Crowd Space: A Predictive Crowd Analysis Technique

IOANNIS KARAMOUZAS, Clemson University
NICK SOHRE, University of Minnesota
RAN HU, Facebook & University of Minnesota
STEPHEN J. GUY, University of Minnesota

Over the last two decades there has been a proliferation of methods for simulating crowds of humans. As the number of different methods and their complexity increases, it becomes increasingly unrealistic to expect researchers and users to keep up with all the possible options and trade-offs. We therefore see the need for tools that can facilitate both domain experts and non-expert users of crowd simulation in making high-level decisions about the best simulation methods to use in different scenarios. In this paper, we leverage trajectory data from human crowds and machine learning techniques to learn a manifold which captures representative local navigation scenarios that humans encounter in real life. We show the applicability of this manifold in crowd research, including analyzing trends in simulation accuracy, and creating automated systems to assist in choosing an appropriate simulation method for a given scenario.

CCS Concepts: • **Computing methodologies** → *Animation*; *Multi-agent planning*; Scene understanding; • **Mathematics of computing** → *Dimensionality reduction*;

Additional Key Words and Phrases: Crowd simulation, validation, entropy, manifold learning

ACM Reference Format:
Ioannis Karamouzas, Nick Sohre, Ran Hu, and Stephen J. Guy. 2018. Crowd Space: A Predictive Crowd Analysis Technique. *ACM Trans. Graph.* 37, 6, Article 186 (November 2018), 14 pages. https://doi.org/10.1145/3272127.3275079

1 INTRODUCTION

The last two decades have seen a dramatic rise in the number of techniques used in simulating real-world phenomena, from animating fluid and sound to modeling hyperelastic materials and human crowds [Bridson 2015; Lee 2010; Pelechano et al. 2016; Sifakis and Barbic 2012]. In many fields, this growth in choice of simulation methods raises many questions for domain experts in choosing appropriate simulation techniques. This is especially true in the field of human crowd simulation, where there are a large number of simulation methods that all preform well in some scenarios, but none of which performs "best" in all scenarios. However, choosing the right simulation method can be very important. When crowds are used in games and movies, poor simulation performance can lead to unnatural crowd behavior, quickly breaking the immersion. When crowd simulations are used to guide event planning and building design, poor simulation models can lead to bad practices affecting thousands of people.

Domain experts in crowd simulations, as in many fields, have a wealth of informal knowledge about what simulation approaches to use when. Many experts will suggest not to use reactive methods such as social forces or boids [Helbing et al. 2000; Reynolds 1987] unless the simulation task involves a small number of agents in sparse scenarios. Likewise, many researchers tend to feel geometric methods such as ORCA [van den Berg et al. 2011] are likely to struggle in medium density scenarios where agile maneuvers are important, but do well in dense scenarios where efficient accounting for constraints is key. This kind of informal, unquantified knowledge is crucial to the proper choice of simulation method, but cannot always be reliably communicated in an objective and universally accesible fashion.

In this paper, we propose to leverage data-driven learning techniques to develop a method to compactly represent this type of meta-information about crowd simulation scenarios. To this end, we introduce *Crowd Space*, a low-dimensional manifold learned from human trajectories, which represents the space of likely interaction scenarios that humans can encounter (see Figure 1). We also learn a continuous labeling of this manifold based on a novel method that allows us to evaluate the local simulation accuracy.

Ultimately, our work provides a method for formalizing and quantifying knowledge about what types of simulation techniques to use in different scenarios. To that end, we propose the following contributions:

- *Crowd Space.* An agent-centric, low-dimensional manifold representing a large variety of crowd interaction scenarios.
- An agent-centric formulation of the entropy metric of [Guy et al. 2012] that allows the estimation of a simulation's local accuracy.
- An analysis of key trade-offs between simulation methods, along with a learning-based approach to automatically predict the best (i.e., most accurate) simulation methods to use in a given scenario.

2 RELATED WORK

2.1 Multi-Agent Navigation Methods

There is a significant amount of research in robotics, traffic engineering, and computer graphics on planning paths for multiple agents such as virtual characters, pedestrians, and cars. The most common approach is to decompose global planning from local collision avoidance. Typically, a graph is used to capture paths which are collision-free with respect to the static part of the environment,

Authors' addresses: Ioannis Karamouzas, Clemson University, ioannis@clemson.edu; Nick Sohre, University of Minnesota, sohre007@umn.edu; Ran Hu, Facebook & University of Minnesota, huran.carol@gmail.com; Stephen J. Guy, University of Minnesota, sjguy@umn.edu.

0730-0301/2018/11-ART186 $15.00
https://doi.org/10.1145/3272127.3275079

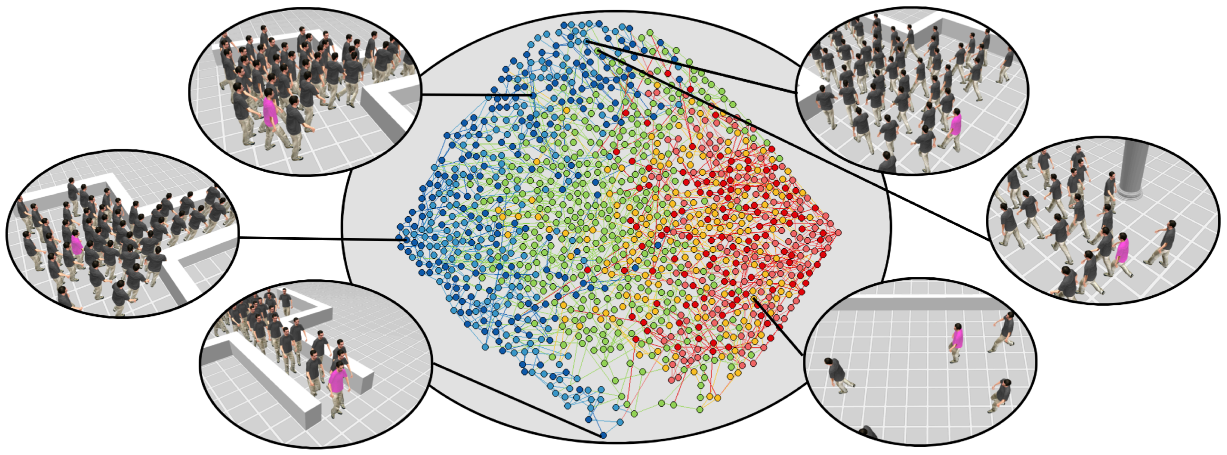

Fig. 1. Crowd space embedding using PCA. The space is obtained by sampling 15,000 60D mpd-based local scenario descriptors gathered from six training datasets (2,500 descriptors per dataset), and taking the first 6 principal components. Here, we visualize this 6D space by random sampling of non-overlapping descriptors and plotting them according to their first two principle components. The resulting space clusters together agents with similar local conditions, even if they come from different datasets. In this 2D projection, the horizontal axis tends to correspond to density (more dense on the left) and the vertical axis whether the agent is entering, leaving, or located in the middle of congestion (top, bottom, middle, respectively). The edges in the 2D embedding denote nearest neighbors.

and a local collision avoidance technique is then used to navigate each agent around other nearby agents and obstacles. Our work here will focus on the local navigation, as many different methods rely on the same global planning approaches (e.g., artist-designed roadmaps), and variation in global planning tend to have less influence on the collective behavior of the agents than variation in local planning. (We refer the reader to the recent books of [Kapadia et al. 2015; Pelechano et al. 2016] for a more extensive review of state-of-the-art local navigation and global planning approaches.)

While a large variety of local navigation methods have been proposed, many popular methods can be classified into one of five classes: a) *reactive* approaches where agents rely primarily on distance-based functions to avoid collisions with nearby agents and obstacles [Helbing et al. 2000; Helbing and Molnár 1995; Reynolds 1999]; b) *geometrically-based* approaches where agents compute collision-free velocities using sampling or optimization [Pettré et al. 2009; van den Berg et al. 2011, 2008], c) *vision-based* approaches where the steering behavior of each agent is directly correlated to its (simulated) visual stimuli [Hughes et al. 2015; Kapadia et al. 2012; Moussaïd et al. 2011; Ondřej et al. 2010]; d) *gradient-based* approaches where each agent tries to independently minimize an energy or cost function that accounts for future collisions [Dutra et al. 2017; Karamouzas et al. 2014; Wolinski et al. 2016]; and e) *example-based* approaches that use human data to update the velocities of simulated agents [Charalambous and Chrysanthou 2014; Ju et al. 2010; Lerner et al. 2007]. Here, we do not consider example-based approaches directly, as their accuracy depends directly on how closely their training dataset matches our testing dataset, rather than how well their mathematical models capture human behavior.

There are, of course, navigation methods that do not fit easily into one of these categories. For example, some methods focus on explicitly accounting for grouping and other social dynamics [Ren et al. 2017], level-of-detail methods [Kulpa et al. 2011], and methods that closely model the biomechanics of body orientation and footstep patterns [Singh et al. 2011; Stuvel et al. 2016]. Furthermore, with recent advancements in deep learning, approaches have been introduced that seek to improve the collision avoidance behavior and the locomotion of the agents [Holden et al. 2017; Long et al. 2017; Peng et al. 2017].

2.2 Data-driven evaluation and analysis

Perhaps the most intuitive way to analyze the accuracy of a simulated crowd is by defining a number of test-case scenarios along with evaluation metrics that can be used to collect statistics about the behavior of the agents. For example, Reitsma and Pollard [2007] used task-based metrics to evaluate the animation capabilities of virtual agents, while the work in [Kapadia et al. 2009, 2011; Singh et al. 2009] focus on the steering behavior of the agents. Such metrics have also been used by data-driven evaluation techniques to determine how similar are simulated pedestrians to real ones extracted from video crowd footage [Lerner et al. 2010]. This include approaches that automatically detect outlying behavior [Charalambous et al. 2014] and optimization-based techniques that tune the parameters of the agents to enable simulations that best fit the given data [Berseth et al. 2014; Wolinski et al. 2014]. Evaluation techniques that focus on the macroscopic behavior of the agents have also been proposed, such as the work in [Wang et al. 2017] that uses Bayesian inference to learn path patterns and compare them to reference data, and the entropy metric work [Guy et al. 2012] that estimates in a robust way the similarity between the aggregate motion of real pedestrian and simulated trajectories.

Our work is complementary to the crowd evaluation techniques mentioned above. In particular, we extend the entropy work to focus on the predictive accuracy of a simulation method at a per-agent level. Furthermore, as compared to previous solutions, our method learns to automatically predict the most accurate simulation methods for a given interaction scenario. We note that there is also a significant amount of work on perceptually evaluating simulations. In particular, user studies have been proposed in different fields to assess the visual quality of videos, rendering techniques, and fluid simulation methods (see, e.g., [Aydin et al. 2010; Um et al. 2017]. Furthermore, perceptual studies have been extensively used in the field of character animation and crowd simulation, with recent work focusing on the visual realism and expressiveness of animated crowds [Durupinar et al. 2017; Hoyet et al. 2013; McDonnell et al. 2009]. While related to our work, the goal of these approaches is somewhat orthogonal to ours, as they focus on the perceptual quality rather than the predictive accuracy of simulations.

3 BACKGROUND AND DEFINITIONS

While the question of how to analyze different crowd simulation techniques can take many forms, our work focuses on the following two questions: i) Given a simulation method, in what crowd scenarios does this method perform the best? And ii) Given a particular crowd scenario, which methods produce the most accurate simulations?

Formally, we define the above terms as:

- **Crowd Scenario**: A crowd simulation scenario, $\mathcal{S}$, consists of a set of n agents, each with its own initial position, goal position, goal speed, radius, and (optionally) a set of obstacles demarcating impassable regions.
- **Simulation Method:** A simulation method is a function which takes as input a *crowd scenario*, and returns as output a trajectory for each agent.
- **Simulation Accuracy:** The accuracy of a *simulation method* is a measure of how closely the resulting trajectories match those which would have been produced by humans in the same *crowd scenario.*

In particular, we note that there are many different plausible measures of simulation accuracy (see, e.g., [Singh et al. 2009; Um et al. 2017]). Here, we focus on the notation of *predictive* accuracy as introduced by the Entropy Metric [Guy et al. 2012], that is: given a representative collection of real human trajectories, what is the likelihood that a given simulation method would produce similar trajectories in similar circumstances.

3.1 Generating Scenarios from Human Data

To best estimate the accuracy of a given technique in a particular crowd scenario, we should have access to a collection of representative trajectories that real humans took in the same (or very similar) situations. While it would be best to randomly sample a well distributed range of crowd scenarios and ask human subjects to enact these situations, this approach is impractical to achieve at scale. Instead, we invert this strategy and infer likely crowd scenarios given previously observed trajectories from real-world crowd videos. By breaking real-world crowd datasets and videos into small time steps we can infer hundreds of new crowd scenarios by estimating a position, velocity, radius, goal position and goal speed for each agent. Appendix A gives details on how we estimate each of these terms. Importantly, deriving crowd scenarios from real-world data allows us to know what paths real people took in each of these scenarios.

Crowd Datasets. Because all of our crowd scenarios are derived from real-world datasets, it it is important to choose a large, varied collection of human data for training. Here, we use ten different human trajectory datasets representing efforts from a variety of researchers, crowd densities, motion heterogeneity, and so on. These include: i) sparse crowds (average space-time density less than 0.5 ppl/m^2) captured at a snowy intersection [Pellegrini et al. 2009] and in front of a church [Lerner et al. 2010], labeled *Snow_eth* and *Church*, respectively; ii) Medium-density (between 0.5-2 ppl/m^2) groups of shoppers walking down a street at two different times of day, *Zara1* and *Zara2*, and students on a college campus, *Students* [Lerner et al. 2007]; iii) Dense groups (average density greater than 2 ppl/m^2) of participants in various scenarios including walking through differently shaped bottlenecks, labeled *Bottleneck* and *Bottleneck-long* [Seyfried et al. 2009], interacting as two large streams moving past each other through a train station hallway, *Berlin-180*, and two large streams of people interacting perpendicularly *Berlin-90* [Lämmel and Plaue 2014]; iv) And, finally, results from experiments with small groups walking simultaneously down a hallway, *One-way* [Ju et al. 2010].

3.2 Simulation Methods

We investigate several different local navigation methods, representing a wide variety of different simulation techniques. In general, we selected methods which are either recently proposed or in wide-spread use. This includes reactive *Social Forces* [Helbing et al. 2000], the geometrically-based methods of *T-Model* [Pettré et al. 2009] and *ORCA* [van den Berg et al. 2011], the *Vision-based* method of [Moussaïd et al. 2011] which uses a sampling-based approach to optimize an anticipatory energy function, and predictive force-based approaches of *PAM* [Karamouzas et al. 2009] and *TTC Forces* [Guy and Karamouzas 2015]. Where possible, we obtained the latest, revised implementation directly from the authors, though in the case of the *Social Forces* and *Vision-based* models we provided our own implementation.

Note, we exclude the PowerLaw model proposed in [Karamouzas et al. 2014] as it was explicitly trained on part of the data used to create our learning method. Likewise, we avoid data-driven approaches such as the ones proposed in [Charalambous and Chrysanthou 2014; Lerner et al. 2007] due to similar concerns with over-fitting. Finally, we do not consider continuum-based solutions or those which require explicit global solvers such as those proposed in [Karamouzas et al. 2017; Narain et al. 2009; Treuille et al. 2006] as these methods tend to be too slow for the entropy-based accuracy analysis we employ (Section 4) and can have difficulty with agents who enter and exit a simulation.

3.3 Organization

The next section details our approach to evaluating the time-varying accuracy with which a simulation captures the behavior of real-world pedestrians in various settings. In Section 5, we detail our approach to building a crowd scenario manifold to facilitate simulation analysis. In Section 6, we use the resulting Crowd Space to analyze the relative accuracy of the aforementioned simulation methods in various scenarios. Finally, Section 7 introduces a method to automate the process of predictive crowd analysis by formulating it as a multi-label classification problem.

4 AGENT-CENTRIC ENTROPY

In order to facilitate our predictive analysis of various crowd simulation techniques, we must first establish an evaluation metric. Here, we wish to focus on the accuracy of a simulation technique; that is, how closely a simulation matches the behavior of real pedestrians in similar scenarios. Much of the previous work in this area has focused on studying indirect measures of accuracy, such as comparing the local density of simulated agents to those found in database of typical human motion as in [Lerner et al. 2010] and [Charalambous et al. 2014], or has used user-defined metrics to evaluate a path such as smoothness or number of collisions [Kapadia et al. 2009; Singh et al. 2009]. While these approaches are valuable metrics of path quality, they do not directly address the predictive accuracy a simulation method may have in a given scenario. Closer to our goal is the *entropy* metric proposed in [Guy et al. 2012], which uses a stochastic inference framework to directly estimate prediction accuracy of a method given a set of pedestrian trajectories in a manner which is robust to the large amount of sensor noise present in real-world datasets. However, the original formulation of this metric has two issues we seek to address:

(1) *Fixed Simulation Size.* The original entropy metric assumes the number of agents is static throughout a dataset. In practice, agents will likely enter and leave the scene quite frequently in any dataset lasting more than a few seconds.
(2) *Global Accuracy Estimates.* The original entropy metric provides a single accuracy estimate per simulation method per dataset. In reality, the accuracy of a simulation varies spatially (per-agent), and over time (for a given agent).

In this section, we propose a per-agent entropy metric which (1) allows agents to enter and exit a scene over time and (2) provides a per-agent, ego-centric estimate of how well a given crowd simulation method would predict the motion of each agent in the crowd. Together, these changes remove the fundamental heterogeneity assumption found in the original entropy metric. As a result, we can now identify localized problem regions within an otherwise high-quality simulation method, greatly improving the metric's applicability for crowd analysis.

4.1 Approach Overview

Similar to the original entropy metric approach, we seek to measure the degree to which a simulation method is likely to produce the behavior captured in a real-world dataset. As in [Guy et al. 2012], we follow an information-theoretic formulation, and treat measuring accuracy as a likelihood estimation problem. The overall approach to computing the entropy metric is as follows: first, decompose the observed crowd data into several single time step prediction tasks by using Bayesian filtering (e.g. Kalman filtering); second, add zero-mean Gaussian noise with covariance Q to each simulation step and run localized simulations per-time-step to estimate the effect of Q; and third, estimate the size and shape of Q which maximizes the likelihood of the observed data via Expectation Maximization (EM). The smaller the Gaussian Q needed to reproduce the data, the more accurate the simulation method.

The key difference from our approach and [Guy et al. 2012] is in how the dataset is decomposed into the per-time-step prediction tasks. Guy et al. represented the simulation state as a joint state over all agents in an entire frame of a simulation (necessitating a fixed number of agents throughout the data); here, we instead estimate a time-varying value of Q for each agent independently. This is done by treating each agent as a *central agent* in a one-off simulation with all of its nearby neighbors. The process is run independently for each agent, and is used to generate a value of Q per central agent, per-time-step (as described below). To produce a final accuracy estimate, we accumulate values of Q across several seconds and refer to the average of these Q value as $\widetilde{Q}$. The per-agent, per-time step accuracy score is then the entropy of this distribution:

$$e(\widetilde{Q}) = \frac{1}{2} log((2\pi e)^d det(\widetilde{Q})), \tag{1}$$

where d is the dimensionality of $\widetilde{Q}$.

It is important to stress that even though we produce a new accuracy estimate for each agent, for each time step our modified entropy metric is not estimating the instantaneous accuracy for this time step, rather it is estimating the accuracy over the entire timerange over which we accumulate values to estimate $\widetilde{Q}$. Here, we use a 5 s window as most agents in our dataset have tracked interactions for 5 s or less.

4.2 Computing the Per-Agent Entropy Metric

The approach we use to decompose real crowd data into small simulation tasks is detailed in Algorithm 1. Here, we use an Ensemble Kalman Smoother (EnKS) as a state estimation technique [Evensen 2003]. Our EnKS-based approach works by iteratively estimating the most likely internal agent state $\mathbf{a}_k$ at timestep k based on external observations over the entire dataset. We assume we know the position observation function $\mathbf{h}$, the trajectory observations $\mathbf{z}$ from both before and after the current timestep, an estimate of the state at the previous timestep $\mathbf{a}_{k-1}$, and a prediction function f which advances the state to the next timestep. Here, the function f is the crowd simulation under consideration by the entropy metric.

EnKS takes a two stage approach of first predicting the new state for the next timestep, $f(\mathbf{a}_k)$, based on the prediction function (predict step), and then correcting this prediction based on the observations (correct step). The effect of simulation uncertainty is modeled through the use of noise. Rather than running a single simulation to predict the next agent state, m simulations are run each with a sample of Gaussian noise with covariance $\widetilde{Q}$ added to the prediction. An agent's state, then, is actually represented by an ensemble of different states. (A similar step adds noise of fixed size R to the observations.) At each timestep, the correct step fits a Gaussian to this ensemble of samples, and updates the state estimation based on the Kalman gain equation.

After the ensemble of states have been estimated via EnKS, we can estimate the noise covariance, Q, which would maximize the

ALGORITHM 1: EnKS for Estimating Agent's States

```
Input  : Measured Real-World States $Z = \{z_{t_0} \cdots z_{t_p}\}$, Appearance
         Period $p$, Crowd Simulation technique $f$, Estimated Average Error
         Variance $\widetilde{Q}$
Output : Estimated Agent State Distribution $\hat{A} = \{\hat{a}_{t_0} \cdots \hat{a}_{t_p}\}$
foreach $k \in t_1 \cdots t_p$ do
    // Predict Step
    foreach $i \in 1 \cdots m$ do
        Draw $q_{k-1}^{(i)}$ from $\widetilde{Q}_P$
        $\hat{a}_k^{(i)} = f(\hat{a}_{k-1}^{(i)}) + q_{k-1}^{(i)}$
        Draw $r_k^{(i)}$ from $R$
        $\hat{z}_k^{(i)} = h(\hat{a}_k^{(i)}) + r_k^{(i)}$
    $\bar{z}_k = \frac{1}{m}\sum_{i=1}^{m} \hat{z}_k^{(i)}$
    $Z_k = \frac{1}{m}\sum_{i=1}^{m} (\hat{z}_k^{(i)} - \bar{z}_k)(\hat{z}_k^{(i)} - \bar{z}_k)^T$
    // Correct Step
    foreach $j \in 1 \cdots k$ do
        $\bar{a}_j = \frac{1}{m}\sum_{i=1}^{m} \hat{a}_j^{(i)}$
        $\sum_j = \frac{1}{m}\sum_{i=1}^{m} (\hat{a}_j^{(i)} - \bar{a}_j)(\hat{z}_k^{(i)} - \bar{z}_k)^T$
        foreach $i \in 1 \cdots m$ do
            $\hat{a}_j^{(i)} = \hat{a}_j^{(i)} + \sum_j Z_k^{-1}(z_k - \hat{z}_k^{(i)})$
```

ALGORITHM 2: Maximum Likelihood Estimation for an Agent

```
Input  : Estimated State Distribution $\hat{A}_P = \{\hat{a}_{t_0} \cdots \hat{a}_{t_p}\}$
Output : Estimated Average Error Variance $\widetilde{Q}_P = \{\widetilde{Q}_{t_0} \cdots \widetilde{Q}_{t_p}\}$
foreach $k \in t_0 \cdots t_p$ do
    $Q_k = 0$ , $\widetilde{Q}_k = 0$
foreach $k \in t_0 \cdots t_{p-1}$ do
    foreach $i \in 1 \cdots m$ do
        $Q_k += \frac{1}{m}(\hat{a}_{k+1}^{(i)} - f(\hat{a}_k^{(i)}))(\hat{a}_{k+1}^{(i)} - f(\hat{a}_k^{(i)}))^T$
// Average Window
foreach $k \in t_0 \cdots t_p$ do
    $\widetilde{Q}_k = \frac{1}{2w+1}\sum_{I=k-w}^{k+w} Q_{(P,I)}$
```

likelihood of the observed trajectory using Maximum Likelihood Estimation (Algorithm 2). This gives us a new $\widetilde{Q}$ which we can now apply in a new round of EnKS. We can run this process until the estimate for $\widetilde{Q}$ converges. (Note an agent may serve as a non-central agent for several other central agents, but this will not change its Q value.) Once $\widetilde{Q}$ is known we can measure its size using Equation 1, providing us with the final estimate of the simulation error.

Importantly, this approach assumes that the error between the predicted state and the observed state can be well modeled by a Gaussian when performing the Maximum Likelihood Estimation. We can test the validity of this hypothesis using a Kolmogorov-Smirnov test [Massey Jr 1951]. Figure 2 shows p-values of the Gaussian fitness test for simulation errors when using ORCA as the prediction function, distributed across 10 randomly selected agents extracted from four datasets (Zara1, Students, Berlin-90, and One-way). Across the 2,000 collected samples (one per agent, per time step), almost none was able to reject the null (Gaussian) hypothesis at the 5% significance level, which indicates that the error between observed and predicted state is generally Gaussianly distributed. In addition, the Gaussians show a strong tendency to be zero-meaned (e.g., the average $\mathbf{p}_x$ error has a mean value of 0.011 ± 0.057), suggesting an unbiased Q. Appendix B shows similar results for additional simulation methods and provides further analysis.

Implementation Details. In our results, we represent each agent's simulation state A with an ensemble of m=2000 simulation state samples for each time step $\{\hat{a}_{t_0} \cdots \hat{a}_{t_p}\}$, and run 3 iterations of EnKS and MLE for each agent each time step. Each agent is assumed to exist continuously from time t_0 until time t_p; if an agent exits and reenters, it is treated as a separate agent. We use the current position

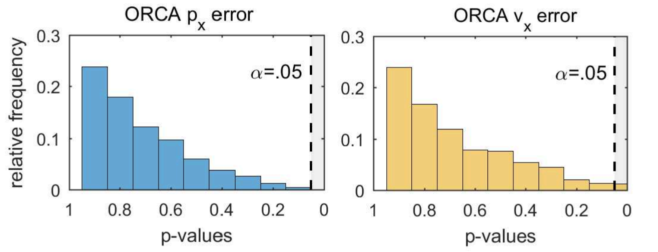

Fig. 2. Analysis of Gaussian fit in four datasets based on one-sample Kolmogorov-Smirnov test. Here, almost all ORCA agents fail to reject the null hypothesis of a Gaussian model of error distribution at $\alpha = .05$, suggesting the true distribution is well approximated by a Gaussian.

and velocity for each time step t, and assume the goal position is externally estimated as discussed in Section 3.1.

The prediction function f is executed over a period of 125ms to match the framerate of the pedestrian datasets. For each central-agent, we choose all neighbors within a 5 m radius to use as non-central simulation agents. In theory, entropy scores have no lower bound (a perfect simulation would have a score of negative infinity). In practice, we observe only finite values which we scale such that all scores lie between -2 and 2.

4.3 Metric Analysis

Our modified approach to the entropy metric allows for a per-agent, per-time step analysis of simulation accuracy, while retaining many of the key robustness properties of the originally proposed metric of [Guy et al. 2012]. This is due to the fact that the entropy metric compares the predictions made by a given simulation technique to the distribution of decisions latent in a given dataset; this is in contrast to other crowd analysis techniques which focus on comparing predicted trajectories to observed trajectories.

As a point of comparison, we consider two alternative classes of trajectory-based error estimation: i) a *distance-based* metric where the error is the L2 distance between the observed and predicted trajectories at any time step (as was used in [Lerner et al. 2010]); and ii), a *neighbor-based* metric where the error is based on the distance between an agent and its nearest neighbor as compared to the relative

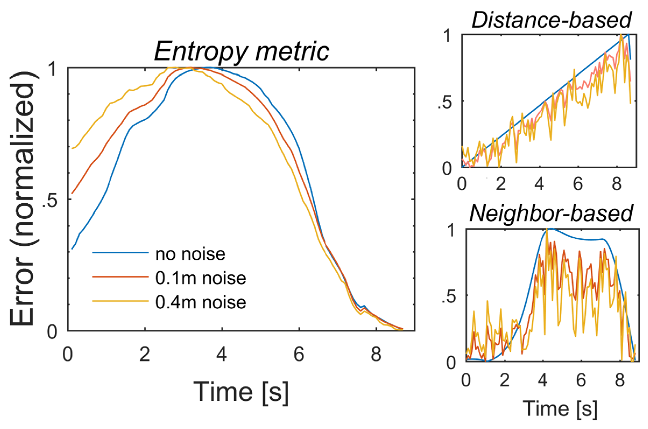

Fig. 3. A comparison of the time-varying error in an ORCA simulation as measured by our entropy metric, an absolute distanced-based error metric, and a nearest neighbor-based error metric in a two-person crossing scenario from [Pettré et al. 2009]. All graphs are normalized to run from 0 to 1 and report the mean error of the two agents.

distances seen in the reference data (as was used in [Charalambous et al. 2014]). Figure 3 shows results from these three different types of metrics on 2-person interaction data from [Pettré et al. 2009] where two pedestrians cross paths perpendicularly to each other. The corresponding simulations were obtained using ORCA.

Comparing the different metrics reveals a fundamental difference in how they measure error. The distance-based metric allows a small but consistent error in the predicted speed of the agents to accumulate over time revealing a strong sensitivity to estimating the initial state correctly and showing that the metric fails to capture the notion of the instantaneous error that we seek in the entropy metric. In contrast, both the entropy metric and the neighbor-based metric capture a peak in error at the point were the two pedestrians are most interacting (~4s). However, the neighbor-based metric continues to show a high error well after the agents have passed each other, whereas the entropy metric quickly approaches zero. This ability to estimate the current simulation accuracy without being overly influenced by past behavior will be exploited later in Sections 6 and 7 to allow us to treat each frame of the training data as an independent dataset.

Robustness to Noise. To determine how robust different metrics are to sensor noise, we considered the previous scenario and added independent and identically distributed noise to the positions of the reference pedestrians. Figure 3 shows the corresponding results for different noise magnitudes v, sampled from the uniform distribution on a disc centered at $\mathbf{0}$ with radius v. Note that the graphs have all been normalized to run from 0 to 1 in order to focus on the shape of the estimated error over time. Both the distance-based and neighbor-based approaches show very noisy outputs as a result of the input noise. However, the entropy metric shows very little sensitivity to this input noise, retaining a smoothly varying estimate of error with an almost identical point of “peak” error even with a large amount of noise added to the data.

Table 1. Both real humans paths and crowd simulations can change drastically under small perturbations of initial state. In the 6-person circle scenario, we estimate the effect of small perturbations for various metrics by adding a small offset to the start and goal position in the data, and looking at the coefficient of determination (r^2) of the time varying errors estimated by various metrics with and without these perturbations. The entropy metric is more robust to these perturbations than the two other metrics. Likewise, mirroring the paths taken by two people swapping positions creates an equally likely trajectory, and the time-varying entropy metric values after this mirroring are highly correlated with the values derived from the unmodified data.

Scenario	Data Modification	Dist	NN	Entropy
6-person circle	Perturbed .1m	.89	.52	.97
6-person circle	Perturbed .4m	.83	.20	.92
2-person swap	Mirrored paths	.75	.97	.99

Non-determinism in Human Motion. Two different people given the exact same navigation task (i.e., same start, goal, and neighbors) might make very different decisions. The entropy metric accounts for this non-determinism by treating the reference human data as a single sample from the true distribution of plausible pedestrian responses to a given scenario. The results of this can be seen if we apply small perturbations to the start and goal positions in reference data. Here, we look at a challenging scenario from [Wolinski et al. 2014] where 6 people walk to antipodal points in a circle. Because of the near-symmetry of the scenario, small perturbations in the initial position will cause the simulations (or real people) to significantly change their path to the goal (e.g., clockwise vs. counter-clockwise motion). Table 1 shows the correlation of the errors reported by the various metrics with and without these perturbations using ORCA as simulation. Here, the entropy metric shows very high correlation between the results with and without the perturbations showcasing its robustness to these types of large changes; our metric retains its high correlation because it treats the reference data as a sample from a distribution of likely trajectories, rather than as the ground truth. Finally, we can complete a similar experiment to that shown in [Guy et al. 2012] where we use as reference data trajectories of two agents exactly swapping positions, passing each other on the right. By exactly mirroring the velocities of the agents, we can generate a different, but equally plausible, set of trajectories for the same start-goal pair (where agents now pass on the left). Table 1 reports the correlation between the Vision-based simulation error when computed against the original trajectories and when against the mirrored trajectories. Where the distance-based metric shows a big change in the error (due to the change in passing side), the entropy metric shows nearly perfect correlation between errors.

4.4 Applications

Because our formulation generates a per-agent, per-timestep estimation of simulation accuracy, it allows for applications not possible with the original entropy metric. For example, our metric can be used to detect hot-spots in a dataset that are poorly captured by a simulation. Figure 4 shows the heatmap of our agent-centric entropy scores for ORCA on the Bottleneck dataset. Our method reveals that ORCA works well in the center of the hallway flow, but

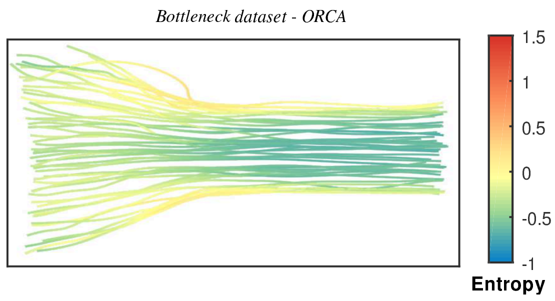

Fig. 4. Our improved entropy metric provides spatially (and temporally) varying estimates of simulation error (lower is better). Here, an ORCA simulation of the Bottleneck scenario shows almost no error in the center of the bottleneck (green), but matches the data less well near the edges of the hallway (yellow).

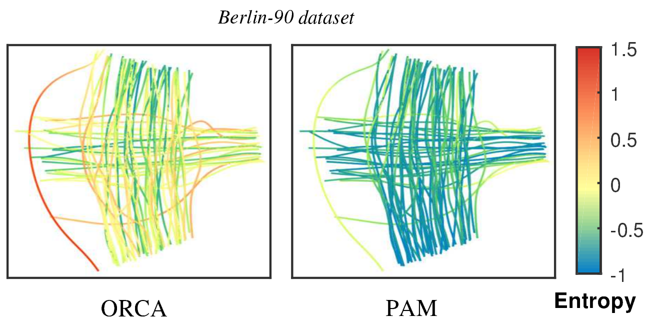

Fig. 5. A comparison between ORCA and PAM in the Berlin-90 scenario. While ORCA matches some agents well, PAM has consistently lower error.

has higher error for agents who are walking close to the walls of the bottleneck. Figure 5 shows a comparison between two simulations on the Berlin-90 dataset. Here, we can see ORCA and PAM work equally well for some of the trajectories (i.e., both with very low error), but for many agents flows are better captured by PAM, especially for those on the edges of the crowd. This type of analysis can also be used for outlier detection (as in [Charalambous et al. 2014]) and can be applied to help in per-agent tuning of simulation parameters (see supplemental video and Appendix D).

5 CROWD SPACE

In addition to estimating how accurate a simulation is for a given crowd state, we would also like to represent the state of the crowd in a compact and generalizable fashion. To this end, we introduce Crowd Space, an agent-centric manifold representing the local conditions in a crowd. There are two key features which we seek in this manifold. One: it should provide a natural way to represent a wide range of crowd scenarios. Two: similar crowd scenarios should lie near each other on the manifold (to provide a clear semantic interpretation of local regions of the manifold).

Here, we use the human trajectories from Section 3.1 to build the Crowd Space. Our approach requires two steps. First, we must find a structured descriptor which provides a fixed-width vector representation of the current crowd state. Secondly, we apply a dimensionality reduction technique to automatically produce a low-dimensional representation which faithfully captures the higher dimensional structure. The result is a compact representation of the state of the crowd, appropriate for visual analysis, and prediction tasks. We described each of these steps in turn.

5.1 Per-Agent Crowd Descriptor

There are many challenges in representing the current state of the crowd with a single fixed-size descriptor. For one, the state of the crowd can be very different across different regions of space. Additionally, as the size of the crowd grows and shrinks, the size of the descriptor would need to change to maintain the same fidelity in description. Here, we propose to overcome both of these challenges by using a separate egocentric representation of the crowd state for each agent. The full crowd is then represented by an unordered collection (i.e., a set) of these per-agent descriptors. That is, rather than storing the global crowd structure explicitly, our approach relies on the emergent effect this structure has on each individual agent's local conditions.

Agent State Descriptor. We use a local descriptor of a given agent's local crowd state that is based on the oriented, spatially varying, relative distances and velocities to nearby agents. In this way, we can capture both how the local density changes around the agent (based on positions) and how it is likely to change in the near future (from relative velocities). In practice, we found that it was sufficient to combine the relative distance and velocity in a single quantity known as the *minimal predicted distance* (MPD) between an agent and its nearby neighbors [Olivier et al. 2012]. Here, we compute the MPD as the closest distance between the two interacting parties assuming a linear extrapolation of their goal velocities (we assume agent goal velocities, but not instantaneous velocities, are part of the crowd scenario specification). Formally, given an agent A, the MPD to its nth neighbor is computed as:

$$\min_{\tau \geq 0} \|(\mathbf{p} - \mathbf{p}_n) + (\mathbf{v} - \mathbf{v}_n)\,\tau\|, \tag{2}$$

where $\mathbf{p}$ and $\mathbf{v}$ denote the current position and goal velocity of the agent A, and $\mathbf{p}_n$ and $\mathbf{v}_n$ denote the position and goal velocity of the neighbor, respectively.

In order to capture how MPD varies spatially, we use an egocentric descriptor, $\mathbf{x}$, which records the local crowd conditions from the perspective of the agent as MPD values along evenly-spaced radial intervals around the agent A. Similar to many visibility-based approaches [Charalambous and Chrysanthou 2014; Kapadia et al. 2012; Ondřej et al. 2010], we consider a local neighborhood around A, and assume a local coordinate system centered at the agent and oriented towards its goal velocity. As such, in constructing the agent descriptor, $\mathbf{x}$, the first ray is always oriented towards the agent's goal direction (and the rest sampled in a clockwise fashion) so as to provide a meaningful overall orientation to the rays. For each interval we determine the closest agent or static obstacle to agent A using ray tracing. If such a neighbor exists, we store the MPD to this agent.

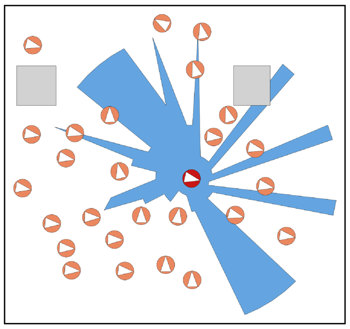

Fig. 6. Example of a minimal predicted distance (MPD) descriptor that captures the crowd scenario as seen from the perspective of the red agent.

Similar to [Charalambous and Chrysanthou 2014], we found 60 angular segments to be sufficient for estimating the agent's descriptor. In addition, we only consider neighboring agents that are within a 5 m radius from the agent A, and static obstacles that are 1 m away. If no neighbor is traced along a given ray, the corresponding MPD value is set to a default maximum of 5 m. We also clamp the MPD values to the maximum value, and normalize the results accordingly. Formally, the descriptor $\mathbf{x} \in \mathbb{R}^{60}$ of an agent is a vector of 60 MPD values, where each value denotes a mapping $[-\pi, \pi] \mapsto [0, 1]$. In total, we extracted 7,946 frames from the ten crowd datasets described in Section 3.1. From the 7,946 scenarios, we computed 68,086 descriptors. As an example, Figure 6 visualizes the descriptor corresponding to the red highlighted agent.

Importantly, this MPD-based formulation uses only information that is available in the crowd scenario description (e.g., it does not rely on any future crowd positions). This allows us to compute a descriptor from just the scenario description, while still being inherently anticipatory in nature, accounting for the expected future interactions of the agents through their relative positions and velocities.

5.2 Manifold Learning

We can use the approximately 68,000 MPD descriptors from real pedestrians to directly learn a low-dimensional Crowd Space manifold which captures the structure and variation seen in the crowd datasets. In order to create balanced training data which is representative of the entire variety of training scenarios, we randomly select an equal number of stratified samples from each dataset.

Generically, let $X \in \mathbb{R}^{m \times d}$ be stratified training data obtained after concatenating the m, d-dimensional, descriptors from a subset of agents present in the crowd datasets. We are interested in learning a mapping from X to a new representation $Y \in \mathbb{R}^{m \times d'}$, where typically $d' \ll d$. Here, we use Principal Component Analysis (PCA), to learn a linear projector operator $Y = XP$, where P is an $d \times d'$ matrix whose columns are the eigenvectors of the covariance $X^T X$. By keeping only the first d' principal components, a reduced representation can be obtained while controlling the percentage of variability that is retained.

While PCA provides the optimal linear model for dimensionality reduction, it is important to note that our framework does not depend on this method specifically. Other popular (non-linear) dimensionality reduction techniques include Multidimensional scaling (MDS) [Torgerson 1952], Isomap [Tenenbaum et al. 2000], multilayer denoising autoencoders (SDAE) [Vincent et al. 2010], and Local Linear Embedding (LLE) [Roweis and Saul 2000] (projections using these approaches are shown in Appendix E). We can quantitatively measure these various techniques in terms of how well they maintain the relative distances and neighborhood relationships seen in the original high-dimensional data [Venna et al. 2010].

To estimate how well relative distances were retained in the low dimensional space, we used the six datasets depicted in Figure 1 and extracted 2,500 MPD descriptors per dataset. We then took a random subset of 10,000 distances between pairs of samples in the original 60D space and the same sample pairs in various lower dimensional spaces, and computed the Spearman's rank correlation coefficient between each low-D set and the 60D space. Ideally, these distances should be exactly correlated indicating the projection preserves all of the global structure of the full space. In practice, Table 2a shows that PCA maintains the highest level of correlation across all methods tested and across different number of output dimensions. Corresponding scatterplots, known as Shepard diagrams [Shepard 1980], are shown in Appendix E.

We can focus more on the quality of the local structure by computing the generalization error of each space [van Der Maaten et al. 2009] (as estimated by a 1-NN classifier of the source video dataset for each given MPD descriptor). Table 2b reports these results, where again PCA shows the best performance among the different methods. These findings are consistent with prior work that has suggested that many non-linear techniques do not outperform classical principal component analysis for many real-world tasks [van Der Maaten et al. 2009; Venna et al. 2010].

For any dimensionality reduction technique, the optimal number of reduced dimensions d' depends on the task at hand. For the purposes of visualization, 2D projections ($d' = 2$) are typically the most appropriate. While using too few dimensions fails to capture all of the structure of the data, using too many increases the risk of overfitting and requires more training data to achieve the same amount of generalizability. In fact, for our dataset, the generalization error of the original 60D data is slightly higher than with PCA at 6-10 dimensions, due to the reduced space capturing less of the unimportant variation (i.e., noise). Below, unless otherwise stated, we use a d' of 6, to balance these tradeoffs.

5.3 Crowd Space Analysis

Figure 1 shows a 2D projection of the crowd space obtained with PCA using 2,500 MPD descriptors from six datasets, where each sample is colored according to the dataset from which it was generated. Even this 2D projection captures a semantic meaningful crowd space. For example, the primary direction (x-axis) of the embedding corresponds closely to the density of the scenario, going from regions of dense interactions (left) to sparse interactions (right). The orthogonal axis in this projection is less direct, but is closely related to whether agents in the crowd are heading into or out of congestion (top and bottom, respectively) or if they are in the middle of it (center). In addition, similar agent-based descriptors tend to

Table 2. Analysis of various dimensionality reduction techniques at different numbers of output dimensions. (a) Rank-correlation analysis of how well the reduced space preserves the (high-dimensional) distances between data points. (b) The generalization error as measured by 1-NN video dataset misclassification rate. For both metrics, PCA performs the best across all output dimensions.

(a) Distance Correlations (higher is better)

# Dim.	PCA	SDAE	MDS	Isomap	LLE
2D	**.931**	.824	.896	.925	.478
6D	**.975**	.963	.931	.970	.347
10D	**.985**	.956	.944	.971	.256

(b) Generalization Error (lower is better)

# Dim.	PCA	SDAE	MDS	Isomap	LLE
2D	**.442**	.464	.474	.448	.524
6D	**.152**	.232	.542	.213	.376
10D	**.113**	.200	.545	.196	.326

naturally cluster together independent of the crowd dataset they came from, indicating the ability of our linear projector to learn the high-dimensional structure of the MPD-based input data.

Classification Accuracy. We can quantify how well the derived Crowd Space captures high-level, semantically meaningful information about the structure of the crowd, by training a k-nearest neighbors (k-NN) classifier using this space. Here, we use the label of which dataset a given MPD descriptor came from as training. Because nearby descriptors in a (smooth) low-D space should share consistent labels, a neighbor-based classifier should be able to reliably label novel samples with high accuracy.

Formally, given a scenario descriptor $\mathcal{I} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ (i.e., consisting of n per-agent MPD descriptors), we need to predict its dataset label y which can take one of K different known values. To do so, we need to estimate the probability $p(y_j = 1 \mid P, \mathcal{I})$ for each $j \in [1, K]$. Since the scenario consists of n agents, we use a plurality voting scheme to determine such probability (relying on a maximum likelihood assumption). In particular, we use the learned operator P to project each agent descriptor $\mathbf{x}_i \in \mathcal{I}$ to the PCA crowd space and identify its k-closest training examples. The most represented dataset label by the $k \times n$ nearest neighbors is considered as the $\mathcal{I}$'s predicted dataset label.

Results. Here we focus on the six datasets used to learn the space depicted in Figure 1. To prevent overfitting, we partitioned the scenarios from each dataset into ten consecutive subsets of frames and employed a 10-fold cross validation, where in each fold one of the subsets of every dataset is retained as testing data and the remaining nine subsets are used as training scenarios. This allows us to use all scenarios both for training and testing, with each scenario being used for testing exactly once. The results from the 10 folds can then be averaged to determine the classification accuracy which varies with the number of components used to define the PCA space. With a 2D space shown in Figure 1, we have 63.31% accuracy. Increasing the number of components increases the accuracy until a plateau at six components resulting in 80.45% classification accuracy. Figure 7 shows the corresponding confusion matrix. Misclassified scenarios tend to appear primarily between datasets that share many similar features such as Bottleneck and Bottleneck_long (both dense bottleneck scenarios) or Snow_eth and Students (both have agents who commonly overtake slower pedestrians), further validating our PCA-based dimensionality reduction approach.

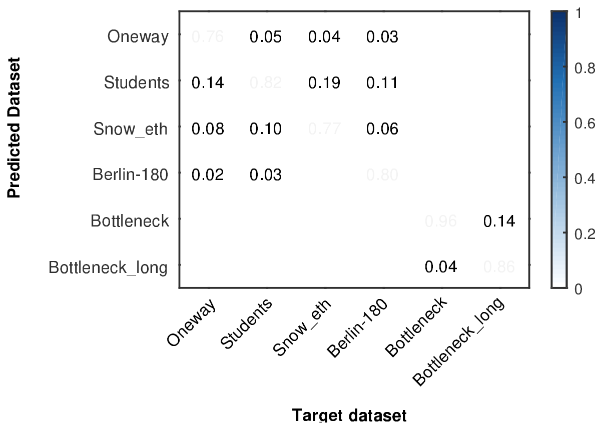

Fig. 7. Percentage of correct and incorrect scenario classification for six crowd datasets obtained using PCA with 6 principal components and our voting scheme with 5 nearest neighbors. Datasets that contain many crowds in similar scenarios tend to be confused (e.g., both Students and Berlin-180 contain bidirectional flows of pedestrians) . All reported results are the averages over 10 folds.

6 SIMULATION METHOD ANALYSIS

We can now combine our localized estimate of simulation accuracy (Section 4) with our Crowd Space (Section 5) to generate a map of the expected accuracy of different simulation methods across different potential crowd scenarios. The result is shown in Figure 8, where each local descriptor sample is colored according to its corresponding accuracy as measured by our modified entropy metric. This type of visualization can provide insight both on the (absolute) accuracy of a given simulation method in different types of scenarios, and as to the relative merit of different simulation methods on different types of scenarios.

Recall that in the 2D projection of the Crowd Space used here (c.f. Figure 1), the horizontal axis corresponds strongly to the local density and the north (or south) portions represent regions were agents are entering into (or exiting from) local congestion. With these correlations in mind, several important trends can be seen in Figure 1. For example, nearly all simulation methods perform better in low density scenarios than high density ones, reflecting the fact that human-like navigation in dense crowds remains a challenging problem for the field. Additionally, while there is a loose ordering of accuracy between the methods (Social Forces generally

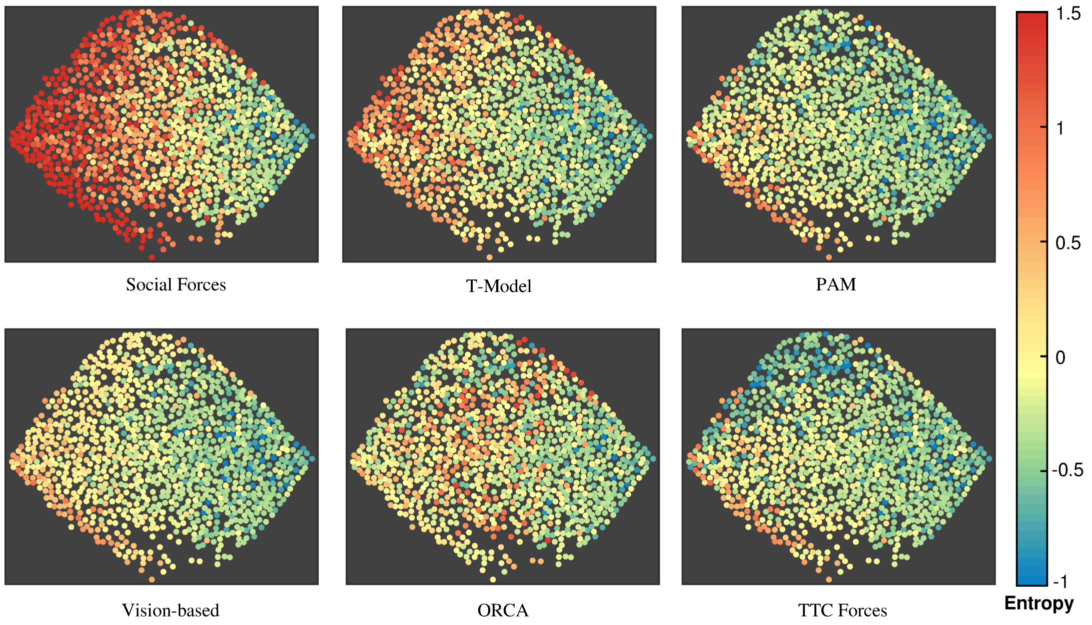

Fig. 8. Mapping of the local simulation accuracy (as computed by our modified entropy metric) for the Crowd Space shown in Figure 1. The lower the entropy score, the better the simulated agent matches the behavior of the real pedestrian.

under-performing and TTC forces generally over-performing), each method has some regions where it performs very strongly.

Some methods, such at PAM and TTC Forces, show a strong up-down asymmetry on the y-axis, especially in dense scenarios, where they have worse performance at the bottom of the Crowd Space (i.e., exiting scenarios). This could suggest that these methods underestimate the effect of neighbors that are behind an agent. This is especially likely in the case of TTC forces where very close agents are ignored if an agent is moving away from them (i.e., time to collision is less than 0).

It is also interesting to see where similar methods differ in performance. ORCA and T-Model were both derived from analyzing the geometry of collision dynamics but show fairly different entropies. The T-Model accounts only for pair-wise interactions whereas ORCA agents more robustly account for the simultaneous set of all neighbors. This may be what contributes to ORCA outperforming T-Model in dense scenarios. However, this comes as a cost as ORCA typically under-performs in medium density crowds (center of the space), and is noticeable worse than the T-Model here.

We can facilitate a direct comparison between these two methods by highlighting for each point in the Crowd Space whether there are more nearby neighboring samples with lower entropy metric scores for one method than another. Figure 9a shows a direct pair-wise comparison for every point in Crowd Space based on if ORCA or T-Model is the predicted winner (if there is a tie, both are considered winners). As can be seen, there are several regions where each method outperforms the other. Figure 9b shows a similar comparison between PAM and TTC Forces (two methods based on anticipatory avoidance forces). Here, PAM and TTC forces have very similar performance in many scenarios, but TTC Forces has consistently higher accuracy in dense scenarios. In the following section, we propose a method to automatically apply this type of predictive analysis in determining the best simulation method to use on new crowd scenarios.

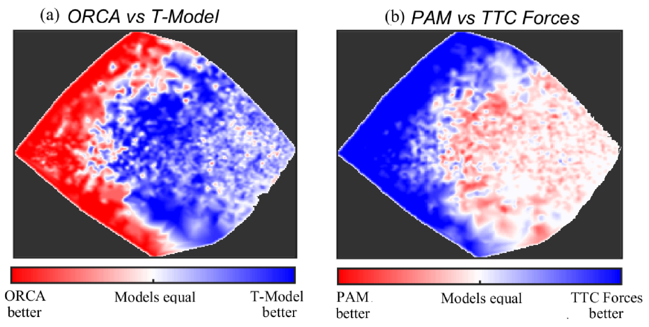

Fig. 9. Pairwise model comparison between (a) ORCA and T-Model and (b) PAM and TTC Forces. Coloring indicates what percentage of the 5 nearest neighboring samples have the lowest predicted simulation error (with a mean filtering applied for smoothing). White regions indicate the two methods performed equally well.

7 CROWD SIMULATION PREDICTION

Here, we consider the task of predicting which simulation method is the most accurate to use in a given scenario. We assume that because there is uncertainty in measuring the accuracy of a simulation method, multiple approaches may be equally well suited for any

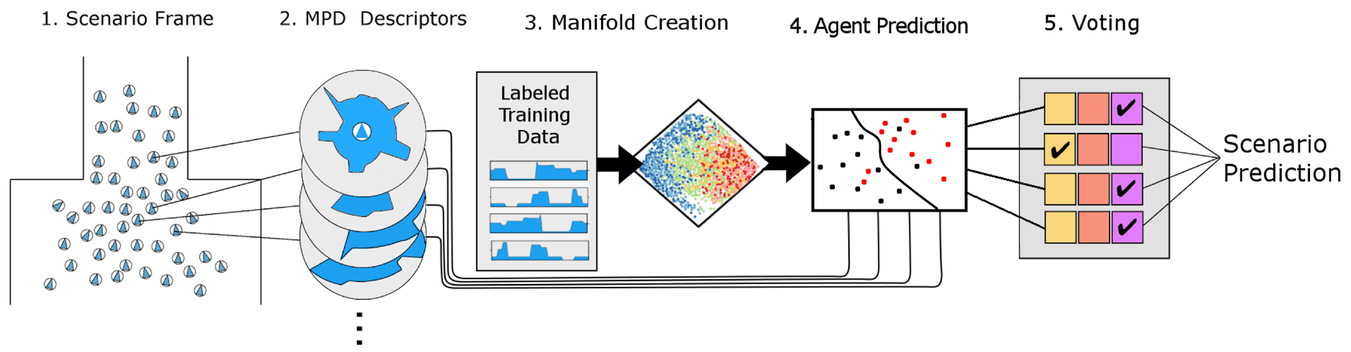

Fig. 10. (a) Overview of our simulation method prediction pipeline. Given a new crowd scenario (1), we decompose it to a collection of per-agent descriptors (2), where each descriptor provides an egocentric representation of the scenario from the eyes of the agent based on the concept of minimal predicted distance. Having learned an embedding from real world crowd datasets, we project each agent descriptor to the embedding (3), and employ a voting scheme (4) to predict the class of the input interaction scenario.

given scenario leading to the potential of multiple best methods for a given scenario.

7.1 Voting-Based Framework

The intuitive, neighbor-based, approach to analyzing the best method for a given scenario presented in the previous section can be formalized as a k-nearest neighbor framework. That is, we look at which samples in our data lie nearby, as measured on the low-dimensional manifold, to the feature descriptors that make up the crowd scenario in question. However, this approach will provide a separate label for each agent in the scenario. To develop a single label for the scenario, each agent votes for each simulation method based on whether or not its percentage of nearby neighbors is above a (learned) threshold. Any simulation method with enough votes above the learned threshold is declared a predicted winner for this crowd scenario. An overview of this approach is given in Figure 10.

More formally, we assume we are a given a scenario descriptor $\mathcal{I} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ consisting of n MPD descriptors, and we want to estimate the performance of K crowd simulation methods. To solve this problem, we associate each agent-based MPD descriptor in our training data with a binary vector $\mathbf{t} \in \{0,1\}^K$, where its j^{th} element denotes whether the method $j \in [1,K]$ is suited for simulating the agent. To compute the vector $\mathbf{t}$ for the agent, we use for each method its corresponding entropy metric (see Figure 8) to determine whether it has the best entropy score, or it is within a small threshold of the best score. Given the scenario $\mathcal{I}$ in question, we project each agent descriptor $\mathbf{x}_i \in \mathcal{I}$ to the crowd space and compute its k-nearest neighbors. For each simulation method j, we can then determine the percentage of $k \times n$ neighbors that can be simulated by this method. If this percentage is above a threshold, we consider the method j suitable for the testing scenario $\mathcal{I}$.

Parameter Tuning. This prediction approach leaves a number of hyperparameters opened for tuning. This includes how many neighbors to consider in the k-NN neighborhood, and what threshold of voting is needed to declare a simulation method as a winner. All of these were fit using a three-way testing-training-validation split as detailed in Appendix F.

7.2 Prediction Accuracy

We evaluated the prediction performance on scenarios from the ten datasets described in Section 3.1. To compute the accuracy of the resulting classifier, we used a 10-fold cross validation scheme. Because a simulation method is very likely to exhibit similar behavior in scenarios extracted from the same dataset, we employed a leave-4-out cross validation scheme, where we randomly select six of the datasets as training data and the remaining four datasets are retained as testing data. In each fold, we randomly select an equal number of MPD descriptors from the training datasets to re-learn the PCA space, and compute the accuracy from the testing datasets.

Accuracy results from a single fold are shown in Figure 11a based on 4 datasets that were not in the training environment. Since this is a multi-label and multi-class classification problem, the standard definition of accuracy is not directly applicable. Instead, we report one minus the hamming loss normalized over all testing examples and classes [Tsoumakas and Katakis 2006]. Here, the Church dataset has the lowest accuracy, likely because several of the participants are walking in intentionally silly or playful manner that is unlike any of the other training data used in the fold's training.

The average accuracy over all 10 folds is 83.56% ± 4.57% (here scenarios where no simulation method is winning for the majority of agents are removed from testing). Importantly, as shown in Figure 11b, our approach had a 87.04% (± 3.53%) success rate in predicting the best simulation method to use for a given scenario (rank-1 accuracy). In almost every case where we mis-predicted the winner, we instead predicted the second best method (i.e., the method with the second highest vote total). Our accuracy at predicting either the best or second best method (rank-2 accuracy) was 97.53% (± 1.71%).

7.3 Deep Neural Networks

The overall, voting-based prediction framework described in Figure 10 can naturally support different methods for manifold learning (step 3) and different agent prediction methods (step 4). While k-NN

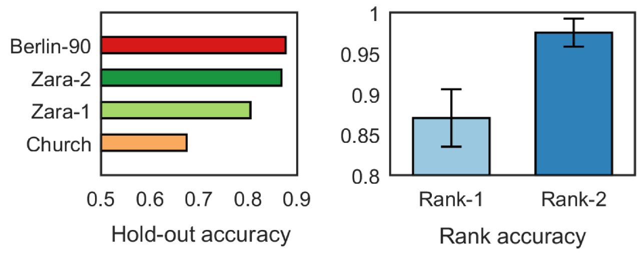

Fig. 11. (left) Method prediction accuracy for the datasets not used in the training of Figure 1. (right) Rank accuracy averaged across 10 folds. Our pipeline predicts the best method 87% of the time and one of the two best methods 97% of the time.

prediction across a PCA-learned manifold gives high prediction accuracy, deep neural networks provide a natural alternative, especially given the large amount of data we have available for training. Instead of PCA, for example, we can employ a deep network based on stacking layers of denoising autoencoders (SDAE), where the input units of each autoencoder are first corrupted by additive Gaussian noise, to learn robust representations of complex data [Vincent et al. 2010]. Instead of k-NN, we can use a fully connected network to predict the class of an agent, based on it's SDAE-encoded features.

To create our deep network, we pretrained a 100-100-100 SDAE network using a zero mean Gaussian noise with 0.1 variance. For each autoencoder we used an affine mapping followed by hyperbolic tangent to encode the visible units to a hidden representation. An affine+tanh decoder is then used to map the hidden representation back to the input space and reconstruct the input data. Each autoencoder was pretrained using conjugate gradient minimization with a squared reconstruction error loss function. As with PCA, the input data $X \in \mathbb{R}^{m \times 60}$ was obtained after concatenating m MPD descriptors from a subset of agents present in the crowd datasets. Using the pretrained SDAE network, we connected K logistic units to the top layer that denote the feasibility of using each method to simulate an input agent, and fine-tuned the whole network by minimizing the cross-entropy error with respect to the winning simulation method(s) for the agent.

Similar to k-NN, we can use each output of this network to determine the probability that a method is likely to have the highest accuracy. If this percentage is above a (learned) threshold, we mark this method as a winner. As with the k-NN-based predictions, we used a 10-fold cross validation scheme where we stratified our folds by dataset using a 6-4 pattern (i.e., train on 6 datasets, test on the 4 holdout, repeat 10 times), to compute the prediction accuracy. Overall, the two methods performed similarly with deep neural network showing a slight improvement over k-NN both in the rank-1 accuracy (88.59% vs 87.04%) and rank-2 accuracy (98.46% vs 97.53%). However, in some situations this small increase in accuracy may be worth the additional training time.

7.4 Applications

Because the overall prediction framework we propose requires no information besides the initial crowd state, it can be used in part of an animation assistance tool to provide automatic recommendations of the best simulation method to use in a given scenario. Importantly, this framework can be applied to entirely new configurations never seen in training, needing only the environment and initial and goal conditions of the crowd. As a result, we can evaluate the likely accuracy of a simulation method *without needing to run the simulation* so long as the local conditions do not significantly change over the lifetime of the agents (which is true for all our datasets); if the agents enter substantially new conditions, though, it would be necessary to gather new MPD descriptors and predictions.

In the supplementary video, we show results for two user-created scenarios: a bi-directional hallway scenario, and a circle scenario where agents are tasked to walk to antipodal points in circle, a task unlike any found in our training data. Here, our framework predicts the Vision model will outperform ORCA, and the subsequent simulations show the motion from ORCA is indeed more halting and less smooth. For truly user-drawn scenarios, there is no known right motion, so we cannot compute entropy numbers to compute our prediction accuracy. However, the supplementary video shows results from several of the hold-out scenarios from our folds, highlighting the predictive power of our framework in data that it has not seen in training.

8 DISCUSSION AND CONCLUSION

We introduced a new method for analyzing crowd simulation accuracy based on a novel low dimensional crowd space. We have shown both how this space can provide interesting insights into the relative merit of different simulation techniques, and how our proposed per-agent, voting-based classifier can be used to automate predictions of the best method to use in a given scenario with high accuracy. To this end, we also introduced a modification of the entropy metric to allow localized (per-agent, per-time step) estimations of simulation accuracy. This modified entropy metric can also be used as an analysis tools on its own by highlighting aspects of real-world crowd datasets that different simulation methods fail to capture.

It is interesting to consider how our method may be applied to predict perceptual quality (e.g., as estimated from a user study) rather than simulation accuracy (as computed by our Entropy metric). In theory, it is possible to simply replace the entropy labeling with a “looked best" manual labeling. However, visually inspecting simulations is a tedious task, especially given the large datasets needed to get high-quality results. Additionally, while visual simulation quality is often correlated with prediction accuracy, there are other important aspects such as the rendering, animation, camera placement, etc., which can have a large impact. Experts and lay-people may also differ on which simulations look best, and small errors in accuracy that are difficult to notice at first may be magnified after seeing them happen repeatedly.

Limitations. There are some important limitations of our approach that are worth emphasizing. Most notable, being a data-driven approach, the quality of our results is limited to the scope and size of datasets used in training. Relatedly, individual pedestrians are rarely tracked for more than several seconds, preventing us from training on long term changes in simulation accuracy. Likewise, the difficulty of inferring a person's true desired position and velocity from noisy trajectories can create error in the crowd scenarios we extract from the tracked pedestrian datasets. In terms of the learning approach, our descriptor of a crowd scenario is broken into an unstructured collection of per-agent, MPD-based descriptors. This representation can lose some of the long-range structure inherent in certain scenarios.

Additionally, our proposed voting-based framework treats each agent equally, which can dampen the effect of extreme outliers (e.g., the negative effect of a small number of very poorly simulated agents can be underestimated). Overall, the high accuracy of our approach suggest these issues had only a limited effect on the scenarios contained in our datasets, but there are of course many other important simulation scenarios than those we were able to test. Another limitation stems from what is being measured as accuracy through the entropy metric. The statistical approach used here can be biased in favor of methods that predict the correct average behavior but fail to capture some of the interesting (though unusual) deviations from average.

Future work. Looking forward, we see many possibilities for expanding the scope of this work and increasing its potential impact. In addition to using the Crowd Space to predict simulation accuracy (as measured by the modified entropy metric), we can look at other important measures of quality such as path smoothness, energy minimization, or number of collisions. A similar manifold-based approach might also be applicable to other domains where multiple competing simulation methods are popular such as character animations or modeling the motion of plants and animals. Though, to be useful, there would need to exist large datasets of reference real world motions.

We also hope to address some of the above limitations in future work. In particular, ideas from object recognition which use multi-scale, image-based (i.e., sampling-based) feature descriptors to represent objects at different scales may also be applied to explicitly capture the structure in crowd simulations. We would also like to incorporate additional simulation methods to our framework, such as the recent work in [Wolinski et al. 2016] that provides a probabilistic collision prediction scheme for the agents. We note that in our current work we tested the simulation accuracy of six methods using their default simulation parameters (as provided in the corresponding papers and/or software libraries). However, besides different methods, our approach can also support the testing of the same method but simulated with different sets of parameters, a mixture of the two, etc. As such, we are excited about the prospect of building a tool which can provide visual analysis and automated insights as to the best simulation models to use in various scenarios.

Lastly, we are interested in exploring the potential of using our approach to directly drive a simulation method. This could be in the form of an AI which chooses a new simulation method for each frame (or each agent) based on the best option for the current conditions.

ACKNOWLEDGMENTS

We thank Jan Ondřej for sharing his *T-model* code, and Rahul Narain for useful discussions. This work was supported in part by the National Science Foundation under grants IIS-1748541, CHS-1526693, and CNS-1544887.

REFERENCES

Tunç Ozan Aydin, Martin Čadík, Karol Myszkowski, and Hans-Peter Seidel. 2010. Video Quality Assessment for Computer Graphics Applications. *ACM Transactions on Graphics* 29, 6, Article 161 (2010), 12 pages.

Glen Berseth, Mubbasir Kapadia, Brandon Haworth, and Petros Faloutsos. 2014. SteerFit: Automated Parameter Fitting for Steering Algorithms. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 113–122.

Robert Bridson. 2015. *Fluid simulation for computer graphics*. CRC Press.

Panayiotis Charalambous and Yiorgos Chrysanthou. 2014. The PAG Crowd: A Graph Based Approach for Efficient Data-Driven Crowd Simulation. *Computer Graphics Forum* 33, 8 (2014), 95–108.

Panayiotis Charalambous, Ioannis Karamouzas, Stephen J. Guy, and Yiorgos Chrysanthou. 2014. A Data-Driven Framework for Visual Crowd Analysis. *Computer Graphics Forum* 33, 7 (2014), 41–50.

Funda Durupinar, Mubbasir Kapadia, Susan Deutsch, Michael Neff, and Norman I Badler. 2017. Perform: Perceptual approach for adding ocean personality to human motion using laban movement analysis. *ACM Transactions on Graphics* 36, 1 (2017), 6.

Teófilo Dutra, Ricardo Marques, Joaquim B. Cavalcante-Neto, Creto Augusto Vidal, and Julien Pettré. 2017. Gradient-based steering for vision-based crowd simulation algorithms. *Computer Graphics Forum* 36, 2 (2017).

Geir Evensen. 2003. The ensemble Kalman filter: Theoretical formulation and practical implementation. *Ocean dynamics* 53, 4 (2003), 343–367.

Stephen J. Guy and Ioannis Karamouzas. 2015. A Guide to Anticipatory Collision Avoidance. In *Game AI Pro 2: Collected Wisdom of Game AI Professionals*, Steve Rabin (Ed.). A K Peters/CRC Press, Chapter 19, 195–208.

Stephen J Guy, Jur Van Den Berg, Wenxi Liu, Rynson Lau, Ming C Lin, and Dinesh Manocha. 2012. A statistical similarity measure for aggregate crowd dynamics. *ACM Transactions on Graphics (TOG)* 31, 6 (2012), 190.

Dirk Helbing, Illés Farkas, and Tamas Vicsek. 2000. Simulating dynamical features of escape panic. *Nature* 407, 6803 (2000), 487–490.

Dirk Helbing and Péter Molnár. 1995. Social Force Model for Pedestrian Dynamics. *Physical Review E* 51 (1995), 4282–4286.

Daniel Holden, Taku Komura, and Jun Saito. 2017. Phase-functioned neural networks for character control. *ACM Transactions on Graphics* 36, 4 (2017), 42.

Ludovic Hoyet, Kenneth Ryall, Katja Zibrek, Hwangpil Park, Jehee Lee, Jessica Hodgins, and Carol O'sullivan. 2013. Evaluating the distinctiveness and attractiveness of human motions on realistic virtual bodies. *ACM Transactions on Graphics* 32, 6 (2013), 204.

Rowan Hughes, Jan Ondřej, and John Dingliana. 2015. DAVIS: density-adaptive synthetic-vision based steering for virtual crowds. In *Motion in Games*. ACM, 79–84.

Eunjung Ju, Myung Geol Choi, Minji Park, Jehee Lee, Kang Hoon Lee, and Shigeo Takahashi. 2010. Morphable crowds. *ACM Transactions on Graphics* 29 (2010), 140:1–140:10. Issue 6.

Mubbasir Kapadia, Nuria Pelechano, Jan Allbeck, and Norm Badler. 2015. Virtual crowds: Steps toward behavioral realism. *Synthesis Lectures on Visual Computing: Computer Graphics, Animation, Computational Photography, and Imaging* 7, 4 (2015), 1–270.

Mubbasir Kapadia, Shawn Singh, Brian Allen, Glenn Reinman, and Petros Faloutsos. 2009. Steerbug: an interactive framework for specifying and detecting steering behaviors. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 209–216.

Mubbasir Kapadia, Shawn Singh, William Hewlett, Glenn Reinman, and Petros Faloutsos. 2012. Parallelized egocentric fields for autonomous navigation. *The Visual Computer* 28, 12 (2012), 1209–1227.

Mubbasir Kapadia, Matt Wang, Shawn Singh, Glenn Reinman, and Petros Faloutsos. 2011. Scenario space: characterizing coverage, quality, and failure of steering algorithms. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 53–62.

Ioannis Karamouzas, Peter Heil, Pascal van Beek, and Mark H. Overmars. 2009. A Predictive Collision Avoidance Model for Pedestrian Simulation. In *Motion in Games*. 41–52.

Ioannis Karamouzas, Brian Skinner, and Stephen J. Guy. 2014. Universal Power Law Governing Pedestrian Interactions. *Physical Review Letters* 113 (2014), 238701. Issue 23.

Ioannis Karamouzas, Nick Sohre, Rahul Narain, and Stephen J. Guy. 2017. Implicit Crowds: Optimization Integrator for Robust Crowd Simulation. *ACM Transactions on Graphics* 36, 4 (2017).

Richard Kulpa, Anne-Hélène Olivierxs, Jan Ondřej, and Julien Pettré. 2011. Imperceptible relaxation of collision avoidance constraints in virtual crowds. *ACM Transactions on Graphics* 30, 6 (2011), 138.

Gregor Lämmel and Matthias Plaue. 2014. Getting out of the way: Collision-avoiding pedestrian models compared to the realworld. In *Pedestrian and Evacuation Dynamics 2012*. Springer, 1275–1289.

Jehee Lee. 2010. Introduction to Data-driven Animation: Programming with Motion Capture. In *ACM SIGGRAPH ASIA 2010 Courses*. Article 4, 50 pages. https://doi.org/10.1145/1900520.1900524

Alon Lerner, Yiorgos Chrysanthou, and Dani Lischinski. 2007. Crowds by example. *Computer Graphics Forum* 26 (2007), 655–664.

Alon Lerner, Yiorgos Chrysanthou, Ariel Shamir, and Daniel Cohen-Or. 2010. Context-Dependent Crowd Evaluation. *Computer Graphics Forum* 29, 7 (2010), 2197–2206.

Pinxin Long, Wenxi Liu, and Jia Pan. 2017. Deep-Learned Collision Avoidance Policy for Distributed Multiagent Navigation. *IEEE Robotics and Automation Letters* 2, 2 (2017), 656–663.

Frank J Massey Jr. 1951. The Kolmogorov-Smirnov test for goodness of fit. *Journal of the American statistical Association* 46, 253 (1951), 68–78.

Rachel McDonnell, Michéal Larkin, Benjamín Hernández, Isaac Rudomin, and Carol O'Sullivan. 2009. Eye-catching Crowds: Saliency Based Selective Variation. *ACM Transactions on Graphics* 28, 3, Article 55 (2009), 55:1–55:10 pages.

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

Rahul Narain, Abhinav Golas, Sean Curtis, and Ming C. Lin. 2009. Aggregate Dynamics for Dense Crowd Simulation. *ACM Transaction on Graphics* 28, 5 (2009), 122:1–122:8.

Anne-Hélène Olivier, Antoine Marin, Armel Crétual, and Julien Pettré. 2012. Minimal predicted distance: A common metric for collision avoidance during pairwise interactions between walkers. *Gait Posture* 36, 3 (2012), 399–404.

Jan Ondřej, Julien Pettré, Anne-Hélène Olivier, and Stéphane Donikian. 2010. A synthetic-vision based steering approach for crowd simulation. *ACM Transactions on Graphics* 29, 4 (2010), 1–9.

Nuria Pelechano, Jan M Allbeck, Mubbasir Kapadia, and Norman I Badler. 2016. *Simulating Heterogeneous Crowd with Interactive Behaviors.* CRC Press.

Stefano Pellegrini, Andrea Ess, Konrad Schindler, and Luc Van Gool. 2009. You'll never walk alone: Modeling social behavior for multi-target tracking. In *IEEE International Conference on Computer Vision.* 261–268.

Xue Bin Peng, Glen Berseth, KangKang Yin, and Michiel Van De Panne. 2017. Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning. *ACM Transactions on Graphics* 36, 4 (2017), 41.

Julien Pettré, Jan Ondřej, Anne-Hélène Olivier, Armel Crétual, and Stéphane Donikian. 2009. Experiment-based Modeling, Simulation and Validation of Interactions between Virtual Walkers. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation.* 189–198.

Paul S.A. Reitsma and Nancy S. Pollard. 2007. Evaluating motion graphs for character animation. *ACM Transactions on Graphics* 26, 4 (2007), 18.

Zhiguo Ren, Panayiotis Charalambous, Julien Bruneau, Qunsheng Peng, and Julien Pettré. 2017. Group Modeling: A Unified Velocity-Based Approach. In *Computer Graphics Forum*, Vol. 36. 45–56.

Craig W. Reynolds. 1987. Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics* 21, 4 (1987), 24–34.

Craig W. Reynolds. 1999. Steering Behaviors For Autonomous Characters. In *Game Developers Conference.* 763–782.

Sam T Roweis and Lawrence K Saul. 2000. Nonlinear dimensionality reduction by locally linear embedding. *science* 290, 5500 (2000), 2323–2326.

Armin Seyfried, Oliver Passon, Bernhard Steffen, Maik Boltes, Tobias Rupprecht, and Wolfram Klingsch. 2009. New insights into pedestrian flow through bottlenecks. *Transportation Science* 43, 3 (2009), 395–406.

Roger N Shepard. 1980. Multidimensional scaling, tree-fitting, and clustering. *Science* 210, 4468 (1980), 390–398.

Eftychios Sifakis and Jernej Barbic. 2012. FEM Simulation of 3D Deformable Solids: A Practitioner's Guide to Theory, Discretization and Model Reduction. In *ACM SIGGRAPH 2012 Courses.* Article 20, 50 pages.

Shawn Singh, Mubbasir Kapadia, Petros Faloutsos, and Glenn Reinman. 2009. SteerBench: a benchmark suite for evaluating steering behaviors. *Computer Animation and Virtual Worlds* 20, 5-6 (2009), 533–548.

Shawn Singh, Mubbasir Kapadia, Glenn Reinman, and Petros Faloutsos. 2011. Footstep navigation for dynamic crowds. *Computer Animation and Virtual Worlds* 22, 2-3 (2011), 151–158.

Sybren Stuvel, Nadia Magnenat-Thalmann, Daniel Thalmann, A Frank van der Stappen, and Arjan Egges. 2016. Torso Crowds. *IEEE Transactions on Visualization and Computer Graphics* (2016).

Joshua B Tenenbaum, Vin De Silva, and John C Langford. 2000. A global geometric framework for nonlinear dimensionality reduction. *science* 290, 5500 (2000), 2319–2323.

Warren S Torgerson. 1952. Multidimensional scaling: I. Theory and method. *Psychometrika* 17, 4 (1952), 401–419.

Adrien Treuille, Seth Cooper, and Zoran Popović. 2006. Continuum crowds. *ACM Transactions on Graphics* 25, 3 (2006), 1160–1168.

Grigorios Tsoumakas and Ioannis Katakis. 2006. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining* 3, 3 (2006).

Kiwon Um, Xiangyu Hu, and Nils Thuerey. 2017. Perceptual Evaluation of Liquid Simulation Methods. *ACM Transactions on Graphics* 36, 4, Article 143 (2017), 143:1–143:12 pages. https://doi.org/10.1145/3072959.3073633

Jur van den Berg, Stephen J. Guy, Ming C. Lin, and Dinesh Manocha. 2011. Reciprocal n-body Collision Avoidance. In *International Symposium of Robotics Research.* 3–19.

Jur van den Berg, Ming C. Lin, and Dinesh Manocha. 2008. Reciprocal Velocity Obstacles for real-time multi-agent navigation. In *IEEE International Conference on Robotics and Automation.* 1928–1935.

Laurens van Der Maaten, Eric Postma, and Jaap Van den Herik. 2009. Dimensionality reduction: a comparative review. *Journal of Machine Learning Research* 10 (2009), 66–71.

Jarkko Venna, Jaakko Peltonen, Kristian Nybo, Helena Aidos, and Samuel Kaski. 2010. Information retrieval perspective to nonlinear dimensionality reduction for data visualization. *Journal of Machine Learning Research* 11 (2010), 451–490.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research* 11, Dec (2010), 3371–3408.

He Wang, Jan Ondrej, and Carol O'Sullivan. 2017. Trending Paths: A New Semantic-Level Metric for Comparing Simulated and Real Crowd Data. *IEEE Transactions on Visualization & Computer Graphics* 23, 5 (2017), 1454–1464.

David Wolinski, Stephen Guy, Anne-Hélène Olivier, Ming Lin, Dinesh Manocha, and Julien Pettré. 2014. Parameter Estimation and Comparative Evaluation of Crowd Simulations. *Computer Graphics Forum* 33, 2 (2014), 303–312.

David Wolinski, Ming C. Lin, and Julien Pettré. 2016. WarpDriver: context-aware probabilistic motion prediction for crowd simulation. *ACM Transactions on Graphics* 35, 6 (2016), 164.