EUROGRAPHICS 2011 / M. Chen and O. Deussen
(Guest Editors)

Volume 30 (2011), Number 2

Deformable Motion: Squeezing into Cluttered Environments

Myung Geol Choi, Manmyung Kim, Kyung Lyul Hyun and Jehee Lee

Seoul National University

Figure 1: *Our deformable motion allows animated characters to navigate through highly constrained environments in realtime interactive control system. (left) Many cylinder world. (middle) Jump and climb. (right) Probabilistic roadmap.*

Abstract

We present an interactive method that allows animated characters to navigate through cluttered environments. Our characters are equipped with a variety of motion skills to clear obstacles, narrow passages, and highly constrained environment features. Our control method incorporates a behavior model into well-known, standard path planning algorithms. Our behavior model, called deformable motion*, consists of a graph of motion capture fragments. The key idea of our approach is to add flexibility on motion fragments such that we can situate them into a cluttered environment via constraint-based formulation. We demonstrate our deformable motion for realtime interactive navigation and global path planning in highly constrained virtual environments.*

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

1. Introduction

Interactively controlling animated characters in complex virtual environments is a recurring problem in computer graphics. The problem has previously been addressed by employing state-space searching techniques and constructing precomputed structures for environments and control policies. Even with remarkable advances in relevant research fields, steering animated characters through large, tightly constrained environments is still challenging because many of existing techniques are either memory-intensive or computation-intensive. The requirement for compelling character motion poses even more challenges to the already difficult problem.

Data-driven animation using motion capture data has become a standard practice in character animation. A number of techniques have been developed to add flexibility on captured human motion data by editing joint trajectories, warping motion paths, blending a family of parameterized motions, and adapting motion to new characters and environments. However, situating motion data in a highly constrained environment requires appreciable computational resources for interactive graphics applications.

We present an interactive technique that allows animated characters to navigate through cluttered environments. The characters are equipped with a variety of behavior skills to clear obstacles, narrow passages, and highly constrained environment features. Behavior skills are represented as a graph of motion capture data. Our character strings a sequence of motion fragments together to animate itself. The key idea of our approach is to consider the sweep of each motion fragment as a three-dimensional deformable object, which we can "squeeze" into a cluttered environment. Our deformable motion can be incorporated into many existing path planning and state-space search techniques to facilitate interactive character control in challenging virtual environments. In this paper, we demonstrate our deformable motion for real-time interactive navigation and path planning by adopting limited-horizontal state-space search, probabilistic roadmaps, and rapidly-exploring random trees.

2. Related Work

Motion data are captured from specific subjects, at specific environments, and in specific style and mood. There exists a vast literature on reusing motion data at different conditions than they were originally captured. To name a few, Gleicher [Gle98] retargetted the motion of one character to another via optimization. Lee and Shin [LS99] addressed interactive motion editing with arbitrary accuracy using hierarchical displacement mapping. Motion editing techniques have further been elaborated to exploit physics-based motion style [LHP05], interactively manipulate a group of pedestrians [KLLT08] and synchronized multiple character motions [KHKL09], and retarget highly-constrainted interaction motions to diverse sizes of bodies [HKT10]. Statistical analysis can provide a priori of human motions for motion data editing [GMHP04, MCC09].

Constructing a graph of motion fragments has become a common practice in character animation [LCR*02, KGP02]. Motion graphs have evolved to assimilate a family of parameterized motions [SO06, HG07, SH07]. Animating a character using motion graphs requires a control method that searches through motion graphs to retrieve a series of motion fragments, along which the character is animated. A number of control methods have been explored including A*-search [KGP02, SH07], limited-horizon search [LCR*02], and dynamic programming [AFO03]. The control policies can be explicitly constructed and tabulated via reinforcement learning [LL04, TLP07, MP07, LZ08, LLP09].

Many of path planning algorithms assume a simple rigid mover that can translate and rotate in any direction without restriction [LaV06]. Recently, path planning with a sophisticated mover model, such as deformable models [GSLM05, MLM08], articulated figures [LK05, ZPM03], biped robots [KNK*03] has been extensively explored. Brock and Khatib [BK00b] introduced a local planner called elastic strips for mobile manipulation robots. The algorithm partially updates an existing trajectory with dynamically changing environments at runtime. This method is extended to satisfy task constraints of a manipulation robot [BKV02]. The path planning algorithm combined with a motion graph allows an animated character to navigate through complex virtual environments. Choi el al. [CLS03] randomly sampled footprint configurations and built a probabilistic roadmap by connecting footprints using available motion fragments. Similar techniques based on probabilistic roadmaps have been studied by other research groups [PLS03, EAPL06]. Particularly, Pettre et al. [PLS03] cleared minor interferences with environments by warping arms and spine trajectories. Lamarche [Lam09] applied a motion blending technique for navigation in a passage with varying height. While going through the passage, the character crocuches down or stands up smoothly to adapt his body to an arbitrary height of ceiling. Reitsma and Pollard [RP04] evaluated motion graphs for character navigation by "unrolling" the graph in a specific enviroment. The resulting structure is conceptually similar to a probabilistic roadmap. Lau and Kuffner [LK06] demonstrated that precomputation with motion data considerably improves the search speed at runtime. Lee and Lee [LCL06] addressed character motion planning in a complex virtual environment, such as jungle gym, by encapsulating motion data captured at one place of the environment to reuse at another place.

Many existing techniques suffer from bottlenecks at narrow passages and highly constrained places [HKL*98]. Global path planning algorithms are usually equipped with a local planner that decides if there exists a short-term path between two configurations. The use of a powerful local planner often alleviates difficulty at bottlenecks. Our work does not attempt to replace any of aforementioned searching/planning methods. Our deformable motion provides an effective local planner that supplements existing methods to clear challenging environment features. With deformable motion, many of existing techniques explore much less states to find a clear path through narrow passages, achieving significant speedup and saving on memory usage.

3. Deformable Motion

Situating motion data into an environment poses several types of constraints. At first, motion data should not penetrate through the ground, walls, and obstacles (non-penetration constraints). The animated character's feet, hands, and other body parts should have contacts with environment objects in right place at right time (contact constraints). A continuous stream of motion data are typically represented as a sequence of motion fragments. Each fragment should meet its predecessor and successor with smooth position, orientation, velocity changes (continuity constraints). Each motion fragment may have intrinsic constraints. For example, the character in the air should not change its moving direction (motion constraints).

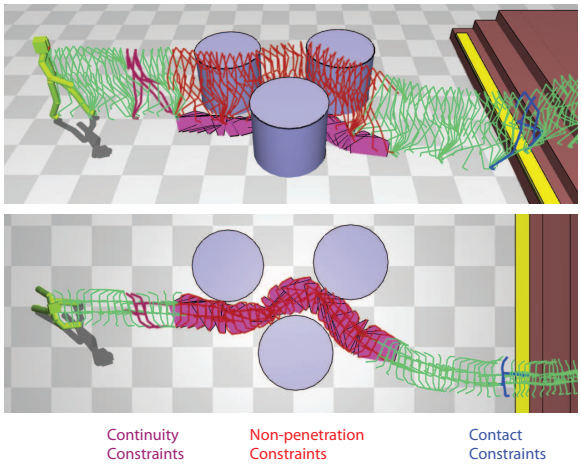


Figure 2: *Constraints for situating motion data into the environment. A series of motion fragments should be connected to each other smoothly. There should not be any interpenetration. The convex boundaries of character poses are shown in pink. The character should step on the right place (the yellow region) to start walking on the stairs.*

We first consider a simple mechanism for interactive character control. Elaborated control strategies will be discussed later in the next section. The character is equipped with a repertoire of action skills, described as motion fragments. Whenever it finishes one action, it has to choose the best next action to proceed among available choices. To do so, it has to evaluate all available candidates with respect to environment conditions and its goal (such as a target location specified by the user) subject to non-penetration, contact, continuity, and motion constraints.

We consider the sweep of each motion fragment as a flexible object that undergoes deformation subject to user constraints. A general deformable model could be constructed by generating a sweep of mass points along joint trajectories and connecting them with each other using a web of damped springs. Alternatively, a volumetric FEM model could be constructed by tetrahedrizing mass points. Recent advances in deformable model research would allow us to physically simulate the deformable model subject to any type of constraints that can be written as mathematical equations. However, our stringent performance requirement for interactive control prevented us from employing such a fully general approach. Instead, in this section, we present a simple, efficient, approximate technique based on motion editing techniques.

3.1. Constraints

We consider four types of constraints that are either identified automatically or annotated manually at relevant environment objects or motion fragments.

Continuity constraints are automatically specified to pin down the position and direction of the character at the beginning of each motion fragment. This allows smooth transitioning between motion fragments. Pinning position and direction on a motion path can be formulated as linear equations.

Non-penetration constraints are automatically identified at runtime and formulated as inequalities. We designed virtual environments with convex objects for efficient collision detection. All concave features have been decomposed into convex polyhedra.

Contact constraints specify the interaction between the character and the environment. For example, the stance foot should be within a certain range from the first step of stairs and the body should face within a certain range of direction to start walking on the stairs. We annotated such preconditions at stairs and other objects to climb and jump over. Contact constraints are formulated as inequalities that specify the plausible range of position and direction of any body part.

Motion constraints specify how each motion fragment would deform. Deforming broad jump allows broader/narrower jumps, but does not allow the bending of its motion path while the character being in the air. Conversely, locomotion is resilient to bending, but susceptible to stretching or squashing. Extended double stance phases (e.g., standing still) do not allow neither type of deformation. Each motion fragment is annotated what type of deformation is allowed.

3.2. Deformation

Motion deformation is computed at two levels. We first consider a three-dimensional path of the character's root (see Figure 2). The motion path is considered as an elastic string, which undergoes deformation subject to given constraints in an as-rigid-as-possible manner. Then, the character's poses at individual frames are further refined to squeeze into narrow passages. We employ a two-dimensional path editing method by Kim et al. [KHKL09] for path deformation. Our requirements necessitate its generalization in several directions such that it can cope with three-dimensional motion paths, inequality constraints, and deformation types.

Two-dimensional Path Editing. Let $\{(x_i, y_i, z_i)\}$ be the three-dimensional path of a motion fragment. The path editing algorithm of Kim et al. works on its two-dimensional projection $\{(x_i, z_i)\}$ subject to pinning (position and direction) constraints, which can be formulated as linear equations. The path editing algorithm consists of two steps: shape preserving deformation and scale compensation. At the first step, the goal of shape preservation often entails large scaling. The second step compensates scaling artifacts to maintain the original length of the path. The Laplacian formulation of shape preservation leads to a linear system $A\bar{\mathbf{p}} = \mathbf{0}$,

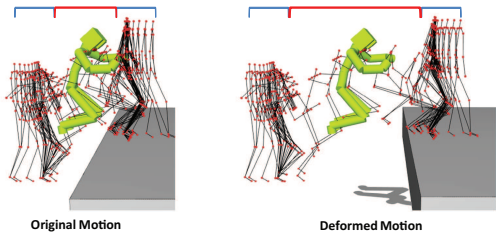


Figure 3: *Stretching a jump motion results in the extended flight phase while leaving the double-stance phases before and after the flight unchanged.*

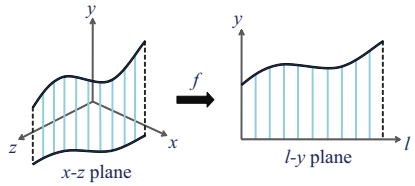


Figure 4: *3D path editing*

where $\bar{\mathbf{p}} = (\bar{x}_0, \bar{z}_0, \bar{x}_1, \bar{z}_1, \bar{x}_2, \bar{z}_2, \cdots)^\top$ is a long vector representing a deformed 2D path at the first step of the algorithm (see [KHKL09] for detailed derivation). Combining it with soft constraints $B\bar{\mathbf{p}} = \mathbf{b}$ forms an over-determined linear system:

$$M_1\bar{\mathbf{p}} = \left(\frac{W_1A_1}{B}\right)\bar{\mathbf{p}} = \begin{pmatrix} \mathbf{0} \\ \mathbf{b} \end{pmatrix} = \mathbf{m}_1, \qquad (1)$$

where W_1 is a diagonal matrix weighing how well the path bends subject to constraints. Similarly, the second step for scaling compensation forms an over-determined system:

$$M_2\hat{\mathbf{p}} = \left(\frac{W_2A_2}{B}\right)\hat{\mathbf{p}} = \begin{pmatrix} W_2\mathbf{a}_2 \\ \mathbf{b} \end{pmatrix} = \mathbf{m}_2, \qquad (2)$$

where $\hat{\mathbf{p}}$ is the final deformed path and W_2 is a diagonal matrix weighing how well the path stretches. For locomotion data with alternating single-support phases, we weigh more on the scale compensation step to discourage excessive stretching. For jumping data with flight phases, we weigh more on shape preservation to encourage stretching over bending (See Figure 3). Extended double stance phases are simply considered as a rigid segment of the path and handled using hard constraints. Putting hard constraints $H\bar{\mathbf{p}} = \mathbf{h}$ together, the first and second step of the algorithm become augmented linear systems

$$\begin{pmatrix} M_1^\top M_1 & H^\top \\ H & 0 \end{pmatrix}\begin{pmatrix} \bar{\mathbf{p}} \\ \lambda_1 \end{pmatrix} = \begin{pmatrix} M_1^\top \mathbf{m}_1 \\ \mathbf{h} \end{pmatrix}, \qquad (3)$$

$$\begin{pmatrix} M_2^\top M_2 & H^\top \\ H & 0 \end{pmatrix}\begin{pmatrix} \hat{\mathbf{p}} \\ \lambda_2 \end{pmatrix} = \begin{pmatrix} M_2^\top \mathbf{m}_2 \\ \mathbf{h} \end{pmatrix}, \qquad (4)$$

where λ_1 and λ_2 are Lagrange multipliers. Solving two augmented systems sequentially completes the 2D path editing.

Three-dimensional Motion Paths. Our virtual environment is three-dimensional and our virtual characters walk up and down the stairs, climb up to and jump down from any object in the environment. The two-dimensional path editing algorithm does not generalize easily to deal with three-dimensional paths because the Laplacian-based formulation for shape preservation leads to non-linear equations. Instead, we apply the 2D algorithm twice in horizontal and vertical planes to circumvent the non-linear formulation (see Figure 4). The first (horizontal) phase is same as the 2D algorithm. We project the 3D motion path $\{(x_i, y_i, z_i) | 1 \le i \le n\}$ onto the ground plane and deform its 2D projection $\{(x_i, z_i)\}$ to satisfy constraints in the plane. For the second (vertical) phase, we consider a curved manifold parallel to the y-axis that traces along the deformed 2D path. The arc length parameterization along the deformed 2D path $\{(\hat{x}_i, \hat{z}_i)\}$ flattens the curved manifold to generate another plane curve $\{(l_i, y_i)\}$,

$$l_i = \sum_{j=1}^{i} \sqrt{(\hat{x}_i - \hat{x}_{i-1})^2 + (\hat{z}_i - \hat{z}_{i-1})^2}, \qquad (5)$$

to which we apply the 2D editing algorithm again to satisfy constraints in the $l-y$ plane. The deformed 2D curve $\{(\tilde{l}_i, \tilde{y}_i)\}$ finally transforms back to the 3D curved manifold:

$$\begin{aligned} \tilde{x}_i &= \frac{(\hat{l}_i - l_k)\hat{x}_{k+1} + (l_{k+1} - \hat{l}_i)\hat{x}_k}{(\hat{l}_i - l_k) + (l_{k+1} - \hat{l}_i)} \\ \tilde{y}_i &= \hat{y}_i \\ \tilde{z}_i &= \frac{(\hat{l}_i - l_k)\hat{z}_{k+1} + (l_{k+1} - \hat{l}_i)\hat{z}_k}{(\hat{l}_i - l_k) + (l_{k+1} - \hat{l}_i)}. \end{aligned} \qquad (6)$$

where k is chosen such that $l_k < \hat{l}_i < l_{k+1}$. $k = 1$ if $\hat{l}_i \le l_1$ and $k = n-1$ if $l_n \le \hat{l}_i$.

Inequality Conditions. Though both non-penetration and contact constraints are formulated as inequalities, they affect motion paths differently. Non-penetration constraints are repulsers, while contact constraints are attractors. We resolve inequality constraints by incrementally adding supplementary equality constraints (see Figure 5). Each repulser having intersection with an animated character adds an equality constraint that pushes off the deepest penetration point out to the nearest safe location. Let $\mathbf{q}$ be the deepest penetration point on the body at frame i and $\mathbf{q}_d$ be the nearest non-penetrating point. The root position $\mathbf{p}_i$ at frame i should be translated by $\mathbf{q}_d - \mathbf{q}$ to resolve the penetration. The supplementary constraint is

$$\tilde{\mathbf{p}}_i = \mathbf{p}_i + (\mathbf{q}_d - \mathbf{q}) \qquad (7)$$

The penetration depth between convex obstacles and convex bounding volumes can be computed efficiently [KLM02]. Each attractor adds an equality constraint to pull its asso-

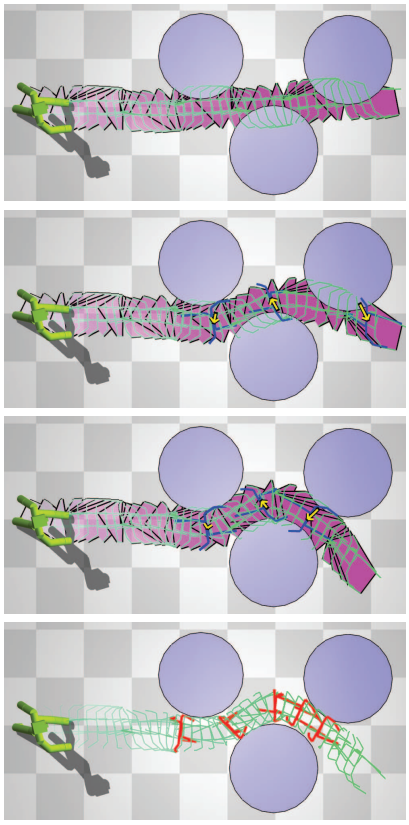


Figure 5: *Resolving non-penetration constraints. (left) Original motion fragment. (middle left) The first iteration. (middle right) The second iteration. (right) Motio refinement at pose-level.*

ciated end-effector into the contact region if they are sufficiently close. Let $\mathbf{q}$ and ϕ be the position and orientation of the end-effector when they are closest to their plausible ranges at frame i and $\mathbf{q}_d$ and ϕ_d be their desired values within the plausible ranges. It poses a supplementary position/direction pinning constraint at the root trajectory:

$$T(\tilde{\mathbf{p}}_i, \tilde{\theta}_i) = T(\mathbf{q}_d, \phi_d) \cdot T^{-1}(\mathbf{q}, \phi) \cdot T(\mathbf{p}_i, \theta_i) \qquad (8)$$

where $T\big((x,y,z),\theta\big) = \begin{pmatrix} cos\theta & 0 & sin\theta & x \\ 0 & 1 & 0 & y \\ -sin\theta & 0 & cos\theta & z \\ 0 & 0 & 0 & 1 \end{pmatrix}$. This process repeats until all inequalities are satisfied or adding a new supplementary constraint does not improve the result.

Fullbody refinement. In general, non-penetration constraints cannot be fully satisfied via path editing at narrow passages. We further refine motion data by using inverse kinematics and hierarchical displacement mapping [LS99]. The refinement process is similar to path editing. Each repulser identifies the deepest penetrating point on the skeleton at any frame and adds an inverse kinematics constraint to push it off to the nearest safe location. A collection of inverse kinematics constraints thus specified deform the motion data smoothly by using hierarchical displacement mapping. This repeats until there is no more penetration or penetration does not improve any more.

4. Interactive Control

Our deformable motion can be incorporated into various search and path planning algorithms. In this section, we discuss the use of our deformable motion in three algorithms. One is a local planner based on limited-horizon best-first search, which allows realtime interactive control. The others are global path planners based on probabilistic roadmaps and rapidly-exploring random trees that facilitate challenging motion planning in highly constrained environments.

4.1. Limited-Horizon Search

A*-search algorithm is a standard technique that finds the least-cost path from a given initial configuration to a goal. Though it has been widely used, it is often not suitable for realtime interactive applications because finding the optimal path is computationally demanding. Limited-horizon best-first search algorithm is identical to A*-search algorithm except that it focus on achieving constant frame rates than finding the optimal path.

Deformable Motion in Search Algorithms. The character is provided with a set of motions from which it can choose its next movement. A*-search algorithm systematically expands their future choices until the optimal path is found. Whenever a new choice of motion is examined, our deformable motion is employed to situate the motion into the context. If the motion should be deformed beyond a user-specified threshold, that choice is pruned. In our experiments, we allow stretching of 90% to 110% and bending of 2 degrees at each frame. The limited-horizon search algorithm follows the same procedure, but expands a limited number of motion choices at every frame of simulation. Therefore, it may terminate before the optimal path is found and possibly ends up with a suboptimal choice. The limited-horizon search is useful for realtime applications for which interactivity is more important than optimality. It is also effective as a local planner being part of global path planning algorithms.

Cost and Heuristics. The cost function for taking a series of motion fragments is:

$$C = \sum \left(c_1 E_{\text{path}} + c_2 E_{\text{pose}} + E_{\text{motion}} D\right), \qquad (9)$$

where E_{path} is the energy of path deformation, E_{pose} is the total sum of weighted joint angle changes, E_{motion} is the preference of taking each motion fragments, and D is the distance travelled. In our experiments, straight walking is preferred

over all the other motions. The admissible heuristics function assumes zero path deformation, zero pose deformation, the most preferred motion choices, and the straight distance to the goal.

Amortizing Computation. Searching with motion graphs performs at the granularity of motion fragments. Therefore, searching algorithm is needed only at the end of each fragments and all the other frames simply play back without much computation. This simple strategy makes the character to move intermittently at runtime. We amortize the computation load evenly over all frames to achieve constant frame rates. The searching algorithm expands a constant number of motion choices and stalls at every frame. At the next frame, it resumes to search unexplored choices further. This procedure repeats until it reaches the end of a fragment and select the next motion choice based the cumulative search result. Amortizing computation achieves a constant frame rate at runtime for realtime interactive system.

4.2. Global Path Planning

A **probabilistic Roadmap** consists of a number of nodes (representing the character's position, direction, and pose) randomly sampled from the free space of the environment. A directional edge connects a pair of nodes if a local planner can find a short-term path between them. Including start and goal configurations into a roadmap, planning the character's motion through a virtual environment is reduced to graph-searching through the roadmap. The probabilistic roadmap could be constructed to densely cover the entire environment at the preprocessing phase [BK00a]. Precomputing a dense roadmap for large, high-dimensional configuration space demands significant computational resources, but allows runtime path-finding queries to be handled efficiently. Alternatively, a lazy evaluation approach constructs a roadmap on-the-fly for each path-finding query. The lazy evaluation attempts to find a trade-off between enormous precomputation costs and runtime performance. We construct a probabilistic roadmap in a lazy manner. Given start and goal configurations, we incrementally add random samples to the roadmap until a path between the start and the goal is found. Each new sample is connected to the existing roadmap by using our local planner based on deformable motion and the limited-horizon A*-search algorithm. The lazy evaluation allows us to roughly measure the minimal number of nodes required to achieve a goal.

Rapidly-exploring random trees (RRTs) are intended for solving single-query path planning problems without preprocessing [KL00]. There exist several variants of RRTS. Our implementation is based on its simplest variant combined with our deformable motion. A RRT is rooted at the start point and grows to explore the free space by iteratively sampling new nodes. We randomly sample a free configuration (position and direction) and find its closest node belong to the search tree. We then examine if a motion fragment heading towards the new sample configuration starting from its closest node can be situated in the environment subject to given constraints. If so, a new branch is added to the tree. This process repeats until the search tree reaches the goal location.

	motion(starting foot)	frame size	Cylinder Worlds	PRM	Jump and Climb
walks	half cycle of a walk(L)	20	o		o
	half cycle of a walk(R)	20	o		o
	one cycle of a right-truning (L)	44	o		o
	one cycle of a right-truning (R)	44	o		o
	one cycle of a left-truning (L)	44	o		o
	one cycle of a left-truning (R)	44	o		o
	U-turning(L)	60	o		
	U-turning(R)	64	o		
	transition from normal-walk to side-walk(R)	40	o		
	transition from normal-walk to side-walk(L)	39	o		
	transition from side-walk to normal-walk (L)	21	o		
	transition from side-walk to normal-walk(R)	21	o		
	one cycle of a side-walk(L)	41	o		
	one cycle of a side-walk(R)	42	o		
interactions	crawl(L)	113			o
	jump onto box(L)	125			o
	going up stair(R)	66			o
	going up stair(L)	78			o
	going down stair(R)	67			o
	going down stair(L)	66			o
	jump(R)	76			o
	jump(L)	52			o
	climbing up block(R)	152			o
	climbing up block(L)	115			o
	climbing down block(R)	113			o
	climbing down block(L)	99			o
short steps	turning in place-right45	33		o	
	turning in place-left45	39		o	
	turning in place-right90	43		o	
	turning in place-left90	50		o	
	turnning in place-180	59		o	
	short step - front	39		o	
	short step -right	34		o	
	short step -left	45		o	

Figure 6: *Motion data in our experiments.*

5. Experimental Results

All motion data in our experiments were originally captured using a Vicon optical motion capture system at the rate of 120 frames/second and then down-sampled to 30 frames/second. Motion data include walking, turning in 45, 90, 180 degrees, side walking, short stepping, jumping over, waling up/down stairs, and climbing up/down blocks (see Figure 6). Each motion fragment is 20 frames (0.66 second) to 150 frames (5 second) long. We built a motion graph manually from the data set. The motion graph has 7 to 8 outgoing branches at every node.

Many Cylinder World. We applied our limited-horizon local planner to an environment with many cylindrical obstacles (see Figure 1(left)). Each cylinder is about the shoulder height. The clearance between obstacles is narrower (45cm to 50cm) than the character's shoulder width, so the character must either squeeze in or side steps to pass through the obstacles. The planner explored about 50 to 60 nodes at each branch. Even with such a simple, standard search technique, our deformable model made it possible for the character to navigate through obstacles fluidly. The computation load was evenly amortized to achieve the animation of 30 frames/second without massive precomputation. Collision detection was the bottleneck of computation. Our im-

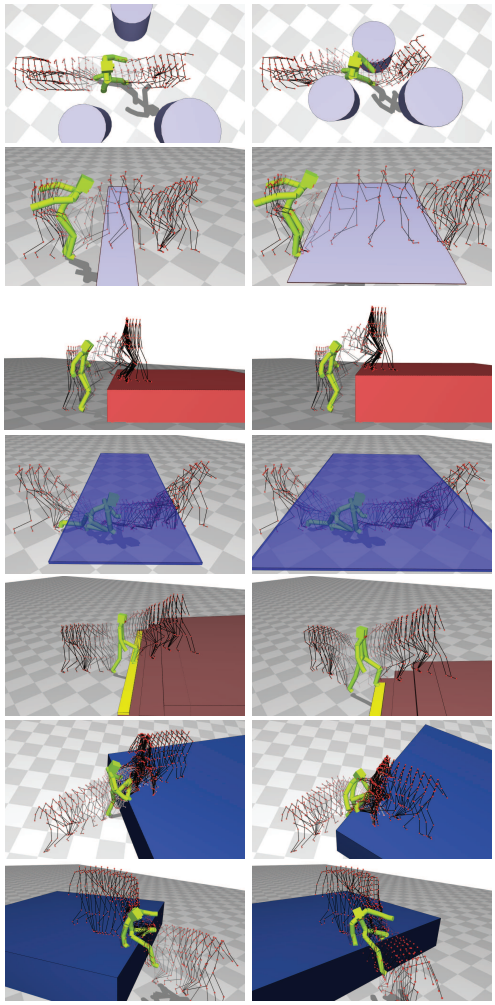


Figure 7: *The deformable motion fragments used for jump and climb example. The captured motion data (Left) are deformed to fit into environment features (Right).*

plementation of penetration depth computation was not optimized. Better implementation would achieve much better performance.

Jump and Climb. The Jump-and-Climb environment includes stairs, blocks to climb up, and obstacles to jump over or crawl below (see Figure 1(middle)).

We employed uniform spatial partitioning to cull collisions between the character and the objects. The contact constraints annotated at each object worked effectively to align motion fragments with respect to the environment features at runtime (see Figure 7). The example uses five motion fragments that comes with contact constraints: JumpUp, StairUp, StairDown, ClimbUp and ClimbDown. For JumpUp motion, we set contact constraints at both feet when they first land on the top surface of the block(see third row in Figure 7). For StairUp and StairDown motions, the foot should land within the top surface of the first stairstep to start to walk up / down the stairs (see fifth row in Figure 7). While the character climbs up / down a block, the contact state changes dynamically. Instead of specifying contact constraints at hands and feet, we pinned down the position and orientation of the character's root with respect to the block's coordinate system(see sixth and last row in Figure 7). Our local planner allowed the user to control the character interactively in the environment with highly constrained features. The motion graph consists of 17 deformable motion fragments. The limited-horizon algorithm searched about 300 motion choices at every branch, which roughly corresponds to the size of a fully-expanded search tree of depth two. Amortized computation at each individual frame divides the workload by the number of frames (23 frames in the shortest fragment and 152 frames in the longest). The average computation at each frame takes about 10 msec.

Comparison Test. We quantitatively compared our deformable motion and linear warping in a series of environments with progressively larger obstacles (see Figure 9). The linear warping of a motion fragment steers its moving trajectory by evenly rotating the character pose by θ/n at every frame with respect to its previous frame to achieve θ-degree steering in total. We conducted experiments six times for each environment with different random seeds and averaged the number of expanded nodes. The RRT algorithm with a linear warping local planner works well in a sparse environment with abundant free space, but its performance drops down rapidly with denser environments. The number of expanded nodes increases exponentially with the density of the environment. In comparison, our deformable motion allows the RRT planner to find a clear path by exploring much less nodes. The size of the search tree does not exhibit an exponential tendency until a clear path does not exist any more.

Narrow Passages. A probabilistic roadmap was constructed for a very tight environment (see Figure 1(right)). The minimal clearance is below 30 cm, so even side stepping motion should squeeze in through the narrowest passage. The roadmap was constructed in a lazy manner such that it is fully connected through every pair of obstacles. Such a roadmap required 72 nodes and 192 edges. Direct quantitative comparison with previous approaches is difficult because the environment in our experiments does not allow solution paths when a local planner based on linear warping or linear blending is used. Our roadmap is two orders of magnitude smaller than the roadmap of a similarly-sized environment reported in Choi et al [CLS03].

Dynamic Environment. Our deformable motion can cope with dynamically-changing environments if the changes are not drastic. Figure 8 shows an animated character jumping over between boxes, of which heights change

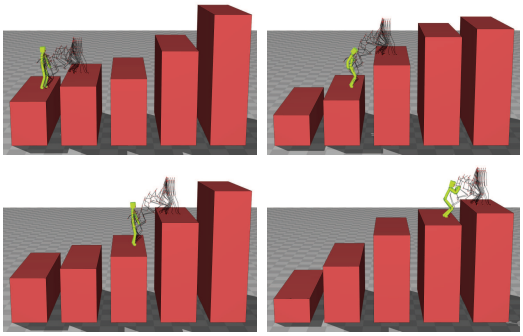
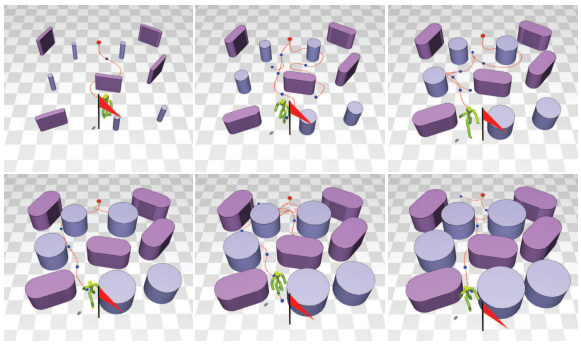


Figure 8: *Navigating in a dynamic environment.*



Free Space	Deformable Motion	Linear Warping
93%	10.71(0.28)	9.43(1.29)
84%	14.00(1.29)	30.43(13.29)
70%	14.57(4.29)	63.86(36.71)
53%	29.00(7.14)	1007.29(803.57)
44%	32.00(10.43)	∞
33%	81.43(43.86)	∞

average # of sampling (failure)

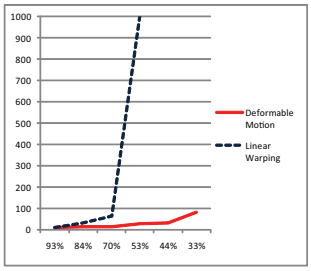


Figure 9: *The RRT-based path planning algorithms with different local planners are evaluated with a series of progressively denser environments. The x-axis of the graph is the density (the ratio of free space) of the environment and the y-axis is the number of expanded nodes.*

continuously over time. Contact constraints at the beginning and end of each jump modulates the deformable motion continuously to deal with dynamic changes.

6. Discussion

Our deformable motion provides a powerful local planner for data-driven animated characters, which enables even simple search/planning algorithms to find clear paths through narrow passages and highly constrained environments. Situating motion data into the context was explicitly formulated using four types (continuity, non-penetration, contact, and motion) of constraints. The recent advances in motion editing techniques allowed motion data to be deformed efficiently subject to constraints. Our deformable motion is a simple, yet effective and practical solution that can be used in a wide variety of virtual environment applications.

We built succinct motion graphs manually from a set of motion fragments. Although the graph construction is not a burden with tens of motion fragments, recent research result [ZNKS09] could be employed to automate the construction process.

The body of our character consists of simple rigid bodies, such as cylinders and boxes, which simplify the penetration calculations. Handling path and skin deformations in a uniform fashion would allow us to achieve improved realism in character animation.

Acknowledgements

This research was partly supported by MKE/MCST strategic technology development program (Project No. 2008-F-033-02).

References

[AFO03] ARIKAN O., FORSYTH D. A., O'BRIEN J. F.: Motion synthesis from annotations. *ACM Transactions on Graphics (SIGGRAPH 2003) 22*, 3 (2003). 2

[BK00a] BOHLIN R., KAVRAKI L. E.: Path planning using lazy prm. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'2000)* (April 2000), vol. 1, pp. 521–528. 6

[BK00b] BROCK O., KHATIB O.: Real-time replanning in high-dimensional configuration spaces using sets of homotopic paths. In *Proceeding of the IEEE International Conference on Robotics and Automation (ICRA'2000)* (2000), pp. 550–555. 2

[BKV02] BROCK O., KHATIB O., VIJI S.: Task-consistent obstacle avoidance and motion behavior for mobile manipulation. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA'2000)* (2002), pp. 388–393. 2

[CLS03] CHOI M. G., LEE J., SHIN S. Y.: Planning biped locomotion using motion capture data and probabilistic roadmaps. *ACM Transactions on Graphics 22*, 2 (2003), 182–203. 2, 7

[EAPL06] ESTEVES C., ARECHAVALETA G., PETTRÉ J., LAUMOND J.-P.: Animation planning for virtual characters cooperation. *ACM Transactions on Graphics 25*, 2 (2006). 2

[Gle98] GLEICHER M.: Retargeting motion to new characters. In *Proceedings of SIGGRAPH 98* (July 1998), pp. 33–42. 2

[GMHP04] GROCHOW K., MARTIN S. L., HERTZMANN A., POPOVIĆ Z.: Style-based inverse kinematics. *ACM Transaction on Graphics (SIGGRAPH 2004) 23*, 3 (2004), 522–531. 2

[GSLM05] GAYLE R., SEGARS W., LIN M. C., MANOCHA D.: Path planning for deformable robots in complex environments. In *Proceedings of Robotics: Systems and Science* (2005). 2

[HG07] HECK R., GLEICHER M.: Parametric motion graphs. In *Proceedings of the 2007 symposium on Interactive 3D graphics and games* (2007), pp. 129–136. 2

[HKL*98] HSU D., KAVRAKI L. E., LATOMBE J.-C., MOTWANI R., SORKIN S.: *On Finding Narrow Passages with Probabilistic Roadmap Planners*. A.K. Peters, 1998, pp. 141–153. 2

[HKT10] HO E. S. L., KOMURA T., TAI C.-L.: Spatial relationship preserving character motion adaptation. *ACM Transactions on Graphics 29*, 4 (2010), 1–8. 2

[KGP02] KOVAR L., GLEICHER M., PIGHIN F.: Motion graphs. *ACM Transactions on Graphics (SIGGRAPH 2002) 21*, 3 (2002), 473–482. 2

[KHKL09] KIM M., HYUN K. L., KIM J., LEE J.: Synchronized multi-character motion editing. *ACM Transactions on Graphics (SIGGRAPH 2009) 28*, 3 (2009). 2, 3, 4

[KL00] KUFFNER J. J., LAVALLE S. M.: Rrt-connect: An efficient approach to single-query path planning. In *Proceeding of the IEEE International Conference on Robotics and Automation (ICRA'2000)* (2000), pp. 995–1001. 6

[KLLT08] KWON T., LEE K. H., LEE J., TAKAHASHI S.: Group motion editing. *ACM Transactions on Graphics (SIGGRAPH 2008) 27*, 3 (2008). 2

[KLM02] KIM Y. J., LIN M. C., MANOCHA D.: DEEP: Dual-space expansion for estimating penetration depth between convex polytopes. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA'2002)* (2002). 4

[KNK*03] KUFFNER J. J., NISHIWAKI K., KAGAMI S., INABA M., INOUE H.: Motion planning for humanoid robots. In *Proceedings of International Symposium on Robotics Research (ISRR'03)* (October 2003). 2

[Lam09] LAMARCHE F.: Topoplan: a topological path planner for real time human navigation under floor and ceiling constraints. *Computer Graphic Forum (EUROGRAPHICS'2009) 28*, 2 (2009), 649–658. 2

[LaV06] LAVALLE S. M.: *Planning Algorithms*. Cambridge University Press, 2006. 2

[LCL06] LEE K. H., CHOI M. G., LEE J.: Motion patches: building blocks for virtual environments annotated with motion data. *ACM Transactions on Graphics (SIGGRAPH 2007) 26*, 3 (2006). 2

[LCR*02] LEE J., CHAI J., REITSMA P. S. A., HODGINS J. K., POLLARD N. S.: Interactive control of avatars animated with human motion data. *ACM Transactions on Graphics (SIGGRAPH 2002) 21*, 3 (2002), 491–500. 2

[LHP05] LIU C. K., HERTZMANN A., POPOVIĆ Z.: Learning physics-based motion style with nonlinear inverse optimization. *ACM Transactions on Graphics (SIGGRAPH 2005) 24*, 3 (2005), 1071–1081. 2

[LK05] LAU M., KUFFNER J. J.: Behavior planning for character animation. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA '05)* (2005), pp. 271–280. 2

[LK06] LAU M., KUFFNER J. J.: Precomputed search trees: Planning for interactive goal-driven animation. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA '06)* (2006), pp. 299–308. 2

[LL04] LEE J., LEE K. H.: Precomputing avatar behavior from human motion data. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA '04)* (2004), pp. 79–87. 2

[LLP09] LEE Y., LEE S. J., POPOVIĆ Z.: Compact character controllers. *ACM Transactions on Graphics (SIGGRAPH ASIA 2009) 28*, 6 (2009). 2

[LS99] LEE J., SHIN S. Y.: A hierarchical approach to interactive motion editing for human-like figures. In *Proceedings of SIGGRAPH 99* (1999), pp. 39–48. 2, 5

[LZ08] LO W.-Y., ZWICKER M.: Real-time planning for parameterized human motion. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA '08)* (2008), pp. 29–38. 2

[MCC09] MIN J., CHEN Y.-L., CHAI J.: Interactive generation of human animation with deformable motion models. *ACM Transaction on Graphics 29*, 1 (2009), 1–12. 2

[MLM08] MOSS W., LIN M. C., MANOCHA D.: Constraint-based motion synthesis for deformable models. *Computer Animation and Virtual Worlds 19*, 3-4 (2008), 421–431. 2

[MP07] MCCANN J., POLLARD N. S.: Responsive characters from motion fragments. *ACM Transactions on Graphics (SIGGRAPH 2007) 26*, 3 (2007). 2

[PLS03] PETTRE J., LAUMOND J.-P., SIMEON T.: 3d collision avoidance for digital actors locomotion. In *Proceeding of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'2003)* (2003), vol. 1, pp. 400–405. 2

[RP04] REITSMA P. S. A., POLLARD N. S.: Evaluating motion graphs for character navigation. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA '04)* (2004), pp. 89–98. 2

[SH07] SAFONOVA A., HODGINS J. K.: Construction and optimal search of interpolated motion graphs. *ACM Transactions on Graphics (SIGGRAPH 2007) 26*, 3 (2007). 2

[SO06] SHIN H. J., OH H. S.: Fat graphs: Constructing an interactive character with continuous controls. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA '06)* (2006). 2

[TLP07] TREUILLE A., LEE Y., POPOVIĆ Z.: Near-optimal character animation with continuous control. *ACM Transactions on Graphics (SIGGRAPH 2007) 26*, 3 (2007). 2

[ZNKS09] ZHAO L., NORMOYLE A., KHANNA S., SAFONOVA A.: Automatic construction of a minimum size motion graph. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA '09)* (2009). 8

[ZPM03] ZHANG L., PAN J., MANOCHA D.: Motion planning of human-like robots using constrained coordination. In *Proceedings of IEEE-RAS International Conference on Humanoid Robots (Humanoids09)* (2003). 2