On-line Locomotion Generation Based on Motion Blending

Sang Il Park[*] Hyun Joon Shin[†] Sung Yong Shin[‡]

Division of Computer Science
Korea Advanced Institute of Science Technology

Abstract

Locomotion such as walking, jogging, and running is one of the most basic forms of daily human motions. However, the previous methods can hardly generate the convincing locomotion of a character following a curved path with a desired speed and style. Based on scattered data interpolation, we propose a novel approach for on-the-fly generation of convincing locomotion, given parameters such as speed, turning angle, and style, on top of others given in the previous approaches. We first present an incremental scheme for timewarping to align the example motion clips of various speeds. Then, we provide a novel scheme for joint angle blending which guarantees similar poses to have similar representations. Finally, we show how to adapt the blended motion to the target character and the environment in an on-line, real-time manner. The resulting motions are not only convincing but also effectively controlled to reflect animator's intention. Our approach is efficient enough for on-line applications such as real-time animation systems and video games.

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

Keywords: Animation, Animation with Constraints, Human Body Simulation

1 Introduction

Locomotion such as walking, jogging, and running is one of the most basic forms of daily human motions. Therefore, it is natural to observe scenes with characters in locomotion frequently in computer games and character animations. The importance of locomotion generation is also revealed by commercial computer animation packages equipped with locomotion generation functions in various forms. The great demand on locomotion generation has provided a driving force for continuing research efforts.

However, the previous methods can hardly generate the convincing locomotion of a character following a curved path with a desired speed and style [20]. Based on mathematical models and fragments of code to describe a human locomotion, procedural approaches are able to produce a variety of locomotion efficiently, but results have often revealed some artifacts because of their kinematics-based nature [2, 3, 4, 5]. Dynamics-based simulations have been tried to automatically generate physically correct motions [12, 15, 17, 22]. Recent motion capture-based approaches have yielded realistic locomotion in real time. However, they lack high-level controllability to synthesize a curved locomotion with given aspects [6, 10, 23, 27].

In this paper, we present a novel approach for on-the-fly generation of convincing locomotion that satisfies time-varying constraints on its trajectory, speed, and style, which are given in an on-line manner. Our basic idea is to generate a motion by blending a set of example motion clips captured from live human motions. Relying on blending, our approach is efficient enough to guarantee real-time performance. Using live-captured motion clips, it also allows us to obtain convincing motions of high quality. Finally, parameterizing example motions properly, it achieves controllability over various aspects of motion such as speed, turning angle, and style.

To generate a motion by blending a set of example motion clips, we should address three issues: First, we need to provide a timewarping scheme to align the motion clips of a wide range of speed. Second, the conventional methods using Euler angles are required to preprocess the example motions for effective motion blending to ensure that similar poses use similar joint angle representation. Without this preprocessing, similar postures would be represented rather differently to cause difficulty in blending. Finally, since the target character has a different size and proportions from an actor (or puppeteer), a blended motion needs to be adapted to the target character in an on-line manner. To tackle the first issue, we present an incremental scheme for timewarping to align the example motion clips of various speeds. We solve the second issue by providing a novel scheme for blending joint angles based on quaternion algebra. To address the last issue, we compute the target stance foot print position at each frame by blending those of the example motions and then adapt the blended motion to the target character and the environment in an on-line, real-time manner.

The remainder of this paper is organized into several sections: We begin our discussion with reviewing previous results in Section 2 and then provide an overview of our scheme in Sections 3. Parameterization of given example motion clips is provided in Section 4. In Section 5, we describe our motion blending scheme in detail. We demonstrate experimental results in section 6. Finally, we conclude this paper in Section 7.

2 Related work

2.1 Locomotion generation

There have been an abundance of research results to generate human locomotion. An excellent survey of these efforts can be found in [20]. Generally, previous work of generating human locomotion can be classified into three categories: procedural generation, dynamics simulation, and motion capture-based approaches.

Procedural approaches are mainly based on kinematics. Bruderlin and Calvert [3, 4, 5] generated locomotion by simulating an inverted pendulum for a stance leg. Boulic and Thalmann [2] ex-

[*]e-mail: sipark@jupiter.kaist.ac.kr
[†]e-mail: joony@jupiter.kaist.ac.kr
[‡]e-mail: syshin@jupiter.kaist.ac.kr

ploited inverse kinematics to prevent invalid situations such as foot penetration into the ground. Recently, procedural locomotion generation has been available in commercial animation packages such as Boston Dynamics "BDI Guy", Credo Interactive "Life Forms Studio", 3D Studio Max "Character Studio", and Motion Factor "Motivate".

Dynamics simulation approaches focus on generating physically feasible locomotion. The basic idea of these approaches is to design an efficient controller which computes the actuator forces to generate a desired motion. Raibert and Hodgins [22] provided hand-designed controllers to produce physically realistic locomotion for a simple biped robot. Hodgins *et at.* [12] extended those controllers to more complex human models. Ko and Badler [15] incorporate dynamic balance control into kinematic locomotion generation. Lazro *et al.* [17] suggested limit cycle control to generate a balanced walking.

Motion capture-based approaches exploit example motions to generate a desired motion. Unuma *et al.* [28] applied the Fourier analysis to motion data for interpolating and extrapolating the human locomotion such as walking motion. Bruderlin and Williams [6] adapted the conventional signal processing techniques to modify an animated motion. Guo and Rovergé [10] and Willey and Hahn [29] provided an interpolation technique for example motions located regularly in the parameter space. Rose *et al.* [23] and Sloan *et al.* [26] presented scattered data interpolation techniques which are suitable for blending example motions located irregularly in the parameter space. Sun and Metaxas [27] presented a hybrid approach, coupling a procedural approach and a motion capture-based approach.

2.2 Unit quaternion interpolation

For interpolation of two unit quaternions, *slerp* (spherical linear interpolation) has been used [21, 25]. One popular approach for interpolation of multiple unit quaternions is to interpolate the components of quaternions individually and then to re-normalize the result for satisfying the unitariness condition [1]. However, this re-normalization is known to incur side effects such as singularity and unexpected distortion. To avoid such re-normalization, the exponential and logarithm mapping has been used [9, 11]. Johnstone and Williams [13] suggested a rational mapping between $\mathbb{S}^3$ and $\mathbb{R}^3$. Lee and Shin [19] proposed a general framework for constructing the time-domain filters for orientation data. Their scheme satisfies such important filter properties as coordinate-invariance, time-invariance, and symmetry. Buss and Fillmore [7] provided a method for computing the weighted spherical averages of sample points on d-dimensional sphere based on least squares minimization.

2.3 Motion retargeting

Motion retargeting is to adapt a motion created for one articulated figure to another figure with identical structure but different segment lengths [8]. Gleicher [8] gave an optimization technique using the spacetime formulation for motion retargeting. Lee and Shin [18] provided an interactive motion editing technique based on hierarchical curve fitting. They also presented a fast inverse kinematics solver adopting the notion of an elbow circle given by Korein and Badler [16]. Shin *et al.* [24] suggested an importance-based approach for on-line motion retargeting. They provided the notion of dynamic importance of an end-effector and introduced a fast, robust inverse kinematics solver to realize the important aspect of the end-effector according to its importance value.

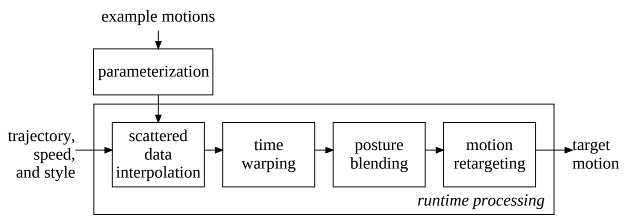

Figure 1: **Overall structure.**

3 Overview

Based on the framework of motion blending proposed by Rose *et al*. [23] and Sloan *et al*. [26], our approach for locomotion generation consists of two main parts: parameterization and motion blending as shown in Figure 1. In addition to presenting the overall structure of our approach, we show how we extend the previous approaches in each of the parts specifically for locomotion generation. This extension, we believe, is non-trivial and can be applicable to motion generation in general.

As preprocessing, we parameterize the example motions to place them in a parameter space. Rose *et al*. [23] introduced the notion of *verbs* and *adverbs* for this purpose. They classified the motions of the similar structure as a verb and placed them in the space parameterized by adverbs. In their original parameterization, different styles and speeds of locomotion gave rise to different verbs. For example, walking and running were different verbs. These verbs were connected by using a 'verb graph' for their seamless transition. However, this transition makes it hard to control the speed and style of locomotion in an on-line manner. Therefore, for on-line control, we combine those verbs into one. On top of the original parameters, we adopt three parameters explicitly: style, speed, and turning angle for better control of locomotion. As a result, the locomotion becomes a 'large' verb, which would have been represented by a verb graph, otherwise. Given an example motion, we automatically compute its quantitative parameters such as speed and turning angle. The other parameters are specified interactively by a user.

Provided with a vector of parameters, we blend the parameterized example motions to generate the corresponding motion at each frame. Based on a multidimensional scattered data interpolation, we divide motion blending into four steps: weight computation, timewarping, posture blending, and motion retargeting. For a given vector of parameters, we first compute the contribution (called weight) of each example motion to the target motion using cardinal basis functions [26]. Then, the features of each motion are aligned by timewarping. Based on *keytimes*, which are important instants of locomotion such as heel-strikes and toe-offs [10, 23], Rose *et al.* aligned the example motions using timewarping functions each of which maps the actual time of an example motion onto the *generic time* [23]. However, the weight may change dynamically from frame to frame as the parameter vector changes, and the range of speed over example motions may also be quite large. In this case, the original version may cause a blended actual time of the target motion to go reversely, that is, go back to the past, with respect to the generic time. To address this issue, we introduce an incremental approach for timewarping which blends the timewarping functions of the example motions incrementally for guaranteeing the monotonicity of the blended actual time as long as the weight values are non-negative.

Next, we blend the timewarped example motions. For their effective blending, we provide a novel scheme based on the result in [19]. Our scheme adopts unit quaternions rather than Euler an-

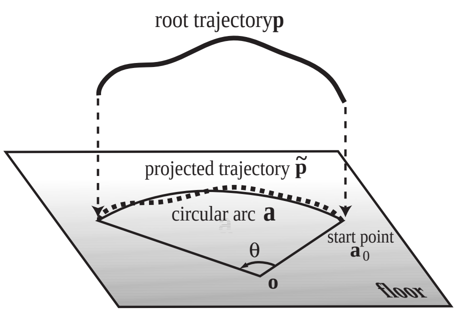
root trajectory $\mathbf{p}$
projected trajectory $\tilde{\mathbf{p}}$
circular arc $\mathbf{a}$
start point $\mathbf{a}_0$
θ
$\mathbf{o}$
floor

Figure 2: **Circular arc as an approximation of the root trajectory.**

gles to represent joint angles for a consistent parameterization over example motions. We finally apply the blended motion to a target character. When the size and proportions of the target character are different from those of the actual human performer, we cannot apply the blended posture directly to the target character, which may cause artifacts such as foot sliding or penetration. For avoiding those artifacts, we first compute the target stance foot position at each frame by blending those of the example motions and then employ an importance-based approach for motion retargeting [24], which adapt the motion to the target character and the environment in an on-line, real-time manner.

4 Locomotion Parameterization

For effective control over a variety of locomotion, we focus on three parameters: style, speed, and turning angle. The rest of parameters for an example motion are specified interactively as discussed in [23]. We have two types of locomotion, walking and running to which we assign zero and one, respectively, as their values of the style parameter. Parameters such as speed and turning angle will be computed from each example motion.

With some loss of generality, suppose that each example motion is short enough for its speed and turning angle to be invariant over its duration. For a lengthy motion with non-homogeneous speed and turning angle, we need some preprocessing to decompose it into short motions satisfying this assumption. Since a motion of constant speed and turning angle traces a circular trajectory, we approximate the root trajectory $\mathbf{p}$ of a motion as a circular arc $\mathbf{a}$ that best fits the projected trajectory $\tilde{\mathbf{p}}$ on the floor. Notice that a straight line is a circular arc of infinite radius.

As shown in figure 2, let the circular arc $\mathbf{a}$ subtend an angle θ starting from a point $\mathbf{a}_0$ on the circle of radius r centered at $\mathbf{o}$. We find the circular arc $\mathbf{a}$ by least-squares fitting which minimizes the distance between $\tilde{\mathbf{p}}$ and $\mathbf{a}$ as follows:

$$\text{mimimize} \quad \sum_{i=1}^{N_f} [\tilde{\mathbf{p}}_i - \mathbf{a}(\theta_i;\mathbf{o},\mathbf{a}_0,\theta)]^2 \quad \text{over} \quad \mathbf{o},\mathbf{a}_0,\theta, \tag{1}$$

where $\tilde{\mathbf{p}}_i$ is the point of the projected root trajectory at the ith frame, $\mathbf{a}(\theta_i)$ is the point on $\mathbf{a}$ after rotating $\mathbf{a}_0$ by the angle $(i \times \frac{\theta}{N_f-1})$ about $\mathbf{o}$, and N_f is the number of the frames. Let l and T be the length of the arc $\mathbf{a}$ and the duration of the motion, respectively. Then, the speed of the motion is $\frac{l}{T}$, and its turning angle is $\frac{\theta}{T}$.

5 Motion Blending

Given a vector of parameters at each frame, we first determine the weights of the example motions. We then perform timewarping to synchronize the example motions. Next, we compute the target motion by blending them with respect to their weights. Finally, given a trajectory on the floor, we adapt the blended posture to the target character and the environment to follow the trajectory through motion retargeting.

5.1 Weight Computation

We employ a multidimensional scattered data interpolation technique suggested by Sloan *et al.* [26]. Their approach is based on the interpolation scheme of Rose *et al.* [23] who used radial basis functions. Incorporating cardinal basis functions, Sloan *et al.* reformulated this scheme to provide a more efficient interpolation method. While the original version interpolates each degree of freedom at every frame, they computed the weights of the example motions for the given parameter vector and blended them with respect to these weights.

Given the vector $\mathbf{p}$ of parameters, the weight $w_i(\mathbf{p})$ of the ith example motion is defined as

$$w_i(\mathbf{p}) = \sum_{l=0}^{N_p} a_{il}A_l(\mathbf{p}) + \sum_{j=1}^{N_e} r_{ij}R_j(\mathbf{p}), \tag{2}$$

where $A_l(\mathbf{p})$ and a_{il} are the linear basis functions and their coefficients, $R_j(\mathbf{p})$ and r_{ij} are the radial basis functions and their coefficients, respectively. N_p and N_e are the number of the parameters and that of the examples. For interpolating the example motions exactly, the weight of the ith example motion is one at $\mathbf{p}_i$ and becomes zero at $\mathbf{p}_j$, $i \neq j$, that is, $w_i(\mathbf{p}_j) = 1$ for $i = j$ and $w_i(\mathbf{p}_j) = 0$ for $i \neq j$.

Ignoring the second term of Equation (2), we first solve for the linear coefficients a_{il} to fix the first term:

$$w_i(\mathbf{p}) = \sum_{l=0}^{N_p} a_{il}A_l(\mathbf{p}). \tag{3}$$

The linear bases are simply $A_l(\mathbf{p}) = \mathbf{p}_l$, which is the lth component of $\mathbf{p}$, and $A_o(\mathbf{p}) = 1$. We employ a least squares method to determine the unknown coefficients a_{il} of the linear bases using the parameter vector $\mathbf{p}_i$ of each example and its weight $w_i(\mathbf{p}_i)$.

We employ the radial basis functions to interpolate the residuals $\bar{w}_i(\mathbf{p}) = w_i(\mathbf{p}) - \sum_{l=0}^{N_p} a_{il}A_l(\mathbf{p})$ for all i. The radial basis function $R_j(\mathbf{p})$ is a function of the Eculidean distance between $\mathbf{p}$ and $\mathbf{p}_j$ in the parameter space:

$$R_j(\mathbf{p}) = B\left(\frac{\|\mathbf{p}-\mathbf{p}_j\|}{\alpha}\right), \quad \text{for } 1 \leq j \leq N_e, \tag{4}$$

where $B(\cdot)$ is the cubic B-spline function, and α is the dilation factor, which is the minimum separation to the nearest other example in the parameter space. The coefficients r_{ij} are calculated by solving the linear system:

$$\mathbf{r}\mathbf{R} = \bar{\mathbf{w}}, \tag{5}$$

where $\mathbf{r}$ is an $N_e \times N_e$ matrix of the unknown coefficients r_{ij}, and $\mathbf{R}$ and $\bar{\mathbf{w}}$ are the matrices of the same size defined by the radial basis functions and the residuals, respectively, such that $\mathbf{R}_{ij} = R_i(\mathbf{p}_j)$ and $\bar{\mathbf{w}}_{ij} = \bar{w}_i(\mathbf{p}_j)$. Provided with the solutions for a_{il} and r_{ij}, we obtain the weight of each example motion from Equation (2).

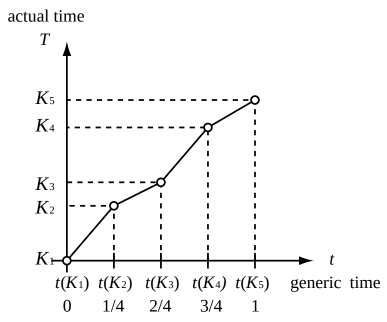

Figure 3: **The piecewise linear mapping between the actual time T and the generic time t.**

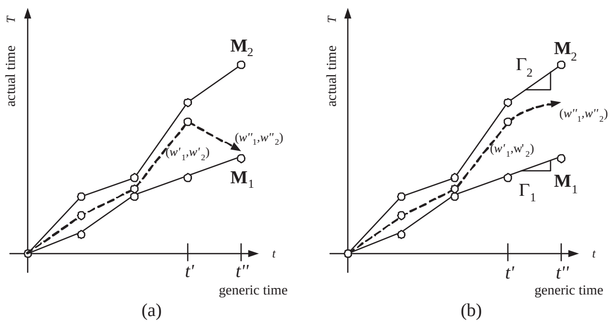

Figure 4: **(a) Timewarping of Rose *et al.*; (b) Our incremental timewarping.**

5.2 Incremental Timewarping

To blend the example motions of various speed, we align the example motions by employing a notion of keytimes which are important instances of a motion such as moments of heel-strikes and toe-offs [10, 23]. All example motions consist of the same sequence of keytime phases, that is, the example motions start with the same foot and take the same number of steps. We denote an actual keytime by $K_i, 1 \leq i \leq N_k$, where N_k is the number of the keytimes.

Based on the keytimes, timewarping is defined as a piecewise linear mapping of actual time $T \in [K_1, K_{N_k}]$ onto generic time $t \in [0,1]$. Given an actual time T, the corresponding generic time $t(T)$ is

$$t(T) = \left((m-1) + \frac{T - K_m}{K_{m+1} - K_m} \right) \frac{1}{N_k - 1} \tag{6}$$

for the largest m such that $T > K_m$. Figure 3 shows an example of this timewarping with 5 keytimes.

Once all example motions have reparameterized with the generic time, we can compute the actual time of a blended motion at a given generic time. Sloan *et al.* [26] computed the actual time of the target motion by weighted summing the actual times of the example motions at a given generic time, that is,

$$T(t) = \sum_{i=1}^{N_e} w_i T_i(t), \tag{7}$$

where w_i and $T_i(t)$ are the weight value and the actual time of the ith example at the generic time t, respectively. This approach works well when the weight of each example motion is fixed during the whole cycle of the motion. In general, the weight may change dynamically from frame to frame according to the parameter vector, and the speed variation over example motions may also be quite large. In this case, a blended actual time may go reversely, that is, go back to the past, with respect to the generic time. For example, see the generic time interval $[t', t'']$ as given in Figure 4(a).

To blend those motions of various speed with time-varying weights, we propose an incremental timewarping technique. The basic idea is to blend the change rates of the actual times with respect to the generic time rather than the actual times themselves and to accumulate the weighted change rate for incrementally updating the current actual time of the blended motion. With this incremental approach, we can guarantee the monotonicity of the blended timewarping function as long as the weight values are nonnegative. From Equation (6), we define the untimewaping function, which maps the generic time t onto the actual time T:

$$T(t) = K_m + \{(N_k - 1)t - (m-1)\}(K_{m+1} - K_m). \tag{8}$$

The change rate of the actual time is the same as the first derivative of the untimewarping function. Thus, given the generic time t and its change $\triangle t$, we compute the change $\triangle T$ of actual time as follows:

$$\triangle T = \left(\sum_{i=1}^{N_e} w_i(t) \cdot \Gamma_i(t) \right) \cdot \triangle t, \tag{9}$$

where $w_i(t)$ and $\Gamma_i(t)$ are the weight and the derivative value of the untimewarping function of the ith motion at time t, respectively.

Figure 4(b) illustrates our incremental timewarping. For two motions $\mathbf{M}_1$ and $\mathbf{M}_2$, let the derivatives of untimewarping functions at the generic time t' be Γ_1 and Γ_2, respectively. Given the actual time T' of the blended motion at the generic time t', the actual time at the generic time $t' + \triangle t$ is

$$T' + \triangle T = T' + (w_1(t')\Gamma_1 + w_2(t')\Gamma_2)\triangle t, \tag{10}$$

where $w_1(t')$ and $w_2(t')$ are the weights for example motions at the generic time t'.

5.3 Posture Blending

We generate the target posture at a given generic time by blending the corresponding postures of example motions at the same generic time. The posture of an articulated body is defined by the position of the root segment, its orientation, and the joint angles. We blend not only the joint angles but also the root positions and orientations of the example postures to obtain the posture of the target character, which will be adapted to a given trajectory.

We take a simple weighted sum to blend the root positions. However, due to the non-linearity of the orientation space, this scheme can not be applied directly to blending orientation data such as root orientations and joint angles. With Euler angles, it is non-trivial to ensure that similar poses use similar Euler angles. We provide a new orientation blending scheme to address this problem, based on orientation filtering [19]. Our scheme uses unit quaternions to represent orientations.

We begin joint angle blending with computing the reference orientation of each joint at each generic time step. Due to antipodal equivalence, it is well-known that an orientation has two equivalent quaternion representations. To ensure that similar poses have similar representations, we need to place all quaternion representations for the example poses in the hemisphere on the unit quaternion space $\mathbb{S}^3$ defined by the reference orientation.

The basic idea for blending orientations is to transform the orientation data into their analogues in a vector space with respect to a reference orientation, to compute their weighted sum, and then to

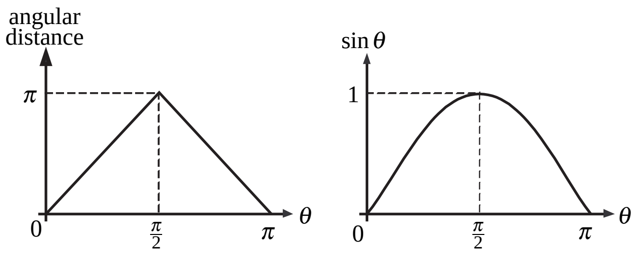

Figure 5: **Distance metrics: (a) angular distance; (b) proposed distance metric.**

transform the result back to the orientation space. The main connection between unit quaternions and vectors is the logarithm and exponential maps. By the logarithm map, a unit quaternion $\mathbf{q}$ is transformed into its corresponding displacement vector $\mathbf{v}$ with respect to a reference orientation $\mathbf{q}_*$, that is,

$$\mathbf{v} = \log(\mathbf{q}_*^{-1}\mathbf{q}). \tag{11}$$

Here, if $\mathbf{q}$ lies outside the hemisphere defined by $\mathbf{q}_*$, that is, $\|\log(\mathbf{q}_*^{-1}\mathbf{q})\| > \pi/2$, then we use $-\mathbf{q}$ rather than $\mathbf{q}$ to ensure its consistent representation. This also guarantees that a posture has its unique representation. The original orientation (or its antipodal equivalence) is recovered by the exponential map:

$$\mathbf{q} = \mathbf{q}_* \exp(\mathbf{v}). \tag{12}$$

Geometrically, these logarithm and exponentiation give mappings between the tangent space $T_{\mathbf{q}_*}\ \mathbb{S}^3 \equiv \mathbb{R}^3$ at the reference unit quaternion $\mathbf{q}_*$ and the unit quaternion space $\mathbb{S}^3$ [14, 19]. The logarithm map of a unit quaternion is not well defined at $-\mathbf{I} = (-1,0,0,0)$, and the exponential and logarithm maps provide a natural, non-singular parameterization for "small" angular displacements [19]. Thus, before mapping orientation data into their vector displacements, we choose the reference orientation that is as "close" to all example unit quaternions as possible.

To choose the reference orientation, we first define the distance metric between two quaternions. Buss and Fillmore [7] used the geodesic norm $\theta = \|\log(\mathbf{q}_1^{-1}\mathbf{q}_2)\|$ as their distance metric to find the spherical average of points on a d-dimensional sphere $\mathbb{S}^d$. However, since they were interested in averaging points on $\mathbb{S}^d$, they did not address the problem caused by the antipodal equivalence property of the unit quaternion space. For an antipodal pair, $\mathbf{q}$ and $-\mathbf{q}$ in $\mathbb{S}^3$, their distance measure is $\|\log\left((-\mathbf{q})^{-1}\mathbf{q}\right)\|$ which is π, even though they represent an identical orientation. We suggest a new distance metric to avoid such a problem. This metric also leads to an efficient algorithm for finding the reference orientation.

We begin with the angular distance between two quaternions to address the antipodal equivalence problem. Since $\mathbf{q}$ and $-\mathbf{q}$ represent the same orientation, the angular distance between two quaternions, $\mathbf{q}_1$ and $\mathbf{q}_2$ is

$$\mathrm{dist}(\mathbf{q}_1,\mathbf{q}_2) = \min\left(\|\log\left(\mathbf{q}_1^{-1}\mathbf{q}_2\right)\|, \|\log\left(\mathbf{q}_1^{-1}(-\mathbf{q}_2)\right)\|\right). \tag{13}$$

Therefore, the reference quaternion $\mathbf{q}_*$ is obtained by minimizing the sum of squared distances:

$$E = \sum_{i=1}^{N_e} \|\mathrm{dist}(\mathbf{q}_*,\mathbf{q}_i)\|^2. \tag{14}$$

However, this distance metric is not differentiable at $\theta = \frac{\pi}{2}$ as shown in Figure 5(a). Thus, we introduce an alternative distance metric based on a sinusoidal function:

$$\mathrm{dist}(\mathbf{q}_1,\mathbf{q}_2) = \sin(\|\log(\mathbf{q}_1^{-1}\mathbf{q}_2)\|). \tag{15}$$

This metric not only approximates the angular distance well but also is differentiable at any point in $[0,\pi)$ as illustrated in Figure 5(b).

With the new distance metric, the reference orientation $\mathbf{q}_*$ can be found by minimizing the objective function:

$$E = \sum_{i=1}^{N_e} \sin^2\theta_i, \tag{16}$$

where $\theta_i = \|\log(\mathbf{q}_*^{-1}\mathbf{q}_i)\|$. Since $\sin^2\theta_i = (1-\cos^2\theta_i)$ and $\cos\theta_i = \mathbf{q}_i^T\cdot\mathbf{q}_*$, we have

$$E = \sum_{i=1}^{N_e}\left(1-(\mathbf{q}_i^T\cdot\mathbf{q}_*)^2\right). \tag{17}$$

To find $\mathbf{q}_*$ that minimizes E subject to the unitariness constraint of a quaternion, we employ the lagrangian multiplier method:

$$\frac{\partial E}{\partial \mathbf{q}_*} = \lambda\frac{\partial C}{\partial \mathbf{q}_*}, \tag{18}$$

where $C = 1 - \|\mathbf{q}_*\|^2$ and λ is the lagrangian multiplier. Plugging Equation (17) into Equation (18), we have

$$\left(\sum_{i=1}^{N_e}\mathbf{q}_i\cdot\mathbf{q}_i^T\right)\mathbf{q}_* = \lambda\mathbf{q}_* \quad \text{or} \quad \mathbf{A}\mathbf{q}_* = \lambda\mathbf{q}_*, \tag{19}$$

where $\mathbf{q}_*$ is represented as a 4×1 vector, $\mathbf{A}$ is a 4×4 matrix, and λ is a real number. The problem of finding $\mathbf{q}_*$ in Equation (19) is a typical eigenvector problem. Since a 4×4 matrix has a maximum of four eigenvectors, we examine of those solutions to choose the best one which minimizes the objective function E as given in Equation (17).

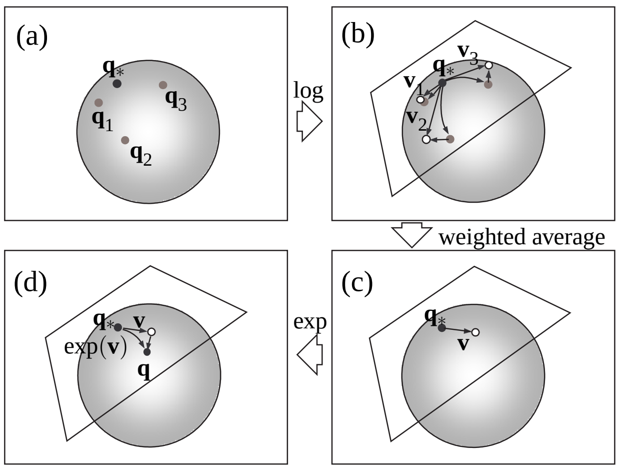

Figure 6: **Unit quaternion blending.**

Figure 6 illustrates our unit quaternion blending scheme. Given the reference orientation $\mathbf{q}_*$ (Figure 6(a)), we transform each example orientation $\mathbf{q}_i$ into its corresponding displacement vector $\mathbf{v}_i$ through the logarithm map, that is, $\mathbf{v}_i = \log(\mathbf{q}_*^{-1}\mathbf{q}_i)$ (Figure 6(b)). Then, we blend $\mathbf{v}_i$, $1 \le i \le N_e$ with respect to their weights to obtain the displacement vector $\mathbf{v} = \sum_{i=1}^{N_e} w_i\mathbf{v}_i$ (Figure 6(c)). Finally,

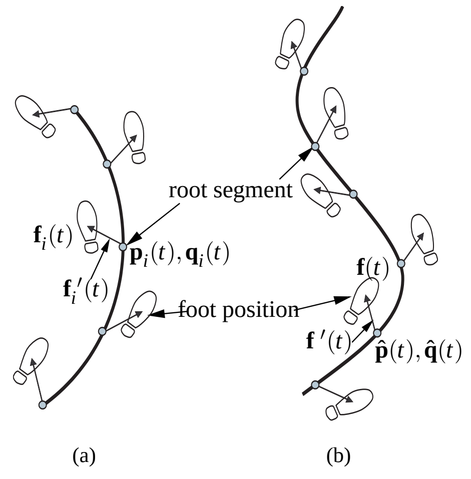

Figure 7: **Foot positions in the global and the local coordinate frames: (a) example motion; (b) target motion.**

we compute the blended orientation $\mathbf{q}$ by transforming $\mathbf{v}$ back to the orientation space and apply it to $\mathbf{q}_*$, that is, $\mathbf{q} = \mathbf{q}_* \exp(\mathbf{v})$ (Figure 6(d)).

Now, we adapt the blended root position and orientation to a given trajectory. We first adjust the root position. Let $\mathbf{p}^t(t)$ and $\mathbf{d}^t(t)$ be the curved trajectory provided by a user and the tangent vector of the trajectory at time t. Here, we assume that $\mathbf{p}^t(t)$ is a planar curve given on the floor. We compute the target root position $\hat{\mathbf{p}}(t)$ so that its projection on the floor is coincident with $\mathbf{p}^t(t)$ while preserving its elevation:

$$\hat{\mathbf{p}}(t) = \mathbf{p}(t) + (\mathbf{p}^t(t) - \tilde{\mathbf{p}}(t)), \tag{20}$$

where $\mathbf{p}(t)$ is the blended root position and $\tilde{\mathbf{p}}(t)$ is its projection on the floor. We then adjust the blended orientation of the root. The direction $\mathbf{d}(t)$ for the character to move forward is obtained by blending the tangent vectors of the arcs that approximate the projected root trajectories of the example motions. We determine the target root orientation $\hat{\mathbf{q}}(t)$ such that the forward direction $\mathbf{d}(t)$ coincides with $\mathbf{d}^t(t)$. Let ϕ be the angular distance between $\mathbf{d}(t)$ and $\mathbf{d}^t(t)$. The unit quaternion to rotate by ϕ about the unit normal vector $\mathbf{n}$ of the floor is $e^{\mathbf{n}\phi/2}$. Therefore, the target root orientation $\hat{\mathbf{q}}(t)$ is

$$\hat{\mathbf{q}}(t) = e^{\mathbf{n}\phi/2}\mathbf{q}(t), \tag{21}$$

where $\mathbf{q}(t)$ is the blended root orientation.

5.4 Motion Retargeting

When the target character has a different size and proportion from the actual human performer, we are not able to simultaneously preserve both the joint angles of the blended posture and its end-effector positions. Therefore, there may be artifacts such as foot sliding and penetration. To obtain a convincing locomotion of the target character, we need to adjust the blended posture in an on-line, real-time manner while keeping the characteristics of the original motion. For this purpose, we employ an importance-based approach for on-line motion retargeting introduced by Shin *et al.* [24].

To employ their approach, we have to provide the target stance foot position as input data at each frame. We obtain it by blending the foot positions of the example motions. To do that, we first represent each of them in the local coordinate frame of its root segment. For the *i*th example motion, the foot position $\mathbf{f}_i'(t)$ at time t in the local coordinate frame is

$$\mathbf{f}_i'(t) = \mathbf{q}_i^{-1}(t)\left(\mathbf{f}_i(t) - \mathbf{p}_i(t)\right)\mathbf{q}_i(t), \tag{22}$$

where $\mathbf{f}_i(t)$ is the foot position of the ith example motion at time t measured in the global coordinate frame, and $\mathbf{p}_i(t)$ and $\mathbf{q}_i(t)$ are the position of the root segment and its orientation, respectively. The target foot position $\mathbf{f}'(t)$ is obtained by blending $\mathbf{f}_i'(t)$, $1 \le i \le N_e$ with respect to their weights w_i:

$$\mathbf{f}'(t) = \sum_i^{N_e} w_i \mathbf{f}_i'(t). \tag{23}$$

Given the target root position $\hat{\mathbf{p}}(t)$ and orientation $\hat{\mathbf{q}}(t)$ which define the local coordinate frame of the root segment, the target foot position $\mathbf{f}(t)$ in the global coordinate frame is

$$\mathbf{f}(t) = \hat{\mathbf{q}}(t)\ \mathbf{f}'(t)\ \hat{\mathbf{q}}^{-1}(t) + \hat{\mathbf{p}}(t). \tag{24}$$

A target foot position may vary from time to time in accordance with its weight change. Therefore, we force the target foot position to be fixed while the foot contacts the floor. The duration of contact can be easily obtained from the keytimes. When the foot is approaching or contacting the floor, we change the joint angles to keep the foot position $\mathbf{f}(t)$. Otherwise, we keep the joint angles to preserve the motion characteristics inherited from the example motions.

6 Experimental Results

For our experiments, we used twenty example motion clips of human locomotion: ten walking motions, five slow running motions, and five fast running motions. They are different from each others in speed and turning angle. We computed the speed and turning angle of each example as described in Section 4. Every motion contains two steps and starts with the right step. We interactively tagged the keytimes at the instances of heel-strikes and toe-offs. We use a human model of 43 degrees of freedom: 6 DOF's for the root position and orientation, 9 DOF's for the body joints, and 28 DOF's for the limbs.

Our first experiment is to generate the locomotion that follows a given curved trajectory of specific style and speed. Figures 9(a) through 9(c) exhibit the resulting motions with the same curved trajectory, which is represented as a B-spline curve. We generated the locomotion that follows the trajectory by exploiting the turning angle parameter obtained from the tangent vector of the trajectory at each frame. As shown in the figures, the resulting motions are also effectively controlled in style and speed by the corresponding parameters.

In next experiment, we demonstrate the on-line, real-time capability of our motion generation scheme. We took a sequence of mouse pointer positions as on-line input data and made the target character chase the pointer. Based on the sampled pointer positions at each frame, we computed the speed and turning angle parameters. The style of the motion was selected in accordance with the speed of the mouse pointer. As shown in Figure 10, we used running motions when the character moves fast; otherwise, we used walking motions.

For the final experiment, we generated locomotion on a terrain. The terrain was represented by a NURBS surface of which control points were placed on a regular grid, and their heights were randomly perturbed. To adapt a blended motion to the terrain, we adjusted the height of a target foot position so that it lies on the ground. Thus, the resulting motion did not suffer from artifacts such as foot sliding and penetration even on the terrain as shown in Figure 11.

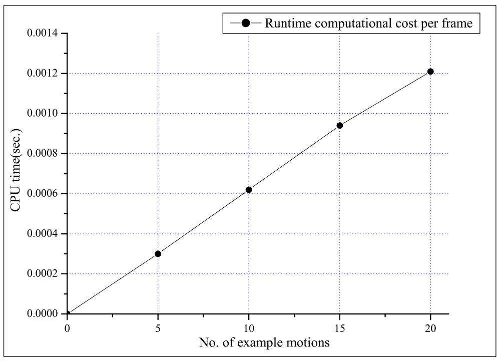

Figure 8: **Computation times of our algorithm obtained by varying the number of example motions**

Figure 8 gives the overall performance of our approach excluding rendering time. The experiments were performed on an IBM compatible PC with Pentium III 800MHz CPU and 512 Mb memory. The timing data were obtained by varying the number of example motions. The figure shows that the computation time is linearly proportional to the number of example motions. With twenty example motions, it took 1.2 milliseconds per frame to produce a target motion. Thus, we are able to generate more than 800 frames per second.

7 Conclusion

In this paper, we describe a novel approach to produce convincing locomotion of human-like characters in real time. Our goal is on-the-fly generation of locomotion that satisfies time-varying constraints on its trajectory, speed, and style given in an on-line manner. The basic idea of this paper is blending a set of pre-recorded example motions to achieve the desired motion based on the motion blending techniques suggested by Rose *et al.* [23] and Sloan *et al.* [26]. We extend their approaches specifically for locomotion generation in several ways: In order to blend the example motions of a wide range of speed, we provide an incremental approach for timewarping, while addressing the artifacts caused by the previous approaches. We present a novel scheme for blending joint angles based on quaternion algebra to guarantee their consistent representations. To apply the blended motion to characters of various size and proportion, we compute the target positions of feet by blending those of the example motions and then adapt the motion to the target character and the environment in an on-line manner. The experimental results demonstrate that the proposed approach can produce convincing motions in real time. We believe that our approach can be extended to be applicable to motion generation in general.

Acknowledgement

This work was supported by the NRL (National Research Laboratory) project of KISTEP (Korea Institute of Science & Technology Evaluation and Planning).

References

[1] R. Azuma and G. Bishop. Improving static and dynamic registration in an optical see-through hmd. *Computer Graphics (Proceedings of SIGGRAPH 94)*, pages 197–204, July 1994.

[2] R. Boulic and D. Thalmann. Combined direct and inverse kinematics control for articulated figures motion editing. *Computer Graphics Forum*, 11(4):189–201, 1992.

[3] A. Bruderlin and T. Calvert. Goal-directed dynamic animation of human walking. *Computer Graphics (Proceedings of SIGGRAPH 89)*, pages 233–242, 1989.

[4] A. Bruderlin and T. Calvert. Interactive animation of perseonalized human locomotion. In *Proceedings of Graphics Interface 93*, pages 17–23, 1993.

[5] A. Bruderlin and T. Calvert. Knowledge-driven, interactive animation of human running. In *Proceedings of Graphics Interface 96*, pages 213–221, 1996.

[6] A. Bruderlin and L. Williams. Motion signal processing. *Computer Graphics (Proceedings of SIGGRAPH 95)*, pages 97–104, 1995.

[7] S. R. Buss and J. P. Fillmore. Spherical averages and applicataions to spherical splines and interpolation. *ACM Transactions On Graphics*, 2001. To appear.

[8] M. Gleicher. Retargeting motion to new characters. *Computer Graphics (Proceedings of SIGGRAPH 98)*, pages 33–42, 1998.

[9] S. Grassia. Practical parameterization of rorations using the exponential map. *The Journal of Graphics Tools*, 3(3):29–48, 1998.

[10] S. Guo and J. Robergé. A high-level control mechanism for human locomotion based on parametric frame space interpolation. In *Proceedings of Eurographics Workshop on Computer Animation and Simulation 96*, pages 95–107, Aug. 1996.

[11] G. Hanotaux and B. Peroche. Interactive control of interpolations for animation and modeling. In *Proceedings of Graphics Interface '93*, pages 201–208, 1993.

[12] J. Hodgins, W. Wooten, D. Brogan, and J. O'Brien. Animating human athletics. *Computer Graphics (Proceedings of SIGGRAPH 95)*, pages 71–78, 1995.

[13] J. Johnstone and J. Williams. Rational control of orientation for animation. In *Proceedings of Graphics Interface '93*, pages 179–186, 1995.

[14] M. J. Kim, M. S. Kim, and S. Y. Shin. A general construction scheme for unit quaternion curves with simple high order derivatives. *Computer Graphics (Proceedings of SIGGRAPH 95)*, pages 369–376, 1995.

[15] H. Ko and N. I. Badler. Animating human locomotion with inverse dynamics. *IEEE Computer Graphics and Applications*, 16(2):50–59, 1996.

[16] J. U. Korein and N. I. Badler. Techniques for generating the goal-directed motion of articulated structures. *IEEE Computer Graphics and Applications*, pages 71–81, 1982.

[17] J. Laszlo, M. van de Panne, and E. Fiume. Limit cycle control and int application to the animation of balancing and walking. *Computer Graphics (Proceedings of SIGGRAPH 96)*, pages 155–162, 1996.

[18] J. Lee and S. Y. Shin. A hierarchical approach to interactive motion editing for human-like figures. *Computer Graphics (Proceedings of SIGGRAPH 99)*, pages 39–48, 1999.

[19] J. Lee and S. Y. Shin. General construction of time-domain filters for orientation data. *IEEE Transactions on Visualization and Computer Graphics*, 2001. To appear.

[20] M. Multon, L. France, M-P. Cani-Gascuel, and D. Debunne. Computer animation of human walking: a survey. *Journal of Visualization and Computer Animation*, 10:39–54, 1999.

[21] D. Pletinckx. Quaternion calculus as a basic tool in computer graphics. *The Visual Computer*, 5:2–13, 1989.

[22] M. Raibert and J. Hodgins. Animation of dynamic legged locomotion. *Computer Graphics (Proceedings of SIGGRAPH 91)*, pages 349–358, 1991.

[23] C. Rose, M. F. Cohen, and B. Bodenheimer. Verbs and adverbs: Mulidimensional motion interpolation. *IEEE Computer Graphics and Applications*, 18(5):32–40, Sept. 1998.

[24] H. J. Shin, J. Lee, M. Gleicher, and S.Y. Shin. Computer puppetry: An importance-based approach. *ACM Transactions On Graphics*, 20(2):67–94, Apr. 2001.

[25] K. Shoemake. Animating rotation with quaternion vurves. *Computer Graphics (Proceedings of SIGGRAPH 85)*, pages 245–254, 1985.

[26] P. Sloan, C. F. Rose, and M. F. Cohen. Shape by example. In *Proceedings of 2001 ACM Symposium on Interactive 3D Graphics*, pages 135–144, 2001.

[27] H. C. Sun and D. M. Metaxas. Automating gait generation. *Computer Graphics (Proceedings of SIGGRAPH 2001)*, pages 261–270, 2001.

[28] M. Unuma, K. Anjyo, and T. Tekeuchi. Fourier principles for emotion-based human figure animation. *Computer Graphics (Proceedings of SIGGRAPH 95)*, pages 91–96, 1995.

[29] D. J. Wiley and J. K. Hahn. Interpolation synthesis for articulated fiture motion. *IEEE Computer Graphis and Applications*, 17(6):39–45, 1997.

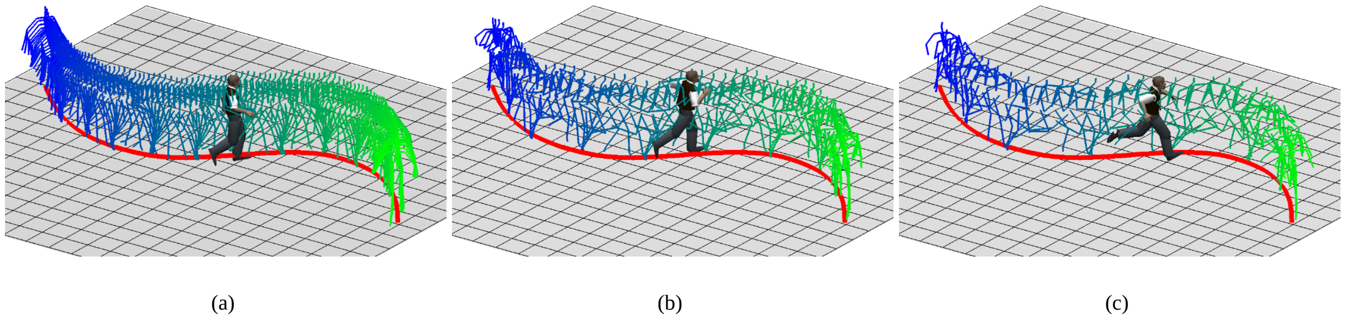

Figure 9: **Motion following the trajectory: (a) walking; (b) slow running; (c) fast running.**

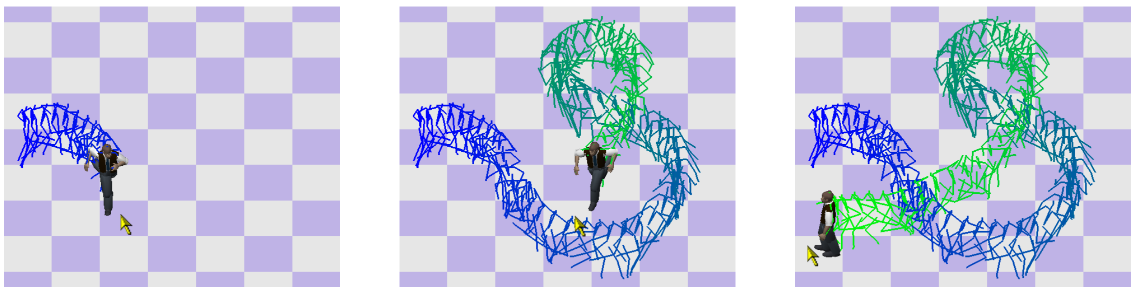

Figure 10: **Locomotion chasing the mouse pointer.**

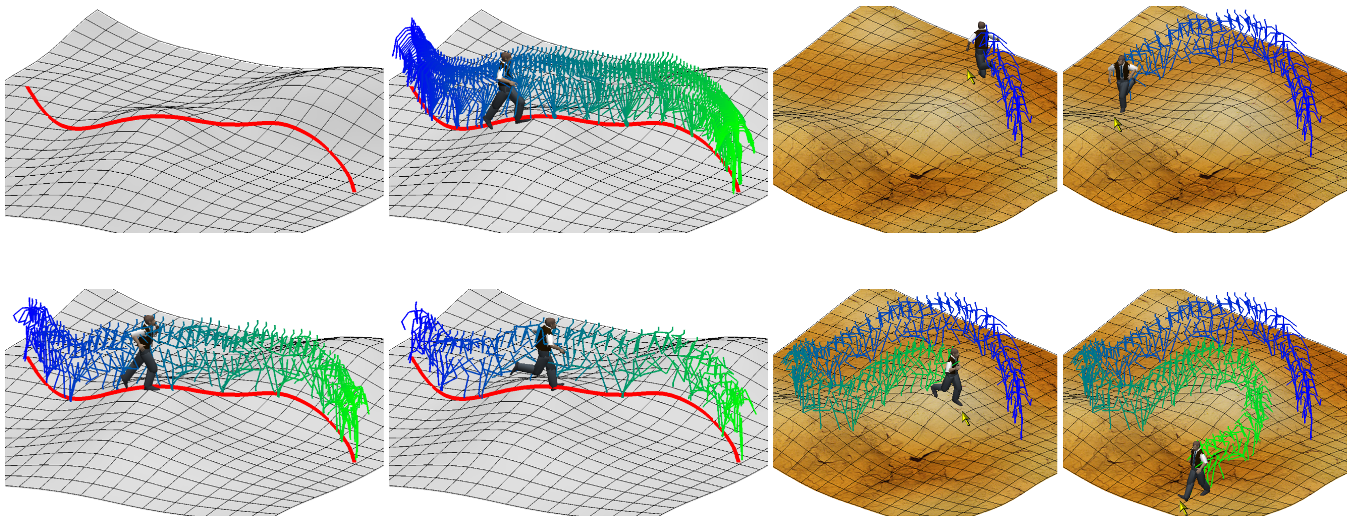

Figure 11: **Locomotion on a terrain.**