A Probabilistic Approach to Planning Biped Locomotion with Prescribed Motions

Min Gyu Choi Jehee Lee Sung Yong Shin

CS/TR-2001-162

February 6, 2001

K A I S T

Department of Computer Science

A Probabilistic Approach to Planning Biped Locomotion with Prescribed Motions

Min Gyu Choi Jehee Lee Sung Yong Shin

Abstract

Typical high-level directives for locomotion of human-like characters are encountered frequently in animation scripts or interactive systems. 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. Combining two novel ideas, that is, probabilistic path planning and hierarchical motion fitting, 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 attached 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.

1 Introduction

1.1 Motivation and Objectives

The recent advance in 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 [7, 10, 27, 37, 41, 43]. However, motion planning with such motion clips has not been explored, yet.

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 [14, 26] can hardly achieve such directives with captured motion clips.

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 using a set of prescribed motions. Combining two novel ideas, that is, probabilistic path planning [19] and hierarchical motion fitting [27], our scheme finds a sequence of input motion clips and that of target footprints, and then retargets the motion sequence to yield a desired motion that follows the footprints. The scheme enables a human-like figure to perform a variety of motions such as running in a plain region, 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 four categories: biped locomotion, human navigation, probabilistic path planning, and motion editing.

Biped locomotion: Generating realistic biped locomotion has received increasing attention in computer animation. Bruderlin and Calvert [6] 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.* [5] exploited biomechanical data to utilize their intrinsic dynamics. Ko and Badler [21] 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. Raibert and Hodgins [34] showed that hand-designed controllers can produce physically realistic legged locomotion. Hodgins *et al.* [13] extended those controllers to more complex models such as human athletics. Hodgins and Pollard [12] also described an automatic method to adapt existing controllers to new characters. An optimization-based scheme was proposed by van de Panne [42] to generate biped locomotion from given footprints. This scheme was further extended to quadruped locomotion [39].

Human navigation: Reynolds [36] 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.* [40] simulated reactive behaviors of artificial fishes with synthetic visions. Noser *et al.* [30] presented a local navigation model for human-like characters using synthetic visions. They also simulated a human memory model for avoiding obstacles [31]. Kuffner and Latombe [24] addressed a similar problem for dynamic environments. Reich *et al.* [35] suggested a real-time model for human navigation in an uneven terrain. They adopted simulated sensors to detect geometric features such as obstacles. Bandi and Thalmann [2] discretized a synthetic environment into a 3D uniform grid to search paths for autonomous characters.

Probabilistic path planning: Barraquand and Latombe [4] elaborated a randomized path planning technique, that is originally invented for escaping local minima in a potential field. Thereafter, Kavraki and Latombe [17] and Overmars and Sv̌estka [32] independently and jointly proposed similar methods [19] 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 in 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 [3, 16, 18]. Koga *et al.* [22] combined randomized path planning and inverse kinematics to automatically generate animation for human arm manipulation. Kalisiak and van de Panne [15] presented a grasp-based motion planning algorithm in a constrained 2D environment with designated handholds and footholds. Kindel *et al.* [20] 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 Willams [7] 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 [43] proposed a motion warping technique for the same purpose. Unuma *et al.* [41] used Fourier analysis techniques to interpolate and extrapolate motion data in the frequency domain. Lamouret and van de Panne [25] discussed a variety of issues in reusing motion clips. Rose *et al.* [37] generated seamless transitions between motion clips using spacetime constraints [8]. Gleicher [10] simplified the spacetime problem for motion retargetting, that is, adapting a pre-existing motion of a character for another character of the same structure and different size. Employing an optimization technique, he was able to achieve an interactive performance for motion editing. To accelerate this approach, Lee and Shin [27] presented a hierarchical displacement mapping technique based on the multilevel B-spline approximation. They also gave a fast inverse kinematics solver adopting the notion of an elbow circle given by Korein and Badler [23].

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 [14, 26] can hardly achieve a realistic animation of a human-like figure. On the other hand, motion editing techniques [10, 27, 37] 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 combine two novel ideas, that is, probabilistic path planning [19] and hierarchical motion fitting [27]. 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 [19]. For efficiency, we represent the configuration of the figure with that of its stance foot, called as a foothold, rather than posture itself. The roadmap can be modeled as a directed graph whose node represents a valid sample 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 a node to the other with a prescribed motion while preserving its liveness.

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 [1]. 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 [9] 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. We have acquired an input motion sequence in roadmap search 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 scheme [27] 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 randomized 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 c-space (configuration space) with the 2D position (u, v) of the stance foot and its resting yaw θ on the ground to reduce the dimension 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. 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, that 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 prohibitively. Observing that a node is usually not connected with those nodes that are far apart, 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 [33], 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 can be transformed, within a specified tolerance, to have its first and last footprints coincided 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 [11, 28, 29].

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.

Suppose that we attempt to connect two nodes of the roadmap with the motion M. Let $\mathbf{f}_s = (\mathbf{p}_s, \mathbf{q}_s)$ and $\mathbf{f}_e = (\mathbf{p}_e, \mathbf{q}_e)$ be their foothold configurations, respectively. To transform the footprint sequence $\mathbf{f}(M)$, we first adjust the positions of the footprints and then their orientations (See Figure 1).

To adjust position differences, we initially 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 $\mathbf{p}_e$. Accordingly, we scale the parameters of locomotion such as step length, velocity, turning angle and so on.

Now, we adjust orientation differences. Employing the logarithmic map $\log(\cdot)$ and exponential map $\exp(\cdot)$ of the quaternion algebra [38], 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 with the chord length parameterization of $\mathbf{p}_j$, $1 \leq j \leq m$. That is,

$$\mathbf{q}'_j = \mathbf{q}_j \exp((1 - t_j) \cdot \log(\mathbf{q}_1^{-1}\mathbf{q}_s) + t_j \cdot \log(\mathbf{q}_m^{-1}\mathbf{q}_e)), \tag{3}$$

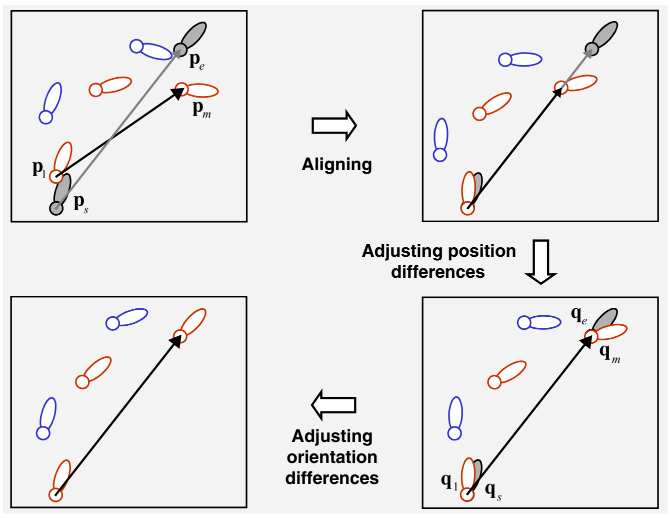


Figure 1: Footprint transformation

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.

2.3 Cost Function

Each directed edge of the roadmap has its cost that measures the efforts for a character to move from the head node to the tail 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. The cost of an edge can be computed from those factors.

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 $\mathbf{p}_j$, $1 \leq j \leq m$, 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 is that 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 liveness of the original 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 jump” 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:

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

where w_d and w_r are user-controllable weighting factors. In addition to the above terms, other factors such as obstacles along the edge may also be incorporated into the cost function.

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 [17, 19]. We adopt a heuristic method [17] to facilitate better interconnection. For each node v belonging to the node set V of the roadmap, a probability density function is defined by

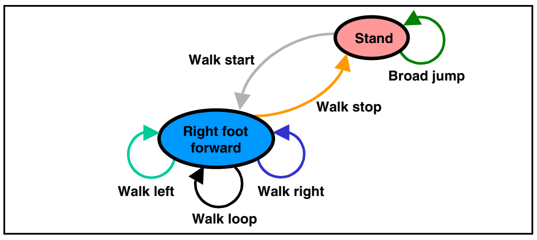
$$P(v|V) = \frac{1}{i_v + o_v + 1} / \sum_{u \in V} \frac{1}{i_u + o_u + 1}, \tag{7}$$

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 node interconnected poorly 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 [16, 17, 19].

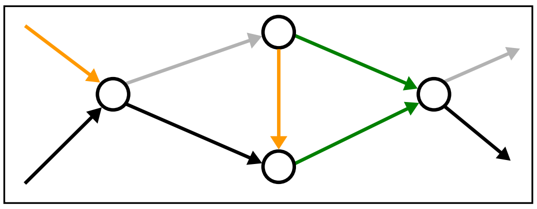
3 Roadmap Search

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.

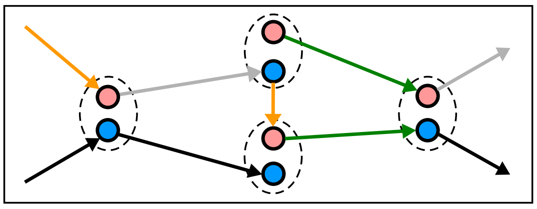
Before moving on to roadmap search, we describe a posture transition graph [1] in detail since it plays a crucial role in ensuring connectivity between motion clips. A node of the posture transition graph represents a valid posture of a biped figure, and an edge represents a motion clip. If the motion 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 may employ techniques [27, 37] for generating the transitions among motion clips to build the posture transition graph.



(a) Posture transition graph



(b) Roadmap



(c) Augmented roadmap

Figure 2: Augmented roadmap with posture transition graph

3.1 Path Finding

Given two foothold, $\mathbf{f}_x$ 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 search the roadmap for a minimum-cost path between x and y. When any of those two nodes is not connected to the roadmap, we employ a random walk to eventually connect it to the roadmap [4].

Searching the roadmap needs special care since motion clips are not guaranteed to stitch with each other. 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

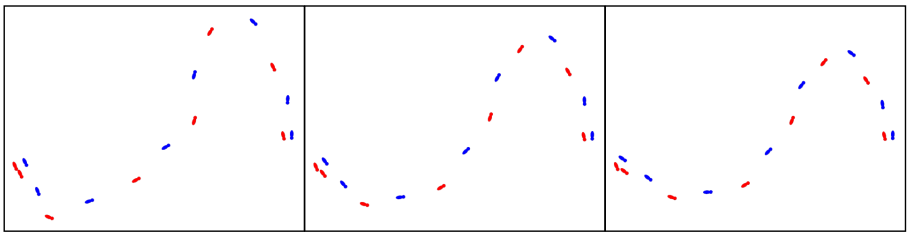
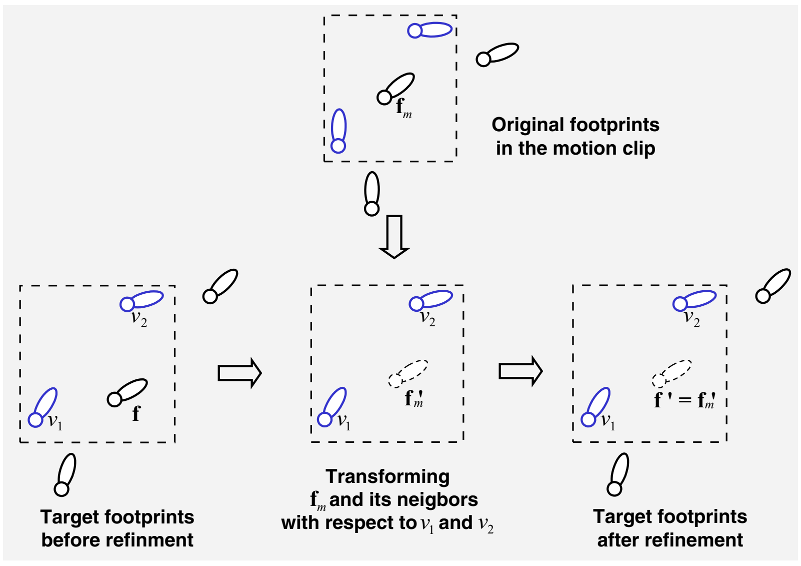


Figure 3: A sequence of footprints is repeatedly refined. The number of iterations is 0 (left), 4 (middle), and 8 (right).

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 [9].

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. Finally, we remove the nodes from G' that are not connected with any other nodes. 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 at most $k|V|$ 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 to move from a start node to a goal.

If path planning fails frequently, we sample the c-space more densely to add new



$\mathbf{f}_m$
Original footprints in the motion clip
v_2
v_1
$\mathbf{f}$
Target footprints before refinment
v_2
v_1
$\mathbf{f}_m'$
Transforming $\mathbf{f}_m$ and its neigbors with respect to v_1 and v_2
v_2
v_1
$\mathbf{f}' = \mathbf{f}_m'$
Target footprints after refinement

Figure 4: Footprint refining

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.

3.2 Footprint Refining

A target footprint sequence has been obtained from the motion clips tagged on the edges of the path in the roadmap. As explained in Section 2.2, the footprint sequence may differ from the original sequence in the motion clips within a specified tolerance. This difference may vary from edge to edge. To smooth the difference along the path, we give a local refining scheme that adjusts each footprint successively by referring to the neighboring footprints of the original, while ensuring the validity of the adjusted footprints. Applying the local refining scheme to every footprint in the sequence back and forth repeatedly along the path, we propagate the difference rather uniformly (See Figure 3).

We adopt the local planner developed in Section 2.2 for smoothing the difference.

For each footprint $\mathbf{f}$ in the target footprint sequence, we take its corresponding footprint $\mathbf{f}_m$ and its adjacent footprints in the original motion clip (See Figure 4). 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 $\mathbf{f}'$.

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 [10], we can formulate this task as a motion retargetting problem [10, 27, 37]. 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. Using the initial guess, we can employ the hierarchical motion fitting scheme [27] to retarget the input motion sequence for the footprints.

Since motion retargettting is well-described in [10, 27], 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 $\mathbf{f}(A)$

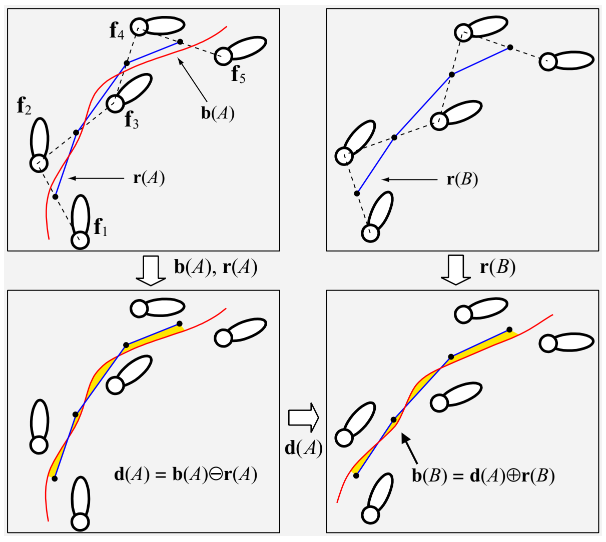


Figure 5: Body trajectory estimation

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 5). 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 $\mathbf{b}(A)$,

$$\mathbf{d}(A) = \mathbf{b}(A) \ominus \mathbf{r}(A), \tag{8}$$

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

$$\begin{aligned} \mathbf{b}(B) &= \mathbf{d}(B) \oplus \mathbf{r}(B) \approx \mathbf{d}(A) \oplus \mathbf{r}(B) \\ &= (\mathbf{b}(A) \ominus \mathbf{r}(A)) \oplus \mathbf{r}(B). \end{aligned} \tag{9}$$

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

$$(\bar{\mathbf{p}}_j, \bar{\mathbf{q}}_j) = (\frac{1}{2}(\mathbf{p}_j + \mathbf{p}_{j+1}), \mathbf{q}_j(\mathbf{q}_j^{-1}\mathbf{q}_{j+1})^{\frac{1}{2}}) \tag{10}$$

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 $[\bar{t}_1, \bar{t}_{m-1}]$ [38]. 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 $\mathbf{f}(B)$.

We now compute the displacement map $\mathbf{d}(A) = \mathbf{b}(A) \ominus \mathbf{r}(A)$. An ordered pair $(\mathbf{p}_i^r, \mathbf{q}_i^r)$, $\bar{t}_1 \le i \le \bar{t}_{m-1}$, 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 $\mathbf{b}(A) = \{(\mathbf{p}_i^b, \mathbf{q}_i^b) \in \mathbb{R}^3 \times \mathbb{S}^3 | 1 \le i \le n\}$ and $\mathbf{r}(A) = \{(\mathbf{p}_i^r, \mathbf{q}_i^r) \in \mathbb{R}^3 \times \mathbb{S}^3 | \bar{t}_1 \le i \le \bar{t}_{m-1}\}$, we define their displacement map $\mathbf{d}(A) = \{(\mathbf{u}_i, \mathbf{v}_i) \in \mathbb{R}^3 \times \mathbb{R}^3 | \bar{t}_1 \le i \le \bar{t}_{m-1}\}$ measured in the local coordinate system for $\mathbf{r}(A)$ as follows:

$$\begin{aligned} \mathbf{d}(A) &= \mathbf{b}(A) \ominus \mathbf{r}(A) \\ &= \{(\mathbf{p}_i^b, \mathbf{q}_i^b) \ominus (\mathbf{p}_i^r, \mathbf{q}_i^r) | \bar{t}_1 \le i \le \bar{t}_{m-1}\} \\ &= \{((\mathbf{q}_i^r)^{-1}(\mathbf{p}_i^b - \mathbf{p}_i^r)\mathbf{q}_i^r, \log((\mathbf{q}_i^r)^{-1}\mathbf{q}_i^b)) | \bar{t}_1 \le i \le \bar{t}_{m-1}\}. \end{aligned} \tag{11}$$

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

$$\begin{aligned} \mathbf{b}(B) &= \mathbf{d}(A) \oplus \mathbf{r}(B) \\ &= \{(\mathbf{u}_i,\ \mathbf{v}_i) \oplus (\mathbf{p}_i, \mathbf{q}_i) | \bar{t}_1 \le i \le \bar{t}_{m-1}\} \\ &= \{(\mathbf{q}_i \mathbf{u}_i \mathbf{q}_i^{-1} + \mathbf{p}_i, \mathbf{q}_i \exp(\mathbf{v}_i)) | \bar{t}_1 \le i \le \bar{t}_{m-1}\}. \end{aligned} \tag{12}$$

$\mathbf{b}(B)$ 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^{TM} plug-in on top of the Microsoft Windows^{TM} 2000. 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. The motion clips are sampled at the rate of 24 frames per second.

Our first experiment is for planning a walking motion of a human-like character. Figure 6 exhibits the resulting motion on a terrain in an island, in which foot planting is not allowed at any point in the sea. We use the posture transition graph with a set of live-captured motion clips for walking as depicted in Figure 2(a). 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 accompanying video clips.

The next experiment exhibits the capability of our planning scheme to cope with a difficult environment by using various motions. As shown in Figure 7, 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.

Our final experiment is for an environment with obstacles. As illustrated in Figure 8, the environment is a room with several pieces of furniture as obstacles. For collision detection, we employ a similar method in [11]. 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.

Table 1 summarizes the input data and the results of our experiments. For each example, our path planner performs three major steps: roadmap construction, roamap search, and motion generation. The running time of the rodmap 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

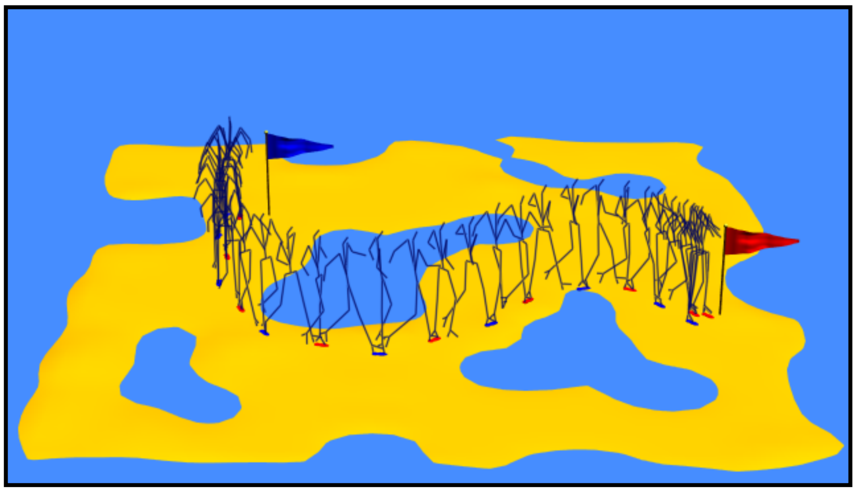


Figure 6: Simple terrain

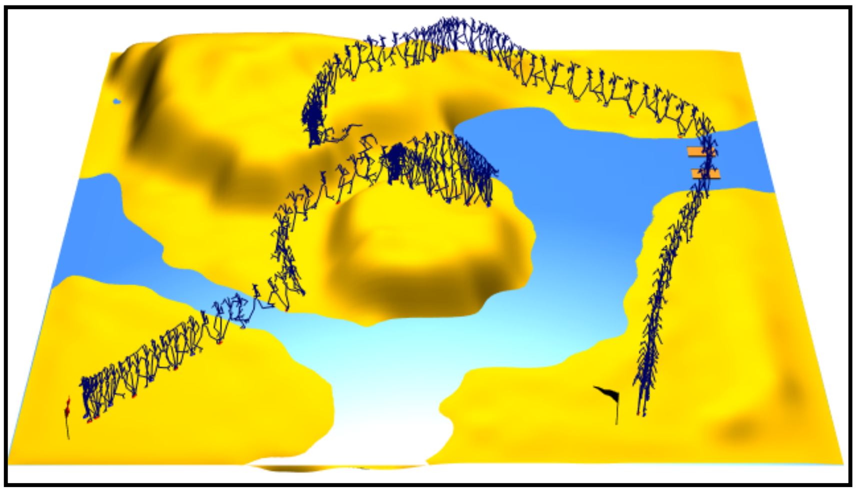


Figure 7: Complex terrain

Table 1: 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.

		Fig. 6	Fig. 7	Fig. 8
# 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	34.250	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
	retargetting time	0.090	0.670	0.250
	retargetting 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

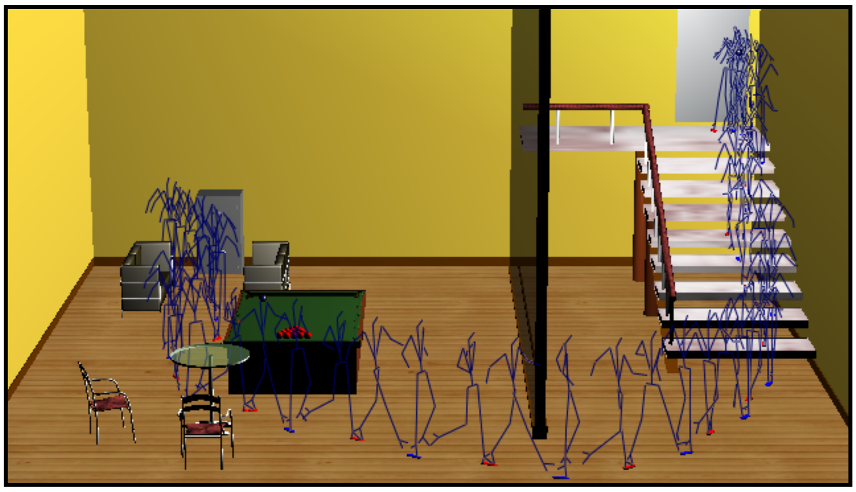


Figure 8: Room with furniture

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 an 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 [4, 15, 22] and probabilistic path planning with roadmaps [3, 16, 17, 18, 19, 32]. 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, we can find a path between any pair of configurations very efficiently. Our approach is particularly effective to perform repetitive point-to-point locomotion generations in the same environment.

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 c-space of postures, we sample that of footholds while accessing motion clips via edges incident to and from nodes representing footholds. Although this makes roadmap search a little complicated, the dimension of the c-space is reduced dramatically, and thus a moderate number of samples reflects well the connectivity of the c-space. This enables us to search a minimum-cost path at an interactive performance. Alternatively, one may prefer the pelvis configuration for the same purpose. Unlike end-effectors such as feet and hands, the pelvis cannot be used for interaction 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 c-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 c-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 [3, 16, 18] 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 c-space [3, 16, 18].

7 Conclusions

Animation scripts or interactive systems frequently 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 [19] and hierarchical motion fitting [27], 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 on an uneven terrain with a variety of motions, such as running in a plain region, walking over stepping stones, and jumping over a crevice.

Currently, our motion planner produces locomotion guided by the footholds at the nodes of the roadmap. We may use other features such as "handholds [15]" 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.

References

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

[2] S. Bandi and D. Thalmann. Space discretization for efficient human navigation. *Computer Graphics Forum*, 17(3):195–206, 1998.

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

[4] J. Barrauqand and J.-C. Latombe. Robot motion planning: A distributed representation approach. *Int. J. Robotics Research*, 10(6):628–649, 1991.

[5] R. Boulic, N. M. Thalmann, and D. Thalmann. A global human walking model with real-time kinematic personification. *The Visual Computer*, 6:344–358, 1990.

[6] A. Bruderlin and T. W. Calvert. Goal-directed animation of human walking. *Computer Graphics (Proc. SIGGRAPH '89)*, 23(4):233–242, 1989.

[7] A. Bruderlin and L. Williams. Motion siganl processing. *Computer Graphics (Proc. SIGGRAPH '95)*, 29:97–104, August 1995.

[8] M. F. Cohen. Interactive spacetime control for animation. *Computer Graphics (Proc. SIGGRAPH '92)*, 26(2):293–302, July 1992.

[9] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, pages 269–271, 1959.

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

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

[12] J. K. Hodgins and N. S. Pollard. Adapting simulated behaviors for new characters. *Computer Graphics (Proc. SIGGRAPH '97)*, 31:153–162, 1991.

[13] J. K. Hodgins, W. L. Wooten, D. C. Brogan, and J. F. O'Brien. Animating human athletics. *Computer Graphics (Proc. SIGGRAPH '95)*, 29:71–78, August 1995.

[14] Y. Hwang and N. Ahuja. Gross motion planning – a survey. *ACM Computing Surveys*, 24(3):219–291, 1992.

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

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

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

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

[19] L. Kavraki, P. Sv̌estka, J.-C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high dimensional configuration space. *IEEE. J. Robotics and Automation*, 12(4):566–580, 1996.

[20] R. Kindel, D. Hsu, J.-C. Latombe, and S. Rock. Kinodynamic motion planning amidst moving obstacles. *Submitted to IEEE Int. Conf. Robotics and Automation*, 2000.

[21] H. Ko and N. I. Badler. Animating human locomotion with inverse dynamics. In *Proc. IEEE Computer Graphics and Applications*, pages 50–59, March 1996.

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

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

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

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

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

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

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

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

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

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

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

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

[34] M. H. Raibert and J. K. Hodgins. Animation of dynamic legged locomotion. *Computer Graphics (Proc. SIGGRAPH '91)*, 25:349–358, 1991.

[35] B. D. Reich, H. Ko, W. Becket, and N. I. Badler. Terrain reasoning for human locomotion. In *Proc. Computer Animation '94*, pages 76–82, 1994.

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

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

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

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

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

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

[42] M. van de Panne. From footprints to animation. *Computer Graphics Forum*, 16(4):211–223, 1997.

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