower the μ, the better the performance. We have experimented with different test-case scenarios in a number of virtual environments

# Groups	# Agents	Group phase (msec/frames)	Member phase (msec/frames)	Total time (msec/frames)
100	256	9.17	1.41	10.58
200	510	21.04	3.15	24.19
300	751	33.18	4.92	38.10
400	1004	48.24	8.01	56.25
500	1253	64.44	9.43	73.87

Table 8.2: Performance of our approach for a varying number of small groups. The reported times are simulation only.

and found out that by considering three intermediate formations, our method generates smooth motions without imposing any significant overhead to the CPU usage.

To test the overall performance of our approach, we selected a varying number of groups consisting of 2-3 members and placed them randomly across the obstacle-free environment. Each group had to advance towards a random goal position avoiding collisions with the other moving groups; when it had reached its destination, a new goal was chosen. Table 8.2 summarizes the results we obtained for our benchmark scenario. The third column indicates the average time that is needed per simulation step to compute the first phase of our approach and the fourth column reports the average computation time of the second phase. Note that during the benchmark, the character rendering was disabled, since our goal was to analyze the motion planning cost of our algorithm.

As can be inferred from the table, the group planning phase dominates the overall performance of our algorithm. In contrast, computing the new velocities of the group members influences the running time of our approach marginally. This is expected, since most of the collisions are resolved in the first phase. In the second phase, the underlying agent-based method is only used to handle interactions between the group members, as well as imminent collisions between the group members and a limited number of neighbouring groups and agents.

The last column of the table shows that, in all of the test-cases, our method ran at interactive rates (ranging from 14 to 95 fps, depending on the crowd density). Since our current implementation is unoptimized and uses only one CPU thread for computing the motions of the agents, it is clear that our approach can be used for real-time simulation of small groups of characters.

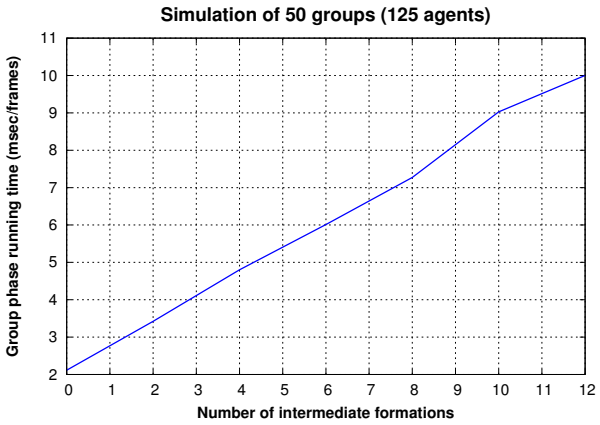


Figure 8.16: Average computation time of the group planning phase as a function of the number of intermediate evaluated formations.

8.6 Discussion

In this section, we elaborate on the parameters of our model. We also qualitatively compare our approach with prior work and discuss some of its limitations.

8.6.1 Model Parameters

The behaviour of each group depends on the weights of the three cost terms ($\kappa_1, \kappa_2, \kappa_3$) of the objective function shown in Equation 8.9. In our simulations, we set the default values to $\kappa_1 = 1.0$, $\kappa_2 = 0.5$, $\kappa_3 = 0.2$. In all of our test cases, these parameters generated the best agreement with the empirical observations, leading at the same time to visually convincing simulations. However, these parameters can be adapted to provide a wide variety of avoidance behaviours. In general, a high penalty for the deviation or time-to-collision cost term will force the group to adapt to a river-like formation, whereas a relatively high penalty for the deformation cost will make the group stick to its desired line-abreast formation.

A strong link also exists between the anticipated reactions of a group and the admissible orientation O_F and speed U_F domains of each candidate formation. As can be inferred from Figure 8.7, the maximum orientation deviation $\Delta\theta^{\max}(ttc)$ and thus, the O_F, is completely controlled by the user-defined parameters t_α, tc_{mid}, $tc_{\min}$, $d_{\max}$ and d_{mid}. The parameter t_α defines the maximum time that the group anticipates a collision. The threshold tc_{mid} regulates the start of the constant part of the function, whereas the $tc_{\min}$ threshold defines collisions that are imminent to the group. The parameter $d_{\max}$ determines the maximum angle that the group is allowed to deviate from its desired direction of motion. Finally, the parameter d_{mid} defines the deviation angle during the constant interval of the function. In general, the steeper the first part of $\Delta\theta^{\max}$ ($tc_{\text{mid}} \leq ttc < t_\alpha$), the earlier the anticipation. In addition, when the constant part of $\Delta\theta^{\max}$ is long, the orientation changes remain low and get stronger as ttc decreases, reaching to a peak when ttc tends to 0. Regarding the admissible speed domain (see Equation 8.5), the higher the $tc_{\min}$ threshold, the stronger the speed adaptation is. In all of our experiments, we set the default values to: $t_\alpha = 6\,\text{s}$, $tc_{\text{mid}} = 4\,\text{s}$, $tc_{\min} = 1.5\,\text{s}$, $d_{\max} = \pi/2$, and $d_{\text{mid}} = \pi/4$. These values were determined by the analysis of pedestrian interactions data in controlled scenarios. Finally, we refer the reader to Chapter 7 for the default parameters used in the second phase of our algorithm and their impact on the avoidance behaviour of the group members.

8.6.2 Comparisons

The flocking technique introduced by Reynolds [181] generates convincing motions, but is mainly applicable to model the collective behaviour of herds, flocks or schools. Later, Reynolds extended the flocking technique to incorporate additional steering behaviours, such as the leader-follower behaviour [182]. Based on his work, a considerable amount of research has emerged to model virtual crowds using simple local rules (see e.g [136, 165]). However, all these approaches are less suited for simulating small groups of virtual humans, since they do not take into account the way that friends or couples walk next to each other and how groups are perceived and avoided by other groups and walking individuals.

More recently, Singh *et al.* [198] proposed an intuitive method that enables agents to reach their destinations while maintaining a specific geometric formation. In their approach, safety plays an important role in the stability of the group formations. The

requirements, though, for controlling a small group of friends are not the same. In such groups, the communication between the group members significantly affects the configuration of the group. The dynamics and stability of group formations have also received a lot of attention from the robotics community and numerous approaches have been introduced (e.g. [6,34,124]). Although related to our research, these approaches are focused on different kind of applications, such as military applications and automated highway systems, where natural (human) behaviour is less important. Consequently, the evaluated formations are different from the formations adopted by pedestrian groups in real-life.

Our current research is closely related to the recent work of Moussaïd *et al.* [148], since it is elaborated from their empirical observations regarding the spatial organization of pedestrian groups. Using these observations, Moussaïd and his colleagues devised a social force model to account for small groups. Their approach generates macroscopically plausible behaviour. However, it lacks anticipation and prediction. Therefore, the groups interact when they get sufficiently close resulting in unnatural motions and oscillatory behaviours that become more obvious in large and cluttered environments. In contrast, our approach uses the velocity space to plan the avoidance maneuvers of the groups ensuring smooth and oscillation-free motions. Furthermore, rather than let group patterns emerge during the simulation, our method explicitly takes into account a number of empirically observed configurations. This not only provides a more robust crowd model, but also generates the correct microscopic (local) behaviour for the pedestrian groups.

Peters and Ennis [169] have also recently proposed a model to simulate plausible behaviours of small groups. Similar to our approach, their method takes into account the different spatial patterns observed in real pedestrian groups. Nevertheless, their approach is rule-based often requiring careful parameter tuning to obtain a desired simulation result. In contrast, our model automatically solves interactions between groups without the use of any explicit rules. The behaviour of each group depends only on the weights of the three cost terms of our objective function, as explained in Section 8.6.1.

Another approach to model pedestrian groups is the continuum crowd formulation [218] which unifies global planning and local collision avoidance into a single framework. Although this method exhibits emergent phenomena observed in real crowds, it is mainly suited for the simulation of a limited number of large homogeneous groups and is prohibitively computationally expensive for planning the movements of a large number of small groups that have distinct characteristics and goals.

More recently, Kwon *et al.* [114] have proposed an approach for synthesizing realistic group motions based on a mesh editing scheme. A similar idea was also employed in [210] to control the spatiotemporal arrangements of groups of multiple individuals in a number of applications. However, both approaches aim to govern the macroscopic behaviour of agent groups, whereas our goal is to simulate group interactions at the microscopic level. Additionally, the running times of both techniques are too high for real-time interactive applications. Finally, Ju *et al.* [97] have introduced an example-based approach that synthesizes virtual crowds from captured and simulated crowd data. Their approach can create, blend and combine groups of any size having arbitrary formations. However, it is primarily designed for generating group motions for offline crowd simulations and, hence, has a different goal as compared to our model.

8.6.3 Limitations

Although our algorithm can deal with dynamic and challenging scenarios by adapting the group formation, the final motions of the group members are dependent on the local avoidance mechanism of the second phase of our algorithm (see Section 8.4.2). Thus,

since no local method can absolutely guarantee collision-free motion, some collisions may occur in rather complex environments. It should also be noted that our method only aims to control the local interactions of pedestrian groups. As a result, a global path planning approach should be used to guide the global motions of the groups such that they cannot get trapped in local minima due to the presence of obstacles (see, e.g., [57, 118], as well as Chapter 3). It must also be pointed out that our method is not intended for simulating groups of characters in densely packed scenarios. In these scenarios, the focus is on the macroscopic simulation of the crowd, and thus, approaches based on continuum dynamics and/or fluid models should be used [151,218].

Finally, in this work, we focus on typical interactions among pedestrians and thus, we restrict ourselves to small pedestrian groups. However, we believe that our two-level approach is applicable to groups of any type and size. Assuming that a group can choose among a number of different configurations, in the first-phase, an optimal formation needs to be determined. Similar to Equation 8.9, a specific cost function can be used taking into account parameters such as the safety of the group, the preferred group formation, the communication of the group members, etc. In the second phase, an agent-based method can be employed to resolve collisions between the group members. Note that increasing the group size influences the performance of our approach marginally, since the running time of the group planning phase is dominated by the number of the evaluated formations rather than the size of the group. We can also trade off realism for further performance gain by planning the maneuvers of the groups using a geometrically-based avoidance model (e.g, [66,226]), instead of the data-driven approach presented in Section 8.3.

8.7 Conclusions

We presented a novel method for simulating the local motions of small groups of virtual humans. The intuition behind our approach is based on observed behaviour of real crowds; in real life, couples, friends or families tend to walk next to each other forming groups of two or three members. Similarly, our model considers pairs and triples of characters and uses a two-step algorithm to ensure that the groups will stay as coherent as possible while avoiding collisions with other groups, individuals and static obstacles. We have demonstrated the potential of our method through a wide range of test-case scenarios. We also evaluated our model using a number of quantitative quality metrics and validated our simulations with real-world data. In all of our benchmarks, our method was able to predict the emergence of empirically observed walking patterns generating smooth and visually convincing motions.

Currently, we are developing a tool to semi-automatically track pedestrians in outdoor environments. This will allow us to gain more insight into the avoidance behaviour of small groups of pedestrians and extend our model accordingly. Joining and splitting of pedestrian groups is also one of the research topics under investigation. In our current implementation, the group members stay as coherent as possible and only split up if a collision is imminent. However, in real-life, a group may split upon encountering another group or a small obstacle. For example, a group of pedestrians can easily move on both sides of a light pole, instead of trying to avoid it on the same side. Therefore, we would like to conduct an experimental study to better understand the decisions of the group members in such situations and further improve the avoidance model used in the second phase of our algorithm.

Chapter 9

Conclusion

In this thesis, we studied motion planning algorithms for crowd simulation purposes. In particular, we focused on real-time path planning techniques for computing the global motions of multiple virtual characters and on methods for simulating the local interactions between individuals as well as groups of characters. In the remainder of this concluding chapter, we summarize the main contributions of the thesis and discuss some ideas for future research.

9.1 Contributions

In the first part of the thesis, we introduced the Indicative Route Method (IRM) as a new path planning approach in interactive virtual worlds and games (Chapter 3). In the IRM, a so-called indicative route defines the preferred global route of the character, whereas a collision-free corridor around this route gives the character flexibility to locally adapt its motion when collisions with other characters and dynamic obstacles need to be avoided; while the character globally follows its indicative route to the goal, it can locally deviate from this route, as long as it stays within the corridor. Such a corridor is extracted from the Explicit Corridor Map (ECM) data structure, which stores a network of two-dimensional corridors along the edges of the medial axis of the environment. The final motion of the character inside the corridor is governed by a potential field approach resulting in a smooth path.

We have experimentally shown that the IRM can be used for real-time planning and navigation of thousands of characters. The method is fast, flexible and leads to believable paths. Additionally, it is very generally applicable and does not impose any limitation on the global behaviour of the characters, as indicative routes can be computed using any of the well-known methods in the motion planning literature or even specified manually by the user to encourage certain character behaviour. In Chapter 4, for example, we presented a simple A*-based algorithm that improves the indicative routes of the characters by taking the crowd density into account. In Chapter 2, we also devised three simple, yet effective, techniques for creating variants of homotopic paths that characters can follow when their indicative routes lie on the medial axis of the environment. Such variation not only provides a more challenging and less predictable opponent for the user in a (serious) game, but also enhances the realism of the simulations allowing characters to spread over the environment and thus, making the virtual worlds to look more lively and appealing.

We also applied the IRM to the multi-group path planning and navigation problem in Chapter 5. Here, we proposed a novel approach based on techniques from Integer Linear Programming to choreograph through space-time the indicative routes of large

heterogeneous groups of virtual humans. Our space-time planning formulation allows the IRM to identify and prevent deadlock situations, successfully resolve congestion and generate different macroscopic behaviours and natural looking motion patterns observed in real pedestrian crowds.

Besides demonstrating believable path planning behaviour, the virtual characters should also be able to move toward their desired locations in a human-like manner, resolving a bewildering amount of local interactions and avoiding collisions with each other. This problem is very challenging, since real humans exhibit behaviours of enormous complexity and subtlety making their simulation a rather difficult task. Therefore, in the second part of the thesis, we tried to address some of these challenges.

First, we proposed a predictive model based on (social) forces for solving interactions between virtual pedestrians that have converging trajectories (Chapter 6). Similar to real pedestrians, each virtual pedestrian anticipates possible future collisions with other pedestrians and then makes an efficient move to avoid them. Our technique ensures visually compelling simulations and smooth avoidance maneuvers that result in shorter and less curved paths as compared to previous solutions. It is extremely fast, simple to implement and improves the emergence of self-organized phenomena as we have demonstrated on a wide range of examples. As such, and given the current-state-of-the-art, we believe that there is hardly any room left for improvement in force-based local collision avoidance.

In Chapter 7, we also addressed the issue of realistic collision avoidance among virtual humans using a velocity-based model; at every simulation step, an optimal new velocity is returned for each character that minimizes the risk of collisions with other characters, the deviation from its desired velocity and the amount of effort that the character requires to adapt its motion. The derived model is elaborated from experimental interaction data between real pedestrians and is based on the simple hypothesis that pedestrians take early and energy-efficient actions to avoid collisions by slightly adapting their directions and/or speeds. Simulations created with our system run at interactive rates and lead to visually compelling motions. In addition, the velocity formulation provides better control over the avoidance behaviour of the characters compared to the force model in Chapter 6. Consequently, it is much easier to regulate the results of the simulations and generate desired crowd movements, especially when attempting to steer thousands of characters through the virtual world.

Finally, our last contribution was to extend the aforementioned model to simulate the walking behaviour of small groups of pedestrians. In particular, in Chapter 8, we introduced a novel algorithm to ensure that the group members will safely navigate toward their goals, forming at the same time walking patterns similar to the ones observed in real-life. We highlighted the potential of our method through challenging scenarios and evaluated the results from our simulations using visual and numerical comparisons with video footages of real pedestrian crowds.

9.2 Future Work

In this section, we discuss some overall directions for future work in the areas of global planning, local collision avoidance and crowd evaluation. We refer the reader to the concluding section of each chapter for a detailed discussion about how our proposed techniques can be extended and improved in the future.

Global Planning

We are confident that our proposed IRM framework has a lot of potential and can form the algorithmic basis for further research. In our current implementation, we assume that the simulated characters move on a plane or a terrain. Therefore, an obvious avenue for future work would be to extend our planner to two-and-a-half or three dimensional space. Preliminary results toward this direction look quite promising [231].

A future challenge would also be to address dynamic and changeable environments. In modern computer games and training applications, virtual worlds are highly dynamic and interactive; obstacles may move, appear and disappear within a simulation, paths can become partially or fully blocked due to the interaction of the user, whereas several objects can change their configurations over time (e.g. doors can open or close, traffic lights turn red or green, etc). However, in the IRM, we assume that the virtual environment is static and known. Consequently, each character decides about its global indicative route only once, that is before leaving from its start position. This may occasionally lead to unrealistic behaviours, e.g. characters waiting for a narrow passage to be freed whereas the nearby routes are less crowded or characters trying to follow paths that are not valid anymore. In contrast, in real-life, people constantly perceive the environment and may reconsider their path choices and replan their routes e.g. due to crowd density, safety reasons, limited or no accessibility.

As a result, we should also revise the IRM and devise an adaptive planning framework so that a virtual character can dynamically adapt its indicative route based on local information. Over the past twenty years, efficient algorithms have been proposed that are able to replan by repairing previous solutions instead of (re)planning from scratch [109, 134, 204]. Such approaches can be integrated into the IRM enabling our framework to replan indicative routes in real-time. Note, though, that in order to extract valid corridors around the indicative routes, the ECM data structure should also be updated in real-time when changes in the environment are observed. This can be achieved by locally repairing its underlying medial axis using incremental techniques (see, e.g., [62]).

Local Planning

Another area for improvement concerns the local interactions between virtual humans. In the two methods proposed in Chapters 6 and 7, as well as in current state-of-the-art techniques for local collision avoidance, e.g. [156, 173, 227], the animation is separated from path computation. Typically, a collision-free trajectory is locally planned for each virtual character (using physical forces or velocity vectors) and then, an animation is fitted along the trajectory using motion capture data. Nevertheless, such an approach cannot take into account the biped nature of real humans leading to unrealistic animations, since movements like sidesteps and backwards motions cannot be accurately performed. Another limitation is that existing local planners model humans as discs having a fixed radius. However, in real-life, people may "squeeze" themselves to cross, for example, an area of high crowd density. In such interaction scenarios, some physical contact can occur ("brushing" of shoulders), which is usually resolved by slightly turning the shoulder and performing an almost imperceptible sidestep [240]. In contrast, in a simulated crowd, a virtual human tries to avoid physical contact at all cost and hence, it will stop when a collision is imminent.

Based on these observations, we believe that a more realistic steering method can be obtained by planning a sequence of footsteps for a virtual human instead of computing

collision-free velocities or obstacle-avoidance forces (see, e.g., [20, 112, 200]). The generated footsteps can be followed exactly using existing animation techniques, like the ones proposed in [223, 224], resulting in a single framework that unifies local collision avoidance and motion synthesis. Such a footstep-driven approach can alleviate the problems mentioned above and increase the believability of the resulting animations, capturing a vast repertoire of movements that cannot be simulated by current local planners, e.g. side-stepping, backwards stepping, angling of upper body, etc. Consequently, in the future, we would like to extend/map several of the concepts for local collision avoidance introduced in this thesis to a footstep-based planner.

Crowd Evaluation

Throughout this thesis, we evaluated our proposed motion planning algorithms against a wide range of challenging scenarios and statistically compared them to existing techniques based on a number of quantitative quality metrics. Although such evaluation metrics provide a quick insight into the overall quality of a crowd model, they are not sufficient to fully and objectively assess the resulting simulations. In real-life, many parameters can affect the path choices that we make, such as the environment, the task we have to perform, our knowledge of the world, the presence and motion of other people, and possibly our emotional state. Hence, simply assigning a numerical value to a simulation (such as the average traveling time) cannot accommodate the different contexts that affect the decisions of the virtual humans. Consider, for example, an individual that has to thread through a dense crowd. Some collisions or near collisions may occur as the individual tries to reach its destination. However, the same type of behaviour at a low-density scenario is considered as highly unnatural.

To address these issues, data-driven evaluation techniques have also been explored in this thesis. In Chapter 7, for example, we compared the trajectories of simulated virtual characters to the ones obtained from motion captured data of real humans. However, such a validation protocol was restricted to simple interactions between pairs of virtual characters. To be able to evaluate multiple and complex character interactions and adapt the evaluation for different contexts, Lerner *et al.* [129] have recently proposed a data-driven approach that is based on video footages of real crowds. In their approach, crowd videos are analyzed and then used to create a database of examples. Each example represents a unique pedestrian behaviour and stores part of the pedestrian's trajectory along with the attributes that influence the motion of the pedestrian. Given a simulation, a similar analysis is performed on the trajectories of the virtual characters, resulting in a set of simulated queries that are compared to the observed examples. For each query, the most similar example in the database is retrieved based on a similarity measure and an evaluation score is returned providing a detailed assessment of the character's behaviour at that specific moment in time. In Chapter 8, we extended the work in [129] to evaluate the behaviour of small groups of virtual characters.

A disadvantage of the aforementioned evaluation techniques is the need to obtain the right crowd video, as well as to track the individual trajectories in it, which usually requires a lot of manual work. In addition, if the input video is not representative of the simulation and does not contain enough examples, several simulated behaviours that should be considered typical would possibly be identified as problematic. Despite these limitations, data-driven matching approaches provide a much more intuitive way to evaluate a crowd simulation, since they change the question from "how natural is a simulated crowd" to the easier one of "how similar is the simulated crowd to a real one". To date, these methods have relied on a small set of example behaviours. Therefore, an

obvious way to overcome their limitations is to obtain more example data, for instance by exploiting crowd videos from Internet websites. We can also explore existing computer vision algorithms to automatically track pedestrian trajectories over the obtained video sequences (see, e.g., [15]). In the future, we hope to address these challenges and develop a general framework for evaluating the quality of crowd simulation models.

Bibliography

[1] I. Altman. *The Environment and Social Behavior: Privacy, Personal Space, Territory, and Crowding*. Monterey, CA: Brooks/Cole Publishing Co., 1975.

[2] G. Arechavaleta, J-P. Laumond, H. Hicheur, and A. Berthoz. The nonholonomic nature of human locomotion: a modeling study. In *International Conference on Biomedical Robotics and Biomechatronics*, pages 158–163, 2006.

[3] O. Arikan and D.A. Forsyth. Interactive motion generation from examples. *ACM Transactions on Graphics*, 21(3):483–490, 2002.

[4] R.C. Arkin. Cooperation without communication: Multiagent schema-based robot navigation. *Journal of Robotic Systems*, 9(3):351–364, 1992.

[5] T. Balch and R.C. Arkin. Behavior-based formation control for multirobot teams. *IEEE Transactions on Robotics and Automation*, 14(6):926–939, 1998.

[6] T. Balch and M. Hybinette. Social potentials for scalable multi-robot formations. In *IEEE International Conference on Robotics and Automation*, pages 73–80, 2000.

[7] J. Barraquand, B. Langlois, and J.-C. Latombe. Numerical potential field techniques for robot path planning. *IEEE Transactions on Systems, Man, and Cybernetics*, 22:224–241, 1992.

[8] J. Barraquand and J.-C. Latombe. Robot motion planning: A distributed representation approach. *International Journal of Robotics Research*, 10:628–649, 1991.

[9] O.B. Bayazit, J.-M. Lien, and N.M. Amato. Better group behaviors in complex environments using global roadmaps. In *8th International Conference on Artificial life*, pages 362–370, 2003.

[10] M. Bennewitz, W. Burgard, and S. Thrun. Priority schemes for decoupled path planning techniques for teams of mobile robots. *Robotics and Autonomous System*, 41:89–99, 2002.

[11] D. Bertsimas and J.N. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, 1997.

[12] R. Boulic. Relaxed steering towards oriented region goals. In *Motion in Games*, volume 5277 of *Lecture Notes in Computer Science*, pages 176–187. Springer-Verlag, 2008.

[13] E. Bouvier and P. Guilloteau. Crowd simulation in immersive space management. In *Eurographics workshop on Virtual Environments and Scientific Visualization*, pages 104–110, 1996.

[14] A. Braun, S.R. Musse, L.P.L. de Oliveira, and B.E.J. Bodmann. Modeling individual behaviors in crowd simulation. In *Computer Animation and Social Agents*, pages 143–148, 2003.

[15] M.D. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. Van Gool. Online multiperson tracking-by-detection from a single, uncalibrated camera. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33:1820–1833, 2011.

[16] D.C. Brogan and J.K. Hodgins. Group behaviors for systems with significant dynamics. *Autonomous Robots*, 4:137–153, 1997.

[17] C. Burstedde, K. Klauck, A. Schadschneider, and J. Zittartz. Simulation of pedestrian dynamics using a two-dimensional cellular automaton. *Physica A: Statistical Mechanics and its Applications*, 295(3-4):507–525, 2001.

[18] M.E. Caplan and M. Goldman. Personal space violations as a function of height. *Journal of Social Psychology*, 114:167–171, 1981.

[19] S. Chenney. Flow tiles. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 233–242, 2004.

[20] J. Chestnutt, M. Lau, G. Cheung, J. Kuffner, J. Hodgins, and T. Kanade. Footstep planning for the Honda ASIMO humanoid. In *IEEE International Conference on Robotics and Automation*, pages 629–634, 2005.

[21] J.A. Cheyne and M.G. Efran. The effect of spatial and interpersonal variables on the invasion of group controlled territories. *Sociometry*, 35(3):477–489, 1972.

[22] F.Y. Chin, J. Snoeyink, and C.A. Wang. Finding the medial axis of a simple polygon in linear time. In *International Symposium on Algorithms and Computation*, pages 382–391, 1995.

[23] H. Choset, K.M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L.E. Kavraki, and S. Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, first edition, 2005.

[24] C.M. Clark, T. Bretl, and S. Rock. Applying kinodynamic randomized motion planning with a dynamic priority system to multi-robot space systems. In *IEEE Aerospace Conference*, pages 3621–3631, 2002.

[25] R.R. Clements and R.L. Hughes. Mathematical modelling of a mediaeval battle: the battle of Agincourt, 1415. *Mathematics and computers in simulation*, 64(2):259–269, 2004.

[26] M.B. Cluss, E.A. Crane, M.M. Gross, and B.L. Fredrickson. Effect of emotion on the kinematics of gait. In *American Society of Biomechanics*, 2006.

[27] J.S. Coleman and J. James. The equilibrium size distribution of freely-forming groups. *Sociometry*, 24(1):36–45, 1961.

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

[29] N. Courty and T. Corpetti. Crowd motion capture. *Computer Animation and Virtual Worlds*, 18(4-5):361–370, 2007.

[30] CPLEX 11.0. User's Manual. Technical report, ILOG SA, Gentilly, France, 2008.

[31] R.C. Dalton. The secret is to follow your nose: Route path selection and angularity. *Environment and Behavior*, 35(1):107–131, 2003.

[32] M. de Berg, M. van Kreveld, M.H. Overmars, and M. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin, Germany, 2000.

[33] Mark DeLoura. *Game Programming Gems*. Charles River Media, Inc., 2000.

[34] J.P. Desai, J.P. Ostrowski, and V. Kumar. Modeling and control of formations of nonholonomic mobile robots. *IEEE Transactions on Robotics and Automation*, 17(6):905, 2001.

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

[36] S. Dobbyn, J. Hamill, K. O'Conor, and C. O'Sullivan. Geopostors: a real-time geometry/impostor crowd rendering system. In *Symposium on Interactive 3D Graphics and Games*, pages 95–102, 2005.

[37] S. Dobbyn, R. McDonnell, L. Kavan, S. Collins, and C. O'Sullivan. Clothing the masses: Real-time clothed crowds with variation. *Eurographics Short Papers*, pages 103–106, 2006.

[38] F. Durupinar, J. Allbeck, N. Pelechano, and N. Badler. Creating crowd variation with the OCEAN personality model. In *7th International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1217–1220, 2008.

[39] M. Erdmann and T. Lozano-Pérez. On multiple moving objects. *Algorithmica*, 2:477–521, 1987.

[40] C. Ericson. *Real-time collision detection*. Morgan Kaufmann, 2005.

[41] B. Faverjon. Object level programming of industrial robots. In *IEEE International Conference on Robotics and Automation*, pages 1406–1412, 1986.

[42] Federal Highway Administration, U.S. Department of Transportation. *U.S. FHWA Manual on Uniform Traffic Control Devices*, 2003 with revisions 1 and 2 incorporated edition, 2003.

[43] D. Ferguson and A. Stentz. Field D*: An interpolation-based path planner and replanner. *International Symposium on Robotics Research*, pages 239–253, 2007.

[44] F. Feurtey. Simulating the collision avoidance of pedestrians. Master's thesis, University of Tokyo, 2000.

[45] P. Fiorini and Z. Shiller. Motion planning in dynamic environments using Velocity Obstacles. *International Journal of Robotics Research*, 17:760–772, 1998.

[46] L. Fleischer and M. Skutella. Quickest flows over time. *SIAM Journal on Computing*, 36(6):1600–1630, 2007.

[47] L.R. Ford Jr and D.R. Fulkerson. Constructing maximal dynamic flows from static flows. *Operations Research*, pages 419–433, 1958.

[48] L.R. Ford Jr and D.R. Fulkerson. A suggested computation for maximal multi-commodity network flows. *Management Science*, pages 97–101, 1958.

[49] L.R. Ford Jr and D.R. Fulkerson. *Flows in networks*. Princeton University Press, 1962.

[50] T. Fraichard. Trajectory planning in a dynamic workspace: A 'state-time' approach. *Advanced Robotics*, 13:75–94, 1999.

[51] S. Fritz and S. Carver. Accessibility as an important wilderness indicator: Modelling naismith's rule. In *GIS RESEARCH UK: 6th Annual Conference*, 1998.

[52] C. Fulgenzi, A. Spalanzani, and C. Laugier. Dynamic obstacle avoidance in uncertain environment combining PVOs and occupancy grid. In *IEEE International Conference on Robotics and Automation*, pages 1610–1616, 2007.

[53] J. Funge, X. Tu, and D. Terzopoulos. Cognitive modeling: knowledge, reasoning and planning for intelligent characters. In *26th annual conference on Computer graphics and interactive techniques*, pages 29–38, 1999.

[54] R. Gayle, A. Sud, M.C. Lin, and D. Manocha. Reactive deformation roadmaps: Motion planning of multiple robots in dynamic environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3777–3783, 2007.

[55] R. Geraerts. Planning short paths with clearance using explicit corridors. In *IEEE International Conference on Robotics and Automation*, pages 1997–2004, 2010.

[56] R. Geraerts, A. Kamphuis, I. Karamouzas, and M.H. Overmars. Using the Corridor Map Method for path planning for a large number of characters. In *Motion in Games*, volume 5277 of *Lecture Notes in Computer Science*, pages 11–22. Springer-Verlag, 2008.

[57] R. Geraerts and M.H. Overmars. The Corridor Map Method: A general framework for real-time high-quality path planning. *Computer Animation and Virtual Worlds*, 18:107–119, 2007.

[58] R. Geraerts and M.H. Overmars. Enhancing corridor maps for real-time path planning in virtual environments. In *Computer Animation and Social Agents*, pages 64–71, 2008.

[59] M. Gérin-Lajoie, C.L. Richards, and B.J. McFadyen. The negotiation of stationary and moving obstructions during walking: anticipatory locomotor adaptations and preservation of personal space. *Motor control*, 9(3):242–269, 2005.

[60] E. Goffman. *Relations in public : microstudies of the public order*. New York: Basic books, 1971.

[61] R. Golledge. Path selection and route preference in human navigation: A progress report. *Spatial Information Theory: A Theoretical Basis for GIS*, 988:207–222, 1995.

[62] P.J. Green and R. Sibson. Computing Dirichlet tessellations in the plane. *Computer Journal*, 21(2):168–173, 1978.

[63] M.M. Gross, E.A. Crane, and B.L. Fredrickson. Effort-shape and kinematic assessment of bodily expression of emotion during gait. *Human Movement Science*, 31(1):202–221, 2012.

[64] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer-Verlag, second edition, 1993.

[65] S.J. Guy, J. Chhugani, S. Curtis, D. Pradeep, M. Lin, and D. Manocha. PLEdestrians: A least-effort approach to crowd simulation. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 119–128, 2010.

[66] S.J. Guy, J. Chhugani, C. Kim, N. Satish, M.C. Lin, D. Manocha, and P. Dubey. Clearpath: highly parallel collision avoidance for multi-agent simulation. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 177–187, 2009.

[67] S.J. Guy, S. Kim, M.C. Lin, and D. Manocha. Simulating heterogeneous crowd behaviors using personality trait theory. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 43–52, 2011.

[68] A. Hall, S. Hippler, and M. Skutella. Multicommodity flows over time: Efficient algorithms and complexity. *Theoretical Computer Science*, 379:387–404, 2007.

[69] E.T. Hall. *The Hidden Dimension*. Garden City, N.Y.:Doubleday, 1966.

[70] D. Halperin, M.H. Overmars, and M. Sharir. Efficient motion planning for an L-shaped object. *SIAM Journal on Computing*, 21:1–23, 1992.

[71] J.J. Hartnett, K.G. Bailey, and C.S. Hartley. Body height, position, and sex as determinants of personal space. *Journal of psychology*, 87:129–136, 1974.

[72] L. Heigeas, A. Luciani, J. Thollot, and N. Castagné. A physically-based particle model of emergent crowd behaviors. In *Graphicon*, pages 5–10, 2003.

[73] D. Helbing. A fluid-dynamic model for the movement of pedestrians. *Complex Systems*, 6:391–415, 1992.

[74] 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.

[75] D. Helbing, L. Buzna, and T. Werner. Self-organized pedestrian crowd dynamics and design solutions. *Traffic Forum 12*, 2003.

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

[77] D. Helbing and P. Molnár. Social force model for pedestrian dynamics. *Physical Review E*, 51:4282–4286, 1995.

[78] D. Helbing and P. Molnár. Self-organization phenomena in pedestrian crowds. In *F. Schweitzer (Eds.), Self-organization of Complex Structures: From Individual to Collective Dynamics*, pages 569–577. Gordon and Breach, 1997.

[79] D. Helbing, P. Molnár, I.J. Farkas, and K. Bolay. Self-organizing pedestrian movement. *Environment and Planning B*, 28:361–384, 2001.

[80] L.F. Henderson. On the fluid mechanics of human crowd motion. *Transportation research*, 8(6):509–515, 1974.

[81] K.E. Hoff III, J. Keyser, M. Lin, D. Manocha, and T. Culver. Fast computation of generalized Voronoi diagrams using graphics hardware. In *26th annual conference on Computer graphics and interactive techniques*, pages 277–286, 1999.

[82] L. Hongwan, F.K. Wai, and C.H. Chor. A study of pedestrian flow using fluid dynamics. Technical report, National University of Singapore, 2003.

[83] S. Hoogendoorn and P.H.L. Bovy. Gas-kinetic modeling and simulation of pedestrian flows. *Transportation Research Record*, 1710:28–36, 2000.

[84] S. Hoogendoorn and P.H.L. Bovy. Normative pedestrian behaviour theory and modelling. In *Michael A.P. Taylor (Ed.), Transportation and traffic theory in the 21st century*, pages 219–245, 2002.

[85] S. Hoogendoorn and W. Daamen. Self-organization in pedestrian flow. In *S.P. Hoogendoorn, S. Luding, P.H.L. Bovy, M. Schreckenberg, D.E. Wolf (Eds.), Traffic and Granular Flow '03*, pages 373–382. Springer-Verlag, 2005.

[86] S.P. Hoogendoorn and P.H.L. Bovy. Pedestrian route-choice and activity scheduling theory and models. *Transportation Research Part B: Methodological*, 38(2):169–190, 2004.

[87] B.K.P. Horn. The curve of least energy. *ACM Transactions on Mathematical Software*, 9(4):441–460, 1983.

[88] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock. Randomized kinodynamic motion planning with moving obstacles. *International Journal of Robotics Research*, 21:233–255, 2002.

[89] S. Huerre, J. Lee, M. Lin, and C. O'Sullivan. Simulating believable crowd and group behaviors. In *ACM SIGGRAPH ASIA 2010 Courses*, pages 13:1–13:92, 2010.

[90] R.L. Hughes. A continuum theory for the flow of pedestrians. *Transportation Research Part B: Methodological*, 36(6):507–535, 2002.

[91] R.L. Hughes. The flow of human crowds. *Annual Review of Fluid Mechanics*, 35:169–182, 2003.

[92] V.T. Inman. Human locomotion. *Canadian Medical Association Journal*, 94(20):1047–1054, 1966.

[93] L. Jaillet and T. Siméon. A PRM-based motion planner for dynamically changing environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 2, pages 1606–1611, 2004.

[94] M.R. Jansen and N.R. Sturtevant. Direction maps for cooperative pathfinding. In *4th Artificial Intelligence and Interactive Digital Entertainment Conference*, pages 185–190, 2008.

[95] S.E.M. Jansen and H. van Welbergen. Methodologies for the user evaluation of the motion of virtual humans. In *International Conference on Intelligent Virtual Agents*, volume 5773 of *Lecture Notes in Artificial Intelligence*, pages 125–131. Springer, 2009.

[96] X. Jin, J. Xu, C.C.L. Wang, S. Huang, and J. Zhang. Interactive control of large-crowd navigation in virtual environments using vector fields. *IEEE Computer Graphics and Applications*, 28(6):37–46, 2008.

[97] E. Ju, M.G. Choi, M. Park, J. Lee, K.H. Lee, and S. Takahashi. Morphable crowds. *ACM Transactions on Graphics*, 29:140:1–140:10, 2010.

[98] M. Kallmann, H. Bieri, and D. Thalmann. Fully dynamic constrained delaunay triangulations. *Geometric Modelling for Scientific Visualization*, 3:74–123, 2003.

[99] A. Kamphuis and M.H. Overmars. Finding paths for coherent groups using clearance. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 19–28, 2004.

[100] M. Kapadia, S. Singh, W. Hewlett, and P. Faloutsos. Egocentric affordance fields in pedestrian steering. In *Symposium on Interactive 3D Graphics and Games*, pages 215–223, 2009.

[101] L.E. Kavraki, P. Švestka, J.-C. Latombe, and M.H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.

[102] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research*, 5:90–98, 1986.

[103] P. Khosla and R. Volpe. Superquadratic artificial potentials for obstacle avoidance and approach. In *IEEE International Conference on Robotics and Automation*, pages 1778–1784, 1988.

[104] B. Klinz and G.J. Woeginger. Minimum-cost dynamic flows: The series-parallel case. *Networks*, 43:153–162, 2004.

[105] B. Kluge and E. Prassler. Reflective navigation: Individual behaviors and group behaviors. In *IEEE International Conference on Robotics and Automation*, pages 4172–4177, 2004.

[106] H. Klüpfel, T. Meyer-König, J. Wahle, and M. Schreckenberg. Microscopic simulation of evacuation processes on passenger ships. In *4th International Conference on Cellular Automata for Research and Industry: Theoretical and Practical Issues on Cellular Automata*, pages 63–71. Springer-Verlag, 2000.

[107] R.L. Knoblauch, M.T. Pietrucha, and M. Nitzburg. Field studies of pedestrian walking speed and start-up time. *Transportation Research Record, No. 1538*, pages 27–38, 1996.

[108] D. Koditschek. Exact robot navigation by means of potential functions: Some topological considerations. In *IEEE International Conference on Robotics and Automation*, pages 1–6, 1987.

[109] S. Koenig and M. Likhachev. Improved fast replanning for robot navigation in unknown terrain. In *IEEE International Conference on Robotics and Automation*, pages 968–975, 2002.

[110] Y. Koren and J. Borenstein. Potential field methods and their inherent limitations for mobile robot navigation. In *IEEE International Conference on Robotics and Automation*, pages 1398–1404, 1991.

[111] L. Kovar, M. Gleicher, and F. Pighin. Motion graphs. *ACM Transactions on Graphics*, 21(3):473–482, 2002.

[112] J. Kuffner, K. Nishiwaki, S. Kagami, M. Inaba, and H. Inoue. Motion planning for humanoid robots. In *International Symposium of Robotics Research*, pages 365–374, 2005.

[113] J.J. Kuffner and S.M. LaValle. RRT-connect: An efficient approach to single-query path planning. In *IEEE International Conference on Robotics and Automation*, pages 995–1001, 2000.

[114] T. Kwon, K.H. Lee, J. Lee, and S. Takahashi. Group motion editing. *ACM Transactions on Graphics*, 27(3):1–8, 2008.

[115] T. Kwon and S.Y. Shin. A steering model for on-line locomotion synthesis. *Computer Animation and Virtual Worlds*, 18(4-5):463–472, 2007.

[116] M. Kyriakou and Y. Chrysanthou. Texture synthesis based simulation of secondary agents. In *Motion in Games*, volume 5277 of *Lecture Notes in Computer Science*, pages 1–10. Springer-Verlag, 2008.

[117] Y.C. Lai, S. Chenney, and S.H. Fan. Group motion graphs. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 281–290, 2005.

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

[119] J.-C. Latombe. *Robot Motion Planning*. Kluwer, 1991.

[120] M. Lau and J.J. Kuffner. Precomputed search trees: planning for interactive goal-driven animation. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 299–308, 2006.

[121] S.M. LaValle. *Planning algorithms*. Cambridge University Press, 2006.

[122] S.M. LaValle and J.J. Kuffner. Randomized kinodynamic planning. In *IEEE International Conference on Robotics and Automation*, pages 473–479, 1999.

[123] S.M. LaValle and J.J. Kuffner. Rapidly-exploring random trees: Progress and prospects. In *International Workshop on the Algorithmic Foundations of Robotics*, pages 293–308, 2000.

[124] J.R.T. Lawton, R.W. Beard, and B.J. Young. A decentralized approach to formation maneuvers. *IEEE Transactions on Robotics and Automation*, 19(6):933–941, 2003.

[125] J. Lee, J. Chai, P.S.A. Reitsma, J.K. Hodgins, and N.S. Pollard. Interactive control of avatars animated with human motion data. *ACM Transactions on Graphics*, 21(3):491–500, 2002.

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

[127] N.E. Leonard and E. Fiorelli. Virtual leaders, artificial potentials and coordinated control of groups. In *IEEE Conference on Decision and Control*, pages 2968–2973, 2001.

[128] A. Lerner, Y. Chrysanthou, and D. Lischinski. Crowds by example. *Computer Graphics Forum*, 26:655–664, 2007.

[129] A. Lerner, Y. Chrysanthou, A. Shamir, and D. Cohen-Or. Context-dependent crowd evaluation. *Computer Graphics Forum*, 29(7):2197 – 2206, 2010.

[130] D. Leven and M. Sharir. Planning a purely translational motion for a convex object in two-dimensional space using generalized voronoi diagrams. *Discrete and computational Geometry*, 2(1):9–31, 1987.

[131] P. Leven and S. Hutchinson. A framework for real-time path planning in changing environments. *International Journal of Robotics Research*, 21(12):999–1030, 2002.

[132] M.A. Lewis and K.H. Tan. High precision formation control of mobile robots using virtual structures. *Autonomous Robots*, 4(4):387–403, 1997.

[133] T.-Y. Li and H.-C. Chou. Motion planning for a crowd of robots. In *IEEE International Conference on Robotics and Automation*, pages 4215–4221, 2003.

[134] M. Likhachev, D. Ferguson, G. Gordon, A. Stentz, and S. Thrun. Anytime dynamic A*: An anytime, replanning algorithm. In *International Conference on Automated Planning and Scheduling*, pages 262–271, 2005.

[135] Y. Limon Duparcmeur, H. Herrmann, and J.P. Troadec. Spontaneous formation of vortex in a system of self motorised particles. *Journal de Physique I*, 5(9):1119–1128, 1995.

[136] C. Loscos, D. Marchal, and A. Meyer. Intuitive crowd behaviour in dense urban environments using local laws. *Theory and Practice of Computer Graphics*, 2003.

[137] G.G. Lovas. Modeling and simulation of pedestrian traffic flow. *Transportation Research Part B: Methodological*, 28(6):429–443, 1994.

[138] Massive Software - Simulating Life. `http://www.massivesoftware.com/` (accessed 2012).

[139] M. Mataric. Minimizing complexity in controlling a mobile robot population. In *IEEE International Conference on Robotics and Automation*, pages 835–835, 1992.

[140] M.J. Mataric. Designing emergent behaviors: From local interactions to collective intelligence. In *International Conference on Simulation of Adaptive Behavior: From Animals to Animats 2*, pages 432–441, 1992.

[141] R. McDonnell, M. Larkin, S. Dobbyn, S. Collins, and C. O'Sullivan. Clone attack! perception of crowd variety. *ACM Transactions on Graphics*, 27(3):26:1–26:8, 2008.

[142] R. McDonnell, M. Larkin, B. Hernández, I. Rudomin, and C. O'Sullivan. Eye-catching crowds: saliency based selective variation. *ACM Transactions on Graphics*, 28(3):55:1–55:10, 2009.

[143] MetaVR Inc. MetaVR real-time PC-based 3D visual simulation. `http://www.metavr.com` (accessed 2012).

[144] J.S. Milazzo, N.M. Rouphail, J.E. Hummer, and D.P. Allen. Effect of pedestrians on capacity of signalized intersections. *Transportation Research Record*, 1646:37–46, 1998.

[145] M. Mononen. Recast navigation: Navigation-mesh construction toolset for games. Google Project: `http://code.google.com/p/recastnavigation` (accessed 2012).

[146] J.M. Montepare, S.B. Goldstein, and A. Clausen. The identification of emotions from gait information. *Journal of Nonverbal Behavior*, 11(1):33–42, 1987.

[147] M. Moussaïd, D. Helbing, S. Garnier, A. Johansson, M. Combe, and G. Theraulaz. Experimental study of the behavioural mechanisms underlying self-organization in human crowds. *Proceedings of the Royal Society B: Biological Sciences*, 276(1668):2755–2762, 2009.

[148] M. Moussaïd, N. Perozo, S. Garnier, D. Helbing, and G. Theraulaz. The walking behaviour of pedestrian social groups and its impact on crowd dynamics. *PLoS ONE*, 5(4):e10047, 2010.

[149] S.R. Musse, C.R. Jung, J. Jacques Jr, and A. Braun. Using computer vision to simulate the motion of virtual agents. *Computer Animation and Virtual Worlds*, 18(2):83–93, 2007.

[150] S.R. Musse and D. Thalmann. A model of human crowd behavior: Group interrelationship and collision detection analysis. In *Computer Animation and Simulation*, pages 39–51, 1997.

[151] R. Narain, A. Golas, S. Curtis, and M.C. Lin. Aggregate dynamics for dense crowd simulation. *ACM Transactions on Graphics*, 28:1–8, 2009.

[152] F.P.D. Navin and R.J. Wheeler. Pedestrian flow characteristics. *Traffic Engineering*, 39:30–36, 1969.

[153] D. Nieuwenhuisen, A. Kamphuis, and M.H. Overmars. High quality navigation in computer games. *Science of Computer Programming*, 67(1):91–104, 2007.

[154] N.J. Nilsson. A mobius automation: an application of artificial intelligence techniques. In *1st International Joint Conference on Artificial Intelligence*, pages 509–520, 1969.

[155] C. Ó'Dúnlaing, M. Sharir, and C.K. Yap. Retraction: A new approach to motion planning. In *ACM Symposium on Theory of Computing*, pages 207–220, 1983.

[156] J. Ondřej, J. Pettré, A.-H. Olivier, and S. Donikian. A synthetic-vision based steering approach for crowd simulation. *ACM Transactions on Graphics*, 29(4):1–9, 2010.

[157] Opensteer: Steering behaviors for autonomous characters. `http://opensteer.sourceforge.net/` (accessed 2012).

[158] M. Oshita and Y. Ogiwara. Sketch-based interface for crowd animation. In *10th International Symposium on Smart Graphics*, pages 253–262, 2009.

[159] M.H. Overmars. Point location in fat subdivisions. *Information Processing Letters*, 44:261–265, 1992.

[160] M.H. Overmars and P. Švestka. A probablisitic learning approach to motion planning. Technical Report UU-CS-1994-03, Department of Computer Science, Utrecht University, the Netherlands, 1994.

[161] S. Paris and S. Donikian. Activity-driven populace: a cognitive approach to crowd simulation. *IEEE Computer Graphics and Applications*, 29(4):34–43, 2009.

[162] S. Paris, S. Donikian, and N. Bonvalet. Environmental abstraction and path planning techniques for realistic crowd simulation. *Computer Animation and Virtual Worlds*, 17(3-4):325–335, 2006.

[163] S. Paris, J. Pettré, and S. Donikian. Pedestrian reactive navigation for crowd simulation: a predictive approach. *Computer Graphics Forum*, 26(3):665–674, 2007.

[164] S. Patil, J. van den Berg, S. Curtis, M.C. Lin, and D. Manocha. Directing crowd simulations using navigation fields. *IEEE Transactions on Visualization and Computer Graphics*, 17:244–254, 2011.

[165] N. Pelechano, J.M. Allbeck, and N.I. Badler. Controlling individual agents in high-density crowd simulation. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 99–108, 2007.

[166] 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.

[167] J. Peng and S. Akella. Coordinating multiple robots with kinodynamic constraints along specified paths. *International Journal of Robotics Research*, 24:295–310, 2005.

[168] K. Perlin. An image synthesizer. *Computer Graphics*, 19(3):287–296, 1985.

[169] C. Peters and C. Ennis. Modeling groups of plausible virtual pedestrians. *IEEE Computer Graphics and Applications*, 29(4):54–63, 2009.

[170] J. Pettré, P.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.

[171] J. Pettré, M. Kallmann, and M.C. Lin. Motion planning and autonomy for virtual humans. In *ACM SIGGRAPH 2008 classes*, pages 42:1–42:31, 2008.

[172] J. Pettré, 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*, pages 81–90, 2005.

[173] J. Pettré, J. Ondrej, A.-H. Olivier, A. Crétual, and S. Donikian. Experiment-based modeling, simulation and validation of interactions between virtual walkers. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 189–198, 2009.

[174] M. Pfetsch. Multicommodity Flows and Column Generation, Lecture Notes. Zuse Institute Berlin, 2006.

[175] S. Poole. *Trigger Happy: Videogames and the Entertainment Revolution*. Arcade Publishing, 2000.

[176] C. Prins. Multi-unit pathfinding and column generation. Master's thesis, Utrecht University, the Netherlands, 2010.

[177] F. Qiu and X. Hu. Modeling group structures in pedestrian crowd simulation. *Simulation Modelling Practice and Theory*, 18(2):190 – 205, 2010.

[178] S. Quinlan and O. Khatib. Elastic bands: Connecting path planning and control. In *IEEE International Conference on Robotics and Automation*, pages 802–807, 1993.

[179] S. Rabin. *AI Game Programming Wisdom 4*. Charles River Media, Inc., 2008.

[180] J.H. Reif and H. Wang. Social potential fields: A distributed behavioral control for autonomous robots. *Robotics and Autonomous Systems*, 27(3):171–194, 1999.

[181] C.W. Reynolds. Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics*, 21(4):24–34, 1987.

[182] C.W. Reynolds. Steering behaviors for autonomous characters. In *Game Developers Conference*, pages 763–782, 1999.

[183] C.W. Reynolds. Interaction with groups of autonomous characters. In *Game Developers Conference*, pages 449–460, 2000.

[184] E. Rimon and D.E. Koditschek. Exact robot navigation using artificial potential fields. *IEEE Transactions on Robotics and Automation*, 8:501–518, 1992.

[185] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 1994.

[186] RVO Library v1.1. `http://gamma.cs.unc.edu/RVO/` (accessed 2012).

[187] A. Safonova and J.K. Hodgins. Construction and optimal search of interpolated motion graphs. *ACM Transactions on Graphics*, 26(3):106, 2007.

[188] G. Sánchez and J.-C. Latombe. Using a PRM planner to compare centralized and decoupled planning for multi-robot systems. In *IEEE International Conference Robotics and Automation*, 2002.

[189] M. Schreckenberg and S.D. Sharma (Eds.). *Pedestrian and evacuation dynamics*. Springer, 2001.

[190] J.T. Schwartz and M. Sharir. On the piano movers' problem: I. The case of a two-dimensional rigid polygonal body moving amidst polygonal barriers. *Communications on Pure and Applied Mathematics*, 36:345–398, 1983.

[191] J.T. Schwartz and M. Sharir. On the piano movers' problem: II. General techniques for computing topological properties of real algebraic manifolds. *Advances in Applied Mathematics*, 4:298–351, 1983.

[192] J.T. Schwartz and M. Sharir. On the piano movers' problem: III. Coordinating the motion of several independent bodies: The special case of circular bodies moving amidst polygonal obstacles. *International Journal of Robotics Research*, 2:46–75, 1983.

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

[194] S. Shimoda, Y. Kuroda, and K. Iagnemma. Random motion to escape local minima potential field navigation of high speed unmanned ground vehicles on uneven terrain. In *IEEE International Conference on Robotics and Automation*, pages 2839–2844, 2005.

[195] D. Silver. Cooperative pathfinding. In *1st Artificial Intelligence and Interactive Digital Entertainment Conference*, pages 117–122, 2005.

[196] T. Simeon, S. Leroy, and J.-P. Laumond. Path coordination for multiple mobile robots: A resolution-complete algorithm. *IEEE Transactions on Robotics and Automation*, 18(1):42–49, 2002.

[197] S. Singh, M. Kapadia, P. Faloutsos, and G. Reinman. Steerbench: a benchmark suite for evaluating steering behaviors. *Computer Animation and Virtual Worlds*, 20(5-6):533–548, 2009.

[198] S. Singh, M. Kapadia, P. Faloutsos, and M. Schuerman. Situation agents: agent-based externalized steering logic. *Computer Animation and Virtual Worlds*, 21(3-4):267–276, 2010.

[199] S. Singh, M. Kapadia, B. Hewlett, G. Reinman, and P. Faloutsos. A modular framework for adaptive agent-based steering. In *Symposium on Interactive 3D Graphics and Games*, pages 141–150, 2011.

[200] S. Singh, M. Kapadia, G. Reinman, and P. Faloutsos. Footstep navigation for dynamic crowds. *Computer Animation and Virtual Worlds*, 22(2-3):151–158, 2011.

[201] J. Snape, J. van den Berg, Stephen J. Guy, and D. Manocha. Independent navigation of multiple mobile robots with hybrid reciprocal velocity obstacles. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5917–5922, 2009.

[202] R.S. Sobel and N. Lillith. Determinant of nonstationary personal space invasion. *Journal of social psychology*, 97:39–45, 1975.

[203] R. Sommer. *Personal Space: The Behavioral Basis of Design*. Prentice-Hall, Englewood Cliffs, 1969.

[204] A. Stentz. The focussed D* algorithm for real-time replanning. In *International Joint Conference on Artificial Intelligence*, volume 14, pages 1652–1659, 1995.

[205] G. Still. *Crowd Dynamics*. PhD thesis, University of Warwick, August 2000.

[206] P. Stucki. Obstacles in pedestrian simulations. Master's thesis, Swiss Federal Institute of Technology ETH, 2003.

[207] A. Sud, E. Andersen, S. Curtis, M.C. Lin, and D. Manocha. Real-time path planning for virtual agents in dynamic environments. In *IEEE Virtual Reality*, pages 91–98, 2007.

[208] A. Sud, R. Gayle, E. Andersen, S. Guy, M. Lin, and D. Manocha. Real-time navigation of independent agents using adaptive roadmaps. In *ACM symposium on Virtual Reality Software and Technology*, pages 99–106, 2007.

[209] M. Sung, L. Kovar, and M. Gleicher. Fast and accurate goal-directed motion synthesis for crowds. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 291–300, 2005.

[210] S. Takahashi, K. Yoshida, T. Kwon, K.H. Lee, J. Lee, and S.Y. Shin. Spectral-based group formation control. *Computer Graphics Forum*, 28(2):639–648, 2009.

[211] F. Tecchia, C. Loscos, and Y. Chrysanthou. Visualizing crowds in real-time. *Computer Graphics Forum*, 21(4):753–765, 2002.

[212] F. Tecchia, C. Loscos, R. Conroy, and Y. Chrysanthou. Agent behaviour simulator (ABS): A platform for urban behaviour development. *ACM/EG Game Technology Conference*, 2001.

[213] K. Teknomo. Application of microscopic pedestrian simulation model. *Transportation Research Part F: Traffic Psychology and Behaviour*, 9(1):15 – 27, 2006.

[214] D. Thalmann, P. de Heras Ciechomski, C. O'Sullivan, and S. Dobbyn. Populating Virtual Environments with Crowds. In *Eurographics 2006 Tutorial Notes*, volume 2, pages 869 – 964, 2006.

[215] D. Thalmann and S.R. Musse. *Crowd simulation*. Springer-Verlag New York Inc, 2007.

[216] The Transportation Association of Canada. *Manual of Uniform Traffic Control Devices for Canada*, 2002 updated edition, 2002.

[217] Transportation Research Board, National Research Council, Washington, D.C. *Highway Capacity Manual*, 2000.

[218] A. Treuille, S. Cooper, and Z. Popović. Continuum crowds. *ACM Transactions on Graphics*, 25(3):1160–1168, 2006.

[219] X. Tu and D. Terzopoulos. Artificial fishes: Physics, locomotion, perception, behavior. In *21st annual conference on Computer graphics and interactive techniques*, pages 43–50, 1994.

[220] B. Ulicny and D. Thalmann. Towards interactive real-time crowd behavior simulation. *Computer Graphics Forum*, 21(4):767–775, 2002.

[221] B.J.H. van Basten and A. Egges. Evaluating distance metrics for animation blending. In *4th International Conference on Foundations of Digital Games*, pages 199–206, 2009.

[222] B.J.H. van Basten, S.E. Jansen, and Ioannis Karamouzas. Exploiting motion capture to enhance avoidance behaviour in games. In *Motion in Games*, volume 5884 of *Lecture Notes in Computer Science*, pages 29–40. Springer-Verlag, 2009.

[223] B.J.H. van Basten, P. Peeters, and A. Egges. The step space: example-based footprint-driven motion synthesis. *Computer Animation and Virtual Worlds*, 21(3–4):433–441, 2010.

[224] M. van de Panne. From footprints to animation. *Computer Graphics Forum*, 16(4):211–223, 1997.

[225] M. van den Akker, R. Geraerts, H. Hoogeveen, and C. Prins. Path planning for groups using column generation. In *Motion in Games*, volume 6459 of *Lecture Notes in Computer Science*, pages 94–105. Springer, 2010.

[226] J. van den Berg, S.J. Guy, M.C. Lin, and Dinesh Manocha. Reciprocal n-body collision avoidance. In *International Symposium of Robotics Research*, pages 3–19, 2009.

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

[228] J. van den Berg and M.H. Overmars. Prioritized motion planning for multiple robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2217–2222, 2005.

[229] J. van den Berg and M.H. Overmars. Roadmap-based motion planning in dynamic environments. *IEEE Transactions on Robotics and Automation*, 21:885–897, 2005.

[230] J. van den Berg, S. Patil, J. Sewall, D. Manocha, and M. Lin. Interactive navigation of multiple agents in crowded environments. In *Symposium on Interactive 3D Graphics and Games*, pages 139–147, 2008.

[231] W. van Toll, A.F. Cook IV, and R. Geraerts. Navigation meshes for realistic multi-layered environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3526–3532, 2011.

[232] L. Verlet. Computer Experiments on Classical Fluids. I. Thermodynamical Properties of Lennard-Jones Molecules. *Physical Review*, 159(1):98–103, July 1967.

[233] J. Vleugels and M. Overmars. Approximating Voronoi diagrams of convex sites in any dimension. *International Journal of Computational Geometry and Applications*, 8(2):201–222, 1998.

[234] P.K.C. Wang. Navigation strategies for multiple autonomous mobile robots moving in formation. *Journal of Robotic Systems*, 8(2):177–195, 1991.

[235] Biwi walking pedestrians dataset. `http://www.vision.ee.ethz.ch/datasets/index.en.html` (accessed 2012).

[236] Crowd data. `http://graphics.cs.ucy.ac.cy/research/downloads/crowd-data` (accessed 2012).

[237] R. Wein, J. van den Berg, and D. Halperin. The Visibility-Voronoi complex and its applications. In *Annual Symposium on Computational Geometry*, pages 63–72, 2005.

[238] M. Whittle. *Gait analysis: an introduction*. Elsevier, 2007.

[239] D. Wilkie, J. van den Berg, and D. Manocha. Generalized velocity obstacles. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5573–5578, 2009.

[240] M. Wolff. Notes on the behaviour of pedestrians. *People in places: The sociology of the familiar*, pages 35–48, 1973.

[241] S. Wolfram. Statistical mechanics of cellular automata. *Reviews of Modern Physics*, 55(3):601–644, 1983.

[242] S. Wolfram. *Cellular automata and complexity. Collected papers*. Reading, MA: Addison-Wesley Publishing Company, 1994.

[243] L.A. Wolsey. *Integer programming*. Wiley, New York, 1998.

[244] J. Wu and Z. Popović. Realistic modeling of bird flight animations. *ACM Transactions on Graphics*, 22(3):895, 2003.

[245] Y. Yang and Brock O. Elastic roadmaps: Globally task-consistent motion for autonomous mobile manipulation in dynamic environments. In *Robotics: Science and Systems*, 2006.

[246] H. Yeh, S. Curtis, S. Patil, J. van den Berg, D. Manocha, and M. Lin. Composite agents. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 39–47, 2008.

[247] B. Yersin, J. Maïm, P. Ciechomski, S. Schertenleib, and D. Thalmann. Steering a virtual crowd based on a semantically augmented navigation graph. In *First International Workshop on Crowd Simulation*, pages 169–178, 2005.

[248] B. Yersin, J. Maïm, F. Morini, and D. Thalmann. Real-time crowd motion planning. *The Visual Computer*, 24(10):859–870, 2008.

[249] D. Zhu and J.-C. Latombe. New heuristic algorithms for efficient hierarchical path planning. *IEEE Transactions on Robotics and Automation*, 7:9–20, 1991.

[250] G.K. Zipf. *Human Behavior and the Principle of Least Effort*. Addison-Wesley, 1949.

Publications

The chapters of this thesis are based on the following publications:

Chapter 2

I. Karamouzas and M. H. Overmars. Adding variation to path planning. *Computer Animation and Virtual Worlds*, 19(3-4):283–293, 2008.

Chapter 3

I. Karamouzas, R. Geraerts, and M. Overmars. Indicative routes for path planning and crowd simulation. In *FDG '09: Proceedings of the 4th International Conference on Foundations of Digital Games*, pages 113–120, 2009.

M. Overmars, I. Karamouzas, and R. Geraerts. Flexible path planning using corridor maps. In *ESA '08: Proceedings of the 16th annual European Symposium on Algorithms*, volume 5193 of *Lecture Notes in Computer Science*, pages 1–12. Springer-Verlag, 2008.

Chapter 4

I. Karamouzas, J. Bakker, and M. H. Overmars. Density constraints for crowd simulation. In *ICE-GIC '09: Proceedings of IEEE Games Innovation Conference*, pages 160–168, 2009.

Chapter 5

I. Karamouzas, R. Geraerts, and A. F. van der Stappen. Space-time Group Motion Planning. In *WAFR 2012: The Tenth International Workshop on the Algorithmic Foundations of Robotics*, to appear, 2012.

Chapter 6

I. Karamouzas, P. Heil, P. van Beek, and M. H. Overmars. A predictive collision avoidance model for pedestrian simulation. In *MIG '09: Proceedings of Motion in Games*, volume 5884 of *Lecture Notes in Computer Science*, pages 41–52. Springer-Verlag, 2009.

Chapter 7

I. Karamouzas and M. Overmars. Simulating human collision avoidance using a velocity-based approach. In *VRIPHYS '10: The 7th EG Workshop on Virtual Reality Interaction and*

Physical Simulation, pages 125–134, 2010.

I. Karamouzas and M. Overmars. A velocity-based approach for simulating human collision avoidance. In *IVA '10: Proceedings of 10th International Conference on Intelligent Virtual Agents*, volume 6356 of *Lecture Notes in Artificial Intelligence*, pages 180–186, 2010.

Chapter 8

I. Karamouzas and M. Overmars. Simulating and evaluating the local behavior of small pedestrian groups. *IEEE Transactions on Visualization and Computer Graphics*, 18(3):394–406, 2012.

I. Karamouzas and M. Overmars. Simulating the local behaviour of small pedestrian groups. In *VRST '10: Proceedings of the 17th ACM Symposium on Virtual Reality Software and Technology*, pages 183–190, 2010.

Samenvatting

Mensenmenigten zijn alomtegenwoordig in de echte wereld en het simuleren van menigten wordt in ons dagelijks leven steeds belangrijker: computergames, animatiefilms, maar ook serieuze toepassingen zoals menselijke analyse, training en opleiding zijn slechts enkele van de gebieden waarin virtuele karakters die een menigte simuleren vaak worden gebruikt. Een van de fundamentele taken die deze karakters moeten uitvoeren is enerzijds het plannen van hun paden tussen verschillende locaties in de virtuele wereld, en anderzijds het op een mensachtige manier naar een gewenste locatie bewegen, zonder met elkaar en met de omgeving te botsen. Dit is het het centrale thema van dit proefschrift.

Het eerste deel van dit proefschrift richt zich op de planning van paden van hoge kwaliteit in interactieve virtuele werelden. Eerst presenteren we drie eenvoudige doch effectieve methoden voor het creëren van varianten van homotopic paden, die een karakter afhankelijk van zijn start- en eindpositie kan volgen (hoofdstuk 2). Dit zorgt niet alleen voor een meer uitdagende en minder voorspelbare tegenstander voor de gebruiker in een (serieus) spel, maar verhoogt ook het realisme van de game-ervaring en training. Een individu neemt in het echte leven ook niet altijd precies dezelfde route bij het oplossen van een typisch probleem in de padplanning. Bovendien kunnen verschillende personen enigszins verschillende routes volgen, zelfs als hun start- en eindposities vrijwel gelijk zijn. Om geloofwaardig en aanvaardbaar te zijn, moeten virtuele personages dus ook vergelijkbaar gedrag vertonen. We tonen aan dat dergelijke padvariatie kan worden verkregen door het plaatselijk wijzigen van paden langs de mediale as van de omgeving. Er worden drie verschillende benaderingen voorgesteld. De eerste maakt gebruik van een coherente geluidsfunctie om "gecontroleerde" willekeurige paden te creëren. De tweede techniek creëert virtuele banen die karakters kunnen volgen om hun bestemmingen te bereiken, terwijl de laatste benadering een karakter toestaat bochten af te snijden en een kortere en meer directe pad naar zijn doel te volgen.

In hoofdstuk 3 introduceren we de *Indicative Route Method* (IRM) als een nieuwe aanpak in de padplanning in interactieve virtuele werelden en games. In de IRM stuurt een *indicatieve route* de algemene beweging van het karakter aan, terwijl een potentiaalveld benadering wordt gebruikt om het karakter lokaal door een *botsingsvrije corridor* om deze route heen te leiden, wat resulteert in een C^1-continuous pad. De corridor biedt de karakters voldoende lokale flexibiliteit wanneer dynamische obstakels (of andere bewegende figuren) moeten worden vermeden; terwijl het karakter globaal zijn indicatieve route naar het doel volgt, kan het lokaal van de route afwijken zolang het binnen de corridor blijft. Zoals we zullen aantonen, kan de IRM goed worden gecombineerd met de bestaande schema's ter voorkoming van lokale botsingen waardoor real-time simulatie van duizenden karakters door complexe virtuele werelden mogelijk is. Deze methode

is snel, flexibel en leidt tot vlotte paden. Bovendien beperkt dit in vergelijking met de meeste bestaande planners het globale gedrag van de karakters niet. Indicatieve routes kunnen automatisch worden berekend met een van de welbekende werkwijzen uit de literatuur omtrent bewegingsplanning, of zelfs door de gebruiker handmatig worden berekend om bepaald gedrag van het karakter aan te sporen. In hoofdstuk 4, bijvoorbeeld, presenteren we een eenvoudig op A* gebaseerd algoritme dat de indicatieve routes die karakters het liefst volgen verbetert door rekening te houden met de densiteit van de menigte. Een dergelijke benadering van densiteit vermindert het aantal lokale interacties tussen de karakters, wat resulteert in tijd- en energie-efficinte bewegingen.

In het laatste hoofdstuk van deel I, passen we de IRM toe op het probleem van de multi-groep padplanning en navigatie. Hier stellen we een nieuwe aanpak voor die is gebaseerd op technieken uit de *Integer Linear Programming* om de indicatieve routes van heterogene groepen van virtuele karakters doorheen de ruimte-tijd te choreograferen. De berekende routes worden dan een input aan de IRM om de uiteindelijke paden van de karakters bepalen. Onze formulering van ruimte-tijdplanning stelt de IRM in staat impassesituaties te identificeren en voorkomen, met succes opstoppingen op te lossen en verschillend macroscopisch gedrag en natuurlijke bewegingspatronen te genereren. Over het geheel genomen stelt onze aanpak ons in staat om complexe planningsproblemen met een of meerdere groep(en) karakters in real-time op te lossen, zoals we aantonen door middel van een aantal uitdagende scenario's.

Naast het tonen van geloofwaardig gedrag wat betreft padplanning, moeten de virtuele karakters ook in staat zijn om hun bewegingen lokaal aan te passen en dynamische interacties met elkaar op een mensachtige wijze uit te voeren. Dit probleem is zeer uitdagend, omdat het gedrag van echte mensen enorme complex en subtiel is, wat het maken van een simulatie een vrij moeilijke taak maakt. Daarom hebben we in het tweede deel van dit proefschrift geprobeerd om een aantal van deze uitdagingen aan te pakken.

Als eerste presenteren we in hoofdstuk 6 een voorspellend model op basis van *(sociale) krachten* voor het oplossen van interacties tussen virtuele karakters die convergerende trajecten hebben. Net als echte mensen, anticipeert elk virtueel karakter op mogelijke toekomstige botsingen met andere karakters en maakt dan een efficinte beweging om deze te vermijden. Onze techniek zorgt voor visueel aantrekkelijke simulaties en vlotte uitwijkmanoeuvres, die leiden tot kortere en minder gebogen paden in vergelijking met eerdere oplossingen. Het is zeer snel, eenvoudig te implementeren en verbetert de opkomst van zelfgeorganiseerde verschijnselen, zoals we demonstreren met een breed scala aan voorbeelden.

In hoofdstuk 7 besteden we ook aandacht aan het probleem van real-time en natuurlijk ogend vermijdingsgedrag tussen virtuele karakters met behulp van een op *snelheid gebaseerd model*; bij elke simulatiestap wordt de optimale nieuwe snelheid teruggebracht voor elk karakter dat het risico van botsingen met andere karakters, de afwijking van zijn gewenste snelheid en de inspanning die het karakter moet leveren om zijn beweging aan te passen minimaliseert. Het afgeleide model wordt uitgewerkt op grond van de wisselwerking tussen echte mensen in gecontroleerde experimenten en stoelt op de eenvoudige hypothese dat individuen snelle en efficinte maatregelen treffen om botsingen te voorkomen door hun richting en snelheid iets aan te passen. Met ons systeem gecreëerde simulaties draaien op interactieve snelheden en leiden tot visueel overtuigende bewegingen. Verbeteringen ten opzichte van een van de meest populaire, op snelheid gebaseerde methoden, het Reciprocal Velocity Obstacles model, zijn ook merkbaar. Zoals we zullen aantonen is ons model in goede overeenstemming met onze experimentele interactiegegevens en legt het ook belangrijke opkomende verschijnselen uit

de voetgangersliteratuur bloot. Bovendien biedt de snelheidsformulering betere controle over het vermijdingsgedrag van karakters in vergelijking met het fysiek-gebaseerde model uit hoofdstuk 6. Daardoor is het veel gemakkelijker om de resultaten van de simulaties te reguleren en gewenste massabewegingen te genereren, vooral bij een poging om duizenden karakters door de virtuele wereld heen te sturen.

Tenslotte breiden we in hoofdstuk 8 ons eerder genoemde op snelheid gebaseerde model uit om het lokale gedrag van kleine groepen van karakters te simuleren. De idee achter onze aanpak is gebaseerd op waargenomen gedrag van echte menigten: in het echte leven loopt de meerderheid van de voetgangers niet alleen, maar in kleine groepjes van twee of drie personen, zoals koppels of vrienden die samen lopen. Ook ons model houdt rekening met paren en groepjes van drie karakters en maakt gebruik van een tweestaps algoritme om ervoor te zorgen dat de leden van de groep veilig in de richting van hun doel kunnen navigeren, terwijl zij looppatronen vormen die vergelijkbaar zijn met patronen die worden waargenomen in het echte leven. We wijzen op de mogelijkheden van onze methode door middel van een breed scala van uitdagende scenario's. We vergelijken onze techniek ook met bestaande oplossingen en evalueren de resultaten van onze simulaties met behulp van visuele en numerieke vergelijkingen met videomateriaal van echte menigten voetgangers. In alle experimenten kan ons algoritme het ontstaan van empirisch vastgestelde groepspatronen die vlotte en overtuigende bewegingen genereren voorspellen.

Acknowledgments

First of all, I would like to thank my promotor, Mark Overmars, who was also my daily supervisor during the first three years of my PhD. Mark, your expertise and sharp vision was invaluable to me in addressing several of the problems mentioned in this thesis and successfully completing my doctoral. I really appreciated our fruitful discussions, the freedom that you gave me to conduct research of my own interest and, mostly, the trust that you have placed in me all these years.

I would also like to express my sincere gratitude to Frank van der Stappen, who, due to Mark's leave of absence, had the difficult job of supervising me during the last year of my PhD. Frank, thanks for helping me with my thesis write-up and for your suggestions and contributions to my PhD work. I really enjoyed our talks about football, politics and, of course, research and I am grateful for all your efforts and advice.

Next, I would like to thank the members of the reading committee, Yiorgos Chrysanthou, Serge Hoogendoorn, Steve LaValle, Carol O'Sullivan and Remco Veltkamp for the time they devote to read my thesis and for their helpful comments.

Special thanks go to Ben van Basten, Sander Jansen and Saskia Groenewegen, who have been my office mates throughout my PhD. Ben and Sander, I really appreciated the fun that we had and the support we gave each other. Seriously, guys, I will always remember our drinking sessions at "The Basket" and the social events in Utrecht along with Thomas Geijtenbeek, Heinrich Kruger and the other members of our research group.

I also want to thank all of my colleagues at the department for the pleasant working environment, and Arjan Egges, Roland Geraerts and Arno Kamphuis, in particular, for the many interesting and insightful discussions and for the good times that we had during coffee breaks, meetings and conferences. Arno and Roland, thanks for all your help and your active participation in my research.

Finally, I would like to thank all of my friends and my family for their endless support.

Curriculum vitae

Ioannis Karamouzas was born on November 6, 1982 in Athens, Greece. After finishing high school in the city of Serres, he studied computer science at the University of Macedonia, Thessaloniki, Greece from where he graduated in 2004 with a B.Sc. in Applied Informatics. In the year of 2005, he obtained his M.Sc. degree in Advanced Computer Science from The University of Manchester, UK. Two years later, he moved in the Netherlands and started his Ph.D. in computer science at Utrecht University under the supervision of Prof. dr. Mark H. Overmars. In 2011, he completed his thesis there.