RESEARCH ARTICLE

Combining path planners and motion graphs

B. J. H. van Basten*, A. Egges and R. Geraerts

Center for Advanced Gaming and Simulation, Utrecht University, PO Box 80.089, 3508 TB Utrecht, The Netherlands

ABSTRACT

Natural locomotion of virtual characters is very important in games and simulations. The naturalness of the total motion strongly depends on both the path the character chooses and the animation of the walking character. Therefore, much work has been done on path planning and generating walking animations. However, the combination of both fields has received less attention. Combining path planning and motion synthesis introduces several problems. In this paper, we will identify two problems and propose possible solutions. The first problem is selecting an appropriate distance metric for locomotion synthesis. When concatenating clips of locomotion, a distance metric is required to detect good transition points. We have evaluated three common distance metrics both quantitatively (in terms of footskating, path deviation and online running time) and qualitatively (user study). Based on our observations, we propose a set of guidelines when using these metrics in a motion synthesizer. The second problem is the fact that there is no single point on the body that can follow the path generated by the path planner without causing unnatural animations. This raises the question how the character should follow the path. We will show that enforcing the pelvis to follow the path will lead to unnatural animations and that our proposed solution, which uses path abstractions, generates significantly better animations. Copyright © 2011 John Wiley & Sons, Ltd.

KEYWORDS
computer animation; distance metrics; motion capture; motion synthesis; path planning

***Correspondence**
B.J.H. van Basten, Center for Advanced Gaming and Simulation, Utrecht University, PO Box 80.089, 3508 TB Utrecht, The Netherlands.
E-mail: basten@cs.uu.nl

1. INTRODUCTION

Natural movement of characters is crucial in games and simulations. In many virtual environments characters need to walk from a point A to a point B. The quality of this movement highly depends on both the path that the character chooses and the walking animation itself. If a character collides with the environment or the animation suffers from foot skating the motion of the character will not be perceived as realistic.

Basically, this problem can be considered a *motion planning* problem. The posture of an articulated character with n rotational degrees of freedom can be described by an $n+3$ dimensional vector $q = (t_x, t_y, t_z, \theta_1, \ldots, \theta_n)^T$ where $t = (t_x, t_y, t_z)^T$ is the global root translation. We will define the $(n+3)$-dimensional space as the *configuration space* $\mathcal{C}$ of the character. Note that one pose of the character corresponds to a point in this space. The problem can then be defined as follows: given a character A (represented as a point) moving in a configuration space $\mathcal{C}$ amidst a collection of fixed rigid obstacles $\mathcal{B}$, and a start configuration s and goal configuration g for A, find a continuous path $\mathcal{P}$ from s to g avoiding contact with $\mathcal{B}$. An important additional constraint is that the resulting animation also needs to be perceived as natural.

Planning such whole-body motions in configuration space is very difficult. In general, humanoid articulated characters have many degrees of freedom and therefore the configuration space is very high-dimensional. Next to that, these dimensions can also be highly dependent and the dynamics of human motion are hard to encapsulate in constraints and optimization functions.

Therefore, in practice the problem is often simplified by separating the path planning and animation. First, a path planner generates a smooth, collision-free path $\mathcal{P}_{\text{plan}}$. Then the animation system generates a walking animation that drives the character to follow the path $\mathcal{P}_{\text{plan}}$ as closely as possible. Figure 1 shows an example of such a two-step process. The Corridor Map Method [1] is used as path planner and a motion graph approach is used as motion synthesizer [2].

Combining a path planner with an animation system seems a trivial task. There are, however, some major issues that none of the existing methods address. In this paper, we will give an overview of various (combinations of) path planning and animation systems and we will address two problems that arise when choosing a decoupled approach.

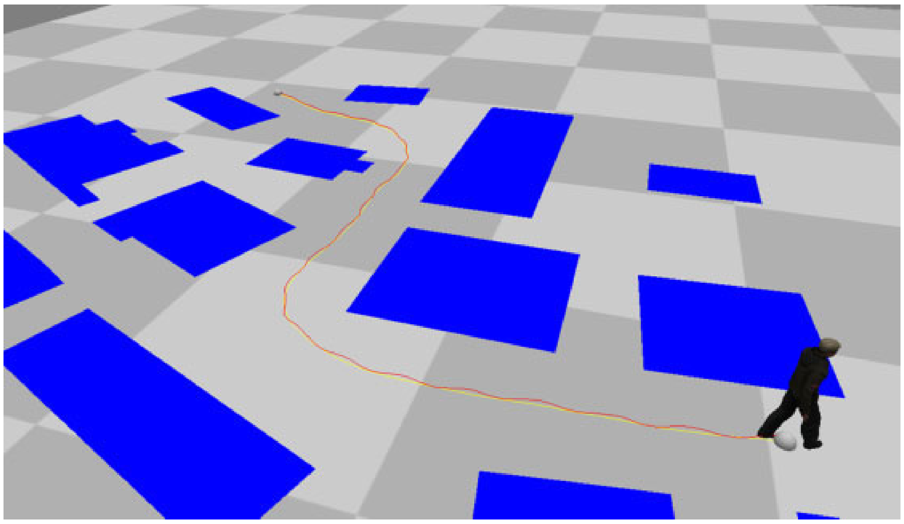


Figure 1. Motion synthesis techniques such as a motion graph can generate locomotion along a smooth obstacle-free path.

Our provided solutions can easily be incorporated in a motion synthesizer. Since the integration can be done in the preprocessing phase, it will not affect the performance of the motion synthesizer. Note that the results presented in this paper are not only useful for motion graph based animation systems but for any motion synthesizer that depends on a path that a character needs to follow.

1.1. Choosing Posture Distance Metrics

In general, when concatenating two animations it is necessary that the end postures of the first animation resembles the beginning postures of the second animation. Typically, posture distance metrics are used to automatically determine these resembling clips of motion. However, no research has been done on the effect of the distance metric on the generated animation in terms of foot skating, path deviation and online running time. Foot skating and path deviation are crucial in the domain of locomotion, the running time is important for real-time applications. The experiments we have carried out will provide insight into the effect of using a given distance metric to generate locomotion along a path by using a motion synthesizer. Based on our evaluations, we propose a set of guidelines for using distance metrics in a motion synthesizer. These guidelines can then be used construct motion synthesis systems that are better adapted to the needs of animators and researchers.

1.2. Preventing Unnatural Pelvis Oscillation

The path generated by a path planning algorithm is a parametric curve that represents a simplification of the path the character needs to follow. Generally it is not clear which (body) part of the character should follow this path. In many systems the path is interpreted as the desired trajectory of the projection of the pelvis on the plane. However, when one takes a closer look at the trajectory of the pelvis during locomotion it appears that the pelvis oscillates during such a motion. When the motion synthesizer forces the pelvis to closely follow the desired path one loses this natural oscillation or *wiggle*, resulting in an unnatural motion. We provide a solution to the loss of oscillation by using path abstractions instead of the pelvis trajectory. This will lead to more natural animations while still being able to enforce path constraints.

2. RELATED WORK

In this section we will give an overview of related work in both path planning as well as computer animation. The section concludes with an overview of work that combines both fields.

2.1. Path Planning

Introduced in 1968, the A*-algorithm is one of the first planners [3]. This planner is still popular because of its simplicity and ability to find the shortest path in a grid of free cells covering the environment. Nevertheless, resulting paths tend to have little clearance to the obstacles and can be aesthetically unpleasant, so care must be taken to smooth them.

Like the A*-algorithm, the Potential-Field method [4] operates on a grid. Nevertheless, smooth paths can be produced by following the direction of the steepest descent of the potential toward the goal. While it is possible to compute a potential without local minima [5], the computation is rather expensive, and, hence, may compromise the real-time performance.

Roadmap-based methods, such as Visibility graphs [6], Rapidly-exploring Random Trees [7] and Probabilistic Roadmap Methods (PRM) [8,9], do not have local minima and usually ensure that a path can be found if one exists. These methods build a roadmap graph which represents the free space in the environment. Because this graph can be constructed off-line, real-time performance can be achieved when a path is extracted from this graph. In addition, their strength is that they can be applied to problems with many degrees of freedom. Nevertheless, they lack flexibility because they output a fixed path extracted from a one-dimensional graph. In addition, the paths are jerky. While optimization algorithms exist, they are still too slow for real-time performance [10].

Recently, the concept of path planning inside corridors has been introduced [1,11–13]. Such a corridor is defined as a sequence of empty disks. Because the union of these disks is two-dimensional, corridors facilitate creating collision-free smooth paths with a certain amount of minimum clearance to the obstacles. Such a path minimizes the chance that the animated character collides with obstacles in the environment.

When a global path has been derived, the character locally has to manoeuvre around dynamic obstacles and other characters. Often a local planner (for example based on forces [14] or geometric techniques [15]) is needed to *steer* the character along this global path. Singh *et al.* [16] developed a framework to evaluate such steering techniques.

2.2. Animation

Generating animations of human walking has received a lot of attention during the past decades. There are several classes of techniques, each having its own advantages and disadvantages (see Welbergen *et al.* [17] for an overview of different animation techniques). An excellent survey on generating locomotion has been written by Multon *et al.* [18]. The techniques can be classified into three classes. *Procedural* techniques generate locomotion from scratch by using algorithms based on empirical and biomechanical concepts. These techniques offer a high-level control, yet in general are not perceived as realistic. One procedural technique we would like to mention is the locomotion system by Boulic *et al.* [19] that explicitly calculates the pelvis oscillation using biomechanical models. *Physics-based* techniques simulate locomotion using dynamics and physical properties of the body. These techniques yield realistic animations, but offer less control than procedural techniques and can be computationally expensive. A third class comprises *example-based* approaches. Existing motions are reused to generate a clip of locomotion. Often these motions are recorded by using motion capture systems. Basically, there are two main example-based techniques. *Motion concatenation* techniques stitch clips of motion together [2,20,21]. Often, all possible good transitions between motions are precomputed and stored in a graph-like structure such as a *motion graph* [2]. *Motion parameterization* techniques interpolate between existing motions to generate motions corresponding to a specific abstract parameter [22–24] such as the end-effector position. The former yields more natural animations while the latter offers a higher level of control. Combinations of motion parameterization and concatenation have also been investigated [25].

2.2.1. Distance Metrics.

To automatically concatenate or parameterize motions, a notion of resemblance is crucial. For example, to transition from one motion M_1 to a motion M_2, the end of motion M_1 should resemble the beginning of motion M_2. Various distance metrics can be used to detect these transition points [2,20,21]. Distance metrics for character poses are also important in many other fields and techniques, such as parametric motion graphs [25], but also in motion retrieval systems [26,27], performance analysis [28], time (or motion) warping [25], and transition generation [29]. Quite a few different metrics have been proposed, such as metrics based on joint angles [21], principal components [30,31] and point clouds [2]. We would also like to mention the metric proposed by Li *et al.* [32], which measures the effort needed to perform a transition. Similarly, Rose *et al.* [33] present a transition-generating algorithm that tries to minimize the joint torque. Ikemoto *et al.* [29] present a multi-way transition-generating technique that blends the source and target motion with intermediate motions determined by searching in a clustered space. To achieve this clustering, they use the distance metric by Kovar *et al.* [2]

Although many distance metrics exist, there is little known about how they compare to each other. Until now, a metric is compared only with an improved version of itself. Usually this improvement means optimizing the weights for specific joints to generate better transitions. For example, Matsunaga *et al.* [34] present a dynamics-based weight assignment scheme. The weight for a specific body part is based on the displaced mass and encountered friction of that body part during a transition. This weight assignment scheme is used to set the weights of the distance metric presented by Kovar *et al.* [2] resulting in less foot skating. Wang *et al.* [35] present an evaluation of the distance metric of Lee *et al.* [20]. They compare two sets of weights for this metric. The first set contains the original weights, as stated in the original distance metric paper, the second set contains automatically determined weights by solving a least-square minimization, after selecting some good and bad blends. They use a cross-validation and a user study to show that their set of weights results in a better set of blends.

2.2.2. Evaluation of Motion.

Much work has been done on evaluating generated motions. Safonova *et al.* [36] analyze the correctness of interpolated motion and present some simple modifications such that, in some constrained situations, the resulting motions adhere to physical constraints. For example, when one wants to blend two jumping motions of a character, linearly interpolating the center of mass of the character instead of the root position will result in a center of mass that has linear momentum, as should be the case during flight. Reitsma *et al.* [37] observe by user studies that errors in horizontal velocity and added accelerations are easier observed by humans than errors in vertical velocity and added decelerations. Ikemoto *et al.* [29] also evaluate transitions by foot skating and by evaluating the *zero moment point* (ZMP). The ZMP is the point on the ground plane at which the moment of the ground reaction forces is zero. In a physically valid motion, this point should be on or within the support polygon.

Ren *et al.* [38] present a tool to evaluate the naturalness of animations using a set of statistical models based on natural example data. However, their test set of unnatural motions only contains motion transitions based on a joint-angle metric. Reitsma *et al.* [39] present quality metrics to evaluate the (global) quality of motion graphs. Two of these quality metrics represent the coverage of the environment and to what extent the resulting path is close to the shortest path. These metrics are approximated by embedding the entire motion graph in a 4D grid. This scheme resembles our path deviation criterion, but it is limited to a *fixed environment* (i.e., an environment that is fully known beforehand). We also look at foot skating and we perform a qualitative analysis of blends that can benefit any environment.

2.3. Combination of Path Planning and Animation

Some research has been conducted on combining path planning and character animation. Choi *et al.* [40] create a map of footprints based on a probabilistic roadmap. They search this roadmap to find a collision-free path after which displacement mapping is used to adjust the motions to fit the target footprints. Sung *et al.* [41] also plan a path by using a probabilistic roadmap technique and generate a motion (by using a motion graph variant) which approximately follows the path. This motion is then adjusted to follow the path more accurately. Lau and Kuffner [42] precompute paths and motion clips based on a motion database and search over this graph-like structure for navigation and synthesis. This method implicitly uses the animation data to define the navigation space. Efforts have been made to speed up the search process by precomputing the possible positions that are reachable for the character by using a motion graph [43]. Srinivasan *et al.* [44] present a conceptually similar work. In Pettré *et al.* [45,46], the path planning and animation techniques are separated which means another path planner can be used to achieve similar results. They use a PRM to compute the path and interpolate motion clips to generate the character animation. Circular arcs are used to represent the root trajectory. Kamphuis *et al.* [47] also separate the path planner from the animation system. They assume the existence of a collision-free path. Then the animation technique based on the work of Pettré *et al.* [45] is used to generate an animation along the path. However, specific cyclic motion clips, such as walk cycles, are required to generate natural looking motions. Safonova and Hodgins [48] use a large parametric motion graph to follow a user-based (or automatically determined) sketch of path. This technique provides greater accuracy because it allows for interpolating two paths (or motions) in the motion graph and employing a global search over this enhanced graph. Some aspects of this technique (blend in corresponding contact phases, globally minimizing energy) might reduce the loss of oscillation, but it might not totally solve the problem. This is a computationally expensive offline technique that ideally is applied in a fixed environment.

There are also techniques that do not decouple the path planning and animation. Treuille *et al.* [49] concatenate animations of individual walk cycles. The next cycle is selected such that it safely moves the character to the goal and yields a good transition from the previous step. They train their controller using reinforcement learning. Unfortunately, although their technique is very fast, no guarantees can be made over the global optimality of the path that the character follows.

Attempts have been made to plan whole-body movement from scratch using techniques from robotics. For humanoid robots, the motion does not need to be natural [50], but for computer animation of virtual humans this is a requirement. Zhang *et al.* [51] present such a technique for static manipulation tasks in highly constrained environments. A sampling-based planner is used to generate an initial motion after which motion capture data are used to improve the naturalness of the motion. Unfortunately, generating natural locomotion from scratch is not yet feasible using these kinds of approaches.

3. MOTION SYNTHESIS

Motion synthesizers typically concatenate clips of motion. For our experiments, we will use the standard motion graph as described by Kovar *et al.* [2]

A motion graph is a directed graph whose edges correspond to clips of motion. We can create a trivial motion graph where the original motions are single edges. To seamlessly blend from one motion M_1 to another motion M_2 (or to add an edge between the corresponding edges) a distance metric is required to determine the good transition points. To incorporate dynamic properties of the motion, these distance metrics are often applied on a window of frames.

Because a motion is fundamentally the same when rotated about the vertical axis and translated in the plane, the motions in the database need to be aligned. We use the closed-form alignment function described in Reference [2]. The distance metric then determines the distance between all pairs of frames in the database, resulting in a distance matrix. Examples of these distance matrices for each of the three distance metrics are shown in Figure 2. Only local minima in this matrix are considered as transitions. Finally,

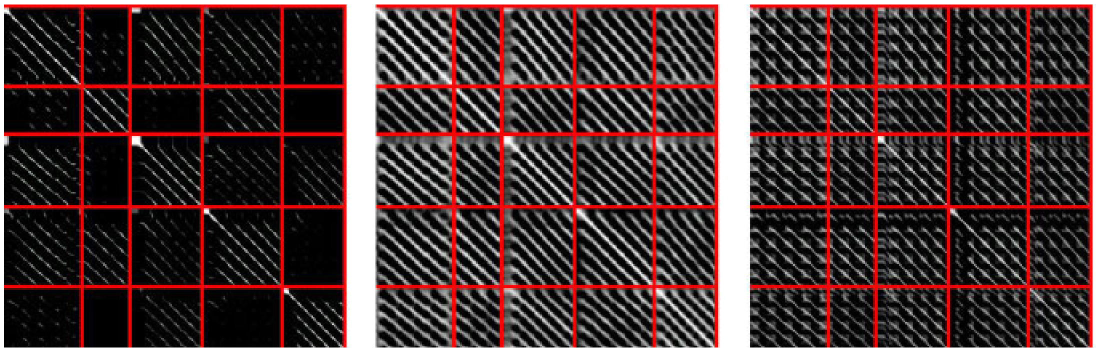


Figure 2. Distance matrices for the point cloud, joint angle and PCA metric. The distance is indicated by color (high distance = dark, low distance = light).

the graph is pruned by using the largest strongly connected component, which avoids dead-ends.

On-line path synthesis is performed by an incremental branch-and-bound algorithm. While the traversed distance of the motion is smaller than the length of the path, a branch-and-bound algorithm is employed, checking the search space n frames around the current node looking for the path that follows the path best. When an edge sequence of n frames is determined, the first m frames are selected and the node at frame m is picked as the new starting point for another search. For a more elaborate explanation on the motion graph and the branch-and-bound search algorithm, we refer the reader to the original article [2].

We would like to stress that motion concatenation algorithms consist of more components than just the distance metric. Several choices for parameters have to be made, as already indicated by Lamouret *et al*. [52] For example, one needs to choose which motions should be in the database as well as the duration of a transition [53]. In this paper, we only focus on the distance metric. The settings for all the other relevant parameters are motivated in the sections describing the experimental setups.

4. QUALITY MEASURES

We will use several quality measures to quantify the results of the techniques we propose. These measures will be described in this section.

4.1. Foot Skating

To measure foot skating, one needs to determine when the foot is planted. Simply checking the height of the foot with respect to the ground is insufficient. Motion capture data always contains some noise and artifacts due to targeting. When swinging the foot forwards when walking, the foot may come very close to (or even hit or penetrate) the ground [54]. Some techniques also incorporate the velocity of the foot, which is small during a foot plant. However, foot skating can introduce a large velocity so these methods will not detect all foot plants. Also, more advanced techniques exist, such as machine learning classifiers [54]. To avoid the issues that occur when using a height-based or velocity-based detector, we use a foot plant detection algorithm based on both height and time. First, the height-based test delivers us a set of foot plants. In addition to the height-test, we only accept foot plants if the foot plant occurs in a group of adjacent frames. We observed that this method works well in the case of walking animations. After determining the foot plants, the foot skating per frame is determined by calculating the sum of the distance between the ankle, subtalar and toe at that frame and the previous frame for both legs. The total foot skating is then defined as the sum of the foot skating over all frames.

4.2. Path Deviation

The resulting path of the animation P' is the projection of the root on the plane. We reparameterize the paths on traversed distance d and integrate the squared planar distance of the actual animation trajectory P' with length l and the desired path P_{plan}. So, for an animation M the path deviation $e_{\text{dev}}(M)$ is defined as

$$e_{\text{dev}}(M) = \int_{d=0}^{l} \left\| P'(d) - P_{\text{plan}}(d) \right\|^2$$

4.3. Wave Measures

It is known that the pelvis oscillates during walking [55]. This oscillation of the pelvis can be considered as a wave. Such a wave has a certain *wavelength* and *amplitude*. Note that the phase is not a good quality measure, for it depends on the foot that the animation starts with, which is dependent on the starting node in the motion graph.

The goal of our proposed solution is that the pelvis in the generated motions oscillates the same way as in the recorded motions. Therefore these techniques should yield motions with similar wavelength and amplitude. In contrast, the standard motion graph branch-and-bound algorithm implicitly

Table 1. The measurements of the recorded motions for the three test paths.

Measure	Straight	Low curve	High curve
Foot skating (cm)	0.20	0.28	0.39
Amplitude (cm)	3.91	3.55	4.51
Wavelength (cm)	57.83	55.31	49.06
Amplitude dev. (cm)	0.55	0.78	1.10
Wavelength dev. (cm)	1.33	3.73	6.44

tries to minimize the oscillation and thus should have a different outcome of wavelength and amplitude. We will determine the average wavelength and amplitude of the curvature function κ of the pelvis trajectory. It is straightforward to determine the average wavelength and amplitude of the wiggle of κ. Let $\mathbb{E}$ be the set of local extrema of the curvature function $\kappa(t)$. Then the wavelength w of two successive points $e_i, e_{i+1} \in \mathbb{E}$ is determined as two times the distance between those points $w(e_i, e_{i+1}) = 2d(e_i, e_{i+1})$. Then the average of all measured wavelengths is calculated by $\mu_{\text{wave}} = \frac{1}{n-1} \sum_{i=1}^{n-1} w(e_i, e_{i+1})$ where n is the number of extrema in $\mathbb{E}$. The amplitude of the wiggle is calculated in a similar way. First, the amplitude of a local extrema e_i is defined as $\text{amp}(e_i) = |\kappa(e_i)|$. Then the average of all amplitudes is calculated by $\mu_{\text{amp}} = \frac{1}{n} \sum_{i=1}^{n} \text{amp}(e_i)$.

Ideally, one would expect that the wavelength and amplitude of the pelvis in a generated motion should have an equal variance as in original motion. Therefore we will also measure the standard deviation σ_{wave} and σ_{amp} of the wavelength and amplitude. In Table 1 we show the wave and footskate measurements of the recorded motions corresponding to a straight path, low curvature path and high curvature path. Using these measured values, we can compare generated motions with real motions.

4.4. Online Running Time

The online running time is measured as the time needed by the branch-and-bound algorithm. Building up the graph is not taken into account, since this can be done during pre-processing. The running time of the branch-and-bound algorithm is an especially important measure for interactive applications, where motion needs to be generated on-the-fly. Running times are on a Pentium Intel DualCore 2.4 GHz with 1 GB of RAM.

5. CHOOSING POSTURE DISTANCE METRICS

In this section we will describe the first problem, namely selecting the best posture distance metric. As already said, to minimize the visual artifacts that can be introduced by blending two clips, transitions should only be added between frames that "resemble" each other. Therefore, a distance metric is required that quantifies the resemblance between frames/poses. Many different metrics exist, and the behavior and properties of a particular metric have a big influence on where transitions are added, and thus also on the quality of resulting motions. However, little is known about how the use of a given distance metric influences a motion synthesizer when generating locomotion along a path. In this section, we will analyze the effect of using a distance metric in a motion graph both quantitatively and qualitatively. The quantitative analysis is in terms of foot skating, path deviation and online running time. The qualitative analysis is done by a user study, where users rate the quality of blends corresponding to different distance error values.

5.1. Posture Distance Metrics

In this section, we will discuss the three metrics that are evaluated in this paper.

5.1.1. Joint Angles.

The most basic distance metric is based on calculating the difference between joint angle values and optionally, a number of derivatives such as velocity and acceleration. Several researchers have used this metric or a variation on it. For example, Arikan *et al.* [21] compare animations by evaluating the joint position, joint velocity, torso velocity, and torso acceleration. For a given frame/pose i, let us define $p_i \in \mathbb{R}^3$ as the global (root) position and $q_{i,k} \in S^3$ as the unit quaternion describing the orientation of a joint $k \in$ the joint set $\mathbb{J}$. We will evaluate the metric by Lee *et al.* [20] The distance between a frame a and a frame b is defined as follows

$$d(a, b) + \nu d(\dot{a}, \dot{b})$$

where $d(a,b)$ describes the weighted differences of joint angles and global position, and the second term $d(\dot{a}, \dot{b})$ represents the weighted (Euclidean) differences of the joint velocities. The constant ν defines the influence of velocity on the total distance. The joint angle differences are calculated as follows

$$d(a, b) = \| p_a - p_b \|^2 + \sum_{k \in \mathbb{J}} w_k \left\| \log(q_{b,k}^{-1} q_{a,k}) \right\|^2$$

The first term describes the squared difference between the global (root) positions of frame a and b. The second term describes the weighted sum of the orientation differences. Note that because of the Euclidean summing operation, the orientations need to be written in a suitable format, in this case the exponential map [56].

5.1.2. Geometrical.

The distance metric that was used in the original motion graph paper by Kovar *et al.* [2] was based on *point clouds*. For each frame, a point cloud is constructed, which, ideally, is a simplification of the character's mesh. To account for velocity and acceleration differences, the authors compare a window of frames. The final distance is the minimal weighted sum of the squared distances between point clouds given an arbitrary rotation about the vertical (y) axis and translation on the (x,z) plane

$$d(a, b) = \min_{\theta,x_0,z_0} \sum_i w_i(a_i - T_{\theta,x_0,z_0} b_i)^2$$

where a and b are the two point clouds, and the linear transformation T_{θ,x_0,z_0} rotates a point around the y-axis by θ degrees and then translates it by (x_0, z_0). The index i is over the number of points in the point cloud. This optimization has a closed form solution, for which we refer the reader to Reference [2].

5.1.3. Principal Components.

Several authors have proposed to use principal components as a representation of posture [30,31]. A principal component analysis (PCA) is performed on the rotations of the skeleton joints. Postures can then be compared in principal component space. The advantage of this approach is that dependencies between joints are taken into account in the distance metric. Since PCA is ideally applied to linear spaces, the rotations are expressed in the exponential map representation. Suppose that the frames are represented by an N-dimensional vector of Principal component (PC) values. The distance between two frames a and b is then defined as the weighted Euclidean distance between the two vectors (w_i is the weight of variable i)

$$d(a, b) = \sqrt{\sum_{i=0}^{N} w_i(a_i - b_i)^2}$$

Forbes *et al.* [31] have shown that not the entire set of PC values need to be used for calculating this distance, but only the most important ones. By using a reduced number of PCs, a more efficient distance calculation can be achieved, although the conversion from and to the exponential map is still required.

5.2. Experimental Setup

We evaluate the three different distance metrics discussed in the previous section on a qualitative and a quantitative level. In the following sections, we will discuss these evaluations in more detail.

5.2.1. Choosing Weights.

One of the major difficulties of configuring a distance metric is the assignment of weights to joints or body parts. Larger weights will result in a larger influence of these parts on the distance value. In this section we will describe the weight configuration of the metrics. The weights for the three metrics described in this section hold for all frames in the window.

For the joint-angle distance metric, we use the weight set determined by Wang *et al.* [35] instead of the manually determined weights proposed in the original work [20]. Furthermore, we do not take into account the difference in absolute root position (the first term in the distance function), because this is only useful in fixed environments. Second, considering the fact that we only use forward walking motions, we set the weight ν for the joint velocity very low (0.0001). The joint velocity serves primarily to distinguish between forward and backward motions [20], yet we consider only forward walking motions with roughly the same velocity in our database.

For the point-cloud metric, we mimic the approach of Matsunaga *et al.* [34] by assigning a higher weight to joints which displace a lot of mass and which are in contact with the environment. Because we mainly deal with locomotion animation, the feet are the only body parts that are in contact with the environment. Since they also carry almost all body mass, our weight scheme assigns a high weight to the feet, and this weight decreases when going up the articulated body (until the root). The hips, knees, ankles, subtalars, and feet have a weight of 0.6, 0.7, 0.8, 0.9, and 1.0, respectively. The other joints have a weight of 0.5. Note that it is not possible to have an exact equal weight set for the joint-angle and point-cloud metric because both metrics work on another posture representation. Therefore there exists no one-on-one mapping between weights.

Finally, the PC weights are defined as the values of the Eigen vector of the PC matrix [30], since these values correspond to the occurrence in the data of each PC value. We use all principal components, although it is also possible to use less [31].

For both experiments, we have evaluated the distance metrics over a window of 10 frames.

5.2.2. Quantitative Evaluation.

We evaluate the three metrics using three quality measures: foot skating, path deviation, and on-line graph search time. For each distance metric, we will generate a number of different motion graphs. Each motion graph is evaluated using a long test path (33 meters). This path is a "zigzag" and requires both straight locomotion and curved locomotion. Especially in the case of locomotion, the curvature of the path greatly influences how the outcoming motion is constructed.

We evaluate the graphs of four different *complexities*. We define this motion graph complexity as a triple (N,E,K) consisting of the number of nodes, edges, and keyframes that comprise the graph. These numbers are listed in Table 2. The three different metrics result in graphs with a different node-edge-frame ratio. We have tried to match the graph complexities as closely as possible. After creating these graphs, the branch-and-bound algorithm described above generates the animations according to the path. The starting node in the graph is randomized. For the search algorithm, we set n to 80 and m to 25, as proposed in the original paper. Recall that n is the number of frames we look ahead and m is the number of frames we select from the n frames to add to the animation. Initially, we align the animation with the

Table 2. The complexities of the graphs based on the 3 metrics.

Metric	# edges	# nodes	# frames
Geom I	271	167	1420
Geom II	644	352	3532
Geom III	1013	493	5894
Geom IV	1263	542	7911
Angular I	253	160	1508
Angular II	632	365	3331
Angular III	1017	516	5720
Angular IV	1264	582	7535
PCA I	258	161	1437
PCA II	638	352	3498
PCA III	1024	479	6096
PCA IV	1268	534	8027

first segment of the test path, so that the character does not need to turn around before being able to follow the path. All quantitative measures are measured over 20 executions of the branch-and-bound algorithm for each path. No motion editing or correction is done after generating the animations with the branch-and-bound algorithm.

To evaluate the metrics, we use a database that contains five animations of locomotion containing in total 832 frames. Each animation is a clip of locomotion with a specific curvature (straight ahead, left and right curves, and tight left and right curves). Although this is a fairly small database, the content is sufficient for the paths we use as query.

5.2.3. Qualitative Evaluation.

We perform an on-line user study (251 participants) to evaluate the quality of blends occurring between different motions. The goal of this study is to gain insight in the *meaning* of a given distance value. By having a better view on what a certain distance value means, one can more easily set correct parameters in a motion synthesizer. The participants are visitors of graphics and game-related Internet forums [57,58]. In the user study, we present the users 40 clips of locomotion of approximately 5 seconds out of a pool of 50 motions. The pool of motions consisted of 45 blends of the motions described above corresponding to different distance values from all three distance metrics. Next to these 45 blends, the pool contained five original motions of 5 seconds to obtain a ground truth. These clips are presented in random order.

The blends occur at random moments (between 1 and 4 seconds) in the shown animations. As already mentioned, all blends are over 10 frames, using SLERP for the rotation and linear interpolation for the root translation. No motion editing has been done on the resulting animations. We use the same geometric model for all clips, because the geometric model influences the perception of the viewer [59]. The subjects are asked to grade the motion realism of the clips on a scale of 1 to 10. Our hypothesis is that there should be a strict inverse proportional relationship between the distance and the grade of the subject.

The thresholds we select for the pool are the upper limit of the first 0.5, 1.0, 2.0, 8.0, and 32.0% of the distances of the frames in the database. We have not opted for a uniform selection of the transitions between a distance of 0 and the maximum distance, because this would result in mostly blends with high distances, which are not interesting for our study. We want to focus primarily on the lower distances, where the distinction between blend and original motion becomes less obvious. An additional advantage of our proposed selection scheme is that it applies to all metrics. Note that a presented movie is a sample for all three metrics, for we can determine the distance between the two blended frames. A screenshot of the on-line questionnaire can be seen in Figure 3.

5.3. Results

In this section we will present the results from the quantitative and qualitative evaluations of the distance metrics.

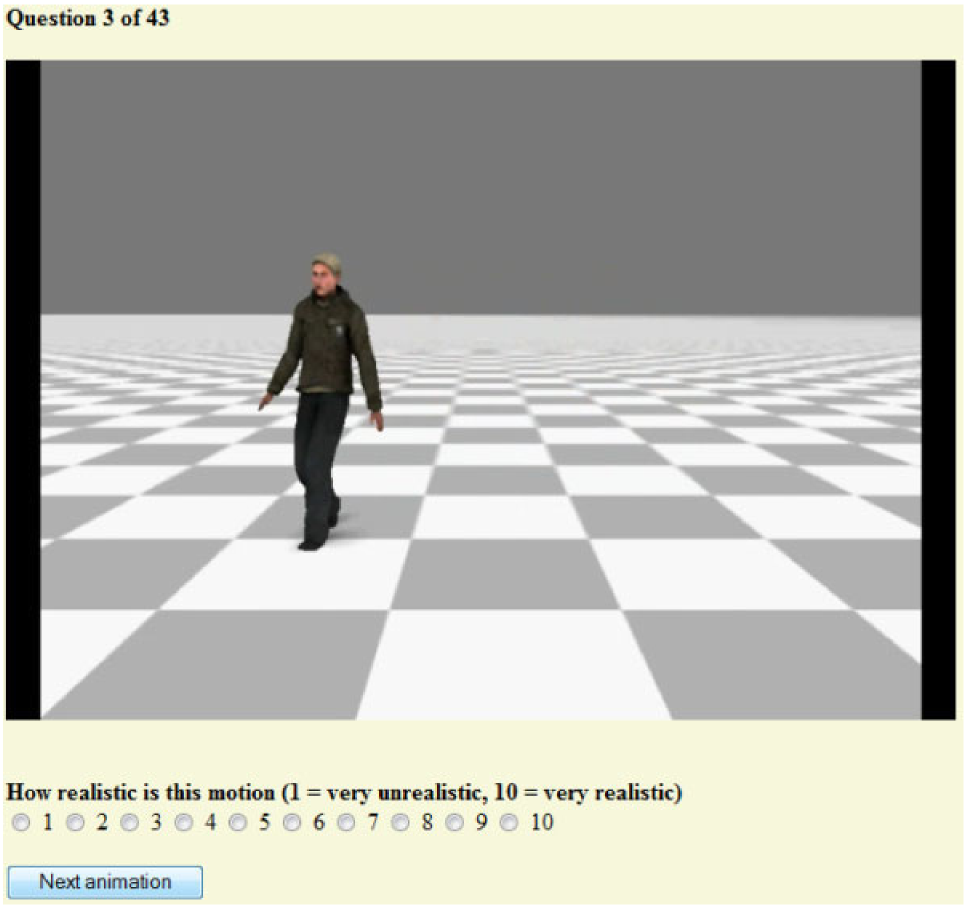
5.3.1. Path Deviation.

The resulting animations of the quantitative analysis were 35–40 seconds long. As was expected, the coarsest graphs (complexity I) result in animations with a high path deviation. We see that the path deviation of the graph constructed using the point-cloud metric is higher than the graph constructed using the joint-angle metric (see Table 3). The main reason for this difference is that the point-cloud metric is geometry-based. A point cloud built over a curved locomotion differs from one built over a straight locomotion. This is primarily due to the fact that the point cloud is built over a window of frames. This problem also occurs when we require a blend to an animation consisting of locomotion at a much higher velocity, where the point cloud generated for the faster animation will be much more stretched out. Decreasing the window size will not resolve this issue. When looking at the difference of a single posture from a straight locomotion and a single posture of a curved locomotion, it appears that the upper body is slightly twisted. In case of a joint-angle based metric, this will affect the angles of a small number of joints. In case of the point-cloud metric, it will affect a large set of points, mostly in the upper body. The result of this difference is that the point-cloud metric results in a graph with more self-blends: blends going *from* and *to* the same animation. The point-cloud metric prefers blends *from* and *to* animations of locomotion with the same curvature, resulting in generated animations that show more path deviation.

The path deviation from the PCA-based graph is also often higher than the joint-angle based graph. Like the

Table 3. The path deviation (in cm^2) for the graphs.

Path d.	I	II	III	IV
Geom	2034946.2	160938.4	78837.7	67054.7
Angular	1485244.7	82974.5	75676.1	50273.2
PCA	1274669.5	163120.6	82585.6	54404.1


Question 3 of 43

How realistic is this motion (1 = very unrealistic, 10 = very realistic)

◯ 1 ◯ 2 ◯ 3 ◯ 4 ◯ 5 ◯ 6 ◯ 7 ◯ 8 ◯ 9 ◯ 10

Figure 3. A screenshot of the on-line questionnaire.

point-cloud metric, the graphs generated with the PCA metric contain more self-blends. This is because the weight set of the joints in the PCA metric is different from the weight set of the joint-angle metric. Many weights in the weight scheme of [35] are set to zero, whereas they can be non-zero in case of the PCA. We evaluated the influence of the joints in the PCA metric by multiplying the transposed (absolute) PCA matrix (consisting of Eigenvectors) with the Eigenvalues. After normalizing the weights such that the sum of the weights equals the sum of weights determined by [35] it turns out that some joints greatly influence the PCA metric, such as the spine joint, whereas they have been assigned zero weight in Wang's weights. The PCA metric, however, assigns a high influence to the shoulder and elbow joints, similar to Wang's weight set.

In conclusion, when low path deviation is an important concern in an animation system, the joint-angle metric seems to be the best choice.

5.3.2. Foot Skating.

We measured the foot skating of the results of the four graphs per metric. The results are shown in Table 4. The point-cloud based graph results in the least foot skating, followed by the PCA-based graph. This is clearly an advantage of a metric in the geometrical domain. The fact that the PCA metric performs better than the joint-angle metric might again be due to the fact that the PCA assigns tighter weights than the joint-angle metric.

The number of edges in a graph is limited by the local minima in the distance matrix. Our observation is that the joint-angle metric introduces a lot less local minima. Hence, to generate a large graph, more local minima need to be added to be able to generate enough edges. This can be done by decreasing the window size, since the window acts as a low-pass filter on the distance matrix. However, reducing the window size might result in more foot skating. In conclusion, if low foot skating is crucial, the point-cloud metric performs best.

Table 4. The foot skating (in cm) for the graphs.

Skating	I	II	III	IV
Geom	1608.17	1604.40	1829.22	1975.81
Angular	1664.72	1987.65	2482.57	3174.35
PCA	1528.57	1895.00	2088.76	2340.05

Table 5. The time (in seconds) needed for the branch-and-bound algorithm.

Time	I	II	III	IV
Geom	2.16	9.94	41.15	125.53
Angular	2.65	3.86	14.91	21.53
PCA	0.58	5.49	41.98	80.86

5.3.3. Running Time.

We also looked at the running time of the branch-and-bound algorithm for each distance metric. The branch-and-bound algorithm runs much slower on the point-cloud generated graph than the graphs generated by the other two metrics. Table 5 shows that the joint-angle metric is often faster than the other two metrics.

We have found a few possible explanations for the slow-down of the point cloud-cloud metric. The point-cloud metric results in a graph that contains highly connected sub-graphs where the search algorithm mostly resides. We have observed a correspondence between the average out-degree of the encountered nodes in the search algorithm and the running time. The average out-degree of the encountered nodes in the search algorithm on the point-cloud based graph is higher than the average out-degree of the nodes encountered in graphs generated by the other two metrics.

Also, the occurrence of small local cycles can increase the time needed for a search. The search space can grow rapidly if the search returns to the starting node early in the process. For the graphs generated by the PCA and point-cloud metric, the average length of the shortest path from each node to itself (excluding the empty path) is around 8–9 edges, whereas it is around 10–11 for the graph based on the joint-angle metric.

Another observation was that most transitions are cluttered at certain pieces of motions in case of the PCA and point-cloud. For example, for all interblends (blends between different animations and hence, different curvature) in the point-cloud based graph, only 4% was transitioning to animation 5 (tight left curve). For the interblends of the PCA based graph 13% was transitioning to animation 5 and for the interblends of the joint-angle based graph 19% were transitioning to animation 5. We also observed this clutter of transitions in the distance matrices (of which examples are shown in Figure 2) where they correspond to local minima. For completeness, the branching factor of the branch-and-bound algorithm did not differ much between the graphs generated by the three metrics.

5.3.4. User Study.

In this section, we will discuss the results of our user study. Surprisingly, we did not find a strictly inverse proportional relationship between the distance metric and the grade of the subject. There appears to be a plateau for all three metrics at the smaller distances, as can be seen in Figures 4–6. The red lines (indicating a grading of 7.65) represent the ground truth. Note that it is not possible to plot all relations in a single graph, for there is no 1-on-1 mapping between the distance metrics.

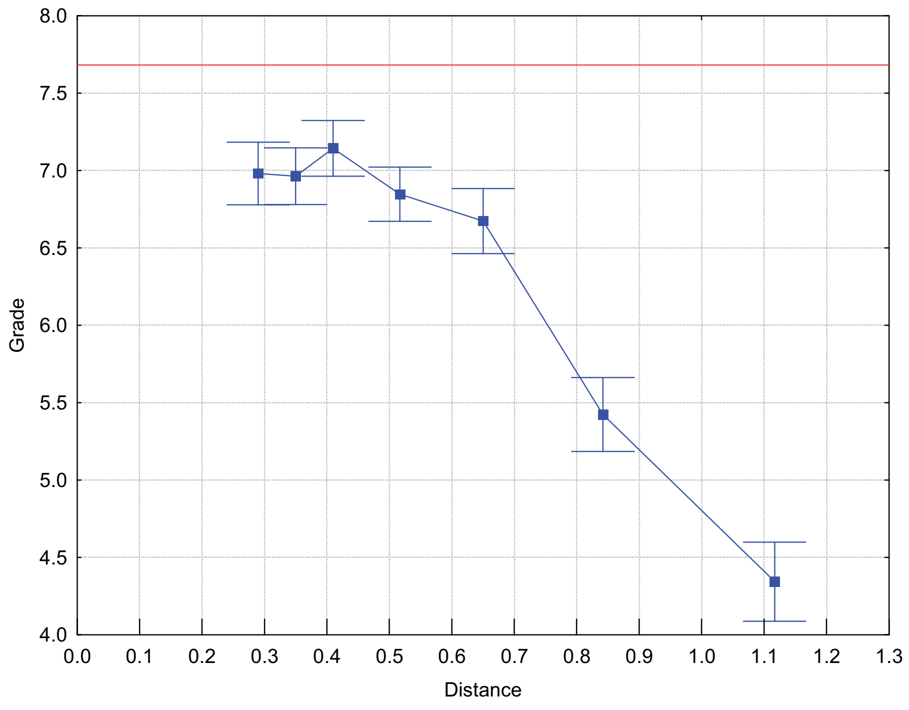


Figure 4. The relation between the distance from the PCA metric and the average grade given by the subjects. Vertical bars denote 0.95 confidence interval.

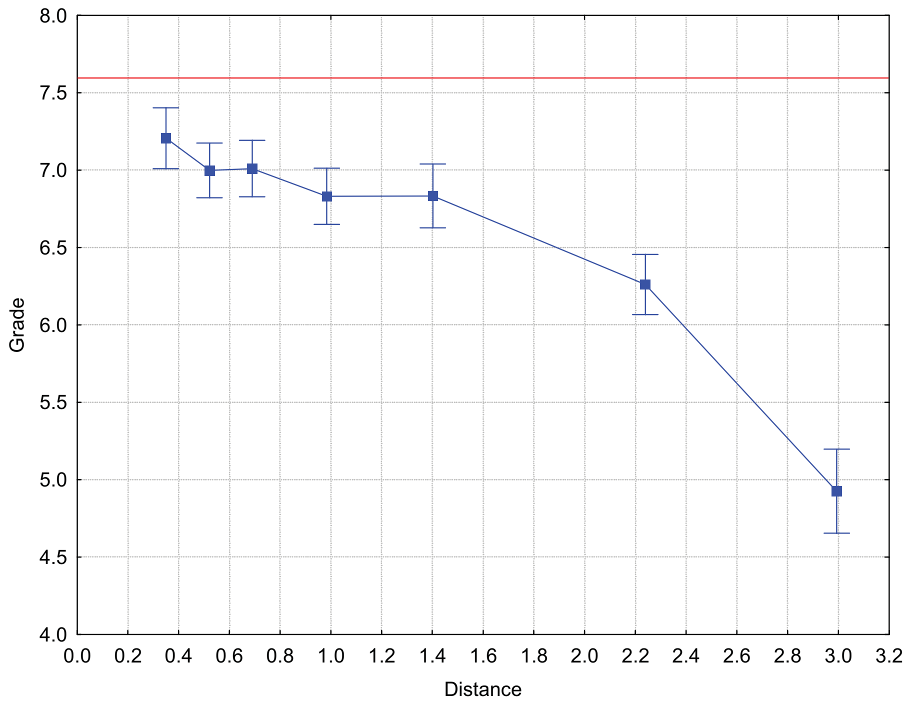


Figure 5. The relation between the distance from the joint angle metric and the average grade given by the subjects. Vertical bars denote 0.95 confidence interval.

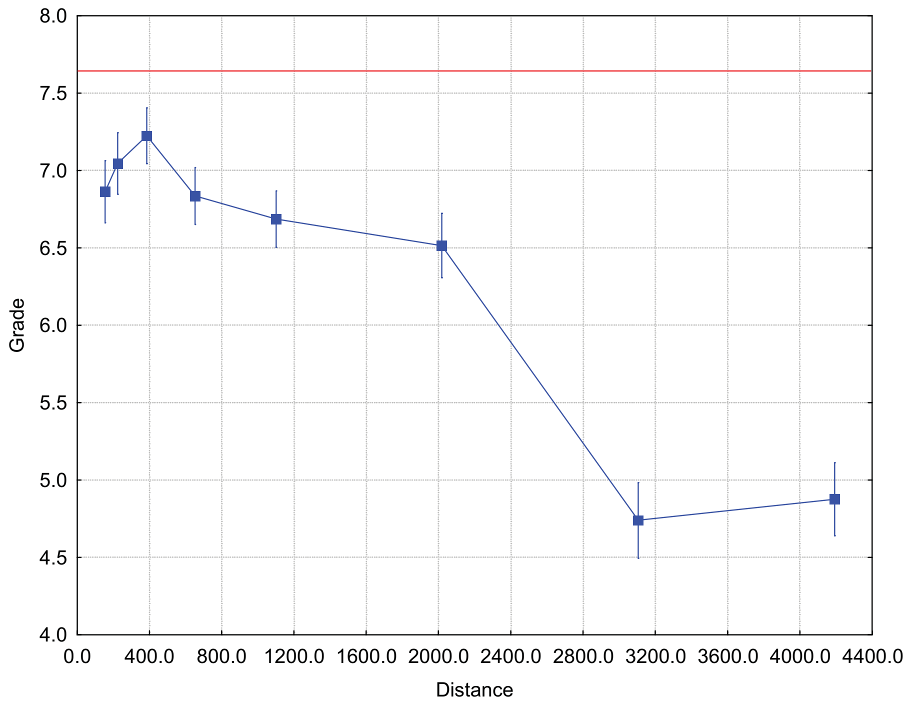


Figure 6. The relation between the distance from the point cloud metric and the average grade given by the subjects. Vertical bars denote 0.95 confidence interval.

We have performed a *post hoc* analysis (Tukey, $p = 0.05$) on the results to determine the sequential significance between the distances. For the point cloud metric, the only significant decrease is found from the thresholds 2018 to 3105. Indicating that the perceptual plateau ends somewhere in that bin. These two thresholds correspond to motion graphs with complexities ($N = 701$, $E = 2923$, $K = 22980$) to ($N = 715$, $E = 3208$, $K = 25704$) where N, E, and K are the number of nodes, edges, and keyframes, respectively.

For the PCA metric, the first two significant decreases of grading is found from 0.650 to 0.842 and 0.842 to 1.117, indicating bin 5 to 6 and 6 to 7. The perceptual plateau ends somewhere between 0.65 and 0.842. These two thresholds correspond to motion graphs with complexities ($N = 606$, $E = 1544$, $K = 10118$) to ($N = 663$, $E = 1852$, $K = 12645$). As for the joint angle metric, the first significant decreases of grading is found from 1.402 to 2.2395 and 2.2395 to 2.994, indicating bin 5 to 6 and 6 to 7. The perceptual plateau ends somewhere between 1.402 and 2.2395. These two thresholds correspond to motion graphs with complexities ($N = 528$, $E = 1053$, $K = 5961$) to ($N = 606$, $E = 1341$, $K = 8094$).

The plateau implies that in some cases more complex graphs can be constructed with a higher threshold, without sacrificing blend quality, as long as the selected threshold is on the plateau.

None of the grades, given for the metrics, approached the grade given for the ground truth animations. Ideally, humans should not be able to distinguish between computer-generated animations and motion captured animations. An animation Turing test has already been proposed by [60]. Our work contains such a Turing test: the computer-generated blends for the smallest distances did not get a grade as high as the one given for the ground truth. Thus, the blends generated by all metrics with our selected thresholds and weights did not pass the Turing test.

6. PREVENTING UNNATURAL PELVIS OSCILLATION

As already shown by the previous problem, a generated animation can not always be guaranteed to exactly follow a planned path. Errors may occur due to curvature limits in the motion database. If a path planner generates a path with a high curvature that is not present in the motion database, the character might not exactly follow the path. A possible solution for this type of problem is to apply a path editing technique [61]. However, this assumes that we know what a path actually represents: the path P_{plan} generated by a path planning algorithm is a parametric curve that represents a simplification of the path the character needs to follow. Generally it is not clear which (body) part of the character should follow this path. In many systems the path is interpreted as the desired trajectory of the projection of the pelvis on the plane. However, when one takes a closer look at the trajectory of the pelvis during locomotion it appears that the pelvis oscillates during such a motion. Early experiments in biomechanics already showed that the center of mass oscillates both horizontally and vertically during locomotion [55]. When the motion synthesizer forces the pelvis to closely follow the desired path one loses this natural oscillation or *wiggle*, resulting in an unnatural motion. Not enforcing the path constraint will lead to path deviation, but over-enforcing the path constraint leads to unnatural motion.

In this section, we propose a solution to this problem by using a *path abstraction*. We will show that using an abstract path P_{abstract} instead of the traditional pelvis trajectory will lead to more natural motions while still being able to enforce path constraints. We propose several ways of obtaining an abstract path from an existing animation and we will compare their advantages and limitations. Path abstractions can easily be incorporated in motion synthesizers. Because it can be done in the preprocessing phase, it will not affect the performance of the motion synthesizer.

We can approach this problem from two sides. We could incorporate the pelvis oscillation in the paths of a path planner, but this has some major drawbacks. The oscillation is highly dependent on the character and we would need to know in which phase of the locomotion cycle we want to begin. For example, we would need to know which foot we will swing first. Therefore we do not change the path but we change the representation of the character motions in the database. We do not try to force the pelvis to follow the presented path, but an abstraction of the character motion.

The motion graph search algorithm is the only component of the path planning and animation system that needs to be changed, not the motions, motion graphs or paths. Again, we use the standard search algorithm for motion graphs [2]. While the traversed distance of the motion is smaller than the length of the path, this branch-and-bound algorithm is employed, checking the search space n frames around the current node looking for the motion clip that follows the path P_{plan} best. To find the motion clip that follows the path best often the projected trajectory of the pelvis on the plane is compared to the query path P_{plan}. This results in losing the natural oscillation of the pelvis as explained in the previous sections. In our method, when selecting a new motion clip from the graph, we compare the the query path P_{plan} with a path abstraction P_{abstract} instead of the pelvis trajectory.

6.1. Path Abstraction Techniques

In this section we will present the four path abstraction techniques. Although most of them act as a low-pass filter, the resulting path abstraction does not need to be the same. We will denote P_{wiggle} as the original oscillating pelvis trajectory projected on the ground plane, P_{abstract} is the trajectory of the path abstraction. Note that all path abstraction techniques basically remove the local pelvis oscillation. We will show in subsection "Results" that applying these path abstraction techniques on the database, as described in the previous section, will result in animations that exhibit a natural pelvis oscillation in terms of amplitude and wavelength (see subsection "Wave Measures").

6.1.1. Joint Combination (JC).

When examining recorded motion, it appears that there is no joint without oscillation. We cannot choose a particular joint as indication of the character's global motion, although the combination of several joints might be interesting. While the *trunk* is a good indication of the character's direction [62], the position of the trunk still suffers from an oscillation. Fortunately, some joints appear to follow a path that resembles the mirrored version of the pelvis path. These are the feet, ankles and knees. We will determine P_{abstract} by interpolating the trajectories of these joints.

6.1.2. Joint Orientation Extrema (JOE).

We have observed that the global path roughly follows the midpoints of two consecutive local extrema of the pelvis trajectory. Since extrema are easily determined in monotone functions we will transform the non-monotone path such that it is x-monotone. We can then determine extrema by finding zero-crossings of the first derivative. P_{abstract} is then determined by connecting the midpoints of the segments between consecutive extrema.

6.1.3. Gaussian Smoothing (GS).

The third path abstraction method is Gaussian smoothing. It is suitable for smoothing arbitrary curves and is used in image processing to filter out high frequency distortions. The wiggle of the pelvis can be viewed as such a distortion of the global path of the character. An approximation of P_{abstract} is obtained by applying Gaussian smoothing to the path of the pelvis. The pelvis trajectory P_{wiggle} is represented by two parametric functions $x(t)$ and $y(t)$ where t indicates the path length. These are both convolved with a one-dimensional Gaussian kernel $G_\sigma(t)$ of standard deviation σ.

$$G_\sigma(t) = \frac{1}{\sigma\sqrt{2\pi}} e^{((-t^2)/(2\sigma^2))}$$

$X(t)$ is defined as the convolution of $x(t)$ with Gaussian kernel $G_\sigma(t)$ for some value of σ: $X(t) = G_\sigma(t) \otimes x(t)$. $Y(t)$ is defined similarly.

Differentiation commutes with convolution, so $X(t)' = G'_\sigma(t) \otimes x(t)$, and $X(t)'' = G''_\sigma(t) \otimes x(t)$, where $G'_\sigma(t)$ and $G''_\sigma(t)$ are the first and second order derivatives of the Gaussian kernel. $X(t)$ and $Y(t)$ represent a smoothed version of P_{wiggle} with a smoothing factor of σ. $X(t)$ and $Y(t)$ can be used as an approximation for P_{abstract}.

A problem with any averaging filter is that the resulting path is shrunk toward the center of its supporting circle. Each point is filtered by using its neighbors that in both directions curve toward the center of the supporting circle. A solution to this problem was provided by Lowe[63]. He states that the shrinkage is dependent on the amount of smoothing and the local curvature. We can use the known value of σ and the measured curvature of the smoothed curve to compensate for the degree of shrinkage that must have occurred. Since we do not actually know the value of the original curve radius r, we need to use the second derivative of the smoothed curve X. Given σ and the measured curvature of the smoothed path the shrinkage Δx at a point t is defined as

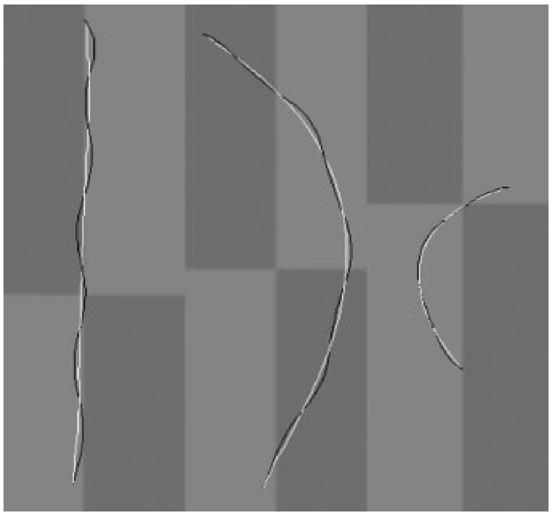


Figure 7. We can remove the oscillation of the pelvis (black) by applying a Gaussian filter (white).

$$\Delta x(t) = r(1 - e^{(-\sigma^2)/(2r^2))})$$

where r is numerically determined using the second derivative of the smoothed function

$$X''(t) = \frac{e^{(-\sigma^2/(2r^2))}}{r}$$

A lookup table is constructed that maps the second derivative of convolution $X''(t)$ on the shrinkage error value Δx. Intermediate values are linearly interpolated. For each point on the path the appropriate error value is looked up and subtracted from the smoothed coordinate value. Several solutions exist to convolve the Gaussian kernel when it extends beyond the end of the path. We will use the method of Rosin [64] where the path is extended by 3σ at both ends by duplicating and reflecting the path (see Figure 7 for Gaussian filtering on the pelvis oscillation). We use a σ of 0.75 meter, corresponding to one step cycle.

6.1.4. B-Spline (BS).

For the last path abstraction method, we follow the idea of Gleicher [61]. He uses a uniform cubic B-Spline with six knots located at uniform intervals in the trajectory of the pelvis. It can be interesting to see how well this method performs in comparison to the already proposed methods.

Given a set of control points P, for each segment i we determine the trajectory of P_{abstract} using

$$P_i(t) = [t^3 t^2 t 1]\frac{1}{6}\begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{pmatrix}\begin{pmatrix} p_{i-1} \\ p_i \\ p_{i+1} \\ p_{i+2} \end{pmatrix}$$

for $t \in [0, 1]$.

6.2. Experimental Setup

To evaluate the result of each path abstraction technique we compare the generated animations with real recorded motions. We have motion-captured a human walking along three paths: a straight path, a low curvature path and a high curvature path. We manually extracted the test paths and used them as input to the motion graphs. This allows us to compare the generated animations with the original recorded motions. We will generate motions using the five different path abstraction techniques (including the standard pelvis-based method). Note that we do not change the motions in the database, but we change the representation. In the branch-and-bound algorithm we do not compare the pelvis trajectory to the query path, but to the path abstractions. In total we have five different versions of the search algorithm, each corresponding to a different path abstraction technique. Since the transition threshold of the motion graph greatly influences the quality of the resulting animation, we will use seven different threshold values. We will use a common joint-angle based distance metric [65]. For each threshold a different motion graph is constructed. The properties of the graphs for the seven thresholds are described in Table 6. These thresholds vary from a really small graph, which barely allows for following a path at all, to a large graph which is able to strongly enforce the pelvis to follow the path. For all thresholds each technique will generate 40 animations per test path. We build up a motion graph using five recorded walking motions of varying curvature (done by the same human as the test paths). The total number of frames was 683.

As said, we will use seven different transition thresholds, ranging from 0.5 to 3.5. Each threshold will lead to a different motion graph. The higher the threshold the more transitions are allowed and the denser the resulting motion graph will be. In Table 6 the number of edges, nodes and frames per graph are shown.

Table 6. The complexities of the graphs for different thresholds

Threshold	# nodes	# edges	# frames
0.5	79	114	653
1.0	295	528	2874
1.5	475	925	5202
2.0	566	1300	8046
2.5	599	1468	9403
3.0	611	1563	10235
3.5	613	1584	10425

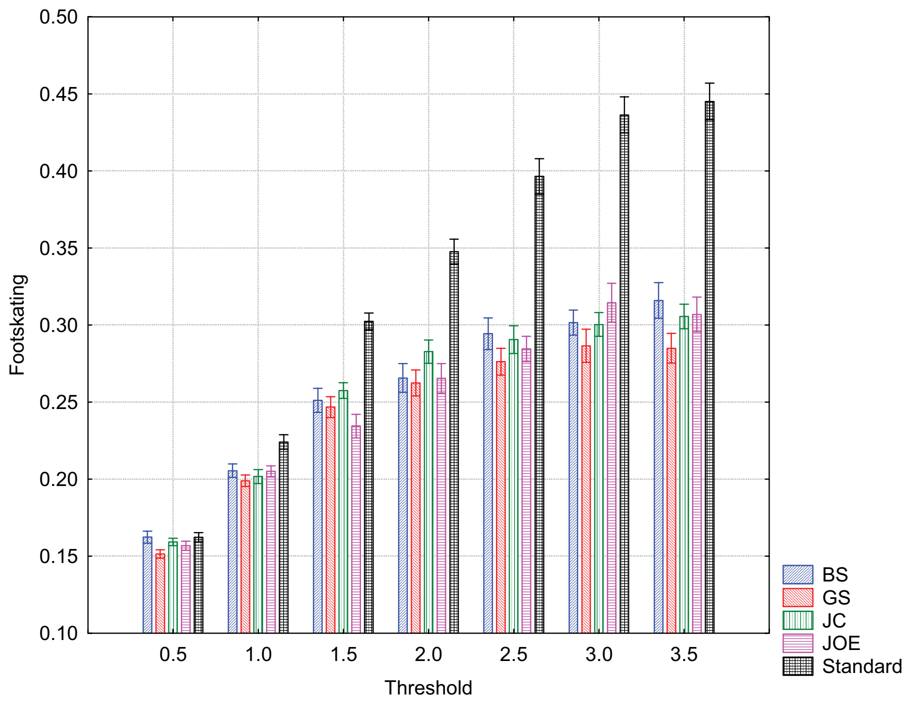


Figure 8. The average foot skating for each method and technique over all paths. Vertical bars denote standard error of mean.

6.3. Results

In this section we will discuss the results of the techniques separately for each quality measure. To analyze the techniques statistically, we employ three repeated measures ANOVA, one for each quality measure. Whenever significant effects have been found, *post hoc* analysis (Tukey, $p = 0.05$) was performed for pair-wise comparison. For some examples of resulting motions of the path abstraction techniques, we refer the reader to the accompanying demo video.

6.3.1. Foot Skating.

We found an effect of technique on foot skating ($F(4, 156) = 367.40$, $p < 0, 001$). The foot skating for each method and technique over all paths is depicted in Figure 8. On the horizontal axis the seven thresholds used during the graph creation ranging from 0.5 to 3.5 are shown. Pairwise comparison showed that the standard method generates significantly more foot skating (over all paths and thresholds) than all other techniques (all $p < 0.001$) and the GS method generates significantly less (all $p < 0.01$). We have observed that for the high curvature path, the foot skating of all techniques is lower than the foot skating of recorded motion. This is mainly because the motion graph is not able to generate animations with such a high curvature. It is known that there are not many transitions in the graph going to animations with such a high curvature [65].

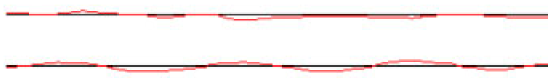


Figure 10. Two resulting pelvis trajectories (red) for the query paths (black). As can be clearly seen, the Gaussian smoothing path abstraction (below) preserves the natural oscillation of the pelvis whereas the standard method (top) does not.

6.3.2. Amplitude.

We found an effect of technique on the absolute difference of the amplitude of the resulting animations and the original recordings ($F(4, 156) = 630.32$, $p < 0, 001$). The amplitude difference for each technique over all paths and thresholds is depicted in Figure 9. It is clear that the path abstraction techniques are able to preserve the wiggle with a similar amplitude as the recorded motion, as their difference is significantly smaller (all $p < 0.001$) (see Figure 10). Overall, the amplitude differences from the GS and JC method are significantly smaller than the other techniques (all $p < 0.004$). We have observed that for the standard method the amplitude decreases as the threshold increases and becomes much smaller than the amplitude of recorded motions. This typically illustrates the problem of loss of oscillation. For the high curvature path we have observed that the amplitudes of all the path abstraction techniques are at a similar level as the recorded motion, except for the threshold of 0.5. This is because the motion graph is quite small and is unable to correctly follow the path and often generates straight locomotion, which has a smaller amplitude than curved locomotion.

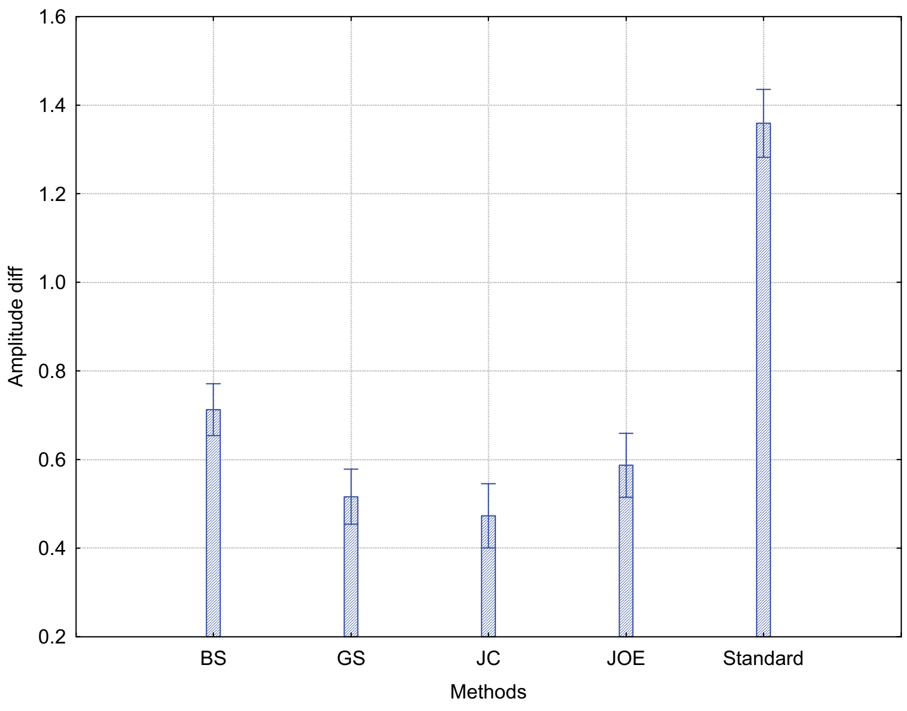


Figure 9. The overall average amplitude difference for each technique. Vertical bars denote standard error of mean.

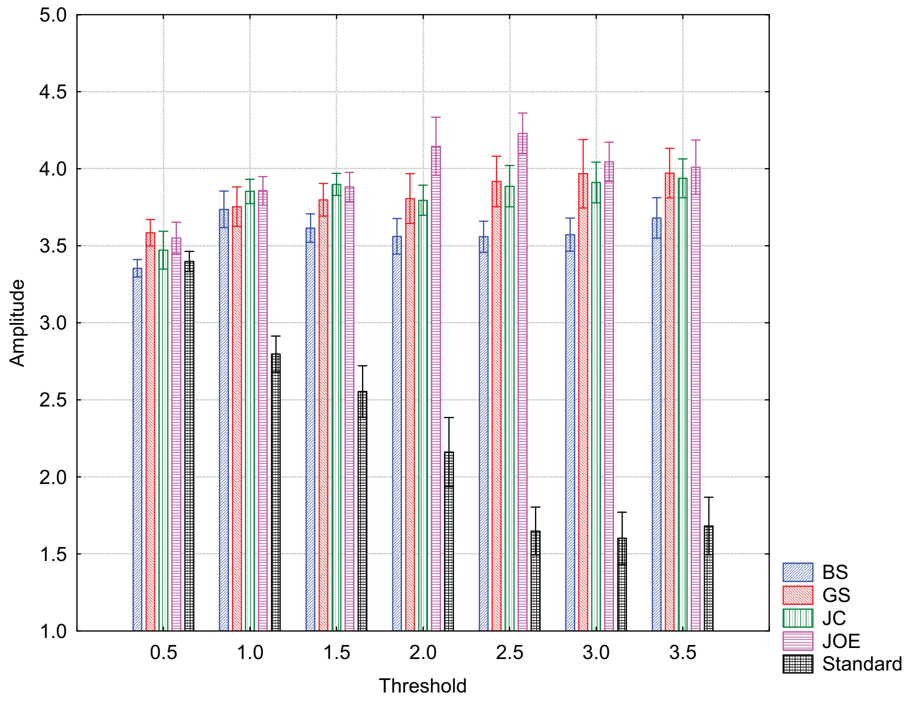


Figure 11. The standard method is not able to preserve the pelvis oscillation. Vertical bars denote standard error of mean.

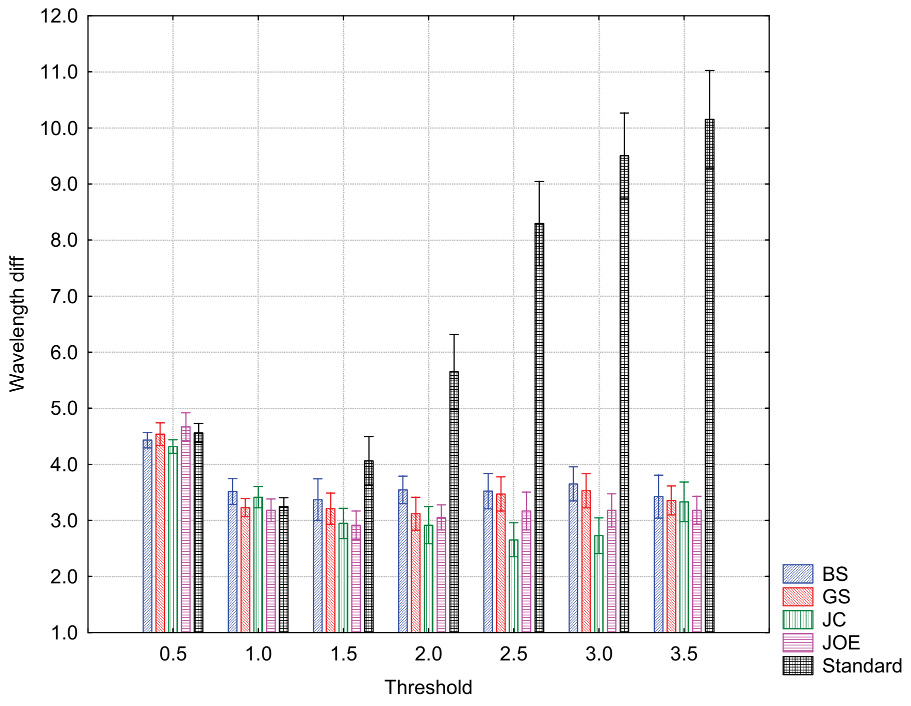


Figure 12. The wavelength difference for each technique over all paths. Vertical bars denote standard error of mean.

Figure 11 typically illustrates the problem of loss of oscillation. Here we see the decrease of amplitude for the standard method for the straight path, whereas the amplitude of the other methods remains intact.

6.3.3. Wavelength.

We found an effect of technique on the absolute difference of the wavelength of the generated animations and recorded motions ($F(4, 156) = 317.35$, $p < 0.001$).

Overall, the path abstraction techniques result in motions whose wavelength is closer to natural values than the standard method (all $p < 0.001$), as is illustrated in Figure 12. For the standard method the wavelength decreases as the threshold increases for straight and low curvature paths. In contrast, the wavelength for the path abstraction techniques remains stable and close to natural values. For the high curvature path we see different results; in contrast to the straight and low curvature path the path abstraction techniques do not perform significantly better.

We observed that the standard deviation of amplitude and wavelength for all path abstractions differs from the standard deviations of recorded motions. However, the standard deviation of the wavelength of the standard method continuously increases as the threshold increases, whereas for the path abstraction it remains stable for all thresholds.

7. CONCLUSION

We investigated two problems that arise when combining path planners and motion synthesizers.

We have performed an evaluation of three different distance metrics on a quantitative as well as qualitative level. We have shown that each metric has its advantages and disadvantages. The joint-angle metric results in graphs that generate animations with the least path deviation. In case of path planning in highly constrained areas, this is a nice property, because collisions with obstacles should be avoided. Second, the search on a graph generated by the joint-angle metric is the fastest, which is particularly useful in interactive applications. The point-cloud based graph results in the slowest search, yet in the least foot skating. Unfortunately, the path deviation is the highest. The PCA metric results in a faster synthesis and a lower path deviation compared to the point-cloud metric. It introduces less foot skating than the joint-angle metric, yet more path deviation. It is also slightly slower, but one does not need to set weights. Following these guidelines, an animator can make a better choice in selecting the right metric for the animation he wants to create. We have also shown that in some cases a more complex graph may be constructed with a higher threshold, while not sacrificing blend quality. Especially at lower metric distances, humans do not significantly grade these blends as being less natural.

We have also seen that defining what the path of the character means is very important. Like many motion synthesizers, the standard version of the motion graph uses the pelvis trajectory as an approximation of the path of the character. The standard motion graph method induces significantly more foot skating. Also the levels of the amplitude and wavelength of the oscillation show that the standard method is not able to preserve the oscillation of the pelvis. This translates to a far less natural looking animation for the standard method in comparison to the path abstraction techniques. Of the different abstraction techniques, the joint combination method is the easiest to implement, but it gives less good results than other path abstraction techniques. The results could be improved by further investigating possible joint combinations. The Joint Extrema method is useful because there are no parameters that need to be set. However, this method will only work for motions where a clear oscillation is present. The B-spline-based method might suffer from corner cutting. This obviously depends on the placement of the knots. Overall, we found that the Gaussian smoothing filter gives the best results for foot skating and oscillation preservation. In the accompanying demo video we show different results for both the standard method and the Gaussian smoothing technique.

8. FUTURE WORK AND CHALLENGES

Many parameters influence the resulting behavior of a metric. The performance of the metrics for foot skating and path deviation may be affected by choosing other parameter values. Also, some artifacts in the synthesized animations can be corrected in post-processing. For example, high path deviation can be solved in a later stage by using a path editing technique [61]. Foot skating can also be removed [66] as a post-processing step. However, adapting motions may introduce other unnatural artifacts. We plan to evaluate metrics for other actions than locomotion, such as fighting, manipulation, dancing and so on. In these cases other evaluation measures would be important, such as physical correctness of jumps or end-effector position. The limitation of our path abstraction techniques is that we only consider planar locomotion. For other types of (loco)motions, the oscillation will be different or even absent. Most of the path abstraction techniques we presented might therefore not work anymore. Also, our database consisted of animations recorded from a single person. It might be interesting to incorporate humans of varying gender and height.

Many challenges arise in combining path planning and animation. First of all, the search space for planning full body movement through an environment is very large. Decoupling of path planning and animation is already a rigorous step in which one ignores many dependencies between the path planning and body movement. This decoupling introduces many problems. As we already have seen, there is no compatibility between the path of a path planner and the motion of a character. What part of the character needs to follow the path? Such questions need to be answered. The determined path can have a big influence on the required animation. When the path planner determines a path up a flight of stairs other animations are needed than when the path remains planar.

Every animation technique has only a certain space of motions it can generate. Ideally, a path planner should take those restrictions into account. For example, some path planners are able to cope with dynamic obstacles such as other characters. To avoid dynamic obstacles we need an animation system that is highly responsive such that the direction of character animation can be instantly adapted. When an animation system has a low responsiveness, dynamic obstacles might not be avoided. A path planner can also augment a generated path with additional information to assist the animation system. For example, clearance information can be incorporated into the path. This allows an animation system to select animations that are collision-free.

Motion parameterization (see "Related Work" section) offers a higher level of control than motion concatenation. In highly constrained environments, for example a narrow passage, the character might need to follow the path exactly. For a highly situated walk engine, a concatenation of parameterized walk cycles followed by editing the abstracted pelvis path might be a promising approach.

Nevertheless, we expect a tighter coupling between path planners and motion synthesis techniques to generate natural motions in games and simulations.

ACKNOWLEDGEMENTS

This research has been supported by the GATE project, funded by the Netherlands Organization for Scientific Research (NWO) and the Netherlands ICT Research and Innovation Authority (ICT Regie). Part of this research has been funded by the Metaverse1 project.

REFERENCES

1. Geraerts R, Overmars MH. The corridor map method: a general framework for real-time high-quality path planning. *Computer Animations and Virtual Worlds* 2007; **18**(2): 107–119.
2. Kovar L, Gleicher M, Pighin F. Motion graphs. In *SIGGRAPH*, ACM Press, New York; 2002; 473–482.
3. Hart PE, Nilsson NJ, Raphael B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics* 1968; **4**: 100–107.
4. Barraquand J. Automatic motion planning for complex articulated bodies. 1991. *Technical Report*, Paris Research Laboratory.
5. Connolly CI, Grupen RA. Harmonic control. In *IEEE International Symposium on Intelligent Control*, 1998; 503–506.
6. Latombe J-C. Robot Motion Planning. Kluwer: Heidelberg, Germany, 1991.
7. Kuffner JJ, LaValle SM. RRT-connect: an efficient approach to single-query path planning. In *IEEE International Conference on Robotics and Automation*, 2000; 995–1001.
8. Kavraki LE, Švestka P, Latombe J-C, Overmars MH. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation* 1996; **12**: 566–580.
9. Barraquand J, Kavraki LE, Latombe JC, Li TY, Motwani R, Raghavan P. A random sampling scheme for path planning. *International Journal of Robotics Research* 1997; **16**: 759–774.
10. Geraerts R, Overmars MH. Creating high-quality paths for motion planning. *International Journal of Robotics Research* 2007; **26**: 845–863.
11. Kamphuis A, Overmars MH. Finding paths for coherent groups using clearance. In *Eurographics/ACM SIGGRAPH Symposium on Computer Animation*, 2004; 19–28.
12. Pettré J, de Heras Ciechomski P, Maïm J, Yersin B, Laumond J-P, Thalmann D. Real-time navigating crowds: scalable simulation and rendering. *Computer Animation and Virtual Worlds* 2006; **17**: 445–455.
13. Karamouzas I, Geraerts R, Overmars MH. Indicative routes for path planning and crowd simulation. In *The Fourth International Conference on the Foundations of Digital Games*, 2009; 113–120.
14. Karamouzas I, Heil P, van Beek P, Overmars MH. A predictive collision avoidance model for pedestrian simulation. In Motion in Games, Second International Workshop, MIG 2009, Zeist, The Netherlands, November 21-24, 2009. Volume 5884 of Lecture Notes in Computer Science, Egges A, Geraerts R, Overmars MH (eds). Springer-Verlag: Berlin, Heidelberg, 2009; 41–52.
15. Boulic R. Relaxed steering towards oriented region goals. *Lecture Notes in Computer Science 5277, MIG 2008*, 2008; 176–187.
16. Singh S, Kapadia M, Faloutsos P, Reinman G. SteerBench: a benchmark suite for evaluating steering behaviors. *Computer Animation and Virtual Worlds* 2009; **20**: 533–548.
17. van Welbergen H, van Basten BJH, Egges A, Ruttkay Zs, Overmars MH. Real Time Animation of Virtual Humans: A Trade-Off Between Naturalness and Control. Eurographics Association: Munich, 2009; 45–72.
18. Multon F, France L, Cani-Gascuel MP, Debunne G. Computer animation of human walking: a survey. *The Journal of Visualization and Computer Animation* 1999; **10**(1): 39–54.
19. Boulic R, Thalmann NM, Thalmann D. A global human walking model with real-time kinematic personification. *The Visual Computer* 1990; **6**(6): 344–358.
20. Lee J, Chai J, Reitsma PSA, Hodgins JK, Pollard NS. Interactive control of avatars animated with human motion data. In *SIGGRAPH*, ACM Press, New York, 2002; 491–500.
21. Arikan O, Forsyth DA. Interactive motion generation from examples. In *SIGGRAPH*, ACM Press, New York, 2002; 483–490.

22. Wiley DJ, Hahn JK. Interpolation synthesis of articulated figure motion. *IEEE Computer Graphics Application* 1997; **17**(6): 39–45.
23. Park SI, Shin HJ, Shin SY. On-line locomotion generation based on motion blending. In *SCA 02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, New York, 2002; 105–111.
24. Kwon T, Shin SY. Motion modeling for on-line locomotion synthesis. In *SCA 05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, New York, 2005; 29–38.
25. Heck R, Gleicher M. Parametric motion graphs. In *I3D 07: Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games*, ACM Press, New York, 2007; 129–136.
26. So CKF, Baciu G. Entropy-based motion extraction for motion capture animation: motion capture and retrieval. *Computer Animated Virtual Worlds* 2005; **16**(3–4): 225–235.
27. Tang JKT, Leung H, Komura T, Shum HPH. Emulating human perception of motion similarity. *Computer Animated Virtual Worlds* 2008; **19**(3–4): 211–221.
28. Chua PT, Crivella R, Daly B, *et al*. Training for physical tasks in virtual environments: Tai chi. In *VR 03: Proceedings of the IEEE Virtual Reality 2003*, IEEE Computer Society, Washington, DC, 2003; 87.
29. Ikemoto L, Arikan O, Forsyth D. Quick transitions with cached multi-way blends. In *I3D 07: Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games*, ACM, New York, 2007; 145–151.
30. Egges A, Molet T, Magnenat-Thalmann N. Personalised real-time idle motion synthesis. In Pacific Graphics. IEEE Computer Society: Washington, DC, 2004; 121--130.
31. Forbes K, Fiume E. An efficient search algorithm for motion data using weighted PCA. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2005.
32. Li L, McCann J, Faloutsos C, Pollard N. Laziness is a virtue: motion stitching using effort minimization. In *Short Papers Proceedings of EUROGRAPHICS*, 2008.
33. Rose C, Guenter B, Bodenheimer B, Cohen MF. Efficient generation of motion transitions using spacetime constraints. In *SIGGRAPH*, ACM Press, New York, 1996; 147–154.
34. Matsunaga M, Zordan VB. A dynamics-based comparison metric for motion graphs. In *Computer Graphics International (CGI)*, 2007.
35. Wang J, Bodenheimer B. An evaluation of a cost metric for selecting transitions between motion segments. In *SCA 03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Eurographics Association, Aire-la-Ville, Switzerland, 2003; 232–238.
36. Safonova A, Hodgins JK. Analyzing the physical correctness of interpolated human motion. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ACM Press, New York, 2005; 171–180.
37. Reitsma PSA, Pollard NS. Perceptual metrics for character animation: sensitivity to errors in ballistic motion. *ACM Transactions on Graphics* 2003; **22**(3): 537–542.
38. Ren L, Patrick A, Efros AA, Hodgins JK, Rehg JM. A data-driven approach to quantifying natural human motion. *ACM Transactions on Graphics* 2005; **24**(3): 1090–1097.
39. Reitsma PSA, Pollard NS. Evaluating motion graphs for character animation. *ACM Transactions on Graphics* 2007; **26**(4): 18.
40. Choi MG, Lee J, Shin SY. Planning biped locomotion using motion capture data and probabilistic roadmaps. *ACM Transactions on Graphics* 2003; **22**(2): 182–203.
41. Sung M, Kovar L, Gleicher M. Fast and accurate goal-directed motion synthesis for crowds. In *Proceedings of the 2005 Symposium on Computer Animation*, ACM, New York, 2005; 291–300.
42. Lau M, Kuffner JJ. Behavior planning for character animation. In *Proceedings of the 2005 Symposium on Computer Animation*, ACM, New York, 2005; 271–280.
43. Lau M, Kuffner JJ. Precomputed search trees: planning for interactive goal-driven animation. *Proceedings of ACM SIGGRAPH*, 2006; 299–308.
44. Srinivasan M, Metoyer RA, Mortensen EN. Controllable real-time locomotion using mobility maps. *Proceedings of Graphics Interface*, 2005. 51–59.
45. Pettré J, Laumond J-P, Siméon T. A 2-stages locomotion planner for digital actors In *Proceedings of the 2003 Symposium on Computer Animation*, Eurographics Association, Aire-la-Ville, 2003; 258–264.
46. Pettré J, Simeon T, Laumond JP. Planning human walk in virtual environments. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2002; 3048–3053.
47. Kamphuis A, Pettré J, Overmars M, Laumond J-P. Path finding for the animation of walking characters. *Poster Proceedings of Eurographics*, 2005; 8–9.
48. Safonova A, Hodgins JK. Construction and optimal search of interpolated motion graphs. In *ACM SIGGRAPH 2007 Papers*, 2007; 106.
49. Treuille A, Lee Y, Popović Z. Near-optimal character animation with continuous control. In *ACM SIGGRAPH 2007 Papers*, 2007; 7.
50. Yoshida E, Laumond JP, Esteves C, Kanoun O, Sakaguchi T, Yokoi K. Whole-body locomotion,

manipulation and reaching for humanoids. In *Motion in Games*, Springer: Heidelberg, Germany, 2008; 221.
51. Zhang L, Pan J, Manocha D. Motion planning and synthesis of human-like characters in constrained environments. In *Motion in Games 2009, Volume 5884 of Lecture Notes in Computer Science*, Springer: Heidelberg, Germany, 2009; 138–145.
52. Lamouret A, van de Panne M. Motion synthesis by example. In Computer Animation and Simulation, Springer-Verlag New York, Inc.: New York, 1996; 199–212.
53. Wang J, Bodenheimer B. Computing the duration of motion transitions: an empirical approach. In *2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Grenoble, France, August 2004; 337–346.
54. Ikemoto L, Arikan O, Forsyth DA. Knowing when to put your foot down. In *Proceedings of the Symposium on Interactive 3D Graphics and Games*, ACM Press, New York, 2006; 49–53.
55. Saunders JB, Inman VT, Eberhart HD. The major determinants in normal and pathological gait. *The Journal of Bone and Joint Surgery* 1953; **35**(3): 543.
56. Grassia FS. Practical parameterization of rotations using the exponential map. *Journal Graphics Tools* 1998; **3**(3): 29–248.
57. OGRE. *Open Source 3D Graphics Engine Website*. Available at: www.open3d.org [Accessed on July 2, 2008].
58. YoYo Games. *Game Maker Website*. Available at: www.yoyogames.com/make [Accessed on July 2, 2008].
59. Hodgins JK, O'Brien JF, Tumblin J. Perception of human motion with different geometric models. *IEEE Transactions on Visualization and Computer Graphics* 1998; **4**(4): 307–316.
60. Hodgins JK, Wooten WL, Brogan DC, O'Brien JF. Animating human athletics. In *SIGGRAPH*, ACM Press, New York, 1995; 71–78.
61. Gleicher M. Motion path editing. In *I3D '01: Proceedings of the 2001 Symposium on Interactive 3D Graphics*, ACM, New York, 2001; 195–202.
62. Arechavaleta G, Laumond J-P, Hicheur H, Berthoz A. The nonholonomic nature of human locomotion: a modeling study. *International Conference on Biomedical Robotics and Biomechatronics*, Pisa, Italy, 2006; 158–163.
63. Lowe DG. Organization of smooth image curves at multiple scales. *Proceedings of 2nd ICCV*, 1988; 558–567.
64. Rosin P. Representing curves at their natural scales. *Patent Recognition* 1992; **25**(11): 1315–1325.
65. van Basten BJH, Egges A. Evaluating distance metrics for animation blending. In *Proceedings. of the 4th International Conference on Foundations of Digital Games*, ACM, New York, 2009; 199–206.
66. Kovar L, Schreiner J, Gleicher M. Footskate cleanup for motion capture editing. In *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ACM, New York, 2002; 97–104.

AUTHORS' BIOGRAPHIES

Ben van Basten is a PhD candidate at the Games and Virtual Worlds group at Utrecht University. He received his MSc in Geometry, Imaging and Virtual Environments (GIVE) in 2005 from the same university. His research interest include animation and motion planning.

Arjan Egges is an Assistant Professor at the Games and Virtual Worlds group in the Department of Information and Computing Sciences, Utrecht University in the Netherlands. He obtained his PhD at MIRALab-University of Geneva, Switzerland on the topic of emotion and personality models, in combination with automatically generated face and body motions using motion capture data. His current research focuses on the integration of motion capture animation with navigation and object manipulation tasks, as a part of the Dutch funded GATE project. Furthermore, he heads the motion capture lab and he teaches several courses related to games and computer animation. Arjan is also an associate editor of the Computer Animation and Virtual Worlds journal published by Wiley and he is one of the co-founders of the annual Motion in Games conference.

Roland Geraerts completed his PhD degree in Computer Sciences at Utrecht University in 2006. His dissertation dealt with a comparison and analysis of sampling-based motion planning techniques, in particular variants of the Probabilistic Roadmap Method. In addition, he also studied quality aspects of paths and roadmaps. Since 2006, he is continuing his work as Assistant Professor and focuses on path planning and crowd simulation in games.