Planning Biped Locomotion using Motion Capture Data and Probabilistic Roadmaps

MIN GYU CHOI
Korea Advanced Institute of Science and Technology
JEHEE LEE
Seoul National University
SUNG YONG SHIN
Korea Advanced Institute of Science and Technology

Typical high-level directives for locomotion of human-like characters are useful for interactive games and simulations as well as for off-line production animation. In this paper, we present a new scheme for planning natural-looking locomotion of a biped figure to facilitate rapid motion prototyping and task-level motion generation. Given start and goal positions in a virtual environment, our scheme gives a sequence of motions to move from the start to the goal using a set of live-captured motion clips. Based on a novel combination of probabilistic path planning and hierarchical displacement mapping, our scheme consists of three parts: roadmap construction, roadmap search, and motion generation. We randomly sample a set of valid footholds of the biped figure from the environment to construct a directed graph, called a roadmap, that guides the locomotion of the figure. Every edge of the roadmap is associated with a live-captured motion clip. Augmenting the roadmap with a posture transition graph, we traverse it to obtain the sequence of input motion clips and that of target footprints. We finally adapt the motion sequence to the constraints specified by the footprint sequence to generate a desired locomotion.

Categories and Subject Descriptors: I.3.7 [**Computer Graphics**]: Three-dimensional Graphics and Realism—*Animation*; G.3.0 [**Probability and Statistics**]: Probabilistic Algorithms—*Path Planning*

General Terms: Animation, Path Planning

Additional Key Words and Phrases: Biped locomotion, Human navigation, Motion editing and adaptation, Probabilistic path planning

This research was supported by grants from the NRL (National Research Laboratory) program of KISTEP (Korea Institute of Science & Technology Evaluation and Planning).
Author's address: M. G. Choi, Department of Electrical Engineering & Computer Science, Korea Advanced Institute of Science and Technology, 373-1 Kusong-dong, Yusong-gu, Taejon, 305-701, Korea; J. Lee, ERC for Advanced Control and Instrumentation, Seoul National University, San 56-1, Shillim-dong, Kwanak-ku, Seoul, Korea; S. Y. Shin, Department of Electrical Engineering & Computer Science, Korea Advanced Institute of Science and Technology, 373-1 Kusong-dong, Yusong-gu, Taejon, 305-701, Korea;

1. INTRODUCTION

1.1 Motivation and Objectives

The advent of motion capture systems offers a convenient means for acquiring realistic motion data. Due to the success of such systems, realistic and highly detailed motion clips are commercially available and widely used for producing visually convincing animations of human-like 3D characters in a variety of applications, such as animation films and video games. Efforts have been focused on editing and manipulating live-captured motion clips to provide effective ways of adapting motion clips to the desired constraints specified by animators [Bruderlin and Williams 1995; Gleicher 1998; Lee and Shin 1999; Rose et al. 1996; Unuma et al. 1995; Witkin and Popović 1995]. However, motion planning with such motion clips has not been explored, yet.

In this paper, we present a new planning scheme that produces a natural-looking motion for a human-like biped figure to move from a given start position to a goal position using a set of prescribed motions. An animation scenario is composed of certain tasks, each of which can be fulfilled with a sequence of motions. Task-level motion planning with canned motion clips can provide realistic motions at the early stage of the animation design for rapid motion prototyping, that facilitates early validation of an animation. It can also support an interactive animation, where a user-controlled character interacts with a synthetic environment. To generate realistic motions in such an interactive animation, the user needs task-level directives. However, traditional motion planning techniques [Hwang and Ahuja 1992; Latombe 1991] cannot achieve such directives with captured motion clips.

When the set of motions is limited to moving along a continuous path such as walking and running, our goal might simply be achieved by separating path planning from motion generation. That is, one can first plan a continuous moving path and then generate a motion along the path with captured motion data. However, this approach cannot properly take into account the intrinsic property of biped locomotion: A biped character can move from one place to another over disconnected regions by following a sequence of discrete footholds. Therefore, the continuous path does not reflect well all the motion clips available.

Based on a novel combination of probabilistic path planning [Kavraki et al. 1996] and hierarchical displacement mapping [Lee and Shin 1999], our scheme finds a sequence of input motion clips and that of target footprints simultaneously, and then retargets the motion sequence to yield a desired motion that follows the footprints. Provided with a rich set of canned motion clips, the scheme enables a human-like figure to perform a variety of motions such as running on flat terrain, jumping over a crevice, and walking over stepping stones to reach the goal.

1.2 Related Work

Planning locomotion of a human-like figure is related to several different areas of research. For our convenience, we classify these related works into five categories: biped locomotion, human navigation, probabilistic path planning, motion editing, and example-based motion synthesis.

Biped locomotion. Generating realistic biped locomotion has received increasing attention in computer animation. An excellent survey on this field can be found in [Multon et al. 1999]. Bruderlin and Calvert [1989] presented a goal-directed dynamic approach that generates a desired walking motion for given parameters such as velocity, step length, and step frequency. Boulic et al. [1990] exploited biomechanical data to utilize their intrinsic dynamics. Ko and Badler [1996] also presented a similar technique to produce dynamically-balanced walking that was further generalized for curved path walking. They also were able to generate footprints automatically. To generate a balanced walking, Laszlo et al. [1996] suggested the notion of limit cycle control that adds closed-loop feedback to open-loop periodic motions. Raibert and Hodgins [1991] showed that hand-designed controllers can produce physically realistic legged locomotion. Hodgins et al. [1995] extended those controllers to other motions of human athletics. Hodgins and Pollard [1997] also described an automatic method to adapt existing controllers to new characters. An optimization-based scheme was proposed by van de Panne [1997] to generate biped locomotion from given footprints. This scheme was further extended to quadruped locomotion [Torkos and van de Panne 1998]. Ko and Cremer [1995] proposed a real-time method to generate human locomotion from positional input streams. Chung and Hahn [1999] developed a hierarchical motion control system to generate walking motion along a path on uneven terrain. Sun and Metaxas [2001] exploited sagittal elevation angles to automate gait generation. Recently, procedural models for biped locomotion have also been available as commercial animation packages, such as Boston Dynamics "BDI Guy", Credo Interactive "Life Forms Studio", 3D Studio Max "Character Studio", and Avid "SOFTIMAGE|RTK". These packages generate biped locomotion from the user-specified moving trajectory of a character or his/her footprints.

Human navigation. Reynolds [1987] introduced reactive behaviors to simulate groups of simple creatures such as flocks of birds, herds of land animals, and schools of fishes. Terzopoulos et al. [1994] simulated reactive behaviors of artificial fishes with synthetic visions. Noser et al. [1996] presented a local navigation model for human-like characters using synthetic visions. They also simulated a human memory model for avoiding obstacles [Noser et al. 1995]. Kuffner and Latombe [1999] addressed a similar problem for dynamic environments. Reich et al. [1994] suggested a real-time model for human navigation over uneven terrain. They adopted simulated sensors to detect geometric features such as obstacles. Marti and Bunn [1994] considered large-scale terrain represented as a height field over a 2D uniform grid. Bandi and Thalmann [1998] discretized a synthetic environment into a 3D uniform grid to search paths for autonomous characters.

Probabilistic path planning. Barraquand and Latombe [1991] elaborated a randomized path planning technique, that is originally invented for escaping local minima in a potential field. Thereafter, Kavraki and Latombe [1994] and Overmars and Sv̌estka [1994] independently and jointly proposed similar methods [Kavraki et al. 1996] that randomly sample the configuration space as preprocessing, to construct a roadmap and then search the roadmap for a path during the planning stage. These methods have demonstrated good performance empirically for difficult problems,

such as navigating car-like robots with non-holonomic constraints and robots with many degrees of freedom. In recent years, theoretical foundations for such empirical successes have been established in some restricted cases [Barraquand et al. 1997; Kavraki et al. 1995; Kavraki et al. 1996; Overmars and Svěstka 1994]. Koga et al. [1994] combined randomized path planning and inverse kinematics to automatically generate animation for human arm manipulation. To coordinate multiple robots, Svěstka and Overmars [1998] combined roadmaps for multiple robots into a roadmap for the single composite robot. Kalisiak and van de Panne [2000] presented a grasp-based motion planning algorithm in a constrained 2D environment with designated handholds and footholds. Kindel et al. [2000] developed a path planner for a robot with dynamic constraints and verified its effectiveness both in real and simulated environments.

Motion editing. There have been a variety of efforts to develop motion editing tools. Bruderlin and Williams [1995] adopted signal processing techniques to manipulate animated motions. They introduced displacement mapping to alter a canned motion clip while preserving its detailed characteristics. Witkin and Popović [1995] proposed a motion warping technique for the same purpose. Unuma et al. [1995] used Fourier analysis techniques to interpolate and extrapolate motion data in the frequency domain. Rose et al. [1998] introduced the framework of "verbs and adverbs" to interpolate example motions with a combination of radial basis functions and low order polynomials. Lamouret and van de Panne [1996] discussed a variety of issues in reusing motion clips. Rose et al. [1996] generated seamless transitions between motion clips using spacetime constraints [Cohen 1992]. Gleicher [1998] simplified the spacetime problem for motion retargetting, that is, adapting a preexisting motion of a character for another character of the same structure and different size. Employing an optimization technique, he was able to achieve interactive performance for motion editing. To accelerate this approach, Lee and Shin [1999] presented a hierarchical displacement mapping technique based on the multilevel B-spline approximation. They also presented a fast inverse kinematics solver adopting the notion of an elbow circle given by Korein and Badler [1982]. Gleicher [2001] presented a method for path-based editing of existing motion data.

Example-based motion synthesis. Inspired by the work on "video texture" [Schödl et al. 2000], a number of researches have explored a method for synthesizing a new motion from captured motion data. Kovar et al. [2002] introduced the notion of a motion graph to represent transitions between poses of the captured motion data: A node of this graph represents a pose, and two nodes are connected by a directed edge if they can be followed from one to the other in an acceptable splice. They used branch and bound to search the graph for a motion following a sketched trajectory. Lee et al. [2002] also represented captured motion data with a similar graph structure, and provided effective user interfaces for interactive applications. Arikan and Forsyth [2002] built a hierarchical motion graph and applied a randomized search to extracting motions from the graph, which satisfy user-specified constraints such as their durations and joint angles at given keyframes. Pullen and Bregler [2002] developed a method for enhancing roughly-keyframed animation with captured motion data. To capture the stochastic and dynamic nature of motions, Li et al. [2002]

developed a two-level statistical model by combining low-level linear dynamic systems with a high-level Markov process. They used the model to produce a dance motion with variations in fine details. There are also similar researches that exploit two-level statistical models for motion synthesis [Bowden 2000; Brand and Hertzmann 2000; Molina-Tanco and Hilton 2000].

1.3 Overview

Given start and goal positions in a virtual environment, our objective is to find a sequence of motions of a biped figure to move from the start to the goal. Conventional motion planning techniques in robotics [Hwang and Ahuja 1992; Latombe 1991] typically generate very efficient mechanical movements rather than lifelike natural-looking motions desired in computer animation applications. On the other hand, motion editing techniques in computer grahics [Gleicher 1998; Lee and Shin 1999; Rose et al. 1996] are not equipped with a high-level planning capability to yield a desired motion. To rapidly plan convincing motions of the human-like character with high-level directives, we use a novel combination of probabilistic path planning [Kavraki et al. 1996] and hierarchical displacement mapping [Lee and Shin 1999]. Our scheme consists of the following three steps: roadmap construction, roadmap search, and motion generation. We briefly describe each of them to give an overall view on our scheme.

Roadmap construction. Given a virtual environment, we randomly sample valid configurations of a biped figure to construct a roadmap [Kavraki et al. 1996]. For efficiency, we represent the configuration of the figure with that of its stance foot, called a foothold, rather than its body posture. The roadmap can be modeled as a directed graph whose nodes represent valid samples of the configuration space, that is, the position and orientation of the stance foot. A pair of nodes are connected by an edge if the biped figure can move from one node to the other with a prescribed motion while preserving its lifelikeness.

Roadmap search. Once a roadmap is constructed, the path planning problem is reduced to a constrained minimum-cost path problem on a directed graph, that is, finding a minimum-cost path such that each pair of consecutive edges in the path share a node in a posture transition graph [Badler et al. 1994]. A node of the posture transition graph represents a posture, and two nodes are connected by a directed edge representing a motion clip. After augmenting the roadmap with the posture transition graph for ensuring the connectivity between motion clips, we adopt a minimum-cost path algorithm [Dijkstra 1959] to search for two primary pieces of information: a sequence of input motion clips and that of target footprints.

Motion generation. Our last task is to generate realistic locomotion from the sequence of input motion clips and that of target footprints obtained from the roadmap. In roadmap search, we have acquired an input motion sequence by simply stitching the motion clips attached to the edges along the minimum-cost path. Therefore, the footprints of the input motion sequence may yield some deviations from the target footprints transformed to coincide with the footholds at the nodes on the path. We address this problem in two steps: First, a target motion is estimated to provide a better initial guess. Then, the hierarchical displacement

mapping [Lee and Shin 1999] is employed with this initial guess to retarget the input motion for the target footprints.

The remainder of the paper is organized as follows. After presenting our probabilistic scheme to construct a roadmap in Section 2, we describe how we can search a sequence of input motion clips and that of target footprints from the roadmap in Section 3. In Section 4, we present how to generate a desired motion from the results of roadmap search. Section 5 demonstrates experimental results of our planning scheme. In Section 6, we discuss several issues on our scheme. Finally, we conclude this paper and describe future work in Section 7.

2. ROADMAP CONSTRUCTION

2.1 Node Generation and Connection

A foothold $\mathbf{f} = (\mathbf{p}, \mathbf{q})$, represents the configuration of a stance foot, where $\mathbf{p} \in \mathbb{R}^3$ and $\mathbf{q} \in \mathbb{S}^3$ denote its position and orientation, respectively. However, we will parameterize the foothold configuration space with the 2D position (u, v) of the stance foot and its resting yaw θ on the ground to reduce the dimensionality of the space. Provided with (u, v, θ), we can always recover the full configuration $\mathbf{f} = (\mathbf{p}, \mathbf{q})$ by extracting the height and the resting pitch and roll from environment geometry. Using parameters, u, v, and θ, we randomly sample footholds to generate the nodes of a roadmap under the assumption that each parameter value has a uniform distribution over an interval. After sampling a foothold, we test whether it is valid, that is, a figure can safely put a foot on the ground with this configuration. For example, the figure cannot place its foot on water in our first and second experiments. In the third experiment, the foot cannot collide with obstacles. The fourth experiment enforces both constraints. A prescribed number of valid configurations are sampled in this way and retained in the roadmap. In our experiments, a few thousand samples have given high fidelity in planning a path with the roadmap.

Each newly generated node is added to the roadmap using a fast local planner, which will be described in Section 2.2. As the number of nodes in the roadmap increases, the time to connect a new node with the others grows rapidly. Observing that a node can only be connected with nearby nodes, we choose the K-closest neighbors to the new node as the candidates for connection for a given positive integer K. A successful connection of the new node to or from one of the K-closest nodes yields a directed edge between them. From a result in computational geometry [Preparata and Shamos 1985], finding the K-closest neighbors requires a non-trivial amount of computation for a large number of points. In our current implementation, we exploit spatial partitioning techniques to speed up the process of finding the K-closest nodes. Specifically, we keep the randomly-generated nodes in a spatial data structure such as uniform cells. For each cell, we choose one of its nodes as their representative. Given a new node, we first sort the cells in the increasing order of distances from their representatives to the new node and then choose K nodes, while visiting the cells in the same order to approximate the K-closest neighbors.

2.2 Local Planner

To connect a pair of given nodes in the roadmap, the local planner checks whether they can be connected with a motion clip, that is, whether or not the motion clip (or its portion) can be transformed, within a specified tolerance, to have its first and last footprints coincide with the footholds at the two nodes, respectively (See Figure 1). Successful transformation gives a sequence of footprints. The connection between the two nodes is discarded, if any footprint of this sequence is invalid, that is, not safely placed on the ground. When every footprint is valid, the nodes are connected with the edge unless the transformed motion causes any collision with obstacles. Collision detection can be performed with efficient methods in [Gottschalk et al. 1996; Lin and Manocha 1995; Mirtich and Canny 1995].

From now, we focus on how to transform the footprints. Given a motion clip M, the footprint sequence $\mathbf{f}(M)$ is obtained by interactively marking the moments of heel-strike and toe-off. A stance foot is held on the ground from its heel-strike time to its toe-off time to give a footprint. Suppose that M consists of n frames and has m footprints. Letting the j-th stance foot be on the ground for a time interval, $[t_j - \Delta_j, t_j + \Delta_j]$, we specify $\mathbf{f}(M)$ as follows:

$$\mathbf{f}(M) = \{\mathbf{f}_j | \mathbf{f}_j = (\mathbf{p}_j, \mathbf{q}_j, t_j, \Delta_j), 1 \leq j \leq m\}. \tag{1}$$

Here, $\mathbf{f}_j$ represents the j-th footprint, $\mathbf{p}_j \in \mathbb{R}^3$ and $\mathbf{q}_j \in \mathbb{S}^3$ denote its position and orientation, respectively, t_j is the middle of its heel-strike and toe-off times, and $2\Delta_j$ is its duration of stance. Moreover, $t_j \leq t_{j+1}$ and $1 \leq t_j \pm \Delta_j \leq n$ for all j.

Without loss of generality, we assume that the Euclidean distance $||\mathbf{p}_k - \mathbf{p}_1||$ from the first footprint $\mathbf{f}_1$ to the k-th footprint $\mathbf{f}_k$ increases monotonically in the direction from $\mathbf{f}_1$ to the last footprint $\mathbf{f}_m$ as the index k increases. For a motion with a non-monotonic footprint sequence, we can always split it into two or more short-sized motions to satisfy this assumption. Let $\mathbf{f}_{k,l}(M)$, $1 \leq k < l \leq m$, be the partial footprint sequence of $\mathbf{f}(M)$ from $\mathbf{f}_k$ to $\mathbf{f}_l$, and $M_{k,l}$ be its corresponding motion segment.

Let $\mathbf{f}_s = (\mathbf{p}_s, \mathbf{q}_s)$ and $\mathbf{f}_e = (\mathbf{p}_e, \mathbf{q}_e)$ be the foothold configurations of the given two nodes, respectively. Using a motion M, we are to connect them from $\mathbf{f}_s$ to $\mathbf{f}_e$. The local planner examines each footprint of $\mathbf{f}(M)$ successively to find a partial footprint sequence $\mathbf{f}_{k,l}(M)$ such that the difference between $||\mathbf{p}_k - \mathbf{p}_l||$ and $||\mathbf{p}_s - \mathbf{p}_e||$ is minimized, and that the start and last postures of its motion segment $M_{k,l}$ are similar to those of the motion M within a given tolerance [Lee et al. 2002], respectively. Suppose that the local planner has examined up to the footprint $\mathbf{f}_{k_1}$ and found $\mathbf{f}_{k_1,l_1}(M)$ that minimizes $\big|||\mathbf{p}_{k_1} - \mathbf{p}_{l_1}|| - ||\mathbf{p}_s - \mathbf{p}_e|||\big|$ among the partial footprint sequences starting from $\mathbf{f}_{k_1}$, whose first and last footprints satisfy the latter condition. Then, for the next footprint $\mathbf{f}_{k_2}$ at which the posture is similar to the start posture of the motion M, we only need to examine $\mathbf{f}(M)$, starting from the last footprint previously examined for determining $\mathbf{f}_{l_1}$, to find the footprint $\mathbf{f}_{l_2}$ such that $\big|||\mathbf{p}_{k_2} - \mathbf{p}_{l_2}|| - ||\mathbf{p}_s - \mathbf{p}_e|||\big|$ is minimized and that the posture at $\mathbf{f}_{l_2}$ is also similar to the last posture of the motion M. Thus, by examining the partial footprint sequences successively from the first footprint, we can choose $\mathbf{f}_{k,l}(M)$ that will be used to connect the footholds $\mathbf{f}_s$ and $\mathbf{f}_e$ in $O(m)$ time. Hereafter, for convenience of explanation, we will regard the partial footprint sequence $\mathbf{f}_{k,l}(M)$

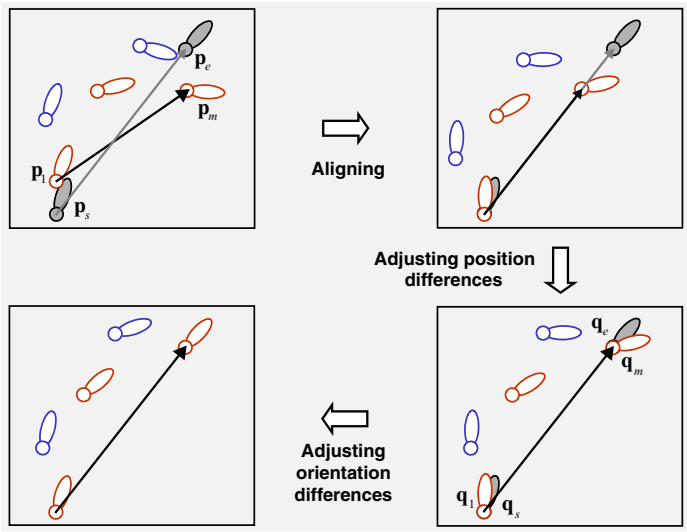


Fig. 1. Footprint transformation

thus chosen as the full footprint sequence $\mathbf{f}(M)$.

To transform the footprint sequence $\mathbf{f}(M)$, we first adjust the positions of the footprints and then their orientations (See Figure 1). In order to adjust the positions, we apply a single affine transformation to the footprint positions. Initially, we translate $\mathbf{p}_j$ for all j to make the position $\mathbf{p}_1$ of the first footprint coincide with $\mathbf{p}_s$. Then, to place the position $\mathbf{p}_m$ of the last footprint as close to $\mathbf{p}_e$ as possible, we rotate $\mathbf{p}_j$ for all j, at first, about the axis perpendicular to the ground plane, and then, about the axis lying on the same plane and perpendicular to the direction vector $\mathbf{p}_e - \mathbf{p}_s$. Here, both axes pass through $\mathbf{p}_1$. For later orientation adjustment, we update the footprint orientation $\mathbf{q}_j$, $1 \leq j \leq m$, with those rotations. In general, $\mathbf{p}_m$ does not lie exactly on $\mathbf{p}_e$ with those rotations. However, they are lying on the line connecting $\mathbf{p}_s$ and $\mathbf{p}_e$. Scaling $\mathbf{p}_j$ for all j by the factor

$$s = \frac{\|\mathbf{p}_s - \mathbf{p}_e\|}{\|\mathbf{p}_m - \mathbf{p}_1\|}, \tag{2}$$

we can make $\mathbf{p}_m$ lie on $\mathbf{p}_e$.

Now, we adjust orientation differences. Employing the slerp (spherical linear interpolation) introduced by Shoemake [1985], we propagate the orientation differences at the two nodes, $\log(\mathbf{q}_1^{-1}\mathbf{q}_s)$ and $\log(\mathbf{q}_m^{-1}\mathbf{q}_e)$, to the intermediate footprints between the first and the last. To obtain a new orientation $\mathbf{q}'_j$ of the j-th footprint, we linearly interpolate the differences in the orientation space with the chord length parameterization of $\mathbf{p}_j$, $1 \leq j \leq m$, and then apply the rotation to $\mathbf{q}_j$. That is,

$$\mathbf{q}'_j = \mathbf{q}_j \mathrm{slerp}(t_j, \mathbf{q}_1^{-1}\mathbf{q}_s, \mathbf{q}_m^{-1}\mathbf{q}_e) \tag{3}$$

where $t_j = \frac{\sum_{k=2}^{j} \|\mathbf{p}_k - \mathbf{p}_{k-1}\|}{\sum_{k=2}^{m} \|\mathbf{p}_k - \mathbf{p}_{k-1}\|}$ for all j.

To preserve the quality of the motion clip M, we need to check whether the

last two non-rigid transformations can be done within given thresholds. For the position difference, we use the ratio $|1-s|$ to measure how much the length of the chord from $\mathbf{p}_1$ to $\mathbf{p}_m$ is stretched or contracted, where s is given by Equation (2). To measure the orientation difference, we employ the tight upper bound on the angular difference of rotation in the interpolation given in Equation (3), that is, $\max(\|\log(\mathbf{q}_1^{-1}\mathbf{q}_s)\|, \|\log(\mathbf{q}_m^{-1}\mathbf{q}_e)\|)$. If both the position and orientation differences are within their threshold values, we achieve a successful transformation. In our experiments, the threshold values for the position and orientation differences were chosen empirically as 0.25 and 0.25π, respectively.

2.3 Cost Function

An edge of a directed graph is specified by an ordered pair of nodes called tail and head nodes, respectively. Each edge of the roadmap has its cost that measures the effort for a character to move from the tail node to the head node with the tagged motion clip. Searching the roadmap for a path is guided by the edge costs. With the well-designed edge costs, an intended sequence of motions can be obtained by finding the minimum-cost path from the start node to the goal in the roadmap. Our cost function of each edge incorporates a set of factors that measure the motion clip in diverse perspectives.

One of the most intuitive and important factors is the distance for a character to travel. For obtaining the distance c_d^p to travel with a motion clip, we sum up the distances between every pair of adjacent footprints, that is, $c_d^p = \sum_{j=2}^{m} \|\mathbf{p}_j - \mathbf{p}_{j-1}\|$, where $\mathbf{p}_j$, $1 \le j \le m$, is the position of the j-th footprint. A somewhat similar and slightly different factor c_d^t is the number of frames of a motion clip. c_d^t is a measure in the time domain whereas c_d^p in the space domain. Thus, we incorporate both terms in a single function,

$$c_d = c_d^p + w_b \cdot c_d^t, \tag{4}$$

with a user-provided weighting coefficient w_b.

In Section 2.2, we have employed the local planner to adapt a live-captured motion to the footholds at a pair of nodes of each edge. The adaptation is required to adjust both the position and orientation differences between the footholds and the pair of extreme footprints of the motion clip. These differences indicate how much the motion clip is degraded to satisfy the foothold constraints. In order to preserve the lifelike nature of the captured motion clips, we need to minimize their adaptation while satisfying the constraints. Let c_r^p and c_r^o be the position and orientation differences, respectively. To measure the degree of adaptation, we use a weighted sum of the both differences,

$$c_r = c_r^p + w_m \cdot c_r^o, \tag{5}$$

where w_m is a weighting factor addressing the metric difference between the position and the orientation.

Besides distance and adaptation costs, we may also consider user's preference to certain motion clips. For example, "walking" motions are more desirable than "broad jumping" in a normal case. To incorporate such preference, we consider a preference cost c_p that is reciprocal to user's preference. Then, the final cost function for an edge is defined as a weighted sum of the former two costs multiplied

by the last cost:

$$c = c_p \cdot (c_d + w_r \cdot c_r), \tag{6}$$

where w_r is a user-controllable weighting factor. In addition to the above terms, other factors such as obstacles along the edge may also be incorporated into the cost function. In our experiments, the weighting factors w_b, w_m and w_r were chosen empirically as $0.5, 1.0$ and 10.0, respectively.

2.4 Enhancement

If a sufficiently large number of nodes were generated, then the roadmap would uniformly cover almost entire regions of the c-space. However, with a moderate number of nodes, uniform sampling does not guarantee that the roadmap is connected well in "difficult" regions of the c-space such as narrow passages [Kavraki and Latombe 1994; Kavraki et al. 1996]. We adopt a heuristic method [Kavraki and Latombe 1994] to facilitate better interconnection. For each node v belonging to the node set V of the roadmap, a probability density function is defined by

$$P(v|V) = \frac{1}{i_v + o_v + 1} / \sum_{u \in V} \frac{1}{i_u + o_u + 1}, \tag{7}$$

where i_v is the number of edges incident to v, and o_v is the number of edges incident from v. The underlying idea of this function is based on the fact that a poorly-interconnected node lies in a difficult region with a high probability and thus needs more samples for better connectivity. We choose a node v from the node set V of the roadmap with probability $P(v|V)$ to introduce an additional node near the node v. The number of additional nodes between one third and one half of that of initial nodes is known empirically to give a good performance [Kavraki et al. 1996; Kavraki and Latombe 1994; Kavraki et al. 1996].

3. ROADMAP SEARCH

In this section, we describe how to search the roadmap for a minimum-cost path, that gives a sequence of desired motions and that of target footprints, provided with start and goal configurations. We will further refine the target footprints using the local planner presented in Section 2.2.

Before moving on to roadmap search, we start with describing a posture transition graph [Badler et al. 1994] since it plays a crucial role in ensuring connectivity between motion clips. A node of the posture transition graph represents a valid posture of a biped figure, and a directed edge represents a motion clip. If the motion clip of an edge can be connected to that of another edge, then the head node of the former edge coincides with the tail of the latter (See Figure 2(a)). In addition, each edge is tagged with a footprint sequence obtained from the corresponding motion clip. We employ techniques [Lee and Shin 1999; Rose et al. 1996] for generating the transitions among motion clips to build the posture transition graph.

3.1 Path Finding

Given start and goal footholds, $\mathbf{f}_x$ and $\mathbf{f}_y$, our objective here is to find a sequence of input motion clips and the target footprints, which can be done in two steps: We first add two nodes x and y corresponding to $\mathbf{f}_x$ and $\mathbf{f}_y$ to the roadmap and then

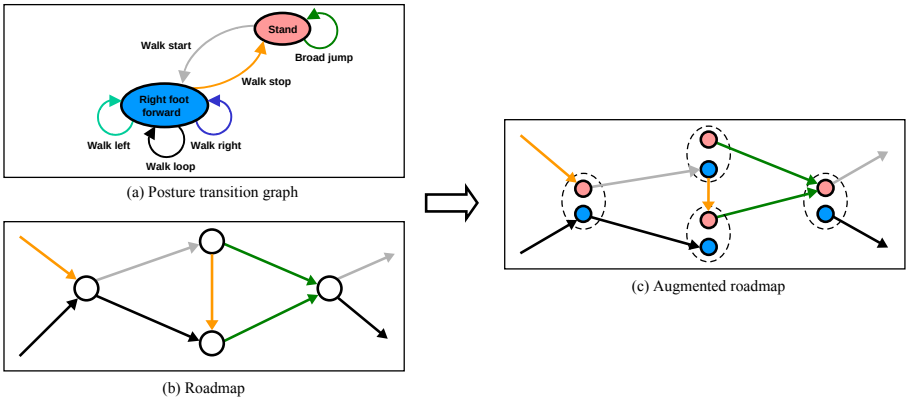


Fig. 2. Augmented roadmap with posture transition graph

search the roadmap for a minimum-cost path from x to y. When any of those two nodes cannot be connected to the roadmap with our local planner, we employ a random walk to connect it to the roadmap [Barraquand and Latombe 1991].

Searching the roadmap needs special care to select motion clips that can be spliced. Suppose that we have traversed the roadmap to arrive at the node v via an edge e incident to v. To go further from v, we take another edge e' that is incident from the node v and whose corresponding motion clip can be stitched with that of the edge e. At each node v encountered, we need to memorize the one-level history, that is, the incident edge to the node v used for entering. We also need to refer to the posture transition graph to check if the motion tagged on e can make transition to that on e' via a common posture. Therefore, we cannot directly apply a minimum-cost path algorithm [Dijkstra 1959] to the roadmap.

Let $G(V, E)$ be the directed graph representing our roadmap. To avoid both memorizing the history and referring to the posture transition graph during path search, we transform $G(V, E)$ into a new directed graph $G'(V', E')$ so that any connected path of G' yields a feasible motion sequence. Suppose that the posture transition graph has k nodes, $u_i, 1 \leq i \leq k$, each representing a posture. Then, using the nodes of the posture transition graph, we split every node v of $G(V, E)$ to make a set of k nodes, $\{(v, u_i)|i = 1, 2, \cdots, k\}$ of G'. Each node (v, u_i) possesses both the posture at the node u_i of the posture transition graph and the foothold at the node v of the roadmap G (See Figure 2). A pair of nodes, (v_1, u_i) and (v_2, u_j) of G' admit a directed edge from (v_1, u_i) to (v_2, u_j) if and only if the motion tagged on the edge from v_1 to v_2 in G is also tagged on the edge from u_i to u_j in the posture transition graph. Since each node of G' is associated with a posture as well as a foothold, any pair of edges in G' that are incident to and from a node guarantee that the motion clip tagged on the edge incident to the node can make transition to that on the other, and thus any connected path in G' provides a feasible motion sequence. Consequently, we can directly apply the minimum-cost path algorithm to the graph G'. Since G' has $k|V|$ nodes and $|E|$ edges, we can find a sequence of motion clips and their target footprints in $\mathrm{O}(k|V| \log k|V| + |E|)$ time.

If path planning fails frequently, we sample the configuration space more densely

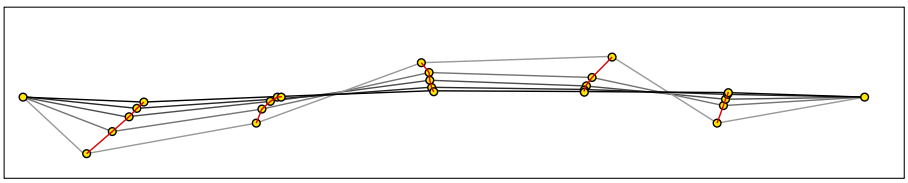


Fig. 3. A sequence of footholds is iteratively straightened up.

to add new nodes. The roadmap can be expanded incrementally on the fly with our local planner. When a small portion of the environment changes, new nodes can be sampled from the same portion of the new environment while removing those nodes lying on that portion of the old environment. Another kind of failure can occur due to lack of right motion clips for connection. We empirically handled this problem in our experiments by choosing a sufficiently large number as a threshold to discriminate the latter failure from the former.

3.2 Footprint Refining

Consider a target footprint sequence that has been obtained from the motion clips tagged on the edges of the path in the roadmap. Since we have sampled the footholds of the roadmap randomly, the target footprints may reveal some artifacts: The character may walk in zigzag along the path. In addition, as explained in Section 2.2, the footprint sequence along each edge may differ from that of the original motion clip within a specified tolerance. This difference may vary from edge to edge. To reduce such artifacts due to random sampling, we first modify the target footholds slightly along the path by smoothing them, and then adjust the target footprints locally to make them more similar to the original ones.

Path straightening. To straighten up a zigzag-shaped moving path, we consider the target footholds as data points on a curve in $\mathbb{R}^3 \times \mathbb{S}^3$ and apply a smoothing filter developed for motion data. While smoothing the target footholds, we change the motion M previously attached on an edge using the same type of motions as M. Here, the same type of motions are assumed to have the same start and last postures so that the connectivity of the motions at each foothold can be guaranteed. For example, "30-degree right turn" may be replaced by "15-degree right turn" or "straight walk" as the path is becoming straightened. Iteratively smoothing the target footholds and changing the motions attached on the edges, we gradually straighten up the zigzag-shaped moving path (See Figure 3).

Let $\mathbf{f}_j = (\mathbf{p}_j, \mathbf{q}_j) \in \mathbb{R}^3 \times \mathbb{S}^3, 1 \leq j \leq m$, be the configurations of the target footholds. To smooth both position and orientation in a coherent way, we employ the general filtering scheme for motion data [Lee and Shin 2001; 2002]. In particular, we adopt a binomial filter mask $(\frac{1}{16}, \frac{4}{16}, \frac{6}{16}, \frac{4}{16}, \frac{1}{16})$. For position smoothing, the filtering scheme yields a function $\mathcal{F}(\cdot)$ that takes the weighted average of the adjacent positions along the path with the filter mask:

$$\mathbf{p}'_j = \mathcal{F}(\mathbf{p}_j) = \frac{1}{16}(\mathbf{p}_{j-2} + 4\mathbf{p}_{j-1} + 6\mathbf{p}_j + 4\mathbf{p}_{j+1} + \mathbf{p}_{j+2}).$$

For orientation smoothing, the filtering scheme transforms the orientation data into

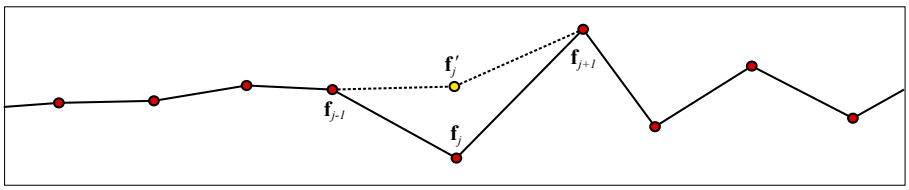


Fig. 4. Conceptual view of path straightening

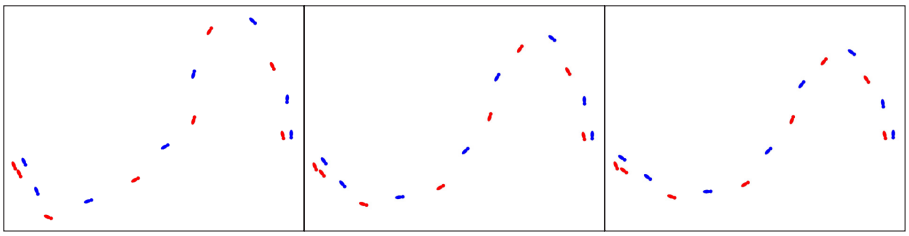


Fig. 5. A sequence of footprints is repeatedly refined. The number of iterations is 0 (left), 2 (middle), and 8 (right).

their analogies in a vector space, applies a filter mask on them, and then transforms the displacement back to the orientation space:

$$\begin{aligned}\mathbf{q}'_j &= \mathcal{H}(\mathbf{q}_j) = \mathbf{q}_j \exp(\mathcal{F}(\mathbf{v}_j) - \mathbf{v}_j) \\ &= \mathbf{q}_j \exp(\frac{1}{16}(-\omega_{j-2} - 5\omega_{j-1} + 5\omega_j + \omega_{j+1})),\end{aligned} \tag{8}$$

where $\mathbf{v}_j = \sum_{i=0}^{j-1} \log(\omega_i)$ and $\omega_j = \log((\mathbf{q}_j)^{-1}\mathbf{q}_{j+1}) \in \mathbb{R}^3$. For detailed derivation, see the results in [Lee and Shin 2002].

While smoothing each foothold configuration successively along the path, we also ensure the feasibility of the path with the new foothold. Suppose that we are now computing the j-th foothold configuration. We check if the foothold configuration $\mathbf{f}'_j$ is valid and if $\mathbf{f}'_j$ can be connected safely from $\mathbf{f}_{j-1}$ and to $\mathbf{f}_{j+1}$ using the local planner (See Figure 4). If both conditions are satisfied, we replace the foothold configurations $\mathbf{f}_j$ with $\mathbf{f}'_j$. Otherwise, we linearly interpolate $\mathbf{f}_j$ and $\mathbf{f}'_j$ with a given weighting factor. If the interpolated configuration does not satisfy the above two conditions, we lower the weighting factor to obtain a new configuration that is closer to $\mathbf{f}_j$. This process is repeated a few times to obtain a configuration satisfying the conditions. If we fail to find such a configuration after a given number of iterations, we take $\mathbf{f}_j$ itself as the filtered foothold $\mathbf{f}'_j$.

Difference smoothing. The second step of footprint refining examines the local window at each footprint in sequence, which covers a small portion of the footprint sequence, and evaluates how much it is different from the original footprint sequence of the corresponding portion of the input motion data. To propagate the difference smoothly along the path, we present a local refining scheme that adjusts each footprint successively by referring to the neighboring footprints of the original,

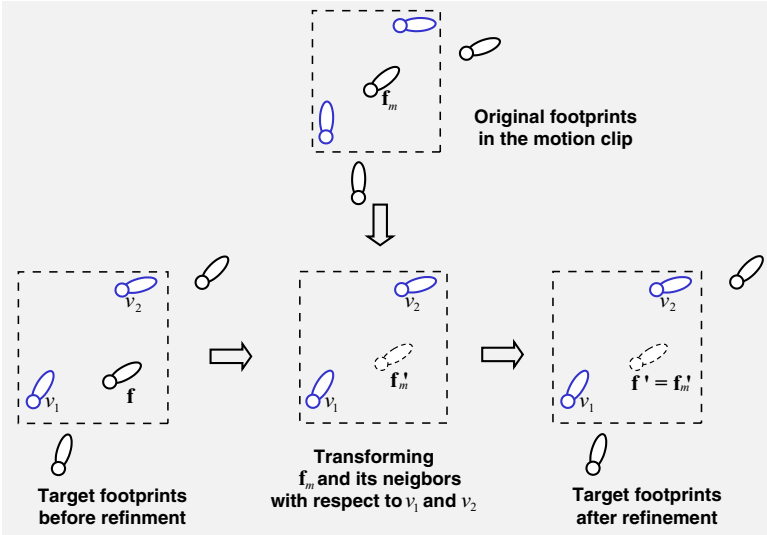


Fig. 6. The conceptual view of difference smoothing

while ensuring the validity of the adjusted footprints. Applying the local refining scheme to every footprint in sequence back and forth repeatedly along the path, we propagate the difference rather uniformly (See Figure 5).

We adopt the local planner described in Section 2.2 for smoothing the difference. For each footprint $\mathbf{f}$ in the target footprint sequence, we take the corresponding footprint $\mathbf{f}_m$ and its two adjacent footprints in the original motion clip (See Figure 6). To apply the local planner to $\mathbf{f}_m$ together with its two neighboring footprints, we conceptually interpret the two footprints adjacent to the footprint $\mathbf{f}$ as the footholds representing two hypothetical nodes, v_1 and v_2 of the roadmap, respectively. The planner adjusts the footprint sequence composed of the three footprints, that is, the footprint $\mathbf{f}_m$ and its two neighbors, with respect to the footholds in the hypothetical nodes v_1 and v_2 to obtain a new footprint $\mathbf{f}'$.

However, we have difficulty in refining a footprint $\mathbf{f}$ when $\mathbf{f}$ is shared by the two footprint sequences of the motion clips A and B. Then, $\mathbf{f}_m$ is not only the last footprint of the motion A but also the first footprint of the motion B. Thus, each of the motion clips has only one footprint adjacent to $\mathbf{f}_m$. To apply our local refining scheme to $\mathbf{f}$, we take $\mathbf{f}_m$ and the two neighboring footprints after transforming the original footprint sequence of the motion B such that its first footprint coincides with the last footprint of the motion A.

During the local refining, instead of simply replacing the original footprint $\mathbf{f}$ with the new one $\mathbf{f}'$, we linearly interpolate them with a given weighting factor. If the interpolated configuration is not valid, we lower the weighting factor to obtain a new candidate that is closer to the original footprint $\mathbf{f}$. This process is repeated a few times to obtain a valid one. If we fail to find a valid one after a given number of iterations, we take $\mathbf{f}$ itself as the footprint.

4. MOTION GENERATION

With the sequence of input motion clips and that of target footprints available, we finally generate a biped locomotion from the start position to the goal in this section. Interpreting each of those footprints as a variational constraint over a time interval [Gleicher 1998], we can formulate this task as a motion retargetting problem [Gleicher 1998; Lee and Shin 1999; Rose et al. 1996]. For this problem, it is well-known that the initial body trajectory is very important for the convergence of numerical optimization and the quality of the result. The body trajectory is usually represented by the position and orientation of the root segment. By analyzing the input motion sequence and the target footprints, we estimate the body trajectory of the target motion. Together with the joint angles of the motion clips, this yields an initial guess for the target motion at every frame. Finally, we apply the hierarchical displacement mapping [Lee and Shin 1999] with this initial guess to retarget the input motion sequence for the target footprints.

Since motion retargetting is well-described in [Gleicher 1998; Lee and Shin 1999], we will concentrate on how to estimate the body trajectory of the target motion for deriving the initial guess. For the input motion sequence A, let $\mathbf{f}(A)$ and $\mathbf{b}(A)$ denote its footprint sequence and body trajectory, respectively. For the unknown target motion sequence B, $\mathbf{f}(B)$ and $\mathbf{b}(B)$ can be defined similarly. Here, our objective is to estimate the target body trajectory $\mathbf{b}(B)$. B is implicitly specified by $\mathbf{f}(B)$ together with A. Suppose that A and $\mathbf{f}(A)$ consist of n frames and m footprints, respectively. Tracing the posture of the root segment in every motion clip in sequence, we can easily acquire $\mathbf{b}(A)$ for all frames of A.

In order to estimate $\mathbf{b}(B)$, we exploit a relationship between $\mathbf{b}(A)$ and $\mathbf{f}(A)$ (See Figure 7). We first derive a reference trajectory $\mathbf{r}(A)$ of the motion sequence A from its footprints $\mathbf{f}(A)$, and then compute its displacement $\mathbf{d}(A)$ to $\mathbf{b}(A)$,

$$\mathbf{d}(A) = \mathbf{b}(A) \ominus \mathbf{r}(A), \tag{9}$$

where $\mathbf{r}(A)$ and the operator $\ominus$ will be defined later. Notice that each of $\mathbf{b}(A)$, $\mathbf{d}(A)$, and $\mathbf{r}(A)$ consists of vector and orientation components. Assuming that A and B are similar within a small tolerance, we have

$$\begin{aligned} \mathbf{b}(B) &= \mathbf{d}(B) \oplus \mathbf{r}(B) \approx \mathbf{d}(A) \oplus \mathbf{r}(B) \\ &= (\mathbf{b}(A) \ominus \mathbf{r}(A)) \oplus \mathbf{r}(B). \end{aligned} \tag{10}$$

With $\mathbf{b}(A)$ directly picked up from the original motion clips, we need to compute reference trajectories $\mathbf{r}(A)$ and $\mathbf{r}(B)$ to determine $\mathbf{b}(B)$. Provided with $\mathbf{f}(A) = \{\mathbf{f}_j | \mathbf{f}_j = (\mathbf{p}_j, \mathbf{q}_j, t_j, \Delta_j), 1 \le j \le m\}$, the body center (the center of the root segment) is expected to lie above the "mid-posture" of every pair of consecutive footprints. Let the mid-posture of $\mathbf{f}_j$ and $\mathbf{f}_{j+1}$ be

$$(\bar{\mathbf{p}}_j, \bar{\mathbf{q}}_j) = (\frac{1}{2}(\mathbf{p}_j + \mathbf{p}_{j+1}), \mathbf{q}_j(\mathbf{q}_j^{-1}\mathbf{q}_{j+1})^{\frac{1}{2}}) \tag{11}$$

at $\bar{t}_j = \lfloor \frac{t_j + t_{j+1}}{2} \rfloor$, $1 \le j < m$. Interpolating those mid-postures piecewisely for each adjacent pair in sequence, we obtain a continuous reference trajectory for A over the interval $[\bar{t}_1, \bar{t}_{m-1}]$ [Shoemake 1985]. We resample the trajectory at every frame in $[\bar{t}_1, \bar{t}_{m-1}]$ to have $\mathbf{r}(A) = \{(\mathbf{p}_i^r, \mathbf{q}_i^r) \in \mathbb{R}^3 \times \mathbb{S}^3 | \bar{t}_1 \le i \le \bar{t}_{m-1}\}$. $\mathbf{r}(B)$ can also be obtained in the same way from the target footprint sequence $\mathbf{f}(B)$.

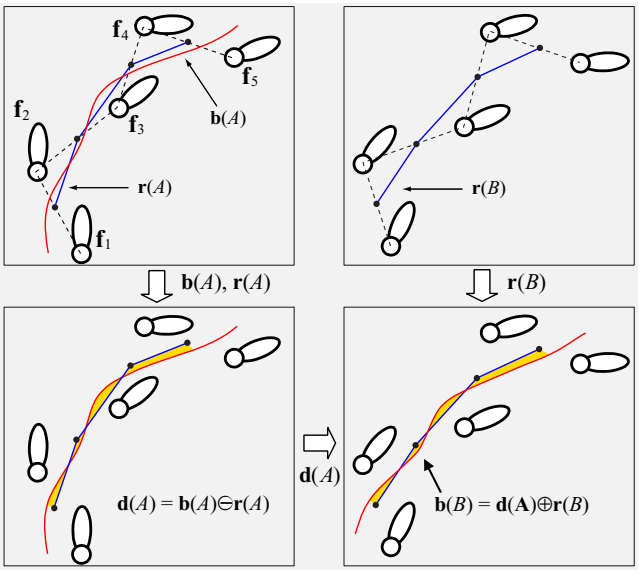


Fig. 7. Body trajectory estimation

We now compute the displacement map $\mathbf{d}(A) = \mathbf{b}(A) \ominus \mathbf{r}(A)$. An ordered pair $(\mathbf{p}_i^r, \mathbf{q}_i^r)$, $\bar{t}_1 \leq i \leq \bar{t}_{m-1}$, of position and orientation components of $\mathbf{r}(A)$ specifies a rigid transformation that maps a point $\mathbf{u}$ in $\mathbb{R}^3$ to a point $\mathbf{u}'$ in $\mathbb{R}^3$, that is, $\mathbf{u}' = \mathbf{q}_i^r \mathbf{u} (\mathbf{q}_i^r)^{-1} + \mathbf{p}_i^r$. Here, the vector $(x, y, z) = \mathbf{u} \in \mathbb{R}^3$ is considered as a purely imaginary quaternion $(0, x, y, z) \in \mathbb{S}^3$. Given $\mathbf{b}(A) = \{(\mathbf{p}_i^b, \mathbf{q}_i^b) \in \mathbb{R}^3 \times \mathbb{S}^3 | 1 \leq i \leq n\}$ and $\mathbf{r}(A) = \{(\mathbf{p}_i^r, \mathbf{q}_i^r) \in \mathbb{R}^3 \times \mathbb{S}^3 | \bar{t}_1 \leq i \leq \bar{t}_{m-1}\}$, we define their displacement map $\mathbf{d}(A) = \{(\mathbf{u}_i, \mathbf{v}_i) \in \mathbb{R}^3 \times \mathbb{R}^3 | \bar{t}_1 \leq i \leq \bar{t}_{m-1}\}$ measured in the local coordinate system for $\mathbf{r}(A)$ as follows:

$$\begin{aligned} \mathbf{d}(A) &= \mathbf{b}(A) \ominus \mathbf{r}(A) \\ &= \{(\mathbf{p}_i^b, \mathbf{q}_i^b) \ominus (\mathbf{p}_i^r, \mathbf{q}_i^r) | \bar{t}_1 \leq i \leq \bar{t}_{m-1}\} \\ &= \left\{ \left((\mathbf{q}_i^r)^{-1} (\mathbf{p}_i^b - \mathbf{p}_i^r) \mathbf{q}_i^r, \log\left((\mathbf{q}_i^r)^{-1} \mathbf{q}_i^b \right) \right) | \bar{t}_1 \leq i \leq \bar{t}_{m-1} \right\}. \end{aligned} \tag{12}$$

Finally, letting $\mathbf{r}(B) = \{(\mathbf{p}_i, \mathbf{q}_i) | \bar{t}_1 \leq i \leq \bar{t}_{m-1}\}$, we obtain the body trajectory $\mathbf{b}(B)$:

$$\begin{aligned} \mathbf{b}(B) &= \mathbf{d}(A) \oplus \mathbf{r}(B) \\ &= \{(\mathbf{u}_i,\ \mathbf{v}_i) \oplus (\mathbf{p}_i, \mathbf{q}_i) | \bar{t}_1 \leq i \leq \bar{t}_{m-1}\} \\ &= \left\{ \left(\mathbf{q}_i \mathbf{u}_i \mathbf{q}_i^{-1} + \mathbf{p}_i, \mathbf{q}_i \exp(\mathbf{v}_i) \right) | \bar{t}_1 \leq i \leq \bar{t}_{m-1} \right\}. \end{aligned} \tag{13}$$

$\mathbf{b}(B)$ is defined over frames from $\bar{t}_1$ to $\bar{t}_{m-1}$. To extend $\mathbf{b}(B)$ over all frames, we take the portion of $\mathbf{b}(A)$ between 1 and $\bar{t}_1$, and stitch it with $\mathbf{b}(B)$ such that its position and orientation coincide with those of $\mathbf{b}(B)$ at $\bar{t}_1$ through a rigid transformation. We can obtain $\mathbf{b}(B)$ from $\bar{t}_{m-1}$ to n symmetrically.

5. EXPERIMENTAL RESULTS

Our planning scheme is implemented in C++ as an Alias|Wavefront MAYA^{R} plug-in on top of the Microsoft Windows XP. Experiments are performed on an Intel Pentium^{R} PC (PIII 800 MHz processor and 512 MB memory) with commercially available motion clips. We use a human model of 43 DOFs: 6 DOFs for the pelvis position and orientation, 3 DOFs for the spine, 7 DOFs for each limb, and 3 DOFs for each of the neck and head.

Our first experiment is for planning a walking motion. Figure 8 exhibits the resulting motion in an island with three puddles, in which foot planting is not allowed at any point on water. We have used the posture transition graph with a set of live-captured motion clips for walking, which is similar to Figure 2(a). Specifically, we have prepared 13 motion clips, such as "walk start", "walk stop", "walk straight", "turn left", "turn right", and several curved walking motions. The terrain is represented as a NURBS surface of which control points are placed on a regular grid, and their y-coordinates (heights) are perturbed above or below the sea level. By setting the sea level to zero, a valid footprint has a non-negative height. For collision avoidance, we use a heuristic method that detects a collision when the height of the swing foot lies below the environment at any frame. The flow of our planning scheme is visualized in the movie clips available at `http://cg.kaist.ac.kr/~min/motion_planning`.

The second experiment exhibits the capability of our planning scheme to cope with a difficult environment by using various motions. As shown in Figure 9, the terrain has complex features such as a crevice and a small stream. Motions such as "broad jumping" and "running" are added to the motion repertoire. The running motion is given a preference over the others when a local geometry has a small height variation. We can observe that such motions are properly used for overcoming the environment.

The third experiment is for an environment with obstacles. As illustrated in Figure 10, the environment is a room with several pieces of furniture as obstacles. For collision detection, we employ a similar method in [Gottschalk et al. 1996]. A pair of nodes are connected with an edge representing a motion clip when the conservative bounding volume swept by the motion clip does not intersect with the rectangular bounding box of any obstacle.

The final experiment is to demonstrate the effectiveness of our planning scheme in a highly-constrained environment. As illustrated in Figure 11, we process a sequence of queries in a dungeon consisting of three distinct regions: The first region is a room with several pieces of old furniture, some of which are placed near to each others to form narrow passages. The second region is a moat with irregularly-distributed marble columns. The third region is a burial chamber. To navigate a character through this dungeon, we have prepared a total of 58 motions such as forward walking, lateral walking, backward walking, jumping, turning and hurdling. The responses of the first and third queries show that lateral walking and turning are properly used to pass through narrow passages. For the second query, low preference to jumping results in a motion sequence including jumping and curved walking rather than jumping only. For the fourth query, backward walking is selected at the end of the narrow passage because there is not enough space to turn around.

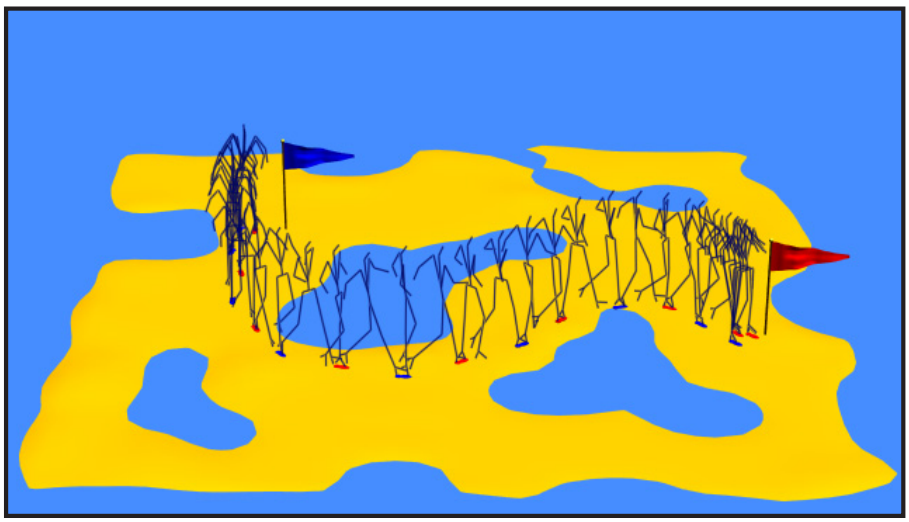


Fig. 8. Simple terrain

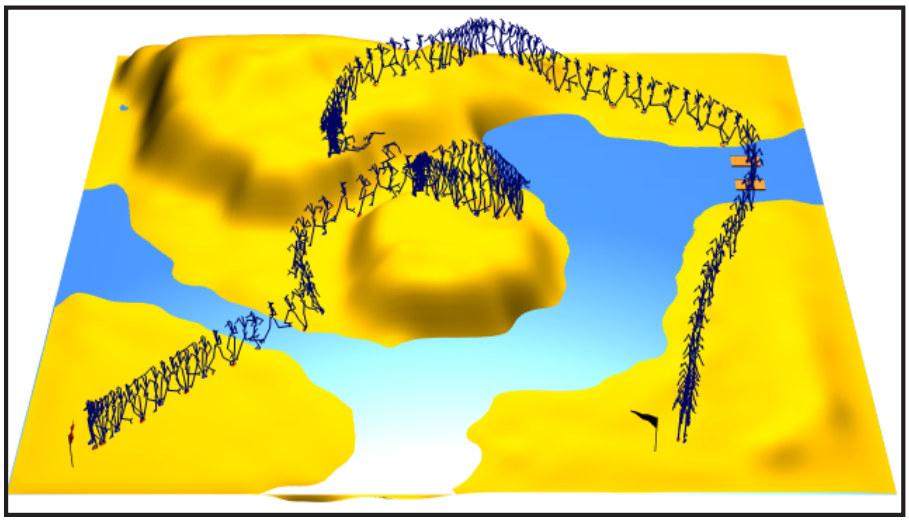


Fig. 9. Complex terrain

The fifth query gives rise to hurdling an obstacle to reach the exit of the room. The final query is to find a sequence of motions over the marble columns, which are not connected with each others and irregularly distributed. Unlike traditional path planning schemes for human navigation [Bandi and Thalmann 1998; Kuffner and Latombe 1999; Noser et al. 1995; Reich et al. 1994], our planning scheme seeks a sequence of discrete footprints rather than a continuous moving path. Thus, such a sequence can be obtained successfully.

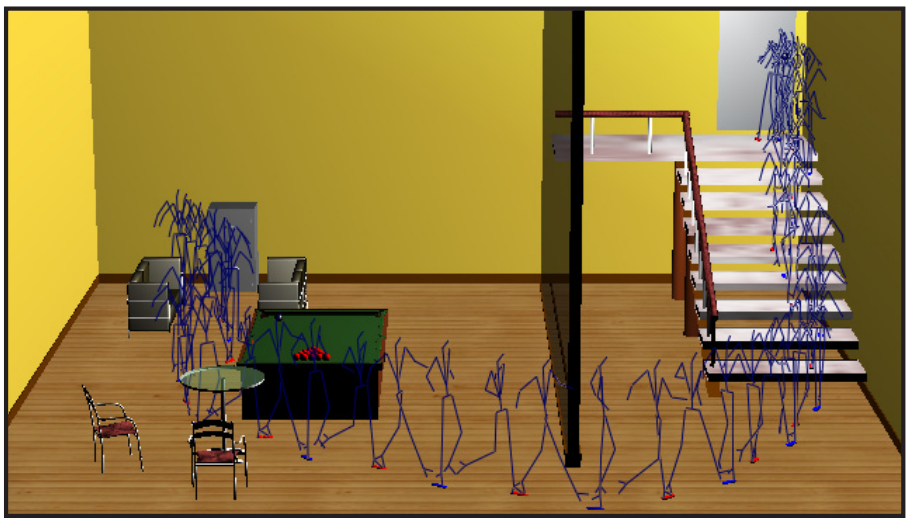
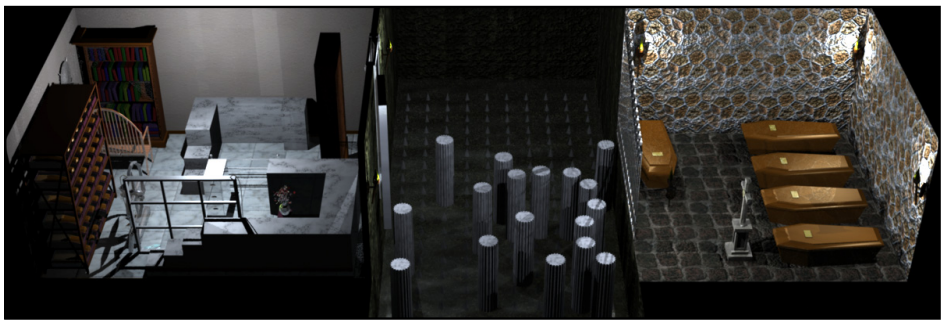


Fig. 10. Room with furniture

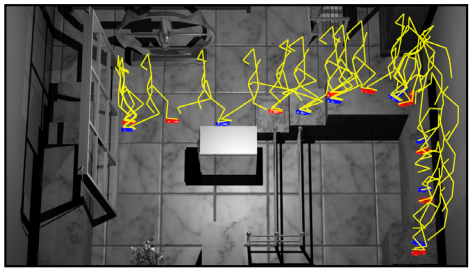
Table I summarizes the input data and the results of our experiments. For each example, our path planner performs three major steps: roadmap construction, roadmap search, and motion generation. The running time of the roadmap search shows its dependency on the number of nodes and that of edges in the roadmap. In each of the examples, the roadmap construction time dominates the others as expected. The construction time is a function of motion clips and environment geometry. The motion generation time roughly depends on the number of frames generated. After constructing the roadmap, we can produce more than 1000 frames per second in all experiments. Since the roadmap construction can be considered as preprocessing, our motion planning scheme exhibits interactive performance in our experiments.

6. DISCUSSION

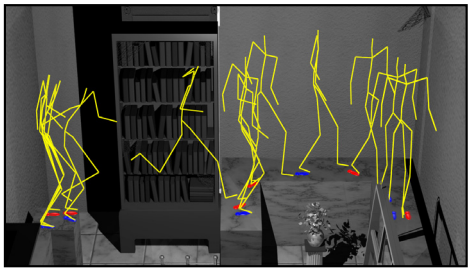
Potential field planner vs. Roadmap planner. There have been two major streams of randomized techniques for path planning: randomized path planning with potential fields [Barraquand and Latombe 1991; Kalisiak and van de Panne 2000; Koga et al. 1994] and probabilistic path planning with roadmaps [Barraquand et al. 1997; Kavraki et al. 1996; Kavraki and Latombe 1994; Kavraki et al. 1995; Kavraki et al. 1996; Overmars and Sŭestka 1994]. A randomized path planning scheme employs a potential field to guide the search for a path to the goal while avoiding the obstacles. To escape from a local minimum in the potential field, this scheme is usually equipped with random walks. A probabilistic path planning scheme constructs a roadmap by random sampling to guide the path search. In the probabilistic scheme, most heavy computations are done in the preprocessing phase, that is, roadmap construction. Once a roadmap is constructed, a path between any pair of configurations can be found very efficiently. Therefore, the roadmap method is particularly effective for handling repetitive point-to-point queries in a tightly-constrained static



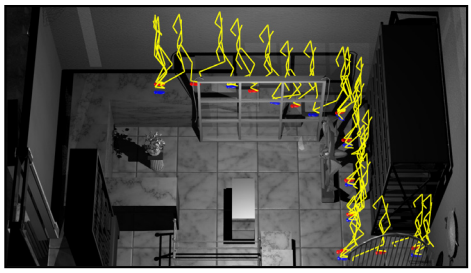
(a) The environment



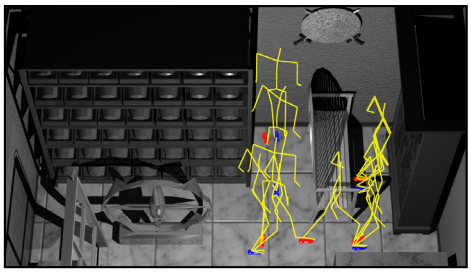
(b) The 1st query



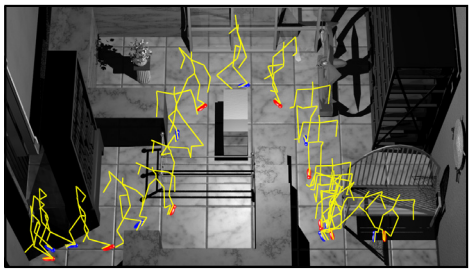
(c) The 2nd query



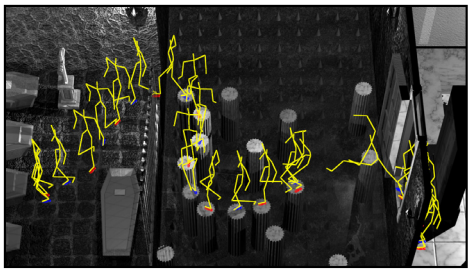
(d) The 3rd query



(e) The 4th query



(f) The 5th query



(g) The 6th query

Fig. 11. A sequence of motion planning queries in a dungeon

Table I. Performance data. N and M are the numbers of initial nodes sampled and the additional nodes for enhancement, respectively. E is the number of edges connected. Timing data give CPU time in seconds.

		Exp. 1	Exp. 2	Exp. 3
# of motion clips		13	29	13
roadmap construction	N (# of initial nodes)	2000	3000	2000
	M (# of additional nodes)	1000	1500	1000
	K (# of candidates)	100	100	100
	E (# of edges)	88703	266419	87617
	# of edges/node	28.81	59.20	29.21
	construction time	4.250	99.050	38.880
roadmap search	path finding time	0.020	0.250	0.060
	# of motion clips on a path	5	28	14
	# of iterations in refining	4	4	4
	footprint refining time	0.020	0.120	0.080
motion generation	# of frames	197	1139	467
	initial estimation time	0.010	0.020	0.010
	retargeting time	0.090	0.670	0.250
	retargeting time/frame	0.001	0.001	0.001
total time (excluding preprocessing)		0.140	1.060	0.400
average time/frame (excluding preprocessing)		0.001	0.001	0.001

		Exp. 4					
# of motion clips		58					
roadmap construction	N (# of initial nodes)	3000					
	M (# of additional nodes)	1500					
	K (# of candidates)	100					
	E (# of edges)	122975					
	# of edges/node	27.33					
	construction time	104.610					
query		1st	2nd	3rd	4th	5th	6th
roadmap search	path finding time	0.030	0.030	0.030	0.030	0.030	0.030
	# of motion clips on a path	7	5	9	4	8	11
	# of iterations in refining	4	4	4	4	4	4
	footprint refining time	0.020	0.020	0.030	0.010	0.030	0.060
motion generation	# of frames	286	181	304	131	293	365
	initial estimation time	0.010	0.010	0.010	0.010	0.010	0.010
	retargeting time	0.110	0.060	0.110	0.050	0.110	0.130
	retargeting time/frame	0.001	0.001	0.001	0.001	0.001	0.001
total time (excluding preprocessing)		0.170	0.120	0.180	0.100	0.180	0.230
average time/frame (excluding preprocessing)		0.001	0.001	0.001	0.001	0.001	0.001

environments as shown in Figure 11.

Postures vs. Footholds. A reasonable human model in computer graphics has about 40 degrees of freedom. Planning motions with such high degrees of freedom directly in the configuration space is still computationally demanding even with probabilistic motion planning techniques. Instead of sampling the configuration space of postures, we sample that of footholds while accessing motion clips via edges incident to and from nodes representing footholds. Although this makes the roadmap search a little complicated, the dimensionality of the configuration space is reduced greatly, and thus a moderate number of samples reflects well the connectivity of the configuration space. This enables us to search a minimum-cost path at interactive performance. Alternatively, one may prefer the pelvis configuration for the same purpose. However, unlike feet and hands, the pelvis is rarely used to interact with external environments. Thus, the footholds make it easier to plan motions such as jumping over crevice and walking over stepping stones.

Regular sampling vs. Random sampling. For a configuration space of low dimension such as our foothold space, regular sampling is a possible choice to construct a roadmap. In regular sampling, all grid points in the free configuration space are sampled as the nodes to guarantee their uniform coverage of the space. However, for an environment with dense obstacles, a grid of high resolution is required to ensure a good connectivity of the roadmap. In this case, random sampling yields a well-connected roadmap with a moderate number of samples due to the heuristic scheme for additional sampling [Barraquand et al. 1997; Kavraki et al. 1996; Kavraki et al. 1995] as given in Section 2.4. Moreover, compared to regular sampling, random sampling requires much less samples to achieve high fidelity in the sense that the roadmap covers uniformly almost entire regions of the configuration space [Barraquand et al. 1997; Kavraki et al. 1996; Kavraki et al. 1995].

7. CONCLUSIONS

Animation scripts or interactive systems require high-level directives for locomotion of a character on a virtual environment. To facilitate rapid motion prototyping and task-level motion generation for interactive applications, this paper presents a new scheme for planning a natural-looking motion for a human-like biped figure to move from a given start position to a goal with a set of prescribed motion clips. Combining probabilistic path planning [Kavraki et al. 1996] and hierarchical displacement mapping [Lee and Shin 1999], we find a sequence of motion clips and that of target footprints from the start to the goal, and then retarget the motion sequence to follow the target footprint sequence. Given a rich set of motion clips, our scheme enables a human-like figure to move over uneven terrain with a variety of motions, such as running on flat terrain, walking over stepping stones, and jumping over a crevice.

Our motion planner produces locomotion guided by the footholds at the nodes of the roadmap. We may use other features such as "handholds" [Kalisiak and van de Panne 2000] as well to produce different kinds of motions, for example, crossing a river with a rope and climbing a rock, sitting on a chair, and so on. Currently, our scheme is for motion planning of a single character in a static environment. We hope to extend our work for dynamic obstacles and multiple characters in future. Our scheme would handle dynamic obstacles by repeating motion planning at a regular interval with the static obstacles conservatively bounding them. To cope with multiple characters, we may employ a theoretical result [Svěstka and Overmars 1998] that combines roadmaps for multiple robots into the roadmap for the single composite robot consisting of all the robots. Finally, we are interested in a scalability issue. With a large set of motion clips, there could be a large number of edges in the roadmap, so that path finding would also be time-consuming. To accelerate path finding in such a case, we may reduce the number of multi-edges connecting the same pair of nodes in the roadmap by clustering them according to their corresponding motion types and filtering them during roadmap search to finally choose a motion.

ACKNOWLEDGMENTS

The authors wish to thank the referees for their useful comments and suggestions.

REFERENCES

Arikan, O. and Forsyth, D. 2002. Interactive motion generation from examples. *ACM Transactions on Graphics (Proc. SIGGRAPH 2002) 21,* 3, 483–490.

Badler, N. I., Bindiganavale, R., Granieri, J. P., Wei, S., and Zhao, X. 1994. Posture interpolation with collision avoidance. In *Proc. Computer Animation '94*. 13–20.

Bandi, S. and Thalmann, D. 1998. Space discretization for efficient human navigation. *Computer Graphics Forum 17,* 3, 195–206.

Barraquand, J., Kavraki, L., Latombe, J.-C., Li, T. Y., Motwani, R., and Raghavan, P. 1997. A random sampling scheme for path planning. *Int. J. Robotics Research 16,* 6, 759–774.

Barraquand, J. and Latombe, J.-C. 1991. Robot motion planning: A distributed representation approach. *Int. J. Robotics Research 10,* 6, 628–649.

Boulic, R., Thalmann, N. M., and Thalmann, D. 1990. A global human walking model with real-time kinematic personification. *The Visual Computer 6,* 6, 344–368.

Bowden, R. 2000. Learning statistical models of human motion. In *IEEE Workshop on Human Modeling, Analysis and Synthesis, CVPR2000*.

Brand, M. and Hertzmann, A. 2000. Style machines. *Computer Graphics (Proc. SIGGRAPH 2000) 34*, 183–192.

Bruderlin, A. and Calvert, T. W. 1989. Goal-directed animation of human walking. *Computer Graphics (Proc. SIGGRAPH '89) 23*, 233–942.

Bruderlin, A. and Williams, L. 1995. Motion signal processing. *Computer Graphics (Proc. SIGGRAPH '95) 29*, 97–104.

Chung, S.-K. and Hahn, J. K. 1999. Animation of human walking in virtual environments. In *Proc. Computer Animation '99*. 4–15.

Cohen, M. F. 1992. Interactive spacetime control for animation. *Computer Graphics (Proc. SIGGRAPH '92) 26*, 293–302.

Dijkstra, E. W. 1959. A note on two problems in connection with graphs. *Numerische Mathematik 1*, 269–271.

Gleicher, M. 1998. Retargetting motion to new characters. *Computer Graphics (Proc. SIGGRAPH '98) 32*, 33–42.

Gleicher, M. 2001. Motion path editing. In *Proc. ACM Symp. Interactive 3D Graphics*. 195–202.

Gottschalk, S., Lin, M. C., and Manocha, D. 1996. OBBtree: A hierarchical structure for rapid interference detection. *Computer Graphics (Proc. SIGGRAPH '96) 30*, 171–180.

Hodgins, J. K. and Pollard, N. S. 1997. Adapting simulated behaviors for new characters. *Computer Graphics (Proc. SIGGRAPH '97) 31*, 153–162.

Hodgins, J. K., Wooten, W. L., Brogan, D. C., and O'Brien, J. F. 1995. Animating human athletics. *Computer Graphics (Proc. SIGGRAPH '95) 29*, 75–78.

Hwang, Y. and Ahuja, N. 1992. Gross motion planning – a survey. *ACM Computing Surveys 24,* 3, 219–291.

Kalisiak, M. and van de Panne, M. 2000. A grasp-based motion planning algorithm for character animation. In *Proc. CAS '2000 – Eurographics Workshop on Simulation and Animation*. 43–58.

Kavraki, L., Kolountzakis, M., and Latombe, J.-C. 1996. Analysis of probabilistic roadmaps for path planning. In *Proc. IEEE Int. Conf. Robotics and Automation*. 3020–3025.

Kavraki, L. and Latombe, J.-C. 1994. Randomized preprocessing of configuration space for fast path planning. In *Proc. IEEE Int. Conf. Robotics and Automation*. 2138–2145.

Kavraki, L., Latombe, J.-C., Motwani, R., and Raghavan, P. 1995. Randomized query processing in robot motion planning. In *Proc. 27th Annual ACM Symp. Theory of Computing (STOC)*. 353–362.

Kavraki, L., Švestka, P., Latombe, J.-C., and Overmars, M. H. 1996. Probabilistic roadmaps for path planning in high dimensional configuration space. *IEEE. Trans. Robotics and Automation 12,* 4, 566–580.

Kindel, R., Hsu, D., Latombe, J.-C., and Rock, S. 2000. Kinodynamic motion planning amidst moving obstacles. In *IEEE Int. Conf. Robotics and Automation*. 537–543.

Ko, H. and Badler, N. I. 1996. Animating human locomotion with inverse dynamics. *IEEE Computer Graphics and Applications 16,* 2, 50–29.

Ko, H. and Cremer, J. 1995. VRLOCO: Real-time human locomotion from positional input streams. *Presence: Teleoperations and Virtual Environments 5,* 4, 1–15.

Koga, Y., Kondo, K., Kuffner, J., and Latombe, J.-C. 1994. Planning motions with intentions. *Computer Graphics (Proc. SIGGRAPH '94) 28*, 395–408.

Korein, J. U. and Badler, N. I. 1982. Techniques for generating the goal-directed motion of articulated structures. *IEEE CG&A 2,* 9, 71–81.

Kovar, L., Gleicher, M., and Pighin, F. 2002. Motion graphs. *ACM Transactions on Graphics (Proc. SIGGRAPH 2002) 21,* 3, 473–482.

Kuffner, J. and Latombe, J.-C. 1999. Fast synthetic vision, memory, and learning for virtual humans. In *Proc. Computer Animation '99.* 118–127.

Lamouret, A. and van de Panne, M. 1996. Motion synthesis by example. In *Proc. CAS '96 – Eurographics Workshop on Simulation and Animation.* 199–212.

Laszlo, J., van de Panne, M., and Fiume, E. 1996. Limit cycle control and its application to the animation of balancing and walking. *Computer Graphics (Proc. SIGGRAPH '96) 30*, 155–162.

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

Lee, J., Chai, J., Reitsma, P., Hodgins, J. K., and Pollard, N. 2002. Interactive control of avartars animated with human motion data. *ACM Transactions on Graphics (Proc. SIGGRAPH 2002) 21,* 3, 491–500.

Lee, J. and Shin, S. Y. 1999. A hierarchical approach to interactive motion editing for human-like figures. *Computer Graphics (Proc. SIGGRAPH '99) 33*, 395–408.

Lee, J. and Shin, S. Y. 2001. A coordinate-invariant approach to multiresolution motion analysis. *Graphical Models 63,* 2, 87–105.

Lee, J. and Shin, S. Y. 2002. General construction of time-domain filters for orientation data. *IEEE Trans. Visualization and Computer Graphics 8,* 2, 119–128.

Li, Y., Wang, T., and Shum, H.-Y. 2002. Motion texture: A two-level statistical model for character motion synthesis. *ACM Transactions on Graphics (Proc. SIGGRAPH 2002) 21,* 3, 465–472.

Lin, M. C. and Manocha, D. 1995. Fast interference detection between geometric models. *The Visual Computer 11,* 10, 542–561.

Marti, J. and Bunn, C. 1994. Automated path planning for simulation. In *Conf. AI, Simulation and Planning, AIS94.*

Mirtich, B. and Canny, J. F. 1995. Impulse-based simulation of rigid bodies. In *Proc. ACM Symp. Interactive 3D Graphics.* 181–188.

Molina-Tanco, L. and Hilton, A. 2000. Realistic synthesis of novel human movements from a database of motion capture examples. In *Proc. IEEE Workshop on Human Motion.* 137–142.

Multon, F., France, L., Cani, M.-P., and Debunne, G. 1999. Computer animation of human walking: a survey. *The Journal of Visualization and Computer Animation 10,* 3, 39–54.

Noser, H., Pandzic, I. S., Capin, T. K., Thalmann, N. M., and Thalmann, D. 1996. Playing games through the virtual life network. In *Proc. Alife '96.*

Noser, H., Renault, O., Thalmann, D., and Thalmann, N. M. 1995. Navigation for digital actors based on synthetic vision, memory, and learning. *Computers and Graphics 19,* 1, 7–19.

Overmars, M. H. and Švestka, P. 1994. A probabilistic learning approach to motion planning. In *Proc. Workshop on Algorithmic Foundations of Robotics.* 19–37.

Preparata, F. P. and Shamos, M. I. 1985. *Computational Geometry: An Introduction.* Springer-Verlag.

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

Raibert, M. H. and Hodgins, J. K. 1991. Animation of dynamic legged locomotion. *Computer Graphics (Proc. SIGGRAPH '91) 25*, 319–358.

Reich, B. D., Ko, H., Becket, W., and Badler, N. I. 1994. Terrain reasoning for human locomotion. In *Proc. Computer Animation '94*. 77–82.

Reynolds, C. W. 1987. Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics (Proc. SIGGRAPH '87) 21*, 25–34.

Rose, C., Cohen, M. F., and Bodenheimer, B. 1998. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics and Applications 18,* 5, 32–40.

Rose, C., Guenter, B., Bodenheimer, B., and Cohen, M. F. 1996. Efficient generation of motion transitions using spacetime constraints. *Computer Graphics (Proc. SIGGRAPH '96) 30*, 147–154.

Schödl, A., Szeliski, R., Salesin, D. H., and Essa, I. 2000. Video textures. *Computer Graphics (Proc. SIGGRAPH 2000) 34*, 489–498.

Shoemake, K. 1985. Animating rotation with quaternion curves. *Computer Graphics (Proc. SIGGRAPH '85) 19*, 245–54.

Sun, H. C. and Metaxas, D. N. 2001. Automating gait generation. *Computer Graphics (Proc. SIGGRAPH 2001) 35*, 261–269.

Svĕstka, P. and Overmars, M. H. 1998. Coordinated path planning for multiple robots. *Robotics and Autonomous Systems 23,* 4, 125–152.

Torkos, N. and van de Panne, M. 1998. Footprint-based quadruped motion synthesis. In *Proc. Graphics Interface '98*. 151–160.

Tu, X. and Terzopoulos, D. 1994. Artificial fishes: Physics, locomotion, perception, behavior. *Computer Graphics (Proc. SIGGRAPH '94) 28*, 43–50.

Unuma, M., Anjyo, K., and Takeuchi, R. 1995. Fourier principles for emotion-based human figure animation. *Computer Graphics (Proc. SIGGRAPH '95) 29*, 91–96.

van de Panne, M. 1997. From footprints to animation. *Computer Graphics Forum 16,* 4, 211–223.

Witkin, A. and Popović, Z. 1995. Motion warping. *Computer Graphics (Proc. SIGGRAPH '95) 29*, 105–108.