Visual Comput (2006) 22: 194–209
DOI 10.1007/s00371-006-0376-9

ORIGINAL ARTICLE

Pascal Glardon
Ronan Boulic
Daniel Thalmann

Robust on-line adaptive footplant detection and enforcement for locomotion

Published online: 7 February 2006

P. Glardon (✉) · R. Boulic · D. Thalmann
Ecole Polytechnique Fédérale
de Lausanne (EPFL),
Virtual Reality Lab,
CH-1015 Lausanne, Switzerland
pascal.glardon@epfl.ch

Abstract A common problem in virtual character computer animation concerns the preservation of the basic foot-floor constraint (or footplant), consisting in detecting it before enforcing it. This paper describes a system capable of generating motion while continuously preserving the footplants for a real-time, dynamically evolving context. This system introduces a constraint detection method that improves classical techniques by adaptively selecting threshold values according to motion type and quality. The footplants are then enforced using a numerical inverse kinematics solver. As opposed to previous approaches, we define the footplant by attaching to it two effectors whose position at the beginning of the constraint can be modified, in order to place the foot on the ground, for example. However, the corrected posture at the constraint beginning is needed before it starts to ensure smoothness between the unconstrained and constrained states. We, therefore, present a new approach based on motion anticipation, which computes animation postures in advance, according to time-evolving motion parameters, such as locomotion speed and type. We illustrate our on-line approach with continuously modified locomotion patterns, and demonstrate its ability to correct motion artifacts, such as foot sliding, to change the constraint position and to modify from a straight to a curved walk motion.

Keywords Motion anticipation · Animation with constraints · Human body simulation

1 Introduction

The animation of virtual characters remains a challenging topic in computer graphics. Common human activities, for example, walking, jogging, running or jumping are widely applied, especially in computer games. In such a context, the variety of generated motions is fundamental. On the one hand, it is necessary to produce animations controlled by high-level parameters, like changing the locomotion style and speed to walk around in a virtual environment. On the other hand, small continuous variations of those parameters increase and sustain the believability of character movements. These variations have to be performed on-line, reactive to user's requests or to autonomous agents.

Concurrently, the resulting animation should be as realistic as possible, notably by maintaining basic physical constraints. Among them, keeping the foot planted on the floor during a period of time (referred to as footplant in this paper) has entailed the elaboration of numerous methods, divided into two distinct stages: the detection of a footplant and its enforcement.

Traditional methods [2, 27] that detect the start and end of a footplant use global thresholds on the position and velocity of the feet, independently of the motion type con-

taining the footplants. However, threshold parameters for a light walk are not compatible with a high dynamic run. In addition, these methods assume that the motions are free from noise and artifacts such as foot sliding. Therefore, in this paper, we tackle a first problem: footplant detection in real-time. We propose an adaptive on-line method that takes into account the nature and quality of the motion to determine threshold values automatically. In addition, this detection technique utilizes the motion structure to determine restricted periods of time when a constraint has to be investigated.

Concerning the second stage, many methods enforcing footplants are based on specialized IK (inverse kinematics) solvers [20, 24, 37]. All, except the method of Kovar et al. [20] can be applied on-line, for example for motion retargeting [29], or to maintain foot constraints during the transition between two motion capture clips [23]. However, these approaches are unadapted when the original position needs to be modified at the constraint time. In practice, imperfect input motions may contain a footplant whose foot trajectory ends above the ground. In this case, the constraint position has to be re-positioned on the ground, while ensuring a smooth motion correction. Hence, we address a second problem in this paper: real-time constraint re-positioning and enforcement. We correct a posture with the new constraint positions by using a numerical IK algorithm, robust enough to work on-line. In order to avoid an abrupt change in the constraint location, we introduce an ease-in phase based on a displacement map, allowing us to modify the motion towards the new posture smoothly. Similarly, an ease-out phase allows us to seamlessly release the constraint. In addition, we improve the motion realism by defining two effectors (heel and toe) on each foot that control the footplant.

To deal with the two main problems presented in this paper, it is necessary to obtain future frames of the current animation. Actually, as the constraint detection method is adaptive, it needs to know the foot position in advance. For footplant enforcement, the modified posture at the start of the constraint is required at the beginning of the ease-in phase. In this case, future motion information is also crucial. However, the motion is generated on-line, producing a continuous stream of frames with uncertain variations due to the parameter changes. Therefore, to compensate these two paradoxical goals, we introduce a novel approach capable of anticipating the motion. According to the motion state, described with its current and eventually desired high-level parameters, our method computes future information so as to react correctly in the present.

A complete overview of our system is depicted in Fig. 1. To illustrate our methodology, we focus on locomotion patterns, providing walking and running cycles. At any time, the user can determine either current high-level parameters, or desired ones, which have to be reached after a specific period of time. In addition, effectors can be configured in order to modify the foot constraint location.

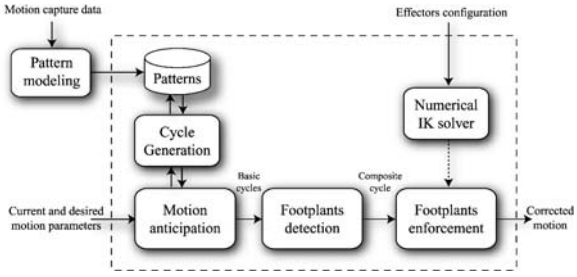

Fig. 1. System overview

According to the varying parameter set, the motion anticipation module computes multiple anticipated frames, each stemming from a different basic cycle. Footplants are also detected on these basic cycles, and are enforced using our numerical IK solver. In addition, we refer to the postures from anticipated frames as a composite cycle. Postures from this cycle are used to ensure a smooth transition from the unconstrained state to the constrained state.

The main contribution of this paper is to propose a method efficient enough to encapsulate both footplant detection and enforcement in a single system, where motions are generated on the fly and whose parameters may potentially vary endlessly. We explore different applications of this method: correction of input motions having feet artifacts (sliding footplant slightly above the ground), stylistic production by changing constraint locations, motion modification from a straight walk to a curved one.

The next section reviews related work. Section 3 describes our motion anticipation method. Section 4 describes footplant detection in detail, while Sect. 5 describes its enforcement. Then we illustrate our methodology with results in Sect. 6. Finally, we conclude this paper by discussing the limitations of our method and possible extensions.

2 Related work

The goal of this paper is to present an on-line system that integrates the detection of constraints and their enforcement using an IK-based motion editing technique. For illustration purposes, this system is coupled with a real-time locomotion engine. The related works are, therefore, classified into three research directions: locomotion generation, constraint detection and motion editing.

2.1 Locomotion generation

Many works have been dedicated to the locomotion of virtual humans [28]. Two classes of methods can be distinguished: off-line and on-line, both offering different levels of precision in the resulting motion, in particular regarding footplant preservation.

Off-line methods. To produce off-line animations, the keyframing technique allows an animator to specify key postures at specific key times. Using appropriate software [26], skilled designers can control the motion in detail. However, this technique is quite labor-intensive, as any motion parameter change entails the animators to modify every keyframe. Kinematics approaches generate motions from parameters such as position feet or speed value. Extended from one of the pioneering works of Girard and Maciejewski [11], step-driven methods [8, 40] have been developed. Motions are generated by giving a pre-defined set of foot positions (footprints) and timing information. This data is generally computed by a motion planning technique, which has to be as interactive as possible to be comfortable for animators.

Dynamics approaches aim to describe a motion by applying physics laws. As an example, Wooten and Hodgins [43] propose control algorithms based on a finite state-machine to describe a particular motion and proportional-derivative servos to compute the forces. However, even if these methods produce physically correct animations, the configuration of their algorithms remains difficult. It is not easy to determine the influence of each parameter on the resulting motions.

On-line methods. The procedural animation technique refers to the generation of motions by applying different successive parameterized algorithms on input data. This definition is not strict: keyframe-based techniques may also be considered as a succession of algorithms. However, for procedural approaches, the input data does not define the motion explicitly, but consists of a parameter set describing initial conditions used by the algorithms. Many methods based on empirical data and bio-mechanical observations are able to generate walking [3] or running patterns [4], reactive to given user parameters. Other similar approaches take into account the environment, walking on uneven or sloped terrains [38] or climbing stairs [9]. Despite their real-time capability, all these methods lack in realism, as the legs' motion is considered symmetrical, for example. To increase the motion believability, Ko and Badler [18] propose to add inverse dynamics to control character balance. Another class of on-line animation techniques re-uses original motion capture data. Treated as a time-varying signal, a new motion can be generated by modifying its frequency bands [5] or its Fourier coefficients [39]. Other methods [29, 32] define each motion by B-spline coefficients. New motions are then computed by setting weights on the various original motions and performing interpolations using polynomial and RBF (radial basis function) functions. The Kovar and Gleicher [19] method wraps input motions into a data structure that ensures consistent time-warping, root alignment and constraint matching. Finally, Glardon et al. [12] apply PCA (principal component analysis) on a motion capture database composed of various walking and running cycles. The generation of new parameterized motions is performed by interpolation and extrapolation into a hierarchical structure of PCA spaces.

This latter class of techniques produces realistic and high-level parameterized motions (e.g. velocity, walk-run ratio), but suffers artifacts, as the foot constraint is not perfectly preserved. Post-processing correction methods, such as *footskate cleanup* [20] may be applied, but cancel the real-time reactivity of the motion generation.

2.2 Constraint detection

In this paper, we consider a constraint from its geometrical point of view: for a given body part, a goal position has to be reached and enforced within a period of time.

A category of constraint detection methods allows us to compute the proximity between two points of interest, a virtual human's finger and a table, for example. To determine this proximity only at possible relevant frames, Bindiganavale and Badler [2] introduce the notion of effector acceleration zero-crossing. Hreljac and Marshall [17] use a similar technique to determine the heel-strike and toe-off times in walking motion. Their results are compared with measures performed on force platforms. In [23, 27], the authors propose another approach by setting thresholds on the position and velocity of the feet to detect footplants. Under these threshold values, a foot joint is considered to be fixed on the floor.

Liu and Popović [25] present a generic constraint detection method by finding points on a character's body that stay fixed in space for some period of time. In addition, close constraints are merged if the duration between constraints is under specified threshold values. Salvati et al. [35] extend this method to detect constraints relative to moving objects in a scene (e.g. a hand touching a ball).

In addition to being off-line, all of these techniques prove to be unreliable if the original motion is noisy. Actually, threshold values vary according to the input motion properties. We illustrate this problem further in this paper by describing situations where thresholds should be dependent of the motion parameters. We also propose a technique for improving the constraint detection in such a case.

2.3 Motion editing

Motion editing techniques aim to slightly modify, retarget or correct (e.g. by applying constraints) an existing motion.

Off-line methods. Constraint-based techniques, discussed and classified in [15], alter an original motion while preserving some specific geometrical features. Among them, space-time constraint [14, 24] or physically-based approaches [25, 30] provide effective tools to interactively manipulate a motion clip by changing its properties. How-

ever, these methods are computationally expensive and it remains intuitively difficult to provide a correct mathematical formulation corresponding to the desired motion modifications. Recent works tend to improve this weakness, either by reducing the motion dimension [34], or by offering more intuitive motion control [36]. Alternative approaches apply IK for motion editing. Specialized analytical methods are presented by Kovar et al. [20] to clean up footskate from motion capture data as a post-processing step. However, this method is not appropriate for rigid bodies (such as those based on [16]), as the leg length may vary. Another approach introduced by Lee and Shin [24] allows motions to be edited hierarchically, improving their modifications at each iteration. In [22], the authors propose to modify a motion by applying a numerical IK method setting priorities on effectors and controlling the center of mass. Finally, Yamane et al. [44] use a similar IK solver to generate animation given an object manipulation task and a motion database.

On-line methods. Only a few methods are able to edit motion in real-time. Rose et al. [33] propose to perform an interpolation of several input motions to reach a given end-effector position. In [37], an analytic inverse kinematics is applied to perform on-line motion retargeting. This method is integrated into the locomotion generation described in [29]. Rose et al. [32] use an optimized numerical solution to apply footplants on motions. Finally, a hybrid solution is provided by Choi and Ko [7] by using an inverse rate control technique to retarget a motion. To be computationally competitive, these on-line methods control the position of a single joint per footplant whose detection is performed in a pre-processing stage. While powerful, all these motion editing techniques need improvements to be integrated into an on-line motion generation system, whose footplants are automatically detected and enforced. Actually, it is necessary to substitute the off-line animator's task, consisting in building up the end-effector trajectories, by an automatic process. These trajectories allow the foot to be re-positioned in order to keep it fixed on the floor level during a constraint.

2.4 Motion anticipation

Butz et al. [6] classify anticipatory mechanisms in four categories: implicit, payoff, sensorial and state-based. We focus on the latter category, dealing with mechanisms in which predictions about future states directly influence current behavioral decision making. In general, anticipation is applied to explore unknown virtual environments, where virtual agents are equipped with multi-sensory systems [10]. A collaborative multi-agent context may also be based on anticipation in order to predict the internal state of other autonomous virtual agents [41]. To our knowledge, only a few works deal with motion anticipation. Labbé et al. [21] propose to anticipate periodic movement trajectories in a prey-predator situation. Steering methods [31] can anticipate a motion according to a given desired speed and/or a target to be reached. However, these techniques consider the virtual human as a simple 3D model composed of six degrees of freedom, instead of a complete articulated system.

To conclude this survey, we summarize the main drawbacks of the previous methods:

- Constraint detection methods are unreliable in an on-line context.
- Constraint enforcement methods do not consider the re-positioning of the foot.
- Anticipation methods based on a real-time locomotion engine do not provide future body postures.

Therefore, our method aims to provide a complete on-line system for character animation: motion generation, adaptive constraints detection and finally smooth constraint re-positioning and enforcement. In order to achieve this goal, we introduce the notion of motion anticipation in the next section.

3 Motion anticipation

We define the term of motion anticipation as the ability to generate future motion postures. In the case of off-line animation processes, these postures are directly disposable. The particular case of on-line motion generation is much more complicated. In point of fact, traditional animation methods [29, 32] compute the postures frame by frame, according to an elapsed Δt time between two animation updates. To anticipate postures, it is necessary to have a method that is efficient enough not to alter the real-time motion generation process, in terms of update rate. In addition, the anticipation has to take into account possible parameter variations, such as changes in the locomotion speed.

To address on-line anticipation, we base our approach on the motion modeling method presented in [12], which generates a motion as a whole locomotion cycle as opposed to the standard frame by frame approach. By applying a PCA algorithm on a motion capture database, hierarchical PCA spaces have been constructed. As this PCA algorithm considers *eigencycles* and not *eigenframes*, an entire time-normalized walking or running cycle is computed at once, according to given motion parameters. This method is very efficient and takes only 0.3 ms to compute a 25 frame cycle. Note that original motions have been captured on a treadmill.

3.1 Current posture computation

A motion may be represented as a continuous function of time

$$\boldsymbol{M}(t) = (\boldsymbol{p}_r(t), \boldsymbol{q}_r(t), \boldsymbol{q}_1(t), \ldots, \boldsymbol{q}_n(t)), \quad (1)$$

where $\boldsymbol{p}_r(t)$ and $\boldsymbol{q}_r(t)$ represent the global position and orientation of the root node, and $\boldsymbol{q}_i(t)$, for $i > 0$, the local transformation of the i-th joint.

To compute $\boldsymbol{M}(t)$ at a given time $t = t_i$ (i.e. a posture), we need to generate a locomotion cycle $\boldsymbol{L}$ first, determined by a parameter vector $\boldsymbol{w}$ whose elements each describe a motion characteristic (e.g. speed, type of locomotion, personification). This cycle is regularly sampled into frames $\mathcal{F}_i = \boldsymbol{L}(\varphi_i)$ on a normalized time $\varphi = 0 \ldots 1$ referred to as the locomotion phase. A locomotion cycle can be written as

$$\boldsymbol{L}(\boldsymbol{w}) = (\mathcal{F}_0, \ldots, \mathcal{F}_m) \tag{2}$$

and its corresponding continuous function as

$$\boldsymbol{L}(\boldsymbol{w}, \varphi) = (\widehat{\boldsymbol{p}}_r(\varphi), \widehat{\boldsymbol{q}}_r(\varphi), \boldsymbol{q}_1(\varphi), \ldots, \boldsymbol{q}_n(\varphi)). \tag{3}$$

We bear in mind that $\widehat{\boldsymbol{p}}_r(\varphi)$ and $\widehat{\boldsymbol{q}}_r(\varphi)$ contain only the translation and orientation of treadmill locomotion.

Then, the current time t_i has to be dynamically wrapped into the normalized time, by updating the locomotion phase from φ_{i-1} to φ_i. Actually, to the elapsed time $\Delta t = t_i - t_{i-1}$ corresponds a $\Delta\varphi$ computed by a frequency function described in [12]. Hence the new locomotion phase is

$$\varphi_i = \varphi_{i-1} + \Delta\varphi \pmod 1 \tag{4}$$

and is multiplied by the total frame number of a cycle so as to obtain the frame index i, and so, the unwrapped frame $\mathcal{F}_i$ of $\boldsymbol{L}(\boldsymbol{w}, \varphi)$.

Finally, $\boldsymbol{M}(t_i)$ has to be constructed in the global coordinate system because the root node of a cycle $\boldsymbol{L}$ is expressed in a local coordinate system aligned to the body. In addition, this node contains only local translation and orientation oscillations of the motion, as the original cycles of the motion model are performed on a treadmill. Therefore, according to the two current motion parameters, linear velocity $\boldsymbol{v}$ and angular speed ω, $\boldsymbol{p}_r(t_i)$ and $\boldsymbol{q}_0(t_i)$ of $\boldsymbol{M}(t_i)$ are modified with respect to the elapsed time $\Delta t = t_i - t_{i-1}$:

$$\begin{aligned} \boldsymbol{p}_r(t_i) &= \boldsymbol{p}_r(t_{i-1}) - \widehat{\boldsymbol{p}}_r(\varphi_{i-1}) + \widehat{\boldsymbol{p}}_r(\varphi_i) + \boldsymbol{v}\Delta t, \\ \boldsymbol{q}_r(t_i) &= \boldsymbol{q}_r(t_{i-1}) \times \widehat{\boldsymbol{q}}_r(\varphi_{i-1})^{-1} \times \widehat{\boldsymbol{q}}_r(\varphi_i) \times \mathrm{Rot}(\omega\Delta t), \end{aligned} \tag{5}$$

where $\mathrm{Rot}(\alpha)$ defines the rotation of the yaw angle α.

3.2 Anticipated posture computation

To anticipate the motion, our method generates not only $\boldsymbol{M}(t_i)$, but also any future posture $\boldsymbol{M}(t_i + \Delta T)$, with $\Delta T \geqslant \Delta t$, in real-time and at any point in time t_i. Figure 2 illustrates the anticipation of 13 postures, with a constant ΔT between each posture. The yellow bodies on the far left in the images represent the current postures, whereas the others are anticipated. To explain the posture computation, we consider two contexts: one where the parameter vector $\boldsymbol{w}$ does not vary (Fig. 2, left), and one where it continuously varies (Fig. 2, right).

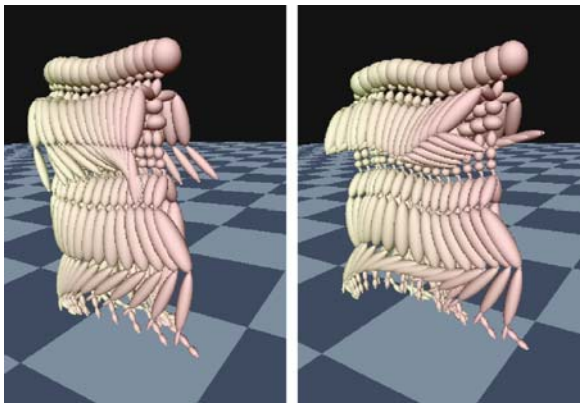

Fig. 2. Motion patterns with anticipation. The yellow posture (far left) represents the current frame, while the others (from yellow to red, towards the right) are anticipated postures. *Left:* invariant motion parameters. *Right:* variation of the speed parameter

We imagine the creation of a buffer containing n anticipated postures, with a constant ΔT between postures. For the first context (we assume that the angular speed is null), the j-th posture $\boldsymbol{M}(t_i + j\Delta T)$, for $j = 1 \ldots n$, is computed analogously to $\boldsymbol{M}(t_i)$. As $\boldsymbol{L}(\boldsymbol{w})$ contains all the cycle frames, only the root node has to be updated according to Eq. 5 where $\Delta t := j\Delta T$.

The second context considers a continuous variation of $\boldsymbol{w}$. Let $\boldsymbol{w}_j$ be the parameter vector at time $t_j = t_i + j\Delta T$. Hence, the computation of the j-th posture at time t_j involves the generation of a new motion pattern $\boldsymbol{L}(\boldsymbol{w}_j)$. We refer to the set of all these n new patterns as composite cycles. Then, the root node of each $\boldsymbol{M}(t_j)$ is sequentially updated, starting from $j = 1$ until $j = n$, to take into account the continuous parameter variation. Equation 6 described one update step, where $\Delta T = t_j - t_{j-1}$.

$$\begin{aligned} \boldsymbol{p}_r(t_j) &= \boldsymbol{p}_r(t_{j-1}) - \widehat{\boldsymbol{p}}_r(\varphi_{j-1}) + \widehat{\boldsymbol{p}}_r(\varphi_j) + \boldsymbol{d}(\Delta T) \\ \boldsymbol{q}_r(t_j) &= \boldsymbol{q}_r(t_{j-1}) \times \widehat{\boldsymbol{q}}_r(\varphi_{j-1})^{-1} \times \\ &\qquad \widehat{\boldsymbol{q}}_r(\varphi_j) \times \mathrm{Rot}\left(\frac{\omega_j + \omega_{j-1}}{2}\Delta T\right), \end{aligned} \tag{6}$$

where ω_j is the angular speed at time t_j.

The function $\boldsymbol{d}(\Delta T)$ approximates the translation component of a curved trajectory. Assuming that ΔT is small enough to consider the parameter variation between t_{j-1} and t_j as linear, the translation approximation can be written as

$$\boldsymbol{d}(\Delta T) = \begin{pmatrix} \sin\left(\alpha_{j-1} + \frac{\omega_j + \omega_{j-1}}{2}\Delta T\right) \frac{\|\boldsymbol{v}_j\| + \|\boldsymbol{v}_{j-1}\|}{2}\Delta T \\ 0 \\ \cos\left(\alpha_{j-1} + \frac{\omega_j + \omega_{j-1}}{2}\Delta T\right) \frac{\|\boldsymbol{v}_j\| + \|\boldsymbol{v}_{j-1}\|}{2}\Delta T \end{pmatrix} \tag{7}$$

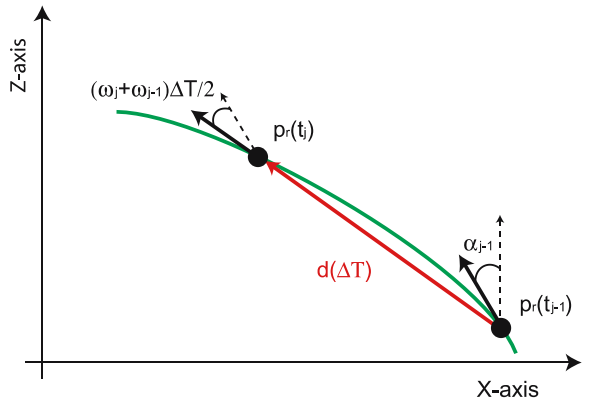

Fig. 3. The original root trajectory (green curve) approximated by the function $\boldsymbol{d}(\Delta T)$, illustrated by the red line, for a given $\Delta T = t_j - t_{j-1}$

where $\boldsymbol{v}_j$ and α_j are the linear velocity and the yaw angle at time t_j, respectively. In our coordinate system, a null yaw angle coincides with a locomotion direction along the Z-axis. Figure 3 illustrates the approximated root translation (the red line) of an original curved motion (the green curve).

In practice, the anticipation of the local orientation of a given joint is not sufficient. We also need to know its position in the global coordinate system, in particular for foot joints in order to compute their Euclidian distance to the floor. With this aim in mind, the human body hierarchy has to be traversed, starting from the root node until the desired node, by multiplying every local node orientation. This operation is expensive, and it severely limits the number of anticipated frames given a continuous varying real-time animation context.

In short, given a future parameter variation (e.g. characterized by a set of parameter values to be reached within a desired period of time), the proposed method is able to anticipate postures and the Cartesian location of body segments in a global coordinate system. This material is used for footplant detection and enforcement, as explained in the next two sections.

4 Footplant detection

A virtual human's activity can be segmented into many different constraint types, allowing a formal description of its motion. One of the most important constraint types is the footplant, defined as a period of time during which a foot or part thereof (e.g. ankle, toe) remains in a fixed position with respect to the ground. In this paper, a footplant considers two joints: the ankle (or heel) and the metatarsal (or toe) joints, according to the standard H-ANIM [16] skeleton. The knowledge of constraint information is crucial in many situations. For example, it allows us to structurally align motions [29, 32], to determine the transition period between two motions [13] or to correct artifacts such as footskate [20]. A footplant has to be detected with methods that have to return precise results. Actually, a too short footplant duration would not totally correct the foot sliding. Conversely, a too long duration would stretch the leg when reaching the joint limits, resulting in motion discontinuities.

4.1 Thresholds on Cartesian position and speed

The problem of footplant detection can be solved by using a standard method based on the current vertical position (or height) and translation speed of the foot, as described in [23, 27]. At each animation frame, foot constraints are

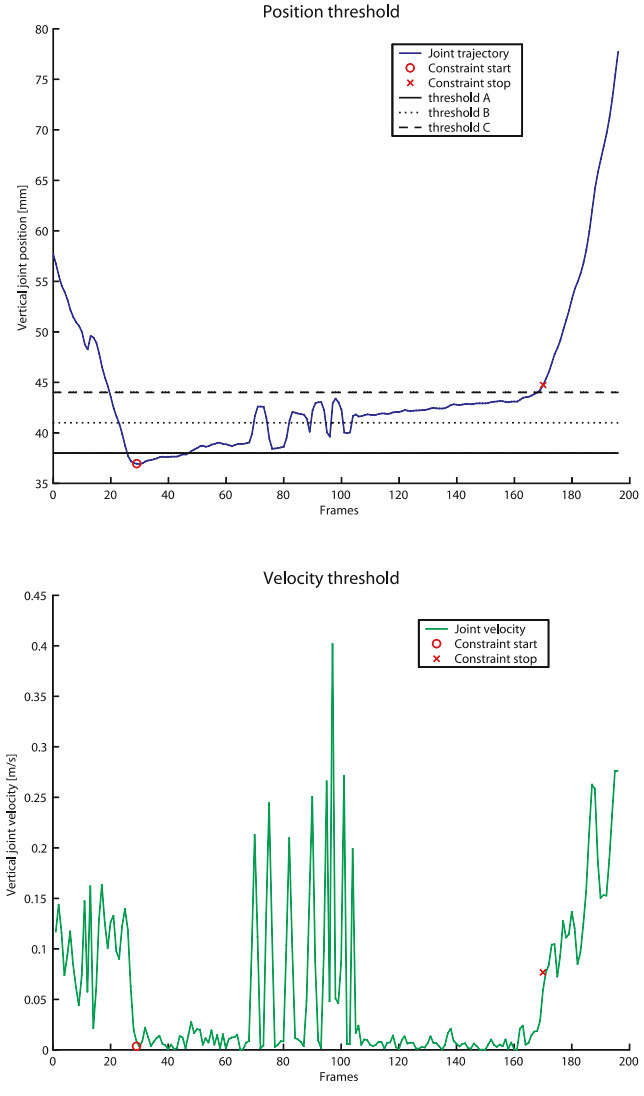

Fig. 4. Problem of threshold values for footplant detection on noisy data: the correct footplant start frame (circle) and end frame (cross). *Top:* the vertical trajectory of the foot. *Bottom:* the speed curve of the foot

checked. The foot is considered to be in contact with the ground when its position and linear speed are lower than specific thresholds.

However, this approach is not reliable for noisy or badly calibrated motion capture data. Figure 4 illustrates the vertical position and speed values of a foot in function of time, for noisy data. Here, the determination of a position threshold ε_p is difficult. If the threshold value is too small (see threshold A in Fig. 4, left), the footplant is too short. By increasing the value, the footplant may be split into several pieces (see threshold B), or may be too long (see threshold C). This footplant splitting problem exists for the setting of the speed threshold ε_s as well (Fig. 4, right).

To tackle this problem, the computed joint position at frame i may be replaced by an average of the $n+1$ positions from frame $i-\frac{n}{2}$ to frame $i+\frac{n}{2}$, using our anticipation method for future frames. In this way, the influence of peaks in noisy data is reduced. However, the difficulty remains to determine an appropriate n value. Our experiments have shown that this method fails to detect short footplant duration when increasing n.

In addition to the footplant splitting problem, the ε_p and ε_s thresholds are dependent on the input motions, due to their quality or their characteristics. During a single motion capture sequence, the quality may change, for example when one or more foot markers have been clumsily moved by the performer. Hence, the reference floor level moves up or down, entailing the need to modify ε_p. Different motion characteristics also result in adapting the ε_s threshold. For instance, Fig. 5 depicts the ankle speed over time, for a fast running and a slow walking motion. If a unique ε_s is used, the footplant fails at least for one of both motions. In addition, an appropriate threshold for the heel is not necessarily valid for the toe.

One solution consists in determining an empiric threshold function, whose given motion characteristics return the corresponding ε_p and ε_s values. However, this approach is unpractical due to the significant number of parameters influencing the thresholds: motion speed, motion capture quality, performer style, joint type.

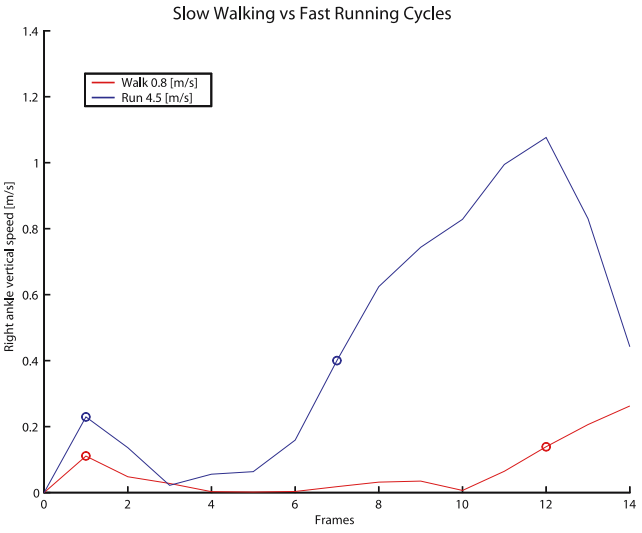

Fig. 5. Right ankle linear speed value comparison between a slow walking (red) and a fast running (blue) motion. The circles indicate the start and end of the footplant

4.2 Adaptive positional threshold

We overcome these difficulties by detecting footplants on locomotion patterns in an on-line manner, using our motion anticipation technique. Roughly speaking, our method uses a vertical position (height) threshold only. This latter is adaptively computed by investigating a specific fixed set of frames of the input patterns.

By avoiding the use of the speed threshold, we remove its problematic setting and we improve the detection method efficiency. Actually, when the foot position is under ε_p, the speed threshold allows us to identify unconstrained foot motion above the floor. This typically occurs during walking patterns, when the constrained foot leaves the floor and goes forward for the next step. Therefore, in our patterns we aim at defining a period of time during which we ensure that the foot is fixed to the ground if its position is under ε_p.

We make use of the observations performed on time-normalized walking and running patterns [13]. From them, we assert that for a given type of locomotion (walk or run), a foot joint constraint occurs *roughly* inside a fixed time interval, regardless of other motion characteristics. Therefore, our method investigates this time interval, referred to as the enclosed frames defined from $\mathcal{F}_b$ to $\mathcal{F}_e$, in order to detect the correct constraint, specified between the frame indexes $F'_s = \text{idx}(\mathcal{F}'_b)$ and $F'_e = \text{idx}(\mathcal{F}'_e)$. The $\text{idx}(\mathcal{F})$ function returns the F frame index of the $\mathcal{F}$ given frame.

Instead of using a fixed threshold value, ε_p is computed adaptively according to the current $\boldsymbol{L}(\boldsymbol{w})$ pattern. From it and for a given foot joint, we attach a set of frames $\boldsymbol{l} = (\mathcal{F}_b, \ldots, \mathcal{F}_e)$. Then, the global heights h_b and h_e of this joint are extracted at the respective bounding frames $\mathcal{F}_b$ and $\mathcal{F}_e$. We assume that the mean h of these two values is related to the pattern properties, namely its locomotion speed and style, in order to determine ε_p adaptively. To confirm, we compare various h values computed for the right heel and left toe, at different locomotion speeds and for five subject capture styles. Figure 6 illustrates the results for walking patterns and Fig. 7 for running ones, where each cell contains the corresponding mean height, in meters.

From these pictures, we infer that h is dependent on pattern parameters and, therefore, that the adaptive ε_p is a function of h. Clearly, the ε_p threshold has to be greater than the min minimal vertical position on $\boldsymbol{l}$, but smaller

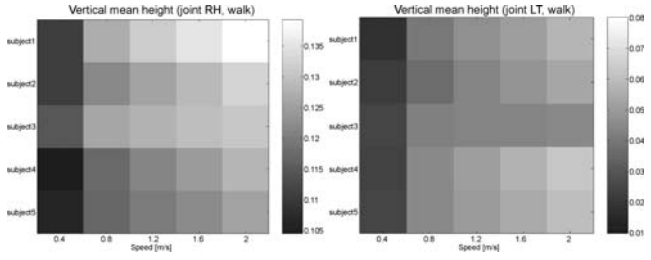

Fig. 6. Joint's mean height for bounding frames $\mathcal{F}_b$ and $\mathcal{F}_e$ for walking patterns at various speeds. *Left:* right heel comparison. *Right:* left toe comparison

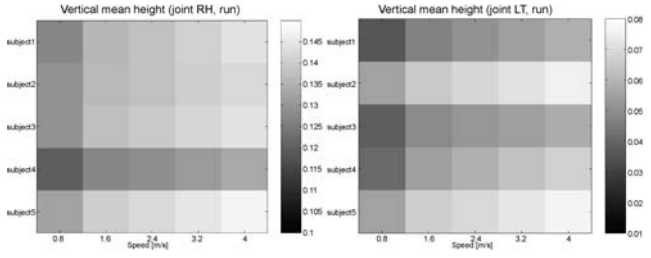

Fig. 7. Joint's mean height for bounding frames $\mathcal{F}_b$ and $\mathcal{F}_e$ for running patterns at various speeds. *Left:* right heel comparison. *Right:* left toe comparison

than h, as described in Eq. 8. We introduce the δ constant defined in $[0 \ldots 1]$, characterizing the correctness of the default bounding frames $\mathcal{F}_b$ and $\mathcal{F}_e$. As an example, by setting δ close to 1, it means that $\mathcal{F}_b$ and $\mathcal{F}_b'$ are close to each other. Figure 8 schematizes the method parameters.

$$\varepsilon_p = \min + \delta(h - \min). \tag{8}$$

In practice, the start frame index F_b' of a footplant for a given joint is detected as follows. We start by comparing the joint's vertical position at frame $\mathcal{F}_b$ (left circle on the curves in Fig. 9) with ε_p (dashed lines in Fig. 9) computed in Eq. 8. If the position at frame $\mathcal{F}_b$ is already below

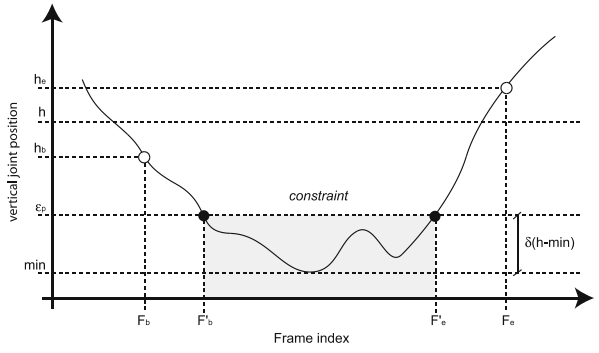

Fig. 8. Schematic representation of the constraint detection method parameters

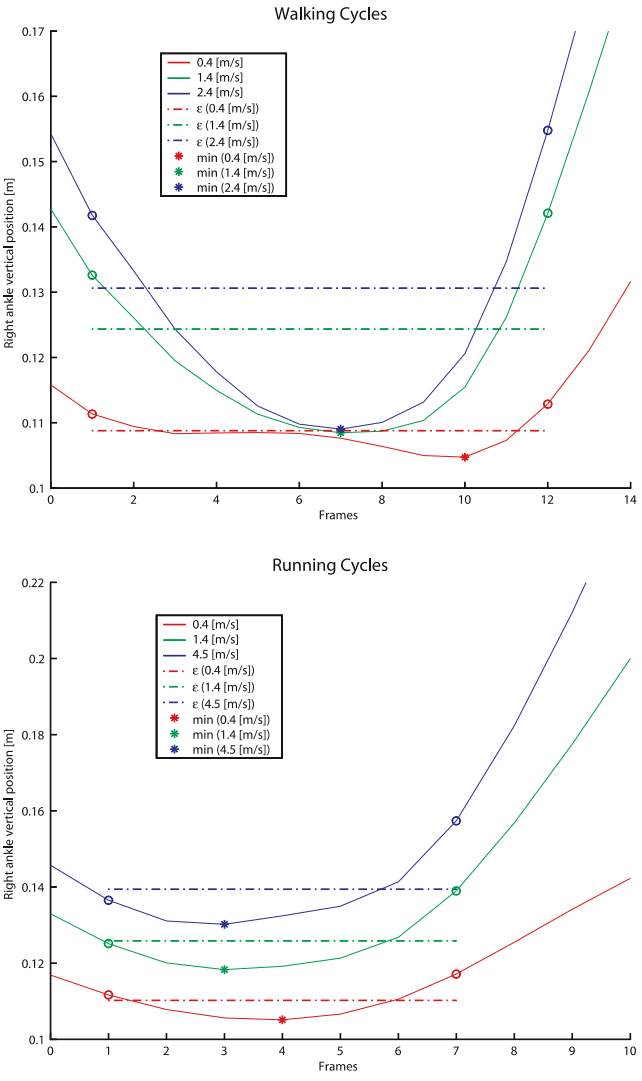

Fig. 9. Examples of our motion data for the right ankle vertical position. The circles indicate the bounding frames. *Top:* different speed values for walking. *Bottom:* different speed values for running

the threshold (e.g. the curve representing a 4.5[m/s] run on Fig. 9, bottom), idx($\mathcal{F}_b$) is assigned to F_b'. Otherwise, as long as position is above the threshold, the comparison is reported to the next frame. When the comparison is stopped, the current frame index is assign to F_b'. Analogously, F_e' is determined by starting at frame $\mathcal{F}_e$, and the threshold comparison is reported to the previous frame as long as the current frame value stays above the threshold. In this way, our method ensures that a footplant never splits off into small pieces.

4.3 On-line detection

At first sight, this detection method seems inappropriate for an on-line context. The movement needs to be known

in advance to obtain h_b and h_e, in order to determine the threshold. Thanks to the anticipation, each frame $\mathcal{F}_i$ of the $\boldsymbol{L}(\boldsymbol{w})$ cycle can be used at any time. Therefore, when a new $\boldsymbol{L}(\boldsymbol{w})$ is generated, the detection method is applied on the four joints describing both feet, providing a start and end frame index per constraint in the time-normalized pattern.

5 Footplant enforcement

We address the problem of re-positioning and enforcing a footplant by using the numerical IK algorithm incorporating the priorities presented in [1]. This method ensures that at least the high priority end-effector goal is achieved at best before considering lower priority constraints.

5.1 Inverse kinematics model

We attach two positional end-effectors on each foot, one for the heel and the other for the toe. In order to limit the computation time of the IK algorithm, we simplify the IK configuration by defining two independent IK chains, one for each leg. A chain contains six DOF's, from the hip joint to the toe joint.

Basically our footplant enforcement method works as follows. In an unconstrained state, the effector attached to a foot joint is driven by the locomotion pattern. As soon as its corresponding constraint has to be applied, the effector goal is re-positioned on the floor and remains fixed during the whole constraint duration. To ensure motion continuity, the effector trajectory is smoothed around the constraint, by defining ease-in and ease-out phases. We therefore apply the displacement map technique [5, 42] by describing a displacement map $\boldsymbol{d}(t)$ in order for the corrected motion $\boldsymbol{M}_{cor}(t) = \boldsymbol{M}(t) \oplus \boldsymbol{d}(t)$ to satisfy the detected footplants.

Let $\mathcal{C}$ be a constraint defining a position goal $\mathcal{C}_p$ for the effector E, active from t_1 to t_2. The desired effector trajectory $E(t)$ is built from its original trajectory $E_0(t)$ as:

$$E(t) = \begin{cases} E_0(t) + \left(1 - \Gamma\left(\frac{t-t_1+\sigma}{\sigma}\right)\right)\left(\mathcal{C}_p - E_0(t_1)\right) \\ \quad \text{for } t_1 - \sigma \leqslant t < t_1 \\ \mathcal{C}_p \\ \quad \text{for } t_1 \leqslant t \leqslant t_2 \\ E_0(t) + \Gamma\left(\frac{t-t_2}{\sigma}\right)\left(\mathcal{C}_p - E_0(t_2)\right) \\ \quad \text{for } t_2 < t \leqslant t_2 + \sigma \\ E_0(t) \\ \quad \text{otherwise} \end{cases}, \quad (9)$$

where σ is the ease-in and ease-out duration. The Γ function is the descending cubic step function, described in Eq. 10 and plotted in Fig. 10,

$$\Gamma(t) = 2t^3 - 3t^2 + 1 \quad \text{for } 0 \leqslant t \leqslant 1. \quad (10)$$

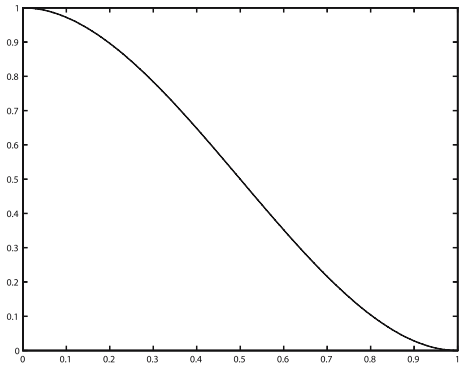

Fig. 10. The descending cubic step function used for the ease-in/out phases

Similarly to Lee et al. [24], we consider a motion as a set of independent character postures. At each time $t = t_i$, the original posture $\boldsymbol{M}(t_i)$ is modified by applying IK with end-effector trajectories defined in Eq. 9. Figure 11 illustrates the original and corrected vertical trajectories of the two effectors attached to the right foot. In addition, we choose to set highest priorities on the ankle's effectors: their constraints occur first and have significant visual impact. The IK solver therefore ensures that, at least, the ankle constraint is enforced, before trying to enforce the toe constraint. This effect is visible in Fig. 11 (right), as the corrected toe position trajectory goes down just after the release of its constraint. At this instant, the IK algorithm is not able to compute a solution ensuring a correct position for both effectors, due to an important foot sliding observed in the too noisy input data. However, the position error is under 0.01 m.

5.2 Ease-in with anticipation

The end-effector trajectories are constructed in order to ensure smooth motion generation around the constraint. Examining Eq. 9, one can observe that while performing the ease-in phase, the original position of the effector E_0 at (the future) time t_1 is required.

Thanks to the anticipation, this position and its corresponding constrained position $\mathcal{C}_p$ are computed as follows. At the beginning of the ease-in phase, namely at time $t_1 - \sigma$, the anticipated posture $\boldsymbol{M}(t_1)$ is computed to extract $E_0(t_1)$. This position, which corresponds to the original one of its attached foot joint, is re-positioned to be on the ground and stored in $\mathcal{C}_p$. Figure 12 illustrates the current posture (yellow) at time $t - \sigma$ and the anticipated one (green) at time t_1 for each effector.

5.3 Ease-in activation

Before enforcing a footplant, we have to know its starting time t_1 in order to activate its ease-in process at time $t_1 - \sigma$. This step is performed using our footplant detection method, applied differently according to three scenar-

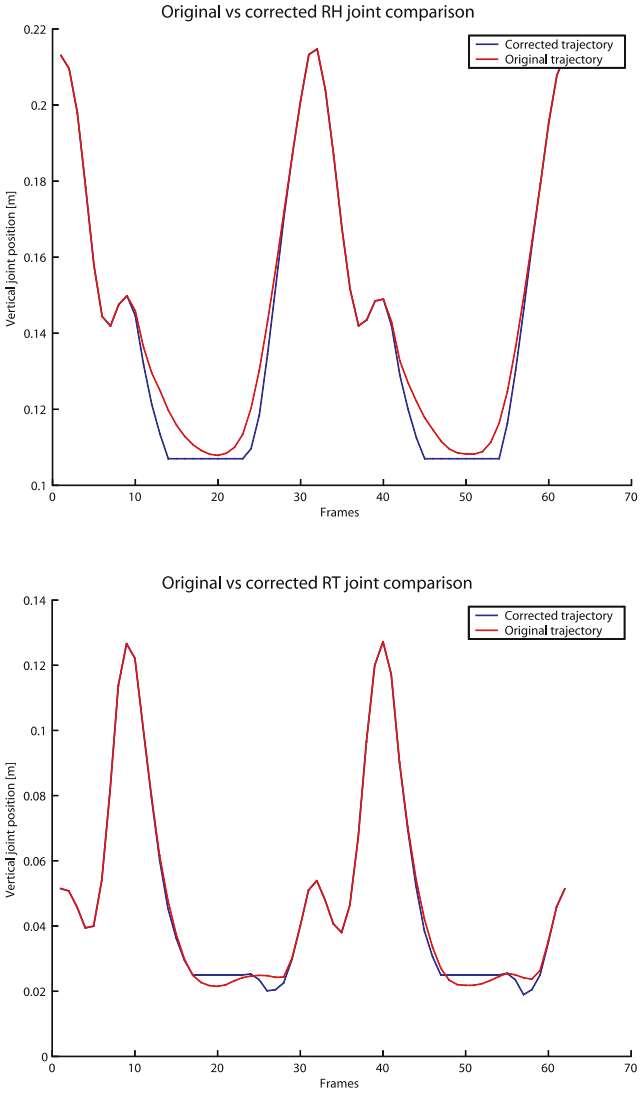

Fig. 11. Original $E_0(t)$ and corrected $E(t)$ vertical effector trajectories for the right foot.*Top:* effector attached to the ankle. *Bottom:* effector attached to the toe

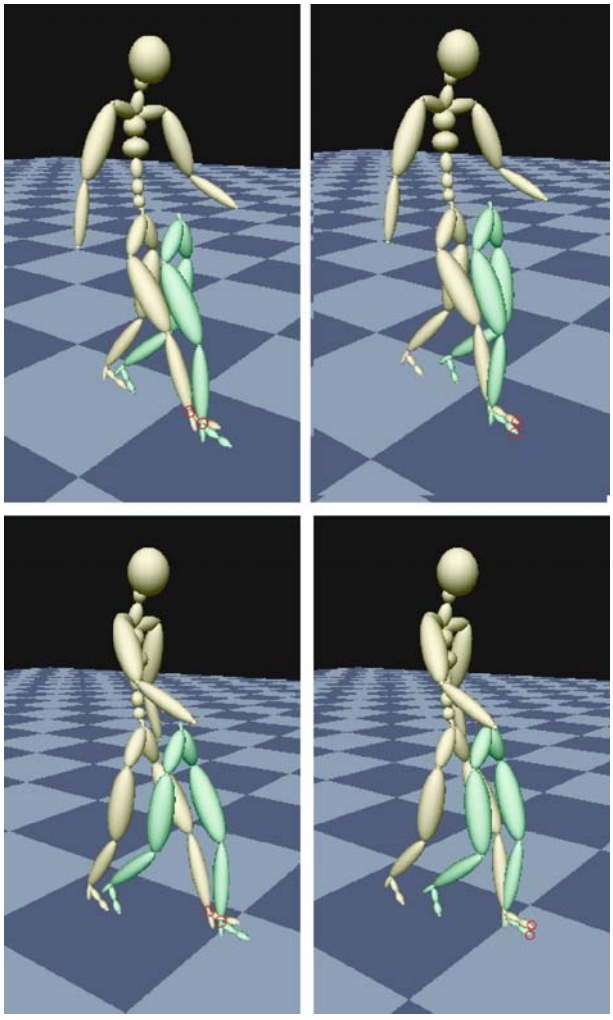

Fig. 12. The start of an ease-in phase for each effector. Current (yellow) and anticipated postures (green) are computed. *From left to right and top to bottom:* Effectors are attached to: right ankle, right metatarsal, left ankle and left metatarsal

ios: constant motion parameter, planned motion parameter variation and modification of a planned motion parameter variation.

5.3.1 Constant motion parameter

In this first case, the motion parameter vector $\boldsymbol{w}$ is constant over time. At the initialization stage, triggered at time t_i (when $\boldsymbol{w}$ is set), a new cycle $\boldsymbol{L}(\boldsymbol{w})$ is computed and footplants are detected with our method. To systematically check if there is already a footplant to activate, the $[t_i \ldots t_i + \sigma]$ time interval is regularly sampled into $n\Delta t$ intervals. For each j-th interval, the time $t_i + j\Delta t$ is used to compute its corresponding locomotion phase φ_j. This value is multiplied by the $\boldsymbol{L}(\boldsymbol{w})$ frame number to obtain the corresponding $\text{idx}(\boldsymbol{F}_j)$ frame index in the normalized time. If this frame index is more than or equal to the F'_b of a foot joint, a constraint occurs at time $t_i + j\Delta t$. As $t_i + j\Delta t < \sigma$, the ease-in phase is immediately activated. After this initialization, at any t time, the future $t + \sigma$ time is transformed into the normalized time to detect whether a constraint needs to be activated.

5.3.2 Planned motion parameter variation

In this case, a new $\boldsymbol{w}'$ has to be reached within a given duration Δm, involving a linear change of $\boldsymbol{w}'$ over time. We therefore assign a $\boldsymbol{w}'_j$ parameter vector to each future time $t_i + j\Delta t$, for which a new $\boldsymbol{L}(\boldsymbol{w}'_j)$ locomotion pattern is generated and footplants are detected. Then, analogously to the previous case, the corresponding $\text{idx}(\boldsymbol{F}_i)$ frame index for time $t_i + j\Delta t$ is extracted and compared to F'_b. In

practice, these operations are too expensive for real-time. Hence, frame indexes F'_b and F'_e are pre-computed by applying the detection method on motions with various parameter vectors (7 and 11 different normalized speed values for walking and running patterns, respectively). Theses frame indexes are then used to determine the ones of $\boldsymbol{L}(\boldsymbol{w}'_i)$ by applying linear interpolation. Our results confirm the pertinence of this approach as can be seen in Sect. 6.

5.3.3 Modification of a planned motion parameter variation

This last case handles the situation in which a planned variation is interrupted by a new one at time t'. For each future $t_i + j\Delta t$ time, a new locomotion pattern is computed and footplants are detected, similarly to the previous case. However, if an effector is in an ease-in phase when the variation is interrupted, its trajectory has to be carefully modified, as its goal position $\mathcal{C}_p$ and time t_1 have changed, due to the new motion parameter configuration. Figure 13 illustrates this modification schematically. First, the effector trajectory $E_0(t)$, depicted by the blue line, is continuously modified to reach the $\mathcal{C}_p$ constraint. This creates a new trajectory $E(t)$, depicted by the dashed blue line. Due to the motion parameter changes at a time t' during the ease-in phase, a new $E'_0(t)$ original effector trajectory is provided (green curve). We therefore have to modify $E(t)$ in order to reach the new goal position $\mathcal{C}'_p$ at a new time t'_1. We modify Eq. 9 in order for the effector trajectory, modified from $E'_0(t)$ to remain smooth. Therefore, from time t' the effector trajectory, depicted by the red dashed line, is updated as follows:

$$E(t) = E'_0(t) + \left(1 - \Gamma\left(\frac{t-t'}{t'_1-t'}\right)\right)\left(E'_0(t') - E(t')\right) + \Gamma\left(\frac{t-t'}{t'_1-t'}\right)\left(\mathcal{C}'_p - E'_0(t'_1)\right). \quad (11)$$

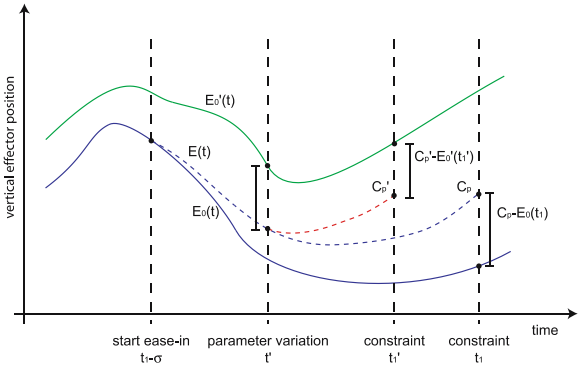

Fig. 13. Step variation during the ease-in phase. The blue and the green curves correspond to the original effector trajectories, the dashed blue and red curves to the modified ones

Figure 14 illustrates the smoothness of the modified effector trajectories. The animation is generated first by continuously increasing the walking speed. This process is then interrupted by another parameter variation, consisting in a transition to a running motion.

6 Experimental results

First we describe the different components of our system prior to presenting some on-line application examples.

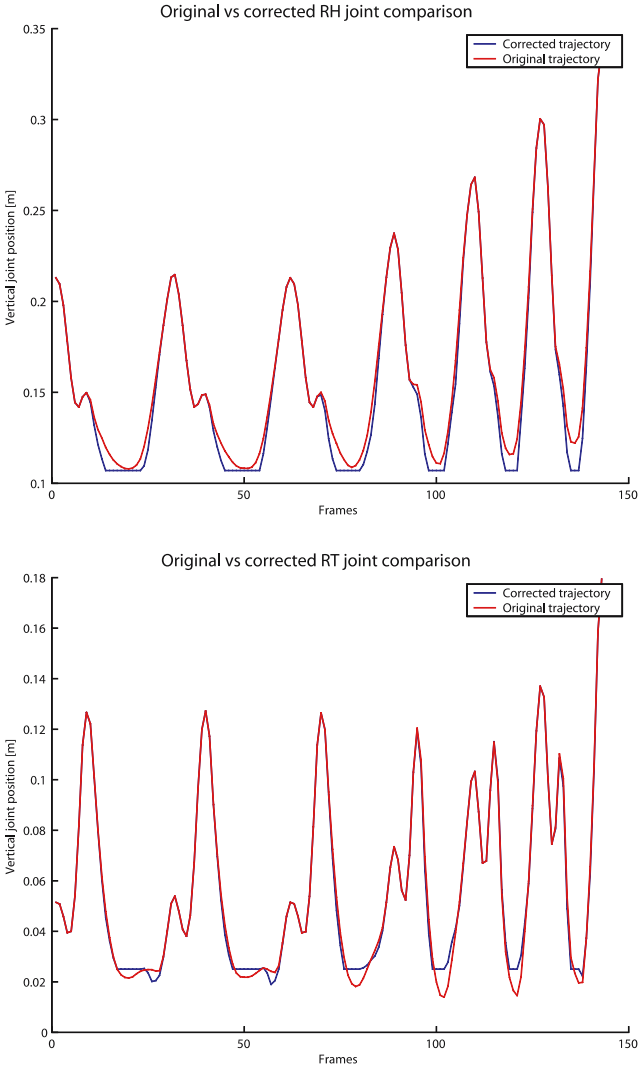

Fig. 14. Original $E_0(t)$ and corrected $E(t)$ vertical effector trajectories for the right foot. The animation first describes a walk that continuously speeds up before being interrupted by a continuous transition to a run. *Top:* effector attached to the ankle. *Bottom:* effector attached to the toe

6.1 Putting it all together

For our experiments, we used a motion capture database composed of 180 walking and 140 straight line running cycles, differing in performer styles and locomotion speeds. A locomotion cycle is generated in 0.3 ms on a 2.2 CPU GHz machine, and is composed of 25 frames, animating an H-ANIM body with 60 DOF's. To speed up the motion anticipation computation, only lower body postures are computed. All the 25 posture configurations, including the joint global positions, are calculated within 1.3 ms.

Then enclosed frame set $\mathcal{I}$ has to be determined, for walking and running cycles, and for each joint needing floor contact detection. We use the values summarized in Table 1. These frames' indexes define the interval in which the constraint detection algorithm is performed. In our experiments, δ is equal to 0.4, and the exact starting and ending constrained index frames F'_b and F'_e for all joints, are computed in less than 0.05 ms.

After detecting the constraint relative to an effector, the method continuously checks whether a constraint has to be enforced in the near future, in order to activate the ease-in phase. The computation time for footplant enforcement depends on the number of joints whose trajectory is modified by the IK algorithm. On average, 1.8 ms are necessary to smoothly enforce the two footplants, representing four constraints.

Finally, we have tested our method with a continuous motion parameters variation and we obtained a maximal computational cost of 4.7 ms per frame. This is the most expensive case, as it entails pattern computation for each future time $t_i + j\Delta t$.

6.2 Foot sliding correction

The first experiment consists in cleaning up the foot sliding and penetration into the ground. Our original motion shows some artifacts, as illustrated at the top of Fig. 15. This is due to the input motion capture data and the PCA algorithm used to reduce data dimensionality. The footplants are then correctly detected and re-positioned, no more foot sliding is perceptible and IK does not introduce discontinuities during the animation.

Table 1. Frame interval for each constrained joint, observed on locomotion cycles of 25 frames

Joint's name	idx($\mathcal{F}_b$) (walk)	idx($\mathcal{F}_e$) (walk)	idx($\mathcal{F}_b$) (run)	idx($\mathcal{F}_e$) (run)
Right ankle	1	12	1	7
Right metatarsal	3	15	1	9
Left ankle	13	26 (mod 25)	12	19
Left metatarsal	15	29 (mod 25)	13	22

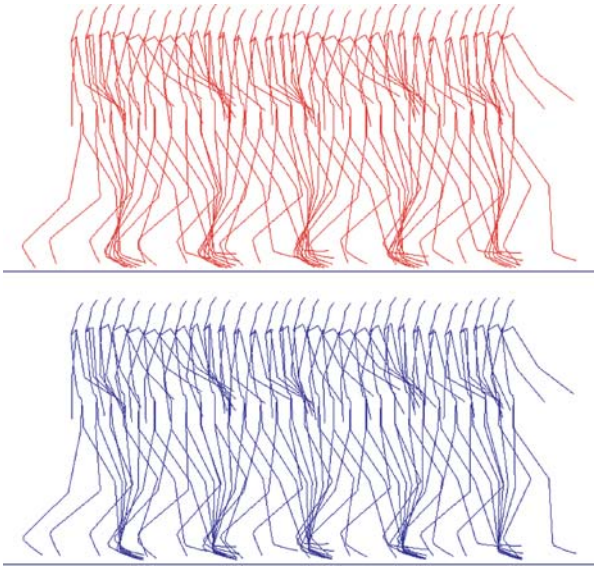

Fig. 15. Foot sliding clean up. *Top:* original motion. *Bottom:* modified motion

6.3 Stylistic edition

Another application of our methodology is to create small stylistic variations of the generated motion, by modifying the location of one (or more) end-effector during its constraint. In our example, the right toe constrained position is displaced, as depicted in Fig. 16, to produce a pigeon-toed effect. At the beginning of the ease-in phase, we modify the anticipated toe joint position corresponding to the start of the constraint. This modification consists in

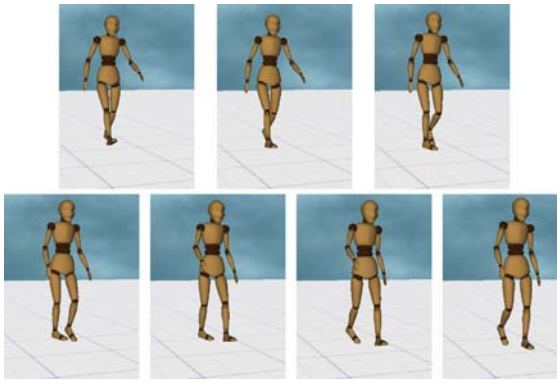

Fig. 16. Modification of the right toe position produces a right pigeon-toed walking motion

a rotation, having its center placed at the right ankle joint, around a perpendicular axis to the ground. The rotation angle is set in order to direct the toe towards the left ankle.

6.4 Continuous parameter variation

The on-line reactivity to user parameter modification is an important aspect of our method. At any time, the user or the AI driving the autonomous agent can define a new motion parameter set that has to be reached in a given period of time. The smooth parameter evolution is performed by a controller. In our implementation, we control the parameter evolution linearly. Imagine a slow walking pattern. By setting a higher speed value and changing the locomotion from walk to run, the motion will simultaneously accelerate and start to run (see Fig. 17). Therefore, the motion is continuously modified while guaranteeing that the footplants detection and enforcement are still coherent with the corresponding parameter values.

6.5 Curved path

In the last experiment, our straight-line locomotion cycles are adapted so as to produce a curved path, illustrated in Fig. 18. The motions' angular speed is continuously changed and, therefore, modifies the yaw angle of the root node. Thanks to our method, the foot remains fixed to the floor during the constraint, while the root rotates. When the constraint is relaxed, the ankle first smoothly reaches its original trajectory. The toe is fixed to the ground a little longer, allowing it to rotate around its position.

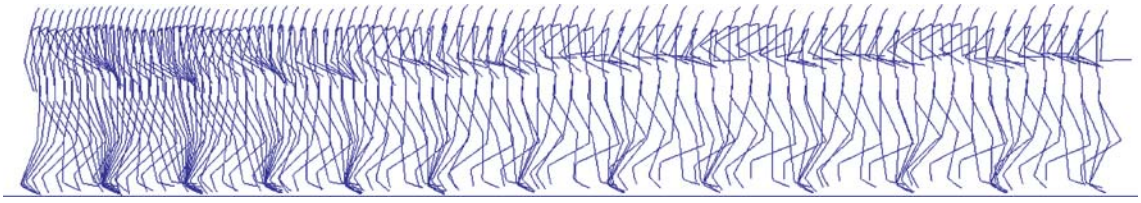

Fig. 17. Continuous parameter variation, from a walk at 0.8 m/s to a run at 1.9 m/s (parameter variation within 3 s)

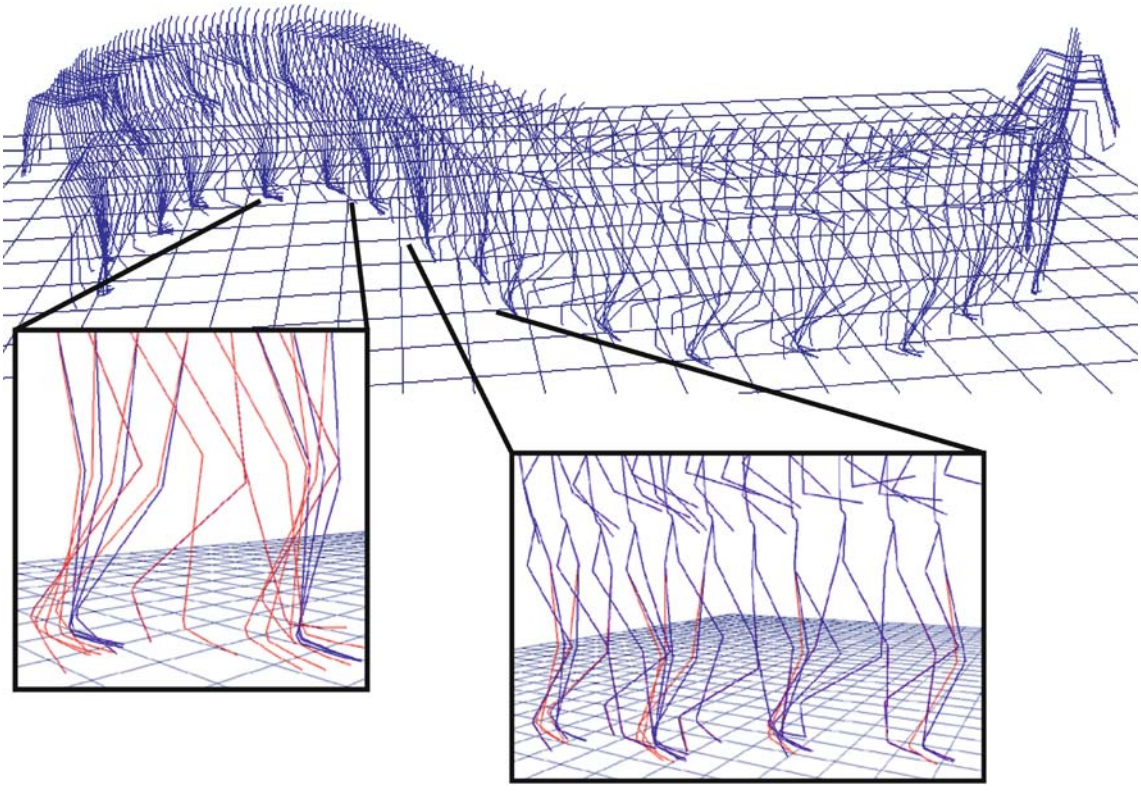

Fig. 18. Walking to running with angular speed variation. The red body in the zoomed frames corresponds to the original motion

7 Conclusion and discussion

To correct a keyframed animation, an animator has to detect footplants manually by labeling the constrained frames. Then the enforcement of these constraints is performed using methods either based on numerical or analytical inverse kinematics. In this paper, we improve this process by presenting an on-line animation system based on the generation of anticipated postures to detect, reposition and apply footplants.

The detection algorithm is robust because it considers the properties of the motion (e.g. speed, type of locomotion, human size) from which the foot constraints have to be extracted on-line. For a given motion, we define adaptively only one vertical position threshold value for each joint describing a foot. The threshold intrinsic normalization allows the method to work for any human size. A numerical IK solver with priorities is applied to re-position and enforce a footplant, described by two end-effectors. The priorities set on these end-effectors ensure that at least one goal position is exactly reached. Our method maintains smooth end-effector trajectories by anticipating postures at a constraint.

The generated motion continuity has also been evaluated. We can modify the goal position for a given footplant, allowing the introduction of some smooth stylistic variations in the original motion. We also modify the root vertical orientation to perform curved motion, leading to very satisfying results. The coherence of the method is ensured, as the motion parameters vary continuously. Finally, the method is not computationally expensive, because even in the worst situation, where parameters have to be updated at each time step, a frame update is performed in less than 5 ms.

A limitation of our method concerns the IK method, which is per-frame based. Theoretically, discontinuities may occur, as the current modified frame is computed relative to the original one. Therefore, it could happen that the IK solver finds very different resulting postures for two original consecutive frames, leading to discontinuities between the modified frames. In practice, we never observed motion discontinuities during our experiments. An alternate approach would be to compute the current modified frame by applying IK to the previously modified one.

In our example, the curved motion generation may produce important end-effector position modifications in the case of important angular speed with low linear speed. In such an extreme situation the constraint duration of the foot on the outer curve could be reduced.

In the future, we plan to apply this method to other motion pattern classes, such as jumping or tiptoeing dance motions. In such a context, the footplant detection method has to be improved, as the enclosing frame interval is more difficult to establish. In addition, the order of the foot constraints may be inverted, i.e. by constraining the toe before the ankle. In this case, our IK method would simply exchange the priorities on the corresponding effectors.

Finally, we believe that the introduced method of motion anticipation is a promising starting point for smoothly avoiding potential collisions with nearby obstacles. Indeed, we can benefit from the possibility of predicting postures to check their proximity to other key elements in the environment in addition to the floor.

Acknowledgement The authors would like to thank Benoît Le Callennec for Maya support and IK help, as well as Helena Grillon for careful proofreading.

The present research and the IK library used for the posture correction were funded by the Swiss National Science Foundation. The Maya licenses have been granted by Alias through their research donation program.

References

1. Baerlocher, P., Boulic, R.: An inverse kinematic architecture enforcing an arbitrary number of strict priority levels. Visual. Comput. **20**(6), 402–417 (2004)
2. Bindiganavale, R., Badler, N.: Motion abstraction and mapping with spatial constraints. Lecture Notes in Computer Science, Vol. 1537, pp. 70–83 (1998)
3. Boulic, R., Ulciny, B., Thalmann, D.: Versatile walk engine. J of Game Development **1**(1), 29–50 (2004)
4. Bruderlin, A., Calvert, T.: Knowledge-driven, interactive animation of human running. In: Graphics Interface '96, pp. 213–221, Canadian Information Processing Society. Toronto, Ontario, Canada (1996)
5. Bruderlin, A., Williams, L.: Motion signal processing. In: Proceedings of ACM SIGGRAPH, Annual Conference Series, pp. 97–104 (1995)
6. Butz, M., Sigaud, O., Gerard, P.: Anticipatory Behavior in Adaptive Learning Systems. Springer, Berlin Heidelberg New York (2003)
7. Choi, K., Ko, H.: Online motion retargetting. J. Visual. Comput. Anim. **11**, 223–235 (2000)
8. Choi, M., Lee, J., Shin, S.: Planning biped locomotion using motion capture data and probabilistic roadmaps. ACM Trans. Graph. (2003)
9. Chung, S., Hahn, J.: Animation of human walking in virtual environments. In: Proceedings of Computer Animation, Geneva, IEEE Computer Society (1999)
10. Conde, T., Thalmann, D.: An artificial life environment for autonomous virtual agents with multi-sensorial and multi-perceptives features. Comput. Anim. Virtual World **15**, 311–318 (2004)
11. Girard, M.: Interactive design of 3-D computer-animated legged animal motion. In: Proceedings of ACM Symposium on Interactive 3D Graphics, pp. 131–150 (1987)
12. Glardon, P., Boulic, R., Thalmann, D.: A coherent locomotion engine extrapolating beyond experimental data. In: Proceedings of Computer Animation and Social Agent, pp. 73–83, Geneva (2004)
13. Glardon, P., Boulic, R., Thalmann, D.: On-line adapted transition between locomotion and jump. In: Proceedings of Computer Graphics International, pp. 44–49, IEEE Computer Society (2005)
14. Gleicher, M.: Motion editing with spacetime constraints. In: Proceedings of ACM Symposium on Interactive 3D Graphics, pp. 139–148 (1997)

15. Gleicher, M.: Comparing constraint-based motion editing methods. Graphical Models **63**(2), 107–134 (2001)
16. H-ANIM: Humanoid animation working group. www.hanim.org (2005)
17. Hreljac, A., Marshall, R.: Algorithms to determine event timing during normal walking using kinematic data. J. Biomech. **33**(6), 783–786 (2000)
18. Ko, H., Badler, N.: Animating human locomotion with inverse dynamics. IEEE Comput. Graph. Applic. **16**(2), 50–58 (1996)
19. Kovar, L., Gleicher, M.: Flexible automatic motion blending with registration curves. In: Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 214–224 (2003)
20. Kovar, L., Schreiner, J., Gleicher, M.: Footskate cleanup for motion capture editing. In: Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 97–104 (2002)
21. Labbé, V., Sigaud, O., Codognet, P.: Anticipation of periodic movements in real time 3D environments. In: Proceedings of ABiALS Workshop, Los Angeles (2004)
22. Le Callennec, B., Boulic, R.: Interactive motion deformation with prioritized constraints. In: Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation (2004)
23. Lee, J., Chai, J., Reitsma, P., Hodgins, J., Pollard, N.: Interactive control of avatars animated with human motion data. In: Proceedings of ACM SIGGRAPH, Annual Conference Series (2002)
24. Lee, J., Shin, S.: A hierarchical approach to interactive motion editing for human-like figures. In: Proceedings of ACM SIGGRAPH, Annual Conference Series, pp. 39–48 (1999)
25. Liu, K., Popović, Z.: Synthesis of complex dynamic character motion from simple animations. In: Proceedings of ACM SIGGRAPH, Annual Conference Series, pp. 408–416 (2002)
26. Maya®: Alias systems corp. www.alias.com (2005)
27. Menardais, S., Kulpa, R., Arnaldi, B.: Synchronisation for dynamic blending of motions. In: Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation (2004)
28. Multon, F., France, L., Cani-Gascuel, M., Debunne, G.: Computer animation of human walking: a survey. J. Visual. Comput. Anim. **10**(1), 39–54 (1999)
29. Park, S., Shin, H., Shin, S.: On-line locomotion generation based on motion blending. In: Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation (2002)
30. Popović, Z., Witkin, A.: Physically based motion transformation. In: Proceedings of ACM SIGGRAPH, Annual Conference Series, pp. 11–20 (1999)
31. Reynolds, C.: Steering behaviors For autonomous characters. In: Proceedings of Game Developers Conference, pp. 763–782 (1999)
32. Rose, C., Cohen, M., Bodenheimer, B.: Verbs and adverbs: Multidimensional motion interpolation. IEEE Comput. Graph. Applic. **18**(5), 32–41 (1998)
33. Rose, C., Sloan, P., Cohen, M.: Artist-directed inverse-kinematics using radial basis function interpolation. In: Proceedings of Eurographics, vol. 20(3) (2001)
34. Safonova, A., Hodgins, J., Pollard, N.: Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. In: Proceedings of ACM SIGGRAPH, Annual Conference Series (2004)
35. Salvati, M., Le Callennec, B., Boulic, R.: A generic method for geometric contraints detection. In: Proceedings of Eurographics, short presentation (2004)
36. Shin, H., Kovar, L., Gleicher, M.: Physical touch-up of human motions. In: Proceedings of Pacific Graphics, pp. 194–203, IEEE Computer Society (2003)
37. Shin, H., Lee, J., Shin, S., Gleicher, M.: Computer puppetry: an importance-based approach. ACM Trans. Graph. **20**(2), 67–94 (2001)
38. Sun, H., Metaxas, D.: Automating Gait Generation. In: Proceedings of ACM SIGGRAPH, Annual Conference Series (2001)
39. Unuma, M., Anjyo, K., Takeuchi, R.: Fourier principles for emotion-based human figure. In: Proceedings of ACM SIGGRAPH, Annual Conference Series, pp. 91–96 (1995)
40. van de Panne, M.: From footprints to animation. Comput. Graph. Forum **16**(4), 211–223 (1997)
41. Veloso, M., Stone, P., Bowling, M.: Anticipation: a key for collaboration in a team of agents. In: Proceedings of Conference on Autonomous Agents (1998)
42. Witkin, A., Popović, Z.: Motion warping. In: Proceedings of ACM SIGGRAPH, Annual Conference Series, pp. 105–108 (1995)
43. Wooten, W., Hodgins, J.: Simulating leaping, tumbling, landing and balancing humans. In: Proceedings of IEEE International Conference on Robotics and Automation (2000)
44. Yamane, K., Nakamura, Y.: Natural motion animation through constraining and deconstraining at will. IEEE Trans. Visual. Comput. Graph. **9**(3), 352–360 (2003)

PASCAL GLARDON is a research assistant and pursuing a PhD in the Virtual Reality Laboratory at the Swiss Federal Institute of Technology (EPFL). His research interests include the generation and the control of virtual character animations, as well as their interaction with virtual environments. He received his Master's degree in Computer Science in 2000 at the Swiss Federal Institute of Technology, Zurich (ETHZ). In 2001 he worked as a software engineer at a company specialized in security document printing.

RONAN BOULIC is a Senior Researcher, Lecturer and PhD Director at the Swiss Federal Institute of Technology, Lausanne (EPFL). He is working in the Virtual Reality Lab and his research interests include realistic motion synthesis for virtual humans and robot. He received his PhD degree in Computer Science in 1986 from the University of Rennes, France, at the INRIA-IRISA Research Institute. He received his Habilitation degree from the University of Grenoble, France, in March 1995. Ronan Boulic is co-author of 83 research papers, among which 21 have appeared in international peer-reviewed journals. He served as paper chair of the 2004 SIGGRAPH-Eurographics Symposium on Computer Animation. He is a senior member of IEEE and a member of ACM and Eurographics.

DANIEL THALMANN is Professor and Director of The Virtual Reality Lab (VRlab) at EPFL, Switzerland. He is a pioneer in research on virtual humans. His current research interests include real-time virtual humans in virtual reality, networked virtual environments, artificial life, and multimedia. Daniel Thalmann has been Professor at The University of Montreal and Visiting Professor/Researcher at CERN, University of Nebraska, University of Tokyo, and Institute of System Science in Singapore.

He is coeditor-in-chief of Computer Animation and Virtual Worlds (formerly Journal of Visualization and Computer Animation), and member of the editorial board of the Visual Computer and four other journals.

Daniel Thalmann has been member of numerous program committees, the program chair of several conferences and chair of the Computer Graphics International '93, Pacific Graphics '95, ACM VRST '97, and MMM '98 conferences. He was program cochair of IEEE VR 2000. He has also organized five courses at SIGGRAPH on human animation and crowd simulation.

Daniel Thalmann has published more than 400 papers on graphics, animation, and virtual reality. He is coeditor of 30 books included the recent "Handbook of Virtual Humans", published by John Wiley and Sons and coauthor of several books. He was also codirector of several computer-generated films with synthetic actors, including a synthetic Marilyn shown on numerous TV channels all over the world.

He received his PhD in Computer Science in 1977 from the University of Geneva and an Honorary Doctorate (Honoris Causa) from the University Paul-Sabatier in Toulouse, France, in 2003.