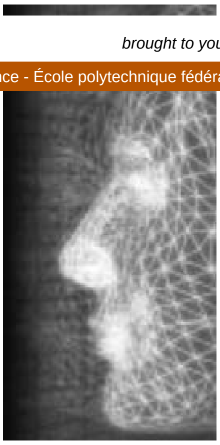
COMPUTER ANIMATION AND VIRTUAL WORLDS
Comp. Anim. Virtual Worlds (in press)
Published online in Wiley InterScience
(www.interscience.wiley.com) DOI: 10.1002/cav.210



motion synthesis by combining motion models and PIK

By Schubert R. Carvalho*, Ronan Boulic and Daniel Thalmann

This paper explores the issue of interactive low-dimensional human motion synthesis. We compare the performances of two motion models, i.e. Principal Components Analysis (PCA) or Probabilistic PCA (PPCA), for solving a constrained optimization problem within a low-dimensional latent space. We use PCA or PPCA as a first step of preprocessing to reduce the dimensionality of the database to make it tractable, and to encapsulate only the essential aspects of a specific motion pattern. Interactive user control is provided by formulating a low-dimensional optimization framework that uses a Prioritized Inverse Kinematics (PIK) strategy. The key insight of PIK is that the user can adjust a motion by adding constraints with different priorities. We demonstrate the robustness of our approach by synthesizing various styles of golf swing. This movement is challenging in the sense that it is highly coordinated and requires a great precision while moving with high speeds. Hence, any artifact is clearly noticeable in the solution movement. We simultaneously show results comparing local and global motion models regarding synthesis realism and performance. Finally, the quality of the synthesized animations is assessed by comparing our results against a per-frame PIK technique. Copyright © 2007 John Wiley & Sons, Ltd.

Received: 15 May 2007; Accepted: 23 May 2007

KEY WORDS: human motion synthesis; motion models; prioritized inverse kinematics

Introduction

Nowadays, it is common to leverage on motion captured databases to improve interactive human motion synthesis by constructing a low-dimensional latent space from the data.[1–5] Constrained optimization can be performed within the latent space to obtain movements enforcing a new set of constraints. However, reducing the dimensionality of the motion data space is not sufficient to guarantee the quality of the solution. This is in part due to the style variability characterizing the motion database. Moreover, to produce acceptable synthesized results challenging issues might also be treated, such as: handling the size of the database to achieve faster synthesis results,[2] dealing with artifacts added on the solution animation,[1,3,5] and allowing the user to interactively synthesize an animation by adding constraints.[1,2,4]

In this paper, we propose a solution for the problem of interactive low-dimensional human motion synthesis, by using a framework that combines motion models and Prioritized Inverse Kinematics (PIK). We compare two motion models, one based on Principal Components Analysis (PCA) and the other on Probabilistic PCA (PPCA), for solving a low-dimensional constrained optimization. These models are used to reduce the dimensionality of the database to make it tractable, and to enclose the key aspects of a specific motion pattern. PIK allows the user to add constraints with different levels of priorities while interactively editing an animation. To produce realistic results and to improve synthesis performance, we exploit the restricted space provided by local models (i.e., models learned from the same motion style).

We demonstrate the performance and robustness of our approach on the specific case of golf by synthesizing various styles of golf swing—such as the ones shown in Figure 7. This movement is challenging because it is highly synergistic and requires a great precision

*Correspondence to: S. R. Carvalho, VRLab—Ecole Polytechnique Fédérale de Lausanne, EPFL IC ISIM VRLAB Station 14, 1015 Lausanne, Switzerland. E-mail: schubert.carvalho@epfl.ch

Figure 1. With our approach the user needs to adjust just one key frame with PIK to realistically synthesize the motion as a whole.

while moving with high speeds.[6–9] We show that by using our system the user can precisely synthesize a motion just by adjusting one key frame (e.g., the hitting position in the case of golf, Figure 1) and the system automatically synthesize the movement as a whole. Our experiments show that, when local models are used, the synthesis process runs faster and more realistic animations are produced. We observed that the PCA motion model produced better quality results in extrapolation situations compared to PPCA. Finally, the quality of the synthesized animations is assessed by comparing our results against a per-frame PIK motion editing technique.[10]

Related Work

Motion editing is a well known approach used to produce new motions from existing ones. Constrained-based techniques enable the user to specify constraints over the entire motion or at specific times while editing an animation. Most approaches[10–14] solve the synthesis problem on the full-postural space described by the characters joint angles. The problem of low-dimensional human motion synthesis has been already addressed by some authors.[1,2,4,5] Safanova *et al.*[1] proposed a motion synthesis framework able to synthesize new motions by optimizing a set of constraints within a low-dimensional space constructed with PCA. They used IK, just on the characters limbs, as a second step process to clean undesirable artifacts. Glardon *et al.*[5] developed an integrated walking and running engine able to extrapolate data beyond the space described by the PCA basis. IK was also used to prevent feet sliding by exploiting the predictive capability of the model. Grochow *et al.*[2] proposed a non-linear PCA to solve the low-dimensional human motion synthesis problem. However, their approach cannot handle large data sets due to the complexity of the model. Shin *et al.*[4] proposed a framework for low-dimensional motion synthesis by using linear models. They provided a 2D grid describing a low-dimensional representation of the data where the user can synthesize a motion by dragging the mouse pointer on this grid. However, the user cannot add constraints on this 2D grid.

Overview

We formulate a low-dimensional optimization framework with PIK. The key feature of this technique is that prioritized constraints are sorted into layers of priorities. The priority strategy ensures that the most important constraints are satisfied first and the less important ones are satisfied as much as possible without disturbing the vital constraints.[15] Basically, our approach is divided into two main steps:

1. *Learning motion models*: We learn motion models from motion captured data of people performing the same activity many times. Each motion is represented as a normalized motion vector containing N poses (section Motion Representation). Given the motion database, we use a motion model such as PCA or PPCA to find a low-dimensional latent space that can efficiently encapsulate the important characteristics of a specific motion pattern.
2. *Low-dimensional optimization with PIK*: We adapted the Resolved Motion Rate Control (RMRC) proposed by Whitney[16] to solve the low-dimensional optimization problem, by considering a set of user specified constraints with different priorities. To achieve this end, we exploit the chain rule by combining two Jacobian matrices—one relating the position or orientation of a set of end-effectors with a pose—and the other relating this pose with a basis. The final Jacobian relates the increment of the motion model parameters to the displacement of the end-effector in space. The solution is iteratively solved at regular time steps.

Learning Motion Models

Motion Representation

In this section, we recall how a motion can be represented as a normalized motion vector. Let us define a character pose, $\boldsymbol{\Theta}$, as a state vector describing the 3D global position (P_1) and 3D global orientation (θ_1) of the root node, and a set of joint angles (θ_n):

$$\boldsymbol{\Theta} = \left[P_1, \theta_1, \theta_j, \ldots, \theta_n\right] \tag{1}$$

θ_j is the local transformation of the jth joint expressed by exponential map.[17] The number of degrees of freedom (DoF) of the skeleton model is: $3(n+1)$. As each person tends to perform the same activity with some variability in speed, each training motion is time warped normalized using quaternion spherical interpolation,[18] to alleviate time variation duration and to align key postures.[19,20] A motion is then represented as a normalized line vector of the form:

$$\boldsymbol{\Psi} = \left[\boldsymbol{\Theta}_{t_0}, \boldsymbol{\Theta}_{t_k}, \ldots, \boldsymbol{\Theta}_{t_1}\right], \quad 0 \leq k \leq 1 \tag{2}$$

Each motion has dimension: $1 \times D$, $D = 3(n+1) * N$. Where N is the total number of poses. $\boldsymbol{\Theta}_{t_k}$ is a posture corresponding to the normalized time t_k.

PCA Motion Model

Given a set of training motions, the PCA efficiently approximates the motion pattern with a linear combination of the mean motion,

$$\boldsymbol{\Psi}_\circ = \frac{1}{d}\sum_{i=1}^{d} \boldsymbol{\Psi}_i,$$

(d is the number of motions), and a set of Principal Components (PCs).[21] To perform PCA, we compute the eigenvectors of the motion data covariance matrix $\mathbf{S}$:

$$\mathbf{S} = \mathbf{Y}\mathbf{Y}^{\mathrm{T}} \tag{3}$$

$\mathbf{Y}$ is the $d \times D$ mean-subtracted motion matrix whose ith line is written as:

$$\mathbf{Y} = (\boldsymbol{\Psi}_i - \boldsymbol{\Psi}_\circ), \quad 1 \leq i \leq d,$$

$\mathbf{S}$ is a $d \times d$ matrix. Then, the eigenvectors are computed by Singular Value Decomposition (SVD) of $\mathbf{S}$. The SVD gives the matrices $\mathbf{V}$, $\mathbf{U}$, and $\boldsymbol{\Lambda}$, such that:

$$\mathbf{S} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{U}^{\mathrm{T}} \tag{4}$$

where the columns of $\mathbf{V}$ and $\mathbf{U}$ are orthonormal vectors, and $\boldsymbol{\Lambda}$ is a $d \times d$ diagonal matrix containing the non-negative eigenvalues ordered by decreasing magnitude: $\lambda_i \geq \lambda_{i+1}$. By projecting all the motions on the eigenvector matrix $\mathbf{U}$:

$$\widehat{\mathbf{E}} = \sum_{i=1}^{d} \mathbf{u}_i^{\mathrm{T}}(\boldsymbol{\Psi}_i - \boldsymbol{\Psi}_\circ)$$

A linear mapping is constructed to obtain the *eigen-motions*: $\widehat{\mathbf{E}}_i$, $1 \leq i \leq d$[21] (the term *eigen-motions* will refer to the PC learned from motion data). Therefore, any motion $\boldsymbol{\Psi}$ can be approximated as:

$$\boldsymbol{\Psi} \approx \boldsymbol{\Psi}_\circ + \sum_{i=1}^{q} \mathbf{u}_i \widehat{\mathbf{E}}_i \tag{5}$$

where, $\mathbf{u} = (u_1, \ldots, u_q)$ are the Principal Coefficients (PCs) that characterize the motion and $q \leq d$ controls the fraction of the total variance of the training data that is captured by the sub-space denoted by $\rho(q)$:

$$\rho(q) = \frac{\sum_{i=1}^{q} \lambda_i}{\sum_{i=1}^{d} \lambda_i} \tag{6}$$

q represents the number of *eigen-motions* required to approximate a learned motion pattern. Following the notation of Equation (5), a pose can be defined as a function of the scalar coefficients, $\{\mathbf{u}_i\}$ and the normalized time t_k:

$$\psi(t_k, u_1, \ldots, u_q) \approx \boldsymbol{\Psi}_\circ(t_k) + \sum_{i=1}^{q} \mathbf{u}_i \widehat{\mathbf{E}}_i(t_k) \tag{7}$$

$\widehat{\mathbf{E}}_i(t_k)$ represents an *eigen-pose* and $\boldsymbol{\Psi}_\circ(t_k)$ the mean pose for a specific normalized time.

PPCA Motion Model

As the second motion model, we use the PPCA to estimate a set of q-dimensional latent variables $\widetilde{\mathbf{E}}$ (*eigen-motions*) from a set of d-dimensional motion data vectors $\mathbf{\Psi}$. Let us recall the parameters of the PPCA:[22]

$$\sigma^2 = \frac{1}{d-q}\sum_{j=q+1}^{d}\lambda_j \tag{8}$$

$$\mathbf{W} = \mathbf{U}_q(\mathbf{\Lambda}_q - \sigma^2\mathbf{I})^{1/2}\mathbf{R} \tag{9}$$

Differently from PCA, where the PCs with less variance are discarded, in PPCA they are modeled as noise. The noise term σ^2 represents the average loss per discarded dimension. $\mathbf{W}$ corresponds to the q-dimensional *loadings* matrix of the model. $\mathbf{U}_q$ is the $d \times q$ eigenvector matrix of $\mathbf{S}$, with corresponding eigenvalues in the $q \times q$ diagonal matrix $\mathbf{\Lambda}_q$, $\mathbf{R}$ is an arbitrary $q \times q$ orthogonal rotation matrix, and $\mathbf{I}$ is the $q \times q$ identity matrix.

During Maximum Likelihood Estimation (MLE) the matrix $\mathbf{R}$ is set to the identity. However, to recover the true principal axis, $\mathbf{U}_q$, this matrix needs to be computed. According to Bishop *et al.*,[22] $\mathbf{R}$ is the eigenvectors of the $q \times q$ matrix:

$$\mathbf{W}^{\mathrm{T}}\mathbf{W} = \mathbf{R}^{\mathrm{T}}(\mathbf{\Lambda}_q - \sigma^2\mathbf{I})\mathbf{R} \tag{10}$$

In PPCA the PCs and the *eigen-motions* are computed, respectively, as:

$$\widetilde{\mathbf{W}} = \mathbf{W}(\mathbf{W}^{\mathrm{T}}\mathbf{W})^{-1}$$

and

$$\widetilde{\mathbf{E}} = \sum_{i=1}^{d}\mathbf{w}_i^{\mathrm{T}}(\mathbf{\Psi}_i - \mathbf{\Psi}_\circ)$$

We refer the interested reader to Reference [22] for more details about these equations. Following the notations of Equations (5) and (7) any motion $\mathbf{\Psi}$ can be approximated as follows:

$$\mathbf{\Psi} \approx \mathbf{\Psi}_\circ + \sum_{i=1}^{q}\widetilde{\mathbf{w}}_i\widetilde{\mathbf{E}}_i \tag{11}$$

And a pose:

$$\psi(t_{\mathrm{k}}, \widetilde{w}_1, \ldots, \widetilde{w}_q) \approx \mathbf{\Psi}_\circ(t_k) + \sum_{i=1}^{q}\widetilde{\mathbf{w}}_i\widetilde{\mathbf{E}}_i(t_k) \tag{12}$$

Models Comparison

Synthesizing golf swings confronts the system with challenging problems, such as: dealing with different styles of swings, time precision (e.g., the hitting position of the golf club head) and the swing is normally executed in high speeds.[7–9] To handle these problems, we build motion models to operate within the *motion* space instead of the *pose* space.[19]

We learned motion models from golf swing motions played on three different types of terrain: (1) golf swing executed on a flat ground, for a total of 16 motion; (2) golf swing executed on an up slope ground for angles ranging from 0.5° to 5.0° (anti-clockwise), by increments of 0.5°, for a total of 16 motions; (3) golf swing executed on a down slope ground for angles ranging from 0.5° to 5.0° (clockwise), by increments of 0.5°, for a total of 16 motions.

The golf swing motions executed on the up and down slopes were synthetically produced from the flat ground motions by using a motion editing system.[10] Note that these synthetic motions were produced by only adjusting the position of the feet according to the ground inclination, because adjusting the position of the golf club head showed to add discontinuities. In what follows we learned three local models and one global model for both PCA and PPCA, respectively. Local models: one motion model for flat ground swings, one for up slope swings, and one for down slope swings. Global model: a model for the combined flat, up and down slope ground swings (i.e., a multi-pattern model).

The percentage of the database, $\rho(q)$, as a function of the *eigen-motions* for the global and local models is shown in Figure 2. For the local models the percentage is practically the same because the up and down slope

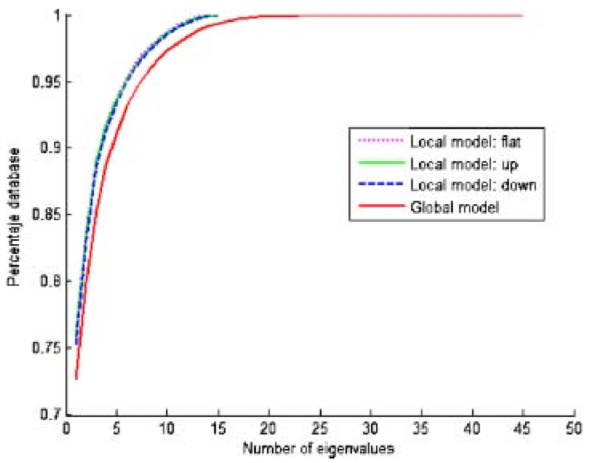


Figure 2. Percentage of the database that can be generated with a given number of eigen-motions. This percentage is the same for the PCA and PPCA motion models because they have the same eigenvalues, λ_i.

motions were produced synthetically. We found that for the locals and global models $q = 15$ *eigen-motions* out of a possible of 16 and 48 captured approximately 100% of the total variance. Note that considering a high database percentage can prevent the addition of artifacts (e.g., foot sliding).[19]

Low-Dimensional Optimization With PIK

Constraints

Constraints tend to fall into two categories:[1,23,24] geometric constraints, such as a point position in space; and physical constraints, such as velocities and forces. Our system is able to handle geometric constrains, such as position or orientation of end-effectors. These constraints are more intuitive for interactive motion synthesis because the user can edit a motion just by dragging the character's end-effector to a new position.

All the constraints are specified directly on the character body and the user needs just to adjust for a single pose end-effectors goals and by associating a priority to them. As a second step, the constraints are projected onto the low-dimensional motion space before the optimizer solves for a pose that is very close to the *eigen-poses* of a specific motion pattern.

Constraints such as position of the center of mass or joint angle limits are discarded, because the solution is achieved within the latent space of physically balanced motions. Likewise it is not necessary to explicitly provide for constraints on accelerations and velocities because, by construction, these constraints are intrinsically modeled through the *eigen-motions*.

Low-Dimensional IK

In this section, we show how to specify a set of constraints and how to solve them within the low-dimensional latent space.

Given a pose of a virtual character as a function of the normalized time and a set of PCs: $\psi(t_k, \alpha_1, \ldots, \alpha_q)$, where α_i can be either u_i or $\widetilde{w}_i$ (Equations (7) and (12), respectively). And a task to satisfy a set of c-constraints:

$$\Delta\mathbf{x} = (\Delta \mathrm{x}_0, \ldots, \Delta \mathrm{x}_c),$$

where $\Delta \mathrm{x}_i$ the ith m-dimensional constraint. We recast the low-dimensional IK problem based on the RMRC technique[15,16] as follows:

$$\Delta\alpha = J_\alpha^{\dagger\xi} \Delta\mathbf{x} + \mathrm{P}_{N(J_\alpha)}\mathbf{z} \tag{13}$$

with

$$\mathrm{P}_{N(J_\alpha)} = \mathbf{I}_q - J_\alpha^{\dagger} J_\alpha$$

$\Delta\alpha$ is the optimal PCs increment, J_α is the $(mc \times q)$ Jacobian relating the tasks' increment with the PCs, $\mathrm{P}_{N(J_\alpha)}$ is the projection operator onto the Null-space of J_α, $J_\alpha^{\dagger\xi}$ is the damped pseudoinverse, $J_\alpha^{\dagger}$ is the pseudoinverse, $\mathbf{I}_q$ is the $q \times q$ identity matrix and $\mathbf{z}$ is an arbitrary vector expressing a variation of a posture on the *eigen-motions* space.[23,25] The Jacobian J_α can be easily computed with the chain rule:[19]

$$J_\alpha = J_{\boldsymbol{\Theta}_{t_k}} J_{\mathbf{E}(t_k)} \tag{14}$$

$J_{\boldsymbol{\Theta}_{t_k}}$ is the Jacobian expressing the full dimensional joint angle space (the root node is included), and the Jacobian $J_{\mathbf{E}(t_k)}$ can be interpreted as the linear mapping onto the low-dimensional *eigen-motions* space—where $\mathbf{E}(t_k)$ can be either $\widehat{\mathbf{E}}(t_k)$ or $\widetilde{\mathbf{E}}(t_k)$ (Equations 7 and 12, respectively). By using the proposed formulation (Equation (13)) the problem can easily becomes over-constrained due to the restrictive space provided by the low-dimensional motion space or if more than one pose is constrained during the synthesis process. We alleviate this problem by constructing *eigen-motions* of the same behavior, where constraints can be met, and by constraining only one frame of the motion. However, if no solution exists to the IK problem, a solution that minimizes the error norm is found.

Solving Multiple Constraints With PIK Within the Latent Space

In this section, we show how constraints with different priorities can be solved, how our system can automatically synthesize the motion as a whole, and how artifacts can be managed. The complete solution that extends Equation (13) and solves the low-dimensional optimization problem with PIK is described by Algorithm 1. The solution is found by iteratively minimizing the error norm.[19] j is the current priority-layer, p is the total number of priority-layers, and $J_{\alpha_j}^{\mathrm{A}}$ is the augmented Jacobian,[23] here defined as:

$$J^{\mathrm{A}}_{\alpha_j} = \begin{bmatrix} J_{\alpha_1} \\ J_{\alpha_2} \\ \vdots \\ J_{\alpha_j} \end{bmatrix} \tag{15}$$

Algorithm 1. Low-dimensional optmization.

```
1: $\tilde{\alpha} \leftarrow \alpha$; $\Delta\tilde{\alpha}_\circ \leftarrow \mathbf{0}$; $J_{\mathbf{E}(t_k)} \leftarrow \mathbf{E}(t_k)$, $\mathrm{P}_N(J_{\tilde{\alpha}_0}) \leftarrow \mathbf{I}_q$
2: while not converged do
3:   Compute $\{J^{\mathrm{A}}_{\alpha_j}, \Delta\mathbf{x}\}$
4:   for $j = 1$ to $p$ do
5:     $\Delta\hat{\mathrm{x}}_j \leftarrow \Delta\mathrm{x}_j - J_{\tilde{\alpha}_j}\Delta\tilde{\alpha}_{j-1}$
6:     $\tilde{J}_{\tilde{\alpha}_j} \leftarrow J_{\tilde{\alpha}_j}\mathrm{P}_{N\left(J^{\mathrm{A}}_{\tilde{\alpha}_{j-1}}\right)}$
7:     $\Delta\tilde{\alpha}_j \leftarrow \Delta\tilde{\alpha}_{j-1} + J^{\dagger\xi}_{\tilde{\alpha}_j}\Delta\hat{\mathrm{x}}_j$
8:     $\mathrm{P}_{N\left(J^{\mathrm{A}}_{\tilde{\alpha}_j}\right)} \leftarrow \mathrm{P}_{N\left(J^{\mathrm{A}}_{\tilde{\alpha}_{j-1}}\right)} - \tilde{J}^{\dagger}_{\tilde{\alpha}_j}\tilde{J}_{\tilde{\alpha}_j}$
9:   end for
10:  $\Delta\tilde{\alpha} \leftarrow \Delta\tilde{\alpha}_p + \mathrm{P}_{N(J^{\mathrm{A}}_{\tilde{\alpha}_p})}\mathrm{z}$
11:  $\tilde{\alpha} \leftarrow \tilde{\alpha} + \Delta\tilde{\alpha}$
12:  $\mathbf{\Theta}_{t_k} \leftarrow \mathbf{\Psi}_\circ(t_k) + \tilde{\alpha}\mathbf{E}(t_k)$
13: end while
14: $\tilde{\mathbf{\Psi}} \leftarrow \mathbf{\Psi}_\circ + \sum_{i=1}^{q} \tilde{\alpha}_i\mathbf{E}_i$
```

Note that the $\tilde{\alpha}$ vector defines one full motion resulting from the constraint on one pose $\mathbf{\Theta}_{t_k}$. Our systems can synthesize the others frames when the convergence is completed (line 14 of Algorithm 1). Moreover, continuity between frames is intrinsically enforced through the latent space.

The key advantage of associating a higher priority to a constraint is the guarantee to converge to a motion that at least enforce the most important ones in terms of realism (e.g., feet on the ground). An important parameter used in our system to smooth out artifacts is the regularization factor ξ (Equation 13), giving a high value for the damping stabilize singular context but at the cost of a slower convergence. Such singular context happens when the PIK solver tries to achieve unreachable poses, i.e, poses that are not in the database. We exploit the approach proposed by Maciejewsky *et al.*[26] to compute an appropriate damping value. This value is a function of the minimum singular value of J_α.

Experiments

In this section, we analyze the performance and robustness of our framework and show a number of synthesized results generated for a virtual character with 93 DoF. To synthesize a motion with our approach the user needs to specify the final goal position and the priority value of each end-effector just for one frame—the one that should be adjusted—and the optimizer solves for a pose that is very close to the *eigen-poses* of a specific motion pattern. As a second step, the system automatically updates the solution for the others poses. The motion synthesis framework proposed in this paper was integrated into the Autodesk/Maya software as plug-in and MEL scripts. In all the experiments reported in this paper we add constraints near the golf club head to control the hit position of the golf club, and in some cases on the feet to treat slope situations. Figure 3 shows a common constraining configuration used in our system to synthesize a golf swing motion. All the experiments were run on a 3.2 GHz Pentium Xeon(TM) with 1 GB memory.

Convergence Performance

The convergence performance of our system depends on many parameters: the choice of a motion model, the priority given for constraints, and the intrinsic parameters of the optimizer (i.e., the damping factor, $\xi = 10$, and the number of iterations, which we set to 1500). The computing cost of each iteration is approximately 5.3 ms.

We verified the convergence performance of our system by synthesizing the same golf swing played on a

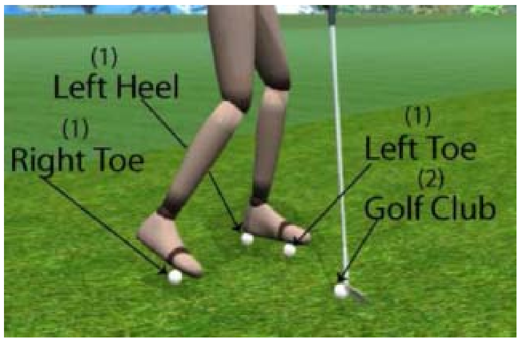


Figure 3. A typical constraint configuration used to edit a golf swing. 1 means the highest priority and 2 is the second one, see Algorithm 1 for more details.

flat ground in three different constraints configurations: first, by attaching one constraint near of the golf club head, to synthesize a flat ground swing just by shifting the hit position; second, by attaching three constraints on the feet with highest priority and one constraint near the golf club head with lower priority, to synthesize an up slope swing with 5°; third, by attaching three constraints on the feet and one constraint near the golf club with the same priorities, to synthesize an up slope swing with 5°. The hit position was the same in all three cases. For each configuration, we used eight motion models: one local model learned from flat swings, one local model learned from up slope swings, one local model learned from down slope swings and a multi-pattern model for the combined flat, up, and down slope swings, for PCA and PPCA, respectively.

The optimization convergence runs faster when a local model of the same behavior of the synthesized motion was used. Giving different levels of priority for each set of constraints also provided faster convergence solutions than giving the same importance for all the constraints, thanks to the priority strategy formulation used in our system.[23] Figure 4(a) and (b) shows the convergence results for the local models learned from flat motions and for the multi-pattern model. The '*y*' axis shows the error norm. We found similar convergence results, as shown in Figure 4, for the local models learned from up and down slopes swings.

Local Versus Global Models

The low-dimensional space provided by local models produced realistic results for synthesized animations of the same behavior of the *eigen-motions*. We verified this by synthesizing a flat ground swing motion on an up slope ground. When we used *eigen-motions* learned from a database containing up slope swing motions all the synthesized results demonstrated realism, but *eigen-motions* learned from down slope or flat grounds swings produced less realistic results. In the last case, we noticed that the algorithm converged for the adjusted frame producing a realistic pose, but the *eigen-motions* were not able to synthesize the motion as a whole. This happened because the *eigen-motions* should have a similar information of slope to propagate this information through the motion.

The low-dimensional space provided by global models learned from the flat, up, and down slopes golf swings motions generated more synthesized motions. We observed this by synthesizing a flat ground swing motion in different situations: considering flat ground and up and down slopes grounds with different degrees of inclination, and considering different hit positions. In the majority of the cases plausible synthesized animations were produced. However, we noticed that the algorithm needed more iterations to converge and to consequently produce similar quality results achieved by local models.

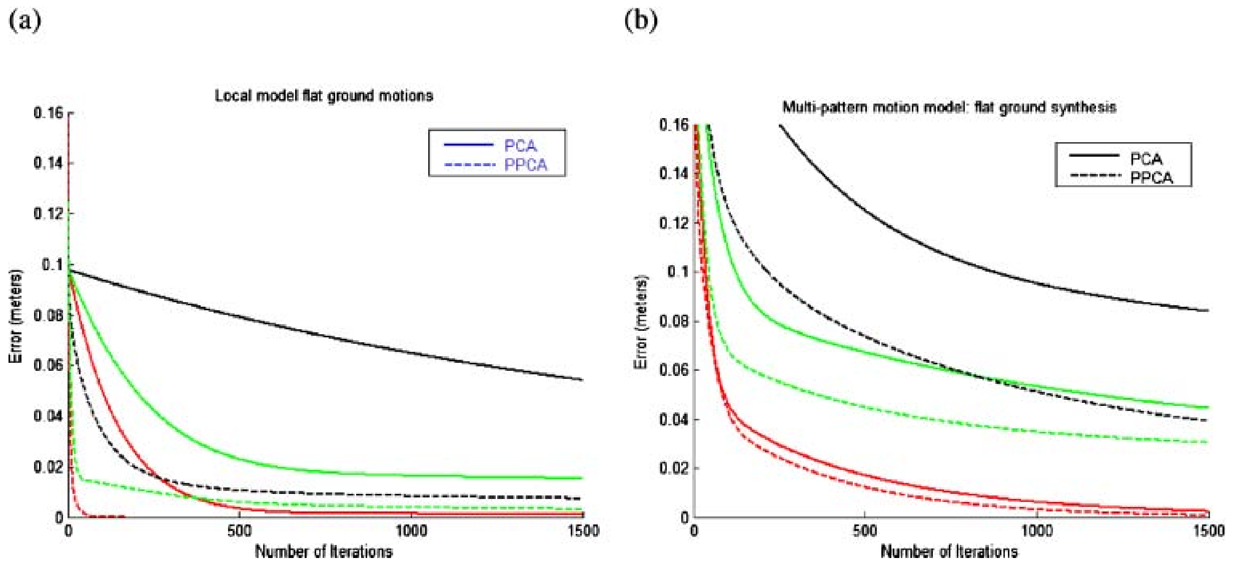


Figure 4. Convergence performance. The red, green, and black lines corresponding to the first, second, and third constraints configuration (see 'Convergence Performance' subsection for more details). (a) Local model learned from flat swings; (b) global model learned from flat, up, and down swings.

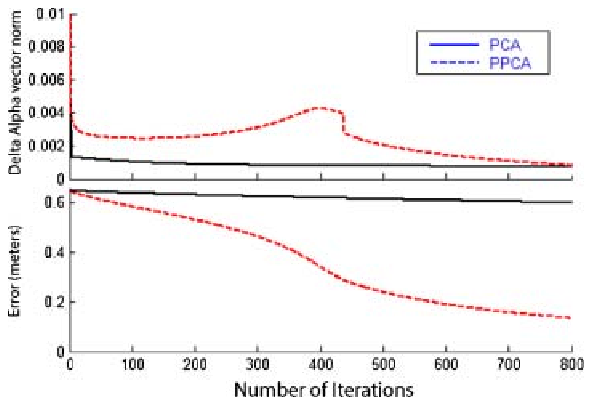


Figure 5. Extrapolation context.

PCA Versus PPCA

We verified that in some experiments (e.g., when the optimizer tried to achieve unreachable poses) the PCAs motion models produced synthesis results with much less artifacts compared to the PPCAs ones; for the same value of the regularization factor, (ξ), and the same number of iterations.

Figure 5 shows the error norm and the delta alpha vector norm—$\|\Delta\alpha\|$—of an extrapolation context. Note the smoother convergence of PCA compared to PPCA.

Although PPCA has demonstrated a faster convergence, it produced results with discontinuities in some joints.

Figure 6 shows the trajectory of the left elbow joint in quaternion space, for the original and the synthesized motions obtained with PCA and PPCA, by using local models (similar results were found with global models). However, these artifacts were gracefully removed by increasing the value of the damping factor.

Comparisons With Per-Frame PIK

In this section, we compare our approach against a per-frame PIK motion editing technique[10] regarding performance, robustness, simplicity, and realism, by

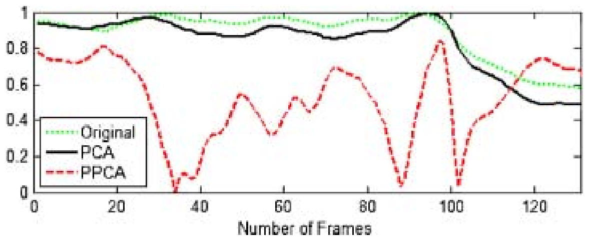


Figure 6. Rotation trajectory of the left elbow.

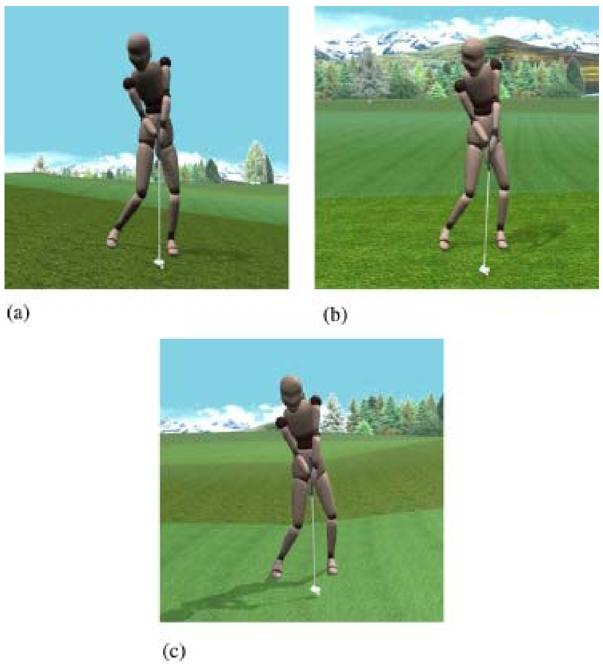


Figure 7. Synthesized motions, by using our system, from the swing motion shown in Figure 1. (a) 7° down slope ground; (b) flat ground; (c) 7° up slope ground.

synthesizing golf swing motions executed on three different types of terrain: flat, up, and down slope grounds (Figure 7, submitted video). To edit a motion with the per-frame PIK the user has to specify the trajectory and the priority of each end-effector and the ease-in and ease-out time intervals for which the constraints will be activated. Note that this technique can add more significant deformations to an input motion than our approach.

For a fair comparison of both techniques, in the per-frame technique we set the parameters of the optimizer as suggested by the authors (first line of Table 1) to obtain the best tradeoff between realistic synthesized motions and computing time performance. We made the same with our approach (second line of Table 1) and we used the PCAs local models. For the experiments, we synthesized the same golf swing motion, with

First case	Second case	Third case
$I = 100$, $D = 10$	$I = 100$, $D = 10$	$I = 100$, $D = 10$
$I = 25$, $D = 1.5$	$I = 400$, $D = 0.8$	$I = 400$, $D = 0.8$

Table 1. Optimizer parameters values I is the number of iterations and D is the damping.

132 frames, executed on a flat ground (Figure 1) in three different cases: first, by synthesizing a golf swing executed on a flat ground just by shifting the hit position of the golf club head, Figure 7(b); second, by synthesizing a golf swing executed on a flat ground by considering an up slope ground with 7° anti-clockwise and also by shifting the hit position of the golf club head, Figure 7(c); and third, by synthesizing a golf swing executed on a flat ground by considering a down slope ground with 7° clockwise and also by shifting the hit position of the golf club head, Figure 7(a). Note that the 7° slope was not learned by the *eigen-motions*. The adjusted frame, 94, and the position of the golf club with respect to the hands of the virtual character were the same in all situations.

By using the per-frame approach in all the three cases, we needed to add five constraints: three on the feet with highest priority (activated frames 1–132), to prevent foot sliding or to elevate the feet according to the ground inclination; one on the center of mass (activated frames 1–132) with lower priority, to prevent unbalance poses; and one near the golf club head (activated frames 90–99) with middle priority, to control the hit position. The computing cost for the first case was approximately 80 second and for the second and third cases was approximately 105 second, without visualization. We also noticed inter penetration of the arms in some situations and always the presence of discontinuities (e.g., body balance) around the hit position (see, submitted videos). Such artifact is usually fixed with a pass filtering stage that may alter constraint enforcement.

On the contrary, with the proposed approach only one constraint was necessary in the first case—the golf club head constraint—to achieve a globally continuous motion. The computing cost for the first case was approximately 0.2 second, without visualization. For the second and third cases, we needed to add three more constraints in addition to the golf club head: three on the feet with highest priority to control the position of the feet according to the ground inclination. The computing cost for the second and third cases was approximately 3 second, without visualization. Our approach provided more fluid motions and faster results because we just needed to constrain one key event: the hit frame.

Conclusions

We have presented a new approach for interactive low-dimensional human motion synthesis, by combining motion models and prioritized inverse kinematics. We developed a constrained optimization framework to solve the synthesis problem within a low-dimensional latent space. In this space, a movement enforcing a new set of user specified constraints could be obtained in just one step. We demonstrated the performance, robustness, and simplicity of our approach by synthesizing various styles of golf swings; and by comparing the quality of the synthesized results against a per-frame PIK technique. We showed that by constraining and adjusting just one key frame our approach provided faster and more fluid results without the need of an additional filtering stage. Moreover, building *motion* models instead of *pose* models demonstrated two important advantages, i.e., our system could automatically synthesize the movement as a whole without introducing artifacts and impose continuity between frames through the latent space.

Given an appropriate database containing motion samples of the same style, we can build local models and use them to improve motion synthesis. The experiments reported in this paper showed that local models provided faster synthesis results compared to multi-pattern models. Likewise, by giving constraints different priorities also demonstrated an improvement in synthesis performance. In interpolation contexts, i.e., when the optimizer solved for a motion closer to the database motions, PCA and PPCA models produced very similar synthesized results. In extrapolation contexts, the PCA produced smoother convergence compared to the PPCA. However, we attenuated this problem by increasing the value of the damping factor.

Extrapolation and interpolation of the golf swings could be performed from 0° to 8° for up and down slopes, beyond this value the algorithm converges toward a result minimizing the error norm. We believe that this problem can be solved by increasing the variability of the motion database (e.g., by adding higher slopes information of the same swing style).

Our approach is well suited to deal with deformations and retargeting problems. In this paper, we have mainly focused our study on synthesizing motions that need a great precision while moving in high speeds. Our approach can also be exploited for a wider range of coordinated motions (e.g., baseball, tennis, jumping, boxing, and others).

ACKNOWLEDGEMENTS

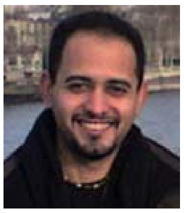
Many thanks to Mireille Clavien for the rendering and the video production showed in this paper. The authors would like to thank to Autodesk/Maya for their donation of Maya software. Many thanks to Benoît Le Callennec, to providing access to his motion editing system (with the support of the Swiss National

Fund, grant no. 200020-109989). This project was sponsored by the EPFL—Sport and Rehabilitation Engineering program.

References

1. Safonova A, Hodgins JK, Pollard NS. Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. *ACM Transactions on Graphics* 2004; **23**(3): 514–521.
2. Grochow K, Martin SL, Hertzmann A, Popovi Z. Style-based inverse kinematics. *ACM Transactions on Graphics* 2004; **23**(3): 522–531.
3. Glardon P, Boulic R, Thalmann D. A coherent locomotion engine extrapolating beyond experimental data. In *Proceedings of CASA*, July 2004; pp. 73–84.
4. Shin HJ, Lee J. Motion synthesis and editing in low-dimensional spaces: research articles. *Computer Animation and Virtual Worlds* 2006; **17**(3–4):219–227.
5. Glardon P, Boulic R, Thalmann D. Robust on-line adaptive footplant detection and enforcement for locomotion. *The Visual Computer* 2006; **5**(6): 194–209.
6. Farrally MR, Cochran AJ, Crews DJ, *et al*. Golf science research at the beginning of the twenty-first century. *Journal of Sports Sciences* 2003; **21**: 753–765.
7. Gehrig N, Lepetit V, Fua P. Golf club visual tracking for enhanced swing analysis tools. In *British Machine Vision Conference*, Norwich, UK, 2003.
8. Urtasun R, Fleet D, Fua P. Monocular 3-d tracking of the golf swing. In *CVPR*, vol. 1, June 2005; pp. 932–939.
9. Raymond AP. The physics of golf. *Reports on Progress in Physics* 2003; **66**(2): 131–171.
10. Le Callennec B, Boulic R. Interactive motion deformation with prioritized constraints. *Graphical Models* 2006; **68**(2): 175–193. Special Issue on SCA 2004.
11. Gleicher M. Comparing constraint-based motion editing methods. *Graphical models* 2001; **63**(2): 107–134.
12. Kulpa R, Multon F, Arnaldi B. Morphology-independent representation of motions for interactive human-like animation. In *EUROGRAPHICS*, vol. 24, August–September 2005; pp. 343–352.
13. Lee J, Shin SY. A hierarchical approach to interactive motion editing for human-like figures. In *Proceedings of ACM SIGGRAPH*, 1999.
14. Liu L, Zhao-qi W, Deng-Ming Z, Shi-Hong X. Motion edit with collision avoidance. In *Proceedings of the WSCG*, January 2006; pp. 303–310.
15. Baerlocher P, Boulic R. An inverse kinematic architecture enforcing an arbitrary number of strict priority levels. *The Visual Computer* 2004; **20**(6): 402–417.
16. Whitney DE. Resolved motion rate control of manipulators and human prostheses. *IEEE Transactions on Man-Machine Systems* 1969; **10**(2): 47–53.
17. Grassia FS. Practical parameterization of rotations using the exponential map. *Journal of Graphics Tools* 1998; **3**(3): 29–48.
18. Shoemake K. Animating rotation with quaternion curves. In *SIGGRAPH'85*. ACM Press: New York, NY, USA, 1985; 245–254.
19. Carvalho SR, Boulic R, Thalmann D. Motion pattern preserving ik operating in the motion principal coefficients space. In *Proceedings of 15th WSCG*, January–February 2007; pp. 97–104.
20. Urtasun R, Fleet DJ, Fua P. Temporal motion models for monocular and multiview 3d human body tracking. *Computer Vision and Image Understanding* 2006; **104**(2): 157–177.
21. Jolliffe IT. *Principal Component Analysis*. Springer-Verlag: New York Inc. 1986.
22. Tipping ME, Bishop CM. Mixtures of probabilistic principal component analysers. *Neural Computation* 2006; **11**(2): 443–482.
23. Baerlocher P. *Inverse Kinematics Techniques of The Interactive Posture Control of Articulated Figures*. Phd thesis, cole Polytechnique Fdral de Lausanne, 2001.
24. Gleicher M. Motion editing with spacetime constraints. In *Proceedings of SI3D'97*. ACM Press: New York, NY, USA, 1997; 139–148.
25. Maciejewski AA. Dealing with the ill-conditioned equations of motion forarticulated figures. *IEEE Computer Graphics and Applications* 1990; **10**: 63–71.
26. Maciejewski AA, Klein CA. Numerical filtering for the operation of robotic manipulators through kinematically singular configurations. *Journal of Robotic Systems* 1988; **15**: 527–552.

Authors' biographies:



Schubert Ribeiro de Carvalho is a Ph.D. candidate and Research Assistant in the Virtual Reality Lab at the Swiss Federal Institute of Technology, Lausanne (EPFL), under the supervision of Dr Daniel Thalmann and Dr Ronan Boulic. He received his M.Sc. degree in Computer Science in 2005 from the State University of Campinas (UNICAMP), Brazil and B.S. degree in Computer Science in 2001 from the Federal University of Pará (UFPA), Brazil. His research interests include human motion modeling and synthesis, low-dimensional constrained optimization, motion planning, and computer animation and robotics. His doctoral thesis involves the development of parametric models of athletic motions.

Ronan Boulic is a Senior Researcher, Lecturer, and Ph.D. Advisor at the Swiss Federal Institute of

Technology, Lausanne (EPFL). He is working in the Virtual Reality Lab and his research interests include 3D interaction, motion capture, modeling, and synthesis for virtual humans and robots. He received Ph.D. in Computer Science in 1986 from the University of Rennes, France, at the INRIA-IRISA research institute. He received the Habilitation degree from the University of Grenoble, France, in March 1995. Ronan Boulic is co-author of 76 research papers that appeared in international peer-reviewed conferences, and of 26 papers that appeared in international peer-reviewed journals. He has been the chairman of the Eurographics Workshop on Computer Animation and Simulation 1996, program committee member of Eurographics, ACM SCA, Computer Animation, and Computer Graphics International, and he was part of the organization committee of Eurographics'03 and '05. He was a senior reviewer for Eurographics'04, and Paper co-chair of the Eurographics/SIGGRAPH Symposium on Computer Animation 2004 in Grenoble. He is a senior member of IEEE and of ACM, and member of Eurographics.

Daniel Thalmann is a 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, computer animation, and 3D interaction. Daniel Thalmann has been a 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 the President of the Swiss Association of Research in Information Technology and one Director of ERCIM. He is Co-Editor-in-Chief of the *Journal of Computer Animation and Virtual Worlds*, and member of the editorial board of the *Visual Computer* and four other journals. Daniel Thalmann was member of numerous Program Committees, Program Chair, and Co-Chair of several conferences. He is the Program Co-Chair of CGI 2007 and Virtual Rehabilitation 2007, and the Conference Co-Chair of CASA 2007, SCA 2007, and ACM VRST 2008. He has also organized five courses at SIGGRAPH on human animation. Daniel Thalmann has published more than 400 papers in Graphics, Animation, and Virtual Reality. He is co-editor of 30 books, and co-author of several books including the *Handbook on Virtual Humans*, published by John Wiley and Sons. He received his Ph.D. in Computer Science in 1977 from the University of Geneva and an Honorary Doctorate (Honoris Causa) from University Paul-Sabatier in Toulouse, France, in 2003.