DENSE CROWDS OF VIRTUAL HUMANS

SYBREN A. STÜVEL

© 2016 Sybren Anton Stüvel
Cover design by Sybren A. Stüvel
ISBN: 978-90-393-6515-1

Typeset in GVIM and LaTeX
Printed in The Netherlands by Printservice Ede

This publication, *Dense Crowds of Virtual Humans* by Sybren A. Stüvel, is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License. To view a copy of this license, visit `http://creativecommons.org/licenses/by-nc-nd/4.0/`

DENSE CROWDS OF VIRTUAL HUMANS

DICHTE VIRTUELE MENSENMASSA'S

(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 20 april 2016 des middags te 12.45 uur

door

SYBREN ANTON STÜVEL

geboren op 26 april 1979
te Opmeer

Promotor:	Prof.dr. M. van Kreveld
Copromotoren:	Dr.ir. A. Egges Dr.ir. A.F. van der Stappen

Dit proefschrift werd mogelijk gemaakt met financiële steun van nationale projecten COMMIT/ en COMMANDS.

CONTENTS

1 Introduction 1
- 1.1 Definition of terms . . . 3
- 1.2 Thesis Outline . . . 3

2 Background 5
- 2.1 Coordinate systems . . . 5
- 2.2 Agents, characters, and crowds . . . 7
  - 2.2.1 Crowd agents . . . 7
  - 2.2.2 Character shape representation and control . . . 9
  - 2.2.3 Animation . . . 10
  - 2.2.4 Path planning . . . 13
  - 2.2.5 Path following . . . 14
- 2.3 PD Controllers . . . 15
- 2.4 Statistics . . . 16
  - 2.4.1 Linear regression . . . 16
  - 2.4.2 Binomial test to determine bias . . . 18

3 Hierarchical Structures for Collision Checking 21
- 3.1 Related Work . . . 23
- 3.2 Hierarchical structures for detecting collisions . . . 24
  - 3.2.1 Bounding Cylinder Hierarchy . . . 25
  - 3.2.2 Oriented Bounding Box tree . . . 27
- 3.3 Comparison . . . 28
- 3.4 Properties of the BCH . . . 31
  - 3.4.1 Residual distance . . . 32
  - 3.4.2 Represented volume . . . 34
- 3.5 Conclusion . . . 35

4 Perception of Collisions between Virtual Characters 39
- 4.1 Related work . . . 40
- 4.2 Experiment design . . . 42
- 4.3 Pilot experiment . . . 44
  - 4.3.1 Overview . . . 44
  - 4.3.2 Experiment . . . 47

4.3.3 Results 47
4.4 Main experiment 49
4.4.1 Overview and variables 49
4.4.2 Image generation and questionnaire 51
4.4.3 Results 52
4.5 Discussion 56
4.6 Conclusion 59

5 An Analysis of Manoeuvring in Dense Crowds 61
5.1 Related work 63
5.2 Experiment 64
5.2.1 Experiment design 65
5.2.2 Execution 67
5.2.3 Representation 68
5.3 Results 69
5.3.1 Modelling direction of movement 70
5.3.2 Capsule-shaped agent representations 75
5.4 Discussion and Conclusion 76

6 Torso Crowds 81
6.1 Related work 82
6.2 Setting and Problem Formulation 84
6.3 Active agents 86
6.3.1 Limited-horizon path planning 88
6.3.2 Torso rotation planning 91
6.3.3 Performing the planned movement 92
6.4 Passive agents 93
6.4.1 Space finding 94
6.4.2 Orientation finding 96
6.4.3 Avoidance of active agents 98
6.4.4 Turning desire into action 99
6.5 Walls, doors, and other obstacles 100
6.6 Results 101
6.6.1 Validation and parameter optimization 102
6.6.2 Examples and comparison with disc-based simulation . 106
6.7 Character animation 110
6.7.1 Lower body orientation estimation 110
6.7.2 Skeletal animation 111
6.8 Conclusion 112

7 Conclusion and Closing Remarks 115
7.1 Summary and contributions 115

7.2 Limitations and future directions 116

Bibliography **119**

Samenvatting **133**

Acknowledgements **137**

Publications **139**

Curriculum Vitae **141**

CHAPTER 1

INTRODUCTION

Crowd, noun:

A large number of persons gathered so closely together as to press upon or impede each other; a throng, a dense multitude.
(The earlier term from 13th c. was ‘press’.)

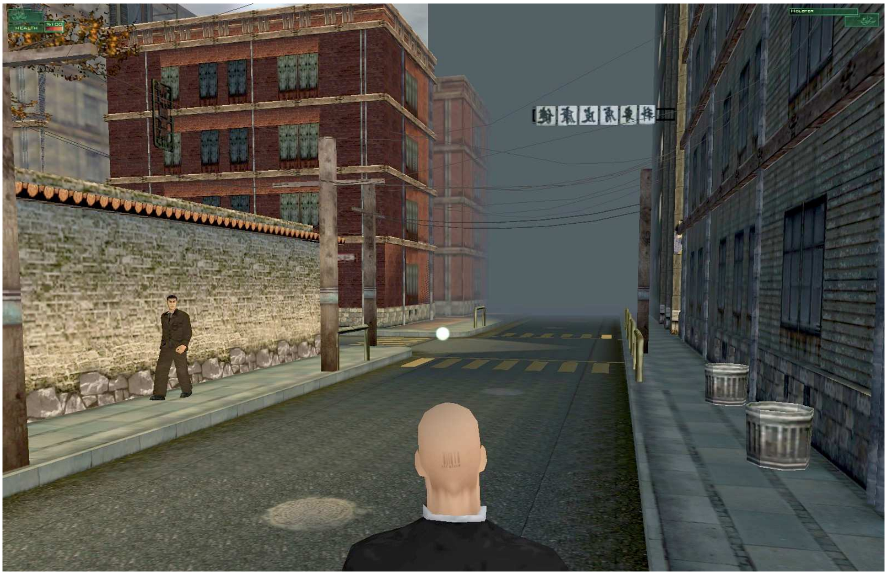
1623 Shakespeare & J. Fletcher Henry VIII iv. i. 58:

Among the Crowd i’ th’ Abbey, where a finger
Could not be wedg’d in more.

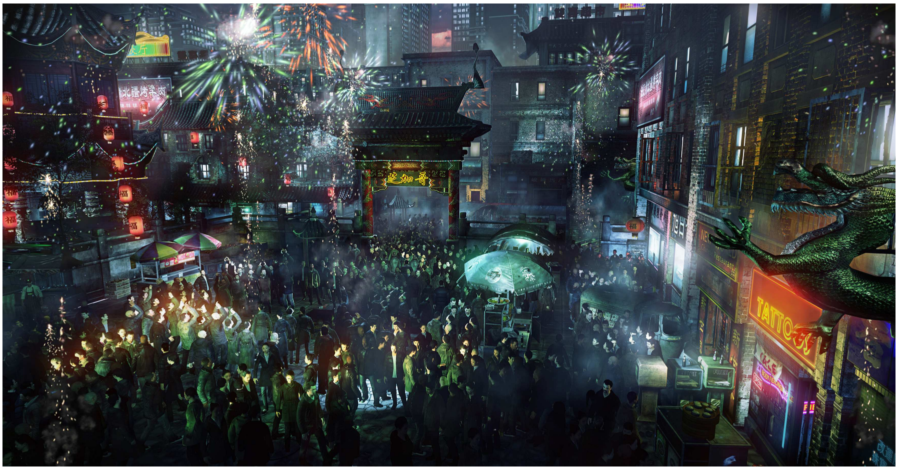
Source: Oxford English Dictionary

Throughout history, people have moved from the country into cities. The number of people in the world keeps growing, resulting in the highest local population densities of all time. Furthermore, the scale of events and venues such as football stadiums and concert halls increases. As a result, we see more, larger crowds of people; their safety and well-being becomes more important, while at the same time becoming more difficult to predict. Recent history has shown how quickly a situation can take a turn for the worse when a large crowd is involved. Examples are the crushing disasters at the Hillsborough stadium in 1989 and the Love Parade in Duisburg in 2010. Computer simulations of such events can help in various ways. Police, riot control and medical staff can use real-time crowd simulation for training purposes, and test the effectiveness of their crowd management strategies in a safe and controlled virtual environment. City council members and event planners can also use simulations to investigate crowd flow and find potentially dangerous bottlenecks. Even before construction starts, building designs can be evaluated for crowd flow and evacuatability. In these simulations, it is vital that the behaviour of the simulated crowd is representative of that of real people.

With developments in gaming hardware, realism of games has improved; characters have become more plausible, and it has become possible to populate virtual worlds with high numbers of such characters. The addition of background crowds to such games gives the player an increased sense of presence, and helps avoid the appearance of a ‘ghost town’. A good example can be seen when comparing the IO Interactive game *Hitman: Codename 47* (2000, Figure 1.1a) with the latest game in the series, *Hitman: Absolution* (2012,



(a) Hong Kong, as depicted in *Hitman: Agent 47* (2000).



(b) Chinatown in Chicago, as depicted in *Hitman: Absolution* (2012).

FIGURE 1.1: Two examples of city scenes in Hitman games. Images © IO Interactive.

Figure 1.1b). Where in the first game the city of Hong Kong, known for its very high population density [CoHK15], is depicted as empty and deserted, its eventual successor shows cities full of people, even though in reality those cities have a lower population density. Furthermore, in the latter game, crowds are a much more integrated component of the gameplay.

The main motivation for the research reported on in this thesis is the desire to increase realism in simulations of dense crowds. Most simulation methods use very simple, symmetrical shapes, such as discs or points, to represent members of the crowd. Their orientation is directly related to their velocity, and as a result characters rotate instantaneously. The common disc shape also results in non-humanoid behaviour in dense situations; when many agents are pressed into a small area, this shape becomes indirectly visible; their motion reminds of beer bottles pushing against each other in a factory. This led to the use of a capsule shape in Chapters 5 and 6, which also allows for more realistic motion by allowing side-stepping, walking backward, and twisting the torso to manoeuvre through narrow openings in the crowd.

Various metrics for comparing crowd simulation performance, most notably SteerBench [SKFR09], focus on the avoidance of collisions. Although such avoidance in itself is realistic, a simulation of a dense crowd that shows no collisions at all is not. This led us to produce a fast collision detection technique in Chapter 3. Since that technique employs an approximation scheme, we were also interested in the ability of people to recognise collisions, and to investigate how well the choices made in that chapter suit our human observation. This user study and its findings are presented in Chapter 4.

1.1 | DEFINITION OF TERMS

We use the words *participant*, *person* and *people* to mean actual human beings. The words *agent* and *crowd agent* are used interchangeably, and indicate an abstract representation of people, often associated with a simple shape (see Section 2.2.1). A *character model*, sometimes shortened to *character*, is a three-dimensional model of a human (see Section 2.2.2). The term *character animation* is understood as the movement of individual body parts of characters, such as walking, waving, falling down, et cetera, whereas *motion*, *movement*, or *manoeuvring* refers to the displacement of the entire body. The term *crowd animation* refers to the animation of characters in a crowd. It is used in contrast to *crowd simulation*, which is generally understood as the simulation of agent movement.

1.2 | THESIS OUTLINE

The remainder of this thesis is divided into six chapters, each covering a different aspect of dense crowd animation. Each chapter, except for Chapter 2, places the research it describes within the body of knowledge in science on that topic.

Chapter 2 provides fundamental background information on coordinate systems, distance metrics, notation, character and crowd animation, PID controllers, and statistical methods.

Chapter 3 discusses a fast collision detection strategy based on hierarchies of cylinders. As people in dense crowds have a high chance to collide, fast collision detections are important in such crowds. Single cylinders are often used to detect collisions between virtual characters; by refining the single cylinder using gradually smaller cylinders we detail this shape, and make the collision detection more precise.

Chapter 4 describes a user study into the ability of human observers to recognise collisions between virtual characters. We observe that people are generally good at recognising non-collisions, and less so at recognising collisions. Furthermore, we identify several parameters significant to this recognition.

Chapter 5 describes an experiment, with a real crowd of 23 participants in a motion capture studio, to obtain information on the crowd manoeuvring behaviour of people. We observe that a generalized Voronoi diagram of the participants can be used to predict the direction in which they manoeuvre through the crowd.

Chapter 6 describes a crowd simulation method aimed at the simulation of dense crowds, and is based on the results of the user studies. It introduces a capsule shape to represent people in the crowd, and supports different behaviour for actively manoeuvring people and people that have no incentive to move. It also discusses a character animation method for holonomic motion.

Chapter 7 concludes the thesis by providing an overview of the accomplished work. It also discusses limitations of the work, and covers possible avenues for future research.

CHAPTER 2

BACKGROUND

This chapter provides background knowledge for the remainder of this thesis. Section 2.1 discusses the choice of coordinate systems, distance metrics, and notation. Section 2.2 describes character animation and crowd simulation; simple abstract representations are discussed first, and then extended to a human shape, deformation, animation, and finally integration with crowd simulation systems. In Section 2.3, we give an overview of the history and use of the most-used feedback control mechanism, the PID controller, which we use in the crowd simulation system discussed in Chapter 6. Finally, Section 2.4 discusses linear regression, a statistical technique to fit a model to a system, which is used in Chapter 4 to interpret the result of a user study.

2.1 | COORDINATE SYSTEMS

In this section we discuss coordinate systems and distance metrics commonly used in the field of computer graphics. We live in an environment consisting of three spatial dimensions and one temporal dimension: space-time. In this section we discuss spatial coordinates, whereas in the next section we also look at the addition of time.

Any point in space can be uniquely addressed by a list of coordinates, one for each spatial dimension, which express the signed distance to a set of orthogonal coordinate axes. In three-dimensional space, these coordinates are usually given the letters x, y and z. Unfortunately, there are different interpretations of the directions denoted by these letters. It seems that everybody agrees on the 2D coordinate system of a monitor: x for left-right and y for up-down. However, the introduction of a third coordinate for displaying 3D scenes is still controversial. Arguing from the point of view that the third dimension expands on the 2D projection of the monitor, some people feel that z should indicate the in-out dimension. As a result, the ground plane of their virtual environment is modelled as the xz plane, with the y direction still indicating up-down. In contrast, in the field of crowd simulation, computations are often performed on the ground plane; the height of the ground is then ignored. These 2D simulations use the first two available letters: x and y. Many other real-life problems are also solved by projection of the 3D world onto the ground plane, whereas few problems are solved by projecting onto the remaining two planes. For those reasons, in this thesis, we denote the ground plane as xy, and use z for the up-down axis.

To differentiate between the individual coordinates (or just any scalar) $x \in \mathbb{R}$ and the vector $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$, we denote vectors with a bold symbol. The differential of a scalar or vector is denoted with a dot, such as $\dot{x}$ or $\dot{\mathbf{x}}$, when it is clear from the context over which variable the differential is taken.

A general distance metric between two points $\mathbf{u}$ and $\mathbf{v}$ is given by the p-norm:

$$d_p(\mathbf{u}, \mathbf{v}) = ||\mathbf{v} - \mathbf{u}||_p = \left(\sum_{i=1}^{n} |v_i - u_i|^p \right)^{1/p} ,$$

where n is the number of dimensions. For $p = 1$ this gives the Manhattan distance, which is used as distance metric in those cellular automaton crowd simulations (see Section 2.2.5) where diagonal movement is not allowed. The special case $p = \infty$ results in the Chebyshev distance, which is the greatest difference between the coordinates along any of the dimensions. The most commonly used distance metric is the Euclidean distance obtained by $p = 2$, which we use in this thesis.

Regardless of the distance metric used, there are multiple ways to define the distance between objects. We define an object by the collection of points occupied by the object. As is common in the field of computer animation, in this thesis we assume that objects are closed, i.e. the boundary is part of the object. Given two objects $A, B \subset \mathbb{R}^n$, their distance is defined as the minimum distance between their points:

$$d(A, B) = \min \{ d(\mathbf{a}, \mathbf{b}) \mid \mathbf{a} \in A, \mathbf{b} \in B \}$$

This metric can also be used when the objects overlap, in which case $d(A, B) = 0$. Unfortunately, this does not provide any indication as to the amount of overlap. This is measured by the *penetration depth*, denoted as $\pi(A, B)$, which is defined as the minimum translation distance required to bring the pair in touching contact [AGHP$^+$00, vdB01]:

$$\pi(A, B) = \min \{ ||\mathbf{t}|| \mid \mathrm{int}(A + \mathbf{t}) \cap B = \emptyset, \mathbf{t} \in \mathbb{R}^n \}$$

Computing the penetration depth for non-convex objects, such as virtual humans, is a difficult problem. It is generally solved by decomposing the object into convex pieces, and finding their penetration depth [KOLM02, HTKG04]. The two metrics for object distance and penetration depth can be combined into a continuous distance function $d_c(A, B)$, by taking $d(A, B)$ when $A \cap B = \emptyset$, and $-\pi(A, B)$ otherwise. Finally, we can compute the Hausdorff distance [Hau14], which is defined as:

$$d_H(A, B) = \max \left\{ \max_{\mathbf{a} \in A} \min_{\mathbf{b} \in B} d(\mathbf{a}, \mathbf{b}), \max_{\mathbf{b} \in B} \min_{\mathbf{a} \in A} d(\mathbf{a}, \mathbf{b}) \right\}$$

Informally, the Hausdorff distance is the longest distance one can be forced to travel by an adversary who chooses a point in one of the two sets, from where you must travel to the other set. As such, if two objects have similar shape and placement, they will be close in the Hausdorff distance [ZZYS13]. For the humanoid shapes typically used in character animation, the opposite is also true, a property we use in Chapter 3.

2.2 | Agents, characters, and crowds

In this section we discuss various aspects of crowds of virtual characters. We look at the shape and behaviour of crowd members, mechanisms to deform those shapes, and methods to animate those deformations.

2.2.1 | Crowd agents

The term *agent* as used in computer science stems from the field of artificial intelligence, and, confusingly, has no clear definition. Nwana describes an agent as referring to a component of software and/or hardware which is capable of acting exactingly in order to accomplish tasks on behalf of its user [Nwa96]. In the author's view, agents are (or should be) disembodied bits of 'intelligence'[1] In this thesis we borrow this broad definition, and define a *crowd agent* as an abstract entity that can make steering and behavioural decisions on behalf of a character in the crowd. A crowd agent is mobile, in the sense that it has an associated placement in the virtual environment that can change over time. It engages in deliberative planning, in order to achieve coordination with other agents, and to reach a certain abstract goal. Even though there are crowd simulation systems that have explicit cooperation [KHHL12], usually coordination is limited to collision and congestion avoidance [vdBLM08, vTCG12].

A human body has many ways in which it can move, so called *degrees of freedom*. An adult human skeleton has an estimated total of 244 degrees of freedom [ZP12]. Many of these DoF can be redundant in certain situations; for example, when placing your hand at a fixed spot on a surface, when your shoulder stays in the same spot your elbow can still move around. However,

---

[1] The term 'intelligence' is explicitly left undefined by Nwana, but is understood to include the ability to learn.

this redundancy disappears when the goal is to place your elbow at a fixed spot. This redundancy and the large number of degrees of freedom make it hard to plan realistic full body motion for a single body, and even more difficult for an entire crowd. As described above, a crowd agent makes steering and behavioural decisions on behalf of a character. To increase computation performance, these decisions are generally based on a simplified representation of that character. Even though agents are abstract, disembodied entities, this representation is usually referred to as the agent's *shape*.

The character's behaviour is determined by the agent; the chosen shape determines the range of possible behaviours. Most crowd simulation systems represent characters as points (x_i, y_i) [Hen71, KS05, JCG13] or discs (x_i, y_i, r_i) [NGCL09, vdBGLM11, CGZM11] moving in the ground plane. The big advantage of these shapes are their simplicity. Due to the rotational symmetry, they have two positional DoFs (x, y); the disc's radius is generally not a DoF, as the agents do not change size. Since crowd simulation systems generally try to avoid collisions, distance checks are frequent, and discs allow for simple and fast distance computations. Some simulations extend the disc model by taking the character's height into account, resulting in a cylindrical representation. If this height is used for path planning, but the collision detection is still performed in the ground plane, the simulation is considered 2.5-dimensional. The disc and cylinder models are well suited for sparse crowds of forward-walking characters. However, when the crowd becomes dense, the agent shape becomes visible as characters slide against each other in a circular pattern.

As defined in standard textbooks [Cra68], a *holonomic* system is a system that has the same number of independent degrees of freedom as the number of generalised coordinates to locate the system [WA92]. The placement of a crowd agent on the ground plane is defined by three generalised coordinates[2]: two for the position and one for the orientation of the agent. Disc- and point-based crowd simulation methods generally assume that the orientation of the agent is determined by the linear velocity vector. The agent is thus always oriented along its path, which effectively removes a degree of freedom. Consequently, such systems are called *nonholonomic*.

Agent representations more complex than points and discs are used as well. Singh et al. use multiple discs [SKRF11] to represent a single agent, and plan their motion using a predictive footstep model. This produces a wider range of motions, including holonomic motion such as side-stepping.

Although the disc and cylinder models are used in the majority of crowd simulation systems, we observe that they actually are a rather bad match for a human shape in Chapter 3. Instead, we use a capsule-shaped agent in Chapters 5 and 6, which fits the human shape better than a disc, and allows the planning of side-stepping and backward-stepping motions.

---

[2] Here we assume that the crowd agent has a non-articulated, rigid shape.

2.2.2 | CHARACTER SHAPE REPRESENTATION AND CONTROL

Approaches to represent a shape in the virtual environment can be generally divided into two categories: a *boundary representation* and a *solid representation*. Suppose we would like to define a model of a planet. Using a boundary representation, we might write the equation of a sphere that roughly coincides with the planet's surface. Using a solid representation, we would describe the set of all points that are contained in the sphere[LaV06]. Solid representations are hardly used in the field of character modelling and animation, as the boundary is sufficient for most purposes, and gives more freedom to the modeler by allowing non-manifold objects.

Boundary representations can be divided further into algebraic and polyhedral models. *Algebraic* models use algebraic expressions, such as $\{\mathbf{x} \mid d(\mathbf{x}, \mathbf{c}) = r\}$ for a sphere around point $\mathbf{c}$ and radius r. Collision detection checks are trivial in such simple representations, but quickly become impossible to solve analytically when the expression becomes more complex. Other operations also become harder; for example, a cartoon style upper arm could be represented by an oblong ellipse, but modifying the algebraic formula to show the bulging of biceps is rather complex. *Polyhedral* representations work around these issues by approximating the character shape with connected small, planar, polygonal patches. Such a representation is called a *mesh*, and consists of three types of components: points in space called *vertices*, which are connected by *edges*, which in turn connect and delimit planar *faces*.

By manipulating the vertex coordinates, the model can be deformed. Although this gives ultimate control over the shape of the character, it is also hard to work with. Not only does it take effort to specify all positions of all vertices, it is also hard to ensure that limbs bend correctly at the joints, or that their length remains constant. Furthermore, the deformation cannot be easily transferred to other meshes, as it is dependent on the exact vertex structure.

Different techniques have been developed that allow describing a deformation in more abstract terms, or make it adaptable to different 3D objects. *Free-form deformation* [SP86], also known as lattice deformation, uses a limited set of control points around a three-dimensional model. By moving those control points, the enclosed model is deformed. This allows for rapid modelling of objects by deforming simple shapes, but still suffers from the same downsides as direct vertex manipulation; care has to be taken to correctly bend at joints or to keep limbs at the correct length. *Linear blend skinning* (LBS), also known as skeletal subspace deformation (see Figure 2.1), was established by Magnenat-Thalmann et al. [MTLT88]. Even though there are notable improvements to their method, such as dual quaternion skinning [KCvO07], LBS is still the most common technique seen in 3D software and game engines. In contrast to the animation of the character mesh itself, a skeleton-based

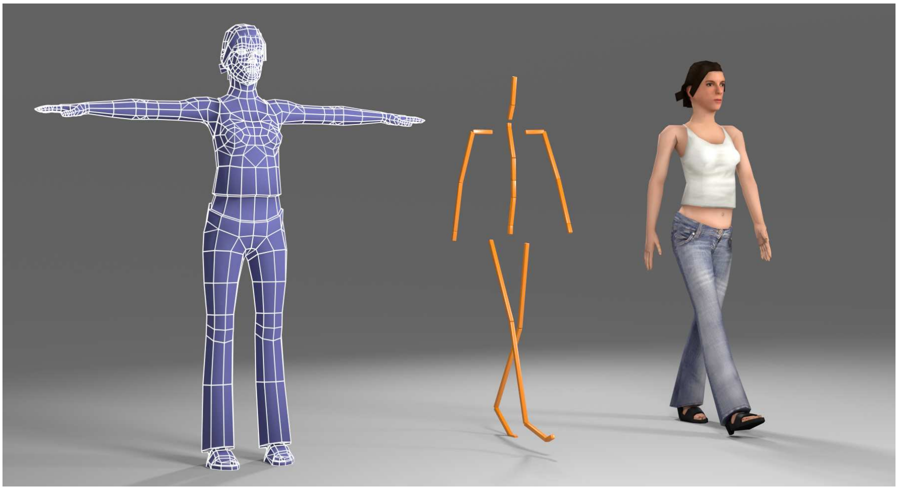


Figure 2.1: Mesh in rest pose (left), skeletal structure (middle), and the resulting deformed and textured mesh (right).

approach allows for model-independent animations, as well as fewer parameters to manipulate. Although the animation is model-independent, it is still dependent on the structure and proportions of the skeleton. Usually, this skeleton is constructed in a hierarchical fashion, with several branches for the limbs and head. The base of the hierarchy is called the *root joint*, and is usually placed in the centre of mass of the character. It also defines the position of the character in cases where a single point is required. The animation data can be adjusted to a skeleton of identical structure but different proportions using a retargeting technique [Gle98].

2.2.3 | Animation

The Oxford English Dictionary describes *animation* as ‘the action or process of imparting life, vitality, or (as a sign of life) motion’. In contrast, Parent foregoes the aspect of imparting life, and describes *computer animation* as any computer-based computation used in producing images intended to create the perception of motion [Par12]. In the field of computer graphics, animation is often seen as a synonym for just ‘motion graphics’, which is how the term is used in this thesis as well. Motion is obtained by introducing the concept of *time* to otherwise static graphics. In this section, we follow the description of animation by Egges [Egg06]. In its simplest form, a computer animation is represented as a set of values F, also called a *frame*, that can change over time: $A : t \mapsto F$. The continuous function A is defined for all timekeys $t \in [t_s, t_e] \subset \mathbb{R}$. In general, $A(t)$ is just a set of time-varying parameters.

In character animation these parameters control the placement and shape of the character. Its continuity is important, as it provides the character's shape at each moment in time. Since the number of timekeys in any non-empty interval is infinite, it is not possible to explicitly specify a frame F for each timekey. Instead, most animation systems rely on a discrete sequence of *keyframes*, each associated with a distinct timekey. To obtain the required continuity, an interpolation technique is used. In computer animation, the term 'keyframe' has been generalised to apply to any variable whose value is set at specific time keys, and from which values for the intermediate frames are interpolated according to some predescribed procedure [Par12]. The choice of interpolation technique depends on the representation of the keyframes, as well as the required *look* of the animation.

An important point to consider when one conceives an animation system, is its usability from the animator's point of view. When we look at the design process for animations, there are several commonly used techniques:

Manual approach: an animation is constructed from a set of keyframes, manually designed by an animator. Many Open Source and commercial packages (such as Blender [ble15] and Maya [may15]) are available that allow an animator to create keyframes. Since the animator has complete control over the resulting animation, this is a very popular approach. However, the manual creation of a convincing, life-like animation requires a lot of time and skill.

Pre-recorded animations: an animation is recorded using a motion capture or tracking system (such as Vicon [vic15] or MotionStar [mot15]). The MotionStar system uses magnetic tracking, which suffers much less from occlusion issues than the Vicon tracking system, which uses a multi-camera optical tracking approach. The Vicon system, however, has a higher spatial precision, which is why in many commercial applications, such as games or movies, an optical motion capture system is used to drive the character motions. A motion capture system can be an excellent time-saver, because a lot less manual work is required to produce animations. A major disadvantage of using such a system, is that the animations need to be adapted so that they look good on different characters, and that the system is limited to recording existing people, animals and objects. Due to the large amount of data produced by motion capture systems, generally in the order of 100 frames per second, such adaptation can be cumbersome. Also, when using recorded motion clips, one must address the problems of selecting suitable clips from the recorded corpus, and prevent unnatural transitions between those clips.

Procedural animations: an animation is defined by a set of mathematical formulae. Periodic motions can be defined by period functions, such as the sine function. By careful combination of such functions, animations such as walking [BC89] or waving can be constructed. Contrary to the discrete representation of the manual and pre-recorded approaches, which need interpolation rules to obtain the required continuity of $A(t)$, procedural animations are explicitly defined as continuous functions. Perlin used tiling pseudo-random noise textures to blend between different motions [Per95]. Such animation systems are light-weight, as no memory is required for keyframes or motion capture data. However, a procedural walk cycle system generally provides only a limited set of high-level parameters, such as velocity, step length and step frequency [BC89].

Physics simulation: an animation is produced by simulating mass and force. By employing Newtonian laws of physics, physically correct motions are generated. Although such simulations can be computationally expensive, the resulting animations generally look life-like. Dynamically simulated characters were first proposed 30 years ago [AG85, WB85]. Characters animated in this way can respond to external perturbations without requiring pre-recorded or pre-constructed animation data specifically tailored for those perturbations [FvdPT01]. Furthermore, it allows for plausible animation of non-existing creatures.

Two main approaches to animation and character control are kinematic and dynamic control. *Kinematic control* refers to the movement of objects irrespective of the forces involved in producing the movement [Par12]. For example, the manual, pre-recorded and procedural approaches described above are concerned with kinematic control. A downside of kinematic control is that physical correctness, or even believability, is not guaranteed. This can be seen in many crowd simulations and games, where characters can instantaneously rotate, slide their feet along the ground, or do not respond to collisions. However, owing to the absolute, direct control over the character's pose, kinematic control is widely used. *Dynamic control* is concerned with computing the underlying forces that are then used to produce movement. Among the dynamic control algorithms are the methods based on physics simulation. In such simulations, it is hard to attain kinematic constraints, such as placing a hand at a specific position on a table, even though such control is often desired in character animation. PD controllers (see Section 2.3) are often used to produce forces that guide a dynamically controlled character into a specific pose. Another major issue with physically based systems is stability. Often the simulation requires very small update steps, in the order of a millisecond per step, to produce stable results. Dynamic control methods are not limited to character animation; other examples are the simulation of hair, cloth and particles. Kinematic and dynamic control can be combined to animate different aspects of a character, by kinematically controlling the body pose and then dynamically simulating hair and clothing.

2.2.4 | PATH PLANNING

Many crowd simulation techniques produce a path for each agent to follow, towards some goal position. Path planning methods can be subdivided into flow-based and rule-based models, cellular automata, agent-based models, and road maps. These approaches not only have an impact on the produced paths, but also have their own implicit or explicit agent shapes. For a general overview of crowd simulation techniques and topics, we refer to the books by Thalmann and Musse [TM13], and Pelechano et al. [PAB08].

Flow-based models are macroscopic, and simulate crowds as a whole. Examples are the seminal work by Henderson et al. who used fluid dynamics [Hen71, Hen72], or the gas kinetics model by Kerr and Spears [KS05]. In these models, characters are represented by particles, i.e. orientation-less, compressible discs. Due to the simple representation, and by putting aside individual behaviour, these models can simulate very large crowds.

Rule-based models, such as Reynolds' *boids* [Rey87] and *autonomous characters* [Rey99], can create realistic looking, complex behaviours and are suitable for low and medium density crowds, such as flocks of birds. The motion and behaviour of these boids are determined by rules, and creating those rules can be difficult. These models were designed for simple, uniform flocks, but when humans are concerned, we expect more individuality and detail than for a flock of birds far away in the sky. Those simple models are not suitable for modelling contact between agents, as they were built to avoid collisions. When contact cannot be avoided, we humans will move out of the way by side-stepping or simply collide and push through the crowd. In these models, characters are represented by oriented particles, i.e. points in $\mathbb{R}^3$ that are associated with some geometry to model their orientation.

Cellular automata discretise floor space into cells. Every character can occupy exactly one cell, and vice versa. The character is implicitly modelled as a cell-shaped object, usually an axis-aligned square. This results in even spacing between agents, which will become unnatural when densities are high. Pushing behaviour is also impossible with this approach. However, it is computationally simple and easy to implement, and can often be seen in older strategy games such as the Command & Conquer series [Lar02]. An alternative approach, which also employs discretised floor space, stores information used by the crowd in each floor grid cell, such as the desired velocity of the characters in that cell [LMM03, Che04, Ali11]. Such an approach can simulate large crowds, but is limited in the number of goals for the agents to move to. It also does not allow for individual preference of characters, so it is impossible to model a heterogeneous crowd.

Agent-based methods employing social forces, or planning in velocity space, such as the Reciprocal Velocity Obstacles (RVO) model [vdBGLM11, CGZM11, vdBLM08], generally allow for reasonably high density crowds of abstract agents, while supporting individual behaviour of those agents. However, there are limits to their effectiveness and realism at higher densities, as we will discuss in Chapter 6. RVO can be extended to support physical interaction with obstacles and the environment [KGM13]. Social forces, introduced by Helbing et al. [HM95], are not directly exerted by the environment, but are used to represent the internal motivation of the individuals, such as the attraction to a target position and avoidance of other people. This concept was further extended by Pelechano et al. by combining these internal forces with physiological aspects of the character and geometrical information of the surroundings [PAB07].

In the described methods, the members of the crowd generally have limited knowledge about their environment, and only take very local information into account. As a result, they can get stuck in places where real humans would not. To solve this, higher level planning is needed, such as *road maps* [SKG05, BBLA02, Ger10]. The high-level paths can be used directly, or processed further [KGO09].

2.2.5 | PATH FOLLOWING

Visualizing animated human characters based on the output of a crowd simulation system is not a trivial task. Some crowd simulation systems avoid this by visualizing only the abstract agent representation [KGM13, Ali11]. Other methods directly use a corpus of predefined animation data to drive the characters [KHHL12, JPCC14], also producing realistically animated characters. A downside to these approaches is that high-level planning, such as specific characters moving towards their exact goal positions, is harder to do, as the required motions may be missing from the corpus.

The most commonly used approach is to place human characters at the simulated agents' positions and orientations. Such crowd simulation methods produce paths for characters to follow, defining the character's global position $\mathbf{p}(t)$ and orientation $\theta(t)$ at each moment in time t. After placement of the character, the body is animated using walk cycles [vdBGLM11, SAC$^+$08, PAB07]. This results in an animated character, but, due to a lack of coordination between the animation and simulation subsystems, the animations often show artefacts such as foot skating or unnatural animation speeds. Foot skating is caused by the simplified agent model. The agent can slide over the ground plane with a linear motion. However, when walking, a human body sways left and right, and accelerates and decelerates at every step. As a result, there is no single point in the body that exactly follows the agent's path [vBEG11]. A carefully constructed walk cycle can model this swaying around a central pivot point, and let that pivot point follow the path. This prevents foot skating when the agent is moving forward in a straight line, but, due to the simplicity of the walk cycle approach, it is not a solution for curved paths. In Chapter 6, we extend the common forward-walking walk cycle approach with backward walking, sidestepping, and diagonal steps.

Singh et al. [SKRF11] plan foot placements in their crowd simulation system. Contrary to the agent path described above, which does not correspond to any point in the body, this model uses a biomechanical inverted pendulum model to produce believable foot placement, steered by a space-time planner. Not only does this prevent foot skating, it also allows for tighter collision bounds and more precise control over character placement and motion. Once the foot placements are planned, several methods are available [vdP97, CH99, VBSE11] to animate a virtual character.

In contrast to the described approach, of first planning a path, and then planning an animated character that follows that path, full-body motion capture data has been used directly as the basis for crowd animation techniques. Lee et al. [LCL06] introduced 'motion patches', later extended by Yersin et al. [YMPT09] and Kim et al. [KHHL12]. These approaches use precomputed human motion, often obtained from motion capture, to animate and stitch together cyclic and collision-free behaviour. A small number of people (in certain cases just a single person) are recorded simultaneously, and multiple recordings are stitched together to form a crowd. As such, these patches could possibly be intertwined into a crowd of high density, but due to the low-density recording, the individuals in the crowd will still show low-density behaviour. Furthermore, due to the high-level planning of these methods, planning the motion of individuals, such as specific characters moving towards their respective goal positions, is much harder to do.

2.3 | PD Controllers

Physical simulation techniques are used for animation (also see Section 2.2.3). In the field of character animation, proportional-integral-derivative controllers (*PID controllers*) are employed to move physically simulated characters into their desired poses [FvdPT01, YLvdP07]. The PID controller is the most commonly used mechanism for control systems, and was first described in a scientific journal by Minorsky in 1922 [Min22]. These feedback control systems are used in many industrial control applications [AH96], such as controlling temperature of the wort at a beer brewery, controlling the joint angles in a robotic arm, or automatic steering of battleships. It is the latter application in which the PID controller originated.

A PID controller is a single-input, single-output controller that consists of three components. The input is the difference between the desired *set point* and the current *process value*; this difference is known as the *error*. For example, in the case of the beer brewery, the set point is the desired temperature of the wort, the process value is the current temperature, and the error is the signed difference between those. The first component of the PID controller is directly proportional to the error (P). The second component is proportional to the integral of the error (I), and counters systematic drift by taking into account previous errors, such as heat escaping due to a lack of insulation of the kettle. The third component is proportional to the derivative of the error (D), and prevents overshoot of the set point. For the beer brewery, it causes the gas burner to throttle down before reaching the desired temperature. The output is the required change in the *control variable*, such as the throttle for the gas burner in the case of the beer brewery, or the rudder angle of a battleship. It takes the form

$$u(t) = K_p e + K_i \int_0^t e(\tau) d\tau + K_d \frac{de(t)}{dt} \,,$$

where $e(t)$ is the current error, the output $u(t)$ indicates the change in control value, and $t \in [0, \infty)$ is the current time. The $K_p, K_i, K_d \in [0, \infty)$ factors are known as *gains*, and influence the controller's behaviour. The appropriate values depend on the controller's application, and algorithms for finding these values are subject to research [MMC96, Luy96, WC02] For an in-depth discussion of PID controllers we refer to Åström and Hägglund [AH06].

In the field of character animation, a slightly simplified model is used. The integral term counters systematic drift, which is relevant only for longer-running processes and of little importance to the rapid changes in pose of an animated character. The integral term is removed by choosing $K_i = 0$, resulting in a PD controller. In Chapter 6, we employ PD controllers to steer agents towards a desired placement in a crowd simulation system.

2.4 | STATISTICS

In this section we discuss two statistical methods. Section 2.4.1 explains linear regression, which is commonly used to fit a model to observations. Section 2.4.2 discusses the use of a binomial test to determine bias, for example of participants in a user study.

2.4.1 | LINEAR REGRESSION

A problem that occurs in many forms of scientific research is the fitting of a model to observations of some system. It is obvious that having more observations results in a better understanding of the system, leading to an overdetermined system (i.e. more observations than degrees of freedom in the model). The most commonly used approach is the minimization of the sum of square residuals, where the residual is defined as the difference between the observed value and the value the model predicts. This 'method of least squares' was first published by Adrien Marie Legendre in 1805 [Leg05]. However, its invention is somewhat controversially accredited to Carl Friedrich Gauss, who claimed to have invented the method as early as 1795, when he was eighteen years old, but did not publish it until 1809 [Gau09]. Gauss, however, was the first to link the method to probability, and provided algorithms for the computation of estimates [Sti81]. A stable numerical solution was formulated by Golub in 1965 [Gol65].

Linear regression is the application of the least squares method to a linear model. Such a model takes a set of parameters x_1, x_2, …, x_n, and provides an expected outcome $E[y]$ by taking a linear combination of the parameters:

$$E[y] = B_0 + B_1 x_1 + B_2 x_2 + \cdots + B_n x_n$$

Note that a linear model is not necessarily limited to straight lines, as long as it is linear in the B_i variables. An example is $E[y] = B_0 + B_1 x_1^2$. In a similar way, the interaction between two parameters can be investigated by choosing $x_i = x_j x_k$. The result of a linear regression is usually shown in a table similar to Table 2.1. The standardized coefficients β indicate the *relative importance* of the factors; these are basically the same coefficients as B, after performing a z-transform on the data. The z-transform modifies the data such that it has an average 0 and standard deviation 1. This removes any scaling factor and offset. For example, temperatures measured in ^{o}C would result in the same β as when measured in ^{o}F. This allows only for comparison between factors in the same linear regression model, and not between models.

As with many statistical methods, there is a *null hypothesis*; p denotes the probability that this null hypothesis is true, given the observations. In the case of linear regression, the null hypothesis states that the average observation $\hat{y}$ is a good model for the system, i.e. $E[y] = \hat{y}$, implying that the parameters x_i have no influence on $E[y]$. The t-statistic is well known from Student's t-test [Stu08]; it is a measurement of the statistical significance of the parameter to the rejection of the null hypothesis, and used to compute p. When $p < 0.05$ it is generally assumed that the null hypothesis can be rejected, and thus that variance in x_i results in a significant variance in $E[y]$.

Next to statistical significance, it is interesting to see *how much* of the variance in the observations can be explained by the model. The R^2 value mentioned in each regression analysis table caption denotes the percentage of the variance in y that matches the variance in the factors x_i. In other words, R^2 indicates the percentage of the variance explained by the model. Note that this does *not*

	Coefficients		**Standardized Coefficients**	**Significance**	
	B	Std.Err.		t	p
(Constant)	$B_0 = 0.59$	0.11		5.32	0.000
x_1	$B_1 = 0.12$	0.16	$\beta_1 = 0.28$	0.79	0.446
x_2	$B_2 = 0.10$	0.04	$\beta_2 = 1.12$	2.78	0.017
$x_1 \times x_2$	$B_3 = -0.08$	0.05	$\beta_3 = -0.76$	-1.54	0.148

TABLE 2.1: Example of the result of a linear regression analysis; $\mathrm{R}^2 = 80\%$

imply that this model is correct only for R^2 of the cases. Given N observations with parameters $\mathbf{x}_i$ and observed outcomes y_i, average observation $\hat{y}$, and model $f(\mathbf{x})$, R^2 is computed as the ratio of the sum of squared residuals SS_{reg} to the total variance of the observations SS_{tot}:

$$\begin{aligned} SS_{reg} &= \sum_{i=1}^{N} (y_i - f(\mathbf{x}_i))^2 \\ SS_{tot} &= \sum_{i=1}^{N} (y_i - \hat{y})^2 \\ \mathrm{R}^2 &= 1 - \frac{SS_{reg}}{SS_{tot}} \end{aligned}$$

R^2 is a deceptively simple metric. Choosing a linear model solely based on high values for R^2 can result in over-fitting the data, and in an overly complex model; adding more parameters x_i increases the degrees of freedom in the model, which will always result in a better fit of the data. Similarly, in Chapter 4, we avoid a more complex model that fits better, since we cannot explain why this more complex model would be correct.

There are more ways in which a linear regression analysis can produce the wrong results. Care should be taken with the resulting model; as it is based only on the observations, it cannot be assumed to be fit for extrapolation. As a simple example, one could study the effect of a training, by analyzing the effect of the number of days of training (x_1) on the effectiveness of the training, expressed as the number of correct questions in an exam (y). If the found $\mathbf{B}_1$ is positive, after a certain number of days of training, the model would predict more correct questions than there are questions in the exam.

2.4.2 | BINOMIAL TEST TO DETERMINE BIAS

Linear regression can provide a model, for example of the number of correct answers in a test, but it does not provide information on the bias of those answers. In Chapter 4, we compute a model that predicts how accurate people are at recognising a certain situation. This model, however, does not provide information about the type of error when the accuracy is low. A *type 1* error is the incorrect rejection of a true null hypothesis, also known as a *false positive* (FP). A *type 2* error is the opposite: the failure to reject a false null hypothesis, also known as a *false negative* (FN).

The ratio FP:FN provides information to the bias in the observations, and its significance can be computed using a binomial test. The *binomial test* is an exact test to compare the distribution of observations with an expected distribution, and can only be applied when there are two categories of observations (in this case, 'false positive' and 'false negative') [FLP13]. When there is no bias, the FP:FN ratio of N observations will show the same distribution as the ratio of heads:tails in N fair coin tosses; this is the null hypothesis. The binomial test results in the probability p that, given the observations, these distributions are indeed equal. When $p < 0.05$, it is 95% certain that the null hypothesis can be rejected, and we can interpret the FP:FN ratio as significantly different from fair coin tosses, and thus biased.

Care should be taken to apply this analysis to the correct data set. For example, when the aggregated FP and FN counts of a user study are used, a significant bias toward FP of one participant can cancel out a significant bias toward FN of another participant. Computing and reporting the bias per participant provides a more detailed and meaningful view.

CHAPTER 3

HIERARCHICAL STRUCTURES FOR COLLISION CHECKING

Simulating a crowded scene requires tight packing of virtual characters, as shown for example in the side view in Figure 3.1 and the top-down view in Figure 3.2. In such cases, collisions are likely to occur, and the choice in collision detection shape will influence how characters are allowed to intermingle. When a crowd is densely packed, collision detection on all characters can become computationally expensive. A brute-force approach to collision detection would check every face of one character mesh with all faces of the others. This is unattainable for a real-time crowd, so a smarter approach is needed. Most agent-based crowd simulation systems use a simplification, where crowd agents are modelled as a cylinder sliding on a ground plane, animating a character inside this cylinder using a walk cycle [HM95, PAB07, OPOD10, AMTT12]. Typically, the character's orientation is identified with the cylinder's velocity, and the cylinder's radius is independent of the character's pose. For sparse to medium-density crowds, this method is often sufficient. However, when the density of the crowd increases such that the movement of

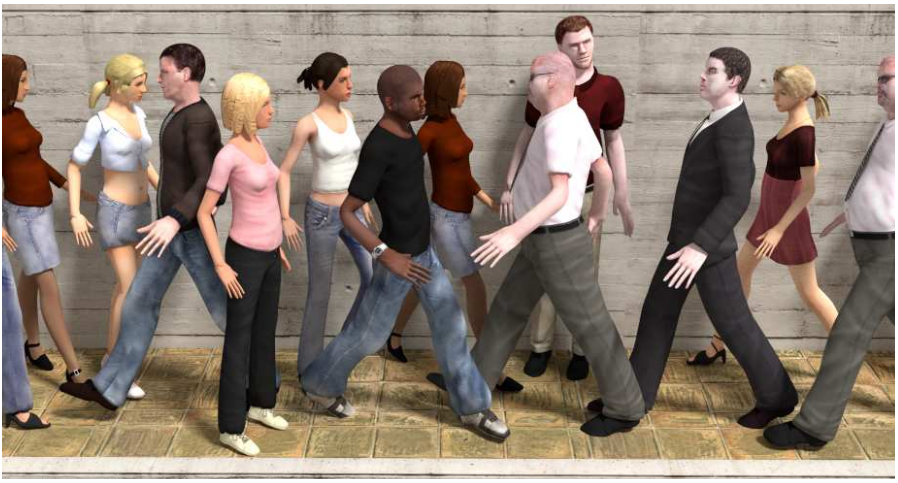


FIGURE 3.1: A crowded pavement.

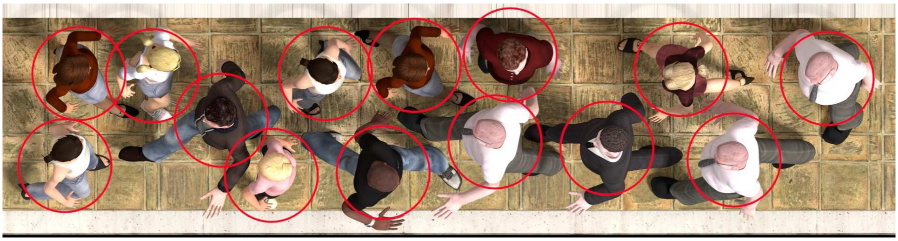


Figure 3.2: A top-down view of the crowd shown in Figure 3.1, using cylinders with a 30 cm radius. Typical in literature are radii between 25 cm and 40 cm (also see Section 3.4.2).

an agent may be impaired, a more precise representation of the space that virtual characters occupy is needed. An example can be seen in Figure 3.2: in such a densely crowded situation, cylinders of any fixed radius would result in either undetected penetrations (when body parts are outside the cylinders) or superfluous empty space between characters.

As a possible solution to the problem described above, we introduce the *bounding cylinder hierarchy* (BCH), a bounding volume hierarchy that uses vertical cylinders as bounding shapes. Since the BCH is a generalization of the single cylinder, we expect that this representation can be easily integrated with existing crowd simulation systems. To be able to compare the BCH with existing collision detection structures, namely the oriented bounding box tree and the shape most widely used in crowd simulation, the single cylinder, we set criteria based on query performance, construction speed and represented volume. Based on the outcome of a comparative experiment, we show which collision structure is preferable for which use case. To get an indication of possible crowd densities, we investigate how close characters can be before collision is detected, and finally propose a critical maximum depth for the BCH. Here we only look at collision *detection*, not collision *perception*. The latter provides a useful measure for optimizing real-time graphics applications, and is studied in Chapter 4 of this thesis.

Note that we focus on the problem of detecting collisions between two given characters, which is known as the *narrow phase* in collision detection. We do not consider the preceding *broad phase*. The broad phase finds pairs of objects that may collide, and eliminates pairs that are far away from each other, usually employing space partitioning structures such as voxel grids [Rey06] or space partitioning trees [VLOG10]. The narrow phase then takes these pairs of objects, and performs the actual collision test. In this chapter we look at discrete collision detection: for every time step in the simulation, stationary shapes are intersected. This is also known as *interference detection*. We also do not consider the computation of the penetration depth. Penetration depth is important for the computation of a physically plausible collision response.

Collision is usually only detected after two objects have some measure of interpenetration, and the penetration depth is then used as a measure of the collision force [HTKG04]. Computing this penetration depth requires a volumetric representation of the colliding objects. However, most applications use a boundary representation, where objects consist of a polyhedral mesh. As a result, we are limited to detecting intersections of the boundaries of the objects.

Related work is discussed in the next section. Section 3.2 formalizes bounding volume hierarchies for collision detection, and defines the bounding cylinder hierarchy (BCH) and OBB tree within that formal definition. Section 3.3 details our experimental comparison between the single cylinder, the BCH and OBB tree. In Section 3.4, we investigate different properties of the BCH. We discuss possible future work and conclude in Section 3.5.

3.1 | RELATED WORK

Applications of collision detection include robotics, computer aided design and manufacturing, and of course crowd simulation. It is common that simplified shapes are used in order to reduce computational complexity, such as a set of bounding shapes for the head, the torso and the limbs. By using algebraic definitions, such shapes can be used for collision detection. However, finding the correct algebraic shapes that tightly fit a given mesh can be cumbersome.

A different approach to speed up the narrow phase is the use of model partitioning techniques such as bounding volume hierarchies (BVHs) that allow for quick determination of non-intersection. Commonly used BVHs are sphere trees [Hub96], axis-aligned bounding box trees [vdB97], oriented bounding box trees [GLM96] and k-DOP trees [KHM$^+$98].

Other authors have also compared different BVHs. Gottschalk et al. [Got00] proved that oriented bounding box (OBB) trees are superior in terms of query performance over sphere trees and axis-aligned bounding box (AABB) trees for surface based BVHs. Their method is widely used, and will be used for comparison with the bounding circle hierarchy. Van den Bergen showed that AABB trees are easy to adjust after the object has been deformed, so for real-time adaptation of the collision hierarchy AABB trees are preferred [vdB97]. Both box-based methods use a bounding shape with straight edges and square corners, which is not a good match for the human shape. This in itself is not an issue as the hierarchy contains all the necessary details, but when the maximum recursion depth is limited (as we investigate in Section 3.4) this becomes relevant. Sphere trees do not have such square corners, and have been proven useful for the estimation of penetration depth [OD99]. Collision tests on spheres are simple as they are independent of the character's rotation.

However, a sphere bounding a virtual character would contain much empty space. For a more detailed survey on collision detection techniques we refer to the work of Kockara et al. [KHI$^+$07].

In this chapter, we introduce the *bounding cylinder hierarchy* for discrete collision detection, also known as interference detection. The cylinder is the prevalent shape in crowd simulation, and is widely accepted as a rough approximation of the human shape. By employing a hierarchical refinement strategy, we ensure that the BCH represents a much tighter fit than possible with a single cylinder. The cylinder's rotational symmetry allows for fast intersection tests and efficient storage.

3.2 | HIERARCHICAL STRUCTURES FOR DETECTING COLLISIONS

In the previous section, we have described a family of bounding volume hierarchies that are commonly used for collision or interference detection. This section presents a formal definition of such structures, providing us with a common reference frame to compare members of this family. We introduce the bounding cylinder hierarchy as a hierarchical generalization of the commonly used cylinder, and redefine the oriented bounding box tree within the terms of our reference frame. Section 3.3 will use these definitions in a comparative experiment. We use polyhedral character meshes, and assume the mesh consists of triangular faces; non-triangular faces can be triangulated without loss of generality.

For an object P the ingredients for a hierarchical collision structure $\mathcal{H}(P)$ are:

1. A family of shapes, such as cylinders, boxes or spheres;
2. A finite tree structure, where every node ν contains a bounding volume $B(\nu)$ of the aforementioned shape family, and represents a sub-object $P(\nu) \subset P$;
3. A subdivision strategy, defining nodes in tree layer $i+1$ given the nodes in layer i;
4. A stop criterion for the subdivision.

Let ν be any node in the tree, and $C(\nu) = \{\mu_1, \ldots, \mu_k\}$ denote its child nodes. We impose the following requirements for the hierarchical structures we consider in this chapter, where $int(X)$ denotes the interior of X:

- $P(\nu) = \bigcup_{\mu \in C(\nu)} P(\mu)$;
- For all $\mu, \mu' \in C(\nu)$, $\mu \neq \mu'$: $int(P(\mu)) \cap int(P(\mu')) = \emptyset$.

So in other words, $P(\nu)$ is partitioned into the sub-objects associated with the members of $C(\nu)$. A crucial property of bounding volumes is that for any query object Q and node ν, if $Q \cap B(\nu) = \emptyset$ then $Q \cap P(\nu) = \emptyset$.

The above properties rule out certain bounding volume hierarchies, such as using bounding boxes for torso (root node) and head, arms and legs (child nodes), or the elliptical and cylindrical collision shapes by Dube et al. [DTT11]

3.2.1 | BOUNDING CYLINDER HIERARCHY

In this section, we define the Bounding Cylinder Hierarchy (BCH). It refines the cylindrical representation commonly used in crowd simulation. Note that in this chapter we use *bounding* shapes for collision detection of characters (and parts of characters), which is not always the case in the field of animation; often approximations are used that do not necessarily contain the entire character, in order to artificially allow increased crowd densities at the expense of realism (also discussed in Chapter 4). We define the structure $\mathcal{BCH}(P)$ as follows.

1. A vertical cylinder is used as the bounding shape $B(\nu)$.
2. The tree is binary; the root represents the entire object P with its smallest enclosing cylinder.
3. $P(\nu)$ is subdivided by a vertical separation plane. This plane is defined by a point (for we will test several approaches, described below), and a normal vector. To find this vector, we consider the projections of $P(\nu)$ onto the two horizontal global coordinate axes; the axis for which the projection has the largest extent defines the normal. $P(\mu_1)$ and $P(\mu_2)$ are defined as a partition of $P(\nu)$ by that plane – details are given below.

   $B(\mu_i)$ is defined as the smallest enclosing cylinder of $P(\mu_i)$.
4. When the radius of $B(\nu)$ is less than a predefined threshold, subdivision stops.

$P(\mu_1)$ and $P(\mu_2)$ are separated by a vertical plane, hence their interiors are disjoint. As $P(\mu_1) \cup P(\mu_2) = P(\nu)$, it follows that $P(\nu)$ is partitioned properly. A binary split was chosen in favour of a split into four parts, as the latter can result in longer, thinner subdivisions that are not a good match for a cylindrical bounding shape. Furthermore, splitting into four parts can

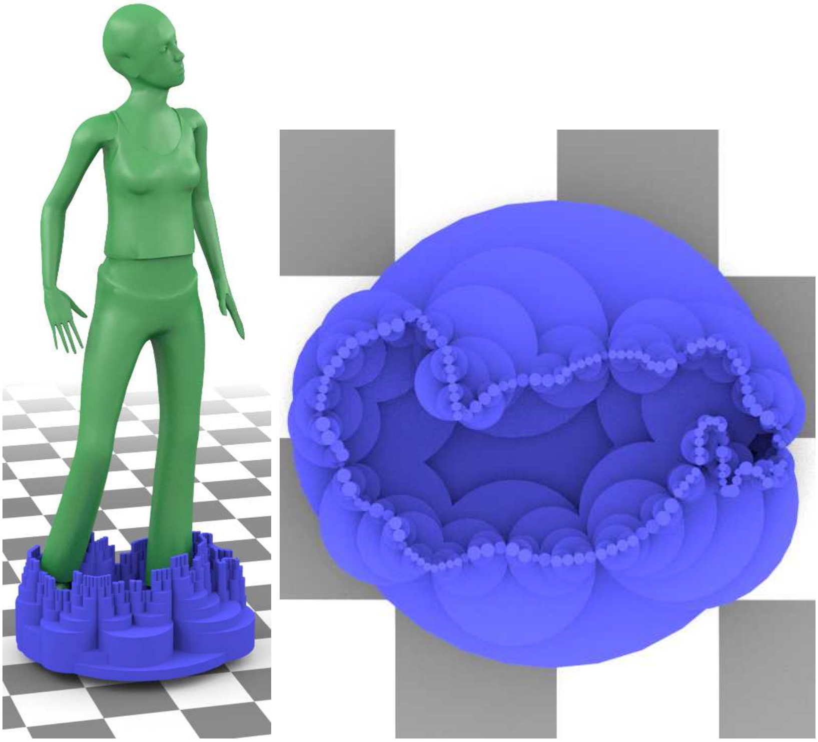


Figure 3.3: Visualization of the *contour* BCH for the green mesh. The cylinders are vertically cropped, so that the green mesh is still visible in this figure.

be seen as a specialized, more restricted variant of binary splitting. In our implementation we consider the bounding cylinders to be of infinite height, effectively ignoring the height of the character. This is common practice in crowd simulations using single cylinders. Future development could include the height information in intersection tests.

We have investigated four methods of representing $P(\nu)$ and chosen appropriate subdivision schemes accordingly.

Contour: In this approach we only consider the contour from the top view of $P(\nu)$. The separation plane is then defined by its normal as described in item 3 in the list above, and the centroid of the contour polygon. The centroid is commonly used, as opposed to the centre of the bounding box, as it often results in a more balanced tree and better query performance. We have chosen to use the centroid of the contour rather than that of the projected mesh vertices, in order to prevent bias to more detailed areas.

The contour is interpreted as a sequence of line segments $S = \{S_0, \ldots, S_n\}$. For each segment the centre point is computed; depending on the side of the separating plane the centroid lies on, the segment is used to define $P(\mu_1)$ or $P(\mu_2)$, taking care that neither set will be empty.

When S only contains a single line segment, and the segment is longer than twice the threshold radius, the line segment is split into two halves and decomposed as described above. This results in very little overlap between the cylinders associated with the leaf nodes of the hierarchy, as shown in Figure 3.3. The separation of line segments into smaller segments is not performed when there are multiple line segments in S, as doing so would result in a much larger number of cylinders in the hierarchy.

Contour (centre): This approach is almost identical to the *contour* approach, differing only in that, rather than the centroid, it uses the centre point of the axis-aligned bounding box.

All projected triangles: In this method we eliminate the need to compute the contour explicitly, by simply including all triangles of the character mesh into the hierarchy. This will allow for faster construction at the cost of an increase of the query time. This approach mimics the decomposition used by Gottschalk et al. [GLM96]. A triangle is never subdivided into smaller pieces, but is placed into a subdivision $P(\mu_i)$ depending on the position of its centroid with respect to the separating plane. The separation plane is chosen as described above, except that the centroid of the mesh vertices is used. Subdivision stops when a node contains a single triangle. Since they are not subdivided, we cannot get arbitrarily small cylinders. However, we do have the ability to perform efficient, exact intersection tests as every leaf in the tree contains exactly one triangle.

Single cylinder: This method could be described as the *all projected triangles* method, but using an infinite number of triangles for the stop criterion. This results in a hierarchy with one node, containing only a single bounding cylinder. As this form is so common, we handle it as a special case.

3.2.2 | ORIENTED BOUNDING BOX TREE

The OBB tree by Gottschalk et al. [GLM96] is a well-known and widely used method for collision detection, and thus forms a good comparison for the BCH. It follows the formal definition given at the start of this section:

1. An oriented box is used as the bounding shape.

2. The tree is binary; the root represents the entire object P with its oriented bounding box.
3. Triangles of the mesh are stored in the leaves, based on which side of a separating plane their centroid lies. For the exact spatial subdivision rules we refer to [GLM96].
4. When a node contains only a single triangle, subdivision stops.

At every internal node, every triangle in $P(\nu)$ is represented by exactly one child node. This ensures that $P(\nu)$ is partitioned properly. At the leaf nodes, the OBB collision test performs triangle-triangle intersection tests. We define three techniques to construct the OBB tree:

Full: The OBB tree is populated with the triangles of the original mesh. This is thus an exact, three-dimensional representation of the mesh. Even though the BCH and the other OBB approaches all use a projection onto the ground plane, we are interested in a comparison with an exact representation of the mesh.

Contour: We compute the contour of the projection of the mesh, as in Section 3.2.1. The line segments that make up this contour are used to populate the OBB tree.

All projected triangles: We use all triangles of the character mesh projected onto the ground plane to populate the hierarchy, as in Section 3.2.1. Our hypothesis is that this will be less efficient to query than the *contour* case, but faster to construct. It may be faster to query than the *full* case.

3.3 | COMPARISON

In order to compare the BCH and OBB trees, we perform two timing experiments. Each experiment uses two character meshes, one male and one female (see Figure 3.4) of approximately 2850 triangles each. We use motion capture recordings totalling 212 seconds of walking, turning, side-stepping and idling motions; such motions are common in crowds. For each of the two meshes, we randomly select 100 motion capture frames, resulting in a total of 200 randomly posed character meshes. For the experiments, those are regarded simply as a collection of triangles; the fact that they were posed using an articulated structure was of no relevance, and we do not use any metrics that depend on such a structure. Tests are run multiple times to reduce jitter and to average out external factors, on an Intel Core i7 3630QM 3.2 GHz laptop. The BCH radius threshold is set at 1 cm.

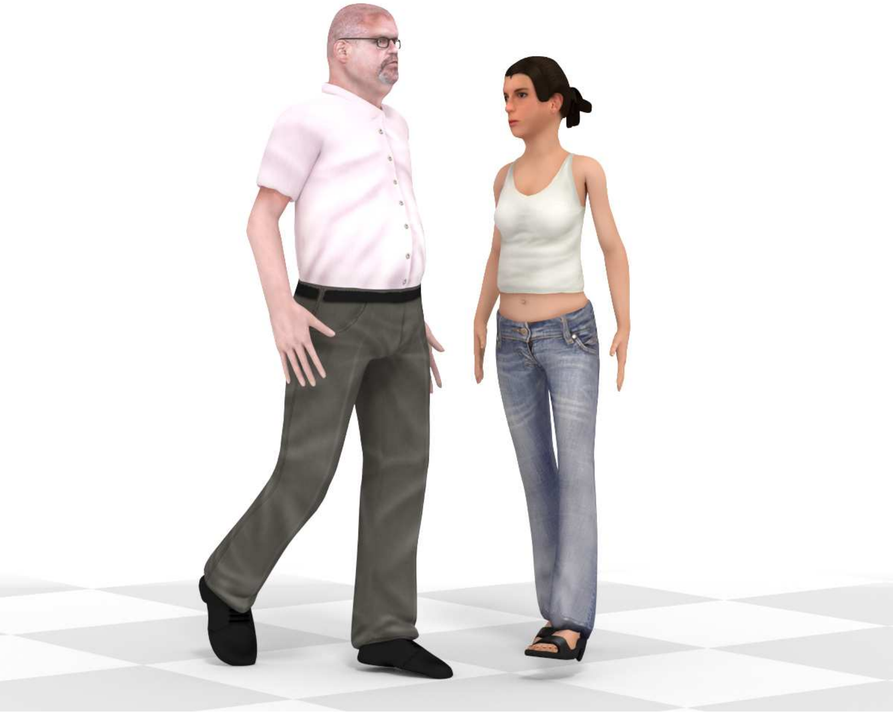


Figure 3.4: The two characters we used for testing. We removed non-manifold edges such as hair strands to allow for volume computations.

We represent characters by their boundaries and do not see them as solids; we consider two objects as colliding when their boundaries intersect. This means that we will not be able to detect the case where one character completely envelopes another. In our intended scenario this will not be an issue, as this cannot happen given the character shapes and possible poses.

In our first experiment we measure the time required to construct the representations for each of the 200 random poses; the results are shown in Figure 3.5a, where the 'BCH single cylinder' row shows the time needed to compute a single bounding cylinder for the posed mesh using a linear programming method [MSW96]. The time to load the mesh data from disk is excluded from the test.

The second experiment measures the duration of intersection tests. Each of our 500 test cases is created by selecting two random poses, a random translation along the ground plane and a rotation about the up-vector. The process of randomization is repeated until a true collision is detected using the exact *full* OBB method. This presents us with a worst case test set, as

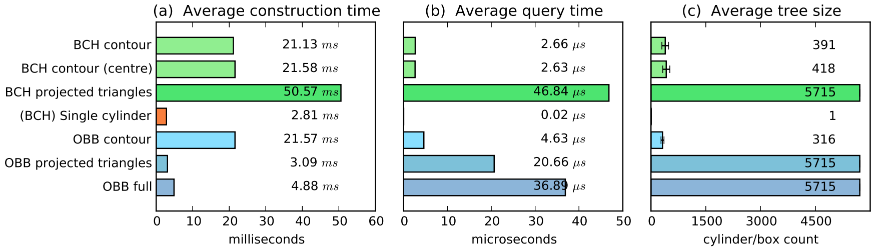


Figure 3.5: Comparison between the BCH and OBB tree variants.

negative tests often do not require descending the entire tree where positive tests do. All tests are binary, i.e. only report whether an intersection is found.

We run 10,000 test iterations, where each iteration tests all 500 test cases sequentially, preventing over-optimistic results due to caching. To illustrate the effect of caching, we have performed the experiment in a different order by running each individual test case 10,000 times. This resulted in an approximately 15% faster performance, but as this would not reflect a real use case, it will not be included in our results.

The experiments show a query time of 2.66 μs for the *BCH contour* method, and 4.63 μs for the *OBB contour* method; in our scenario the BCH is 74% faster to query than the OBB tree (see Figure 3.5b). In general, the amount of overlap between cylinders of the same hierarchical layer of a BCH is higher than the overlap between boxes in the corresponding OBB tree. However, the simplicity of intersection tests between cylinders makes up for this when using our *contour* approach. The more or less fat shape of the projected mesh is a reasonably good fit for the cylinders, whereas the OBB tree is more efficient for long, thin shapes such as individual triangles. This is clearly shown in the difference in intersection speed between the BCH and OBB *all projected triangles* cases, where the trees are of the same size and both well balanced, but due to the lesser amount of overlap the OBB tree is more than twice as fast to query.

There is no significant difference between the *centre* and *centroid* BCH methods, with respect to construction and query duration. Since using the centroid results in a slightly smaller tree (see Figure 3.5c), the remainder of this chapter will use this approach.

For interactive, multi-character purposes, use of the BCH requires a preprocessing step in which the data structures are computed, as this cannot be done at interactive rates for multiple characters (yet). However, the resulting structure is nearly 8 times faster to query than the 20.66 μs of the fastest-to-construct *projected triangles* OBB method.

The OBB tree provides an exact collision test, whereas the precision of the BCH depends on the threshold radius. For cases where exactness is more important than performance the OBB tree would be a more suitable representation. We measure this in Section 3.4.

The implementation by Gottschalk et al. was used in all three OBB cases. The *contour* tests were performed using this 3D implementation, where the contour line segments are input as degenerate triangles. A pilot experiment showed that this did not cause significantly different results compared to using non-degenerate triangles. Our experimental results thus reflect a worst case scenario; an alternative implementation optimized for two-dimensional shapes could provide different results, but was not available.

The BCH reports a collision when the cylinder at a leaf node collides with another cylinder at some leaf node. By instead reporting a collision when a given maximum recursion depth is reached, we can allow for a *level of detail* approach. A possible application would be in a crowd simulation scenario. More precision is required close to the camera than further away in the crowd, as most of the collisions will be occluded [KOOP11]. Crowd density could also be used to influence the maximum recursion depth, reverting to a single cylinder for low-density crowds. A similar technique could be applied to the OBB tree, although at coarser levels the corners of the bounding boxes may induce unnatural behaviour and visual artefacts. At the coarsest level the BCH collapses to a single cylinder, which has already been proven to be useful for crowd simulations.

These experiments are purely quantitative, and do not consider the quality of the results. A qualitative experiment is performed in Chapter 4.

3.4 | PROPERTIES OF THE BCH

In the previous section we have shown that the BCH is a suitable collision structure for virtual characters. This section investigates the BCH further. One of the goals of this thesis is to look at ways to effectively deal with crowds of high density. We look at the effect of employing a maximum recursion depth for intersection tests, to see how much closer meshes can get to each other, before a collision is reported, by using a more detailed representation.

In real life, a collision between two people occurs when there is a collision between the two occupied volumes. The more precise the collision shape represents this volume, the smaller the risk of false positives. In Section 3.4.2 we investigate the represented volumes of the BCH and compare those to the commonly used single cylinder.

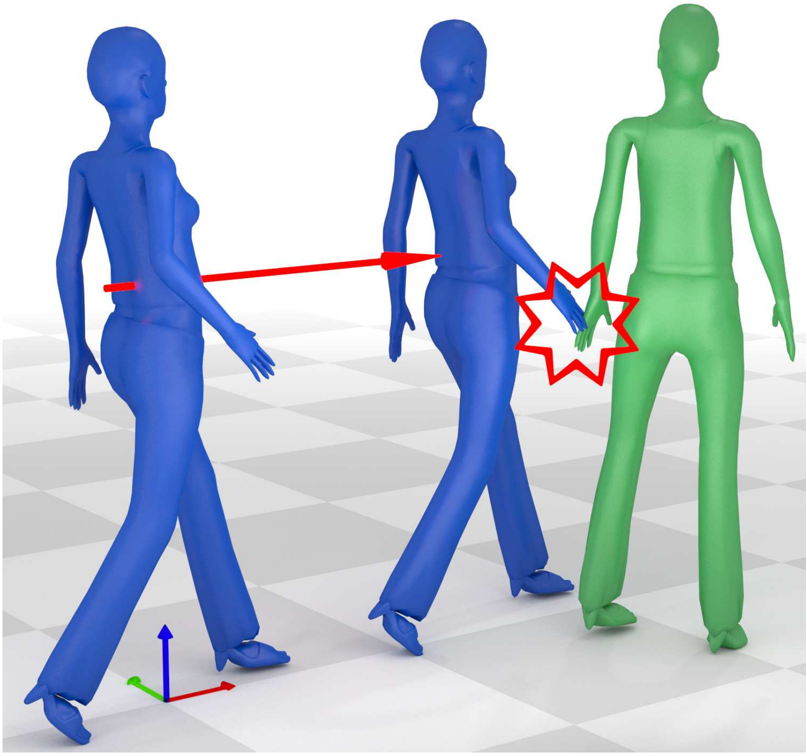


Figure 3.6: The distance test, where one character is slid towards the other until collision is detected.

These experiments have been performed using the same 200 randomly posed meshes as described in Section 3.3. Also for these experiments, the posed meshes were regarded simply as a collection of triangles.

3.4.1 | Residual distance

Our premise is that a better-fitting collision shape will allow for tighter packing of virtual characters, as less false positives will be detected. In this section, we investigate the residual distance between two characters when we detect a collision. Often, the root joint is used as *the* position of a character, and the planar difference in root joint positions denotes the distance between them. However, the choice of root joint can be arbitrary, and the reported distance should also depend on the pose. Instead of the root joint distance, we use the actual distance between the two meshes (see Section 2.1); when this distance is zero, there is a true collision.

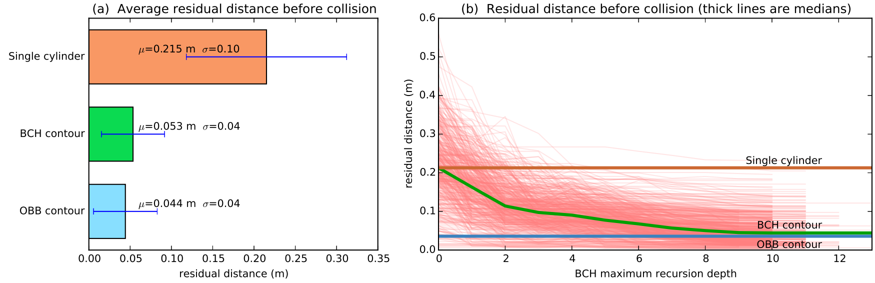


Figure 3.7: Residual distance between meshes before collision. (a) The bar chart shows the average residual distance between two characters at which a collision is reported. (b) The graph shows a thin red line for each test case and thick lines for medians. The dark green, curved line shows the median residual distance obtained with the BCH.

We use a coordinate frame where the x- and y-axes span the ground plane. In our experiment we randomly orient two posed meshes. We place one at the origin, and the other at $(2, y)$ where $y \in [-0.5, 0.5]$ is chosen such that the meshes will collide. We then slide the first mesh along the positive x-axis until a collision is reported, as shown in Figure 3.6. In the resulting configuration, the actual distance between the two meshes is reported as the *residual distance*. We chose this approach over explicitly computing the distance at which a collision would occur, as our approach requires no knowledge of the actual collision detection algorithm and is very simple to implement. We repeated this process for all 200 meshes in a total of 500 random orientations, and for several collision detection approaches.

The bar chart in Figure 3.7a shows the average residual distance for the minimum bounding cylinder, the *contour* BCH and *contour* OBB tree. The *full* OBB method is not included in the figure, as this is an exact collision detection scheme, so the distance is zero with zero standard deviation.

At an average residual distance of 5 cm, the *BCH contour* method allows characters to get significantly closer to each other before a collision is reported than the 20 cm allowed by a single cylinder. The situation shown in Figure 3.1 is impossible to attain when representing characters with a single cylinder. However, even in this dense situation the contours do not intersect.

On average the *OBB contour* allows for a residual distance of 4 cm, which is 1 cm closer than the *BCH contour* allows. This is explained by the fact that the BCH is an approximation; in our experiment we used a threshold radius of 1 cm, which is reflected in these results.

To investigate the *level of detail* possibilities of the BCH we limit the recursion depth of collision tests. Figure 3.7b shows the distance at which a collision is detected given a maximum recursion depth d. The single cylinder always has the same distance as $d = 0$. A higher d results in a tighter fit of the mesh, except in some specific cases (for example the small peak in one of the test cases at $d = 3$). At $d = 9$ the distance is only 1% larger than at $d = \infty$, so for many practical purposes this represents a critical maximum recursion depth. As the number of reachable nodes in the tree is $O(2^d)$, imposing such a limit will increase query performance.

3.4.2 | REPRESENTED VOLUME

When checking for collisions using simplified shapes, one aims to use a collision shape that approximates the actual shape in a sufficiently precise manner. To see how well a cylinder matches a human shape, we have compared for all 200 random poses: the smallest enclosing axis-aligned cylinder and the volume represented by the BCH. We normalized the volumes by the volume of the mesh, to make it easier to compare between meshes of different sizes.

We posed the meshes using linear blend skinning, also known as skeletal subspace deformation. This is a conventional skinning method that does not preserve volume [GB08]. However, we do not use heavily deforming poses; we measured a standard deviation of 0.01 m^3. The average of the measured volumes was used for normalization. We define the volume represented by a BCH as the volume of the union of the cylinders stored in the leaf nodes of the *all projected triangles* representation. For simplicity we use a discretization by drawing the bases of the cylinders onto a high-resolution image ($>$ 6 megapixels/m^2), counting the drawn pixels and multiplying with the character's height.

The results can be seen in Figure 3.8, with numeric data in Table 3.1. From this, we can observe that the cylinder may not be the best representation of the human shape, as it represents a volume approximately seven to ten times larger than the actual character.

The radii most often used in literature are around 0.25 m [KGO09, vTCG12, HFV00], or around 0.40 m [Ger10, AMTT12], and sometimes up to 2.00 m [GKLM11]. We used character models that are slightly taller than the world average (1.74 m for the female character, 1.88 m for the male character), probably because the models were created in The Netherlands. The average radii we have found for the random poses are consistent with what is used in literature. However, given our results, we can safely say that for average-sized characters a radius of 0.25 m will cause unnoticed intersections (i.e. false negatives), whereas a radius of much more than 0.40 m will cause more spacing between characters than strictly necessary (i.e. false positives). In Chapter 4, we investigate the perception of collisions, and report on the observed bias of the participants towards false positives or negatives.

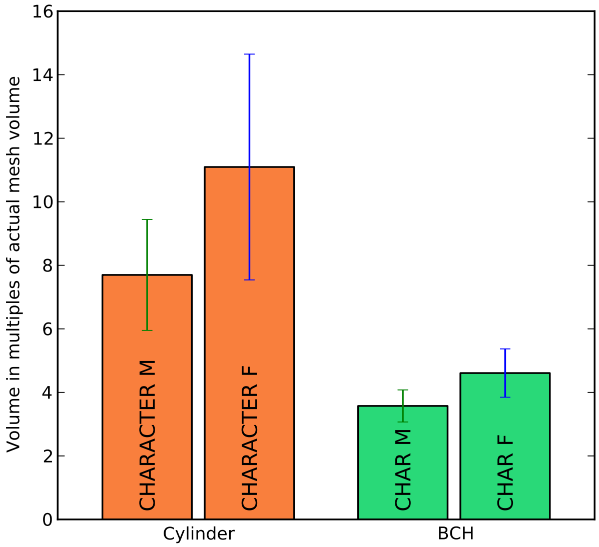


Figure 3.8: Volumes occupied by the minimal bounding cylinder and by the BCH. The volumes are expressed as multiples of the actual mesh volume, to allow comparison between different meshes.

			average	**median**	**stdev**
Cylinder radii (m)		Female	0.34	0.31	0.05
		Male	0.42	0.40	0.05
Volumes (norm.)	Cylinder	Female	11.09	9.50	3.55
	BCH	Female	4.61	6.99	0.76
	Cylinder	Male	7.70	6.99	1.75
	BCH	Male	3.57	3.34	0.51

Table 3.1: The observed radii of the minimal spanning cylinders, and the difference in volumes occupied by those cylinders and the BCH. All volumes have been normalized by the mesh volume to allow comparison between different meshes.

3.5 | Conclusion

We have introduced a bounding cylinder hierarchy for efficient collision checking with a flexible level of detail. We have compared it with the commonly used OBB tree, and seen that in the scenario we tested – humanoid character collisions – the BCH provides a better query-time performance.

We suspect that the BCH is more resistant to an increase in detail of the model than the OBB tree. The latter typically contains all triangles of the mesh, and thus depends on the mesh complexity. The BCH complexity is bound by the threshold radius, providing tunable mesh simplification for collision checking.

In contrast to many single-cylinder systems, we have consistently used the smallest bounding cylinder of the posed character. This resulted in a worst case scenario, as often in crowd simulations a fixed pose-independent radius is used. In such a case, visualizing a dense crowd is guaranteed to show undetected interpenetration of body parts, which we want to avoid. We have shown that the volume that is represented by the BCH is much smaller than that of a single bounding cylinder, and that the BCH also allows characters to move much closer together before collision is detected.

The walking, turning, side-stepping and idling motions we used in our experiment are fairly commonly seen in real crowds. When performing such motions, people are fairly cylinder-like, in that they are standing straight up, and are taller than they are wide. It would be interesting to see how well the BCH and other representations compare when different motions are used, such as cartwheeling, belly flopping and breakdancing.

It is trivial to compute the distance between two cylinders. This is important for crowd simulation systems, as many distance checks are performed between neighbouring crowd agents. Such distance checks provide a more detailed indication of proximity than a binary collision check. A fast distance metric between BCHs would enable a crowd simulation system to use the character poses in their proximity computation, and thus produce a more realistic results. We leave the construction of such a distance metric to future work. However, in Chapters 5 and 6 we introduce a capsule shape for collision detection, which fits better than a cylinder; distances between capsules are also relatively easy to compute.

In our current implementation, the vertical plane used in the construction of the BCH is always aligned with a horizontal coordinate axis, as suggested by Kolowski [Kol98]. Using a similar covariance analysis as used by Gottschalk in the OBB construction could result in a more optimal subdivision and faster query times, at the potential expense of longer construction times.

The cylinders in the BCH span the full height of the pose. To facilitate a closer fit, we could horizontally slice the mesh, and create a BCH for each slice. Possible slices would be for the legs, torso, and head. This would provide a tighter fit at only a constant factor in processing and storage costs. Furthermore, such subdivision would allow for different levels of detail in collision checking for different body parts.

We have limited our comparison to static poses. The BCH for one pose is completely unrelated to the BCH of the next pose of the character. We are interested in investigating an optimization technique to find a set of animated cylinders that fit an entire animation. Not only would this allow for more efficient storage, but such temporal coherence would also make interpolation between motions possible. The method by Van Basten et al. [VBSE11] is an example in which predefined animation segments are interpolated and where such preprocessing is possible.

The BCH represents an approximation of the character's shape. It is constructed such that it never produces a false negative, but does produce false positives. This is a common approach in robotics, where the collision between robots can result in expensive damage. However, this is not the case for virtual characters, as they can freely move through each other. In a virtual environment, the precision of collision detection is not determined by possible damage, but rather by the available computational power, the requirements to the level of realism, and the accuracy of the observer. In the next chapter, we try to measure the latter aspect, and follow up with a qualitative study to find out how well people can recognise collisions in both animations and static poses.

Chapter 4

Perception of Collisions between Virtual Characters

In the previous chapter, we described the Bounding Cylinder Hierarchy (BCH) and Oriented Bounding Box (OBB) collision detection techniques. The BCH represents an approximation of the character shape, and so does the OBB when applied to the ground projection of the character. However, many other collision detection schemes, including the OBB applied to the original character shape, aim at exactness, which is vital in areas like computer aided design and robotic product manufacturing. Such exactness may not be the best approach for collision detection between virtual characters. People observing virtual characters may not be able to recognise collisions in certain configurations, and thus certain optimisations could exploit this to improve collision detection performance without sacrificing perceived quality, or to provide a better match between observed and detected collisions. Exactness seems even less crucial in crowds of virtual characters; people observing a crowd of virtual characters do not always have all the information to determine whether a collision occurs.

This chapter presents an investigation into the accuracy of human observers with regard to the recognition of collisions between virtual characters. We have performed two user studies into the perception of collisions between virtual characters, to determine how accurate human observers can classify a situation as ‘colliding’ or ‘not colliding’. A pilot experiment investigates the perception of static images. The main experiment uses video to explore the effects of movement; we have investigated the angle between the characters and the severity of the (near) collision, and present a statistical model for the expected accuracy.

Our results show that the average observer has a bias towards negative (‘not colliding’) answers, mostly in cases of minor collisions, and that the accuracy of the answers has an asymmetrical relation with the severity of the (near) collisions. To conclude we suggest a technique to improve performance of collision handling possible collision shape, and a simplification scheme that matches human perception. Contrary to our approach in Chapter 3, this simplification is based on inner approximations rather than the coarse cylindrical outer approximations that are commonly used in animations. Furthermore, we investigate the difference in perception of lower-body and upper-body collisions, which, to our knowledge, has not been explored yet.

The rest of the chapter is organised as follows. Section 4.1 discusses related work. The overall experiment design is described in Section 4.2. The pilot experiment is described in Section 4.3, and the main experiment in Section 4.4. The implications are discussed in Section 4.5. Section 4.6 concludes the chapter.

4.1 | RELATED WORK

The perception of collisions between large numbers of objects was studied by O'Sullivan et al. [ORC99, OD01]. They showed that when the simulation becomes more complex, such as when looking at animated characters rather than simple geometric shapes, observers 'rely on their own naïve or commonsense judgements of dynamics, which are often inaccurate' [OD01]. We will come back to this later in this section.

Perceptual studies of virtual characters have been performed with regards to motion, emotion, timing and sound [EMO10, MEDO09, EHEM13]. An experiment by Hoyet et al. [HMO12] investigated the perception of causality in virtual interactions, dealing with pushing interactions between characters. Their focus lies on the perceived realism of a scene, after applying alterations commonly found in virtual environments such as games. In contrast, our experiment does not focus on perceived realism, but on whether collisions can be perceived at all.

Perception of collisions between a real user and a virtual entity has also been studied. DeLucia [DeL13] investigated the perception of collision with respect to traffic safety, i.e. collision between moving obstacles and a stationary observer. Olivier et al. [OOP$^+$10] performed a user experiment to assess whether real humans are also able to accurately estimate a virtual human motion before collision avoidance, and conclude that, when an observer is in front of a simple display, judgement of crossing order was easier than recognition of future collisions. This shows that perceiving collisions, at least when one virtual character is involved, is nontrivial. They continue to show that the 'bearing angle', the angle at which one entity sees the other, plays a large role in the perception of collisions.

Kulpa et al. present an experiment of both the perception of crowds of virtual humans, and an accompanying LOD technique for collision detection

[KOOP11]. Their focus lies on the accuracy of human observers in recognising collisions, based on various parameters such as camera distance, horizontal and vertical camera angle, and character distance. They measure the latter as the distance between the characters' root joints, which, although easy to compute, provides only a rough estimate for the distance between the two characters. In contrast, in this chapter we use the actual distance between the character shapes, as defined in Section 2.1. This metric also determines whether there is actually a collision or not. Another contrast to the aforementioned work by Kulpa et al. lies in the placement of the camera. We place the camera such that the collision itself is maximally visible. Furthermore, rather than having the characters walk along parallel paths, we consider crossing paths of the characters, and measure the effect of the angle between those paths on the perception of the collision.

To speed up collision detection algorithms, it is common to forego the possibly complex shape of the object, and use a simplified shape instead. *Level of detail* (LOD) techniques can generate such shapes, most notably applied to model simplification for rendering acceleration [CVM$^+$96, LWC$^+$02]. LOD techniques have seen less emphasis in the area of collision detection, and, as we have done in Chapter 3, mostly focus on the simplification of the colliding shapes [DO00, YSLM04]. Otaduy and Lin [OL03] introduced a technique that also considers the velocity and view size of the objects, and allows for time-critical detection in a similar way as introduced by Hubbard [Hub94]. Apart from the velocity-based LOD technique, these techniques do not focus on human perception of collisions, and taking this into account could lead to better algorithms. O'Sullivan et al. [OCV$^+$02] incorporated LOD techniques not only in rendering and collision handling, but also in the animation and behavioural algorithms. In the discussion section we explore possible adaptations of LOD techniques to bring them in line with our findings on human perception.

We have performed two user studies. The first experiment is a pilot experiment, the main goal of which is to determine reasonable ranges of parameters to be used in the main experiment. The pilot experiment uses static poses rather than animated characters, since it is nearly impossible to find animated cases corresponding to specific desired parameter values, even though we are aware of the limitation that the lack of movement may make it difficult to fully assess the situation. As such, the participants' answers may depend on common-sense judgements, which O'Sullivan and Dingliana [OD01] described as inaccurate. However, their conclusions are based on geometric shapes, and it will be interesting to see how accurate or inaccurate the results are for human shapes. The main user experiment uses animated characters, since we aim at applying our results to collision detection strategies for moving characters. We examine the effect of the severity of the (near) collision, and the angle between the paths of the two characters.

4.2 | EXPERIMENT DESIGN

In this section, we describe the common experiment design, which is shared by both the pilot and main experiments. In both user studies, we show rendered 3D scenes involving two virtual characters in an otherwise empty virtual world. One of the characters is male, the other is female. The two characters are posed using previously recorded motion capture data of a walking person. The following invariants are taken into account.

- No *sharp* shadow is rendered, as that would effectively provide a second angle of view.
- The ground plane is evenly textured, and blends into a solid background, such that it gives the user some sense of perspective without distracting.
- Characters are fully textured and rendered using smooth shading. This provides the most realistic rendering of our character models, while maintaining the exact triangular shape used in the distance and collision computations.
- The characters are placed such that the (near) collision occurs in the vicinity of the origin.
- The camera is placed at an eye height of 1.75 metres and slightly looking downward as to show both characters from head to toe, mimicking the viewpoint of a human observer in a similar real-life situation. The downward angle is adjusted such that the point at 0.89 metres above the origin is at the centre of the view.
- The camera's field of view is chosen to mimic a 50 mm lens on a 35 mm ('full-frame') camera, which is known to result in a perspective distortion similar to that of the human eye.
- Lights are attached to the camera at an offset; lighting is constant with respect to the camera angle.

Participants are presented with an online web-based questionnaire. Before starting the test, they are instructed that any physical contact between the displayed characters (including the slightest touch) is considered 'a collision'. Each participant is shown a scene, advancing to the next after the question 'Do these characters collide?' is answered. The questions are binary; it is only possible to answer 'yes' or 'no'. Answer buttons are always visible, and can be used at any time. Participants have to click on their answer, and then on a confirmation button, which is placed equidistant to the 'yes' and 'no' buttons (see Figure 4.1). This ensures that the mouse has to travel a similar

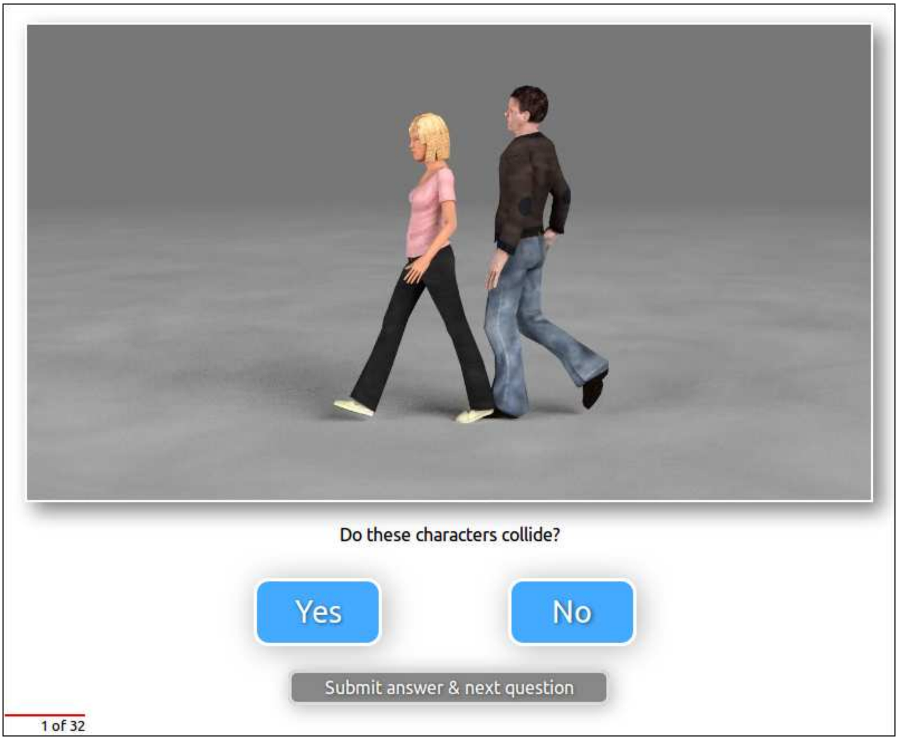

Do these characters collide?

Figure 4.1: Screenshot of the questionnaire.

distance regardless of the answer to the previous question, preventing bias towards repeating answers.

The questionnaire, for both the pilot and main experiment, was open to any participant, who were sourced among colleagues, students, members of computer science and game development forums, and several other non-computer science forums. A small reward was raffled off among interested participants that completed the survey.

Four types of answers are considered: true positive (TP) when there was a collision and it was recognised as such; false positive (FP) when there was no collision but it was recognised as one; true negative (TN) when there was no collision and recognised as such; false negative (FN) when there was a collision but not recognised as one. Accuracy A is computed in a similar way as by Kulpa et al. [KOOP11], as the fraction of correct answers

$$A = \frac{TP + TN}{TP + FP + TN + FN}$$

Absolute uncertainty (i.e. pure guesswork) would result in $A = 50\%$. With continuous data, absolute uncertainty would result in a normal distribution around $A = 50\%$; analysis of the standard deviation would be needed to determine whether the observed distribution differs significantly from pure guesswork. However, due to the binary nature of our data, the standard deviation contains no information regarding the spread of the answers.

4.3 | Pilot experiment

In this section we describe our pilot experiment. We investigate the ability of observers to recognise collisions between virtual characters in *static* situations. Using static images allows us to test a wider range of situations that are difficult to create in an animated context, especially given the requirements that there is only a single collision and that the forward velocity is more or less constant.

4.3.1 | Overview

This section describes the pilot experiment design, invariants and variables. The invariants as described in Section 4.2 are taken into account, and shadow is not rendered at all.

The characters are placed on the ground (xy) plane such that the distance between their meshes (see the definition of $d(A, B)$ in Section 2.1) is D_m metres. Both are initially placed in a random pose at the origin, and then moved apart along the x-axis until the desired distance is obtained. A negative value for D_m models the penetration depth. For simplicity of computation, we use a reasonable approximation, and define it as the minimum distance to travel along the x-axis in order to separate the two meshes[1]. To generate such cases we use a two-step approach. First the characters are placed at $D_m = 0$ as described above, then they are both moved along the x-axis towards the origin by $D_m/2$; by moving both meshes, the collision will still be near the origin. The line segment L is defined by the closest points on the two meshes, hence $||L|| = \max(0, D_m)$ (assuming L is unique). When $D_m < 0$, $||L||$ is a degenerate line segment, and defined as the point where the meshes touched in the first step of the two-step approach we described earlier.

The camera is placed at a random distance D_c to the centroid of L and a random angle, and an image is rendered. A selection of these images are then used in the user experiment; details are presented in Section 4.3.2. The *front*

---

[1] Although theoretically there is the possibility that this metric is very different from the penetration depth, in our test cases this difference is only small.

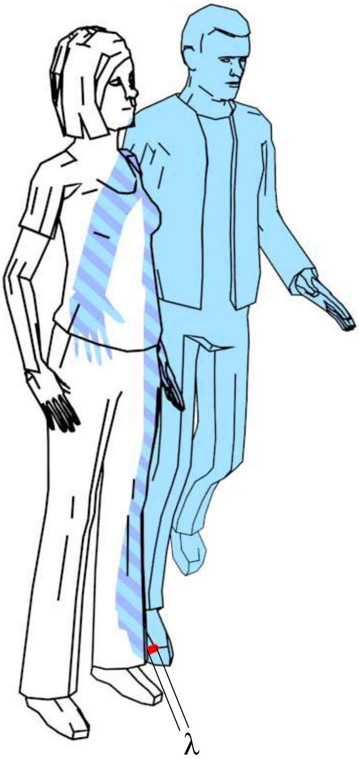
and *rear* character are defined respectively as the characters closest to and furthest from the camera, based on their root joint positions.

Our pilot experiment considers three variables. The first two variables are randomly sampled from a suitable distribution, and used as input to generate the images used in the experiment. The other variable is derived from the randomly generated scene.

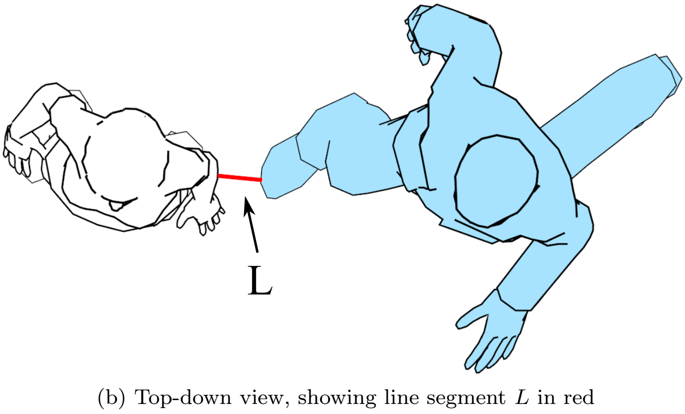
- Mesh-mesh distance D_m was chosen uniformly from the interval $[-0.09, 0.15]$, in metres. We do not use any image with $|D_m| < 0.001$.
- Camera distance $D_c \in [4, 16]$, from camera to centroid of L, in metres. In the case of animated characters, Kulpa et al. [KOOP11] found that up to a certain projected size of the characters camera distance had little influence on accuracy. We are interested to see if this holds for static situations as well. This variable is chosen from an exponential distribution, such that more samples are chosen at smaller distances. When a character is close to the camera, perspective distortion is stronger and resulting effects are easier to measure. The lower bound is chosen such that characters fit entirely inside the camera frustum.
- Variable $\lambda \in [0, \infty)$ measures the length of the visible (i.e. not occluded by the front character) part of L, measured in metres. λ is undefined when the characters are colliding, as there is no visible gap between the characters in those cases. Note that this metric does not denote the gap between the *silhouettes* of the characters; there are many cases in which the visible part of L lies in front of the rear character, such as depicted in Figure 4.2a.

We want to investigate the role of these variables in the perception of collision detection. Given a configuration of two characters, variables D_m and λ are relatively hard to compute, since computing L is a non-trivial task. This means that these variables cannot be directly used in a crowd simulation system. Nevertheless, we suspect that they model important aspects of the perception of the observers. With respect to variable D_m, we expect the accuracy to be the lowest around $D_m = 0$, with a linear positive dependency between $|D_m|$ and the accuracy of observers. We expect a positive linear correlation between the accuracy and λ, as a more visible 'gap' should result in higher accuracy.

Note that the camera angle is not directly part of the variables we consider. Even though the angle of the camera with respect to the walking direction of the characters has been shown in previous work to be relevant to perception [KOOP11], our characters do not share a single direction, hence this metric loses its meaning.



(a) Front view as seen from the camera, showing λ as the visible part of line segment L in red. The striped area indicates the occluded part of the rear character.



(b) Top-down view, showing line segment L in red

FIGURE 4.2: Front and top-down view of the experiment setup.

4.3.2 | EXPERIMENT

To generate the images, the characters were posed using a randomly selected frame from a motion capture corpus consisting of stepping and walking motions. An additional random orientation around their up-axis prevented correlation between the test cases and the absolute orientation recorded in the motion capture lab. The images are rendered at a resolution of 800×600 pixels.

Each variable's range is uniformly split up into eight bins, as shown in Figure 4.3. Each image is assigned a bin index for each of the three variables. The interval $[-0.09, 0.15]$ metres allows us to place $D_m = 0$ at a bin boundary, separating colliding and non-colliding images into different bins. A random sampling technique described by Wand and Straßer [WS02] is used to ensure at least 22 images per bin, resulting in a total of 373 images.

Participants are presented with an online web-based questionnaire, as described in Section 4.2. Each image is shown for 6 seconds and is then hidden; this timeout ensures that all participants look at an image for a more or less equal duration. In order to prevent bias towards positive or negative answers, we include both colliding (i.e. $D_m < 0$) and non-colliding (i.e. $D_m > 0$) images in the experiment. An exact 50%/50% distribution for any single participant who completes the survey is ensured, and approximated for participants that do not.

Per participant, a random subset of the test cases is shown. This allows us to use a large test set without forcing participants to answer all 373 questions. Image selection is biased towards images in bins containing the least number of answers, providing a more even spread of answers over the bins than when uniformly selecting images.

4.3.3 | RESULTS

A total of 212 actively participating users provided 9179 answers, averaging 43 answers per participant. The accuracy over all participants was 72% for this experiment.

For each variable, a graph is shown in Figure 4.3. These graphs show the likelihood that the participants correctly identified the situation, averaged over the images in each bin. The solid blue graph shows the *average accuracy* per bin, with the error bars indicating the standard deviation. The scatter plot shows a dot for each image in the survey. The dark red dashed graph shows the *trend*, and consists of one or two linear pieces.

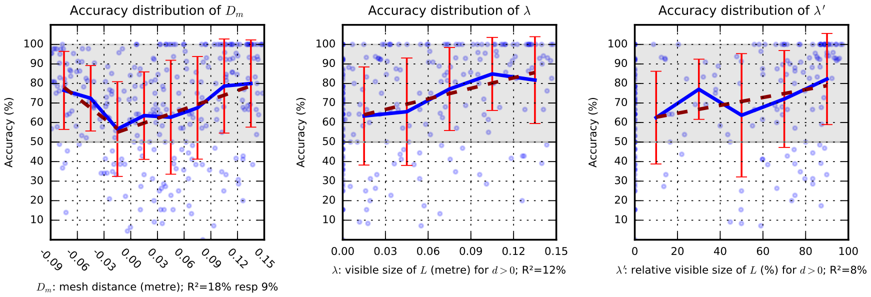


FIGURE 4.3: Plots of the results of the static experiment. Results are binned; the blue, continuous line shows the accuracy of each bin, with red error bars denoting the standard deviation. The red dotted line shows the linear fit of the accuracy. Note that the R^2 numbers relate only to a single parameter of our model with respect to the entire variance in the observations.

To define the *trend* of the accuracy, we investigate, in order of simplicity, a linear function or a piecewise linear function. We accept the simplest function that describes the data well. An analysis of the variance shows how well the found trend fits the data (R^2).

As the projected size of λ is dependent on D_m, we also investigated the visible percentage $\lambda' = \lambda / D_m$ to remove this dependency. However, as can be seen in Figure 4.3, the results are less relevant than for λ ($R^2 = 8\%$).

The effect of camera distance D_c (not included in graph) is negligible, resulting in $R^2 = 0\%$ for camera distances up to 12 metres. This confirms that the findings by Kulpa et al.[KOOP11] are also applicable to static situations.

Computing the average over all participants, we see that 53% of the answers was ‘not colliding’ and 47% was ‘colliding’. We follow the binomial approach detailed in Section 2.4.2 to compute the number of participants that are neutral, err towards false positives, or err towards false negatives. Using a 95% confidence interval, none of the participants had a significant bias towards false positives, i.e. incorrectly answering ‘colliding’. 163 participants did not show any bias, whereas 115 participants showed a significant bias towards false negatives, i.e. incorrectly answering ‘not colliding’. These results may impact strategies for collision detection, as those are generally aimed at the prevention of false negatives; this analysis will be performed in the main experiment as well, to investigate whether the same bias towards false negatives is seen in animated situations.

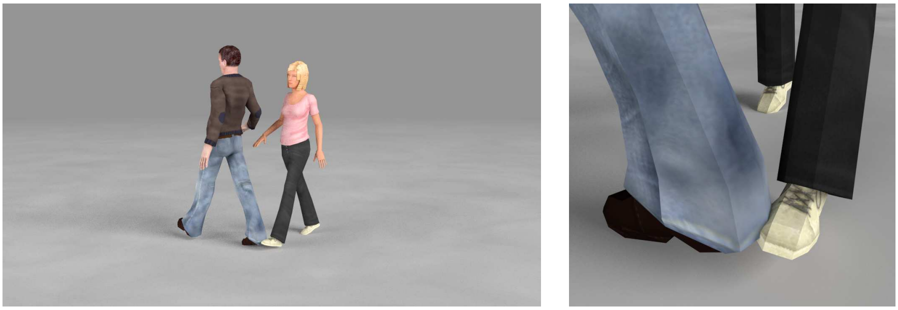


FIGURE 4.4: Still from one of the videos used in the animated experiment. Only 9% of the participants recognised the left scenario as a collision. The feet of the characters intersect, as can be seen from the side view on the right.

4.4 | MAIN EXPERIMENT

In this section we describe our main experiment, in which we investigate the ability of observers to recognise collisions between virtual characters in animated scenarios. Using animated characters, we aim for our results to be applicable to other animated situations, such as crowds of virtual characters.

4.4.1 | OVERVIEW AND VARIABLES

This section provides an overview of the main experiment design. The two characters are animated using previously recorded motion capture data of a person walking in a straight line. Collision responses were not animated; participants could not discriminate colliding from non-colliding scenarios by looking at the animated behaviour. Figure 4.4 shows an example still from one of the animations.

The same invariants as in Section 4.2 are taken into account, albeit with two differences with the pilot experiment. Firstly, to improve the perceived realism, and to visually ground the characters on the floor plane, shadows are rendered. These are very soft (i.e. no hard edges) by employing multiple, large light sources, preventing a second angle of view onto the scene. Secondly, since Kulpa et al. [KOOP11] showed that there is no statistical significance of the camera distance, the camera is placed at a fixed distance of 6.6 metres from the collision point.

The characters are placed on the ground (XY) plane, such that their paths cross at an angle α. By modifying this angle, the starting positions, and animation offsets, a total of 16 colliding and 16 non-colliding scenarios were constructed. To allow reasoning about *the* collision, we make sure that for

Label	**Colliding**: I_V	**Label**	**Non-colliding**: D_m
LOW	0.5 cm^3s	LOW	0.5 cm
MODEST	12.5 cm^3s	MODEST	1.0 cm
CONSIDERABLE	67.2 cm^3s	CONSIDERABLE	3.0 cm
HIGH	132.0 cm^3s	HIGH	5.0 cm

Table 4.1: Severity labels for the colliding and non-colliding cases, based on a small pilot experiment.

the colliding cases there was only a single, continuous time interval in which the characters were intersecting. As a result, $\alpha = 0$ could not be investigated, as it would be impossible to create a scenario with a single collision of the intended severity.

To give the best view of the (near) collision the camera is placed on either the positive or negative side of one of the two bisectors of the character's paths. For each video we manually select which of those four possible positions provides the best view. This provides us with a worst case scenario, as when looking at animated characters in general, often the user will not have an unobstructed view of the collision.

Our experiment considers four variables, defined below. The first two are selected from a set of predefined values. The character animation is adjusted as described earlier, to produce a video that adheres to those parameters. The second two parameters are derived from this animation, and allow us to perform more statistical analyses on our data in order to find out which component is important for the recognition of collisions. The variables are defined as follows.

- Character angle $\alpha \in \{45, 90, 135, 180\}$ degrees. This defines the angle between the forward vectors of the characters.
- The severity S of the (near) collision labeled as LOW, MODEST, CONSIDERABLE or HIGH, and expressed either as intersection volume integrated over time (I_V) when colliding, or as the minimum mesh distance (D_m) otherwise. See Table 4.1 for the values used; each scenario used one of the displayed values, precise up to one decimal.
- Collision duration τ is derived from the animation created to obtain the first two parameters. This variable is only defined for colliding videos.
- Average intersection volume $I_A = I_V/\tau$. This variable is defined only for colliding videos.

The severity labels have different meaning for colliding and non-colliding cases. In the colliding cases, there is a temporary overlap between the two characters. This is expressed in the size of the intersecting volume (in cm^3) integrated over time (in seconds), giving us the integral I_V in cm^3s. Both aspects (size and duration) are important to quantify the potential recognisability of the collision, as even a small intersection will be seen when existing for a long enough time. In the non-colliding cases, the severity was defined as the minimum distance between the meshes D_m, which is identical to the D_m parameter as defined in Section 4.3.1, except that now this minimum is taken over the entire spatio-temporal domain.

To find a suitable range for the collision severity, we have conducted a small pilot experiment with three participants. It took the same form as the actual experiment, and used the following values for the variables:

- $\alpha \in \{45, 90, 135, 180\}$ degrees
- $I_V \in \{0.5, 30, 150, 300\}$ cm^3s
- $D_m \in \{0.00, \pm 0.05, \pm 0.10, \pm 0.20\}$ metres

This small pilot showed that for the larger values of the collision severities I_V and D_m, in respectively the colliding and non-colliding cases, it was very easy to recognise a (non-)collision. Removing these values allows us to have a finer granularity in the lower, more interesting range, without increasing the number of required videos. As stated before, the values used for the final experiment are shown in Table 4.1.

4.4.2 | IMAGE GENERATION AND QUESTIONNAIRE

This section describes the details of the main user experiment. Two textured character models were selected for the experiment; every image uses the same two models with the same textures to prevent dependence on the appearance. We ensured a high contrast between arms and legs of both characters, to make distinguishing the two characters as easy as possible.

We used four slightly different walking motions to reduce the learning effect, and ensured that the two characters never used the same motion in the same video. For each combination of variables, and for both colliding and non-colliding, we generated a video at a standard resolution of 1280×720 pixels at 30 frames per second. Each video was 2.5 seconds long. The time of the (near) collision was randomly chosen between the 50% and 75% mark, to prevent a learning effect. The starting position of the character and the offset into the walking animation were chosen manually, in order to be able to ensure a (near) collision of the intended severity.

Participants are presented with an online web-based questionnaire (see Section 4.2 and Figure 4.1). Each participant is shown all 32 video clips in random order. Each clip is played once, advancing to the next video after the question ‘Do these characters collide?’ is answered.

4.4.3 | RESULTS

The data of the animated experiment is based on the answers of 164 participants, each providing exactly 32 answers. In total 195 people participated; 30 did not complete the survey, and one participant completed the survey on a mobile device. That mobile user’s data was not considered, as the small size of the screen makes recognising collisions harder. The accuracy over all participants was 68% for this animated experiment. Note that the difference between this result and the 72% accuracy observed in the pilot experiment does *not* imply that people are better at recognising static scenarios, since there are more differences between the experiments than just being static or animated. The hardest to recognise collision is shown in Figure 4.4. For the colliding cases, i.e. the scenarios in which $FP = TN = 0$, the accuracy is 58%. For the non-colliding cases, i.e. in which $TP = FN = 0$, the accuracy is 79%.

In order to understand the relation of our variables α and S, and the expected accuracy $E[A]$, we apply linear regression analysis (see Section 2.4.1). The analysis is performed on all answers, and not just on the averages per bin; this implicitly takes variance of the answers into account. Firstly, we linearise our input by finding as simple as possible functions $f_1(\alpha)$ and $f_2(S)$. Secondly, we use a statistical software package to find the best-fitting B_0, B_1, B_2 and B_3 such that:

$$E[A] = B_0 + B_1 f_1(\alpha) + B_2 f_2(S) + B_3 f_1(\alpha) f_2(S)$$

Since S is expressed differently for colliding and non-colliding scenarios, we perform the linear regression method for each separately.

Before applying the linear regression analysis, we need to find suitable functions $f_1(\alpha)$ and $f_2(S)$. The α graph in Figure 4.5 shows a more or less sine-like shape, which could indicate a relation between $E[A]$ and the size of the projection of one of the trajectories onto the other. We choose $f_1(\alpha) = \sin(\alpha)$; as rotations are periodic, we expect the influence of α on $E[A]$ to be periodic as well, supporting the choice for a periodic linearisation[2]. The I_V graph is fairly linear, except for the data point at LOW. We use $f_2(I_V) = I_V$, but we may consider a different linearisation in the future; we will get back to this in

---

[2] We also investigated $f_1(\alpha) = \alpha$, $f_1(\alpha) = \alpha^2$, $f_1(\alpha) = \cos(\alpha)$, and $f_1(\alpha) = \sin(\alpha + \pi/4)$, but all these variants resulted in a lower R^2 than $f_1(\alpha) = \sin(\alpha)$.

	Coefficients		**Standardized Coefficients**	**Significance**	
	B	Std.Err.		t	p
(Constant)	$B_0 = 0.89$	0.10		8.98	0.000
$\sin(\alpha)$	$B_1 = -0.73$	0.13	$\beta_1 = -0.77$	−5.54	0.000
I_V	$B_2 = 0.01$	0.00	$\beta_2 = 0.02$	3.21	0.008
$\sin(\alpha) \times I_V$	$B_3 = 0.00$	0.00	$\beta_3 = 0.52$	−1.95	0.074

TABLE 4.2: Linear regression model for the colliding cases; $R^2 = 78\%$.

Section 4.5. We feel that the accuracy distribution curve of D_m is sufficiently linear, resulting in $f_2(D_m) = D_m$.

Results of the linear regression are shown in Tables 4.2, 4.3, 4.4, and 4.5, with the B_i coefficients in the second column. The R^2 value mentioned in each caption denotes the percentage of the variance explained by these models. Any row with $p < 0.05$ is considered *significant*, and with $p < 0.01$ considered *strongly significant*.

When the characters are colliding, a linear combination of $\sin(\alpha)$ and I_V predicts 78% of the variance in A (see Table 4.2). The interaction between the two variables is not significant ($p = 0.07$). Since I_V is the volume of the intersection integrated over time, we can split its value into average volume I_A and duration τ, to investigate which aspect is more important to the correct classification of the video by observers. This results in the model shown in Table 4.3, with $R^2 = 80\%$. Even though there is variation in A that we did not capture in our model, such as the characters' exact poses at the moment of collision, our model is significant to A. The interaction between I_A and τ is expressed as I_V, and is not included in this analysis due to its insignificance. Interestingly, with $\beta_1 = -0.77$, the angle between the characters is the most important factor. The sign of β_1 indicates a negative correlation, as the minimum accuracy was measured at $\alpha = 90^o$. The duration of the collision is slightly less important, with $\beta_3 = 0.52$. The average volume of the collision is insignificant ($p = 0.879$).

In the non-colliding cases, the model based on $\sin(\alpha)$, D_m, and their interaction seems to predicts 47% of the variance in A (see Table 4.4). However, since a linear model always produces a better fit when there are more parameters, we have to remove the non-significant parameters from the analysis[3]. This is also reflected in Figure 4.5, which shows how $\sin(\alpha)$ alone explains 52% of the variance of the colliding cases, but only 2% of the non-colliding cases. The model based on only D_m, shown in Table 4.5, results in $R^2 = 34\%$. Even though this R^2 is moderate, the results are significant.

---

[3] This was not necessary for the analysis shown in Table 4.3; due to the very small β_2, the outcome would not change significantly.

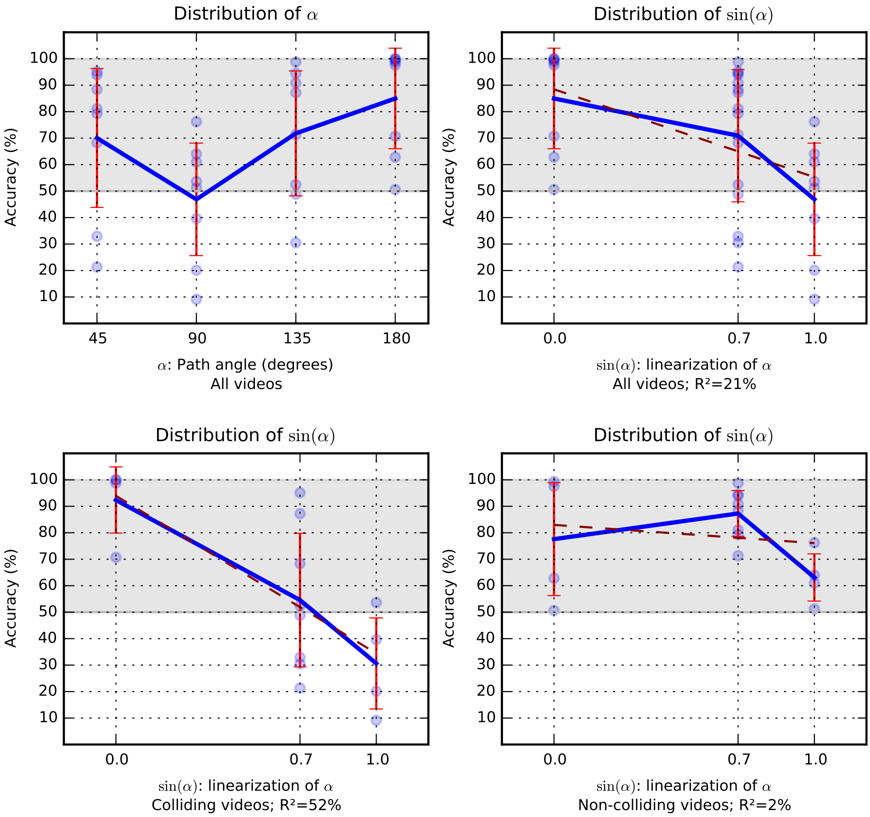


FIGURE 4.5: Plots of the contribution of α to the results of the animated experiment. The blue line shows the accuracy of each bin, with red error bars denoting the standard deviation. The dashed red line indicates the linear fit.

	Coefficients		**Standardized Coefficients**	**Significance**	
	B	Std.Err.		t	p
(Constant)	$B_0 = 0.72$	0.10		7.58	0.000
$\sin(\alpha)$	$B_1 = -0.63$	0.11	$\beta_1 = -0.77$	5.88	0.000
I_A	$B_2 = 0.00$	0.00	$\beta_2 = 0.02$	0.16	0.879
τ	$B_3 = 1.47$	0.45	$\beta_3 = 0.52$	3.29	0.007

TABLE 4.3: Linear regression model for the colliding cases; $R^2 = 80\%$. Interaction between the variables is insignificant ($p > 0.7$) and not included in this table.

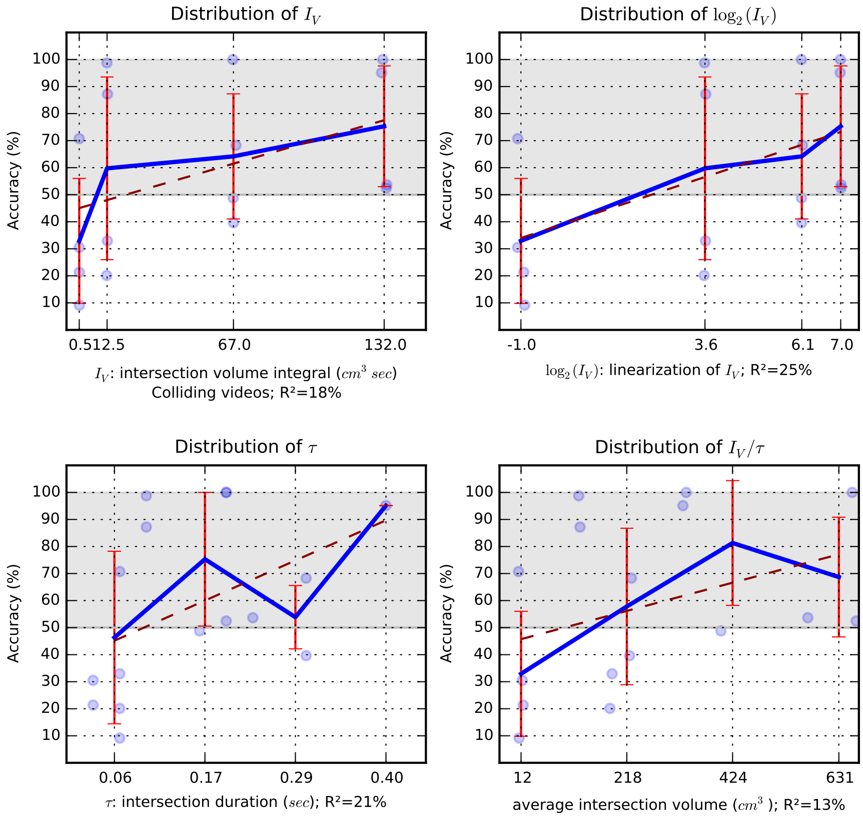


FIGURE 4.6: Plots of the contribution of I_V, I_A and τ to the results of the animated experiment. The blue line shows the accuracy of each bin, with red error bars denoting the standard deviation. The dashed red line indicates the linear fit. The graph of $\log_2(I_V)$ is discussed in Section 4.5.

	Coefficients		**Standardized Coefficients**	**Significance**	
	B	Std.Err.		t	p
(Constant)	$B_0 = 0.59$	0.11		5.32	0.000
$\sin(\alpha)$	$B_1 = 0.12$	0.16	$\beta_1 = 0.28$	0.79	0.446
D_m	$B_2 = 0.10$	0.04	$\beta_2 = 1.12$	2.78	0.017
$\sin(\alpha) \times D_m$	$B_3 = -0.08$	0.05	$\beta_3 = -0.76$	−1.54	0.148

TABLE 4.4: Linear regression model based on $\sin(\alpha)$ and D_m for the non-colliding cases; $R^2 = 47\%$.

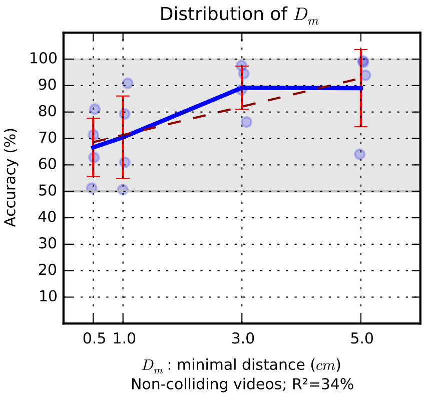


FIGURE 4.7: Plot of the results of the animated experiment. The blue line shows the accuracy of each bin, with red error bars denoting the standard deviation. The dashed red line indicates the linear fit.

	Coefficients		**Standardized**	**Significance**	
	B	Std.Err.	**Coefficients**	t	p
(Constant)	B_0=0.66	0.06		11.08	0.000
D_m	B_1=0.05	0.02	β_1 = 0.59	2.70	0.017

TABLE 4.5: Linear regression model based on D_m for the non-colliding cases; $\mathrm{R}^2 = 34\%$.

Computing the average over all participants, we see that 60% of the answers was ‘not colliding’ and 40% ‘colliding’. To obtain more detail of the nature of this apparent bias, we investigate the ratio of false positives and false negatives for each participant. A two-tailed binomial test using a 95% confidence interval (as detailed in Section 2.4.2) showed that 72 of the participants did not have a bias, and no participants had a bias towards false positives. The remaining 92 participants had a bias towards false negatives; in other words, the majority of the participants significantly erred towards answering ‘not colliding’.

4.5 | DISCUSSION

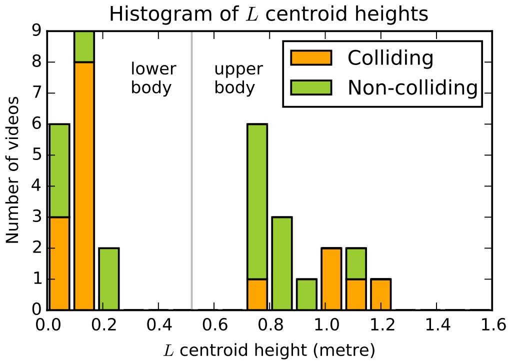
Looking at our findings, we can conclude that in general the subjects were better at recognising non-collisions than collisions. The results from both experiments show the same trends, and although they cannot directly be mapped onto each other, this could indicate that trends observed in static situations may be applicable to animated scenarios as well.

In our main experiment, an intersection volume integral of 0.5 cm^3s resulted in an accuracy of only 33%. Apparently, for the average observer, it is the most difficult to classify a scenario as 'colliding' or 'not colliding' when there is a small amount of interpenetration. This is also observed in the pilot experiment, where we see a remarkable dip in accuracy in the interval $D_m \in [-0.03, 0.00)$. The bias towards answering 'not colliding', observed in both experiments, corroborates these observations. This knowledge may be used to speed up collision detection algorithms. A simplified version of the mesh could be created, taking care that it is an 'inner approximation' bounded by the original mesh. By ensuring a Hausdorff distance [Hau14] of at most 1.5 cm the total penetration of two such meshes would be at most 3.0 cm and fall within the interval of minimal accuracy. The algorithm to create a simple mesh that meets those requirements, and the effect on both the perception of collisions and the performance of collision detection, is an interesting open problem. These observations also seem to indicate that, for collision detection between humanoid shapes, a bounding volume collision detection scheme may not be the best choice. Employing a *bounded* volume method representing an inner approximation could be more efficient, and a better match for our perception.

Intersections are allowed in certain commercial crowd simulation systems, such as the implementation in IO Interactive's *Hitman: Absolution*; apparently large game companies assume that people do not mind such partial intersections [SEE14]. Such an approach also allows for denser crowds, simply by decreasing the personal radius of the characters, without sacrificing too much believability. The low importance of the intersection volume I_V (see Table 4.2), coupled with the high importance of the angle α also suggests varying the effective personal radius for collision checking based on the angle between the paths of the checked characters.

From the λ accuracy graph, it is clear that the more visible the gap between the characters, the easier it is to see that they do not collide. Note that this metric does not denote the gap between the *silhouettes* of the characters - there are many cases in which the visible part of L lies in front of the rear character.

Alternate linearisations for α, I_V and D_m might produce a better fitting model. The I_V accuracy graph resembles a logarithmic curve, so we have also investigated $f_2(I_V) = \log_2(I_V)$, resulting in the curve shown in the top left of Figure 4.6. This model is a tighter fit for the data ($\text{R}^2 = 82\%$ for the entire model, instead of 80%). However, even though visually a logarithm may fit the graph well, this does not imply that it is the best model to use. For



(a) Animated experiment

FIGURE 4.8: Histogram of the height of the (near) collisions, clustered into 'upper' and 'lower' collisions.

this reason, we have kept the linearisation simple, and leave more complex linearisations to future research.

By using the height of the (near) collision, we separate the stimuli into 'upper body' and 'lower body'. Figure 4.8 shows the distribution of the height of the collision, or the average height of the two closest points in non-colliding cases, and confirms that such a distinction is sensible. We use k-means clustering (k=2) to separate the test cases into 'upper body' and 'lower body' clusters, and apply the same analysis as before to each cluster individually. The accuracy in the main experiment shows a remarkable difference, with $A = 54\%$ and $A = 85\%$ for respectively the upper and lower body collisions. The overall FN:FP ratio is also different, with 19% : 81% for the upper body and 80% : 20% for the lower body. A per-participant binomial bias analysis showed interesting differences between upper-body collisions and lower-body collisions: We observed a stronger bias towards answering 'not colliding' for lower-body collisions than for the upper-body collisions (see Table 4.6). These results for upper- and lower-body differences are only preliminary; for every

Collision height	**FP**	**Neutral**	**FN**
Entire body	0	72	92
Upper body	0	86	78
Lower body	0	130	34

TABLE 4.6: The number of participants showing a biased error. 'FP' shows a bias towards false positives, i.e. perceiving a collision where there is none. 'FN' shows a bias towards false negatives, i.e. perceiving no collision when in fact there is one. 'Neutral' shows the number of participants that had no such bias.

pair of parameters (α, S) there was only one sample, i.e. only an upper or a lower body collision. We leave studying these differences to future work.

4.6 | CONCLUSION

In this chapter, we have conducted a perceptual experiment to determine the accuracy of human observers in determining whether two virtual characters collide. We have identified an asymmetry in the recognition of collisions, a critical penetration depth interval where the accuracy is minimal, and proposed a level of detail technique that utilizes this knowledge to speed up collision detection. New collision response criteria that increase performance and allow denser crowds by focusing on pairs of characters have been introduced.

Care should be taken in those cases where crowd behaviour is changed based on any camera-related metric. When crowd simulation is used to mimic real humans, for example to evaluate evacuation scenarios, such view-dependent behaviour will change the outcome of the simulation. When crowd simulation is used in games, view-dependent behaviour could be exploited to gain unfair advantage over other players. For example, one could turn off collision detection of crowd agents when they are not in view of the player; this would make traversing a crowd easier when walking backward than when walking forward.

Simplified shapes are often used in physics simulation software. Future research could investigate whether the results in this chapter are applicable only to humanoid shapes or generalize to other objects or even abstract geometric shapes.

In our surveys we have not rendered crisp shadows and ambient occlusion. This simplifies rendering; shadowless rendering is also used in commercial applications [SEE14] to enable real-time rendering of crowds. It would be interesting to see the effect of different types of shading and lighting on the perception of the crowd in general and collisions in particular.

The backgrounds were rendered as simple as possible, to ensure our results depend only on the two virtual characters and the camera position. The effects of the background behind the characters, especially when visible in the space between the characters, is still an open research question.

We observed an asymmetry in the recognition of collisions, and a bias towards answering ‘not colliding’. These effects could have several causes. Firstly, the characters did not employ a collision response animation. Because of this, and because real humans do not intersect each other when colliding, the non-colliding and colliding scenarios could be classified as respectively realistic and unrealistic, causing this bias towards realistic scenarios. Secondly, participants may have focused on the spot where they anticipated a collision. In cases where they anticipated incorrectly, such focus may have caused them to miss the collision. Since the other way around cannot occur, this likely contributed to the observed bias. Thirdly, we also observed that most of the collisions occurred between the hands or the feet. This was likely caused by the use of a simple walk animation that was not adapting to the proximity of the other character. Real humans would probably be able to slightly change their hand or foot position to avoid a collision without changing their own global position or trajectory. Expecting such behaviour may have also accounted for the bias towards answering ‘not colliding’. In the next two chapters, we use this bias by choosing a representation that allows for some undetected collisions. In future work, it would be interesting to see how collision avoidance and response animations influence this bias specifically, and the perception of collisions in general.

CHAPTER 5

AN ANALYSIS OF MANOEUVRING IN DENSE CROWDS

While walking through any crowd, a person balances several desires, such as reaching some goal position, avoiding collisions with others, and conserving energy. Crowd models generally try to mimic this behaviour by planning short paths that avoid collisions. However, when the crowd density increases, choosing a collision-free path becomes more difficult. Furthermore, in such high-density crowds, one can observe regular steps, side-steps, and *torso twists*; people rotate their upper body to decrease their width perpendicular to the motion path, in order to squeeze through the narrow spaces between other crowd members. Rather than *walking*, such motions are better described as *manoeuvring* through the crowd. In our daily lives, we often face such crowded situations, such as a busy lift, bus or pop concert. In such cases, the members of the crowd are often stationary, moving only when someone wants to pass.

In this chapter, we describe an experiment aimed at understanding such dense crowd manoeuvring behaviour, by recording and analysing a dense crowd. Apart from the common approach to crowd analysis, where only the position of each person in the crowd is recorded, we also record and analyse the torso orientations. To the best of our knowledge, this has not been done before in the context of dense crowds. The underlying motivation is to obtain a data set that supports an implementation of such behaviour in a crowd simulation algorithm (which we implement in Chapter 6). In our experiment, the participants form a crowd of such density that it is sparse enough to manoeuvre through, but dense enough to require torso rotations in order to do so; an example is shown in Figure 5.1. We explicitly ignore the lower body in our analysis. As we have shown in Chapter 4, small intersections between characters are hard to see in general, and even harder to see in a dense crowd. Furthermore, it has also been shown that the shoulder movements through space are a good indication of the movement of the entire body [ALHB08a]. Our accompanying video at `https://stuvel.eu/video/dense-crowd-manoeuvring` shows one of the trials we have recorded, including aspects that play a role in our analysis, namely a capsule-based representation and a generalized Voronoi diagram of that representation.

In each trial, the person in the centre of the crowd must manoeuvre towards a predefined goal position, while the other participants remain more or less stationary inside a circle drawn around that centre. To reduce ambiguity in the analysis of the data, in each trial only a single participant manoeuvres towards this goal position.

In this chapter, we show that in dense crowds people follow generalized Voronoi diagrams based on a line-segment representation of the agents. Two possible methods for defining those line segments are investigated, and we show that the medial axis of a capsule representation of the torso is a good choice for such line segments. We identify relations between instantaneous speed and plan-ahead distance, and between the torso orientations and goal positions of the moving participants. We also show that there is no apparent correlation between linear and angular velocities of those participants' torsos.

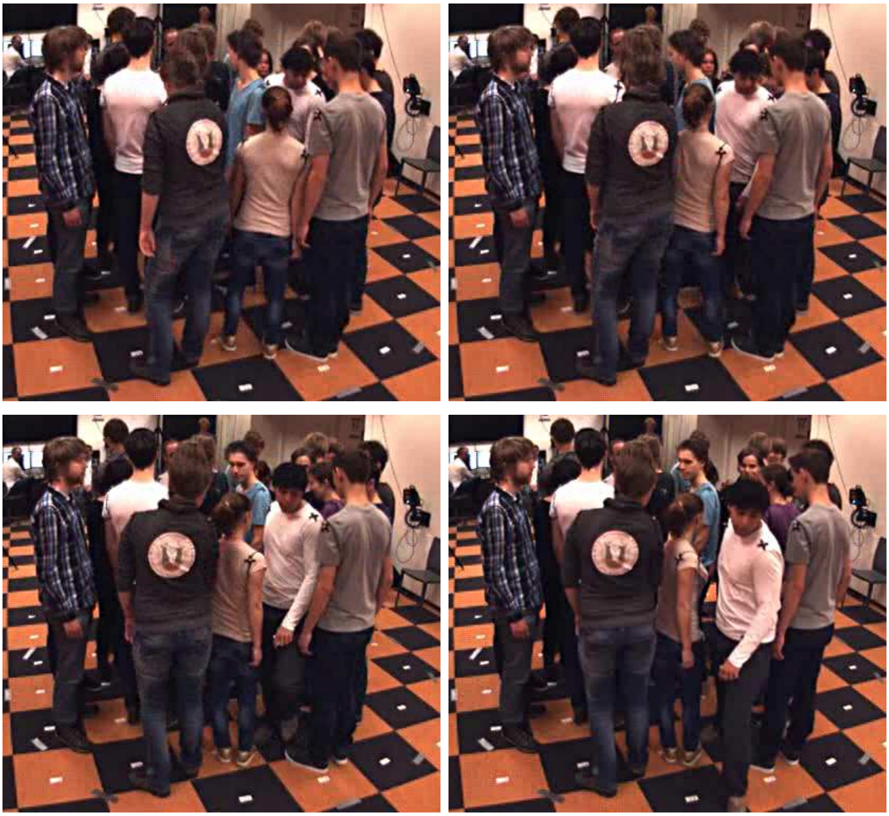


Figure 5.1: Stills of one of the trials, showing a participant escaping the crowd.

The rest of the chapter is organised as follows. Section 5.1 discusses related work. The details and execution of the experiment are described in Section 5.2, with analysis of the results in Section 5.3. The implications and possible future work are discussed in Section 5.4, which concludes the chapter.

5.1 | RELATED WORK

The study of crowd behaviour spans a large research area, ranging from computer vision techniques to assess human behaviour [ZL14] to the simulation of evacuation scenarios [ZZL09], and application in (serious) games. In this section, we focus on the works relevant to the study of *dense crowds*.

Two possible ways of simultaneously capturing the motions of multiple individuals in a crowd are video recordings and motion capture systems. *Video recordings* are most often used; as these require no markers to be attached to the participants, the data are easier to obtain. A common approach to process a video stream is either pure visual comparison [NGCL09], or the use of feature point tracking in order to determine movement of people in a crowd [SBTM08, RSLA11, CKGC14]. Feature point tracking is also used by Lee et al. [LCHL07] to *train* a crowd simulation system such that it exhibits behaviours imitating real human crowds. The opposite of using video tracking to improve crowd models is also possible, by using a crowd simulation model to improve the result of feature tracking in videos [MOS09, BM14]. All these works use the video data to determine the positions of the recorded people; under the assumption of nonholonomic motion (see Section 2.2.1), the orientation is determined from the velocity vector. Such techniques are not suitable for studying the torso twisting behaviour in crowds, as firstly the position alone is not enough to study the torso twist, and secondly the nonholonomy assumption is no longer valid for dense crowds [MTL10, TFP^{+}10]. For these reasons, we have chosen to not employ video tracking for our experiment in favour of using a *motion capture system*, which can precisely track individual body parts.

Motion capture systems have been used by Wolinski et al. [WJGO^{+}14] to optimize parameters for various crowd simulation systems to increase similarity between the simulated and recorded crowd behaviour. Jelic et al. [JARLP12] and later Lemercier et al. [LJK^{+}12] used motion capture to study following behaviour in crowds, by recording people following each other in a circular fashion. In all those works, the participants are represented as moving discs on the ground plane, even though a motion capture system might have been able to capture more detailed motions. Hence, as with the video-driven techniques described earlier, the resulting data are unsuitable for the study of torso twisting behaviour. Truong et al. [Tru10] used a motion capture system to study the nonholonomic and holonomic behaviours in human motion, and extend the work by Arechavaleta et al. [ALHB08b] to include holonomic motion. They distinguish between those two behaviours by observing the orientation of the shoulders with respect to the velocity vector of the body. To capture the motion, three motion capture markers are placed on the upper body and three on the legs. Truong et al. observed only a single participant at the time, allowing for an unobstructed view from the motion capture cameras. We use a similar approach, except that in our experiment we actually study a crowd, so any marker below the shoulders is likely to be occluded by the other participants. Therefore, we use markers on the shoulders only. The motion of the line segment spanned by those markers has been shown by Arechavaleta et al. [ALHB08a] to be a good fit for human motion prediction, further strengthening our choice to investigate torso motions. In this chapter we improve on the work by Arechavaleta et al., in the context of the analysis of dense crowd manoeuvring, by defining the length of the line segment in a different way.

Current crowd simulation techniques are able to simulate crowds of high densities. However, most represent agents as points [JCG13] or discs [vdBGLM11, KSG14, CGZM11, NGCL09] (also see Section 2.2). By employing such a rotationally symmetric representation, planning the torso twist is impossible.

In our experiment, we compare the motions of people in dense crowds to the motions predicted by a Generalized Voronoi Diagram. It has been shown that such diagrams can be used to steer medium-density crowds [SAC+08]. However, it is unknown whether people in dense crowds also follow a Voronoi Diagram. To our knowledge, we are the first to investigate this. Another aspect we investigate is the relation between linear and angular velocities of the actively manoeuvring participants. Hicheur et al. [HVR+05] have shown that a close relationship exists between linear velocity and path curvature, but have not investigated the angular velocity of the torso. We expect to find a negative correlation between linear and angular velocity; turning at higher speeds may be harder (to plan for), while at the same time it may be harder to walk faster while turning.

5.2 | EXPERIMENT

This section describes the experiment design, execution, and representation of the participants in our analysis. In short, the goal of the experiment is to obtain information on how people manoeuvre through dense crowds. We observe and analyse torso motions; even though we do not explicitly consider hip rotations, their effect is encoded in the motion of the torso.

5.2.1 | Experiment design

The experiment consists of a repetition of trials. In each trial, the person in the centre of the crowd, the designated *walker*, manoeuvres towards a predefined goal position, while the other participants remain more or less stationary. The experiment is aimed at providing ground truth data, which can be analysed to further understand crowd behaviour, as well as provide empirical data to improve crowd simulation methods. We are particularly interested in the following aspects:

- Typical values for linear and angular velocity, for the crowd densities we consider, and the possible relation between them;
- Typical values for the angle between the normal of the torso and the direction towards a goal position;
- A geometric model that predicts the chosen path through the crowd.

When recording people using an optical motion capture system, marker occlusions are very common. When the number of people increases, the chance that occlusions occur also increases, and more so when those people stand closer together. This makes it impossible to capture the full-body motion of each member of a large, dense group. Consequently, we chose to reduce the marker set, and place them on the body in areas that are least likely to be occluded from the overhead cameras. Three motion capture markers are attached to each participant: one on the left shoulder, and two on the right. The asymmetry in the marker layout allows us to distinguish between the participants' left and right shoulders. Each participant started at a predetermined position, simplifying the identification and labelling during the post-processing of the recorded data.

A circle with a radius of 1.75 metres was drawn in the middle of the motion capture studio. All participants except the walker were prohibited from leaving this circle during each trial, to allow for a consistent overall crowd density and a symmetric distribution of the participants around the centre. The location and size of the circle depended on the limitations of the motion capture system as well as the intended crowd density. Around the circle, the letters A-H, printed on A4 paper, were hung 2 meters high and 45 degrees apart, as shown in Figure 5.3. The motion capture system employs fourteen cameras recording at 120 Hz.

The experiment trials were grouped into sets. For each set, a single participant was chosen for the role of walker. For each walker, we recorded a set of seven trials. The walker received a randomized set of task cards: a random subset of six of the eight letters, and a question mark. He/she was instructed to keep

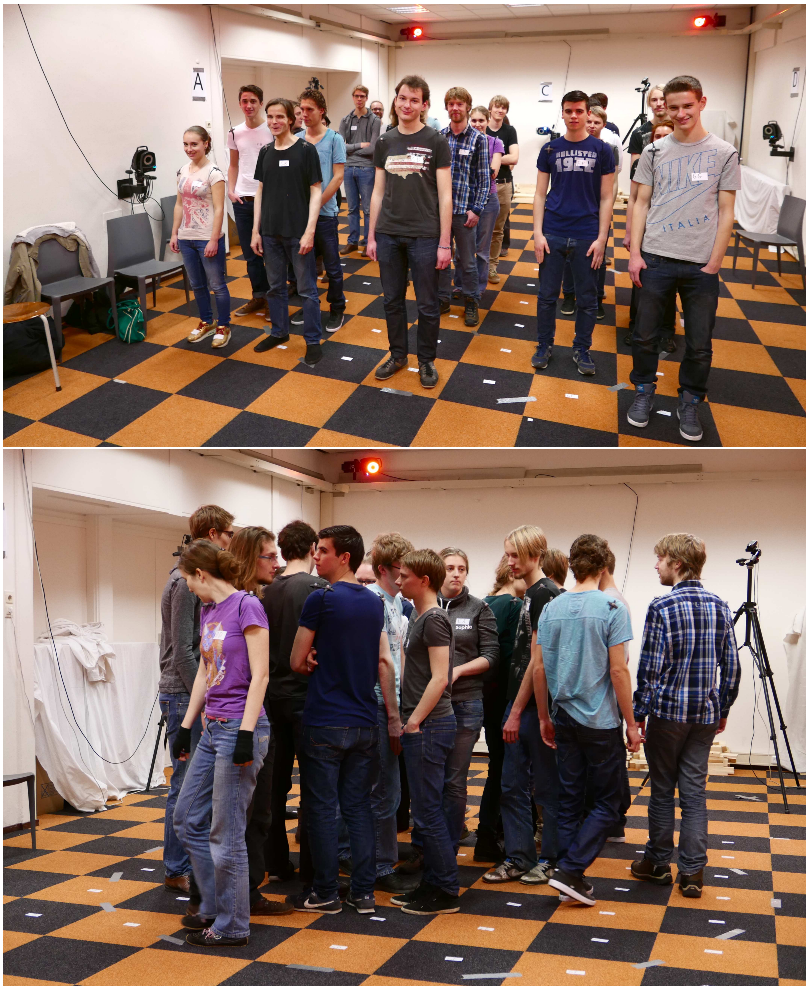


FIGURE 5.2: Predefined starting positions (top) and randomization step (bottom).

these cards hidden from the other participants. As two letters were excluded for each walker, the other participants would not be able to predict the tasks.

Participants were invited on the notion that the intended goal of the experiment was to test the limits of our motion capture lab: to see how many people it can hold, and how many markers it can track simultaneously. This way, the behaviour of the participants would not be influenced by any knowledge about the actual goal of the experiment. Furthermore, the non-walker participants were asked to treat the situation as a densely packed bar, and to crowd together to such a degree that it would be non-trivial but still possible to manoeuvre through. As for dealing with the walker, we asked participants

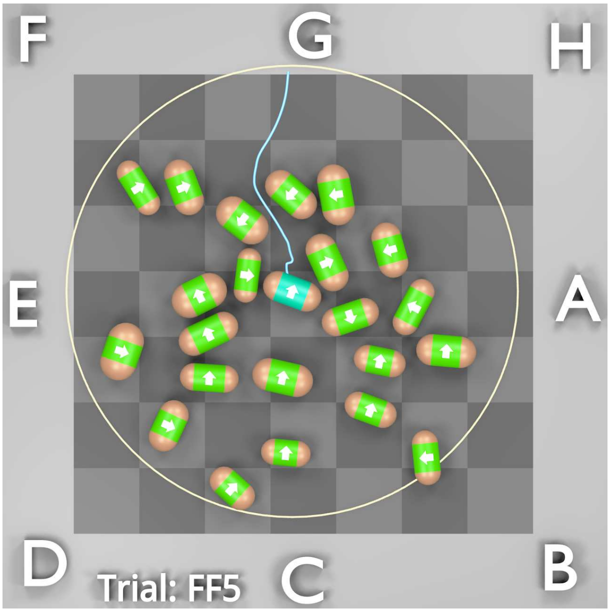


Figure 5.3: Top view of a single trial, showing the blue walker in the starting configuration, and letter G as the goal. The light blue curve indicates the path of the participant.

to let him/her through as they would have in similar situations in real life and not anticipate their movement too much.

5.2.2 | Execution

Each participant was measured for their shoulder width, chest thickness, and distance between the left and right shoulder markers. The latter measurement allows us to model the participants as a line segment translating and rotating on the ground plane, which we used for most of our analyses (see Section 5.2.3). The other dimensions are used to define an alternative capsule-shaped representation, which will discussed in Section 5.3.2. Furthermore, we recorded each participant's name, age, and gender, and starting position for each trial.

Each *walker* was hand-picked from the participants on the basis of height and gender. We chose participants of average height (compared to the participants, not the country or world population), on the assumption that being too short or too tall will influence the behaviour – we leave the influence of height on the behaviour in crowd to future work. We alternated the gender of the walker to eliminate gender bias.

Each trial consisted of the following steps:

1. All participants move to their predefined starting positions. Recording starts.
2. The *walker* moves to the centre of the circle and rotates to face letter G, while the other participants walk around in a more or less random fashion. This ensures the crowd is different in each trial, and gives the walker ample time to covertly inspect his/her set of cards to determine the task to perform.
3. When the crowd is sufficiently randomized and evenly distributed around the walker, a verbal signal is given. The participants move to fill possible gaps in their vicinity, and then stop walking.
4. The walker manoeuvres through the crowd. Depending on the task, he/she tries to reach the goal letter, or, when the task card indicates a question mark, tries to exit the crowd in the easiest direction of his/her own choice.
5. When the task is complete, recording stops, and the next trial begins.

A total of 23 people (16 male, 7 female) participated in the experiment, with an average age of 24 years ($\sigma = 8.4$). Their average chest width was 0.44 metre ($\sigma = 0.03$), and chest thickness 0.23 metre ($\sigma = 0.03$). Seven participants took the role of walker, and a total of 47 usable trials were recorded; two recordings were rejected due to a technical issue and a participant not adhering to the task.

5.2.3 | REPRESENTATION

After the experiment, all motion capture data was post-processed, manually labelled, and mapped to an abstract agent representation for analysis purposes. Each agent consists of a line segment, defined by the ground projections of the left shoulder marker and the centroid of the two right shoulder markers. The centroid of the segment defines the position of the abstract agent. The *torso normal* of the agent is defined as the normal of the line segment facing the front of the torso. Hence, it is always perpendicular to the line segment, regardless of the direction of motion. This process is repeated for each participant at each frame of the motion capture data, and results in the set of agents translating and rotating in the ground plane used in our analysis. This representation has been shown to be suitable for human motion analysis [ALHB08a].

5.3 | Results

In this section we describe our experimental results, based on an analysis of the abstract representation of the motion capture data, as described in the previous section. At the start of each trial, the walker turns towards the target position, investigates the situation, and shifts balance in order to start manoeuvring. This initial turning towards the target position occurred naturally, and was not instructed. What follows is a period of *dense manoeuvring*, until the walker exits the crowd; the recordings from this period are the subject of our analysis.

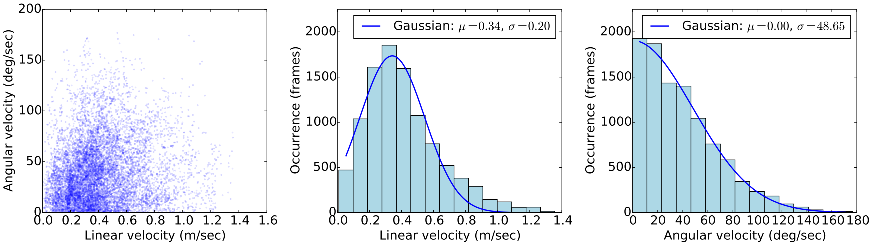


Figure 5.4: Linear and angular velocities, sampled for each motion capture frame. The scatter plot shows that there is no significant correlation between the two velocities. The histograms show the densities of the samples, separately for linear and angular velocities, and fitted Gaussian curves.

The linear and angular velocities observed in our recordings are shown in Figure 5.4. These were computed for each frame, by numerical differentiation per frame and applying a smoothing filter to improve numerical stability. The average linear velocity is 0.41 m/s ($\sigma = 0.23$ m/s), and the average angular velocity is 39 deg/s ($\sigma = 31$ deg/s), with maxima at respectively 1.36 m/s and 176 deg/s. Figure 5.4 shows a large spread of the velocities in a near-Gaussian distribution. Fitting a linear correlation results in $\mathrm{R}^2 = 0.01$; in other words, variance in the linear velocity explains only 1% of the variance in angular velocity. We can conclude that we have found no correlation between angular and linear velocities. This means that a crowd simulation method can plan linear and angular velocity independently, as we do in our Torso Crowd method (see Chapter 6).

The angle τ between the torso normal of the walker and the vector towards the target position tells us something about how often people keep their body oriented towards their target, and which angles are generally preferred. A histogram of these angles is shown in Figure 5.5. Fitting a Gaussian curve using a minimum-squared-error-approach shows an average angle $\hat{\tau} = 45^o$ ($\sigma = 35^o$). The right-hand graph in the same figure shows the percentage of time in which this angle is within a certain range. The relation between the angle limit and the time spent within that limit is more or less linear between 0 and 72 degrees, and then gradually flattens out until reaching its maximum of 100% at $\tau = 120^o$. Angle $\tau \leq 50^o$ in 50% of the time, and $\tau \leq 84^o$ in 90% of the time. These statistics are used in our Torso Crowds simulation method, in order to choose the direction in which characters rotate while navigating the crowd.

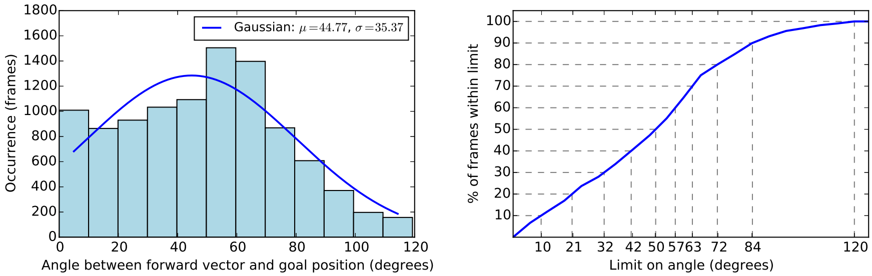


FIGURE 5.5: Angle between the agents' torso normal and to-goal vector. The right-hand graph shows the percentage of recorded frames for which that angle is within a certain limit.

For the 'question mark' tasks the letter most often chosen was E, with 4 out of 7 walkers choosing this letter. Second was the letter G with 2 out of 7. The third letter picked was C. The letter E is to the left of the walkers, letter G directly in front, and letter C directly behind the walkers. It is interesting to see that one participant took the effort to turn 180^o to find an easy way out of the crowd. We intend to use the recordings of these fastest possible crowd escape tasks in future work, to analyse crowd simulation performance for such situations.

5.3.1 | MODELLING DIRECTION OF MOVEMENT

It is a known fact that people minimize their perceived energy use when walking [Zip49, GCC$^+$10]. We predict that this holds true for dense crowds as well, and hypothesize that in such a crowd people no longer attempt avoiding single individuals, but choose an opening between pairs of individuals to pass through, and thus:

1. Move towards the *midpoint* of the opening, as this provides a path with the smallest probability of collision;
2. Move towards the *area of largest clearance* behind that midpoint when the planning agent is closer to the other individuals than the individuals are to each other.

A *midpoint* is defined as the middle of the opening between two agents. In our model, we use the middle of the line segment that represents the shortest distance between the agents' line segments. By definition, this point lies on an edge of a Generalized Voronoi Diagram (GVD) [LDI81], or on an extension of such an edge when a third agent is close by (as described in item 2 above). In such a case, the midpoint would lie on a GVD edge if the third agent were to be removed; see the right-hand image in Figure 5.6 for an example. This Voronoi edge is called the *corresponding edge* of the midpoint. The GVD is defined as a pair (V, E) of vertices V and arc-shaped edges E that partition the ground plane. Each cell of this partition is defined by an agent, and consists of all points that are closer to that agent than to any other. The centre of the *area of largest clearance* behind the aforementioned opening corresponds to a vertex of the GVD.

In our analyses we simplify the GVD by using the agents' line segments (see Section 5.2.3), rather than using their exact shape. In this section, we will use the line segments defined by the shoulder markers. In Section 5.3.2 we will investigate an alternative line segment, namely the medial axis of a torso-fitting capsule, that produces even better results.

To verify our expectations, we have analysed the difference between the actual direction of movement and the direction predicted by the expectations described above. In our analysis, we use three different methods to test our expectations, first per hypothesis, and then integrally:

MIDPOINT Only *midpoints* are considered, regardless of the distance to the planning agent. This corresponds to testing only the first hypothesis.

VERTEX Only *vertices* of the Voronoi cell defined by the planning agent are considered. This corresponds to testing only the second hypothesis, regardless of the distances.

LIMITED MIDPOINT *Midpoints* are considered, but limited to the closest point on their corresponding edges. This corresponds to both hypotheses.

Figure 5.6 shows the difference between the *midpoint* of the two green agents in dark red, and the *vertex* that indicates the start of the path between them in light orange. In both cases, the LIMITED MIDPOINT approach would use the dot that is furthest away from the cyan planning agent.

We aim to compare the directions predicted by the methods described above with the short-term direction of movement of the participant, in the order of magnitude of up to one second to capture short-term human decision making; we are not interested in the long-term overall direction, as we know it will approach the vector from the starting point to the goal position. This direction

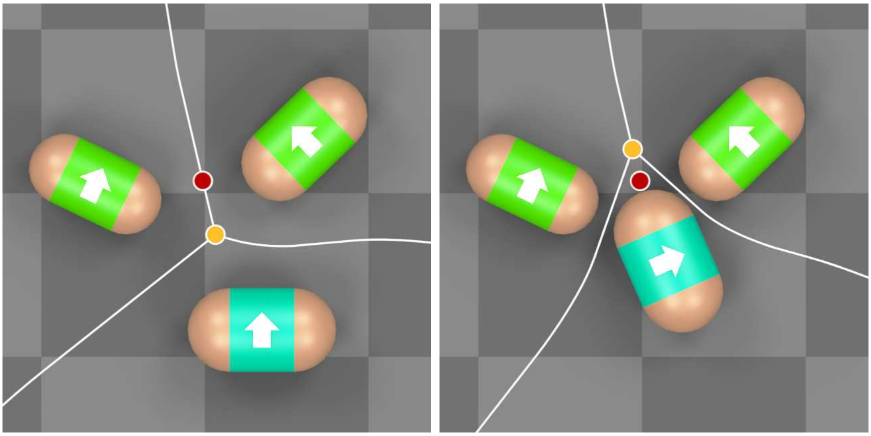


FIGURE 5.6: Two situations that show the difference between the *midpoint* (dark red) of the two green agents, and the *vertex* (light orange) that indicates the start of the path between them. The planning agent is displayed in cyan.

cannot be directly determined by the instantaneous velocity vector, since this vector varies too rapidly. Such rapid changes are especially noticeable at low velocities, where merely shifting weight can rotate the velocity vector by 180^o. To filter out these variations, we consider the recorded path of the agent, and obtain the vector from the current position to the path at a given Euclidean distance of $D \in [0.05, 1.5]$ metres (see Figure 5.7). The participant's position at this distance D is uniquely defined due to the nature of our recordings, as the participants move more or less monotonically towards their target. By choosing this distance too small, the resulting direction vector will mimic the instantaneous velocity vector and thus vary rapidly and be an ill match for the overall direction of the participant. On the other hand, choosing this distance too large will result in a direction vector that matches the long-term behaviour of the participant, instead of the short-term behaviour we are interested in. The maximum velocity we have measured during the experiment is 1.4 m/s; we are interested in short-term decisions in the order of one second, resulting in an upper bound of 1.4 metre. We have chosen our interval slightly larger than strictly necessary to ensure that the optimal distance falls within the parameters of the experiment. The search space $D \in [0.05, 1.5]$ is sampled uniformly into 10 bins; we have observed that dividing into more fine-grained bins does not produce significantly different results. Distance D is named the *plan-ahead distance*, as it could indicate the distance people consider in order to plan their direction. An example is shown in Figure 5.7. To our best knowledge, the relation between the agent's instantaneous speed and the preferred plan-ahead distance has not been studied in the context of dense crowds.

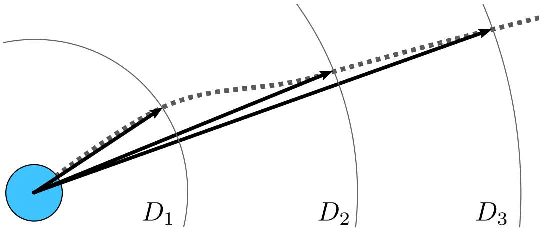


Figure 5.7: Direction vectors determined by three different plan-ahead distances D_1, D_2 and D_3. The cyan dot indicates the position of the participant, and the dotted line shows their recorded path.

As mentioned in the introduction of this section, our analysis will be limited to the time span of *dense manoeuvring*, which ends when the participant exits the dense crowd. We continually observe the participant's Voronoi cell; when the cell is incident to a Voronoi cell defined by the bounding box, we no longer consider the participant to be in the dense crowd. This definition works well in practice. An additional criterion makes this definition slightly more strict; by limiting the surface area of the Voronoi cell to 0.32 m^2 we truncate a small number of trials and ensure that all analysed data describe dense manoeuvring. An important aspect of our definition is that it does *not* consider the aggregate crowd density, but categorizes the situation of individual agents.

Now that we can compute a direction of movement and define the time span of dense manoeuvring, we can test our hypothesis. Since there is a correlation between crowd density and walking speed [Daa04], we have included the speed of the agent as a variable in our analysis. Figure 5.10 shows the graphs for the MIDPOINT, VERTEX and LIMITED MIDPOINT testing methods. For each recorded frame F, we determine the instantaneous speed of the walker $s(F)$. We partition the measured speeds into twelve bins, allowing us to average similar results from different frames. This number of bins was chosen by balancing a high enough bin count to obtain detailed data, and low enough count to ensure a sufficiently large number of data points in each bin. Speed $s(F)$ determines the bin column in which the data for that frame is plotted. We then vary plan-ahead distance D for each frame F incrementally to determine direction of movement vectors $\mathbf{d}$, an example of which is shown in Figure 5.7. This plan-ahead distance determines the row in the graph. We then compute error $e(\mathbf{d}, P)$ with respect to a prediction P, as described in the next paragraph. The error is stored at the appropriate bin; the graphs in Figure 5.10 show the average error (left) and standard deviation (middle) of the binned errors, and the number of data points in each bin (right).

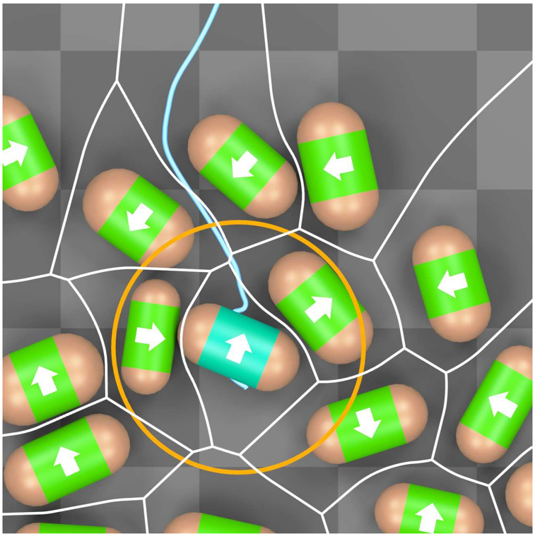


FIGURE 5.8: Recorded motion is shown as a light blue trajectory. The yellow circle indicates a plan-ahead distance of 0.4 metre.

The error $e(\mathbf{d}, P)$ depends on prediction P, which is the set of points considered by the different prediction methods MIDPOINT, VERTEX and LIMITED MIDPOINT. The error is defined as the minimal angle between the direction of movement $\mathbf{d}$ and the prediction points in P

$$e(\mathbf{d}, P) = \min_{\mathbf{p} \in P} \left[\arccos \left(\frac{\mathbf{d}}{||\mathbf{d}||} \cdot \frac{\mathbf{p} - \mathbf{x}}{||\mathbf{p} - \mathbf{x}||} \right) \right] \; ,$$

where $\mathbf{x}$ is the position of the participant.

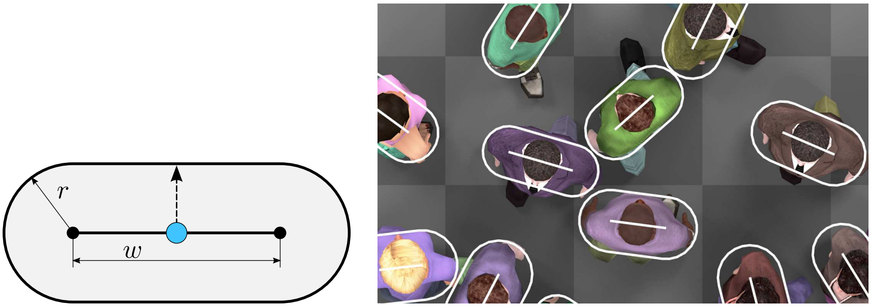
For each binned speed we find the plan-ahead distance that results in the smallest predicted error e. These bins are marked with a white dot in Figure 5.10, and indicate the most likely plan-ahead distance L for each speed v. The figure shows that, on average, $L = 0.34$, with a minute dependence on speed v; differences between the approaches are discussed in the next paragraph. As we try to predict future motions based on the static situation in the current frame F, and the participants can obviously change their trajectory dynamically at any moment in time, it is impossible to obtain a zero error. However, the low average error in these bins (≈ 7.2 degrees) shows that there is only a small difference between the direction of movement of the participants, and the directions predicted by the Generalized Voronoi Diagram of the crowd.

The white lines in Figure 5.10 indicate the best linear regressions through the bins of minimal error, weighted by the sample density. The MIDPOINT and

LIMITED MIDPOINT methods show a plan-ahead distance of approximately 0.40 metres (see Figure 5.8), whereas the VERTEX method shows a distance of 0.24 metres. This difference is caused by the VERTEX method generally predicting points closer to the participant than the other methods. From the minimal influence of the speed on the regression line we can conclude that people in dense crowds seem to generally plan for a constant distance, regardless of their speed.

5.3.2 | Capsule-shaped agent representations

In the previous section, we have shown that people manoeuvring in dense crowds tend to follow a Generalized Voronoi Diagram (GVD). We used a simple representation for each participant: a line segment spanned by the motion capture markers on the shoulders. Even though this is an accepted way to capture human motion [ALHB08a], it is based on more or less arbitrary points on the shoulders, and the distance from the line segment to the edge of the participant's torso is not constant in all directions. Hence, the GVD of the participants will differ from the GVD defined by those line segments. This difference can be reduced by slightly changing the length of the used line segment such that the distance from the edge of the torso to the line segment is more or less constant. The result is a capsule-shaped representation of the torso, as shown in Figure 5.9a. When using this representation, the inter-agent distances differ only by a constant amount from the distances between the line segments, in all directions.



(a) Schematic view of the capsule representation. The dashed arrow indicates the *torso normal* of the agent.

(b) Top view of a motion captured situation. It can clearly be seen how the capsule model fits the torso of the characters. We can show this for our character models only, as we did not have an orthographic overhead camera during the experiment.

Figure 5.9: Our abstract crowd agent representation, consisting of an oriented line segment with a radius.

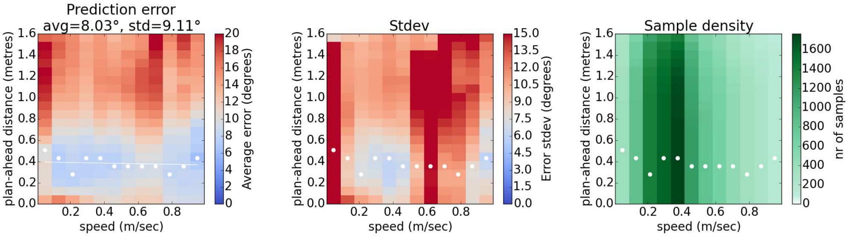
Each capsule consists of a line segment of length w inflated by a radius r, as shown in Figure 5.9a. For each participant, the measured chest thickness T determines radius $r = T/2$, and the measured chest width W determines the line segment length $w = W - 2r$. Figure 5.9b shows an overhead view of a motion captured situation, and demonstrates how well the capsule shapes fit the torsos. Of course, the legs extend from the capsule, but this is a common property of cylinder-based crowd simulations too, hence not an issue specific to the chosen representation. This approach resulted in a shortening of the line segments by an average of 10 cm ($\sigma = 3$ cm).

We have repeated the same analysis as described in Section 5.3.1, replacing the line segments based on motion capture markers by line segments based on the capsule representation. Figure 5.11 shows the graphs for the LIMITED MIDPOINT, MIDPOINT and VERTEX testing methods. As expected, the capsule-based representation results in a smaller error, hence a better model for the participants' motions. The best results are obtained with the LIMITED MIDPOINT method, resulting in an error of only 6.65^o ($\sigma = 5.33^o$); this translates to an 8% smaller error and 30% smaller standard deviation. This is corroborated by visually comparing the graphs shown in Figures 5.10 and 5.11; we can observe that this representation and method also result in the smoothest distribution of the error, and smallest standard deviation with the smoothest distribution, and hence produces the most reliable results.

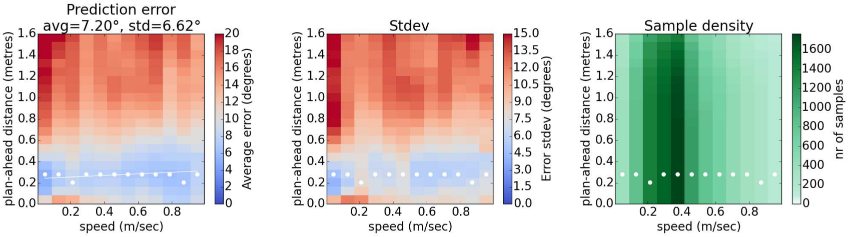
5.4 | Discussion and Conclusion

Our most prominent result is the correlation between dense crowd manoeuvring patterns and Generalized Voronoi Diagrams. We generated this Voronoi diagram by modelling the participants as line segments, and analysed the motions of the agents. Our results show that with an average error of less than 7^o ($\sigma = 5^o$) our LIMITED MIDPOINT method successfully matches the paths of our participants. In comparison, representing agents using shoulder marker line segments results in not only a higher average error, but also higher fluctuations in the standard deviation. We expect that this Voronoi-based model allows for a crowd simulation algorithm that mimics people's behaviour in dense crowds. Our capsule-based line segment representation will result in more natural, human-like behaviour in such a simulation, and is used in our crowd simulation method described in Chapter 6.

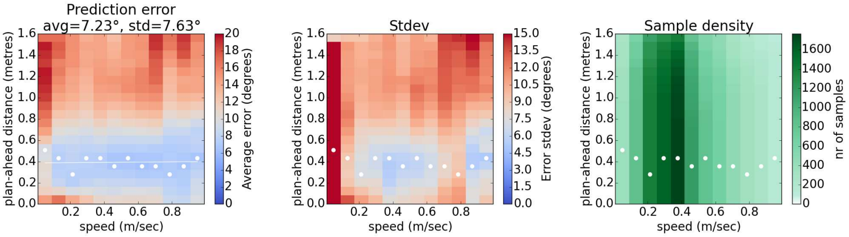
Our model shows an average error of less than 7^o, which seems small. However, given that we introduced a new method of crowd analysis, the question whether this error is good or bad remains to be answered. A perception study may give more information about usable bounds on this error, where the resulting crowd behaviour is considered humanoid.



(a) MIDPOINT method; $L = -0.03v + 0.40$.



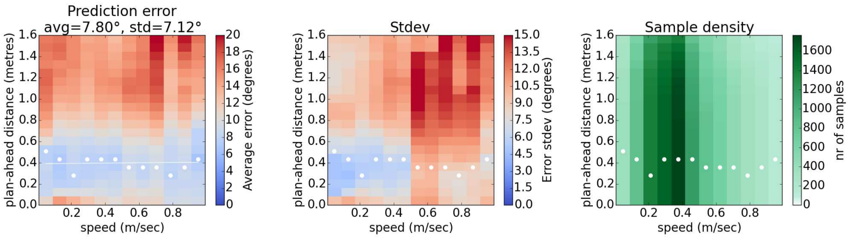
(b) VERTEX method; $L = 0.08v + 0.24$.



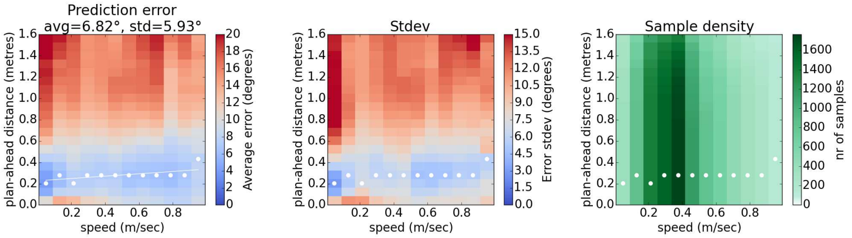
(c) LIMITED MIDPOINT method; $L = 0.01v + 0.39$.

FIGURE 5.10: (a), (b), and (c) show results of our analysis for the *motion capture marker* based approach described in Section 5.3.1. The agent's instantaneous speed is shown on the x-axis; the chosen plan-ahead distance D is shown on the y-axis. The colour indicates the average error over all recorded trials (left), standard deviation (middle) and number of data points (right). The white dots indicate the bins with the smallest error.

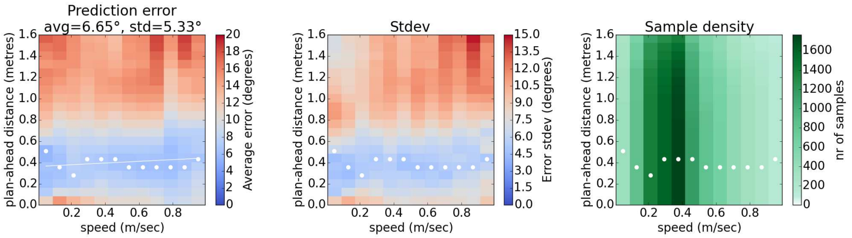
We have measured the participants in the directions that are relevant to an analysis performed on the ground projection of the data. It would be interesting to investigate the influence of the participant's relative height to the other participants on their behaviour, as this may both influence the participant as well as the others in their proximity.



(a) MIDPOINT method; $L = 0.02v + 0.39$.



(b) VERTEX method; $L = 0.11v + 0.23$.



(c) LIMITED MIDPOINT method; $L = 0.08v + 0.36$.

FIGURE 5.11: (a), (b), and (c) show results of our analysis for the *capsule* based approach described in Section 5.3.2. The agent's instantaneous speed is shown on the x-axis; the chosen plan-ahead distance D is shown on the y-axis. The colour indicates the average error over all recorded trials (left), standard deviation (middle) and number of data points (right). The white dots indicate the bins with the smallest error.

Similar to Hicheur et al. [HVR+05], who investigated the relationship between linear velocity and path curvature, we looked at the relation between linear velocity and angular velocity of the torso. Contrary to our expectations, we have found no such relation. Furthermore, we could study the relation between linear velocity and the proximity to other torsos in the forward direction. In other words, investigate whether people slow down when getting close to

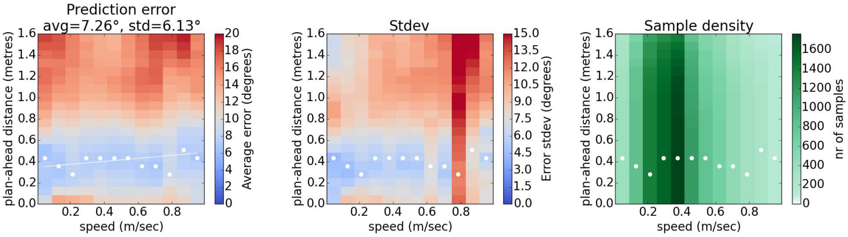


FIGURE 5.12: LIMITED MIDPOINT method, but using **points** instead of line segments to represent the agents; $L = 0.14v + 0.34$.

others, and manoeuvre slowly when squeezing between others? We have seen examples of such behaviour in our experiment, but leave a thorough analysis to future work.

As we focus on torso orientations, the orientation of the head was not recorded during our experiment. A future experiment, employing a cap with motion capture markers, could record head movement. This may produce more natural results when applying full-body animation, as well as provide interesting data on the impact of vision on manoeuvring behaviour.

During the experiment, the participants let us know that they felt uncomfortably close to each other. We suspect that this is not only caused by the higher density; after all, in daily life there are denser crowds where even manoeuvring is sheer impossible. However, in such cases there is often something that draws away the attention of the crowd, such as a performing artist, whereas in our case there was little to focus on except each other. It is interesting to investigate the effect of such a distractor on the natural density of a crowd. Finally, the experiment was performed in a controlled environment. We would like to perform similar experiments using real crowds, for example during a large concert.

CHAPTER 6

TORSO CROWDS

The shapes most often used to represent characters in crowd simulations are points and discs. In sparse crowd simulations, such a simple shape works well; the chosen representation does not have a large impact on the behaviour, as there is ample space around the agents. However, when the agents move very close to each other, motion follows shape. A common motion in dense crowds is the twisting of the torso, to squeeze through an opening between people. Points and discs are ill suited for such situations, as the rotational symmetry prohibits planning of such twist. Instead, in this chapter, we investigate an agent representation based on the torso. By employing the capsule-shaped representation introduced in Chapter 5, which is closer to the human shape (see Figures 5.9b and 6.1), we expect to obtain more realistic human-like motions than disc-based crowd simulation methods.

Contrary to other crowd simulation systems, which often focus on the movement of the entire crowd, our method distinguishes between passive agents that have no incentive to move from their present location, and active agents that try to manoeuvre through the crowd towards a goal position. We introduce the concept of a *focus point* for crowd agents, which allows for more control and more realistic, social, and complex behaviour. Furthermore, we validate the active agent behaviour using ground truth data, obtained in the experiment described in Chapter 5; our proposed model produces equivalent

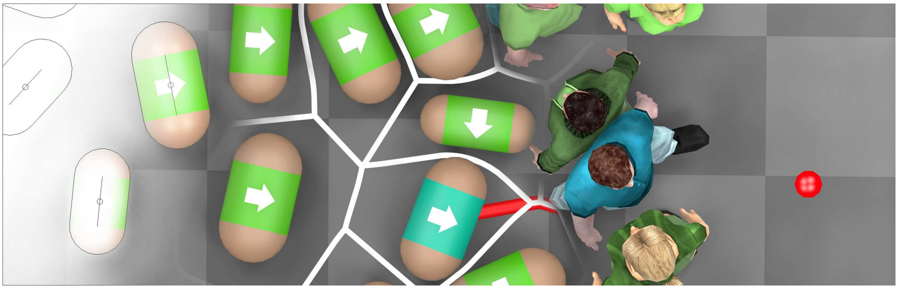


FIGURE 6.1: Our crowd simulation model, showing different stages of the process. From left to right: the agent representation, computation of the Voronoi diagram, planning a path towards a goal position, and finally the animation of torso-twisting virtual characters.

paths for 85% of the validation set. We mostly consider the motions of the upper body, i.e. the torso. The lower body is considered only at the final visualization step, where humanoid body animation is generated. We present a character animation technique that uses the results from our crowd model to generate torso-twisting and side-stepping characters. In this chapter, *torso twist* is defined as the rotation of the torso relative to the agent's trajectory.

Our implementation does not focus on computational speed. The computation of an exact Voronoi diagram comprises the majority of the execution time, and can be quickly approximated using a GPU-based technique such as described by Sud et al. [SAC$^+$08]. Instead, our focus lies on obtaining realistic results that match our ground truth data. Regardless, the simulations we included in the accompanying video are all simulated in real-time. The accompanying video is available at `https://stuvel.eu/video/torso-crowds`.

The rest of the chapter is organised as follows. Section 6.1 discusses related work. Section 6.2 provides a description of the overall problem setting, followed by the design of active (Section 6.3) and passive (Section 6.4) agents. Section 6.5 discusses obstacles, and explains how anticipation of predictable movement is modelled. We show our results, compare with a cylinder-based simulation method and with ground truth obtained from motion capture, and describe several simulated scenarios in Section 6.6. Section 6.7 describes the animation technique used to display the moving crowd agents as walking humanoid figures. Section 6.8 discusses future work, and concludes the chapter.

6.1 | RELATED WORK

As we described in Chapter 2, there are many approaches to simulating crowds. Each leads to different behaviour in dense situations, which we briefly describe here. *Flow-based* methods are, due to their macroscopic nature, particularly suitable for high-density simulations. However, such approaches model a global optimum, whereas humans generally behave less optimally and can even get stuck in very dense situations. *Cellular automaton* approaches are computationally inexpensive and thus support large crowds. They also result in even spacing between agents, which will appear unnatural when densities are high. Both flow-based and cellular automaton systems do not consider how people move, and thus generally do not result in believable human crowds. *Agent-based* methods generally avoid agent collisions at all costs, even to the point where all agents stop moving. In contrast, in actual dense crowds people frequently bump into each other. This is reflected in our method, as by design our agents prioritize motion over collision avoidance.

The common agent shapes, points and discs, have proven to be suitable to simulate abstract (i.e. non-humanoid) crowds of any density. However, when using such a simple, rotationally symmetric representation, it becomes hard to animate more detailed human motion. This results in artefacts such as interpenetration of characters, unnatural distances between characters, and a lack of torso rotations. Furthermore, we have shown that a disc does not accurately represent the actual volume occupied by the character in 3D space (see Chapter 3). In Section 6.6, we show the importance of the agent shape in dense crowds, not just for realism of the motions, but also to support higher densities without getting stuck. Singh et al. use multiple discs [SKRF11] to represent an agent, and plan their motion using a footstep model. This approach allows for denser crowds than body-enclosing discs, and offers realistic walking animations. However, the ability to simulate dense crowds remains to be investigated. Our agent-based method extends the walk cycle approach, by employing multiple walk cycle animations and, obviously, twisting the torso. To our knowledge, our method is the first to use a capsule as agent representation in crowd simulations.

In Section 3.4, we investigated the residual distance between the meshes when a collision is reported, and compared the volumes occupied by the bounding cylinder and by the BCH. To further validate the choice for the capsule as collision detection shape, we have performed the same two analyses using the capsule, extruding it over the character's height. Since collision detection using the capsule approach allows for false negatives, whereas the other collision detection strategies discussed in that chapter do not, the results are not directly comparable: any false negative, where the character meshes intersect when the capsule shapes do not, are reported as a zero residual distance. The average residual distance obtained in that way is 2.3 cm (σ=3.5 cm), which is closer than the single cylinder (20 cm) or BCH (5 cm). The volume represented by the extruded capsule is closer to the actual mesh volume than the single cylinder and BCH representations. For respectively the female and male characters, the extruded capsule volumes were 3.21 and 2.45 times larger than the character mesh volume. See Figure 6.2 for a comparison with the cylinder and the BCH. As the capsule is pose-independent, the standard deviations are zero; the minute variations in character mesh volume due to the linear blend skinning are ignored.

Our method employs Voronoi diagrams for planning motions through the crowd. The edges of such a diagram represent the paths of maximum clearance between agents; intuitively this corresponds well with the desire of people to minimize perceived effort when walking [Zip49, GCC^{+}10]. In Chapter 5, we showed that in a dense crowd people indeed move along such paths. The Explicit Corridor Map method by Geraerts [Ger10] uses city-scale generalized Voronoi diagrams for path planning. Sud et al. [SAC^{+}08] perform path planning based on 1$^{\text{st}}$ and 2$^{\text{nd}}$ order Voronoi diagrams, containing information about respectively the closest agent and the closest pair of agents. They employ a path scoring technique slightly resembling our proposed method.

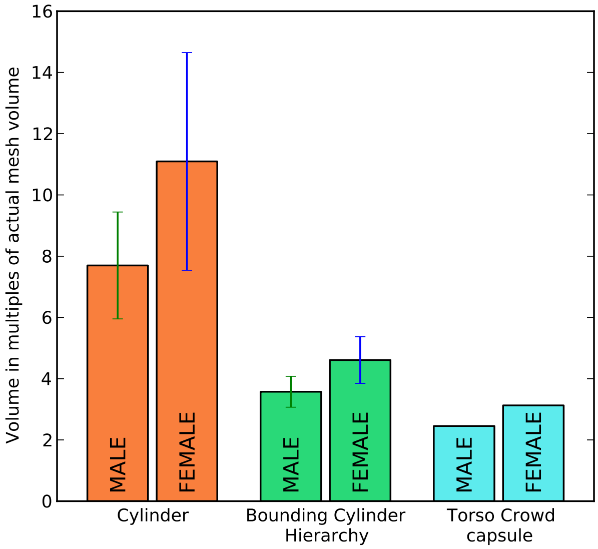


FIGURE 6.2: Volumes occupied by the minimal bounding cylinder, the BCH, and the extruded Torso Crowd capsule. The volumes are expressed as multiples of the actual mesh volume, to allow comparison between different meshes.

While their article promotes speed of computation, our approach focuses on a richer character representation, and validation against real crowd data.

6.2 | SETTING AND PROBLEM FORMULATION

Our crowd simulation system considers the torso as the main moving element. The algorithm is based on our findings described in Chapter 5, where we observed and recorded dense crowd behaviour. In that experiment, participants were given the task of manoeuvring through the crowd to predefined points. The movement of the crowd was recorded using a motion capture system, and these data serve as a ground truth for the behaviour of people actively manoeuvring through dense crowds. Our Torso Crowd model is designed to support the observed motions of the crowd-escaping participants, and believable simulation of essentially stationary people.

When observing dense crowds in general, and the previously mentioned recordings specifically, it is clear that torso rotations are critical when manoeuvring through a dense crowd. To support such rotations, our agents extend the common disc-based crowd agents, as shown in Figure 6.3. The common agent is defined as a point with a radius r'; our agent model extends this point to a line segment of length ℓ, with a (probably different) radius r, resulting in a *capsule* shape, which is also known as a *race track*. This extension eliminates the rotational symmetry, thereby making it possible to plan torso rotations. Note that the capsule forms a generalization of the circular crowd agent, as such a form may be obtained by taking $\ell = 0$ and increasing the radius by $\ell/2$ to compensate for the reduction in width.

People standing still in a crowd behave differently from people trying to reach a certain goal position. In order to model these differences in behaviour, two types of agents are used in our crowd simulation technique. *Active agents* move to reach their goal position, whereas *passive agents* mostly stay in place, moving only to make room for other agents. Section 6.3 describes the behaviour of the active agents, while Section 6.4 describes the passive agents. Different aspects of the *environment*, such as walls, doors, and other obstacles, are handled by our method in a unified way; this is described in Section 6.5. The goals of the agents are determined by a high-level planner, which is scenario-dependent and not described further. When an active agent reaches its goal, depending on the intended scenario, the agent optionally switches to passive behaviour. Similarly, when the high-level planner provides a passive agent with a new goal position, that agent will switch to active behaviour.

The crowd consists of N agents A_i, $i \in [1, \ldots, N]$. Each agent A_i in reference placement is defined as the Minkowski sum of a line segment of length ℓ_i centred around the origin and aligned with the x-axis, referred to as the *central axis*, and a disc of radius r_i. The placement of an agent is represented by a pair $(\mathbf{a}_i, \theta_i)$, where $\mathbf{a}_i$ and θ_i are the agent's position and orientation. For ease of discussion, we denote the direction of the forward-facing normal of the torso of agent A_i in placement $(\mathbf{a}_i, \theta_i)$ by $\mathbf{n}_i$, the continuous set of points covered by its central axis by $\mathbf{s}_i$, and its linear velocity vector as $\dot{\mathbf{a}}_i$. These concepts will be detailed in the following sections.

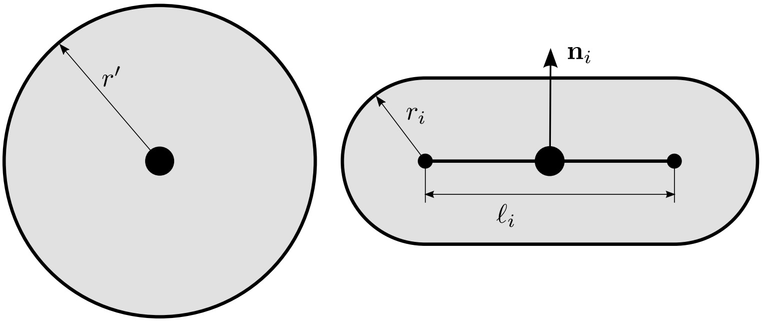


Figure 6.3: Two types of crowd agent representation. On the left a common crowd agent: a point with a radius. On the right our crowd agent: a line segment with a radius.

6.3 | ACTIVE AGENTS

From observation of the previously mentioned ground truth data, and dense crowds in general, we formulate the following assumptions as basis for our active agent model.

- People tend to choose a comfortable path, that is, maximize clearance, by avoiding areas of very high density. Occasionally, a less comfortable path may be chosen, when the discomfort is only temporary and the path leads to an area of larger clearance.
- People tend to minimize perceived energy use [Zip49], and thus prefer short, straight paths.
- People generally move in the direction of their goal, but divert from the shortest path when it is obstructed or when an alternative path is significantly more comfortable.
- Averaging at 0.4 m/sec, the traversing speed through a dense crowd is relatively low (see Section 5.3). This, combined with the dynamic nature of crowds and possibly a lack of overview of the situation, makes long-distance planning of exact paths to the goal impractical.

The generalized Voronoi diagram (GVD) is a partitioning of the plane. In our crowd model, a cell is defined for each agent, being the set of all points that is closest to that agent. The GVD is represented as a pair $\{V, E\}$ of vertices V and edges $E \subset V \times V$ that represent the boundaries of those cells, where the edges are arcs (possibly with zero curvature). Every point on an edge or a vertex is equidistant to its neighbouring crowd agents. These edges form the medial axis between the agents, and thus represent a more or less comfortable path that maximizes local clearance. In Chapter 5, we observed that people in dense crowds indeed follow such paths (also see Figure 6.4).

In real situations, observing surrounding people, planning a path, and manoeuvring through the crowd, are intertwined in a continuous process. However, people are not continually reconsidering all their options all the time, but rather make more or less discrete decisions. Our agents reflect this behaviour by replanning their actions at a moderate rate.

Similar to Sud et al. [SAC$^+$08], all active agents use the same GVD for planning their motion. The passive agents use a slightly altered GVD, modelling all doors as closed as described in Section 6.4. As a result, the GVD needs to be computed at most twice per simulation update, regardless of the crowd size and frequency of planning. For simplicity of computation, we use the central axes $\mathbf{s}_i$ to compute the GVD, rather than the agent shapes themselves. Due

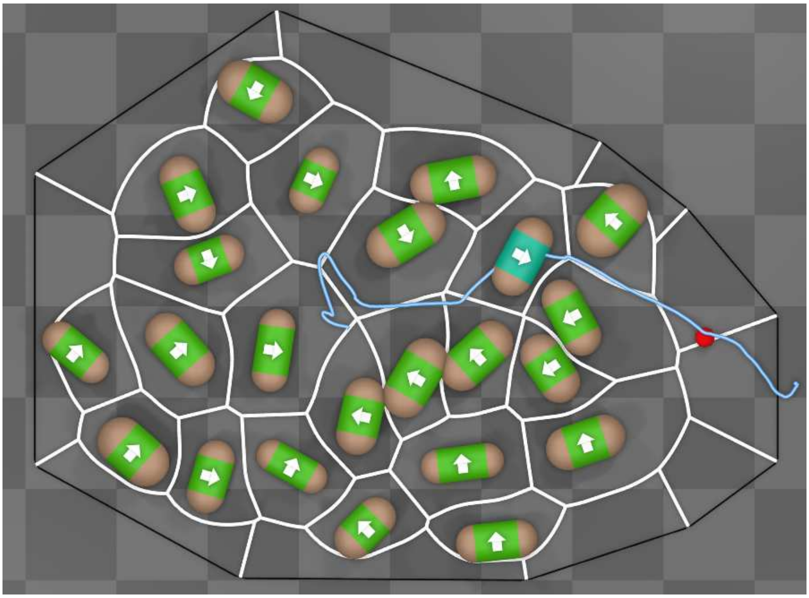


FIGURE 6.4: Top-down view of a real, motion captured crowd, with the generalized Voronoi diagram in white lines.

to the nature of the capsule, where the distance from the axis to the edge of the shape is constant, this approximated GVD is very similar to the exact GVD (see Figure 6.5); the distances between the corresponding edges are in the order of magnitude of the differences between the agent radii. Such small differences, in our case in the order of centimetres [SdGvdSE15], are unlikely to cause noticeable changes in the crowd's behaviour.

Our Torso Crowd method is suitable for simulating dense crowd manoeuvring, and based on experimental observations of such behaviour. However, in situations where the crowd is not dense, we have no proof of validity. As a consequence, our implementation switches to the less complex and thus faster RVO2 crowd simulation algorithm [vdBGLM11] when agents move out of the dense crowd. This can be seen in the accompanying video, when agents walk out of a lift and into an empty hallway. We reuse the definition of *dense* situations from Section 5.3.1, namely those situations where there is an average of at least three humans per square metre, as measured by the area of their Voronoi cell. Since we can measure this density on a per-agent basis, this decision is also made for each agent individually.

In the next subsections, we discuss the planning and execution of the agent's movement. Firstly, similar to real people, a desired position is planned, taking into account potential torso twists needed to reach that position. Since

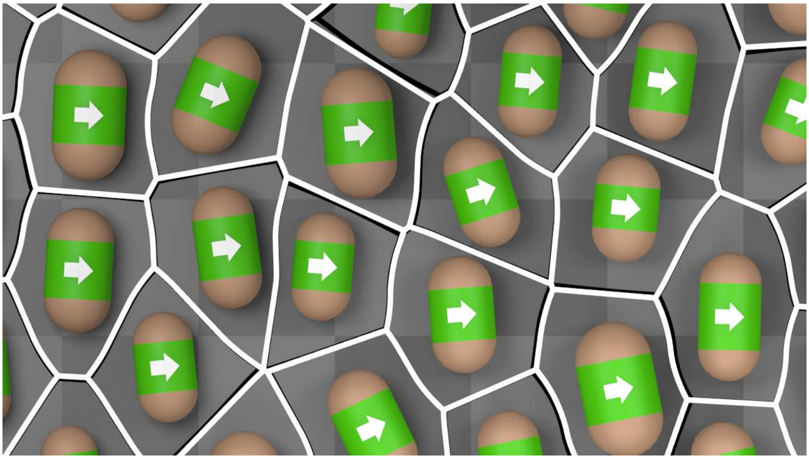


Figure 6.5: The exact Generalized Voronoi Diagram of the capsules (black) and the approximated diagram of their central axes (white). The distance between corresponding edges depend on the difference in capsule radius.

the available clearance at the planned position poses a bound on the torso orientation, this orientation is planned in a second step.

6.3.1 | Limited-horizon path planning

To plan the movement of an active agent, the following steps are taken:

1. Find paths by exploring the vicinity in the GVD of the Voronoi cell containing the agent.
2. Compute a score for each path, and determine the best-scoring path.
3. Compute the desired agent orientation at the start of the path, accounting for available clearance.

The GVD provides proximity information in a natural way; the cell of the active agent represents its proximity, and the outgoing edges of that cell's vertices form paths between agents in its direct vicinity. The search is initialized by taking these outgoing edges, i.e. the edges that only have a single vertex incident to the active agent's Voronoi cell. This set is then extended, parametrized by given values for Euclidean distances H_D and H_ϵ, and edge count limit H_C, as follows: the outgoing edges are followed depth-first until either distance H_D, or edge count limit H_C is reached. For the latter limit, edges shorter than H_ϵ are ignored (as circled in Figure 6.6). Such short edges

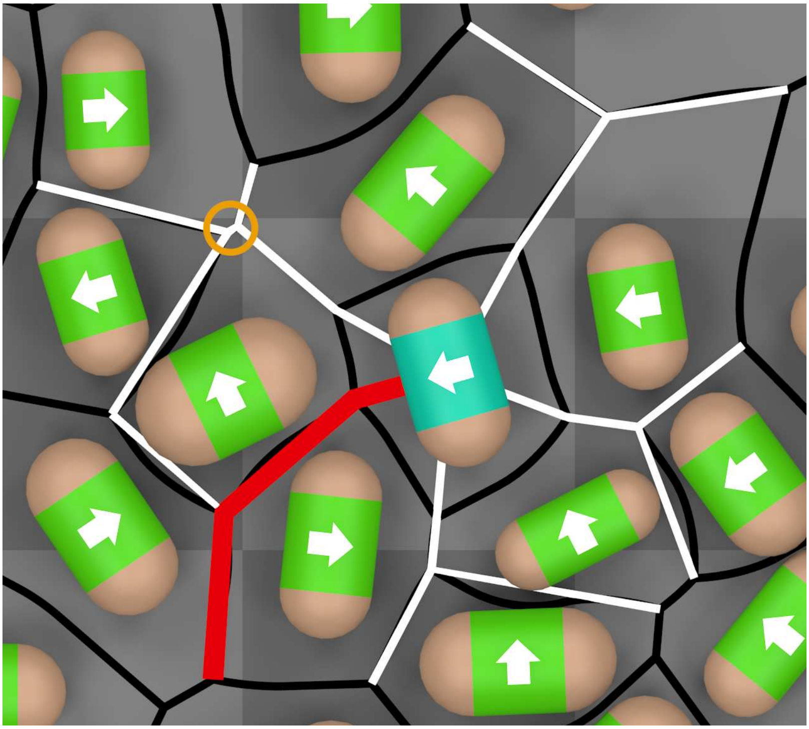


FIGURE 6.6: Candidate paths in white lines, with the best-scoring path as a thick, red line. The circled edge was shorter than H_ϵ. The paths are extended to the agent position. The goal position is bottom left outside the frame. Note that the paths along curved Voronoi edges are just drawn as straight line segments for simplicity.

occur, for example, when four agents are almost equidistant to a point, and the clearance between the agents would likely be perceived as a single space. Hence, such edges are unlikely to correspond to human perception. Even though this approach could theoretically lead to a path consisting of an arbitrarily large number of edges, such a situation does not occur in dense crowds when using crowd agents of more or less realistic human-like sizes. The resulting path P consists of a sequence of GVD edges; following the path should bring the agent closer to its goal.

After a set of candidate paths is found, each path is given a score. The agent will attempt to use the path with the highest score. The composite score function $S(i, P)$ takes agent A_i and path P. It enforces the behaviour of real people in dense crowds, based on our observations in Chapter 5. As all score functions should be balanced to make a final decision as to the best possible path, they are combined into a weighted sum:

$$S(i,P) = w_g S_g(i,P) + w_c S_c(P) + w_l S_l(P) + w_m S_m(i,P) \ ,$$

where $S_g(i,P)$, $S_c(P)$, $S_l(P)$ and $S_m(i,P)$ are score functions, and w_g, w_c, w_l and w_m are weights given to these sub-scores. Values for these weights are determined in Section 6.6.1. In the following descriptions of each score function, $\mathbf{p}_0$ and $\mathbf{p}_f$ respectively indicate the initial and final vertex positions of path P. Note that the path's final point $\mathbf{p}_f$ does not necessarily correspond to the agent's goal position $\mathbf{g}_i$, due to the limit on the path length described earlier.

Score function $S_g(i,P)$ drives the agent towards its goal. It measures how well the path leads to the goal position $\mathbf{g}_i$, expressing the distance, from the end of the path to the goal, as a ratio of the total Euclidean distance to the goal. This normalization ensures that the resulting score is independent of the absolute distance to the goal:

$$S_g(i,P) = 1 - \frac{|\mathbf{g}_i - \mathbf{p}_f|}{|\mathbf{g}_i - \mathbf{a}_i|} \ .$$

Score function $S_c(P)$ measures the clearance radius along the path, ensuring that the agent prefers comfortable routes with large clearances. It consists of two components. The first component stems from the moderate rate replanning principle. It assumes that a person plans a motion towards a more spacious area; when this area is reached, a new decision can be made. The second component prefers motion along paths with as much clearance as possible. The GVD structure enables efficient computation of clearance radius $C(\mathbf{x})$ at any point $\mathbf{x} \in \mathbb{R}^2$.

$$S_c(P) = w_c^F C(\mathbf{p}_f) + w_c^A \frac{1}{|P|} \int_{\mathbf{x} \in P} C(\mathbf{x}) \ d\mathbf{x} \ ,$$

where $|P|$ indicates the total arc length of P, $\mathbf{x} \in P$ are the collection of all points along path P, and w_c^F and w_c^A are weights for respectively the final and the average clearance of the path. Due to implementation details of our GVD library, we only had access to the minimal clearance of edges, and approximated the integral using discretized summation. This score function also serves as a term to minimize the relative rotation of the torso with respect to the motion trajectory, due to the way the clearance information is used to plan torso orientations (as described in Section 6.3.2).

The conservation of energy can be broken down into two components: the minimization of the distance travelled, and the effort required to travel that distance. Score function $S_l(P)$ models the first component, and measures path length. This function combines with $S_g(i,P)$ into the preference of short paths leading to the goal:

$$S_l(P) = -\sum_{e \in P} |e| \,.$$

Score function $S_m(i, P)$ represents the second component of energy conservation, by penalizing changes in momentum, i.e. sharp turns. Since we can safely assume that the mass of the agent is constant, any change in momentum is explained by a change (in the direction of) the velocity vector, which in turn can be modelled by the cosine similarity of the current velocity and the direction towards the path:

$$S_m(i, P) = \frac{\dot{\mathbf{a}}_i \cdot \mathbf{e}_0}{|\dot{\mathbf{a}}_i||\mathbf{e}_0|} \,,$$

where $\mathbf{e}_0 = \mathbf{p}_0 - \mathbf{a}_i$, the vector connecting the agent to the starting point of the path.

A more elaborate alternative for $S_m(i, P)$ could compute the weighted integral of the curvature along $\mathbf{e}_0$ and P, with the weight inversely proportional to the distance from the agent. This would take the curvature of the entire path into account, emphasizing more immediate momentum changes. However, our proposed approach is simpler, and seems to be sufficient in practice. Furthermore, due to the agent replanning while it is en route to its goal, effectively the curvature of the entire path is taken into account.

6.3.2 | Torso rotation planning

Once the best path P has been chosen, which determines the next torso position, the torso *orientation* T_o is determined. For this we use the torso normal $\mathbf{n_i}$ of agent A_i. Torso orientation T_o consists of two components: heading T_h and torso twist T_t. The first component, T_h, represents a common nonholonomic walking motion along the start of the path. Its computation is trivial and not described here. The second component, torso twist T_t, adjusts for the minimal clearance along the start of the path. The clearance at later parts of the path is of less importance for the current torso twist planning, due to the moderate-rate replanning principle. The first edge of the path lies between two neighbouring agents, and ends at a point of local maximum clearance[1] behind those two agents. It is this part of the path that is used for the planning of the torso twist. The clearance at a point indicates the distance from that point to the nearest agent capsule. To maximize the time for the agent to smoothly change its torso orientation towards the desired twist, we compute the minimal clearance c along the first edge of path P. The torso

---

[1] This statement is not strictly true for Voronoi diagrams in general. However, it is valid for our dense crowds in constrained environments

twist T_t can then be expressed in degrees as:

$$T_t = \begin{cases} 0^o & : c - r_i \geq w_i \\ \arccos\left(\frac{c-r_i}{w_i}\right) & : 0 < c - r_i < w_i \\ 90^o & : c - r_i \leq 0 \end{cases}$$

where $w_i = \ell_i/2 + r_i$ is the half-width of the capsule. This results in two possible orientations, both of which will fit the available clearance equally well: $T_h + T_t$ and $T_h - T_t$. In Section 5.3, we observe that, while manoeuvring through a dense crowd, people tend to aim their torso normal towards their goal position. The absolute angle between the torso normal and the vector to their goal is limited to 90^o for 90% of the time, and never more than 120^o. Consequently, we choose $T_o = T_h \pm T_t$ such that this angle is minimized.

6.3.3 | PERFORMING THE PLANNED MOVEMENT

Like most crowd simulation systems, our system is not able to avoid every collision, and since collisions frequently occur in real dense crowds, this is actually desirable. As the likelihood of collisions increases with density, it is critical to detect and handle collisions in a correct way. For this, we use a physics engine, as such engines are optimized for efficient collision detection and handling. Since instantaneous displacement of objects can interfere with realistic collision handling in such an engine, our agents are moved using virtual forces. Once the desired position $\mathbf{p}_0$ and orientation T_o have been computed, each agent employs proportional-derivative controllers to steer towards the planned configuration. For our implementation we use the Blender Game Engine [ble15], which contains a physics simulation engine based on Bullet [bul15].

So far we have discussed the general approach for an active agent. Based on observations from real crowds, we deviate from this approach when a character starts to move towards a goal. In Section 5.3, we observed that, before they start manoeuvring, people orient their torso towards their goal. Similar behaviour is incorporated into our crowd model. When an agent becomes active and starts planning its movements, it performs the same planning steps as described in the previous subsections. However, it discards the planned position $\mathbf{p}_0$, and rotates on the spot towards the planned orientation T_o. Subsequent planning steps are performed as described earlier.

The agents' maximum translation speed is enforced by a parameter of the physics engine, which is controlled by an agent-dependent function $m(i)$ that interpolates between a minimum and maximum speed based on the available clearance at the planned location. In our implementation, we use the following function:

$$m(i) = \begin{cases} (M_x - M_n)\left(\frac{c}{r_i}\right)^{\gamma} + M_n & \text{if } c < r_i \\ M_x & \text{otherwise} \end{cases}$$

where r_i is the radius of agent A_i, c is the available clearance at the agent's planned location $\mathbf{p}_0$, M_n and M_x are minimum and maximum bounds on the speed, and γ is a curvature tuning parameter. In our implementation, we obtained polite behaviour using $M_n = 0.05$ m/sec, $M_x = 0.6$ m/sec and $\gamma = 2$. By choosing $\gamma > 1$ and M_n fairly low, the agent slows down when the crowd density is high, giving the other agents ample time to make space before moving forward. A more aggressive agent can be modelled by lowering γ and increasing M_n.

Before planning a new path, an agent performs a ray-cast on the GVD. When a direct, linear path from the agent to the goal is available with a minimal clearance radius of the agent's half-width $\ell_i/2 + r_i$, the agent foregoes the planning stage, moves directly to the goal position, and stops there. This approach is suitable when the goal can actually be easily reached by the agent. Alternatively, for example when the goal position is occupied by another agent, the agent detects that the goal is within a threshold distance, and stops manoeuvring.

6.4 | PASSIVE AGENTS

In this section, we discuss the behaviour of *passive* crowd agents, which, in contrast to active agents, do not have an explicit target to navigate to. We consider two sometimes contradictory motivations for their placement: finding local comfort, and rotation towards a focus point. Our passive agents locally optimize their placement, making themselves as comfortable as possible, i.e. maximize the clearance around them, given the constraints of their immediate surroundings. Similar to the planning method for active agents, we separate the planning of translation and rotation. The translation $\mathbf{t}_S$ to reach a more comfortable placement is described in Section 6.4.1. When the geometry of the environment and the configuration of the crowd allow for it, a trade-off is made between rotating to a comfortable orientation and a rotation towards a focus point. This can be the centre of a chatting group of people, the charismatic front man of a performing band, or simply the floor number display of the lift. To our knowledge, we are the first to use such a focus point in a crowd simulation system. The rotation ϕ_S from the current to the desired orientation is described in Section 6.4.2. Passive members of a crowd temporarily accept a less comfortable position in order to make way for someone else to pass; this avoidance by translating ($\mathbf{t}_A$) and rotating (ϕ_A) is described in Section 6.4.3.

In Section 6.4.4 we show how these desires are combined into the agent's motion.

6.4.1 | SPACE FINDING

Passive agents try to loosely maintain their position. For example, even when a lift is crowded, the door is open, and outside the lift is a plethora of space, agents waiting in the lift will remain in that lift. Manoeuvring to a different area, such as stepping out of the lift, is considered active behaviour, and is described in the previous section; note that agents can switch from passive to active behaviour when required. We use walls and doors (see Section 6.5) to delineate areas in the environment. To restrict the space finding algorithm to the agents' current area, our passive agents consider all doors as closed, regardless of their actual state. However, the agents do search for a better place to stand in their direct vicinity; this is what we call *space finding* behaviour. This results in a translation vector $\mathbf{t}_S$ from their current position to a more spacious position. Effectively it is a combination of comfort optimization and avoidance of passive agents.

Whether the space finding algorithm is engaged depends on the space around the agents. We assume that our passive agents like to stand in a spot where there is enough space surrounding them. When that is the case, i.e. the distance to the nearest neighbouring agent or obstacle is larger than a certain threshold, they remain stationary, even though there may be even more space available to them; the agent is marked as *happy* with its current placement, and will not engage the space finding algorithm (so $\mathbf{t}_S = \mathbf{0}$). This threshold can be configured individually for each agent, and can be a function of culture, scenario, or the geometry of the surroundings.

In tighter situations, our passive agents move to maximize their comfort. To obtain nearby candidate positions of increased comfort, agents consider points of maximum clearance between their surrounding neighbours. By definition, such points correspond with vertices of a Generalized Voronoi Diagram (GVD, described in Section 6.3) of those neighbours. Such a *local Generalized Voronoi Diagram* $\mathcal{L}_i$ of agent A_i is the GVD defined by $\mathcal{N}_i$, where $\mathcal{N}_i$ is the set of neighbouring agents and obstacles of agent A_i. $\mathcal{N}_i$ can be efficiently extracted from the GVD of the entire crowd, by iterating over the edges of the cell containing A_i, and taking the agents or obstacles on the opposite side of the edges. Note that agent A_i itself is *not* included in $\mathcal{L}_i$ (as shown in Figure 6.7). The vertices of $\mathcal{L}_i$ correspond to local clearance maxima, and thus potentially comfortable positions for the agent to move to.

People try not to spend too much energy [Zip49], and will accept a marginally more cramped situation when walking to a better spot would take a significant

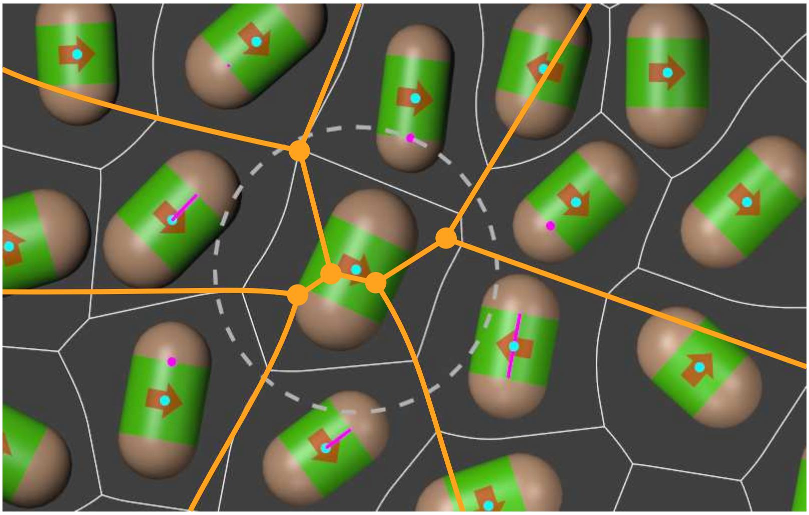


Figure 6.7: Example of a local Generalized Voronoi Diagram (GVD), with points of maximal local clearance, in orange. The GVD of the crowd is shown in white. The features that define the local GVD are shown in magenta. The dashed circle shows the clearance of the agent.

effort. To take this into account, we use the following energy minimization function to balance the gain (more available space) with the expended energy (the distance to travel to that space). All vertices $\mathbf{v}_j \in \mathcal{L}_i$ are considered potential better positions, and are given an energy cost

$$\begin{aligned} E(\mathbf{a}_i, \mathbf{v}_j) &= \frac{|\mathbf{v}_j - \mathbf{a}_i|}{C(\mathbf{v}_j) - C(\mathbf{a}_i)} \\ \mathbf{v}_d &= \arg\min_{\mathbf{v}_j \in \mathcal{L}_i} E(\mathbf{a}_i, \mathbf{v}_j) \\ \mathbf{t}_S &= \mathbf{v}_d - \mathbf{a}_i \end{aligned}$$

where $C(\mathbf{x})$ indicates the clearance around $\mathbf{x}$; $\mathbf{v}_d$ denotes the vertex with the lowest energy cost, and determines the agent's space finding translation vector $\mathbf{t}_S$. This scoring is efficient; we have found that, in practice, 89% of the time $\mathcal{L}_i$ contains no more than three vertices, with an average of 2.7 vertices (σ=0.7). The energy function $E(\mathbf{a}_i, \mathbf{v}_j)$ is intentionally left simple. Although it is not based on actual knowledge on the human decision making process, it works well in practice. We leave it to future work to discover whether the preference of position is indeed linear in clearance and distance.

6.4.2 | ORIENTATION FINDING

When agents are squeezed into a small area, they rotate themselves to fit the available space. However, if the constraints allow for it, the agents focus on a given point (a performing band on a stage, floor number display of a lift, etc.). This results in a rotation ϕ_S from the current orientation of the agent towards a desired orientation. The *focus point* is environment- and scenario-dependent, and can of course change over time and be different for each person or agent. It is denoted as $\mathbf{f}_i$ for agent A_i. The accompanying video shows the effect of this focus point. A group of agents have a focus point in the centre, and the video demonstrates the effect of increased density on this group: the group stays together, even though the focus point has no direct influence on the position of the agents (see Figure 6.12).

In the remainder of this section, p is the passive agent's index number, so $\mathbf{a}_p$ indicates its position. The angle between the agent's torso normal $\mathbf{n}_p$ (see Section 6.2) and the vector to its focus point $\mathbf{f}_p$ is defined as

$$\alpha_f = \angle(\mathbf{f}_p - \mathbf{a}_p, \mathbf{n}_p)\,,$$

where $\angle(\mathbf{x}, \mathbf{y})$ indicates the signed angle between two vectors on the interval $(-\pi, \pi]$.

When marked as *happy* with their current placement (which depends on the available clearance, as described in Section 6.4.1), our crowd agents rotate such that their torso normal points towards their focus point. In this case, we take $\phi_S = \alpha_f$.

The shape of the available space is the dominant factor in someone's orientation when that space is tight; one rotates to fit the little space available. The narrower the space, the less important any focus point becomes. To include this behaviour in our model, we inspect the shape of the agent's Voronoi cell. Since this cell contains all points that are closer to the agent than to any other agent, it is a good model for their available space. The *width* of the cell is defined as the minimal distance between two parallel tangents to the cell. The direction of these tangents are a common measure for the oblong direction of the cell. However, this direction is not stable under small variations in agent configurations, so we use a more robust approach. To obtain a vector that indicates the overall orientation of the space, a Principal Component Analysis (PCA) [Hyv70] is applied. Such an analysis is applied to a point cloud, to determine its dominant direction. Since it cannot be applied to continuous shapes, intuitively we could sample the interior of the Voronoi cell to obtain such a point cloud. However, to increase computational performance, we limit this approach to the sampled cell edges; considering the results, this is sufficient. The result of the PCA consists of the eigenvalues and eigenvectors of a covariance matrix; when ordered from large to small by their absolute

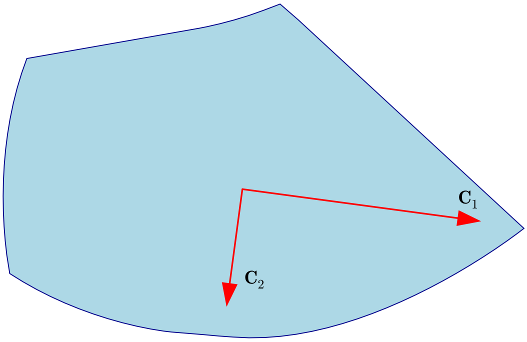


Figure 6.8: Example of a Voronoi cell, with the first and second principal component in red, and scaled by their eigenvalues.

eigenvalues c_1 and c_2, the eigenvectors indicate the first and second principal components $\mathbf{C}_1$ and $\mathbf{C}_2$. In the remainder of this section, we assume that the absolute eigenvalues are ordered by magnitude, i.e. c_1 is the absolute eigenvalue belonging to eigenvector $\mathbf{C}_1$.

When the Voronoi cell of a passive agent has no clear orientation, the eigenvectors hold little information, and the eigenvalues will be more or less equal. In this case, the agent rotates towards the focus point. When the shape of the cell is elongated, and thus relevant for the orientation of the crowd agent, the first principal component aligns with the cell's shape (see Figure 6.8). This relevance is indicated by a large difference between the first and second eigenvalue of the covariance matrix, i.e. $c_1 - c_2 \geq \epsilon_2$ (ϵ_1 will be introduced later as a lower bound). In this case, there are two possible orientations for the agent, in which the agent's central axis $\mathbf{s}_p$ aligns with either $\mathbf{C}_1$ or $-\mathbf{C}_1$; the agent chooses the orientation that minimizes α_f. If there is no focus point, α_f is not defined, and the agent chooses the orientation that requires the smallest rotation from its current orientation:

$$\alpha_c = \angle(\pm\mathbf{C}_1, \mathbf{s}_p) \ .$$

To ensure smooth transition between α_c and α_f, we blend between them depending on the eigenvalue difference:

$$\phi_S = \begin{cases} \alpha_c & \text{if } \ \epsilon_2 \leq c_1 - c_2 \\ I(\alpha_c, \alpha_f, t) & \text{if } \ \epsilon_1 \leq c_1 - c_2 < \epsilon_2 \\ \alpha_f & \text{if } \ c_1 - c_2 < \epsilon_1 \ , \end{cases}$$

where $\epsilon_1 < \epsilon_2$, $I(\alpha_c, \alpha_f, t)$ indicates angular linear interpolation along the shortest arc for $t = (c_1 - \epsilon_1)/(\epsilon_1 - \epsilon_2)$. In our implementation, we use $\epsilon_1 = 0.015$ and $\epsilon_2 = 0.045$.

We investigated simpler approaches for finding the orientation of the shape, such as a PCA on just the Voronoi vertices, or taking the two points furthest apart on the Voronoi edges. However, our approach of performing a PCA on the Voronoi edges produced the most stable and realistic results.

6.4.3 | AVOIDANCE OF ACTIVE AGENTS

The behaviour of passive and active agents is quite different. Passive agents move slower, and try to divide the available space between them. Active agents move faster (when allowed by the constrained environment), and, more importantly, try to reach a specific goal. These differences are also reflected in the way that passive agents perform agent avoidance. This section describes how they avoid *active* agents; avoidance of passive agents is handled by the space-finding algorithm, which is described in Section 6.4.1.

Since distant agents have a negligible probability of colliding with the passive agent, only those nearby are avoided. Of the active agents that are within an avoidance distance d_i of the passive agent, measuring distance between the agents' capsules, the nearest K, with indices $\{i_1, \ldots, i_K\}$, are considered for avoidance. In our implementation we used $d_i = 0.4r_i$ and $K = 4$. The avoidance distance d_i can be varied to model observant (using larger d_i) or unaware (using smaller d_i) behaviour, and is not necessarily related to the agent's radius. In the following description of the avoidance behaviour, we denote the index of the active agent that is to be avoided as $i \in \{i_1, \ldots, i_K\}$, and the index of the passive agent as p. Agents that move away from the avoiding agent, i.e. where $(\mathbf{a}_i - \mathbf{a}_p) \cdot \mathring{\mathbf{a}}_i > 0$, are safely ignored, as their motion is sufficient to avoid any collisions.

The avoidance behaviour consists of two components, a rotation ϕ_A and a translation $\mathbf{t}_A$. The passive agent rotates to minimize its width in the active agent's direction of movement, and it translates to move out of the way. The active agent's position $\mathbf{a}_i$ and velocity vector $\mathring{\mathbf{a}}_i$ are used to determine a first-order approximation of its future trajectory.

Passive agent A_p rotates to reduce its width perpendicular to $\mathring{\mathbf{a}}_i$, allowing A_i as much space as possible to pass. ϕ_A is chosen such that the central axis $\mathbf{s}_p$ aligns with either $\mathring{\mathbf{a}}_i$ or $-\mathring{\mathbf{a}}_i$, depending on which produces the smallest rotation:

$$\phi_i = \angle(\pm\mathring{\mathbf{a}}_i, \mathbf{s}_p)$$

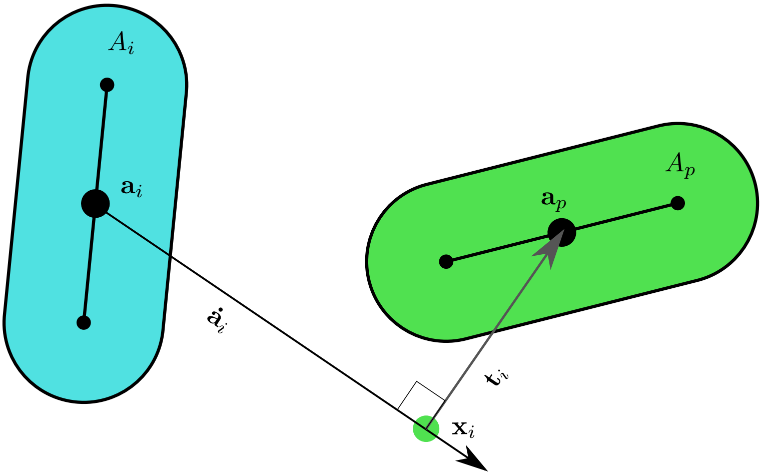


FIGURE 6.9: Active agent avoidance; the passive agent A_p (green) will move to avoid the active agent A_i (cyan). Arrow $\mathbf{t}_i$ indicates the resulting avoidance vector.

The final rotation ϕ_A is the sum of the individual rotations ϕ_i. This summation is very simple; we are interested in a more refined approach, such as computing the rotation to avoid the one agent that is most likely to collide, based on its position and velocity. The avoidance of other agents could then be performed once that agent has been avoided. The investigation of more elaborate methods is left as future work.

To step out of the way of agent A_i, the passive agent translates perpendicular to the velocity vector $\dot{\mathbf{a}}_i$ (see Figure 6.9). For the active agent, we determine the line through $\mathbf{a}_i$ and oriented along $\dot{\mathbf{a}}_i$. For the passive agent, we determine the line orthogonal to $\dot{\mathbf{a}}_i$ and intersecting $\mathbf{a}_p$. The intersection point $\mathbf{x}_i$ of those lines determines translation vector $\mathbf{t}_i$:

$$\mathbf{t}_i = \frac{1}{\delta_i} \frac{\mathbf{a}_p - \mathbf{x}_i}{|\mathbf{a}_p - \mathbf{x}_i|}$$

with dampening factor $\delta_i > 0$. The dampening factor can be agent-specific, to allow for different personality traits. A high dampening factor will make the agent slower to respond than a low dampening factor. In the accompanying video we used $\delta_i = 200$ for all agents. The final agent avoidance translation vector $\mathbf{t}_A$ is the sum of individual translations $\mathbf{t}_i$.

6.4.4 | TURNING DESIRE INTO ACTION

The previous subsections described methods to obtain a vector towards more space $\mathbf{t}_S$, a rotation ϕ_S towards a focus point or to align with the available space, and translation $\mathbf{t}_A$ and rotation ϕ_A to avoid active agents. This section describes how our method selects which translation and rotation to use to produce the agent's motion.

The space finding translation vector $\mathbf{t}_S$ is applied only when certain conditions are met. Firstly, we make an assumption based on the principle of energy minimization. We assume that people accept a marginally worse situation when manoeuvring into a better spot would use significantly more effort than standing still. In our algorithm, the clearance at the found point must be significantly better than the agent's current situation; we use a threshold value of 125% of the agent's current clearance. Not only does this produce more natural results (an irregular distribution of free space among the crowd), it also prevents oscillation between points of similar clearance. Secondly, when making space for someone to pass (see Section 6.4.3), people generally accept a worse situation, as it will only be temporarily. However, people try to move towards an open space if one is available and can be reached while still allowing someone to pass, since this will make it both easier for the passing person and more comfortable for the avoiding person. To model this, space finding vector $\mathbf{t}_S$ is applied only when agent avoidance and space finding result in a translation in roughly the same direction; in other words, when the dot product $\mathbf{t}_S \cdot \mathbf{t}_A > 0$. When these are more or less opposite, only the agent avoidance is performed. The same approach is taken for ϕ_S and ϕ_A; if both rotate in the same direction, they are combined, otherwise only ϕ_A is applied.

To determine the movement of the agent, the planned position $\mathbf{p}_0$ and torso orientation T_o (described in Section 6.3) are defined as:

$$\begin{aligned} \mathbf{p}_0 &= \mathbf{a}_p + \mathbf{t}_A + \begin{cases} \mathbf{t}_S & \text{if } \mathbf{t}_S \cdot \mathbf{t}_A > 0 \\ \mathbf{0} & \text{otherwise} \end{cases} \\ T_o &= \theta_p + \phi_A + \begin{cases} \phi_S & \text{if } \phi_A \phi_S > 0 \\ 0 & \text{otherwise ,} \end{cases} \end{aligned}$$

where θ_p is the passive agent's current orientation. The movement of the agent is controlled in the same way as described for active agents in Section 6.3, with the exception of the maximum translation speed, which is constant for passive agents.

6.5 | WALLS, DOORS, AND OTHER OBSTACLES

In order to model realistic scenarios, our method supports straight walls, doors and polygonal obstacles. To integrate these into the crowd behaviour, they are all modelled as line segments and included as additional sites in the generalized Voronoi diagram (GVD). As a result, the GVD contains line segment sites for agents, walls, doors and obstacles. All these are interpreted by the crowd agents as impenetrable obstacles. Due to the frequent computation of the GVD, obstacles and walls can move dynamically. Agents will adjust for this, but do not anticipate those movements. Such anticipation is left as future work.

Doors are modelled as special wall segments that can be enabled when the door is closed, and disabled when the door is opened. As described in Section 6.4.1, doors are interpreted differently by active and passive crowd members. When a door is open, its line segment simply is not inserted into the active agents' GVD at the next simulation update. The GVD for the passive agents always inserts door line segments, to ensure that the space finding algorithm does not cross area boundaries.

In real life, people anticipate the movement of others. Anticipation in crowd simulation has been studied before [PPD07, KHvBO09, SYN12]; in these works, crowd agents predict other agents' actions by their movement, and adapt their own motion to avoid collisions. Our crowd model takes the opposite approach; our active agents actively notify passive agents of their intentions, by placing information in the environment, similar to the approach by Yeh et al. [YCP+08]. As a real-life example of the intended behaviour, consider a person entering a lift; people appear to mentally model the space required for that person, and make space accordingly. Since the final orientation of the person is not known a-priori, a point would be sufficient to model this. In our simulation, active agents insert point obstacles in the passive GVD, at their goal position $\mathbf{g}_i$. As a result, the passive agents make space around this position, sooner than the avoidance behaviour would. This is applicable only in situations where the active agent's behaviour is predictable, such as when entering or exiting a lift or bus, which is why it is an optional feature of our crowd simulation method.

6.6 | RESULTS

In this section, we validate our Torso Crowd model against a real crowd, in order to find values for parameters that result in human-like behaviour. Furthermore, we investigate our model by looking at several scenarios. We also compare our model with a disc-based crowd simulation: Reciprocal Velocity Obstacles [vdBGLM11].

6.6.1 | VALIDATION AND PARAMETER OPTIMIZATION

To validate our crowd model behaviour, we used motion capture data of a real crowd. This data set contains the torso width and thickness of each participant, and a recording of their locations and torso orientations during each of 47 trials. These recorded motions represent human behaviour in a real situation, and thus form a suitable ground truth for our parameter optimization and model verification. We do note that the recordings were performed in a controlled environment, and thus may not be a faithful representation of day to day scenarios. We leave evaluation using real crowds, for example using video analysis, to future work. The data set is used to validate the behaviour of the active agents, and the passive agents in the interior of the crowd. In the experiment, the active participants had the concrete, reasonably realistic task of manoeuvring through a crowd to a given point. The rest of the crowd had to stand still in a dense configuration, which was necessarily synthetic for the participants at the edge of the crowd due to the set-up of the experiment. To compare the behaviour of the active participants with our crowd simulation system, we look at topological equivalence, rather than Euclidean distance between paths, as the exact positions of the paths are highly dependent on the behaviour of the passive crowd members. The behaviour of the passive crowd agents was compared to the participants by observing how much they align themselves with passing active agents. The parameters for the passive agents are simpler and more intuitive than those for the active agents, and were chosen based on visual inspection of the simulation results of the scenarios described in Section 6.6.2.

Our crowd model uses a number of parameters that determine the behaviour of the active agents, as described in Section 6.2. These parameters, with their optimized values, are shown in Table 6.1. To optimize these parameters, we used the following approach:

1. *Conversion:* The motion capture data is converted to our abstract agent representation, enabling us to input captured situations into our crowd simulation method.
2. *Test sets:* We choose N random frames from the recorded motion capture data. We ensure that each of the chosen frames represents a different situation. The set of frames is separated into two distinct, equally sized, randomly chosen subsets $\mathcal{T}$ for parameter tweaking and $\mathcal{V}$ for verification.
3. *Parameter tweaking:* For each frame in $\mathcal{T}$, the choices of the path planning algorithm are compared with the choices of the participant. We adjust parameters and repeat the comparison, until either all choices made by the path planner are equal to the choices made by the participants, or no more improvements can be made. When the planned path

category	parameter	value	parameter	value
Planner horizon	H_C	3	H_D	1.50 m
	H_ϵ	0.05 m		
Score function weights	w_c	2.30	w_g	1.41
	w_l	0.21	w_m	1.00
Clearance weights	w_c^F	0.1	w_c^A	0.9

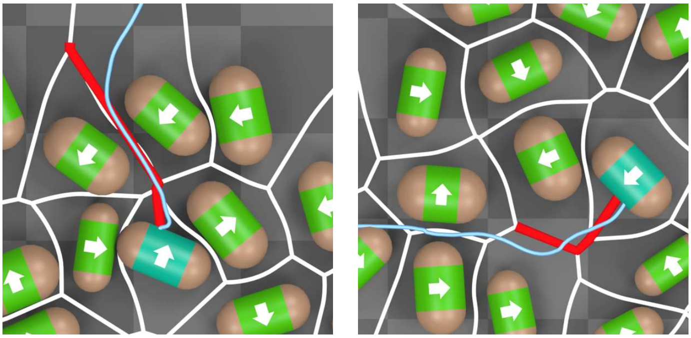
Table 6.1: The path planner parameters obtained from our comparison with our ground truth data. All values were obtained by manual optimization.

passes between the same agents as the participant's motion, they are considered equal.

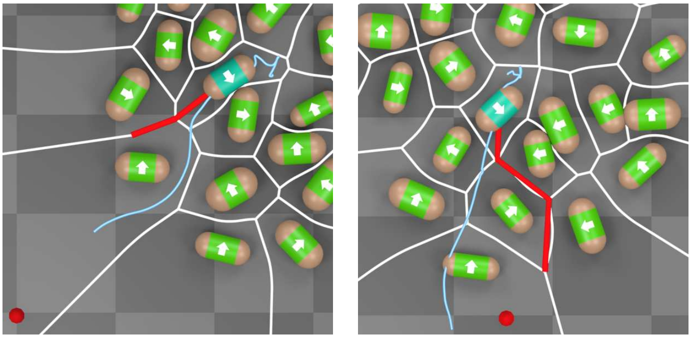
4. *Verification:* For each frame in $\mathcal{V}$, the same type of comparison is performed, as a verification of the parameters. We also measure the difference in planned and recorded torso twist.

We used $N = 80$ to tweak and verify our parameters. Little adjustment was needed during the tweaking phase, resulting in the parameters displayed in Table 6.1. To prevent over-fitting to our motion capture data, we also validated against the behaviour observed in the simulations seen in the accompanying video. During the verification phase, the path planner chose a path that was topologically equivalent to the participants in 85% of the cases. Figure 6.10a shows examples of such correctly planned paths. In four of the six cases where the planner diverted from the recorded data, the planned path was equally plausible (see Figure 6.10b). In the recordings of the other two cases, at the exact frame used for validation, the participant shifted weight from one foot to the other while otherwise stationary, which resulted in a large change in the instantaneous momentum vector and thus in a different path being chosen (see Figure 6.10c); within 1/30 second after the test frame, the planner chose the same path as the participant in both cases. Of course this is not an issue when using simulated data, as our system does not model this weight shifting.

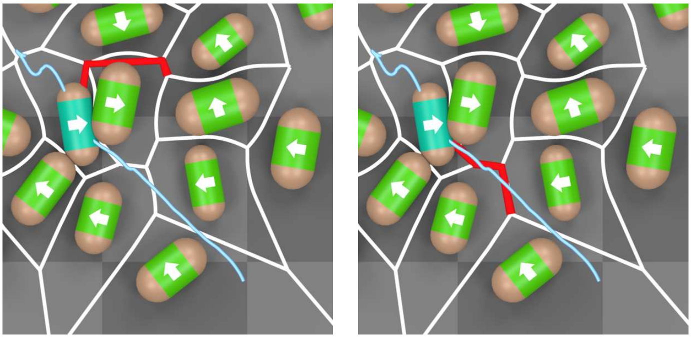
The verification of our model also includes a comparison between the planned and recorded torso orientations for the 34 test cases where the predicted path was topologically equivalent to the path of the recorded participant. To remove the influence of local path variations, we compare the torso *twists*, since these are relative to the agent's and participant's own paths. The twist is defined as the angle between the torso normal and the torso's instantaneous velocity vector (as described in Section 6.3.2). For each verification frame, our Torso Crowd method is used to plan the agent's next short-term target position $\mathbf{p}_0$ and torso twist T_t. The recording is then forwarded to the time



(a) We consider the blue and red paths as topologically equal, since they choose the same route between the same agents.



(b) We consider those two paths as topologically different, since they choose a different route towards the goal, but equally plausible routes towards the goal position.



(c) In the left image, the planner took an unnatural decision, due to noise in the recorded velocity vector. However, 1/60 second later (right image) the planner made the same choice as the participant.

FIGURE 6.10: Comparison between the planned paths (thick red line) and the recorded motion capture data (thin blue line).

where the participant reaches $\mathbf{p}_0$, after which his/her torso twist T'_t is determined. The error is then expressed as the signed difference between the twists: $E = T_t - T'_t$ where the sign of error E indicates whether our planner over-estimates (with E positive) or under-estimates (with E negative) the required twist. In 10 cases we under-estimated the required twist. We classify one of those cases as outlier; it showed a -42^o difference due to the participant moving at that angle even though it was not needed given the available space. In the other under-estimated cases the average error was small at -12^o, and the error was never more than -16^o. In 25 of the 34 cases, we over-estimated the required twist. This is easily explained by the fact that the plan is based on the GVD, which represents the current situation. In the recorded data, it is clear to see that the passive participants make space for the active participant, resulting in more available space, hence less torso twist is required. The average error when over-estimating was 22^o. The largest error in the predicted twist was 76^o. However, in this case the planned global torso orientation was reached within 0.8 seconds after reaching the planned position. Note that we compare the torsos at the moment in time where the distance between the recorded participant and the planned position is minimal. In 16 of the 34 test cases, either the planned torso twist T_t or the global torso orientation T_o is approached (within a 2^o error margin) within 0.5 seconds from that moment in time. This indicates that the recorded participant rotates at a slightly different rate, but still assumes the planned configuration shortly before or after. The average of the absolute error is quite small at 19^o, and the median of 16^o indicates that more than half of the predictions have a smaller-than-average error. We can conclude that our method for simulating active agents corresponds well with the ground truth data.

The avoidance behaviour of the passive participants was also investigated, to confirm that they show the alignment behaviour we model in Section 6.4.3. Since our aim is the simulation of dense crowds, we discarded the participants at the edge of the crowd, and limited this analysis to those that are in a dense situation (see Section 5.3). Their continuous motion was segmented into *avoidance actions*, which are defined as a period in which the participant shows a translation and/or rotation in order to make way for the active participant. In our data set, all avoidance actions consisted of at least a translation (average 0.09 m, $\sigma = 0.07$ m), which allowed us to find the peak in translation speed, and use the local minima around this peak to define the start and end timekeys of each avoidance action. At both timekeys, we measured the angle between the passive participant's torso segment $\mathbf{s}_p$ and the active participant's velocity vector $\dot{\mathbf{a}}_i$. By analysing 94 avoidance actions, we found that at the start of the avoidance action, the average angle was 42^o ($\sigma = 25^o$), and at the end timekey it was 30^o ($\sigma = 22^o$). A paired-samples T-test on the angles shows that this is a strong significant difference ($p < 0.0001$), indicating that there is indeed a trend to align with the active agent's velocity vector. The specific values of the observed averages are of relative importance;

as we measured the angle at translation-dependent points in time, we did not account for any anticipation or other temporal effects. Doing so may produce stronger results, which is left for future research. The simulated avoidance behaviour is parametrized, and can be adjusted to mimic these findings.

6.6.2 | EXAMPLES AND COMPARISON WITH DISC-BASED SIMULATION

We have modelled several scenarios to test our crowd simulation method. As we focus on situations where a large part of the crowd stands still, typical tests where the entire crowd moves do not suffice. Furthermore, in dense crowds people often bump into each other, so a benchmarking method that penalizes collisions, such as SteerBench [SKFR09], will produce unrealistic scores. Instead, we have chosen to use a lift and a hallway to model crowded spaces. All scenarios are simulated at real-time on a single CPU core of a modern PC. The scenarios are shown in the accompanying video. For each scenario, we first show the simulated agents, and then animated characters that follow the motions of those agents.

Small lift In this scenario, the lift visits various floors, and on each floor agents get in or out of the lift (see Figure 6.11). This scenario shows the typical division of the available space seen in lifts: one person by itself stands more or less in the middle of the lift, while the space gets divided when more people enter. While waiting for their floor, the agents turn towards a common focus point: the floor indicator panel above the door. When agents leave the lift, the remaining space is used by the remaining agents. Note that, mimicking real life, the space is *not* optimally divided amongst the agents. Instead, agents around a gap, where an agent stood before leaving the lift, benefit most from the newly available space. The effect of the insertion of the immediate goal of active agents, as described in Section 6.5, can clearly be seen when passive agents make space as an active agent enters the lift.

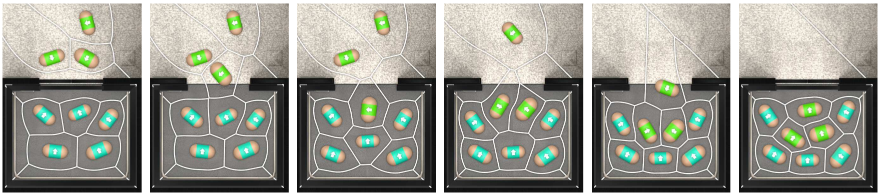


FIGURE 6.11: Stills of the “small lift” scenario. Three agents enter the lift, while the others make space.

Large lift This scenario demonstrates what happens when a group of agents share a focus point, and the density of the crowd increases. The three green agents (see Figure 6.12) share a focus point that is positioned at the centroid of their positions. The other agents in the simulation do not have a focus point. The behaviour of the agents entering the lift is not necessarily natural, since half of them have been scripted to move to the back of the lift. This behaviour is more disruptive to the agents already present, and thus forms a more interesting scenario. Even though the focus point has no direct influence on the agents positions, the three agents stay together.

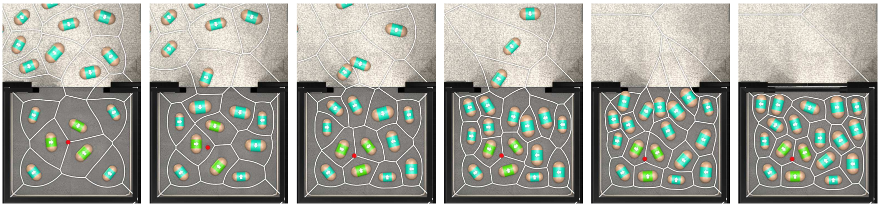
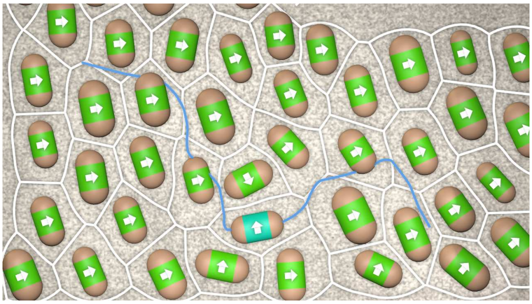


FIGURE 6.12: Stills of the "large lift" scenario. The three green agents have a common focus point (the red dot).

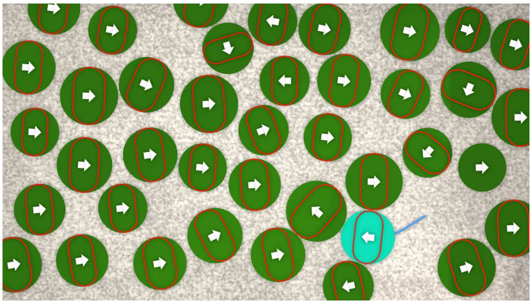
Hallway In this scenario we show a character manoeuvring through a larger crowd in a hallway. One agent tries to manoeuvre towards its goal position, while the remainder of the crowd is stationary; those agents have a zero preferred velocity. We use this scenario to compare the behaviour of our Torso Crowd model with a widely accepted crowd simulation model: Reciprocal Velocity Obstacles (RVO) [vdBGLM11]. This comparison does not aim specifically at RVO; we just use RVO as an example of a good and widely used disc-based crowd simulation model. All passive Torso Crowd agents share the same focus point, out of view on the right-hand side.

Since RVO models agents as discs, we need to convert our capsule representation. We keep in mind that the agents actually represent humanoid shapes; making the RVO agents narrower will result in many undetected intersections. Therefore, the radius is chosen such that the disc encloses the torso capsule, as shown in Figure 6.13b. The blue line shows how far the agent was able to move: in such a dense, stationary crowd, the disc-shaped agents are too big to manoeuvre, while this density is not a problem for Torso Crowd (see Figure 6.13a). One of the underlying issues is that RVO agents make space only to avoid collisions. When the active agent slows down to avoid a collision, the surrounding agents move with only half the speed necessary to avoid the collision. This forces the active agent to slow down even more, finally forcing it to stand stand still. Its velocity vector then becomes zero and holds no information, and the agents in its surroundings will no longer move.

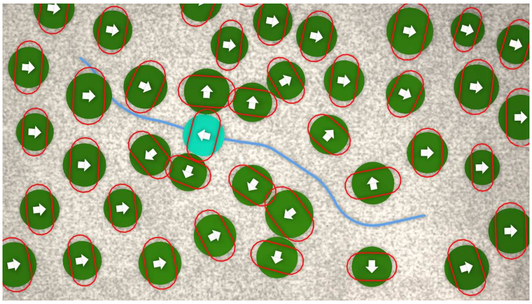
To give the RVO agents more space, we reduce the agent radii, such that the surface area of the agent's ground projection is equal to that of the capsule. This makes the RVO agents narrower but still thicker than the Torso Crowd agents, and results in an equal ground area for RVO and Torso Crowd. The



(a) Torso Crowd finds a path to the goal.



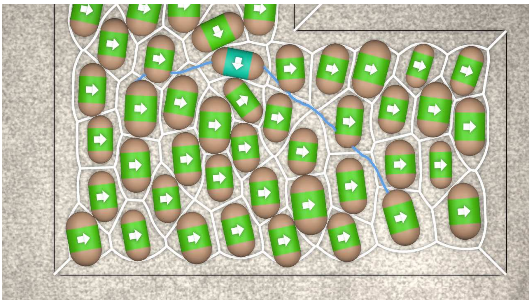
(b) RVO with the same width as the capsules does not find a path to the goal.



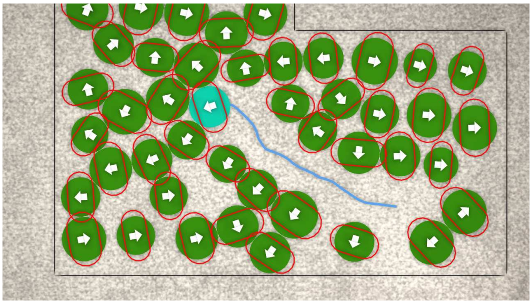
(c) RVO with the same surface area as the capsules to allow manoeuvring.

FIGURE 6.13: Motion paths of a Torso Crowd agent, and RVO approaches. The RVO agents are displayed with a capsule shape overlay, to visualize intersections between agent-driven humanoid characters.

RVO agent can then successfully navigate the crowd, at the expense of intersections between the characters. Figure 6.13c shows this situation, with red capsules to visualize the character torsos. Statistics on our choice of agent sizes are shown in Table 6.2; the average width of 0.44 metres matches the average torso width (measured shoulder to shoulder) of the participants of



(a) Torso Crowd finds a path to the goal.



(b) RVO with the smaller agents, with the same surface area as the capsules, does not find a path to the goal.

Figure 6.14: Motion paths through an even denser crowd. RVO does not find a path to the goal, while Torso Crowd does. The RVO agents are displayed with a capsule shape overlay, to visualize intersections between agent-driven humanoid characters.

Simulation	**shape**	**min**	**max**	**mean**
Torso Crowd	capsule	0.382	0.504	0.443
Regular RVO	disc	0.382	0.504	0.443
Same-area RVO	disc	0.280	0.399	0.345

Table 6.2: Agent diameters used in the comparative scenario, in metres. For capsule agents the diameter is defined as $2r_i + \ell_i$, whereas for disc agents this is $2r_i$.

our motion capture experiment (Chapter 5). We can further increase the crowd density; even the smaller RVO agents move slowly, and eventually do not find a path to the goal (Figure 6.14b). Our Torso Crowd model still handles this situation, and allows the agent to manoeuvre to its goal position (Figure 6.14a).

We can observe more differences. The Torso Crowd agent takes a longer path through the crowd, as it has been configured to avoid areas of low clearance (i.e. agents that stand close together). The RVO agent tries to maintain the shortest path by preferring velocities directly towards the goal position. Another difference is that RVO agents are limited to nonholonomic behaviour; an agent cannot take a step backward or to the side to make room for a passing agent, resulting in unrealistic instantaneous rotations when a human character is animated in its place. Where the passive Torso Crowd agents fill up the space in the wake of the blue agent to make themselves more comfortable, the green RVO agents remain stationary. These results show that the disc shape is not suitable for the simulation of dense crowds. We can also conclude that our Torso Crowd model shows a wider range of motions.

6.7 | CHARACTER ANIMATION

In order to display a humanoid crowd, the motions of the crowd agents need to be mapped to humanoid characters. This poses an under-specified problem. Since only the torso motion is simulated, the lower body orientation needs to be reconstructed before further body animation is possible. In this section we first describe our lower body estimation method, and then the proposed skeletal animation method.

6.7.1 | LOWER BODY ORIENTATION ESTIMATION

The motion data that contains the torso positions and orientations, either from our Torso Crowd simulation or a motion capture recording, is represented as mappings $\mathbf{T}_p : \mathbb{R} \to \mathbb{R}^2$ and $T_o : \mathbb{R} \to \mathbb{S}^1$, from time to respectively position and orientation in the ground plane. Derivative $\dot{\mathbf{T}}_p(t)$ is computed by numerical differentiation. Jitter caused by the numerical approach is filtered out using Gaussian smoothing [Cul71].

The lower body orientation is estimated based on two observations. Firstly, when manoeuvring, the lower body is oriented more or less in the same direction as the torso, and slightly turned towards the direction of motion. Secondly, the lower body cannot instantly change its orientation, so the resulting orientations need to change smoothly. Together, these observations lead to a smoothing scheme based on torso orientation $T_o(t)$.

The lower body estimation is expressed as a function $L_o(t)$, representing the angle of movement relative to the torso orientation. Together with $\mathbf{T}_p(t)$ and $T_o(t)$, it is used for the skeletal animation system described in the next section.

Firstly, we smooth the torso orientation $T_o(t)$. Ordinarily, when smoothing a signal, the smoothed signal lags behind the original. Since the lower body should introduce the motion, as per the first observation described earlier, we use this smoothing lag to our advantage by altering a simple Infinite Impulse Response filter such that it can be evaluated in reversed time:

$$T'_o(t) = T'_o(t+\delta) + (1-\beta)\Big(T_o(t) - T'_o(t+\delta)\Big)$$

where δ is the duration of a simulation frame, 1/60 second in our implementation, and $\beta \in [0,1]$ determines the amount of smoothing. ($\beta = 1$ results in a perfectly smooth, hence constant $T'_o(t)$, and $\beta = 0$ results in no smoothing). Here and in the next equation, the minus sign denotes the signed angular difference on the interval $(-\pi, \pi]$ over the shortest arc. In our implementation, we obtained sufficient smoothing using $\beta = 0.9$. The smoothing filter is applied in reverse time, thus $T'_o(t+\delta)$ is evaluated before $T'_o(t)$. As a result, the smoothed signal “lags in front” of the original signal, producing the desired motion.

Secondly, we compute the angle between the smoothed torso orientation $T'_o(t)$ and the trajectory of the motion $\dot{\mathbf{T}}_p(t)$:

$$L_o(t) = \angle\dot{\mathbf{T}}_p(t) - T'_o(t)$$

where $\angle\dot{\mathbf{T}}_p(t)$ denotes the signed angle of the velocity vector with the world x-axis.

6.7.2 | Skeletal animation

Once the lower body orientation $L_o(t)$ is determined, we can animate the skeletal structure that determines the character’s pose. A commonly used technique for crowd animation is the use of a single walk cycle to animate characters at various speeds, where the animation playback rate depends on each character’s walking speed. Such an approach is simple to implement, but does not support holonomic motion (such as side-stepping). Furthermore, it results in a direct dependency between walking speed and cadence (steps per minute). However, when people change their walking speed, both the cadence and stride length change [KWJ85]. This change in stride length cannot be captured in a single walk cycle, producing unnatural results. To address these issues, the basis for our animation technique is two sets of ten gender-specific walk cycles, consisting of an idle animation (0.0 m/sec), eight slow (0.4 m/sec) walk animations in different directions, and a faster (1.0 m/sec) straight forward walk. Each walk cycle has a fixed pivot point, and hip swing occurs around this point, resulting in a natural hip swing even when we fix the pivot point to a straight path on the ground plane. The eight slow animations consist of straight forward and backward walking, left and right sidestepping, and diagonal steps in four directions. The speed of 0.4 m/sec was chosen for those animations as it was found to be the average speed when manoeuvring through a dense crowd (see Section 5.3).

To produce a character that walks at the correct speed, the joint angles of the animations are blended using weights that depend on the speed of the crowd agent $|\dot{\mathbf{T}}_p(t)|$ and the direction of the motion relative to the torso $L_o(t)$. The speed determines whether we blend between the idle animation and slow walking, and both the speed and direction determine whether we blend between slow and fast walking. The animation's pivot point is positioned at $\mathbf{T}_p(t)$, and oriented at $T_o(t) + L_o(t)$ around the world up-axis. Constraints are placed on the spine bones to incrementally rotate the torso such that it is oriented at $T_o(t)$ around the world up-axis, producing the required torso twists. An example is shown in Figure 6.15.

For simplicity, the head animation is part of the predefined walk cycle animations. This produces slightly more believable results than simply aligning the head with either the torso orientation or the motion trajectory. Future research could combine our technique with a vision-based system, or results from a gaze-in-crowd experiment, to produce natural head movement.

6.8 | CONCLUSION

In this chapter we introduced a novel crowd simulation method, based on the manoeuvring of a number of people in otherwise stationary dense crowds. By extending the common disc-based agent representation to capsules, we are able to plan upper body twisting based on available clearance. Such torso twisting is critical for believable dense crowd manoeuvring.

Our method has been validated against data obtained from real crowd behaviour. The active agent behaviour matches paths chosen by humans in 85% of the cases, and produces different but equally plausible paths in 10% of the cases. The method's parameter values were manually optimized; it would be interesting to investigate automatic parameter tuning such as proposed by Wolinski et al. [WJGO$^+$14] and Berseth et al. [BKHF14]. Even though we used a simplified Voronoi diagram, the resulting behaviour is a close match to the ground truth (as shown in Section 6.6.1). The majority of our validation focused on the behaviour of active agents; further comparison, with different ground truth data, could improve realism of the passive crowd members as well, and could strengthen our design decisions, such as the space-finding behaviour, the assumption that passive agents do not move to different rooms, and show what role focus points play in real crowds.

Regardless of the method to obtain the parameters, it is likely that their scope is limited to high-density situations. Since the planning of torso twist

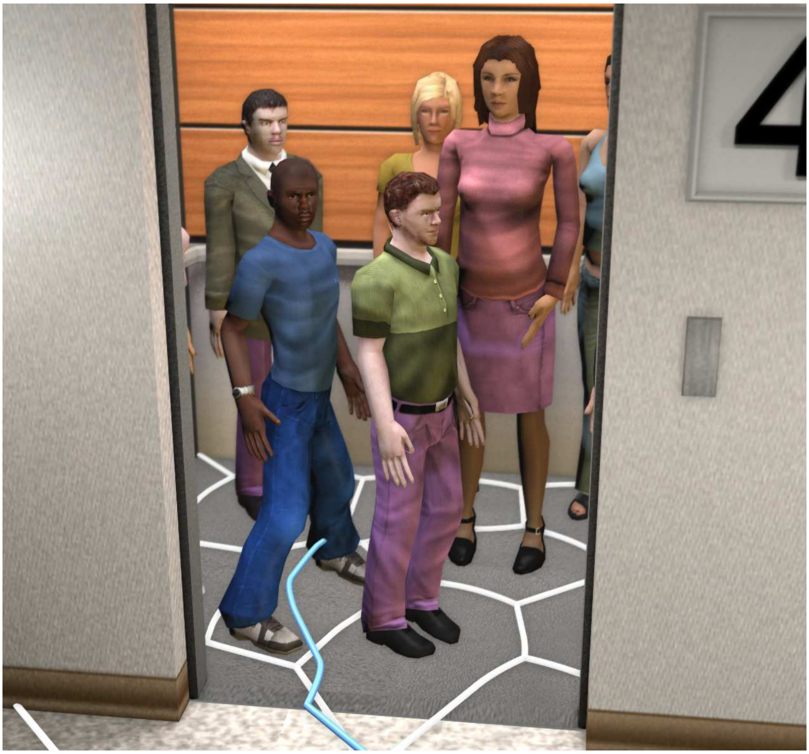


Figure 6.15: A crowd of animated, human characters in a lift. The man in the blue clothing is the active character, whose torso twist is clearly visible. The path is shown in blue on the ground plane.

is no longer a necessity in lower-density crowds, our crowd simulation system switches between our proposed method and a different agent-based method aimed at regular locomotion, depending on the density of the crowd. Alternatively, our system could be extended to handle lower densities, by employing density-dependent parameter values; this should be relatively straightforward, since our density metric is agent-oriented, and the parameters are already adjustable for each agent. It would also be interesting to add velocity-based path planning to our method; for example, the change of clearance over time could be used to prefer small-but-growing openings in the crowd over larger-but-shrinking ones.

The passive agents use a generalized Voronoi diagram to find comfortable places to stand. There is no distinction between agents and walls in such a diagram, and, by definition, it is symmetric in the front and rear of agents. These aspects result in artefacts, such as agents standing too far away from walls, or agents standing too far away from each other when back to back.

A possible solution may be found in a multiplicatively or additively weighted generalized Voronoi diagram [OBSC09], which might also be useful to model the asymmetrical nature of people's personal space [Hay81]. However, since there are no suitable, robust implementations available, we are unable to implement such an approach at this time, and leave this to future work.

In our scenarios, each active agent was appointed a fixed, scenario-specific goal position. When that goal is reached, the agent switches to passive behaviour. Due to the dynamic nature of the crowd, the scripted goal position may not be the most comfortable (see Section 6.4.1), and the agent will move to a desirable point after reaching the goal. When approaching the goal, the active agent could use a local GVD to find a comfortable position in the goal area, and actively move there before switching to passive behaviour.

We presented an animation system that shows walking characters in a crowd using the motions obtained from the simulation. Our system uses a kinematic approach, hence it does not respond to inter-character collisions. Due to the density of the crowd, however, such collisions are likely to occur. We are currently investigating a method employing physics-based characters that follows our torso planning method [KKE15]. Such a system would be able to respond to collisions in a physically correct way, and be used to plan lower-body motion. Another interesting way to extend our model is based on the observation that in dense crowds people often use their arms for navigation. Not only are they used to physically make space, but also for notification as to the intent to pass between people, and as a tactile addition to visual information about one's neighbours in the crowd. A different approach to improving the result of the animation system would be the integration of a footstep-based method, such as described by Singh et al. [SKRF11] and subsequently improved by Berseth et al. [BKF15]; we expect that planning both footstep positions and torso orientations may lead to more natural results.

Further research could extend the Torso Crowd model to allow for a crowd of mostly active agents. It would be interesting to add a velocity component similar to RVO to the planner. Furthermore, the Torso Crowd representation could be employed to reduce the energy needed to manoeuvre a crowd for other crowd simulations. For example, our passive agents anticipate the motions of the active agents, and move aside and twist their torso to make space. Such behaviour can also be observed in less dense crowds, in cases where twisting the torso is not a geometric necessity for someone to pass, but does provide them with a more energy-efficient path. This happens, for example, when making space for someone running towards a train. This shows that torso planning is not limited to dense crowds.

Chapter 7

Conclusion and Closing Remarks

In this thesis we presented a wide range of topics related to the simulation of dense crowds. Our research has resulted in a number of scientific contributions, which we will summarise in this chapter. We also discuss some of the limitations of our work, and of crowd simulation in general, and cover possible avenues for future research.

7.1 | Summary and contributions

In Chapter 3 we introduced the Bounding Cylinder Hierarchy (BCH) as a fast method for collision detection between virtual characters. Its performance is high due to two choices. Firstly, the BCH encodes the contour of the ground projection of a posed character, and thus reduces the three-dimensional problem to a two-dimensional one, at the expense of an increase in false positives. Secondly, we used a tunable threshold radius to determine the precision of the approximation. For different collision detection schemes, namely the BCH and the single cylinder, we measured the average distance between two characters when a collision is reported, and observed a practical limit on the recursion depth to obtain near-optimal results (compared to the result obtained using an unlimited recursion depth). Finally, we showed that an enclosing cylinder contains a order of magnitude larger volume than the posed character itself, and that the BCH and the capsule shape introduced Chapter 5 fit the character much better.

We performed a user study in Chapter 4, to investigate the accuracy with which people can classify a situation with two virtual characters as ‘colliding’ or ‘non-colliding’. We found that, with an accuracy of 72%, people are generally moderately good at recognising collisions. We also observed a bias in the erroneous answers towards ‘non-colliding’, which indicates that people mostly make mistakes when observing colliding scenarios; people thus are generally better at recognising non-collisions than collisions. We conclude that a conservative collision detection strategy that avoids false negatives, which is very common in robotics due to the potentially destructive result of collisions, does not match our perception very well. Unfortunately, this is also applicable to our BCH method. For virtual characters, we suggest a collision detection scheme employing an 'inner approximation' bounded by the character mesh.

In Chapter 5 we performed an experiment to obtain data about the manoeuvring of people in dense crowds. The participants were recorded in a motion capture studio, and subsequently modelled using line segments representing their torsos. For each trial, one participant escaped the crowd, while the other participants remained more or less stationary. We observed that the manoeuvring participants followed edges of a generalized Voronoi diagram based on the shoulder line segment. We investigated two line segments, one between motion capture markers on the shoulders, and the medial axis of a torso-enclosing capsule shape. The latter line segment showed improvement in predicting the participants' movements over the former, strengthening our choice for a capsule-based crowd model.

Based on the capsule shape model introduced in Chapter 5, and the observation in Chapter 4 that such a shape should not avoid all collisions at all costs, we introduced a novel crowd simulation method in Chapter 6. Our model supports efficient manoeuvring through dense crowds, and, by employing a rotationally asymmetric representation, supports the planning of torso twists typically observed in dense crowds. We also distinguish between active and passive agents, and introduce the concept of a focus point to influence the orientation of individual agents and agent groups. We compared our model with an established disc-based model, and showed that, for high crowd densities, our model performs better. Furthermore, we presented an animation technique for virtual characters that allows not only forward walking motion, but also side-stepping, diagonal stepping and backward walking. By basing the method on actual perception and behaviour of real people, we created a more realistic crowd simulation method for dense crowds than current disc-based models.

7.2 | LIMITATIONS AND FUTURE DIRECTIONS

Even though we provided several contributions to the scientific state of the art, there are still many avenues for improvements. Ideas for future work stem from the identification of limitations, which we discuss in this section.

Many crowd simulation methods, Torso Crowd included, separate the planning of motion and body animation. Crowd agent motion is based on simplified shapes, while knowledge of the articulated humanoid structure is only applied in the animation step. As a result, the overall character motion cannot take into account any conflict or constraint in that final step, such as conflicting collision states of the agent and character shapes, constraints like putting a hand on a shoulder of another crowd member, or ‘simply’ physically correct walking animations without foot skating. The footstep-based method by Singh et al. [SKRF11] is an exception to this approach, and we believe that a general solution lies in the unification of motion and animation planning. Such unification could open up possibilities for crowd members to extend their set of possible actions, and allow them to climb over obstacles, push people out of the way, and generally show the type of behaviour seen in riots and panic situations. Furthermore, in real dense crowds, people use their hands, arms, and gaze to communicate their desired direction of movement. In contrast, many crowd simulation systems only use collision avoidance as means of ‘communication’. Making this communication more explicit could also improve the realism of simulated crowds.

The Bounding Cylinder Hierarchy, introduced in Chapter 3, can be seen as a step towards the integration of character pose and agent motion planning. However, using it effectively in a dynamic environment necessitates further research in three areas. The first area is the support for animated characters. Currently such characters are supported by preprocessing pre-existing animations. A method to update the BCH hierarchy in real-time, similar to the work of Van der Bergen [vdB97], would conserve memory and allow collision detection for poses that are not part of pre-existing animations. The second area of research is the creation of a fast distance metric and a method to efficiently compute tangent vectors. Both are used in collision avoidance algorithms; both are also trivial for the cylinder shape, which we think is another reason this shape is so commonly used in crowd simulation methods. Finally, the BCH was constructed with the avoidance of collisions in mind. In our perceptual user study (see Chapter 4), we observed that this common but conservative approach may lead to behaviour that is perceived as unrealistic. Adjusting the collision detection algorithm to form an inner approximation could solve this.

To conclude, we would like to see the validation of crowd simulations using real crowds to become more prevalent. A fundamental limitation of such validations is posed by current markerless video tracking techniques. We are not aware of any such method that can track both position *and orientation* of members in a crowd. Often this limitation is caused by the inherent occlusions in dense crowds. The advent of affordable drones equipped with high-resolution cameras could possibly pose a technical solution to this limitation; by filming straight down from a high altitude, an almost orthogonal projection onto the ground plane could be obtained, making tracking orientations tractable. This would allow the collection of data of bigger crowds in more realistic environments than possible in a motion capture studio.

BIBLIOGRAPHY

[AG85] W.W. Armstrong and M.W. Green. The dynamics of articulated rigid bodies for purposes of animation. *The Visual Computer*, 1(4):231–240, 1985.

[AGHP+00] P.K. Agarwal, L.J. Guibas, S. Har-Peled, A. Rabinovitch, and M. Sharir. Penetration depth of two convex polytopes in 3D. *Nordic J. of Computing*, 7(3):227–240, 2000.

[AH96] K.J. Åström and T. Hägglund. *PID Control*, page 198. CRC Press and IEEE Press, 1996.

[AH06] K.J. Åström and T. Hägglund. *Advanced PID Control*. ISA - The Instrumentation, Systems, and Automation Society; Research Triangle Park, NC 27709, 2006.

[ALHB08a] G. Arechavaleta, J.-P. Laumond, H. Hicheur, and A. Berthoz. On the nonholonomic nature of human locomotion. *Autonomous Robots*, 25(1-2):25–35, 2008.

[ALHB08b] G. Arechavaleta, J.-P. Laumond, H. Hicheur, and A. Berthoz. An optimality principle governing human walking. *IEEE Transactions on Robotics*, 24(1):5–14, 2008.

[Ali11] R. Alizadeh. A dynamic cellular automaton model for evacuation process with obstacles. *Safety Science*, 49(2):315 – 323, 2011.

[AMTT12] B.F. Allen, N. Magnenat-Thalmann, and D. Thalmann. Politeness improves interactivity in dense crowds. *Computer Animation and Virtual Worlds*, 23(6):569–578, 2012.

[BBLA02] O. Burchan Bayazit, J-M. Lien, and N.M. Amato. Roadmap-based flocking for complex environments. In *10th Pacific Conference on Computer Graphics and Applications*, pages 104–113, 2002.

[BC89] A. Bruderlin and T. W. Calvert. Goal-directed, dynamic animation of human walking. *SIGGRAPH Comput. Graph.*, 23(3):233–242, 1989.

[BKF15] G. Berseth, M. Kapadia, and P. Faloutsos. Robust space-time footsteps for agent-based steering. In *Computer Animation and Social Agents (CASA)*, 2015.

[BKHF14] G. Berseth, M. Kapadia, B. Haworth, and P. Faloutsos. SteerFit: Automated parameter fitting for steering algorithms. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, New York, NY, USA, 2014. ACM.

[ble15] Blender, 2015. http://www.blender.org/, accessed 2015-10-14.

[BM14] A. Bera and D. Manocha. Realtime Multilevel Crowd Tracking using Reciprocal Velocity Obstacles. *IEEE International Conference on Pattern Recognition*, 2014.

[bul15] Bullet Physics, 2015. http://www.bulletphysics.org/, accessed 2015-04-02.

[CGZM11] S. Curtis, S.J. Guy, B. Zafar, and D. Manocha. Virtual tawaf: A case study in simulating the behavior of dense, heterogeneous crowds. In *IEEE International Conference on Computer Vision Workshops*, pages 128–135, 2011.

[CH99] S.-K. Chung and J.K. Hahn. Animation of human walking in virtual environments. In *Computer Animation, 1999. Proceedings*, pages 4–15, 1999.

[Che04] S. Chenney. Flow tiles. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 233–242. Eurographics Association, 2004.

[CKGC14] P. Charalambous, I. Karamouzas, S.J. Guy, and Y. Chrysanthou. A data-driven framework for visual crowd analysis. *Computer Graphics Forum*, 33(7):41–50, 2014.

[CoHK15] Census and Statistics Department of Hong Kong. Population, 2015. http://www.censtatd.gov.hk/hkstat/sub/so20.jsp, accessed 2016-01-09.

[Cra68] S.H. Crandall. *Dynamics of Mechanical and Electromechanical Systems*. New York: McGraw-Hill, 1968.

[Cul71] J. Cullum. Numerical differentiation and regularization. *SIAM Journal on Numerical Analysis*, 8(2):pp. 254–265, 1971.

[CVM$^+$96] J. Cohen, A. Varshney, D. Manocha, G. Turk, H. Weber, P. Agarwal, F. Brooks, and W. Wright. Simplification envelopes. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, pages 119–128, New York, NY, USA, 1996. ACM.

[Daa04] W. Daamen. *Modelling Passenger Flows in Public Transport Facilities.* Delft University Press, 2004.

[DeL13] P.R. DeLucia. Effects of size on collision perception and implications for perceptual theory and transportation safety. *Current Directions in Psychological Science*, 22(3):199–204, 2013.

[DO00] J. Dingliana and C. O'Sullivan. Graceful degradation of collision handling in physically based animation. *Computer Graphics Forum*, 19(3):239–248, 2000.

[DTT11] C. Dube, M. Tsoeu, and J. Tapson. A model of the humanoid body for self collision detection based on elliptical capsules. In *IEEE international conference on Robotics and Biomimetics*, pages 2397–2402, 2011.

[Egg06] A. Egges. *Real-time animation of interactive virtual humans.* PhD thesis, University of Geneva, 2006.

[EHEM13] C. Ennis, L. Hoyet, A. Egges, and R. McDonnell. Emotion capture: Emotionally expressive characters for games. In *Proceedings of Motion on Games*, MIG '13, pages 31:53–31:60, New York, NY, USA, 2013. ACM.

[EMO10] C. Ennis, R. McDonnell, and C. O'Sullivan. Seeing is believing: Body motion dominates in multisensory conversations. In *ACM SIGGRAPH 2010 Papers*, SIGGRAPH '10, pages 91:1–91:9, New York, NY, USA, 2010. ACM.

[FLP13] J.L. Fleiss, B. Levin, and M.C. Paik. *Statistical Methods for Rates and Proportions.* John Wiley & Sons, 2013.

[FvdPT01] P. Faloutsos, M. van de Panne, and D. Terzopoulos. Composable controllers for physics-based character animation. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 251–260, New York, NY, USA, 2001. ACM.

[Gau09] C.F. Gauss. *Theoria Motus Corporum Coelestium in Sectionibus Conicis Solem Ambientium.* Cambridge Library Collection - Mathematics. Cambridge University Press, reissue edition (may 19, 2011) edition, 1809.

[GB08] J. Gain and D. Bechmann. A survey of spatial deformation from a user-centered perspective. *ACM Trans. Graph.*, 27(4):107:1–107:21, 2008.

[GCC^{+}10] S.J. Guy, J. Chhugani, S. Curtis, P. Dubey, M. Lin, and D. Manocha. Pledestrians: a least-effort approach to crowd simulation. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '10, pages 119–128, Aire-la-Ville, Switzerland, Switzerland, 2010. Eurographics Association.

[Ger10] R. Geraerts. Planning short paths with clearance using explicit corridors. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1997–2004. IEEE, 2010.

[GKLM11] S.J. Guy, S. Kim, M.C. Lin, and D. Manocha. Simulating heterogeneous crowd behaviors using personality trait theory. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 43–52, 2011.

[Gle98] M. Gleicher. Retargetting motion to new characters. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '98, pages 33–42, New York, NY, USA, 1998. ACM.

[GLM96] S. Gottschalk, M.C. Lin, and D. Manocha. OBBTree: a hierarchical structure for rapid interference detection. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, pages 171–180. ACM, 1996.

[Gol65] G. Golub. Numerical methods for solving linear least squares problems. *Numerische Mathematik*, 7(3):206–216, 1965.

[Got00] S. Gottschalk. *Collision queries using oriented bounding boxes.* PhD thesis, The University of North Carolina, 2000.

[Hau14] F. Hausdorff. *Grundzüge der Mengenlehre.* Veit and Company, Leipzig, 1914.

[Hay81] L.A. Hayduk. The shape of personal space: An experimental investigation. *Canadian Journal of Behavioural Science/Revue canadienne des sciences du comportement*, 13(1):87, 1981.

[Hen71] L.F. Henderson. The statistics of crowd fluids. *Nature*, 229(5284):381–383, 1971.

[Hen72] L.F. Henderson. On the fluid mechanics of human crowd motion. *Transportation Research*, 8(6):509–515, 1972. Revised in 1974.

[HFV00] D. Helbing, I. Farkas, and T. Vicsek. Simulating dynamical features of escape panic. *Nature*, 407(6803):487–90, 2000.

[HM95] D. Helbing and P. Molnár. Social force model for pedestrian dynamics. *Physical Review E*, 51:4282–4286, 1995.

[HMO12] L. Hoyet, R. McDonnell, and C. O'Sullivan. Push it real: Perceiving causality in virtual interactions. *ACM Trans. Graph.*, 31(4):90:1–90:9, 2012.

[HTKG04] B. Heidelberger, M. Teschner, R. Keiser, and M. Müller M. Gross. Consistent penetration depth estimation for deformable collision response. In *Vision, Modeling, and Visualization 2004: Proceedings*. IOS Press, 2004.

[Hub94] P.M. Hubbard. *Collision Detection for Interactive Graphics Applications*. PhD thesis, Brown University, 1994.

[Hub96] P.M. Hubbard. Approximating polyhedra with spheres for time-critical collision detection. *ACM Transactions on Graphics*, 15(3):179–210, 1996.

[HVR$^+$05] H. Hicheur, S. Vieilledent, M.J.E. Richardson, T. Flash, and A. Berthoz. Velocity and curvature in human locomotion along complex curved paths: a comparison with hand movements. *Experimental Brain Research*, 162(2):145–154, 2005.

[Hyv70] L. Hyvärinen. Principal component analysis. In *Mathematical Modeling for Industrial Processes*, volume 19 of *Lecture Notes in Operations Research and Mathematical Systems*, pages 82–104. Springer Berlin Heidelberg, 1970.

[JARLP12] A. Jelic, C. Appert-Rolland, S. Lemercier, and J. Pettré. Properties of pedestrians walking in line: Fundamental diagrams. *Physical Review E*, 85:036111, 2012.

[JCG13] N. Jaklin, A. Cook, and R. Geraerts. Real-time path planning in heterogeneous environments. *Computer Animation and Virtual Worlds*, 24(3-4):285–295, 2013.

[JPCC14] K. Jordao, J. Pettré, M. Christie, and M.-P. Cani. Crowd sculpting: A space-time sculpting method for populating virtual environments. *Computer Graphics Forum*, 33(2):351–360, 2014.

[KCvO07] L. Kavan, S. Collins, J. Žára, and C. O'Sullivan. Skinning with dual quaternions. In *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games*, I3D '07, pages 39–46, New York, NY, USA, 2007. ACM.

[KGM13] S. Kim, S.J. Guy, and D. Manocha. Velocity-based modeling of physical interactions in multi-agent simulations. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '13, pages 125–133, New York, NY, USA, 2013. ACM.

[KGO09] I. Karamouzas, R. Geraerts, and M. Overmars. Indicative routes for path planning and crowd simulation. In *Proceedings of the 4th International Conference on Foundations of Digital Games*, FDG '09, pages 113–120, New York, NY, USA, 2009. ACM.

[KHHL12] M. Kim, Y. Hwang, K. Hyun, and J. Lee. Tiling motion patches. In *Proceedings of the 11th ACM SIGGRAPH / Eurographics Conference on Computer Animation*, EUROSCA'12, pages 117–126, Aire-la-Ville, Switzerland, Switzerland, 2012. Eurographics Association.

[KHI$^+$07] S. Kockara, T. Halic, K. Iqbal, C. Bayrak, and R. Rowe. Collision detection: A survey. In *IEEE International Conference on Systems, Man and Cybernetics*, pages 4046–4051. IEEE, 2007.

[KHM$^+$98] J.T. Klosowski, M. Held, J.S.B. Mitchell, H. Sowizral, and K. Zikan. Efficient collision detection using bounding volume hierarchies of k-dops. *IEEE Transactions on Visualization and Computer Graphics*, 4(1):21–36, 1998.

[KHvBO09] I. Karamouzas, P. Heil, P. van Beek, and M.H. Overmars. A predictive collision avoidance model for pedestrian simulation. In *Motion in Games*, volume 5884 of *Lecture Notes in Computer Science*, pages 41–52. Springer Berlin Heidelberg, 2009.

[KKE15] Z. Kavafoglu, E. Kavafoglu, and A. Egges. Robust balance shift control with posture optimization. In *Proceedings of the 8th ACM SIGGRAPH Conference on Motion in Games*, pages 183–192, 2015.

[Kol98] J.T. Kolowski. *Efficient Collision Detection for Interactive 3D Graphics and Virtual Environments.* PhD thesis, State Univ. of New York, 1998.

[KOLM02] Y.J. Kim, M.A. Otaduy, M.C. Lin, and D. Manocha. Fast penetration depth computation for physically-based animation. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '02, pages 23–31, New York, NY, USA, 2002. ACM.

[KOOP11] R. Kulpa, A.-H. Olivierxs, J. Ondřej, and J. Pettré. Imperceptible relaxation of collision avoidance constraints in virtual crowds. *ACM Trans. Graph.*, 30(6):138:1–138:10, 2011.

[KS05] W. Kerr and D. Spears. Robotic simulation of gases for a surveillance task. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2905–2910, 2005.

[KSG14] I. Karamouzas, B. Skinner, and S.J. Guy. Universal power law governing pedestrian interactions. *Phys. Rev. Lett.*, 113:238701, 2014.

[KWJ85] C. Kirtley, M.W. Whittle, and R.J. Jefferson. Influence of walking speed on gait parameters. *Journal of Biomedical Engineering*, 7(4):282–288, 1985.

[Lar02] P. Larson. Command and Conquer. Technical report, Stanford University, 2002.

[LaV06] S.M. LaValle. *Planning Algorithms.* Cambridge University Press, 2006.

[LCHL07] K.H. Lee, M.G. Choi, Q. Hong, and J. Lee. Group behavior from video: A data-driven approach to crowd simulation. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '07, pages 109–118, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.

[LCL06] K.H. Lee, M.G. Choi, and J. Lee. Motion patches: Building blocks for virtual environments annotated with motion data. *ACM Trans. Graph.*, 25(3):898–906, 2006.

[LDI81] D. T. Lee and R. L. Drysdale III. Generalization of voronoi diagrams in the plane. *SIAM Journal on Computing*, 10(1):73–87, 1981.

[Leg05] A.M. Legendre. *Nouvelles Méthodes pour la Détermination des Orbites des Comètes.* Firmin Didot, 1805.

[LJK$^+$12] S. Lemercier, A. Jelic, R. Kulpa, J. Hua, J. Fehrenbach, P. Degond, C. Appert-Rolland, S. Donikian, and J. Pettré. Realistic following behaviors for crowd simulation. *Computer Graphics Forum*, 31(2pt2):489–498, 2012.

[LMM03] C. Loscos, D. Marchal, and A. Meyer. Intuitive crowd behavior in dense urban environments using local laws. In *Theory and Practice of Computer Graphics*, pages 122–129, 2003.

[Luy96] W.L. Luyben. Tuning proportional-integral-derivative controllers for integrator/deadtime processes. *Industrial & Engineering Chemistry Research*, 35(10):3480–3483, 1996.

[LWC$^+$02] D. Luebke, B. Watson, J.D. Cohen, M. Reddy, and A. Varshney. *Level of Detail for 3D Graphics.* Elsevier Science Inc., New York, NY, USA, 2002.

[may15] Autodesk maya, 2015. http://www.autodesk.com/maya, accessed 2015-10-14.

[MEDO09] R. McDonnell, C. Ennis, S. Dobbyn, and C. O'Sullivan. Talking bodies: Sensitivity to desynchronization of conversations. *ACM Trans. Appl. Percept.*, 6(4):22:1–22:8, 2009.

[Min22] N. Minorsky. Directional stability of automatically steered bodies. *Journal of ASNE*, 42(2):280–309, 1922.

[MMC96] D. Misir, H.A. Malki, and G. Chen. Design and analysis of a fuzzy proportional-integral-derivative controller. *Fuzzy Sets and Systems*, 79(3):297 – 314, 1996.

[MOS09] R. Mehran, A. Oyama, and M. Shah. Abnormal crowd behavior detection using social force model. In *IEEE conference on Computer Vision and Pattern Recognition*, pages 935–942, 2009.

[mot15] Motionstar, 2015. http://www.ascension-tech.com/, accessed 2015-11-05.

[MSW96] J. Matoušek, M. Sharir, and E. Welzl. A subexponential bound for linear programming. *Algorithmica*, 16(4-5):498–516, 1996.

[MTL10] K. Mombaur, A. Truong, and J.-P. Laumond. From human to humanoid locomotion-an inverse optimal control approach. *Autonomous Robots*, 28(3):369–383, 2010.

[MTLT88] N. Magnenat-Thalmann, R. Laperrire, and D. Thalmann. Joint-dependent local deformations for hand animation and object grasping. In *Proceedings on Graphics Interface '88*, pages 26–33, 1988.

[NGCL09] R. Narain, A. Golas, S. Curtis, and M.C. Lin. Aggregate dynamics for dense crowd simulation. *ACM Trans. Graph.*, 28(5):122:1–122:8, 2009.

[Nwa96] H.S. Nwana. Software agents: an overview. *The Knowledge Engineering Review*, 11:205–244, 1996.

[OBSC09] A. Okabe, B. Boots, K. Sugihara, and S.N. Chiu. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*, volume 501. John Wiley & Sons, 2009.

[OCV+02] C. O'Sullivan, J. Cassell, H. Vilhjálmsson, J. Dingliana, S. Dobbyn, B. McNamee, C. Peters, and T. Giang. Levels of detail for crowds and groups. *Computer Graphics Forum*, 21(4):733–741, 2002.

[OD99] C. O'Sullivan and J. Dingliana. Real-time collision detection and response using sphere-trees. *Spring Conference on Computer Graphics*, pages 83 – 92, 1999.

[OD01] C. O'Sullivan and J. Dingliana. Collisions and perception. *ACM Trans. Graph.*, 20(3):151–168, 2001.

[OL03] M.A. Otaduy and M.C. Lin. CLODs: Dual hierarchies for multiresolution collision detection. In *Proceedings of the 2003 Eurographics/ACM Siggraph Symposium on Geometry Processing*, SGP '03, pages 94–101, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[OOP+10] A.-H. Olivier, J. Ondřej, J. Pettré, R. Kulpa, and A. Crétual. Interaction between real and virtual humans during walking: Perceptual evaluation of a simple device. In *Proceedings of the 7th Symposium on Applied Perception in Graphics and Visualization*, APGV '10, pages 117–124, New York, NY, USA, 2010. ACM.

[OPOD10] J. Ondřej, J. Pettré, A. Olivier, and S. Donikian. A synthetic-vision based steering approach for crowd simulation. *ACM Trans. Graph.*, 29(4):123:1–123:9, 2010.

[ORC99] C. O'Sullivan, R. Radach, and S. Collins. A model of collision perception for real-time animation. In Nadia Magnenat-Thalmann and Daniel Thalmann, editors, *Computer Animation and Simulation '99*, Eurographics, pages 67–76. Springer Vienna, 1999.

[PAB07] N. Pelechano, J.M. Allbeck, and N.I. Badler. Controlling individual agents in high-density crowd simulation. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 99–108. Eurographics Association, 2007.

[PAB08] N. Pelechano, J. Allbeck, and N. Badler. *Virtual Crowds: Methods, Simulation, and Control.* Synthesis Lectures on Computer Graphics and Animation. Morgan & Claypool Publishers, San Rafael, 2008.

[Par12] R. Parent. *Computer animation: Algorithms and Techniques, Third edition.* Morgan Kaufmann/Elsevier, 2012.

[Per95] K. Perlin. Real time responsive animation with personality. *IEEE Transactions on Visualization and Computer Graphics*, 1(1):5–15, 1995.

[PPD07] S. Paris, J. Pettré, and S. Donikian. Pedestrian reactive navigation for crowd simulation: a predictive approach. *Computer Graphics Forum, Eurographics 2007*, pages 665–674, 2007.

[Rey87] C.W. Reynolds. Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the 14th annual conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, pages 25–34, 1987.

[Rey99] C.W. Reynolds. Steering behaviors for autonomous characters. In *Proceedings of Game Developers Conference*, pages 763–782. Miller Freeman Game Group, 1999.

[Rey06] C.W. Reynolds. Big fast crowds on PS3. In *Proceedings of the 2006 ACM SIGGRAPH Symposium on Videogames*, pages 113–121, 2006.

[RSLA11] M. Rodriguez, J. Sivic, I. Laptev, and J.-Y. Audibert. Data-driven crowd analysis in videos. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1235–1242, 2011.

[SAC$^+$08] A. Sud, E. Andersen, S. Curtis, M.C. Lin, and D. Manocha. Real-time path planning in dynamic virtual environments using multiagent navigation graphs. *IEEE Transactions on Visualization and Computer Graphics*, 14(3):526–538, 2008.

[SBTM08] S. Saxena, F. Brémond, M. Thonnat, and R. Ma. Crowd behavior recognition for video surveillance. In Jacques Blanc-Talon, Salah Bourennane, Wilfried Philips, Dan Popescu, and Paul Scheunders, editors, *Advanced Concepts for Intelligent Vision Systems*, volume 5259 of *Lecture Notes in Computer Science*, pages 970–981. Springer Berlin Heidelberg, 2008.

[SdGvdSE15] S.A. Stüvel, M.F. de Goeij, A.F. van der Stappen, and A. Egges. An analysis of manoeuvring in dense crowds. In *Proceedings of the 8th ACM SIGGRAPH Conference on Motion in Games*, MIG '15, pages 85–90, New York, NY, USA, 2015. ACM.

[SEE14] S.A. Stüvel, C. Ennis, and A. Egges. Mass population: Plausible and practical crowd simulation. In M.C. Angelides and H Agius, editors, *Handbook of Digital Games*, chapter 6, pages 146–174. Wiley-IEEE Press, 2014.

[SKFR09] 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.

[SKG05] M. Sung, L. Kovar, and M. Gleicher. Fast and accurate goal-directed motion synthesis for crowds. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '05, pages 291–300, New York, NY, USA, 2005. ACM.

[SKRF11] 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.

[SP86] T.W. Sederberg and S.R. Parry. Free-form deformation of solid geometric models. *SIGGRAPH Comput. Graph.*, 20(4):151–160, 1986.

[Sti81] S.M. Stigler. Gauss and the Invention of Least Squares. *The Annals of Statistics*, 9(3):465–474, 1981.

[Stu08] Student. The probable error of a mean. *Biometrika*, 6(1):1–25, 1908.

[SYN12] Y. Suma, D. Yanagisawa, and K Nishinari. Anticipation effect in pedestrian dynamics: modeling and experiments. *Physica A: Statistical Mechanics and its Applications*, 391(1-2):248–263, 2012.

[TFP$^+$10] T.-V.-A. Truong, D. Flavigne, J. Pettre, K. Mombaur, and J.-P. Laumond. Reactive synthesizing of human locomotion combining nonholonomic and holonomic behaviors. In *3rd IEEE RAS and EMBS International Conference on Biomedical Robotics and Biomechatronics*, pages 632–637, 2010.

[TM13] D. Thalmann and S.R. Musse. *Crowd Simulation.* Springer-Verlag London, 2 edition, 2013.

[Tru10] T.V.A. Truong. Unifying nonholonomic and holonomic behaviors in human locomotion. 2010.

[vBEG11] B.J.H. van Basten, A. Egges, and R. Geraerts. Combining path planners and motion graphs. *Computer Animation and Virtual Worlds*, 22(1):59–78, 2011.

[VBSE11] B.J.H. Van Basten, S.A. Stüvel, and J. Egges. A hybrid interpolation scheme for footprint-driven walking synthesis. In *Proceedings of Graphics Interface*, pages 9–16, 2011.

[vdB97] G. van den Bergen. Efficient collision detection of complex deformable models using AABB trees. *Journal of Graphics Tools*, 2(4):1–13, 1997.

[vdB01] G. van den Bergen. Proximity queries and penetration depth computation on 3D game objects. In *Game Developers Conference*, volume 170, 2001.

[vdBGLM11] J. van den Berg, S.J. Guy, M. Lin, and D. Manocha. *Reciprocal n-Body Collision Avoidance*, volume 70 of *Springer Tracts in Advanced Robotics*, pages 3–19. Springer Berlin Heidelberg, 2011.

[vdBLM08] J. van den Berg, M.C. 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.

[vdP97] M. van de Panne. From footprints to animation. *Computer Graphics Forum*, 16(4):211–223, 1997.

[vic15] Vicon motion systems, 2015. http://www.vicon.com/, accessed 2015-11-05.

[VLOG10] G. Vigueras, M. Lozano, J.M. Orduña, and F. Grimaldo. A comparative study of partitioning methods for crowd simulations. *Applied Soft Computing*, 10(1):225–235, 2010.

[vTCG12] W.G. van Toll, A.F. Cook, and R. Geraerts. Real-time density-based crowd simulation. *Computer Animation and Virtual Worlds*, 23(1):59–69, 2012.

[WA92] M. West and H. Asada. Design of a holonomic omnidirectional vehicle. In *IEEE International Conference on Robotics and Automation*, pages 97–103 vol.1, 1992.

[WB85] J.P. Wilhelms and B.A. Barsky. Using dynamic analysis to animate articulated bodies such as humans and robots. In Nadia Magnenat-Thalmann and Daniel Thalmann, editors, *Computer-Generated Images*, pages 209–229. Springer Japan, 1985.

[WC02] Y.-G. Wang and W-J Cai. Advanced proportional-integral-derivative tuning for integrating and unstable processes with gain and phase margin specifications. *Industrial & Engineering Chemistry Research*, 41(12):2910–2914, 2002.

[WJGO+14] D. Wolinski, S. J. Guy, A.-H. Olivier, M. Lin, D. Manocha, and J. Pettré. Parameter estimation and comparative evaluation of crowd simulations. *Computer Graphics Forum*, 33(2):303–312, 2014.

[WS02] M. Wand and W. Straßer. Multi-resolution rendering of complex animated scenes. *Computer Graphics Forum*, 21(3):483–491, 2002.

[YCP+08] H. Yeh, S. Curtis, S. Patil, J. van den Berg, D. Manocha, and M. Lin. Composite agents. In *Proceedings of the 2008 ACM SIGGRAPH/EUROGRAPHICS Symposium on Computer Animation*, pages 39–47, 2008.

[YLvdP07] K. Yin, K. Loken, and M. van de Panne. Simbicon: Simple biped locomotion control. *ACM Trans. Graph.*, 26(3), 2007.

[YMPT09] B. Yersin, J. Maïm, J. Pettré, and D. Thalmann. Crowd patches: Populating large-scale virtual environments for real-time applications. In *Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games*, I3D '09, pages 207–214, New York, NY, USA, 2009. ACM.

[YSLM04] Sung-Eui Yoon, Brian Salomon, Ming Lin, and Dinesh Manocha. Fast collision detection between massive models using dynamic simplification. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, SGP '04, pages 136–146, New York, NY, USA, 2004. ACM.

[Zip49] G.K. Zipf. *Human Behavior and the Principle of Least Effort.* Addison-Wesley Press, 1949.

[ZL14] F. Zhao and J. Li. Pedestrian motion tracking and crowd abnormal behavior detection based on intelligent video surveillance. *Journal of Networks*, 9(10), 2014.

[ZP12] V. Zatsiorsky and B. Prilutsky. *Biomechanics of Skeletal Muscles.* Human Kinetics 10%, 2012.

[ZZL09] X. Zheng, T. Zhong, and M. Liu. Modeling crowd evacuation of a building based on seven methodological approaches. *Building and Environment*, 44(3):437 – 445, 2009.

[ZZYS13] K. Zheng, Y. Zheng, N.J. Yuan, and S. Shang. On discovery of gathering patterns from trajectories. In *IEEE 29th International Conference on Data Engineering (ICDE)*, pages 242–253, 2013.

SAMENVATTING

Door de geschiedenis heen zijn mensen verhuisd van het platteland om zich te concentreren in de steden; tevens blijft de wereldbevolking gestaag groeien. Als gevolg zien we nu de hoogste populatiedichtheden ooit. Daarnaast worden evenementen, en evenementslocaties zoals voetbalstadions en concerthallen, steeds groter. Hierdoor zien we steeds vaker grote samenscholingen van mensen; hun veiligheid en welzijn wordt belangrijker, en tegelijkertijd moeilijker te voorspellen. Recente gebeurtenissen laten zien hoe snel een situatie om kan slaan wanneer er een groot publiek bij betrokken is. Voorbeelden zijn de verpletterende rampen in het Hillsborough stadion in 1989 en de Love Parade in Duisburg in 2010. Computersimulaties van dergelijke gebeurtenissen kunnen op verschillende manieren helpen. Politie, *riot control* en medische medewerkers kunnen simulaties gebruiken voor trainingsdoeleinden, en voor het testen van de effectiviteit van verschillende crowd management-strategieën in een veilige en gecontroleerde virtuele omgeving. Gemeenteraadsleden en evenementsorganisatoren kunnen ook simulaties gebruiken om mensenstromingen te onderzoeken, en voor het vinden van potentieel gevaarlijke knelpunten. Zelfs voordat de bouw begint kunnen gebouwontwerpen worden doorgerekend op doorstroming en ontruimbaarheid. In deze simulaties is het essentieel dat het gedrag van de gesimuleerde menigte representatief is voor het gedrag van echte mensen.

Door ontwikkelingen in gaming hardware is het realisme van computer games sterk verbeterd; karakters ogen en gedragen zich steeds realistischer, en het is mogelijk geworden om virtuele werelden te bevolken met hoge aantallen van dergelijke karakters. De toevoeging van mensenmassa's in games geeft de speler een sterker gevoel van aanwezigheid in een 'echte' virtuele wereld, en het voorkomt dat de omgeving er uit ziet als een spookstad. Een goed voorbeeld is te zien wanneer we IO Interactive's game *Hitman: Codename 47* (2000, zie Figuur 1.1a) vergelijken met de nieuwste game in de serie, *Hitman: Absolution* (2012, zie Figuur 1.1b). Waar de eerste game de stad Hong Kong, bekend om zijn zeer hoge bevolkingsdichtheid [CoHK15], afgeschildert als leeg en verlaten, toont de opvolger uit 2012 steden vol met mensen, ook al hebben deze steden in werkelijkheid een lagere bevolkingsdichtheid. Verder is in deze laatste game de mensenmassa een meer geïntegreerd onderdeel van de spelervaring dan in de voorgangers uit de serie.

De belangrijkste motivatie voor het onderzoek gerapporteerd in dit proefschrift is het verlangen om het realisme van simulaties van dichte menigten te verhogen. De meeste simulatiemethoden maken gebruik van zeer eenvoudige symmetrische vormen, namelijk schijven of punten, om de mensen in de menigte te representeren. De oriëntatie van deze vormen wordt direct gerelateerd aan de bewegingsrichting, met als gevolg dat karakters instantaan om hun as kunnen draaien. De gebruikelijke schijfvorm veroorzaakt ook onmenselijk gedrag in dichte situaties; wanneer veel *agents* worden samengedrukt in een klein gebied, wordt de vorm indirect zichtbaar, doordat de bewegingen lijken op die van langs elkaar schuivende bierflesjes in een fabriek. Deze observaties leiden in Hoofdstukken 5 en 6 tot het gebruik van een capsulevorm, welke meer lijkt op de vorm van de menselijke torso dan een schijf of punt, en tevens de mogelijkheid geeft tot meer realistische bewegingen, zoals opzij stappen, achteruit lopen, en het verdraaien van de torso om door smalle openingen in de menigte te manouvreren.

Metrieken om de prestaties van verschillende menigtesimulatiemethoden te vergelijken, zoals Steer Bench [SKFR09], richten zich in het algemeen op de het vermijden van botsingen. Hoewel het ontwijken van botsingen op zichzelf realistisch is, is een simulatie van een dichte menigte die geheel geen botsingen vertoont dat niet. Dit leidde ons tot het produceren van een techniek voor het snel detecteren van botsingen, gepresenteerd in Hoofdstuk 3. Aangezien deze techniek gebruik maakt van een benadering van de menselijke vorm, waren we ook geïnteresseerd in de mogelijkheden van mensen om botsingen tussen virtuele karakters te herkennen, en hebben we onderzocht hoe goed deze benadering past bij onze menselijke waarneming. De gebruikersstudie en onze bevindingen worden gepresenteerd in Hoofdstuk 4.

Deze thesis is onderverdeeld in zeven hoofdstukken, met een introductie in Hoofdstuk 1.

Hoofdstuk 2 behandelt fundamentele achtergrondinformatie over coördinatenstelsels, afstandsmetrieken, wiskundige notatie, karakter- en mensenmassa-animatie, PID controllers, en enkele statistische methoden.

Hoofdstuk 3 introduceert een snelle botsingsdetectiestrategie gebaseerd op hiërarchieën van cylinders. Enkele cylinders worden veel gebruikt voor het detecteren van botsingen tussen virtuele karakters; door middel van steeds kleinere cylinders kunnen we deze vorm detailleren, en de botsingsdetectie preciezer maken.

Hoofdstuk 4 beschrijft een gebruikersonderzoek naar de mogelijkheid van menselijke observanten om botsingen tussen virtuele karakters te herkennen. We zien dat mensen over het algemeen goed zijn in het herkennen van nietbotsingen, maar minder goed in het herkennen van botsingen, en identificeren enkele parameters die significant zijn voor deze herkenning.

Hoofdstuk 5 behandelt een experiment met een echte mensenmassa van 23 participanten in een motion capture studio, met als doel het verzamelen van informatie over het mensenmassamanouvreergedrag. We observeren dat een Voronoi diagram van de participanten gebruikt kan worden voor het voorspellen van de richting waarin ze zich door de mensenmassa bewegen.

Hoofdstuk 6 beschrijft een mensenmassasimulatiemethode specifiek gericht op het simuleren van mensenmassa's van hoge dichtheid, gebaseerd op de resultaten van de voorgaande hoofdstukken, en introduceren een capsule-vorm als representatie van mensen in de massa. De mensenmassasimulatiemethode ondersteunt verschillend gedrag voor mensen die actief door de massa manouvreren en voor mensen die geen aanleiding hebben om dat te doen. Het hoofdstuk behandelt ook een karakteranimatiemethode voor holonomische bewegingen, zoals het zetten van stappen opzij en achteruit.

Hoofdstuk 7 sluit de thesis af door een overzicht te geven van het verwezenlijkte werk. Tevens wordt er ingegaan op limitaties van ons werk, en behandelen we mogelijke toekomstige onderzoeksrichtingen.

ACKNOWLEDGEMENTS

First of all, I would like to thank my copromotors Arjan Egges and Frank van der Stappen for offering me this great opportunity to perform my own research for four years. I have greatly enjoyed working together, and I'm especially impressed by our personal contact, often hilarious meetings, and strong focus on the scientific qualities of our joint research. I thank my promotor Professor Marc van Kreveld for his support and thorough attention to detail. All three gentlemen have always been supportive and ready to discuss scientific matters, and their feedback on my articles and thesis has been invaluable.

I sincerely thank Professors Nadia Magnenat-Thalmann and Daniel Thalmann for providing me with the means to visit them in Singapore on two occasions, and work with them at the Institute of Media Innovation, Nanyang Technological University. Our collaboration has had a large impact on the direction of my research. I am very grateful for these opportunities, and look forward to working together in the future.

I am grateful for the time and effort spent by Professors Daniel Thalmann, Dirk Heylen, Elmar Eisemann, Peter Werkhoven, and Dr. Julien Pettré by forming my promotion committee and providing me with feedback on this thesis. Norman Jaklin, Wouter van Toll, Arne Hillebrand and Roland Geraerts have my gratitude for providing, and helping with, the software I used to compute the generalized Voronoi diagrams that play such a big role in my work. For their support in statistical matters, I thank Peter de Waal, Cathy Ennis and Ad Feelders. I also thank my colleagues Michael Wand, Zümra Kavafoğlu, Nicolas Pronost, Mart Hagenaars, Thijs de Goeij, Zerrin Yumak, Rohit Dubey and Chax Rivera for our fruitful and enjoyable time working together. I also thank Ton Roosendaal and the Blender developers for their help with Blender and the changes I needed to get my crowd simulation system to work in the Blender game engine.

Of course I want to thank my parents Rob and Lies, and my in-laws Coen and Nelleke, for their support, pride and enthusiasm. I thank Ork de Rooij and Ton Wessling for being awesome friends and having my back at my defence. My band The Soundabout, Rutger, Peter, Bart and Mathijs, and Mahjong friends Jolanda and Erik, have been instrumental in my life's pleasure in the past four years. I also thank Hjalti Hjálmarsson for introducing me to the pain of manual character animation, and being an eye opener for a computer scientist like myself.

Finally I thank my wife Marit, for her support on all different levels, acceptance of my grumpy responses when I'm too focused on my work, eagerness to understand everything I'm doing, and tremendous love.

PUBLICATIONS

This thesis is based on the following publications:

Chapter 3

SYBREN A. STÜVEL, NADIA MAGNENAT-THALMANN, DANIEL THALMANN, ARJAN EGGES, AND A. FRANK VAN DER STAPPEN. *Hierarchical Structures for Collision Checking between Virtual Characters.* Computer Animation and Virtual Worlds, issue 25, pages 333–342, 2014.

Chapter 4

SYBREN A. STÜVEL, A. FRANK VAN DER STAPPEN, AND ARJAN EGGES. *Perception of Collisions between Virtual Characters.* Under review for publication in Computer Animation and Virtual Worlds.

Chapter 5

SYBREN A. STÜVEL, THIJS F. DE GOEIJ, A. FRANK VAN DER STAPPEN, AND ARJAN EGGES. *An Analysis of Manoeuvring in Dense Crowds.* Proceedings of Motion in Games, Paris, 2015

SYBREN A. STÜVEL, THIJS F. DE GOEIJ, A. FRANK VAN DER STAPPEN, AND ARJAN EGGES. *An Analysis of Manoeuvring in Dense Crowds.* Technical report UU-CS-2015-006, Department of Information and Computing Sciences, Utrecht University

Chapter 6

SYBREN A. STÜVEL, NADIA MAGNENAT-THALMANN, DANIEL THALMANN, A. FRANK VAN DER STAPPEN, AND ARJAN EGGES. *Torso Crowds.* Accepted for publication in IEEE Transactions on Visualization and Computer Graphics.

CURRICULUM VITAE

Sybren A. Stüvel was born on 26 April 1979 in Gouwe, municipality Opmeer, The Netherlands. He finished his VWO (pre-university education) at the Berger Scholengemeenschap in Bergen in 1997. He obtained his Bachelor of Science degree in computer science at the University of Amsterdam in 2006, and received his Master of Science degree with honours at Utrecht University in 2010, on the topic of character animation. During and after his studies at both universities, he worked as software engineer at different companies, most notably Eyefi Interactive in Amsterdam, and Chess in Haarlem. In 2011, he started a PhD at Utrecht University, for which he completed his thesis in 2016.