SteerFit: Automated Parameter Fitting for Steering Algorithms

Glen Berseth, Petros Faloutsos, Mubbasir Kapadia

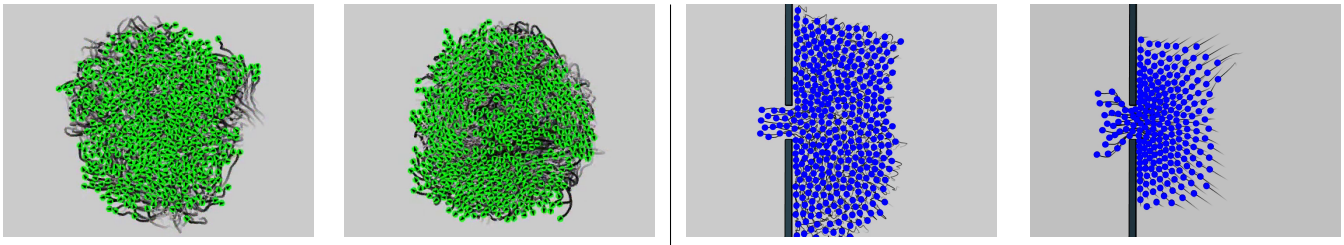
Technical Report EECS-2014-02

March 1 2014

Department of Electrical Engineering and Computer Science
4700 Keele Street, Toronto, Ontario M3J 1P3 Canada

SteerFit: Automated Parameter Fitting for Steering Algorithms

Glen Berseth (glenpb@cse.yorku.ca)
Petros Faloutsos (pfal@cse.yorku.ca)
Mubbasir Kapadia (mubbasir.kapadia@gmail.com)



(a) Scenario I : default (b) Scenario I: optimal time (c) Scenario II: default (d) Scenario II: optimal effort

Figure 1: *Comparison of simulations using default [(a), (c)] and optimized [(b), (d)] parameters. Left: Agents are initially in a circle with anti-diametric goals. The **ORCA** algorithm, optimized to reduce time-to-completion, completes the task* **twice** *as fast as its default configuration and exhibits a less turbulent pattern. Right: The **SF** algorithm, optimized to minimize effort, requires a* **third** *of the energy spent by its default configuration, and produces a faster room evacuation with tighter packing and smoother motion.*

Abstract

In the context of crowd simulation, there are a diverse set of algorithms that model *steering*, the ability of an agent to navigate between spatial locations, while avoiding static and dynamic obstacles. The *performance* of steering approaches, both in terms of quality of results and computational efficiency, depends on internal parameters that are manually tuned to satisfy application-specific requirements. This paper investigates the effect that these parameters have on an algorithm's performance. Using three representative crowd simulators and a set of established performance criteria, we perform a number of large scale optimization experiments that optimize an algorithm's parameters for a range of objectives. Our experiments show that parameter fitting has a significant impact on the visual fidelity of the simulations produced, the crowd's localized and macroscopic behaviours, as well as the computational efficiency of the steering algorithm. For example, our method automatically finds optimal parameters to minimize turbulence at bottlenecks, reduce building evacuation times, produce emergent patterns, and increase the computational efficiency of an algorithm, in one case by a factor of two. Our study includes an in-depth statistical analysis of the correlations between algorithmic parameters, and performance criteria. The proposed methodology can be applied to any steering algorithm using any set of performance criteria. To our knowledge, this is the first attempt at studying the relationship between a steering algorithm's parameters and its performance, based on parameter fitting.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation;

Keywords: steering,peformance analysis, parameter tuning

1 Introduction

Simulating groups of autonomous virtual humans (agents) in complex, dynamic environments is an important issue for many practical applications. A key aspect of autonomous agents is their ability to navigate (steer) from one location to another in their environment, while avoiding collisions with static as well as dynamic obstacles. The requirements of a steering approach differ significantly between applications and application domains. For example, computer games are generally concerned with minimizing computational overhead, and often trade off quality for efficiency, while evacuation studies often aim to generate plausible crowd behaviour that minimizes evacuation times while maintaining order.

There is no definitive solution to the steering problem. Most of the established methods are designed for specific classes of situations (scenarios), and make different trade-offs between quality and efficiency. The fine balance between these often competing performance criteria is governed by algorithm specific parameters that are exposed to the user. Some of these parameters have intuitive direct effects. For example, the radius of a *comfort zone* affects how close agents may come to each other, while the *neighbour horizon* limits the distance from an agent within which other agents are considered during steering, which significantly influences both the predictive power and computational efficiency of the associated method. However, even when the parameters are fairly intuitive, their combined effect, or their effect on the macroscopic behaviour of a large crowd is not always easy to predict. For this reason, the inverse question is particularly interesting. That is, given a performance criterion (metric), or a trade-off between performance metrics, or pattern of behaviour, can we automatically select the parameter values of a steering algorithm that will produce the desired effect? This is a timely and important question, and the main focus of our work.

We present a methodology for automatically fitting the parameters of a steering algorithm to minimize *any combination* of performance metrics, across *any* set of environment benchmarks in a general, model-independent fashion. Using our approach, a steering algorithm can be optimized for success, quality with respect to distance, time, or energy consumption of an agent, its computational performance, similarity to ground truth, user-defined custom metrics, or a weighted combination of any of the above. Optimizing an algorithm's parameters across a representative set of challenging scenarios provides a parameter set that generalizes to many situations. A steering approach may also be fitted to a specific benchmark (e.g., a game level), or a benchmark category (e.g., evacuations) to hone its performance for a particular application. Our method is model-independent and can be applied to any steering algorithm using any set of performance criteria. To our knowledge, this is the first study of the relationship between a steering algo-

rithm's parameters and its performance, based on parameter fitting.

We demonstrate our proposed methodology using three established agent-based algorithms: (1) **ORCA**: a predictive technique that uses reciprocal velocity obstacles for collision avoidance [van den Berg et al. 2008a], (2) **PPR**: a hybrid approach that uses rules to combine reactions, predictions, and planning [Singh et al. 2011], and (3) **SF**: a variant of the social forces method for crowd simulation [Helbing et al. 2000]. We thoroughly study these algorithms and compute their optimal parameter configurations for different metric combinations on a representative scenario set of local agent interactions and large-scale benchmarks. For example, our method automatically finds optimal parameters to minimize turbulence at bottlenecks, reduce building evacuation times, produce emergent patterns, and increase the computational efficiency of an algorithm, in one case by a factor of two. Cross-validation shows that optimal parameter values generalize on average across scenarios that were not part of the test set. Our study includes an in-depth statistical analysis of correlations between algorithmic parameters, and performance criteria, however, because of space limitations most of it can be found in the supplemental material.

The main contributions of this paper are twofold. First, we propose a model-independent solution that automatically fits a steering algorithm's parameters to maximize its performance, and we demonstrate its effectiveness with a use-case analysis of three popular crowd simulation techniques. Second, we show that parameter fitting over appropriate test sets can be an effective tool for studying and analyzing a steering algorithm, and in particular for gaining insights on the effect its internal parameters on the algorithm's performance.

2 Related Work

Since the seminal work of [Reynolds 1987; Reynolds 1999], crowd simulation has been studied from many different perspectives. We refer the readers to comprehensive surveys [Pelechano et al. 2008; Huerre et al. 2010; Thalmann and Musse 2013] and present a broad review below.

Continuum-based techniques [Treuille et al. 2006; Narain et al. 2009] model the characteristics of the crowd flow to simulate macroscopic crowd phenomena. Particle-based approaches [Reynolds 1987; Reynolds 1999] model agents as particles and simulate crowds using basic particle dynamics. The social force model [Helbing et al. 2005; Brogan and Hodgins 1997] simulates forces such as repulsion, attraction, friction and dissipation for each agent to simulate pedestrians. Rule-based approaches [Lamarche and Donikian 2004; Pelechano et al. 2007; Sud et al. 2007; van den Berg et al. 2008b] use various conditions and heuristics to identify the exact situation of an agent. Data-driven methods [Lee et al. 2007; Lerner et al. 2007; Ju et al. 2010; Metoyer and Hodgins 2003; Musse et al. 2007; Patil et al. 2011] use existing video or motion capture data to derive steering choices that are then used in virtual worlds, and recent work [Ondřej et al. 2010] demonstrates a synthetic vision-based approach to steering. The works of [Paris et al. 2007; van den Berg et al. 2011] use predictions to perform steering in environments populated with dynamic threats. Predicting potential threats ahead of time results in more realistic steering behaviours.

Crowd Evaluation. There has been a growing recent trend to use statistical analysis in the evaluation and analysis of crowd simulations. The work in [Lerner et al. 2010] adopts a data-driven approach to evaluating crowds by measuring its similarity to real world data. Singh *et al.* [2009] proposes a compact suite of manually defined test cases that represent different steering challenges and a rich set of derived metrics that provide an empirical measure of the performance of an algorithm. Recent extensions [Kapadia et al. 2011] propose a representative sampling of challenging scenarios that agents encounter in crowds to compute the coverage of the algorithm, and the quality of the simulations produced. Density measures [Lerner et al. 2010] and fundamental diagram-based comparisons [Seyfried et al. 2010] use aggregate metrics for quantifying similarity. The work in [Guy et al. 2012; Pettré et al. 2009] measures the ability of a crowd simulator to emulate the behavior of a real crowd dataset by measuring its divergence from ground truth. [Musse et al. 2012] presents a histogram-based technique to quantify the global flow characteristics of crowds.

Perceptual experiments rely on real people to quantify the plausibility of simulated crowds. The work in [Ennis et al. 2011] measures the effects of external factors such as camera position on the perceived fidelity of crowds. The work in [McDonnell et al. 2008] explores the perception of variety in appearance and motion in crowds. Kulpa *et al.* [2011] measures the perceptual fidelity of relaxing collisions in crowds.

Parameter Optimization. Parameter fitting is widely used in visual effects [Bruckner and Moller 2010] to automate the tuning of model parameters to meet certain user-defined criteria. The resulting optimization problems tend to involve non-convex, and highly-dimensional spaces. For these problems evolutionary strategies are preferred because they generally have less parameters to tune and do not require the computation of derivatives. Such techniques have been successfully demonstrated on a diverse set of application domains [Ha et al. 2013; Wang et al. 2010]. By selecting the right set of parameters, researchers have shown improvements in a crowd simulator's ability to match recorded crowd data [Pettré et al. 2009; Lemercier et al. 2012; Pellegrini et al. 2009; Davidich and Koester 2011].

Although prior work has entertained the notion of parameter tuning in certain specific cases, a comprehensive study and a methodology to identify the relation between a steering algorithm's parameters and performance objectives has not been performed yet. Such a study is an important and timely next step, and it is the main focus of this paper.

3 Parameter Fitting Methodology

We present an optimization based framework for automatically fitting the parameters $\mathbf{v} \in \mathbb{V}$ of an algorithm, $A_{\mathbf{v}}$. Our framework automatically selects optimal parameter values $\mathbf{v}^* \in \mathbb{V}$ such that the performance of $A_{\mathbf{v}^*}$ minimizes certain performance criteria, over a set of benchmarks (test set). The next sections describe the elements involved in this problem and our approach to solving it.

3.1 Steering Algorithms

Our approach can be applied to any steering algorithm. For demonstration reasons, we use the following established algorithms that model different steering approaches.

1. **PPR**. [Singh et al. 2011] presents a hybrid framework that combines reaction, prediction and planning. It is an example of a rule based method for agent based steering and has 38 independent parameters. For example, *avoidance-turn-rate* defines the turning rate adjustment speed in proportion to the typical speed and *query-radius* controls the radius around an agent that **PPR** uses to predict collisions with other objects and agents.

2. **ORCA**. [van den Berg et al. 2011] is a very popular method that uses optimal reciprocal collision avoidance to efficiently

steer agents in large-scale crowds. A subset of its independent parameters are: *max-neighbors*, the maximum number of nearby agents that an agent will take into consideration when making steering choices, *max-speed*, the maximum speed that an agent may travel with, and *time-horizon*, the minimal time for which an agent's computed velocity is safe with respect to other agents.

3. **SF**. [Helbing et al. 2000] uses hypothetical social forces for resolving collisions between interacting agents in dense crowds. In addition to general parameters similar with the other methods, each social force model has associated parameters that govern its relative influence. The effect of some of these parameters on the emergent dynamics of a crowd simulation has been studied before.

The complete set of parameters and their default values for **ORCA** are shown in Table 4. The other two algorithms have much larger parameters sets which can be found in the supplemental material.

3.2 Test Sets

In the context of parameter fitting, we can employ a general test set that tries to cover a wide range of situations, for example the one proposed in [Kapadia et al. 2011], or a specific test set with situations that appear in a particular application domain, for example building evacuations. It is also worth noting that certain performance metrics may have more meaning for specific test sets. For example, computational efficiency is more meaningful for situations that involve sufficiently large numbers of agents.

For demonstration reasons, we focus our study on the following test sets.

Large Scale Sets. $\mathcal{S}$ contains most of the large-scale benchmarks in Table 1 that define large environments with many agents. $\mathcal{S}^v$ is a set of similar but different large-scale benchmarks that will be used to validate the results of parameter optimization on previously unseen cases (cross validation).

Benchmark	# Agents	Description
`Random`	1000	Random agents in open space.
`Forest`	500	Random agents in a forest.
`Urban`	500	Random agents in an urban environment.
`Hallway`	200	Bi-directional traffic in a hallway.
`Free Tickets`	200	Random agents to same goal, then disperse.
`Bottleneck`	1000	Tight bottleneck.
`Bottleneck evac`	200	Evacuation through a narrow door.
`Concentric circle`	250	circle with target on opposite side.
`Concentric circle`	500	circle with target on opposite side.
`Hallway`	400	4-way directional traffic.

Table 1: *Large scale benchmarks.*

Representative Set. The representative scenario set, $\mathcal{R}$, includes 5000 samples of a wide range of local interactions. It is designed to include challenging local scenarios and to exclude trivial or invalid cases. We construct it in a fashion similar to [Kapadia et al. 2011], following these general guidelines: (a) The reference agent is placed near the center of the scenario, (b) agent targets are placed at the environment boundary, and (c) non-reference agents are distributed at locations that maximize the likelihood that their static paths will intersect the reference agent's static path to its target. We use the same method to generate another set of the same size, $\mathcal{R}^v$, for cross-validation.

Combined Test Set. The union of the large scale set, $\mathcal{S}$, and the representative set, $\mathcal{R}$, $\mathcal{T} = \mathcal{S} \cup \mathcal{R}$ is the main test set that we use for algorithm analysis and parameter fitting in a statistically significant general fashion.

Combined Validation Set. Similarly, the combined cross-validation set is $\mathcal{T}^v = \mathcal{S}^v \cup \mathcal{R}^v$.

Custom Scenario Set. A user can specify a subset of scenarios in $\mathcal{T}$ or even design custom benchmarks to focus the parameter fitting on application-specific requirements. Random permutations in the environment configuration and agent placement can generate multiple samples of a custom benchmark category. For example, one can create a set of test cases that capture two-way traffic in orthogonally crossing hallways as is common in large buildings.

Ground Truth Test Set. The are a few publicly available data sets of recorded crowd motion, which can be used to test the ability of steering algorithms to match real world data. We use a ground truth test set $\mathcal{G}$, published by [Seyfried et al. 2010] for our experiments.

3.3 Normalized Performance Measures

Given an appropriate test set, we want to compute normalized quantities (metrics) that characterize important aspects of a steering algorithm's performance. Recently a number of intuitive performance metrics have been proposed that include: (a) the fraction of scenarios that an algorithm is unable to solve in a representative set of scenarios, (b) quality measures with respect to distance travelled, total time taken, or energy consumption of an agent, (c) computational performance of the algorithm, and (d) statistical similarity with respect to ground truth. The specific metrics that we use in our experiments are briefly described below. For more details see [Kapadia et al. 2011; Guy et al. 2010; Guy et al. 2012]. One can also define custom metrics to meet application-specific requirements.

Failure rate. The coverage $c(A_{\mathbf{v}})$ of a steering algorithm $A_{\mathbf{v}}$ over a test set, $\mathcal{T}$, is the ratio of scenarios that a steering algorithm successfully completes in $\mathcal{T}$. An algorithm successfully completes a particular scenario if the reference agent reaches its goal within a conservative time limit without any collisions, and the total number of collisions among non-reference agents is less than the number of agents in the scenario. We define the failure rate as the complement of coverage $d(A_{\mathbf{v}}) = 1 - c(A_{\mathbf{v}})$. It captures the ratio between the number of scenarios not successfully solved and the total number of scenarios in $\mathcal{T}$, and it has obvious bounds in $\mathcal{T}$.

Distance Quality. For a single small scale scenario, s, we define the distance quality metric, $q^{\mathrm{d}}(A_{\mathbf{v}})$ of an algorithm $A_{\mathbf{v}}$ as the complement of the ratio between the length of an ideal optimal path, o_s^d, and the length of the path that the reference agent followed, a_s^d:

$$q^{\mathrm{d}}(A_{\mathbf{v}}) = 1 - \frac{o_s^d}{a_s^d}. \tag{1}$$

The ideal optimal path is the shortest static path from the agent's initial position to its goal after line-of-sight smoothing [Pinter 2001]. If the algorithm does not successfully complete the scenario then the associated distance quality metric is set to the worst-case value of 1. For a large-scale scenario we compute $q^{\mathrm{d}}(A_{\mathbf{v}})$ as the average over all agents, and for a set of scenarios, we computed it as the average over the set.

Time Quality. It characterizes how much longer the reference agent took to reach its goal compared to an ideal optimal time. The ideal optimal time for a single scenario,o_s^t, corresponds to the agent reaching its goal when moving with its desired velocity along the ideal optimal path. Similarly to the distance quality metric, time quality is defined as:

$$q^{\mathrm{t}}(A_{\mathbf{v}}) = 1 - \frac{o_s^t}{a_s^t}, \tag{2}$$

where a_s^t is the time it took the agent to reach its goal in the scenario s. If the algorithm does not successfully complete the scenario then the metric is set to the worst-case value of 1. For large scale scenarios this metric represents the average over all agents, and for a set of test cases the average over the set.

PLE Quality. The principal of least effort characterizes the energy expenditure of a reference agent over a path traveled [Guy et al. 2010] as follows:

$$p^e = m \int_{t_{start}}^{t_{end}} (e_s + e_w)|\mathbf{v}|^2 dt, \tag{3}$$

where e_s and e_w are commonly used energy terms for the average person [Whittle 2007], and the mass, m is set to 1 in our experiments. The PLE quality metric, $q^{\mathrm{e}}(A_{\mathbf{v}})$, is computed similar to the other metrics as follows:

$$q^{\mathrm{e}}(A_{\mathbf{v}}) = 1 - \frac{o^e}{a^e}, \tag{4}$$

where o_s^e = optimal-path-length $\times$ $(e_s + e_w)$ is the ideal optimal effort and a^e the actual effort of the agent. If the algorithm does not successfully complete the scenario the metric is set to the worst case value of 1. For many agents and/or test cases the metric is computed in the average sense.

Computational Efficiency. The computational efficiency $e(A_{\mathbf{v}})$ metric is the average CPU time consumed by all agents in all scenarios in a test set $\mathcal{S}$. Unlike the above normalized metrics, it is not straightforward to provide an ideal upper bound for e. To provide a basis for normalization, we assume that 10% of all computational resources are allocated to the steering algorithm. Hence, the maximum time allocated to a steering algorithm every frame is n_{des}^{-1} seconds for a desired framerate of n_{des} fps. For every scenario s, the maximum time t_{max}^s allocated to every steering agent per frame is $(N \cdot n_{\text{des}})^{-1}$ seconds, where N is the number of agents in s. Let t_{avg}^s be the average time spent per frame for all agents to reach a steering decision. The average computational efficiency e over a test set $\mathcal{S}$ is computed as follows:

$$e(A_{\mathbf{v}}) = 1 - \frac{\sum_{s \in \mathcal{S}} e_{\mathrm{s}}(A_{\mathbf{v}})}{|\mathcal{S}|}, \quad e_{\mathrm{s}}(A_{\mathbf{v}}) = \frac{t_{\text{max}}^{\mathrm{s}}}{t_{\text{avg}}^{\mathrm{s}}} \tag{5}$$

where $e_{\mathrm{s}}(A_{\mathbf{v}})$ is the efficiency of $A_{\mathbf{v}}$ for a particular scenario s, and $|\mathcal{S}|$ is the cardinality of the test set $\mathcal{S}$.

The desired framerate, n_{des}, provides an ideal upper bound for efficiency, analogous to the ideal upper bounds of the other metrics, and allows us to define a normalized efficiency metric. Normalized metrics can be combined more intuitively into optimization objectives in the forthcoming analysis. Alternatively, we could set the desired framerate to a very high value for all algorithms and deal with scaling issues later.

Similarity to Ground Truth. In addition to quantitatively characterizing the performance of a steering algorithm, we can also measure its ability to match ground truth. We compute a simulation-to-data similarity measure $g(A_{\mathbf{v}}, \mathcal{G})$ [Guy et al. 2012] which computes the entropy measurement of the prediction errors of algorithm $A_{\mathbf{v}}$ relative to a given example dataset, such as the test set $\mathcal{G}$ defined in Section 3.2.

3.4 Parameter Optimization

Given a set of performance metrics such as the ones defined in Section 3.3, $\mathbb{M} = \langle d, q^{\mathrm{d}}, q^{\mathrm{t}}, q^{\mathrm{e}}, e \rangle$, we can define an objective function as a weighted combination of these metrics:

$$f(A_{\mathbf{v}}, \mathbf{w}) = \sum_{m_i \in \mathbb{M}} w_i \cdot m_i, \tag{6}$$

where $\mathbf{w} = \{w_i\}$ contains the weights which determine the relative influence of each individual metric. By choosing different sets of metrics and associated relative weights, we can define custom objectives.

For a steering algorithm $A_{\mathbf{v}}$ with internal parameters $\mathbf{v} \in \mathbb{V}$, a set of test set, and a desired objective $f(A_{\mathbf{v}}, \mathbf{w})$, our goal is to find the optimal parameter values $\mathbf{v}_{\mathbf{w}}^*$ that minimize the objective over the test set. Mathematically this can be formulated as a minimization problem:

$$\mathbf{v}_{\mathbf{w}}^* = \arg\min_{\mathbf{v} \in \mathbb{V}} f(A_{\mathbf{v}}, \mathbf{w}). \tag{7}$$

This is generally a non-linear and non-convex optimization problem for the independent parameters, $\mathbf{v} \in \mathbb{V}$. The Covariance Matrix Adaptation Evolution Strategy technique (CMA-ES) [Hansen and Ostermeier 1996; Hansen 2011] is one of the many methods that can solve such problems. We chose CMA-ES because it is straightforward to implement, it can handle ill-conditioned objectives and noise, but, more importantly, it has support for mixed integer optimization. It is very common for steering algorithms to involve integer parameters that are discontinuous.

The CMA-ES algorithm terminates when the objective converges to a minimum, when very little improvement is made between iterations or after a fixed number of iterations. Each iteration evaluates the objective for 8 parameter samples. In most of our experiments the algorithm converged within 1000 evaluations.

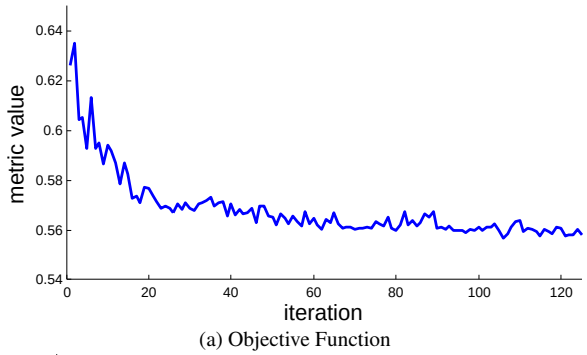
For practical reasons we have to limit the range of the algorithm's parameters. The bounds are chosen separately for each parameter based on intuition, physical interpretation of the parameter, or default values provided by the algorithm's creators. Limiting the values of an algorithm's parameters transforms the problem of optimizing over a unbounded domain to a bounded one, which generally decreases the number of iterations needed for the optimization to converge. The supplementary document reports the chosen minimum and maximum bounds for each parameter of **PPR**, **ORCA** and **SF** for reference.

Example: Figure 2 illustrates an optimization process. The parameters of **ORCA** $\mathbf{v}$ = {max speed, neighbour distance, time horizon, time horizon obstacles, max neighbours} are optimally fitted to an equally weighted combination of metrics over the test set $\mathcal{T}$. After 45 iterations the optimization converges to approximately 10% better objective value. A quick observation shows that the optimization has reduced the number of neighbours that the algorithm considers for each agent, max neighbours, from 10 to 2.

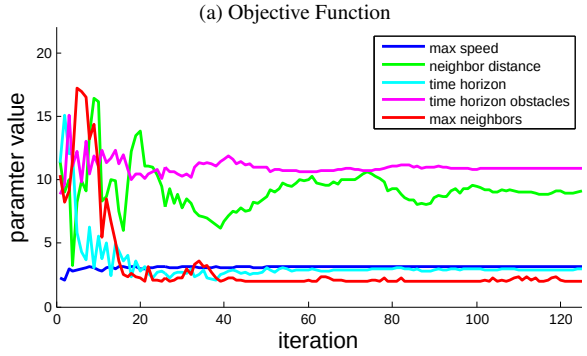
4 Large Scale Study

In this section, we study the effects of parameter fitting using the combined test sets, $\mathcal{T}$, $\mathcal{T}^v$. Our goal is to identify whether parameter fitting has a significant effect in a statistical sense, and to gain some understanding on the relation between algorithmic parameters and performance. The next section focuses on specific cases and examples.

For each of the three algorithms, **PPR**, **ORCA** and **SF**, we compute the optimal parameter values for each of the five metrics, failure rate $d(A_{\mathbf{v}})$, distance quality $q^{\mathrm{d}}(A_{\mathbf{v}})$, time quality $q^{\mathrm{t}}(A_{\mathbf{v}})$, PLE $q^{\mathrm{e}}(A_{\mathbf{v}})$, efficiency $e(A_{\mathbf{v}})$, as well as a uniform combination of these metrics, u, over the entire combined set, $\mathcal{T}$. For comparison, we also



(a) Objective Function



(b) **ORCA** parameter values

Figure 2: *Optimizing **ORCA** parameters to minimize the uniformly weighted combination of metrics over the test set $\mathcal{T}$. Each iteration is equal to 8 metric evaluations. As can be seen convergence occurs around 45 iterations.*

compute the same metrics for all algorithms with their parameters set to default values. The results are shown in Table 2.

f

Looking at the table, we notice that with the default parameters **PPR**, **ORCA** and **SF** cannot solve respectively 39%, 56%, and 26% of the sampled scenarios. Using the optimal parameter selection for **PPR**, the algorithm only fails in 9% of the scenarios, an improvement of 30% over the default settings. The significant optimization in time quality, $q^{\mathrm{t}}(A_{\mathbf{v}})$, for the **PPR** algorithm is impressive as well. **ORCA** does not show significant results over the metrics with the exception of q^{t}. On the other hand **SF** shows impressive improvement over most metrics, achieving the smallest failure rate d and the minimum energy expenditure, q^{e}. The supplementary document provides the corresponding optimal parameter values for these experiments.

Validation. We verify the statistical significance of the results shown in Table 2 in two ways. First, we observe that for all three algorithms and for all the scenarios in the test set, $\mathcal{T}$, which are more than 5000, the optimization did not time out but actually converged to at least a local minimum. In the context of numerical optimization that is a sufficiently strong indication that the results are not random. Second, we perform a cross validation study on an equally large test set of similar, but previously unseen scenarios, $\mathcal{T}^v$. Comparing the values of the objectives for the default parameters of the algorithms, and for the optimized ones, we see that the optimized parameters on average perform better even on scenarios that were not used during the optimization. The full cross-validation study can be found in the supplementary document.

Having established the validity of the results we can now identify potential relationships, correlations and insights related to the parameters, the performance metrics and the behaviour of an algorithm.

$A_{\mathbf{v}}$	$\mathbf{v}$	d	q^{d}	q^{t}	q^{e}	e	u
PPR	DEF	0.39	0.49	0.56	0.53	0.96	0.58
	OPT	0.09	0.20	0.07	0.28	0.89	0.34
ORCA	DEF	0.56	0.61	0.56	0.67	0.75	0.62
	OPT	0.47	0.56	0.30	0.63	0.67	0.55
SF	DEF	0.26	0.41	0.50	0.45	0.87	0.50
	OPT	0.04	0.20	0.29	0.23	0.78	0.32

Table 2: *Comparison of failure rate d, distance quality q^{d}, time quality q^{t}, effort quality q^{e}, computational efficiency e, and a uniform combination of all metrics u for the three steering algorithms using: (a) DEF: default parameter values and (b) OPT: optimal parameter values.*

Relationship between performance metrics. It is interesting to investigate whether relationships exist between performance metrics. For example, does optimizing for distance quality, q^{d}, also optimizes time quality, q^{t}? To answer such questions, we compute the value of each metric obtained with parameter values that are optimized for the other metrics, Table 3.

We observe that the optimal parameters for distance quality, $q^{\mathrm{d}}(A_{\mathbf{v}})$, produce near-optimal results for failure rate, $d(A_{\mathbf{v}})$ in the case of both **PPR** and **ORCA**. However, the opposite does not hold true. Optimizing for failure rate, $d(A_{\mathbf{v}})$, does not yield optimal results for distance quality,$q^{\mathrm{d}}(A_{\mathbf{v}})$.

A correlation analysis produces a clearer picture of the dependencies across metrics for a given algorithm. Because of space limitations we only discuss the correlation between metrics in the case of **ORCA**. We generate 1000 samples in the parameter space of **ORCA**, and use them to compute each metric over the 5008 cases in $\mathcal{T}$. We then compute the Spearman correlation coefficients between pairs of metrics, shown in Table 6. We can identify the following correlations:

1. A weak negative correlation between computational efficiency, $e_{\mathrm{s}}(A_{\mathbf{v}})$, and the other metrics.
2. A strong negative correlation between time quality, $q^{\mathrm{t}}(A_{\mathbf{v}})$, and effort quality, $q^{\mathrm{e}}(A_{\mathbf{v}})$, which in general can be expected, as faster motion requires more energy to achieve.
3. A weak positive correlation between time quality, $q^{\mathrm{t}}(A_{\mathbf{v}})$, and distance quality, $q^{\mathrm{d}}(A_{\mathbf{v}})$. This is also expected since a shortest path on average results in shorter completion time.

Relationship between parameters and metrics. It is interesting to identify which parameters change in relation to the objectives, and study the trade-offs that the algorithms essentially make with these changes. Because of space limitations we present the relevant data for **ORCA** in Tables 4and 5, and refer the user to the supplemental material for the supporting data on the other two algorithms. However, we present a few observations for all three algorithms to demonstrate that our methodology reveals interesting introspective insights for more than one algorithm.

To optimize failure rate, $d(A_{\mathbf{v}})$, **PPR** chooses very high values for predictive avoidance parameters and minimal values for speed thresholds, and trades off performance by selecting higher spatial querying distances. This results in slow moving agents that can manoeuvre in tight spaces.When optimizing distance quality $q^{\mathrm{d}}(A_{\mathbf{v}})$ **PPR** changes different speed multipliers in an attempt to minimize any extra distance covered around corners. To improve computational efficiency,e, **PPR** minimizes parameters that would trigger

	ORCA						PPR						SF					
	d	q^{d}	q^{t}	q^{e}	e	u	d	q^{d}	q^{t}	q^{e}	e	u	d	q^{d}	q^{t}	q^{e}	e	u
d	0.47	0.46	0.49	0.48	0.65	0.48	0.09	0.09	0.15	0.12	0.32	0.13	0.04	0.05	0.05	0.05	1.00	0.05
q^{d}	0.59	0.56	0.58	0.57	0.71	0.57	0.23	0.20	0.26	0.23	0.44	0.26	0.20	0.20	0.20	0.20	1.00	0.20
q^{t}	0.39	0.52	0.30	0.63	0.43	0.32	0.61	0.64	0.07	0.30	0.73	0.06	0.30	0.28	0.29	0.28	1.00	0.29
q^{e}	0.73	0.66	0.71	0.63	0.79	0.71	0.41	0.42	0.34	0.28	0.57	0.34	0.24	0.23	0.24	0.23	1.00	0.23
e	0.72	0.74	0.71	0.74	0.67	0.74	0.98	0.96	0.97	0.94	0.89	0.90	0.83	0.83	0.83	0.83	0.80	0.83
u	0.59	0.59	0.56	0.61	0.65	0.55	0.46	0.46	0.36	0.38	0.59	0.34	0.32	0.32	0.32	0.32	0.96	0.32

Table 3: *Comparison of failure rate d, distance quality q^{d}, time quality q^{t}, effort quality q^{e}, computational efficiency e, and a uniform combination of all metrics u for the three steering algorithms. Each column holds the values of all metrics that are computed with the parameters that optimize only for the metric corresponding to the column.*

Parameter	DEF	Min	Max	d	q^{d}	q^{t}	q^{e}	e	u
max speed	2	1	3.20	3.20	2.15	3.20	1.52	3.14	3.14
neighbor distance	15	2	22	17.39	13.37	14.75	12.08	8.18	8.99
time horizon	10	2	16	16	3.71	2	2.72	8.44	2.92
time horizon obstacles	7	2	16	12.30	16	9.60	11.81	2	10.92
max neighbors	10	2	22	8	11	2	15.03	2	2

Table 4: *The parameters of **ORCA**: default values (DEF), bounds (Min, Max), and optimal values obtained for each metric separately, and a uniform combination of the metrics.*

changes in its planned path, which would require an expensive path replanning operation.

To minimize failure rate and meet the time limit, **ORCA** raises its time horizon to increase the number of agents it considers in its velocity calculations, and increases its max speed so that agents cover as much distance as possible. For distance quality, $q^{\mathrm{d}}(A_{\mathbf{v}})$, **ORCA** reduces max speed just like **PPR**.

In general, **SF** reduces acceleration parameters to minimum values for all quality metrics to prevent agents from overreacting.

Looking directly at the correlation coefficients, for **ORCA** in Table 5, we can make the following observations:

1. For **ORCA**, the maximum number of neighbours considered has the highest correlation with most metrics. The max speed seems to be the second most important parameter. It affects effort quality, $q^{\mathrm{e}}(A_{\mathbf{v}})$, negatively, and time quality $q^{\mathrm{t}}(A_{\mathbf{v}})$ positively.

2. For **PPR**, the max speed factor, which is a multiplier that increases the speed of an agent, is strongly correlated with the efficiency metric, e, and has a negative effect on all quality metrics.

3. For **PPR**, the size of the neighbourhood area, and the distance to the furthest local target seem to be the parameters most strongly correlated with efficiency, e.

4. For **SF**, the parameters with the highest correlation to computational efficiency, e, have to do with proximity forces. When these are increased, agents push each other away forcefully, decreasing the likelihood that they will interact again in the the next frame.

5. The parameters of **SF** that affect the quality measures the most are the wall repulsion coefficients.

The above analysis is not meant to be definite or complete, but rather to demonstrate that the proposed methodology is an effective way to optimize, probe and analyze the behaviour of a steering algorithm in relation to its parameters, over a small or large set of test cases. The next section presents a number of more focused examples and other practical applications of our methodology.

ORCA	d	q^{d}	q^{t}	q^{e}	e
d	1	1.00	0.20	0.35	-0.18
q^{d}	1.00	1	0.21	0.36	-0.16
q^{t}	0.20	0.21	1	-0.63	-0.02
q^{e}	0.35	0.36	-0.63	1	-0.01
e	-0.18	-0.16	-0.02	-0.01	1

Table 6: *Spearman correlation coefficients between performance metrics for a set of 1000 parameter samples with **ORCA**.*

5 Applications and Results

The previous section demonstrates that it is both beneficial and revealing to fit the parameters of a steering algorithm to performance objectives over a large set of test cases. This section presents a series of experiments that demonstrate the potential applications of parameter fitting for more specific cases. We refer the reader to the accompanying video for a visual demonstration of the results, and additional experiments.

Circlular benchmark. A popular and challenging scenario, that is often used to test the effectiveness of a steering algorithm, distributes the agents on a circular fashion with diametrically opposite goals. Such a configuration forces dense simultaneous interactions in the middle of the circle. Using a group of 500 agents, we compare the results of **ORCA** with the default, and optimized parameter values that aim to minimize time quality, $q^{\mathrm{t}}(A_{\mathbf{v}})$. With the optimal parameter set, **ORCA** takes 50% less time to complete the benchmark, and exhibits a more organized emerging behaviour. Agents seem to form groups that follow a smooth curved trajectory, Figure 1 (left). For reference, the optimal parameter set in this case is: {max speed: 3.2, neighbour distance: 13.63, time horizon: 2.32, time horizon obstacles: 5.30, max neighbours: 7}.

Room evacuation. Evacuation benchmarks are important for a range of application domains. In this benchmark a group of 500 agents must exit a room. For this experiment we use the social forces, **SF**, method with the default, and optimized parameter values that minimize the effort quality metric, $q^{\mathrm{e}}(A_{\mathbf{v}})$. **SF** with optimal parameters spends 66% less energy on average per agent, exhibits tighter packing, and visibly reduces the turbulence of the crowd's behaviour, Figure 1 (right).

Parameter	d	q^{d}	q^{t}	q^{e}	e
max speed	0.02	0.03	−0.34	0.58	0.14
neighbour distance	−0.09	−0.07	−0.13	−0.03	0.03
time horizon	−0.12	−0.08	0.10	0.04	0.07
time horizon obstacles	−0.09	−0.09	0.17	0.04	0.11
max neighbors	0.42	0.47	0.54	0.29	0.37

Table 5: *Spearman correlation coefficients between five metrics and the parameters of **ORCA**. The maximum amount of neighbours considered seems to have a significant effect on all metrics. For the effort metric, q^{e}, the maximum speed parameter has a great negative effect.*

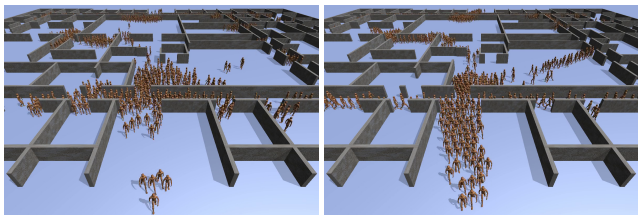


Figure 3: *Office evacuation with **ORCA**. Default parameters (left), parameters optimized for time quality (right).*

Office evacuation. A more challenging evacuation scenario that involves 1000 agents in a more complex, office-like ground floor. Optimizing **ORCA** for time quality, $q^{\mathrm{t}}(A_{\mathbf{v}})$, reduces the average time it takes to exit the building by almost 60%. In addition, it exhibits higher crowd density, and higher throughput at the doors, Figure 3.

Optimizing for Ground Truth. There are a few methods that use recorded crowd motion to influence and direct virtual crowds. In this experiment, we simply show that our methodology can support this approach. We optimize the behaviour of the three test algorithms to match real world data contained in the ground truth test set, $\mathcal{G}$, Section 3.2. Our experiments showed that in most cases, the optimization was able to significantly alter the resulting steering behaviour, and increase the similarity to the recorded data. Table 7 reports the reduction in the entropy metric, g, (increase in similarity) as a result of parameter optimization for all three algorithms, and two different benchmarks. Figure 4, shows a comparison between default (left), ground truth (middle), and optimized (right) results for the bi-directional hallway benchmark using **ORCA**.

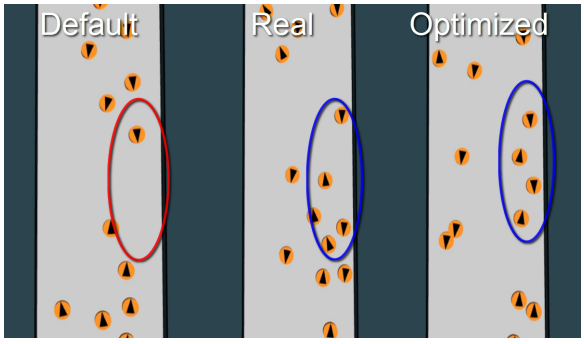


Figure 4: *Parameters optimized for similarity to ground truth (middle) produce better results (right) than default parameters (left).*

Dynamically Adapting Steering Parameters. Our methodology can create a large number of samples that relate parameter values to performance metrics. It is an obvious next step to model that landscape, and learn the relation between parameters and performance metrics, using, perhaps, non-linear, manifold learning techniques. We leave this task for future work. In the mean time, we explore a couple of simple interactive examples. Figure 5 shows a snapshot from an interactive demo that allows the user to switch dynamically between optimal parameter values that correspond to different objectives.

$A_{\mathbf{v}}$	$\mathbf{v}$	2-agent-crossing	bi-directional hallway
PPR	`DEF`	3.42	3.40
	`OPT`	1.92	2.27
ORCA	`DEF`	2.12	2.95
	`OPT`	0.63	2.20
SF	`DEF`	3.74	3.62
	`OPT`	3.10	2.76

Table 7: *Comparison of entropy metric values before and after optimization to match real world data. `DEF`: default parameter values, `OPT`: optimal parameter values.*

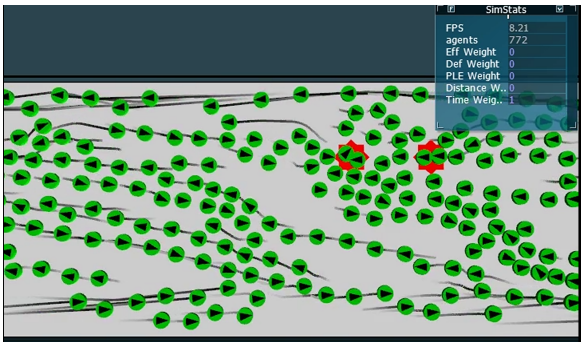


Figure 5: *A prototype system for interactively setting the relative weights of the metrics in the objective. When the weights change, the algorithm's parameters are set automatically to the corresponding optimal values.*

The accompanying video demonstrates additional examples, including multi-objective optimizations, and optimizations that improve computational efficiency.

5.1 Implementation details

The primary factors that affect the computational performance of the optimization are the size of the test set, the number and range of parameters that are fitted, and the number of agents in the test cases. Although CMA-ES is an efficient optimization method, fitting a large number of parameters over a sizeable test set is computationally expensive. For reference, a 12 core, 2.4 GHz, 12 GB, computer, using 10 parallel threads, takes in the order of 20 hours to optimize an algorithm over the 5008 scenarios in the test set, $\mathcal{T}$.

6 Conclusion

We have presented a methodology to fit a steering algorithm's parameters to user defined objectives over arbitrary test sets. Our framework and methodology are general. Most elements can be tailored to the needs of a particular application. For example, one can use different performance metrics, objectives, test sets, and optimization methods.

More importantly, we also show that parameter fitting not only can be used to improve the performance of an algorithm, but it can also serve as an analysis tool to produce a detailed view of an algorithm's behaviour relative to its internal parameters. This detailed view can be the basis of a thorough introspective analysis that allows both developers and end-users to gain insights on the performance and behaviour of an algorithm.

References

BROGAN, D. C., AND HODGINS, J. K. 1997. Group behaviors for systems with significant dynamics. *Auton. Robots 4*, 1, 137–153.

BRUCKNER, S., AND MOLLER, T. 2010. Result-driven exploration of simulation parameter spaces for visual effects design. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (Nov.), 1468–1476.

DAVIDICH, M., AND KOESTER, G. 2011. Towards automatic and robust adjustment of human behavioral parameters in a pedestrian stream model to measured data. In *Pedestrian and Evacuation Dynamics*, R. D. Peacock, E. D. Kuligowski, and J. D. Averill, Eds. Springer US, 537–546.

ENNIS, C., PETERS, C., AND O'SULLIVAN, C. 2011. Perceptual effects of scene context and viewpoint for virtual pedestrian crowds. *ACM Trans. Appl. Percept. 8*, 2 (Feb.), 10:1–10:22.

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 SCA*, Eurographics Association, 119–128.

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 Trans. on Graphics 31*, 6, 11.

HA, S., MCCANN, J., LIU, C. K., AND POPOVIĆ, J. 2013. Physics Storyboards. *Computer Graphics Forum (The proceeding of Eurographics 2013)*.

HANSEN, N., AND OSTERMEIER, A. 1996. Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation. In *Evolutionary Computation, 1996, Proceedings of IEEE International Conference on*, 312–317.

HANSEN, N. 2011. A CMA-ES for Mixed-Integer Nonlinear Optimization. Research Report RR-7751, INRIA, Oct.

HELBING, D., FARKAS, I., AND VICSEK, T. 2000. Simulating dynamical features of escape panic. *Nature 407*, 6803, 487–490.

HELBING, D., BUZNA, L., JOHANSSON, A., AND WERNER, T. 2005. Self-organized pedestrian crowd dynamics: Experiments, simulations, and design solutions. *Transp. Science 39*, 1, 1–24.

HUERRE, S., LEE, J., LIN, M., AND O'SULLIVAN, C. 2010. Simulating believable crowd and group behaviors. In *ACM SIGGRAPH ASIA 2010 Courses*, ACM, New York, NY, USA, SA '10, 13:1–13:92.

JU, E., CHOI, M. G., PARK, M., LEE, J., LEE, K. H., AND TAKAHASHI, S. 2010. Morphable crowds. In *ACM SIGGRAPH Asia 2010 papers*, ACM, New York, NY, USA, SIGGRAPH ASIA '10, 140:1–140:10.

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 SCA*, ACM, SCA '11, 53–62.

KULPA, R., OLIVIERXS, A.-H., ONDŘEJ, J., AND PETTRÉ, J. 2011. Imperceptible relaxation of collision avoidance constraints in virtual crowds. In *Proceedings of the 2011 SIGGRAPH Asia Conference*, ACM, New York, NY, USA, SA '11, 138:1–138:10.

LAMARCHE, F., AND DONIKIAN, S. 2004. Crowd of virtual humans: a new approach for real time navigation in complex and structured environments. In *Computer Graphics Forum 23*.

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 SCA*, Eurographics Association, 109–118.

LEMERCIER, S., JELIC, A., KULPA, R., HUA, J., FEHRENBACH, J., DEGOND, P., APPERT-ROLLAND, C., DONIKIAN, S., AND PETTRÉ, J. 2012. Realistic following behaviors for crowd simulation. *Comput. Graph. Forum 31*, 2, 489–498.

LERNER, A., CHRYSANTHOU, Y., AND LISCHINSKI, D. 2007. Crowds by example. *Computer Graphics Forum 26*, 3 (September), 655–664.

LERNER, A., CHRYSANTHOU, Y., SHAMIR, A., AND COHEN-OR, D. 2010. Context-dependent crowd evaluation. *Comput. Graph. Forum 29*, 7, 2197–2206.

MCDONNELL, R., LARKIN, M., DOBBYN, S., COLLINS, S., AND O'SULLIVAN, C. 2008. Clone attack! perception of crowd variety. In *ACM SIGGRAPH 2008 papers*, ACM, New York, NY, USA, SIGGRAPH '08, 26:1–26:8.

METOYER, R. A., AND HODGINS, J. K. 2003. Reactive pedestrian path following from examples. In *Proceedings of CASA 2003*, IEEE Computer Society, 149–156.

MUSSE, S. R., JUNG, C. R., JACQUES, JR., J. C. S., AND BRAUN, A. 2007. Using computer vision to simulate the motion of virtual agents: Research articles. *Comput. Animat. Virtual Worlds 18*, 2 (May), 83–93.

MUSSE, S. R., CASSOL, V. J., AND JUNG, C. R. 2012. Towards a quantitative approach for comparing crowds. *Computer Animation and Virtual Worlds 23*, 1, 49–57.

NARAIN, R., GOLAS, A., CURTIS, S., AND LIN, M. C. 2009. Aggregate dynamics for dense crowd simulation. In *ACM SIGGRAPH Asia 2009 Papers*, ACM, New York, NY, USA, SIGGRAPH Asia '09, 122:1–122:8.

ONDŘEJ, J., PETTRÉ, J., OLIVIER, A.-H., AND DONIKIAN, S. 2010. A synthetic-vision based steering approach for crowd simulation. *ACM Trans. Graph. 29*, 4 (July), 123:1–123:9.

PARIS, S., PETTRÉ, J., AND DONIKIAN, S. 2007. Pedestrian reactive navigation for crowd simulation: a predictive approach. In *EUROGRAPHICS 2007*, vol. 26, 665–674.

PATIL, S., VAN DEN BERG, J., CURTIS, S., LIN, M., AND MANOCHA, D. 2011. Directing crowd simulations using navigation fields. *Visualization and Computer Graphics, IEEE Transactions on 17*, 2 (feb.), 244 –254.

PELECHANO, N., ALLBECK, J. M., AND BADLER, N. I. 2007. Controlling individual agents in high-density crowd simulation. In *Proceedings of SCA*, Eurographics Association, 99–108.

PELECHANO, N., ALLBECK, J. M., AND BADLER, N. I. 2008. *Virtual Crowds: Methods, Simulation, and Control*. Synthesis Lectures on Computer Graphics and Animation. Morgan & Claypool Publishers.

PELLEGRINI, S., ESS, A., SCHINDLER, K., AND VAN GOOL, L. 2009. You'll never walk alone: Modeling social behavior for multi-target tracking. In *Computer Vision, 2009 IEEE 12th International Conference on*, 261–268.

PETTRÉ, J., ONDŘEJ, J., OLIVIER, A.-H., CRETUAL, A., AND DONIKIAN, S. 2009. Experiment-based modeling, simulation and validation of interactions between virtual walkers. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ACM, New York, NY, USA, SCA '09, 189–198.

PINTER, M. 2001. Toward more realistic pathfinding. *Gamasutra. com*.

REYNOLDS, C. W. 1987. Flocks, herds and schools: A distributed behavioral model. In *Proceedings of ACM SIGGRAPH*, ACM, 25–34.

REYNOLDS, C. W. 1999. Steering behaviors for autonomous characters. *Game Developers Conference 1999*, 9602, 763–782.

SEYFRIED, A., BOLTES, M., KHLER, J., KLINGSCH, W., PORTZ, A., RUPPRECHT, T., SCHADSCHNEIDER, A., STEFFEN, B., AND WINKENS, A. 2010. Enhanced empirical data for the fundamental diagram and the flow through bottlenecks. In *Pedestrian and Evacuation Dynamics 2008*, W. W. F. Klingsch, C. Rogsch, A. Schadschneider, and M. Schreckenberg, Eds. Springer Berlin Heidelberg, 145–156.

SINGH, B. S., KAPADIA, M., FALOUTSOS, P., AND REINMAN, G. 2009. Steerbench : a benchmark suite for evaluating steering behaviors. *Computer Animation And Virtual Worlds 20*, February, 533–548.

SINGH, S., KAPADIA, M., HEWLETT, B., REINMAN, G., AND FALOUTSOS, P. 2011. A modular framework for adaptive agent-based steering. In *Proceedings of I3D*, ACM, New York, NY, USA, I3D '11, 141–150.

SUD, A., GAYLE, R., ANDERSEN, E., GUY, S., LIN, M., AND MANOCHA, D. 2007. Real-time navigation of independent agents using adaptive roadmaps. In *Proceedings of VRST*, ACM, 99–106.

THALMANN, D., AND MUSSE, S. R. 2013. *Crowd Simulation, Second Edition*. Springer.

TREUILLE, A., COOPER, S., AND POPOVIĆ, Z. 2006. Continuum crowds. *ACM Trans. Graph. 25*, 3, 1160–1168.

VAN DEN BERG, J., LIN, M. C., AND MANOCHA, D. 2008. Reciprocal velocity obstacles for real-time multi-agent navigation. In *Proceedings of ICRA*, IEEE, 1928–1935.

VAN DEN BERG, J., PATIL, S., SEWALL, J., MANOCHA, D., AND LIN, M. 2008. Interactive navigation of multiple agents in crowded environments. In *Proceedings of I3D*, 139.

VAN DEN BERG, J., GUY, S. J., LIN, M., AND MANOCHA, D. 2011. Reciprocal n-body collision avoidance. In *Robotics Research*, vol. 70 of *Springer Tracts in Advanced Robotics*. Springer Berlin Heidelberg, 3–19.

WANG, J. M., FLEET, D. J., AND HERTZMANN, A. 2010. Optimizing walking controllers for uncertain inputs and environments. *ACM Trans. Graph. 29*, 4 (July), 73:1–73:8.

WHITTLE, M. 2007. *Gait analysis: an introduction*. Butterworth-Heinemann.