Group Motion Editing

Taesoo Kwon
Seoul National University

Kang Hoon Lee
Kwangwoon University

Jehee Lee
Seoul National University

Shigeo Takahashi
University of Tokyo

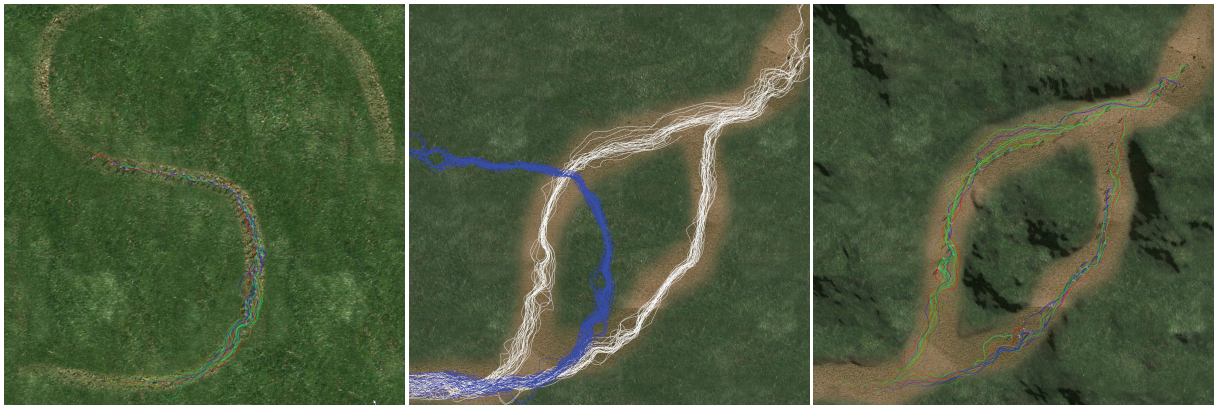

Figure 1: *Battlefield. An existing group motion on the left is adapted to a significantly different environment on the right. In the middle, the blue curves represent the original trajectories, and the white curves are the edited ones.*

Abstract

Animating a crowd of characters is an important problem in computer graphics. The latest techniques enable highly realistic group motions to be produced in feature animation films and video games. However, interactive methods have not emerged yet for editing the existing group motion of multiple characters. We present an approach to editing group motion as a whole while maintaining its neighborhood formation and individual moving trajectories in the original animation as much as possible. The user can deform a group motion by pinning or dragging individuals. Multiple group motions can be stitched or merged to form a longer or larger group motion while avoiding collisions. These editing operations rely on a novel graph structure, in which vertices represent positions of individuals at specific frames and edges encode neighborhood formations and moving trajectories. We employ a shape-manipulation technique to minimize the distortion of relative arrangements among adjacent vertices while editing the graph structure. The usefulness and flexibility of our approach is demonstrated through examples in which the user creates and edits complex crowd animations interactively using a collection of group motion clips.

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

Keywords: Group Motion Editing, Crowd Simulation, Human Motion, Character Animation

1 Introduction

Crowd scenes appear frequently in recent feature animation films and video games. Typical examples include pedestrians walking in the street, soldiers fighting in a battle, and spectators watching a performance. State-of-the-art techniques in crowd animation make it possible to synthesize convincing animations of virtual crowds by simulating each individual that chooses its actions based on rules, decision models, or force fields. However, these techniques often require careful parameter tuning to achieve desired results and still lack precise control of individuals. In practice, multiple runs of simulation with different parameters on the same scene have their own advantages and disadvantages. Achieving overall satisfactory results often requires laborious trial-and-error. We propose an interactive editing scheme that complements such simulation-based techniques in a way that animators have direct control over animated crowd behaviors. Using our system, an animator can selectively edit and combine some portions of the simulation results to achieve globally satisfactory results.

Manipulating the motion of multiple characters by repositioning only a few characters would facilitate the editing of large-scale crowds consisting of hundreds or thousands of characters. In order to provide the user with plausible and predictive outcomes, group motion editing needs to preserve not only each individual trajectory but also the group formation of individuals. Although a number of methods have been developed for editing single character motions, only a few of them addressed the interactive editing of multiple character motions.

We represent a group motion as a novel graph structure, in which each vertex represents the location of an individual at a sampled frame. Connecting edges encode individual moving trajectories and neighborhood formations. Given a graph structure, we employ a mesh editing method [Igarashi et al. 2005] for manipulating group motions. The user can deform a group motion and stitch two independent group motions while avoiding collisions between characters/obstacles and maintaining the characteristics of the original group motion data.

The practical usefulness and flexibility of our approach is demon-

strated by two examples; adapting an existing group motion along a S-shaped path to a different environment with branched paths (see Figure 1), and authoring a large-scale crowd animation in a complex town environment by using a few short motion clips.

2 Background

Synthesizing realistic group motions has been extensively explored by computer graphics researchers in the last two decades. Many existing techniques employ agent models, in which each agent perceives its own state and decides the following actions based on a set of predefined rules [Reynolds 1987; Musse and Thalmann 1997; Pelechano et al. 2005; Shao and Terzopoulos 2005]. Alternative approaches directly model the global flow of crowds either by designing velocity fields manually [Chenney 2004], or by specifying the governing equations of the flow that continuously update the velocity fields according to the distribution of crowds [Hughes 2003; Treuille et al. 2006]. Recently, several researchers presented data-driven methods of constructing group behavior models based on a set of simulated group formations [Lai et al. 2005], or captured motion data of real human crowds [Lee et al. 2007; Lerner et al. 2007; Courty and Corpetti 2007; Paris et al. 2007]. Although a number of useful techniques exist for creating convincing group motions, few approaches have been presented for editing existing group motions.

Motion editing has been an important issue in character animation since the advent of motion capture technology. For animating characters in various figures, environments, and scenarios in an intuitive manner, a variety of editing methods have been developed, which allow motion clips to be manipulated with constraints [Gleicher 1997; Lee and Shin 1999], concatenated [Rose et al. 1996], interpolated [Rose et al. 1998; Mukai and Kuriyama 2005], and rearranged [Lee et al. 2002; Kovar et al. 2002; Arikan et al. 2003]. Our goal is to provide users with a similar level of flexibility in editing group motions.

Our approach is based on the recent studies on detail-preserving shape editing [Igarashi et al. 2005; Sorkine et al. 2004; Lipman et al. 2005]. Maintaining the local arrangement of vertices allows users to intuitively manipulate two-dimensional and three-dimensional shapes without severely distorting shape details. More recently, this detail-preserving approach has been extended to edit a sequence of deforming meshes [Xu et al. 2007].

3 Graph Construction

Our system allows multiple motion clips to be edited in a shared timeline so that a large crowd animation can be created using many tractable motion clips. A group of characters can be divided into multiple groups, and multiple groups can be merged together to form a larger group. A single group motion clip can be split into multiple clips in the time domain, and multiple clips can be stitched together to create a longer clip as well. Each motion clip can be relocated in the timeline, and deformed/transformed to arbitrary locations and orientations in a virtual environment.

Each group motion clip consists of a set of two-dimensional moving trajectories of individual characters. For the simultaneous manipulation of the multiple trajectories while preserving spatial relations among individuals, we build a graph structure in which edges connect the vertices sampled from individual moving trajectories, both temporally and spatially. Specifically, a graph $\mathbf{G}$ is characterized by the number of sampled frames T and the number of individuals N. For each regularly down-sampled frame $i \in [1, 2, \cdots, T]$, we create a set of vertices $\{\mathbf{v}_{i,j}\}_{j=1}^{N}$, where vertex $\mathbf{v}_{i,j}$ defines the two-dimensional location of individual j at frame i as shown in Figure 2 (b). We down-sampled motion trajectories at every second

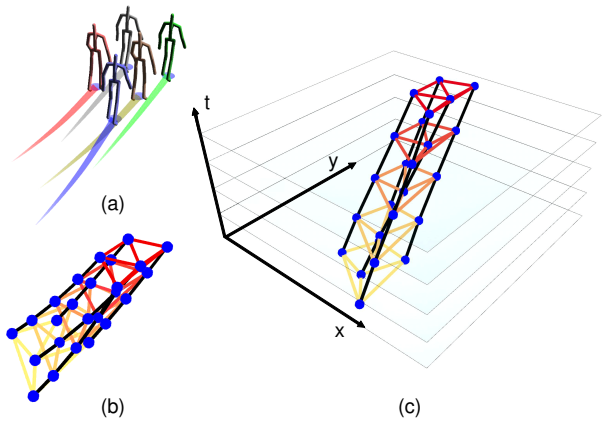

Figure 2: *Graph representation of a group motion clip. (a) A short group motion clip. (b) A graph constructed from the clip. (c) Conceptual view of the graph. The graph encodes time-varying group formations.*

for efficient editing of large-scale group motions. The vertices are then interconnected by two sets of edges, *formation edges* (colored edges in the figure) and *motion edges* (black edges). As shown in Figure 2(c), our graph structure can be viewed as a collection of temporally layered two-dimensional planes on which time-varying group formations are annotated. The formation edges connect the vertices of a graph at each plane, and the motion edges connect two adjacent planes.

Formation edges represent the neighborhood relationships between individuals. A pair of vertices in the same plane is connected with each other by a formation edge if the corresponding individuals should maintain their relative formations such that the positional change of one leads to the positional change of the other. However, we usually have no explicit knowledge of such relationships because group motion data simply is a set of independent trajectories. Moreover, neighborhoods can vary according to time because each individual character often does not keep pace with other characters, and moves at variable speeds. It is thus very challenging to correctly identify the formational relationships. Our system relies on a heuristic solution for defining such relationships; the Delaunay triangulation of vertices at each plane (time-slice) produced reasonable connectivity that associates nearby characters based on spatial distances. Despite many desirable properties of Delaunay triangulation, we can have some undesirable connections such as a long edge between two distant characters. To alleviate this, our system allows users to manually adjust the connectivity.

Motion edges correspond to sampled trajectories. For every vertex, we create two adjacent motion edges to connect it with the vertex of the corresponding character at the previous plane and the next plane, respectively. Each vertex on the first and last planes has only one adjacent motion edge.

4 Editing Group Motions

Once a graph has been constructed from a motion clip, a user can deform the graph by pinning or dragging vertices to desired locations interactively. Whenever a vertex is repositioned, our system recalculates every vertex position to minimize the shape distortion of the original graph while satisfying the positional constraints.

To measure the distortion of a deformed graph $\mathbf{G}'$ from the original graph $\mathbf{G}$ in a coordinate-invariant manner, we consider local fea-

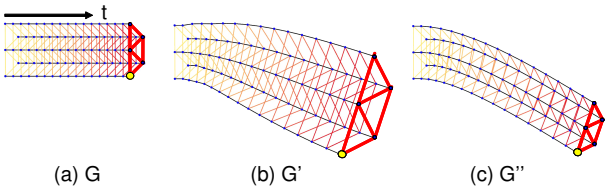

Figure 3: *Graph deformation. In source animation (a), five individuals are marching in two rows. The source animation is deformed by dragging the yellow-colored vertex. (b) Laplacian deformation could result in unnatural scaling artifacts. (c) The scale compensation step alleviates the artifacts.*

tures that encode the relative position of each vertex with respect to its adjacent vertices. Let us define a local feature $\mathbf{c}$ for a vertex $\mathbf{v}$ as an ordered triple of vertices $(\mathbf{u}, \mathbf{v}, \mathbf{w})$, where $\mathbf{u}$ and $\mathbf{w}$ are adjacent to $\mathbf{v}$. Then, the vertex $\mathbf{v}$ in the middle can be described in a local coordinate system defined by the neighboring vertices $\mathbf{u}$ and $\mathbf{w}$ as follows:

$$\mathbf{v} = \mathbf{u} + c_x(\mathbf{w} - \mathbf{u}) + c_y\mathbf{R}(\mathbf{w} - \mathbf{u}) \quad (1)$$

where $\mathbf{R}$ is a 90-degree rotation matrix in a counter-clockwise direction. Two-dimensional vector (c_x, c_y) can be regarded as a local coordinate of $\mathbf{v}$ with respect to the coordinate system defined by origin $\mathbf{u}$ and two orthogonal bases $(\mathbf{w} - \mathbf{u})$ and $\mathbf{R}(\mathbf{w} - \mathbf{u})$. For notational convenience, we define a function $\mathbf{f_c}$ that returns the desired location of $\mathbf{v}$ by linearly combining adjacent vertices $\mathbf{u}$ and $\mathbf{w}$ based on the local coordinate from $\mathbf{c}$:

$$\mathbf{f_c}(\mathbf{u}, \mathbf{w}) = \mathbf{u} + c_x(\mathbf{w} - \mathbf{u}) + c_y\mathbf{R}(\mathbf{w} - \mathbf{u}). \quad (2)$$

Now, we can measure the difference between any original feature $\mathbf{c} = (\mathbf{u}, \mathbf{v}, \mathbf{w})$ and its deformed feature $\mathbf{c}' = (\mathbf{u}', \mathbf{v}', \mathbf{w}')$, where $\mathbf{u}, \mathbf{v}, \mathbf{w} \in \mathbf{G}$ and $\mathbf{u}', \mathbf{v}', \mathbf{w}' \in \mathbf{G}'$, by the squared distance between $\mathbf{f_c}(\mathbf{u}', \mathbf{w}')$ and $\mathbf{v}'$ as follows:

$$D(\mathbf{c}, \mathbf{c}') = \|\mathbf{f_c}(\mathbf{u}', \mathbf{w}') - \mathbf{v}'\|^2. \quad (3)$$

Using the distortion metric $D(\cdot)$ defined above, we formulate our motion editing process as a constrained least-squares optimization problem of finding the optimal arrangement of free vertices that minimizes the distortion of local features. The optimization can be efficiently performed by using a sparse linear system solver.

This quadratic distortion metric, however, has an undesirable property; it is invariant on the uniform scaling of local features. This property allows original features to be unnaturally enlarged, or shrunken since such scale distortions are not penalized by the objective function. This problem has been reported in the field of mesh editing. To alleviate the problem, Igarashi and his colleagues [2005] proposed a two-step optimization scheme that solves two least-squares problems sequentially. We extend this two-step algorithm to deal with group motion editing.

An input graph $\mathbf{G}$ is first deformed to yield an intermediate graph $\mathbf{G}'$ that has minimal overall distortion of local features while allowing free translation, rotation, and uniform scaling (see Figure 3(a) and (b)). Unnecessary scaling effects are then compensated to produce the final deformed graph $\mathbf{G}''$ in the subsequent scale-adjustment step (see Figure 3(c)). The scale adjustment method of [Igarashi et al. 2005] first scales deformed triangles to best match the original ones, and then reconstructs a final mesh from the scaled triangles. While effective in eliminating the scale distortion, it sometimes introduces artifacts such as highly-distorted, near-degenerate triangles, especially when the graph undergoes a large

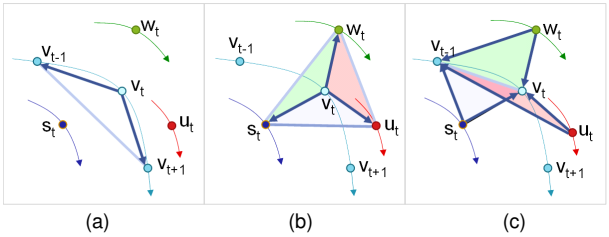

Figure 4: *Three types of triangular features. (a) Temporal feature. (b) Spatial feature. (c) Spatiotemporal feature.*

deformation. Directly applying the method to group motion editing could cause motion artifacts such as sudden velocity changes in deformed trajectories. Instead, we apply the scale-adjustment method only to adjust the spatial group formation at each time frame. Temporal distortions such as stretched/squeezed motion trajectories are handled separately by using a time-warping method explained in Section 6.2.

In the following subsection, we describe how to manipulate a group motion represented by graph $\mathbf{G}$ based on scale-free Laplacian deformation. The resulting graph $\mathbf{G}'$ represents a deformed group motion, which is further processed to compensate scaling artifacts in Section 4.2.

4.1 Laplacian Deformation

An intermediate graph $\mathbf{G}'$ should minimize the distortion of local features while satisfying the positional constraints. We consider three types of triangular features from original graph $\mathbf{G}$: spatial, temporal, and spatiotemporal features. A spatial feature of vertex $\mathbf{v}$ is defined by an adjacent triangle lying on the time plane containing $\mathbf{v}$ (see Figure 4(b)). A triangle consisting of $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{w}$ defines three spatial features, $(\mathbf{u}, \mathbf{v}, \mathbf{w})$, $(\mathbf{v}, \mathbf{w}, \mathbf{u})$, and $(\mathbf{w}, \mathbf{u}, \mathbf{v})$. A temporal feature is defined by a pair of adjacent motion edges. Specifically, a temporal feature of vertex $\mathbf{v}_t$, sampled at time t, corresponds to an ordered triple of vertices $(\mathbf{v}_{t-1}, \mathbf{v}_t, \mathbf{v}_{t+1})$ where $\mathbf{v}_{t-1}$ and $\mathbf{v}_{t+1}$ denote the previous and following vertices of $\mathbf{v}_t$, respectively (see Figure 4(a)). A spatiotemporal feature is defined by a motion edge and a formation edge. A spatiotemporal feature of vertex $\mathbf{v}_t$ is defined by $(\mathbf{v}_{t-1}, \mathbf{w}_t, \mathbf{v}_t)$, where $\mathbf{v}_{t-1}$ is the previous vertex of $\mathbf{v}_t$, and $\mathbf{w}_t$ is an adjacent vertex of $\mathbf{v}_t$ in the same time frame (see Figure 4(c)). Using these features, the objective function is defined as follows:

$$E_1(\mathbf{G}') = E_S(\mathbf{G}') + \alpha E_T(\mathbf{G}') + \beta E_{ST}(\mathbf{G}') \quad (4)$$

where $E_S(\cdot)$, $E_T(\cdot)$, and $E_{ST}(\cdot)$ measure the distortions of spatial, temporal, and spatiotemporal features, respectively. The weight values α and β control the relative importance among them.

Spatial features. $E_S(\cdot)$ measures the distortion of neighborhood formations:

$$E_S(\mathbf{G}') = \sum_{\mathbf{c}' \in \mathbf{C}'_S} w(\mathbf{c}) \|\mathbf{f_c}(\mathbf{u}', \mathbf{w}') - \mathbf{v}'\|^2, \quad (5)$$

where $\mathbf{C}'_S$ is a set of spatial features in which both $\mathbf{u}'$ and $\mathbf{w}'$ are connected to $\mathbf{v}'$ by formation edges, and $w(\mathbf{c})$ is the inverse of $\|\overrightarrow{\mathbf{vu}}\|^2 + \|\overrightarrow{\mathbf{vw}}\|^2$ for discounting distant formations.

Temporal features. $E_T(\cdot)$ reflects the distortion in individual moving trajectories.

$$E_T(\mathbf{G}') = \sum_{\mathbf{c}' \in \mathbf{C}'_T} \|\mathbf{f_c}(\mathbf{u}', \mathbf{w}') - \mathbf{v}'\|^2, \quad (6)$$

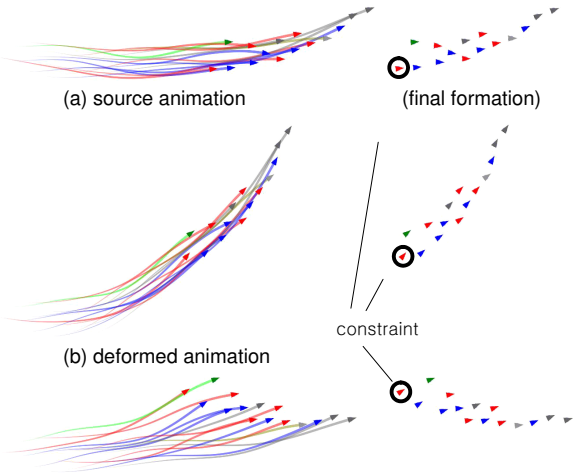

Figure 5: *A source animation (a) is deformed by dragging a vertex that belongs to the final frame shown on the right side of the figure. The first two frames are constrained to be pinned down. (b) The use of spatiotemporal term enforces the orientation of the group formation to match its moving direction. (c) Without the spatiotemporal term, the formation tends to rotate regardless of its moving direction.*

where $\mathbf{C}'_T$ is a set of temporal features in which both $\mathbf{u}'$ and $\mathbf{w}'$ are connected to $\mathbf{v}'$ by motion edges.

Spatiotemporal features. The third type of features represent the spatiotemporal relationships among vertices. Intuitively, these features prevent arbitrary rotation of group formation with respect to its moving direction (see Figure 5).

$$E_{ST}(\mathbf{G}') = \sum_{\mathbf{c}' \in \mathbf{C}'_{ST}} \|\mathbf{f_c}(\mathbf{u}', \mathbf{w}') - \mathbf{v}'\|^2, \tag{7}$$

where $\mathbf{C}'_{ST}$ is a set of spatiotemporal features in which $\mathbf{v}'$ is connected to $\mathbf{u}'$ and $\mathbf{w}'$ by a formation edge and a motion edge, respectively.

4.2 Scale Compensation

The scale adjustment begins by fitting every spatial feature in $\mathbf{G}$ to best match the corresponding spatial features in the intermediate graph $\mathbf{G}'$. The least-square fitting of each feature can be achieved by minimizing (see [Igarashi et al. 2005] for details):

$$f_{\mathbf{c}} = \|\mathbf{v}^F - \mathbf{v}'\|^2 + \|\mathbf{u}^F - \mathbf{u}'\|^2 + \|\mathbf{w}^F - \mathbf{w}'\|^2, \tag{8}$$

where $\mathbf{c}$ is a spatial feature and $\mathbf{v}^F$ is constrained to satisfy $\mathbf{v}^F = \mathbf{f_c}(\mathbf{u}^F, \mathbf{w}^F)$. Once we have fitted every spatial feature, the deformed graph $\mathbf{G}''$ can be obtained by minimizing

$$E_2(\mathbf{G}'') = E_{SC}(\mathbf{G}'') + \alpha' E_T(\mathbf{G}'') + \beta' E_{ST}(\mathbf{G}''). \tag{9}$$

Here, the new energy function $E_{SC}(\cdot)$ replaces $E_S(\cdot)$ in Equation (4):

$$E_{SC}(\mathbf{G}'') = \sum_{\mathbf{c}'' \in \mathbf{C}''_F} \left(\|\overrightarrow{\mathbf{v}''\mathbf{u}''} - \overrightarrow{\mathbf{v}^F\mathbf{u}^F}\|^2 + \|\overrightarrow{\mathbf{v}''\mathbf{w}''} - \overrightarrow{\mathbf{v}^F\mathbf{w}^F}\|^2 \right). \tag{10}$$

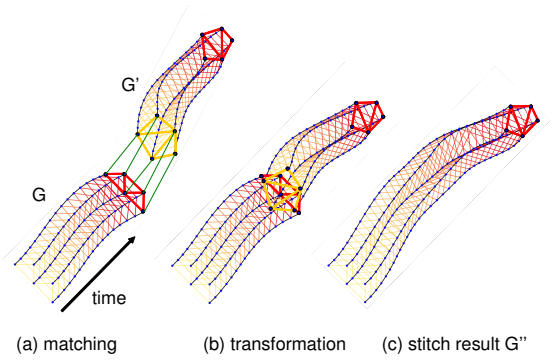

Figure 6: *Group motion stitching.*

Note that only the spatial features are scaled. The rationale is that fitting temporal and spatiotemporal features can lead to sudden velocity changes, which are susceptible to human eyes compared to the spatial irregularity in scales.

Given the objective functions E_1 and E_2, the graph deformation process is formulated as a two-step constrained least-squares optimization problem. To construct a sparse linear system that can be efficiently solved, we employ a Lagrange multiplier scheme instead of variable elimination used in [Igarashi et al. 2005] because it is easier to implement without the needs for variable reordering and special treatments of terms in the objective function involving constrained variables. The downside of a Lagrange multiplier scheme is its slightly larger system matrix compared to that of variable elimination. In our experiments, however, no noticeable performance difference was observed in solving the sparse linear system because the number of variables is usually much larger than the number of constraints. It should also be noted that a Lagrange multiplier scheme can handle general linear equality constraints, e.g., enforcing the velocity/acceleration of a character at specific time, pinning the relative position of a character with respect to the others, and constraining a character to move along a line. In particular, our system uses linear constraints to handle collision avoidance between motion trajectories as explained in Section 6.1.

5 Stitching Group Motions

Our system allows the user to merge two graphs into a single, longer graph. Given two graphs $\mathbf{G}$ and $\mathbf{G}'$ having the same number of individuals N, we build a new graph $\mathbf{G}''$ that concatenates two group motion clips sequentially in time. $\mathbf{G}''$ has $T = t + t' - 1$ time frames, where t and t' are the time frames of $\mathbf{G}$ and $\mathbf{G}'$, respectively.

Stitching two graphs requires three steps. At first, we need to establish one-to-one correspondences between individuals in two groups (see Figure 6(a)). Secondly, two motion clips should be aligned via rigid transformation (see Figure 6(b)). Finally, two group motions are concatenated by smoothly morphing group formations at the boundary (see Figure 6(c)). The first step of the procedure can be formulated as a bipartite graph matching algorithm [Belongie et al. 2002] that searches a best matching (one-to-one correspondences) between two sets of individuals while minimizing the total distance between corresponding pairs. Once the matching is found, we can align two motion clips by translating and rotating them to best match the boundary [Kovar et al. 2002].

The final step of stitching involves smooth blending of group formations. Simple linear blending of individual trajectories does not

generate desired results because the group formation is not appropriately reflected. Instead, we blend triangular (spatial, temporal, and spatiotemporal) features and construct a concatenate graph by solving the linear system presented in Section 4. This approach generates a smooth transitioning of the group formation while maintaining the detail characteristics of individual moving trajectories.

Blending temporal features. The individual trajectories should be maintained in the concatenate group motion. Therefore, we copy the temporal features of $\mathbf{G}$ to the first $(t-1)$ frames of concatenate motion $\mathbf{G}''$ and temporal feature of $\mathbf{G}'$ to the last $(t'-1)$ frames of $\mathbf{G}''$. Temporal features at intervening frame t are obtained by linearly interpolating the temporal features at frame $t-1$ and frame $t+1$.

Blending spatial and spatiotemporal features. In order to compute the smooth blending of spatial/spatiotemporal features over the concatenate timeline, we construct an intermediate graph $\bar{\mathbf{G}}$ that is a simple blending of vertex positions. The spatial and spatiotemporal features of the intermediate graph are then copied to $\mathbf{G}''$, in which all three-types of triangular features are combined and solved to produce the actual stitching of two input group motions. The intermediate graph $\bar{\mathbf{G}}$ is computed as follows: Let $\{\mathbf{w}_{i,j}\}_{j=1}^{N}$ be the frames of $\bar{\mathbf{G}}$. The frame $\mathbf{w}_{t,j}$ at intervening frame t averages the last frame of $\mathbf{G}$ and the first frame of $\mathbf{G}'$. All remaining frames are determined by applying a smooth displacement mapping to $\mathbf{G}$ and $\mathbf{G}'$. For frame $i < t$, frames $\mathbf{v}_{i,j}$ of $\mathbf{G}$ are displaced such that $\mathbf{w}_{i,j} = \mathbf{v}_{i,j} + f_a(t-i)(\mathbf{w}_{t,j} - \mathbf{v}_{t,j})$, where the blending function is

$$f_a(\Delta t) = 1 - 3\left(\frac{\Delta t}{t}\right)^2 + 2\left(\frac{\Delta t}{t}\right)^3. \qquad (11)$$

Similarly, we can determine frames $\mathbf{w}_{i,j}$ for $t+1 < i < t+t'-1$ by displacing the vertices of $\mathbf{G}'$.

6 Postprocess

In the postprocess phase, group motions are further refined to avoid collision, re-timed to cope with stretched or shrunken trajectories, and re-sampled to produce smooth trajectories by using spline interpolation. The fullbody motion of each character is then computed using a locomotion synthesis method presented by [Kwon and Shin 2007].

6.1 Collision Avoidance

The deformation or stitching of a group motion can lead to collision or insufficient clearance between individuals even though our editing method tries to maintain their relative formation. We use an iterative method for resolving collisions (see Figure 7 and 8). The collision avoidance algorithm begins by approximating each individual trajectory by a time-parameterized piecewise linear curve. If two trajectories are closer than a certain threshold at any time instance, our system pulls the trajectories away by 10% of the threshold and repeats this process until all collisions are resolved. To do so, we find the exact moment when the distance between two trajectories is minimized and deforms both trajectories such that the closest pair of points on the trajectories are pulled away. Note that the exact time and the closest pair of points can be computed analytically with piecewise linear curves. The point being pushed is expressed as a linear combination of two temporally adjacent vertex positions. Therefore, the displacement of the point can be formulated as a linear constraint, which is incorporated into the linear system derived in Section 4. For efficiency, the deformation of the graph is computed with a subgraph containing only the vertices and motion edges corresponding to colliding trajectories.

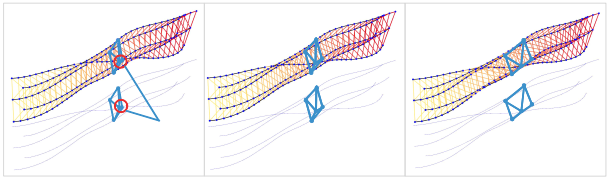

Figure 7: *Collision avoidance for a small ($N = 5$) group of characters.*

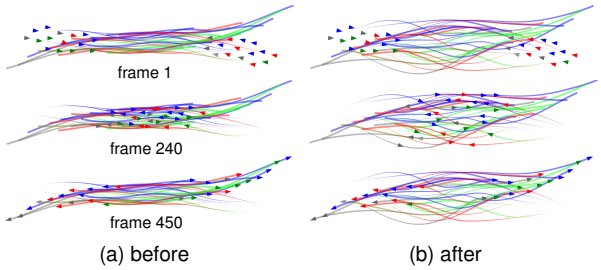

Figure 8: *Collision handling. (a) Two groups of crowds are bumping into each other. (b) Collisions are resolved iteratively while fixing the original formations at the first and last frames. Resolving all collisions required 1997 iterations, which took about four seconds.*

We also provide tools for avoiding collision with obstacles consisting of convex polygons and circles. If any point on a piecewise linear trajectory is inside an obstacle, we pull the deepest penetrating point toward the nearest point on the boundary of the obstacle. This iteration continues until all collisions between characters and obstacles are resolved. Figure 9 shows a challenging collision avoidance example.

6.2 Time Warping

The increased (or decreased) length of a deformed trajectory makes the character to move faster (or slower) than its original speed (see

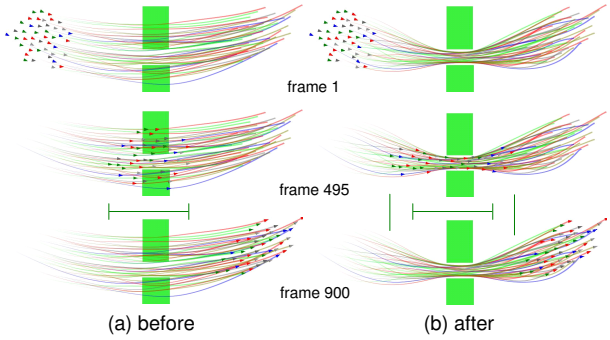

Figure 9: *Forcing a circular group (a) through a narrow opening such that the group must elongate to pass. In this specific example, penetrating points inside obstacles are pulled toward the opening. Resolving all collisions required 34733 iterations, which took about 163 seconds.*

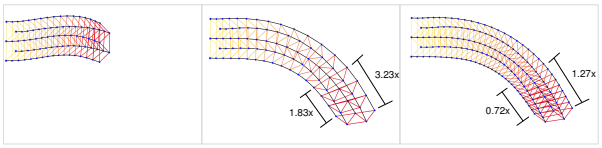

Figure 10: *Timewarping. (a) Source animation. (b) Due to deformation, the individual trajectories are stretched and thus the characters on the deformed trajectories have to walk faster than on the original trajectories. (c) Timewarping re-parameterizes the trajectories to minimize excessive speedup and slowdown.*

Figure 10). This speed change is often irregular; the character has to walk faster at some time interval on the deformed trajectory and walk slower on other intervals. Such an irregular speed change is usually undesirable in motion editing. The goal of our time warping method is to allow characters in a deformed group motion clip to move as closely to their original speeds as possible. It should be noted that all characters in motion data should speedup or slowdown at the same rate. We have considered the possibility of allowing non-uniform individual time-warping, but do not aware of any nice method that allows a group of individually time-warped characters to keep their pace.

We formulate the time-warping problem as a least-squares optimization where the objective function is defined as

$$E_t = \sum_{i=1}^{T}\sum_{j=1}^{N}\left(\frac{\|\mathbf{v}'_{i+1,j} - \mathbf{v}'_{i,j}\|}{\Delta t'_i} - \frac{\|\mathbf{v}_{i+1,j} - \mathbf{v}_{i,j}\|}{\Delta t}\right)^2, \quad (12)$$

where T is the number of time frames in the deformed group motion, N is the number of individuals in the group, $\mathbf{v}_{i,j}$ are the original vertex positions, $\mathbf{v}'_{i,j}$ are deformed positions, Δt is the time interval of the original motion data. Note that Δt is constant because the input motion data was sampled uniformly in time for constructing the graph representation. $\Delta t'_i$ is the non-uniform time interval between i-th and $(i+1)$-th time frames in the deformed graph. Taking the inverse of $\Delta t'_i$ as optimization variables, non-uniform time intervals can be efficiently computed by using a sparse linear system solver

Finally, smooth individual moving trajectories are obtained from the time-warped graph by using non-uniform cubic interpolating splines that pass through vertex $\mathbf{v}'_{i,j}$ at time t'_i [Bartels et al. 1987].

7 Experimental Results

In order to demonstrate the flexiblity and usefulness of our approach, we created three crowd animations, two-groups bumping, battlefield and downtown. For the first experiment, we edited the resulting motion of the collision handling example shown in Figure 7. For the last two experiments, we edited existing group motion clips obtained from rule-based crowd simulation, which generates a set of two-dimensional trajectories. We synthesized the final animation with multiple full-body characters that follow the two-dimensional trajectories by using an online locomotion synthesis method [Kwon and Shin 2007].

Two-groups bumping. The first example shows that our editing scheme works well with characters walking in a variety of different directions. In the source motion shown in Figure 11 (a), two groups of characters are bumping into each other and passing through without collisions. As shown in Figure 11 (b), such inhomogeneous

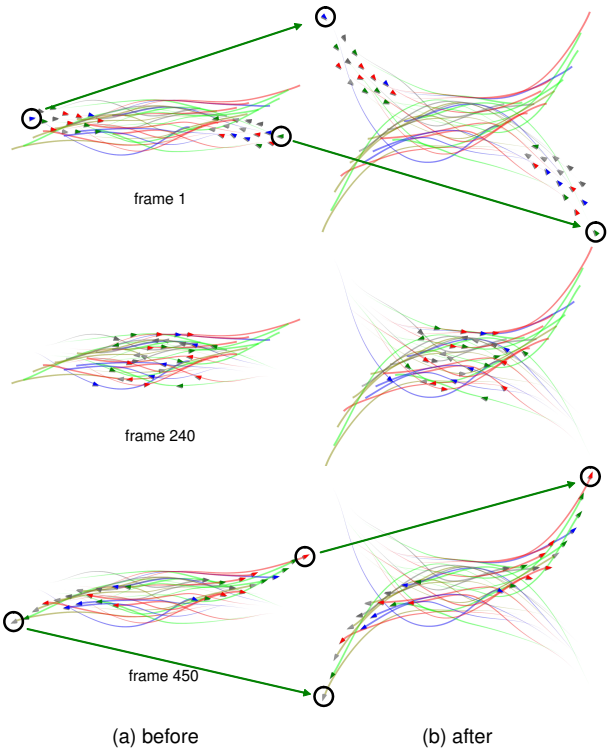

Figure 11: *A group of characters moving in two different directions. (a) Original motion. (b) Editing by dragging four vertices.*

groups are manipulated as a whole by dragging four vertices located at corners of each group at the first and the last frame. The resulting group motion does not contain any collisions without additional collision avoidance step.

Battlefield. The second experiment demonstrates that an existing crowd animation can be flexibly adapted to a new environment. In the source group motion shown in Figure 1 (left), fifty characters are densely streaming through a curved path. As shown in Figure 1 (middle), the trajectories of all characters were first rigidly transformed to be located at the starting position of the new environment having branched paths. The final group motion shown in Figure 1 (right) was generated in two steps. In the first step, we transformed the original trajectories such that the head and tail of the trajectories are located at the beginning and end of the new path, respectively. This was simply done by selecting a group of vertices, and then dragging those vertices to the desired locations. In the second step, we separated the group into two subgroups to follow branches in the path. To do so, we manually picked characters in a subgroup and then divided the group motion graph into two subgraphs. The trajectories of each group were edited again to fit to the corresponding path. It took about five minutes to perform the entire editing process (see Figure 12).

Town. The last example shows that our scheme can be used to author large-scale crowd animation by using only a few, short motion clips. We created a complex crowd scene consisting of about three hundreds characters by duplicating, merging, deforming, and stitching 19 motion clips, which are acquired by choosing five to fifty individuals from the battlefield example for 10 to 30 second duration. Those clips are copy-and-pasted onto the town environment, and then edited to construct a complex animation in which a large number of characters wandering around while avoiding colli-

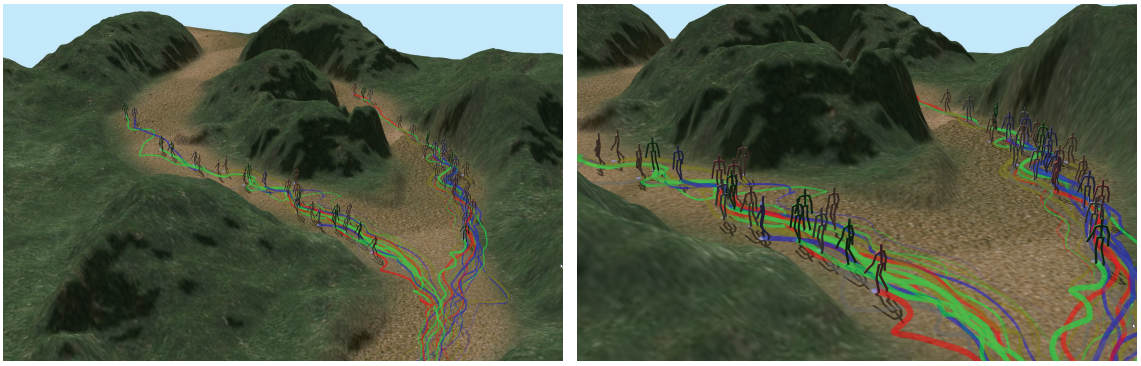

Figure 12: *Snapshots of the crowd animation in a battle field scene.*

Figure 13: *Snapshots of the crowd animation in a downtown scene.*

sion with each other (see Figure 13). The total editing time required to create this thirty second animation clip was about five hours.

In both examples, each editing operation took less than one second on average and four seconds for deforming the largest group. Table 1 summarizes the timing information.

8 Discussion

We introduced a novel group motion editing method that allows animators to manipulate existing group motion data interactively. The detail-preserving approach developed in mesh editing research was successfully extended to deal with group motions. The advantage of our method is that the user can interactively manipulate multiple character motions as a whole and still have direct, precise control over individual trajectories.

Our approach has several limitations. A large deformation of a group motion can lead to unnatural speedup/slowdown of individual motions. Although our time-warping scheme mitigates such artifacts to some extent, there are cases where characters cannot maintain their original formation unless some characters move extremely fast. We allow the user to handle such cases interactively rather than try to remove all artifacts automatically.

We chose to use the mesh editing algorithm presented by Igarashi et al. [2005] because the algorithm is easy-to-implement and based on an efficient linear system formulation. Our system can actually take any other similar mesh editing method such as canonical constrained quadratic programming [Xu et al. 2007]. Probably, a more flexible non-linear optimization method would allow us to model high-level group behaviors and inter-personal interactions.

Even with an efficient linear formulation, our system cannot handle a large crowd consisting of thousands characters at interactive rate. The crowd animations in feature films and video games are even bigger. It would be often undesirable that slight modification of an individual motion in a crowd scene results in the change of the entire crowd animation. A better approach would be to take only small part of the crowd animation into consideration, while leaving the remaining parts intact. In this way, we can edit a large crowd interactively.

Our work currently focuses on group locomotion, in which multiple characters walk or run in various directions. An interesting direction for future work is to deal with a wider variety of group behaviors, possibly captured from crowd animations, such as chatting in a party, lining up at a ticket booth, and audience in a performance. It would also be interesting to apply our interactive editing method to such complex scenes as Olympic mass games that are difficult to generate by using simulation-based methods.

Acknowledgements

We would like to thank the reviewers for their constructive comments and suggestions. We are also grateful to all the members in Zoiment Inc. for allowing us to use their mesh data. This work was supported by the IT R&D program of MKE/IITA (2008-F-033-01).

References

ARIKAN, O., FORSYTH, D. A., AND O'BRIEN, J. F. 2003. Motion synthesis from annotations. *ACM Transactions on Graphics (SIGGRAPH 2003) 22*, 3, 402–408.

Table 1: *Statistics on the final group motion clips. The maximum computation time was measured by performing a deformation operation on the largest group in the scene.*

	# of frames	# of sample frames	# of characters	# of groups	total # of vertices	# of vertices in the largest graph	maximum computation time
Two-groups bumping	450	15	30	1	450	450	219ms
Battlefield	3958	132	50	2	6600	6600	3803ms
Downtown	900	30	348	15	10440	1500	677ms

BARTELS, R. H., BEATTY, J. C., AND BARSKY, B. A. 1987. *An introduction to splines for use in computer graphics & geometric modeling*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

BELONGIE, S., MALIK, J., AND PUZICHA, J. 2002. Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Anal. Mach. Intell. 24*, 4, 509–522.

CHENNEY, S. 2004. Flow tiles. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 233–242.

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

GLEICHER, M. 1997. Motion editing with spacetime constraints. In *I3D '97: Proceedings of the 1997 Symposium on Interactive 3D graphics*, 139–148.

HUGHES, R. L. 2003. The flow of human crowds. *Annual Review of Fluid Mechanics 35*, 169–182.

IGARASHI, T., MOSCOVICH, T., AND HUGHES, J. F. 2005. As-rigid-as-possible shape manipulation. *ACM Transactions on Graphics (SIGGRAPH 2005) 24*, 3, 1134–1141.

KOVAR, L., GLEICHER, M., AND PIGHIN, F. 2002. Motion graphs. *ACM Transactions on Graphics (SIGGRAPH 2002) 21*, 3, 473–482.

KWON, T., AND SHIN, S. Y. 2007. A steering model for on-line locomotion synthesis. *Computer Animation and Virtual Worlds 18*, 4–5, 463–472.

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., AND SHIN, S. Y. 1999. A hierarchical approach to interactive motion editing for human-like figures. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, 39–48.

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.

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.

LIPMAN, Y., SORKINE, O., LEVIN, D., AND COHEN-OR, D. 2005. Linear rotation-invariant coordinates for meshes. *ACM Trans. Graph. 24*, 3, 479–487.

MUKAI, T., AND KURIYAMA, S. 2005. Geostatistical motion interpolation. *ACM Transactions on Graphics (SIGGRAPH 2005) 24*, 3, 1062–1070.

MUSSE, S. R., AND THALMANN, D. 1997. A model of human crowd behavior: Group inter-relationship and collision detection analysis. In *Computer Animation and Simulation '97*, 39–51.

PARIS, S., PETTRÉ, J., AND DONIKIAN, S. 2007. Pedestrian reactive navigation for crowd simulation: a predictive approach. *Computer Graphics Forum (Eurographics 2007) 26*, 3, 665–675.

PELECHANO, N., O'BRIEN, K., SILVERMAN, B., AND BADLER, N. 2005. Crowd simulation incorporating agent psychological models, roles and communication. In *V-CROWDS '05: Proceedings of the First International Workshop on Crowd Simulation*, 24–25.

REYNOLDS, C. W. 1987. Flocks, herds and schools: A distributed behavioral model. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, 25–34.

ROSE, C., GUENTER, B., BODENHEIMER, B., AND COHEN, M. F. 1996. Efficient generation of motion transitions using spacetime constraints. In *Proceedings of SIGGRAPH 1996*, 147–154.

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

SHAO, W., AND TERZOPOULOS, D. 2005. Autonomous pedestrians. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 19–28.

SORKINE, O., COHEN-OR, D., LIPMAN, Y., ALEXA, M., RˊOSSL, C., AND SEIDEL, H.-P. 2004. Laplacian surface editing. In *SGP '2004: Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, 175–184.

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

XU, W., ZHOU, K., YU, Y., TAN, Q., PENG, Q., AND GUO, B. 2007. Gradient domain editing of deforming mesh sequences. *ACM Transactions on Graphics (SIGGRAPH 2007) 26*, 3, 84.