INTERACTIVE SYNTHESIS OF CONSTRAINED MOTION FROM EXAMPLE MOVEMENTS

Jaroslav Semančík and Josef Pelikán and Jiří Žára
Faculty of Mathematics and Physics
Charles University in Prague
Czech Republic
email: {Jaroslav.Semancik|Josef.Pelikan}@mff.cuni.cz, zara@fel.cvut.cz

ABSTRACT
We present a technique for interactive synthesis of a continuous human motion specified by given constraints (e.g. footprints positions) using a library of example movements. The library contains basic movements (e.g step, jump, kick) obtained by motion capture. Several variations of each movement are captured for different parameter values (e.g. steps of various lengths and heights). Then the movements for any parameter values resulting from the constraints are created as a multidimensional scattered interpolation of examples in parameter space. Smooth transitions between successive interpolated movements are performed to yield a seamless continuous motion. The method is very flexible and it runs in extremely high performance appropriate for time-critical interactive applications like video-games. We demonstrate it on a human walking with variable step length by our implementation.

KEY WORDS
human figure animation, motion library, multidimensional interpolation, motion capture, motion blending

1 Introduction

Realistic motion of a virtual character (*avatar*) plays the same role in human figure animation by computer as a photorealistic rendering for still images. We are very sensitive to human motion; even a schematic character with "life-like' ' motion is accepted by human eye as more natural than a photorealistic model with unwieldy artificial motion. The reason is that we encounter human motion daily and we are perfectly familiar with it. Hence a computer animated character without a convincing motion acts disturbing in virtual environments.

A unique source of the most realistic life-like motion is *motion capture* – recording of a real motion performed by an actor. A library of captured basic movements can be prepared and used in runtime to animate an avatar as we see in current video-games. However, convincing motion of an interactively controlled avatar must adapt to constraints imposed by a virtual terrain around the avatar and a user control (e.g. location to reach). Variations of the captured movements are needed to generate in real-time instead of applying a single library motion directly. Examples of such constraining environments are any uneven terrain or a ford with stones standing out of the water and a problem of crossing the river dry-shod.

We propose a method to generate continuous human motion by varying of basic movements stored in a library. When some example variations with different parameter values are provided for each avatar movement, infinite number of new ones can be synthesized by interpolation of the examples. A multidimensional scattered interpolation runs in space of parameters and it blends the examples with proper weights to get a variation with desired parameter values. The parameters describe and quantify style (e.g. weariness or happiness) of a particular example motion or its physical properties (e.g. length of a step, height of a jump, etc.). Finally, the interpolated movements must be smoothly joined together to avoid snapping between successive movements.

2 Related Work

A library of example motions was first introduced in [1] where a single best-fit motion is chosen from the library and adjusted and in [2] where motion-captured examples are dense regular samples in a parameter space. A more general approach [3] further enhanced in [4] performs a multidimensional scattered interpolation on sparse irregular samples using radial basis functions. While [3] is focused mostly on stylistic parameters of the motion and [4] on inverse kinematics tasks, our goal is a continuous motion constrained by a terrain. A similar problem of scattered data interpolation is encountered in [5] to construct a parametric motion model. Unlike the cardinal basis used in [4] and in our work it interpolates the captured motion data compressed by principal component analysis.

Human gait over an uneven terrain is generated by interpolation in [6]. However, the method is restricted to lower body and a walking motion and it performs only 2D interpolation of examples. A task similar to ours is treated also by [7] but it does not address the problem of reducing the interpolation error. Various older approaches to generate human walk are surveyed in [8].

Another branch of recent research [9, 10, 11, 12] uses a larger corpus of unlabeled captured motion and constructs a graph of possible frame to frame transitions among all

motions in the corpus. A new motion is synthesized by finding a path in the graph satisfying user-specified requirements. The produced motion thus consists of original captured frames only and any variation of the motion that has not been captured cannot be synthesized except a path modification.

Statistical approaches [13, 14] provide greater variability to generated motion. A stochastic model is learned first and then used to synthesize a novel motion statistically similar to the original.

3 Motion Representation and Blending

We use a standard articulated figure (or a *skeleton*) as an avatar model. Modeling of a human body attached to the skeleton is beyond the scope of our work. Due to known problems with interpolation of rotations parameterized by Euler angles (see [15]) we use unit quaternions instead. Motion is thus a set of joint rotations represented as time-varying quaternion-valued signals $\mathbf{r}(t)$ and joint translations (for joints changing their position with respect to their parent in the articulated hierarchy, typically only the root) as vector-valued signals $\mathbf{t}(t)$

$$\mathbf{M}(t) = \big\{\mathbf{t}_i(t) \in \mathbb{R}^3, \mathbf{r}_j(t) \in \mathbb{S}^3 \mid i = 1, \ldots, n_t,\ j = 1, \ldots, n_r\big\}. \quad (1)$$

Moreover each example movement contains an increasing sequence of keytimes $\{K_i\}_{i=0}^{n_K}$ and a vector of parameters $\mathbf{p}$. Keytimes are times of significant events in the motion such as heel-strikes or toe-offs and they are used in blending of motions. Components of $\mathbf{p}$ are parameter values of the movement.

A blend of given motions $\mathbf{M}_1, \ldots, \mathbf{M}_n$ and respective weights $w_1, \ldots, w_n$ where $\sum w_i = 1$ is a motion similar to $\mathbf{M}_i$ proportionally to w_i, i.e. a weighted sum of motions

$$\mathbf{M} = \sum_{k=1}^{n} w_k \mathbf{M}_k. \quad (2)$$

The sum of motions above is a motion whose i-th joint translation is a weighted sum of the i-th joint translations across all blended motions. When blending joint rotations there is a complication due to the fact that both unit quaternions q and $-q$ represent the same rotation. Some of them may need to be negated before summing to get them to one hemisphere of rotations and thus to obtain a correct blend.

To get a blended motion one cannot simply evaluate all source motions for a fixed time but for times equivalent in the sense of motion structure. Duration of structural elements of the motion may be different in each example and blending a moving foot with a foot fixed on the ground results in ugly feet sliding. The equivalent times are the ones having the same *generic time* (see [3]) with respect to keytimes of a motion $\mathbf{M}$

$$g_{\mathbf{M}}(t) = i + (t - K_i)/(K_{i+1} - K_i), \quad \text{where } K_i \le t < K_{i+1}. \quad (3)$$

Keytimes of a blended motion $\mathbf{M}$ are computed as weighted sums of keytimes in source motions

$$K_{\mathbf{M},i} = \sum_{k=1}^{n} w_k K_{\mathbf{M}_k,i}, \quad \text{for } i = 1, \ldots, n_K. \quad (4)$$

To evaluate a blended motion $\mathbf{M}$ for a time t_0 the respective generic time $g_{\mathbf{M}}(t_0)$ is computed and all source motions are blended in that generic time. The complete algorithm is:

1. for $i = 1, \ldots, n_K$: compute keytimes $K_{\mathbf{M},i}$ by (4)
2. compute $g_0 = g_{\mathbf{M}}(t_0)$ by (3)
3. for $i = 1, \ldots, n$: evaluate source motions $\mathbf{M}_i$ for time $g^{-1}_{\mathbf{M}_i}(g_0)$
4. for $i = 1, \ldots, n_t$: blend i-th joint translation $\mathbf{t}_i$
5. for $i = 1, \ldots, n_r$:
   (a) adjust quaternions for $\mathbf{r}_i$ in all the motions to one hemisphere
   (b) blend i-th joint rotation $\mathbf{r}_i$
   (c) normalize the result

4 Synthesis of a Movement Variation

We synthesize new variations of human movements by blending of several example variations with known parameters. Let a new variation is to be created for a given parameter vector $\mathbf{p}$. If $\mathbf{p}$ is equal to one of the example's parameter vector the resulting movement should be identical to that example. Otherwise the respective variation is obtained by interpolation in space of parameters. The interpolated motion is synthesized by blending of examples, hence a set of weights to blend examples with must be found by interpolation for the given $\mathbf{p}$. The weights are known only in examples' parameter vectors $\mathbf{p}_i$ scattered in the parameter space – in $\mathbf{p}_i$ all weights are zero except the $w_i = 1$.

The interpolation should be continuous, smooth and well scalable to higher dimensions as number of parameters may be often greater than two or three. Interpolation of scattered data using radial basis functions (RBF) represents such an interpolation scheme perfectly suitable here. We use the method of Sloan, Rose and Cohen from [4] with cardinal basis functions and B-spline kernels. Each weight function $w_i(\mathbf{p})$ is a sum of a linear approximation and a RBF correction. The linear component is a hyperplane fitted to known weights at parameter vectors of examples by the least squares method. Radial basis functions correct the residuals between the hyperplane and the desired weights at examples $w_i(\mathbf{p}_j) = \delta_{ij}$ (where δ is the Kronecker delta). We refer to [4] for details. Most of the weights are zero due to compact support of B-spline kernel functions so the blending of examples needs only a few motions to actually evaluate.

Let $\hat{\mathbf{M}}$ be a movement interpolated for a given parameter vector $\hat{\mathbf{p}}$. If the parameters are not emotional but

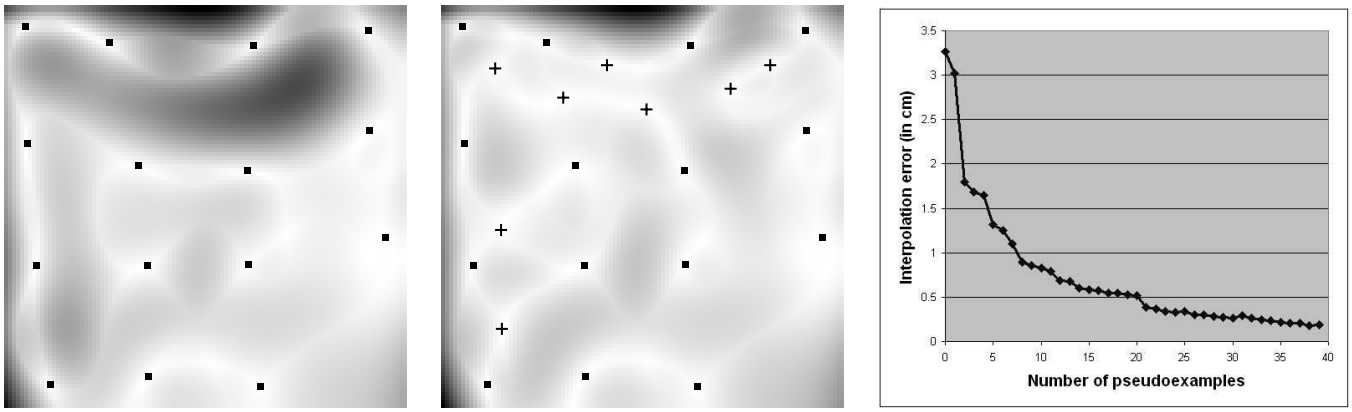

Figure 1. *Left image:* Error of interpolation shown in parameter space for an interpolated motion with two parameters. Points in darker regions correspond to synthesized motions with higher interpolation error. Black dots depict parameter vectors of example motions. *Middle image:* Interpolation error after refinement by pseudoexamples. Added pseudoexamples are depicted as crosses. *Right image:* Relation between number of added pseudoexamples and interpolation error.
Steps with various initial and final length were used as examples for demonstration in the above images.

physically measurable (e.g. length of a step), one can measure that actual parameter values of generated $\hat{\mathbf{M}}$ are often a little different than desired $\hat{\mathbf{p}}$. This is caused by nonlinearity of the parameter space. Interpolation error – a difference between $\mathbf{p}$ and actual parameters $\mathbf{a}(\mathbf{p})$ of a movement interpolated for $\mathbf{p}$ is demonstrated in Figure 1 on the left. However, there may be some $\mathbf{p}$ nearby $\hat{\mathbf{p}}$ such that $\hat{\mathbf{p}} = \mathbf{a}(\mathbf{p})$ or at least with lesser error $\|\hat{\mathbf{p}} - \mathbf{a}(\mathbf{p})\|$. We minimize the error[1] $e_{\hat{\mathbf{p}}}(\mathbf{p}) = \|\hat{\mathbf{p}} - \mathbf{a}(\mathbf{p})\|$ for a desired parameter vector $\hat{\mathbf{p}}$ and create $\hat{\mathbf{M}}$ for such $\mathbf{p}$ where the error is minimal. Minimum of $e_{\hat{\mathbf{p}}}(\mathbf{p})$ is found by steepest slope method from $\hat{\mathbf{p}}$ as a starting iteration.

The search described above is too expensive to perform in realtime each time a new movement is created. But once completed, the best weights for a movement with parameters $\hat{\mathbf{p}}$ are known. As introduced by [4], that blended motion may be added as a new "pseudoexample" to refine the library. The pseudoexample is most useful for a parameter vector $\mathbf{p}$ where the interpolation error $e(\mathbf{p}) = \|\mathbf{p} - \mathbf{a}(\mathbf{p})\|$ is maximal. A volume V of parameter space the examples lies within is sampled for $e(\mathbf{p})$ and the error in the samples is maximized by steepest slope method. The maximization is bounded by the volume V. An interpolated movement for the parameter vector where $e(\mathbf{p})$ is maximal is then created as explained in previous paragraph and added to the set of examples. That refinement of the library is repeated for a given number of cycles or until the maximal interpolation error is lesser than a given threshold. Result of the refinement is shown in the middle of Figure 1.

It is not necessary to create those pseudoexamples as complete motions represented as described in Section 3. Each pseudoexample is rather just a parameter vector and a set of corresponding weights to blend real examples with. The pseudoexamples just influence shape of the weight functions $w_1(\mathbf{p}), ..., w_n(\mathbf{p})$ used to blend real examples.

[1] The ℓ_∞ norm is used in the error function.

5 From Individual Movements to Continuous Motion

Most of human motion is a sequence of some basic movements (e.g. step, jump, turn, reach) with varying parameters (e.g. length, height, reach location, etc.). The individual interpolated movements are created by blending of fixed examples in the library that must share a common coordinate frame. All examples are normalized – oriented to start with identity rotation and translated to start at origin. That alignment propagates to all interpolated movements – each one specifies avatar's position and rotation with respect to start of the movement. A global position and rotation of the avatar at start of the current movement (a *base position/rotation*) is maintained and the relative offsets are transformed from that base coordinate frame to world coordinates each time the avatar's posture is evaluated.

In a closer look one can notice that bending forward or backward is a constituent of the motion and only a horizontal component of the initial rotation, i.e. rotation around a vertical axis is insignificant and should be eliminated from the example movements. A reason for this exclusiveness of the y-axis is vertical direction of the gravity force. Only that horizontal rotation is maintained then in base rotation of the avatar.

The following takes place when a movement is finished and a new one should be created: Position of the avatar at the end of the movement is stored to base position and a horizontal component of avatar's rotation to base rotation. A kind of a new movement and its parameters are determined by user control and terrain constraints or simply read from a prepared choreography. We consider those

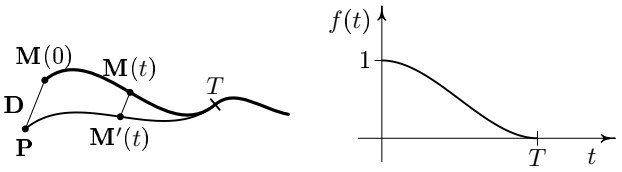

Figure 2. Adjustment of a movement $\mathbf{M}$ to start in a posture $\mathbf{P}$ (left image). A difference posture $\mathbf{D}$ is eased-out and added to the motion on an interval $[0, T]$. The used cubic ease-out function is shown in the right image.

data to be given, computing of parameters by path planning or interaction with a terrain are out of the scope of this paper. Finally, the new movement is created by interpolation.

For a seamless continuous motion the last posture of a finished interpolated movement must be identical to the first posture of the subsequent one. To generate transitions between different kinds of movements the library must contain also "transition movements". However, since the parameters don't determine the initial and final parameters uniquely and also due to interpolation error a visible snapping appears between successive movements and must be additionally smoothed.

Let the last posture of a movement that has just finished be $\mathbf{P}$. The beginning of next movement $\mathbf{M}$ is adjusted to start at $\mathbf{P}$ as follows: A "difference posture" is calculated as $\mathbf{D} = \mathbf{P} - \mathbf{M}(0)$. That difference posture is scaled by an ease-out function dropping from 1 to 0 on some interval $[0, T]$ at the beginning of the movement $\mathbf{M}$ and added to $\mathbf{M}$. The effect is illustrated in Figure 2. A cubic ease-out function $f(s) = 3(1-s)^2 - 2(1-s)^3$ is used where $s = t/T$.

A "posture arithmetic" for postures $\mathbf{A}$, $\mathbf{B}$ and scalar c is defined as

$$\begin{aligned}
\mathbf{A} + \mathbf{B} &= \left\{ \mathbf{t}_{\mathbf{A},i} + \mathbf{t}_{\mathbf{B},i},\ \mathbf{r}_{\mathbf{B},j}\mathbf{r}_{\mathbf{A},j} \mid i, j = 1, \ldots, n_t, n_r \right\} \\
\mathbf{A} - \mathbf{B} &= \left\{ \mathbf{t}_{\mathbf{A},i} - \mathbf{t}_{\mathbf{B},i},\ \mathbf{r}_{\mathbf{B},j}^{-1}\mathbf{r}_{\mathbf{A},j} \mid \forall i, j \right\} \\
c\mathbf{A} &= \left\{ c\mathbf{t}_{\mathbf{A},i},\ \mathbf{r}^c_{\mathbf{A},j} = \mathrm{slerp}(\mathbf{1}, \mathbf{r}_{\mathbf{A},j}, c) \mid \forall i, j \right\}
\end{aligned}$$

where $\mathbf{1}$ is an identity quaternion and "slerp" is a spherical linear interpolation of quaternions. Note that addition and subtraction of postures are not commutative as is implied from non-commutativity of quaternion multiplication.

One can easily verify that the adjusted motion

$$\mathbf{M}'(t) = f(s)\mathbf{D} + \mathbf{M}(t) \tag{5}$$

satisfies $\mathbf{M}'(0) = \mathbf{P}$ and $\mathbf{M}'(T) = \mathbf{M}(T)$ as shown in Figure 2. The smoothing above runs for postures in world coordinate frame, after transformation by base position and rotation of the avatar.

6 Motion Library

Example movements stored in a library are organized into groups of the same kind, e.g. a step by left foot, a step by right foot, a jump, etc. Such a group consists of the movement variations for various parameters. The examples can be prepared by any method, we use motion captured data mapped onto a skeleton. Data are sampled at 30 fps and interpolated to get a continuous signal for synthesis. Individual example movements must be extracted from a captured motion first, then normalized, their keytimes identified and finally annotated by parameter values.

Detection of start and end of a particular example movement in a longer captured motion depends on kind of the movement. A procedure for each kind must be written, we outline the one for a walking motion in section Results. Representative variations of each movement are then chosen by hand as examples.

Normalization of examples translates the movements and rotates them horizontally to start at origin with face in direction of z-axis.[2] All skeleton's root rotation samples are transformed by inverse of a horizontal component of the root's initial rotation. Thus a vertical rotation (i.e. bending of the body) is preserved. Let q be a unit quaternion of root's initial rotation. Let z' be the z-axis rotated by q and then projected to a horizontal plane and α be the angle between z' and the original z-axis. Then the horizontal component of q is a rotation around vertical y-axis by angle α. The respective quaternion is $q_h = (0, \sin\frac{\alpha}{2}, 0, \cos\frac{\alpha}{2})$. Besides translation of the movement to start at origin all samples of root's translation must be also rotated by q_h^{-1}.

Keytimes and parameters of an example movement, their number and meaning depends again on kind of the movement and needs a proper analysis procedure. We demonstrate detection of keytimes and measuring of parameter values for a walking motion in the next section.

7 Results

We have tested our method on a walking motion and a standing jump. Walking is considered as alternating of two kinds of movements – a step by left foot and a step by right foot. The steps were extracted as a motion between two consecutive double support phases, precisely between centers of time intervals when both feet are fixed on the ground. One foot is fixed during the whole step the other is moving. A foot is considered as fixed when both its velocity and height are lower than a defined threshold. Keytimes are identified using the intervals of fixed feet. There are four keytimes for each step: start, toe-off (double-support phase ends here), heel-strike (next double-support phase starts here), end. A general step would have 7 parameters: relative position of feet, i.e. a vector from the fixed foot position to the moving one at start of the step (3 components),

[2] We use a right-handed coordinate frame with x-axis pointing right, y-axis pointing up and z-axis pointing out of the screen.

the same at the end of the step and an angle between initial and final horizontal rotation of the body. The vectors are taken in coordinate frame of the skeleton root at start and end of the step respectively. Currently we have data available for straightline steps having two parameters – lengths in initial and final double support phases i.e. the relative feet positions in direction of forward axis only. The steps were performed for three different lengths – short, medium and long together with transitions to/from still standing giving altogether 15 examples for the left step as well as for the right step. Some of them are shown in Figure 3. The motions were captured by an optical motion capture system Vicon at the rate 120 Hz resampled down to 30 Hz. Due to short duration of the steps (about 1 second) the space needed to store an example movement is only about 10 KB and size of the entire library about 320 KB.

From those example steps a straightline step of any initial and final length is synthesized. Using the proposed smoothing between successive steps the root translation and all joint rotations are continuous and the avatar walks seamlessly with given steps lengths. A generated walk is shown in Figure 4. Unfortunately, a visible feet sliding is present in the motion (especially for short steps) due to discrepancy between interpolated translation of the root and rotation of joints. It can be fixed by using constraints or as in [16].

Error of the interpolation is up to 3.3 cm in a volume of parameter space spanned by examples. Thanks to linear component of the interpolation the steps may be also extrapolated outside of that volume but the error rapidly grows there. The interpolation error after adding about 10 pseudoexamples dropped under 1 cm (see Figure 1) and in the pseudoexamples themselves it was less than 0.01 mm in our experiments.

A standing jump movement has the same keytimes as a step and only one parameter – length of the jump. Four jumps of different length were captured to synthesize a jump of arbitrary length.

Speed of the runtime system is very favorable. Our experimental implementation not specially optimized evaluates 3300 steps per second on a low-end 1.2 GHz Intel Celeron processor. Rendering of the avatar is not comprised in that number, just the blending of example steps and calculating weights for given parameters when a new step is synthesized. Refinement of the motion library as a preprocessing step takes up to one minute.

Video sequences showing some of the example steps as well as the synthesized walk shown in Figure 4 are available at the project webpage
`http://cgg.ms.mff.cuni.cz/~semancik/phd/`.

8 Conclusions and Future Work

The proposed method interactively produces human motion adapted to a constraining virtual environments and user control. Its strengths are realism provided by motion capture, flexibility and runtime performance that makes it interesting for video games. Nevertheless a constraint system must be integrated further to eliminate some undesired effects such as sliding of end-effectors that are supposed to be fixed as it very degrades the produced motion.

A number of challenging questions arises for future research. We planned and chose example movements by hand. It would be much more convenient or even necessary for larger motion libraries to identify suitable examples in a continuous motion automatically. A future vision of examples acquisition is an automatic extraction of representative movements (i.e. a set of movements that is sparse and spanning an interested portion of parameter space) from a long captured motion of a freely moving actor. Other interesting issues are automatic transitions between motions of a different structure or combining motion of individual body parts.

Acknowledgments

We would like to thank Bohemia Interactive Studio and specially to Jan Hovora and Jakub Mareš for capturing and processing the motion data.

References

[1] Alexis Lamouret and Michiel van de Panne. Motion synthesis by example. In *Proceedings of the Eurographics workshop on Computer animation and simulation '96*, pages 199–212. Springer-Verlag New York, Inc., 1996.

[2] Douglas J. Wiley and James K. Hahn. Interpolation synthesis of articulated figure motion. *IEEE Comput. Graph. Appl.*, 17(6):39–45, 1997.

[3] Charles Rose, Michael F. Cohen, and Bobby Bodenheimer. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics and Applications*, 18(5):32–41, /1998.

[4] Michael F. Cohen Charles F. Rose III, Peter-Pike J. Sloan. Artist-directed inverse-kinematics using radial basis function interpolation. *Computer Graphics Forum*, 20(3), 2001.

[5] Ik Soo Lim and Daniel Thalmann. Construction of animation models out of captured data. *Proceedings of IEEE International Conference on Multimedia and Expo (ICME 2002)*, 1:829–832, August 2002.

[6] Harold C. Sun and Dimitris N. Metaxas. Automating gait generation. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 261–270. ACM Press, 2001.

[7] Sang Il Park, Hyun Joon Shin, and Sung Yong Shin. On-line locomotion generation based on motion blending. In *Proceedings of the 2002 ACM SIG-*

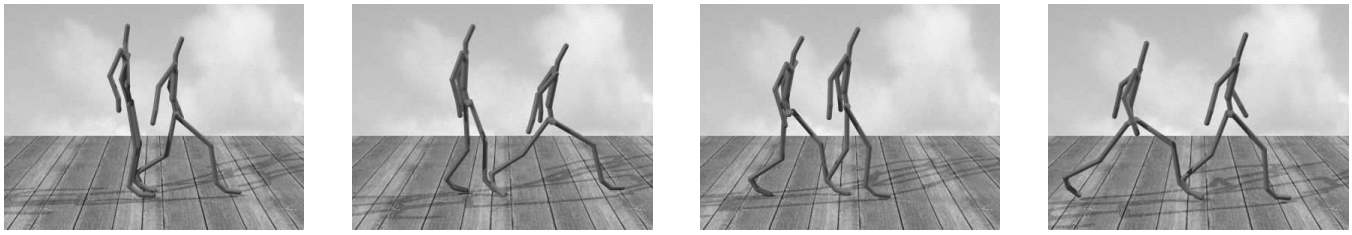

Figure 3. Four of examples for a step by right foot. The first and last posture of the step is shown in each image.

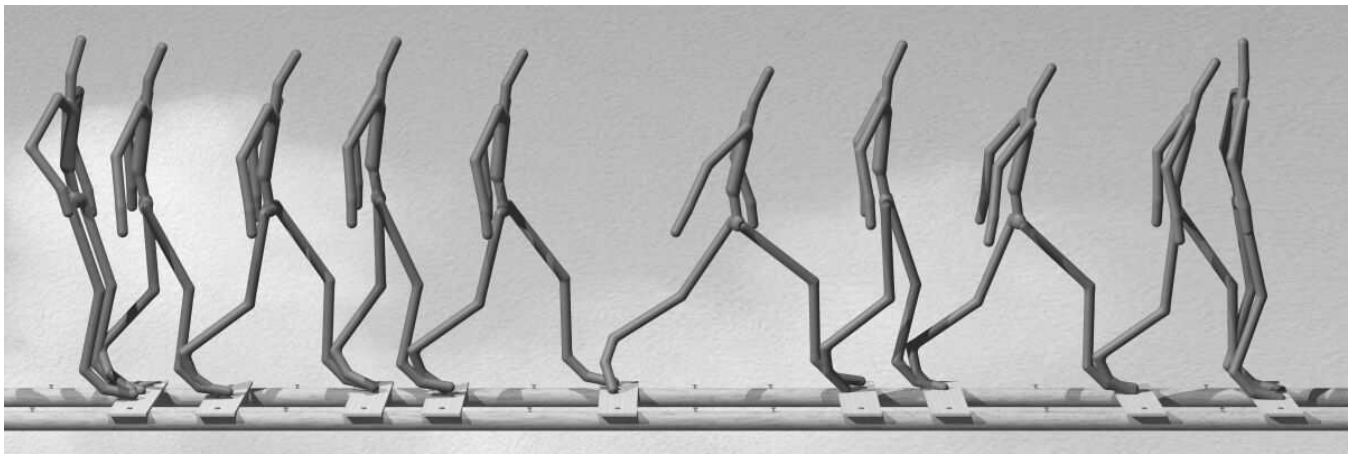

Figure 4. A synthesized walk on a constraining terrain. Proper steps are generated for arbitrary distances between slabs on the ladder. Selected frames of the motion are shown.

GRAPH/Eurographics symposium on Computer animation, pages 105–111. ACM Press, 2002.

[8] Franck Multon, Laure France, Marie-Paule Cani-Gascuel, and Giles Debunne. Computer animation of human walking: a survey. *The Journal of Visualization and Computer Animation*, 10(1):39–54, 1999.

[9] Lucas Kovar, Michael Gleicher, and Frédéric Pighin. Motion graphs. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 473–482. ACM Press, 2002.

[10] Michael Gleicher, Hyun Joon Shin, Lucas Kovar, and Andrew Jepsen. Snap-together motion: assembling run-time animations. In *Proceedings of the 2003 symposium on Interactive 3D graphics*, pages 181–188. ACM Press, 2003.

[11] Okan Arikan and D. A. Forsyth. Interactive motion generation from examples. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 483–490. ACM Press, 2002.

[12] Jehee Lee, Jinxiang Chai, Paul S. A. Reitsma, Jessica K. Hodgins, and Nancy S. Pollard. Interactive control of avatars animated with human motion data. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 491–500. ACM Press, 2002.

[13] Matthew Brand and Aaron Hertzmann. Style machines. In Kurt Akeley, editor, *Siggraph 2000, Computer Graphics Proceedings*, pages 183–192. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000.

[14] Yan Li, Tianshu Wang, and Heung-Yeung Shum. Motion texture: a two-level statistical model for character motion synthesis. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 465–472. ACM Press, 2002.

[15] Alan Watt and Mark Watt. *Advanced animation and rendering techniques*. ACM Press, 1991.

[16] Lucas Kovar, John Schreiner, and Michael Gleicher. Footskate cleanup for motion capture editing. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 97–104. ACM Press, 2002.