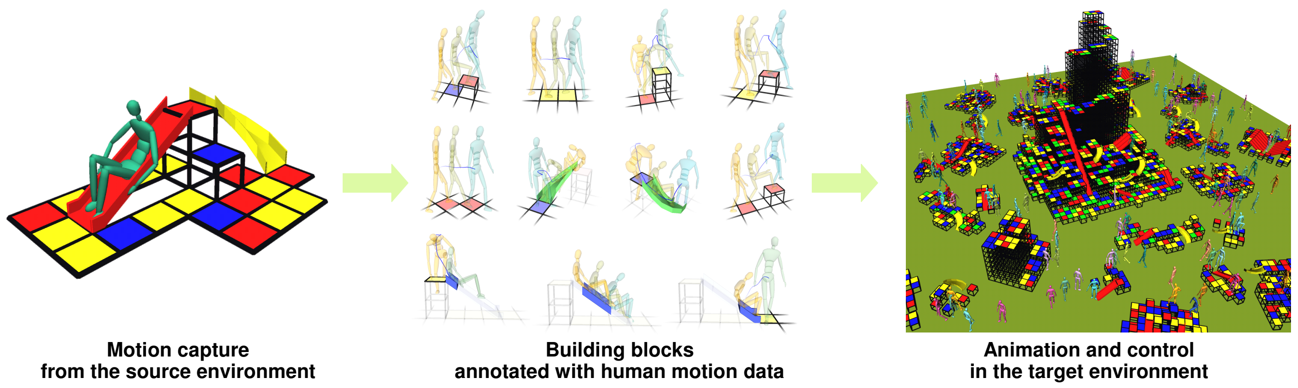
Motion Patches: Building Blocks for Virtual Environments Annotated with Motion Data


Motion capture
from the source environment
Building blocks
annotated with human motion data
Animation and control
in the target environment

Abstract

Real-time animation of human figures in virtual environments is an important problem in the context of computer games and virtual environments. Recently, the use of large collections of captured motion data has added increased realism in character animation. However, assuming that the virtual environment is large and complex, the effort of capturing motion data in a physical environment and adapting them to an extended virtual environment is the bottleneck for achieving interactive character animation and control. We present a new technique for allowing our animated characters to navigate through a large virtual environment, which is constructed using a set of building blocks. The building blocks can be arbitrarily assembled to create novel environments. We annotate each block with a *motion patch*, which informs what motions are available for animated characters within the block. The versatility and flexibility of our approach are demonstrated through examples in which multiple characters are animated and controlled at interactive rates in large, complex virtual environments.

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

Keywords: Interactive character animation, human motion, motion capture, virtual environments, path planning

1 Introduction

The real-time animation and control of human figures in complex virtual environments have been an important problem in computer graphics. A number of techniques have been developed for animating human figures: the motion of the figures may be keyframed, simulated, procedurally defined, or live captured. We are particularly interested in the last technique using a database of recorded motion. This data-driven approach can create a rich variety of natural human behavior from a large data set recorded from live actors. However, applying data-driven approaches to human figures in a complex environment is still challenging because a large motion set must be searched in real-time in order to select motions appropriate to given situations, and the selected motion must be adapted to match the environment.

The virtual environments we intend to design are often too big to be physically realized and accommodated in a motion capture studio. Motion capture in such a large environment is certainly difficult, if not impossible. One approach is to divide a large environment into small pieces, then record motion interacting with each piece separately, and finally combine motion data into a single, large piece of data covering the entire environment. In this way, the character's motion in a large environment can be collected. However, this approach requires a rich amount of motion data at every portion of the environment, and thus the motion capture process can be quite laborious and painful. A significant amount of redundancy exists in the motion set because it includes the same motion performed in different parts of the environment. This redundancy can be avoided by reusing motion data captured at one location to animate characters at other locations.

We present a new technique for allowing our animated characters to navigate through and interact with a large virtual environment in real-time. In order to reuse motion data collected from one environment to another, the key idea of our approach is to identify a set of building blocks from the source environment, embed recorded motion data in these blocks, and fit them to the target environment so that the embedded motion data can be transferred between the environments. The set of motion data embedded in a building block is called a *motion patch* and informs what motions are available for animated characters within the block. The overview of our approach is as follows:

- **Data collection and processing.** We first build a physical environment in the motion capture studio and collect motion data viable in the environment. The recorded motion data is preprocessed such that the frames in the motion data sets are sorted into groups. The clustering allows us to construct and stitch motion patches efficiently.
- **Patch construction.** A set of building blocks is identified by analyzing the geometric structure of the environment and the recorded motion data. The motion patch embedded in each

building block forms a directed graph to achieve flexibility in the character's motion within the block.

- **Fitting to the target environment.** Once the target environment is created, the set of building blocks can be fitted in the target environment such that the environment is covered by the embedded motion patches. By establishing the connections between motion patches, the target environment can be automatically annotated with a rich, connected set of motion data, which allows characters to navigate through the environment. We implemented two types of motion patches: tilable and non-tilable patches. Non-tilable patches are connected to each other by finding intersections at overlapping regions, while tilable patches can be seamlessly tiled in a regular pattern without much effort.
- **Animation and control.** Our system allows the user to change the environment dynamically at runtime and control animated characters in the environment interactively. The motion patches make it possible to achieve real-time performance in finding collision-free paths through complex virtual environments and animating multiple characters while avoiding collisions between them.

2 Background

Animation and control of human figures in synthetic environments have been studied in computer graphics for the last decade [Bandi and Thalmann 1997; Bindiganavale et al. 1994; Jung et al. 1994; Noser et al. 1995; Thorne et al. 2004]. This problem is very difficult because the human motion is high-dimensional, and many degrees of freedom need to be choreographed in a coordinated, collision-free, human-like manner.

Captured motion data have frequently been used in character animation for reproducing the naturalness of human movements. Recently, a number of researchers have explored the method of representing a significant amount of motion data as a directed graph and using it to animate and control human figures [Arikan and Forsyth 2002; Kovar et al. 2002; Lee et al. 2002; Pullen and Bregler 2002]. This method has further been explored for better controllability [Arikan et al. 2003; Hsu et al. 2004; Stone et al. 2004], efficient search [Lee and Lee 2004], parameterizing and blending motions [Kovar and Gleicher 2004; Park et al. 2004], synchronizing with sound [Kim et al. 2003], parameter tuning [Wang and Bodenheimer 2004], classification [Kwon and Shin 2005], simulating group behavior [Lai et al. 2005], responsiveness to external forces [Arikan et al. 2005], and evaluating the effectiveness of a motion graph in a specific environment [Reitsma and Pollard 2004].

Lee and his colleagues [2002] showed the capability of the graph-based motion representation to reuse motion data collected from a "poles and holes" terrain environment to an extended terrain environment. Their character control algorithm is based on state space search techniques and barely achieves its interactive performance (approximately 5 to 10 frames per second) with a single character. We address the same problem with much flexibility in selecting target environments and aiming to achieve real-time performance with multiple characters. To do so, we annotate the target environment with a repertoire of character motions available at each location. This annotation makes path planning and state space search in the environment very efficient and straightforward. Our work is related to animation techniques that annotate environment objects with the availability of motions [Abaci et al. 2005; Shao and Terzopoulos 2005; Sung et al. 2004]

Our problem is also related to the path planning of articulated figures in the presence of obstacles, which is a classical problem in a wide range of disciplines. There is a vast amount of literature on path planning [Latombe 1991]; however, only a few works addressed human motion planning, which yields a high-dimensional (typically, 20 to 100) configuration space. With such a high-dimensional configuration space, most optimal path planning algorithms are either computation-intensive for searching through configuration spaces exhaustively or memory-intensive for maintaining discretized configuration spaces. Many researchers used a low-dimensional configuration space (e.g., body and footprint locations) and randomized sampling techniques for producing character animation of locomotion, crawling, and climbing [Choi et al. 2003; Kalisiak and van de Panne 2001; Kuffner et al. 2001; Lau and Kuffner 2005; Pettre et al. 2003; Shiller et al. 2001; Sung et al. 2005], and grasping and manipulating an object [Koga et al. 1994; Yamane et al. 2004]. Our motion patches can be thought of as a way to create discrete configuration spaces memory-efficient without losing the diversity and subtle details of captured motion data.

Reitsma and Pollard [2004] embedded human motion data in a square tile and allowed a character to navigate a large environment covered with a regular grid of square tiles. We extend their work and deal with several challenges, which include embedding motion data into arbitrarily-shaped objects other than square tiles, allowing animated characters to navigate around and interact with these objects in a collision-free manner, and allowing locomotion and the other types of human motion to be handled in a uniform way.

3 Data Acquisition and Processing

All of the motion data used in our experiments were captured from a Vicon optical system with sixteen cameras at the rate of 120 frames/second and then down-sampled to 30 frames/second for real-time display. Our motion capture studio allows an effective capture region of 4 by 4 by 2.5 meters. Motion capture data contains trajectories for the position and orientation of the root node (pelvis) as well as relative joint angles for each body part.

We built several physical environments, each of which can be accommodated in our motion capture studio (see Figure 1). Each environment was designed to include some geometric features of the target (possibly much larger) environment we intend to create. All of the geometric features distributed in the physical environments can be rearranged and assembled to reconstruct the target environment. We also prepared polygonal models that match the physical environments.

In motion capture sessions, we collected a variety of human motions that are viable in a given environment. Our motion database contains about 70 minutes of data (see Table 1). To create the database, our subject walked, climbed, and slid in the playground and walked up and down the stairs. In the office environment, our subject approached the chair, sat down, stood up, and left repeatedly. We also captured our subject walking, stopping, idling, chatting, and making a presentation in an empty environment. Motion data were recorded in long clips so that seamless, natural transitions between motions could be captured. Our subjects were instructed to repeat each action several times to capture variations in each action. We denote the set of recorded motion data in the database as $\{\mathbf{p}_i | i = 1, \cdots, n\}$, where each frame $\mathbf{p}_i$ is the joint angle representation of a specific pose of the character.

Contact. The motion data is automatically preprocessed to annotate body-environment contact information at each frame. A body segment and an environment object are considered to be in contact

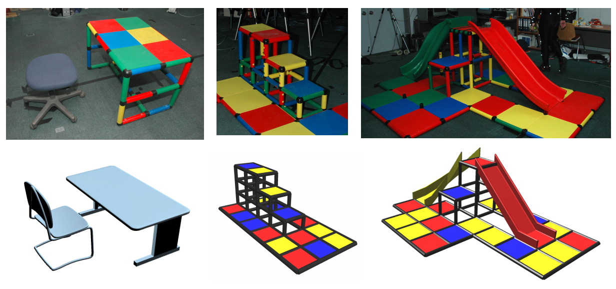


Figure 1: The physical environments we built in our motion capture studio for collecting human motion data and their polyhedral models. The playground construction kit was used for building the desk, the steps, and the playground. We removed the back of the chair in order to avoid the occlusion of reflective markers from motion capture cameras.

Motion sets	Environments		# of	Time
	source	target	frames	(sec)
walk/climb/slide	playground(S)	playground(L)	51169	1706
idle	empty	office	3685	123
chat	empty	office	3653	122
dispute	empty	office	3213	107
presentation	empty	office	3179	106
sit down/stand up	desk/chair	office	11778	393
work at the desk	desk/chair	office	5122	171
chat at the desk	desk/chair	office	10171	339
walk up/down	stairs	office	19421	324
walk-stop-walk	empty	office	4385	146
walk	empty	both	23911	797

Table 1: The motion sets collected for our experiments. Each data set was captured in the source environment and used for animating characters in the target environment.

if any joint adjacent to the segment is sufficiently close to the object and its velocity is below some threshold.

Clustering. The construction and stitching of motion patches involve searching pairs of similar frames that allow smooth transition from one patch to another. At a preprocessing phase, we sort motion frames into groups to accelerate this similarity search. The agglomerative hierarchical k-means algorithm [Duda et al. 2000] is employed to guarantee that the distance between any frames in a group is below a user-specified threshold. The threshold is determined through experimentation such that transitions within a group can generate smoothly blended motions. Here, in order to measure the distance between motion frames, we consider six features of the frames. All of the features were selected such that the distance became invariant under horizonal translation and rotation about the vertical (up direction) axis. Given two frames $\mathbf{p}_i$ and $\mathbf{p}_j$, the features are:

- *Joint angles:* The differences in all of the joint angles are considered. In our system, joint angles are described with unit quaternions and the difference between two unit quaternions q_i and q_j is measured by their geodesic distance $\min(\|\log(q_j^{-1}q_i)\|, \|\log(q_j^{-1}(-q_i))\|)$.

- *Root height:* The difference $(y_i - y_j)$ in the height of the root nodes from the reference ground plane is considered. The horizontal coordinates of the root nodes are disregarded.

- *Root orientation:* The orientations of the root nodes are also described with unit quaternions. To effectively ignore the rotation about the vertical axis in the difference between two orientations q_i and q_j, we first rotate q_i about the vertical axis so that it can be brought as close to q_j as possible. The optimal rotation about the vertical axis $\hat{y}$ is denoted as $\exp(\theta\hat{y})$, where θ is the rotation angle about the vertical axis. Then, the geodesic distance between $q_i' = \exp(\theta\hat{y})q_i$ and q_j is invariant under rotation about the vertical axis. θ is computed as the solution of finding the closest point on the geodesic curve $G(\theta) = \exp(\theta\hat{y})$ from the unit quaternion $q = q_j q_i^{-1} = (w, v)$, where $w \in \mathbb{R}$ and $v \in \mathbb{R}^3$.

$$\theta = \begin{cases} -\alpha + \frac{\pi}{2}, & \text{if } q \cdot G(-\alpha + \frac{\pi}{2}) > q \cdot G(-\alpha - \frac{\pi}{2}), \\ -\alpha - \frac{\pi}{2}, & \text{if } q \cdot G(-\alpha + \frac{\pi}{2}) < q \cdot G(-\alpha - \frac{\pi}{2}), \end{cases} \tag{1}$$

where $\tan\alpha = w/(v \cdot \hat{y})$.

- *The velocities of joints, root translation, and root rotation:* The joint velocity at a joint is computed as a difference between angles at the next and the previous frames, that is, $v_i = \log(q_{i-1}^{-1}q_{i+1})/2$. The linear and angular velocities of the root node can also be given by differencing the next and the previous frames. We represent the linear and angular velocities of the root node with respect to a local, moving coordinate system attached to the root node. This allows the velocities to be rotation-invariant.

The above six features are weighted, squared, and summed to compute the squared distance between motion frames. For effective clustering of motion frames, contact with the environments is an important perceptual feature of motion, which is not reflected in the distance between motion frames. Given a motion frame, we consider the contact states at three successive frames including the previous and next frames for evaluating the similarity between motion frames. In the process of agglomerative clustering, motion frames are considered to be dissimilar if their contact states or the contact states of their neighboring frames are different.

4 Motion Patch Construction

We intend to create virtual environments that consist of a reasonably small number of *unit objects*. A unit object is defined by its geometric shape, which cannot be modified or changed, and the bounding box. The virtual environment may include multiple instances of each individual unit and the location of each instance in the environment is described by a rigid transform. We denote the environment as $\{(u_1, T_1), (u_2, T_2), \cdots, (u_m, T_m)\}$, where u_i is a unit object and T_i is a rigid transform with respect to the world coordinate system. The motion of a character may occur either on a single unit or across multiple units. Accordingly, each *building block* for virtual environments consists of either one or more units such that any recognizable human action can be embedded in a single building block.

A pose in the motion data belongs to a unit object if the pose is either in contact with the geometry of the unit or intersects its bounding box. The pose can belong to more than one unit simultaneously. A segment of motion frames $\{\mathbf{p}_s, \cdots, \mathbf{p}_e | 1 \le s \le e \le n\}$ in the database is supposed to be embedded in a building block if every frame in the segment belongs to one of its units.

Given an environment and the set of motion data captured in that environment, the algorithm for identifying a set of building blocks of length k is as follows (see Figure 2). We begin with an empty set

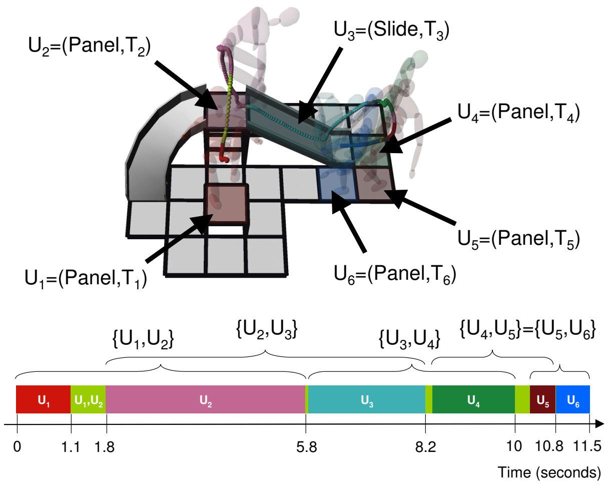


Figure 2: Identifying building blocks from motion data. The environment consists of three unit objects: the ground panels, the straight slide, and the curved slide. A short segment of motion capture data is used for uncluttered illustration. The root trajectory is depicted as a series of small spheres. The motion frames at the bottom of the figure are partitioned depending on to which unit objects the frames belong. The frames colored in lime green belong to two objects simultaneously and correspond to the character's motion transitioning from one object to another. Our algorithm identified four building blocks: $\{U_1,U_2\},\{U_2,U_3\},\{U_3,U_4\},\{U_4,U_5\}$. Note that $\{U_4,U_5\}$ and $\{U_5,U_6\}$ are identical.

of building blocks. The motion data set is scanned to find a segment of successive frames that belongs to any k units. If a building block consisting of the same k units already exists, we simply register the segment of frames to the existing block. Otherwise, we create a new block and register the segment to the new one. When we decide if two building blocks are equivalent, the relative locations and rotational symmetry of unit objects are considered to match the shapes of the blocks. However, the absolute locations of unit objects with respect to a global, fixed coordinate system are disregarded. In this way, motions recorded in different parts of the environment can be gathered and embedded in a building block.

The embedding of motion data in each individual building block forms a directed graph to allow characters to navigate within the block. We assume that all motion frames in the database have been sorted into groups at a preprocessing phase. We approximate each block using a regular grid of cells. Each cell is indexed by two-dimensional location (x,y), yaw orientation θ, and the index p of groups of the character's poses. A frame of the motion data belongs to a cell if the root of the body is located within the cell and the pose belongs to the p-th group. The cells become the nodes of the directed graph. The connecting transitions between cells are created if any motion segment embedded to the block allows transition from one cell to the other. More specifically, the connecting transitions between cells are formed as follows (see Figure 3). For any motion segment, we check all of its sub-sequences by double looping over the frames. For every sub-sequence longer than 20 frames (0.66 second), we insert a connecting transition between the cells where the sub-sequence originates and terminates. The minimum length constraint is required to avoid too frequent transitions in animation. We repeat this procedure for every motion segment embedded in the block.

Once the graph is constructed, the character can navigate through the block traversing the graph. Since a path traversing the graph

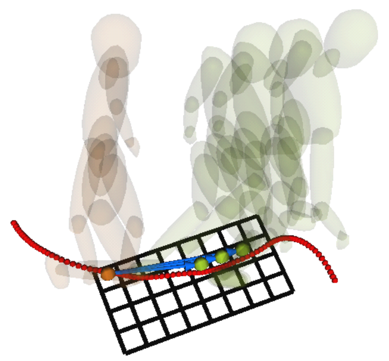


Figure 3: Forming connecting transitions between 4D cells within a motion patch. A motion segment passing through the cell marked by the red sphere forms links to the green cells, which are subsequently visited by the motion segment, if the pose group indexes in the cells match the poses in the motion segment at incident frames.

corresponds to a series of motion segments, the animation of characters along the graph involves the concatenation of motion segments at cells. Transition from one motion segment to another is made smooth by blending the segments in and out and enforcing foothold constraints, as done in [Lee et al. 2002].

Block size. We do not have a simple rule to determine the size of building blocks. Through experiments, we have found several heuristic rules. The size of blocks should be determined by the content of capture motion data. Each individual block should be large enough to accommodate any single recognizable action in the data set. With walking motion data, for example, we select the square ground panel as a unit object, of which the width and height are about the same as the distance travelled by two cycles of walking. There exists a trade-off, however, for different block sizes. Large blocks allow rich connections through the embedded motion patches, while smaller blocks provide flexibility in designing virtual environments.

5 Stitching Motion Patches

The user is provided with a user interface system that allows the user to design a target environment using a set of unit objects. Our system fits a set of building blocks automatically into the target environment such that the target environment is covered by the building blocks (see Figure 4). Transitions between motion patches are established where the patches overlap. Given two overlapping patches, the transition from cell (x_1,y_1,θ_1,p_1) of one patch to cell (x_2,y_2,θ_2,p_2) of the other patch is formed if the pose indexes are identical and the distance between the cell centers and the angle between the yaw orientations are below certain thresholds. In our experiments, the thresholds are the same as the size of the cells. The position threshold ranges from 5 to 10cm and the orientation threshold ranges from 5 to 10 degrees.

Once the connections among motion patches are established, we obtain a large, connected graph covering the target environment. Some connecting transitions between cells should be disabled to avoid collision and dead ends. For each motion patch, we first disable transitions that cause the collision between the animated character and the environment. Then, we run a strongly connected component algorithm [Tarjan 1972] on the graph and prune connecting transitions that are not contained in the largest strongly connected component.

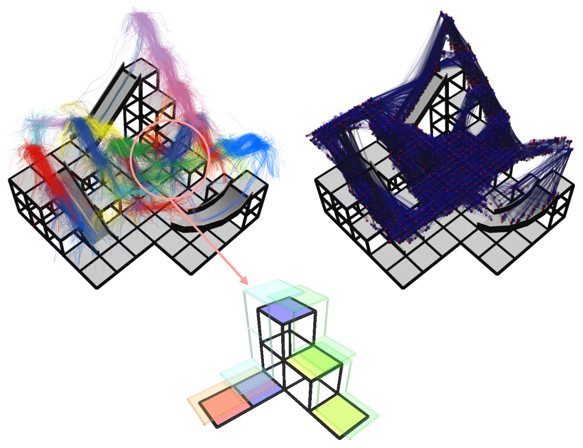


Figure 4: (Top left) The motion patches covering the target environment are depicted as the root trajectories rendered in different colors. (Bottom) Building blocks are fitted into the target environment by shape matching. (Top right) The motion patches are stitched and then pruned to leave a single strongly connected component.

Memory efficiency. In the worst case, the memory requirement of motion patches would scale in proportion to the square of the number of frames in the database, because the directed graph in the patches could have $O(n^2)$ connecting transitions for n nodes. In practice, the memory requirement is much less than the worst case for several reasons. At first, the motion database is divided into subsets, and each subset of motion frames constructs a motion patch. Therefore, the memory requirement is asymptotically not proportional to the size of the entire data sets, but related to the size of the largest motion patch. Secondly, since the database contains many unrelated motions, the directed graph is actually sparse in practice. The storage cost of the graph is also mitigated by the clustering of motion frames. The cluster-to-cluster transition graph is significantly smaller than the frame-to-frame transition graph. Finally, the storage cost for creating multiple instances of a motion patch is modest, because each instance does not need to include motion data and their graph structure. In each instance, we maintain only an array of bits that mark cells and transitions disabled by either obstruction or insufficient connections.

6 Tilable Patches

We also implemented a special type of patches that can be tiled in a regular pattern, such as square ground panels and stairs. In this section, we will explain the construction of square ground patches that allow the character to enter and leave the patch in any of four directions. The other types of tilable patches can be constructed in a similar way.

The tilable motion patch has pairs of matching boundary edges and a number of *entries* and *exits* sampled on the edges. Any entry on a boundary edge always has its corresponding exit on the opposite edge and vice versa. This boundary condition allows motion patches to be aligned and tiled without much effort. Entries and exits have the properties of the position and yaw orientation of the root of the character and the index of the pose groups.

Entries and exits are connected by motion segments chosen from in put motion data (see Figure 5). More specifically, a motion segment can be used to link an entry-exit pair if the poses at the first and last frames belong to pose groups indicated at the entry and exit, respectively, and the motion segment can be edited within a user-specified tolerance to match the incoming/outgoing positions and orientations at both ends. To form connections between entry-exit pairs, we check all possible subsequences of input motion data by double looping over the frames.

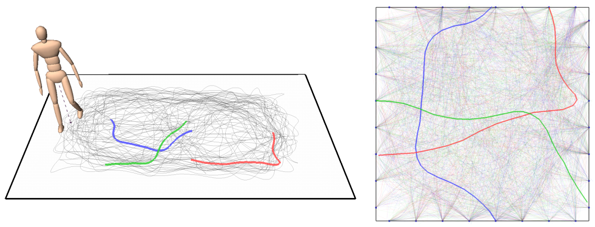


Figure 5: The construction of square ground patches. (Left) We collected the motion of our subject walking around for about ten minutes. The motion data are depicted as the root trajectories that are projected on to the ground. (Right) The square patch has entries and exits sampled evenly on the boundary. The connections between the entries and the exits are formed by selecting appropriate motion segments from the walking motion data.

The connections in a patch should be rich enough to provide a variety of motion. The richness of connections depends not only on the size of input motion data, but also on the extent to which we are willing to edit the motion segments. We use a simple linear model presented by Reitsma and Pollard [2004] for editing the path of motion. The linear model allows the root position and orientation to be changed linearly from the start to the end of the segment. The total amount of position and orientation changes allowed at the end is linearly proportional to the distance travelled.

Pruning. A tilable motion patch is *safe* if two conditions are satisfied. The first condition is that every entry must have at least one out-going motion for each of the four directions. Even with a large collection of motion data, we may not be able to find enough connecting motions for some entries if the entries are heading toward a corner of the patch. We remove those entries and their corresponding exits on the opposite edges to avoid dead ends and ensure enough flexibility in steering animated characters. The second condition requires that motion patches should produce a strongly connected graph when they are tiled on a surface. This condition ensures that the entire motion sets can be fully utilized in animating characters. We consider a torus constructed from the motion patch by gluing both pairs of opposite edges. On the torus, motion segments embedded in the patch create a boundary-less directed graph. We run Tarjan's algorithm [Tarjan 1972] on the boundary-less graph to identify the largest strongly connected component and disable connecting motions that are not contained in the strongly connected component. To ensure both conditions simultaneously, we enforce two safety conditions repeatedly and alternatingly until no connection is pruned.

Connection between patches. Tiled patches can be connected to other patches overlaid on the tiled surface (see Figure 6). To find the intersections between patches efficiently, we partition the internal region of tilable patches into 4D cells, which are indexed by 2D location, yaw orientation, and the index of the pose groups. Each cell maintains a list of motion segments passing through the cell. This cell partitioning allows us to find the coincident frames efficiently within a certain threshold from the overlapped patches. The characters can transit from one patch to another at those coincident frames.

Avoiding obstacles. In the presence of obstacles on the tiled sur-

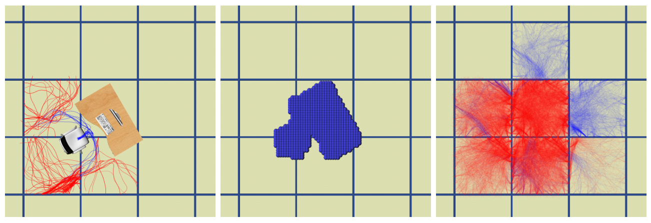


Figure 6: The desk and chair on the tiled surface. (Left) The red motion segments embedded in the ground panel and the blue motion segments embedded in the desk and chair have coincident frames, at which characters are allowed to make a transition between the patches. (Center) The blue cells in the ground panel are concealed by the desk and chair and thus disabled. (Right) The red motion segments are disabled because they are obstructed by the obstacles. The blue motion segments are disabled because they have all following motions disabled.

face, motion segments obstructed by the obstacles need to be disabled. This can be done simply by marking the cells concealed by the obstacles and then disabling all motion segments passing through the marked cells. After disabling some motion segments, we have to check if their adjacent motion segments have at least one out-going transition. Otherwise, those adjacent segments would cause dead ends and should be disabled. This procedure is repeated until no more motion segments are disabled. In practice, this procedure can be finished in one or two iterations.

7 Animation and Control

We built a simple interactive editing system for designing various virtual environments. Our system allows the user to create, remove, or drag building blocks interactively to change the environment. Accordingly, motion patches are created, connected to each other, or detached from each other in order to dynamically update the connected structure of motion patches in the environment. The user can also control each character by specifying either a goal location to move to or a specific behavior such as idling, chatting, and roaming around.

The environment is annotated with a graph of motions covering the whole area of the environment. The animation and control of characters along the graph is computationally very efficient. At every node of the graph, characters are provided with a set of motions immediately available at that moment and can proceed with any choice among available motions. Assuming that the decision is made based on local information, the computation time for animation and control is often negligible, and the total computation time is dominated by rendering the scenes on the computer screen.

The capability of global path planning is an important feature of our system for controlling characters in complex environments. Optimal path planning algorithms, such as Dijkstra's shortest path algorithm, require an explicit representation of the entire connected graph, which is often implausible for large environments. The storage efficiency of our approach is largely due to the distributed representation of the motion graph annotated in the environment. We actually do not maintain an explicit representation of the graph, but the graph is implicitly maintained in the connections among motion patches. We address this problem in a hierarchical manner. In our system, the motion data are organized in a two-layer structure. The higher layer is a directed graph of which nodes are motion patches. The higher layer maintains the connectivity among motion patches.

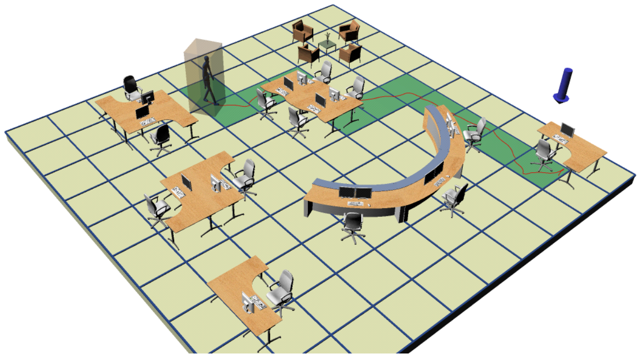


Figure 7: Two-layer path planning.

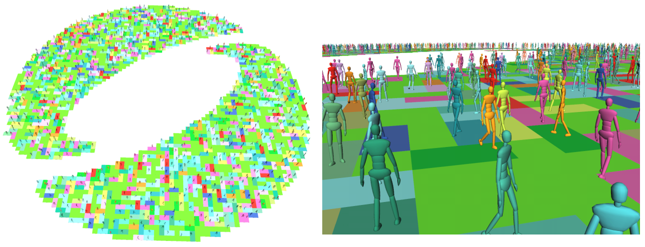


Figure 8: (Left) One thousand of animated characters on the SIGGRAPH logo. (Right) A close-up view.

The lower layer is a directed graph of motion segments embedded in motion patches. Our two-layer path planning algorithm first finds a path to the goal at the resolution of patches in the higher layer and then refines the path by running a shortest path algorithm through the lower layer (see Figure 7). In this way, we can run path planning algorithms time-efficiently within limited storage space.

8 Experimental Results

The timing data provided in this section was measured on a 2.8GHz Intel Pentium 4 computer with 2Gbyte main memory and an nVidia Quadro FX1100 graphics accelerator.

Performance. To evaluate the performance of our system, we created one thousand of animated characters on a grid of walk patches (see Figure 8). A 13-minute sequence of motion was captured and used to construct the tilable walk patch. Collisions between characters are avoided approximately at the resolution of building blocks. At any instance, each character occupies two blocks (the one it belongs and the other one on which it will move to shortly) and avoids the collision with the other characters by preventing them from entering the occupied blocks. Our system required about 66 seconds to create 300 frames (10 seconds) of video images. Actually, rendering dominated the computation time. Our system required only 2.8 seconds to create the same animation with video disabled. It means that the motion of one thousand characters is computed and controlled at a rate of more than 100 frames per second.

Playground. We recorded motions of about 28 minutes duration (51169 frames) from the playground environment (see Figure 9). In the recorded data, our subject walked, climbed, and slid repeatedly in the playground. The clustering of motion frames produced 4928 pose groups. From the source environment, we selected five unit objects by hand: the ground panel, the curved slide, two outer

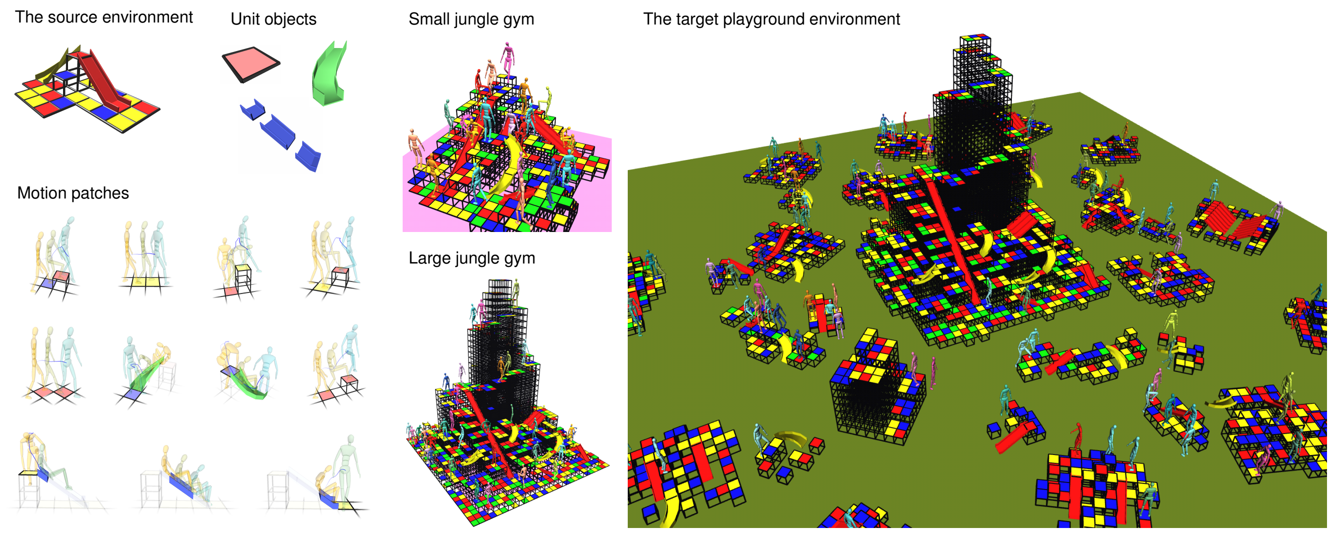


Figure 9: From the source playground environment, we selected five unit objects that include the square ground panel, the curved slide, and the straight slide divided into three pieces. Our algorithm identified eleven building blocks. Motion data collected from the source environment were transferred to the target environments using motion patches.

parts of the straight slide, and its intervening part. Our system identified eleven building blocks. Each building block consists of two unit objects, except for the intervening part of the straight slide, which consists of a single unit. The intervening part can be duplicated and strung together in order to generate arbitrarily long slides. 536 instances of motion patches were fitted in the small jungle gym through shape matching (the eleven patches were instantiated 132, 181, 21, 75, 62, 3, 3, 36, 5, 13, and 5 times, respectively). These patches produced a strongly connected component with 43,516 cells and 346,507 transitions. Our system required about 4 minutes and 30 seconds to annotate the small jungle gym environment. This computation time includes 0.7 second for shape matching, 257.7 seconds for stitching patches, 10 seconds for generating transitions, and 4 seconds for finding a strongly connected component. 2063 instances of motion patches were fitted in the large jungle gym. Stitching these patches produced a strongly connected component with 227,583 cells and 1,474,971 transitions. The target playground environment is equipped with 32 jungle gyms, which are connected to a 30 by 30 grid of walk patches. 8775 instances of motion patches were required to cover the whole environment, which is annotated with a motion graph with 16,745,035 nodes and 15,326,250 transitions. This graph encodes about 6847.57 hours of motion.

Office. For the office example, 40 minutes of motion were collected, and the motion data were sorted into 2337 pose groups (see Figure 10). We constructed eight motion patches for the office example, except for the tilable walk patch that is used in all of our examples. The desk and the chair are considered as a single unit. Three patches (sit down/stand up, work at the desk, and chat at the desk) are embedded in the desk-and-chair block. Five patches (idle, chat, dispute, presentation, and walk-to-stop) describes human behaviors in an empty space and are embedded in the square ground panel. Our system required less than one second to create, remove, or drag a desk-and-chair block on the grid of walk patches. The location of a behavior patch is not fixed with respect to a world coordinate system, but determined appropriately when the character makes a transition to the patch such that the transition could be made immediately.

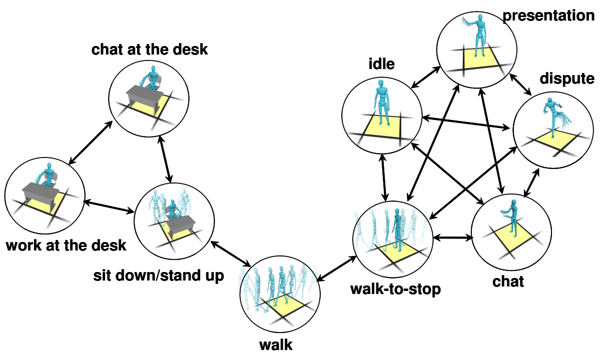
9 Discussion

Motion patches are useful for graphics applications in which multiple characters are animated and controlled at interactive rates in a large virtual environment. Motion patches are simple, versatile, and easy to implement. The primary advantage of our approach is that it scales well with the size of motion data and the complexity of virtual environments. Computer games can benefit from the versatility and compactness of motion patches. In many computer games, we see characters that can move only in four or eight axis-aligned directions. Our motion patches provide diversity in choosing motions through rich connections within patches even if the patches are tiled regularly on a grid.

All of the building blocks consist of either one or two unit objects in our experiments. This is because our motion data sets do not include character poses touching more than two units simultaneously. We have had difficulty building bigger and heavier physical environments in our motion capture studio because of its limited space and the occlusion of reflective markers from motion capture cameras. Commercial studios have larger spaces and more cameras to capture more challenging motions, such as climbing in the playground, which use both hands and feet to touch several units simultaneously. We would need blocks consisting of more than two units to accommodate those motions.

In our experiments, it was reasonably easy to select the set of unit objects and design the source environment by hand. However, with an enormously large and complex target environment, inspecting the entire structures and selecting unit objects can be tedious and laborious. Ideally, we wish that our system would be able to take target environment data, such as architectural CAD data, as input, analyze the environment data automatically to identify a set of unit objects, and suggest several source environments that could be physically realized in the motion capture studio.

Our approach is particularly suitable for artificial environments, such as architectural structures and urban areas, in which regularities in geometric features are abundant. One limitation of our approach is that we need too many building blocks for building natural, irregular environments. This could be alleviated by allowing



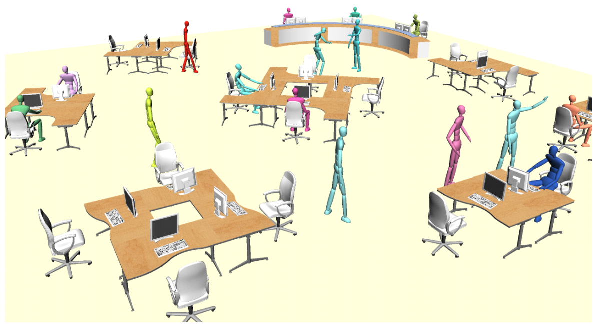


Figure 10: Office. (Top) The connectivity among motion patches. (Bottom) Multiple characters animated and controlled in the office environment.

the free-form deformation of motion patches. We might be able to fit the target environment approximately with a reasonably small number of deformable patches. Motion data embedded in the deformable patches could be edited accordingly using an off-the-shelf motion editing technique [Gleicher 1998; Lee and Shin 1999].

Another limitation of our approach is that physics is simply ignored in our framework. The characters on very long slides do not accelerate properly because the character's motion was created by repeating the ones recorded from a short slide. We arbitrarily accelerated characters on long slides to make them more realistic. Combining data-driven techniques with physically based methods will open up many possible directions for future research.

References

ABACI, T., CÍGER, J., AND THALMANN, D. 2005. Planning with smart objects. In *WSCG (Short Papers)*, 25–28.

ARIKAN, O., AND FORSYTH, D. A. 2002. Interactive motion generation from examples. *ACM Transactions on Graphics (SIGGRAPH 2002) 21*, 3, 483–490.

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.

ARIKAN, O., FORSYTH, D., AND O'BRIEN, J. 2005. Pushing people around. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 59–66.

BANDI, S., AND THALMANN, D. 1997. A configuration space approach for efficient animation of human figures. In *Proc. of IEEE Non Rigid and Articulated Motion Workshop*, IEEECS Press.

BINDIGANAVALE, R., GRANIERI, J. P., WEI, S., ZHAO, X., AND BADLER, N. I. 1994. Posture interpolation with collision avoidance. In *Proceedings of Computer Animation '94*, 13–20.

CHOI, M. G., LEE, J., AND SHIN, S. Y. 2003. Planning biped locomotion using motion capture data and probabilistic roadmaps. *ACM Transactions on Graphics 22*, 2, 182–203.

DUDA, R. O., HART, P. E., AND STORK, D. G. 2000. *Pattern Classification*. Wiley-Interscience.

GLEICHER, M. 1998. Retargeting motion to new characters. In *Proceedings of SIGGRAPH 98*, 33–42.

HSU, E., GENTRY, S., AND POPOVIĆ, J. 2004. Example-based control of human motion. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 69–77.

JUNG, M. R., BADLER, N. I., AND NOMA, T. 1994. Animated human agents with motion planning capability for 3d-space postural goals. *The Journal of Visualization and Computer Animation 5*, 4, 225–246.

KALISIAK, M., AND VAN DE PANNE, M. 2001. A grasp-based motion planning algorithm for character animation. *The Journal of Visualization and Computer Animation 12*, 3, 117–129.

KIM, T., PARK, S. I., AND SHIN, S. Y. 2003. Rhythmic-motion synthesis based on motion-beat analysis. *ACM Transactions on Graphics (SIGGRAPH 2003) 22*, 3, 392–401.

KOGA, Y., KONDO, K., KUFFER, J., AND LATOMBE, J. 1994. Planning motions with intensions. *Proceedings of SIGGRAPH '94 28* (July), 395–408.

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

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

KUFFNER, J. J., NISHIWAKI, K., KAGAMI, S., INABA, M., AND INOUE, H. 2001. Footstep planning among obstacles for biped robots. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'01)*.

KWON, T., AND SHIN, S. Y. 2005. Motion modeling for on-line locomotion synthesis. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 29–38.

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.

LATOMBE, J. C. 1991. *Robot Motion Planning*. Kluwer Academic Publishers.

LAU, M., AND KUFFNER, J. J. 2005. Behavior planning for character animation. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 270–280.

LEE, J., AND LEE, K. H. 2004. Precomputing avatar behavior from human motion data. In *SCA '04: Proceedings of the 2004*

Motion	Unit objects		Size	Cell partitioning			Tilable	
patches	# of units	objects	$(cm)^2$	grids	cells	transitions	entries/exits	connections
playground1	2	two panels	80×40	16×8	9037	235026		
playground2	2	two panels	80×40	16×8	30199	246705		
playground3	2	two panels	80×40	16×8	1978	83291		
playground4	2	two panels	80×80	16×16	2837	23584		
playground5	2	two panels	80×80	16×16	2012	39832		
playground6	2	panel and curved slide	123×83	24×16	1578	66826		
playground7	2	panel and curved slide	121×82	24×16	1008	18013		
playground8	2	two panels	120×40	24×8	876	32065		
playground9	2	panel and slide up	59×42	11×8	1079	27956		
playground10	1	slide middle	83×42	16×8	212		1	17
playground11	2	panel and slide down	58×42	11×8	581	6293		
idle	1	panel	120×120	24×24	73	610		
chat	1	panel	120×120	24×24	74	1054		
dispute	1	panel	120×120	24×24	1453	105571		
presentation	1	panel	120×120	24×24	130	4324		
sit down/stand up	1	panel-desk-chair	160×160	32×32	771	0		
work at the desk	1	panel-desk-chair	120×120	24×24	84	741		
chat at the desk	1	panel-desk-chair	120×120	24×24	1428	53879		
walk-to-stop	1	panel	120×120	24×24	116	0		
walk	1	panel	120×120	24×24	75973		144	7967

Table 2: The motion patches constructed in our experiments.

ACM SIGGRAPH/Eurographics Symposium on Computer Animation, 79–87.

LEE, J., AND SHIN, S. Y. 1999. A hierarchical approach to interactive motion editing for human-like figures. In *Proceedings of SIGGRAPH 99*, 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.

NOSER, H., RENAULT, O., THALMANN, D., AND THALMANN, N. M. 1995. Navigation for digital actors based on synthetic vision, memory, and learning. *Computer & Graphics 19*, 1, 7–19.

PARK, S. I., SHIN, H. J., KIM, T., AND SHIN, S. Y. 2004. On-line motion blending for real-time locomotion generation. *Computer Animation and Virtual Worlds 15*, 3, 125–138.

PETTRE, J., LAUMOND, J.-P., AND SIMEON, T. 2003. A 2-stages locomotion planner for digital actors. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 258–264.

PULLEN, K., AND BREGLER, C. 2002. Motion capture assisted animation: Texturing and synthesis. *ACM Transactions on Graphics (SIGGRAPH 2002) 21*, 3, 501–508.

REITSMA, P. S. A., AND POLLARD, N. S. 2004. Evaluating motion graphs for character navigation. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 89–98.

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

SHILLER, Z., YAMANE, K., AND NAKAMURA, Y. 2001. Planning motion patterns of human figures using a multi-layered grid and the dynamics filter. In *Proc. of the International Conference on Robotics and Automation 2001*.

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.

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

SUNG, M., KOVAR, L., AND GLEICHER, M. 2005. Fast and accurate goal-directed motion synthesis for crowds. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 291–300.

TARJAN, R. 1972. Depth first search and linear graph algorithms. *SIAM Journal of Computing 1*, 146–160.

THORNE, M., BURKE, D., AND VAN DE PANNE, M. 2004. Motion doodles: An interface for sketching character motion. *ACM Transactions on Graphics (SIGGRAPH 2004) 23*, 3, 424–431.

WANG, J., AND BODENHEIMER, B. 2004. Computing the duration of motion transitions: An empirical approach. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 337–346.

YAMANE, K., KUFFNER, J. J., AND HODGINS, J. K. 2004. Synthesizing animations of human manipulation tasks. *ACM Transactions on Graphics (SIGGRAPH 2004) 23*, 3, 532–539.