Generating and Ranking Diverse Multi-Character Interactions

Jungdam Won, Kyungho Lee, Carol O'Sullivan, Jessica K. Hodgins, Jehee Lee

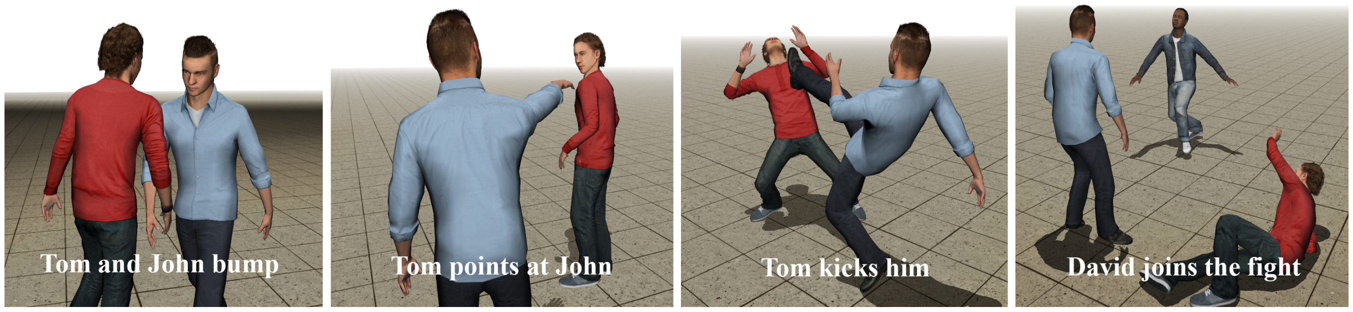

Figure 1: *We animate data-driven scenes with multi-character interactions from high-level descriptions. Many plausible scenes are generated and efficiently ranked, so that a small, diverse and high-quality selection can be presented to the user and iteratively refined.*

Abstract

In many application areas, such as animation for pre-visualizing movie sequences or choreography of dancing or other show performances, only a high-level description of the desired scene is provided as input, either written or verbal. Such sparsity, however, lends itself well to the creative process, as the choreographer, animator or director can be given more choice and control of the final scene. Animating scenes with multi-character interactions can be a particularly complex process, as there are many different constraints to enforce and actions to synchronize. Our novel 'generate-and-rank' approach rapidly and semi-automatically generates data-driven multi-character interaction scenes from high-level graphical descriptions composed of simple clauses and phrases. From a database of captured motions, we generate a multitude of plausible candidate scenes. We then efficiently and intelligently rank these scenes in order to recommend a small but high-quality and diverse selection to the user. This set can then be refined by re-ranking or by generating alternatives to specific interactions. While our approach is applicable to any scenes that depict multi-character interactions, we demonstrate its efficacy for choreographing fighting scenes and evaluate it in terms of performance and the diversity and coverage of the results.

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

Keywords: Character Animation, Multi-Character Interaction, Animation Authoring, Motion Databases, Fight Scenes, Pre-visualization

1 Introduction

Written or verbal descriptions of action scenes for movies (e.g., a screenplay), television and theatre are often quite high-level and sparse. The scriptwriter, director or choreographer specifies only the key events in the scene and omits details such as the exact location, style and timing of each individual action. To aid communication and provide the full choreography, visual aids such as sketches, miniatures, and simple animations are often employed. Fight scenes are particularly difficult to choreograph as the timing and details of the events are critical to advancing the story in a convincing fashion and it is generally not possible to shoot many takes.

Our aim is to allow a user to rapidly and semi-automatically choreograph scenes with multiple characters interacting from sparse, high-level graphical descriptions, and to iteratively modify these scenes at interactive rates. The user's work process is thereby facilitated by efficiently and intelligently proposing several diverse and high quality scenes that satisfy the desired structure, which can be iteratively refined as required by generating new or modified candidate scenes with every iteration.

To achieve our goals, we present a novel *generate-and-rank* approach that takes as input a high-level scene specification, in the form of a graph built of sentence-like structures consisting of verbs, subjects, and objects. In practice, this graph could be manually built based on the instructions of a director or animator, or automatically extracted from a written text (e.g., a screenplay). From this specification we automatically generate a large number of plausible scenes from a database of annotated motion capture clips, and then efficiently rank these candidates in order to recommend a small and diverse selection of the highest quality scenes. For the purposes of this paper, we define a scene to be *plausible* if it matches all the elements of the specification (i.e., it 'satifies' the graph in that all the interactions are performed), whereas *quality* is a more subjective measure of how well the scene actually looks and/or meets the needs of the user.

The first problem to be addressed is therefore generating plausible candidate scenes and evaluating their quality based on prior information consisting of a set of metrics. In our implementation we evaluate priors such as how many individual clips are needed; how much they need to be edited in order to satisfy the description; how well the interactions are coordinated; and the extent to which interpenetrations between characters are avoided. The relative weights of these priors are set by the user based on their quality requirements. For example, a previsualization artist may care more about the overall staging of the scene and less about motion glitches or interpenetrations, whereas animators may require higher production standards. Additional quality measures could easily be added, such as saliency or other perceptual metrics.

The second challenge faced is how to rank the plausible scenes in order to select the subset of candidate scenes to present to the user. The most obvious consideration is of course quality, as defined above, but ranking based on this factor alone will not meet the needs of most users, as the set of highest quality results could all be very similar and therefore not provide enough choice. Therefore, three further closely-related criteria are equally important: the *diversity* of the selection; how much *coverage* is provided of the full space of possible solutions; and how representative each selected scene is of a larger cluster of scenes, i.e., its level of *similarity* to other, non-selected, scenes. Even though animated multi-character scenes are structurally quite different from webpages or images, the ranking approach is conceptually the same, in that prior information about individual items and the relationships between them are utilized to determine the order in which results are presented and thereby reduce redundancy. We therefore adapt some ideas from several modern ranking algorithms for our purposes [Brin and Page 1998; Jing and Baluja 2008; Mei et al. 2010].

Contributions: We support the creativity of the scene choreographer by allowing them a large degree of choice and flexibility in creating their desired scenario, while ensuring that the final result will contain consistent and high quality interactions between the characters. Our motion synthesis algorithms are, to our knowledge, the first to generate a fully-connected interaction scene from a sparse description. Our ranking approach, borrowing new ideas from other domains such as web and image searching, provides users with a diverse set of the highest quality options from which they can select and refine those that best suit their needs.

Our novel generate-and-rank approach comprises four technical innovations: a graphical model for high-level scene specification (Section 3); a probabilistic motion synthesis algorithm that analyzes the specification to understand the structure of multi-character interactions in individual scenes (Section 4); an efficient and intelligent scene ranking process (Section 5); and several refinement options that allow the user to have detailed control of the results (Section 6).

2 Related Work

There have been many previous systems that generate animations from high-level descriptions. Perlin's *Improv* system [Perlin and Goldberg 1996] creates actors that respond to users and to each other in realtime, using procedural animation and a rule-based behavior engine that takes as input English-style script describing the actors' communication and decision-making skills. Similar AI approaches have been proposed to generate group behaviours during social interactions and other scenarios: Pedica et al. [2010] model the territorial dynamics of social interactions; Yu and Terzopoulos [2007] simulate pedestrians in urban settings using hierarchical decision networks, while Funge et al. [1999] introduce the cognitive modeling language CML, to endow their autonomous characters with enough intelligence to work out a detailed sequence of actions to satisfy a brief behavior outline or "sketch plan". Most conceptually similar to our approach, Calvert and Ma's [1996] *Life Forms* system presents a selection of keyframed motion sequences from a large database based on a choreographer's sketch of a dance motion, and provides support for further editing using procedural animation and inverse kinematics. The authored dance sequences can then be viewed on multiple virtual characters. We derive inspiration from this approach in the design of our user interface.

With path planning methods, only a start and goal are provided for the character, and the system must automatically move them along the generated trajectory. Such motion controllers can be learned based on motion capture data [Lo and Zwicker 2008], and parameterized to correspond to specific motion skills using reinforcement learning [Levine et al. 2011]. Choi et al. [2011] generate and traverse a graph of motion capture fragments, which may be deformable in order to situate them into highly constrained virtual environments. Other sparse descriptions include simple motion sketches that are satisfied by efficiently searching through a motion graph and generating the interpolated motion trajectory [Safonova and Hodgins 2007], or a timeline painted with simple annotations selected from a domain-tailored vocabulary, again using optimized searching to deliver interactive authoring [Arikan et al. 2003]. These approaches do not however efficiently provide a high level of scene variations while satisfying the constraints imposed by multi-character interactions.

Another related body of work involves the automatic generation of conversational motions. The Behavior Expression Animation Toolkit (BEAT) takes as input a typed script and generates appropriate and synchronized nonverbal behaviors and synthesized speech [Cassell et al. 2001]. Stone et al. [2004], and later Levine et al. [2010], use a database of recorded speech (composed of short, clearly-delimited phrases), while Neff et al. [2008] analyzes video to create a statistical model of a specific performer's gesturing motions, and then generates full-body animation in the style of that individual for any novel input text.

A major requirement for our work is to provide a user with a wide variety of candidate scenes, from which they can intuitively select and refine. We have incorporated design elements from several relevant systems. Marks et al. [1997] introduced the concept of *Design Galleries*, which present the user with a broad selection of perceptually different graphics or animations by varying sets of parameters. The main criteria for their approach are dispersion (i.e., sampling the full set of possible outputs) and arrangement (i.e., providing an intuitive browsing and selection interface). Similarly, *Many-Worlds Browsing* exploits the speed of multibody simulators to compute numerous example simulations, and allows the user to browse and refine them interactively [Twigg and James 2007], while *Physics Storyboards* have been proposed to focus on key events and outcomes to accelerate the process of tuning interactive, procedural animations [Ha et al. 2013].

Relevant to our requirement to offer a variety of interactions that can be further refined, Agrawal et al. [2013] present an optimization framework for generating diverse variations of physics-based character motions, while Liu et al. [2010] use randomized sampling and forward dynamics simulation to reconstruct many variations on a given motion capture trajectory. Ye et al. [2012] also use randomized sampling for synthesizing a variety of physically simulated hand motions. Jain et al. [2009] present an interactive motion editing tool for creating dynamic scenes with human and object interaction, which allows the animator to directly change the trajectories of the human or objects and simultaneously render the effect of the edits on the entire scene. Optimization has also been used to synthesize more complex human behaviors, including cooperative actions involving multiple characters and their interactions with their environment and each other [Al Borno et al. 2013; Mordatch et al. 2012].

Focussing on the difficult problem of synthesizing multi-character interactions, Kim et al. [2009] uses Laplacian motion editing to allow the user to interactively manipulate synchronized multi-actor motions. Ho et al. [2010] use an *Interaction Mesh* to edit and retarget close interactions between body parts of single or multiple characters, while Shum et al. [2010] apply min-max searching to produce scenes of two character interactions to follow multiple story-lines. Game theory and motion graphs have also been used to control the motion of two characters in an adversarial game [Wampler et al. 2010]. The scope of motion synthesis algorithms based on min-max searching and adversarial games are often limited to interactions between two characters.

Many of scalable multi-character motion synthesis algorithms make use of *Motion Patches*. The original work by Lee et al. [2006] tackles the problem of capturing motion data for a large and complex virtual environment by constructing a set of building blocks that can be arbitrarily assembled to create novel environments. Each block is annotated with a *Motion Patch*, which describes the animations available for characters within that block. Even though the original work explores character animation in complex environments, its followup studies focus on creating a dense crowd of interacting characters. Crowd Patches involves a similar approach [Yersin et al. 2009] for large-scale urban scenes. Support has also been provided to simulate close interactions between characters, using tileable *Interaction Patches* [Shum et al. 2008]. However, fully connecting scattered interaction patches when the connectivity is not simple (i.e., with cycles) is difficult. Patch tiling by Kim et al [2012] and an MCMC method by Hyun et al [2013] generate fully-connected (spatially coordinated and temporally synchronized) interaction scenes, with the focus on removing all dead-end connections. Resulting scenes tended to be random rather than controlled and detailed scene descriptions could not be satisfied.

3 Graphical Scene Description

The goal of our authoring system is to generate a variety of action scenes based on a high-level written or verbal description (e.g., Figure 2a), which should specify the individual motions of the characters, what events occur, and how the characters interact with each other. We provide a graphical user interface to design a diagram of the scene description, called an *event graph* (Figure 2c(i)), which exploits subject-verb-object structures to describe events. (While we do not currently perform any natural language processing to automatically create the specification from a written script, this would be a useful enhancement.) In this event graph, actions are represented by *event nodes* if those actions can be described using a clause with a verb, one or more subjects, and possibly objects and phrases. The directional edges of the graph describe the temporal sequencing of events and the transitions of an character from one event to the other. Each event node can have multiple incoming and outgoing edges that correspond to the characters' actions; either both characters are actively involved, or one is *active* (red arrows in event nodes) and the other *responsive* (blue arrows in event nodes). We currently only allow each event to involve one or two characters for ease of implementation and clarity of presentation, and leave the case of event nodes depicting simultaneous interactions between three or more characters for future work.

To provide high quality, data-driven animation of the events described in the scene, a comprehensive database of suitable motions should be constructed. For the fighting scenarios we use as a demonstration of our approach, we captured one hour's worth of data from professional stunt-men and actors (Figure 2b). Motion capture sessions were conducted following standard procedures. We provided twelve high-level themes (e.g., men fighting over a woman, burglar in house, classroom bully, big vs. small guy.) to the actors, which they interpreted and choreographed themselves and also performed some free-style acting/stunt scenarios. Any general or other domain-specific databases could also be used as long as they contain a sufficient variety of relevant interactions. We manually label the motions of each individual motion captured actor using an appropriate vocabulary (though this process could be automated using existing motion classification approaches) consisting of verbs (e.g., bump, push), adverbs (e.g., gently) and phrases (e.g., fall-down). A single motion clip for an actor may have multiple labels (e.g., slap, gently, on-the-face). From viewing the original interaction pairs, the constituent motion clips are labelled appropriately (e.g., slap-active, slap-responsive) as it is possible to pair an active motion with a responsive one that was not simultaneously captured (Partner Changing in Section 6). We also identified physical contacts between body parts (e.g., fist on opponent's face) and the environment (e.g., foot on ground) in a semi-automatic manner and added this information to each motion clip's annotation. Initially, thresholding on distance and relative velocity are used to detect contacts, for which we rectify any false detection manually. More general motion clips (e.g., wandering) are labeled as 'Idle', and are used to make connecting paths between scene events. The plausibility of these 'filler' motions could easily be enhanced by adding new labels to refine the annotations.

From the labelled data we build a *motion graph*, which describes the connectivity between motion clips in the database [Lee et al. 2002]. Traversing the graph allows longer motion sequences to be generated. In Figure 2c(i), the character John follows a sequence of event nodes (Walk, Bump, Point, Push, Punch, Kick). Motion clips with an action verb are called *event clips* and can be associated with an event node that has the same verb label. All plausible motion sequences for John should therefore begin with a Walk event clip, followed by the other event clips in order. Each pair of subsequent event clips may be connected through a sequence of Idle motion clips. From the event clips and their rich connectivity, a large number of plausible motion sequences for each scene character can be generated that satisfy the event graph with respect to label matching. Combinations of these motion sequences for all scene characters constitute a plausible scene if their relative position, direction and timing match when interactions and physical contacts occur. A summary of some definitions and notations we use in this and subsequent sections is provided in Table 1.

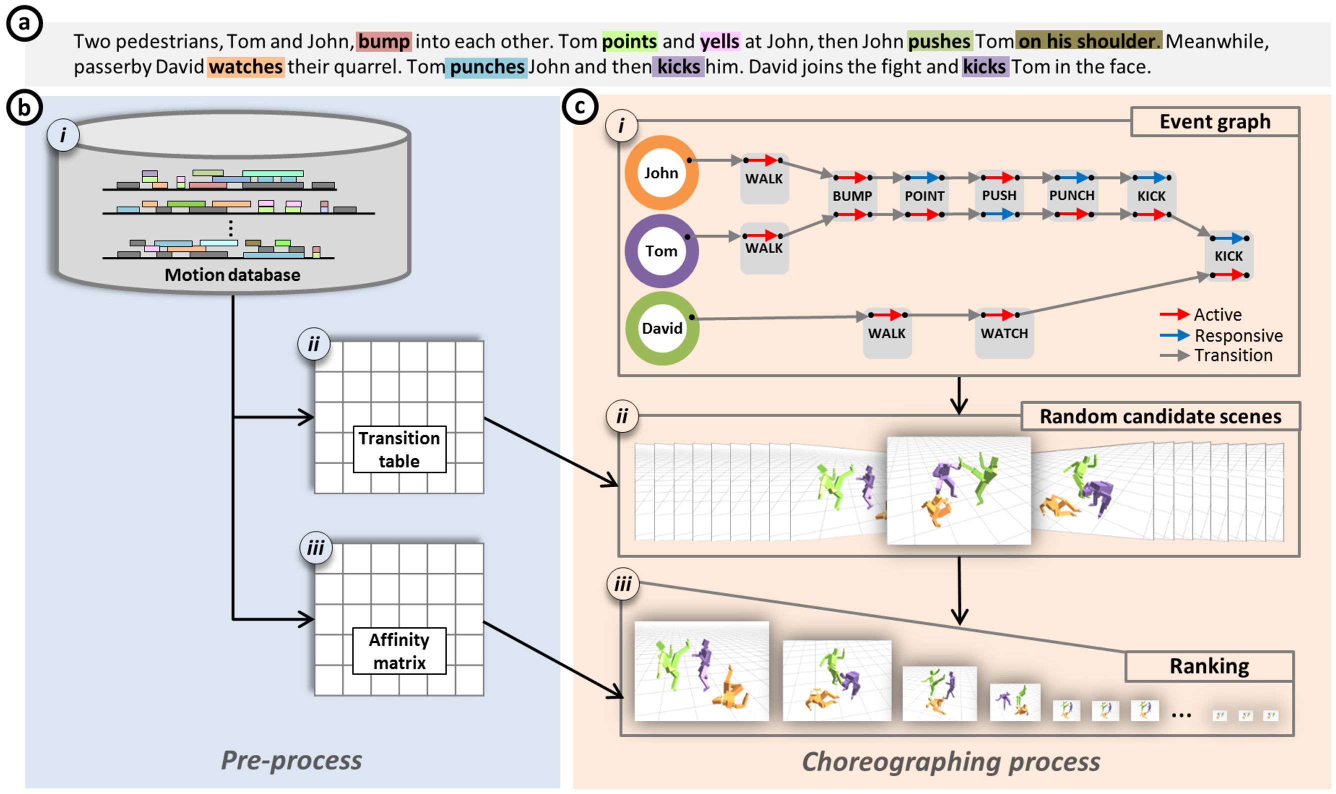
Two pedestrians, Tom and John, bump into each other. Tom points and yells at John, then John pushes Tom on his shoulder. Meanwhile, passerby David watches their quarrel. Tom punches John and then kicks him. David joins the fight and kicks Tom in the face.

Figure 2: *Overview. (a) The text script of a fight scene. Action verbs and phrases are highlighted. (b) Our motion databases have its frames labeled using our system vocabulary. We also pre-compute a collection of random connecting paths for every pair of action verbs to build a* Transition Table *(ii), and an* Affinity Matrix *(iii) to encode the similarity between all pairs of motion clips. (c) At runtime, the user designs an event graph that matches the text script. Our system generates a collection of random candidate scenes from the event graph and ranks them to present a few top-ranked plausible scenes to the user.*

Name	**Description**
Character c_i	scene character
Event Node n_i	single- or multi-character event
Event Graph	connects event nodes to make scene
Event Clip: e_i	motion clip associated with event node
Event Path: $p_{i,j}$	sequence of event nodes between n_i and n_j
Cycle: $c(p_i, p_j)$	when two paths share start and end nodes
Transforms: T	move **a**ctive and **r**esponsive characters
T_i^a, T_i^r	move a and r characters *through* event clip e_i
$T_{i,j}^a, T_{i,j}^r$	move a and r *between* event clips e_i and e_j
T_i^{ar}, T_i^{ra}	a and r's relative position and orientation in clip e_i
$t_i^a, t_{i,j}^a, t_i^r, t_{i,j}^r$	associated times
Bodies: B	body coordinate systems and parts
B_i^k	coordinate system of character c_i at frame k
$b_i^{k,m}$	position of c_i's body part m at frame k
t_i^k	associated time

Table 1: *Definitions and Notation*

4 Candidate Scene Generation

The successful application of our *generate-and-rank* approach relies on our ability to generate a large collection of plausible candidate scenes. Naïve random walks in the motion graph rarely generate convincing motions because multiple characters will not appear coordinated and too many transitions can result in low-quality motion sequences. In this section, we present a new algorithm to produce a variety of multi-character motion sequences that satisfy the event graph.

Our synthesis algorithm is probabilistic and thus we cannot guarantee that each individual scene generated by our algorithm is always plausible with respect to all conditions and requirements. We present two ideas to alleviate the problem. The first idea is our cycle analysis of the event graph, which allows us to pick plausible scenes with high probability. Secondly, many candidate scenes are generated in the *generate-and-rank* framework and top-ranked scenes among them are highly likely to be plausible within error tolerance.

Connecting Paths: Each event node can be associated with many event clips (Figure 3). Pairs of (active, responsive) event clips are associated with each event node involving two characters. Consider event clips e_i and e_j associated with two sequential event nodes. A motion path between e_i and e_j through the 'Idle' part of the motion graph creates a seamless sequence of connected motions. There may be many such motion paths available between any two event clips, with each path requiring different amounts of body transla-

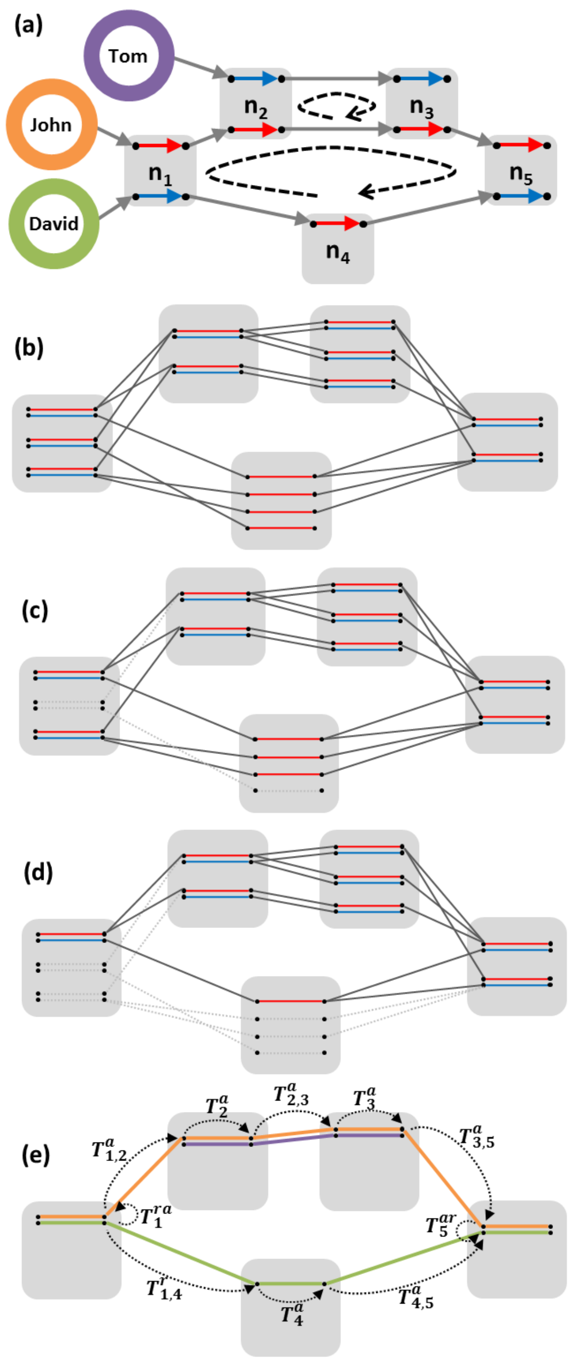

Figure 3: *Scene Generation: (a) a simple event graph and (b) its motion cascade. We prune event clips that (c) have no outgoing branches and (d) are not cycle-feasible. (e) a candidate scene instantiated from the motion cascade.*

tion, rotation, and time.

We pre-compute a collection of connecting paths for each pair of event clips to build a transition table (Figure 2b(ii)), the columns and rows of which respectively correspond to sources and destinations of transitions between event clips. Each transition (i, j) contains a variety of paths from e_i to e_j that complete within a certain time limit, or is empty if the shortest path takes too long. In our experiments, the time limit is set to four seconds, and we picked up to 300 paths for each transition (i, j). This collection is expected to cover variations in body translation, rotation and timing while performing the transitional motions from e_i to e_j. We also require each individual path to have as few branching transitions through the motion graph as possible, so we order connecting paths by increasing number of branches and picked paths with fewer branches first. In this way, most paths we generated have only a few branches, thereby minimizing jerky motions.

Motion Cascade: Many event clips and their rich connectivity form a underlying structure that is embedded in an event graph. We call this structure a *motion cascade*. In Figure 3, we show a simple event graph (a) and its embedded motion cascade (b) for illustration. Event node n_4 is a single-character node, while the others involve two characters, one of whom is active (in red) and the other responsive (in blue). The edges are solid black if there are any pre-computed paths available between event clips. Even through we draw the edge as a single line, many connecting paths might be available. Given a motion cascade, an instance of a multi-character scene is generated by picking an event clip e_i for each individual node n_i (or a pair of active-responsive event clips for a two-character node) and choosing a connecting path for each individual pair of subsequent event clips.

A sequence of event nodes in the event graph forms an *event path* p_i and, if two such paths share the same start and end event nodes, they form a *cycle* $c(p_i, p_j)$. John (orange) and David (green) traverse paths $p_1 = (n_1, n_2, n_3, n_5)$ and $p_2 = (n_1, n_4, n_5)$, respectively, and their paths coincide at n_1 and n_5 to form a cycle. This means that they interact for event n_1, go their separate ways for other events, and then meet again for event n_5. A pair of active-responsive event clips at the start of a cycle is *cycle-feasible* if they both have connecting paths to the event clips of another pair at the end.

Figure 3 also shows that some event clips are 'dead-ends', i.e., either one or both characters have no outgoing branches (c), or they are part of a pair that is not cycle-feasible (d). Such action clips should be pruned before we begin to instantiate our candidate scenes. Whenever any event clip is pruned, we trace its path forwards and backwards to remove any further inconsistencies (e.g., see node n_1 in (c), where the edges of these pruned paths are dotted grey). This process could also introduce a potential dead-end, so building a scene is an iterative process of repeatedly picking event clips and connecting paths one by one from the motion cascade and pruning potential dead-ends.

Motion Selection For Each Cycle: The motion cascade has the potential to produce a very large number of combinations of motion sequences that constitute plausible scenes. However, instantiating one as a candidate is a non-trivial task if the connectivity of the event graph is complex. We first discuss how to generate plausible motion sequences along a single cycle and the full version of the algorithm will be presented later in this section.

We define a set of *transforms* for each event clip. T^a_i and T^r_i are 3×3 homogeneous matrices describing two-dimensional translation on the ground plane and rotation about the vertical axis, which bring the body of the active and responsive characters respectively from the beginning of event clip e_i ~~through~~ to its end, while $T^a_{(i,j)}$ and $T^r_{(i,j)}$ move the characters *between* event clips. At the start of a pair of active-responsive motions, the relative position and orientation of the interacting characters are represented by the matrix T^{ar}. Similarly, at the end of the active-responsive pair, the relative position and orientation of the interacting characters are represented by the matrix T^{ra}. Times $t^a_i, t^r_i, t^a_{i,j}, t^r_{i,j}$ denote the associated lengths of the aforementioned event clips and connecting paths in time.

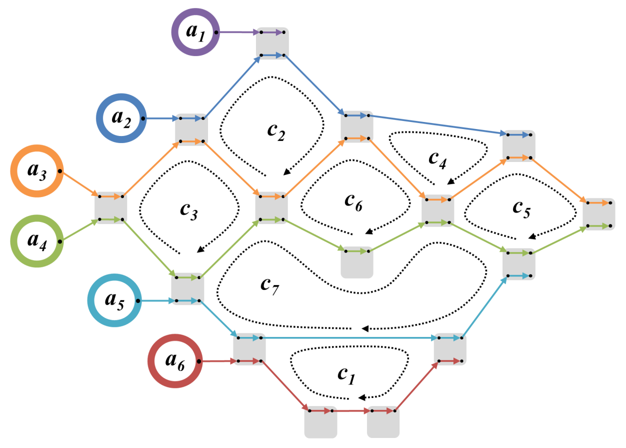

Figure 4: *Cycle ordering*

Therefore, given the cycle for John and David, Equation (1) and Equation (2) respectively impose spatial coordination and temporal synchronization between motion sequences for the scene instance in Figure 3(e)).

$$T_1^{ra}\, T_{1,2}^a\, T_2^a\, T_{2,3}^a\, T_3^a\, T_{3,5}^a\, T_5^{ar} = T_{1,4}^r\, T_4^a\, T_{4,5}^a \tag{1}$$
$$t_{1,2}^a + t_2^a + t_{2,3}^a + t_3^a + t_{3,5}^a = t_{1,4}^r + t_4^a + t_{4,5}^a \tag{2}$$

~~**Motion sampling:**~~ Randomly picking event clips and connecting paths among available options rarely meets the cycle conditions in Equation (1) and Equation (2). We sample many such motion sequences along two adjoining paths and choose the best pair among them. The best pair of motion sequences thus obtained are highly likely to be plausible within error tolerance if sufficiently many samples are picked and examined. In our experiments, we picked 10,000 random samples for each cycle. The spatiotemporal error is measured using

$$\xi = \|\mathbf{v}' - \mathbf{v}\| + w_a|\theta' - \theta| + w_t|\delta' - \delta| \tag{3}$$

where $\mathbf{v} \in R^2$ and $\theta \in R$, respectively, are the translation and rotation components of the left side of Equation (1), $\delta \in R$ is the left-hand side of Equation (2), $\mathbf{v}'$, θ', δ' are from the right-hand side of the respective equations, and w_a, w_t are weights. In our experiments, we determined the weights $w_a = 1.12$ and $w_t = 0.38$ such that the averages of translation, rotation, and time of all available event clips and connecting paths are normalized.

If one of two adjoining paths is much longer than the other (i.e., more event nodes along the path), we may not be able to find any plausible motion sequences that match, because our system limits the lengths of connecting paths. The solution we use is to insert *pseudo event nodes*, where the character idles for a short while in order to fill in the gaps of our sparse description. In this way, we can generate arbitrarily long connections to match adjoining paths, or to create idling behaviors at the beginning and end of the scene if no description is provided for certain characters.

Cycle Ordering: The event graph, in general, has a number of nested, adjacent cycles, which should be visited in an appropriate order. Consider the event graph in Figure 4. If we select the motions of a_2 and a_3 first to form two cycles c_2 and c_4, and the motions of a_4 and a_5 later to form another cycle c_7, then the motions of the third character pair a_3 and a_4 would have already been decided before we can examine whether their motions are well matched around middle cycle c_6. When we visit each individual cycle, at least one side of the cycle should remain undecided so that we retain the freedom to coordinate interactions between characters.

Algorithm 1 Scene generation from a motion cascade

```
    // Preprocessing
1:  Build a table of connecting paths

    // At runtime
2:  Decide the order of cycles in the event graph
3:  for # of candidate scenes do
4:      for each cycle do
5:          Prune infeasible event clips
6:          Pick motion sequences for a free path
7:      end for
8:      Motion editing and time warping
9:  end for
```

Algorithm 1 shows the overall process for producing a candidate scene, and we now discuss the cycle ordering in Line 2. Our cycle ordering algorithm chooses the last order cycle first and eliminates it from the event graph, then repeat this until only one remains, which will be the first order cycle. In Figure 4, it removes cycles from c1 to c7, while our motion synthesis algorithm will visit them backward from c7 to c1 (Line 4 to 7). We choose a cycle at each iteration as follows: we pick any simple event path $p_{i,j}$ between event nodes n_i and n_j. A simple event path is a path that is only composed of two degrees internal nodes. If there exists an alternative path $p'_{i,j}$ between the two event nodes, then p and p' form a candidate cycle and there exist another cycles which share p' with it except when only one cycle remains. Among many candidate cycles, we pick the one with the shortest shared path and eliminate p at each iteration. Whenever the motion synthesis algorithm visits a new cycle, its p' is already decided and p remains undecided. For example, in Figure 4, the path traversed by a_6 forms cycle c_1 with a_5's path. Cycle c_1 will be picked first because it has the shortest shared path of length 1. Then we eliminate a_6's path from the graph and repeat the algorithm until we have no more paths to eliminate. This algorithm can guarantee that there will be a *free path*, for which motion sequences are not determined yet, whenever a new cycle is visited.

Generalized Paths and Cycles: The definitions of event paths and cycles generalize further in two respects. First, a single event path can be traversed by multiple characters in relay. Secondly, a path can be traversed backwards in time. This assumption means that the event graph can be undirected, and we do not need two paths to define a cycle anymore. A generalized cycle can be defined as a single path that traverses the event graph and returns to its start node. Even though this generalization may seem extreme, our algorithm readily works with generalized paths and cycles without major modification, thus allowing the algorithm to deal with a wider range of graph configurations that it otherwise could not handle. Figure 5 shows such a case, which does not have two adjoining forward paths, but does have a generalized cycle. The transformations along such a cycle are inter-personal when characters take turns, and inverted when backward edges are chosen, which should result in the identity transformation. The generalized cycle in Figure 5 poses spatial and temporal conditions as follows:

$$T_1^{ra}\, T_{1,2}^a\, T_2^{ar}\, (T_{3,2}^a)^{-1} (T_3^{ra})^{-1} T_{3,4}^r (T_4^{ar})^{-1} (T_{1,4}^r)^{-1} = I \tag{4}$$
$$t_{1,2}^a - t_{3,2}^a + t_{3,4}^r - t_{1,4}^a = 0 \tag{5}$$

With these generalized paths and cycles, we can guarantee that a collection of motion sequences constitute a *plausible* scene if the motion sequences are *plausible* along individual cycles. This property allows us to account for the complex connectivity of the

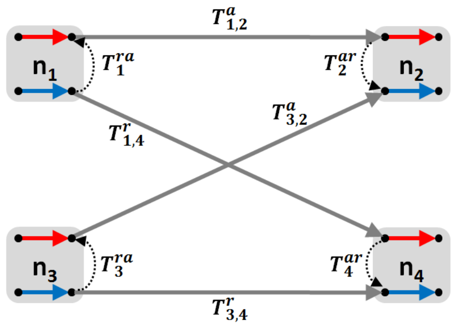

Figure 5: *A generalized cycle with backwards traversal*

event graph simply by examining individual cycles one-by-one, and therefore makes our algorithm simple, efficient, and easy to implement.

Motion Editing: Even though interactions are carefully coordinated via cycle analysis, there will still be mismatches in the characters' positions and timings. We use motion editing and time warping to rectify these residual artifacts. Consider a scene consisting of individual characters' motions, each of which is a seamless sequence of motion clips. Interpersonal constraints are defined at motion frames in which interactions and physical contacts occur. Let the indices i and j denote characters, k and l frames~~, and m and n body parts~~. Let B_i^k be the body-attached coordinated system and t_i^k is the time of character i at frame k ~~and $b_i^{k,m}$ be the position of his body part m~~. Each interaction applies three constraints on the body, body parts and time respectively:

$$(B_i^k)^{-1}B_j^l\left((\tilde{B}_i^k)^{-1}\tilde{B}_j^l\right)^{-1} = I \tag{6}$$

$$t_i^k - t_j^l = 0 \tag{7}$$

$$\|b_i^{k,m} - b_j^{l,n}\| - \|\tilde{b}_i^{k,m} - \tilde{b}_j^{l,n}\| = 0 \tag{8}$$

where the tilde indicates the reference values measured from the original motion capture data. The first constraint ensures that the relative position and orientation between characters are preserved ~~, and the second preserves the relative position of end-effectors (e.g., the fist and the face when one character punches the face of another)~~. The ~~third~~second equation favors precise synchronization of an action and its response. We employ Laplacian motion editing by Kim et al. [2009] to solve this constrained optimization problem, as it makes smooth changes on a collection of motions to meet the constraints, while minimizing motion deformation.

5 Scene Ranking

The plausibility or quality of an animated scene can be quite subjective, depending on the target application and/or the viewer's personal goals and preferences. We take inspiration from webpage and image ranking research and apply some of those concepts to the problem of ranking our choreographed scenes. A common theme that underpins modern ranking algorithms is the characterization of relationships between the items to be ranked. The PageRank algorithm originally exploited in the Google search engine uses hyperlinks between webpages to build a ranking graph [Brin and Page 1998]. A webpage is considered important if it has many incoming hyperlinks from important webpages. The algorithm determines the ranking of webpages, which correspond to graph nodes, based on prior information (any immediate measure of ranking that might not be fully reliable) and propagation of prior ranking across hyperlinks.

Although images do not have explicit links between them, Jing and Baluja [2008] proposed the VisualRank algorithm that creates a link structure among images based on image similarity metrics. An image is therefore considered important if it is similar to important images. We adopt this idea to construct a ranking graph of animated scenes. The similarity between scenes serves as edge weights, and the prior ranking of each individual scene is computed based on our plausibility measure. Propagating prior ranking across visual similarity links results in top-ranked scenes being at the centre of the overall distribution.

Yet another factor we have to consider is the diversity of top-ranked scenes. Unlike images, it is difficult to quickly browse through many candidate scenes, so only a few scenes can be suggested to the user at each iteration of the choreography cycle. Therefore, it is important to have a few top results that are visually different from each other. We employ the idea of diversity ranking [Mei et al. 2010] that pursues a balance between centrality and diversity in the top ranked results, and also ensures that the full space of candidate scenes is sampled. We can see from Figure 6 that the top ten scenes selected using our algorithm based on diversity ranking provides high diversity, as they are less similar to each other; and coverage, as the candidate scenes are sampled more evenly throughout the space of plausible solutions.

In order to apply these ideas from modern ranking algorithms, we need some metrics to efficiently evaluate the plausibility and similarity of scenes to facilitate interactive work flow. We therefore define the plausibility P of the scene as the weighted sum of five metrics. Motion editing with multiple actors often involves a trade-off between the degree of deformation and the accuracy of constraint satisfaction. Although allowing large deformations would satisfy all constraints precisely, we often want to limit the degree of deformation to achieve better quality of motion while allowing small residual mismatches in constraints. P_{deform} measures the degree of editing needed to fit motions together for multi-actor coordination, which is calculated using a weighted sum of Laplacian deformation energies for spatial and temporal warping, calculated during motion editing [Kim et al. 2009]. P_{residual} is the weighted sum of residuals in Equations (6)–(8) and P_{col} penalizes collisions and inter-penetrations between actors. The diversity metric P_{div} penalizes multiple occurrences of identical actions, because viewers often respond negatively when they spot exactly same actions appearing repeatedly in a single scene. Finally, P_{pref} allows the animator to directly specify his or her preferences on individual actions. In our experiments, the weight values are 2.0 and 0.5 for the spatial and temporal components of both P_{deform} and P_{residual}, 3.0 for collision, 1.0 for diversity, and 0.0 for user preference.

The animated scene consists of important actions and responses, with more neutral connections between these events. Our similarity measure compares actions and connections at different levels of detail. The similarity between two scenes $\mathcal{S}$, $\mathcal{S}'$ is formulated as follows:

$$\begin{aligned}\text{dist}(\mathcal{S},\mathcal{S}') = &\frac{\sum_i \text{dist}(e_i, e_i')}{\text{(\# of event nodes)}} + w_p \frac{\sum_j \text{dist}(p_j, p_j')}{\text{(\# of event edges)}} + \\ &w_s\left(1 - \frac{|E(\mathcal{S}) \cap E(\mathcal{S}')|}{|E(\mathcal{S}) \cup E(\mathcal{S}')|}\right)\end{aligned} \tag{9}$$

$$\text{similarity}(\mathcal{S},\mathcal{S}') = \frac{1}{\text{dist}(\mathcal{S},\mathcal{S}') + \epsilon} \tag{10}$$

where $\text{dist}(e_i, e_i')$ and $\text{dist}(p_j, p_j')$ are the dissimilarities between two event clips and two connecting paths respectively. For the

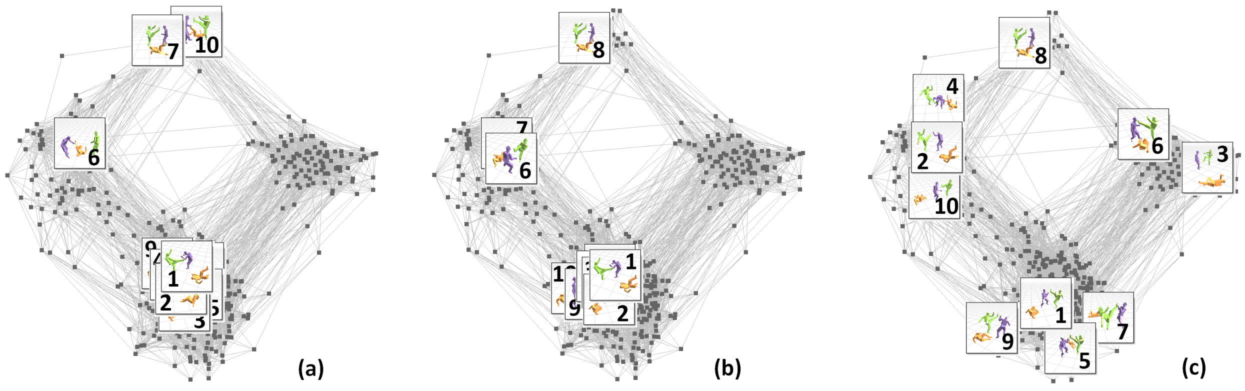

Figure 6: *Scene ranking. Dimensionality reduction by MDS (multi-dimensional scaling) depicts a collection of 200 scenes in a two-dimensional plane. The Top-10 ranked scenes were chosen based on (a) quality only, (b) PageRank + visual similarity, and (c) our algorithm based on diversity ranking, which demonstrates better diversity and coverage than both others.*

computation of action similarity, we use dynamic time warping to align motions in time. The dissimilarity between individual poses is computed based on 39 binary features, suggested by Müller et al. [2006]. We compute the similarity between all pairs of event clips to construct an *affinity matrix* during a preprocessing phase (Figure 2b(iii)). For connecting paths, a detailed comparison of full-body poses and time alignment is not helpful. Instead, we only compare their spatial transformations T and the length in time using Equation (3). The third term indicates the duplication of event clips in two scenes we are comparing. Scene S is composed of a set of event clips $E(\mathcal{S})$ and $|E(\mathcal{S})|$ is its cardinality. If a set of event clips appear in both $E(\mathcal{S})$ and $E(\mathcal{S}')$ in different orders, these two scenes would be perceptually very similar to each other, but the dissimilarity term would not recognize their similarity. The third term compares the duplication of event clips regardless of their ordering.

6 Scene Refinement

The ability to refine the top ranked results is an essential component of our generate-and-rank approach, which allows the scene choreographer to have immediate and detailed control over the results. The user is provided with a range of refinement options through an appropriate user interface. The computation cost for the refinement varies depending on how deep a dive into the hierarchy of the process flow is needed to execute it.

Interactive Manipulation: The user may often find highly ranked scenes to be satisfactory except for small glitches that can be easily fixed. He may want to remove collisions between characters, require an character to face a particular direction or to reach a particular location at a particular time. Any scene generated from our system is readily annotated with interaction constraints. Therefore, we can use Laplacian motion editing while maintaining the integrity of multi-character coordination, which is the most immediate and direct type of refinement we facilitate.

Re-ranking and Re-generation: The user can adjust weights of rank metrics, which are then reflected in the next round of ranking. Re-ranking candidate scenes can be done quickly in about one second. A more expensive option is re-generation, which involves either a change to the event graph, or adding a new set of motion data and labels. In the former case the motion cascade should be rebuilt to generate a new set of candidate scenes, which takes just a few minutes for simpler examples and may take up to an hour for the largest example we tested. The latter, more extreme, change requires the motion database to be updated from scratch, which can take a few hours.

Action Replacement: Given any scene, we might wish to replace a particular event clip (or a pair of active-responsive clips) with another, while leaving the remaining scene intact. The new event clips at an event node will be connected to the remaining part of the scene by choosing appropriate connecting paths along incident event edges. A brute-force approach examines all possible combinations, which takes $O(N^p)$ computation time, where N is the number of available connecting paths and p is the number of (both incoming and outgoing) event edges. If the event node has two characters, action replacement takes $O(N^4)$ time. We suggest a more efficient algorithm of $O(pN^3)$ time complexity. For simplicity of algorithm description, we assume that a pair of active-responsive event clips in a two-character event node is replaced with another pair (a, r), so $p = 4$. From earlier, T^a and T^r are the associated transformations, while T^{ar} and T^{ra} are the inter-personal transformations between the partners at the start and end of the event clips (Figure 7). Characters a and r are supposed to be connected to the remaining part of the scene through four connecting paths. Homogeneous matrix C_i for $1 \le i \le 4$ denotes the position and orientation at the end of a connecting path, where it should be incident with either a or r. We need to choose four connecting paths such that the error E_{connect}, i.e., the sum of misalignment distances, is minimized:

$$\begin{aligned} E_{\text{connect}} = \; & \text{dist}(C_1^{-1}C_2, T^{ar}) + \text{dist}(C_2^{-1}C_3, T^{r}) + \\ & \text{dist}(C_3^{-1}C_4, T^{ra}) + \text{dist}(C_4^{-1}C_1, (T^{a})^{-1}) \quad (11) \end{aligned}$$

Our algorithm is based on dynamic programming. Assuming that we first choose the n-th connecting path for C_1, deciding the other three paths requires the construction of a table of $p \times N$ fitness values, by solving the recurrence equations:

$$V(1, j) = \text{cost}(1, n, j) \quad (12)$$
$$V(i, j) = \min_k V(i-1, k) + \text{cost}(i, k, j) \quad (13)$$

where $\text{cost}(i, k, j)$ is the misalignment of choosing the k-th connecting path for C_i and j-th connecting path for $C_{(i+1) \bmod 4}$, as-

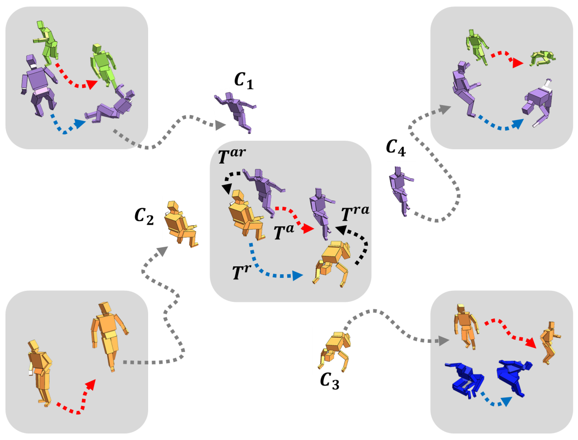

Figure 7: *Action replacement*

suming that the preceding connecting paths have been chosen optimally. $V(i, j)$ is the accumulated misalignment of choosing j-th connecting path for $(i + 1)$-th event edge assuming that previous choices of connecting paths are optimal. Because of the cyclic ordering of the event edges, index i is modulo 4. The table entries are filled based on dynamic programming. Backwards tracing from $V(4, n)$ identifies a cyclic path through the table. We repeat the dynamic programming for each n to examine all possible combinations and choose the best one that minimizes the error E_{connect} in Equation (11). Even though we only explained about spatial coordination of connecting paths, temporal synchronization is also considered to compute E_{connect}. So, the *dist* function of Equation (11) is computed by Equation (3) using spatial and temporal error. In practice, the brute-force $O(N^4)$ algorithm takes about 10 seconds to find the optimal set of connecting paths, while our algorithm takes less than one second to achieve an order of magnitude performance gain.

Partner Changing: Even though the motion cascade readily provides very many candidate scenes, sometimes richer variability is required for a specific event node, for which a limited number of event clips are available. Partner Changing is the process of combining active and responsive motion clips captured separately in order to enrich variability. Consider two pairs of action-response clips (a, r) and (a', r'). Each pair has inter-personal transformations at some frames in which interactions or physical contacts occur. Each 3×3 homogeneous transformation describes the relative position and orientation of two interacting characters. Partner changing generates new crossing pairs (a, r') and (a', r), if there exists a correspondence between interactions in the two original pairs, while corresponding transformations are similar within user-specified thresholds. The averages of corresponding transformations serve as interaction constraints for the new crossing pairs. Laplacian motion editing with halfway constraints solves for motion deformations that match crossing pairs well.

7 Results

We demonstrate the power and scalability of our generate-and-rank approach through a variety of examples. Figure 8 shows the event graphs for additional examples. Each example has a story of events.

- **Payback:** Three guys are sitting on the ground, stretching and exercising. Another guy bugs and irritates them repeatedly. They all stand up and pay him back for the irritation.
- **Tournament:** Eight people fight in an elimination tournament. The winner goes through to the next round, while the loser falls down and stays on the ground. The final winner cheers for victory.
- **Movie:** We recreate a scene based on a fight sequence from an actual movie (Snatch, ©Columbia Pictures, 2000), where two guys point, yell, punch, kick, grab, and throw each other. Although all our motions had been captured with no reference to this scene, our system was able to emulate the original sequence very well.
- **Random fight:** Twenty actors fight randomly with each other (Figure 9). This example demonstrates the scalability of our approach: the event graph includes 268 events and 320 edges, and our algorithm identified 52 cycles in the graph.

Our system generates 1000 candidate scenes for each example. Each candidate scene is between 30 to 80 seconds long and has a very high-dimensional representation (scene-time $\times$ frames per second $\times$ number of actors $\times$ degrees of freedom per pose). Multi-dimensional scaling (MDS) allows us to visualize the level of similarity in our high-dimensional data. As Figure 8 shows, each scene has a clustered distribution. The top ten results selected using quality only and PageRank with visual similarity links tend to be quite similar and do not offer much coverage of all clusters. Our algorithm based on diversity ranking, however, always selects a highly relevant yet diverse set of scenes for the top ranked results, which broadly cover the space of candidate scenes.

Performance statistics are measured on a desktop PC equipped with an Intel Xeon CPU E5-2680 (8 cores, 2.7 GHz) and 32GB main memory, except for the random fight example. Creating such a large scene is memory intensive, so we computed the random fight example on another machine with four processors of an Intel Xeon CPU E7-4870 (2.4 GHz) and 1TB memory. Our database consists of 78,502 frames of motion data cleaned up and labeled. The motion graph constructed from the database includes 70,861 frames (about 40 minutes long) of deadend-free, strongly-connected components. The motion database has 25 verb labels and 20 phrase labels (Table 2). In the preprocessing phase, the construction of a motion graph, an affinity matrix, and a table of connecting paths took 5.7, 10.6, and 88.5 minutes, respectively. Therefore, rebuilding these structures from scratch takes about 105 minutes of computation in total.

The runtime computation for each example is summarized in Table 3. The units are in meters for distance, radians for angle, and seconds for time, unless otherwise specified. The breakdown of the runtime computation shows that motion editing and motion selection around each individual cycle are the most time-consuming components. Kim et al [2009] suggested an acceleration technique based on frame sub-sampling, which we have not yet incorporated into our system, but which we expect will deliver an order of magnitude improvement in performance. The computation time for motion selection depends on the number of samples we pick for each individual cycle. In principle, we have to test more samples for longer cycles since they may provide more diversity of motion choices. Table 4 shows how the number of samples may affect the motion quality for the three-person example. The quality improves as the number of samples increases and the improvement plateaus at about 10,000 to 20,000 samples per cycle. Currently, we pick 10,000 samples per cycle regardless of its length, and there are opportunities for further improvement by picking samples adaptively.

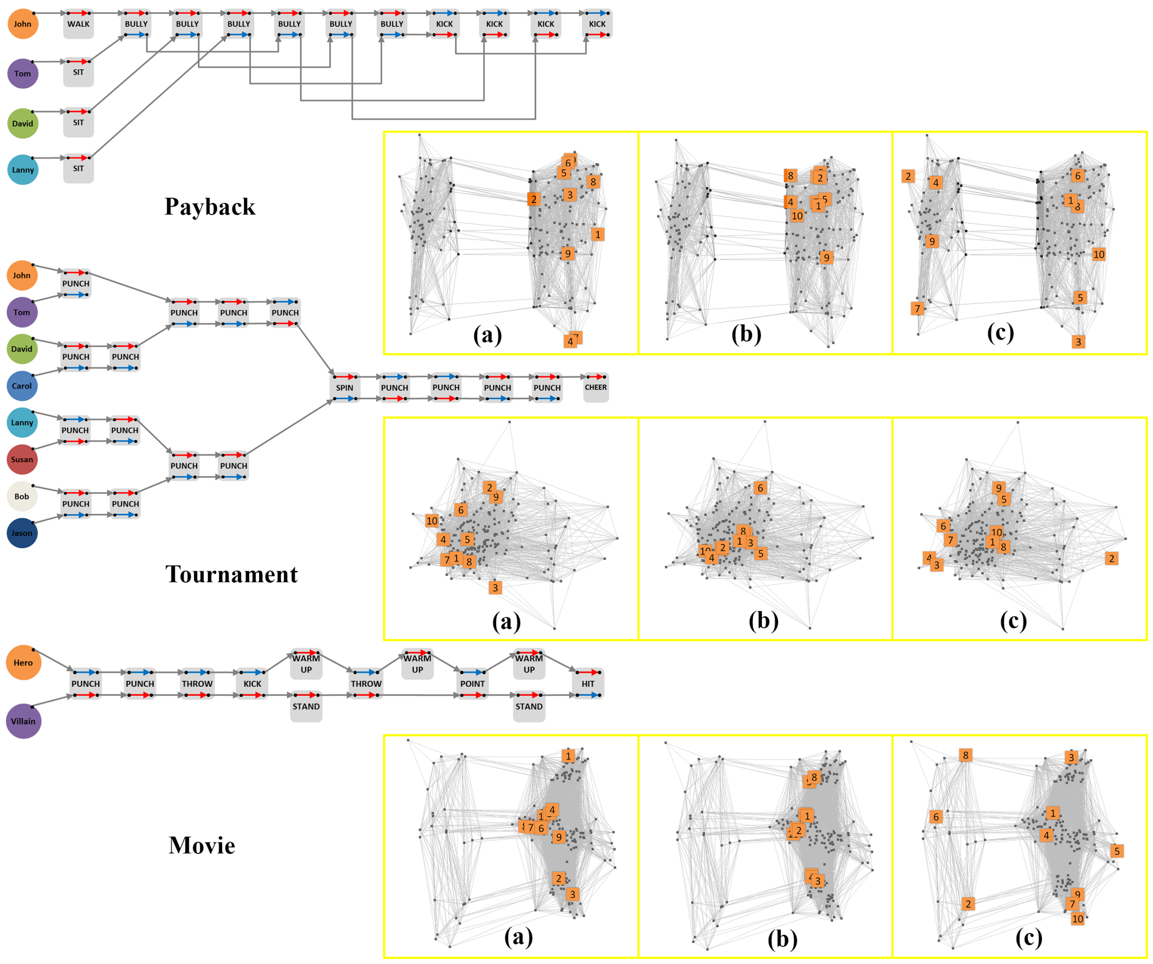

Figure 8: *The event graph and MDS visualizations of examples. Top ten results are selected by (a) quality only, (b) PageRank + visual similarity, and (c) our algorithm.*

8 Conclusions and Future Work

Even though our focus has been on choreographing fight scenes in this paper, our approach could be easily extended to deal with other types of scenes where there is a requirement for the coordination of multiple interacting actors, such as dancing, sports or social interactions. We focused on fighting scenes because they are particularly difficult to synthesize and therefore present a significant challenge. The problem becomes simpler the fewer physical contacts there are between the actors. Instead, other factors, such as facial expression, lip synch, gaze direction and gesture, would become more important. These factors have been the subject of many studies and are highly complementary to and compatible with our approach. Potential application areas include pre-visualization of action scenes for movies or TV, interactive storytelling, crime scene reenactments, and choreography of ensemble dances or ballet duets. The key challenge would be the effort required to build domain-specific motion databases and system vocabularies.

Our system currently can handle up to two actors for each event. This limitation is not inherent to our approach, but was rather chosen for convenience of system implementation and clarity of exposition. In principle, our algorithms can readily handle more general forms of events with arbitrarily many subjects and objects. However, the diversity of solutions and therefore the level of control afforded the choreographer will be reduced when smaller two person events are so tightly synchronized as to be combined into one bigger event.

One limitation of our approach is the handling of causal chains. Consider the script: 'Jack punches Tom. Tom dodges and kicks him back.' where the punch event is the direct cause of the dodge event and both events occur almost simultaneously. The dodge event causes another kick event immediately afterwards to form a chain of causality, which could generate very dense interactions. It also does not allow room for extended connections between subsequent events and might require that the sequence had been included in the motion capture session. Dealing effectively with long causal chains and dense interactions is a challenging goal for future research.

Another promising direction for future work is to incorporate physics simulation into our generate-and-rank framework. Physics simulation can provide rich variability and realism without expanding the size of motion databases. As shown by Twigg and James [2007], generate-and-browse is a viable option for steering complex multi-body dynamics simulations, from which our system could benefit greatly in order to generate more realistic interactions and collision effects. The perceptual plausibility of the causality between the selected action-response motion pairs is also an important factor that should be evaluated and taken into account, as shown by Hoyet and colleagues [2012], and physical simulation could also enhance perceived visual quality.

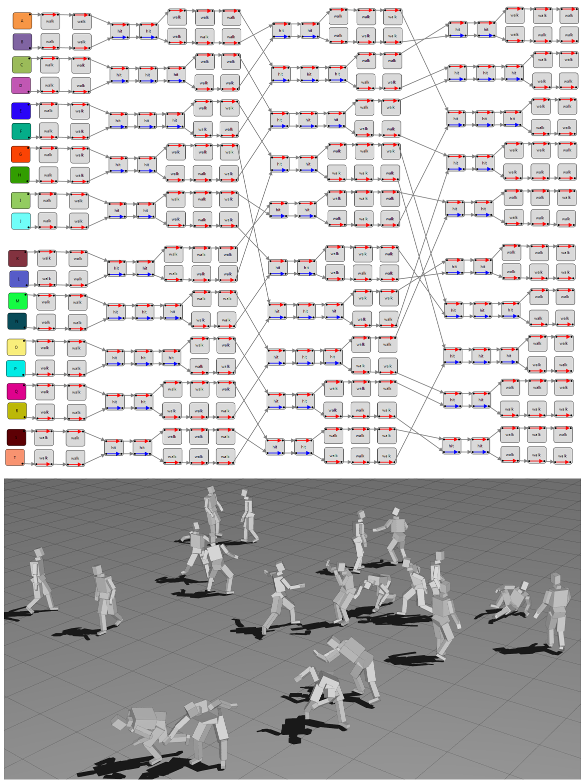

Figure 9: *Random fighting*

Action verbs			Phrases	
Single Action	***Hit***	***Interaction***	***Body part***	***State***
bend	kick	block	head	standing
cheer	punch	bump	chest	sitting
faint	slap	chase	stomach	bending
lie	***Bully***	face	pelvis	lying
sit	dig	grab	upper leg	***Strength***
stand	nudge	handshake	lower leg	low
walk	press	point	foot	medium
warm up		spin	shoulder	high
		step on	upper arm	***Direction***
		throw	lower arm	left
		push	hand	right

Table 2: *The vocabulary of action labels*

		Three people	Tournament	Payback	Movie
Graph size	Persons	3	8	4	2
	Event nodes	10	18	14	12
	Event edges	13	27	19	17
	Cycles	4	11	7	6
	Pseudo nodes	2	16	39	18
Play time		33.2	59.8	57.1	83.4
Time	Cycle ordering	0.014	0.061	0.057	0.022
	Pruning	0.255	21.201	5.426	0.200
	Motion selection	252.907	1183.268	1913.907	718.634
	Motion editing	78.376	1055.058	2624.184	529.917
	Ranking graph construction	2.663	7.863	9.306	5.503
	Ranking	0.361	0.422	0.357	0.350
	Total	334.576	2267.873	4553.236	1254.626

Table 3: *Runtime performance*

Camera views and sound effects are also important elements of scene choreography. For the examples described in this paper, we manually specified camera views and annotated sound effects. A straightforward extension would be to include audio information in the motion database, allowing us to synthesize sound effects to match the constructed scenes. Stunt directors not only choreograph fight action sequences but also have expert knowledge on how to best showcase them by selecting the camera view that achieves maximum visual impact. Incorporating such domain expertise into the design of our motion databases would also be very valuable, in the same way that Calvert and Ma [1996] incorporated the input of expert dance choreographers.

References

AGRAWAL, S., SHEN, S., AND VAN DE PANNE, M. 2013. Diverse motion variations for physics-based character animation. In *Proceedings of the 2013 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*.

AL BORNO, M., DE LASA, M., AND HERTZMANN, A. 2013. Trajectory optimization for full-body movements with complex contacts. *IEEE Trans. Visualization and Computer Graphics 19*, 8, 1405–1414.

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.

BRIN, S., AND PAGE, L. 1998. The anatomy of a large-scale hypertextual web search engine. *Computer Networks 30*, 1-7, 107–117.

CALVERT, T. W., AND MAH, S. H. 1996. Choreographers as animators: Systems to support composition of dance. In *Interactive Computer Animation*, Prentice-Hall, N. Magnenat-Thalmann and D. Thalmann, Eds., 100–126.

CASSELL, J., VILHJÁLMSSON, H. H., AND BICKMORE, T. 2001. Beat: The behavior expression animation toolkit. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, 477–486.

CHOI, M. G., KIM, M., HYUN, K., AND LEE, J. 2011. Deformable motion: Squeezing into cluttered environments. *Computer Graphics Forum (Eurographics 2011) 30*, 2, 445–453.

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

HA, S., MCCANN, J., LIU, K., AND POPOVIC, J. 2013. Physics storyboards. *Computer Graphics Forum (Eurographics 2013) 32*, 133142.

HO, E. S. L., KOMURA, T., AND TAI, C.-L. 2010. Spatial relationship preserving character motion adaptation. *ACM Transactions on Graphics (SIGGRAPH 2010) 29*, 4.

# of connecting paths	Total scenes		Top ranked scenes	
	Spatial	Temporal	Spatial	Temporal
100	0.04460	0.00019	0.01206	0.00011
300	0.03002	0.00020	0.00592	0.00005
500	0.02995	0.00018	0.00515	0.00005
700	0.02740	0.00024	0.00321	0.00003
900	0.03071	0.00023	0.00243	0.00006

Sampling per cycle	Total scenes		Top ranked scenes	
	Spatial	Temporal	Spatial	Temporal
1	0.69220	0.01923	0.38662	0.00499
100	0.22817	0.00295	0.05972	0.00075
1,000	0.11445	0.00092	0.02875	0.00032
10,000	0.06182	0.00038	0.00802	0.00008
100,000	0.03002	0.00020	0.00592	0.00005

Table 4: *Motion quality with respect to the number of connecting paths and cycle sampling for the three-person example. Spatial and temporal Laplacian energies normalized per frame indicate the degrees of spatial deformation and time warping. Low energies indicate better quality results. The average energies over all 1000 candidate scenes and the top 8 scenes are provided.*

HOYET, L., MCDONNELL, R., AND O'SULLIVAN, C. 2012. Push it real: Perceiving causality in virtual interactions. *ACM Transactions on Graphics (SIGGRAPH 2012) 31*, 4.

HYUN, K., KIM, M., HWANG, Y., AND LEE, J. 2013. Tiling motion patches. *IEEE Transactions on Visualization and Computer Graphics 19*, 11, 1923–1934.

JAIN, S., AND LIU, C. K. 2009. Interactive synthesis of human-object interaction. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, 47–53.

JING, Y., AND BALUJA, S. 2008. VisualRank: Applying pagerank to large-scale image search. *IEEE Transactions on Pattern Analysis and Machine Intelligence 30*, 1877–1890.

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

KIM, M., HWANG, Y., HYUN, K., AND LEE, J. 2012. Tiling motion patches. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 117–126.

LEE, J., CHAI, J., REITSMA, P., HODGINS, J., AND POLLARD, N. 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., AND LEE, J. 2006. Motion patches: building blocks for virtual environments annotated with motion data. *ACM Transactions on Graphics (SIGGRAPH 2006) 26*, 3.

LEVINE, S., KRÄHENBÜHL, P., THRUN, S., AND KOLTUN, V. 2010. Gesture controllers. *ACM Transactions on Graphics (SIGGRAPH 2010) 29*, 4.

LEVINE, S., LEE, Y., KOLTUN, V., AND POPOVIĆ, Z. 2011. Space-time planning with parameterized locomotion controllers. *ACM Transactions on Graphics 30*, 3.

LIU, L., YIN, K., VAN DE PANNE, M., SHAO, T., AND XU, W. 2010. Sampling-based contact-rich motion control. *ACM Trans. Graph. 29*, 4.

LO, W.-Y., AND ZWICKER, M. 2008. Real-time planning for parameterized human motion. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 29–38.

MARKS, J., ANDALMAN, B., BEARDSLEY, P. A., FREEMAN, W., GIBSON, S., HODGINS, J., KANG, T., MIRTICH, B., PFISTER, H., RUML, W., RYALL, K., SEIMS, J., AND SHIEBER, S. 1997. Design galleries: A general approach to setting parameters for computer graphics and animation. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, 389–400.

MEI, Q., GUO, J., AND RADEV, D. 2010. DivRank: The interplay of prestige and diversity in information networks. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1009–1018.

MORDATCH, I., TODOROV, E., AND POPOVIĆ, Z. 2012. Discovery of complex behaviors through contact-invariant optimization. *ACM Trans. Graph. 31*, 4.

MÜLLER, M., AND RÖDER, T. 2006. Motion templates for automatic classification and retrieval of motion capture data. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 137–146.

NEFF, M., KIPP, M., ALBRECHT, I., AND SEIDEL, H.-P. 2008. Gesture modeling and animation based on a probabilistic recreation of speaker style. *ACM Transations on Graphics 27*, 1.

PEDICA, C., AND VILHJÁLMSSON, H. H. 2010. Spontaneous avatar behavior for human territoriality. *Appl. Artif. Intell. (IVA Special Issue) 24*, 6, 575–593.

PERLIN, K., AND GOLDBERG, A. 1996. Improv: A system for scripting interactive actors in virtual worlds. In *Proceedings of SIGGRAPH 1996*, 205–216.

SAFONOVA, A., AND HODGINS, J. K. 2007. Construction and optimal search of interpolated motion graphs. *ACM Transactions on Graphics (Siggraph 2007)*.

SHUM, H. P. H., KOMURA, T., SHIRAISHI, M., AND YAMAZAKI, S. 2008. Interaction patches for multi-character animation. *ACM Transactions on Graphics (SIGGRAPH ASIA 2008) 27*, 5.

SHUM, H. P. H., KOMURA, T., AND YAMAZAKI, S. 2010. Simulating multiple character interactions with collaborative and adversarial goals. *IEEE Transactions on Visualization and Computer Graphics*, 99.

STONE, M., DECARLO, D., OH, I., RODRIGUEZ, C., STERE, A., LEES, A., AND BREGLER, C. 2004. Speaking with hands: Creating animated conversational characters from recordings of human performance. *ACM Transactions on Graphics (SIGGRAPH 2004) 23*, 3, 506–513.

TWIGG, C. D., AND JAMES, D. L. 2007. Many-worlds browsing for control of multibody dynamics. *ACM Transactions on Graphics (SIGGRAPH 2007) 26*, 3.

WAMPLER, K., ANDERSEN, E., HERBST, E., LEE, Y., AND POPOVIĆ, Z. 2010. Character animation in two-player adversarial games. *ACM Transactions on Graphics 29*, 3.

YE, Y., AND LIU, C. K. 2012. Synthesis of detailed hand manipulations using contact sampling. *ACM Transactions on Graphics (SIGGRAPH 2012) 31*, 4, 41:1–41:10.

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

YU, Q., AND TERZOPOULOS, D. 2007. A decision network framework for the behavioral animation of virtual humans. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 119–128.