Automating Crowd Simulation

From parameter tuning to dynamic context-to-policy adaptation

Beatriz Cabrero Daniel

TESI DOCTORAL UPF / year 2021

THESIS SUPERVISORS
Josep Blat and Ricardo Marques
Department de Tecnologies de la Informació i les Comunicacions

Universitat Pompeu Fabra
Barcelona

To everyone who bears with
me while I grunt and sweat
under a weary life.

Thanks to my advisors for their persistent efforts in keeping me afloat by putting things in perspective.

First, I'd like to thank Dr. Ricardo Marques for his support and guidance throughout the process. For all of his questions, comments, corrections, advice, examples, and for helping me to converge, I am incredibly grateful. This thesis would not have been possible without Dr. Julien Pettré, who combines his extensive knowledge of Crowd Simulation with wild ideas that are the key to propel research forward. Thanks to him, I was invited for a couple of research stays at Inria Rennes Bretagne Atlantique where he leads a group of outstanding researchers who never failed to make me feel at home. Dr. Ludovic Hoyet deserves a special mention for his help and his positivity. He's the best co-author you can ask for; his knowledge about human perception and statistical analysis amazes and frightens me in equal measure. Last but not least, I'd want to express my gratitude to Dr. Josep Blat for his patience, encouragement, wisdom, pragmatism, and stories... I wish I had understood them sooner.

None of this work would have been possible without the promptness of the UPF-DTIC administrative staff's replies and the moral support from my colleagues from the GTI research group and many others, including the Rainbow team at Inria Rennes. They are friends, after all, and my deepest gratitude goes out to them.

Abstract

Computer-generated crowds are becoming more and more frequent in films, video games and safety assessment applications. Many crowd simulation algorithms exist to address the needs of this diverse range of industries. Even though the underlying principles are similar, there are large differences between the resulting synthetic trajectories. Each algorithm has strengths and weaknesses that need to be weighted, and appropriate parameter values for them must be selected as well. These are not easy tasks and Machine Learning algorithms are often used to guide these decisions. In this work we study three of these tasks: parameter tuning, trajectory evaluation, and character policy selection and adaptation. Our results show the usefulness of the proposed methods to evaluate previously unseen synthetic trajectories to find appropriate parameter values for the algorithms without directly relying on real data. Moreover, by classifying the context of characters, we propose a policy adaptation strategy to improve crowd simulations.

Resum

Les multituds simulades per ordinador són cada cop més habituals en cinema, vídeo jocs i en aplicacions relacionades amb la seguretat. Existeixen molts algoritmes per simular multituds per adreçar tal varietat d'indústries. Tot i que els principis subjacents són similars, hi ha diferències entre les simulacions resultants. Cada algoritme té avantatges i inconvenients que s'han de valorar i, a més a més, cal trobar valors pels seus paràmetres. Aquestes no són tasques senzilles i, sovint, es fan servir algoritmes d'aprenentatge automàtic per guiar aquestes decisions. Estudiem tres d'aquestes tasques: donar valor als paràmetres, avaluar trajectòries, i adaptar les polítiques. Els resultats demostren la utilitat dels mètodes proposats per avaluar trajectòries noves per tal de trobar valors apropiats pels paràmetres dels algorismes sense fer servir dades reals directament. A més a més, proposem una estratègia per adaptar la política de cada agent a través del reconeixement del context, millorant les simulacions.

Table of contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

Crowd simulation is a very active research field, with several applications in different industries ranging from entertainment to safety assessment.

Computer-generated crowds are becoming more and more frequent in film making. They can be employed to generate dynamic scenes with believable crowds, hence improving the realism of a take in movies or television shows such as *The Lord of the Rings*, *Game of Thrones*, or *The King*. Another use of crowd simulation in the movie industry is to capture shots that would require a large number of extras and/or putting real people in danger. Battle scenes involving large armies are an example of this. The ability to manipulate the results is a fundamental feature of crowd simulators in both cases. The artist must have an extensive control over the simulation to adjust the crowd's behaviour to recreate a certain behaviour.

Crowd simulators in video games are responsible for steering dynamic *non-player characters* (NPC). From pedestrians walking down the street to warriors in a battle scene, the goal is to populate the interactive environment with plausible crowds. An example of this are the large crowds in *Assassin's Creed Unity*. As opposed to the film industry, where the time to generate the crowd can be relatively long, interactive experiences (in this case, video games) must simulate NPCs in real time. In addition, just as in film making, a high amount of control over the outcome is needed, together with a sufficient level of trajectory believability.

Crowd simulation is also used in the field of safety assessment in order to study and prevent dangerous situations involving thousands of people. An example of this is the *Astroworld Festival crowd crush* (November 2021). For instance, without putting real people's lives in danger, crowd simulations can be used to analyze a fire drill. Using this approach, emergency evacuations can be assessed, problematic areas can be accurately identified, and appropriate decisions can be made in order to create safe environments and contingency plans. Furthermore, crowd simulation can be utilized to examine pedestrian behavior from a high level perspective. This is extremely helpful in determining the relationship between the environment (e.g., topography) and the navigation patterns of a significant number of people. Based on this knowledge, it is then possible to design safe and ergonomic public areas using this information.

The purpose of a crowd simulator is to synthesize the motion of numerous agents, based on a defined model of the crowd motion. Many crowd simulation algorithms have been designed to address the needs of such a diverse range of applications and industries. The expectations about simulation results generally include a set of conditions such as the absence of collisions and of unnatural movements, yet models are typically developed for a specific purpose and the results differ. For instance, the Optimal Reciprocal Collision Avoidance approach (ORCA) by van den Berg et al. [2011] is designed to control mobile robots without communication among them. ORCA is very common and it's often considered a standard for video games too. Nevertheless, for applications where human pedestrians need to be simulated realistically, other motion characteristics need then to be considered, such as the fundamental diagram (that relates the flow of characters with respect to the density) [Jelić et al., 2012; Chattaraj et al., 2009], visual information [Dutra et al., 2017b], or the perception of motion artifacts in 3D visualisations [Kulpa et al., 2011; Molina et al., 2021; Hoyet et al., 2016]. Each approach has advantages and disadvantages that must be considered and weighted for each particular task or application, which makes proposing generic solutions to simulate pedestrians extremely challenging.

1.1 Problem

It has long been established that the simulation algorithms and their parameters have a direct impact on the resulting quality of animations. It is also known that using appropriate parameter values for a specific scenario can improve performance. It's difficult to figure out what parameters values to use, though, even by hand. Furthermore, comparing algorithms is difficult due to the various manners in which they are implemented. To simplify this process, we unified the implementations of a number of well known motion algorithms and created a common interface to test and experiment on them.

Measuring the performance of motion algorithms is usually performed by evaluating the resulting trajectories but assessing the performance of a steering algorithm or parameter values is difficult itself. Data-based approaches face the challenge of finding similarities between simulations and recorded motions. Characters do not learn how to move, but to imitate pre-recorded motions. Other approaches, measure performance according to some metrics. What trajectory characteristics are visually relevant? To what degree a metric based on such characteristics is correlated with human perception? How do we study the admissible value range for these characteristics without using real data directly (would limit the flexibility and range of application)?

Usually, steering algorithms are used from the beginning to the end of a simulation. The question of what steering algorithm is preferable (e.g., does replicate a real trajectory better) appears. Nevertheless, as a character moves, its context changes. How can the context of a character be described? How can we adapt the policy to the context? Can simulations benefit from adapting the characters' motion to their current context?

As the number of published steering algorithms and techniques continues increasing, so does the difficulty of comparing their performances. The most difficult for a non-expert user is to decide what character policy to obtain certain results. Thus, a major objective of this thesis is to propose generic solutions to interconnect the aforementioned process in order to automatize (or to support better informed) decision making.

1.2 Approach

1.2.1 Considered crowd simulation algorithms

Simulators are based on several classes of algorithms, which are intended to generate believable (or, in some cases, realistic) trajectories of the agents in a crowd. Various approaches to this problem have been proposed. We provide next an overview of our approach to crowd simulation.

Macroscopic approaches consider crowds as a whole, modeling it as a single continuous moving matter [Hughes, 2003; Treuille et al., 2006]. On the other hand, *microscopic* crowd simulation algorithms set the principles by which agents move individually and global crowd motion effects are expected to emerge from the interactions between agents. The seminal work of [Reynolds, 1987] explored how to control boids by each following the mean velocity field generated by neighbors. The number of categories of simulation algorithms rapidly grew with force-based models [Helbing and Molnár, 1995; Karamouzas et al., 2014], velocity-based models [Paris et al., 2007; van den Berg et al., 2008; Karamouzas et al., 2009], vision-based models [Ondřej et al., 2010; Dutra et al., 2017a], or data-driven models [Lerner et al., 2007; Charalambous and Chrysanthou, 2014a]. These are few examples of a large body of literature. Many variants specialize their basic algorithms to extend the range of scenarios they can handle. Nevertheless, each category of algorithms will instil one single way to move and interact in agents - a function of the state of the surrounding agents and of the environment - and agents never deviate from this principle. Nevertheless, each algorithm is designed for relatively specific situations, and in spite of the efforts to always increase their validity range, each of them only captures a portion of all behaviours human exhibit in crowds. This can result in lack of realism of the simulated crowds.

In this work, we focus on the microscopic crowd simulation class, where algorithms express models of how each character moves and interacts with its surrounding environment. A detailed explanation of the implementation some algorithms mentioned here is given in Section 2.4.

1.2.2 Evaluation of synthetic trajectories

There are many available microscopic crowd simulation algorithms. Even though the underlying principles are similar, there are large differences between the resulting synthetic trajectories. Each algorithm has strengths and weaknesses and people wishing to simulate a crowd need to decide which to use in each context. Moreover, together with the selection of the algorithm, appropriate parameter values must be selected as well. These are not easy tasks and machine learning algorithms are often used to guide these decisions. Chapter 2 discusses the learning framework that we use throughout this work.

This learning can be done by evaluating the resulting trajectories as a measure of the fitness of a parameter value set. A group of approaches uses paths of real crowds, and evaluate the ability of simulators to reproduce them. The question of comparison metrics is central, and several solutions have been proposed: Guy et al. [2012]; Wolinski et al. [2014a]; Charalambous et al. [2014a], for instance. However, there are drawbacks associated with the use of reference data, e.g., over-fitting due to the limited sample of pedestrian trajectories available. In Chapter 3, we discuss an approach to evaluation which overcomes this problem. The proposed approach served as a first step to automatise the selection of steering algorithms and parameter values.

1.2.3 Generating heterogeneous and dynamic crowds

There is a growing body of literature that recognises the different performance of steering algorithms in different scenarios e.g., van Toll and Pettré [2021]; Yang et al. [2020]. Numerous studies try to find the best parameters for existing steering algorithms, often comparing the results using data-based performance metrics [Guy et al., 2012]. The objective of these works is to aid the selection of policies in order to improve the trajectories resulting from simulation. Karamouzas et al. [2018a] compare the performance of steering algorithms (using default parameters) in terms of distance-to-real-data in different scenarios. They compute how closely each steering algorithm is able to replicate specific real crowd trajectories e.g., which steering algorithm (out of 6) works best in a medium density area when entering a bottleneck corridor. This opens the door to simulating *heterogeneous* crowds, where the rules governing the motion of each character might be different. Some authors have even proposed strategies to profit from two steering strategies so that characters can adapt to *dynamic* conditions such as changes in density. van Toll et al. [2020a] combine agent-based (the Social Forces model) and particle-based approaches (Smoothed Hydrodynamic Particles) through abstraction layers in order to improve the behaviour in high density regions, hence making their policy dynamic.

Chapter 4 describes how we identify, for an heterogeneous crowd, the context of each character at each time step. Moreover, we discuss an strategy to pick the optimal steering algorithm for each context and to dynamically adapt the policy of each character, hence improving the simulation quality.

1.3 Contributions

The contributions presented in this thesis can be summarized as follows:

Models as costs. Local navigation algorithms describe how characters move based on their surroundings. Many algorithms have been proposed, each using different principles and implementation details, so that they are difficult to compare with each other. Section 2.4 describes a novel framework that implements local agent navigation as a cost optimisation in a Velocity Space. We showed that many state-of-the-art algorithms, whose implementation is not always available, can be translated to this framework. This software is used to experiment with different algorithms and parameter values through a unified interface. Moreover, this approach helps understanding the differences between navigation methods and enables objective comparisons and combinations between them.

Citation: Wouter van Toll, Fabien Grzeskowiak, Axel López Gandía, Javad Amirian, Florian Berton, Julien Bruneau, Beatriz Cabrero Daniel, Alberto Jovane, and Julien Pettré (2020b). Generalized microscropic crowd simulation using costs in velocity space. In *Symposium on Interactive 3D Graphics and Games*, I3D '20, NewYork, NY, USA. Association for Computing Machinery. DOI: 10.1145/3384382.3384532

Quality function. We integrate two strategies study the relation between parametric values for simulation techniques and the quality of the resulting trajectories, which had previously been studied, namely, either through perceptual experiments or by comparison with real crowd trajectories. As discussed in Chapter 3, a quality metric, QF, was proposed to abstract from reference data while capturing the most salient features that affect the perception of trajectory realism. QF weights and combines cost functions that are based on several individual, local and global properties of trajectories. These trajectory features were selected from the literature and from interviews with experts. To validate the capacity of QF to capture perceived trajectory quality, we conducted an online experiment that demonstrated the high agreement between the automatic quality score and non-expert users. To further demonstrate the usefulness of QF, we used it in a data-free parameter tuning application able to tune any parametric microscopic crowd simulation model that outputs independent trajectories for characters. The learnt parameters for the tuned crowd motion model maintained the influence of the reference data which was used to weight the terms of QF.

Citation: Beatriz Cabrero Daniel, Ricardo Marques, Ludovic Hoyet, Julien Pettré, and Josep Blat (2021). A perceptually-validated metric for crowd trajectory quality evaluation. *Proc. ACM Comput. Graph. Interact. Tech.*, 4(3). DOI: 10.1145/3480136

Coverage of models. We address the question of choosing the right crowd simulation algorithm with the right parameter values. This is of crucial importance given the large impact on the quality of results. In Chapter 4 we discuss how these two concepts were combined in order to achieve an autonomous and informed mapping from context, representing the environment of a character, to best-performing policy (i.e., simulation algorithm and its parameters). We study the performance of a number of steering policies in a variety of contexts, resorting to the proposed quality function, QF. This analysis allows us to map contexts to the performance of steering policies. Based on this mapping, we demonstrate that distributing the best performing policies among characters improves the resulting simulations. Furthermore, we also propose a solution to dynamically adjust the policies, for each agent independently and while the simulation is running, based on the local context each agent is currently in. We demonstrate significant improvements of simulation results compared to previous work that would optimize parameters once for the whole simulation, or pick an optimized, but unique and static, policy for a given global simulation context.

Citation: Beatriz Cabrero Daniel, Ricardo Marques, Ludovic Hoyet, Julien Pettré, and Josep Blat (2022). Dynamic Combination of Crowd Steering Policies Based on Context. *Submitted to Eurographics'2022.*

All this was integrated in a framework for crowd simulation whose objective is the analysis and synthesis of crowd motions so that the generated trajectories of agents exhibit desired properties. We present a full framework architecture which, unlike previous works, can operate without any real-world measured data. This is achieved through a random scenario generator coupled with a trajectory quality evaluation i.e. QF. As discussed in Section 2.4, the key to this approach lies in the ability of the quality function to approximate trajectory quality. The use of synthetic data instead of real-world data allows to easily change and extend the variety of training situations, as well as to study in-depth the influence of scenario features on the characters' behaviour.

Limitations of these approaches are discussed and suggestions for future work are given. Moreover, Chapter 5 lists a number of unanswered research questions that we feel are important issues for future research. For instance, to widen the applicability range of QF, additional studies about perceptual realism in different situations will be needed. There is also abundant room for further progress in creating the context-to-policy mapping, especially in the variety of contexts used. Further studies, which take these comments into account, are therefore recommended.

Chapter 2

THE LEARNING CROWDS FRAMEWORK

Crowd simulation algorithms determine the way agents move by resorting to a set of rules or by defining steering functions. While the used parameters values for each algorithm affect the way agents move, the underlying principle ruling the agents' movement for each algorithm is "ad hoc" and remains immutable. Moreover, there is a set of open questions when using microscopic approaches: which is the validity range of the rules and steering functions used? How do they compare to each other? How to set simulation parameters in a principled and robust manner for general scenarios? How can the values of the parameters adapt to the situation the agents are in?

These questions have been addressed in previous work mainly by comparing simulation results against reference data, based on dedicated comparison metrics Guy et al. [2012]; Charalambous et al. [2014b], as well as automatic parameter evaluation techniques Wolinski et al. [2014b]. These approaches have two main limitations: (i) the need for reference data (i.e., real world observations); and (ii) the restriction to parameter tuning only, which prevents exploring new policies out of the commonly used steering functions. To enable the study of these and other questions in a flexible way, while overcoming the limitation of data-based approaches, we built a proof-of-concept machine learning framework based on Genetic Algorithms. A *scenario generator* in the framework can reproduce a large set of simulation cases, where features such as the density of agents in the environment or the number of crowd flow directions can be controlled. The learning process and its *fitness* function, that, permits to evaluate the suitability of the trajectories generated, are discussed in detail in Chapter 3.

2.1 State of the art

The crowd simulation research field is concerned with reproducing, predicting and understanding the motion of real human crowds. Generally speaking, the purpose of crowd simulation is thus to compute the motion of a large number of characters (also called agents) in the same space and time. Various approaches to this problem have been proposed. One class of them are called *microscopic* approaches. Algorithms belonging to this class compute the motion for each character independently, in contrast with *macroscopic* approaches, which consider a crowd as a continuous moving matter [Hughes, 2003; Treuille et al., 2006]. In our work, we are mostly interested in the former kind of approaches.

To palliate the lack of variety and realism in the simulated behaviours, several works take the approach of steering agents using pre-recorded examples of motions [Lee et al., 2007; Lerner et al., 2007; Charalambous and Chrysanthou, 2014b]. These approaches face the challenge of processing a high-dimensional database to efficiently search it, and find similarities between simulation agents' state and recorded motions. Agents do not actually learn how to move, but imitate motions that already existed. As an alternative, Wolinski et al. [2014b] try to get microscopic crowd simulators as close as possible to real crowds of reference, by optimizing simulation parameters. Zhao et al. [2017] steer agents to imitate the specific patterns learned in a given environment. Our work considers applying machine learning to train agents how to move e.g., how to reach their goal while avoiding collisions. A major difference with respect to previous work in this direction is that we do not use real crowd motion data for training. Instead, we propose using a quality function that measures how 'well' agents move and interact.

The idea of learning by experience and interaction is not new; many previous works have already applied reinforcement learning for better agents' steering. Efforts typically deal with finding policies that are state-to-action mappings for one or many agents [E. Hart et al., 1972]. These approaches, however, are hindered by the computational complexity of such systems, due to the curse of dimensionality [Todorov, 2009; Thalmeier et al., 2017]. That is why optimal control approximation and inference, along with sampling techniques, are emerging as interesting alternatives in this field [Kappen and Ruiz, 2016; Todorov, 2006]. Finding the expected reward for each possible next state and using that information to choose which action to take, has already been applied in multi-agent systems to simulate pedestrians by Martinez-Gil et al. [2014]. These approaches, though, usually deal with discrete state spaces, discrete agent actions, or very few agents. We learn general behaviours instead: the policy is defined as a parameter vector that controls how much each state feature affects the motion. By doing this, we learn the best parameter values to optimize the expected simulation quality and generate good trajectories according to the aforementioned fitness metric. Unlike our synthetic data-based approach, Wolinski et al. [2014b] rely on real world data, hence limiting the flexibility and range of application of their proposed framework.

2.2 Overview

The framework we propose is composed of two main building blocks (Figure 2.1): the optimizer, and the simulation sampler. Briefly speaking, learning within our framework is performed in an iterative process, looping between simulation and optimization until the *fitness* of the model parameters found is considered satisfactory. This process is described in Section 2.3.

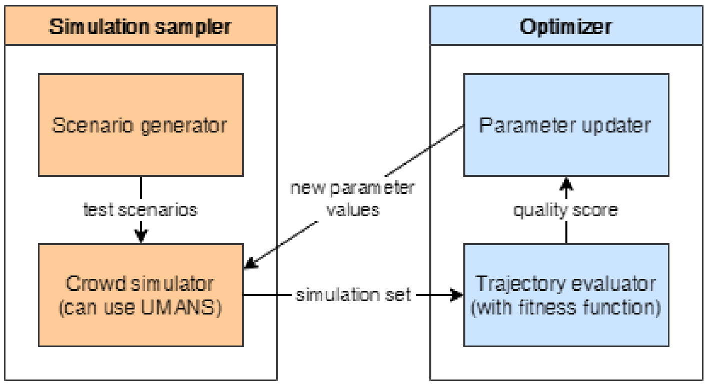


Figure 2.1: Schematic overview of the framework. The simulator sampler (left, orange) tests the parameter values proposed by the optimizer (right, blue) that, in turn, tries to maximise the fitness function score.

On the one hand, the *simulation sampler* receives a set of parameters, θ, and outputs a set of N simulations S $S = \{s_i\}_{i=1}^{N}$, where N is the number of samples used by the learning algorithm, as discussed below. Each simulation, s_i, corresponds to a *crowd trajectory*, the set of all character trajectories of a crowd. Every character has a trajectory in the studied time window that refers to the sequence of 2D positions of one character together with information about its state at every time step. In order to simulate crowds, we can use the original algorithms' implementation (provided by the respective authors) or the *UMANS* software, which is explained in Section 2.4. The scenarios used for this work are generated on the fly in order to test the model parameters in a variety of different situations. On the other hand, the *optimizer* receives a simulation set, S, and studies and returns the average *fitness* over the trajectories in S. Then, it proposes, as discussed in Section 2.3.2, new parameter values to test using the *simulation sampler*.

2.3 Iterative learning of parameter values

We endow the framework with a machine learning module that actively seeks the parameters which maximize motion quality, as defined by the quality function.

2.3.1 Simulation sampling

The simulation sampler (left block in Figure 4.1) receives a set of parameters θ and outputs a set of simulations $S = \{s_1, \ldots, s_n\}$. Each simulation s_i corresponds to the characters' trajectories generated using θ on a given scenario. The simulation sampler has two components: (i) the *scenario generator* generates a set $\Gamma = \{\gamma_1, \ldots, \gamma_n\}$ of n random scenarios on which the parameters are tested; on the other hand, (ii) the *crowd simulator* takes as input the model parameters θ and uses them to perform an end-to-end simulation on each of the scenarios $\gamma_i \in \Gamma$, yielding S.

Scenario generator

We consider a scenario γ_i to be a set of initial positions and goals for each character in a two-dimensional world. Ideally, the simulation vector, S, would contain simulation results using the parameter set θ over the space $\mathcal{G}$ of all possible scenarios. However, such an approach is infeasible. To overcome this problem, we resort to a random scenario generator which draws sample scenarios from $\mathcal{G}$. To guarantee an efficient learning process, care must be taken to ensure that the random scenarios provide appropriate conditions for the agents' learning e.g., the existence of collisions. We refer to the space of scenarios where learning is likely to happen as $\mathcal{G}_l \in \mathcal{G}$. The set of scenarios that is useful greatly depends on the parameters that one intends to learn. For instance, as will be shown later, low density scenarios are not as useful to learn collision avoidance behaviours as densely populated ones. This is because if agents do not experience collisions in the training scenarios, they cannot *learn from experience* the proper values for collision-avoidance related parameters.

The scenario generator aims at randomly generating sets of sample scenarios $\Gamma \in \mathcal{G}_l$, given a fixed number of agents. It assigns an initial position and a goal to each agent. In our implementation, the scenarios generated are square-shaped. Moreover, to avoid introducing border artifacts, we consider a periodic world in which agents leaving the area limits re-enter through the other extremity. The area of the scenarios is a random variable $a \sim \mathcal{U}(a_m, a_M)$, where a_m and a_M correspond to the minimum and maximum scenario area, respectively, that are user provided parameters. The area size a implicitly controls the density of agents, which is a very interesting scenario feature with direct impact on the learned model parameters, as shown by our results.

Crowd simulator

This module can use the implementation of steering algorithms as Velocity Space costs, as discussed in 2.4 or the original algorithm implementation. Each steering algorithm receives a parameter value set, θ, provided by the optimizer. Given a sample scenario $\lambda_i \in \Lambda$ and θ, a steering algorithm produces the corresponding simulation s_i. Each simulation s_i is a sequence of positions and velocities of each agent for a number of time-steps $t = [1, \ldots, T]$, such that $s_i = \{s_i^0, s_i^1, \ldots, s_i^T\}$. To compute each simulation frame s_i^{t+1}, the previous time-step s_i^t and the model parameters θ are passed as argument to the motion model, yielding:

$$s_i^{t+1} = m(s_i^t, \theta)\,, \tag{2.1}$$

where m represents the steering algorithm, implemented as a Velocity Space cost, used. Performing the simulation for a set Λ of n random scenarios sampled by the scenario generator yields the simulation set $S = \{s_1, \ldots, s_n\}$, which is then passed to the evaluator. Any motion model capable of fulfilling this specification is a valid model for being used within our framework.

2.3.2 Parameter optimizer

The optimizer (right block in Figure 4.1) receives a simulation set S, produced by the simulation sampler. First, the *trajectory evaluator* resorts to a quality function, QF, which returns a scalar c representing the average fitness over the trajectories in S. Then, the *parameter updater* proposes a new parameter set to use in the next iteration, based on the performance of θ.

Trajectory evaluator

The evaluator is a key component since it should quantify how "good" a set of parameters proposed by the parameter updater is. For this we specify a simple fitness function, that measures the performance of a parameter set, θ, given by the sum of the quality over all simulations, $s_i \in S$:

$$\mathcal{F}(S) = \sum_i QF(s_i)\,. \tag{2.2}$$

This is performed by resorting to a quality function, QF, whose input are each of the trajectories in the simulation set S produced by the simulation sampler. Chapter 3 moves to discuss the quality function, QF, that is learnt using real data but does not rely on it in order to evaluate previously unseen trajectories. Such a function is thus crucial to the learning framework. It evaluates the values of specific trajectory features in each individual simulation $s_i \in S$.

With this, the problem of finding the optimal parameters, θ^*, which maximise the fitness function, $\mathcal{F}$, can be stated as:

$$\theta^* = \underset{\theta \in \Theta}{\arg\max} \mathcal{F}(S)\,. \tag{2.3}$$

Parameter updater

At each iteration i, the parameter updater proposes a new parameter value set θ_i that will be tested using the *simulation sampler* block. The goal of the parameter updater is to iteratively change the values of θ in order to progressively converge to θ^* that minimises $\mathcal{F}$. In this work, parameters values are mutated using a *Genetic Algorithm*. When a new parameter set is proposed for evaluation, the parameter optimizer uses the quality score given by the *trajectory evaluator* to decide which individuals of the population to keep in order for them to be parents of a new population. This parameter value learning process is discussed in detail in Chapter 3.

2.4 Unified Microscopic Agent Navigation Simulator

The purpose of a local steering algorithm is to compute how a character, c_i, at time t should update its velocity for the upcoming simulation step, s^{t+1}. The general idea is that a character should stay close to its preferred velocity, $\mathbf{v}_{pref} \in \mathcal{R}^2$, while respecting local rules such as collision avoidance with nearby obstacles and other agents. Steering algorithms also receive a set of parameter values, θ, that affect the how the velocity for the subsequent time-step, $\mathbf{v}'$, is selected.

We presented, together with van Toll et al. [2020b], a novel technique to describe local steering algorithms as cost functions that can be minimised in order to select $\mathbf{v}'$. These implementations are gathered in a crowd simulation engine called *Unified Microscopic Agent Navigation Simulator (UMANS)*. This software unifies the implementations of a number of well known microscopic motion algorithms whose implementation is not provided by authors and allows to experiment with them through a common interface.

2.4.1 Steering algorithms as Velocity Space costs

Let the *Velocity Space*, $V \subseteq \mathcal{R}^2$, be the set of all possible velocities a character can have. V can be thought of a disk around a character, c_i, with a radius equal to its maximum speed, s_{max}. We can translate existing steering algorithms into Velocity Space costs, C, that assign to each possible velocity $\mathbf{v}' \in V$ a scalar cost $C(\mathbf{v}')$. A Velocity Space cost can therefore be thought of as the ‘attractiveness’ of choosing $\mathbf{v}'$ as the agent’s next velocity. In this setting, updating the velocity of a character is done by finding which $\mathbf{v}'$ minimises a defined cost.

The cost of a velocity $\mathbf{v}'$ for character c_i can be based on various kinds of information, including its current position, velocity, preferred velocity or goal position; the *distance* between neighbouring characters; the *time to collision* (TTC) assuming that a collision is expected if c_i uses $\mathbf{v}'$ and all neighbours maintain their current velocity; the *distance to collision* (DC), used if a collision is predicted under the same assumptions; the *time to closest approach* (TTCA), that is the same as TTC if a collision is expected; and the *distance at closest approach* (DCA), the distance between them after TTCA seconds. As an example, the Reciprocal Velocity Obstacles (RVO) algorithm uses TTC to evaluate each possible new velocity $\mathbf{v}'$:

$$C(\mathbf{v}') = \frac{w}{TTC} + ||\mathbf{v}_{pref} - \mathbf{v}'||$$

where w is the weight of the cost function, whose value is determined by the *parameter updater*, and TTC depends on the position at time t of character c_i, its radius, and $\mathbf{v}'$. We consider $\mathbf{v}_{pref}$ to be the velocity that would move c_i directly to its goal at a preferred speed $s_{pref} = ||\mathbf{v}_{pref}||$.

It is worth noting that costs, C, are not necessarily smooth and a global optimization might not be able to find an analytical solution. For example, if C uses TTC, the costs of two similar velocities $\mathbf{v}'$ and $\mathbf{v}''$ can be very different if one velocity causes a collision while the other avoids it. Some implementations, therefore, approximate the optimal velocity by sampling multiple ‘candidate’ velocities and choosing the one with the lowest cost.

2.4.2 Crowd simulation loop

We use a simulation loop with frames of a fixed length Δt and N characters simulated simultaneously. At the beginning of the simulation, a spatial hash for character positions is created to facilitate nearest-neighbor computations. Then, for each step, s^t, and character, c_i:

1. All neighbours within r meters of c_i are found, where r is defined in θ. These are the characters that might affect the local navigation of c_i.

2. The preferred velocity, $\mathbf{v}_{pref}$, for c_i is computed.

3. The key step in this loop is computing a new velocity vector $\mathbf{v}'$ for c_i in the next simulation step, s^{t+1}. Each agent can use its own local navigation algorithm, as discussed next.

4. Finally, the velocity v and position p are updated.

This is an independent process that is executed per agent. Therefore, to decrease the simulation's computation time, the work inside these steps can be performed simultaneously for N different characters on parallel threads.

2.4.3 Integration with the framework

Each cost function is a subclass of the abstract class *CostFunction*, and it is used to compute three things: the cost $C(\mathbf{v}')$ for a velocity $\mathbf{v}'$, the gradient $\nabla C(\mathbf{v}')$, and the velocity with minimal cost, v^*. A cost function is accompanied by a set of default or specific parameter values. The cost functions, parameter values and velocity selection method can be specified through XML files. The individual start and goal positions of characters, as well as other internal properties, can also be initialised. This set-up allows to easily test 'variants' of an algorithm in a number of predefined scenarios.

A total of ten different algorithms are described by van Toll et al. [2020b]. The code in *UMANS*, freely available [1], rends similar results to the original implementation (for algorithms whose original implementation is available) or the results described in the original publications [van Toll et al., 2020b].

[1]https://gitlab.inria.fr/OCSR/UMANS

The aforementioned cost functions and optimisation methods are written in platform-independent C++11. This is used by the *crowd simulator* in our framework (see Figure 2.1), as a standalone library controlled through a Python interface. The framework can also use the original code provided by the authors of the steering algorithms, as an external library, whenever it is available.

2.5 Discussion

Previous agent-based techniques have suggested plenty of variables possibly influencing behaviours. With our system, it is possible to determine which model fits better a scenario, or even to simulate higher quality trajectories by combining several approaches. There have been various metrics to evaluate crowds against empirical data Guy et al. [2012]. These measurements are analysed and discussed in Chapter 3, where a perceptually validated quality function is proposed to automatically find parameter values for the steering algorithms.

If characters share their policy (the steering algorithm and parameter values), θ is simply a vector of parameter values. On the other hand, these parameters can constitute a very high dimensional space when different policies for each agent are used for the parameter set θ is then an $m \times N$ matrix where m is the number of parameters of the steering, and N is the number of agents. This distinction, together with more details about the learning process, is discussed in Chapter 3 and a solution to simplify this task is proposed in Chapter 4.

We also find that being able to adapt the policy of every agent dynamically -that is, somewhere in the middle of the simulation- to be able to better steer agents that pass through different scenarios -for instance, different densities- is an interesting improvement with respect to current approaches. For this, equipping the agents with the ability to determine the situation they are in will be very important to switch between policies. This is discussed in Chapter 4. Moreover, being able to group agents, according to the behaviour (policy) or the situation they are in, could be the key to dramatically reduce the computational complexity of the task since policies could be shared among characters.

2.6 Conclusions

We proposed a synthetic data-based machine learning approach to train crowd simulation agents. Using our proposed framework, we automatically find an interesting set of policies (crowd simulation algorithms along with their corresponding learnt parameters) that enable a crowd simulator to meet specific objectives as described by a simulation quality function, described in detail in Chapter 3.

One of the interests of our approach resides on using machine learning without requiring real-world motion-data. Instead, we use a synthetic data-based approach coupled with user-defined simulation quality functions with are used to generate a large variety of training cases and to evaluate the agents behaviour, respectively. We believe that our approach presents many advantages.

An interesting line of work could be using weighted trajectory features alone in order to adapt the motion of agents, reaching an almost model free stage. Examples of these state features could be distance to closest approach, density in the neighbourhood, the context of movement (whether agents are rushing or strolling), etc.

Chapter 3

CROWD TRAJECTORY QUALITY EVALUATION

It has long been established that crowd navigation algorithms and their parameters have a direct impact on the resulting quality of animations. This makes the decision of what model to use and how to pick parameter values difficult. To assist designers in their task, we explore the issue of automatically evaluating the quality of character trajectories in a crowd, and the effect of parameters values. Two paradigms confront each other, each with its advantages and weaknesses:

On the one hand, methods of comparison with real data allow adjusting the parameters to each case, but data is required and the question of comparison criteria arises. Note that the comparison of data also requires objective metrics to measure how distant simulations are from real trajectories. Many metrics have been proposed, but their links with the perceived quality of animations have not been clearly established yet.

On the other hand, perception studies allow to directly judge the quality of animations as perceived by spectators, and therefore perfectly fulfill the objective, but are carried out in long cycles and only allow to estimate parameters that are fixed in advance.

We address the problem of evaluating the quality of crowd simulations by exploring a new method that gathers the advantages of these 2 paradigms. Our approach is to first establish a list of crowd trajectory features which are likely to impact the quality of crowd animations. They are established in discussion with crowd animations experts and concern individual, interaction or global crowd motion features. In a second step, we build a so-called Quality Function QF which maps those features to a single quality score. This function has parameters, the value of which is established from our experts' feedback or estimated from real data. Finally, we perform a perception study with naive users so as to demonstrate that our quality function returns values that correlate with the perceived quality of crowd animations. We demonstrate our approach on the case of *ambient crowds* that can be loosely described as trajectories of heterogeneous pedestrians in the street without any specific behaviour other than walking to their goal (no queuing, no running, no grouping, etc.). For this case, we learn value ranges for each term of QF from real data, we establish a relationship between model parameters and perceived quality, and automatize evaluation to autonomously find parameters for models.

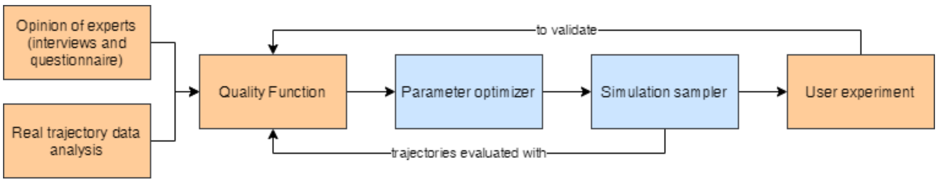

Opinion of experts (interviews and questionnaire)
Real trajectory data analysis
Quality Function
Parameter optimizer
Simulation sampler
User experiment
to validate
trajectories evaluated with

Figure 3.1: Quality function creation overview. The quality function, designed with the help of experts and through the analysis of real data, is validated through a user experiment. The quality function is used in the Learning Crowds framework, represented by the blue boxes.

3.1 State of the art

Crowd simulations result in large sets of individual animation trajectories. Their quality depend on a number of rules by which agents move (simulation models), as well as parameter values to control the simulation. They are not intuitive nor easy to tune and often depend on context.

Our objective is to propose a method to evaluate these simulation results, regardless of the method by which they are generated. We focus on the visual realism of a crowd, i.e. the judgement from spectators of whether a simulation seems real. We can distinguish various approaches to the evaluation of crowd simulations.

The first uses paths of real crowds, to evaluate the ability of simulators to reproduce them. The question of comparison metrics is central, and several adapted solutions have been proposed, for instance Guy et al. [2012], Wolinski et al. [2014a], and Charalambous et al. [2014a]. These metrics consider crowd movement at different scales and take into account the variability of behaviors. However, there are drawbacks associated with the use of reference data, e.g., over-fitting due to the limited sample of pedestrian trajectories available. A broader perspective has been adopted by some authors that, instead of focusing on agent trajectories, measure crowd motion characteristics such as the ratio between the density and the average speed in different cultures like Jelić et al. [2012] and Chattaraj et al. [2009].

To overcome the problems related to the availability of crowd data, some authors study properties that trajectories should exhibit. For instance, Berseth et al. [2016], use a combination of measurements (e.g., path length, failure rate, similarity to ground truth) to automatically tune simulation and/or decide what tuned motion model fits a situation better. Kapadia et al. [2009] propose objective measurable features to compare crowd simulations to real data and detect unrealistic patterns. Another strategy consists in proposing representative scenarios as a benchmark and studying the coverage of different motion models (scenarios they can handle) [Kapadia et al., 2011]. A second category of approach, rather than looking for criteria or data capable of determining the level of realism of a simulation, is to directly evaluate the perceived realism through perception. For instance, McDonnell et al. [2008] and McDonnell et al. [2009] explore the impact of character appearance and motion variations on the perception of crowd heterogeneity, while Turnwald et al. [2015] propose a metric to reproduce the human perception of motion differences. The perception of human motion animation in relation with collisions, that are trajectory events, has also been studied by Hoyet et al. [2016], Molina et al. [2021] and Kulpa et al. [2011]. However, to the best of the authors knowledge, there is no proven link between autonomous quality evaluation methods and human perception of trajectory quality.

Finally, one should notice the recent evolution of simulation techniques towards data-driven models. Recent approaches based on deep learning such as Social-LSTM or Social-GAN and consors, like those by Alahi et al. [2016] or by Gupta et al. [2018], make an implicit evaluation of the generated trajectories (through the loss function, or the discriminator component of a GAN). These methods are rather used to solve trajectory prediction problems, but their use can be adapted to the synthesis of crowd trajectories [Amirian et al., 2019].

Our approach integrates the two first types of approaches for trajectory evaluation, and is designed to preserve the advantages of each: anchoring in objective reference data like data-driven methods, generalising with the help of experts the concept of realism through measurable trajectory features, and linking the evaluations with spectators through a perceptual study. With this approach, evaluations are fast to compute and provide intuitive results, which is very useful to computer animators, while still retaining the information from real data. We demonstrate its usefulness to determine simulation parameter values.

3.2 Overview

The objective of this work is to propose a metric to evaluate the quality of a set of 2D trajectories resulting from crowd simulation. Through the following of this chapter, we call this metric the Quality Function (QF), where by *quality* we mean the *level of perceived realism* of a given trajectory according to general users. However, since determining the relevant trajectory features for estimating such a quality is not straightforward, we propose to rely on a 2-step process. In a first step, we select relevant motion characteristics with the help of experts in the fields of Crowd Simulation and Human Animation. Then, in a second step, the selection is

Table 3.1: Trajectory features discussed with experts (Section 3.3) and used in the quality function (Section 3.4), referred throughout the chapter using the associated 3-letter codes.

Individual features (Code)		
Average walking speed (AWS)	Difference to goal direction (DGD)	Inertia (INE)
Flickering in direction (FDR)	Flickering in speed (FSP)	Goal reaching (GLR)
Difference to comfort speed (DCS)	Angular velocity (AVL)	Trajectory length (LEN)
Interaction features (Code)		
Environment-based density (EDN)	Number of collisions (COL)	Local density (LDN)
Distance to other agents (DTA)	Time to collision (TTC)	Interaction strength (IST)
Time to closest approach (TCA)	Personal space overlap (OVP)	
Interaction anticipation (IAN)	Distance at closest approach (DCA)	
Global features (Code)		
Fundamental diagram (FDG)	Feature values variety (VAR)	Path length (LEN)

validated through a questionnaire that a different set of experts are asked to fill. This feature selection process, together with the features' impact on quality perception according to experts, is detailed in Section 3.3. Once these expert-based relevant features are selected, we measure them in real data and propose a novel quality metric, the Quality Function QF, as described in Section 3.4. In Section 3.5.1, through a perceptual experiment, we show that the proposed metric is able to capture the human perception of trajectory quality. Moreover, Section 3.5.2 illustrates a practical application for automatically determining the parameters of a crowd simulation model by maximising QF. Finally, we present our conclusions in Section 3.7.

3.3 Trajectory features selection

The first step in the development of the QF is to determine what crowd motion characteristics affect the perception of quality. On the one hand, there is an important body of literature providing metrics to evaluate and compare steering algorithms, e.g., absolute difference to ground truth Wolinski et al. [2014a], coverage Berseth et al. [2014], etc. These studies provide insights into what is desirable in a crowd motion. Metrics extracted from related literature, though, are numerous and their definitions often overlap. To only keep trajectory features that have an impact on trajectory quality and are not redundant, we conducted face-to-face interviews with 6 professionals (6+ years of experience) from different leading international animation and crowd simulation companies. The second step is to understand how important each characteristic is and how it can be measured in 2D reference data and synthetic trajectories. To this end, a different set of 9 experts were consulted to validate the features importance and discuss their admissible value ranges.

3.3.1 Face-to-face interviews with experts

To identify trajectory characteristics of crowd motions that are relevant to measure quality, 6 experts were interviewed. Prior to the interview, they received a general explanation of the project. Then, the interviewees selected a number of *trajectory features* to discuss which they believed to affect perceived realism of a simulated crowd, and which can be classified into three groups (see Table 3.1): (i) individual trajectory features, which measure the trajectory of a single agent independently, e.g. difference between walking and comfort speed; in contrast; (ii) interaction features that deal with measures taking into account any pair of agents; the last group is for (iii) global trajectory features aggregate these measurements and study their distribution among the crowd or study the relation between one or more trajectory features. These features were discussed to understand their impact on the perceived quality of crowd motion. Experts were also asked about admissible value ranges for the discussed features and sufficient conditions to consider a crowd trajectory to be of low quality. Some higher-level properties such as coherence in time, the animation layer and scene decorations were also discussed with experts but are not included in this work as they were considered to be outside the scope of “simulating ambient crowds”. Note that some of the proposed features are interrelated, e.g., fundamental diagrams, the relation between a crowd’s flow speed and density, depends on the values of two features.

This is very important for there are behaviours that are undesirable even if some features are individually inside their acceptable value range, but their interrelation in the context of ambient crowds is not.

3.3.2 Questionnaire for experts

Following the interviews, we created an online questionnaire for experts in the fields of crowd simulation and character animation, including questions about all the features that had been deemed to be important. The objective was to obtain qualitative and quantitative information about trajectory features in each group, as well as to ask for additional details, e.g. about their relative importance.

After providing information about their background and occupation, participants then answered questions about the trajectory features listed in Table 3.1. Questions about each trajectory feature were grouped and displayed on the same page, beginning with relevant descriptions and video examples. As can be read in Appendix A, in each page of the questionnaire, experts were asked a number of questions about the related features, providing their answer using a 7-point scale, ranging from "totally disagree" to "totally agree". Experts were asked about the features' importance under different conditions. Note that the formulation of the questions varied slightly to adapt to the feature definitions and units. The goal was to gain a more detailed understanding of how different features and feature values would impact the perceived quality. Moreover, a couple of open answer questions were asked in each page of the questionnaire for experts to give their general opinion about the proposed trajectory features, including their opinion about values, related features, etc. After the feature-specific questions, the experts then answered questions about the general appearance of the crowd, e.g. heterogeneity, as well as to the relative importance of trajectory features. Questions about the impact of trajectory features and of specific values of the features were answered using the following scale: "not important at all", "of very little importance", "of little importance", "of average importance", "important", "very important", and "absolutely essential". A different set of experts (9) partic-

Table 3.2: Reported expertise (from 1 to 5) on Human Animation and Crowd Simulation of the 9 participants.

Participant Field of Expertise	**Reported expertise in Human Animation**	**Reported expertise in Crowd Simulation**
Crowd Simulation Research (3)	{4, 4, 5}	{4, 5, 5}
Video-Game Industry (3)	{2, 4, 5}	{2, 3, 4}
Movie Industry (3)	{3, 4, 4}	{2, 4, 5}

ipated in the survey. Table 3.2 reports their expertise, showing that most respondents worked in the industry (6).

3.3.3 Discussion of expert opinion and of feature selection

Replies were used to validate the selection of trajectory features. Experts overall agreed with the feature definitions and importance. In particular, Figure 3.2 (top) shows on the vertical axis the agreement of experts on a scale from 1 to 7 when questioned about each feature on the horizontal axis. It can be, therefore, said that participants in the survey agreed with the definition of all the selected trajectory features (definitions in Appendix A and in Table 3.4), and that the example videos were representative of the trajectory features in question. This means that the concept of features used for trajectory evaluation was understood and that the mathematical definition of the features is in line with the behaviours and artifacts shown in the videos. Moreover, the survey also validated the use of *ChAOS* [1] to showcase desired undesired trajectory feature values in human trajectories.

Figure 3.2 (bottom) illustrates the experts' opinion on the importance of features. Experts were also asked whether particular values for the trajectory features affected the perceived quality. A minority of experts

[1]https://gitlab.inria.fr/OCSR/chaos

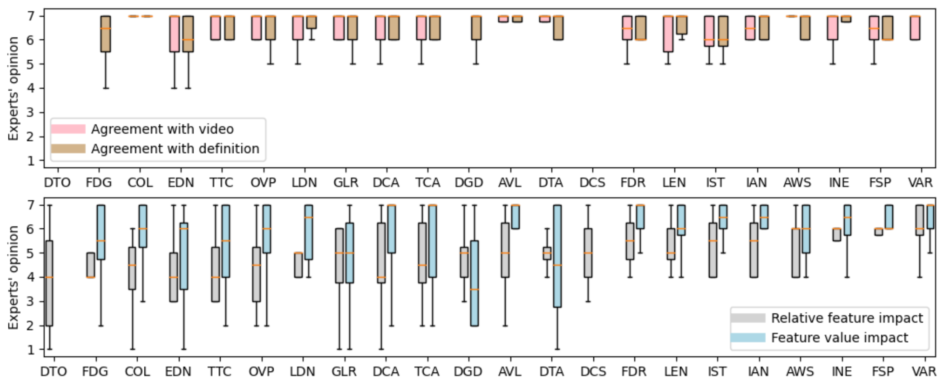


Figure 3.2: Agreement of experts with the statements about feature definitions (top, brown), video examples (top, pink), values (bottom, blue), and importance (bottom, grey). Labels in the horizontal axis correspond to the 3-letter codes in Table 3.1. Boxes represents the central 50% of the answers and whiskers, the remaining 50%. Orange bars represents the median value. The absence of a box means that there was no question in the survey about the definition, video, or importance of that feature.

indicated that some trajectory features were "of little" or "of very little" importance for perceived quality of trajectories. However, all features were on average considered of being at least "of average importance" (values over 4 on the vertical axis), and some "very important". Their opinion on feature value impact (blue) suggests that values outside the admissible range for any of the selected features negatively affect the perceived quality. With this, we conclude that all the features contribute to some degree to the perceptual quality of trajectories, and should therefore be used in the evaluation of crowd trajectory quality.

Answers to the open questions also provided us with valuable information. For instance, E1 highlighted the importance of the available animation system in handling collisions. In his point of view, collisions are "not particularly important if the animations are appropriate for both characters." E2 also reported on this effect by reminding that it is common to "keep agents that intersect slightly because our eye assumes they are not colliding, (as) the human eye often does not register collisions." Such comments are in agreement with previous works on this topic Hoyet et al. [2016] and suggest a difference between intersections (personal space overlap) and actual perceivable collisions (collisions), and as reported by E4 "collisions do happen in real life, so some tolerance is required". However, when such animations are not available, or if the system is not dealing with collisions, "the collision then should be avoided all together, and having a high TTC value would be better" (E1). In this work we focus on trajectories, without making assumptions about the animation system used to visualise characters on such trajectories. Together, these comments highlight the importance of preventing collisions when evaluating trajectories.

However, the importance of avoiding collisions might be mitigated by other factors, as E2 also emphasized the importance of avoiding sharp changes in direction (i.e., high angular velocity values), by commenting that "it is better to collide than to have an abrupt change of direction." E1 also mentioned that it is preferable to avoid sharp changes in velocity, unless "we have captured animations for this and can play those animations appropriately." Similarly, E3 reported that "animation plays a big part" when a collision is imminent (small TTC) and that "sidestepping is perceived as more realistic than turning completely 90 degrees to avoid the collision." Another expert, E5, also commented that "abrupt turns are not found in real trajectories", and that "there is a maximum turning angle to avoid obstacles, the primary reaction is always slowing down." In his point of view "humans have an anticipation of about 30 meters from their surroundings", which could provide an estimate of the distance up to which an agent can be affected by its neighbours.

Experts also commented on the actual trajectory lengths and directions to reach a goal. In particular, according to E2 "limiting travel time to a specific duration makes the resulting simulation look artificial", for it would penalise trajectories going around obstacles and favouring the shortest path and higher walking speeds, which is not necessarily close to that of a real human. E4 also mentioned that "unnecessarily large desired direction differences might work just as well." These comments suggest that trajectory lengths need to be varied enough to avoid impairing realism, and not moving towards the goal at all times might be preferable in some contexts. Of course, travel time is also affected by the walking speed of characters, which experts agreed should be within an admissible range. In particular, they voiced their concerns about displaying walking characters moving very slowly or very fast, as "changing the speed of the animation does affect visual quality" (E1) and "animations cannot be re-timed to change the speed because of motion dynamics" (E2).

Interestingly, experts were unanimous about environment/based density: the distribution of inter-pedestrian distances has an impact on realism since an uniformly distributed crowd is not believable. Many participants voiced their concerns about distances among groups of pedestrians as well. E2 said that people might move in groups and that, depending on the context, "local density among grouping agents might be high, and the closest distance very small." Another expert (E4) suggested that density should be computed as "the ratio of flesh per square meter to account for children", while also highlighting that there are many types of pedestrians for which different state feature values are expected such as "children, disabled people, inebriated people, or people queueing" (E5 adds "menacing people" and "elderlies" to the list). Similarly, E4 pointed out that "fundamental diagram is also a cultural trait", highlighting that cultural differences might also play a role in evaluating trajectory features. Such comments highlight the importance of individual differences, even though we focus (as a first step) on ambient crowds of similar-sized adults without any specific behaviour or cultural traits.

3.4 Quality Function

Our goal is to propose a function for an autonomous quantitative evaluation of crowd trajectory quality. In this context, every character in a crowd has a trajectory in the studied time window, i.e., the length of the reference video or the simulation length, and the term *character trajectory* is used here to refer to the sequence of 2D positions of one character together with information about its state at every time step. The term *crowd trajectory* is then used to refer to the set of all character trajectories of a crowd. The work presented in Section 3.3 provides us with a set of trajectory features, the relevance of which for assessing the believability of a *crowd trajectory* has been validated by experts. These are the basic building blocks with which to construct our quality function. However, to effectively build it, the trajectory features need to be measured, ranked and combined to create the *quality function*.

Constructing QF is a three-fold process. First, metrics for quantitative evaluation for each of the trajectory features are proposed, and real data is studied to obtain reference values for the features. These reference values are treated as a golden set, used to inform the quality function about typical and expected values. With this information, a cost to penalise deviations between the golden set and the features of the evaluated trajectory is proposed. Finally, these independent feature costs are then combined through a weighted sum to determine the quality of the evaluated trajectory.

3.4.1 QF definition

The quality function, QF, is defined as one minus a weighted sum of costs (C_i), where the subscript i indicates the index of the trajectory feature. C_i depends on the distribution of feature values in ground truth data, r_i, and in the trajectory being evaluated, s_i. QF is therefore defined as:

$$QF = 1 - \sum_i \omega_i C_i(s_i|r_i), \tag{3.1}$$

where $i \in [1, \dots, I]$ is the trajectory feature index, I is the number of features (I=21 in our examples), and ω_i is the weight associated with the feature i. The ω_i values are automatically learnt as discussed in the following sections. As identified in Table 3.1, trajectory features can be classified into three categories: individual, interaction and global. Individual and interaction features take different values for each character and time-step of the trajectory while global features take different values only for each time-step of the trajectory. For those trajectory features with values changing for each character, n, and time-step, t, we compute the cost as follows:

$$C_i(s_i|r_i) = \sum_n \sum_t \left(1 - e^{-(s_i^{n,t}-\mu_i)^2/2\sigma_i^2}\right)/(NT), \tag{3.2}$$

where N is the number of characters in the crowd, T is the length of the simulated trajectory, and μ_i and σ_i are the average and standard deviation of the trajectory feature values found in real data r_i. For trajectory features computed across characters at each time-step, t, we compute:

$$C_i(s_i|r_i) = \frac{1}{T} \sum_t \left(1 - e^{-(s_i^{t}-\mu_i)^2/2\sigma_i^2}\right). \tag{3.3}$$

The trajectory feature values for s_i and r_i are determined prior to evaluating trajectories using QF. For this, we assume future linear motion of all the characters in the crowd. We move on to discuss the feature values and the distributions found in reference trajectories.

3.4.2 Trajectory feature values

The values of the features in the evaluated trajectory, s_i, are obtained using the state properties of all the characters. Moreover, the trajectory feature values are compared in the cost functions C_i to a gold distribution, r_i, which we compute in practice from a set of reference data. The dataset used in this work is pending publication; meanwhile, it is available on demand. Following Eq. (3.2), we are interested in the average value, μ_i, and standard deviation, σ_i, for each of the trajectory features in real trajectories.

Character properties can be classified in: static, individual and state properties. Properties like agent radius and mass are static: unchanging and shared by all agents. Individual properties, such as goal position or preferred speed, are different for each character in the crowd but remain constant throughout the trajectory. Lastly, state properties refer to the properties of a character that change at every time-step of a trajectory,

Table 3.3: Common terms for the equations in Table 3.4.

Term	Definition
s_i	Values of feature i in the studied trajectory
r_i	Values of feature i in reference data
T	Total number of time steps in the studied time window
t	Current time step
N	Total number of characters in the crowd
n	Index for the current characters

e.g. current speed, direction, position, etc. Static, individual and state motion properties are used to compute the values of *trajectory features*. Each feature value set s_i will be the result of measuring the values of a particular trajectory feature, f_i, at each time-step, t, and for each character in the crowd, n, if appropriate:

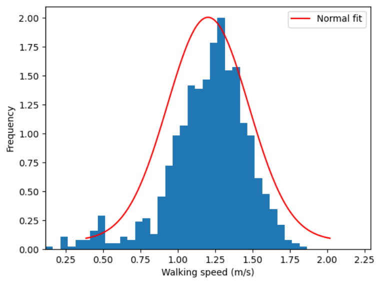
$$s_i = \{f_i(n, t)\}_{n \in [0 \cdots N], t} \tag{3.4}$$

Note that the values of s_i change at every time step as characters move and interact. The time window is not specified in Eq. (3.2) and greatly depends on the application as will be discussed in Section 3.5.1. A detailed account of how the values for each trajectory feature are computed is given by the mathematical definitions provided in Table 3.4.

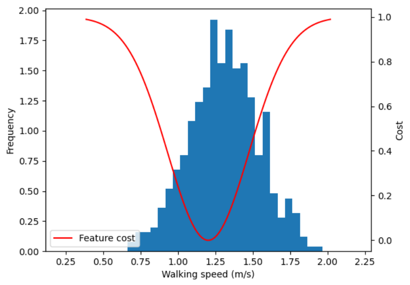
Walking speed is an important term of the quality metric and is used here to illustrate the analysis of feature value distribution. The velocity of the characters of the crowd, the output of the motion model, is observable at every time-step of the simulation, and we can therefore use the values of the walking speed for all characters for all the time-steps of the trajectory. Figure 3.3a shows the distribution of walking speed values in reference data (blue bars), as well as the normal distribution fitted to this data (red line); the corresponding mean (μ_{ws}) and standard deviation (σ_{ws}) of this distribution will be used to compute the cost for walking speed $C(s_{ws}|r_{ws})$. Figure 3.3b shows the distribution of walking speed values found in a sampled synthetic crowd trajectory (blue bars),

Table 3.4: Measurements for the features. Find in brackets where the definition for the terms is.

Feature	Definition and terms	Equation
AWS	Average walking speed of agent n, where $w_{n,t}$ is the current walking speed at time t and N is the number of characters in the crowd.	$\mathcal{S}_{AWS} = \left\{ \sum_{t=0}^{T} \frac{w_{n,t}}{T} \right\}_n$
DCS DGD	Difference between walking speed, $w_{n,t}$, and comfort speed, w_n^*, of agent n at time t. Same equation for the difference between current direction, $\alpha_{n,t}$, and goal direction, α_n^*.	$\mathcal{S}_{DCS} = \{w_n^* - w_{n,t}\}_{n,t}$
FSP FDR	A time-invariant Gaussian filter G is used to obtain $w_{n,t}^G$ and $\alpha_{n,t}^G$, and then subtracted to the original high frequency speed and direction sequence, respectively.	$\mathcal{S}_{FSP} = \left\{w_{n,t} - w_{n,t}^G\right\}_{n,t}$
AVL	The smoothed sequence of direction of movements for each character d_n^G is also used to compute the total angular velocity feature value.	$\mathcal{S}_{AVL} = \left\{w_{t+1}^G - w_t^G\right\}_{n,t}$
LDN	Number of characters $c_{n,t}$ inside an circle around each character n at time t, divided by its area a.	$\mathcal{S}_{LDN} = \left\{ \frac{c_{n,t}}{a} \right\}_{n,t}$
EDN	Average pairwise distance, $d_{n,j}$, between character n and all other characters at time t.	$\mathcal{S}_{EDN} = \left\{ \sum_{j=1}^{N} \frac{d_{n,j}}{(N-1)} \right\}_{n,t}$
DTA DTO	Minimum distance between character n and all other characters, $d_{n,j}$, or obstacles, $d_{n,o}$.	$\mathcal{S}_{DTA} = \left\{ \min\left(\{d_{n,j}\}_j \right) \right\}_{n,t}$
INE	We compute the momentum of the linear system using the velocity, $w_{n,t}$, and mass, m_n.	$\mathcal{S}_{INE} = \{w_{n,t} * m_n\}_{n,t}$
IST	Product between deviation because of neighbour j and deviation from goal direction d^*.	$\mathcal{S}_{IST} = a\left(\alpha_n, \alpha_{n,j}\right) * a(\alpha_n, \alpha_n^*)$
TCA	Assuming linear motion, we compute relative velocity $v_{n,j}$ and position $p_{n,j}$ for each neighbour of character n. $\mathcal{S}_{TCA}$ is the minimum TTCA value for each character and frame.	$\mathcal{S}_{TCA} = \left\{ min\left(-\frac{p_{n,j} * v_{n,j}}{\|v_{n,j}\|^2} \right)_j \right\}_{n,t}$
DCA TTC	Using TTCA, Distance at Closest Approach (DCA) is computed assuming linear movement. Time to collision (TTC) is defined by the value of TTCA when DCA is under a threshold.	$\mathcal{S}_{DCA} = \{p_n + ttca_n * v_n\}_{n,t}$
IAN	For future collisions (small DCA), product between TTCA and difference to desired direction α^*.	$\mathcal{S}_{IAN} = \{ttca_n * (\alpha_n^* - \alpha_{n,t})\}_{n,t}$
COL	Number of times the distance $d_{n,j}$ between n and j is smaller than the sum of the radii, r.	$\mathcal{S}_{COL} = \left\{ \sum_j \left[d_{n,j} < r_n + r_j\right] \right\}_{n,t}$
OVP	Total overlapping area between characters. Personal space radii depend on the application.	$\mathcal{S}_{OVP} = \left\{ \sum_j ovp(n,j) \right\}_{n,t}$
LEN	Length of the path, l_n, divided by the length of the line between the origin, $\hat{p}_0$, and goal, $\hat{p}_g$.	$\mathcal{S}_{LEN} = \left\{ \frac{l_n}{\|\hat{p}_g - \hat{p}_0\|} \right\}_n$
FDG	Relation between density, den_t, and speed, $w_{n,t}$, in the trajectory and in real trajectories. $fd()$ refers to the value of the fundamental diagram in real data.	$\mathcal{S}_{FDG} = fd(den_t) - \frac{\sum_t \sum_n w_{n,t}}{NTden_t}$
GLR	Percentage of characters, n, that reach their goal, $\hat{p}_{n,g}$, in the studied time window [t, T].	$\mathcal{S}_{GLR} = \sum_n \left[\hat{p}_{n,g} = \hat{p}_{n,T}\right]$
VAR	Sum of differences between the variance for feature i in reference, σ_i, and synthetic data, σ_{s_i}.	$\mathcal{S}_{VAR} = \sum_i \|\sigma_i - \sigma_{s_i}\|$



(a) Real data (reference).



(b) Synthetic data.

Figure 3.3: Blue: walking speed distribution in reference data (a) and in a synthetic trajectory (b). Red: normal distribution fitted to the reference data (a), and the trajectory cost assigned to each value (b).

as well as the cost assigned to each feature value according to Eq. (3.2) (red curve). Intuitively, the larger the difference between the feature distribution in the simulation and in the reference data, the higher the cost in Eq. (3.2).

Analysis of the feature values in reference data r_i revealed distributions close to that of a normal distribution (example in Figure 3.3), such that $r_i = \mathcal{N}(\mu_i, \sigma_i)$. In some cases the feature values exhibit a multi-modal shape which would be better explained with a Gaussian mixture, but upon inspection these trajectories were identified to be outside the scope of this project e.g., trajectories with moving and standing agents.

3.4.3 Quality function weights

The weight ω_i of each cost in Eq. (3.1) is related to the influence of each trajectory feature on the quality of crowd trajectories. These weights are learnt using an evolutionary approach that leads to a single set of weights used in QF. Different learning strategies can be used to give values to the QF weights, but Genetic Algorithms (GA) are not prone to be stuck in local minima and provided stable results in our experiments. GA are able to explore new weight value sets and exploit the solutions found in previous iterations through mutation and crossover strategies.

In this work, the performance of characters is measured on a given training set. The training set consists of two types of trajectories: (i) reference data, that receive a score equal to 1, and (ii) unrealistic trajectories, according to experts, that are given a low score. For instance, trajectories generated without avoidance maneuvers are studied and the corresponding feature values are used as predictors and assigned 0 as target value in the training set. Moreover, undesired feature value combinations (discussed in Section 3.3) are also given a 0 target score. In this step we use different scenarios: crossing flows, circle crossing and random. The consistency of the learnt weights across scenarios is discussed in Section 3.5.1.

The average difference between the given quality score, S_R, and the prediction, S_{QF}, is used as the fitness of each individual of the population. For information, the weights learnt for each trajectory feature are reported in Table 3.5; where we see that some features have a more direct impact on the overall evaluation. Nevertheless, all features contribute to the quality score in a significant way, allowing it to detect artifacts that simpler metrics such as the Least Effort Function Guy et al. [2010] could not e.g. unrealistic combinations of speed and density or the apparition of flickering in some interactions.

The quality score S_{QF} obtained with these measurements can be compared to existing metrics in the literature. Since QF shares some of the evaluation principles, the trajectory features, with these metrics, we expect a certain correlation between the values. Nevertheless, QF aggregates multiple numerical values that, according to experts agreed are important to evaluate perceived realism. We can compare, for instance, QF to the Effort Measure Guy et al. [2010] and we obtain a Pearson correlation of around $r = 0.3489$. This is due to QF being based, among other things, on the deviation to the desired velocity of movement through DCS and DGD. Figure 3.4 represents the evaluation of both metrics together with a trend line, which clearly shows the relation between the values. Nevertheless, this figure proves that many trajectories are over

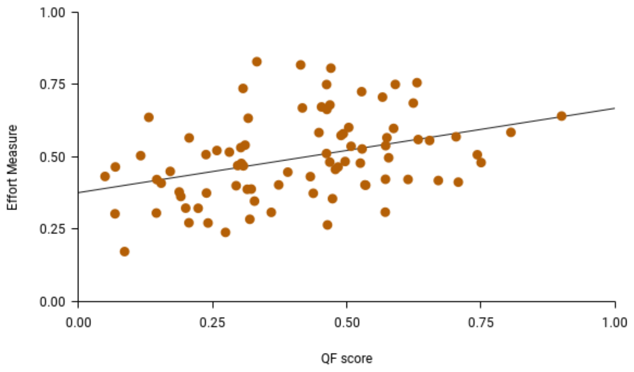


Figure 3.4: Scores (brown), for the same collision-free trajectories, by QF and the Effort Measure (normalised). The trend line (black) shows the relation between the two.

or underrated by the Effort Measure, while QF bases the evaluation on many other feature values. Points under the trend line represent trajectories where QF detects artifacts in some trajectory feature that is not used to compute the Effort. Similarly, points over the trend line represent trajectories where the deviation from desired velocity is due to global feature values e.g. high densities. A discussion on the relationship between these metrics using specific examples can be found in Section 3.5.2.

Correlations between features were also studied prior to learning the weights, to avoid highly correlated features to impair learning, but that none had a correlation coefficient greater than 0.8. All the features were therefore included to be potential predictor variables. We would like to point out that the Distance To Obstacles (DTO) feature was selected by experts (Table 3.1) but was not used in our experiments because relevant information was not available in our reference dataset (our reference trajectories did not include obstacles other than neighbouring characters). Nevertheless, weights for such features can be learned in the future if a different dataset including the relevant information is made available.

Table 3.5: Weights for Eq. (3.2) costs, that depend on the feature values in synthetic and reference data.

Trajectory feature weights					
Individual		**Local**		**Global**	
AWS	0.1995	DTA	0.0586	FDG	0.0515
DCS	0.0258	EDN	0.0087	LEN	0.0587
FDR	0.0590	IST	0.0163	VAR	0.0224
DGD	0.0072	TTC	0.0441		
GLR	0.0074	COL	0.0949		
FSP	0.1054	IAN	0.0085		
INE	0.0275	TCA	0.0698		
AVL	0.0800	DCA	0.0068		
LDN	0.0381	OVP	0.0096		

3.4.4 Consistency check with real data

Once the weights are learnt, crowd trajectories are given a quality score using Eq. (3.1). The quality score obtained in real trajectories is studied using cross-validation. This test confirms that previously unseen real trajectories receive scores higher than synthetic data, $S_{QF} \sim \mathcal{N}(0.9160,\, 0.0817)$. Whilst this alone is not enough to confirm the correctness of the quality function, it partially substantiates its usefulness. The following parts of this chapter, Section 3.5.1 and Section 3.5.2, describe in greater detail the validation and applications of the proposed quality function.

3.5 Validation

3.5.1 User study

To the best of the authors knowledge, none of the existing data-driven metrics for trajectory quality is proven to accurately characterize human perception of trajectories. In the previous sections, a new quality metric QF was proposed, its weights were learnt using reference data and used to assign scores to new trajectories. In this section, we are interested in whether the obtained score correlates with human evaluations of trajectory quality. To answer this question, we conducted an online experiment, where participants were shown pairs of videos and asked to select the more realistic one.

Experiment design

Sixty-six participants took part in our perceptual experiment. All participants (26F/39M/1N; age: 30 ± 9, min=16, max=61) were non-experts in crowd simulation or related fields, and were naive to the purpose of the study. Participants were only informed at the beginning of the experiment that it was about trajectory realism and that data would be treated anonymously. The experiment was conducted through an ad hoc responsive website and the replies were collected on a private server.

Figure 3.5 illustrates the interface shown to users participating in the experiment. Two simulations of the same context (density, scenario, etc.) are shown side by side. Users are asked which looks more realistic. Once a user selects one of the two options, the "Next" button becomes clickable, forcing the participants to choose.

The experiment consisted in showing to participants a number of pairs of videos, presented side by side. The focus of this work is the "realism" of trajectories, not of the animation of characters. To help participants focus on the trajectories, animations therefore had to be believable without masking the underlying trajectory artifacts. Following recommendations on variety in crowds McDonnell et al. [2009], trajectories were animated with a set of 20 male and 20 female characters, with 6 textural variations per character.

There are factors that can influence the perception of quality. For instance, different motion models generate different types of trajectories, with different motion characteristics. Therefore, in order to make the experiment robust, we selected three representative models to generate the experiment videos: (i) a representative of force-based models, Social

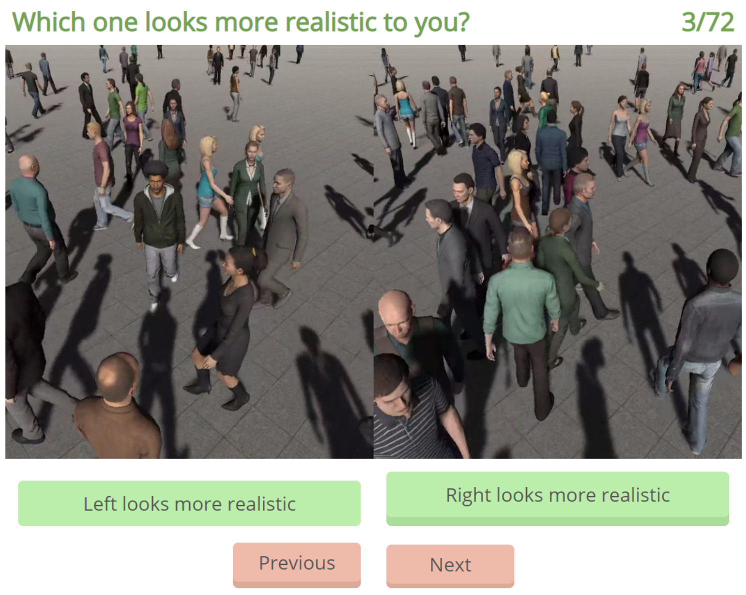

Which one looks more realistic to you?
3/72
Left looks more realistic
Right looks more realistic
Previous
Next

Figure 3.5: User experiment interface with "Left" selected.

Forces; (ii) a representative of velocity-based approaches, RVO; (iii) and a linear prediction-based model, Dutra-Marques. Density can also impact the perception of artifacts so crowds were generated at two density levels: high (more than 3p/m^2) and low (less than 1p/m^2). To study a wider range of interactions, three generic scenarios were also selected: a random one, and two crossing flows at 90º and 135º, respectively (see Figure 3.6). Furthermore, previous work has focused on determining the influence of the point of view on the perception of realism; to ensure the camera position did not affect the results, we therefore chose two points of view (see Figure 3.6): a canonical eye-level camera and a top view.

Trajectories used in this experiment obtain QF scores in the [0, 0.9) range. To study the perception of trajectory visual quality, the videos were classified into quartiles (Q1=lowest quality quartile, Q4=highest quality quartile). As we were interested in evaluating perceptual differences between the quartiles, we included in our experiment comparisons between all quartiles (6 combinations). All combinations between these factors

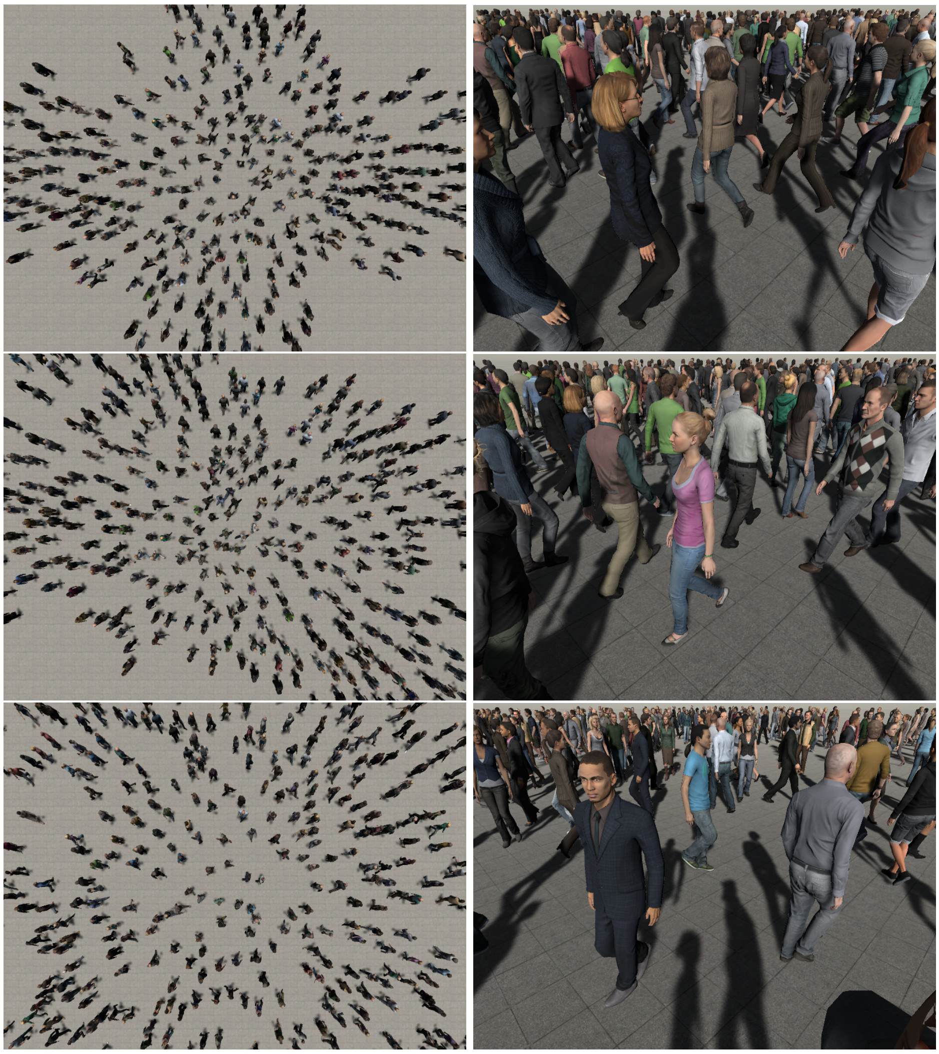


Figure 3.6: Scenarios and point of views used in our experiment. From top to bottom: 2-flows 90º crossing, 2-flows 135º crossing and random scenarios, displayed from top (left column) and eye level (right column) point of views.

were taken into account to create the stimuli, resulting in 288 videos (5 seconds, 960×1080 pixels) created using *ChAOS*: 4 quality levels $\times$ 3 scenarios $\times$ 3 motion models $\times$ 2 densities $\times$ 2 POV $\times$ 2 repetitions.

The experiment was divided into three blocks (one for each scenario). Each participant performed a single block of the experiment, and was shown a random subset of 72 video pairs for this specific scenario (a total of 144 good and bad quality trajectories). Each pair of videos showed crowd motions with different quality scores Qi and Qj, but generated with the same motion model, scenario, density level, and point of view. In the following, such a comparison is referred to as QiQj. The left/right position of the videos on the screen was randomly assigned for each trial. There was no time limit for completion of the experiment, and the critical time of this experiment, i.e. the minimum time needed for users to watch each pair at least once, is 12 minutes. After watching each video in a pair, participants were asked “Which one looks more realistic to you?”, and answered by clicking with the mouse on a button to select a video of the pair. The mouse position was reset to the center of the screen between each trial by pressing a ’Next’ button, to avoid any left/right selection bias.

Once a participant completed the experiment, a table was stored on the server-side containing, for each trial, the ids of the videos shown, the participant answer, as well as the side of the screen the selected video was shown on (left/right). For each participant, the time for experiment completion was also stored, together with the age and gender, asked at the end of the experiment. Replies were then aggregated to study correlations between the participant choices and the quality function score. Out of 66 participants, 24 performed the random scenario, 22 performed the 90° crossing flows scenario, and 20 performed the 135° crossing flows scenario.

Statistical analysis

To analyse participants’ results, we first determine True Positive (TP) answers, i.e., participant selects the video with the highest QF score. With this information, **user accuracy** is computed as the percentage of TP for each user, and used to evaluate the level of agreement of users with QF.

To check for statistical differences in user accuracy, we performed on each within-subject factor (QiQj Comparisons, Motion Models, Densities, POV) a separate 2-way mixed-design repeated measures Analysis of Variance (ANOVA) with between-subject factor Scenario. A full factorial analysis was however not possible as participants were presented with a random subset of 72 video-pair comparisons. Normality was assessed using a Kolmogorov-Smirnov test. All effects were reported at p¡0.05. When we found effects, we further explored the cause of these effects using Bonferroni post-hoc tests for pairwise comparisons. As we did not find any interaction effect of Scenario and Motion Model on user accuracy, these are omitted in the following of the section.

QiQj Comparisons. We first looked at statistical differences between the QiQj comparisons, to evaluate effects of quality comparisons on user accuracy. Results showed a main effect of QiQj ($F_{5,300} = 10.551$, $p \approx 0$). Actual accuracy values per QiQj comparisons are reported in Table 3.6. Post-hoc analysis showed that all QiQj comparisons were perceived on average with a similar accuracy, except for Q1Q2 for which participants demonstrated a significantly lower accuracy. This result suggests that participants had more difficulties in differentiating between videos of the Q1 and Q2 quartiles, but that they were more accurate for comparisons between other quartiles, reaching a 75% accuracy for comparisons between Q1 and Q4. Single t-tests on accuracy were also significantly higher than 50% for each QiQj comparison (all $p < 0.05$), showing that user accuracy was in all cases significantly above chance level.

To better understand the relationship between user preference and quality, we averaged accuracy over trials with the same quartile difference. We organised the data in three Quality Difference levels: QD1 (over Q1Q2, Q2Q3, Q3Q4), QD2 (over Q1Q3, Q2Q4) and QD3 (Q1Q4). A repeated measures Analysis of Variance (ANOVA) with within-subject factor QD showed a main effect of QD on user accuracy ($F_{2,120} = 25.253$, $p \approx 0$). As quality differences are directly related to our QF scores,

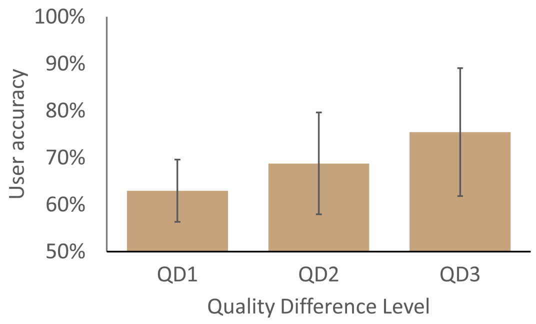


Figure 3.7: Effect of quality difference on user accuracy. Error bars depict the standard deviation.

Table 3.6: User accuracy detailed per quartile-to-quartile comparisons. The quality bins in the rows are preferred over the quality bins in the columns.

	Q1	**Q2**	**Q3**	**Q4**
Q1				
Q2	55±20%			
Q3	68±16%	63±14%		
Q4	75±15%	68±15%	67±17%	

this result demonstrates that users were more likely to correctly identify the highest QF quality videos when the QF quality difference between videos increased (Figure 3.7), and therefore to agree with the QF scores.

Discussion

The results of the user evaluation demonstrated that viewers consider, on average, videos with a higher QF score to be more realistic than videos with a lower QF score, therefore showing their general agreement with the quality function. This agreement can be observed in the QiQj comparisons (Figure 3.7, left), where accuracy was significantly above chance level in all cases, and up to 75% accuracy on the higher quality differences. We also observed that accuracy significantly increased depending on the difference in QF scores between the presented videos. Overall, these results validate that the output of our data-driven QF and the perception of human trajectory quality are in agreement.

However, the user evaluation also demonstrated that other factors can influence the perception of the visual realism of trajectories. Such factors include elements that are not captured by QF and are depending on application choices, such as the point of view. For instance, an eye-level point of view seems to enable users to more accurately perceive visual artifacts in character trajectories, while a top-down viewpoint might render perceiving such artifacts more difficult. We believe this is because users are able to assess the quality of trajectories from a more "natural" point of view, closer to the one we are used to experience as humans. The top view, on the other hand, somewhat masks the artifacts in the trajectories, making it more difficult to evaluate the crowd motion. Such results are therefore complementary to the work of [Kulpa et al., 2011], who showed that collisions are more easily spotted by viewers from a top point of view. Nevertheless, other factors also seem to influence the perception of trajectory features. For instance, the results show that users more easily perceived quality differences for higher density scenarios. Such a difference could be due to several possibilities, such as higher numbers of collisions in higher density scenarios, possibly leading to more sudden direction and velocity changes in low quality trajectories, etc., and should therefore be explored in future studies. Similarly, the range of tested scenarios did not allow us to evaluate the effect of all the trajectory features, since the study was limited to ambient crowds. For instance, the goal to reach was not displayed, which is however a feature influencing S_{QF} score. It is thus unknown whether or to what extend such features affect the perception of trajectory quality.

3.5.2 Applications

In this section, we illustrate one of the possible applications of the proposed metric: tuning parameters of crowd motion models without direct comparisons to real trajectories, i.e., using the pre-trained QF. The goal is therefore to find a parameter set, p^{opt}, of a crowd motion model that maximises the QF score for a generated trajectory or for a set of trajectories. Resorting to QF (which can be seen as a fit-function) instead of directly fitting the model to real trajectories has several advantages. First, in this work, the information in the reference data is transformed into abstract trajectory features selected by experts. This makes the learning process focus only on properties that affect the perception of crowd motion quality instead of simply trying to replicate real trajectories which are, in fact, difficult to gather. Second, over-fitting is a recurrent problem when learning crowd motion model parameters. Models usually describe a limited set of interactions and learning the parameters to replicate real data usually leads to ad hoc solutions. In this work, we propose an abstraction from real data so that the algorithm does not replicate actual trajectories, but instead mimics the values of several abstract properties.

Parameter tuning

QF depends on a number of crowd property values which are initialized using real data (see Section 3.4.3). Theoretically, parameters for any microscopic crowd motion model can be learnt using QF, by evaluating the feature values in the simulated trajectories. The learning algorithm would then attempt to maximise QF by changing the model parameters. In this work, we propose a learning process with a decreasing exploration rate to find parameters, in order to explore a wide parameter domain while still ensuring convergence. Algorithm 1 summarises the learning strategy used to find parameters for the motion model.

Other global optimization techniques can be applied to maximise QF for any trajectory. Finding the most adequate method is, in itself, an interesting topic but out of the scope of this work. It is important to note that, when using different parameter value sets for each agent in the crowd,

Algorithm 1: Parameter tuning

```
Result: parameter set, $p^{opt}$
create initial generation of parameters for chosen crowd motion
  model;
while fitness criteria do
    simulate N crowds with each individual in the generation;
    compute average $S_{QF}$ of trajectories generated with each
      individual;
    select and crossover the fittest individuals;
    mutate children and create new generation of parameters;
end
select fittest individual from last population;
```

the complexity of the optimization explodes due to the large number of parameters to learn, because of the typically large number of characters.

Learning process

There are two possible goals of the learning process: (i) finding p^{opt} for a particular initialization in a given scenario; (ii) finding a generic p^{opt} which can be used for any ambient crowd scenario. In the former goal, a single initialisation (specifying a scenario, a simulation algorithm and the internal characteristics of characters, e.g., initial position, comfort speed, goal direction) is used for tuning the model using QF. In this case, the resulting p^{opt} might not be useful with another crowd initialisation. In the latter goal, the objective is to find a scenario-independent p^{opt} for a selected simulation algorithm, instead of focusing on a particular initialization. During training, multiple crowds are simulated at every iteration to evaluate the performance of each parameter set. The evolutionary process, represented in Figure 3.8, ends when the stop criteria are met. In this case, the genetic algorithm stops iterating if it has reached a user-defined maximum number of iterations, if the fitness increase is below a threshold for several iterations, or if the quality reaches the maximum score.

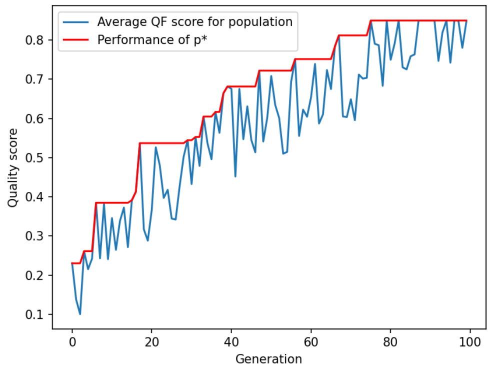


Figure 3.8: Evolution of the average simulation quality S_{QF} with the number of generations. For each generation, a number of trajectories, s_i, are simulated and the average QFscore is computed using Equation 3.1.

Results

Figure 3.9 shows an example of the learning process at different stages using the Dutra-Marques model in a circular scenario (characters initially positioned on a circle and assigned the goal of reaching the diametrically opposite position). In this example, characters share parameter values. Four steps of the learning process are shown (one per column), with increasing quality scores from left to right (each column illustrates a quality quartile Q1-Q4, as in Section 3.5.1). The resulting trajectories, all of the same time length, are shown in the top row, while radar-plots showing the difference to ideal feature values are shown below. At the beginning of the training, the pool of parameters is unlikely to contain parameter values leading to good quality simulations. In Figure 3.9 (top, left), parameters related to goal attraction are not good to guide agents towards their goal, on the other side of the circle. The corresponding radar plot shows that the trajectory features are very different to those found in real data which results in a low QF score for this trajectory ($S_{QF} = 0.22$). QF is specially affected by the number of local features like overlaps and interaction strength (see Table 3.5). As training progresses and the pool of

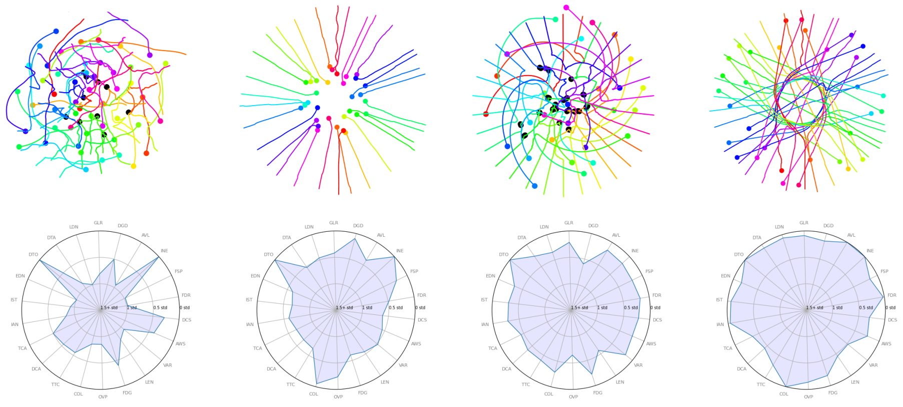


Figure 3.9: Relation between the values of trajectory features in four sample trajectories (from the learning process) and the expected values (relative to those found in real data) expressed in standard deviations. Note that the distance to obstacles features (DTO) feature is always set to 0 for these example trajectories do not contain obstacles other than agents.

parameters evolves, the resulting simulations improve with respect to QF and characters progressively learning to balance between avoiding collisions, attempting to reach their goal, and satisfying the other conditions, e.g., acceptable walking speed, jerkiness. As a result, the right-most plot ($S_{QF} = 0.79$) shows a trajectory created with parameters that balance these implicit rules.

Four examples of circle crossing trajectories with different quality scores are represented in the top row of Figure 3.10. The collisions in the two top, left plots are not represented to facilitate the understanding of the figure but, as can be seen in the corresponding radar plots, collisions are present in the trajectories, affecting the quality score. These four trajectories are, in fact, taken from a parameter tuning process for the Social Forces model. In the beginning of the learning, parameters values are chosen randomly, without any prior information. The first plot, therefore, shows a trajectory with low quality, for characters reach a “deadlock” and slow down for a long while at the center of the circle. This problem is reflected on the corresponding radar plot (bottom) where we can see that trajectory features do not resemble the reference values. In this example, metrics such as the effort measurement proposed by Guy et al. Guy et al. [2010], would also determine this is not a high quality trajectory for the velocity of the characters deviate from their desired one. Berseth et al. Berseth et al. [2014] also propose a number of metrics to evaluate the comparative performance of tuned models such as “failure rate”, “distance quality” or “time quality”. These measures are directly related to one or more trajectory features used in QF. The parameter values are progressively refined in order to improve the quality score of the resulting trajectories. The second column shows an example of this: characters do not stop at the center of the crossing but collisions keep happening and characters create unrealistic paths instead of taking a shorter, smoother route. After a number of iterations, parameter values improve until a solution like the one found in the last column is reached. In this example, all characters reach their goal positions at the other side of the circle and have smoother trajectories. In this case, the effort metric by Berseth et al. Berseth et al. [2014] might give this trajectory a good quality score. Nevertheless, QF receives information about features such as the interaction anticipation time, distance between characters, variability, etc. and gives this trajectory a quality score of $S_{QF} = 0.75$ (lowest $Q4$ score). It is important to note that, the chosen learning algorithm chooses a random pool of parameters as the original parents. For some models, the first pool of parameters might only contain parameter values that lead to quality scores of the second, third and fourth quartiles, $S_{QF} >= 0.25$.

Figure 3.11 shows Reciprocal Velocity Obstacle (RVO) trajectories. In this example, the plots represent a square region of an infinite world where the number of agents is always the same. Characters are initialised belonging to one of two flows that cross: from bottom to top and from left to right. The four columns in this figure are samples taken at different stages of the RVO parameter tuning process. In the left-most figure, we can see one of the first iterations where agents move in a straight line, not avoiding their neighbours. Collisions (black dots) are penalised so this trajectory obtains a very low score. The second column represents a Q2 trajectory. In the top row we can see that an important percentage

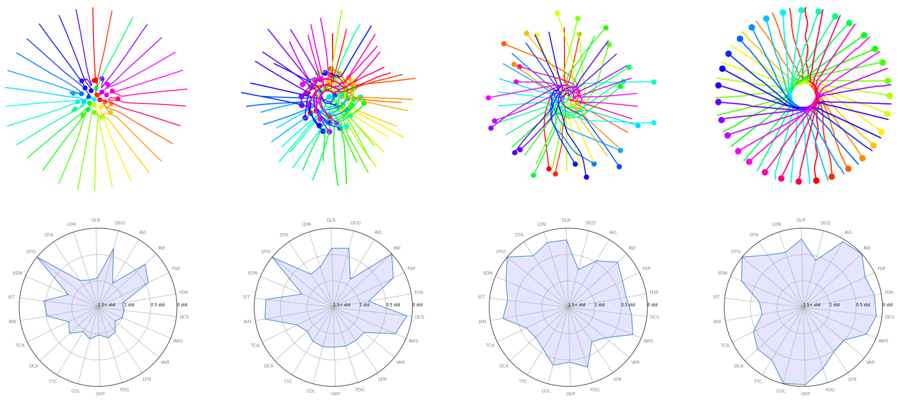


Figure 3.10: Illustration of different trajectories generated with the Social Forces model. Each column corresponds to a quality level and the radar plot in the bottom row show the relationship between the feature values in the trajectories and the reference data used to train QF.

of characters move diagonally as a result of non satisfactory avoidance maneuvers. Since not moving in the desired direction lowers the overall quality, the parameters are changed with generations to avoid this artifact. As a result, trajectories like the one in the third column are preferred. Nevertheless, some residual problems are still present in the simulated crowd: shaky trajectories for some agents (bottom right corner) that are penalised by QF. At the end of the learning process, a parameter set for RVO is chosen, the resulting trajectories receive high quality scores (greater than 0.75). The last column is a sample generated with the learnt parameters. We can see that no collisions are present, agents generally move in their goal direction, there are no shaky trails due to avoidance maneuvers, and all characters move approximately at their comfort speed.

We can also track, as represented in Figures 3.12 and 3.13, independent feature costs. In this case, we study the deviation between the difference to comfort speed (DCS) and the number of collisions (COL). As we can see, in the beginning of the learning process, the the walking speed, $w_{n,t}$, and comfort speed, w_n^*, are the same but collisions are present in the crowd trajectories. Motion parameters are iteratively improved until good

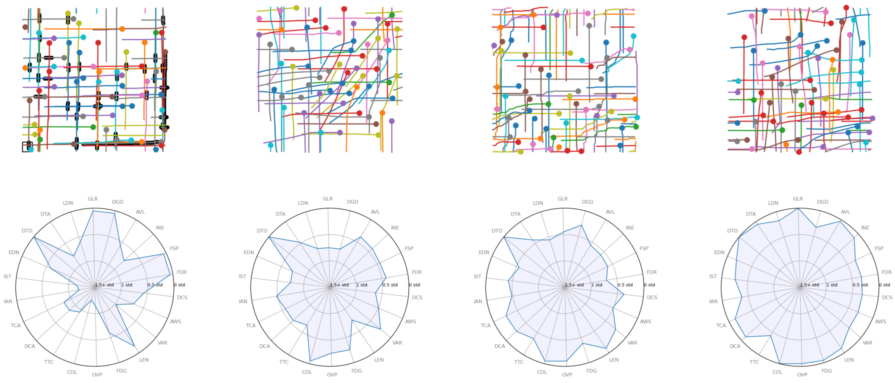


Figure 3.11: Sample 10 second trajectories generated with the RVO model for a crossing flows scenario (2 flows, 90 degrees) in a infinite world (only showing 10x10m region).

trajectories are simulated, according to QF. In the left-most plots, both in Figure 3.12 and Figure 3.13, we can see that no collisions are present (they are heavily penalised in QF and that the the difference to comfort speed is not very high (characters avoid each other by slightly changing their direction and walking speed).

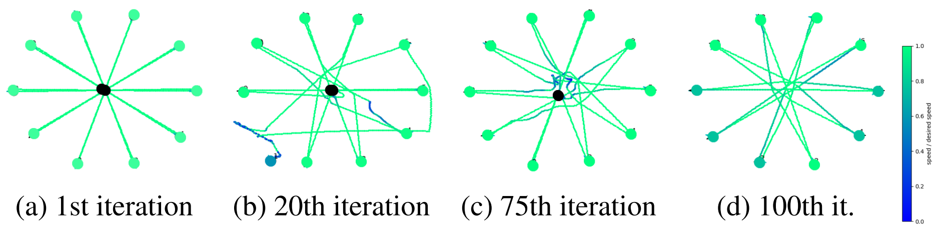
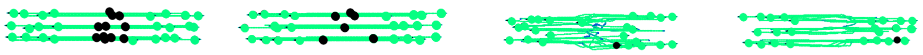


Figure 3.12: Crowd simulation samples, with agents steered using RVO, at various iterations of the learning process. Black dots represent collisions of agents.



(a) 1st iteration (b) 200th iteration (c) 800th iteration (d) 1000th it.

Figure 3.13: Simulation of two flows (25 RVO agents in total) in the flows crossing scenario at different stages of the learning process.

3.6 Discussion

The scenario generator of our framework can reproduce a large set of simulation cases, where features such as the density of agents in the environment or the number of crowd flow directions can be controlled. Its simulation quality function permits to evaluate the suitability of the trajectories generated, based on a parametric policy that can combine several pedestrian motion models. The possibility of studying the optimal parameters and optimal models in different scenarios is only one of the applications of the framework.

Unlike previous works, our framework can operate exclusively on synthetic data, hence overcoming the typical problems of real data-based approaches, such as acquisition, labelling or lack of variety. Based on a random scenario generator coupled with a trajectory quality function, we are able to train virtual agents in a large variety of situations.

Instead, we propose using a fitness function that defines how 'well' agents move and interact. A major difference with the evaluation framework of Wolinski et al. [2014a] is that we do not restrict ourselves to the variation of parameters of existing simulation techniques, although both works have in common that they try to find the main principle by which each agent should move (a steering function), so as to improve simulation results (according to a fitness function).

3.7 Conclusions

We presented a new perceptually-validated quality metric to automatically evaluate crowd trajectories. The proposed evaluation is designed with the help of experts and abstracts from real data. A number of trajectory features, deemed relevant to capture perception of trajectory quality, were selected with the support of experts. Then, measurements were proposed to learn the typical feature values in real data. These features were then combined in the so-called quality function, QF, which returns a score from 0 to 1 for any 2D ambient crowd trajectory. To validate QF we conducted a web-based experiment with non-expert users and we show that there is a high agreement between viewers' perceptions of visual quality and QF scores. Finally, we demonstrate a practical application for tuning parameters of crowd simulation models using QF.

Real trajectory data is only required when training QF. Once this is done, QF can be independently used to evaluate new trajectories, leveraging the feature information previously captured using real data. Besides, our feature-based approach allows avoiding over-fitting the parameters to a specific scenario or dataset by not using reference data directly. With this work, we provide a fully-functional, pre-trained, QF which is ready to use by the scientific community. Note however that the provided set of feature weights can be easily re-targeted to accommodate other use cases. Examples of these use cases could be to include reactions to obstacles, or specific behaviours such as queuing, which were not available in our training dataset. It it also important to point out that specific feature value ranges and weights can be manually adjusted while training QF, which opens the door to automatic generation of, e.g., non-realistic animations requiring overly-smooth trajectories, or simulating a marathon requiring specific speed ranges.

Once the weights are established, QF can be used to train crowd simulation models *without further requiring any real data*. This is one of the main advantages of our approach, as it facilitates parameter tuning for crowd simulation models. Consequently, training the crowd simulation parameters is, in our approach, data-free, while they are strongly influenced by the real data features captured by QF.

However, training a crowd simulation model using QF also has limitations. The main one is the ability of a simulation model to express a large variety of agent interactions. The learning algorithm might output parameter values leading to low S_{QF} values if it is unable to improve the quality score for a maximum number of iterations. For example, in Figure 3.9(bottom, right-most), some trajectory feature values differ from those found in reference data, e.g., average walking speed (faster) and Distance at Closest Approach (smaller). However, in the corresponding trajectories (top row), we see that the learnt parameters led to collision-free, smooth trajectories. This means that the genetic algorithm was able to find a parameter set p^{opt} to maximise QF but, due to the limitations of the model, the parameter sharing, and the symmetry of the scenario, the quality could not be further improved.

Nevertheless, QF has been proved to capture the main characteristics which affect human perception of trajectory quality. Nevertheless, a more complex combination of features could be explored to be able to evaluate other scenarios and behaviours (e.g., different types of agents, grouping, queuing), even though the combination of the features might be application dependent. As an alternative to tuning motion models directly, the more intuitive QF can be manually tuned by artists to fit a particular scene, such as for further penalizing collisions, or enforcing the importance for agents to reach their goal.

We have discussed how QF is useful for ambient crowds, which do not cover all possible crowd compositions or behaviours. Moreover, there is a significant body of work that has shown that not all pedestrians use the same steering strategies in all scenarios (it depends on factors such as age, abilities, cultural differences, etc.). In order to deal with diversity in complex scenarios, future research could focus on a QF-based policy adaption strategy dependent on character properties and their environment. For instance, this could enable characters in low density areas to use simpler motion models, while characters in higher density areas would use more complex models. In order to do this, it might be interesting to model the feature costs as a sum of Gaussian curves, a Gaussian mixture model, instead. Another approach could be to derive a crowd motion model that, in order to compute the optimal velocity at each time step, maximises QF.

For the longer term, we could study more interesting scenarios that are difficult to study or from which data is scarce. With an appropriate quality function, it might be possible to simulate and analyse situations were agents have a complex internal state. With this, we could aim at studying -for instance- groups of pedestrians in extreme situations such as evacuations or how emotions, such as panic, are spread depending on the characteristics of a crowd.

Chapter 4

DYNAMIC COMBINATION OF CROWD STEERING POLICIES BASED ON CONTEXT

While the used simulation parameters drive the way agents move, the underlying principle ruling the agents' movement for each algorithm is "ad hoc" and remains immutable. It has long been established that the simulation algorithms and their parameters have a direct impact on the resulting quality of animations. Extensive research has shown that each algorithm performs well in a limited range of scenarios, e.g., some perform better in high density cases than others. It is also known that using appropriate parameter values for a specific scenario can improve performance.

Nevertheless, as the number of published steering algorithms and techniques continues increasing, so does the difficulty of comparing their performances. This difficulty gave rise to a number of works exploring the questions of evaluating scenario coverage Kapadia et al. [2016], setting simulation parameters Wolinski et al. [2014b] and picking the best performing algorithm Karamouzas et al. [2018b]. Previous work, though, generally considered scenarios in a global manner (e.g., an evacuation, a narrow corridor, or a complex pedestrian crossing) and usually propose a unique technique for setting up a simulation (e.g., an algorithm and parameter values to best simulate a scenario).

However, the question of mixing multiple steering algorithms and of dynamically adjusting parameters during simulation has been quite unexplored. There is a set of open questions when using microscopic approaches: which is the validity range of the rules and steering functions used? How do they compare to each other? How to set simulation parameters in a principled and robust manner for general scenarios? How can the values of the parameters adapt to the situation the agents are in?

The objective of our work is to evaluate the benefit of dynamically adjusting characters' steering policy (i.e., the pair of a specific algorithm and a specific parameter setting) at run time to adjust behaviour of characters to their local situation, a region of a scenario that we call *context*. We consider a simple context definition, based on local density and the directions of the main flows of agents in a uniform region of the scenario. As an example, being part of a dense flow of people crossing another flow would define a context. Initially, we study the quality performance of various steering algorithms on a sample of the context space, at each one separately. To measure, we evaluate the simulated trajectories with a quality metric. At a final stage, we show that quality is further increased when characters periodically estimate their local context and pick one of the best steering policies, effectively adapting their behaviour to local context.

By doing this, we find that being able to adapt the policy of every agent dynamically -that is, somewhere in the middle of the simulation- to be able to better steer agents that pass through different scenarios - for instance, different densities- would be a very interesting improvement with respect to current approaches. Therefore, equipping the agents with the ability to determine the situation they are in will be very important to switch between policies. Our contributions are:

- A definition of a local context for crowd simulation agents based on density and main directions of local flows.

- A demonstration that this context definition is enough to discriminate the performance of various steering policies.

- A mapping from context to best performing steering policy.

- A mechanism to switch between different policies, and demonstrate the benefit of using multiple policies in a single simulation.

4.1 State of the art

This paper proposes novel ways in which traditional crowd steering algorithms can be used to improve the resulting simulations. We discuss here a number of relevant previous works related to traditional steering algorithms, the evaluation of trajectories created by such algorithms, how this information can be used to refine crowd simulations, etc.

4.1.1 Steering algorithms

The crowd simulation research field is concerned with understanding, predicting and reproducing the motion of real human crowds. Crowd simulators are based on several classes of algorithms which are designed to generate realistic trajectories of numerous moving characters. Various approaches to this problem have been proposed. *Macroscopic* approaches consider crowds as a whole, modeling it as a single continuous moving matter Hughes [2003]; Treuille et al. [2006]. *Microscopic* crowd simulation algorithms set the principles by which agents move individually and global crowd motion effects are expected to emerge from the interactions between agents. In Reynolds [1987] seminal work each boid followed the mean velocity field generated by neighbors. The number of categories of simulation algorithms rapidly grew with force-based models Helbing and Molnár [1995]; Karamouzas et al. [2014], velocity-based models Paris et al. [2007]; van den Berg et al. [2008]; Karamouzas et al. [2009], vision-based models Ondřej et al. [2010]; Dutra et al. [2017a], or data-driven models Lerner et al. [2007]; Charalambous and Chrysanthou [2014a]. These are few examples of a large body of literature.

Existing state-of-the-art steering algorithms are difficult to test and compare due to their very different strategies and implementations. To propose a standardisation for such algorithms, van Toll et al. [2020b] propose a holistic interpretation by transforming them into parametric cost functions in velocity space. The behaviours obtained with the algorithms in velocity space are very close to those obtained with the original algorithms. In this paper, the steering algorithms use the implementation in van Toll et al. [2020b].

There is a growing body of literature that recognises the different performance of steering algorithms in different scenarios van Toll and Pettré [2021]; Yang et al. [2020]. Numerous studies try to find the best parameters for existing steering algorithms, often comparing the results using data-based performance metrics Guy et al. [2012]. The objective of these works is to aid the selection of policies in order to improve the trajectories resulting from simulation. Some authors have even proposed strategies to profit from two steering strategies. For instance, van Toll et al. [2020a] combine agent-based (the Social Forces model) and particle-based approaches (Smoothed Hydrodynamic Particles) through abstraction layers in order to improve the behaviour in high density scenarios.

4.1.2 Quality metrics

Crowd simulations result in large sets of individual animation trajectories. Their quality depends on a number of rules by which agents move (simulation models), as well as parameter values to control the simulation. They are not intuitive nor easy to tune and often depend on the scenario to be simulated. Our objective is to propose a method to evaluate these simulation results, regardless of the method by which they are generated. We can distinguish various approaches to the evaluation of crowd simulations. A group of approaches uses paths of real crowds, and evaluate the ability of simulators to reproduce them. The question of comparison metrics is central, and several solutions have been proposed Guy et al. [2012];

Wolinski et al. [2014a]; Charalambous et al. [2014a]: these metrics consider crowd movement at different scales and take into account the variability of behaviors. However, there are drawbacks associated with the use of reference data, e.g., the ability of steering algorithms to replicate some patterns existing in real trajectory data and the limited amount of pedestrian trajectories available, which can lead to over-fitted results.

Research on policy selection (per-scenario or per-character) has been mostly restricted to limited comparisons between synthetic trajectories and real data. A broader perspective has been adopted by some authors that, instead of focusing on agent trajectories, measure crowd motion characteristics such as the ratio between the density and the average speed in different cultures Jelić et al. [2012]; Chattaraj et al. [2009]; Kapadia et al. [2011]. In this work, we use the perceptually validated quality function proposed in Cabrero Daniel et al. [2021] to evaluate the simulation results. This metric abstracts from real data by studying the distribution of a number of motion features and then penalising character motions deviating from what is found to be expected in real human pedestrian trajectories. By using the proposed quality function we remove the need of directly relying in real data to evaluate synthetic trajectories and we can evaluate previously unseen interaction types, always in the scope of ambient crowds (groups of pedestrians that do not show any specific behaviour other than walking to their goal and avoidance maneuvers). The benefit of this approach is that no real trajectory data needs to be gathered in order to evaluate synthetic trajectories. Moreover, the simulated trajectories do not need to resemble the ones found in the original data-sets.

4.1.3 Policy assignations

Closest to our approach is the work of Karamouzas et al. [2018a], which compares the performance of steering algorithms (using default parameters) in terms of distance-to-real-data in different scenarios. They compute how closely each steering algorithm is able to replicate the real trajectory of that character. This gives an insight of which algorithm (out of 6) works best in a type of scenario, e.g. a medium density area when entering a bottleneck corridor. The main difference with our approach is that they pick the best steering algorithm for a previously unseen scenario by studying the initial positions of characters and predicting what type of scenario it is – out of the data sets they use – and then selecting the steering algorithm with higher likely accuracy. Instead, we identify the context of each character at each time step and pick the optimal steering algorithm for the context, obtaining better quality simulations. To evaluate trajectories, the authors of Karamouzas et al. [2018a] propose a simulation accuracy metric, based on the Entropy metric by Guy et al. [2012], that measures the ability of a steering algorithm to create a trajectory similar to that found in real data. Instead, we rely on the metric proposed by Cabrero Daniel et al. [2021] that abstracts from real data. Moreover, the characterisation of characters' contexts, a key point in this work, is different from that of Karamouzas et al. [2018a]. Its authors derive a compact and continuous representation of pedestrian interactions directly from data based on per-agent *minimal predicted distances* (MPD). Instead, we model the context as the description of the dynamics in a neighbourhood of the agent, at each time, and we expect to cover a wide variety of different local interactions.

4.2 Overview

The objective of this work is to propose and compare a number of crowd simulation strategies, including a dynamic adaptation of characters' policy to their *local context*. We propose a policy adaptation technique to improve the overall quality of *crowd trajectories* simulated with traditional steering algorithms. Through the rest of this paper, we discuss and prove how the quality of simulations can be improved by increasing the adaptability of the characters to their surrounding environment.

We demonstrate the usefulness of our approach in the scope of *ambient crowds*, which are defined as groups of pedestrians that do not show any specific behaviour such as, e.g., queuing. The characters which compose the crowd are both homogeneous in the sense that the crowd is composed of similar-sized adults, and heterogeneous because each agent has its own attributes and objectives. Other types of contexts, for specific scenarios or applications, could be defined and evaluated in a similar way, and is further discussed in Section 4.5.

In this work, we analyse the performance of different navigation strategies using an existing quality metric QF Cabrero Daniel et al. [2021] that evaluates the quality of trajectories (orange boxes in Figure 4.1). Then, we propose a context recogniser which allows agents to adapt their navigation strategy depending on their local context (blue loop in Figure 4.1).

With all this information, we evaluate the relative performance of four crowd simulation strategies: (i) all characters in the crowd sharing the steering policy (baseline); (ii) optimising the steering policy for each character (to maximize the simulation quality); (iii) dynamically adjusting the policy of each character to its current context; and (iv) assigning policies, also according to context, following a probability distribution.

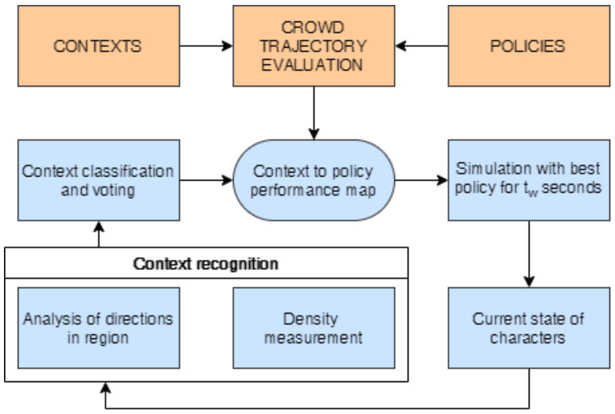

CONTEXTS
CROWD TRAJECTORY EVALUATION
POLICIES
Context classification and voting
Context to policy performance map
Simulation with best policy for t_w seconds
Context recognition
Analysis of directions in region
Density measurement
Current state of characters

Figure 4.1: Overview figure for context to policy mapping (orange), policy distribution among characters, and the context-adaptation loop (blue).

4.3 Methods

The following part of this paper describes in detail the creation a context-to-model map through the evaluation of the resulting trajectories. The analysis is made by studying the quality performance of steering algorithms in a variety of contexts (orange boxes in Figure 4.1). Section 4.3.1 presents the concept of context and details which types of contexts will be used in the paper. A number of steering algorithms is evaluated in each context of Table 4.1. The experimental setup for this analysis is then presented in Section 4.3.2, together with the results.

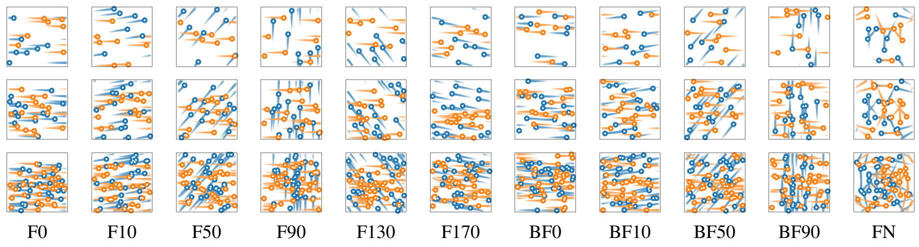


Table 4.1: Representative contexts defined in this work, where a context is defined by the local density and the directions of the main flows of neighbouring agents. Density increases with rows. F stands for Uniform Flow, BF for Bidirectional Flow, and FN for unstructured scenarios (N directions).

4.3.1 Contexts

Evidence from several studies suggests that the performance of different steering algorithms varies depending on the type of simulation they are used for. Crowds can be simulated in a variety of 2D *scenarios*, the latter being defined as a set of N agents together with their initial positions and internal properties (e.g., comfort speed, maximum acceleration, goal direction, radius, etc.). These scenarios can be arbitrarily complex and include a wide range of character interactions such as crossings, bottlenecks, etc. Within a given scenario, different zones can exhibit different interaction features at different points in time. Therefore, we could potentially identify several different *contexts* within the same scenario at a given simulation step. We define contexts as uniform spatial regions to which we associate a characterization of its trajectories based on the local density and the directions of the main flows of neighbouring agents.

To evaluate the performance of different algorithms in different contexts, we discretize the continuous context space into a representative subset of common pedestrian interactions. To this end, we consider three classes of contexts: (i) crossing of two unidirectional flows (F) of agents; (ii) crossing of two bidirectional flows (BF) of agents, where each bidirectional flow contains agents going along the flow in opposite directions; (iii) and unstructured contexts (FN). The crossing contexts are characterized by a bearing angle, which measures the angle at which the two flows cross each other. In order to cover a variety of interaction types, we have defined a total of 6 bearing angles for unidirectional flows crossing (ranging from 0 to 170°), and 4 bearing angles for bidirectional flows crossing (ranging from 0 to 90°). Moreover, we consider three levels of density of agents (low, medium and high) for each context, making a total of 33 representative contexts (11 per density level). The 11 contexts for each density level are shown in Table 4.1.

4.3.2 Context to policy performance map

This section discusses the performance of the navigation policies (i.e., a steering algorithm and its parameters) in each of the 33 different contexts considered in this work.

We decided to evaluate the performance of the following set of representative crowd simulation algorithms for all the contexts defined in the previous section. Each of these algorithms (in ascending order of computational complexity) is implemented in velocity space van Toll et al. [2020b], and will be tuned for each context based on the procedure described in the next section:

- Universal Power Law (PL) Karamouzas et al. [2014]

- Optimal Reciprocal Collision Avoidance (ORCA) van den Berg et al. [2011]
- TtcaDca (TTCA), based on the vision-based algorithm by Dutra et al. [2017a]
- Social Forces (SF) Helbing and Molnár [1995]
- Moussaid (Mou) Moussaïd et al. [2011]
- PLEdestrians (PLE) Guy et al. [2010]
- Reciprocal Velocity Obstacles (RVO) van den Berg et al. [2008]
- Karamouzas (Kar) Karamouzas and Overmars [2011]
- Paris (Par) Paris et al. [2007]

The best parameter setting for each of the considered steering algorithms and for each context is found by maximizing the quality function QF through the iterative process described in Cabrero Daniel et al. [2021]. During this optimization process, the algorithm parameters are not constrained, meaning that we compare algorithms at the "best of their abilities". The performance of each algorithm in each is stored and represented as "score maps" which are summarised in Table 4.2. This map only needs to be computed once and can easily be updated to introduce new steering algorithms.

In order to learn the parameters for each algorithm, we simulated crowds in toric worlds: finite planes where the movement is "wrapped around" i.e. if a character leaves the plane on one side, it appears on the other (and interactions in boundary areas are controlled). We use toric worlds in order to simulate continuous flows and uniform density crowds. In this work, we use 10x10 meter toric world simulations, where the number of characters depends on the density of the respective context (first column in Table 4.2). To find the best parameters for a specific steering algorithm and a specific context, a genetic algorithm is used. In the learning process, each simulation run is initialized with some random variations in the initial character positions, to provide some slight variations of the simulated context. Moreover, the initial seconds of each simulation are discarded in our measurements, as they might contain small artifacts which are not representative of the actual context, e.g. the initial position of characters might lead to strange avoidance maneuvers at the start of the simulation.

To measure the performance of a policy in a context, we study the QF score of 300 seconds of trajectories simulated in each context. Performances for each context and steering algorithm are summarized in Table 4.2, which is one of the core contributions of this work.

All policies using a sampling method to find the next velocity for characters (i.e., ORCA, Moussaid, PLEdestrians, Paris, Karamouzas, and RVO) use a relaxation time equal to 0.5. Unless stated otherwise, the neighbour interaction range for all the algorithms is 3.5 meters. Characters farther than this value will not be taken into account in to compute the next velocity, v', for characters. None of the policies used in this work take into account the contact forces (the coefficient is equal to zero) that apply when characters collide into each other.

In the event that two algorithms have similar average quality for a given context, the algorithm with less computational complexity is preferred. An example of this is a unidirectional flow with low density where the Universal Power Law (PL) is preferred. Similarly, ORCA is often chosen over RVO for being more time efficient. It is interesting to note that ORCA and RVO outperform other algorithms in high density scenarios. On the other hand vision based models, like Moussaïd et al. [2011], tend to work better in more complex scenarios, like unstructured crossings. As shown in the following section, this information can be used in a crowd simulation, to adapt the characters' policy mid-simulation and hence increase the quality of the final result.

4.4 Results

Our goal in this section is to demonstrate that it is possible to improve the quality of crowd simulations in any scenario, i.e., beyond one unique context. In Section 4.4.1, we propose a direct application of the context-to-policy map for this. This technique is based on automatically detecting the local agent context and assigning the agent the respective optimal policy. Then, we study whether we can further improve the results of the optimal policy per context found in Table 4.2. To this end, instead of using a single policy for all agents in a context, we assign the agents a mix of the two best policies found for each context (Section 4.4.2). The relative evaluation of these strategies, over a set of benchmark scenarios, is discussed in Section 4.4.3.

4.4.1 Dynamic context adaptation

When simulating a crowd in a given scenario, it is likely that several different types of interactions between agents will emerge in different sub-regions of the scenario and at different points in time. Therefore, from the perspective of an agent, the context (i.e., the other agents' motion features in a sub-region around the agent) is likely to be dynamic and to change several times during the simulation. We will refer to the dynamic context around a specific character as *local context*. To illustrate this from the perspective of an agent, let us focus, for example, on Figure 4.2. In this scenario we have two flows crossing at a 90-degree angle and the characters' "local context" changes from a single flow "context" to a crossing flows at 90 degrees "context" and back to the single flow "context". This observation, coupled with the results presented in Table 4.2 which show that the optimal policy is context-dependent, motivates our goal of detecting the agents' context during the simulation and, subsequently, to use this information to dynamically adapt the agents' steering policy. Our approach to this problem is described in the following sections.

Context detection

Our approach to the problem of run-time detection of the local context around each agent is: (i) first, we detect the type of motion on a small circular region around the agent of interest; (ii) then, we use this information to map the local context to one of the 33 studied contexts presented in Section 4.3. The relevant features required to perform this mapping operation are the local density and the local distribution of walking directions. The local density is computed by defining a radius r around the current agent, and determining the ratio between the number of agents inside that area and the area of the circle. We have experimentally found that using $r = 4\text{m}$ leads to good results. The local distribution of walking directions is determined by considering the walking direction of agents inside the circular area, filtered over a window of 1 second. Then we extract the main flows resulting from this set of directions by analytically studying the histogram, extracting the main modes. Once we obtain the main directions of the flows, we classify the context depending on the number of flows present (e.g., 1 for F0, 2 for other unidirectional flow crossings, and 4 for bidirectional flow crossings). Then, in the case of unidirectional and bidirectional flow crossing contexts, we compute the bearing angle between the two flows. The density and angle between character flows the local context is classified into one of the previously defined context bins. One more step is performed before selecting the policy to use in the current step, π_s: voting the most often recognised context among the characters inside a sub-region around a given character. The mode for the contexts is then used to select the best policy for the local context of each character.

Smooth policy transition

Changing the policy of a character c, in the middle of the simulation depending on its local context, lc_c, is prone to cause artifacts. This is because two steering algorithms, for very similar situations in consecutive time steps, might compute very different next velocities, $\mathbf{v}'_1$ and $\mathbf{v}'_2$. This could lead to sharp changes in the direction of characters when they enter a new context (related to flickering in direction). In order to ease the transition between algorithms we propose a transition strategy based on overlapping segments and algorithm combination in cost space.

The framework presented in van Toll et al. [2020b] uses combinable cost functions (in velocity space) to reproduce a number of steering algorithms, including those listed in Section 4.3.2. This way, characters transitioning from one context to another can progressively use less an algorithm and more another algorithm and compute the next velocity, $\mathbf{v}'$. The weights for this transition are given by:

$$\begin{aligned} \omega(t) &= (1 + e^{-kt})^{-1} \\ \mathbf{v}' &\leftarrow \omega(t)\pi_1 + (1 - \omega(t))\pi_2 \end{aligned} \tag{4.1}$$

where t stands for the percentage of completion of the transition (mapped from -0.5 to 0.5) and k is the steepness of the transition. In our work, we experimentally found that $k = 9$ results in smooth transitions between algorithms, as can be seen in the Supplementary Videos. The selected $\mathbf{v}'$ for a character corresponds to an admissible velocity that minimises the combined costs of the two algorithms in velocity space.

Nevertheless, some steering algorithms might not be directly compatible for they can return opposite next velocities for the same character state, e.g., avoiding maneuvers turning right or left. A typical example of this is combining a velocity-based model like RVO with a force-based model like Social Forces. Figure 4.3 illustrates this problem, where colored areas represent a simplification of the regions in velocity space where values of the costs functions are minimal. In this example, combining the two policies directly, would not necessarily make sense: the selected $\mathbf{v}'$, optimal for both algorithms (the $\mathbf{v}'$ with lowest overall cost), could mean "not turning" (which could lead to a collision) or even reducing the walking speed to a stop.

Instead of directly combining the steering algorithms, we study the predicted next velocity for both algorithms separately and look for inconsistencies in the outputs of the two algorithms (i.e., if the angle between v_1 and v_2 is greater than a threshold, th). If the two steering algorithms return opposing solutions, the next velocity is selected among the two by studying whether they are consistent with the previous motion and depending on the value of t. If no inconsistencies are present, the velocities computed by the two algorithms are combined using Eq. (4.1).

Simulation loop

Our simulation loop with dynamic context-based policy adaptation is shown in Algorithm 2. The time window (measured in simulation steps) in which the same policy is applied is given by t_w. The algorithm starts by delimiting the circular area $\mathcal{A}$ with radius r centered on the position p of the current agent (line 1). Then, in lines 2 to 5, the local context of the N agents within the area $\mathcal{A}$ is detected, based on the motion features of each area $\mathcal{A}_n$ centered on each of the agents n. Note that this includes the current agent for which we want to adapt the policy. A voting step follows, in which the final local context $\mathcal{C}$ of the current agent is chosen based on the set of contexts $\{c_1, \ldots, c_N\}$ detected for all agents within the area $\mathcal{A}$ (line 6). This context is then used to determine the new motion policy π (line 7), using the context to policy map $\mathcal{P}$ described in Section 4.3. In lines 9 to 15, the transition between the previous policy $\hat{\pi}$ and the new policy π is smoothed out during t_s steps, in case strong motion discrepancies are detected, hence avoiding undesired discontinuities in the simulation. Finally, the remaining simulation steps within the time window t_w are performed using the context-adapted policy π (lines 16 to 19). This process is represented with blue boxes in Figure 4.1.

Results

To present the results of our approach, we first illustrate them with the specific case of a two-flow crossing (from left to right and from bottom to top), simulated either with PL, RVO, or our approach (see Figure 4.2). In Figure 4.2a, we can see that the PL method struggles in regions where the two flows cross, making the characters move diagonally. This is penalised by QF because characters deviate from their desired direction for too long, even excessively moving away from their goal. On the other hand, in Figure 4.2b using RVO we can see that even if the flows are

Algorithm 2: Policy adaptation algorithm for a given agent.

```
Input : $\mathcal{P}$: context to policy map
        $\hat{\pi}$: policy in previous time window
        $p$: agent position
        $r$: radius of circular area defining local context
        $t_w$: number of simulation steps to be performed
        $t_s$: number of simulation steps for policy transition
1  $\mathcal{A} \leftarrow$ getSubRegion($p, r$)
2  for each agent $n \in \mathcal{A}$ do
3      $\mathcal{A}_n \leftarrow$ getSubRegion($n, r$)
4      $c_n \leftarrow$ detectLocalConext($\mathcal{A}_n$)
5  end
6  $\mathcal{C} \leftarrow$ voteContext($\{c_1, \ldots, c_N\}$)
7  $\pi \leftarrow \mathcal{P}(\mathcal{C})$
8  $t_t \leftarrow 1$
9  if $\pi \neq \hat{\pi}$ then
10     for $t$ in $[1, t_s]$ do
11         $\mathbf{v}' \leftarrow$ smoothVelocity($\hat{\pi}, \pi, t$)
12         simulate($t, \mathbf{v}'$)
13     end
14     $t_t \leftarrow t_t + t_s$
15 end
16 for $t$ in $[t_t, t_w]$ do
17     $\mathbf{v}' \leftarrow$ getVelocity($\pi, t$)
18     simulate($t, \mathbf{v}'$)
19 end
```

able to cross each other, characters tend to move apart too much from each other (larger spread of characters across the two flows). Finally, Figure 4.2c shows an example of our policy switching based on the "score map" presented in Table 4.2, where PL is used in low density, unidirectional contexts, and RVO is used in low density, 90 degrees crossings. Overall, Figure 4.2c shows that characters do not deviate too much from their goal direction and switch back to PL as soon as they exit the crossing area, as the majority of characters around them now lead to a change in the distribution of directions (leading to a change of context). This also enables characters to switch back to the less computationally complex PL method, that works well for unidirectional flows.

We can interpret policy adaptation in two ways: (i) changing the steering policy to be able to deal with complex interactions or (ii) "relaxing" the algorithm when the scenario does not require a more time consuming algorithm to correctly solve the interactions. In an extreme case, when distances between neighbours are acceptable (an interaction range of 3 meters is commonly used in the literature) and all agents have the same comfort speed one could use a goal reaching force (without avoidance maneuvers) because there would be no predicted collisions nor unreasonable values for other features.

4.4.2 Mapping context to a distribution of policies

If the steering algorithm and its parameter values are not shared among all characters, the crowd is heterogeneous and characters exhibit different behaviours, typically leading to better simulation results Wolinski et al. [2014b]; Guy et al. [2012]. The following sections are concerned with producing heterogeneous crowds within each context using the information contained in Table 4.2. In particular, Table 4.2 can be used to map contexts to a distribution of policies, instead of mapping a context to a single (optimal) policy as described in the previous sections.

We propose and evaluate two strategies to assign different policies to agents in a context. In the first strategy, presented in Section 4.4.2, we randomly assign each agent one of the two best policies learnt for each context. In the second strategy, presented in Section 4.4.2, we replace the random assignment of policy to each agent by an optimization process which determines to which particular agents should each of the two policies be assigned, so as to maximize the QF score.

Random selection of agent policies

To determine to which extent it is beneficial to combine different policies within the same context we have measured the simulation quality per context using a combination of two policies: the optimal policy p^* for the context, and the second best policy p'^* found for the same context according to Table 4.2. Each agent randomly picks one of these two policies following a probability distribution aimed at keeping the ratio between p^* and p'^* at a desired level.

Figure 4.4 (blue colored bars) shows the average quality across all contexts of crowd trajectories where characters randomly pick among p^* and p'^* following a probability distribution. The results have been generated considering different ratios between p^* and p'^*, ranging from all agents choosing policy p^* to all agents choosing policy p'^*. The blue bars show that the average QF score is higher when characters share the same policy, compared to the case where characters with different policies coexist in the same context. Contrary to expectations, no significant increase in the quality score, S_{QF}, was found compared with using a single policy optimised for a specific context.

Optimized selection of agent policies

Further statistical tests revealed that the average crowd trajectory quality across contexts could be improved by distributing the policies among characters in an informed way. The goal of this learning process is to maximise the quality of the resulting trajectories while maintaining the proportion of characters using each of the two best performing steering algorithms. The relation between the proportion of characters using each of the two best performing steering algorithms and the resulting average quality is represented in Figure 4.4 (orange colored bars). We can therefore conclude that, in contrast to a random assignation of steering algorithms, an informed distribution of the two bests algorithms for each context leads to an improvement in quality. The proportion of characters using a “complimentary” steering algorithm for a particular scenario seems to affect the resulting trajectories’ quality. An explanation for this improvement could be that the crowd simulator avoids some artifacts by changing the steering policy of some of the characters.

We can therefore conclude that the quality of trajectories simulated in a specific context can be further improved when different characters use different policies, even if only a small percentage of characters use a different steering algorithm. Nevertheless, the small increase in S_{QF} is likely to be related to the optimisation of the policies per-context which is done in homogeneous contexts where all agents shared the same policy. There is a risk that the used policies are not well adapted to contexts where characters use different steering strategies, such as in the experiments conducted in this section. As discussed in Section 4.5, the quality might be further improved if instead of using the policies tuned in Section 4.3.2, a mixture-of-policies for each context was learnt instead.

4.4.3 Strategy comparison

To quantitatively assess the effectiveness of the different motion strategies proposed in our paper we have evaluated their performance in a variety of scenarios. This quantitative evaluation of relies on the quality function QF, and on the following benchmark scenarios:

- Two groups of characters moving in opposite directions and crossing in the center. When the groups overlap, the density increases; after crossing, the density returns to the original value.

- Four groups of characters move towards the opposite side of the world, passing through the center. In the crossing, the local context of characters is a bidirectional flows crossing.

- Circle crossing: characters are disposed in a circle around the center of the world; their goal is to reach the opposite side.

- Two unidirectional flows move in the same direction and overlap. Characters in one of the flows have a higher desired speed.
- Random scenarios where every character in the crowd has its own initial position, desired direction and comfort speed.

For each scenario, several evaluations with different initial agent positions are made. The tested motion strategies are:

- S_0, where all characters use the same policy during all the simulation. This strategy should be seen as a baseline strategy;
- S_1, where each character has its own policy which is kept constant throughout the simulation; the algorithm and parameter values are optimised per character with QF as individual fitness measurement.
- S_2, which corresponds to the case where agents can dynamically switch policies during the simulation based on the context to policy map, as described in Section 4.4.1 and Algorithm 2;
- S_3, which corresponds to the case where agents can dynamically switch policies during the simulation but, in contrast with S_2, each context is mapped to a distribution of policies (instead of being mapped to a single policy), as described in Section 4.4.2.

The results for these four strategies are presented in Table 4.3. They show that, compared to the baseline method (S_0), the simulation quality can be improved by an average of 28% by simply tuning the policy so as to fit the current scenario, even if all agents use the same policy (S_1). This is inline with the findings reported by previous works such as Wolinski et al. [2014b] or Karamouzas et al. [2018a]. Moreover, Table 4.3 also confirms the significant benefit brought by dynamically adapting the agents' policy based on their local context (S_3 and S_4). Such strategy, which advances the state-of-the-art by exploring an alternative way to describe the agents' local context and using QF to perform calibration, brings improvements of up to 46% in terms of average simulation quality with respect to the base-line method.

4.5 Limitations and future work

Context. The motion of a crowd can always be decomposed into a number of flows. Following this idea, we introduce a definition of *context* which is based on few main properties of these flows: their density and relative angle. Through the study of a discrete set of contexts, we demonstrate that those properties indeed discriminate various algorithms in their capacity of handling them correctly. We however left for future work a number of other context features that are likely to also influence simulation quality, such as flow rate or non-uniform distributions of flows. We could also have considered other features like the presence of groups or a higher heterogeneity in agents behaviors, etc. Notwithstanding their relatively simple characterisation, this work offers a proof of concept for a dynamic adaptation of steering policies based on local contexts.

Trajectory Evaluation. Not using data (as is the case of our proposed approach) has many advantages such as, for example, not having to collect and process real trajectory data, or not risking over-fitting to a particular data set. However the chosen trajectory evaluation metric (QF Cabrero Daniel et al. [2021]) limits the scope of evaluation to ambient crowds. The approach that we propose might thus be less suited for specific contexts or behaviours not considered in Cabrero Daniel et al. [2021]. Nevertheless, the role of QF is a modular one, and, depending on cases, more appropriate evaluation techniques (already explored in previous work) might be easily considered.

Algorithms. In this paper, we study a variety of crowd steering algorithms, all implemented within the framework of van Toll et al. [2020b]. Some categories of approaches, such as data-driven or reinforcement learning ones, are not covered in this framework and were not considered. This type of approach do not exactly follow similar parameter tuning and evaluation procedure since results more depend on training or example datasets. Data-driven techniques implicitly generate, at some scale, human-like trajectories. Future work is required to adjust our method to these categories of simulation techniques, however, our results offer an insight for improving their design. For example, it may be useful to decompose policies, training or datasets according to contexts.

Policy switching. One could think about many techniques to switch, or even mix, navigation policies. We here prove that a simple method to switch policies is already effective, but many more could be explored, e.g. using multiple tuning policies at all times and voting the next velocity, v', for each character. Moreover, the transition between policies are synchronized for they happen every t_w seconds. A strategy where characters would recognise the context for every time step in the simulation would avoid potential artifacts due to characters switching their policy simultaneously.

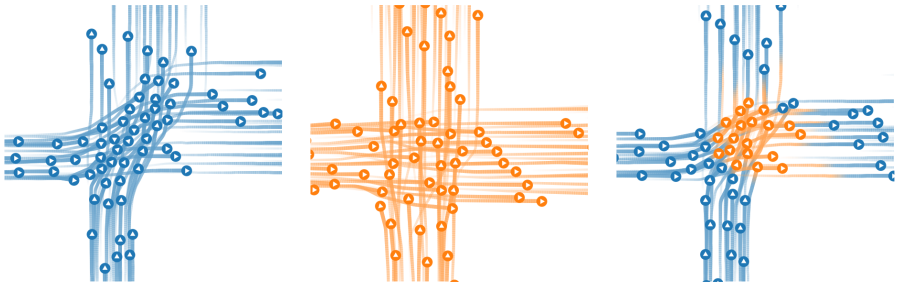
Policy optimisation. We optimized each policy independently to each context. At the same time, we show that mixing policies may further improve results. The natural following step would be to jointly optimize a mixture of policies, i.e., further refine parameters of a mixture of policies to context. It is also probable that *good* combinations of policies is something existing. At least, we may find something that can be observed in real humans who adjust their navigation to the one of their neighbors, e.g., being more careful among inattentive people, or conversely.

4.6 Conclusions

In summary, we have proposed a framework to dynamically adapt the motion policy of characters when simulating large virtual crowds. Our approach is based on a context to policy map which shows for the agents' local context to a set of optimized policies, that are learnt once and for all in a previous step without requiring any real motion data. To this end, we have proposed a discretization of the full context space into a subset of 33 representative contexts and learned the optimal performing policies for each of the contexts. During the simulation, the agents' context is automatically detected and mapped to an optimized policy, which results in a crowd where characters dynamically adapt their motion strategy depending on their situation. Our results demonstrate the benefits of our approach for the crowd simulation quality, exhibiting a significant crowd quality improvement both visually and in terms of a quantitative perceptually-based quality function. Furthermore, the data-independence of our approach opens the path to easily build on and extend our framework to other contexts and policies, which can potentially trigger future research.

		Steering Algorithm								
Density	**Flow**	PL	ORCA	TTCA	SF	Mou	PLE	RVO	Kar	Par
0.5	F0	94	94	89	93	94	86	93	91	86
0.5	F10	92	92	89	91	92	85	92	93	86
0.5	F50	90	91	85	87	91	88	91	88	85
0.5	F90	91	90	89	90	91	87	92	89	85
0.5	F130	90	90	88	90	91	87	91	88	84
0.5	F170	90	90	86	88	91	87	90	87	82
0.5	BF0	91	92	88	88	91	88	93	88	85
0.5	BF10	87	89	83	84	90	85	88	85	81
0.5	BF50	84	86	79	78	87	81	85	82	76
0.5	BF90	88	87	82	90	88	85	88	83	82
0.5	FN	85	85	77	80	86	83	85	81	78
1.0	F0	90	92	80	79	91	87	93	86	81
1.0	F10	90	93	83	87	92	88	91	88	85
1.0	F50	88	89	75	82	89	86	88	90	84
1.0	F90	87	90	72	77	89	84	89	91	82
1.0	F130	87	88	76	81	88	85	88	89	83
1.0	F170	85	89	73	78	88	85	86	81	80
1.0	BF0	86	91	69	73	88	86	89	82	80
1.0	BF10	83	85	74	80	84	82	83	79	80
1.0	BF50	76	77	64	71	78	76	77	72	73
1.0	BF90	77	78	59	66	79	76	77	73	71
1.0	FN	79	82	72	71	81	78	80	76	76
2.0	F0	87	93	76	78	92	88	91	84	84
2.0	F10	82	91	71	70	90	84	92	81	76
2.0	F50	82	84	74	75	84	80	85	79	78
2.0	F90	77	81	68	66	80	76	79	74	72
2.0	F130	73	79	63	53	77	73	77	71	68
2.0	F170	76	78	67	70	77	74	76	72	71
2.0	BF0	76	81	66	64	80	76	79	73	71
2.0	BF10	69	73	61	49	73	67	74	68	64
2.0	BF50	62	70	64	72	71	68	70	64	54
2.0	BF90	61	64	71	72	61	66	69	58	53
2.0	FN	54	55	75	50	53	52	75	44	58

Table 4.2: Context to the performance of each steering algorithm with appropriate parameters. The best performing algorithm is used to map each context to a steering policy. he score in each cell is computed using the quality function QF proposed in Cabrero Daniel et al. [2021].



(a) Universal Power Law ($S_{QF} = 67$) (b) RVO only ($S_{QF} = 84$) (c) PL and RVO ($S_{QF} = 92$)

Figure 4.2: Crowd simulation of two flows of agents crossing at 90º using three different motion strategies, and their respective score S_{QF} according to a quality function. At the left (4.2a), agents are steered using the Power Law model (PL, blue). Note that some agents are dragged along a diagonal towards the up-right direction, hence deviating them from their goal. At the center (4.2b) trajectories are generated using the Reciprocal Velocity Obstacles (RVO, orange). Note that, in this case, characters tend to move apart too much from each other. At the right, trajectories are generated with our approach, where characters dynamically switch motion policy depending on their local context, hence overcoming the motion artifacts of 4.2a and 4.2b. In this example, characters use the PL model, but switch to RVO when in the 90º crossing context. The agents' color encodes their current policy. Our dynamic adaptation results in an increase of the overall quality score, S_{QF}.

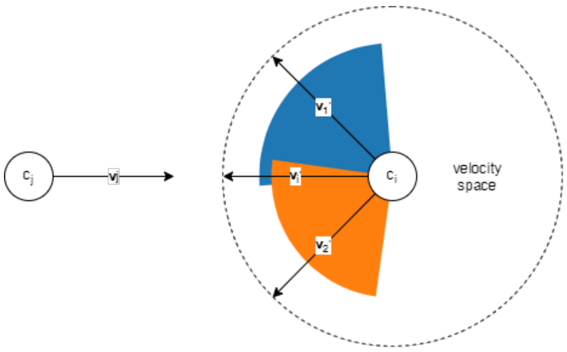


Figure 4.3: Representation of the issues with selecting the next velocity $\mathbf{v}_i'$ for character c_i when using motion combination in velocity space. The best region in velocity space for the first algorithm (blue) slightly overlaps the best region in velocity space for the second algorithm (orange). Nevertheless, the velocities lying in the overlapping area (best for both steering algorithms at once) lead to a collision with c_j.

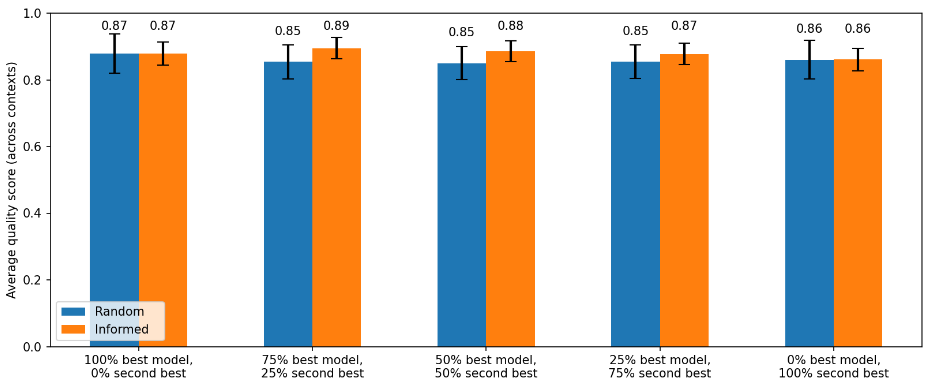


Figure 4.4: Average quality (across all contexts) depending on the proportion of characters using the best performing steering algorithm and the second best performing algorithm.

Strategy	**Mean**	**Variance**
Policy sharing (S_0)	0.61	0.09
Policy per-character (S_1)	0.78 (+28%)	0.06
Switching algorithms (S_2)	0.87 (+43%)	0.06
Switching distr. (S_3)	0.89 (+46%)	0.13

Table 4.3: Average quality of the trajectories resulting from using each of the studied strategies. The percentages in-between brackets for S_1, S_2 and S_3 show the improvement brought by these strategies with respect to the base-line strategy S_0.

Chapter 5

CONCLUSIONS AND FUTURE WORK

Computer-simulated crowds are becoming increasingly common in the film and video gaming industries. Crowd simulation is also used in the field of safety assessment to study and prevent potentially hazardous scenarios e.g. in stadium evacuations or fire drills in office buildings. Existing crowd motion models used to simulate crowds are highly diverse, and no general solutions exist. The results of such simulations are difficult to compare, making it difficult to pick the best suited crowd motion model.

In this work, we looked at how to make decisions about crowd simulation from the perspective of autonomous results evaluation. We've focused on the necessity to assess the performance of various crowd motion models without directly depending on real-world data, which is difficult to gather. Eventually, we linked contexts, previously defined, to appropriate policies. This information can be used to guide the policy selection or even allow the characters in the simulated crowd to adapt their steering strategy dynamically.

Figure 5.1 illustrates the interrelation between the topics and contributions discussed in this thesis. The Learning Crowds framework (blue boxes), described in Chapter 2, uses a fitness function in order to iteratively improve the resulting trajectories. As discussed in Section 2.4, an external module, *UMANS*, is used to generate trajectories for the characters in the crowd and test the performance of a particular parameter value set for a particular steering algorithm. The trajectory evaluation, is done through the Quality Function represented by a green box in Figure 5.1. The Quality Function, QF, is described and validated in Chapter 3. The orange boxes in Figure 5.1 represent the policy to context adaptation loop presented in Chapter 4.

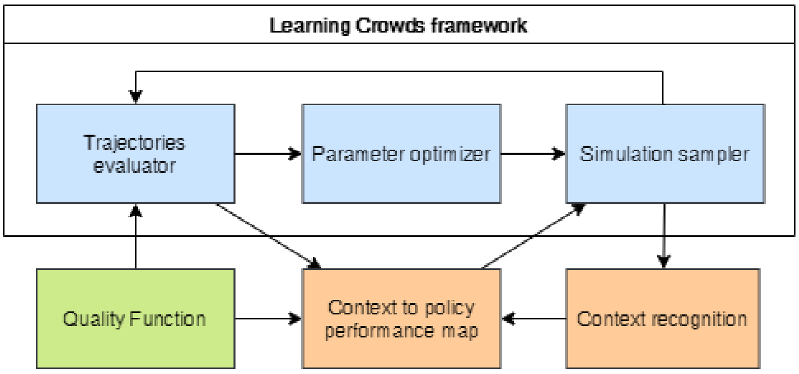


Figure 5.1: Overview of the covered topics and contributions.

5.1 Contributions

Uniform motion algorithm representation (UMANS). To simulate the low-level microscopic behavior of human crowds, a local navigation algorithm describes how characters move based on their surroundings. Many algorithms for this purpose have been proposed, each using different principles and implementation details that are difficult to compare. We first focused on designing a novel framework that describes local agent navigation generically as optimizing a cost function in a velocity space. We showed that many state-of-the-art algorithms can be translated to this framework. This software enables easy experimentation with different algorithms and parameters. This approach helps understand the differences between navigation methods and enable honest comparisons between them.

Automatic trajectory quality evaluation. The study of the relation between parametric values for simulation techniques and the quality of the resulting trajectories had previously been studied either through perceptual experiments or by comparison with real crowd trajectories: we integrated both strategies. A quality metric, QF, was proposed to abstract from reference data while capturing the most salient features that affect the perception of trajectory realism. QF weights and combines cost functions that are based on several individual, local and global properties of trajectories. These trajectory features are selected from the literature and from interviews with experts. To validate the capacity of QF to capture perceived trajectory quality, we conducted an online experiment that demonstrated the high agreement between the automatic quality score and non-expert users. To further demonstrate the usefulness of QF, we used it in a data-free parameter tuning application able to tune any parametric microscopic crowd simulation model that outputs independent trajectories for characters. The learnt parameters for the tuned crowd motion model maintained the influence of the reference data which was used to weight the terms of QF.

Context to policy dynamic adaptation. The question of choosing the right crowd simulation algorithm with the right parameter values is of crucial importance given the large impact on the quality of results. We combined the two previous concepts in order to achieve an autonomous and informed mapping from context, representing the environment of a character, to best-performing policy (i.e., simulation algorithm and its parameters). We study the performance of a number of steering policies in a variety of contexts, resorting to the proposed quality function, QF. This analysis allows us to map contexts to the performance of steering policies. Based on this mapping, we demonstrate that distributing the best performing policies among characters improves the resulting simulations. Furthermore, we also propose a solution to dynamically adjust the policies, for each agent independently and while the simulation is running, based on the local context each agent is currently in. We demonstrate significant improvements of simulation results compared to previous work that would optimize parameters once for the whole simulation, or pick an optimized, but unique and static, policy for a given global simulation context.

Learning Crowds framework. All this was integrated in a framework for crowd simulation whose objective is the analysis and synthesis of crowd motions so that the generated trajectories of agents exhibit desired properties. We presented a full framework architecture which, unlike previous works, can operate without any real-world measured data. This is achieved through a random scenario generator coupled with a trajectory quality evaluation i.e. QF. The use of synthetic data instead of real-world data allows to easily change and extend the variety of training situations, as well as to study in-depth the influence of scenario features on the characters' behaviour.

5.2 Future Work

Not using data (as is the case of our proposed approach) has many advantages such as, for example, not having to collect and process real trajectory data, or not risking over-fitting to a particular data set. However the chosen trajectory evaluation metric, QF, limits the scope of evaluation to *ambient crowds*. Nevertheless, as discussed in Chapter 2, the framework is modular; this allows for other, potentially more appropriate, evaluation techniques to be used instead.

On the one hand, more complex trajectory quality evaluation functions (e.g. based on mixture of Gaussian curves) could be used to detect specific behaviour patterns in character trajectories. Also, a polynomial quality function combining trajectory features, could be proposed. These might lead to better results in terms of performance measurement and therefore improve the parameter tuning results. On the other hand, different trajectory features, currently not considered in Chapter 3, could be added to QF e.g. individual character properties like personality [Durupinar et al., 2011], local crowd properties such as emotion contagion [Alemi et al., 2015], or even global characteristics based on path planning information [Van Toll et al., 2012]. New features might allow the evaluation of trajectories that are not inside the scope of *ambient crowds*.

A wider range of contexts could have been taken into account in order to perform per-character policy switching i.e. learn a larger context to policy map. The main goal would be to study contexts outside the scope of *ambient crowds*, the scope of QF as discussed in Chapter 3. Since QF is not, in principle, suited for those specific contexts and behaviours, a different trajectory evaluation function would need to be proposed.

5.3 Summary

In this work we discuss three of these techniques to advance the field of Crowd Simulation. The proposed methods, automatize crowd simulation-related tasks (i.e., parameter tuning, trajectory evaluation, and character policy selection and adaptation) without relying on real data directly. First, a novel holistic approach to implement crowd simulation algorithms as combinable cost functions. Second, a character trajectory evaluation function that, once trained, can be used to assess the plausibility of trajectories without requiring them to be compared to real data, which is difficult to gather. Third, a method for making informed decisions about crowd simulation algorithms. Nonetheless, despite these promising results, there is still much room for improvement both in the number of steering algorithms used and the range of pedestrian interactions covered. In order to further automatize Crowd Simulation-related decisions, additional studies will need to be undertaken.

Appendices

Appendix A

QUESTIONNAIRE FOR EXPERTS

The survey for experts, aimed at selecting and understanding important trajectory features to evaluate crowd realism, was composed of a relatively long list of questions which, for the sake of space, we only include here. All the questions in the survey for experts can be found in the following lists.

Walking and comfort speed *Walking speed* is an individual property of an agent. Walking speed is the speed at which agents are moving at a particular time-step. Questions asked about this feature: “I agree with this definition of walking speed.”, “The video is showing agents moving at different walking speeds.”, “In general, walking speed is related to the perceived quality of animation trajectories.”, “The agents should present a variety of walking speeds in an unconstrained context where no collision avoidance is required (comfort speed).”, “The walking speed can change because of the environment.”, “If you do not agree that this feature is related to perceived quality, please indicate why.”, and “Do you have anything to add on this feature?”

Flickering in direction and speed *Flickering* in a trajectory is present when an agent exhibits many sharp changes of walking speed and/or direction in a short period of time. Questions asked about these features: "I agree with this definition of flickering.", "The first video displays flickering in direction.", "The second video displays flickering in walking speed.", "Visible flickering in direction negatively affects the perceived quality of trajectories.", "Visible flickering in speed negatively affects the perceived quality of trajectories.", "In an unconstrained scenario (no collision avoidance), flickering is always perceived as unrealistic.", "Is there any scenario in which flickering is not perceived as unrealistic? Which and why?", "If you do not agree that flickering is related to the quality of agent trajectories, please indicate why.", and "Do you have anything to add on this feature?"

Inertia *Inertia* is the resistance of any moving or non-moving object to change its velocity (related to their mass and the forces applied to them). Questions asked about this feature: "I agree with the proposed definition for inertia.", "In this video, the agent at the top shows inertia.", "In this video, the agent at the bottom shows inertia.", "Lack of inertia negatively affects the perceived quality of agent trajectories.", "Humans have a restricted range of admissible acceleration values.", "If you do not agree that displaying inertia is related to the perceived quality of animation trajectories, please explain why.", "If you do not agree that lack of inertia is related to too much acceleration, please explain why.", and "Do you have anything to add on this feature?"

Turning speed, also called *angular velocity*, is measured in radians (or degrees) per second and is used to measure the change in direction of an agent. Questions asked about this feature: "I agree with the definition of turning speed.", "The video is showing different turning speeds for agents.", "Turning speed is related to the perceived quality of animation trajectories.", "There is a maximum turning speed for human walkers.", "As the walking speed of agents increases, their turning capability decreases.", "Changes in direction outside the admissible range of turning angles negatively affect the perceived quality.", "If you do not agree that angular velocity affects the perceived quality of animation trajectories, please specify why.", and "Do you have anything to add on this feature?"

Goal attraction The *direction of movement* of an agent is the current direction an agent is moving in. The *goal direction* can be the direction to a point in space or a desired direction depending on the position of the agent. Questions asked about these features: I agree with the proposed definition for the direction of movement.", "I agree with the proposed definition for the goal direction.", "The agent at the top (unconstrained) is always moving towards its goal.", "The agent at the bottom (unconstrained) is always moving towards its goal.", "The difference between direction of movement and goal direction negatively affects the quality of agent trajectories.", "Agents walk in a straight line in a free environment.", "There is a trade-off between avoidance maneuvers and moving towards the goal.", "The goal direction of an agent can always be inferred from the motion.", "If you do not agree that the direction of movement affects the quality of agent trajectories, please specify why.", "If you do not agree that quality depends on the difference between the direction of movement and the goal direction, please specify why.", and "Do you have anything to add on this feature?"

Reaching the goal means being very close to a particular position at the end of the trajectory of an agent. Questions asked about this feature: "I agree with this definition of goal reaching.", "This video shows an example of goal reaching.", "It is always possible for an agent to reach its goal.", "Not reaching the goal negatively affects the perceived quality of agent trajectories.", "Goal reaching is relevant for the measurement of the quality of the agent trajectory.", "Checking if the goal was reached is needed for the measurement of quality.", "If you do not agree that not reaching the goal negatively affects the perceived quality of trajectories, please indicate why.", and "Do you have anything to add on this feature?"

Density *Local density* refers to the density (persons per square meter) around an agent. For instance, inside a circle centered at the agent. Questions asked about this feature: "I agree with this definition of local density.”, “This video shows a representation of local density.”, “Local density is related to the perceived quality of the agent trajectories.”, “Agents avoid high density areas when possible.”, and “High local density negatively affects the perceived quality of trajectories.”

Environment-based density is the density at subdivisions of the world. Questions asked about this feature: “I agree with this definition of environment-based density.”, “If you do not agree that local density affects the perceived quality of animation trajectories, please indicate why.”, “Do you have anything to add on local density?”, “This video shows a representation of environment-based density in different areas of the environment.”, “The distribution of the environment-based density affects the quality of animation trajectories.”, “People positions are generally not uniformly distributed in their environment.”, “If you do not agree that density affects the perceived quality of animation trajectories, please indicate why.”, and “Is there something you would like to add about density?”

The *pairwise distance between agents* is the distance between a pair of pedestrians Questions asked about these features: “I agree with the proposed definition for distance between agents.”, “This video shows the evolution of distance between two agents.”, “Close distances to other agents can negatively affect the quality of a trajectory.”, “There is a minimum distance to other agents that is considered acceptable.”, “The minimum acceptable distance is reduced when local density increases.”, “The minimum acceptable distance to other agents decreases when relative speed increases.”, “If you do not agree that distance to other agents affects the perceived quality of the trajectories, please explain why.”, “Do you have anything to add on this feature?”, “I understand what the distance to obstacles refers to.”, “When unconstrained, staying far from obstacles is expected in a high quality trajectory.”, “There is a minimum distance to obstacles that is considered desirable.”, “The minimum desired distance to obstacles varies depending on local density.”, “The minimum desired distance to obstacles varies depending on walking speed.”, and “Do you have any comments on distance to obstacles?”

Trajectory length The *travel time* is the total elapsed time (seconds) from the beginning of the motion until the agent stops, reaches its goal, exits the world, or the end of the trajectory is reached. The *total length of a trajectory* is the distance (meters) walked by an agent in the same travel time span. Questions asked about these features: “I agree with this definition of travel time.”, “I agree with this definition of trajectory length.”, “In the video, we can see an example of different travel time.”, “In the video, we can see an example of different trajectory length.”, “The travel time is related to the perceived quality.”, “Trajectories longer than the shortest path in an unconstrained environment (straight line) are perceived as less believable.”, “Collision avoidance might make trajectories longer.”, “When performing collision avoidance, unnecessarily longer trajectories are perceived as less believable.”, and “Do you have any comments on travel time and trajectory lengths?”

Avoidance patterns *Interaction strength* is the amplitude of the acceleration or deceleration caused by collision avoidance. *Anticipation* is the distance (in time-steps or in meters) at which agents adapt their motion to avoid the collision. Questions asked about these features: “I agree with the definition of interaction strength.”, “I agree with this definition for anticipation.”, “In the videos, we can see different interaction strengths for collision avoidance.”, “In the videos, we can see different anticipation times for collision avoidance.”, “The strength of the interactions is related to the perceived quality of trajectories.”, “The anticipation of agents is related to the perceived quality of the trajectories.”, “The interaction will be stronger if a collision is imminent.”, “If you do not think the interaction strength is related to perceived quality, please explain why.”, and “Do you have anything to add on this feature?”

The *Time To Collision* (TTC) is the predicted time until a collision happens. Questions asked about this feature: “I agree with the proposed definition for Time To Collision.”, “This video displays the Time To Collision (TTC).”, “Low TTC values negatively affect the perception of quality of agent trajectories.”, “Low TTC values do not negatively affect the perception of quality, only actual collisions do.”, “If you do not agree that the Time To Collision (TTC) is related to the perceived quality of agent trajectories, please indicate why.", and "Do you have anything to add on this feature?"

The *Time To Closest Approach* is the time until two agents are as close as they would be if they kept their velocities constant. On the other hand, the *Distance at Closest Approach* (DCA) is the minimum distance at which two agents would cross each other if they would keep their velocities constant. Questions asked about these features: "I agree with this definition of Time To Closest Approach (TTCA).", "I agree with this definition of Distance at Closest Approach (DCA).", "This video shows some examples of TTCA and DCA combinations.", "Low Time To Closest Approach (TTCA) values negatively affect the perceived quality of trajectories.", "Small Distance at Closest Approach (DCA) values negatively affects the perceived quality of trajectories.", "I agree that TTCA values are not informative on their own, DCA is needed to interpret if a collision will happen in the near future or not.", "Across DCA and TTCA combinations, only low TTCA combined with low DCA values negatively affect the perceived quality of agent trajectories.", "If you do not agree that combinations of Time To Closest Approach (TTCA) and Distance at Closest Approach (DCA) are related to the perceived quality of animation trajectories, please specify why.", and "Do you have anything to add on this feature?"

Collisions A *collision* happens when the distance between two agents is smaller than the sum of their radius. Questions asked about this feature: "I agree with this definition of what is a collision.", "This video shows the number of collisions at each time-step.", "The number of collisions negatively affects the perceived quality of trajectories.", "If you do not agree that the number of collisions is related to quality of agent trajectories, please indicate why.", and "Do you have anything to add on this feature?"

Agents have a personal space that can *overlap* with that of other agents. Questions asked about this feature: "I agree with the definition of personal space overlap.", "This video shows personal space overlaps.", "The perceived quality is more negatively affected if the overlap between two agents is bigger.”, “The perceived quality is more negatively affected if an overlap lasts longer in time.”, “The perceived quality is more negatively affected if the “perception” of the collision is bigger (e.g. depending on the bearing angle).”, “If the personal space of two agents is overlapping during the whole trajectory, it means they are part of a group.”, “The overlap area of agents of a group does not necessarily negatively affect the perceived quality of agent trajectories.”, “If you do not agree that personal space overlap negatively affects the perceived quality of trajectories, please specify why.”, and “Do you have anything to add on this feature?”

Distributions of feature values A *Fundamental Diagram* is a figure showing the relation between the average walking speed of agents and the density based on experimental data. Questions asked about this feature: “I agree with the proposed definition for Fundamental Diagrams.”, “The range of admissible walking speed values for an agent is affected by the local density around them.”, “For a crowd motion to be perceived as of good quality, average walking speed should follow the fundamental diagram (the denser the slower).”, “If you do not agree that the relationship between density and average walking speed should be the same as in reality for a trajectory to be believable, please explain why.”, and “Do you have anything to add on this feature?”

The *distribution of trajectory lengths* represents the variation in trajectory lengths from agents to walk from their origin position to the final position. Questions asked about this feature: “I agree with the definition for distribution of trajectory lengths.”, “The plot shows the difference in length of a number of trajectories with the same origin and goal position.”, “Different lengths of trajectories from the same origin point to the same final position positively affect the perceived quality of the crowd motion.”, “If you do not agree that the distribution of trajectory lengths (compared to the one of real humans) is important, please tell us why.”, and “Do you have anything to add on this feature?”

Extra questions about variety and heterogeneity as well as relative importance "I agree that *variety* in the values of the state features (e.g. comfort speed, goal position, acceptable TTC values) is related to the *heterogeneity* of the crowd.", "The video shows a heterogeneous crowd with different values for the walking speed state feature.", "The video shows two agents behaving differently to avoid each other.", "Variety improves the perceived quality of the trajectories.", "The environment can affect the walking strategy of an agent (e.g. admissible TTC values, distance to the goal, local density, etc.).", "*Homogeneity* negatively affect the perceived quality of the trajectories.", "A crowd with variety implies that agents exhibit a behaviour that is different from the majority.", "Agents with uncommon trajectories are allowed in some cases to show specific behaviours.", "Do you have any comments on variety?", "What do you think about *coherence* within a trajectory?", and "On a scale from 1 to 7, what is the impact on the following state features in perceived quality of the trajectories of the agents of the crowds: [...]?"

Appendix B

SUPPLEMENTARY VIDEOS

An overview of the Learning Crowds modules, presented in Chapter 2, can be found in [Cabrero-Daniel, 2021a]. The information pipeline is also briefly discussed in the video. To know more about microscopic crowd simulation using costs in Velocity Space, you can watch [Pettre, 2020]. You can hear a detailed explanation about the Quality Function, QF, and its usage in the learning process in [Cabrero-Daniel, 2021b], together with some illustrative examples that accompany those in Chapter 3. These videos, that show the limitations of some steering algorithms in some scenarios, are good motivational examples for the work explained in Chapter 4. Finally, video examples of policy comparisons and sequential policy combinations are available at [Cabrero-Daniel, 2021c].

Bibliography

Alahi, A., Goel, K., Ramanathan, V., Robicquet, A., Fei-Fei, L., and Savarese, S. (2016). Social lstm: Human trajectory prediction in crowded spaces. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 961–971, Las Vegas, NV, USA. IEEE.

Alemi, A. A., Bierbaum, M., Myers, C. R., and Sethna, J. P. (2015). You can run, you can hide: The epidemiology and statistical mechanics of zombies. *Physical Review E*, 92(5).

Amirian, J., van Toll, W., Hayet, J.-B., and Pettré, J. (2019). Data-driven crowd simulation with generative adversarial networks. In *Proceedings of the 32nd International Conference on Computer Animation and Social Agents*, CASA '19, pages 7–10, New York, NY, USA. Association for Computing Machinery.

Berseth, G., Kapadia, M., Haworth, B., and Faloutsos, P. (2014). Steerfit: Automated parameter fitting for steering algorithms. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '14, pages 113–122, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

Berseth, G., Kapadia, M., Haworth, B., and Faloutsos, P. (2016). Steerfit: Automated parameter fitting for steering algorithms. In *Simulating Heterogeneous Crowds with Interactive Behaviors*, pages 229–246. AK Peters/CRC Press.

Cabrero-Daniel, B. (2021a). Automating Crowd Simulation, Chapter 2. https://youtu.be/I-Hsm3xapVM.

Cabrero-Daniel, B. (2021b). Automating Crowd Simulation, Chapter 3. https://youtu.be/w0mvJoJ-zQA.

Cabrero-Daniel, B. (2021c). Automating Crowd Simulation, Chapter 4. https://youtu.be/Xpc3vNgmdL4.

Cabrero Daniel, B., Marques, R., Hoyet, L., Pettré, J., and Blat, J. (2021). A perceptually-validated metric for crowd trajectory quality evaluation. *Proc. ACM Comput. Graph. Interact. Tech.*, 4(3).

Charalambous, P. and Chrysanthou, Y. (2014a). The PAG crowd: A graph based approach for efficient data-driven crowd simulation. *Computer Graphics Forum*, 33(8):95–108.

Charalambous, P. and Chrysanthou, Y. (2014b). The pag crowd: A graph based approach for efficient data-driven crowd simulation. *Computer Graphics Forum*, 33.

Charalambous, P., Karamouzas, I., Guy, S., and Chrysanthou, Y. (2014a). A data-driven framework for visual crowd analysis. *Computer Graphics Forum*, 33.

Charalambous, P., Karamouzas, I., Guy, S. J., and Chrysanthou, Y. (2014b). A data-driven framework for visual crowd analysis. In *Computer Graphics Forum*, volume 33, pages 41–50. Wiley Online Library.

Chattaraj, U., Seyfried, A., and Chakroborty, P. (2009). Comparison of pedestrian fundamental diagram across cultures. *Advances in Complex Systems (ACS)*, 12:393–405.

Durupinar, F., Pelechano, N., Allbeck, J., Güdükbay, U., and Badler, N. I. (2011). How the ocean personality model affects the perception of crowds. *IEEE Computer Graphics and Applications*, 31(3):22–31.

Dutra, T., Marques, R., Cavalcante-Neto, J., Vidal, C., and Pettré, J. (2017a). Gradient-based steering for vision-based crowd simulation algorithms. *Computer Graphics Forum*, 36.

Dutra, T. B., Marques, R., Cavalcante-Neto, J. B., Vidal, C. A., and Pettré, J. (2017b). Gradient-based steering for vision-based crowd simulation algorithms. In *Computer graphics forum*, volume 36, pages 337–348. Wiley Online Library.

E. Hart, P., J. Nilsson, N., and Raphael, B. (1972). A formal basis for the heuristic determination of minimum cost paths. 37:28–29.

Gupta, A., Johnson, J., Fei-Fei, L., Savarese, S., and Alahi, A. (2018). Social GAN: Socially acceptable trajectories with generative adversarial networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2255–2264, Las Vegas, NV, USA. IEEE.

Guy, S. J., Chhugani, J., Curtis, S., Dubey, P., Lin, M., and Manocha, D. (2010). Pledestrians: A least-effort approach to crowd simulation. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '10, page 119–128, Goslar, DEU. Eurographics Association.

Guy, S. J., Van Den Berg, J., Liu, W., Lau, R., Lin, M. C., and Manocha, D. (2012). A statistical similarity measure for aggregate crowd dynamics. *ACM Transactions on Graphics (TOG)*, 31(6):1–11.

Helbing, D. and Molnár, P. (1995). Social force model for pedestrian dynamics. *Physical Review E*, 51(5):4282–4286.

Hoyet, L., Olivier, A.-H., Kulpa, R., and Pettré, J. (2016). Perceptual effect of shoulder motions on crowd animations. *ACM Transactions on Graphics*, 35:1–10.

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

Jelić, A., Appert-Rolland, C., Lemercier, S., and Pettré, J. (2012). Properties of pedestrians walking in line: Fundamental diagrams. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 85:036111.

Kapadia, M., Berseth, G., Singh, S., Reinman, G., and Faloutsos, P. (2016). Scenario space: characterizing coverage, quality, and failure of steering algorithms. In *Simulating Heterogeneous Crowds with Interactive Behaviors*, pages 193–210. AK Peters/CRC Press.

Kapadia, M., Singh, S., Allen, B., Reinman, G., and Faloutsos, P. (2009). Steerbug: An interactive framework for specifying and detecting steering behaviors. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '09, pages 209–216, New York, NY, USA. Association for Computing Machinery.

Kapadia, M., Wang, M., Singh, S., Reinman, G., and Faloutsos, P. (2011). Scenario space: Characterizing coverage, quality, and failure of steering algorithms. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '11, pages 53–62, New York, NY, USA. Association for Computing Machinery.

Kappen, H. J. and Ruiz, H. C. (2016). Adaptive importance sampling for control and inference. *Journal of Statistical Physics*, 162(5):1244–1266.

Karamouzas, I., Heil, P., Beek, P., and Overmars, M. H. (2009). A predictive collision avoidance model for pedestrian simulation. In *Proceedings of the 2nd International Workshop on Motion in Games*, MIG '09, page 41–52, Berlin, Heidelberg. Springer-Verlag.

Karamouzas, I. and Overmars, M. (2011). Simulating and evaluating the local behavior of small pedestrian groups. *IEEE Transactions on Visualization and Computer Graphics*, 18(3):394–406.

Karamouzas, I., Skinner, B., and Guy, S. J. (2014). Universal Power Law Governing Pedestrian Interactions. *Physical Review Letters*, 113(23):238701.

Karamouzas, I., Sohre, N., Hu, R., and Guy, S. J. (2018a). Crowd space: A predictive crowd analysis technique. *ACM Transactions on Graphics*, 37(6).

Karamouzas, I., Sohre, N., Hu, R., and Guy, S. J. (2018b). Crowd space: a predictive crowd analysis technique. *ACM Transactions on Graphics (TOG)*, 37(6):1–14.

Kulpa, R., Olivier, A.-H., Ondřej, J., and Pettré, J. (2011). Imperceptible relaxation of collision avoidance constraints in virtual crowds. *ACM Transactions on Graphics*, 30(6):1–10.

Lee, K. H., Choi, M. G., Hong, Q., and Lee, J. (2007). 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, page 109–118, Goslar, DEU. Eurographics Association.

Lerner, A., Chrysanthou, Y., and Lischinski, D. (2007). Crowds by example. *Computer Graphics Forum*, 26(3):655–664.

Martinez-Gil, F., Lozano, M., and Fernández, F. (2014). Marl-ped: A multi-agent reinforcement learning based framework to simulate pedestrian groups. *Simulation Modelling Practice and Theory*, 47:259 – 275.

McDonnell, R., Larkin, M., Dobbyn, S., Collins, S., and O'Sullivan, C. (2008). Clone attack! perception of crowd variety. *ACM Transactions on Graphics*, 27.

McDonnell, R., Larkin, M., Hernandez, B., Rudomin, I., and O'Sullivan, C. (2009). Eye-catching crowds: Saliency based selective variation. *ACM Transactions on Graphics*, 28.

Molina, E., Ríos, A., and Pelechano, N. (2021). The impact of animations in the perception of a simulated crowd. In Magnenat-Thalmann, N., Interrante, V., Thalmann, D., Papagiannakis, G., Sheng, B., Kim, J., and Gavrilova, M., editors, *Advances in Computer Graphics*, pages 25–38, Cham. Springer International Publishing.

Moussaïd, M., Helbing, D., and Theraulaz, G. (2011). How simple rules determine pedestrian behavior and crowd disasters. *Proceedings of the National Academy of Sciences*, 108(17):6884–6888.

Ondřej, J., Pettré, J., Olivier, A.-H., and Donikian, S. (2010). A synthetic-vision based steering approach for crowd simulation. *ACM Transactions on Graphics*, 29(4):123.

Paris, S., Pettré, J., and Donikian, S. (2007). Pedestrian reactive navigation for crowd simulation: a predictive approach abstract. *Comput. Graph. Forum*, 26:665–674.

Pettre, J. (2020). Generalized microscropic crowd simulationusing costs in velocity space. https://youtu.be/5JkwWQeoJno.

Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. *SIGGRAPH Comput. Graph.*, 21(4):25–34.

Thalmeier, D., Gómez, V., and Kappen, H. J. (2017). Action selection in growing state spaces: control of network structure growth. *Journal of Physics A: Mathematical and Theoretical*, 50(3):034006.

Todorov, E. (2006). Linearly-solvable markov decision problems. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*, NIPS'06, page 1369–1376, Cambridge, MA, USA. MIT Press.

Todorov, E. (2009). Efficient computation of optimal actions. *Proceedings of the national academy of sciences*, 106(28):11478–11483.

Treuille, A., Cooper, S., and Popović, Z. (2006). Continuum crowds. *ACM Transactions on Graphics*, 25(3):1160–1168.

Turnwald, A., Eger, S., and Wollherr, D. (2015). Investigating similarity measures for locomotor trajectories based on the human perception of differences in motions. In *2015 IEEE International Workshop on Advanced Robotics and its Social Impacts (ARSO)*, pages 1–6, Lyon, France. 2015 IEEE International Workshop on Advanced Robotics and its Social Impacts (ARSO).

van den Berg, J., Guy, S. J., Lin, M., and Manocha, D. (2011). Reciprocal n-body collision avoidance. In Pradalier, C., Siegwart, R., and Hirzinger, G., editors, *Robotics Research*, pages 3–19, Berlin, Heidelberg. Springer Berlin Heidelberg.

van den Berg, J., Ming Lin, and Manocha, D. (2008). Reciprocal velocity obstacles for real-time multi-agent navigation. In *2008 IEEE International Conference on Robotics and Automation*, pages 1928–1935, New York, NY. IEEE.

van Toll, W., Braga, C., Solenthaler, B., and Pettré, J. (2020a). Extreme-density crowd simulation: Combining agents with smoothed particle hydrodynamics. In *Motion, Interaction and Games*, MIG '20, New York, NY, USA. Association for Computing Machinery.

van Toll, W., Grzeskowiak, F., Gandía, A. L., Amirian, J., Berton, F., Bruneau, J., Daniel, B. C., Jovane, A., and Pettré, J. (2020b). Generalized microscropic crowd simulation using costs in velocity space. In *Symposium on Interactive 3D Graphics and Games*, I3D '20, New York, NY, USA. Association for Computing Machinery.

van Toll, W. and Pettré, J. (2021). Algorithms for Microscopic Crowd Simulation: Advancements in the 2010s. *Computer Graphics Forum*, 40(2).

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

Wolinski, D., J. Guy, S., Olivier, A.-H., Lin, M., Manocha, D., and Pettré, J. (2014a). Parameter estimation and comparative evaluation of crowd simulations. *Comput. Graph. Forum*, 33(2):303–312.

Wolinski, D., J. Guy, S., Olivier, A.-H., Lin, M., Manocha, D., and Pettré, J. (2014b). Parameter estimation and comparative evaluation of crowd simulations. In *Computer Graphics Forum*, volume 33, pages 303–312. Wiley Online Library.

Yang, S., Li, T., Gong, X., Peng, B., and Hu, J. (2020). A review on crowd simulation and modeling. *Graphical Models*, 111:101081.

Zhao, M., Cai, W., and Turner, S. (2017). Clust: Simulating realistic crowd behaviour by mining pattern from crowd videos: Clust. *Computer Graphics Forum*, 37.