Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2014)
Vladlen Koltun and Eftychios Sifakis (Editors)

SteerFit: Automated Parameter Fitting for Steering Algorithms

Glen Berseth[1], Mubbasir Kapadia[2], Brandon Haworth[1] and Petros Faloutsos[1]

[1]York University
[2]University of Pennsylvania

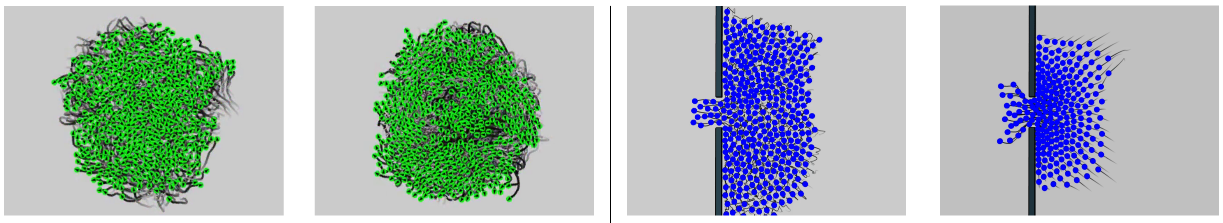

(a) Scenario I : default (b) Scenario I: optimal time (c) Scenario II: default (d) Scenario II: optimal effort

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 smoother, faster and tighter room evacuation.*

Abstract

In the context of crowd simulation, there is a diverse set of algorithms that model steering. *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 steering algorithms 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. 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. We also propose using the Pareto Optimal front as an efficient way of modelling optimal relationships between multiple objectives, and demonstrate its effectiveness by estimating optimal parameters for interactively defined combinations of the associated objectives. The proposed methodologies are general and can be applied to any steering algorithm using any set of performance criteria.*

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. This 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. Given a pattern of behaviour, a performance criterion (metric) or a trade-off between performance metrics, 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 the following: success; quality with respect to distance, time, or energy consumption of an agent; 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.

We demonstrate our proposed methodology using three established agent-based algorithms: (1) **ORCA**: a predictive technique that uses reciprocal velocity obstacles for collision avoidance [vdBGLM11], (2) **PPR**: a hybrid approach that uses rules to combine reactions, predictions, and planning [SKH*11], and (3) **SF**: a variant of the social forces method for crowd simulation [HFV00]. 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, on average, optimal parameter values generalize 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 the complete analysis can be found in the supplemental material.

We also study the interesting and challenging problem of dynamically tuning the parameters of an algorithm to support interactively defined combinations of objectives. For most practical cases, it is not feasible to solve this problem in real-time every time the combination changes. To address this issue we precompute optimal trade-offs between the objectives in the form of a discrete approximation of the Pareto Optimal front. During runtime, our method efficiently estimates the parameters of the algorithm that optimally support a new combination of the objectives.

2. Related Work

Since the seminal work of [Rey87, Rey99], crowd simulation has been studied from many different perspectives. We refer the readers to comprehensive surveys [PAB08, HLLO, TM13] and present a broad review below.

Continuum-based techniques [TCP06, NGCL09] model the characteristics of the crowd flow to simulate macroscopic crowd phenomena. Particle-based approaches [Rey87, Rey99] model agents as particles and simulate crowds using basic particle dynamics. The social force model [HBJW05, PAB07] simulates forces such as repulsion, attraction, friction and dissipation for each agent to simulate pedestrians. Rule-based approaches [LD04, SGA*07] use various conditions and heuristics to identify the exact situation of an agent. Egocentric techniques [KSHF09] model a local variable-resolution perception of the simulation. Data-driven methods [LCHL07, LCL07, JCP*10, BKSB13] use existing video or motion capture data to derive steering choices that are then used in virtual worlds, and recent work [OPOD10] demonstrates a synthetic vision-based approach to steering. The works of [PPD07, vdBGLM11] use predictions to steer in environments populated with dynamic threats.

Crowd Evaluation. There has been a growing recent trend to use statistical analysis in the evaluation and analysis of crowd simulations. The work by Lerner *et al.* [LCSCO10] adopts a data-driven approach to evaluating crowds by measuring its similarity to real world data. Singh *et al.* [SKFR09] 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 [KWS*11] 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 [LCSCO10] and fundamental diagram-based comparisons [SBK*10] use aggregate metrics for quantifying similarity. The work in [GvdBL*12, POO*09] measures the ability of a steering algorithm to emulate the behavior of a real crowd dataset by measuring its divergence from ground truth. [MCJ12] presents a histogram-based technique to quantify the global flow characteristics of crowds. Perceptual studies rely on human factors experiments to measure the variety in appearance and motion [MLD*08], or perceptual fidelity of relaxing collisions [KOOP11] in crowds.

Parameter Optimization. Parameter fitting is widely used in visual effects [BM10] to automate the tuning of model parameters to meet certain user-defined criteria. The resulting optimization problems tend to involve non-convex, and high-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 [HMLP13, WFH10]. By selecting the right set of parameters, researchers have shown improvements in a steering algorithm's ability to match recorded crowd data [JHS07, POO*09, PESVG09, DK11, LJK*12].

Concurrent work [WGO*14] explores parameter estimation for steering algorithms to match reference data for specific scenarios. Our method is not tied to ground truth, and can be used to optimize quantitative metrics such as the computational performance of the algorithm. Additionally, we leverage the use of different test sets, including small-scale interactions and high-density crowds, to obtain optimal parameter values that *generalize* across the space of possible scenarios. To offset the computational burden of optimizing an algorithm for different criteria, we propose a method to precompute the mapping between an algorithm's parameters and objective weights, thus allowing us to dynamically adapt the crowd behaviour at real-time rates.

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 steering approaches. (1) **PPR**: [SKH*11] combines reactions, predictions and planning in one single framework with 38 tunable parameters. (2) **ORCA**: [vdBGLM11] uses reciprocal velocity obstacles for goal-directed collision avoidance. (3) **SF**. [HFV00] uses hypothetical social forces for resolving collisions between interacting agents in dense crowds. These algorithms represent the broad taxonomy of crowd approaches with mutually exclusive parameter sets that can be tuned to produce widely differing variations in the resulting crowd behavior. Additional details of the algorithm parameters can be found in the supplementary document.

3.2. Test Sets

We employ different benchmark sets including local agent interactions and high-density crowds to find the optimal values of an algorithm's parameters that generalize across the wide range of situations that agents encounter in crowds. Note 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.

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. The bottom three scenario are part of $\mathcal{S}^v$. All are designed to stress the steering algorithms computational efficiency.*

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 [KWS*11], 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. We use the representative set because it provides the best sampling of the full space of possible scenarios. Therefore, optimizing for the representative set should give good results in general for any scenario.

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. Here we use statistical significance to contrast against common practice in crowd simulation where results are demonstrated on a very limited number of test cases.

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. There are few publicly available data sets of recorded crowd motion which can be used to test a steering algorithm's ability to match real world data. We use a ground truth test set $\mathcal{G}$, published by [SBK*10], for our experiments.

3.3. Normalized Performance Measures

This section defines a variety of intuitive measures to characterize the performance of a steering algorithm on the test set $\mathcal{T}$. These include: (1) the fraction of scenarios that an algorithm was unable to solve in the representative set of scenarios, (2) quality measures with respect to distance travelled, total time taken, or energy consumption of an agent, (3) computational performance of the algorithm, or (4) statistical similarity with respect to ground truth. The specific metrics we use are briefly described below and we refer the reader to more detailed explanations in [KWS*11, GCC*10, GvdBL*12]. In addition, users may define their own 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 it successfully completes in $\mathcal{T}$. An algorithm successfully completes a particular scenario if the reference agent reaches its goal without any collisions and the total number of collisions among non-reference agents is less than the number of agents in the scenario. The failure rate is the complement of coverage $d(A_\mathbf{v}) = 1 - c(A_\mathbf{v})$.

Distance Quality. The distance quality $q^{\mathrm{d}}(A_\mathbf{v})$ of $A_\mathbf{v}$ for a single scenario s is 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. It is computed as: $q^{\mathrm{d}}(A_\mathbf{v}) = 1 - \frac{o_s^d}{a_s^d}$. The ideal optimal path is the shortest static path from the agent's initial position to its goal. If the algorithm does not successfully complete the scenario then the associated distance quality metric is set to the worst-case value of 1.

Time Quality. Similarly, $q^{\mathrm{t}}(A_\mathbf{v})$ 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 corresponds to the agent reaching its goal when moving with its desired velocity along the ideal optimal path.

PLE Quality. The PLE quality metric is computed as $q^{\mathrm{e}}(A_\mathbf{v}) = 1 - \frac{o^e}{a^e}$, where $o_s^e = 2 \cdot o_s^d \cdot (e_s \cdot e_w)$ is the ideal optimal effort and a^e the actual effort of the agent [GCC*10]. The distance, time, and PLE quality measures can be averaged across a large set of benchmarks to provide aggregate quality measures for a test set.

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 $\mathcal{S}$. 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 $(N \cdot n_{\mathrm{des}})^{-1}$ 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:

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

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 attend to 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 $g(A_\mathbf{v}, \mathcal{G})$ [GvdBL*12] 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:

$$f(A_\mathbf{v}, \mathbf{w}) = \sum_{m_i \in \mathbb{M}} w_i \cdot m_i, \tag{2}$$

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 cases, 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. This can be formulated as a minimization problem:

$$\mathbf{v}^*_{\mathbf{w}} = \arg\min_{\mathbf{v}\in\mathbb{V}} f(A_{\mathbf{v}}, \mathbf{w}). \quad (3)$$

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) [HO96, Han11] 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, it is very competitive in converging to an optimal value in few iterations, and it has support for mixed integer optimization. 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. 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 an 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.

4. Large Scale Study

We study the effects of parameter fitting using the combined test sets, $\mathcal{T}$ and $\mathcal{T}^v$. Our goal is to identify whether parameter fitting has a significant effect and to understand the relation between algorithmic parameters and performance. 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, $u(A_{\mathbf{v}})$, over the entire combined set, $\mathcal{T}$. For comparison, we also compute the same metrics for all algorithms with their parameters set to default values. The results in Figure 2 show a strong increase in optimality for all metrics.

The default parameters for **PPR**, **ORCA** and **SF** cannot solve 39%, 56%, and 26% of the sampled scenarios respectively. 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

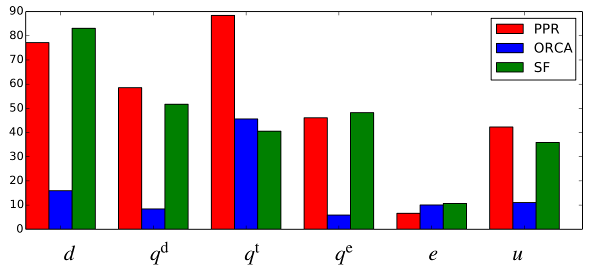


Figure 2: *Relative percent improvement of failure rate d, distance quality q^{d}, time quality q^{t}, effort quality q^{e}, computational efficiency e, and a uniform combination of metrics u for the three steering algorithms.*

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 Figure 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 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.

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 optimize 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 2. We observe that the optimal parameters for distance quality, $q^{\mathrm{d}}(A_{\mathbf{v}})$, produce near-optimal results for failure rate, $d(A_{\mathbf{v}})$, for **PPR** and **ORCA**. However, the opposite does not hold true. Optimizing for failure rate does not yield optimal results for distance quality.

A correlation analysis clarifies the dependencies across metrics for a given algorithm. 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 3. 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,

	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
$d(A_{\mathbf{v}})$	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
$q^{\mathrm{d}}(A_{\mathbf{v}})$	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
$q^{\mathrm{t}}(A_{\mathbf{v}})$	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
$q^{\mathrm{e}}(A_{\mathbf{v}})$	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
$e(A_{\mathbf{v}})$	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
$u(A_{\mathbf{v}})$	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 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. Each coulmn is the optimal parameter set for optimizing that algorithm for that objective. The row value is the result of computing that metric with the columns optimal parameters.*

$q^{\mathrm{t}}(A_{\mathbf{v}})$, and effort quality, $q^{\mathrm{e}}(A_{\mathbf{v}})$, which, in general, can be expected as faster motion requires more energy.

3. A weak positive correlation between time quality, $q^{\mathrm{t}}(A_{\mathbf{v}})$, and distance quality, $q^{\mathrm{d}}(A_{\mathbf{v}})$, as expected since a shorter path often results in shorter completion time.

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 3: *Spearman correlation coefficients between performance metrics for* 1000 *parameter samples with* ***ORCA****.*

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. We present relevant data for **ORCA** in Table 4 and refer the readers to the supplemental material for the supporting data on the other two algorithms.

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. 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 minimize failure rate and meet the time limit, **ORCA** raises its time horizon and increases its max speed. This increases the number of agents it considers in its velocity calculations and ensures agents cover as much distance as possible, respectively. 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 at the correlation coefficients for **ORCA** in Table 4 and in the supplementary material for **PPR** and **SF**, 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 **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.

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 obs	−0.09	−0.09	0.17	0.04	0.11
max neighbors	0.42	0.47	0.54	0.29	0.37

Table 4: *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 large inverse effect.*

The above analysis is not meant to be definite or complete, but rather to demonstrate that the proposed methodology can be notably more effective than manual tuning. The framework 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.

5. Optimal Parameter Mapping for Multiple Objectives

Optimizing a steering algorithm's parameters across a large test set is computationally expensive. The computational complexity increases with the number of parameters and the cardinality of a test set. For example, it takes $\sim$ 20 hours to optimize the 11 parameters of **SF** over the representative test set $\mathcal{T}$. In a weighted multi-objective optimization application, it is desirable to model the relationship between objectives and algorithm parameters. This avoids running an expensive optimization every time we wish to change the associated weights. This can be accomplished by computing the optimal parameters for a discrete set of weighted combinations that can then be interpolated. There are two problems with this approach. First, it can waste significant amounts of computation since each sample point is a result of an independent process that could be visiting the same points in the domain. Second and most important, it is not looking at relationships between the objectives but rather at their weighted combination. Both of these problems can be addressed by computing a *Pareto Optimal Front*. Pareto optimality is a very important concept in optimization which has sparingly been used in computer animation. Our method based on *Pareto Optimality* not only avoids unnecessary computation but also provides a more principled model of the optimal relationships between multiple objectives.

5.1. Pareto Optimality

Pareto Optimality (or Efficiency) refers to a situation where no objective can be improved further without worsening one of the other objectives. The set of points that are Pareto optimal constitute the Pareto Optimal front, a hyper-surface that captures the optimal relationships between the objectives. Computing this front is not trivial and is, in fact, an active area of research. Current state of the art techniques are primarily based on genetic algorithms. We have chosen to use DEAP [FDG*12] and NSGA-II [DPAM02] to estimate the Pareto Optimal front.

A standard evolutionary approach to solving a multi-objective optimization problem models the fitness of samples using a single objective function that is the weighted sum of multiple objectives, where the samples chosen in each iteration minimize the combined objective. In contrast, the goal of Pareto Optimal front approximation is to maximize the hyper-volume constructed by the non-dominated samples (see Figure 3(a)). A point dominates another if it is superior in all Pareto dimensions.

Figure 3(b-d) demonstrates the Pareto Optimal front for three cases. First, we optimize the **ORCA** steering algorithm for e and q^e over a bottleneck scenario. The process and resulting Pareto Optimal front can be seen in Figure 3(b). Second, we optimize the **SF** algorithm for the same scenario and three metrics, e, q^e and $g(A_{\mathbf{v}}, \mathcal{G})$ (the result can be found in Figure 3(c)). The ground truth set, $\mathcal{G}$, is a recording of people funnelling into a small bottleneck, very similar to the scenario used. We optimize for the same objectives with the **ORCA** steering algorithm and the resultant Pareto Optimal front can be see in Figure 3(d). The pareto front is able to capture the non-linear relationships between contradictory objectives and efficiently encodes the tradeoffs between them. For example, optimizing q^e has an adverse effect on $g(A_{\mathbf{v}}, \mathcal{G})$, as shown in Figure 3(c and d).

The Pareto Optimal front provides a principled model of the optimal relationships between the objectives. The number of dimensions is equal to the number of objectives, so for two objectives the result is a 2D curve and for three objectives a 3D surface. For most practical applications three objectives should be sufficient.

5.2. Pareto Optimal Front Interpolation

Having an estimate of the Pareto Optimal front for a set of objectives provides us with the basis to estimate optimal parameters for the associated algorithm with arbitrary combinations of the objectives.

The first step in developing an interpolation model for arbitrary combinations of the objectives is to transform the Pareto Optimal front from objective space to weight space. For m objectives the Pareto Optimal front contains a set of m-dimensional points, $\mathcal{P} = \{\mathbf{b}_p | \forall p = 1, ..., N\}$, including a set of points $\mathcal{P}_O = \{\mathbf{b}^O_p | \forall p = 1, ..., m\}$, that correspond to minimizing each objective while ignoring the others. These latter points have known coordinates in weight space that correspond to the standard unit vectors, and hold the minimum value in the associated dimension.

We transform the Pareto Optimal front from the m-dimensional objective space, $[b_i]$, to the m-dimensional weight space, $[w_i]$, using the following steps: (a) we normalize the Pareto Optimal front so that each dimension maps to $[0, 1]$, (b) we replace each point with its distances from the normalized points in $\mathcal{P}_O$, (c) we project the points, $\mathbf{b}'$, resulting from the previous stage onto the $\sum_i b'_i = 1$ plane and (d) we subtract them from 1. The transformed Pareto Optimal front is now mapped onto a normalized simplex from which we can compute the relative weights of each original point as its barycentric coordinates, (Figure 4).

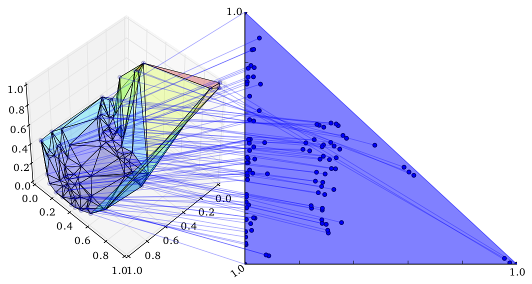
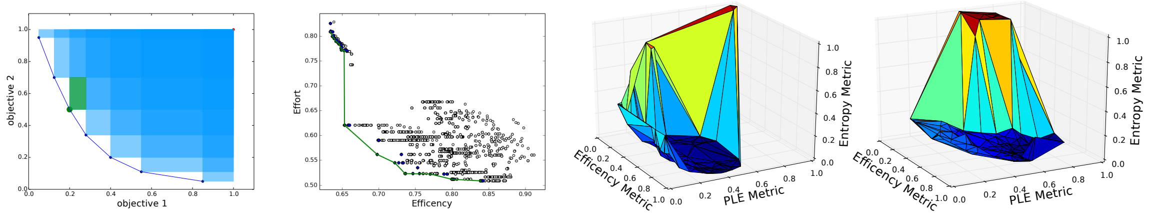


Figure 4: *Projecting a 3D Pareto Optimal front onto a triangular normalized weight domain.*

Having the Pareto Optimal front in weight space, we can now use a standard multidimensional interpolation method such as Mardy quadratics or variants of Shepard's method. A common choice within the Mardy quadratics family of methods is radial basis function interpolation. For three objectives, the associated domain forms a triangle. In this case, given a new set of weights, we can use Delaunay triangulation to compute the three points that make up the bounding simplex whose associated parameters will be interpolated with a standard inverse distance approach.



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

Figure 3: *Each point in (a) dominates any other point in the shaded area defined by that point and adding the green point improves the pareto front equivalent to the green patch it defines. Figure (b) shows the final Pareto Optimal front of non-dominated points (in green) for the **ORCA** steering algorithm over two objectives. Figures (c and d) show the final computed Pareto Optimal front for three objectives for the **SF** and **ORCA** steering algorithms.*

6. Applications and Results

Section 4 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, 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 minimize time quality $q^{t}(A_{\mathbf{v}})$. With the optimal parameters, **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(a and b).

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 as well as optimized parameter values that minimize the effort quality metric, $q^{e}(A_{\mathbf{v}})$. **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(c and d).

Office Evacuation. A more challenging evacuation scenario places 1000 agents in a complex, office-like ground floor. Optimizing **ORCA** for time quality, $q^{t}(A_{\mathbf{v}})$, 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 exits, as seen in Figure 5. Here we use ADAPT [KMB14] to render bipedal characters.

Optimizing for Ground Truth. There are a few methods that use recorded crowd motion to influence and direct virtual crowds. Here, we simply show that our methodology can also support this application. 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. Figure 6 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.

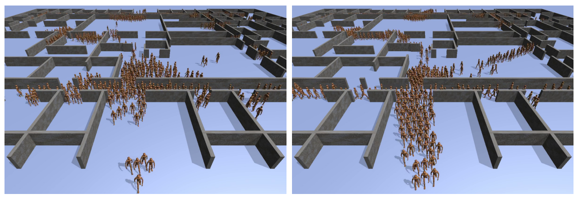


Figure 5: *Office evacuation with **ORCA**. Simulation with parameters optimized for time quality (right) take half the time to complete as compared to the default parameters (left).*

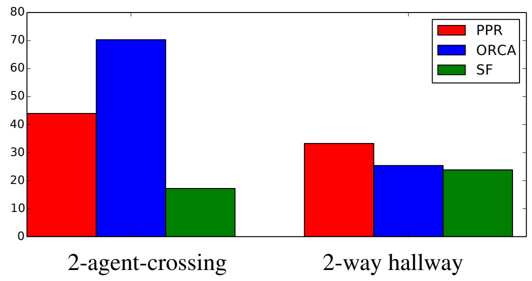


Figure 6: *Relative percent improvement of entropy metric values after optimization on two benchmarks.*

Interactive Parameter Blending. Using a precomputed Pareto Optimal front, Section 5, we can automatically adapt an algorithm's parameters to provide optimal trade-offs for interactively defined combinations of the associated objectives. Figure 7 shows a snapshot of such blending between three objectives. This process is best demonstrated in the accompanying video.

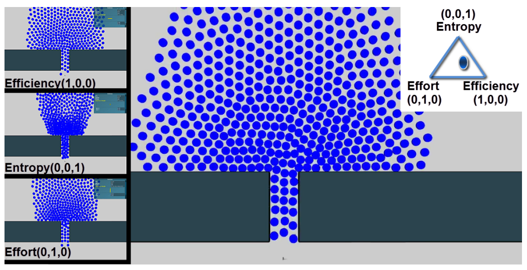


Figure 7: *Blending interactively three objectives (Efficiency, Entropy, and Effort) using a precomputed Pareto Optimal front.*

Implementation details. The primary factors affecting 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 (with hyper-threading), using 10 parallel threads, takes $\sim$ 20 hours to optimize the **SF** algorithm over the test set $\mathcal{T}$. It takes $\sim$ 3 days running 16 parallel threads to compute a Pareto Optimal front with 3 objectives. Interactive blending of the Pareto Optimal front is in realtime.

7. Conclusion

We have presented a framework for optimizing the parameters of a steering algorithm for multiple objectives. Using cross-validation, we show that optimizing over a representative set of scenarios produces optimal parameters that generalize well to new test cases. We have also proposed a method to model trade-offs between the objectives using a Pareto Optimal front. The Pareto Optimal front essentially captures the optimal relationships between objectives. Although our approach can be applied to any number of objectives, three is a practical choice. Thus, we have demonstrated an interactive example that uses the computed Pareto Optimal front to blend between three objectives.

Our study shows 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 range of 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. 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. The supplementary document provides the optimal parameter values of the three steering algorithms for the different objectives which AI developers and enthusiasts can directly use to improve the performance of their crowd simulations. The computational expense of optimizations, especially for large-scale crowds, is one of the reasons why we are committed to sharing our results with the community.

Limitations. Optimization-based methods have certain well-known limitations. For example, it might not be easy or even possible for an optimization process to construct what is essentially a relationship between the parameters of a steering algorithm and global, or long-term, type of objectives. Furthermore, describing desired behaviours as combinations of objectives is not always straightforward and may require experimentation. Although estimating the Pareto Optimal front is much more efficient and effective than naive domain sampling, it still requires significant offline computation.

Future Work. We would like to address heterogeneous crowds by using different parameters per agent or group of agents. We plan to thoroughly investigate the sampling and complexity issues related to the estimation of the Pareto Optimal front, focusing on objectives that are common in crowd simulation.

References

[BKSB13] Boatright C. D., Kapadia M., Shapira J. M., Badler N. I.: Context-sensitive data-driven crowd simulation. In *Proceedings of the 12th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry* (New York, NY, USA, 2013), VRCAI '13, ACM, pp. 51–56. 2

[BM10] Bruckner S., Moller T.: Result-driven exploration of simulation parameter spaces for visual effects design. *IEEE TVCG 16*, 6 (Nov. 2010), 1468–1476. 3

[DK11] Davidich M., Koester G.: Towards automatic and robust adjustment of human behavioral parameters in a pedestrian stream model to measured data. In *Pedestrian and Evacuation Dynamics*. Springer US, 2011, pp. 537–546. 3

[DPAM02] Deb K., Pratap A., Agarwal S., Meyarivan T.: A fast and elitist multiobjective genetic algorithm: Nsga-ii. *Evolutionary Computation, IEEE Transactions on 6*, 2 (Apr 2002), 182–197. 7

[FDG*12] Fortin F.-A., De Rainville F.-M., Gardner M.-A., Parizeau M., Gagné C.: DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research 13* (jul 2012), 2171–2175. 7

[GCC*10] Guy S. J., Chhugani J., Curtis S., Dubey P., Lin M., Manocha D.: PLEdestrians: a least-effort approach to crowd simulation. In *ACM SIGGRAPH/Eurographics SCA* (2010), pp. 119–128. 4

[GvdBL*12] Guy S. J., van den Berg J., Liu W., Lau R., Lin M. C., Manocha D.: A statistical similarity measure for aggregate crowd dynamics. *ACM TOG 31*, 6 (2012), 11. 2, 4

[Han11] HANSEN N.: *A CMA-ES for Mixed-Integer Nonlinear Optimization*. Tech. Rep. RR-7751, INRIA, Oct. 2011. 5

[HBJW05] HELBING D., BUZNA L., JOHANSSON A., WERNER T.: Self-organized pedestrian crowd dynamics: Experiments, simulations, and design solutions. *Transp. Science 39*, 1 (2005), 1–24. 2

[HFV00] HELBING D., FARKAS I., VICSEK T.: Simulating dynamical features of escape panic. *Nature 407*, 6803 (2000), 487–490. 2, 3

[HLLO] HUERRE S., LEE J., LIN M., O'SULLIVAN C.: Simulating believable crowd and group behaviors. In *ACM SIGGRAPH ASIA 2010 Courses*, pp. 13:1–13:92. 2

[HMLP13] HA S., MCCANN J., LIU C. K., POPOVIĆ J.: Physics Storyboards. *Computer Graphics Forum* (2013). 3

[HO96] HANSEN N., OSTERMEIER A.: Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation. In *IEEE International Conference on Evolutionary Computation* (1996), pp. 312–317. 5

[JCP*10] JU E., CHOI M. G., PARK M., LEE J., LEE K. H., TAKAHASHI S.: Morphable crowds. In *ACM SIGGRAPH Asia* (2010), pp. 140:1–140:10. 2

[JHS07] JOHANSSON A., HELBING D., SHUKLA P.: Specification of the social force pedestrian model by evolutionary adjustment to video tracking data. *Advances in Complex Systems 10*, supp02 (2007), 271–288. 3

[KMB14] KAPADIA M., MARSHAK N., BADLER N. I.: Adapt: The agent development and prototyping testbed. *IEEE Transactions on Visualization and Computer Graphics 99*, PrePrints (2014), 1. 8

[KOOP11] KULPA R., OLIVIERXS A.-H., ONDŘEJ J., PETTRÉ J.: Imperceptible relaxation of collision avoidance constraints in virtual crowds. In *ACM SIGGRAPH ASIA* (2011), pp. 138:1–138:10. 3

[KSHF09] KAPADIA M., SINGH S., HEWLETT W., FALOUTSOS P.: Egocentric affordance fields in pedestrian steering. In *Proceedings of the 2009 symposium on Interactive 3D graphics and games* (New York, NY, USA, 2009), I3D '09, ACM, pp. 215–223. 2

[KWS*11] KAPADIA M., WANG M., SINGH S., REINMAN G., FALOUTSOS P.: Scenario space: characterizing coverage, quality, and failure of steering algorithms. In *Proceedings of ACM SIGGRAPH/EG SCA* (2011), pp. 53–62. 2, 3, 4

[LCHL07] LEE K. H., CHOI M. G., HONG Q., LEE J.: Group behavior from video: a data-driven approach to crowd simulation. In *Proceedings of ACM SIGGRAPH/EG SCA* (2007), pp. 109–118. 2

[LCL07] LERNER A., CHRYSANTHOU Y., LISCHINSKI D.: Crowds by example. *CGF 26*, 3 (2007), 655–664. 2

[LCSCO10] LERNER A., CHRYSANTHOU Y., SHAMIR A., COHEN-OR D.: Context-dependent crowd evaluation. *Comput. Graph. Forum 29*, 7 (2010), 2197–2206. 2

[LD04] LAMARCHE F., DONIKIAN S.: Crowd of virtual humans: a new approach for real time navigation in complex and structured environments. In *CGF* (2004), pp. 509–518. 2

[LJK*12] LEMERCIER S., JELIC A., KULPA R., HUA J., FEHRENBACH J., DEGOND P., APPERT-ROLLAND C., DONIKIAN S., PETTRÉ J.: Realistic following behaviors for crowd simulation. *CGF 31*, 2 (2012), 489–498. 3

[MCJ12] MUSSE S. R., CASSOL V. J., JUNG C. R.: Towards a quantitative approach for comparing crowds. *Computer Animation and Virtual Worlds 23*, 1 (2012), 49–57. 2

[MLD*08] MCDONNELL R., LARKIN M., DOBBYN S., COLLINS S., O'SULLIVAN C.: Clone attack! perception of crowd variety. *ACM Trans. Graph. 27*, 3 (2008), 26:1–26:8. 3

[NGCL09] NARAIN R., GOLAS A., CURTIS S., LIN M. C.: Aggregate dynamics for dense crowd simulation. In *ACM SIGGRAPH Asia* (2009), pp. 122:1–122:8. 2

[OPOD10] ONDŘEJ J., PETTRÉ J., OLIVIER A.-H., DONIKIAN S.: A synthetic-vision based steering approach for crowd simulation. *ACM Trans. Graph. 29*, 4 (July 2010), 123:1–123:9. 2

[PAB07] PELECHANO N., ALLBECK J. M., BADLER N. I.: Controlling individual agents in high-density crowd simulation. In *ACM SIGGRAPH/EG SCA* (2007), pp. 99–108. 2

[PAB08] PELECHANO N., ALLBECK J. M., BADLER N. I.: *Virtual Crowds: Methods, Simulation, and Control*. Morgan & Claypool Publishers, 2008. 2

[PESVG09] PELLEGRINI S., ESS A., SCHINDLER K., VAN GOOL L.: You'll never walk alone: Modeling social behavior for multi-target tracking. In *Computer Vision, 2009 IEEE 12th International Conference on* (2009), pp. 261–268. 3

[POO*09] PETTRÉ J., ONDŘEJ J., OLIVIER A.-H., CRETUAL A., DONIKIAN S.: Experiment-based modeling, simulation and validation of interactions between virtual walkers. In *ACM SIGGRAPH/EG SCA* (2009), pp. 189–198. 2, 3

[PPD07] PARIS S., PETTRÉ J., DONIKIAN S.: Pedestrian reactive navigation for crowd simulation: a predictive approach. In *EUROGRAPHICS 2007* (2007), vol. 26, pp. 665–674. 2

[Rey87] REYNOLDS C. W.: Flocks, herds and schools: A distributed behavioral model. In *ACM SIGGRAPH* (1987), pp. 25–34. 2

[Rey99] REYNOLDS C. W.: Steering behaviors for autonomous characters. *GDC 1999*, 9602 (1999), 763–782. 2

[SBK*10] SEYFRIED A., BOLTES M., KÃĎHLER J., KLINGSCH W., PORTZ A., RUPPRECHT T., SCHADSCHNEIDER A., STEFFEN B., WINKENS A.: Enhanced empirical data for the fundamental diagram and the flow through bottlenecks. In *Pedestrian and Evacuation Dynamics 2008*. Springer Berlin Heidelberg, 2010, pp. 145–156. 2, 4

[SGA*07] SUD A., GAYLE R., ANDERSEN E., GUY S., LIN M., MANOCHA D.: Real-time navigation of independent agents using adaptive roadmaps. In *ACM VRST* (2007), pp. 99–106. 2

[SKFR09] SINGH B. S., KAPADIA M., FALOUTSOS P., REINMAN G.: Steerbench : a benchmark suite for evaluating steering behaviors. *Computer Animation And Virtual Worlds 20*, February (2009), 533–548. 2

[SKH*11] SINGH S., KAPADIA M., HEWLETT B., REINMAN G., FALOUTSOS P.: A modular framework for adaptive agent-based steering. In *ACM I3D* (2011), pp. 141–150. 2, 3

[TCP06] TREUILLE A., COOPER S., POPOVIĆ Z.: Continuum crowds. *ACM Trans. Graph. 25*, 3 (2006), 1160–1168. 2

[TM13] THALMANN D., MUSSE S. R.: *Crowd Simulation, Second Edition*. Springer, 2013. 2

[vdBGLM11] VAN DEN BERG J., GUY S. J., LIN M., MANOCHA D.: Reciprocal n-body collision avoidance. In *Robotics Research*, vol. 70. 2011, pp. 3–19. 2, 3

[WFH10] WANG J. M., FLEET D. J., HERTZMANN A.: Optimizing walking controllers for uncertain inputs and environments. *ACM Trans. Graph. 29*, 4 (July 2010), 73:1–73:8. 3

[WGO*14] WOLINSKI D., GUY S., OLIVIER A.-H., LIN M., MANOCHA D., PETTRÉ J.: Parameter estimation and comparative evaluation of crowd simulations. In *Eurographics* (2014). 3