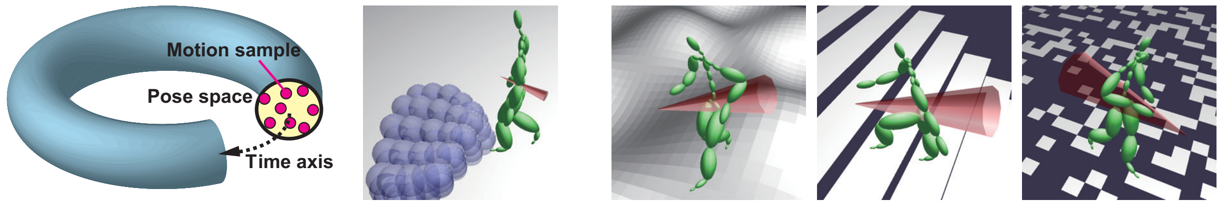
Motion Rings for Interactive Gait Synthesis

Tomohiko Mukai *
Square Enix



(a) Motion ring (b) Precomputed sphere (c) Bumpy terrain (d) Stairs (e) Stepping stones

Figure 1: *Continuous gait motion is synthesized by circulating through a motion ring while a circulation path is controlled to obey user input within the quarter gait cycle. The gait controller also optimizes the circulation path to establish the ground contact constraints by detecting the terrain surface using precomputed spheres.*

Abstract

This paper presents a practical system for synthesizing gait animation in game environments. As well as improving the reality of animation, we should improve the efficiency and the maneuverability of the character, both of which are essential for interactive games. Our system supplies these practical demands by integrating a motion interpolation technique and a sampling-based control mechanism. We introduce a parameterized looped motion data structure, called a motion ring, for synthesizing a variety of cyclic motions. A continuous gait motion is synthesized by circulating through the motion ring while the interpolation parameter is adaptively controlled according to the terrain condition. The gait controller uses a sampling-based precomputation technique which efficiently searches natural foot contact on terrain of an arbitrary surface shape. The interpolation parameter is also controlled to obey the user control within the duration of quarter gait cycle. Although our system slightly sacrifices the physical correctness of the synthesized motion in order to quickly respond to user input, critical visual artifacts such as foot-skating and jerky movement are prevented. We demonstrate the efficiency and versatility of our integrated system by interactively navigating the character on complex, uneven terrain.

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

Keywords: gait synthesis, motion interpolation, responsive animation

1 Introduction

Gait animation of humanoid characters is frequently seen in interactive applications, especially computer games. Recent hardware has the capability to execute a complicated motion synthesis algorithm for improving the animation quality. However, more efficient algorithms are still in great demand because of the limited capacity of memory and computational power of consumer game consoles. Another important requirement is that the animated character has to quickly and stably respond to user input. A slow-reacting character damages game design and displeases users. In other words, we can never use an "advanced" algorithm to generate realistic animation if it lacks responsiveness, stability, or efficiency. Consequently, we should optimize the trade-off between animation quality and computational cost depending on the game design, ensuring quick responsiveness.

Many game engines employ a data-driven, kinematic algorithm because of its high controllability. The typical method repeatedly plays a short motion clip of a straight walk while the position and orientation of the character's root is manipulated directly according to the terrain surface and user input. Although this method requires the minimal amount of computational resources, it causes critical visual artifacts such as foot-skating. Motion graph-based and move tree-based methods generate realistic motion by concatenating short motion clips. However, these methods have a delayed response to user input due to the sparse and finite states in the motion graph.

We think that an interpolation-based method is preferable for our purpose. A new gait motion is efficiently synthesized by interpolating a few sample clips with continuous motion parameters such as turning angle, speed, and walking style. The animated character quickly responds to user input by adjusting the motion parameter. The conventional methods also incorporate the character's surroundings in the motion parameter. For example, the motion of walking on a slope is parameterized by the slope angle. On the other hand, the width and height of the tread are used for controlling stair walking. However, it is not always obvious how to parameterize a complicated environment. Therefore, the existing methods require that the set of motion parameters be manually customized depending on the particular terrain condition.

We propose a practical system for generating an interactive gait animation on uneven terrain in the game environment. Our goal is to efficiently synthesize natural gait animation of a maneuverable character on complex uneven terrain as shown in Figure 1 (c), (d) and (e). To supply these practical demands, our system integrates an interpolation-based method and a sampling-based precomputation technique. The foundational data structure of our system is a parameterized looped motion data structure, called a *motion ring*, the principal axis of which represents the time axis, and the cross-section is the pose space parameterized by the interpolation weight of motion samples as shown in Figure 1 (a). A continuous cyclic

*e-mail:tmki@acm.org

motion is synthesized by circulating through the motion ring while smoothly changing the interpolation weight. We use a sampling-based precomputation algorithm for searching the optimal weight of every half gait cycle according to the terrain surface and the user input. This algorithm uses multiple invisible spheres which store the precomputed relation between interpolation weight and foot-strike point as shown in Figure 1 (b). As the center of the sphere is located at each foot-strike point, the contact constraint is satisfied by retrieving the interpolation weight from the sphere that collides with the environment. Because this approach only uses a general-purpose collision detection techniques to establish the ground contact constraints, there is no need for explicitly parameterizing the shape of the terrain surface. Our main contribution is that we build on previous interpolation-based techniques for optimizing the trade-off among the animation quality, efficiency, and maneuverability of the character. We demonstrate our integrated system efficiently and stably synthesizes an interactive gait animation on complicated uneven terrain with fewer visual artifacts.

1.1 Overview

A motion ring is a parameterized motion data in which its time index is circularly looped; rotations of each joint and velocity and angular velocity of the character's root are respectively continuous at every time frame. Its principal axis represents the time axis, and the cross section is the pose space which is parameterized by interpolation weight of motion samples. A motion ring of gait motion is created by bundling a set of *motion loops* which are each single cycle of looped gait motion. The motion loop is created from a sample of longer than one gait cycle. Our motion looping algorithm consists of three processes: dimensionality reduction of the motion sample, closed-curve fitting to the reduced motion sequence, and synthesis of the motion loop from the closed-curve (Section 3.1). Each motion loop is split into two phases by a *strike frame* which is the time frame when a swinging foot strikes the ground (Section 3.2). After spatially and temporally aligning motion loops (Section 3.3), the motion ring is compactly created using the dimensionality reduction technique (Section 3.4).

Once the motion ring is constructed, a continuous gait motion is synthesized by smoothly changing the interpolation weight during the circulation through the motion ring. Our gait controller searches optimal weights at each strike frame in a step-by-step manner: when a swinging foot strikes the ground, the controller searches the sequence of interpolation weight until the other foot strikes the ground. We introduce an invisible sphere, called a *contact sphere*, for efficiently precomputing the optimal weight. The contact sphere stores a weight value and its center is located at the foot-strike point provided using the stored weight. In the precomputation phase, the multiple contact spheres are sampled so as to infill the reachable region of the swinging foot (Section 4.1). The runtime search detects the contact spheres that collide with the ground, and the optimal one is selected according to the user control (Section 4.2). The retrieved weight is corrected using an iterative algorithm to reduce the positional error between the foot and the ground (Section 4.3). A continuous gait motion is synthesized by repeatedly searching the optimal weight every strike frame, and ease-in and ease-out transition is used to compute the weight of other intermediate frames.

2 Related work

Automatic gait generation algorithms have been of wide interest to the graphics and robotics communities. Physical simulations have the potential to generate an interactive motion which is automatically adapted to environmental changes [Coros et al. 2010; de Lasa et al. 2010; Lee et al. 2010b; chi Wu Zoran Popović 2010]. Nevertheless, data-driven, kinematic techniques provide more natural animation, and the kinematic algorithm is superior to a physics-based method in stability and controllability, which are essential to interactive games.

Typical game engines generate a continuous gait motion by repeatedly playing a single cycle of straight walking. The curved locomotion is synthesized by directly transforming the global position and orientation of the character's root to obey the user input. The greatest benefit of this method is the minimum requirement of computational resources. The direct transformation, however, causes critical visual artifacts such as foot-skating. Inverse kinematics and other post-editing techniques are therefore used to prevent the motion artifacts, which also damages the naturalness of the original motion. Motion graph-based and move tree-based methods generate a long stream of motion by stitching short clips. The graph traversal is optimized according to the user input and environmental constraints [Kovar et al. 2002; Lee et al. 2002; Arikan and Forsyth 2002; Lee and Lee 2004]. Whereas these graph-based techniques robustly synthesize natural animation in real-time, they lack the reactivity of the animated character to the user interaction [Reitsma and Pollard 2007]. Although the maneuverability can be improved by a graph refinement technique [Ikemoto et al. 2007; Ren et al. 2010] or using reinforcement algorithm [Treuille et al. 2007; Lee et al. 2009; Lee et al. 2010a], this problem is essentially unavoidable owing to the discrete and finite states in the motion graph.

Our system is based on a motion interpolation algorithm which synthesizes a new motion clip from the weighted average of similar clips. An infinite number of intermediate motions can be efficiently synthesized with continuous control of interpolation weight. Moreover, scattered data interpolation techniques enable the automatic optimization of the interpolation weight according to a few control parameters [Rose et al. 1998; Rose et al. 2001; Kovar and Gleicher 2004; Mukai and Kuriyama 2005]. Due to the prediction error of these statistical methods, the interpolated motion has to be edited with inverse kinematics or another post-editing technique to reduce the motion artifacts [Park et al. 2002; Park et al. 2004]. The alternative method heuristically generates pseudo-samples by sampling the relation between interpolation weight and the end-effector's position [Rose et al. 2001; Kovar and Gleicher 2004], which inspired our basic concept of the contact sphere. Although kinematic algorithms often neglect the dynamic property of human motion, such as linear and angular momentum, the interpolated motion shows generally acceptable quality because it is always visually similar to given samples.

Our system is closely related to interpolation-based gait synthesis [Sun and Metaxas 2001; Park et al. 2002; Park et al. 2004; Kwon and Shin 2005]. These methods parameterize short motion clips by speed and turning angle of locomotion. A continuous gait animation is generated by concatenating the interpolated clips. They provide efficient and responsive control of the animated character. However, the existing methods have three major factors of performance degradation. The first is that their basic data unit is a short motion clip. It means that the interpolated clips are concatenated using motion blending or extra motion clips for creating smooth transitions. The second is that their parameterization algorithm is based on scattered data interpolation which contains a margin of prediction error, so the resulting animation should be corrected using inverse kinematics. The final and most important factor is that extra motion parameters should be defined for dealing with the environmental conditions. For example, the slope angle of the terrain, and the height and width of the stairs are used for parameterizing gait motion. The gap of the terrain is also treated with the motion parameter. However, it is difficult to automatically parameterize complicated environments. In fact, previous works only demonstrate gait animation on stairs and gently sloped terrain using separate controllers customized for those particular environments. Moreover, online detection of the terrain surface is often time-consuming and memory-inefficient. Our system alleviates these performance problems by using a motion ring and a sampling-based controller.

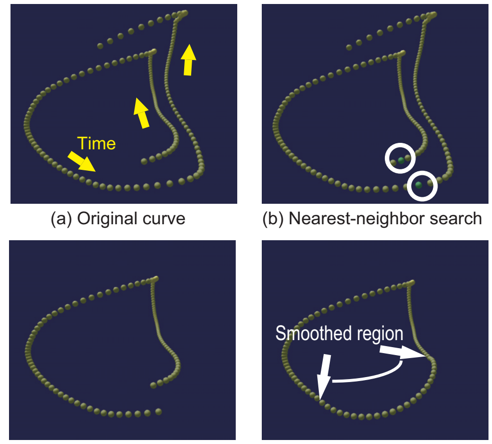
Many techniques have been proposed that combine motion graphs and interpolation techniques. A fat-graph [Shin and Oh 2006] consists of hub nodes and fat edges that represent a static pose and a group of motions parameterized by interpolation weight, respectively. Similarly, a parametric motion graph [Heck and Gleicher 2007] parameterizes both nodes and edges in the motion graph. A motion controller is proposed to find a near-optimal traversal of such parameterized graphs using reinforcement learning [Lo and Zwicker 2008]. Motion ring is assumed to be a variant of these parametric data structures specialized for cyclic motion. Major difference of motion ring to these previous data structures is that a motion ring allows a temporal change of the interpolation weight during interpolating motion clips. Conventional methods synthesize a new motion clip with time-invariant interpolation weight in order to preserve naturalness of motion. In contrast, since the interpolation weight can be quickly modified at any time-frame in the motion ring, our approach enhances a maneuverability of the character in compensation for the slight degradation of animation quality.

3 Motion ring construction

A motion ring represents a time series of interpolated poses whose time index is circularly looped, which is constructed by bundling a set of motion loops of single gait cycles. Given S motion samples $\mathbf{m}_s(t), t = \{1, \cdots, T_s\}, s = \{1, \cdots, S\}$ where T_s denotes the duration of s-th sample, the temporally aligned motion loop $\mathbf{l}_s(t), t = \{1, \cdots, T\}$ is synthesized from each sample, where T is the length of the principal axis of the motion ring. The interpolated pose at the t-th frame is represented as a function $\mathbf{y}(t, \mathbf{w}) = \sum_s w_s \mathbf{l}_s(t)$ where $\mathbf{w} = {}^t[w_1 \ \cdots \ w_S]$ is the interpolation weight. The pose vector $\mathbf{y}(t, \mathbf{w})$ consists of the position and orientation of the root, joint rotations, and a time-warp parameter. By the definition of the motion ring, the pose vector satisfies both the condition of cyclicity; $\mathbf{y}(t + T, \mathbf{w}) = \mathbf{y}(t, \mathbf{w})$, and the condition of continuity; $\mathbf{y}(t, \mathbf{w})$ connects $\mathbf{y}(t+1, \mathbf{w})$ with C_1 continuity. We here note that the continuity of the root configuration is guaranteed on its local velocity and angular velocity: the global position and orientation of the root are actually discontinuous in the motion ring as described in Section 3.3. Their continuity is ensured by aligning the local coordinate system as described in Section 4.

3.1 Motion looping in reduced space

We here introduce an automatic method for creating a motion loop from a motion sample. A motion loop is synthesized by extracting a single gait cycle from a motion sample of longer than one cycle and deforming it so that the final frame smoothly connects to the first. A conventional method creates a smooth transition around the connecting frames using motion blending technique with adequate transition length. However, the transition length and the blending function should be empirically optimized for creating natural motion transition [Wang and Bodenheimer 2008]. In contrast, an automated method extracts a single gait cycle using a motion segmentation technique, and generates optimal transition motion using a spacetime optimization technique. This complex algorithm, however, is too costly because there is only small variation among multiple gait cycles in human motion. Therefore, we use a simplified version of motion segmentation and spacetime optimization based on a dimensionality reduction technique. Our basic idea is to generate a low-dimensional closed-curve which approximates the original motion sequence in the reduced space. This approach significantly reduces the computational complexity because we can now use a simple curve fitting algorithm to generate the closed-curve.



(a) Original curve (b) Nearest-neighbor search (c) Single cycle extraction (d) Closed-curve fitting

Figure 2: *Looping of a running motion in low-dimensional space. (a) Low-dimensional curve of a running motion. (b), (c) Extraction of the single cycle using nearest neighbor search. (d) Closed-curve fitting using least-square method.*

Our system first computes a mapping from a motion sample $\mathbf{m}_s(t)$ to a low-dimensional curve $\mathbf{x}_s(t)$ using classical multi-dimensional scaling (CMDS) [Shin and Lee 2006] as illustrated in Figure 2(a), where the dimensionality of $\mathbf{x}_s(t)$ is set to 4. Secondly, the single cycle of the curve is extracted by searching for the nearest the two points based on the condition of cyclicity (Figure 2 (b), (c)). The distance between two points is computed by the weighted sum of Euclidean distance $\|\mathbf{x}_s(t) - \mathbf{x}_s(t')\|$ and the inverse of the inner product of the differential vectors $1.0/\{\mathbf{x}_s(t + 1) - \mathbf{x}_s(t - 1)\} \cdot \{\mathbf{x}_s(t' + 1) - \mathbf{x}_s(t' - 1)\}$. These two types of distance metrics evaluate the dissimilarity of static pose and that of local velocity of the whole body, respectively. Next, the smooth transition around the gap is generated using least-square fitting. We use a biquadratic Bézier splines for the approximation. Given the time region $[t_b, t_e]$ around the gap, the spline control points $\mathbf{c}_k$ are uniquely determined by the least square method:

$$\arg\min_{\mathbf{c}_1, \cdots \mathbf{c}_K} \sum_{t=t_b}^{t_e} \|\mathbf{x}_s(t) - \sum_{k=1}^{4} B_k(t)\mathbf{c}_k\|, \tag{1}$$

where $B_k(t)$ is a basis function of the biquadratic Bézier spline. The new curve $\tilde{\mathbf{x}}(t)$ is then generated by the spline interpolation $\tilde{\mathbf{x}}(t) = \sum_{k=1}^{4} B_k(t)\mathbf{c}_k$. The remaining free parameter is the time region $[t_b, t_e]$. Any region that is too short causes a discontinuous change in the resulting motion loop, and any region that is too long damages the original motion. The smoothing region $[t_b, t_e]$ is therefore determined to minimize the following objective function:

$$\begin{aligned} \arg \quad \min_{t_b, t_e} \quad & \alpha\|2\mathbf{x}_s(t_b) - \mathbf{x}_s(t_b - 1) - \tilde{\mathbf{x}}_s(t_b + 1)\| \\ + \quad & \alpha\|2\mathbf{x}_s(t_e) - \mathbf{x}_s(t_e + 1) - \tilde{\mathbf{x}}_s(t_e - 1)\| \\ + \quad & (t_e - t_b)^{-1/2} \left\{ \sum_{t=t_b}^{t_e} \|\mathbf{x}_s(t) - \tilde{\mathbf{x}}_s(t)\|^2 \right\}^{1/2}, \end{aligned} \tag{2}$$

where α denotes the weight coefficient, the first and second terms evaluate the continuity at the boundary points, and the third term is the root mean squared error from the original curve. We use the brute force algorithm to search for the optimal solution, and α is set to 10.0.

Finally, the motion loop $\tilde{\mathbf{l}}_s(t), t = \{1, \cdots, \tilde{T}_s\}$, where $\tilde{T}_s$ denotes the duration of the single cycle, is synthesized from the closed-curve $\tilde{\mathbf{x}}_s(t)$ using a non-parametric regression technique. The new pose at the t-th frame is synthesized by the weighted average of all poses in the original motion as $\tilde{\mathbf{l}}_s(t) = \sum b_i \mathbf{m}_s(i)$ where b_i is a blending weight. For the root configuration, this interpolation is performed in the velocity and angular velocity space, and position and orientation are obtained by the temporal integration. We use simple kriging [Wackernagel 2003] to optimize the blending weight as follows:

$$\begin{bmatrix} \mathbf{b} \\ \lambda \end{bmatrix} = \begin{bmatrix} \mathbf{X} & \mathbf{1}_{T_s}, \\ {}^t\mathbf{1}_{T_s} & 0 \end{bmatrix}^{-1} \begin{bmatrix} \chi \\ 1 \end{bmatrix}, \tag{3}$$

where $\mathbf{b} = {}^t[b_1 \ \cdots \ b_{T_s}]$ is blending weight vector, $\mathbf{X}$ is a $T_s \times T_s$ distance matrix in which $\mathbf{X}_{i,j} = \|\mathbf{x}_s(i) - \mathbf{x}_s(j)\|$, χ is a T_s dimensional distance vector in which $\chi_i = \|\mathbf{x}_s(i) - \tilde{\mathbf{x}}_s(t)\|$, $\mathbf{1}_{T_s}$ is a column vector of T_s ones, λ is a Lagrange multiplier, and the last element of the vector in the right-hand side constrains the weights b_i sum up to one.

3.2 Foot-strike detection

Before creating a motion ring, a strike frame, which is the time frame when a swinging foot strikes the ground, is detected in each motion loop. We use a simple algorithm which detects zero-crossings in the acceleration space of the feet [Bindiganavale and Badler 1998]. If multiple frames are detected, our system selects the most appropriate two frames at which the global velocity of the swinging foot is minimized. The temporal difference between those two frames is also constrained so as to be almost equal to the duration of a half of one gait cycle.

3.3 Motion alignment

A motion ring is composed of a set of spatially and temporally aligned motion loops $\mathbf{l}_s(t)$. All motion loops $\tilde{\mathbf{l}}_s(t)$ are temporally aligned using the motion registration algorithm based on dynamic time-warping [Kovar and Gleicher 2003] with key-time constraints on strike frames [Rose et al. 1998]. The incremental time-warping parameter [Kovar and Gleicher 2003] is added to the pose vector.

The spatial alignment is separately performed on each subsequence of half a gait cycle. The time-aligned motion loop is first split into two phases by the strike frame, and the local coordinate system is then defined for each phase as shown in Figure 3. At the first frame of each phase, the local coordinate system is computed so that the local x-axis (one of the horizontal axis) is parallel to the facing direction of the character and the local origin is located at the position of the supporting foot. The position and orientation of the character's root in each phase are expressed in the local coordinate system. As a result, the supporting foot is always located on the local origin at the strike frame for arbitrary interpolation weight.

3.4 Construction of compact motion ring

The data size of the motion ring increases proportional to the number of motion samples, the number of joints, and the number of time frames. We first eliminate redundant motion samples when many samples are available. We use the greedy algorithm to select a reduced set of motion samples. The initial element of the reduced set is the motion sample whose average stride length of two steps is smallest in all samples. Our algorithm then selects the most dissimilar sample from the elements in the reduced set, where the dissimilarity is computed using the distance between foot-strike

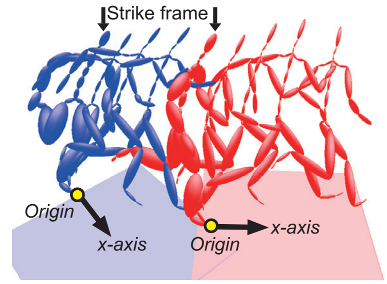


Figure 3: *Local coordinate system of the animated character in which the root configurations of half gait cycle is described.*

points in the local coordinate system. This iteration is terminated until the distance is shorter than a given threshold, which we set to 30cm.

We use principal geodesic analysis (PGA) [Tournier et al. 2009] to reduce the dimensionality of joint rotations, where the number of modes is automatically determined by keeping 99% of the original variations. We here note that the root configuration and the time-warp parameter are not compressed since they have greater influence on the animation quality. In the runtime phase, joint rotations are synthesized by interpolating the reduced principal components and decompressing it. The position and orientation of the root node and the time-warp parameter are separately interpolated. The data compression increases the memory efficiency as well as the computational performance of the online motion interpolation, even though the decompression requires extra computational cost.

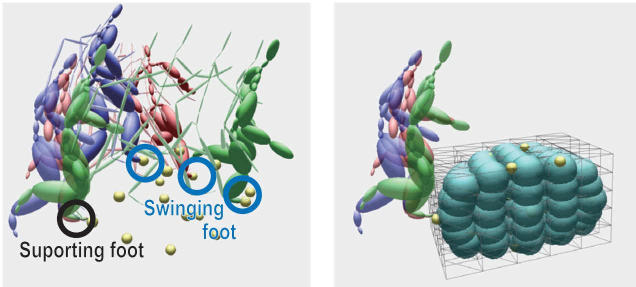
4 Gait synthesis using motion ring

A continuous gait motion is synthesized by circulating through the motion ring while changing the interpolation weight. To efficiently search the sequence of interpolation weight that maintains ground contact constraints, we use a contact sphere for precomputing the relation between interpolation weight and foot-strike point. A contact sphere stores the interpolation weight with which the swinging foot of an interpolated motion strikes the sphere's center. A set of contact spheres is precomputed so as to infill the reachable region of the swinging foot. In the runtime phase, the interpolation weight at the next strike frame is retrieved from a contact sphere which collides with the ground. If multiple spheres are available, the optimal one is selected according to the user input. The interpolation weight is finally corrected by the iterative algorithm based on statistical analysis. The interpolation weight at other intermediate frames is computed using the ease-in and ease-out transition.

4.1 Contact sphere generation

A set of contact spheres is generated using a sampling technique. The straightforward method performs the sampling in the weight space. Given i-th sampled weight $\mathbf{w}_i$, the pose vector is interpolated at the strike frame. Next, the foot-strike point $\mathbf{p}_i$ is computed in the local coordinate system, at where a contact sphere with radius r is then generated. We obtain a set of contact spheres $\{\mathbf{p}_i, \mathbf{w}_i, r\}, i \in \{1, \cdots, N\}$ by repeating this sampling process N times.

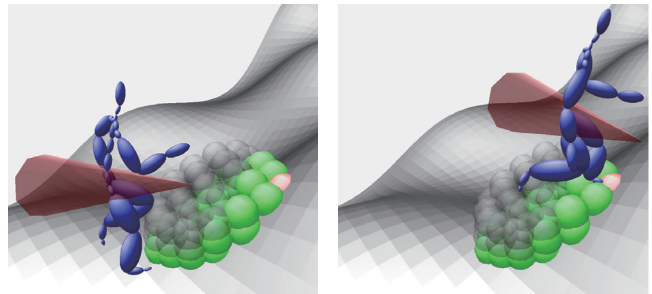
For uniformly sampling the foot-strike points, we employ a scattered data interpolation to predict the interpolation weight from the sampling point. The reachable region of the swinging foot is first approximated by a bounding box of all the sample's foot-strike points $\hat{\mathbf{p}}_s$ as shown in Figure 4. The sampling points $\mathbf{p}_i$ are then arranged in the regular lattice pattern in the bounding box. The



(a) Foot-strike position of motion samples

(b) Sampling contact spheres

Figure 4: *Sampling of contact spheres.*



(a) Selection of optimal contact sphere at the strike frame

(b) Interpolated pose at the next strike frame

Figure 5: *Weight search at strike frame. (a) The interpolation weight is retrieved from the optimal sphere colored in red, which satisfies the directional control, selected among the colliding spheres colored in green. (b) The swinging foot robustly strikes the ground at the next strike frame.*

interpolation weight $\mathbf{w}_i$ at the i-th sampling point $\mathbf{p}_i$ is predicted using simple kriging, similar to Equation 3. The prediction equation is expressed as follows:

$$\begin{bmatrix} \mathbf{w}_i \\ \lambda \end{bmatrix} = \begin{bmatrix} \mathbf{P} & \mathbf{1}_S, \\ {}^t\mathbf{1}_S & 0 \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{\rho} \\ 1 \end{bmatrix}, \tag{4}$$

where $\mathbf{P}$ is the $S \times S$ distance matrix in which $\mathbf{P}_{s,t} = \|\hat{\mathbf{p}}_s - \hat{\mathbf{p}}_t\|$, $\boldsymbol{\rho}$ is the S dimensional distance vector in which $\boldsymbol{\rho}_s = \|\mathbf{p}_i - \hat{\mathbf{p}}_s\|$, $\mathbf{1}_S$ is a column vector of S ones. Since the foot-strike point of the interpolated motion is actually located at different points $\mathbf{p}'_i$ owing to the prediction error, the contact sphere is generated at the actual position as $\{\mathbf{p}'_i, \mathbf{w}_i, r\}$. As a result, the contact spheres are arranged within a smaller region than the bounding box as shown in Figure 4 (b).

4.2 Optimal weight search

A basic method to search the circulation path through the motion ring is to determine the interpolation weight every half gait cycle; at a strike frame τ_i, the sequence of interpolation weight until the next strike frame τ_{i+1} is searched in a step-by-step manner. We explain this basic method here for notational convenience, although we will introduce a more responsive algorithm in Section 4.4. First, the runtime computation detects the contact spheres colliding with the ground at τ_i, which is illustrated by the green spheres in Figure 5. The optimal sphere is then selected among the colliding ones according to the user control, and the interpolation weight $\mathbf{w}(\tau_i)$ is retrieved from the optimal sphere. For instance, if the user directs the character to turn right, a sphere on the right side will be selected, as illustrated by the red sphere in Figure 5. The stride length is controlled by the distance of the contact sphere from the origin of the local coordinate system.

Once $\mathbf{w}(\tau_{i+1})$ is determined, the interpolation weight at other intermediate frames $\mathbf{w}(t)$ where $\tau_i < t < \tau_{i+1}$ are calculated using ease-in and ease-out transition as $\mathbf{w}(t) = (1 - H(t))\mathbf{w}(\tau_i) + H(t)\mathbf{w}(\tau_{i+1})$ where $H(t)$ is the transition function. We use a cubic polynomial $H(t) = t'^2 - 3t'^3 + 1, t' = (t - \tau_i)/(\tau_{i+1} - \tau_i)$ for the smooth transition.

The pose vector at each time frame is synthesized using the interpolation weight. The time frame t is monotonically advanced with the incremental time-warp parameter [Kovar and Gleicher 2003]. The contact sphere and the character are both transformed from the local coordinate system to the global coordinate system. The coordinate transformation can be simply updated every strike frame so that the local origin and local x-axis match the global position of the supporting foot and the global facing direction of the character, respectively.

4.3 Iterative weight correction

By synthesizing the motion using the interpolation weight of the contact sphere, the foot-strike point is located adjacent to the ground. The spatial error up to the radius of the contact sphere can be corrected using a stock IK solver, but kinematic solver often damages the animation quality. For example, the traditional method modifies the foot trajectory so as to satisfy the contact constraints, and edits the joint rotations using the IK solver. However, it is not easy to optimize where and how long the ground contact is constrained for generating smooth and natural motion. Moreover, most IK techniques require tedious parameter tuning to generate natural movement.

We introduce an iterative algorithm to correct the interpolation weight rather than directly adjusting joint rotations. Our system iteratively corrects the interpolation weight so that the swinging foot strikes "somewhere" on the ground. Each iteration starts with the projection of the foot position $\mathbf{f}$ onto the ground. The projected point $\mathbf{g}$ is searched by intersecting the rays passing through the foot position. The optimal weight correction $\triangle\mathbf{w}$ is then predicted based on Equation 4 as follows:

$$\begin{bmatrix} \triangle\mathbf{w} \\ \lambda \end{bmatrix} = \beta \begin{bmatrix} \mathbf{P} & \mathbf{1}_S \\ {}^t\mathbf{1}_S & 0 \end{bmatrix}^{-1} \begin{bmatrix} \triangle\boldsymbol{\rho} \\ 0 \end{bmatrix}, \tag{5}$$

where $\triangle\boldsymbol{\rho}$ is the displacement of the distance vector in which $\triangle\boldsymbol{\rho}_s = \|\mathbf{g} - \hat{\mathbf{p}}_s\| - \|\mathbf{f} - \hat{\mathbf{p}}_s\|$, and β is the positive scaling coefficient less than one. We set the scaling coefficient to $\beta = 0.5$ since large β sometimes results in unstable convergence owing to the prediction error. This equation is derived by subtracting the prediction equation (4) with respect to the foot point $\mathbf{p}$ from that with respect to the projected point $\mathbf{g}$. This linear system is efficiently solved thanks to the precomputable inverse matrix of the right-hand side. The weight $\mathbf{w}$ is iteratively updated as $\mathbf{w} + \triangle\mathbf{w} \rightarrow \mathbf{w}$ until the positional error $\|\mathbf{g} - \mathbf{f}\|$ is less than a threshold ϵ, which we set to 3 cm. This weight correction rapidly converges to the feasible solution in a few iterations. Due to the small degree of the weight correction $\triangle\mathbf{w}$, our system guarantees the synthesized motion always falls within the valid internal space of the motion ring.

4.4 Practical implementation

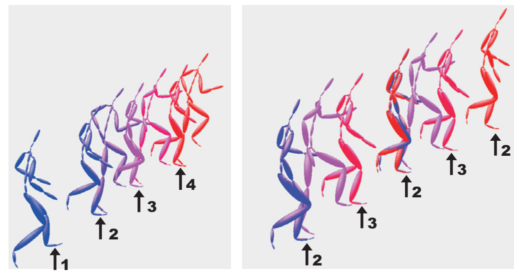
Responsive weight search We have explained the basic algorithm to search the interpolation weight every half gait cycle in

Section 4.2. This method causes time delay in response to user input because the circulation path cannot be changed between strike frames. To reduce the control delay up to the duration of a half gait cycle, the interpolation weight is immediately recomputed according to the user input while avoiding the discontinuous change in synthesized motion. Given the user input at time frame t where $\tau_i < t < \tau_{i+1}$, the weight of the next strike frame $\mathbf{w}(\tau_{i+1})$ is modified by re-selecting the optimal sphere from the colliding ones. The collision detection can be omitted in this recomputation because the contact spheres remain stationary between the strike frames. The weight of the previous strike frame $\mathbf{w}(f_c)$ is then modified using linear extrapolation as $\mathbf{w}(\tau_i) = \{\mathbf{w}(t) - H(t)\mathbf{w}(\tau_{i+1})\}/\{1 - H(t)\}$ in order to ensure the consistency of the current weight $\mathbf{w}(t)$. This method, however, often causes discontinuous change in the synthesized motion. When the user control is applied just before the next strike frame, the interpolation weight is drastically changed in a short period and results in an impossible velocity of the movement. To avoid this problem, user input is ignored for a certain time period before each strike frame. We set the time length to the duration of a quarter gait cycle for the following two reasons. First, the duration of quarter gait cycle is short enough for the interactive games. Secondly, a mid-swing phase of human locomotion shows smaller variation in pose than that around a strike frame. Therefore, a recomputation of the circulation path before the mid-swing phase results in a relatively small change in motion. In contrast, the recomputation around the strike frame causes a noticeable change relative to displacement magnitude of the interpolation weight.

Combination with direct transformation Since our standard method is fully based on motion interpolation, many motion samples are required for dealing with a variety of turning angles, stride lengths, and step heights. For reducing the number of motion samples, we can employ the concept of [Sun and Metaxas 2001] that combines motion interpolation and direct transformation, which we call a *combined method*. This method synthesizes straight walk on uneven terrain using motion interpolation and the curved locomotion is generated by rotating the character's root. The set of motion samples consists of only straight walks, and the contact spheres are generated on the sagittal plane of the character. Given the user input, the character's local coordinate system is rotated around the vertical axis which passes through its origin. This rotation causes less foot-skating because the supporting foot is located on the same origin. The interpolation weight is then recomputed using the responsive search algorithm with detection of collisions between of transformed contact spheres and the terrain surface. Another advantage of the combined method is that the user control is exactly satisfied using the direct transformation instead of the discrete selection among contact spheres. A disadvantage of the combined method is the degradation of the naturalness of synthesized motion in compensation for the improvements in the memory efficiency and controllability.

Efficient collision detection The bottleneck of our system is the collision detection between contact spheres and terrain geometry. We use a well-known data structure, called box-tree, to accelerate the collision detection [Ericson 2005]. By representing the environmental geometry using a binary box-tree, the computational cost is significantly reduced from $O(NG)$ to $O(N \log_2 G)$ where G denotes the number of polygons in the environment. For better performance, we use a two-step traversal of the box-tree using a boundary sphere of all contact spheres. The first breadth-first traversal detects the nodes in the box-tree which collide with the boundary sphere. This traversal terminates at user-specified depth d. The second depth-first traversal resumes from the collided nodes on depth d, and checks whether each contact sphere collides with any leaf node. As this two-step traversal requires $O(d + (\log G - d)N)$ computational steps, the optimal depth d is determined so as to satisfy

Figure 6: *The physical environment in the motion capture studio. The subject walked on the flat floor and several sets that imitate the spiral stairs.*



(a) Motion sample (b) Synthesized motion loop

Figure 7: *Looping of a walking-up motion on the spiral stairs. The motion loop is synthesized from the segment containing the second and third foot-strikes.*

$(\log G - d + 1)N - 1 \geq (\log G - d)N \leq (\log G - d - 1)N + 1$. For further improvement of the worst frame rate, the second traversal is performed over several time frames before the strike frame. When F frames are used for this time-sharing processing, $O((\log G - d)N/F)$ computational steps are required at each frame. We set F to the duration of one quarter of a gait cycle because the user input is neglected for that duration by the responsive search algorithm.

5 Experimental results

5.1 Data acquisition

We built physical sets in our motion capture studio. Each set was designed to imitate eight steps of straight stairs or spiral stairs as shown in Figure 6. We used two different curvatures for the spiral stairs which drew a quarter or a half of a circle every eight steps. We also used two different gradients for each of the stairs. The subject walked on the flat floor, and walked up and down on each set with two different stride lengths. We recorded a total of 43 motion clips of the 81 DOF human figures at 60 Hz for walking and running motions respectively. Each captured motion consists of between 5 and 9 steps, and the middle 3 steps are used as a motion sample for constructing the motion ring.

5.2 Motion ring construction

Figure 7 (a) illustrates a walking-up motion on the spiral stairs, which shows four foot-strikes. The motion loop is synthesized by extracting the segment which contains the second and third foot-strikes as illustrated in Figure 7 (b). Our motion looping technique generates a smooth and foot-skate-free motion loop. A continuous stair walk of an arbitrary number of steps can be generated by repeatedly connecting it without motion blending.

Our standard method selects 15 and 16 motion samples as the reduced set for constructing motion rings of walking and running motions, respectively. The 75-dimensional joint rotation is approximated by 42-dimensional principal components. Each motion sample is temporally aligned to have 60 frames (one second). The data size of each motion ring is less than 200 kB in a single-precision floating point format. On the other hand, the combined method constructs about 130 kB of motion rings from 11 motion samples of straight walking and straight running, respectively. The other parameters are equal to those of the standard method.

5.3 Gait synthesis in complex environments

We constructed three types of virtual environments; bumpy terrain, continuous stairs with irregular step height, and random stepping stones. The user controlled the turning angle and stride length of the locomotion using an analog joystick. We observed that the control delay to the joystick input is almost negligible, and the synthesized motion is smoothly changed according to the user input and the shape of terrain surface. However, the animated characters sometimes fail to obey the user input, especially on the stairs and stepping stones, because the ground contact constraints take priority over the user input. This result shows that the controllability of our system largely depends on the configuration of contact spheres.

We manually connects multiple motion rings of walking, running, and resting motions. These different behaviors are switched using smooth motion transition according to the joystick command. The motion ring of resting motion is constructed using 9 motion samples. Each motion sample maintains the same feet position throughout, so every time frames of the motion ring are regarded as strike frame and one set of contact spheres is shared by all of them. The manually-connected motion rings provided quick and smooth transition between different behaviors, but we observed that it often violates the physical validity such as momentum conservation. This problem could be alleviated by incorporating the mechanism to synthesize the natural transition based on sophisticated motion interpolation [Safonova and Hodgins 2005] or precomputation [Lo and Zwicker 2008].

5.4 Performance evaluation

The computational performance of our standard method and combined method are evaluated by measuring the turnaround time to synthesize animation frame excluding rendering time. We measured the turnaround time for five minutes on an Intel Xeon 3.3GHz PC using one core. The number of contact spheres is 5×5×5 for the standard method and 5×5 for the combined method, respectively. The radius of all contact spheres is fixed to $r = 20$ cm. The test environment is the bumpy terrain as shown in Figure 1 (c), which consists of $10,000$ polygons.

Table 1 summarizes the statistics of the computational performance, and Figure 8 shows the detailed profile of the turnaround time for 800 frames. These result shows that our system takes more computational time than that of existing methods [Park et al. 2002; Park et al. 2004; Mukai and Kuriyama 2005; Kwon and Shin 2005]. We believe that this performance degradation is reasonable because our system provides a more general solution to deal with complex game environments, while still achieving a practical performance and controllability. Both the standard and combined methods show the worst computational load around the strike frames at which the weight search and weight correction are executed. On the other hand, there is no significant difference in the computational performance between them. Although the combined method reduces the computational cost of motion interpolation and collision detection thanks to fewer motion samples and contact spheres, it requires extra computation to execute the collision detection whenever the user control is applied. In fact, the combined method takes much computational time when synthesizing curved locomotion as shown in

Table 1: *Statistics of the computational performance of our standard and combined methods.*

[frames/sec]

	avg.	sd.	min.	max.
Standard method	27,071	3,588	15,412	31,400
Combined method	26,939	3,165	15,821	32,690

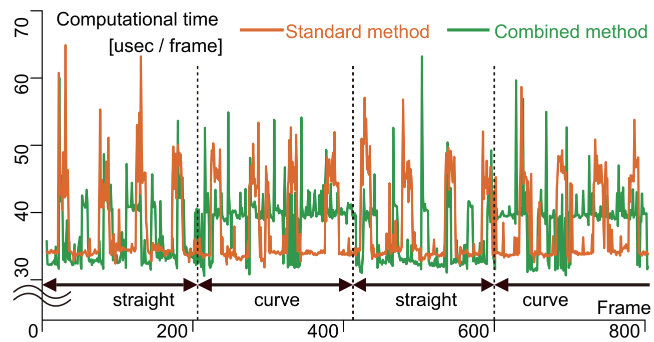


Figure 8: *Profile of computational time of our standard and combined methods.*

Figure 8. These results indicate that the standard method can be used in most cases because of the higher quality of synthesized animation and almost the same computational cost. The combined method is useful when data reduction or high controllability of the character is demanded even if the animation quality is sacrificed.

6 Conclusions

We have proposed an integrated system for synthesizing gait motion in interactive game environments. The fundamental data structure of our system is a parameterized motion loop, called a motion ring. The motion looping algorithm combines a dimensionality reduction technique and a closed-curve fitting algorithm by which a motion loop is automatically generated from a given sample lasting longer than one gait cycle. The motion ring is constructed by bundling a reduced set of motion loops whose redundancy is also eliminated using statistical analysis. A continuous gait motion is efficiently synthesized by circulating through the compact motion ring. Our gait controller searches the interpolation weight using precomputed contact spheres and an iterative weight correction algorithm for preventing foot-skating on complex, uneven terrain. Our system guarantees that the animated character rapidly responds to user input within the duration of a quarter gait cycle.

Our system can be applied for synthesizing a wider variety of cyclic motions other than gait motion. We have demonstrated that different behaviors can be integrated by simply connecting multiple motion rings. However, adequate connection between motion rings is manually created with tedious labor in our current system. We will investigate an automated method to organize a graph structure of multiple motion rings of different behaviors.

Our future work also includes the development of a general framework of environment-driven motion synthesis. Our gait controller measures only the shape of the terrain surface using contact spheres. In practical application, however, the terrain is often annotated with geological conditions. For example, the ground is covered with grass, puddles, snow, and sand, and such annotation data is embedded in the environment itself. Our system is well suited for precisely retrieving such semantic information from the terrain and an object in order to generate reactive behavior to the environmental conditions, the concept of which is similar to motion patches [Lee et al. 2006]. We will examine the applicability of our system to environment-driven motion control.

Acknowledgement

We would like to thank Yutaka Miyauchi and Motion Capture Group at Square Enix for their assistance in the motion capture session, and anonymous reviewers for their constructive comments. We also thank our colleagues for their support.

References

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

BINDIGANAVALE, R., AND BADLER, N. I. 1998. Motion abstraction and mapping with spatial constraints. In *Lecture Notes In Computer Science*, vol. 1537, 70–82.

CHI WU ZORAN POPOVIĆ, J. 2010. Terrain-adaptive bipedal locomotion control. *ACM Transactions on Graphics 29*, 4, 72.

COROS, S., BEAUDOIN, P., AND VAN DE PANNE, M. 2010. Generalized biped walking control. *ACM Transactions on Graphics 29*, 4, 130.

DE LASA, M., MORDATCH, I., AND HERTZMANN, A. 2010. Feature-based locomotion controllers. *ACM Transactions on Graphics 29*, 4, 131.

ERICSON, C. 2005. *Real-Time Collision Detection*. Morgan Kaufmann.

HECK, R., AND GLEICHER, M. 2007. Parametric motion graphs. In *Proc. of ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, 129–136.

IKEMOTO, L., ARIKAN, O., AND FORSYTH, D. 2007. Quick transitions with cached multi-way blends. In *Proc. of ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, 145–151.

KOVAR, L., AND GLEICHER, M. 2003. Flexible automatic motion blending with registration curves. In *Proc. of ACM SIGGRAPH/Eurographics Symposium on Computer Animation 2003*, 214–224.

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

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

KWON, T., AND SHIN, S. Y. 2005. Motion modeling for on-line locomotion synthesis. In *Proc. of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 29–38.

LEE, J., AND LEE, K. H. 2004. Precomputing avatar behavior from human motion data. In *Proc. of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 79–87.

LEE, J., CHAI, J., REITSMA, P. S. A., HODGINS, J. K., AND POLLARD, N. S. 2002. Interactive control of avatars animated with human motion data. *ACM Transactions on Graphics 21*, 3, 491–500.

LEE, K. H., CHOI, M. G., AND LEE, J. 2006. Motion patches: Building blocks for virtual environments annotated with motion data. *ACM Transactions on Graphics 25*, 3, 898–906.

LEE, Y., LEE, S. J., AND POPOVIĆ, Z. 2009. Compact character controllers. *ACM Transactions on Graphics 28*, 5, 169.

LEE, Y., WAMPLER, K., BERNSTEIN, G., POPOVIĆ, J., AND POPOVIĆ, Z. 2010. Motion fields for interactive character animation. *ACM Transactions on Graphics 29*, 5, 138.

LEE, Y., KIM, S., AND LEE, J. 2010. Data-driven biped control. *ACM Transactions on Graphics 29*, 4, 129.

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

MUKAI, T., AND KURIYAMA, S. 2005. Geostatistical motion interpolation. *ACM Transactions on Graphics 24*, 3, 1062–1070.

PARK, S. I., SHIN, H. J., AND SHIN, S. Y. 2002. On-line locomotion generation based on motion blending. In *Proc. of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*.

PARK, S. I., SHIN, H. J., HOON KIM, T., AND SHIN, S. Y. 2004. On-line motion blending for real-time locomotion generation. *Computer Animation and Virtual Worlds 15*, 3-4, 125–138.

REITSMA, P. S. A., AND POLLARD, N. S. 2007. Evaluating motion graphs for character animation. *ACM Transactions on Graphics 26*, 4, 18.

REN, C., ZHAO, L., AND SAFONOVA, A. 2010. Human motion synthesis with optimization-based graphs. *Computer Graphics Forum 29*, 2, 545–554.

ROSE, C. F., BODENHEIMER, B., AND COHEN, M. F. 1998. Verbs and adverbs: Multidimensional motion interpolation'. *IEEE Computer Graphics and Applications 18*, 5, 32–40.

ROSE, C. F., SLOAN, P.-P. J., AND COHEN, M. F. 2001. Artist-directed inverse-kinematics using radial basis function interpolation. *Compuer Graphics Forum 20*, 3, 239–250.

SAFONOVA, A., AND HODGINS, J. K. 2005. Analyzing the physical correctness of interpolated human motion. In *Proc. of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 171–180.

SHIN, H. J., AND LEE, J. 2006. Motion synthesis and editing in low-dimensional spaces. *Computer Animation and Virtual Worlds 17*, 3-4, 219–227.

SHIN, H. J., AND OH, H. S. 2006. Fat graphs: Constructing an interactive character with continuous controls. In *Proc. of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 291–298.

SUN, H. C., AND METAXAS, D. N. 2001. Automating gait generation. In *Proc. of SIGGRAPH 2001*, 261–270.

TOURNIER, M., WU, X., COURTY, N., ARNAUD, E., AND REVERET, L. 2009. Motion compression using principal geodesics analysis. *Computer Graphics Forum 28*, 2, 337–346.

TREUILLE, A., LEE, Y., AND POPOVIĆ, Z. 2007. Near-optimal character animation with continuous control. *ACM Transactions on Graphics 27*, 3, 7.

WACKERNAGEL, H. 2003. *Multivariate Geostatistics*. Springer Varlag.

WANG, J., AND BODENHEIMER, B. 2008. Synthesis and evaluation of linear motion transitions. *ACM Transactions on Graphics 27*, 1, 1.