The step space: example-based footprint-driven motion synthesis

By B. J. H. van Basten*, P. W. A. M. Peeters and A. Egges

Especially in a constrained virtual environment, precise control of foot placement during character locomotion is crucial to avoid collisions and to ensure a natural locomotion. In this paper, we present the step space: a novel technique for generating animations of a character walking over a set of desired foot steps in real-time. We use an efficient greedy nearest-neighbor approach and warp the resulting animation such that it adheres to both spatial and temporal constraints. We use smart aligning and inverse kinematics to ensure spatial constraints and time warping for temporal constraints. We will show that our technique can generate realistic locomotion animations very efficiently even though we impose many constraints on the animation. Copyright © 2010 John Wiley & Sons, Ltd.

KEY WORDS: motion synthesis; speed warping; locomotion; footprint; motion editing

Introduction

Computer animation plays a very important role in games and simulations. In modern games, characters constantly walk through virtual environments. In many cases, especially indoors, this environment may be highly constrained. In the proximity of tables and chairs or in narrow corridors it is crucial that there are no collisions of the character with walls or furniture. In this case, a path-based locomotion controller is not sufficient since actions such as sidestepping, taking smaller steps, stepping backwards, and so on need to be taken. We propose to synthesize character locomotion based on the actual placement of footsteps in the environment. This problem is also known as stone stepping: given a set of query steps, called a foot plan, that contains spatial and temporal constraints of foot placements, generate an animation that adheres to these constraints. Especially in cases where a manipulation is involved (like opening a door), it is impractical to use the root trajectory as a controller for the locomotion. The trajectory of the root during such manipulation can be very complex. As such, footprint-driven animation is already supported in many 3D modeling packages, such as 3D Studio Max\(^1\), showing that footprints provide an intuitive means for creating an animation. Here, the user can define individual foot steps after which a procedural animation system generates locomotion over these steps. Unfortunately, this system is not very versatile, for a slightly more complicated footprint will result in self-intersections or unpredictable animations, as can be seen in Figure 1, where we placed a right foot placement further to the left than regular.

When combining locomotion with other tasks, simply providing a path is an underspecification of the task we want the character to perform. Also, foot placement over many steps is the key determinant of dynamic stability\(^2\). So, foot placement will provide us with information about the balancing of the upper body that can be very useful in case one wants to modify the upper body motion.

Another advantage of planning foot steps is that it reduces the dimensionality of the generic motion planning problem. Instead of planning all degrees of freedom of an articulated body in a high-dimensional configuration space, a planner can now simply plan foot placements over which an animation can be synthesized. We will show that by separating the locomotion planning into footstep planning and animation synthesis, a very efficient locomotion engine can be constructed based on a low-dimensional parameter search space. This paper concentrates on the animation synthesis of footsteps and can be combined with a basic footstep planner as proposed in Reference 3.

*Correspondence to: B. J. H. van Basten, Center for Advanced Gaming and Simulation, Utrecht University, the Netherlands. E-mail: basten@cs.uu.nl

Copyright © 2010 John Wiley & Sons, Ltd.
Related Work

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. An excellent survey on generating locomotion has been written by Multon et al.\(^4\). 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. For example, Boulic et al.\(^5\) present a model for human walking based on biomechanical models. In general, procedural techniques offer high control over the animation, but tend to be unnatural and only useful in particular domains, such as running\(^6\). *Physics-based* techniques simulate locomotion using dynamics and physical properties of the body. These techniques yield realistic animations by applying realistic torques on the joints, but offer less control than procedural techniques and can be computationally expensive. A third class is comprised by *example-based* approaches. Existing motions are reused to generate a clip of locomotion. Basically, there are two main example-based techniques. *Motion concatenation* techniques stitch clips of motion together\(^7\). *Motion parameterization* techniques interpolate between existing motions to generate motions corresponding to a specific parameter\(^8\) 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\(^9\).

Not many techniques offer exact foot placement control. Often, the global pelvis motion is determined first and the leg motion is adapted using inverse kinematics. Van de Panne\(^3\) present a procedural space–time based approach. A physics-based optimizer determines the COM trajectory after which inverse kinematics is used to determine the leg motion. This technique was further extended for quadruped locomotion\(^9\) by Torkos and Van de Panne. Chung and Kahn\(^10\) present a procedural hierarchical system that generates locomotion over footprints laid over uneven terrain. Like Boulic’s technique\(^5\), their motion control is based on biomechanical principles. Coros et al.\(^11\) present a physics-based controller that is aware of a footplan and tries to follow it as closely as possible. Wu et al.\(^12\) present a procedural technique that determines the trajectory of the foot and the COM and enrich the resulting motion with motion capture. An optimization technique is employed to enforce the COM to follow the desired trajectory. Choi et al.\(^13\) build up a roadmap by sampling valid stances of the biped figure. Two stances are connected if they can be connected with an adapted (portion of a) motion clip. If this requires too large transformations or will result in collisions the connection fails. Hierarchical displacement mapping is then used to retarget the input motion to the target footprints. Unfortunately, because a roadmap has to be built in a pre-processing step, this technique only works in fixed environments. Instead of adjusting entire motions, we concatenate single steps. Treuille et al.\(^14\) concatenate entire walking cycles. They use a reinforcement learning technique to select a near-optimal step in terms of path deviation and transition smoothness. This technique.
does not offer exact foot placement control. Oshita\textsuperscript{15} presents an animation tool that selects a specific transition interpolation method for transitions between two motions based on their support phase. Because a foot step itself needs to be represented by many parameters, using a parameterization technique such as Reference \textsuperscript{8} for foot step animations might not yield good results. The number of recorded motions grows exponentially with the number of parameters, so building up a sufficiently covered parameter space will take many samples. Even then there is no guarantee that the resulting motion will end up exactly at the desired foot position, since blending of orientations does not yield a linear path through the parameter space.

We propose a local concatenation approach, based on a database of example steps. Obviously, the query steps will generally not be in the database, therefore we will also present an alignment technique to minimize foot positioning errors. Furthermore, we employ warping to ensure that the spatial (IK) and temporal constraints are met. Most methods plan globally over an entire foot plan and therefore are computationally expensive. We aim at a local approach, resulting in a highly efficient algorithm.

**Building the Step Space**

In this section we will explain how we represent steps and how they are extracted automatically from motion capture data to provide a database of example steps.

**Step Extraction**

During the pre-processing stage, we extract steps from recorded motions. In order to extract individual steps, we need to determine the moments the feet are planted. Determining foot stances is, in general, not trivial due to noise and retargeting errors. However, determining foot downs in locomotion is possible. The foot step detector needs to be precise in order to get a decent transition from one step to the other. We use a height and velocity-based foot step detector\textsuperscript{16}.

Walking is generally distinguished from running in that there is always one foot on the floor and during a brief phase between swings, both feet are in contact with the floor. Therefore it is straightforward to determine the stances belonging to one step.

**Step Representation**

We consider a step to be a displacement of one foot and we represent a step using the 10 parameters that are shown in Figure 2. In this example, a left step, we define the swing foot as the left foot and the right foot as the supporting foot. In our (planar) representation we consider six spatial parameters and four temporal parameters.

We express the spatial parameters in the coordinate frame determined by the supporting foot. The origin of this coordinate frame is placed on the ball of the supporting foot and the \( x \)-axis is parallel to the vector from the ball to the toe. Every foot step consists of two placements \( f_1 \) and \( f_2 \). We represent both placement \( f_1 \) and \( f_2 \) by a local position and orientation \((x_1, y_1, \theta_1)\) and \((x_2, y_2, \theta_2)\). We denote the world position of the foot placements of the swing foot as \( p(f_1) \) and \( p(f_2) \) and the world position of the supporting foot as \( p_{\text{sup}} \). The orientation of a foot placement is the angle between the ball–toe vector of that placement and the \( x \)-axis. Because a foot placement spans multiple frames, we take the average of these parameters over the entire stance.

The temporal parameters are the stance durations \( t_{\text{stance}} \) of both placements, the swing time \( t_{\text{swing}} \) and the stance duration of the supporting foot \( t_{\text{sup}} \). Note that not all parameters are independent. For example, there is a dependency between the swing time and the stance duration of the supporting foot, although they are generally not the same.

Once all parameters have been determined we store the steps in a datastructure that we define as the step space, \( S \). For each step, we also store a pointer to its previous step in the original recording. One could
consider using a kD-tree to store the steps. In our case, the number of samples is relatively small compared to the dimensions of the space, so the speed improvement for querying is negligible.

**Step Synthesis**

In this section we will elaborate on creating animations based on the previously discussed step space. We assume the existence of a foot plan \( F \), comprised by a sequence of \( i \) steps in the same representation as depicted in Figure 2.

**Search Strategy**

For each query step \( F_i \) from our foot plan \( F \), we select the weighted nearest-neighbor in our step space \( S \). We adapt the foot placement \( f_1 \) and the placement of the supporting foot of the next query step \( F_{i+1} \) such that it corresponds to the end pose of the previous step. This reduces the drifting effect. In order to get a smooth transition between the steps we do not use standard posture distance metrics as described in Reference 16, but simply \( d(S_a, S_b) = |(x_{S_a}^2, y_{S_a}^2) - (x_{S_b}^1, y_{S_b}^1)| \) where \( S_a \) and \( S_b \) are two consecutive steps selected from the step space \( S \). Informally, this metric compares the distance of \( f_2 \) of \( S_a \) to the supporting foot of \( S_a \) to the distance of \( f_1 \) of \( S_b \) to the supporting foot of \( S_b \) and basically compares the width of two consecutive steps. Our assumption is that the steps in the foot plan \( F \) are more or less natural and realistic and when two consecutive selected steps from \( S \) resemble the two corresponding consecutive steps from footplan \( F \) they will yield a smooth transition. Note that this minimized metric does not take upper body motions into account. We have observed that blending artifacts on the upper body are visually less bothering.

We have investigated a branch-and-bound algorithm as is done in Reference 7. This looks ahead \( n \) steps and then selects (a part of) the sequence of steps with minimum cost. Unfortunately, the number of steps that need to be considered (and hence, the computation time) grows exponentially. We have found that a greedy approach that only considers one step at a time gives very similar results and is much faster.

Because the parameter space needs to be quite dense, we want to take as many steps into account. The only condition for a step to be neglected in the search process is when its previous step in the original animation is not on the same side as the previous query step, or when it does not have a previous step at all. If this is the case it is not possible to ease-in this step from the previous step, for there is no blend window.

**Step Synthesis**

Let \( S_a \) and \( S_b \) be two consecutive steps selected from the step space \( S \) based on the query steps \( F_a \) and \( F_b \) from footplan \( F \). For simplicity, we will assume that \( S_a \) is a different sided step than \( S_b \). When concatenating two steps, a naive approach would be to align the vector \( \vec{v}_1 = (p(f_2) - p_{sup}) \) from the supporting foot world position \( p_{sup} \) of \( S_a \) to the world position of the second placement \( p(f_2) \) of \( S_a \) with the vector \( \vec{v}_2 = (p_{sup} - p(f_1)) \) from the world position of the first placement \( p(f_1) \) of \( S_b \) to the world position of the supporting foot \( p_{sup} \) of \( S_b \) (see Figure 3). In case \( S_a \) and \( S_b \) are same-sided steps then \( \vec{v}_2 = (p(f_1) - p_{sup}) \).

Let \( \text{prev}(S_b) \) be the previous step of \( S_b \) in its original recording, \( \text{prev}(S_b) \) is the same side as \( S_a \) and therefore we can ease-out from \( S_a \) and ease-in to \( \text{prev}(S_b) \) to get a smooth transition from \( S_a \) to \( S_b \). Obviously we can not blend during double support periods: this will result in foot skating. Therefore, we blend during the swing phase, when only one foot is planted. We linearly

![Figure 3. A straightforward way of aligning steps is by aligning the vectors \( \vec{v}_1 \) and \( \vec{v}_2 \).](image-url)
interpolate to find the corresponding frames in the swings of $S_a$ and $\text{prev}(S_b)$. Note that because we only blend during the swing of the foot, the stance foot remains fixed and there is no footskating at all. Implicitly we are using the stance foot as a root.

We observed that using this alignment strategy resulted in drifting for larger foot plans. We will now propose another alignment strategy that utilizes blending to end up near the query foot placement more accurately. Instead of aligning $\vec{e}_2$ with the previous step vector $\vec{e}_1$ we align it with the query step vector $\vec{v}_q$. $\vec{v}_q$ is defined as the vector from the world position of the supporting foot $p_{\text{sup}}$ of $S_a$ to $p(f_2)$ of $F_a$. This will result in a rotational movement about the supporting foot and will minimize the distance to the query foot placement (see Figure 4). Allowing a pivoting movement has its influence on the search strategy. Because the animation of $S_a$ will blend towards the begin pose of $S_b$, the begin placement $f_1$ of $S_b$ is fairly important. One would like to minimize $d = |\vec{e}_2| - |\vec{v}_q|$. Therefore, we take this difference $d$ of vector length into account in our search strategy as an additional soft constraint with weight $w_d$.

Warping

Due to the dimensionality of the problem the constraints will never be exactly satisfied. In this section, we describe the warping techniques used to ensure that the spatial and temporal constraints are met.

Spatial Warping

In order to eliminate the spatial errors of the foot placements we adapt the animation such that the foot lands exactly on the foot placements of the foot plan. Inverse kinematics algorithms can be quite complex and involve computationally expensive numerical optimizations. Fortunately, there exist closed-form solutions for specific articulated structures such as (7 DoF) human arms or legs. We use an efficient technique from Kovar et al.17. Although this technique was originally used for footskate removal, it is trivial to replace the footskate spatial constraints with our spatial constraints. For efficiency, we implemented a threshold such that a constraint is only enforced when the error is large enough.

Speed Warping

Because $S_a$ and $\text{prev}(S_b)$ do not need to have the same velocity or step size, blending and aligning these steps to create a smooth transition from $S_a$ to $S_b$ can introduce pelvis speed artifacts. The speed of the pelvis during locomotion reaches its maximum during double support and its minimum during the swing18, as can be seen in Figure 5. Here we see the speed of the pelvis in the upper graph plotted against the frame number. The two rows of horizontal bars indicate the left (upper row) and right (lower row) foot downs corresponding to the same frames. We observed that this profile is similar for other locomotion types, such as backwards walking and sidestepping.

In order to resolve the speed errors caused by blending and aligning we will warp the speed of the pelvis such that it complies to a standard speed profile. It is tempting to think that one can derive a standard speed profile from biomechanics. Inman5 provides a normalization formula that shows a linear relationship between step length and step frequency, so on is able to derive the average speed from this formula. One can also estimate the desired speed using the Froude ratio19 that is based on an inverted pendulum model. Unfortunately, we have observed that all these measures show great variation and are not suitable to use as input for an adaptation technique.

Therefore, we will estimate the speed profile corresponding to the new animation by fitting cubics through

Figure 4. Aligning $S_b$ towards the query foot placement (red) offers more accuracy.
the original extrema of the speed of the pelvis of the original steps. In order to achieve speed coherence between the individual steps, we filter the original speed extrema before fitting the cubics. The approximated speed profile can easily be enforced by recalculating the time keys of the frames of the original animation. An example of a resulting speed profile can be seen in Figure 6. The original extrema are depicted as green triangles and the cubic fit of the filtered extrema in blue.

**Time Warping**

The resulting steps from the search algorithm might not totally adhere to the temporal constraints. The extent to which the time requirements are met depend on the weight that is set for timing information in the search algorithm. Nevertheless, some correction might be necessary. Because the result of the search algorithm will have the same number of heel-strikes and lift-offs as the query foot plan, a piecewise linear mapping can be constructed. As a result, the resulting animation time \( t_r \) will be warped to \( t_f \) using the following formula:

\[
t_f = y_a + (t_r - x_a) \frac{(y_b - y_a)}{(x_b - x_a)}
\]

where \( (x_a, y_a) \) and \( (x_b, y_b) \) are consecutive pairs of corresponding events such as heel-strikes and toe-offs. Unfortunately, this mapping is not injective. Several \( t_r \) are being mapped one a single \( t_f \). This is because a left toe-off and right heel-strike can occur really close in time or even simultaneously. Fitting a monotone B-spline through the mapping is possible, yet computationally expensive. Therefore, we only allow heel-strikes.

**Results**

Our database consists of 200 steps (both left and right) taken from various motion capture recordings. We also
have constructed the mirrored versions of these steps. For the nearest neighbor search, we use a weight of 1.0 for the spatial parameters \((x_1, x_2, y_1, y_2)\), 0.25 for the angular parameters \((\theta_1, \theta_2)\), 0.02 for the temporal parameters, 0.3 as weight for the distance metric and an additional 0.3 for \(w_d\). All experiments were executed on an Intel Pentium Centrino 2.4 GHz.

We primarily tested on four foot plans. The first example (Figure 7, upper) is a sequence consisting of five steps containing several consecutive left steps. The second foot plan consists of a long zigzagging sequence of 62 steps. This foot plan was derived from an animation generated by a motion graph using a database of another person. The duration of the resulting animation was 39 seconds. The third example (Figure 7, middle) consisted of a 14 steps forming high curvature locomotion. The last example (Figure 7, lower) is a similar example as the one depicted in Figure 1 consisting of eight steps. Here, one step of a straight locomotion is moved sideways. Our method is able to successfully synthesize an animation of 4.8 seconds over this foot plan.

The technique was implemented as a 2-phase process. First, the initial animation was created. Then the spatial, speed, and time warping were applied in the second phase. For all four foot plans, determining the initial animation took 3 milliseconds per step on average \((\sigma = 1.9\) milliseconds). The warping phase took an additional 1.6 milliseconds per step on average \((\sigma = 0.6\) milliseconds). The duration of a resulting step was 0.60 seconds on average \((\sigma = 0.02\) seconds). Our implementation could be optimized further by integrating these two phases.

We compared our aligner to the naive alignment approach and noticed a big improvement in accuracy of the resulting placement of the feet. Obviously, the closer the initial animation is to the foot plan, the less warping is required. The average planar Euclidean distance per placement for our aligner is 3.58 cm, whereas the naive approach yields an average error of 18.33.

**Conclusion and Future Work**

In this paper we propose a new parameter space, the step space, for solving the stone-stepping problem. We concatenate existing steps that are automatically extracted from motion capture data. These steps are blended and aligned in a smart way such that the amount of required warping is minimized. We showed that our technique is fast and generates good animations for such a highly constrained problem. An additional advantage is that our technique is conceptually simple and relatively easy to implement. Looking only one step ahead means that it is easier to integrate our approach into real-time applications where the character may be controlled by a player and thus the entire footplan is not yet known.

Our model currently only works for planar walking motions. We are investigating non-planar walking motions such as walking up a hill or a pair of stairs, as well as other types of locomotion. Also, a more advanced IK technique might improve the resulting animation. At this point we speedwarp by recalculating...
the timekeys of the animation. This might, however, introduce other (balancing) artifacts in the limb movement. We want to investigate if the results improve when we first change the speed of the pelvis itself (and introduce foot skating), and then applying spatial warping on the limbs.

The dimensionality of the stone stepping problem is a crucial point. In order to improve the initial result, one might want to consider parameterizing on the spatial parameters. This might reduce the error and again require less warping than is the case at this point. We also assume that a footstep planner is available to provide the sequence of footsteps. However, this is definitely not an easy part of the problem since footstep placement will depend on many factors such as the goal of the character and constraints in the environment that oblige the character to sidestep or step backward. Finally, perceived realism is very important for games and simulations. We plan to qualitatively evaluate our results by doing user tests and compare our animations to results from other approaches, such as Reference 13.

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).

References


Authors’ biographies:

Ben van Basten is a Ph.D. candidate at the Games and Virtual Worlds group at Utrecht University. He received his M.Sc. in Geometry, Imaging, and Virtual Environments (GIVE) in 2005 from Utrecht University. His research interest includes animation, motion planning, and games.
**Pieter Peeters** is a student in the Master Program Game and Media Technology at Utrecht University and is currently doing an internship at VRLab in Lausanne, Switzerland.

**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 Ph.D. 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. He 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.