Pacific Graphics 2014
J. Keyser, Y. J. Kim, and P. Wonka
(Guest Editors)

*Volume 33 (2014), Number 7*

# A Data-Driven Framework for Visual Crowd Analysis

Panayiotis Charalambous[1], Ioannis Karamouzas[2], Stephen J. Guy[2] and Yiorgos Chrysanthou[1]

[1]University of Cyprus, Cyprus [2]University of Minnesota, USA

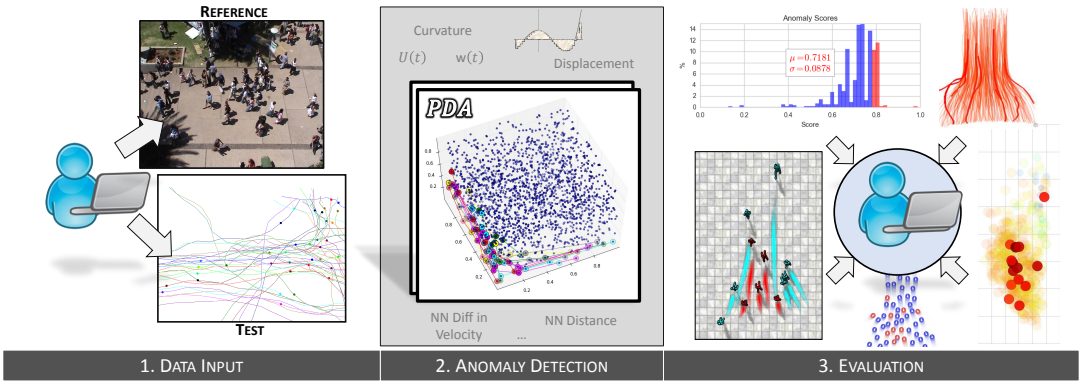



Figure 1: **System Overview** (1) Reference data from real crowds or relevant simulations are compared against testing data from simulations the user wishes to analyze. (2) Using a variety of metrics our framework automatically detects outliers using anomaly detection techniques such as Pareto Depth Analysis. (3) The resulting analysis is then shown as set of different visualizations (including heatmaps, histograms, 2D/3D animation players and a 2D outlier browsing tool) that characterize the test data compared to the reference data – here the most anomalous data are shown in red.

**Abstract**

*We present a novel approach for analyzing the quality of multi-agent crowd simulation algorithms. Our approach is data-driven, taking as input a set of user-defined metrics and reference training data, either synthetic or from video footage of real crowds. Given a simulation, we formulate the crowd analysis problem as an anomaly detection problem and exploit state-of-the-art outlier detection algorithms to address it. To that end, we introduce a new framework for the visual analysis of crowd simulations. Our framework allows us to capture potentially erroneous behaviors on a per-agent basis either by automatically detecting outliers based on individual evaluation metrics or by accounting for multiple evaluation criteria in a principled fashion using Principle Component Analysis and the notion of Pareto Optimality. We discuss optimizations necessary to allow real-time performance on large datasets and demonstrate the applicability of our framework through the analysis of simulations created by several widely-used methods, including a simulation from a commercial game.*

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

## 1. Introduction

Computer generated crowds are commonly used in films, video games, training simulators and virtual environment applications. Realistic simulation of such crowds is an important factor for the level of user immersion and the value of the conclusions one can draw from these simulations. Over the past twenty years, the field of computer graphics has experienced a dramatic increase in the number of tools, approaches and algorithms focusing on creating compelling crowd motions. Currently, several companies spe-

cializing in offline modeling and rendering offer tools to author crowd simulations [MAS] and several game engines have also recently developed modules to assist in modeling crowds [UNI]. As more approaches to virtual crowds are introduced, developing better methods to analyze, evaluate, and improve the quality of these simulations becomes increasingly necessary.

The complexity inherent in a simulation involving multiple interacting characters, makes evaluation a non-trivial, challenging problem. Some issues are easy to identify and measure such as characters colliding with each other or the environment, or characters walking way too fast. Other issues are more subtle: jamming too much at narrow passages or unnecessary backtracking. Some crowd simulation systems avoid these types of problems, however the result might still not be satisfying. A crowd in a city, where each character walks directly to its target in the most efficient way might end up looking robotic and spiritless. This is in contrast to a real-life crowd where many phenomena appear, such as people walking in groups, chatting, wandering from shop to shop and so on. We propose that the challenge of detecting these type of issues in crowd simulations can be addressed through the use of *data-driven* analysis. In our work, reference data (either synthetic or real) is used both to indicate what a correct simulated motion should look like and to automate the task of finding anomalous behaviors.

Our work complements existing research in crowd analysis by focusing on a user-in-the-loop analysis. To that end, we seek to develop a framework which provides real-time, visual feedback on potential anomalies in a crowd, accommodates large data sets, and allows users to quickly test different types of criteria to evaluate behaviors (e.g., unusual speeds or unusual densities). Crucially, we also seek to allow users to choose multiple different and often conflicting criteria *simultaneously*, for example, a user may like to detect anomalous agents whose speed is unusually high given their current density. To find these type of outliers, we will exploit recent advances in machine learning such as the Pareto Depth Analysis (PDA) approach [HXCH12] to reason over multiple criteria in a principled fashion. While, PDA can be too slow to use for real-time analysis, we propose some important modifications which allow us to perform an approximate PDA analysis in real-time.

**Contributions:** We propose a new data-driven crowd analysis framework that formulates the problem of anomalous behavior detection in crowd simulations as an outlier detection problem. Our framework allows users to compare crowd simulations to reference data across a variety of criteria. By visually highlighting behaviors that are likely erroneous in an automated fashion, our framework allows users to quickly find errors in simulations even in large, complex scenarios. We show several applications of our framework by using current state-of-the-art outlier detection techniques to analyze various recent crowd simulation methods in several different scenarios. The main advantages of our approach are:

- It enables unsupervised anomaly detection, without the need for manually labeling any reference data.
- It can identify both atypical behaviors as well as missing features from the simulation by allowing a two-way comparison between a simulation and a known dataset.
- It enables a principled application of multiple evaluation criteria by introducing a new real-time variant of the Pareto Depth Analysis outlier detection method.
- The entire system runs at interactive rates, allowing users to quickly switch between different reference datasets and different evaluation criteria to get a more complete understanding of a method's potential errors (Figure 2).

The rest of the paper is organized as follows. We highlight related work in Section 2. In Section 3, we introduce our crowd analysis framework, and in Section 4 we detail the necessary modification to the Pareto Depth Analysis method for real-time use. We demonstrate the applicability of our framework for crowd simulation and evaluation in Sections 5 and 6. Plans for further research are discussed in Section 7.

## 2. Previous Work

**Crowd Simulation.** Over the past two decades numerous models have been proposed to simulate crowds of interacting characters. The seminal work of Reynolds on flocking has been influential in this field [Rey87]. Since then, many different crowd models have been proposed, including *flow-based* models that dynamically compute vector-fields to guide the crowd motion [TCP06, NGCL09], *force-based* approaches that treat characters as particles and model their interactions using physical forces [HFV00, KHvBO09], and *geometrically-based* approaches that compute collision-free velocities for the characters using sampling or optimization techniques [GCK*09, KSHF09, OPOD10, vdBGLM11]. Most relevant to our work are *data-driven* simulation methods [LCHL07, JCP*10, CC14]. Like our framework, these approaches use example crowd data, but seek to create new simulations rather than compare against it for evaluation. We refer the reader to the survey of Pelechano et al. [PAB08] for a more comprehensive discussion on crowd simulation.

**Crowd Analysis.** The most typical way to assess the correctness of a simulation is by devising a number of meaningful evaluation metrics, such as the time required for the characters to reach their destinations or the average number of collisions. In the animation community, Reitsma and Pollard [RP07] defined task-based metrics for evaluating the quality of individual trajectories, Pelechano et al. [PSAB08] used presence as a metric to validate simulated crowd behaviors, and Singh et al. [SKFR09] proposed a number of predefined test-case scenarios along with different quantitative metrics to objectively assess the steering behaviors of virtual characters. More recently, Guy et al. [GvdBL*12] proposed

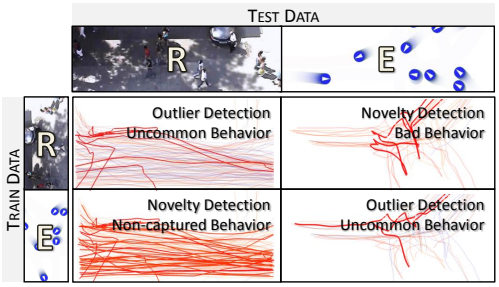



Figure 2: Different results can be produced in the proposed framework depending on which data are used as training and which as testing. In this table, R are reference data from desired behaviors and E are the data under examination.

the entropy metric to estimate how closely a given simulation state matches real-world data.

Most closely related to our work, the Steerbug framework, introduced in [KSA*09], uses a combination of metrics and predefined rules to identify behaviors of interest and atypical behaviors that may reduce the quality of a simulation. In contrast, Lerner et al. [LCSCO10] proposed a data-driven approach that determines how similar the behaviors/trajectories of simulated entities are to the ones obtained from video footage of real crowds. This approach was extended in [KO12] to evaluate the behavior of small pedestrian groups. Conceptually, our framework seeks to capture the advantages of both the user-driven analysis presented in SteerBug and the data-driven driven evaluation of the later two papers in a robust fashion.

## 3. Anomaly Detection Framework

Broadly speaking, our framework seeks to automate the process of analyzing a simulation's output by comparing it to reference data. We assume as input both a set of reference data (real or simulated) that captures typical, desired crowd motion and a set of user-selected evaluation criteria. We then use outlier detection algorithms which when trained can be used to evaluate any new set of trajectories. Both the data to be evaluated, and the reference training data consist of trajectories that track the positions of agents in time. Paths (or segments of these paths) that are found to be anomalous are highlighted for the user using heatmaps, histograms and other visualization approaches as important areas for further investigation (see Figure 1 for an overview).

There are two different modes of using our proposed approach: outlier detection and novelty detection (Figure 2). In pure *outlier detection*, the same data are used for both training and testing purposes. By using the same dataset, trajectories which are uncommon in the reference data will be detected. As we will show, this is useful for finding several types of erroneous behavior that arise from a simulation's poor handling of difficult crowd conditions or unusual local circumstances. Pure outlier detection is not sufficient for identifying all errors. For example, wide-spread and systematic problems in a simulation will not be labeled as erroneous using pure outlier detection (e.g., a simulation where agents all move too quickly).

To perform *novelty detection*, we use different data for training and testing. By using the results from a simulation as testing data and data from humans in a similar environment as training data, we can detect instances where simulated agents act inconstantly with human motion. Importantly, we can perform novelty detection with simulated agents as the training data and real trajectories as testing data. By swapping training and testing data in this way, we are able to detect behaviors performed by the real humans which are not captured by the simulation method being examined.

The choice between outlier detection (same reference and testing data) and novelty detection (different reference and testing data) presents an important tradeoff. Outlier detection requires no special data and can be applied to any simulation, however it will fail to capture systematic errors in a simulation. Novelty detection can find a wider range of erroneous behaviors, though it requires the user to find reference data with similar characteristics to the scenario being analyzed. Section 6 presents results using our framework for both outlier and novelty detection.

### 3.1. Data and Notation

Our framework takes as input two sources of data: training data which serve as a reference of "correct behavior" and testing data in which outliers will be identified. In general, we assume that the reference data provided by the user contains mostly nominal behavior. This assumption allows us to use unlabeled data, and avoids the need for a user annotating behaviors into good and bad.

Each dataset is represented as a time-varying sequence of positions, velocities and orientations for each agent in the scene. This data is typically easy to extract from a simulation. However, when working with human trajectories, it is common to be presented with only (noisy) estimates of individuals positions. In these cases, it is necessary to use filtering methods, and finite difference analysis to infer an individual's velocities and orientations.

**Trajectory Segmentation.** Typically, issues in an agent's motion are localized to a small portion of its trajectory rather than its entire path. To account for this, rather than analyzing entire trajectories, trajectories can be split into smaller segments of equal temporal length. This allows for a finer analysis of the simulations so that local abnormalities are detected and pinpointed. In order not to miss anomalies spanning a segmentation, overlapping segments are used (see Figure 3).

**Data Representation.** Each segment is represented by a collection of metrics which characterize the agent's state

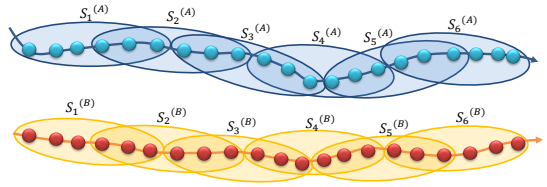



Figure 3: **Segmentation.** Trajectories of two interacting agents are split into several overlapping segments.

along that segment (e.g. average speed, maximum curvature or minimum distance to nearest neighbor). As discussed below, the metrics used to describe a trajectory will define the types of anomalous behavior being detected. Assuming a total of $l$ metrics with each one denoted as $m_j$, $j \in [1,l]$, each of the $n$ possible segments in the training data is represented as a vector:

$$\mathbf{s}_i = [m_1{}^{(i)}, m_2{}^{(i)}, \ldots, m_l{}^{(i)}]^T, i \in [1,n] \quad (1)$$

where the superscript denotes the segment and the subscript indicates the metric. The set of segment representations $\mathbf{S}_T = \{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_n\}$ of the training data represents the space of nominal behaviors and acts as a training database.

We can now formally specify anomaly detection as follows: Given a new testing segment out of a total of new $n_t$ testing data $\mathbf{s}_t = \{m_1{}^{(t)}, m_2{}^{(t)}, \ldots, m_l{}^{(t)}\}$, $t \in [1,n_t]$ ($n_t$ is typically different to $n$), we seek to measure if $\mathbf{s}_t$ lies in the space defined by the nominal data $\mathbf{S}_T$. Typically, normalization based on the combined training and testing data is applied for each of the dimensions of the data to account for scale differences in the different metrics.

### 3.2. Comparison Metrics

When training an anomaly detection method, a user must specify what metric(s) to use for the evaluation. Following the approach presented in [KSA*09], we split our metric into two parts: a *measure* and an *operator*. A measure seeks to characterize the state of an agent at a given timestep (examples include speed, acceleration or number of neighbors). An operator then aggregates the effect of a measure over many time steps (e.g., average, min, or max). Combining an operator with a measure will provide a complete metric, for example two paths may be compared based on their average speed or maximum acceleration.

To assist users in choosing metrics we broadly group our difference measures into three categories: individual, interpersonal, and group (Table 1). Individual similarity measures are based on properties of each agent in isolation. Examples include an agent's speed, acceleration, displacement, curvature, and other criteria that can be defined as a function of the agent's path over time. Interpersonal metrics seek to capture an agent's relationship to its neighboring agent. For example, the average distance to an agent's nearest neighbor may help reveal outliers in density. Finally, group metrics capture how agents behavior compares to their nearby neighbors. Examples include the difference between an agent's velocity and the velocity of all its nearby neighbors, which can capture important aspects of social interactions between individuals. A partial list of measures and operators implemented is given in Table 1 (see Appendix A for a more complete list).

| **Similarity Measures** | **Operators** |
|---|---|
| *Individual*: Speed, Acceleration, Curvature, Displacement<br>*Inter-Personal:* Nearest Neighbor Distance, Nearest Neighbor Speed Diff<br>*Group:* Number of Neighbors, Neighborhood Speed | Average, Standard Deviation, Minimum, Maximum, Sum |

Table 1: **Sample Metrics**. Combining a measure with an operator will produce a metric over a trajectory segment. A full listing of operators and measures is given in the Appendix.

### 3.3. Outlier Detection Methods

Once the simulation data has been divided into segments, reference data has been found and comparison metric(s) have been chosen, outliers can then be detected automatically using state of the art machine learning techniques. We have integrated several different anomaly detection methods within our framework including One Class SVMs, $k$-NN, and $k$-LPE each of which is described briefly below.

**One Class SVM.** One Class Support Vector Machines (SVM) [SWS*00] extend SVMs for classification problems. Here, everything in the input dataset is considered to be nominal and an "optimal" hyperplane is calculated to divide the space into two regions; nominal and anomalous. SVM-based approaches provide a binary classification of each new datapoint as nominal or anomalous.

**$k$-NN Based Approaches.** The simplest approach to anomaly detection is comparing the testing sample $\mathbf{s}_t$ to its closest neighbors in the training dataset $\mathbf{S}_T$. An anomaly score for $\mathbf{s}_t$ is defined as a function of the distances to the $k$ nearest neighbors with values above a threshold indicating an anomalous sample. This threshold value can either be user defined or estimated from the nominal data. The choice of $k$ plays a significant role on the results; a small value examines a very small subset of the data whereas a very large number could lead to over-training. We implemented an approach based on Hsao et al. [HXCH12] who suggest that a good balance can be found by iterating over $k$ until the graph of $k$ nearest neighbors ($k$-NN graph) is fully connected (or within a user defined threshold in case of clustered data).

**$k$-LPE.** Localized p-value Estimation ($k$-LPE) is a recently proposed method [ZS09] for anomaly detection based on k-

Nearest Neighbour Graphs ($k$NNG). Each point $\mathbf{s}_i$ on the graph (of the $n$ training segments $\mathbf{S}_T$), is assigned a score value $R_S(\mathbf{s}_i) \in [0,1]$ based on the distances to their $k$-nearest neighbors. Assuming the $k$-nearest neighbors of $\mathbf{s}_i$ is the ordered list of points $kNN(\mathbf{s}_i) = \{\mathbf{s}_{i,1}, \mathbf{s}_{i,2}, \ldots, \mathbf{s}_{i,k}\}$, the score for each point is then typically defined as the distance to the farthest nearest neighbor $\mathbf{s}_{i,k}$:

$$R_S(\mathbf{s}_i) = d(\mathbf{s}_i, \mathbf{s}_{i,k}) = \sqrt{\sum_{j=1}^{l} (m_j{}^{(i)} - m_j{}^{(k)})^2} \qquad (2)$$

Given a new test point $\mathbf{s}_t$, an anomaly score $p_K(\mathbf{s}_t)$ is estimated based on the following formula of Zhao et al. [ZS09]:

$$p_K(\mathbf{s}_t) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}_{\{R_S(\mathbf{s}_t) \leq R_S(\mathbf{s}_i)\}} \qquad (3)$$

where $\mathbb{I}_{\{.\}}$ is the indicator function, i.e., it returns 1 if the condition is valid and 0 otherwise. Intuitively, Equation (3) can be thought of as indicating what portion of existing points on the $k$NNG of the training data have worse score than the testing point. Or more simply, how much of the training data is more anomalous than a given testing sample. A pre-defined significance level $\alpha$ (e.g., $\alpha = .05$) controls the anomaly detection – a point is considered anomalous if $p_K(\mathbf{s}_t) \leq \alpha$. Importantly, $\alpha$ is exactly the false alarm rate providing an easy to interpret threshold value.

### 3.4. Analysis

**Visualization.** All three outlier detection methods can provide comparable outlier results with carefully tuned threshold selections. However, we recommend using $k$-LPE over SVM or $k$-NN, partly, because of the guarantee on false alarm rates. Moreover, each trajectory segment will be assigned a continuous value between 0 and 1, corresponding to how anomalous it is. This allows the use of $k$-LPE scores in visualization by mapping the scores to a blue-to-red color ramp (see Section 5.1 for further discussion).

**Multiple Criteria.** When using multiple metrics, the distance formula of Equation (2) assumes each metric is weighted equally. In many cases this is undesirable, as agents which are very outlying in one metric but normal in another metric are still outliers. Consider, for example, an agent who takes a very curvy, circuitous path, but walks at a nominal speed; weighting the metrics evenly will mask the fact that the path is an extreme outlier. A popular solution to this issue is known as scalarization, which seeks to find a convex combination of metrics which "best" predict anomalies. Finding these weights is typically a very slow procedure which is exponential to the number of criteria (e.g., if grid search is used), and requires manually labeled data for training purposes. Because these options are unsatisfying, in the next section we will explore methods to detect outliers which account for multiple criteria in a principled fashion.

## 4. Real-Time Multiple-Criteria Evaluation

Often, multiple comparison metrics may be needed to capture a wider range of atypical behaviors. However, most anomaly detection approaches do not naturally handle multi criteria. As discussed above, the typical solution of taking a linear combination of multiple similarity measures (i.e., weighted average of the metrics) has important limitations. More importantly though, there exists outliers which *no linear combination of metrics can detect*! Consider, for example, the metrics of speed and neighbor distance. Assuming that training data comes from an outdoor sidewalk, a simulated agent which moves with near zero speed will not seem anomalous because real people frequently stop to talk to friends or slow down to avoid congestion. Likewise, a simulated agent who has no nearby neighbors is unlikely to be anomalous as people often walk alone. However, an agent who is simultaneously still and not near anyone is anomalous; outside of talking to someone or resolving a collision, it is unusual to stop walking in the middle of a sidewalk. Finding these types of outliers requires a method which captures the dependence of one metric on another.

The concept of Pareto optimality provides a principled way of accounting for multiple criteria simultaneously. In this section, we briefly recap the state-of-the-art in Pareto optimality analysis and discuss modifications necessary to allow the fast data analysis necessary to support a user-in-the-loop workflow.

### 4.1. Pareto Optimality and Dyads

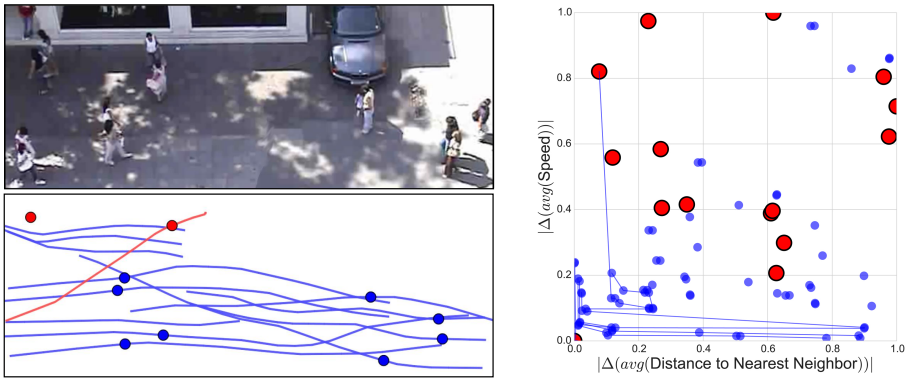
Pareto Optimality is the typical approach for defining optimality in a problem with multiple conflicting criteria. An item is considered Pareto optimal if there is no other item that is better or equal in all the defined criteria. The *Pareto Depth Analysis (PDA)* [HXCH12] method exploits the concept of Pareto Optimality as follows:

First, relationships between all the training segments are encoded using *dyads*, a vector representation of differences for each metric. More formally, a dyad $\mathbf{D}_{i,j} \in \mathbb{R}^l$ between segments $\mathbf{s}_i$ and $\mathbf{s}_j$ is defined as:

$$\mathbf{D}_{i,j} = [d_1(i,j), d_2(i,j), \ldots, d_l(i,j)]^T \qquad (4)$$

where $l$ is the number of metrics and $d_m(i,j)$, $m \in [1,l]$, indicates the difference between segments i and j for metric $m$. Here, $d_m(i,j)$ is typically absolute differences between the segments for criteria $m$.

The set of all possible dyads between training segments are calculated and stored; if there are $N$ segments in a dataset, a total of $\binom{N}{2}$ dyads are calculated – dyads between a segment and itself are not calculated. The set of all dyads $\mathbb{D}$ encodes the differences between each possible pair of training segments. For example, Figure 4b shows dyads plotted for the real-world pedestrian dataset in Figure 4a computed using two metrics; average distance to nearest




(a) (b)

Figure 4: (a) The *Zara* dataset (top) where two apparent outliers with usual behavior have been highlighted in red (bottom). (b) The dyads from these pedestrians are shown based on the criteria of average speed and nearest neighbor distance. The first 10 Pareto fronts are shown as blue lines. The dyads from the apparent outlying pedestrians all lie in deep Pareto fronts (top right of the plot) so the agents would be identified as outliers.

neighbor and average speed ($d_1(i,j) = |\Delta(E(nn_{dist}))|$ and $d_2(i,j) = |\Delta(E(speed))|$ respectively).

After computing dyads, Pareto fronts are found. These are sets of dyads which are Pareto-optimal (i.e., any dyad on the front which is better in one metric is worse in some other metric). These dyads are then removed and the next most Pareto-optimal dyads are taken to form the next Pareto front. This process continues iteratively until every dyad lies on a Pareto front. For each one of these fronts, a rank is given based on the order they are found; i.e., the first Pareto front has rank 1, the second 2 and so on. The lines in Figure 4b indicate the first 10 Pareto fronts for the aforementioned example. Dyads on low rank fronts correspond to trajectory segments which are very similar to other segments, and are considered nominal. Dyads on high rank fronts indicate that the segments are very different in one measure with respect to the other measure (e.g., an unusual path curvature given the speed), and these paths are considered outliers.

To illustrate the concept, in Figure 4a(bottom) we manually highlight in red two agents with visually apparent anomalous behavior. As can be seen in the supplemental video, these agents stand still or watch the building without talking to others or moving for an unusually long time. The dyads corresponding to these apparent outliers are shown in Figure 4b as large red circles. All of these dyads have a high Pareto front depth, and the trajectories are therefore considered outliers by the PDA method.

In general, to evaluate a new segment as a potential outlier, the $k$ closest matching training samples in each of the metrics separately are selected ($k$ can be different for each metric). Dyads between the test sample and these $k$ neighboring samples are calculated and the first Pareto Front that dominates every one of these dyads is found. The average Pareto front depth of all the dyads (normalized from 0 to 1) serves as an evaluation of how anomalous the segment is, and can also be used for coloring heatmaps or determining an anomaly threshold. Typically, values for $k$ are calculated using the $k$NNG approach described in Section 3.3 instead of being manually defined.

### 4.2. Interactive PDA

As described in [HXCH12] the PDA method is too slow to be used in interactive analysis. The dyads and Pareto fronts must be recomputed each time a new set of metrics is chosen which can be a time consuming process preventing quick user interactions. We introduce two modifications to adapt the PDA computation for interactive analysis: randomized PDA and criteria dimensionality reduction.

**Randomized PDA.** To reduce computation cost, we construct Pareto fronts over a randomly chosen subset of the dyads. Crucially, the sampling process is performed on the dyads and not the training samples themselves. This process ensures that data from all the samples are used to construct the Pareto fronts. Our proposed method is especially effective for very large datasets with many dyads, as is the typical case when dividing large datasets into many segments. For the large pedestrian dataset, with 204 trajectories (Table 2) and 4 evaluation criteria, even a large reduction in sampled dyads (up to 90%) had a small affect on the root mean squared error (normalized error of $< 5\%$). See Appendix B for further analysis.

**Criteria Dimensionality Reduction.** When using a long simulation as both training and testing data (i.e., pure outlier detection), the datasets being tested can become very large, greatly increasing the computation time. This problem can be greatly exacerbated if a user wants to find outliers along

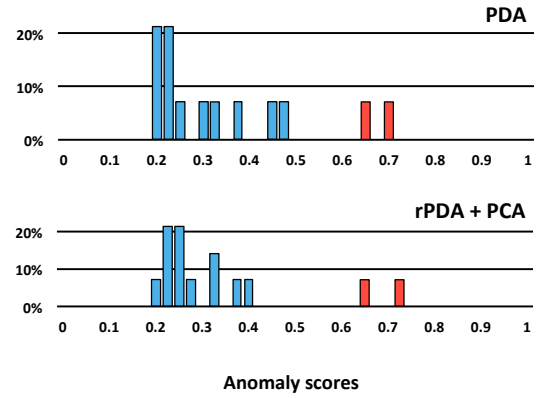



Figure 5: **Distributions of anomaly scores.** Using PCA to reduce the dimensionality from 10 to 4 criteria, finds similar outliers while speeding up computation. The red bars indicate the two idle agents in Figure 4a.

any of the possible evaluation criteria and so chooses dozens of different metrics to analyze simultaneously; the process of finding the appropriate Pareto front for each testing segment is linear to the number of metrics. In these cases, even randomized PDA can take several minutes to evaluate paths. We propose using dimensionality reduction techniques such as Principle Component Analysis (PCA) to solve these issues. Because the testing and training data will be the same, PCA will find a reduced space which represents well both the variation in the training data and the errors in testing data. For novelty detection tasks, this technique may not be safe if the testing and training data vary significantly.

We can illustrate the effect of reducing dimensions in PDA analysis by comparing the outliers found in Figure 4 using 10 different individual and group metrics to those found using the top 4 PCA components. In this way, we can automatically derive new representative criteria without needing any prior knowledge of the scenario being tested. The results for this scenario are shown in Figure 5. The two individuals with anomalous trajectories previously identified are correctly labeled as outliers without any further manual intervention. Overall, randomization along with PCA-based dimensionality reduction lead to over an order of magnitude reduction in computation cost, while calculating similar scores and more importantly similar outliers. Section 5.3 gives a detailed performance comparison, while additional quantitative results are presented in Appendix B.

Pseudocode for this PDA-based, multi-criteria outlier detection technique, including our modifications, is given in Algorithm 1. The algorithm has two phases, training and testing, which can be performed separately. It takes as input the number of metrics $l$, the requested dimensionality $d \leq l$ (which can be given as a variance percentage), the percentage of dyads to keep $0 < p \leq 1$ and a threshold value $\sigma$ for anomaly detection. $\mathbf{B}$ are the eigenvectors of the PCA space.

**Algorithm 1** Randomized PDA on PCA reduced data

**Training**
$\mathbb{S}_{Tr} \leftarrow$ TrainingSegments()
$\mathbb{S}'_{Tr}, \mathbf{B} \leftarrow PCA(\mathbb{S}_{Tr}, d)$
$\mathbb{D} \leftarrow$ GenerateDyads($\mathbb{S}'_{Tr}$)
$\mathbb{D}_r \leftarrow$ RandomSelection($\mathbb{D}, p$)
$\mathbb{PF} \leftarrow$ GenerateParetoFronts($\mathbb{D}_r$)

**Testing**
$\mathbb{S}_{Te} \leftarrow$ TestingSegments()
$\mathbb{S}'_{Te} \leftarrow \mathbf{B}^T \mathbb{S}_{Te}$
**for** $s_t \in \mathbb{S}'_{Te}$ **do**
  $n_b \leftarrow []$ {Neighbors for each criteria}
  **for** $c = 1 \rightarrow d$ **do**
    $nb_c \leftarrow k_c$ nearest neighbors of $s_t \in \mathbb{S}'_{Tr}$
  **end for**
  Create new dyads $\mathbb{D}_n$ between $s_t$ and samples in $nb$
  **for** $\mathbf{D_i} \in \mathbb{D}_n$ **do**
    $e_i \leftarrow Depth(\mathbb{D}_n)$ {Pareto depth of neighbor i}
  **end for**
  $score(s_t) \leftarrow \frac{1}{s} \sum_{i=1}^{s} e_i$
  $s_t$ is anomalous iff $score(s_t) \geq \sigma$
**end for**

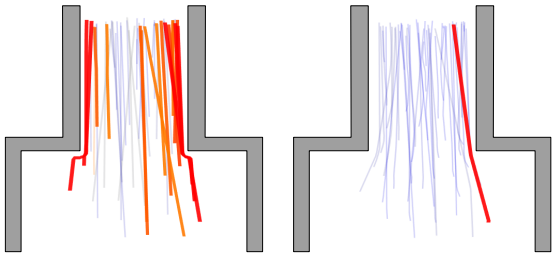
## 5. Framework Usage

We implemented our proposed crowd analysis framework in Python in order to evaluate its applicability in various scenarios. The following reference data was used (after applying a Butterworth filter to remove noise and hip sways):

**Bottleneck:** Group of automatically tracked participants navigating through a 2.5 m wide bottleneck [SPS*09].
**Zara:** Sparse crowd of manually tracked people interacting at a commercial street [LCL07] (Figure 4a).
**Assassin's Creed:** Non-player characters (NPCs) were manually tracked from in game footage [Ubi09] (Figure 7).

### 5.1. Visual Evaluation

Our Python front-end allows the user to analyze crowd simulations and detect erroneous behaviors. After indicating the training/testing data and comparison metrics, several different visualizations are given to the user. By coloring data based on the anomaly score of a trajectory segment on a continuous scale ranging from blue (nominal behavior) to red (highly anomalous behavior), histograms can indicate the distribution of outliers (Figure 5), heatmaps can show where these behaviors happen (Figure 6), and 2D and 3D realtime animations can highlight erroneous behavior over time by changing the agents' colors. We can also display each trajectory segment as a color-coded point in 2D space (using dimensionality reduction like PCA) to allow users to quickly browse through trajectories/segments grouped by the similarity of their metrics. Several example visualizations are given in Figure 1 and Appendix C.



(a) $\tau = 1\,s$ (b) $\tau = 10\,s$

Figure 6: **Parameter Tuning.** Adjusting the time horizon, $\tau$, over which agents can anticipate in ORCA changes the number of outliers. Here, with a small time horizon ($\tau = 1\,s$) agents react to obstacles too late, and have unrealistically straight paths. Larger time horizons results in more reactive agents that match the data better (less outliers).

### 5.2. Improving Crowd Simulations

Our framework can improve existing crowd simulation systems by assisting users in tuning simulation parameters. By using our front-end, different simulation outputs from different parameters can be quickly analyzed for outliers with respect to different evaluation metrics. This interactive use allows us to quickly minimize the number of anomalous trajectories/segments. In addition, such a user-in-the-loop parameter exploration can be combined with automated global optimization techniques for parameter tuning [WGO*14].

Figure 6 gives an example of our framework being coupled with parameter optimization to improve a crowd simulation. Here, we chose curvature and distance to nearest neighbor as evaluation metrics and used the *Bottleneck* dataset to evaluate trajectories simulated with ORCA [vdBGLM11]. We then modified the *anticipation time horizon* of the agents, while keeping the rest of the simulation parameters fixed. The time horizon parameter indicates how far in advance agents react to collisions. With too small of a time horizon, agents either failed to react to close neighbors (paths near others were too straight) or reacted too late to walls (paths near obstacles were too curved). Our visualization quickly highlights both types of outliers (Figure 6a). Using a larger time horizon alleviates both problems and results in fewer outliers (Figure 6b).

### 5.3. Performance

Given the interactive nature of our crowd analysis framework, quick responsiveness to a user's requests is a very important feature. Overall, single-criteria anomaly detection can be performed very fast using $k$-LPE. When using PDA for multiple criteria, our system can run at interactive rates through the introduction of the randomized PDA, and the usage of PCA for dimensionality reduction. Table 2 highlights the runtime performance of our system running pure outlier detection on three datasets using all *group* similarity metrics.

| | | Total Time (s) | | |
|---|---|---|---|---|
| Dataset | # Samples | PDA | rPDA | rPDA + PCA |
| Assassin's | 35 | 2.4 | 0.9 | 0.2 |
| Bottleneck | 176 | 353.9 | 78.7 | 8.3 |
| Zara | 204 | 698.3 | 151.6 | 11.2 |

Table 2: **Performance.** Comparison between PDA, randomized PDA (rPDA), and rPDA with PCA (2 components) using 6 metrics. The randomized methods use 40% of the input dyads. Overall, using both rPDA and PCA leads to a significant performance gain without compromising quality.

Overall, the running time scales almost linearly with the number of constructed pareto fronts. As can be seen from the Table, randomized dyad selection results in about a 4.5x speedup, and incorporating PCA brings the total runtime of Algorithm 1 to be more than an order of magnitude faster than PDA alone. All results were obtained on an Intel 3.4 GHz Core i5 processor (on a single thread).

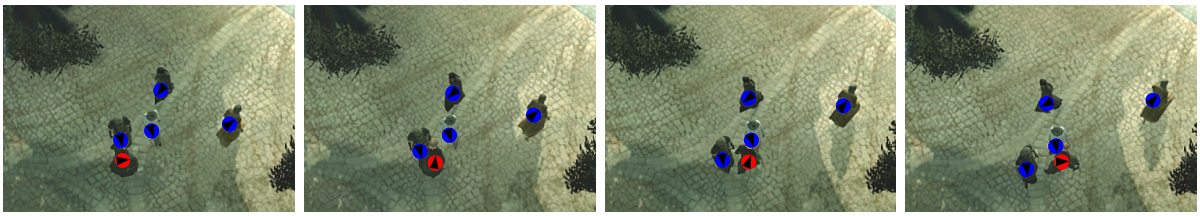
## 6. Simulation Analysis

We applied our tool to all three datasets described in Section 5 to demonstrate the variety of different forms of analysis which can be performed in the proposed framework.

### 6.1. Single Criteria Analysis

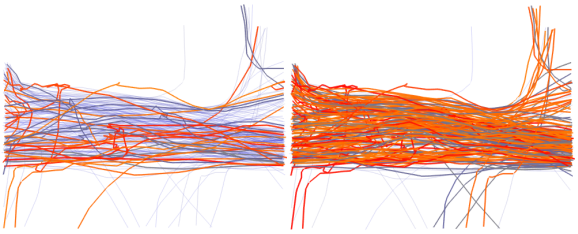
We used the *Bottleneck* as reference data to analyze the quality of three crowd simulation algorithms: the social force model proposed in [HFV00], the velocity-based model presented in [POO*09], and an anticipatory model tuned to closely match the input training data following an approach similar to [WGO*14]. For each method, we created a 50 agent simulation matching closely the conditions of the reference data.

For each simulation we computed the corresponding *entropy metric* scores [GvdBL*12]. As can be seen in the accompanying video, the social force model has the lowest performance, as the agents do not match well the behavior of the real humans. However, both the velocity-based simulation and the tuned one have similar overall flows to the real data, and receive similar entropy scores. Our framework, compliments the entropy score by allowing users to further investigate individual behaviors (rather than aggregate simulation results). Here, we can detect agents that exhibit various erroneous behaviors by some criteria even though the overall flow matches well. The tuned model, for example, has less nearest neighbor outliers than the velocity-based model ($k$-LPE on segments with $\alpha = .05$). Further examples are given in the accompanying video.



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

Figure 7: **Assassin's Creed.** Example outlier detected in a non-playing character in the game Assassin's Creed (red circle). The outlying character can be seen to spin around quickly as it tries to avoid the upcoming collision too late. This outlier was found by applying PDA on path segments after performing principal component analysis on 14 evaluation criteria.



(a) [KO12] model (b) ORCA model

Figure 8: **Missing Social Features.** Simulations that account for small groups show different outliers than those without. The corresponding heatmaps were obtained by applying randomized PDA on speed and NN distance (threshold = 0.4).

### 6.2. Multi-criteria Analysis

**Multi-criteria Novelty Detection.** In this experiment, we used the ORCA framework [vdBGLM11] to simulate 60 agents wandering on a 2D plane. To detect behaviors that are missing from the simulated agents, as compared to real pedestrians, we used the simulated trajectories as training data and the trajectories of the *Zara* dataset as testing. By performing anomaly detection on single criteria independently, a number of missing behaviors can be detected. Using, for example, average speed as an evaluation metric, both those that stand still to chat or run very fast are flagged as outliers. In addition, using the average distance to the nearest neighbor as an evaluation metric, all pedestrians that walk in groups are detected as outliers, since the ORCA model does not account for such groups. However, only the combination of both evaluation criteria using our proposed randomized PDA implementation allowed us to consistently capture both missing behaviors. To validate the accuracy of the randomized PDA, we replaced the ORCA simulation with a simulation that accounts for the local behavior of small pedestrian groups [KO12]. As expected, only pedestrian that have anomalous speed are detected as outliers (see Figure 8).

**Multi-criteria Outlier Detection.** In this experiment, we used the *Assassin's Creed* dataset both for training and testing purposes in order to perform multi-criteria outlier detection and analyze the quality of the manually tracked NPCs. The NPCs global motions are governed by waypoints along the medial axis of the environment, whereas a reactive technique is used to resolve local collisions between them. As such, the characters typically have to be very close to react to each other, which can result in undesired behaviors, such as backward motions and unwarranted oscillations (see outliers in Figure 7). Both the average and standard deviation of all individual measures were used in the analysis. Using PCA to reduce the 14 evaluation metrics down to 2 dimensions resulted in a 14x speedup without compromising the quality of the detected outliers (see Appendix B for details).

## 7. Limitations and Future Work

We have introduced a novel framework for visual crowd analysis. By formulating the analysis problem as an outlier or novelty detection problem, our framework can automatically detect potential erroneous behaviors in a simulation given a collection of arbitrary, user-selection evaluation metrics. We have also shown how to use randomization and dimensionality reduction in conjunction with Pareto depth analysis in order to robustly account for multiple, simultaneous evaluation metrics while still providing good responsiveness necessary for user-in-the-loop iterations.

Our proposed framework has some limitations. For one, PDA-based dimensionality reduction can only be reliably applied when testing and training data are very similar. Additionally, performing novelty detection relies on having external representative data, which may not always be available. We believe that both of these issues can be addressed to some extent by running extensive user studies of many different simulations to develop strong priors on visually acceptable motion even in the absence of training data.

Beyond addressing these limitations, we would like to improve the general speed and robustness of our implementation. For example, enlarging the input training dataset by blending between samples following an approach similar to

Ju et al. [JCP*10] can reduce the number false-positive outliers. Additionally, some aspects of PDA computation can be efficiently parallelized suggesting that a GPU implementation may be effective. Such accelerations may allow interactive analysis of very large crowds, consisting of thousands of agents (see, e.g., [NGCL09]) which are important in some applications.

## Acknowledgments

We would like to thank Kevin S. Xu for helpful discussions on the Pareto Depth Analysis method. This work was partially funded by the Cyprus Research Promotion Foundation and the European Structural Funds for the "VR CAVE" project under contract IPE/NEKYP/0311/02.

## References

[CC14] CHARALAMBOUS P., CHRYSANTHOU Y.: The PAG crowd: A Graph Based Approach for Efficient Data-Driven Crowd Simulation. *Computer Graphics Forum* (2014). 2

[GCK*09] GUY S. J., CHHUGANI J., KIM C., SATISH N., LIN M., MANOCHA D., DUBEY P.: Clearpath: highly parallel collision avoidance for multi-agent simulation. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2009), pp. 177–187. 2

[GvdBL*12] GUY S. J., VAN DEN BERG J., LIU W., LAU R., LIN M. C., MANOCHA D.: A statistical similarity measure for aggregate crowd dynamics. *ACM Transactions on Graphics 31*, 6 (2012), 190. 2, 8

[HFV00] HELBING D., FARKAS I., VICSEK T.: Simulating dynamical features of escape panic. *Nature 407*, 6803 (2000), 487–490. 2, 8

[HXCH12] HSIAO K.-J., XU K., CALDER J., HERO A.: Multi-criteria anomaly detection using pareto depth analysis. In *Advances in Neural Information Processing Systems 25* (2012), pp. 854–862. 2, 4, 5, 6

[JCP*10] JU E., CHOI M., PARK M., LEE J., LEE K., TAKAHASHI S.: Morphable crowds. *ACM Transactions on Graphics 29* (2010), 140:1–140:10. 2, 10

[KHvBO09] KARAMOUZAS I., HEIL P., VAN BEEK P., OVERMARS M.: A predictive collision avoidance model for pedestrian simulation. In *Motion in Games* (2009), vol. 5884 of *LNCS*, Springer, pp. 41–52. 2

[KO12] KARAMOUZAS I., OVERMARS M.: Simulating and evaluating the local behavior of small pedestrian groups. *IEEE Transactions on Visualization and Computer Graphics 18*, 3 (2012), 394–406. 3, 9

[KSA*09] KAPADIA M., SINGH S., ALLEN B., REINMAN G., FALOUTSOS P.: Steerbug: an interactive framework for specifying and detecting steering behaviors. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2009), pp. 209–216. 3, 4

[KSHF09] KAPADIA M., SINGH S., HEWLETT W., FALOUTSOS P.: Egocentric affordance fields in pedestrian steering. In *Symposium on Interactive 3D Graphics and Games* (2009), pp. 215–223. 2

[LCHL07] LEE K., CHOI M., HONG Q., LEE J.: Group behavior from video: a data-driven approach to crowd simulation. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2007), pp. 109–118. 2

[LCL07] LERNER A., CHRYSANTHOU Y., LISCHINSKI D.: Crowds by example. *Computer Graphics Forum 26* (2007), 655–664. 7

[LCSCO10] LERNER A., CHRYSANTHOU Y., SHAMIR A., COHEN-OR D.: Context-dependent crowd evaluation. *Computer Graphics Forum 29*, 7 (2010), 2197 – 2206. 3

[MAS] Massive Software - Simulating Life. http://www.massivesoftware.com/. 2

[NGCL09] NARAIN R., GOLAS A., CURTIS S., LIN M.: Aggregate dynamics for dense crowd simulation. *ACM Transactions on Graphics 28* (2009), 1–8. 2, 10

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

[PAB08] PELECHANO N., ALLBECK J., BADLER N.: Virtual crowds: Methods, simulation, and control. *Synthesis Lectures on Computer Graphics and Animation 3*, 1 (2008), 1–176. 2

[POO*09] PETTRÉ J., ONDREJ J., OLIVIER A.-H., CRÉTUAL A., DONIKIAN S.: Experiment-based modeling, simulation and validation of interactions between virtual walkers. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2009), pp. 189–198. 8

[PSAB08] PELECHANO N., STOCKER C., ALLBECK J., BADLER N.: Being a part of the crowd: towards validating vr crowds using presence. In *Autonomous Agents and Multiagent Systems* (2008), pp. 136–142. 2

[Rey87] REYNOLDS C. W.: Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics 21*, 4 (1987), 24–34. 2

[RP07] REITSMA P. S., POLLARD N. S.: Evaluating motion graphs for character animation. *ACM Transactions on Graphics 26*, 4 (2007), 18. 2

[SKFR09] SINGH S., KAPADIA M., FALOUTSOS P., REINMAN G.: Steerbench: a benchmark suite for evaluating steering behaviors. *Computer Animation and Virtual Worlds 20*, 5-6 (2009), 533–548. 2

[SPS*09] SEYFRIED A., PASSON O., STEFFEN B., BOLTES M., RUPPRECHT T., KLINGSCH W.: New insights into pedestrian flow through bottlenecks. *Transpprtation Science 43*, 3 (2009), 395–406. 7

[SWS*00] SCHÖLKOPF B., WILLIAMSON R. C., SMOLA A. J., SHAWE-TAYLOR J., PLATT J.: Support vector method for novelty detection. *Advances in neural information processing systems 12*, 4 (2000), 582–588. 4

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

[Ubi09] UBISOFT MONTREAL: Assassin's Creed II, 2009. 7

[UNI] Unity Game Engine. http://unity3d.com/unity. 2

[vdBGLM11] VAN DEN BERG J., GUY S. J., LIN M., MANOCHA D.: Reciprocal n-body collision avoidance. In *Robotics Research: The 14th International Symposium ISRR* (2011), vol. 70 of *Springer Tracts in Advanced Robotics*, Springer, pp. 3–19. 2, 8, 9

[WGO*14] WOLINSKI D., GUY S., OLIVIER A.-H., LIN M., MANOCHA D., PETTRÉ J.: Parameter estimation and comparative evaluation of crowd simulations. *Computer Graphics Forum 33*, 2 (2014), 303–312. 8

[ZS09] ZHAO M., SALIGRAMA V.: Anomaly detection with score functions based on nearest neighbor graphs. In *Advances in Neural Information Processing Systems* (2009). 4, 5