Visual Comput (2007) 23: 689–695
DOI 10.1007/s00371-007-0135-6

ORIGINAL ARTICLE

Etienne Lyard
Nadia Magnenat-Thalmann

A simple footskate removal method for virtual reality applications

Published online: 9 June 2007

E. Lyard (✉) · N. Magnenat-Thalmann
MIRALab, University of Geneva
Battelle, Batiment A
7 Route de Drize
CH-1227 Carouge
Switzerland
Lyard@miralab.unige.ch

Abstract Footskate is a common problem encountered in interactive applications dealing with virtual character animations. It has proven difficult to fix without the use of complex numerical methods, which require expert skills for their implementations, along with a fair amount of user interaction to correct a motion. On the other hand, deformable bodies are being increasingly used in virtual reality (VR) applications, allowing users to customize their avatar as they wish. This introduces the need of adapting motions without any help from a designer, as a random user seldom has the skills required to drive the existing algorithms towards the right solution. In this paper, we present a simple method to remove footskate artifacts in VR applications. Unlike previous algorithms, our approach does not rely on the skeletal animation to perform the correction but rather on the skin. This ensures that the final foot planting really matches the virtual character's motion. The changes are applied to the root joint of the skeleton only so that the resulting animation is as close as possible to the original one. Eventually, thanks to the simplicity of its formulation, it can be quickly and easily added to existing frameworks.

Keywords Computer animation · Motion retargeting · Virtual reality

1 Introduction

Virtual reality often uses motion captured data in order to animate virtual humans who populate these environments. As it is very time consuming to capture high quality motions, an existing clip is most likely to be reused for various applications. Thus, unless a clip is applied to only one given character, the motion must be adapted in order to match the new body's characteristics and dimensions. Even though very efficient and reliable methods have already been proposed to solve this problem, our intention is to focus on VR applications while taking into account the trade offs between the quality of the results, the computational complexity and the ease of implementation. Such applications commonly have databases of 3D bodies, or even systems able to generate new bodies according to user requirements [20]. These bodies are animated on-the-fly according to user interaction by applying motions taken from a database or generated at runtime [6, 12, 13].

Users seldom have the skills required in order to perform the motion retargeting themselves and moreover this is a problem they simply do not want to acknowledge. For this reason, we propose here a new footskate removal algorithm which can accommodate most of the usual skeleton simplification and low quality motion clips that are used by VR applications and yet still deliver high quality and reliable results with no user interaction.

In order to face the degradation of the models imposed by the VR designers, our algorithm does not rely on the skeletal animation as it is usually done, but rather on the skin motion. Indeed, it isn't realistic to rely on a model with two points of contact for the skeleton's feet as VR characters rarely have such a feature. Moreover, as the motions are extremely simplified, it is difficult to robustly

estimate these points of contact from the motion data. Considering the skin motion itself directly gives a multiple points of contact model, thus avoiding this difficult estimation step and ensuring a higher quality final animation.

2 Related works

In real human walking motion, feet remain in a fixed position on the ground during small periods of time. Footskate is a common error in character animation when a foot appears to slide on the ground plane. It is introduced by the fact that even tough motions are recorded very accurately on real subjects [17, 24] or are very well designed by hand, a particular movement only matches the specific subject from which it was recorded or the character for which it was originally designed. Thus, when applying a motion to a different body, the global rotations and translations that position the character in the scene no longer match the motion of the limbs.

Numerous approaches have been developed in order to address this mapping. It can be thought of as finding a motion that optimizes a given criterion (e.g., be as close as possible to the original motion) while complying with a set of constraints (e.g., no footskate should remain). Numerical methods estimate the required correction using global optimization algorithms (often referred to as space-time optimization [26]) and inverse kinematics (IK). Such approaches have the advantage of taking the entire motion into account to perform the retarget, and hence deliver high quality results. Gleicher [8, 9] was the first to apply this kind of optimization to the problem of motion editing and retargeting. He managed to keep the high frequency features of a motion by putting a spline layer between the optimization algorithm and the actual values of the motion parameters. Lee and Shin [16] further refined this idea by adopting not only a single spline per piece of motion, but hierarchies of splines which made the final motion easier to control. Choi and Ko [5] used inverse rate control and managed to achieve the retargeting in real time, while Boulic et al. [3] proposed an approach based on null-space projections, which allows to control a strict number of priority for the constraints. All of the works mentioned above, even though not explicitly focused on footskate, can be employed to address this issue. However, due to the large non-linear systems that must be solved at each iteration of the algorithm, they are quite slow compared to other techniques.

Analytical IK methods were developed to quicken and facilitate the manipulation of human figures. Tolani et al. [23] demonstrated how to analytically manipulate a human limb with 7 degrees of freedom. This scheme was later employed by several works, which either used it alone [14], or in conjunction with optimization techniques, thus creating a class of hybrid systems [21].

This idea of hybrid systems, combining several techniques in order to speed up the computation was also exploited by the most recent works in motion adaptation [1, 15, 22]. However, these works do not address the problem of footskate removal. Rather, they consider the physical aspects of a motion [1, 22] or an innovative way to interact with a given clip [15]. Such approaches, even though they may be able to fix the footskating are not designed for this purpose, and improvements may still be found for this specific goal. The only recent work directly dealing with footskate is from Glardon et al. [7] and takes only the skeleton into account along with the use of a numerical IK method to reposition the feet where they should stay put. Unlike previous approaches, ours does not intend to make the motion comply with predefined constraints such as a given foot planting. Rather, we aim to calculate the actual displacement which corresponds to the limbs motion, thus modifying the character's path. This may appear strange at first, but numerous VR applications, for instance a virtual try on, do not focus on the character's path as much as on the actual movements performed by its limbs. Indeed, in order to see how a garment fits on a body or another, the motion of their limbs must be similar whereas their actual path is not of such great importance. Another novel aspect of our approach is that it relies fully on the skin to carry out the computation. This allows the use of any kind of skeletal hierarchy, whether it features both balls and heels joints or not [10]. Finally, we also present a new way to estimate the foot planting. Although quite simple, it has proven to be very robust during our tests on catwalk and tryout animations.

The remainder of this paper is organized as follows: Sect. 3 gives an outline of our method, which is explained further from Sect. 3.1 to 3.3. Section 4 exposes the results we obtained using this method and eventually these results are discussed in Sect. 5.

3 A footskate removal method for simplified characters

VR applications are quite different from entertainment productions. Indeed, VR aims at immersing a user in a real time environment as similar as possible to reality. Thus these applications are usually highly demanding in terms of performances. To meet these requirements, most labs have built their own VR platforms [19, 25], which allow to re-use previously developed components. These frameworks are optimized to ensure maximum performances at runtime, and most of the models assume numerous simplifications in order to allow for rich environments. For instance, VHD++ developed jointly by MIRALab and VRLab does not allow for the resizing of a skeleton at runtime, which leaves the method proposed by [14] unusable within this context. Moreover, VR developers rarely focus on side artifacts and usually prefer to concentrate

on the final user experience, which prevents them from implementing complex methods for only a small benefit. The method we propose here complies with the two previous statements in the sense that it can accommodate rigid skeletons and is very easy to implement. Thus it can be added with only little time and effort to an existing VR framework.

The method can be summarized as follows: first foot plants are estimated, i.e., when should each foot be planted on the ground. Unlike most of the other approaches our algorithm does not constrain the *location* where a foot is planted, but rather the *frame* at which this should happen. This way, the motion itself remains as close as possible to the original, only the path followed by the character is subject to a scale.

The next stage is divided into two separate processes. A first treatment corrects the character's motion along the horizontal axis, and a second one adapts its vertical displacement. This choice was motivated by the observation that in most VR applications, the feet of the character remain rigid throughout the animations. This happens because the feet are attached to only one joint, again for optimization reasons. Thus, as the skin is not deformed accurately, the feet will somewhat penetrate the ground regardless of the retargeting process applied. To correct this, our method accurately plants the feet where they should be in the horizontal plane, while in the vertical direction it minimizes the distance between the ground and the planted foot.

3.1 Feet motion analysis

Depending on the quality of a motion clip, it can be quite tricky to estimate how and when to plant a foot. If the motion is perfect, it should be enough to simply observe that a foot remaining static may be planted. However, feet are rarely motionless. Moreover most of the clips that are repeatedly used in reality are far from perfect and therefore such a simple criterion is insufficient. Previous works focused on proximity rules to extract the planting [2], k-nearest neighbors classifiers [11] or adaptive threshold imposed on the location and velocity of the feet [7]. All the above-mentioned approaches require some human interaction to perform the estimation: even [7] requires at least to specify the kind of motion being performed. As we mentioned previously, our goal is to discard this interaction stage. Instead, we applied a two-step estimation taking the root translation and foot vertices into account. The first step finds out which foot should be planted while the second one refines which part of the sole should remain static. This is achieved by first extracting from the skin mesh the vertices for which the speed has to be calculated. We then isolate the vertices belonging to the feet by using the skin attachment data. Finally, we remove the ones for which the normal is not pointing downward, which leaves us with the sole.

For clarity reasons, we will use t to designate a frame index or a time interval, the unit corresponding to the actual time elapsed between two animation frames.

3.1.1 Foot selection

Given the original root translation ΔR_t from time t to $t+1$ and v_i the vertex which is planted at frame t, we estimate which foot must remain planted at frame $t+1$ by considering the motion $\Delta R'_t$ for which the sole vertex v_j remains planted during the next animation frame. By planting a vertex at time t, we mean that its global coordinates remain constant during the time interval $[t-\frac{1}{2}, t+\frac{1}{2}]$. $\Delta R'_t$ can thus simply be expressed as:

$$\Delta R'_t = o(v_i, t) - o(v_i, t+\delta) + o(v_j, t+\delta) - o(v_j, t+1) \tag{1}$$

where $o(v_i, t)$ is the offset at frame t of vertex v_i from the root, in world coordinates. For this estimation, we take $\delta = \frac{1}{2}$, and $o(v_i, t+\delta)$ is calculated by linear interpolation between t and $t+1$. Once we calculate $\Delta R'_t$ for all the sole vertices, we designate as static the vertex that maximizes the dot product p:

$$p = \frac{\Delta R_t}{\|\Delta R_t\|} \cdot \frac{\Delta R'_t}{\|\Delta R'_t\|} .$$

Indeed, a higher value of this dot product means that if v_j is static at frame $t+1$, the displacement induced will resemble more the original root motion. We discard the magnitude of the vector because we are interested in *where* the character is going and not *how far away* it goes.

3.1.2 Vertex selection

The dot product criterion robustly tells us which foot must be planted. However, the actual vertex picked by this algorithm can sometimes be jerky, e.g., jump from the foot tip to the heel. The reason is that we picked the vertex which keeps the motion as close as possible to the original one, possibly keeping a bit of skating on its way. In order to overcome this issue, we add a second selection process applied on the vertices of the planted foot only. This second process uses the *speed* of the vertices in order to pick the right one. Indeed, if the original motion is not too bad, then the vertex which must be static at a given frame is most likely to be the one moving less. The speed of each vertex is first smoothed along several frames in order to remove some of the data noise (in our experiments, 5 frames appeared to be a good compromise). Second, the least moving vertex is chosen as the static one. The result of this selection over a foot step can be seen in Fig. 1.

We previously assumed that the static vertex in the previous frame must be known in order to estimate the one in the current frame. So for the first frame of the animation, we just use the speed criterion. One could think

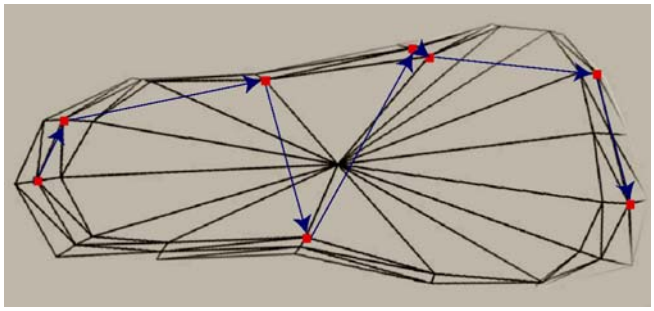

Fig. 1. View of the trajectory of the least moving point over the sole during one foot step. In *black* is a wire frame view of the sole of the character, in *red* are the vertices selected during the step, and eventually the *blue arrows* show the transitions between each point

that the detection would be less accurate because of this, however we did not witness any setback during our experiments.

This algorithm has proven to be quite efficient on the catwalk animations we tried it on. It is even possible to discard the dot product phase of the algorithm but we noticed that this stage of the process significantly improved the robustness of the detection by accurately tagging which foot must be planted. If the original animation clip is too bad, our algorithm may fail to figure out which vertex should be planted. In this case, one still has the possibility to manually label the vertices (or correct the output of the algorithm), as is the case for all the previous motion retargeting methods. However, during our tests, this only happened on complex dance motions, for which it was hard even for the human eye to figure out which foot should be planted or not.

3.2 Root translation correction

As outlined in Sect. 1, the retargeting is split into two phases, namely horizontal and vertical corrections. The horizontal correction introduces a drift of the character over the animation range in order to remove the footskating, while the vertical processing aims at minimizing the distance of the static vertices from the floor. These two separate steps use completely different approaches, which are outlined in the next section.

3.2.1 Horizontal correction

In order to calculate the corrected horizontal translation of the root joint between two frames, once again we use the motion of the vertices. In the previous section, we made the assumption that a static vertex remains during a time interval of at least one frame, centered around the current time instant. However, due to the low sampling of the motion data which is often no more than 25 Hz, this assumption cannot be retained for the actual displacement of the root joint. Thus, we estimate when the transition between two static vertices should happen, again using their speed. As before, the vertex with less speed should remain static, and we estimate the exact time instant between two frames when the transfer should occur.

For doing so, we approximate the speed of each vertex as follows: first the speed of the current and next static vertices v_i and v_j are calculated for frames $t-1$, t, $t+1$ and $t+2$. These velocities are then plotted in 2D and approximated using a Catmull–Rom spline [4], which yields to two parametric curves $V_i(q)$ and $V_j(q)$, $q \in [0, 1]$, as depicted in Fig. 2. Eventually, the particular value q_t corresponding to the cross between v_i and v_j is calculated by solving the cubic equation $V_i(q) = V_j(q)$, which we did using the approach proposed by Nickalls [18].

Now that the exact time $t + q_t$ when the weight transfer occurs is know, the actual position of the vertices at this instant is to be calculated. For doing so, the trajectory of the points between t and $t+1$ is first approximated using again a Catmull–Rom spline. The parametric location t_1 and t_2 of the points over these curves is given by their approximated speeds as follows:

$$t_i = \frac{\int_t^{t+q_t} V_i(q)\mathrm{d}q}{\int_t^{t+1} V_i(q)\mathrm{d}q}, i = 1, 2\,.$$

Having the two offsets $o(v_i, t+q_t)$ and $o(v_j, t+q_t)$, enables us to calculate the new root displacement between frames t and $t+1$ using formula Eq. 1, with $\delta = q_t$.

The translation computed during this step is valid only if the feet deform in a realistic way which – to our experience – they seldom do. Often they remain rigid and this creates a bad vertical translation while the weight is transferred from the heel to the toe during a foot step. This is the reason why, as stated previously, the calculated root translation is only applied on the horizontal directions, as follows:

$$\Delta R_t^{\text{horizontal}} = P \Delta R'_t$$

where P is a 3D to 2D projection matrix.

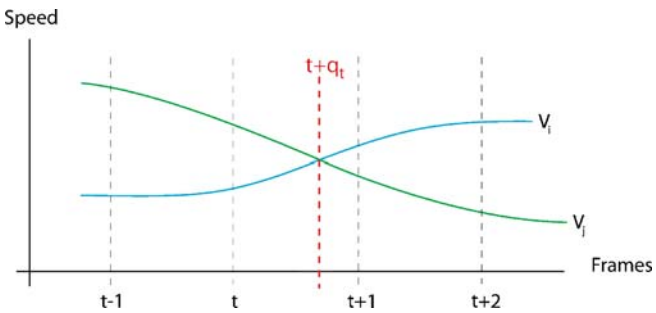

Fig. 2. A conceptual view of the velocity estimation performed in order to determine the exact instant of the weight transfer between two fixed points

3.3 Vertical correction

The horizontal correction introduces some drift of the character compared to the original animation. This effect is desired as it removes the footskating. However, in the vertical direction, no drift should take place otherwise the body will soon be walking in the floor or in the air. We do not want to change the legs configuration for enforcing the correct height of the foot sole because we want to remain as close as possible to the original animation of the limbs. Moreover, strictly enforcing the height of the static vertices to be zero would lead to cumbersome configurations of the legs in order to cope with the rigidity of the feet (remember that the feet seldom deform in VR applications) thus introducing unattractive artifacts.

Instead we chose to act on the root joint translation only, by minimizing the height of the static vertices over the animation. Thus, a small amount of penetration will remain afterwards, which is the price we pay if the feet are rigid and if we do not want to drastically change the look of the animation.

We calculate a single offset and a scale to be applied to the root height trajectory so that static vertices remain as close as possible to the ground throughout the animation, as shown in Fig. 3.

It is quite trivial to calculate the offset to be applied to the root trajectory: if we consider the height h_t in world coordinates of each static point, then the root offset ΔH is simply:

$$\Delta H = -\sum_{t=0}^{N-1} \frac{h_t}{N}$$

where N is the number of frames of the animation.

Once this offset is applied to the root trajectory, the mean of the static vertices height is thus zero. However, they still oscillate above and underneath the ground during the animation. This oscillation will be minimized by the calculation of the scaling factor α.

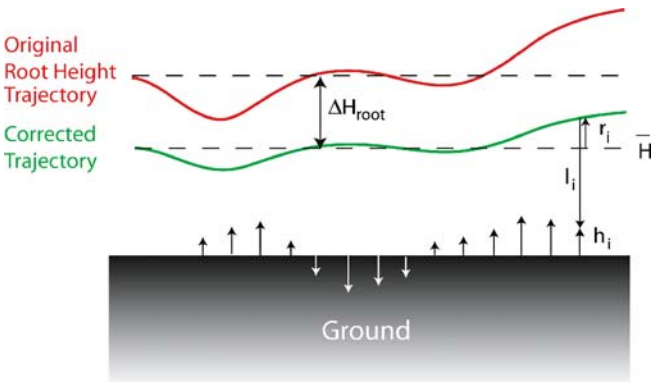

Fig. 3. Scaling of the root joint height

If we consider $\overline{H}$ to be the average root height over the animation, then for each frame its actual height H_t can be written as an offset r_t from this mean value: $H_t = \overline{H} + r_t$. The variance σ^2 of the static points height can be expressed in terms of the root average height $\overline{H}$, the scaling factor α and the relative height l_t of the fixed vertex with respect to the root as follows:

$$N\sigma^2 = \sum_{t=0}^{N-1} h_t^2 = \sum_{t=0}^{N-1} (\overline{H} + \alpha r_t + l_t)^2 .$$

This variance is to be minimized by the scaling factor α, and fortunately this is equivalent to finding the root of a simple second order equation with only one unknown, α. Indeed:

$$N\sigma^2 = \alpha^2 \sum_{t=0}^{N-1} r_t^2 + 2\alpha \sum_{t=0}^{N-1} (r_t.(\overline{H} + l_t)) + \sum_{t=0}^{N-1} (\overline{H} + l_t)^2 .$$

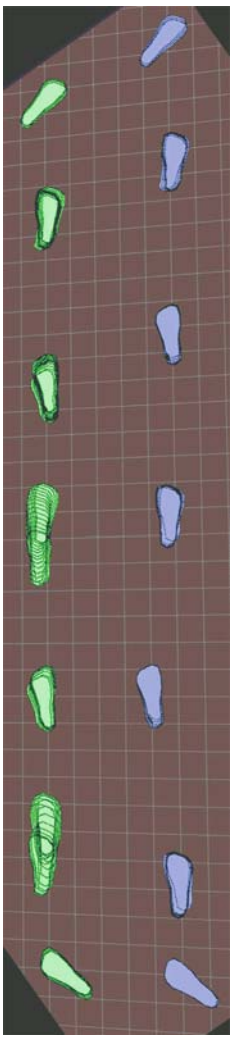

Fig. 4. Two foot prints left by a character animation: on the *left* (in *green*) the original clip and on the *right* (in *blue*) the retargeted clip. The residual sliding that one may notice in the *blue prints* is due to the estimation of the root motion between two steps, which makes both feet move at the same time in order to get a more realistic final motion

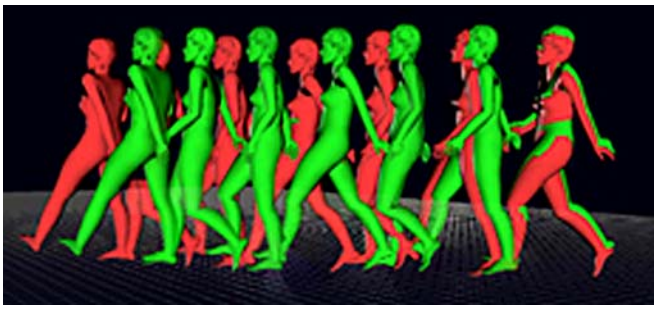

Fig. 5. Superimposed snapshots of a walk with the original animation in *red*, and the corrected one in *green*

As σ^2 and N are always positive, the minimal variance is given by:

$$\alpha = -\frac{\sum_{t=0}^{N-1} r_t.(\overline{H} + l_t)}{\sum_{t=0}^{N-1} r_t^2} .$$

4 Results

Figure 4 exhibits the foot prints left by a retargeted walk (in blue) and by the original walk (in green). One can see that nearly all footskate is corrected. The minor sliding which remains is due to the estimation of the displacement of the root joint between two steps, as described in Sect. 3.2. Indeed, because the weight transfer does not occur exactly in an animation frame, the feet may well slide a bit between two frames when switching from one foot plant to the other. Figure 5 shows animation snapshots of an original animation and its retargeted version. One can see that the drifting effect for getting rid of the foot sliding actually occurs, while the height of the character is modified so that its feet stick to the floor as much as possible.

Due to its simplicity, our method performs very swiftly and can be added to existing systems within moments. Thus even if it cannot accommodate all the motions one may want to use within a VR framework, it can fix many existing clips with only a minimal amount of time and effort.

5 Conclusion

In this paper, we presented a method to remove the footskating of a motion clip, which takes into account the character itself in order to improve the results and speed up the computations. Compared to previous approaches, this method fully relies on the skin, which is what the final animation will display, to perform the retargeting. It can be utilized by casual users as no interaction or expertise is required to obtain good motions. The footskating removal preserves the original movements of the limbs because the corrections are applied to the root joint translation only. Moreover, for most cases, it has the advantage of extracting the foot plants automatically, which prevents time consuming manual work.

Last but not least, in the future we would like to extend our method to be able to apply it to any kind of motion, and more particularly motions such as sliding and jumping. Indeed, during the foot plant extraction, we currently assume that the character always has one foot anchored to the ground, which is not the case in such motions.

Acknowledgement We would like to thank Clementine Lo for proof reading this article, Nedjma Cadi and Marlene Arevalo for the demo movie and Autodesk for their Maya donation. This work was funded by the European Project LEAPFROG IP (FP6-NMP-515810).

References

1. Abe, Y., Liu, K., Popovic, Z.: Momentum-based parameterization of dynamic character motion. In: Proceedings of the ACM Symposium on Computer Animation 2004. ACM, New York (2004)
2. Bindiganavale, R., Badler, N.I.: Motion abstraction and mapping with spatial constraints. Lect. Notes Comput. Sci. **1537**, 70–82 (1998) (URL citeseer.ist.psu.edu/bindiganavale98motion.html)
3. Boulic, R., Le Callennec, B., Herren, M., Bay, H.: Experimenting prioritized IK for motion editing. In: Proceedings of EUROGRAPHICS 2003. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland (2003)
4. Catmull, E., Rom, R.: A class of local interpolating splines. In: Computer Aided Geometric Design, pp. 317–326. Academic, New York (1974)
5. Choi, K., Ko, H.: Online motion retargeting. J. Vis. Comput. Anim. **11**(5), 223–235 (2000)
6. Egges, A., Molet, T., Magnenat-Thalmann, N.: Personalised real-time idle motion synthesis. In: PG '04: Proceedings of the Computer Graphics and Applications, 12th Pacific Conference on (PG'04), pp. 121–130. IEEE Press, Washington, DC (2004)
7. Glardon, P., Boulic, R., Thalmann, D.: Robust on-line adaptive footplant detection and enforcement. Vis. Comput. **22**(3), 194–209 (2006)
8. Gleicher, M.: Motion editing with spacetime constraints. In: Proceedings of the Symposium on Interactive 3D Graphics. ACM, New York (1997)
9. Gleicher, M.: Retargeting motion to new characters. In: Proceedings of SIGGRAPH 1998, Computer Graphics Proceedings, Annual Conference Series, pp. 33–42. ACM/ACM SIGGRAPH, New York (1998)
10. Humanoid Animation Working Group: H-anim. http://hanim.org. Cited May 2006 (2006)
11. Ikemoto, L., Arikan, O., Forsyth, D.: Knowing when to put your foot down. In: SI3D '06: Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games, pp. 49–53. ACM, New York (2006) (DOI http://doi.acm.org/10.1145/1111411.1111420)
12. Kovar, L., Gleicher, M.: Flexible automatic motion blending with registration curves. In: SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 214–224. Eurographics Association,

Aire-la-Ville, Switzerland, Switzerland (2003)
13. Kovar, L., Gleicher, M., Pighin, F.: Motion graphs. In: SIGGRAPH '02: Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques, pp. 473–482. ACM, New York (2002) (DOI http://doi.acm.org/10.1145/566570.566605)
14. Kovar, L., Schreiner, J., Gleicher, M.: Footskate cleanup for motion capture editing. In: Proceedings of the ACM Symposium on Computer Animation 2002. ACM, New York (2002)
15. Lam, W., Zou, F., Komura, T.: Motion editing with data glove. In: Proceedings of the SIGCHI International Conference on Advances in Computer Entertainment Technology, pp. 337–342. ACM, New York (2005)
16. Lee, J., Shin, S.: A hierarchical approach to interactive motion editing for human-like figures. In: Proceedings of SIGGRAPH 1999, Computer Graphics Proceedings, Annual Conference Series. ACM/ACM SIGGRAPH, New York (1999)
17. Motion Analysis: http://www.motionanalysis.com. Cited May 2006 (2006)
18. Nickalls, R.: A new approach to solving the cubic: Cardan's solution revealed. Math. Gaz. **77**, 354–359 (1993)
19. Ponder, M., Papagiannakis, G., Molet, T., Magnenat-Thalmann, N., Thalmann, D.: Vhd++ development framework: towards extendible, component based VR/AR simulation engine featuring advanced virtual character technologies. In: Proceedings of Computer Graphics International, pp. 96–104. CGI, IEEE Press, Washington, DC (2003)
20. Seo, H., Magnenat-Thalmann, N.: An example-based approach to human body manipulation. Graph. Models **66**(1), 1–23 (2004)
21. Shin, H., Lee, J., Shin, S., Gleicher, M.: Computer puppetry: an importance-based approach. ACM Trans. Graph. **20**(2), 67–94 (2001)
22. Tak, S., Ko, H.: A physically-based motion retargeting filter. ACM Trans. Graph. **24**(1), 98–117 (2005)
23. Tolani, D., Goswami, A., Badler, N.I.: Real-time inverse kinematics techniques for anthropomorphic limbs. Graph. Models **62**, 353–388 (2000)
24. Vicon: http://www.vicon.com. Cited May 2006 (2006)
25. VR Juggler: Vr juggler–open source virtual reality tools. http://www.vrjuggler.org. Cited May 2006 (2006)
26. Witkin, A., Kass, M.: Spacetime constraints. In: Proceedings of SIGGRAPH 1988, Computer Graphics Proceedings, Annual Conference Series, pp. 159–168. ACM/ACM SIGGRAPH, New York (1988)

ETIENNE LYARD received his Master's degree in Computer Graphics, Vision and Robotics from the University of Grenoble in 2002. The next year he worked as a reasearch scholar at the Multisensory Computation Lab (Rutgers University). Since then, he has been a research assistant and PhD candidate at MIRALab (University of Geneva). His research interests include physics based modeling, real time applications and human motion.

NADIA MAGNENAT-THALMANN has pioneered research into virtual humans over the last 25 years. She obtained her PhD in Quantum Physics from the University of Geneva. From 1977 to 1989, she was a professor at the University of Montreal where she founded the research lab MIRALab . She was elected Woman of the Year in the Grand Montreal for her pioneering work on virtual humans and presented Virtual Marilyn at the Modern Art Museum of New York in 1988. Since 1989, she has been a professor at the University of Geneva where she recreated the interdisciplinary MIRALab laboratory. With her 30 PhD students, she has authored and coauthored more than 300 research papers and books in the field of modeling virtual humans. She is presently coordinating three European Research Projects (INTERMEDIA, HAPTEX and 3DANATOMICAL HUMANS).