Walk This Way: A Lightweight, Data-driven Walking Synthesis Algorithm

Sean Curtis, Ming Lin, and Dinesh Manocha

University of North Carolina at Chapel Hill,
Chapel Hill, NC, USA
{seanc,lin,dm}@cs.unc.edu
http://gamma.cs.unc.edu/Walking

Abstract. We present a novel, biomechanically-inspired, kinematic-based, example-driven walking synthesis model. Our model is ideally suited towards interactive applications such as games. It synthesizes motion interactively without *a priori* knowledge of the trajectory. The model is very efficient, producing foot-skate free, smooth motion over a large, continuous range of speeds and while turning, in as little as 5 μs. We've formulated our model so that an artist has extensive control over how the walking gait manifests itself at all speeds.

1 Introduction

Data-driven motion synthesis (DDMS) has proven the most common motion synthesis approach in games; DDMS allows detailed art direction and the most basic algorithms are efficient and simple to implement. As rendering has improved rapidly in recent years, these simple DDMS algorithms have lagged behind. Research has produced many new algorithms for creating high fidelity motion. These algorithms aren't always suitable for games. Algorithms focused on quality may exceed a game's computing constraints. These algorithms rely on large databases of motion. Generalized DDMS knows nothing about the underlying motion and cannot distinguish between "good" and "bad" motion. Without an understanding of the motion itself, the synthesized motion must be circumscribed by the samples given. These methods are made practical by the ease with which human motion can be captured and processed. For games which choose an animated style, constructing an equivalently large database is logistically infeasible. So, how can games leverage the art-directable advantages of DDMS approaches while staying within run-time computational budgets and production budgets?

In 2008, Gleicher discussed this question and said, "the future of technology for animated characters in interactive systems lies as a hybrid of [DDMS] and algorithmic approaches." [1] For the practice of game development, eschewing an ideal general DDMS in favor of coupling well-crafted data with intelligent models allows a game to better leverage its artistic assets and production manpower.

Given the ubiquity of locomotion in games, applying such a model to walking data can significantly benefit games. We present a novel walking synthesis model which continuously transforms a single walking example consisting of two straight-line walking steps, without any *a priori* knowledge of the trajectory. The straight-line motion is transformed, on-the-fly, to support arbitrary speeds and turning motion. While the approach is based on fundamental biomechanical principles, the core of the model is a set of user-defined functions (defined in the same manner as keyframe function curves.) This gives artists control, not only over the example data, but over how that motion changes with respect to speed and turning.

Organization In Section 2, we briefly discuss related work. We define concepts related to walking and our model in Section 3. In Section 4 we discuss our model in three layers: steady-state walking, mid-stride acceleration, and turning. Finally, we illustrate and discuss the product of our model in Section 5.

2 Related Work

There has been extensive research in *dynamics-based* motion synthesis. Due to space limitations, we refer the readers recent survey [2] and limit our discussion to data-driven, footstep-based, and procedural methods.

Procedural solutions generate locomotion through functions based on underlying principles of biomechanics [3–8]. For each joint, these approaches define a unique parametric function mapping time and gait parameters directly to values of the degrees of freedom. However, many phenomena are difficult to express closed-form expressions, e.g. high-frequency phenomena, asymmetries, and personality.

DDMS solutions generate new motion from a library of motion clips. Blending approaches create new motion by blending multiple motions [9, 10]. Motion graphs [11, 12] operate by creating a graph of motion clips and then perform a search to find a sequence of adjacent nodes that fit a desired trajectory. The transition mechanism can vary widely (including learned controllers [13].) The clips themselves can be parametrically extended to encompass a space of motions [14, 15]. If the database contains human motion capture, the data encodes the subtle nuances and high-frequency phenomena inherent in human motion (e.g. small twitches or irregular motions.) In addition to requiring large databases, these systems operate at clip resolution. Each clip in the sequence is played from beginning to end; changes in the middle are not supported.

There has also been continued work in footstep-driven walking synthesis. Van Basten et al. presented approaches for generating walking based on a pre-defined sequence of foot steps [16, 17]. Their techniques are essentially a DDMS approach in which the clip selection criteria is a function of the footstep properties. Like in our work, they apply motion warping to better satisfy the constraints, but they apply warping only for small deviations.

Techniques for dynamic blending [18, 19] and interactive editing motion data to create new motion styles using various approaches have also been proposed (e.g. [20–22]).

While our approach applies biomechanical principles employed in many of these works, we apply them in a unique fashion with a different goal in mind. Unlike the procedural approaches we do not parameterize the fine details of motion. We rely on the input clip to define those. Instead, we parameterize the coarse, low frequency phenomena such as stride length. Unlike the DDMS approaches listed, we don't rely on a large database of examples. We use a single example and transform it in a well-principled fashion to achieve large changes in the motion. Finally, we synthesize motion on the fly. Our system requires no *a priori* trajectory or footprint knowledge and responds to an arbitrary sequence of velocity changes. This makes it ideal for driving characters in interactive applications.

3 Walking Gait

In this section, we briefly present key concepts and notation used in the balance of the paper. For more detail on human walking, we recommend [23] and [24].

The walking gait is a form locomotion in which at least one foot is always in contact with the ground and, at times, both are. Our system operates on the intervals defined by changes in contact (i.e. when a foot leaves or makes contact with the ground.) We define the beginning of a step when the new support foot makes contact with the ground at *initial contact* (IC). The swing foot lifts off the ground at *opposite toe off* (OT). The swing foot becomes level with the support foot at *feet adjacent* (FA). Finally, the swing foot makes contact at *opposite contact* (OC) which is the IC event for the next step. We denote the times at which these events occur as T_{IC}, T_{OT}, T_{FA}, and T_{OC}, respectively The fraction of the gait in which both feet are in contact is called *double support time* (DS).

3.1 Gait Properties

A walking gait has many properties [24–26]. These properties define the gait and are exhibited at all speeds, albeit with varying values. We parameterize how six of these properties *change* as speed changes into "gait functions": stride frequency, double support time, foot flexion, vertical hip excursion (apex and nadir) and pelvic swivel. The first two are parameterized with respect to speed and the last four are parameterized with respect to stride length.

Together, stride frequency and double support time define the timing of a step. Stride frequency determines the overall duration of the step and DS time determines the amount of time the swing leg spends in contact with the ground.

Given stride frequency and speed, we can easily compute stride length. Foot flexion, vertical hip excursion, and pelvic swivel are all mechanical techniques which increase a biped's effective stride length and smooth the trajectory of the center of mass [24].

Biomechanical literature [24–26] suggests mathematical models for the naturally occuring relationships. Given "realistic" gait functions this transformation approach produces dynamically consistent motion. In games, however, not all characters walk with a strictly realistic gait. To support arbitrary artistic needs, we model these gait functions as user-editable Hermite curves. Thus, our approach applies equally well to a wide range of walking styles, from realistic to cartoon-like.

3.2 Motion and Motion Warping

We represent the configuration of a skeleton as a state vector, $\mathbf{X} \in \mathbb{R}^n$, where the skeleton has n degrees of freedom. Animation arises from varying this vector with respect to time, $\mathbf{X}(t)$. The time-varying data for each degree of freedom is a channel.

We use motion warping [27] to transform an input motion sample into new motion. Specifically, we use a global time warp function, $\tau(t)$, and warp the i^{th} channel of the motion data, $X_i(t)$, with one of two types of warp functions:

- **Offset**: $X_i'(t) = X_i(t) + o(t)$. The original data is simply offset by a time-varying term, $o(t)$.
- **Scale**: $X_i'(t) = s(t) * X_i(t)$. The original data is scaled by a time-varying scale term, $s(t)$.

We model the warp functions with Hermite splines. Each function, $x(t)$, is defined by an ordered list of constraint tuples: $(t_i, x_i, \dot{x}_i)$, such that $t_i < t_{i+1}$. Section 4 provides the details on how specific warp functions are defined.

4 Gait Transformation

Our approach is based on two simple ideas. First, although human locomotion is a dynamically-complex phenomenon, humans solve it consistently and robustly. Second, the gait properties vary smoothly with speed, and these changes can be empirically observed and directly modeled. Thus a model based on human motion capture data can be transformed into valid walking motion by applying the empirically observed gait property models.

Our approach synthesizes new walking motion by transforming an input clip, based on a set of *gait functions*, the current velocity, and the figure's current configuration, $\mathbf{X}$. We transform the motion in two stages. First we apply motion warping to individual channels in the input clip. Second, the root and swing foot are transformed from straight-line motion into turning motion. Finally, an IK-solver computes joint angles for the degrees of freedom in $\mathbf{X}$.

4.1 Offline processing

The *canonical clip*, $C(t)$, is the input motion data representing a walking motion with several properties. The motion consists of two footsteps, walking in

a straight line along the z-axis. The motion is loopable (i.e. the first and last configurations are identical.) The motion begins at the IC event. And, finally, the joint angle representation of the hip, knee and ankle joints are replaced with an inverse-kinematic (IK) representation for foot position and orientation. Creating a clip with these properties is straightforward. The specific details are not within the scope of this discussion; it can be done algorithmically or by an artist. However, we present the details of the IK representation and solver in Section 4.2.

The input state vector for the canonical clip, $\mathbf{X} \in \mathbb{R}^{22}$, is defined as follows: $\mathbf{X} = \big[\mathbf{Root}\ \mathbf{Ankle_R}\ \mathbf{Ankle_L}\ \mathbf{Toes}\big]$, where $\mathbf{Root} \in \mathbb{R}^6$, including three degrees of rotation and translation, $\mathbf{Toes} \in \mathbb{R}^2$, a single rotational degree of freedom for each foot, and $\mathbf{Ankle_R}, \mathbf{Ankle_L} \in \mathbb{R}^7$ consisting of three dimensions of translation and rotation for each ankle joint and one *pole vector* value. The pole vector value is used to determine the direction the knee faces in solving the IK system (see 4.5.)

Although the canonical clip consists of two steps, each step is considered individually. The balance of the paper will focus on a single step and eschew the descriptions of "left foot" and "right foot" in favor of "support foot" and "swing foot"

Not all of the channels in the canonical clip are warped. For example, our system does not change the side-to-side movement of the pelvis, so we do not warp the root's x-translation. Table 1 enumerates the five channels in $C(t)$ which have motion warping applied. The rotation of the support and swing feet are also transformed to faciliate turning (see Section 4.4.)

Table 1. The five warped channels from $C(t)$, the type of motion warp applied and the times of the constraints in each motion warp.

Joint	Channel	Warp Function	Constraint Times
root	ty	Offset	T_{IC}, T_{OC}, t_{MAX}, t_{MIN}
root	tz	Offset	T_{IC}, T_{OC}
root	ry	Scale	T_{IC}, T_{OC}
swing foot	tx	Offset	T_{OT}, T_{OC}
swing foot	tz	Offset	T_{OT}, T_{OC}

The *gait functions*, F, are the set of six functions that parameterize the properties listed in Section 3.1. They relate speed and stride length to the various gait properties. They serve as the basis for defining the warp functions' constraints.

- $F_f(v)$ maps speed, v, to stride frequency.
- $F_{DS}(f)$ maps stride frequency to double-support time, DS.
- $F_{FF}(l)$ maps stride length, $l = vF_f(v)$ to foot flexion, FF.
- $F_{PS}(l)$ maps stride length to pelvic swivel, PS.
- $F_{AP}(l)$ maps stride length to the apex of vertical hip excursion, AP.
- $F_{ND}(l)$ maps stride length to the nadir of vertical hip excursion, ND.

4.2 Steady-state Gait

The steady-state gait arises after maintaining a constant velocity for multiple steps. By definition, this motion is straight-line walking. The parameter values in the gait functions are fully realized in the steady-state gait.

We create the steady-state gait by evaluating the gait functions and using those results to define the specific constraint tuples for the warp functions. Table 2 shows the constraint values for each of the terms in the warp functions. The constraint definitions use the following variables:

L, the average stride length of C(t).
$l = v * F_v(v)$, the stride length for speed v.
$\Gamma = L - l$, the change in stride length.
β_W, the forward bias from the swing foot. Forward bias is the fraction of the distance in front of the back foot towards the front foot at which the pelvis lies. For the swing foot, this value is defined at T_{IC}.
β_P, the forward bias from the support foot, defined at T_{OC}.
$t_{\max}$, the time at which the pelvis is at its peak (usually shortly after T_{FA}.)
$t_{\min}$, the time at which the pelvis is at its nadir (usually shortly after T_{IC}.)

Table 2. Warp function constraints for steady-state gait. Constraint values marked with * are discussed in Sec. 4.2.

Channel	**Warp Term**	**Constraints** $(t_i, x_i, \dot{x}_i)$
time	τ(t)	(0, T_{IC}, *), ($F_{DS}(F_f(v))/F_f(v)$, T_{OT}, *), ($1/F_f(v)$, T_{OC}, *)
root.tz	$o(t)$	(T_{IC}, $(\beta_W - 1)\Gamma$, 0), (T_{OC}, $\beta_P\Gamma$, 0)
root.ty	$o(t)$	(T_{IC}, *, *), ($t_{\max}$, $F_{AP}(l)$, 0), ($t_{\min}$, $F_{ND}(l)$, 0), (T_{OC}, *, *)
root.ry	$s(t)$	(T_{IC}, $F_{PS}(l)$, 0), (T_{OC}, $F_{PS}(l)$, 0)
swing.tx	$o(t)$	(T_{OT}, 0, 0), (T_{OC}, 0, 0)
swing.tz	$o(t)$	(T_{OT}, 0, 0), (T_{OC}, Γ, 0)

The temporal warp function, $\tau(t)$, changes duration to account for changes in stride frequency and changes the proportion of time spent in double support by moving the world time mapping for the T_{OT} event. The tangents of the curve are calculated to produce smooth, monotonically increasing interpolation between constraints. The root.ty channel's warp constraints are somewhat more elaborate. The bobbing is oscillatory. The time and value of the apex and nadir points of the oscillation defines the behavior. To compute the offset and tangent values at T_{IC} and T_{OC}, we logically extend the peak and valley periodically according to the step duration and then infer the value at T_{IC} and T_{OC} from the cubic interpolation provided by the Hermite spline. Straight-line motion does not alter the warp function for the swing foot's x-translation (see Section 4.4.)

The rotation of the feet due to foot flexion is more complex. Both the support and swing foot's flexion changes with stride length (although the changes to the swing foot are more extreme.) We compute "flexion functions", W_{FF} and P_{FF},

for the swing and support foot, respectively. They are defined as Hermite splines with a set of constraint tuples but the values of the functions are applied quite differently.

Traditional motion warping operates on each channel independently. Rotation is a complex operation in which the various degrees of freedom are interrelated. Orientation for the feet is a "world" orientation and, as such, we cannot modify individual channels with predictable results. Changes in foot flexion change the foot's orientation around its *local* x-axis, regardless of the actual orientation of the foot in gait space. The value of the flexion function is interpreted as an amount to rotate the foot around its x-axis. Table 3 defines the foot flexion constraints in steady state.

Table 3. Foot flexion function constraints. The variable θ_X is the ratio of the support foot's flexion value in the data between T_X and T_{OT}. Similarly, ψ_X is the same for the swing foot.

Foot	**Constraints** $(t_i, x_i, \dot{x}_i)$
Support	$(T_{IC},\ F_{FF}(l) * \theta_{IC},\ 0)$, $(T_{OT},\ 0,\ 0)$ $(T_{FA},\ 0,\ 0)$, $(T_{OC},\ F_{FF}(l) * \theta_{OT},\ 0)$, $(T_{TO},\ F_{FF}(l),\ 0)$
Swing	$(\text{-}T_{MS},\ 0,\ 0)$, $(T_{IC},\ F_{FF}(l) * \psi_{OC},\ 0)$, $(T_{OT},\ F_{FF}(l),\ 0)$, $(T_{OC},\ F_{FF}(l) * \psi_{IC},\ 0)$

4.3 Mid-stride acceleration

In interactive applications, the trajectory a character followsm, $\boldsymbol{P}(t)$, is generated dynamically, directly from user controls or in response to user decisions. At any given display frame, motion must be generated without knowing anything about the future trajectory. More particularly, a velocity change can occur at any point during the walking cycle.

The application maintains a logical position of the game character (usually the product of Euler integration.) The motion generated must adhere to this logical, or *ideal* position. Furthermore, the motion must remain physically plausible. Here we define plausibility with two crieteria. First, feet in contact with the ground cannot move. Second, the joint trajectories of the skeleton must maintain C^1-continuity; we do not allow the current configuration to "pop" nor the rate of change of the configuration to change instantly. Our approach for accomodating mid-stride acceleration satisfies both requirements.

We require that our motion follows the ideal position faithfully. However, constraining the center of mass of the character to that point is unrealistic. When walking a straight line, a real human's center of mass oscillates around the ideal position derived by assuming constant average velocity. To account for this, each skeleton plans its motion such that the physical center of mass oscillates around the ideal position, but perfectly aligns every second step.

When the speed changes mid-stride from v_- to v_+, our system changes the warp functions so that the current configuration is preserved and the position of the center of mass aligns with the ideal position at the end of the step. This depends on determing how much time remains in the current step after the speed change.

We apply a heuristic to determine the remaining time of the step, t_R. If the current gait time is before T_{FA}, we assume that the skeleton can move from its current position, p_C, to a position consistent with the full stride length, p_v, inherent in the steady-state gait for v_+. The remaining time is the time required for the center of mass to travel this distance. If current gait time is after T_{FA} it's the world time the v_+ steady-state gait would take to get from t_C to T_{OC}.

$$t_R = \begin{cases} (p_v - p_C)/v_+ & \text{if } t_C < T_{FA} \\ \tau_+^{-1}(T_{OC}) - \tau_+^{-1}(t_C) & \text{if } t_C \geq T_{FA} \end{cases} \tag{1}$$

We define the new warp function, e.g. $o_+(t)$, in terms of the remaining time, the old warp functions, e.g. $o_-(t)$, and the gait functions. The new warp function constraints completely replace the previous set. Table 4 shows the constraints and uses the following values: τ_v is the temporal warp for the steady-state gait at speed v, τ_- is the temporal warp before the speed change, $\Delta t = \tau_-^{-1}(t_C) - \tau_v^{-1}(t_C)$, and l_+ is the expected stride length at T_{OC} (it may be different from the steady-state stride length.) Finally, to guarantee C^1 continuity in the joint trajectories, we introduce new constraints, consisting of the old warp function values and derivatives at the current time.

Table 4. Transient warp function constraints. Constraints with parameter value T_X are only included if $t_C < T_X$.

Channel	**Warp Term**	**Constraints** $(t, x_i, \dot{x}_i)$
time	$\tau_+(t)$	$(\tau_v(t_C) + \Delta t,\ t_C,\ \dot{\tau}_v(t_C))$, $(\ \tau_v(T_{OT}) + \Delta t,\ T_{OT},\ *)$, $(\tau_v^{-1}(t_{OC}),\ T_{OC},\ *)$
root.tz	$o_+(t)$	$(t_C,\ o_-(t_C),\ \dot{o}_-(t_C))$ $(T_{OC},\ p_v,\ 0)$
root.ty	$o_+(t)$	$(t_C,\ o_-(t_C),\ \dot{o}_-(t_C))$, $(t_{MIN},\ F_{ND}(l_+),\ 0)$ $(T_{OC},\ *,\ *)$
root.ry	$s_+(t)$	$(t_C,\ s_-(t_C),\ \dot{s}_-(t_C))$, $(T_{OC},\ F_{PS}(l_+),\ 0)$
swing.tz	$o_+(t)$	$(t_C,\ o_-(t_C),\ \dot{o}_-(t_C))$, $(T_{OC},\ l_+,\ 0)$

4.4 Turning

Our system can further transform the warped motion data to allow the character to turn as it travels, even exhibiting the subtle phenomenon of leaning into the turn.

Humans lean into turns for the same reason that an inverted pendulum would lean inwards while moving along a circular path, centripetal force. As the centipetal force diminishes, human paths straighten out and they stop leaning to the side. We model this using a critically-damped angular spring. We rotate the position of the root around the straight-line along which the motion is defined. Finally, we transform the line, rotating and displacing it, to achieve turning motion.

To compute the leaning, we define a spring with spring coefficient, k. We assume unit mass and require the system to be critically damped giving us the damping coefficient $c = 2\sqrt{k}$. At run time we compute the centripetal acceleration, $a_c = v * \omega = v_+ * \cos^{-1}(< \hat{v}_+, \hat{v}_- >)/\Delta t$, where $\hat{v}_-$, $\hat{v}_+$, Δt are the direction of the old velocity, new velocity and the elapsed time, respectively. We then apply the force to the spring and integrate the state of the spring. The displacement of the spring, θ, is used to rotate the warped position of the root, $Root_w$ around the z-axis to produce the leaning root, $Root_l$

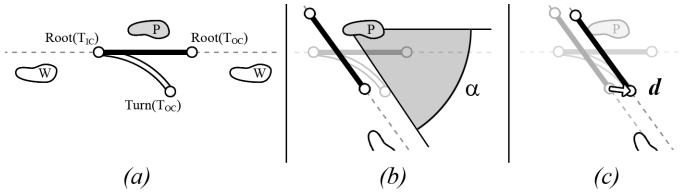

Fig. 1. Illustration of how straight-line motion is transformed to turning. The swing foot is marked W and the support with a P. (a) the initial relationship between straight (black line) and curved (white line) trajectories. (b) the straight-line space is rotated around the support foot α radians to point in the same direction as the final velocity. (c) the space is translated by $\boldsymbol{d}$ to so that the straight-line end point aligns with the ideal position.

To create the effect of turning motion, we apply a principle related to that presented in [8]: we transform the space of the straight-line walk to align with the turning direction. Our solution is different in several respects: first, we do not rotate the full canonical clip. We only transform the root and the swing foot. Second, during double support we perform no transformation at all. Together, these two heuristics guarantee that feet in contact with the ground will not slide.

The final position of the root and swing foot form, $t_C > T_{OT}$, are given by:

$$Root_f(t) = Root_l * R_\alpha + \boldsymbol{d}, \tag{2}$$

$$Swing_f(t) = Swing_w * R_\alpha + \boldsymbol{d}, \tag{3}$$

$$SwingPV_f t = Swing.pv * R_\alpha + \boldsymbol{d}, \tag{4}$$

where R_α is a matrix that performs a rotation of α radians around the y-axis. These turning parameters, α and $\boldsymbol{d}$, are encoded in three more warp-like func-

tions: d_x, d_z and α. For each curve, the initial constraints are $\{(T_{OT}, 0, 0), (T_{OC}, 0, 0)\}$. When velocity direction changes mid-stride, we determine the parameter values at T_{OC} as outlined above and set a constraint at t_C to maintain continuity.

This turning action leads to a particular artifact. By the end of the motion, the support foot is no longer in line with the root and swing foot. This support foot becomes the next swing foot. To maintain continuity its inital position must take into account the misalignment. This is when the warp functions for swing.tx and swing.ry are used. At a step switch they are initialized to have constraints at T_{OT} and T_{OC}. The values at T_{OT} maintain the foot position and orientation from the previous step and at T_{OC} they are zeroed out.

4.5 Inverse Kinematics Solver

We operate on "world" positions and orientations of root and feet in our motion synthesis. We use a simple IK-solver to analytically solve for the joint trajectories. The legs act as a simple two-bone chain with three degrees of freedom at the hip and one at the knee. For a given positions of hip and foot, there are an infinite number of solutions. To resolve this ambiguity, we apply a pole-vector constraint to determine the orientation of the knee. The pole vector is a point in space approximately three body lengths in front of the figure. The time-varying position of this point is extracted from the original motion data. At the default speed, the IK-solver reproduces the canonical clip exactly. As stride length changes, the same pole vector value is used. This keeps the pattern of knee orientation consistent across all speeds.

5 Results and Analysis

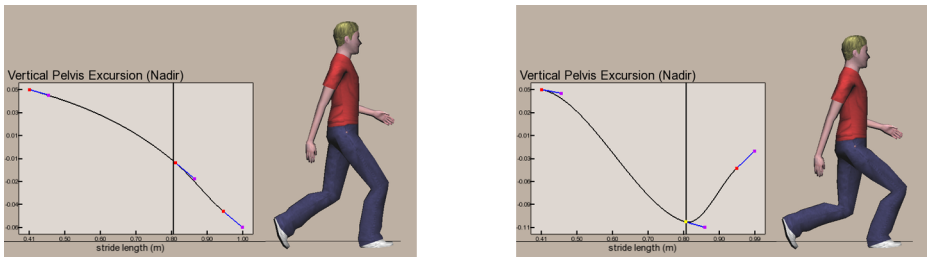

Fig. 2. For a fixed stride length, the vertical position of the pelvis at its nadir has been edited. The figure on the right shows a much lower pelvis in its gait.

Motion synthesis algorithms are best evaluated by viewing the motion they produce. We highlight the features of our algorithm in the accompanying video. We demonstrate the fidelity of the motion following arbitrary trajectories as well as real-time user input. We show the simplicity with which the gait functions

can be modified and the impact that has on the personality of the motion. Finally, we show the scalability of our approach by synthesizing the motion for 800 characters in real-time.

5.1 Performance

We implemented our system in C++ and ran our experiments on an Intel i7 CPU at 2.65 GHz with 6 GB of RAM. Individual skeletons update in 11 μs. For the 800 characters, we updated skeletons in parallel and were able to update all skeletons in less than 4 ms per frame, or 5 μs per skeleton, on average.

5.2 Limitations

Our model is specific to forward walking. Walking backwards or sideways are different gaits. For a more complete walking system, these other, less common gaits, would need to be modeled.

Our model follows the trajectory precisely. The quality of synthesized motion is dependent on the quality (an admittedly unknown quantity) of the trajectory. Furthermore, our system doesn't admit stopping. Depending on speed, a real human requires one or more steps to come to a rest. If the trajectory arbitrarily ends, we have no reasonable mechanism for instantaneously arresting motion.

6 Conclusion

We have presented a novel method for synthesizing walking motion. Our system is lightweight in both memory and computation and, as such, is well-suited to interactive applications with hundreds of characters. The motion is free from foot skating. The personality of the gait is defined by an input clip and the way the gait changes with speed is easily determined by an artist in an intuitive manner. Furthermore, motion is generated "on-the-fly" without latency from input, making our approach suitable for user-controlled avatars as well.

In the future, we will look into incorporating transitions between walking and standing as well as modelling other gaits. We'd also like to explore transformations to accomodate irregular surfaces, uneven terrain and other limitations on valid stepping areas.

References

1. Gleicher, M.:More Motion Capture in Games Can We Make Example-Based Approaches Scale?. Motion in Games, 82-93 (2008).
2. Multon, F., France, L., Cani, M.-P. and Debunne, G.: Computer Animation of Human Walking: a Survey. J. Vis. Comput. Animat. 1. 39–54 (1999).
3. Bruderlin, A. and Calvert, T. W.:Goal-directed, dynamic animation of human walking. Proc. SIGGRAPH SIGGRAPH '89, 233–242 (1989).

4. Bruderlin, A. and Williams, L.: Motion signal processing. Proc. of ACM SIGGRAPH 97104 (1995).
5. Boulic, R., Magnenat-Thalmann, N. and Thalmann, D.: A global human walking model with real-time kinematic personification. The Visual Computer 6. 344–358 (1990).
6. Boulic, R. and Thalmann, D.: Combined direct and inverse kinematic control for articulated figures motion editing. Computer Graphics Forum 4. 189-202 (1992).
7. Ko, H. and Badler, N. I.:Straight Line Walking Animation Based on Kinematic Generalization that Preserves the Original Characteristics. Proceedings Graphics Interface '93, 9–16 (1993).
8. Sun, H. C. and Metaxas, D. N.:Automating gait generation. Proc. SIGGRAPH '01, 261–270 (2001).
9. Park, S. I., Shin, H. J., Kim, T. H. and Shin, S. Y.: On-line motion blending for real-time locomotion generation: Research Articles. Comput. Animat. Virtual Worlds 3-4. 125–138 (2004).
10. Pelechano, N., Spanlang, B. and Beacco, A.:Avatar Locomotion in Crowd Simulation. Proc. CASA, (2011).
11. Kovar, L., Gleicher, M. and Pighin, F.: Motion graphs. ACM Trans. Graph. 3. 473–482 (2002).
12. Gleicher, M.:Graph-based motion synthesis: an annotated bibliography. ACM SIGGRAPH 2008 classes, 1–11 (2008).
13. Treuille, A., Lee, Y. and Popović, Z.: Near-optimal Character Animation with Continuous Control. ACM Trans. Graph. 3. (2007).
14. Heck, R. and Gleicher, M.:Parametric Motion Graphs. Proc. I3D07, (2007).
15. Lau, M., Bar-Joseph, Z. and Kuffner, J.: Modeling spatial and temporal variation in motion data. ACM Trans. Graph. 171:1–171:10 (2009).
16. Basten, vanB. J. H., Peeters, P. W. A. M. and Egges, A.: The step space: example-based footprint-driven motion synthesis. Comput. Animat. Virtual Worlds 433–441 (2010).
17. Basten, vanB. J. H., Stuvel, S. A. and Egges, A.: A hybrid interpolation scheme for footprint-driven walking synthesis. Graphics Interface 9-16 (2011).
18. Menardais, S., Kulpa, R., Multon, F. and Arnaldi, B.: Synchronization for dynamic blending of motions. 325-336 (2004).
19. Kulpa, R., Multon, F. and Arnaldi, B.: Morphology-independent representation of motions for interactive human-like animation. Computer Graphics Forum (Eurographics Special Issue .
20. Shapiro, A., Cao, Y. and Faloutsos, Y.: Style Components. Proc. of Graphics Interfaces (2006).
21. Neff, M. and Fiume, E.: Aesthetic Edits for Character Animation. ACM SIGGRAPH/Eurographics Symposium on Computer Animation 239-244 (2003).
22. Neff, M. and Kim, Y.: Interactive Editing of Motion Style Using Drives and Correlations. ACM SIGGRAPH/Eurographics Symposium on Computer Animation (2009).
23. Whittle, M. W.: Gait Analysis: An Introduction. Elsevier, (2007).
24. Inman, V. T., Ralston, H. J., Todd, F. and Lieberman, J. C.: Human Walking. Williams & Wilkins, (1981).
25. Dean, G. A.: An Analysis of the Energy Expenditure in Level and Grade Walking. Ergonomics 1. 31–47 (1965).
26. Murray, M. P.: Gait as a total pattern of movement. Am. J. Phys. Med. 1. 290–333 (1967).
27. Witkin, A. and Popović, Z.:Motion Warping. Proc. SIGGRAPH '95, 105–108 (1995).