Motion Pattern Encapsulation for Data-Driven Constraint-Based Motion Editing

Schubert Carvalho, Ronan Boulic, and Daniel Thalmann

EPFL VRlab
Station 14, CH 1015 Lausanne, Switzerland
{schubert.carvalho,ronan.boulic,daniel.thalmann}@vrlab.ch
http://vrlab.epfl.ch

Abstract. The growth of motion capture systems have contributed to the proliferation of human motion database, mainly because human motion is important in many applications, ranging from games entertainment and films to sports and medicine. However, the captured motions normally attend specific needs. As an effort for adapting and reusing captured human motions in new tasks and environments and improving the animator's work, we present and discuss a new data-driven constraint-based animation system for interactive human motion editing. This method offers the compelling advantage that it provides faster deformations and more natural-looking motion results compared to goal-directed constraint-based methods found in the literature.

Key words: Motion Models, Motion Editing, Inverse Kinematics, PCA.

1 Introduction

Technological advances in motion capture (mocap) has allowed to provide high quality motions for computer animation. However, the captured animations almost always meet specific needs. Therefore, modifying and reusing these motions in new situations became an increasing area of research known as *motion editing*. In the past few years, there has been a lot of work in motion editing in the graphics community. The proliferation of mocap database has enabled the research and development of data-driven or model-based techniques [1–6]. Basically, these methods focus in constructing a model from mocap data - the data are usually represented by a low-dimensional space known as the *latent space* - to generate new motions from existing ones. Essentially because this representation can be exploited to analyze and synthesize the human motion within a low-dimensional space of physically balanced motions. Furthermore, model-based approaches are capable to generate more natural-looking motions compared to goal-directed ones, but the solutions are limited to the database. On the contrary, goal-directed approaches, for example, the ones based on pseudoinverse techniques that performs optimizations in the joint space - which includes the great majority of constraint-based methods - can achieve a wide range of user-defined tasks. However, their disadvantage is the frequent lack of naturalness when it comes to reproduce human activities.

As an effort of taking the advantages of model-based and goal-directed methods, we introduce a data-driven constraint-based motion editing technique, which combines the particularities of model-based and the versatility of goal-oriented approaches. Our method is based on the connection between linear motion models such as Principal Component Analysis (PCA), which is used to estimate a set of Principal Components (PC) and Principal Coefficients (PCs) [7], and Prioritized Inverse Kinematics (PIK), which is used to provide interactive motion editing. These links allowed us to construct a framework capable to solve a constraint-based optimization problem within the latent space. As a result, system performance is improved compared to pure goal-direct methods. Furthermore, to make easier the animator's work, we build a system to allow animators to generate natural-looking motions from key-frame constraints (i.e., the constraint is specified at specific poses) and key-trajectory constraints (i.e., the constraints are specified over a set of frames representing the trajectory of end-effectors). In order to enforce the spatio-temporal motion flow, continuity is imposed through the PC of the underlying motion pattern. By making use of the PC space, it becomes very efficient to enforce the fluidity of the motion compared to joint space-based methods. Accordingly, the editing becomes more intuitive.

To validate our framework, we have designed a number of experiments to adapt walking jump, reaching and golf swing motions to different situations and environment. The results are also compared against a constraint-based technique that performs deformations in the joint space [8]. After reviewing related work, we start describing the new approach for Data-Driven Constraint-Based Motion Editing.

2 Related Work

Note that, because of the vastness of the subject, this review is not intended to be exhaustive. Constraint-based techniques [9, 8, 10–12] provide a powerful framework to adapt recorded motions to different characters or to new situations from a set of user-specified constraints, such as end-effectors positions and end-effectors trajectories. To enforce a motion deformation an IK solver is normally used to solve user-specified constraints. To preserve the continuity between frames some IK solvers refer to the previous frame in order to choose a solution for the current one. When this is not the case other techniques such as filtering, B-Splines curves, collision avoidance and displacement maps are used to enforce the naturalness between frame. The main drawbacks of these continuity strategies are the time wasted traversing the character's substructures and computing convolution operations that degrade system performance. Data-driven approaches rely on mocap data to restrict the solution space of natural-looking motions. Motion graphs and motion interpolation are used to produce new motions from a database [13–15]. These techniques are not able to handle end-effector constraints that are not represented in the motion database. These techniques present limitations in handling constraints over multiple frames. A

number of researches have developed techniques to synthesize human motion in a low-dimensional space [1, 16, 6, 2, 3], both linear and non-linear models are used. Despite the fact that natural-looking motions can be produced, these techniques present some limitations regarding the database size, handling user-specified constraints or computation performance. A data-driven constraint-based system was proposed in [17], but the system was not capable to handle key-frame trajectory constraints. Recently, Recently, Raunhardt and Boulic [5], have been combined a data-driven and goal-oriented methods to construct a hybrid postural control approach overcoming their limitations. The main focus of the work is to treat the issue of controlling a full-body goal-directed motion (i.e. reach) where locomotion is not necessary. The model is learned from *pose* variations rather than *motion* data. The main difference between this work and ours is that, the optimization is preformed in the joint space rather than the latent space of the underlying motion pattern.

3 Motion Pattern Encapsulation

We refer to the process of learning the underlying motion pattern as *motion encapsulation*. We define a character pose, as a state vector describing the 3D global position, $\mathbf{P}_{root}$ and the 3D global orientation $\mathbf{Q}_{root}$ of the root node, and a set of joint orientations θ, represented by three parameters of the corresponding exponential map [18]:

$$\mathbf{\Theta} = \left\{ \mathbf{P}_{root}, \mathbf{Q}_{root}, \theta_n \right\}. \tag{1}$$

As we use motion capture (mocap) data from real people, and as each person tends to perform the same activity with some variability in speed, the database contains motions that, in general, have different durations. So, a duration normalization is necessary in our approach because the PCA's parameters are estimated from complete motions. The normalization is carried out by using quaternion spherical linear interpolation (Slerp) [19]. A motion is then represented as a line vector of the form:

$$\mathbf{\Psi}_i = \{\mathbf{\Theta}_1 \dots \mathbf{\Theta}_k \dots \mathbf{\Theta}_N\}. \tag{2}$$

$\mathbf{\Theta}_k$ is therefore a pose corresponding to a frame index k. Since a given motion consists of N poses, the motion vector has dimension: $D = (n \cdot N)$.

Once the mocap data are arranged in a matrix form, we use PCA to find a low-dimensional data representation, which efficiently acquires the important nuances of a specific motion pattern. By applying PCA, any normalized motion $\mathbf{\Psi}_i$ can be approximated as:

$$\mathbf{\Psi}_i \approxeq \alpha_i \mathbf{E}_{\mathbf{\Psi}} + \mathbf{\Psi}_\circ \tag{3}$$

where $\mathbf{\Psi}_\circ$ is the mean motion, $\mathbf{E}_{\mathbf{\Psi}}$ are the Principal Components (PC) also referred as the *eigen-motions* and α_i are the so called Principal Coefficients (PCs) that characterize the motion (i.e., the *latent space*). And a pose, here also

referred as an *eigen-pose*, can be computed as a function of the scalar coefficients, α_i $(i = 1, \ldots, m)$ and the frame index k:

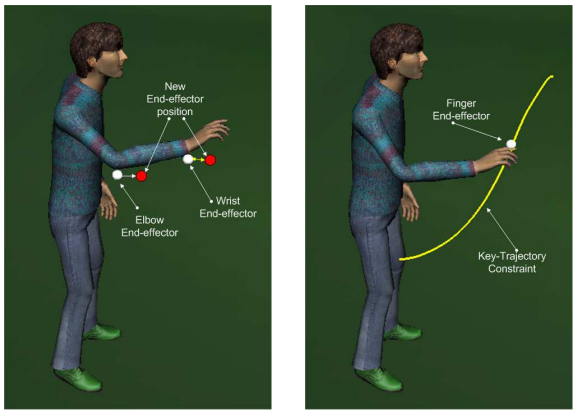
$$\psi\left(k, \alpha_1, \ldots, \alpha_m\right) \cong \alpha_i \mathbf{E}_{\mathbf{\Psi}} + \mathbf{\Psi}_{\circ}. \tag{4}$$

m represents the number of Principal Components that are required to reconstruct a desired percentage of the database [17]. Note that, the PC and the PCs are estimated off-line because the PC remain constant. We learn PCA motion models from three motion patters: walking jumps, golf swings and reaching motions. In the next section, we show how to explore PCA by building a framework to perform optimizations directly in the latent space.

4 Data-Driven Constraint-Based Optimization

4.1 Constraints

Providing interactive ways to manage constraints allow users to easily and rapidly modify preexisting human animations. In particular, geometric constraints, such as the position of an end-effector in the three-dimensional space [20] or the trajectory of an end-effector [8], are more intuitive for interactive manipulation because the user can specify a goal just by dragging an end-effector to a new position. We equip animators with two types of constraints: key-frame and key-trajectory constraints as illustrated in Figure 1, both are firstly created in the joint space and then mapped into the latent space.



(a) (b)

Fig. 1. (a) key-frame constraints. (b) Key-trajectory constraints.

Essentially, key-trajectory constraints are modeled as a *Kochanek-Bartels* spline [21]. This type of constraints allow the animator to edit a motion by specifying end-effectors across the whole motion as continuous trajectories. The aim

of this formulation is to reduce the work of the animator of manually specifying complete end-effector trajectories. On the other hand, with key-frame constraints end-effectors are attached directly on the character's body and dragged to knew positions. Key-frame constraints are pure positional constraints, that is, the effector does not follow any specific trajectory. A key-frame constraint is simply represented by a three-dimensional point x expressed in the so-called *task space*. The animator can use this type of constraints to edit the whole motion by constraining one key-frame on the motion.

4.2 Low-Dimensional Inverse Kinematics (LIK)

In this section, we demonstrate how standard IK techniques can be adapted to our needs. If we consider the pose of a virtual character as a n-dimensional vector (joint space, Eq. 1) and the position of the end-effector, x, as a p-dimensional vector (task space), the IK function can be defined as:

$$\mathbf{\Theta} = f^{-1}(x). \tag{5}$$

To solve this equation by using the well-known *resolved motion rate control* RMRC [22] technique, we need to compute the Jacobian matrix of the forward kinematics function, $x = f(\mathbf{\Theta})$. For a redundant manipulator the damped pseudo-inverse technique can be used [23]:

$$\Delta\mathbf{\Theta} = J(\mathbf{\Theta})^{\dagger^{\xi}}\Delta\mathbf{x} \tag{6}$$

where, $J(\mathbf{\Theta})^{\dagger^{\xi}}$ is the damped pseudo-inverse of the $p \times n$-dimensional joint space Jacobian matrix $J(\mathbf{\Theta}) = \partial x/\partial\mathbf{\Theta}$. The damped factor, ξ, is added to prevent Jacobian's singularities and to stabilize IK solutions. For a redundant manipulator, i.e., $n > p$, the problem is ill-posed (i.e., there is no unique solution). Hence, IK algorithms should determine one solution to Eq. 6 given many possibilities.

As an effort to restrict solutions within the space of feasible motions (i.e., the latent space), we integrate the human behavior directly in the optimization process to obtain more realistic natural-looking solutions. In practice, to compute the pose increment in the latent space we need to solve following equation:

$$\Delta\alpha = J(\alpha)^{\dagger^{\xi}}\Delta x \tag{7}$$

where, $J(\alpha)^{\dagger^{\xi}}$ is the pseudo-inverse of the PCs Jacobian $J(\alpha) = \partial x/\partial\alpha$, which relates a variation Δx in the Cartesian space with a variation $\Delta\alpha$ in the latent space. The computation of $J(\alpha)$ can be easily carried out with the chain rule [24]. So, once we have the new increment $\Delta\alpha$ the new pose can be computed in two steps. First, $\Delta\alpha$ is added with respect to α to obtain the final principal coefficients,α^{υ}:

$$\alpha^{\upsilon} = \alpha + \Delta\alpha. \tag{8}$$

Finally, by using Eq. 4, we compute the new state vector as follows:

$$\psi\left(k, \alpha^{\upsilon}\right) \approx \alpha^{\upsilon}\mathbf{E}_{\mathbf{\Psi}} + \mathbf{\Psi}_{\circ}. \tag{9}$$

In the next section, we show how to extend the current approach to deal with multiple tasks and to resolve their possible conflicts in the latent space.

4.3 Low-Dimensional Prioritized Inverse Kinematics (LPIK)

Let us define a set of p tasks, each one satisfying its goal $\mathbf{g}_j$: $x(\alpha)_j = \mathbf{g}_j$, $j \in [1, \ldots, p]$, and having its corresponding increment, $\Delta x = (\Delta x_1, \ldots, \Delta x_p)$.

In the case of multiple tasks occurring at the same key time k, the optimal solution α^* should satisfy all of them, i.e., $x(\alpha^*)_j = \mathbf{g}_j$, $\forall_j$. However, this may be difficult because some of the tasks may be in conflict. Tasks are said to be in conflict when they are not achievable simultaneously.

To solve a conflict task, we use a technique based on a priority strategy. The major strength of this technique is that prioritized constraints are sorted into priority-layers. As a result, constraints belonging to the highest priority layer are enforced first (e.g., feet on the ground). Then, those of the next priority-layer are satisfied as much as possible without disturbing the previous constraints, and so on. Based on a previous work done by [22, 20], we reformulate the prioritized strategy to work in the latent space as follows [17]:

$$\Delta\alpha = J(\alpha)^{\dagger^{\xi}} \Delta x + \mathrm{P}_{N(J(\alpha))}\mathrm{z} \tag{10}$$

$$\mathrm{P}_{N(J(\alpha))} = \mathrm{I}_m - J^{\dagger}(\alpha)J(\alpha).$$

where $J(\alpha)^{\dagger}$ is the $m \times p$ pseudo-inverse of J_α, $\mathrm{P}_{N(J(\alpha))}$ is the $m \times m$ projector operator onto the null-space of J_α, I_m is the $m \times m$ identity matrix and z is the m-dimensional PCs variation vector. The algorithm that solves Eq. 10 is presented in [17]. During the editing the animator has to setup the optimizer's parameters, that is, the number of iterations and the damping factor.

5 Imposing Continuity

We introduce two techniques for imposing motion continuity based on PCA. First, the strategy elaborated for imposing continuity from key-trajectory constraints uses the eigen-motion space. As in standard per-frame constraint-based motion editing techniques [9, 8], the user needs to relax the motion by using ease-in/ease-out time intervals to prevent discontinuities. Note that, the $\{\alpha\}$ vector describes a complete motion belonging to the latent space. Therefore, solving the problem in the latent space for many poses for determining a unique $\{\alpha^{\upsilon}\}$ satisfying all the user-specified constraints across the motion, can easily over-constraints the problem due to the small size of the latent space. In practice, we ease the constraints by using a frame by frame approach, such that, from to beginning to the ending of the motion sequence the LPIK optimizes the initial $\{\alpha\}$ for each constrained frame obtaining a new set of principal coefficient vectors, such as: $\{\alpha_1^{\upsilon}, ..., \alpha_k^{\upsilon}, ..., \alpha_N^{\upsilon}\}$. Therefore, each pose is computed using Eq. 4. Figure 2 illustrates the process. Second, in our framework, impose continuity

from key-frame constraints is a straightforward process using the Eq. 3. As the principal coefficients are learned from complete motions, each vector $\{\alpha\}$ characterizes a complete animation. Therefore, if the parameter vector is optimized for a specific key, k, resulting from a constraint imposed on a pose $\mathbf{\Theta}_k$, the updated set $\{\alpha^{\upsilon}\}$ defines one full motion belonging to the latent space. Once the solution coefficients are computed, they are also used to define the other poses on the motion by using Eq. 3. Figure 2(b) illustrates the process.

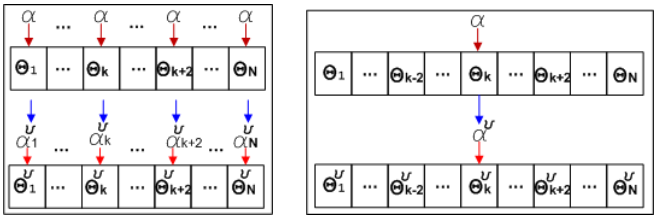


Fig. 2. (a) Once the LPIK converges to a $\{\alpha_k^{\upsilon}\}$ satisfying all the constraints of a pose k, the system uses Eq. 4 to recover the deformed pose: $\mathbf{\Theta}_k^{\upsilon} = \alpha_k^{\upsilon}\mathbf{E}_{\mathbf{\Psi}} + \mathbf{\Psi}_{\circ}$. The process is repeated frame by frame. (b) Motion editing from specific key-time constraints. The complete animation $\{\mathbf{\Psi}^{\upsilon}\}$ is computed as: $\mathbf{\Psi}^{\upsilon} = \alpha^{\upsilon}\mathbf{E}_{\mathbf{\Psi}} + \mathbf{\Psi}_{\circ}$.

6 Overview of the System

6.1 Implementation

The motion editing system that we use as a basis for all experiments has been implemented in three languages: C++, C and Matlab. The system is subdivided in off-line and on-line computations. We used Matlab to construct the PCA motion model. We chose Matlab because it allows easy matrix manipulation, it is stable and it is well adopted in the scientific community. The PCA parameters are stored as text files, which are loaded in the system's data structure when it is started. The computationally expensive calculations, such as the optimization and the motion generation strategy, and the PCA synthesis are all done in C and C++. In all the experiments reported in this chapter are run on a 3.2GHz Pentium Xeon(TM) with 1GB memory.

6.2 System Interface and Functionalities

The data-driven constraint-based motion editing system proposed in this work was integrated into the Autodesk/Maya software as plug-in and MEL scripts. The character model can be loaded from description files, and interactively edited in the application. The deformed animation can be saved for future use, either

as a full-body motion or as a set of latent variables. The window devoted to the management of end-effectors and other objects in the scene is shown in left side of Figure 3. End-effector can be created by selecting among a list of predefined joints available in the window (left side of Figure 3). Another alternative and more direct solution is to pick a point on the surface of a character body with the mouse pointer, in the application window. This allows to control any visible part of the character body just by clicking on it.

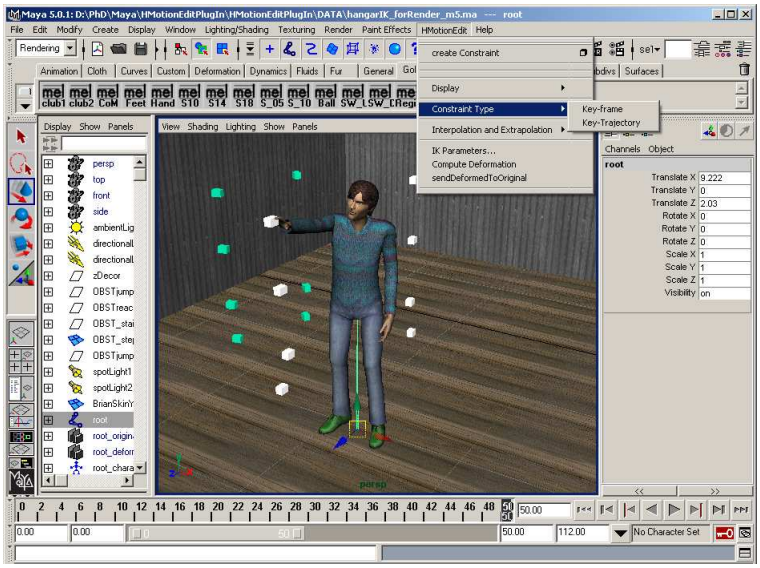


Fig. 3. Data-driven constraint-based motion editing system interface. The green and white cubes represent the reaching positing in the database.

7 Experiments

In this section, we have designed three case studies to evaluate the usability and the performance of the proposed data-driven constraint-based motion editing system. To validate our framework, we perform experiments to compare our approach against a prioritized constraint-based technique that performs deformations in the joint space [8], regarding performance, robustness, simplicity, continuity and realism, by synthesizing golf swing, walking jump and reaching motions. For a fair comparison between both techniques, in the joint constraint-based system, we set the parameters of the optimizer as suggested by the authors. We made the same with our system.

In the first case study, we have retarget a golf swing, with 132 frames, executed on a flat ground to a up slope ground with 7° anti-clockwise, by adjusting

(a) (b)

Fig. 4. The initial hit position is illustrated by a transparent ball. In both methods the golfer hits the ball (a,b), but with the PIK the club head hit the ground (b).

(a) PIK (b) LPIK

Fig. 5. Pink arrow shows the current foot position and the red shows the sliding gap.

the feet position and by shifting the hit position of the golf club head, Figure 4. We used a motion model learned from 16 normalized golf swings played on up slopes, for angles ranging from 0.5° to 5.0° (anti-clockwise), by increments of 0.5°, each motion has 132 frames. We used a database percentage of 98%, which gave $m = 15$ dimensions, and we set the optimizer's parameters to: 100 number of iterations and $\xi = 0.8$. The adjusted posture, at frame 94, and the position of the golf club with respect to the hands were the same in both cases. A key-frame constraint were used for the data-driven approach and key-trajectory constraints were used instead for the the joint approach, as it cannot handle key-frame constraints. It is important to mention that, the 7° slope was not learned by the model. The motion editing system based on the joint optimization generated a less realistic swing motion compared to one based on the latent space. We observed that, the character's left hand lost the contact with the golf club and the club head hit the ground. On the contrary, with the proposed approach, only one key-frame constraint was necessary - on the hit pose - to achieve a globally

continuous motion. The computing performance needed to generate the up slope motion for the joint and latent approaches were 102 and 3 seconds, respectively.

In the second case study, we considered a simple task: the character has to reach an object. We then attached one constraint on the character's hand, at frame 50, to drive the reach in the direction of the goal. The constraint was activated over all frames because the character had to follow the path from start to end. This constraint configuration was used in both systems. We used a motion model learned from 16 normalized reaching motions, such that each motion has 50 frames. We considered a database percentage of 98%, which gave $m = 15$ dimensions, and we set the optimizer's parameters in both systems to: 100 number of iterations and $\xi = 10$. We observed in Figure 5 that the PIK-based system generated a motion in which the character slides to reach the green bar. On the other hand, the latent space generated an animation without introducing motion artifacts foot slides.

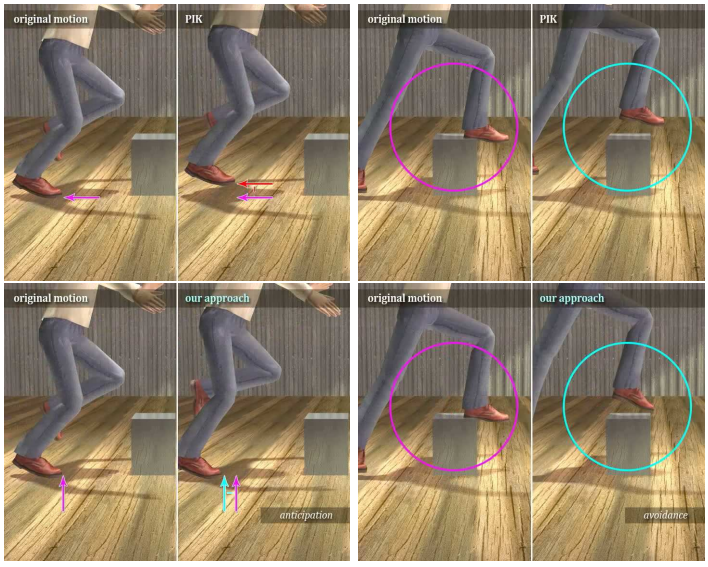


Fig. 6. Fist row shows the input and the generated motion by the PIK-based system. Second row the same for the LPIK-based system. The pink arrow shows the current foot position, the red the sliding gap and the blue the anticipation.

In the last case study, we took a walking jump sequence of $0.8m$ and we modify it to produce a higher jump. By attaching one constraint on the character root node, at frame 15, which is the time of the apex of the jump, and moving the constraint to a higher location. We used key-trajectory constraint for the joint approach and key-frame constraints for the latent one. We have used a motion model learned from 89 normalized motions of five men and one woman performing walking jumps of 3 lengths ranging from $0.4m$ to $1.2m$, by increments

of $0.4m$, such that each motions has 26 poses. We used a database percentage of 95%, which gave $m = 30$ dimensions, and we set the optimizer's parameters in both systems to: 100 number of iterations and $\xi = 10$. The end-effector's final goal was the same in both systems. Figure 6 shows the generated motions from key-trajectory and key-frame constraints. The PIK-based system produced a higher jump motion, but by moving the constraint to a higher position made the character lose ground contact, which is not desired because its not natural. On the other hand, LPIK-based system not only generated a higher jump, but also the resulted animation kept ground contact.

8 Conclusions

In this work, we have proposed a new approach for data-driven constraint-based motion editing. Our technique is based on the link between linear motion models such as Principal Component Analysis (PCA) and Prioritized Inverse Kinematics (PIK). The connection of both techniques allowed us to construct a Low-dimensional Prioritized Inverse Kinematics (LPIK) framework. The solver handles deformation problems within the latent space of some underlying motion pattern. By making use of the pattern space, we increased the possibilities of performing IK in the space of feasible poses.

We have demonstrated that our method achieves a degree of generality beyond the motion capture data. For example, we have retarget a golf swing motion to a 7° up slope using constraints that cannot be satisfied directly by any motion in the database, and have found that the quality of the generated motion was believable. We have seen that our approach is well-suited to deal with deformations and retargeting problems. We have demonstrated the robustness of our approach, deforming three types of movements and comparing the results against a goal-directed constraint-based technique that perform optimizations in the joint space. The system behaves well and robustly, as long as the model has sufficient information to handle the user-specified constraints.

Acknowledgments. The authors would like to thank to Autodesk/Maya for their donation of Maya software. Many thanks to Mireille Clavien for video production. This project was sponsored by the EPFL - Sport and Rehabilitation Engineering program.

References

1. A. Safonova, J. K. Hodgins, and N. S. Pollard. Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. *ACM Trans. Graph.*, 23(3):514–521, 2004.
2. P. Glardon, R. Boulic, and D. Thalmann. Robust on-line adaptive footplant detection and enforcement for locomotion. *Vis. Comput.*, 22(3):194–209, 2006.
3. J. Chai and J.K. Hodgins. Constraint-based motion optimization using a statistical dynamic model. *ACM Trans. Graph.*, 26(3):8, 2007.

4. R. Urtasun, P. Glardon, R. Boulic, D. Thalmann, and P. Fua. Style-based motion synthesis. *Computer Graphics Forum (CGF)*, 23(4):799–812, November 2004.
5. D. Raunhardt and R. Boulic. Motion constraint. *Vis. Comput.*, 25(5-7):509–518, 2009.
6. H.J. Shin and J. Lee. Motion synthesis and editing in low-dimensional spaces. *Comput. Animat. Virtual Worlds*, 17(3‐4):219–227, 2006.
7. I. T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, 1986.
8. B. Le Callennec and R. Boulic. Interactive motion deformation with prioritized constraints. *Graphical Models*, 68(2):175–193, March 2006. Special Issue on SCA 2004.
9. M. Gleicher. Comparing constraint-based motion editing methods. *Graphical models*, 63(2):107–134, 2001.
10. R. Kulpa, F. Multon, and B. Arnaldi. Morphology-independent representation of motions for interactive human-like animation. In *EUROGRAPHICS*, volume 24, pages 343–352, August-September 2005.
11. J. Lee and S.Y. Shin. A hierarchical approach to interactive motion editing for human-like figures. In *Proceedings of ACM SIGGRAPH*, 1999.
12. L. Liu, W. Zhao-qi, Z. Deng-Ming, and X. Shi-Hong. Motion edit with collision avoidance. In *Proceedings of the WSCG*, pages 303–310, January 2006.
13. O. Arikan and D.A. Forsyth. Interactive motion generation from examples. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 483–490, New York, NY, USA, 2002. ACM.
14. L. Kovar, M. Gleicher, and F. Pighin. Motion graphs. *ACM Trans. Graph.*, 21(3):473–482, 2002.
15. T. Mukai and S. Kuriyama. Geostatistical motion interpolation. *ACM Trans. Graph.*, 24(3):1062–1070, 2005.
16. K. Grochow, S. L. Martin, A. Hertzmann, and Z. Popovi. Style-based inverse kinematics. *ACM Trans. Graph.*, 23(3):522–531, 2004.
17. S.R. Carvalho, R. Boulic, and D. Thalmann. Interactive low-dimensional human motion synthesis by combining motion models and pik. *Computer Animation & Virtual Worlds*, 18, 2007. Special Issue of Computer Animation and Social Agents (CASA2007).
18. F. S. Grassia. Practical parameterization of rotations using the exponential map. *Journal of Graphics Tools*, 3(3):29–48, 1998.
19. K. Shoemake. Animating rotation with quaternion curves. In *SIGGRAPH '85*, pages 245–254, New York, NY, USA, 1985. ACM Press.
20. P. Baerlocher. *Inverse Kinematics Techniques of The Interactive Posture Control of Articulated Figures*. Phd thesis, cole Polytechnique Fdral de Lausanne (EPFL) - IC School of Computer and Communication Sciences, 2001.
21. D. H. U. Kochanek and R. H. Bartels. Interpolating splines with local tension, continuity, and bias control. *SIGGRAPH Comput. Graph.*, 18(3):33–41, 1984.
22. D. E. Whitney. Resolved motion rate control of manipulators and human prostheses. In *IEEE Trans. Man-Mach. Syst*, volume 10, pages 47–53, June 1969.
23. A.A. Maciejewski. Dealing with the ill-conditioned equations of motion forarticulated figures. In *Computer Graphics and Applications, IEEE*, volume 10, pages 63–71, May 1990.
24. S.R. Carvalho, R. Boulic, and D. Thalmann. Motion pattern preserving ik operating in the motion principal coefficients space. In *Proceedings of 15-th WSCG*, pages 97–104, January-February 2007.