Morphable Crowds

Eunjung Ju[1] Myung Geol Choi[1] Minji Park[1] Jehee Lee[1] Kang Hoon Lee[2] Shigeo Takahashi[3]
Seoul National University[1] Kwangwoon University[2] The University of Tokyo[3]

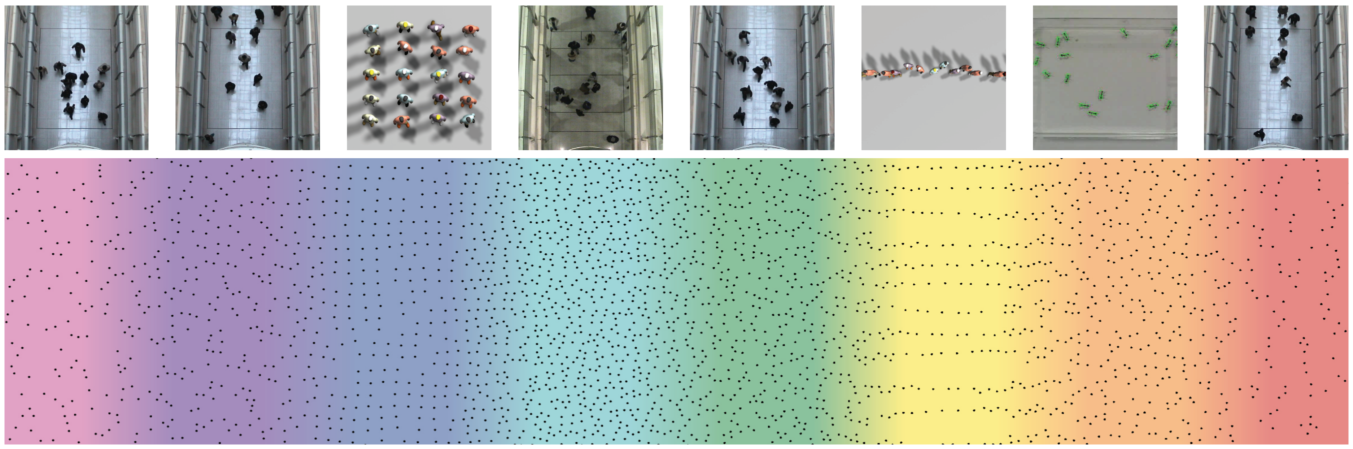

Figure 1: Morphable crowd models synthesize virtual crowds of any size and any length from input crowd data. The synthesized crowds can be interpolated to produce a continuous span of intervening crowd styles.

Abstract

Crowd simulation has been an important research field due to its diverse range of applications that include film production, military simulation, and urban planning. A challenging problem is to provide simple yet effective control over captured and simulated crowds to synthesize intended group motions. We present a new method that blends existing crowd data to generate a new crowd animation. The new animation can include an arbitrary number of agents, extends for an arbitrary duration, and yields a natural-looking mixture of the input crowd data. The main benefit of this approach is to create new spatio-temporal crowd behavior in an intuitive and predictable manner. It is accomplished by introducing a morphable crowd model that allows us to encode the formations and individual trajectories in crowd data. Then, its original spatio-temporal behavior can be reconstructed and interpolated at an arbitrary scale using our morphable model.

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

Keywords: Computer Graphics, Computer Animation, Crowd Simulation, Data-Driven Animation, Human Motion

1 Introduction

Crowd simulation has intensively been studied as an important research theme in computer graphics. Various approaches are available to describe realistic crowd scenes in detail and thus they have been frequently employed in film production, military simulation, and urban planning. For instance, the rule-based model is effective for generating global crowd motions such as flocking. Each character in the model, called agent, is equipped with a set of rules to drive autonomous behaviors reacting to its perceived environmental stimuli. Dynamically evolving flow fields also allow the systematic guidance of agents based on appropriate mathematical models. The realistic behavior of a human crowd can also be recorded and reproduced via data-driven approaches.

A crowd of pedestrians exhibit a particular "style" depending on where/when the crowd is observed and the characteristics of individuals. For example, marching army soldiers would exhibit a regular grid pattern moving in a steady speed, while pedestrians rushing to the train in a crowded subway station would manifest irregular patterns rapidly changing over time. The style of a crowd is somewhat ambiguous, but might be characterized by various factors, such as the density of a crowd, the regularity/persistency of formations, the distribution of individual velocity profiles, the style of individual locomotion, the reaction of individuals to potential collision, and so on.

Synthesizing a particular crowd style or having a control over crowd styles is a challenging problem. Existing crowd simulation algorithms allow us to tune parameters to change some aspects of crowd simulation. However, the scope of style control is inherently restricted by the parameters of governing rules or equations. Data-driven approaches alleviate this problem by enabling virtual crowds to imitate a crowd of real creatures. The example data are often acquired from video that records real crowd behavior. However, finding/chreographing a specific crowd style is cumbersome and changing the style of a recorded crowd is yet another challenge.

One potential approach is a *morphing* technique, which enables us to generate intermediate representations from multiple data. Morphing/interpolation techniques have received great attention as they can generate a continuous span of convincing intermediate representations in many computer graphics applications, including shape design, image processing, and character motion synthesis. Nonetheless, blending multiple data in our context is still a big challenge because we cannot easily establish meaningful correspondence between crowd animations, which are often significantly different both in the spatial arrangements and locomotion styles of individual agents. Even if we could match and align spatial distributions of multiple crowds in some ways, simply interpolating the associated agents' positions at each time step does not maintain the crowd styles inherent in the source data, and thus could result in unexpected spatio-temporal crowd behavior.

We present a novel method of blending crowd animations into a new crowd that exhibits the intervening formations and motion styles at an arbitrary scale. Our blending process begins by introducing a *morphable crowd model* that encodes the distribution of neighborhood formations and locomotion trajectories. A new crowd is then synthesized to match the distribution. As a result, our simulation algorithm can generate an arbitrary number of agents that consistently preserve their local group behavior over an arbitrarily long duration of time.

2 Related Work

Crowd simulation has a long history in computer graphics. Numerous approaches have since investigated achieving high complexity and realism in virtual crowds. These include locally reactive decision making [Guy et al. 2009; Pelechano et al. 2007], guiding potential fields [Treuille et al. 2006], a hybrid of agent-based and continuum approaches [Narain et al. 2009], hierarchical clustering approach for real-time simulation [Sud et al. 2007b], secondary action [Lerner et al. 2009], human perception [McDonnell et al. 2009], navigating using adaptive roadmaps [Sud et al. 2007a], and quality evaluation [Mubbasir Kapadia 2009] to name a few recent work. We refer readers to [Thalmann et al. 2007] for a comprehensive survey on the state of the art in this important research topic.

On the other hand, the data-driven approach to crowd simulation has recently emerged as a new way of reproducing complex crowd behaviors mimicking real crowds. The associated data are usually acquired by recording a bird's eye view of crowds with a camcorder, and then identifying the two-dimensional trajectories of moving subjects, either manually or semi-automatically. Lee et al. [2007] trained a group behavior model from such crowd data, which made each virtual agent react to its perceived state by referring to the captured behavior in similar situations. Lerner et al. [2007] addressed the same problem taking a slightly different approach to motion estimation by defining their matching function. Lai et al. [2005] spliced group motion clips to synthesize a new group motion. Several other researchers employed crowd motion data as a means to calibrate and validate their own group behavior models [Pettre et al. 2009]. Instead of relying on the moving trajectories, Courty et al. [2007] extracted a time series of velocity fields from crowd videos, and then synthesized a new crowd animation by advecting virtual agents along the concatenated sequence of such velocity fields. Peters et al. [2009] analyzed a size of groups, a shape of align and a splitting/merging behavior from a crowd scene or a video to demonstrate clustered pedestrians. Recently Lerner et al. [2010] proposed crowd evaluation approach based on real crowd data and it showed similarity between simulated and real crowd quantitatively.

The data acquisition phase is a common bottleneck across these data-driven methods, especially when the intended crowd scene requires a large diversity of crowd phenomena. The latest work of pedestrian tracking in computer vision is promising to alleviate this laborious process [Ali and Shah 2008; Andriluka et al. 2008]. Another interesting direction is to increase the reusability of crowd motion data once acquired. Kwon et al. [Kwon et al. 2008] presented an approach to deforming and stitching group motions while maximally retaining the neighborhood formations and moving trajectories in the original data. Kim et al. [Kim et al. 2009] extended the idea further to build an interactive multi-character motion editing system that copes with heterogeneous characters, a larger repertoire of actions, interpersonal interaction and temporal synchronization.

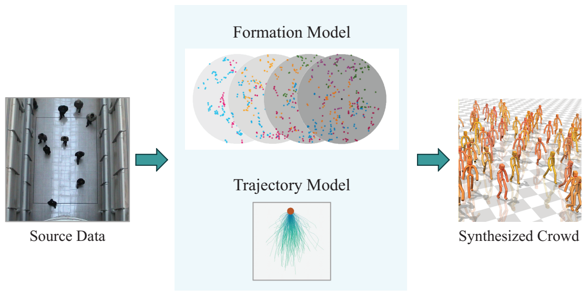

Figure 2: A morphable crowd model.

Blending multiple motion clips increases the reusability of human motion database. It allows a moderate number of clips to construct a continuous space parameterized by controllable features such as moving direction, target location, and emotional status [Rose et al. 1998; Kovar and Gleicher 2004]. Especially for a single character motion, simple linear interpolation can yield a plausible mixture of existing motions since we can easily establish the spatio-temporal correspondences between example motions. However, it is challenging to identify such correspondences between multiple sources of crowd data, because they usually have different spatial and temporal features, including the number of individuals, persistency of formation, regularity of spatial distribution, and velocity/acceleration profiles. There have been crowd blending approaches on directing and controlling based on navigation fields [Peters and Ennis 2009; Pettré et al. 2008]. These approaches focus on controlling and blending simulation fields and steering of crowd based on the fields. Patil et al. [2010] blended multiple input sources such as procedural simulation, captured videos or interactive user sketches. In this work, we introduce a sampling-based approach to construct a morphable crowd model. The morphable model encodes the spatial and temporal characteristics of existing crowd data and reproduces their plausible variations. Though our work is technically based on an extension of data-driven approaches, its application is not limited to the data-driven domain. Our algorithm can take input from any rule-based or flow-field simulation methods to generate inbetween styles.

Generative models have provided a systematic mechanism to synthesize new motions from captured data, and enabled us to produce motion variations while retaining an allowable range of style and dynamics learned from given data. We try to characterize the statistical distribution of individuals to faithfully simulate observed spatial arrangements. Similar motivation can be found in the recent work of various fields in computer graphics. This includes designing morphable textures in the parametric space spanned by texture database [Matusik et al. 2005], generating a compact texture from a large and globally varying image [Wei et al. 2008], factoring repeated contents within and among images [Wang et al. 2008].

Our work is related to the formation control of multi-vehicle sys-

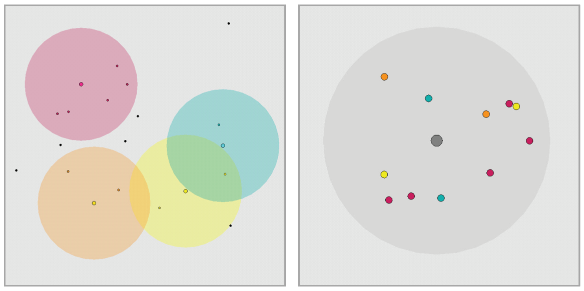

Figure 3: Formation distribution. (Left) Randomly sampling the formation of neighbors from the source crowd data. (Right) Formation samples are overlaid together to form a distribution.

Symbol	Meaning
K	The number of agents in the neighborhood
subscript k	Neighbor index
N	The number of samples in the distribution
superscript n	Sample index
J	The number of formation distributions
subscript j	Distribution index
$\mathbf{p} \in \mathbb{R}^2$	Two-dimension location of an agent
$\bar{\mathbf{p}}^n \in \mathbb{R}^2$	Average location over n
$\hat{\mathbf{p}}_k \in \mathbb{R}^2$	Permutation of $\mathbf{p}_k$
$\mathbb{P}$	A set of two-dimensional agent locations
$\mathbb{N}$	A set of agents in the neighborhood
$\mathbb{D}_j$	A formation distribution
$\mathbb{T}$	A trajectory model

Figure 4: Mathematical notation

tems in robotics, in that moving vehicles need to preserve their intended spatial configurations over a long period. A diversity of approaches has been presented to accomplish this goal. Graph Laplacian analysis has been employed to generate plausible interpolation over a sequence of time-varying formations [Takahashi et al. 2009]. Graphics researchers have also introduced various kinds of spatial constraints on moving agents for better control of coherency in groups [Kamphuis and Overmars 2004], two-dimensional configurations [Silveira et al. 2008], and three-dimensional shapes [Xu et al. 2008].

3 A Generative Model of Crowds

Our algorithm learns a model of crowd movements from observed data that consists of a set of two-dimensional moving trajectories of individuals. Example data were acquired by either processing crowd videos or running a simple rule-based crowd simulator. Some synthetic simulation results were also used to demonstrate various geometric formations and movement patterns.

Our crowd model describes how each individual agent behaves with respect to its neighbors to form group formations and reproduce a range of locomotion styles. Our crowd model is generative in the sense that it is able to generate temporally extended, spatially larger crowds that look perceptually similar to input crowd data (Figure 2). Specifically, our crowd model consists of two sub-models: A formation model that represents the distribution of neighborhood formations and a trajectory model that describes how each individual moves.

Existing data-driven methods [Lee et al. 2007; Lerner et al. 2007] addressed a similar goal of reproducing a recorded crowd simulation by learning a model from training data. Some of features for model learning were discrete-valued and thus did not allow the interpolation between feature values. Our work is distinguished from previous approaches, because our main challenge is blending multiple crowd models. The features for model learning are carefully selected such that we can generate a continuous span of inbetween models.

3.1 Formation Model

A variety of group formations appear and disappear in crowds. Some formations are persistent and some vary over time. Some formations are regular and geometric, while others are random and stochastic. Group formation is determined by the relative location of neighboring individuals. Each individual observes the location of its neighbors and decides its next movement to shape a formation.

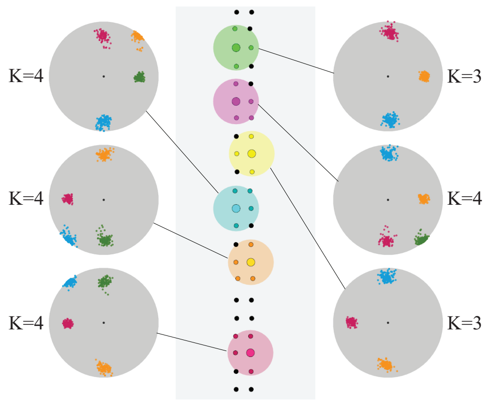

Figure 5: The formation model of a crowd of agents lining up in two rows. Agents are shown as black dots in the middle. We found six formation distributions from the crowd ($J = 6$).

Our formation model is built from a collection of neighborhood formations observed in the training data. We collect formation samples within a specified radius R of each individual (Figure 3). In our experiments, the radius is determined as the average distance to its fourth nearest neighbor. Each sample may have a different number of agents within the radius and thus is represented as a tuple of two-dimensional points $(\mathbf{p}_1, \cdots, \mathbf{p}_K)$, where K is the number of neighboring agents in the sample and $\mathbf{p}_k \in \mathbb{R}^2$ is the two-dimensional relative location of the k-th neighbor (Figure 4 describes the mathematical notation).

We sort formation samples into groups to elucidate the underlying structure of the formation distribution. For example, a crowd of agents in Figure 5 line up in two rows and the formation is near-regular and persistent. The neighborhood formations sampled from the crowd tend to be classified into six groups that characterize the crowd formation. Each group constructs a formation distribution.

We use a simple, incremental method to cluster formation samples. Assume that a set of samples are already classified in several distributions and we want to add a new sample. If the new sample is sufficiently close to any of the distributions, we insert the sample into the closest distribution. Otherwise, we create a new distribution to include the new sample. The key to the clustering algorithm is to decide if a sample can be included in a distribution. Each sample includes a set of unordered two-dimensional points $\mathbb{N} = (\mathbf{p}_1, \cdots, \mathbf{p}_K)$. The new sample cannot be included in a distribution, if it does not have the same number of points as other samples already belonging to the distribution. Otherwise,

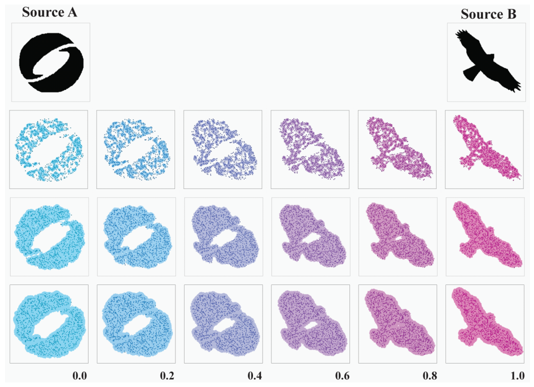

Figure 6: An illustrative example. Two-dimensional source images are reconstructed from random samples and then linearly interapolated. The inside (shaded in cyan and pink) of each reconstructed image is determined by counting samples in the neighborhood of a certain radius. The images in the second row were reconstructed with the smallest radius and the images in the next rows with larger radii.

we decide the order of the points to compare them with samples $\{(\mathbf{p}_1^n, \cdots, \mathbf{p}_K^n) \mid n = 1, \cdots, N\}$ in the distribution. Let $\bar{\mathbf{p}}_k = \frac{1}{N}\sum_n \mathbf{p}_k^n$ be the average location of the k-th points in the samples. To decide the order, we find the bipartite matching [Gibbons 1985] between sample P and the ordered tuple of average locations. The result of bipartite matching is the permutation $(\hat{\mathbf{p}}_1, \cdots, \hat{\mathbf{p}}_K)$ of $\mathbb{N}$ that minimizes the squared distance $E = \sum_k \|\hat{\mathbf{p}}_k - \bar{\mathbf{p}}_k\|^2$. The distribution includes sample $\mathbb{N}$, if E is below a specified threshold. We repeat the sampling procedure until each distribution has sufficient samples. The number of samples depends on the regularity of formations. Random/clustered formations tend to require more samples than regular/persistent formations. For simplicity, we set $N = \lceil 300/K \rceil$ for each distribution throughout the experiments. This number is conservative for the worst case. Most of examples work well with fewer samples. A distribution that does not include a significant (at least 10%) portion of the samples is removed, since it could be outliers.

3.2 Trajectory Model

Our trajectory model is built from a collection of short trajectory segments randomly sampled from source crowd data. Each segment is a sequence of two-dimensional points that represents the trace of an individual for a short interval (one second in our experiments). These short segments are strung together to generate agent trajectories in synthesized crowds. In our experiments, we conservatively sampled 300 samples from each data set. We maintain the segments in a directed graph similar to the one for human motion data [Lee et al. 2002] to avoid abrupt velocity changes in synthesized trajectories. The segments correspond to the graph nodes. Transitioning from one segment to the other is allowed if the corresponding nodes are connected by a directed graph edge.

The graph construction begins by examining each pair of segments and then creating connecting transitions where their velocities match well. Specifically, we create a transition from node i to j if $d_{ij} = \|\mathbf{v}_i^{end} - \mathbf{v}_j^{begin}\|$ is below a certain threshold, where $\mathbf{v}_i^{end} \in \mathbb{R}^2$ is the average velocity on trajectory i for the last 0.2 seconds and $\mathbf{v}_j^{begin} \in \mathbb{R}^2$ is the average velocity on trajectory j for the first 0.2 seconds.

The trajectory graph thus obtained should be further processed to

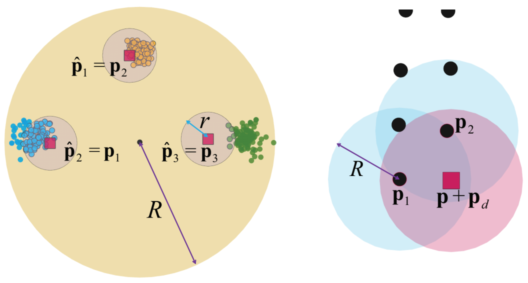
$\hat{\mathbf{p}}_1 = \mathbf{p}_2$
$\hat{\mathbf{p}}_2 = \mathbf{p}_1$
$\hat{\mathbf{p}}_3 = \mathbf{p}_3$
r
R
R
$\mathbf{p}_2$
$\mathbf{p}_1$
$\mathbf{p}+\mathbf{p}_d$

Figure 7: Consistency checking. (Left) The neighborhood $(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3)$ is an unordered tuple of points shown as red squares. The re-arranged neighborhood $(\hat{\mathbf{p}}_1, \hat{\mathbf{p}}_2, \hat{\mathbf{p}}_3)$ is consistent with the distribution if the samples are densely distributed around each point in the neighborhood. (Right) Black dots represent a formation of agents and a new agent $\mathbf{p} + \mathbf{p}_d$, shown as a red square, is added to the formation. To do so, the neighborhood of the new point, shown as a pink disk, should be consistent with the formation model and all agents in the pink disk should also have consistent neighborhoods, shown as blue disks.

prune ill-connected segments and avoid dead-ends. We first pruned segments with fewer than ten out-going transitions to ensure the diversity and flexibility in steering agents. We run a SCC (strongly connected component) algorithm to identify the largest SCC and prune segments that are not included in the SCC to avoid dead-ends. [Lee et al. 2002].

4 Crowd Synthesis

Once our morphable model is constructed from source data, we can generate a new simulation that includes an arbitrary number of agents, extends for an arbitrary duration, and looks similar to the source data. In this section, we describe two algorithms. One algorithm generates a static formation of an arbitrary number of agents that serves as the initial configuration of a synthesized crowd. The other algorithm makes every agent to proceed by the length of a trajectory segment selected from the trajectory model. Although we modeled the spatial formations and individual trajectory patterns in two separate models, our synthesis algorithm enforces a new simulation to exhibit both spatial and temporal traits of the source data simultaneously.

4.1 Plausible Formation

The formation model represents a collection of probability distributions of plausible local arrangements, which are high-dimensional. Both algorithms need to check if any given static formation of agents is consistent with respect to our formation model within a certain threshold. The formation is consistent if the neighborhood of each individual agent is consistent with any of the distributions. The threshold is defined implicitly by checking how densely the samples are populated around a given configuration in the high-dimensional space.

Illustrative example. We begin with a two-dimensional illustrative example. In Figure 6, we have two black-and-white images and 1000 points are sampled from the inside of each image. Each image can be approximately reconstructed from the sample points. A point is inside of the reconstructed image if its neighborhood of a certain radius includes more sample points than a user-specified threshold. Figure 6 shows how the radius affects image reconstruction. The larger radius makes the reconstructed images dilate. Similarly, we reconstruct a high-dimensional distribution of local arrangements from samples. The region of plausible arrangements are determined by adjusting the radius. This sampling-based approach makes the interpolation between high-dimensional functions easy, which is very important for crowd interpolation in the next section.

Checking Consistency. Let $\{\mathbb{D}_j \mid j = 1, \cdots, J\}$ be a collection of formation distributions in the formation model. Consider agent $\mathbf{p}$ and its neighborhood $\mathbb{N} = (\mathbf{p}_1, \cdots, \mathbf{p}_K)$ in the crowd within radius R. We want to check if neighborhood $\mathbb{N}$ is consistent with respect to distribution $\mathbb{D}_j = \{(\mathbf{p}_1^n, \cdots, \mathbf{p}_K^n) \mid n = 1, \cdots, N\}$ for any j. At first, if the number of agents in the neighborhood is equal to the number of samples in the distribution, we compute the permutation $(\hat{\mathbf{p}}_1, \cdots, \hat{\mathbf{p}}_K)$ of $\mathbb{N}$, such that $\sum_k \|\hat{\mathbf{p}}_k - \bar{\mathbf{p}}_k\|^2$ is minimized. The neighborhood is likely to be formed from the distribution if $\mathbf{p}_k^n$'s are densely populated around $\hat{\mathbf{p}}_k$ for each k (Figure 7(left)). The density is measured by counting the number of distributed points $\mathbf{p}_k^n$ within radius r from $\hat{\mathbf{p}}_k$, denoted by $C_r(\hat{\mathbf{p}}_k)$. In our experiments, we set $r = R/5$. $\mathbb{N}$ is consistent with $\mathbb{D}_j$, if the density is above a specified threshold for all k. The threshold is initially set by $(\mu_k - 2\sigma_k)$, where $\mu_k = \frac{1}{N}\sum_n C_r(\mathbf{p}_k^n)$ is the average density and σ_k is its standard deviation. In order to support incremental construction/development of a group formation, partial arrangement $\mathbb{N}' = (\mathbf{p}_1, \cdots, \mathbf{p}_{K'})$ for $K' < K$ passes the consistency checking if its permutation satisfies the density condition.

4.2 Simulation Algorithms

Algorithm 1: Generating the initial formation of a crowd

```
Data: The number $N$ of agents to be generated
      A collection of formation distributions $\mathbb{D}_j$
Result: A set of agent locations $\mathbb{P}$

1  $\mathbb{P} \leftarrow \{(0,0)\}$;
2  for $i \leftarrow 1$ to $N$ do
3      trial ← 0;
4      Reset the threshold for consistency checking;
5      while a new location is not selected do
6          $\mathbf{p} \leftarrow$ pick an agent from $\mathbb{P}$;
7          $\mathbf{p}_d \leftarrow$ pick a relative location from $\mathbb{D}_j$ for any $j$;
8          if Consistent $(\{\mathbb{D}_j\}, \mathbb{P} \cup (\mathbf{p} + \mathbf{p}_d))$ then
9              $\mathbb{P} \leftarrow \mathbb{P} \cup (\mathbf{p} + \mathbf{p}_d)$;
10             break;
           else
11             trial ← trial +1;
12             if trial > max_trial then
13                 Loosen the threshold for consistency checking;
14                 trial ← 0;
```

Initial Formation Generation. Our formation generation algorithm incrementally adds new agents to the crowd one-by-one, until we have a desired number of agents (Algorithm 1). The candidate location $(\mathbf{p} + \mathbf{p}_d)$ is selected by randomly picking an agent $\mathbf{p}$ from the crowd and displacement $\mathbf{p}_d$ from its formation distributions (lines 6–7). This candidate location is accepted if it is consistent with the formation model (lines 8–10). Specifically, the neighborhood $\mathbb{N}_{\mathbf{p}+\mathbf{p}_d}$ of the new point should be consistent with $\mathbb{D}_j$ for any j, and all of its neighbors affected by the new point should have consistent neighborhoods (Figure 7(right)). That is, $\mathbb{N}_{\mathbf{p}'}$ should be consistent for all $\mathbf{p}' \in \mathbb{N}_{\mathbf{p}+\mathbf{p}_d}$. If any of the neighborhoods is inconsistent, the candidate is rejected and we iterate the procedure with a new candidate. If many candidates are rejected in a row, we decrease the consistency checking threshold by σ_k for all k, so candidates can be accepted easily (lines 11–14). In our experiments, max_trial is usually 10 to 20 (line 12). The larger value of max_trial permits control that is more accurate at higher computational cost.

Algorithm 2: Crowd simulation

```
Data: A collection of formation distributions $\mathbb{D}_j$
      A collection of trajectory segments $\mathbb{T}$
      A set of agent locations $\mathbb{P}$
Result: $\mathbb{P}$ is updated

1  $\mathbb{P}_{next} \leftarrow \emptyset$;
2  for $i \leftarrow 1$ to $N$ do
3      Reset the threshold for path warping;
4      $\mathbf{p} \leftarrow$ pick an agent from $\mathbb{P}$;
5      while a new location is not determined do
6          $\mathbf{p}_{next} \leftarrow$ pick a plausible trajectory from $\mathbb{T}$ to advance $\mathbf{p}$;
7          if NoCollision $(\mathbb{P}_{next}, \mathbf{p}_{next})$ and
             Consistent $(\{\mathbb{D}_j\}, \mathbb{P}_{next} \cup \mathbf{p}_{next})$ then
8              $\mathbb{P} \leftarrow \mathbb{P} \setminus \mathbf{p}$;
9              $\mathbb{P}_{next} \leftarrow \mathbb{P}_{next} \cup \mathbf{p}_{next}$;
10             break;
           else
11             if all plausible trajectories were tested then
12                 Loosen the threshold for path warping;
13 $\mathbb{P} \leftarrow \mathbb{P}_{next}$;
```

Simulation. Once the initial crowd formation is obtained, we use Algorithm 2 iteratively to advance the simulation. Each individual agent selects a trajectory segment, which it moves along, from our morphable model. The agent stores the trajectory previously taken and its next trajectory will be selected from those that allow transitioning from its previous trajectory (line 6). The trajectory that moves an agent from $\mathbf{p}$ to $\mathbf{p}_{next}$ is accepted, if it does not cause any collision with other agents and is consistent with the formation model (lines 7–10). If all agents in the source data walked on straight paths, our trajectory model would have only straight trajectory segments and thus the simulated agents could be stuck at obstacles. We allow each path to be warped linearly within a specified threshold to provide enhanced steering flexibility(Figure 8). Initially, the threshold for path warping is set to zero. If no plausible trajectory is accepted (lines 11–12), we loosen the threshold for path warping, so the simulation does not stall.

Acceleration technique. The density calculation is the computational bottleneck of the simulation algorithm. We divide the circular range of a neighborhood into a 100×100 grid of cells, and precomputed the density C_r at the center of each cell. Then, the density computation at runtime can be performed very efficiently. We also employed uniform spatial partitioning to improve efficiency of neighborhood search in the consistency checking phase.

4.3 Hierarchical model

The clustered formation exhibits different traits at different scales. In Figure 9, the individuals tend to form a small group of companions aligning side-by-side at a fine resolution, while a number of small groups are scattered randomly over a wide area. While the fine-level formation is regular and persistent, the coarser-level formation is random and varies over time. We can simulate such a non-uniform group behavior by layering formation models in a hierarchical manner.

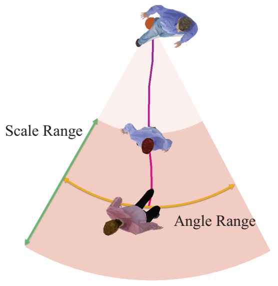

Figure 8: A trajectory segment can be warped to change its steering angle and length within specified thresholds.

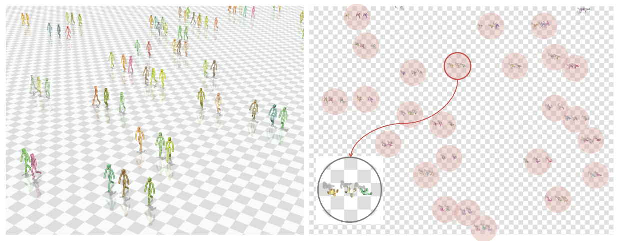

Figure 9: A hierarchical model.

Consider a two-level hierarchical model $(\mathbb{D}_j^{Upper}, \mathbb{D}_j^{Lower}, \mathbb{T})$. Let $\mathbb{P}_l = \{\mathbf{p}_l^0, \mathbf{p}_l^1, \cdots\}$ be a small group of agents and a collection of small groups $\{\mathbb{P}_1, \cdots, \mathbb{P}_L\}$ form a clustered formation at a larger scale. The groups are not necessarily of the same size. Without loss of generality, we assume that the first agent $\mathbf{p}_l^0$ of each group $\mathbb{P}_l$ is the closest to the centroid of the group. The model $(\mathbb{D}_j^{Upper}, \mathbb{T})$ at the upper-level governs the representatives $\{\mathbf{p}_1^0, \cdots, \mathbf{p}_L^0\}$ of the small groups, while the model $(\mathbb{D}_j^{Lower}, \mathbb{T})$ at the lower-level controls the agents in each group. Note that the formation model $\mathbb{D}_j^{Upper}$ should be scaled appropriately to take a small group as its individual.

We discuss the generalization of Algorithms 1 and 2 to deal with a hierarchical model with two levels. Cascading the algorithms in a top-down manner actually allows the hierarchical composition of more than two levels. The hierarchical version of Algorithm 1 begins with the upper-level model to seed new groups. The representatives $\{\mathbf{p}_1^0, \cdots, \mathbf{p}_L^0\}$ are determined based on the upper-level formation model $\mathbb{D}_j^{Upper}$ using the original version of Algorithm 1. The lower-level model is then used to add more agents $\{\mathbf{p}_l^1, \mathbf{p}_l^2, \cdots\}$ around the seed $\mathbf{p}_l^0$ of each group for all $1 \leq l \leq L$ using the same algorithm.

The hierarchical version of Algorithm 2 works in a similar manner. It begins by applying the original algorithm to the representatives of the groups. It makes one representative agent from each group to proceed by using trajectory model $\mathbb{T}$ while maintaining the arrangement among the representatives with respect to the upper-level formation model. Then, all the other agents in each group proceed to keep up with their representative based on the lower-level model $(\mathbb{D}_j^{Lower}, \mathbb{T})$ using Algorithm 2.

5 Interpolating Morphable Crowds

A brute-force approach to interpolating multiple crowd styles is to find the one-to-one correspondence between agents and then blend their spatial locations for each time instance. This brute-force approach does not work for many crowd scenes, because it cannot cope with differences in the number of agents, density, locomotion styles/speed, and their formations. We instead interpolate multiple morphable models to produce inbetween crowd styles. Morphable model interpolation separates the spatial relation among agents from their locomotion. This separation allows us to interpolate two major (spatial and temporal) aspects of crowd behavior appropriately and recombine them in the synthesis phase. In this section, we will first explain how to interpolate two morphable models and then discuss further generalization to blend more than two models.

5.1 Two-Way Blending

Given two morphable models $(\mathbb{D}_j^A, \mathbb{T}^A)$ and $(\mathbb{D}_j^B, \mathbb{T}^B)$, we want to compute their linear interpolation $(\mathbb{D}_j^C, \mathbb{T}^C)$ with weights t and $1-t$ (Figure 10(left)). We blend two trajectory models by randomly selecting pairs of trajectory segments, one from each model, and interpolating them to produce new samples. We generate 300 samples for each intermediate model, the same as for the original models.

Blending formation models is more involved, since each model has a series of formation distributions to be blended. We first establish correspondences between distributions. Correspondences are established between any pair of distributions $\mathbb{D}_j^A$ and $\mathbb{D}_{j'}^B$ that have neighborhood samples of the same size such that $K(\mathbb{D}_j^A) = K(\mathbb{D}_{j'}^B)$. Note that the correspondences between distributions are not one-to-one. A distribution in model A may have multiple counterparts in model B and vice versa. It is also possible that a distribution does not have its counterpart. In this case, we allow a correspondence between distributions having different sample lengths so that all distributions have at least one counterpart. For example, in Figure 10, crowd A has a distribution of K = 2 but crowd B does not have any of the same length. Then, our algorithm selects the closest match of K=3 from crowd B.

Interpolating two corresponding distributions requires finer-level correspondences between samples and points (Figure 10(right)). Let $\bar{\mathbf{p}}_k^A$ and $\bar{\mathbf{p}}_k^B$ be the average locations of the k-th points in the samples of model A and B, respectively. Bipartite matching establishes the correspondence between the average points. Correspondence between $\bar{\mathbf{p}}_k^A$ and $\bar{\mathbf{p}}_{k'}^B$ implies that a set of k-th points in the samples of model A matches a set of k'-th points in the samples of model B. Therefore, we find the point-level correspondence between $\{\mathbf{p}_k^n | n = 1, \cdots, N\}$ of A and $\{\mathbf{p}_{k'}^n | n = 1, \cdots, N\}$ of B also using bipartite matching. Once the point-level correspondences are established, we linearly interpolate corresponding points to generate a continuous span of inbetween formation models.

5.2 Multi-Way Blending

Consider more than two morphable models to be blended and their affine weights w_m, where $\sum w_m = 1$. Multi-way blending of trajectory models is similar to two-way blending. We randomly pick a trajectory segment from each model and blend them with weights to generate a new sample for the inbetween trajectory model. We repeat the sampling procedure until a desired number of interpolated samples is collected.

Two-way blending relies on bipartite matching to establish correspondences between formation models at multiple levels. However, optimal matching does not generalize readily to deal with more than two corresponding sets. For example, tripartite matching is a famous NP-complete problem. Instead of solving an optimal multipartite matching problem, we blend multiple models one-by-one, applying bipartite matching repeatedly. The multi-partite matching

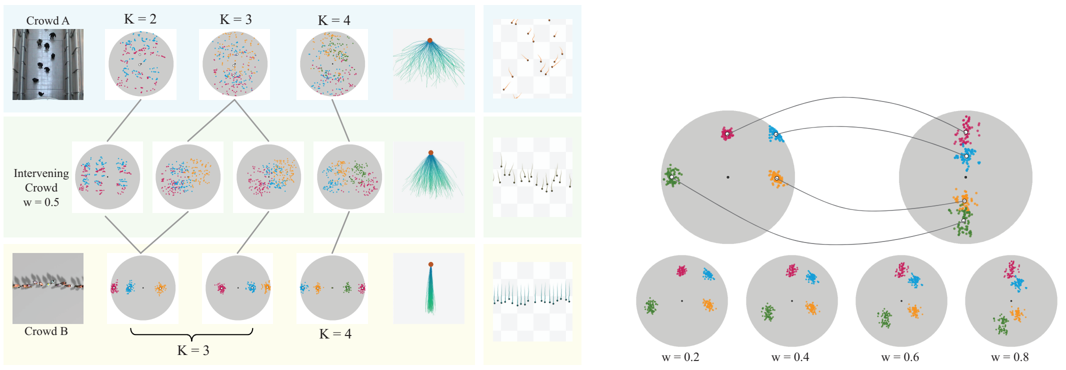

Figure 10: Morphable crowd interpolation. (Left) The correspondence between distributions are established. (Right) Two corresponding distributions are interpolated. The average points of groups match first to identify correspondence between groups. Then, points in each pair of corresponding groups establish correspondences.

Motion data				Generative model	
Style	# of frames	# of agents	# of agents per frame	R(N = 4)	# of clusters
Ants	1907	20	20.00	6.36	6
Chat	201	10	6.78	1.55	5
Stagger	700	24	4.96	4.52	3
Lining	600	17	7.75	2.33	9
Oneway	166	24	9.97	1.96	5
Spectator	201	16	15.55	1.32	4
Stroll	700	20	5.93	3.84	4
Aggressive	619	46	8.18	2.01	6
Army	200	100	100.00	1.72	1
Horizontal	200	80	80.00	1.72	3
Vertical	200	80	80.00	1.74	3
Two Horizontal	200	160	160.00	1.02	6
Two Vertical	200	160	160.00	1.02	6

Table 1: Experimental data. Thirteen motion data were acquired both from real and synthetic crowds.

thus obtained is approximate and dependent on the order of blending models. In our experiments, this approximate matching worked well without introducing any noticeable artifacts.

6 Experimental Results

We demonstrate the power and flexibility of our morphable crowd model through a variety of examples. Thirteen crowd samples were acquired from either real or synthetic crowds (see Table 1). Each crowd data were pre-processed to construct its generative model and find matchings between formation models for crowds blending. They took about one minute to create a single crowd model and establish correspondences between formation distributions on a desktop PC equipped with Intel Core i7 CPU 860 2.8GHz and 4GB main memory. The computation time for each crowd model is about the same. We exhibited some experimental results in our demonstration video and the detailed timing analysis for these results was specified in the supplemental material.

Crowd Synthesis. Each crowd model can create crowd simulation that includes many agents and extends for an arbitrarily long duration. We can observe similar spatial arrangements and locomotion styles in the original and reconstructed data. The simulation algorithm exhibits moderate run-time performance, allowing about two hundred agents to be simulated at 10 fps on a single-threaded CPU. The computation time increases almost linearly with the number of agents. The simulation provides the moving path of every agent. When we synthesize three dimensional character behavior from the path, its facing direction and fullbody motion are estimated in a similar manner done by Lee et al. [2007].

Path following. In Figure 11, a 5-by-5 regular grid of agents moves following a user-drawn path on the ground. The color of agents is green initially and changes gradually to pink as they are following the path. The agents are governed by a formation model learned from persistent, regular-grid formations. This example shows how the formation model and the trajectory model interact with each other. At a sharp turn, the agents outside the corner should walk significantly faster than the agents inside the corner. Their trajectory model was learned from straight walking data and thus does not allow such variations in walking speed. As a result, the formation breaks down naturally at sharp turns, but they quickly recover their formation after the turn. Note that the global shape of the formation after the turn is different than their initial formation. Instead, the recovered formation matches locally to the distribution of the formation model. In this specific example, the agents quickly align with each other in rows and columns after the turn.

Analysis. We performed a qualitative analysis based on a set of statistical measures including the average speed, direction, density of agents, and the spatial randomness of their formation (Figure 12). Given two morphable models, Army and Stroll, a series of inbetween models were generated as the blend weight varies linearly from zero to one. Ideally, all the statistics of the inbetween models should vary linearly. We employed Ripley's K-function from spatial statistics to estimate the randomness, regularity, and clustering of spatial distribution at three distance scales [Ripley 1981]. For any distance d, K-function is defined in our context as $K(d) = \frac{1}{\lambda}E(d)$, where λ is the average density of a crowd, and $E(d)$ is the average number of individuals within a distance d from an arbitrary individual. If individuals in a crowd are distributed in a completely random pattern, $K(d)$ equals to πd^2 for any d. Otherwise, $K(d)$ above or below πd^2 indicates that the distribution exhibits more clustered or regular pattern, respectively. The graphs of the average speed, direction and density demonstrate that the statistics vary as expected though they are not precisely linear. Spatial randomness does not change in proportion to the blend weight. It makes sense because an inbetween model tends to exhibit more randomness in its spatial arrangements than the original models.

7 Discussion

Our morphable crowd model provides the user with an intuitive and predictable control over various crowd styles. We built a parametric style space from a collection of example crowds. A rich variety of crowd styles can be easily created, blended, and combined in

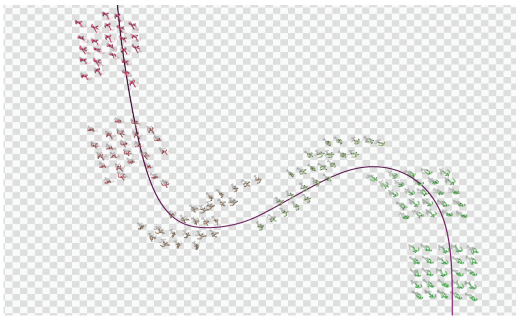

Figure 11: Path following.

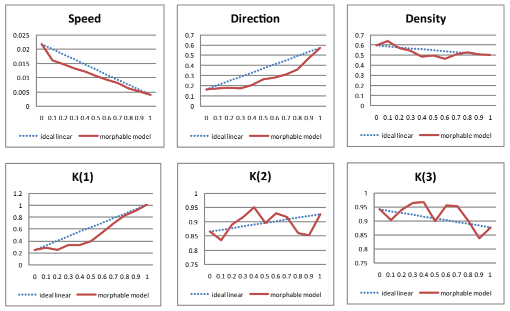

Figure 12: Analysis of morphable model interpolation.

a crowd scene. Two primary sources of example data were rule-based agent simulation and real crowd video. We can also think of an artist-friendly approach, that is, to allow artists to design a short animation of group behavior. Then, our system would be able to create a large scale crowd animation that extends spatially and temporally.

Deciding radius R is important to understand the structure of crowd behavior. A large value of R allows us to capture large patterns in crowd data to generate many formation distributions at higher computational costs. With a small value of R, we could miss some patterns. We had to make a trade-off between the representation power and computational efficiency.

Regardless of the choice of the radius, there are some crowd behaviors our morphable model cannot capture faithfully. The video in Figure 13 recorded a herd of gnus, which shaped a sparse formation at the beginning of the video. They came together gradually to pass through a narrow passage. Our morphable model may capture a mixture of dense and sparse formations, but cannot represent the temporal variation of the formation. It is unclear if each individual model should be powerful enough to capture temporally-varying styles. An alternative approach is to represent the temporal variation as the interpolation of two morphable models.

Our morphable model has several limitations. Currently, we do not take environment features into account except for collision avoidance with obstacles. Incorporating environment features into model learning would allow a wider variety of group behaviors as shown in previous work [Lee et al. 2006; Yersin et al. 2009]. It would be possible to parameterize crowd styles based on environment features. For example, interpolating passing-through-a-narrow-passage and passing-through-a-wider-passage would generate a continuous span of stuck-at-a-bottleneck behaviors. Simulating a very dense crowd is cumbersome in our system. A hybrid approach combined with a local collision avoidance model, such as Guy et al. [2009] and Pelechano et al. [2007], might alleviate the difficulty.

Our morphable model would generalize to represent other unorganized time-series data, such as Lagrangian particles for fluid dynamics simulation. It might be possible to apply our data-driven method to fluid simulation. Fluid example data would be collected through precise, time-consuming simulation and then morphable fluid models would be learned from the example data. Using morphable fluid models, it would be possible to reconstruct new fluid simulation at an arbitrary scale and interpolate morphable models to produce a span of intervening fluids.

Acknowledgement

This research was partly supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology (Project No. 2010-0008243). The first author was partly supported by Korea Research Foundation Grant funded by the Korean Government (MOEHRD)(KRF-2007-511-D00221).

References

ALI, S., AND SHAH, M. 2008. Floor fields for tracking in high density crowd scenes. In *ECCV '08: Proceedings of the 10th European Conference on Computer Vision*, 1–14.

ANDRILUKA, M., ROTH, S., AND SCHIELE, B. 2008. People-tracking-by-detection and people-detection-by-tracking. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition, 2008. (CVPR 2008)*, 1–8.

COURTY, N., AND CORPETTI, T. 2007. Crowd motion capture. *Computer Animation and Virtual Worlds 18*, 4-5, 361–370.

GIBBONS, A. 1985. *Algorithmic Graph Theory*. Cambridge University Press.

GUY, S. J., CHHUGANI, J., KIM, C., SATISH, N., DUBEY, P., LIN, M., AND MANOCHA, D. 2009. Clearpath: Highly parallel collision avoidance for multi-agent simulations. In *SCA '09: Proceedings of the 2009 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 177–187.

KAMPHUIS, A., AND OVERMARS, M. H. 2004. Finding paths for coherent groups using clearance. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 233–242.

KIM, M., HYUN, K. L., KIM, J., AND LEE, J. 2009. Synchronized multi-character motion editing. *ACM Transactions on Graphics (SIGGRAPH 2009) 28*, 3, 1–9.

KOVAR, L., AND GLEICHER, M. 2004. Automated extraction and parameterization of motions in large data sets. *ACM Transactions on Graphics (SIGGRAPH 2004) 23*, 3, 559–568.

KWON, T., LEE, K. H., LEE, J., AND TAKAHASHI, S. 2008. Group motion editing. *ACM Transactions on Graphics (SIGGRAPH 2008) 27*, 3, 1–8.

LAI, Y.-C., CHENNEY, S., AND FAN, S. 2005. Group motion graphs. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 281–290.

LEE, J., CHAI, J., REITSMA, P. S. A., HODGINS, J. K., AND POLLARD, N. S. 2002. Interactive control of avatars animated with human motion data. *ACM Transactions on Graphics (SIGGRAPH 2002) 21*, 3, 491–500.

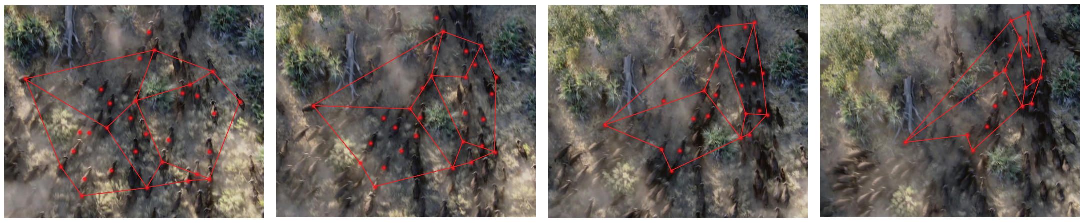

Figure 13: Group formation varying over time.

LEE, K. H., CHOI, M. G., AND LEE, J. 2006. Motion patches: building blocks for virtual environments annotated with motion data. *ACM Transactions on Graphics (SIGGRAPH 2007) 26*, 3, 898–906.

LEE, K. H., CHOI, M. G., HONG, Q., AND LEE, J. 2007. Group behavior from video: a data-driven approach to crowd simulation. In *SCA '07: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 109–118.

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

LERNER, A., FITUSI, E., CHRYSANTHOU, Y., AND COHEN-OR, D. 2009. Fitting behaviors to pedestrian simulations. In *SCA '09: Proceedings of the 2009 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 199–208.

LERNER, A., CHRYSANTHOU, Y., SHAMIR, A., AND COHEN-OR, D. 2010. Context-dependent crowd evaluation. *Computer Graphics Forum (Pacific Graphics 2010) 29*.

MATUSIK, W., ZWICKER, M., AND DURAND, F. 2005. Texture design using a simplicial complex of morphable textures. *ACM Transactions on Graphics (SIGGRAPH 2005) 24*, 3, 787–794.

MCDONNELL, R., LARKIN, M., HERNANDEZ, B., RUDOMIN, I., AND O'SULLIVAN, C. 2009. Eye-catching crowds: Saliency based selective variation. *ACM Transactions on Graphics (SIGGRAPH 2009) 28*, 3, 1–10.

MUBBASIR KAPADIA, SHAWN SINGH, B. A. G. R. P. F. 2009. Steerbug: An interactive framework for specifying and detecting steering behaviors. In *SCA '09: Proceedings of the 2009 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 209–216.

NARAIN, R., GOLAS, A., CURTIS, S., AND LIN, M. 2009. Aggregate dynamics for dense crowd simulation. *ACM Transactions on Graphics (SIGGRAPH Asia 2009) 28*, 6, 1–8.

PATIL, S., VAN DEN BERG, J., CURTIS, S., LIN, M. C., AND MANOCHA, D. 2010. Directing crowd simulations using navigation fields. *IEEE Transactions on Visualization and Computer Graphics 99*, RapidPosts.

PELECHANO, N., ALLBECK, J. M., AND BADLER, N. I. 2007. Controlling individual agents in high-density crowd simulation. In *SCA '07: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 99–108.

PETERS, C., AND ENNIS, C. 2009. Modeling groups of plausible virtual pedestrians. *IEEE Computer Graphics and Applications 29*, 4, 54–63.

PETTRÉ, J., GRILLON, H., AND THALMANN, D. 2008. Crowds of moving objects: navigation planning and simulation. In *SIGGRAPH '08: ACM SIGGRAPH 2008 classes*, ACM, New York, NY, USA, 1–7.

PETTRE, J., ONDREJ, J., OLIVIER, A.-H., CRETUAL, A., AND DONIKIAN, S. 2009. Experiment-based modeling, simulation and validation of interactions between virtual walkers. In *SCA '09: Proceedings of the 2009 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 189–198.

RIPLEY, B. D. 1981. *Spatial statistics*. John Wiley & Sons.

ROSE, C., COHEN, M. F., AND BODENHEIMER, B. 1998. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics and Applications 18*, 5, 32–40.

SILVEIRA, R., PRESTES, E., AND NEDEL, L. P. 2008. Managing coherent groups. *Computer Animatation and Virtual Worlds 19*, 3-4, 295–305.

SUD, A., GAYLE, R., ANDERSEN, E., GUY, S., LIN, M., AND MANOCHA, D. 2007. Real-time navigation of independent agents using adaptive roadmaps. In *VRST '07: Proceedings of the 2007 ACM symposium on Virtual reality software and technology*, ACM, New York, NY, USA, 99–106.

SUD, A., GAYLE, R., GUY, S., ANDERSEN, E., LIN, M., AND MANOCHA, D. 2007. Real-time simulation of heterogeneous crowds. 1–8.

TAKAHASHI, S., YOSHIDA, K., KWON, T., LEE, K. H., LEE, J., AND SHIN, S. Y. 2009. Spectral-based group formation control. *Computer Graphics Forum (Eurographics 2009) 28*, 2, 639–648.

THALMANN, D., O'SULLIVAN, C., YERSIN, B., MAIM, J., AND MCDONNELL, R. 2007. Populating virtual environments with crowds. In *Eurographics 2007 Tutorials*, vol. 2, 869–964.

TREUILLE, A., COOPER, S., AND POPOVIC, Z. 2006. Continuum crowds. *ACM Transactions on Graphics (SIGGRAPH 2006) 25*, 3, 1160–1168.

WANG, H., WEXLER, Y., OFEK, E., AND HOPPE, H. 2008. Factoring repeated content within and among images. *ACM Transactions on Graphics (SIGGRAPH 2008) 27*, 3, 1–10.

WEI, L.-Y., HAN, J., ZHOU, K., BAO, H., GUO, B., AND SHUM, H.-Y. 2008. Inverse texture synthesis. *ACM Transactions on Graphics (SIGGRAPH 2008) 27*, 3, 1–9.

XU, J., JIN, X., YU, Y., SHEN, T., AND ZHOU, M. 2008. Shape-constrained flock animation. *Computer Animation and Virtual Worlds 19*, 3-4, 319–330.

YERSIN, B., MAÏM, J., PETTRÉ, J., AND THALMANN, D. 2009. Crowd patches: populating large-scale virtual environments for real-time applications. In *I3D '09: Proceedings of the 2009 symposium on Interactive 3D graphics and games*, 207–214.