A Statistical Similarity Measure for Aggregate Crowd Dynamics

Stephen J. Guy* UNC-Chapel Hill
Jur van den Berg University of Utah
Wenxi Liu, Rynson Lau City University of Hong Kong
Ming C. Lin, Dinesh Manocha UNC-Chapel Hill

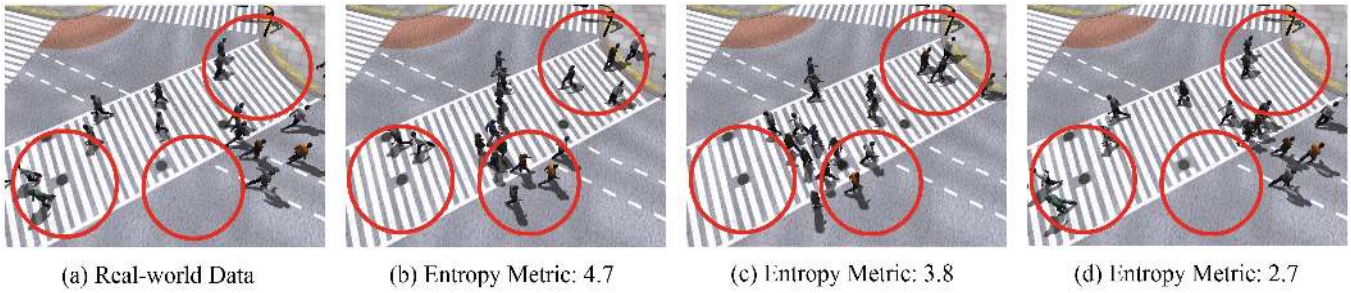


Figure 1: *A comparison between a rendering of real-world crowd data (a), and stills from three different simulation algorithms applied to the same scenario (b-d). Our entropy metric is used to measure the similarity of simulation algorithm to real-world data. A small value of the metric, as in (d), indicates a better match to the data. Differences between the simulations are highlighted with circles.*

Abstract

We present an information-theoretic method to measure the similarity between a given set of observed, real-world data and visual simulation technique for aggregate crowd motions of a complex system consisting of many individual agents. This metric uses a two-step process to quantify a simulator's ability to reproduce the collective behaviors of the whole system, as observed in the recorded real-world data. First, Bayesian inference is used to estimate the simulation states which best correspond to the observed data, then a maximum likelihood estimator is used to approximate the prediction errors. This process is iterated using the EM-algorithm to produce a robust, statistical estimate of the magnitude of the prediction error as measured by its entropy (smaller is better). This metric serves as a simulator-to-data similarity measurement. We evaluated the metric in terms of robustness to sensor noise, consistency across different datasets and simulation methods, and correlation to perceptual metrics.

1 Introduction

Visual simulation of aggregates systems, including human crowds, animal herds, and insect swarms is a growing area of interest in computer graphics, with applications in diverse areas such as social sciences, swarm intelligence, and city planning. For applications in entertainment, providing artists and animators with high-level control while maintaining visual plausibility of motion is often sufficient. However, for many other training and planning applications, such as virtual reality based training, fire-safety planning, and crowd control and management, it is often critical to model accurate motion, in addition to producing a compelling visual rendering. In this context, we define a measure of a simulator's accuracy based on the similarity of the motion from the simulator to the motion captured in real-world observations. While some previous work has studied the visual plausibility of simulation techniques, we present a new metric for quantifying the similarity between a set of real-world observations and any algorithm designed to simulate the aggregate crowd dynamics captured in the data.

*Email: {sjguy,lin,dm}@cs.unc.edu. The first author is currently an Assistant Professor at the University of Minnesota. This work was done while the second and the third authors were at the University of North Carolina (UNC-Chapel Hill).

Evaluating the correctness or predictability of the results from a crowd simulation method presents several interesting challenges, many of which arise from the inherent nature of a crowd as a *complex system*. Complex systems are systems composed of several components or elements that interact to exhibit emergent patterns that cannot be easily predicted from the properties of the individual components alone [Schadschneider et al. 2011; Gallagher and Appenzeller 1999]. Because of issues inherent in these systems, such as uncertainty and non-determinism, the study of complex systems generally must draw on techniques from the fields of statistics, information theory, and non-linear dynamics. We likewise draw on inspiration from these fields, in proposing a new method to compare aggregate simulation methods with real-world data that accounts for these challenges.

Real-world data of crowds is becoming increasingly common, driven in part by recent improvement in sensor technology, such as LiDAR and GPS; the proliferation of high-resolution cameras; and advances in computer vision and motion tracking. However, several aspects of a crowd and its aggregate motions make it difficult to directly compare such data against any simulation results. For example, given two very similar initial states, a small crowd can reach two very different configurations after just a few seconds, because the effects of small changes in states can quickly compound into large differences in the resulting crowd behaviors and motion patterns. This problem is exacerbated by the fact that any data on the motion of aggregate phenomena always comes with noise and uncertainty, making it impossible to know the true state of a crowd with complete accuracy. Additionally, even when presented with the same situation, different individuals tend to make different decisions. Furthermore, each individual can make varying decisions under different emotional states (e.g., happy vs. sad) and other subtle factors. Because of the combined effect of all these uncertainties, it is necessary to treat any real-world data on crowd movements as a noisy sample of possible motions rather than an absolute ground truth, and perform a statistical analysis on the motions and behaviors represented by the observed, example motion of the crowd.

Main result: We introduce the *Entropy Metric* to evaluate the predictability of crowd simulation techniques in terms of similarity to real-world crowd data. Our metric is defined broadly and can be applied to any time-series simulation of aggregate motions in continuous space. In this paper, we focus our discussion on and

illustrate results for data of human crowds.

The Entropy Metric is an *ensemble* measurement of the prediction errors of a given simulation technique relative to a given example set of crowd motions. At a high level, it works based on a two-stage process. First, we estimate a distribution of simulation states which best represents the observed data. Second, the simulator being evaluated is used to predict each subsequent state from the proceeding one. The smaller this prediction error, the better the simulator's ability to reproduce the motion of real-world crowd system represented by the example data. Because these two steps can depend on each other, we use the Expectation Maximization algorithm (EM-algorithm) to interleave these two steps and iterate until convergence.

By only computing prediction error across small simulation timesteps and by maintaining a distribution of likely simulation states, our formulation fundamentally accounts for noise in the measured data, as well as non-determinism in motion, and unmodeled effects of the given crowd simulation method. Moreover, we show that the Entropy Metric is *rankable, predictable, discriminative*, and *robust with respect to sensor noise*. Furthermore, we demonstrate a correlation between the values of the Entropy Metric and perceived motion similarity (as measured by a perceptual study). The Entropy Metric can be used to automatically select a set of appropriate simulation parameters for data-driven crowd simulation to achieve the desired motion and behavior patterns.

The rest of this paper is organized as follows: Section 2 gives a broad characterization of crowd simulation algorithms and introduces our notation. Section 3 describes the theoretical basis of the Entropy Metric and presents an efficient algorithm to compute it. Section 4 demonstrates the application of the metric to several example crowd simulation algorithms by evaluating them on different sets of real-world data. Section 5 analyzes properties of the metric such as its robustness to noise and correlation to user perceptions.

2 Background and Notation

In this section, we give a brief review of algorithms for crowds simulation and data-capture. We also introduce the notation used in the rest of the paper.

Notation We use the following notational conventions throughout this paper: Variables a printed in italics denote scalars or functions, variables $\mathbf{a}$ printed in boldface denote vectors, and variables $\mathbb{A}$ printed in blackboard bold denote vector spaces. Variables A printed in capitals denote (covariance) matrices, and variables $\mathcal{A}$ printed in calligraphic typeface denote probability distributions.

2.1 Simulation State

In the context of this paper, we use the term "crowd" to refer to an aggregate of entities (e.g. people or agents) whose behaviors and dynamics evolve over time. We define a crowd simulation state as follows: for a given simulator, and a given point in time k, the state $\mathbf{x}_k$ of a crowd contains all information about a crowd that is needed to compute its evolution over time. We denote the space of all crowd states by $\mathbb{X}$. For instance, for a crowd consisting of n agents, being simulated by a technique that is based on the position, velocity, and orientation of each agent on a 2D plane, the crowd state space is $\mathbf{x}_k \in \mathbb{X} = \mathbb{R}^{5n}$. Other time-varying aspects, such as the mental state of the agents or dynamic behavior parameters may also be part of the state. We make no specific assumptions about the representation of a crowd state. In addition to the crowd state, simulators may also use constant information, such as constant parameters shared across all agents, and obstacles that define the environment.

2.2 Crowd Simulation and Aggregate Dynamics

At a broad level, the field of crowd simulation can cover many facets of generating human motions and behaviors (such as full-body biomechanics, facial expressions and gestures, and motion dynamics based on the laws of physics). This makes modeling and analyzing all aspects of a crowd highly challenging and can quickly lead to a combinatorial explosion of potential variations. To increase the tractability of the problem, we focus primarily on the aspect of crowd simulation that corresponds to motion data related to the aggregate dynamics of the crowd.

Aggregate Dynamics: Crowds typically are in constant motion, with individual paths changing over time. Formally, we say that if the state of the crowd at time k is $\mathbf{x}_k \in \mathbb{X}$, the crowd has evolved into a state $\mathbf{x}_{k+1} \in \mathbb{X}$ one unit of time later. The rules defining crowd dynamics (that is, the actual rules governing the human motion and behavior) are unknown and likely cannot be defined by simple mathematical models nor derived from first principles. However, we can characterize these unknown dynamics, using an abstract function $f : \mathbb{X} \to \mathbb{X}$, such that:

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k). \tag{1}$$

It should be noted that f is abstract and unknown, and we only use such a formulation to describe crowd evolution. We view a simulator *as an approximation to this function* f; the more accurate and predictive the simulator the better the approximation.

There has been extensive work on computing the pedestrian dynamics or aggregate movement of human-like agents as part of a crowd for more than three decades. These include force-based methods [Helbing and Molnar 1995; Pelechano et al. 2007; Karamouzas et al. 2009], boids and steering models [Reynolds 1987; Reynolds 1999], techniques based on velocity obstacles and geometric optimization [van den Berg et al. 2009; Guy et al. 2010], field-based methods [Patil et al. 2011; Pettre et al. 2009; Ondrej et al. 2010; Sung et al. 2004], continuum models [Treuille et al. 2006; Narain et al. 2009], cognitive models and decision networks [Funge et al. 1999; Yu and Terzopoulos 2007], and example-driven crowd simulation [Lee et al. 2007; Lerner et al. 2007; Pettre et al. 2009]. These methods model different aspects of crowds, including collision avoidance between agents, emergent phenomena, path navigation, high-level cognition and behaviors. All of these methods share a common formulation though, of computing continuous trajectories for each agent to determine the collective dynamics of the motion in the crowd. This commonality leads to the following abstraction of a crowd simulator:

We formulate a simulator as a function $\hat{f} : \mathbb{X} \to \mathbb{X}$ that attempts to approximate the function f:

$$\hat{f}(\mathbf{x}_k) \approx f(\mathbf{x}_k). \tag{2}$$

That is, simulator $\hat{f}$ takes in a state $\mathbf{x}_k$ of the crowd at time k and produces an estimate of the state $\mathbf{x}_{k+1}$ of the crowd at one unit of time later (henceforth referred to as a timestep). We assume that the underlying crowd simulator works in a continuous space over time and our approach may not be applicable to approaches in discretized space (e.g., techniques based on cellular automata). In Section 4.1, we describe the detailed representation of function $\hat{f}$ for some of the commonly used methods.

2.3 Real-World Crowd Data

Empirical datasets of human crowd motion from videos, LiDAR, and GPS sensors are becoming increasingly available, aided by re-

search in computer vision, robotics and pedestrian dynamics on extracting crowd trajectories from sensors and cameras [Seyfried et al. 2010; Lee et al. 2007; Rodriguez et al. 2009; Kratz and Nishino 2011; Pettre et al. 2009]. A recent trend in research has been the combination of crowd tracking algorithms with crowd dynamics models to extract more accurate trajectories or detect abnormal crowd behaviors [Pellegrini et al. 2009; Mehran et al. 2009].

Most of these tracking algorithms represent the position data or the trajectory as time-stamped vectors $\mathbf{z}_k, \mathbf{z}_{k+1}, \ldots$ that provide a partial (and potentially noisy) projection of the true crowd state $\mathbf{x}_k, \mathbf{x}_{k+1}, \ldots$ at the corresponding moment in time. We assume that the relation between the crowd state $\mathbf{x}_k$ and the data $\mathbf{z}_k$ available of the crowd at time k is given by a known function h:

$$\mathbf{z}_k = h(\mathbf{x}_k) + \mathbf{q}_k, \qquad \mathbf{q}_k \sim \mathcal{Q}, \tag{3}$$

where $\mathbf{q}_k$ represents the noise or uncertainty in the real world data, drawn from a constant distribution $\mathcal{Q}$. We assume in this paper that $\mathcal{Q}$ (the sensor uncertainty) is known.

Because our metric measures a simulator's predictability with respect to the set of observed examples, it is important to choose representative data. As our method does not compare a simulator's output to a given set of trajectory data, but rather to the decisions and behaviors captured by the data, it is important to use observed examples which are representative of the behaviors of the real-world crowd. In Sections 4 and 5, we highlight the similarity results on a variety of crowd datasets.

2.4 Validating Crowd Simulators

Crowd simulators have previously been evaluated in terms of perceptual fidelity and other metrics. For example, the work of [Pelechano et al. 2008] evaluates crowd simulations based on quantifying presence in virtual environments. Similarly, [Ennis et al. 2011; Jarabo et al. 2012] measure the perceptual effects of factors such as illumination, camera position and orientation on the perceived fidelity of movement in crowds. In a similar spirit, we also perform a pilot study to measure the correlation between our numerical similarity metric and the perceptual similarity of crowd motion to the validation data.

Other approaches, such as [Singh et al. 2009; Kapadia et al. 2011], present a set of evaluation metrics directly based on the paths generated by a simulator, including path smoothness, number of collisions, or total path length. These metrics are designed to compare the results of different simulations in synthetic environments, but they are not applicable to the evaluation of the similarity between a given simulator and real-world crowd data. Because they provide a different type of motion analysis, these methods should be viewed as complementary to our similarity metric.

2.4.1 Data-driven Crowd Evaluation

Many researchers have proposed methods to measure how closely a simulator matches experimental data. For example, [Pettre et al. 2009] creates a simulation with the same initial conditions as the data and measures the error as a function of the deviation from the recorded trajectories. This approach works well in practice for small numbers of agents, but may not scale to medium or large scenes because of the accumulated, chaotic effect of errors over time.

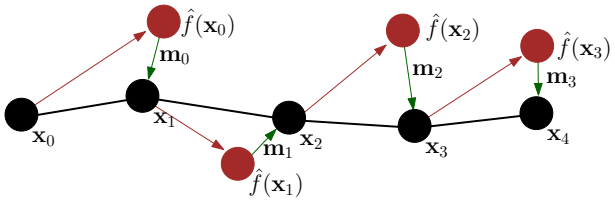
Other approaches, such as the density measure of [Lerner et al. 2009] and fundamental diagram based comparisons such as in [Seyfried et al. 2010], suggest comparing measures based on crowd densities in the output of a simulator with the observed densities in the experimental data. While density-based metrics are applicable to many simulations, densities are not well defined for sparse scenarios, and metrics based on density will be sensitive to noise for these sparse scenarios and those with a small numbers of agents.

In contrast to previous approaches, our metric is applicable to data with both small and large numbers of agents, and to sparse and dense scenarios. Additionally, because our method is based on a robust, statistical interpretation of the validation data as samples of crowd behavior, the entropy metric can account for sensor noise, handle differences in states between simulators, and account for uncertainty in human motion.

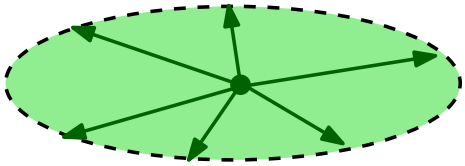
3 Entropy Metric

In this section, we present our Entropy Metric and an efficient algorithm to compute the metric. Fundamentally, we seek to measure the size of the prediction error for a given simulator. That is, given the state of a real crowd $\mathbf{x}_k$, how close does any given simulator $\hat{f}$ comes to predicting the subsequent crowd state $\mathbf{x}_{k+1}$ one timestep later. We denote the error in prediction of the state as the vector $\mathbf{m}_k$ (see Figure 2a). We refer to the distribution of all these error vectors across the entire validation dataset as $\mathcal{M}$ (see Figure 2b). To summarize:

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k) = \hat{f}(\mathbf{x}_k) + \mathbf{m}_k, \qquad \mathbf{m}_k \sim \mathcal{M}. \tag{4}$$



(a) *Computing the prediction error each timestep*



(b) *Distribution of all errors over all timesteps*

Figure 2: *(a) For each timestep, there is some error* $\mathbf{m}$ *(green arrow) between the predicted crowd state from the simulator* $\hat{f}(\mathbf{x}_k)$ *(red dots) and the actual crowd state* $\mathbf{x}_{k+1}$ *(black dots). (b) The collection off all errors (green arrows) over all timesteps is denoted as* $\mathcal{M}$ *(green ellipse). The size of this error distribution, as measured by its entropy, forms our metric (smaller is better).*

The intuition behind our metric lies in the fact that the distribution $\mathcal{M}$ depends on the underlying simulator $\hat{f}$ and encompasses all error and unmodeled effects in the simulator $\hat{f}$, along with potential non-determinism in the function f. A larger value of this distribution $\mathcal{M}$ implies a larger deviation of the simulator from the states represented in the real world data. This implies a higher dissimilarity between the simulator and real world data. Our proposed similarity metric is therefore the size of $\mathcal{M}$, with a smaller $\mathcal{M}$ implying a more accurate simulation with respect to the data.

In order to quantify the size of the error distribution $\mathcal{M}$, we use the notion of *entropy* from information theory as a measure of the unpredictability of a vector $\mathbf{m}$ from the distribution. The *entropy*

of the distribution $\mathcal{M}$ measures the amount of information that is missing from the simulator $\hat{f}$ that would be needed to completely model the function f and capture true crowd motion. As a result, given two simulators, $\hat{f}_1$ and $\hat{f}_2$, the algorithm for which the entropy of $\mathcal{M}$ is lower for some given data is regarded as a better match for that dataset. This leads to the following definition.

Entropy Metric: *The entropy of the distribution $\mathcal{M}$ of errors between the evolution of a crowd predicted by a simulator $\hat{f}$ and by the function f* (lower is better).

A lower value of this entropy implies a smaller error distribution and better similarity with respect to that dataset.

Given this definition of the entropy metric, the underlying challenge is to determine the series of true crowd states $(\mathbf{x}_1 \dots \mathbf{x}_t)$. Because the true states are unknown, we need to estimate them from noisy, real-world crowd data $(\mathbf{z}_0, \dots, \mathbf{z}_t)$. We note that for any given data there are multiple possible crowd simulation states. Instead of inferring one true state $\mathbf{x}_k$, we infer a distribution of likely states $\mathcal{X}_k$. This procedure naturally accounts for data noise and simulator uncertainty. The remainder of this section describes our procedure for estimating the simulations states $\mathcal{X}$ and the error distribution $\mathcal{M}$.

3.1 Computing the Entropy Metric

We compute the Entropy Metric using a two phase process. Firstly, we estimate the crowd states $\mathcal{X}$ from the given validation data. Secondly, for each transition between inferred crowd states, from $\mathcal{X}_k$ to $\mathcal{X}_{k+1}$, we then compute the distribution of prediction errors $\mathbf{m}_k = \mathcal{X}_{k+1} - \hat{f}(\mathcal{X}_k)$ using a maximum likelihood estimator.

To reiterate, the true crowd states and transitions are unknown and must be inferred from the real-world validation data $\mathbf{z}_0, \dots, \mathbf{z}_t$. We estimate the simulator states using *Bayesian Inference* [McLachian and Krishnan 1996]. This is a process which takes observed data $(\mathbf{z})$, a model of how states evolve over time $(\hat{f})$, and an estimate of this model's accuracy $(\mathcal{M})$ to produce an estimate of the likely distribution of true simulation states $(\mathcal{X})$, as in Figure 3.

Unfortunately, this process creates a circular dependency: to estimate the prediction error $\mathcal{M}$ we must know the true crowd states $\mathcal{X}$, and to estimate the true crowd states $\mathcal{X}$ from the data we must know the prediction error $\mathcal{M}$. We solve this problem by taking an iterative approach, first using our best (most recent) guess of $\mathcal{M}$ to infer $\mathcal{X}$, then using this guess of $\mathcal{X}$ to infer $\mathcal{M}$, and continuing to alternate between these two steps until convergence (Figure 3).

This iterative approach to estimation is known as the *EM-algorithm*, and is guaranteed to converge in a coordinate-ascent manner to a locally optimal estimate of the distributions $\mathcal{X}_k$ and $\mathcal{M}$ (in terms of their *likelihood*) given the observed data $\mathbf{z}_0, \dots, \mathbf{z}_t$. This process is summarized in Figure 3 and discussed in detail bellow. This will directly estimate the error distribution $\mathcal{M}$ whose entropy serves as the evaluation metric for the crowd simulator $\hat{f}$. Further discussion of the theoretical foundations of EM and convergence conditions can be found in [McLachian and Krishnan 1996].

3.2 Simplifying Assumptions

There are many difficulties in computing the Entropy Metric exactly, arising from both theoretical and practical issues related to the underlying complexity and non-linear aspects of crowd dynamics, combined with the general non-parametric nature of the distribution $\mathcal{M}$. This makes it necessary to find appropriate approximations and simplifying assumptions which allow us to compute an approximated value of the Entropy metric.

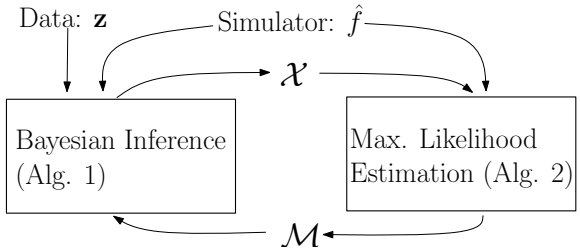


Figure 3: *We estimate the error distribution $\mathcal{M}$ for a crowd simulatior via an iterative process, based on the EM-algorithm. We use Bayesian Inference to estimate the true crowd states $\mathcal{X}$ given data $\mathbf{z}$, a simulator $\hat{f}$, and an error distribution $\mathcal{M}$. Next, we compute the maximum likelihood estimate of $\mathcal{M}$ given the simulator and estimated state distributions $\mathcal{X}$. This process is repeated until convergence.*

Our most important assumption is that all relevant distributions for computing the metric can be modeled as Gaussians. In order to make the computations tractable, we need to represent the distributions $\mathcal{M}$ and $\mathcal{X}_k$ in some parametric form, and it is natural to choose the first two moments (mean and variance) of the distribution as the relevant parameters. Given only the mean and variance this distribution, the maximum entropy principle suggests a Gaussian distribution as it imposes the least additional constraints on the distribution. Additionally, the Central Limit Theorem suggests that if these error distributions are the results of the combination of many independent sources of error, they can be well modeled as a Gaussian.

We therefore represent the distribution $\mathcal{X}_k$ of the state at time k and the error distribution $\mathcal{M}$ as Gaussian. Furthermore, we assume that a crowd state $\mathbf{x}_k$ is composed of the states of each of the n individual agents within the crowd. Hence, if the state of a single agent has dimension d, then the dimension of the composite crowd state is nd. We make three further assumptions regarding the error distribution $\mathcal{M}$: (1) The crowd simulator has no systemic bias in the error of its predictions; (2) The crowd simulator is not systemically more accurate for some agents within a crowd than for others; (3) There is no systemic covariance between the prediction errors of different agents within the crowd. Hence, we can assume that the distribution $\mathcal{M}$ has a zero mean, and that its covariance matrix is block-diagonal;

$$\mathcal{M} = \mathcal{N}(\mathbf{0}, \begin{bmatrix} M & 0 & \cdots & 0 \\ 0 & M & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & M \end{bmatrix}), \tag{5}$$

where M is a $d \times d$ covariance matrix which appears n times along the diagonal. In this case, M models the per-agent error variance of the crowd simulator, and the distribution $\mathcal{M}$ is fully defined by the covariance matrix M. This representation also allows for the use of datasets of crowds of different sizes to evaluate a crowd simulator.

3.3 Computing Crowd State Distributions

Computing the Entropy metric involves estimating the prediction error distribution $\mathcal{M}$. As discussed in Section 3.1, given real-world data $\mathbf{z}_0, \dots, \mathbf{z}_t$, and a simulator $\hat{f}$, we simultaneously estimate the crowd's simulation state $\mathcal{X}$ and error distribution $\mathcal{M}$ using the EM-algorithm. We first describe the mathematical details of estimating the true simulation states from the data.

To estimate the true crowd states $\mathcal{X}$ from the data $\mathcal{Z}$, we use a Bayesian estimation technique based on a variant of the well-known

Kalman filter. A Kalman filter provides an optimal estimate of the true state of a model given noisy data, assuming that the model is linear and the noise is Gaussian. While we make a Gaussian noise assumption, we know that crowd models are highly non-linear. Additionally, a Kalman filter estimates the true state at any timestep based only on past data. However, we typically have data both before and after each timestep and can use both to improve the estimate of the true state at any timestep.

Given the above considerations, we use a method know as Ensemble Kalman Smoothing (EnKS) [Evensen 2003]. The EnKS inference algorithm represents the simulation state using a collection of several samples of possible simulator states (called an ensemble). Each sample is updated based on the non-linear motion model of the crowd simulator, and iteratively modified to correspond to the past and future observed data in a manner consistent with the model error $\mathcal{M}$ and data uncertainty $\mathcal{Q}$. The result is a robust estimate of the true simulation state of each agent for each timestep, based on the global data over all agents over all past and future timesteps.

The EnKS algorithm is known to work particularly well for high-dimensional state spaces and non-linear dynamics [Evensen 2003]. In our case, each distribution $\mathcal{X}_k$ is represented by an ensemble of m samples: $\mathcal{X}_k = \{\hat{\mathbf{x}}_k^{(1)}, \ldots, \hat{\mathbf{x}}_k^{(m)}\}$, and we assume an initial ensemble $\mathcal{X}_0$ is given. The method then proceeds as shown in Algorithm 1. This computes a representation for $\mathcal{X}_0, \ldots, \mathcal{X}_t$, given a current estimate of $\mathcal{M}$ and the trajectory data $\mathbf{z}_0, \ldots, \mathbf{z}_t$.

Algorithm 1: EnKS for estimating crowd states

Input: Measured crowd data $\mathbf{z}_1 ... \mathbf{z}_k$, Crowd Simulator $\hat{f}$, Estimated error variance M
Output: Estimated crowd state distributions $\mathcal{X}_1 ... \mathcal{X}_k$
foreach $k \in 1 \ldots t$ **do**
  // Predict
  foreach $i \in 1 \ldots m$ **do**
    Draw $\mathbf{m}_{k-1}^{(i)}$ from $\mathcal{M}$
    $\hat{\mathbf{x}}_k^{(i)} = \hat{f}(\hat{\mathbf{x}}_{k-1}^{(i)}) + \mathbf{m}_{k-1}^{(i)}$
    Draw $\mathbf{q}_k^{(i)}$ from $\mathcal{Q}$
    $\hat{\mathbf{z}}_k^{(i)} = h(\hat{\mathbf{x}}_k^{(i)}) + \mathbf{q}_k^{(i)}$
  $\bar{\mathbf{z}}_k = \frac{1}{m} \sum_{i=1}^{m} \hat{\mathbf{z}}_k^{(i)}$
  $Z_k = \frac{1}{m} \sum_{i=1}^{m} (\hat{\mathbf{z}}_k^{(i)} - \bar{\mathbf{z}}_k)(\hat{\mathbf{z}}_k^{(i)} - \bar{\mathbf{z}}_k)^T$
  // Correct
  foreach $j \in 1 \ldots k$ **do**
    $\bar{\mathbf{x}}_j = \frac{1}{m} \sum_{i=1}^{m} \hat{\mathbf{x}}_j^{(i)}$;
    $\Sigma_j = \frac{1}{m} \sum_{i=1}^{m} (\hat{\mathbf{x}}_j^{(i)} - \bar{\mathbf{x}}_j)(\hat{\mathbf{z}}_k^{(i)} - \bar{\mathbf{z}}_k)^T$
    foreach $i \in 1 \ldots m$ **do**
      $\hat{\mathbf{x}}_j^{(i)} = \hat{\mathbf{x}}_j^{(i)} + \Sigma_j Z_k^{-1} (\mathbf{z}_k - \hat{\mathbf{z}}_k^{(i)})$

3.4 Computing the Variance M

The second step of the EM-algorithm consists of computing the maximum-likelihood estimate of the distribution $\mathcal{M}$, given the current estimates of the distributions $\mathcal{X}_0, \ldots, \mathcal{X}_t$ of the crowd states as estimated in the first step.

Recall that we do not estimate a single crowd state, but rather a distribution of likely states. Therefore, rather than compute $\mathcal{M}$ directly, we must instead find the most likely value for $\mathcal{M}$ given the inferred distributions of $\mathcal{X}_k$. Since $\mathcal{M}$ is fully defined by the variance M, this is equivalent to maximizing the expected likelihood of M (see Eqn. 5). Further, it is mathematically convenient to maximize the expected *log-likelihood* $\ell\ell(M)$ (as the logarithm cancels against the exponent in the probability density function of a Gaussian distribution), which is equivalent since the logarithm is a monotonic function.

We denote the part of the nd-dimensional crowd state $\mathbf{x}_k$ that contains the state of agent $j \in 1 \ldots n$ as $\mathbf{x}_k[j]$. The expected log-likelihood of variance matrix M is given by:

$$\mathrm{E}(\ell\ell(M)) = -\sum_{k=0}^{t-1} \sum_{j=1}^{n} \mathrm{E}\big((\mathbf{x}_{k+1}[j] - \hat{f}(\mathbf{x}_k)[j])^T M^{-1} \cdot (\mathbf{x}_{k+1}[j] - \hat{f}(\mathbf{x}_k)[j])\big), \qquad \mathbf{x}_k \sim \mathcal{X}_k. \quad (6)$$

Combining Eqn. 6 with the ensemble representations of the distributions $\mathcal{X}_k$, we can compute the maximum likelihood variance M using Algorithm 2.

Algorithm 2: Maximum Likelihood Estimation

Input: Estimated crowd state distributions $\mathcal{X}_1 ... \mathcal{X}_k$, Crowd simulator $\hat{f}$
Output: Estimated error variance M
$M = 0$;
foreach $k \in 0 \ldots t-1$ **do**
  foreach $i \in 1 \ldots m$ **do**
    foreach $j \in 1 \ldots n$ **do**
      $M \mathrel{+}= (\hat{\mathbf{x}}_{k+1}^{(i)}[j] - \hat{f}(\hat{\mathbf{x}}_k^{(i)})[j])(\hat{\mathbf{x}}_{k+1}^{(i)}[j] - \hat{f}(\hat{\mathbf{x}}_k^{(i)})[j])^T$
$M \mathrel{/}= tmn$

The EM-algorithm is initialized with an initial guess for M and $\mathcal{X}_0$, and both steps are repeatedly performed until convergence. The resulting M is a (local-)maximum-likelihood estimate of the error variance of crowd simulator $\hat{f}$.

3.5 Computing the Entropy of $\mathcal{M}$

Given the per-agent variance M as computed above, it remains to compute the entropy of the Gaussian distribution $\mathcal{M}$ of Eqn. (5). This entropy is given by:

$$e(\mathcal{M}) = \frac{1}{2} n \log((2\pi e)^d \det(M)), \quad (7)$$

where n is the number of agents in the crowd, and d is the dimension of the state of a single agent. In order to make our metric independent of the number of agents in the crowd, we normalize the above equation by dividing by n. This gives the entropy of the normal distribution $\mathcal{N}(\mathbf{0}, M)$ that models the per-agent error of the crowd simulator $\hat{f}$. We note that this value is proportional to the log of the determinant of the per-agent variance M, meaning the Entropy Metric follows a log-scale.

4 Implementation and Evaluation

In this section, we demonstrate the application of the Entropy Metric to several crowd simulation algorithms. For each simulation method we evaluate the Entropy Metric on several different sets of simulation parameters, across several different scenarios, each with data gathered from different participants in the scenario. The simulation methods, validation scenarios, and data gathering techniques are described below. The entropy scores for all combinations of parameters, scenarios, and simulators are summarized in Table 1. Because the entropy metric is logarithmic, linear difference in scores corresponds to an exponential difference in performance.

Scenario	RVO-1	RVO-2	RVO-3	SFM-1	SFM-2	SFM-3	Steer-1	Steer-2	Steer-3
Passage-1	3.048	2.329	3.400	6.576	6.581	6.579	6.403	6.490	6.435
Passage-2	1.991	0.690	1.990	5.430	5.458	5.451	4.713	4.748	4.764
Street-1	2.744	3.156	2.800	4.500	4.707	4.665	2.979	3.569	3.838
Street-2	2.709	2.564	2.520	3.793	3.885	3.780	2.660	2.744	3.060
Lab	1.920	1.610	1.230	2.538	2.523	2.509	1.871	1.847	2.305

Table 1: *Entropy Metric for different simulation algorithms on various real-world datasets (lower is better).*

4.1 Simulation Models

We chose three popular simulation methods to test the metric with: a rules-based steering approach, a social-forces model, and a predictive planning approach. Many variants and extensions of all these models have been proposed and widely used in different applications.

Steering Simulator: Steering based simulation approaches use a discrete set of rules to choose agent velocities. We chose a simulation technique based on the classical steering method proposed by Reynolds [1999]. Each agent follows three simple behavior-based rules: steer towards the goal, steer away from the nearest obstacle, and steer away from the nearest person. When obstacles are very close by or when collision are imminent, the avoidance rules are given precedence over the goal-following behavior.

Social Forces Simulator: Social force simulation models use potential fields defined by neighboring agents to impart an acceleration to each agent. We chose a simulation technique based on Helbing's Social Force Model (SFM) [Helbing et al. 2000]. SFM computes the trajectory of each agent by applying a series of forces to each agent that depend on the relative positions and velocities of nearby agents. An agent A receives a repulsive force pushing it away from each neighbor B, denoted as f_{AB}. Moreover, each agent experiences a force pushing it perpendicularly away from the walls or obstacles, denoted as f_W. The magnitude of these forces decreases exponentially with the distance. Each agent also has a goal velocity $\mathbf{v}_A^{pref}$, which is used to compute the desired speed and direction. The simulation function $\hat{f}$ can be summarized as:

$$\mathbf{f}_A^{new} = \frac{\mathbf{v}_A^{pref} - \mathbf{v}_A}{\alpha} + \sum_{A \neq B} f_{AB} + \sum_W f_W \tag{8}$$

where α controls the rate of acceleration.

Predictive Planning Simulator: Predictive, planning based simulators attempt to anticipate collisions based on neighboring agents' positions and velocities and determine new paths which avoid these collisions. We chose a velocity-based formulation called *Reciprocal Velocity Obstacle* as implemented in the RVO2 library [van den Berg et al. 2009]. Each agent navigates by constraining its velocity to those which will avoid collisions with nearby neighbors and obstacles for at least τ seconds. The set of velocity constraints imposed by all the neighbors of an agent A is denoted as RVO_A^τ. An agent is also assumed to have a desired velocity $\mathbf{v}_A^{pref}$. The resulting simulation function $\hat{f}$ can be expressed as:

$$\mathbf{v}_A^{new} = \underset{\mathbf{v} \in RVO_A^\tau}{\operatorname{argmin}} \|\mathbf{v} - \mathbf{v}_A^{pref}\|. \tag{9}$$

The avoidance computation is performed using linear programming, and all agents are assumed to reciprocate (share the responsibility) in avoiding collisions.

For each simulation method, three different sets of parameters were chosen which varying collision radii, preferred speeds, and other internal simulation constants. The resulting simulations that use these parameters are referred to as Steer-1, Steer-2, and Steer-3 for the steering-based approach, SFM-1, SFM-2, and SFM-3 for the social-forces based approach, and RVO-1, RVO-2, and RVO-3 for the predictive planning based approach, respectively.

Further information regarding the implementation of each algorithm are given in Appendix B in the supplementary materials, which details the specific parameters used in each simulation. Additional, this appendix B further describes other implementation details including the specific form of the state vectors, validation data and observation function used to obtain the results in this section.

4.2 Real-World Crowd Data

In order to evaluate the Entropy Metric, we use several sources of data. They correspond to different real-world scenarios (both indoor or outdoor) and have varying number of agents. Each dataset was captured using different sensing hardware (see Table 2).

Lab Setting: This data comes from a study performed in a controlled setting in a motion capture lab. In this scene, two people were placed about 6m apart and were asked to swap their positions [Moussaïd et al. 2011] (see Fig. 4a). We label this benchmark with two agents as *Lab*.

Street Crossing: This data comes from a video of pedestrians walking on a street, which was captured using an overhead camera. The trajectories of each agent were extracted using multi-object tracking [Pellegrini et al. 2009] (see Fig. 4b).

Importantly, we use data from two different capture sessions involving different groups of people crossing the same street. This allows us to test for correlation in the results of metric between different groups of people for the same scenario. The datasets from the two groups are labeled as *Street-1* and *Street-2*.

Narrow Passageway: This data comes from a large indoor experiment designed to capture human exiting behavior through passages of varying sizes [Seyfried et al. 2010]. The experiment involved use of markers and optical tracking equipment to gather high-quality data corresponding to subjects' positions near the entrance of the passageway. The experiment was performed with different exit widths, with each run consisting of hundreds of participants, about 50 of whom were in the tracked area at any given time (see Fig. 4c).

Again, we use data from two different runs of this experiment, the first with a passage of very narrow width of 1m and the second with a wider passage of width 2.5m, denoted as *Passage-1* and *Passage-2*, respectively. This is used to analyze correlation between similar scenarios.

For all five scenarios, we assume that the goal position for each agent is the last tracked position in the dataset and compute $\mathbf{v}^{pref}$ accordingly. However, in some scenarios (such as people moving in a maze) this assumption may not hold. In such cases $\mathbf{v}^{pref}$ can be considered as part of the state and inferred along with other parameters using Bayesian inference.

(a) *Lab (small)* **(b)** *Street (medium)* **(c)** *Passage (large)*

Figure 4: *Our rendering of real-world crowd trajectories used for evaluation.*

Scenario	Agents	Density	Capturing technique
Passage-1	40	2.76	optical tracking+camera
Passage-2	59	2.38	optical tracking+camera
Street-1	18	0.42	overhead camera
Street-2	11	0.37	overhead camera
Lab	2	-	motion capture

Table 2: *Real-world crowd datasets used by our evaluation algorithm. We report the average number of agents per frame and the density of the agents over the tracked area.*

4.3 Results

As can be seen in Table 1, different simulators vary in their ability to capture the motion characteristics of different datasets. Furthermore, for a given simulation method, different parameter sets also score better or worse. This suggests that maximizing the similarity to the data involves choosing not only the right simulator, but the right parameters. Some scenarios resulted in a relatively high Entropy Metric value across all simulators. For example, the best score for Passage-1 (the narrower passage) was worse than the worst score for the Lab scenario. This suggests that all the tested simulators performed poorly in terms of capturing the complex behaviors pedestrians exhibited in the narrow passage scenario.

5 Analysis

While Section 4 provides the results of the Entropy Metric on different simulators and validation datasets, this section analyzes the metric itself. Specifically, we analyze the metric in terms of predictiveness, consistency, robustness to noise, correlation with perceptual similarity, and other important properties. To begin with, we highlight several useful properties of the metric which follow directly from its mathematical definition.

Rankable results: For a given validation dataset, the Entropy Metric provides unique, global rankable results because it computes a single number in $\mathcal{R}$. The result can be ranked uniquely when there are no ties. If the Entropy Metric for $\hat{f}_1$ is lower than the Entropy Metric for $\hat{f}_2$, this implies that simulator $\hat{f}_1$ better captures the aggregate dynamics in that dataset than $\hat{f}_2$.

Discriminative: The data presented in Table 1 highlights the discriminative nature of the Entropy Metric. In contrast to approaches which test a simulator against a discrete set of benchmarks, the Entropy Metric returns a real number, eliminating the risk of ties. This allows us to generate a clear quantitative ranking of different simulators.

Generality: The Entropy Metric makes very few assumptions about the underlying simulator and the real-world data. This is because the Bayesian inference framework is capable of estimating the complete simulation state (position, velocity, orientation, etc.) based on only partial validation data (e.g., only positions). This allows to us to compare simulators that use only position and velocities to others that also account for orientation and accelerations, or other simulation specific parameters.

Scenario	Correlation
Passage-1 & Passage-2	.975
Street-1 & Street-2	.917
Street-1 & Passage-2	.585
Passage-1 & Street-2	.414

Table 3: *The Entropy Metric results on similar datasets such as (Passage-1,Passage-2) or (Street-1,Street-2) are highly correlated. The metric shows lower correlation for different dataset pairs.*

Because the entropy measure directly compares a simulation to reference data, it directly reflects optimizations made to increase the accuracy of simulations. Appendix A in the supplemental material provides a case study showing a correlation between improvements to RVO, and a decrease in the Entropy scores of the resulting simulations. These results can also be seen in the supplemental video.

5.1 Consistency

It is important that the Entropy Metric provides consistency in terms of results across similar datasets. Based on the empirical results presented in Table 1, we can determine that the Entropy Metric has this property. Specifically, the results on similar benchmarks are well correlated with each other. For example, the ordering from best to worst simulators for the benchmarks Passage-1 and Passage-2 does not differ significantly even though the data changes. This suggests the metric can reliably capture some inherent aspect of a simulator's ability to reproduce the movement through a passage.

We can numerically measure the correlation between scenarios using Pearson's correlation coefficient r, which measures correlation on a scale from 0 (uncorrelated) to 1 (exactly correlated). The results from computing the Entropy Metric on two different datasets from the same scenario are highly correlated ($r > .9$), implying a consistency in the metric across similar datasets. The results from different datasets (e.g. Street vs Passage) are much less correlated. This is because different simulators have different abilities to capture different types of motion, which is reflected in the metric. These correlation results are summarized in Table 3.

Simulator	RVO-1	SFM-2	Steer-3
RVO-1	**0.19**	3.17	3.16
SFM-2	4.34	**1.38**	1.58
Steer-3	2.26	1.66	**0.90**

Table 4: *Results from using the Entropy Metric to compare two simulators As expected, the Entropy Metric is the smallest when a simulator is compared to itself.*

5.2 Robustness to Noise

Any data-driven analysis of a simulator needs to consider the effects of noise present in the data. Even in controlled lab settings, there are small amounts of sensor noise, the effects of which can be magnified over time. These effects are even more pronounced when working with data captured from outdoor natural scenes. This data is normally produced by video processing techniques, which can have large amounts of errors compared to data gathered in a controlled lab setting.

We can analyze the effect of noise on the Entropy Metric by artificially adding noise to the validation datasets. In particular, we highlight the results on the data from outdoor street crossing scenario. We add uniformly distributed noise to the data of up to a half meter in size (while keeping $\mathcal{Q}$ constant). We then compute the Entropy Metric for three different simulators (RVO-1, SFM-2, and Steer-3) by varying the noise. The results are shown in Fig 5.

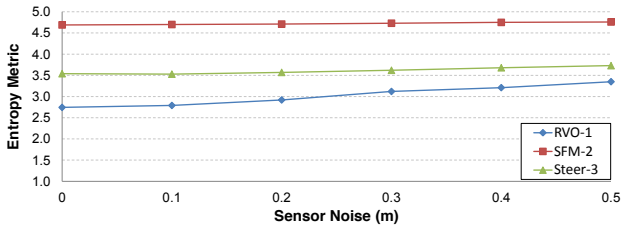


Figure 5: *We evaluate the impact of adding artificial error (uniformly distributed) to the Street-1 benchmark for different simulators. The Entropy Metric is relatively stable to this error and the relative ranking of different simulators does not change.*

As expected, the Entropy score gets worse for all simulators as more noise is added because the error from the noise is being attributed to the simulation. However, the metric handles the noise robustly, with the value of the metric changing in a slow, continuous fashion as large amounts of noise are added. Even with an extreme value of .5m of noise being added randomly every timestep, the metric maintains the same relative ranking between the simulators.

5.3 Similarity between Two Simulators

While the Entropy Metric is designed to compare a simulator to a validation dataset, we can also use it to compare two simulators. This comparison is performed by running a simulator to generate paths and using these paths as the validation dataset for the second simulator. In this way we can compute a score measuring the similarity of two simulators, with a lower score implying a better match between the aggregate dynamics of two simulators. Table 4 shows the results of this comparison on three different simulators. The results are asymmetric mainly due to differences in how sensitive each simulator is to the noise. A symmetric comparison can be achieved by averaging the pairwise differences.

As expected, the similarity matrix is minimized along the diagonal, indicating that each simulation is most closely related to itself. Additionally, the correlation between RVO and two non-predictive models is much worse than the correlation between SFM and Steer (i.e., $\sim$3 vs $\sim$1.5). We speculate that this is due to the fact that RVO uses a predictive approach to collision avoidance, while the other two approaches both use a model based on reactive, distance-based forces.

5.4 Data-driven Crowds and Behavior Modeling

Data-driven approaches are becoming more common in crowd simulation. In part, this is because of their ability to capture complex or subtle behaviors that to not directly map to simulation parameters. Here, we explore the Entropy Metric as it applies to a data-driven crowd simulation designed to capture high-level personality differences. Specifically, we look at simulators based on high-level personality models such as the Five-factor Trait Theory or OCEAN model. This model classifies a personality based on its level of Openness, Conscientiousness, Extraversion, Agreeableness and Neuroticism [Costa and McCrae 1992]. Recent crowd work such as [Durupinar et al. 2008] and [Guy et al. 2011] have proposed data-driven models for each of these personality traits.

In order to evaluate our metric, we analyze one trait (Extraversion), similar results also hold for other data-driven models of high level behaviors. We use a validation dataset consisting of five agents walking past each other with a high degree of Extraversion (as reported by [Guy et al. 2011]), and vary the simulator to have differing amounts of Extraversion. The Entropy results match well when we choose a simulator trained to match a high Extraversion dataset, and the Entropy metric was worse when using a simulator trained for low amounts of perceived Extraversion. As we interpolate between the two simulators using the perceptually linear personality space defined in [Guy et al. 2011], we see a linear improvement in the Entropy metric scores as the simulator moves from less Extraversion to more, see Figure 6. The linearity of the match suggests the Entropy Metric may also be well correlated with a perceptual notion of similarity, this is further explored in Section 5.5.

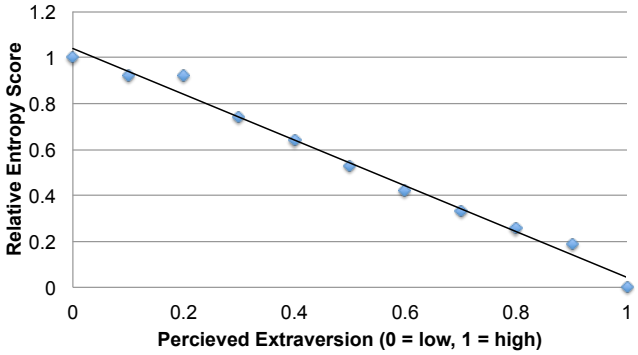


Figure 6: *Comparison of varying levels of the Extraversion personality trait to the Entropy score given an dataset of Extraverted motion. As the data-driven simulation is modified from a less Extraverted model, towards a more extraverted one the Entropy metric with respect to a highly Extraverted dataset decreases.*

5.5 Comparison to Perceptual Evaluation

We conducted a user study to analyze how well the numerical similarity of the Entropy metric corresponds to perceptual similarity. This study involved 36 participants (22 males) and had two sections; the first was designed to directly investigate the correlation between the Entropy metric and perceived similarity, and the second section was designed to analyze how well the metric can predict a user's perceived similarity. When studying perceptual evalu-

ation in crowd videos it is important to note the effect that rendering choices, such as cloned appearance and motion of individuals, can affect a user's perception of simulated crowds [McDonnell et al. 2008]. To mitigate this effect, we rendered all motion users saw with the same visual crowd models and rendering parameters.

In the first section of the study, users were shown two videos. The first was a rendering of the simulation and the second was a rendering of the real-world data. The real-world motion was re-rendered with the crowd rendering system used for the simulated motion. Users were asked to rate the pairs of videos in terms of their similarity to each other on a Likert Scale of 0 (not at all similar) to 10 (very similar); this was done across four different simulators, each with a different entropy score compared to the real-world data. The results are shown in Fig 7.

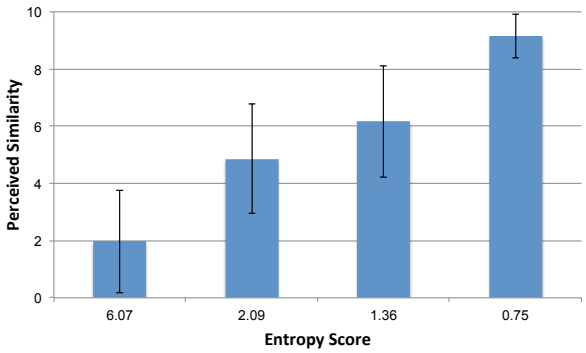


Figure 7: *Comparison of Entropy score (lower is better) to perceived similarity (higher is better) across four different simulators, error bars represent mean absolute deviation in user scores. Simulators with a lower Entropy Metric were consistently given a higher score in terms of perceived similarity to the source video by users.*

As can be seen in Fig 7, when the entropy score is very large (6.07), users gave the simulation a low score for similarity, generally ranging from 0 to 4. When the entropy score was very small (0.75), users gave the simulation a high score for similarity, generally ranging from 8 to 10. For entropy scores in-between, the users gave correspondingly intermediate similarity scores. Numerically, the Entropy Metric and user reported perceived similarity have a Pearson's correlation coefficient of .91, which indicates the Entropy Metric is strongly correlated with perceived similarity.

The second section of the user study was structured using the two-alternative forced choice (2AFC) procedure. For each question, a user was shown videos from two different simulations along with reference video from real-humans walking in the same environment. The users were asked two questions: which simulator matched the real-world data better and whether the two simulators had similar or different behaviors.

A priori, we would expect users to choose the simulator with a lower Entropy score as the one which matched the data better, as this is what the Entropy score seeks to measure. Figure 8 shows the accuracy of this prediction versus the relative differences in Entropy score between the two videos. When the Entropy score was greater than 0.1, the metric correctly predicted the user response with a high accuracy rate, and at statistically significant level ($p \leq .01$).

When the relative difference in Entropy scores between two simulations were very small (less than 0.1), the metric failed to correctly predict user preferences at a statistically significant level. However, for these scenarios the metric correctly classifies the simulators as "very similar", with users classifying these simulations as similar 94.3% of the time.

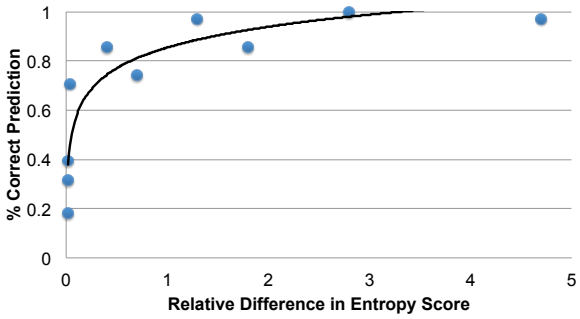


Figure 8: *Relative entropy differences versus user preferences. When the entropy score difference between two simulators was more than 0.1, users agreed with the score's prediction at a statistically significant rate.*

To summarize, the Entropy Metric correlates well with perceived similarity. When the metric indicated a large difference between two simulators, the users agreed with the metric at a statically significant rate (i.e. a rate much greater than chance). When the metric indicated very small differences, users also agreed that the difference was small at a statistically significant rate. This result is significant for those using this metric to design data-driven simulators and for those who want to evaluate simulators used in applications such as games and special effects. By finding simulation parameters that minimize the Entropy Metric, the resulting simulations will visually appear progressively closer to the target data without the need for a human in the loop to evaluate the intermediate results.

These results also provide a meaningful scale for the metric. The sharp inflection point in Figure 8 around 0.1 suggests that this value may be a good estimate of the just-noticeable-difference level in the metric. Likewise, Figure 7 suggests that simulations with an Entropy score less than 1 should be considered very visually similar to the source data and those with a score greater than 6 visually very different.

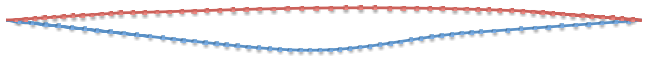
5.6 Motion Uncertainty

As discussed in Section 2.4, many approaches have been suggested for evaluating a simulators ability to match data. The main novelty of the Entropy Metric comes from its ability to effectively handle sensor noise, its correlation with perceptual similarity, and its robust treatment of the uncertainty in human motion. The first two of these features have been discussed above; here, we analyze the effect of motion uncertainty.

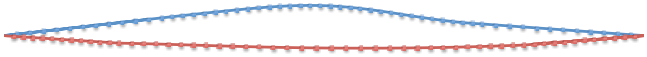
When two people interact, their paths can vary for a variety of reasons. However, small variations early in a path can lead to large deviations later on. These deviations make it hard to compare two trajectories directly. This is what makes comparing trajectories an inappropriate similarity metric. To illustrate this, we used data corresponding to two people swapping their positions in the Lab scenario and mirrored the paths so the participants pass on the right rather than on the left. Both of these paths are equally valid ways of getting between the given start and end positions, and ideally would result in similar validation scores for a simulation.

Since the Entropy Metric does not a compare a simulator to the trajectory data directly, but rather to the decisions latent in that data, it computes nearly identical scores for the two datasets. For example, the RVO-3 simulator scores 1.91 for the original and 1.92 for

the mirrored data. However, if average trajectory differences were used as the metric, the score would change significantly. This is because one simulator can predict passing either on the left or the right, but not both. In contrast, because the Entropy Metric is computed over a *distribution* of likely simulator states, this results in similar scores for two datasets.



(a) *Pass on Left (Entropy for RVO-3: 1.91)*



(b) *Pass on Right (Entropy for RVO-3: 1.92)*

Figure 9: *Two agents (red and blue) switch positions. Regardless of whether the agents pass each other on their left sides (a) or their right sides (b), the Entropy Metric for a given simulator remains almost unchanged.*

6 Assumptions and Limitations

When deriving our approach, we have made some basic assumptions that can help inform the appropriate use of the Entropy Metric for simulation evaluation. Most importantly, we assume that there exists some corpus of representative crowd data for a simulator to be compared to. Because the metric evaluates at the level of individual motion decisions it is important that the validation dataset contains motion which is representative of the types of motions considered realistic. Additionally, the basic formulation of the entropy metric does not directly measure characteristics arising from emergent phenomena, such as density, lane formation, and overall flow rate, though such quantities may be indirectly inferred through the motion of individuals in the crowd and would be of interesting topics for future investigation.

The formulation of the entroy metric assumes that all uncertainties are known and can be modeled with zero-meaned gaussian distributions. Such uncertainties include uncertainty in the validation data, errors in the estimation of the environment, senor noises, etc. This assumption extends to the error of a given simulation technique, which is what the Entropy metric seeks the measure. If any of these errors do not fit this model (for example, a simulation technique which has a systematic bias), our method will naturally account for these by estimating a larger variance for the error distribution. Likewise, any errors in modeling the environment or sensor uncertainty will also increase the estimated error of a simulation (see Section 5.2). By explicitly estimating model uncertainty, the Entropy metric seeks to account for the aggregate effect of the non-determinism in human motion, errors in sensors, and unmodeled sources of error. In this way, our metric can gracefully handle a breakdown of the mathematical assumptions underlying its derivation.

While our metric is simple and provides a consistent, rankable measure of similarity to source data as a single real number, the metric is not invariant to changes in timestep and measurement units. We also assume that the sensing accuracy $\mathcal{Q}$ is known, though it may be possible to extend our approach to unknown distributions.

In contrast to other methods that measuring density, collision counts, and other quantities, this approach instead estimates the error between the motion computations performed by a given simulator and with those captured in the validation dataset. Fundamentally, this type of analysis is applicable to where there is known ground truth motion data, along with knowledge about the environment. Therefore, the Entropy Metric is only defined for a given set of real-world crowd data and cannot directly compare different simulators in the absence of such data. It cannot measure the plausibility of a simulator in the absence of real-world 'ground truth', nor can it be directly applied to scenarios without data to validate against.

7 Conclusion and Future Work

We have introduced an Entropy Metric for evaluating crowd simulators against real-world crowd data. Our metric provides a meaningful quantification that can be used to rank various crowd simulators in a predictive and consistent manner. To the best of our knowledge, this is the first attempt at designing an objective, quantitative evaluation metric that measures the similarity of crowd simulation results with respect to real-world data and explicitly accounts for sensor noise, motion uncertainty, and non-determinism. We have used it to evaluate the performance of steering behaviors, force-based models, and predictive crowd simulation algorithms on different real-world datasets.

In the future, we hope to extend this work to analyze other widely used crowd simulation techniques, such as continuum techniques, vision-based steering, data-driven approaches, foot-step planners, and cognitive models – all of them can be described using continuous formulations introduced in this paper. Furthermore, we would like to apply our metric to a wider variety of real-world crowd data sets that measure many different aspects of motion, including local density, individual behaviors, and apparent personalities. We are also interested in exploring applications of this metric, such as automatic optimization of crowd simulations to data.

Acknowledgements We are grateful comments from the reviewers, and thankful for help from Sujeong Kim and Sean Curtis. This research is supported in part by ARO Contract W911NF-04-1-0088, NSF awards 0917040, 0904990, 100057 and 1117127, and Intel.

References

COSTA, P., AND MCCRAE, R. 1992. *Revised NEO Personality Inventory (NEO PI-R) and Neo Five-Factor Inventory (NEO-FFI)*. Psychological Assessment Resources.

DURUPINAR, F., ALLBECK, J., PELECHANO, N., AND BADLER, N. 2008. Creating crowd variation with the OCEAN personality model. In *Autonomous agents and multiagent systems*.

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

EVENSEN, G. 2003. The ensemble kalman filter: theoretical formulation. *Ocean Dynamics 55*, 343–367.

FUNGE, J., TU, X., AND TERZOPOULOS, D. 1999. Cognitive modeling: Knowledge, reasoning and planning for intelligent characters. *Proc. of ACM SIGGRAPH*, 29–38.

GALLAGHER, R., AND APPENZELLER, T., Eds. 1999. *Science Magazine*, vol. 284. AAAS.

GUY, S., CHUGGANI, J., CURTIS, S., DUBEY, P., LIN, M., AND MANOCHA, D. 2010. Pledestrians: A least-effort approach to crowd simulation. *Proc. of Eurographics/ACM SIGGRAPH Symposium on Computer Animation*, 119–128.

GUY, S. J., KIM, S., LIN, M., AND MANOCHA, D. 2011. Simulating heterogeneous crowd behaviors using personality trait theory. In *Eurographics/ACM SIGGRAPH Symposium on Computer Animation*, The Eurographics Association, 43–52.

HELBING, D., AND MOLNAR, P. 1995. Social force model for pedestrian dynamics. *Physical Review E 51*, 4282.

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

JARABO, A., EYCK, T. V., SUNDSTEDT, V., BALA, K., GUTIERREZ, D., AND O'SULLIVAN, C. 2012. Crowd light: Evaluating the perceived fidelity of illuminated dynamic scenes. *Proc. of Eurographics*. to appear.

KAPADIA, M., WANG, M., SINGH, S., REINMAN, G., AND FALOUTSOS, P. 2011. Scenario space: characterizing coverage, quality, and failure of steering algorithms. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 53–62.

KARAMOUZAS, I., HEIL, P., BEEK, P., AND OVERMARS, M. 2009. A predictive collision avoidance model for pedestrian simulation. *Proc. of Motion in Games*, 41–52.

KRATZ, L., AND NISHINO, K. 2011. Tracking pedestrians using local spatio-temporal motion patterns in extremely crowded scenes. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 99, 1–1.

LEE, H., CHOI, M., HONG, Q., AND LEE, J. 2007. Group behavior from video: a data-driven approach to crowd simulation. In *Proc. of Symposium on Computer Animation*, Eurographics Association, 109–118.

LERNER, A., CHRYSANTHOU, Y., AND LISCHINSKI, D. 2007. Crowds by example. *Computer Graphics Forum (Proceedings of Eurographics) 26*, 3.

LERNER, A., CHRYSANTHOU, Y., SHAMIR, A., AND COHEN-OR, D. 2009. Data driven evaluation of crowds. In *Proceedings of the 2nd International Workshop on Motion in Games*, 75–83.

MCDONNELL, R., LARKIN, M., DOBBYN, S., COLLINS, S., AND O'SULLIVAN, C. 2008. Clone attack! perception of crowd variety. In *ACM Transactions on Graphics (TOG)*, vol. 27, ACM, 26.

MCLACHIAN, G., AND KRISHNAN, T. 1996. *The EM Algorithm and Extensions*. John Wiley and Sons.

MEHRAN, R., OYAMA, A., AND SHAH, M. 2009. Abnormal crowd behavior detection using social force model. In *Proc. of Computer Vision and Pattern Recognition*, 935–942.

MOUSSAÏD, M., HELBING, D., AND THERAULAZ, G. 2011. How simple rules determine pedestrian behavior and crowd disasters. *Proceedings of the National Academy of Sciences 108*, 17, 6884.

NARAIN, R., GOLAS, A., CURTIS, S., AND LIN, M. C. 2009. Aggregate dynamics for dense crowd simulation. *ACM Transactions on Graphics (Proc. of ACM SIGGRAPH Asia) 28*, 5, 122.

ONDREJ, J., PETTRE, J., OLIVIER, A., AND DONIKAN, S. 2010. A synthetic-vision based steering approach for crowd simulation. *ACM Trans. on Graphics 29*, 4, 123:1–123:9.

PATIL, S., VAN DEN BERG, J., CURTIS, S., LIN, M. C., AND MANOCHA, D. 2011. Directing crowd simulations using navigation fields. *IEEE Trans. on Vis. and Comp. Graphics 17*, 2, 244–254.

PELECHANO, N., ALLBECK, J. M., AND BADLER, N. I. 2007. Controlling individual agents in high-density crowd simulation. *Proc. of Symposium on Computer Animation*, 99–108.

PELECHANO, N., STOCKER, C., ALLBECK, J., AND BADLER, N. 2008. Being a part of the crowd: towards validating vr crowds using presence. In *Proc. of 7th Int. Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 136–142.

PELLEGRINI, S., ESS, A., SCHINDLER, K., AND VAN EOOL, L. 2009. You'll never walk alone: Modeling social behavior for multi-target tracking. In *Proc. of Int. Conf. on Computer Vision*, 261–268.

PETTRE, J., ONDREJ, J., OLIVIER, A., CRETUAL, A., AND DONIKIAN, S. 2009. Experiment-based modeling, simulation and validation of interactions between virtual walkers. In *Proc. of Symposium on Computer Animation*, ACM, 189–198.

REYNOLDS, C. W. 1987. Flocks, herds and schools: A distributed behavioral model. *Proc. of ACM SIGGRAPH 21*, 25–34.

REYNOLDS, C. W. 1999. Steering behaviors for autonomous characters. *Game Developers Conference*.

RODRIGUEZ, M., ALI, S., AND KANADE, T. 2009. Tracking in unstructured crowded scenes. In *Computer Vision, 2009 IEEE 12th International Conference on*, 1389–1396.

SCHADSCHNEIDER, A., CHOWDHURY, D., AND NISHINARI, K. 2011. *Stochastic Transport in Complex Systems: From Molecules to Vehicles*. Elsevier.

SEYFRIED, A., BOLTES, M., KÄHLER, J., KLINGSCH, W., PORTZ, A., RUPPRECHT, T., SCHADSCHNEIDER, A., STEFFEN, B., AND WINKENS, A. 2010. Enhanced empirical data for the fundamental diagram and the flow through bottlenecks. *Pedestrian and Evacuation Dynamics 2008*, 145–156.

SINGH, S., KAPADIA, M., REINMANN, G., AND FALOUTSOS, P. 2009. Steerbench: A benchmark suite for evaluating steering behaviors. *Computer Animation and Virtual Worlds 20*, 533–548.

SUNG, M., GLEICHER, M., AND CHENNEY, S. 2004. Scalable behaviors for crowd simulation. *Computer Graphics Forum 23*, 3 (Sept), 519–528.

TREUILLE, A., COOPER, S., AND POPOVIC, Z. 2006. Continuum crowds. *Proc. of ACM SIGGRAPH*, 1160 – 1168.

VAN DEN BERG, J., GUY, S. J., LIN, M. C., AND MANOCHA, D. 2009. Reciprocal n-body collision avoidance. *Proc. of International Symposium on Robotics Research (ISRR)*, 3–19.

YU, Q., AND TERZOPOULOS, D. 2007. A decision network framework for the behavioral animation of virtual humans. In *Proc. of Symposium on Computer animation*, 119–128.

Appendices

A Case Study: Optimizing RVO

The Entropy metric seeks to capture how well a simulator captures the aggregate dynamics of the motion observed in real-world data. For simple scenarios it maybe possible to perform other, simple data analysis, such as measuring average speed or minimum distance between the agents to reliably indicate a failure to capture the recorded motion. Examples include measuring average speed or minimum distance between the agents. This type of analysis is commonly used to improve the simulator and lower the entropy score.

As an example, we chose the Lab scenario of two people passing each other. First we evaluated the Entropy metric with an intentionally poor set of parameters, labeled RVO-A, which resulted in a large Entropy score of 6.07. Next, we improve the simulator by measuring the closest distance between the two participants and using that parameter to set the radius (RVO-B). The simulator's similarity can be further improved by setting the preferred agent speed to be the average speed of the two participants (RVO-C). Finally, we increase the time horizon over which agents plan their collision avoidance to improve the anticipation in motion (RVO-D). As shown in Table 5, each change in the parameters lowers the Entropy score.

The simulations generated from each step of changing the parameters can be seen in the accompanying video. We note that because of the logarithmic scale of the Entropy metric, a reduction in score from 6.07 to 0.75 indicates more than an order of magnitude in reduction of error. The correlation between hand-tuned results and the Entropy metric provide an indication of the metric's ability to capture meaningful errors in terms of aggregate dynamics.

RVO	Metric	Rad.	Pref Speed	Time Horiz	Improvement Method
A	6.07	1.0m	0.9 m/sec	0.1 sec	-
B	2.09	.25m	0.9 m/sec	0.1 sec	Match radius from data
C	1.36	.25m	1.2 m/sec	0.1 sec	Match speed from data
D	0.75	.25m	1.2 m/sec	1.0 sec	Increase planning horizon

Table 5: *As we improve the similarity of the RVO benchmark based on the data from Lab benchmark, the entropy metric decreases. RVO-A is the worst algorithm for this benchmark and RVO-D is the best algorithm (as shown in video). We see a direct correlation between the entropy metric and similarity of the simulation algorithm.*

This result also indicate that the Entropy metric can be used to design data-driven crowd simulation algorithms. Given a motion data from a target set of behaviors, changing the simulator parameters to optimize the Entropy metric will result in a simulator that closely matches the desired behavior.

B Implementation Details

For the results presented in Section 4 we used the same crowd state vector format ($\mathcal{X}_k$), the same observed validation data ($\mathbf{z}_0, \ldots, \mathbf{z}_t$), and the same observation function (h) for all simulators tested. We varied only the simulator ($\hat{f}$). We briefly give details of each below.

State Vector ($\mathcal{X}_k$) The state of each agent was defined as a four dimensional vector of 2D position and 2D velocity.

Validation Data ($\mathbf{z}_0, \ldots, \mathbf{z}_t$) The validation data used was a series of 2D positions for each person being tracked, the data was generally between 10-15Hz.

Observation Function (h) The observation function must be defined to convert an instance of the state vector to the format found in the validation data. In our case, the function h simply returns the position component of the state vector $h((pos.x, pos.y, vel.x, vel.y)) = (pos.x, pos.y)$.

Simulation Function ($\hat{f}$) The details of this function varies between different simulators and each are discussed in more detail below. All simulators share some common features: all agents were restricted to a maximum velocity of 2.5m/s and all simulators considered only the 10 closest neighbors in velocity computations.

B.1 Social Force Simulation Details

Our implementation of a social force model (SFM) approach to simulation is based on the approach detailed by Helbing et al. [2000]. Each agent experiences an avoidance force from all neighboring agents with decreases exponential with distance from the agent, along with frictional and pushing forces that effect agents in contact with each other. All of the *SFM* simulators shared the same force balancing constants: a social scaling force of 2000N, an agent reaction time of 0.5s, a repulsive pushing spring constant of $120,000\ kg/s^2$, and a sliding friction constant of $240,000\ kg/s^2$.

Simulator	Pref Speed (m/s)	Radius (m)	Mass (kg)
SFM-1	1.40	0.16	80
SFM-2	1.10	0.31	80
SFM-3	1.20	0.20	80

Table 6: *Parameters used in various social force simulators.*

B.2 RVO Simulation Details

Our implementation of a predictive planning approach to simulation is based on the Reciprocal Velocity Obstacle (RVO) based approach detailed by Van den Berg et al. [2009]. Each agent chooses a velocity towards its goal which will minimize it's expected collision with all neighboring agents. The *RVO* simulators vary in the radius and preferred speed of the simulated agents as well as the time horizon agents plan over and the maximum distance at which a neighbor can effect planning. The various simulators are detailed below:

Simulator	Pref Speed (m/s)	Radius (m)	Tim Horiz. (s)	Neighbor Dist. (m)
RVO-1	1.44	0.20	1.7	41
RVO-2	1.10	0.20	1000	41
RVO-3	1.20	0.25	1000	200

Table 7: *Parameters used in various RVO simulators.*

B.3 Steering Simulation Details

Our implementation of a steering approach to simulation is based on the approach in OpenSteer by Reynolds [1999]. The final action of each agent is decided by a combination of a force towards the goal, and a force away from the nearest obstacle and pedestrian. All of the *Steer* simulators shared the same blending factor between goal-directed and avoidance forces, they differed in the preferred speed and radius of the agents.

Simulator	Pref Speed (m/s)	Radius (m)
Steer-1	1.44	0.20
Steer-2	1.00	0.30
Steer-3	0.85	0.20

Table 8: *Parameters used in various steering simulators.*