Reconstruction of Human Motions Based on Low-Dimensional Control Signals

Dissertation
zur Erlangung des Doktorgrades (Dr. rer. nat.)
der
Mathematisch-Naturwissenschaftlichen Fakultät
der Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von

Jochen Tautges
geboren in Prüm

Bonn, 01. März 2012

Promotionskommission

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Rheinischen Friedrich-Wilhelms-Universität Bonn.

- Erstgutachter: Prof. Dr. Andreas Weber
- Zweitgutacher: Prof. Dr. Norman I. Badler
- Fachnahes Mitglied: Prof. Dr. Rolf Klein
- Fachfremdes Mitglied: Prof. Dr. Werner Seiler

Tag der mündlichen Prüfung: 27. Juli 2012
Erscheinungsjahr: 2012

Contents

List of Figures

Abstract

This thesis addresses the question to what extent it is possible to reconstruct human full-body motions from very sparse control signals. To this end, we first investigate the use of multi-linear representations of human motions. We show that multi-linear motion models together with knowledge from pre-recorded motion capture databases can be used to realize a basic motion reconstruction framework that relies on very sparse inertial sensor input only. However, due to the need for a semantic pre-classification of the motion to be reconstructed and rather restricting database requirements, the described framework is not suitable for a more general motion capture scenario.

We address these issues in a second, more flexible approach, which relies on sparse accelerometer readings only. Specifically, we employ four 3D accelerometers that are attached to the extremities of a human actor to learn a series of local models of human poses at runtime. The main challenge in generating these local models is to find a reliable mapping from the low-dimensional space of accelerations to the high-dimensional space of human poses or motions. We describe a novel online framework that successfully deals with this challenge. In particular, we introduce a novel method for very efficiently retrieving poses and motion segments from a large motion capture database based on a continuous stream of accelerometer readings, as well as a novel prior model that minimizes reconstruction ambiguities while simultaneously accounting for temporal and spatial variations.

Thirdly, we will outline a conceptually very simple yet very effective framework for reconstructing motions based on sparse sets of marker positions. Here, the sparsity of the control signal results from problems that occurred during a motion capture session and is thus unintentional. As a consequence, we do not control the information we can access, which introduces several new challenges. The basic idea of the presented framework is to approximate

Acknowledgements

First and foremost I want to thank my advisor Andreas Weber for his great support during both my undergraduate and postgraduate studies, for creating an inspiring as well as enjoyable work environment, for always being available even though not always being present, and for not only allowing but even helping me to make my way to the other side of the world before having finished my thesis, which by no means can be taken for granted.

Furthermore I want to thank my former workmates at the University of Bonn and permanent friends Björn Krüger, Arno Zinke, Jan Baumann, and Tomas Lay Herrera for contributing to my work, my knowledge, and my weekends. I owe particular thanks to Björn for the large amount of fast and extremely valuable feedback on my writings, and for making it so easy for me to submit this thesis from the other side of the world.

I would also like to thank J.P. Lewis for his support and advice throughout my work at Weta Digital, part of which is presented in this thesis.

Finally, I want to thank Désirée for accepting my frequent physical absence during the last year, for waking me up in times of mental absence, and for being there for me, then and now.

1 Introduction

Motion capture, the process of recording movement and transferring it onto a digital model, is nowadays widely used in military, entertainment, sports, medical, and computer graphics applications. When used in filmmaking, the term *performance capture* has recently become very popular, describing the attempt to simultaneously capture *all* aspects of an actor's performance (e.g., including his facial expressions) rather than only his overall body movements. Most commercially available (full-body) motion capture systems are based on *optical* sensors (e.g., arrays of calibrated cameras), which are used to triangulate the 3D positions of special markers attached to an actor. While these systems typically allow tracking and recording human motions at very high spatial and temporal resolutions, they are in general very cost-intensive regarding software, equipment, and data acquisition. In addition, they often impose limiting constraints on the actor and the recording environment. In recent years, low-cost motion tracking systems have become popular, especially in video game and sports applications. Examples for this are depth sensor-based systems like Microsoft's Kinect, or accelerometer-based devices like Nintendo's Wii, Nike Plus, or Adidas MiCoach, which can be easily attached to an actor's body or even fit in a shoe. The information obtained from such sensors is, however, often low-dimensional and affected by noise, so that in general high-dimensional motions cannot be inferred without incorporating additional knowledge. Here, data-driven methods, which incorporate such additional knowledge in the form of pre-recorded motion capture databases, have turned out to be a powerful approach. When using low-dimensional control signals to capture full-body motions, many degrees of freedom are necessarily unconstrained. In order to eventually yield high-dimensional full-body animations, these degrees of freedom then need to be synthesized or *reconstructed* using the knowledge embedded in the database. This is why I will refer to the task of creating full-body motions from low-dimensional measurements of a performance as motion *reconstruction* rather than motion capture.

In this thesis I will present three different data-driven approaches to motion reconstruction. The main focus lies on control signals obtained from inertial sensors (Chapters 2 and 3), and I will in particular present a system for reconstructing motions on the basis of sparse *accelerometer* readings only (Chapter 3). Due to their estimative character, motion reconstruction frameworks are in general neither able nor meant to be a substitute for high-quality (full-body) motion capture systems. This holds especially for the film industry, where an actor's movements have to be accurately reproduced rather than estimated. Sometimes, however, very demanding motion capture conditions cause significant gaps in the actually acquired data, which—due to the high costs of a motion capture session—often have to be manually filled by artists afterwards. In addition to the aforementioned techniques that rely on sparsely *designed*, inertial-based control input, I will thus also describe an approach to motion reconstruction that deals with such *unintentionally* sparse, position-based control data (Chapter 4).

2 Motion reconstruction using multi-linear motion models

2.1 Introduction

Motion representations based on linear models together with linear dimensionality reduction techniques like principal component analysis (PCA) have become well-established techniques in motion synthesis applications [Tro02, SHP04, GBT04, BSP$^+$04, CH05, FF05, LZWM05, OBHK05]. Using these methods one neglects information about the motion sequences, such as the temporal order of the frames, information about different actors, or semantic information, which is often given when dealing with motion capture databases.

In the context of facial animation, Vlasic et al. [VBPP05] have successfully applied multi-linear models of 3D face meshes that separably parameterize semantic aspects such as identity, expression, and visemes. Following up work that I started in my diploma thesis [Tau07], Krüger et al. [KTW07, KTMW08] investigated how multi-linear models can be used to represent human (full-body) animations. Here, motion data was encoded in high-order tensors, whose various modes explicitly account for both technical and semantic aspects of not only one motion but an entire class of (semantically) related motions. It was not only shown that reduction techniques based on higher-order singular value decomposition (HOSVD) outperform linear PCA-models in terms of data compression, but also how tensor representations can be used for synthesizing new motions. Building upon these ideas, we created a multi-linear framework that—under certain assumptions—allows us to create naturally looking full-body animations that are driven by extremely sparse control signals. The ideas discussed in this chapter have originally been presented in the paper *Reconstruction of Human Motions Using Few Sensors* [TKZW08]. A very similar usage of motion tensors was later presented by Min et al. [MLC10]. Here, the authors applied multi-linear analysis techniques to construct a generative motion model for synthesis, retargeting, and editing of personalized human motion styles.

When comparing our reconstructed motions with ground-truth motions, we realized that the established approaches to compute a distance between motions on the average error of local joint orientations can fail: Being purely pose-based, the distance measure might fail to detect artifacts like directional flips or jitter (i.e., the distance between the original motion and the reconstructed motion is small although the latter exhibits these artifacts). Therefore, we also present a novel practical distance measure for comparing motions based on quantities represented in a global coordinate frame. Assuming a fixed skeleton topology, our goal is a universal measure that both matches the human perception and is simple enough to be implemented in time-critical environments.

Before diving into motion synthesis, I will briefly recall the basics of multilinear algebra and the use of multi-linear models for representing human motion data [Vas02, RCO05, MK06, KTW07, KTMW08] that will be central to our approach.

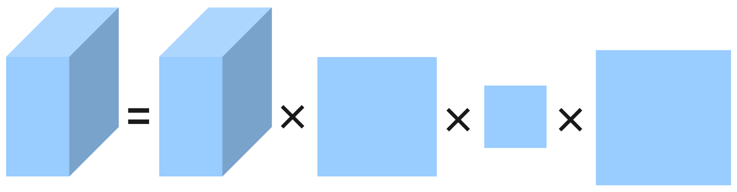


Figure 2.1: Simplified representation of the HOSVD for a tensor of order 3: The data tensor is shown on the left side, its decomposition into a core tensor and three orthonormal matrices on the right side of the equation.

2.2 Basics

2.2.1 Multi-linear algebra

Multi-linear algebra is a natural extension of linear algebra. A *tensor* Θ of *order* $N \in \mathbb{N}$ and type $(d_1, d_2, \ldots, d_N) \in \mathbb{N}^N$ over the real number $\mathbb{R}$ is defined to be an element in $\mathbb{R}^{d_1 \times d_2 \times \ldots \times d_N}$. The number $d := d_1 \cdot d_2 \cdot \ldots \cdot d_N$ is referred to as the *total dimension* of Θ. Intuitively, the tensor Θ represents d real numbers in a multi-dimensional array based on N indices. These indices are also referred to as the *modes* of the tensor, and the vectors spanned by the $k-th$ mode (those aligned with the k-th axis of the tensor) are referred to as mode-k-vectors. As an example, a vector $v \in \mathbb{R}^d$ is a tensor of order $N = 1$, having only one mode. Similarly, a matrix $M \in \mathbb{R}^{d_1 \times d_2}$ is a tensor of order $N = 2$, having two modes that correspond to the columns and rows.

A tensor Θ can be transformed by a higher-order singular value decomposition (HOSVD), sometimes also referred to as *N-mode singular value decomposition*. The result of the decomposition is a *core tensor* Φ of the same size as Θ and associated orthonormal matrices $U_1, U_2, \ldots, U_N$. The matrices U_k are elements in $\mathbb{R}^{d_k \times d_k}$, where $k \in \{1, 2, \ldots, N\}$. Figure 2.1 shows a simplified graphical representation of this decomposition. Mathematically, this decomposition can be expressed in the following way:

$$\Theta = \Phi \times_1 U_1 \times_2 U_2 \times_3 \ldots \times_N U_N. \tag{2.1}$$

This product is defined recursively, where the mode-k-multiplication $\times_k$ with

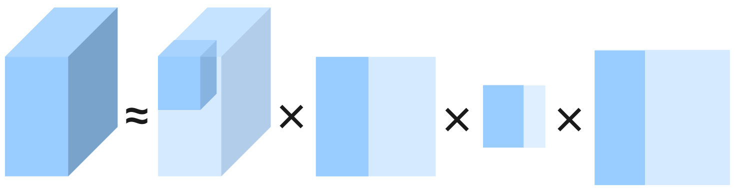


Figure 2.2: Since variance is concentrated in one corner of the core tensor, the data tensor (on the left hand side) can be approximated by truncated versions (drawn in darker blue) of the core tensor and the respective matrices.

U_k replaces each mode-k-vector v of $\Phi \times_1 U_1 \times_2 U_2 \ldots \times_{k-1} U_{k-1}$ for $k > 1$ (and Φ for $k = 1$) by the vector $U_k v$. One important property of Φ is that its elements are sorted in a way that the variance decreases from the first to the last element in each mode [VBPP05]. A reduced model Φ' can be obtained by truncating insignificant components of Φ and the matrices U_k, respectively (see Figure 2.2). In the special case of a 2-mode tensor, this procedure is equivalent to principal component analysis (PCA) [Vas02]. A more detailed description of multi-linear algebra is given in [VBPP05], and a very comprehensive discussion of the HOSVD can be found in [LMV00]. Furthermore, Kolda and Bader [KB09] provide an overview of different higher-order tensor decompositions, their applications, and available software.

2.2.2 Motion tensors

In our case tensors are filled with motion data similar to the way Krüger et al. suggest [KTW07, KTMW08]. A frame is defined by the position of its root node p and quaternions $(q_1 \ldots q_{31})$ describing the orientations of the skeleton segments. A motion is defined to be a sequence of frames. We build two motion tensors, one for the root positions, in the following denoted by Θ_p, and one for the rotational data, in the following denoted by Θ_q. The reason for separating these types of data into two separate tensors is their difference in variance: While the values of unit quaternions are in the interval $[-1, 1]$, the translational offset of the root position is not limited at all.

In these tensors, data is arranged in what is referred to as *technical modes*, which correspond to the structure of the underlying motion capture data,

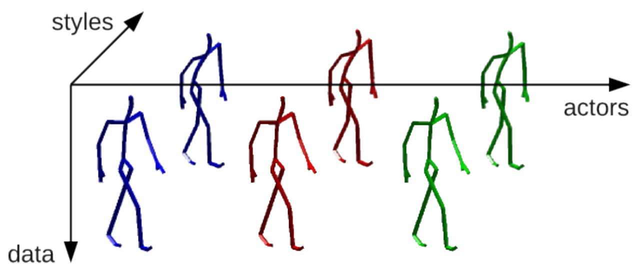


Figure 2.3: Visualization of a motion tensor of order 3, having one technical mode (*data*) and two natural modes (*styles* and *actors*). Each skeleton represents a sequence of poses.

and *natural modes*, which correspond to properties of motions that typically appear in the context of a motion capture session. The technical modes split up into *DOF mode*, *Joint mode* and *Frame mode*, the natural modes into *Style mode*, *Actors mode* and *Repetition mode*. Whenever the size of a tensor is given in this work, the order of its modes will exactly follow this order, with technical modes preceding the natural modes. Due to the structure of the database used in all our experiments (see also Section 2.3.5), the size of the Actors mode will always be 5, and the sizes of both Repetition and Style mode will always be 3. With our skeleton representation having 31 joints, and each local orientation expressed as a 4-dimensional unit quaternion, a typical tensor Θ_q has a dimension of $N = 6$ and a size of $d = 4 \times 31 \times F \times 3 \times 5 \times 3$, with F being the number of frames. Since only one node is considered in a tensor Θ_p storing the translation of the root node, this tensor does not need a Joint mode and its dimension reduces to $N = 5$. Identifying the degrees of freedom of the root node with the axes in 3D space, the size of the DOF mode in this case becomes 3.

Unfortunately, there is no intuitive way of visualizing tensors of order greater than 3. Figure 2.3 thus simplifies the data structure once again for visualization purposes by combining all technical modes into a single data mode and discarding one of the three natural modes. In this visualization of a motion tensor, each skeleton represents a single (large) column vector containing the rotational data of not only a single pose but a full motion sequence.

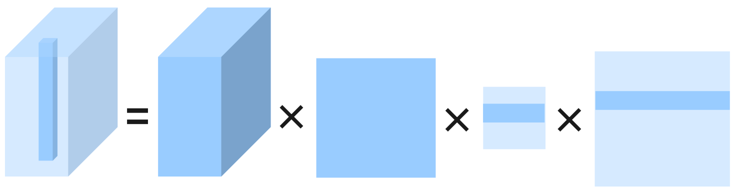


Figure 2.4: By multiplying the core tensor with all matrices corresponding to technical modes (one in this example) and a single row of all matrices corresponding to natural modes (two in this example), one of the original motions (visualized as the darker blue bar inside the data tensor on the left hand side) is reproduced.

2.2.3 Motion synthesis

As described in Section 2.2.1, a data tensor Θ can be decomposed into a core tensor Φ and related matrices $U_1, \ldots, U_N$. In this decomposition, each matrix U_k corresponds to a specific mode (e.g., the Actors mode), and each row in a matrix U_k corresponds to a specific entry of this mode (e.g., a certain actor). Instead of reproducing the complete data tensor Θ (by mode-multiplying Φ with all matrices U_k), this representation also allows us to directly reproduce a single *original* motion, i.e., a motion contained in the data tensor. This is done by first multiplying Φ with each matrix corresponding to a technical mode, and then multiplying the result with only *one row* of each matrix corresponding to a natural mode. Let t be the number of technical modes, n the number of natural modes, and let u_k^i be the i-th row of matrix U_k. Reproducing a motion m then can be expressed in the following way:

$$m = \Phi \times_1 U_1 \ldots \times_t U_t \times_{t+1} u_{t+1}^{i_1} \ldots \times_{t+n} u_{t+n}^{i_n}. \tag{2.2}$$

Figure 2.4 illustrates reproduction of an original motion for a tensor that has one technical and two natural modes.

While multiplying with a single row of each matrix $U_{t+1}, \ldots, U_{t+n}$ reproduces one of the original motions, it is also possible to synthesize a new motion m_{new} by using linear combinations of matrix rows. This can be expressed mathematically in this way:

$$m_{\text{new}}(\lambda_k) = \Phi \times_1 U_1 ... \times_t U_t \times_{t+1} \lambda_{t+1} U_{t+1} ... \times_{t+n} \lambda_{t+n} U_{t+n}, \tag{2.3}$$

with

$$\lambda_k U_k = \begin{pmatrix} \lambda_k^1 \dots \lambda_k^{d_k} \end{pmatrix} \begin{pmatrix} u_k^1 \\ \vdots \\ u_k^{d_k} \end{pmatrix} = \sum_{i=1}^{d_k} \lambda_k^i \; u_k^i =: x_k. \tag{2.4}$$

Motion classification

With the above model in hand, we are able to formulate an optimization problem based on the variables λ_k: Given an input motion m_{input}, our goal is to find values $\hat{\lambda}_k$, with $1 \leq k \leq n$, such that the synthesized motion $m_{\text{new}}(\hat{\lambda}_k)$ best matches this input motion:

$$\hat{\lambda}_k = \min_{\lambda_k} \text{dist}(m_{\text{input}}, m_{\text{new}}(\lambda_k)), 1 \leq k \leq n, \tag{2.5}$$

with $\text{dist}(m_1, m_2)$ being an arbitrary distance measure measuring the dissimilarity of two motions m_1 and m_2.

While in our reconstruction scenario we are dealing with a sparse input signal rather than with a complete input animation, the described framework can already be used for motion *classification*. Optimizing for λ_k then simply relates to classifying the input motion with respect to each natural mode. More precisely, given a motion tensor like the one described in Section 2.2.2, the $\hat{\lambda}_k$ tell us to which extent each actor, style, and repetition contributes to approximating the input motion. Given a weight distribution like this, arbitrary heuristics can be employed to derive a classification result. The most obvious strategy would be to assign the input motion to exactly that actor (style, repetition) that has the highest weight compared to all other actors (styles, repetitions). As there might, however, not always be a clear winner in each considered mode, more sophisticated strategies could for instance try to disambiguate the assignment by (iteratively) re-optimizing a subset of λ_k after excluding candidates with low weights from the optimization. Of course this still does not guarantee an unambiguous classification result.

While classifying a motion with respect to the Repetition mode is a rather unappealing task, optimizing it with respect to Style and Actors mode enables two different but equally interesting applications. Probably the more classical scenario would be to classify a motion with respect to its Style mode: What kind of motion are we dealing with? Optimizing for the Actors mode on the other hand in principle allows for actor identification: Whose motion is this? Note, however, that even Actors and Style Mode only represent different *shades* of the same basic motion class (e.g., walking motions), and that a single motion tensor is in general only capable of storing one such base class. Thus, a meaningful motion tensor-based classification requires some kind of semantic pre-classification: Which is the appropriate base class for this motion? Or in other words: Which tensor must we use? Following these considerations, Krüger [Krü11] has shown that tensor-based motion classification can be beneficial in a two-layered approach, seeking to refine a coarse pre-annotation, which in general is much easier to obtain automatically.

Classification is, however, beyond the scope of this work. Instead, we want to focus on motion *reconstruction* based on a sparse control signal. Before stating the optimization problem in this scenario, I first want to discuss the control signal itself.

2.3 Motion reconstruction

2.3.1 Control signal

The control signal in our reconstruction scenario is provided by up to four Xsens MTx inertial measurement units (IMU) [Xse11] that are attached to an actor's extremities. Each IMU consists of an accelerometer measuring its 3D linear acceleration, a gyroscope tracking changes in its 3D orientation, and a magnetic field sensor pointing towards the magnetic north pole. Fusing the information of all integrated sensors, these devices also provide (real-time) information about their 3D orientation.

In contrast to the system presented later in Chapter 3, we are here not only making use of the accelerometer readings, but also of the orientation information. More precisely, we use the orientation information to transform *local* accelerations a_L—accelerations given in the devices' local coordinate systems, as reported by the accelerometers—into *global* accelerations a_G, accelerations expressed in a fixed global (world) coordinate system. Note that up to this point both local and global acceleration represent an overlay of acceleration due to motion and acceleration due to gravity:

$$a_G = a_G^{\text{motion}} + a_G^{\text{gravity}} \tag{2.6}$$

Knowing the sensor orientation, however, enables us to subtract the acceleration due to gravity, leaving us with the pure acceleration due to motion a_G^{motion}. In the following, this acceleration due to motion (also referred to as *coordinate* acceleration) will serve as our control signal and be denoted as a_{sensor}:

$$a_{\text{sensor}} := a_G^{\text{motion}}. \tag{2.7}$$

In order to compare synthesized motions with this control signal, we simulate the coordinate accelerations of so-called *virtual sensors* that are attached to the virtual actor in the same way as the real sensors are attached to the real actor. Simulating the coordinate accelerations of virtual sensors is a very straightforward process: All we have to do is computing the second time derivatives of the virtual sensors' positional trajectories, which we can easily obtain using standard forward kinematics.

Denoting the simulated coordinate accelerations of virtual sensors by a_{sim} and the actual coordinate accelerations of the real sensors by a_{sensor}, the optimization problem becomes: Find the synthetic motion $\hat{m}_{\text{new}}$ such that the simulated accelerations $a_{\text{sim}}(\hat{m}_{\text{new}})$ derived from this motion best match the actual coordinate accelerations a_{sensor} of the real sensors:

$$\hat{m}_{\text{new}} = \min_{m_{\text{new}}} \operatorname{dist}(a_{\text{sensor}}, a_{\text{sim}}(m_{\text{new}})). \tag{2.8}$$

2.3.2 Distance measure

In order to solve this optimization problem, we first have to define a distance measure for comparing accelerations. While accelerations expressed in a global coordinate system are invariant under motion translation, they are not invariant under motion rotation: Say we have two motions, one of them being an exact copy of the other but being rotated about the vertical axis by 180 degrees. Acceleration vectors derived from both motions then will have the same magnitudes but different directions.[1] When comparing motions, both invariance under motion translation and rotation, however, are in general very desirable properties of a distance measure. To be more precise: We want our distance measure to be invariant under translation *at least* in the horizontal plane and to be invariant under rotation *only* about the vertical axis. Note that a simple comparison of acceleration magnitudes would not only be invariant under arbitrary rotation—a property we usually do not want—but would also neglect some relational information between different sensors that is established by the directions of the accelerations. Comparing poses on the acceleration level is inherently very difficult because of their ambiguity. The easiest way to illustrate this problem is to consider static poses without any measurable coordinate acceleration. In order to reduce this ambiguity, we want to calculate distances between entire motions instead of calculating distances between single frames, which also allows us to make use of the smoothness conditions contained in the motion data.

Our proposed distance measure is based on the one presented by Kovar et al. [KGP02]. Here, the authors defined the distance between two windows of frames as the minimal weighted sum of squared distances between corresponding points in two point clouds, given that an arbitrary rigid 2d transformation may be applied to one of the point clouds. Without loss of generality, let $p_{j,f}$ represent the position of joint j in frame f. The distance between two motions m and m' of arbitrary (but same) length is then defined

[1]To be more precise, the acceleration vectors will be mirrored on a plane perpendicular to the ground plane.

as:

$$\mathrm{dist}(m, m') = \min_{\theta, x_0, z_0} \left(\sum_f \sum_j \|p_{f,j} - T_{\theta,x_0,z_0} p'_{f,j}\|^2 \right), \tag{2.9}$$

where T_{θ,x_0,z_0} is a linear transformation that rotates a point p about the (vertical) y-axis by θ degrees and then translates it by x_0, z_0. This minimization problem has a closed-form solution [KGP02]. While in their framework points represent positional data—in general a downsampling of the virtual character's mesh deformed according to the underlying skeleton's pose—we will identify points with the coordinate accelerations of the sensors. As mentioned before, accelerations are inherently invariant under motion translation, so we do not even have to solve for the translational part of the transformation T and yield:

$$\mathrm{dist}(m, m') = \min_{\theta} \left(\sum_f \sum_s \|a_{f,s} - T_\theta a'_{f,s}\|^2 \right), \tag{2.10}$$

where $a_{f,s}$ represents the 3-dimensional vector of coordinate accelerations reported by sensor s in frame f. The same closed-form solution can be applied to this problem.

2.3.3 Optimization problem

As stated earlier, our optimization problem consists in finding the synthetic motion that best fits the sensor data. In the following, we assume that a semantic pre-classification of the motion to be reconstructed is given, and that the respective motion tensor contains 45 motions spanning a 5-dimensional Actors mode, a 3-dimensional Style mode and a 3-dimensional Repetition mode (cf. Section 2.2.2). Furthermore, let m_{asr} be the motion of actor a, style s and repetition r, and let λ_{asr} be a weight assigned to this motion, $1 \leq a \leq 5, 1 \leq s, r \leq 3$. Given that we want to approximate the original performance by a linear combination of the original motions, the general *linear*

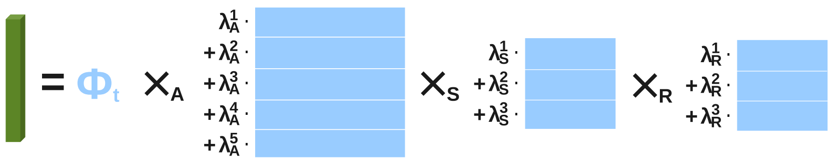


Figure 2.5: In the multi-linear framework, a new motion is synthesized by computing linear combinations of respective matrix rows. The resulting motion (visualized as green bar on the left side) then is a linear combination of original motions. In this example, the tensor has three natural modes (A,S,R) with dimensions 5, 3 and 3, respectively. Φ_t denotes the result of mode-multiplying the core tensor Φ with all matrices related to technical modes.

(as opposed to the multi-linear) approach would calculate m_{new} as

$$m_{\text{new}}(\lambda_{asr}) = \sum_a \sum_s \sum_r \lambda_{asr} \cdot m_{asr}, 1 \leq a \leq 5, 1 \leq s, r \leq 3, \tag{2.11}$$

thus comprising a 45-dimensional optimization problem. Please note that, due to the forward kinematics required for simulating virtual sensor readings, the minimization problem stated in equation 2.8 is non-linear (regardless of the chosen distance measure) and no closed-form solution exists.

By arranging the original motions in a multi-linear model and decomposing it using HOSVD, we yield a representation that effectively reduces the number of variables in the optimization to 11: Instead of having one parameter for each motion ($5 \cdot 3 \cdot 3 = 45$), we only have one for each actor, one for each style, and one for each repetition ($5+3+3 = 11$). Let $\lambda_A = (\lambda_A^1 \dots \lambda_A^5)$ be the weights assigned to the individual actors, $\lambda_S = (\lambda_S^1 \dots \lambda_S^3)$ the weights assigned to the individual styles, and $\lambda_R = (\lambda_R^1 \dots \lambda_R^3)$ the weights assigned to the individual repetitions. Then we can reformulate equation 2.3 as

$$m_{\text{new}}(\lambda_A, \lambda_S, \lambda_R) = \Phi_t \times_A \lambda_A U_A \times_S \lambda_S U_S \times_R \lambda_R U_R, \tag{2.12}$$

with $\Phi_t = \Phi \times_1 U_1 ... \times_t U_t$ being the product of the core tensor Φ and all matrices related to technical modes (Figure 2.5).

2.3.4 Synthesizing motions of arbitrary length

Motion reconstruction as described above is limited to motions of a specific "semantic" length (for locomotions in means of number of steps). For a variety of locomotions, we can, however, overcome this restriction with a slight modification of our method. The basic idea is to partition motion sequences into natural motion *units* and locally optimize these units using appropriately-sized tensors. For locomotions, a single step defines a natural motion unit, and "foot on ground" phases give natural transitions between these motion units. For more general motions, techniques used for motion graphs [KGP02, SO06, HG07, SH07, MP07] have to be employed.

The here described extension of the multi-linear model has originally been developed by Golla [Gol09]. The goal of his thesis, however, slightly differed from ours: While we aim at synthesizing plausible motions based on a sparse sampling of an actual motion performance, he sought to synthesize motions based on low-dimensional user specifications (or constraints), similar to the *Motion Doodles* interface described by Thorne et al. [TBv04]. More precisely, the input for his motion synthesis was a user-drawn (two-dimensional) curve, interpreted as a (smoothed) projection of the skeleton's hip trajectory onto the ground plane. As a consequence, he didn't have to deal with the problem of finding the *correct* partition of the control input: Either the choice was given to the user (thereby giving him control over certain properties of the resulting motion), or it was determined by a simple analysis of the respective motion units in the used tensors. We, however, have to find the correct partition based on our *control signal*.

As a more concrete example, say we want to synthesize a walking sequence containing different step lengths. For this we build two tensors: One storing single steps (of different step lengths) with the right foot, the other storing respective steps with the left foot. In order to make use of these tensors, we now have to partition the input signal into alternating left and right steps. In Golla's framework, the input signal might resemble the curve shown in Figure 2.6 (a). Assuming that the scale of this curve is fixed (e.g., defined as a mapping from image pixels to meters), the easiest way to define a par-

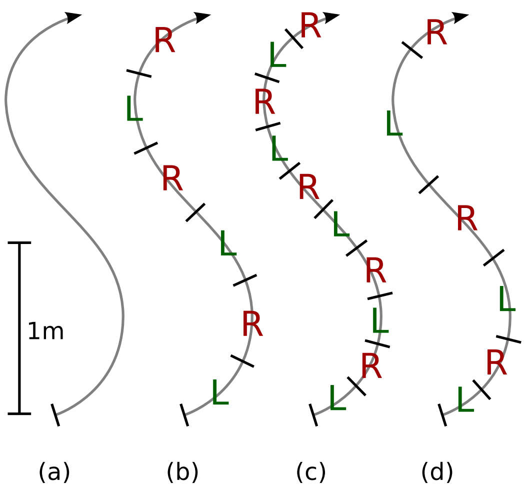


Figure 2.6: (a) Example of a user-specified input curve together with a predefined scale. (b)-(d) Different partitions of the input curve. *R* represents a step with the right foot, *L* a step with the left foot.

tition would be to simply project average step sizes (calculated over the two tensors, and without loss of generality expressed in meters) onto this curve. A possible result of this projection is visualized in Figure 2.6 (b). Note that even the choice with which foot the synthesis should start is completely free. Obviously, different partitions will produce different outcomes. Figure 2.6 (c) shows a partition into very small fragments, consequently leading to a synthesized motion consisting of small steps, and Figure 2.6 (d) shows a partition into fragments of varying lengths.

Once this partition is defined, several methods are possible to synthesize the full motion sequence. Golla basically distinguishes between a *local*, a *global*, and a *window-based* approach. In the local approach, each single step is optimized individually, and the final motion is created by appropriately aligning and concatenating (blending) the resulting synthesized steps. In the global approach, all steps are optimized simultaneously, trying to find the global motion sequence that best follows the control curve while satisfying certain transition constraints. The best results, however, were achieved by the window-based approach. The basic idea of this approach is to simultaneously optimize a window of several consecutive steps, and after each optimization shift this window forward by one step, thereby always taking advantage of the previous outcome. Details can be found in Golla's thesis [Gol09].

The same methods can be applied to our motion reconstruction scenario, given that we manage to find the correct partition of our input signal[2]. With accelerometers attached to the feet, automatic foot step detection for most locomotions, however, is a solvable task.

2.3.5 Motion database

For our approach we need a database of motions that is semantically pre-classified. Using the category names, such a semantic pre-classification is available in the commonly used CMU database [Car04]. However, the collection of motions contained in the CMU database is not sufficient for building a multi-linear model, since most motions are performed by one actor only without any stylistic variation.

For our purposes we found the data provided by the HDM05 motion capture database [MRC+07] more suitable. This database contains more than three hours of systematically recorded and well-documented motion capture data. All motion sequences were performed by five non-professional actors according to the guidelines fixed in a script. The script consists of five parts, where each part is subdivided into several scenes. In addition to the full takes, the HDM05 database also provides a set of short mocap clips that have been cut out of the takes and arranged into a set of roughly 100 motion classes. It is this set of cut-out motion clips that we used for all our experiments. Most of the represented classes contain 10 to 50 different realizations of the same type of motion, covering a broad spectrum of semantically meaningful variations. The resulting *motion class database* contains roughly 1,500 motion clips and 50 minutes of motion data.

[2]Please note that a partition of an acceleration-based control signal means a partition in time rather than in space.

2.4 A novel distance measure

For the numerical evaluation of a reconstruction result, the synthesized motion has to be compared with the original (ground truth) motion. At this, finding a distance measure matching the human perception of motion is a nontrivial task. A well-established approach is to compute a distance based on the average error of local joint orientations [CH05]. However, such methods may be inappropriate if global similarities of poses have to be computed since the hierarchical organization of a skeleton is completely neglected: An error at a parent joint also affects its children. Hence, a local error at a joint at the top of the skeleton hierarchy is likely to have a bigger impact on the global error than the same error at a lower level joint. As a consequence, the resulting globally visible error may be not properly reflected by a distance measure based on local joint orientations. Moreover, using the L_2 norm on Euler Angles directly suffers from the problem of finding an adequate distance measure for this representation of rotations.

In this section, we present a novel practical measure for comparing similarities of motions based on quantities represented in a global coordinate frame. Assuming a fixed skeleton topology, our goal is a universal measure that both matches the human perception and is simple enough to be implemented in time critical environments.

The basic idea is to frame-wise compare the cross product $\vec{c}\,_i^j$ formed by a joint j and two of its child joints a and b (Figure 2.7 left):

$$\vec{c}\,_i^j(a, b, f) = \vec{v}_{j\to a}(f) \times \vec{v}_{j\to b}(f) \tag{2.13}$$

Here, f denotes the frame of a motion i for which the cross product at a joint j is computed, $\vec{v}_{j\to a}$ the vector pointing from j to a, and $\vec{v}_{j\to b}$ the vector pointing to b, respectively. Please note that $\vec{c}\,_i^j$ can be interpreted geometrically as the normal of the triangle spanned by $\vec{v}_{j\to a}$ and $\vec{v}_{j\to b}$ weighted by two times the area of this triangle. Hence, $\vec{c}\,_i^j$ characterizes the orientation and the relative angle of two connected bones. In the following, the frame-based trajectory of $\vec{c}\,_i^j$ is denoted t_i^j.

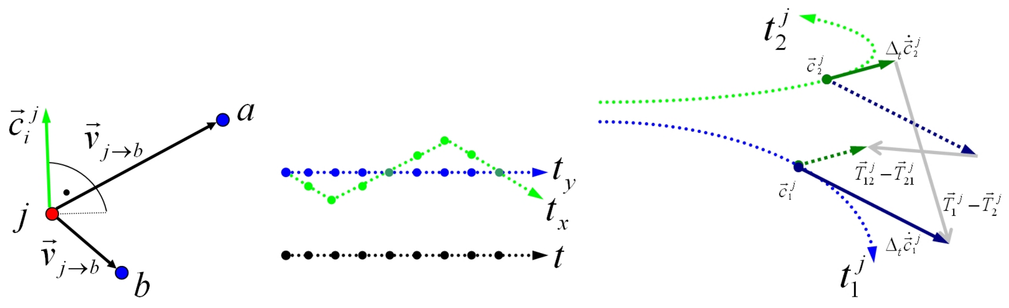


Figure 2.7: Left: Notation. Middle: Comparing two trajectories t_x and t_y to a reference t. Frames are indicated by dots. Note that t_y is just a shifted copy of t. Although the spatial distance is the same for t_x and t_y, t_x clearly differs from t which can be detected by comparing the local Taylor expansions of t_x and t. In this example a purely pose-based approach with frame-wise comparison fails. Right: Illustrating the meaning of T_1^j, T_2^j, T_{12}^j and T_{21}^j. In this example, $D_{1,2}^j = \|T_1^j - T_2^j\|$.

Supposing that two different motions of a joint j (and its child joints) are given, we use a local Taylor expansion of $\vec{c}_{\,i}^{\,j}$ to frame-wise describe the similarity between these two motions. For the two corresponding first-order Taylor expansions $\vec{T}_1^j$ and $\vec{T}_2^j$ around the frame f we get:

$$\vec{T}_1^j(f) = \vec{c}_{\,1}^{\,j}(a,b,f) + \Delta_t\,\dot{\vec{c}}_{\,1}^{\,j}(a,b,f) \tag{2.14}$$

and

$$\vec{T}_2^j(f) = \vec{c}_{\,2}^{\,j}(a,b,f) + \Delta_t\,\dot{\vec{c}}_{\,2}^{\,j}(a,b,f), \tag{2.15}$$

where Δ_t is a time-step and $\dot{\vec{c}}_{\,i}^{\,j}$ is the time derivative of $\vec{c}_{\,i}^{\,j}$. Let moreover $\vec{T}_{12}$ and $\vec{T}_{21}$ be two functions of mixed terms of $\vec{T}_1$ and $\vec{T}_2$:

$$\vec{T}_{12}^j(f) = \vec{c}_{\,1}^{\,j}(a,b,f) + \Delta_t\,\dot{\vec{c}}_{\,2}^{\,j}(a,b,f), \tag{2.16}$$

$$\vec{T}_{21}^j(f) = \vec{c}_{\,2}^{\,j}(a,b,f) + \Delta_t\,\dot{\vec{c}}_{\,1}^{\,j}(a,b,f). \tag{2.17}$$

If the two trajectories are traversed in a similar manner, $\vec{T}_1^j$, $\vec{T}_2^j$, $\vec{T}_{12}^j$ and $\vec{T}_{21}^j$ have to match. Consequently, differences indicate local errors (see also Figure 2.7, middle and right). Based on this observation, our local distance measure $D^j_{1,2}$ with respect to a frame f computes as:

$$D^j_{1,2}(a,b,f) = \max(\|\vec{T}^j_1 - \vec{T}^j_2\|, \|\vec{T}^j_{12} - \vec{T}^j_{21}\|), \tag{2.18}$$

which can be simplified to:

$$D^j_{1,2}(a,b,f) = C^j_{1,2}(a,b,f) + \dot{C}^j_{1,2}(a,b,f) \tag{2.19}$$

with

$$C^j_{1,2}(a,b,f) = \|\vec{c}\,^j_1 - \vec{c}\,^j_2\| \tag{2.20}$$

and

$$\dot{C}^j_{1,2}(a,b,f) = \Delta_t \, \|\dot{\vec{c}}\,^j_1 - \dot{\vec{c}}\,^j_2\|. \tag{2.21}$$

Setting the remaining free parameter Δ_t to

$$\Delta_t = \frac{\|\vec{v}_{j\to a}\|\|\vec{v}_{j\to b}\|}{\|\dot{\vec{c}}\,^j_1\| + \|\dot{\vec{c}}\,^j_2\|} \tag{2.22}$$

scales $\dot{C}^j_{1,2}$ to the range of $C^j_{1,2}$. Now that a similarity measure for a single joint j and two children a and b can be computed we finally generalize this measure to a distance measure $\mathrm{D_{pv}}$ for an arbitrary set of joints by summing over all frames f, all joints j and child joints a, b according to

$$\mathrm{D_{pv}} = \sqrt{\sum_{f=1}^{d_2} \sum_{j=1}^{d_3} D^j(f)}, \tag{2.23}$$

with

$$D^j(f) = \sum_{a=1}^{s_j} \sum_{b=1}^{s_j} (1 - \delta_{ab}) \left(D^j_{1,2}(a,b,f)\right)^2. \tag{2.24}$$

Please note that the error at a joint is implicitly weighted by the length of its bones. This is a desirable property, since longer bones are very likely to dominate the perception of a motion. Moreover, subtle errors like flipped joints are detected by the proposed method. However, although $\mathrm{D_{pv}}$ is invariant under translation, rotating motions yields different results. This is a

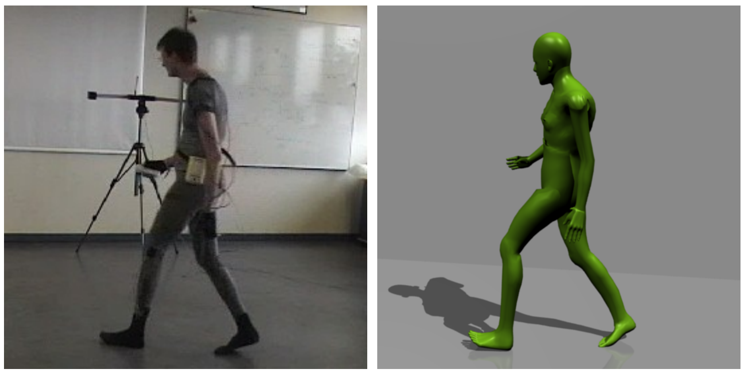


Figure 2.8: Pictures of a walking motion. The left picture is taken from the video, the right picture shows the corresponding pose of the reconstruction. The synthesis was driven by only four inertial sensors.

direct consequence of performing all computations with respect to a global coordinate frame.

2.5 Results

We evaluated the techniques described in the previous sections in two different ways. First, we used real sensor data obtained from four Xsens inertial sensors (cf. Section 2.3.1) attached to the hands and feet of an actor as control input for our motion reconstruction. As we do not have ground truth data in this case, the synthesized motion was compared with a video of the performance. In Figure 2.8, a single frame of the video is shown as a reference, for more results we refer to the supplemented video.

Second, in order to numerically evaluate the outcome of our multi-linear motion synthesis framework, we simulated sensor readings based on motion segments taken from the CMU [Car04] and HDM05 [MRC+07] motion databases and used these as input for our motion synthesis.

In both scenarios, the multi-linear model was built based on the HDM05 motion database as described in Section 2.3.5. Table 2.1 shows the average reconstruction errors as reported by the novel distance measure described in Section 2.4 and the commonly used L_2-distance on joint angles, using

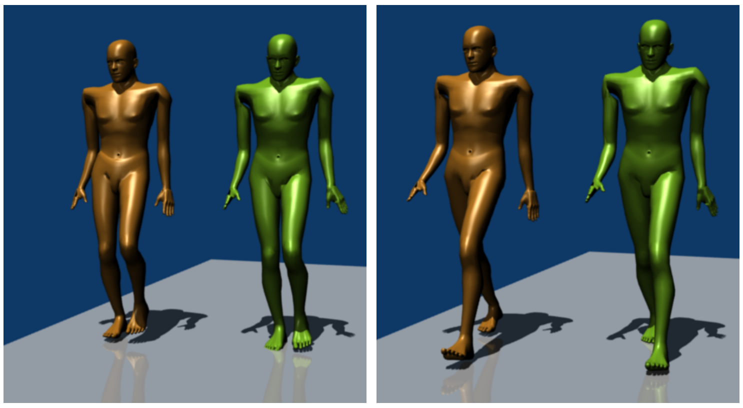


Figure 2.9: Comparison of two frames of an original walking motion (brown) and a reconstruction with our method (green). We only used the acceleration data of the left foot and the left hand to reconstruct these motions.

different sensor setups on two very different kinds of motion (walking and cartwheel). For visual comparison we refer again to the supplemented video. When comparing the numerical outcomes with actual renderings of the reconstructed motions, we see that the novel distance measure better identifies problematic cases. This finding is also supported by a series of experiments we performed with a different motion synthesis framework [TKZW08]. In these tests, the joint angle-based measure failed to penalize artifacts like directional flips or jitter. Our proposed distance measure, however, clearly identified these artifacts that drastically affect the human perception of motions, while assigning small distances to perceptually similar motions.

Finally, Figure 2.10 shows a result of the extension discussed in Section 2.3.4 using the window-based reconstruction scheme.

2.6 Conclusion

The results of this work can be seen as an early proof-of-concept that—under certain assumptions—using a pre-classified motion capture database high-dimensional full-body motions can be reconstructed on the basis of extremely sparse control inputs. In several applications there will be a priori

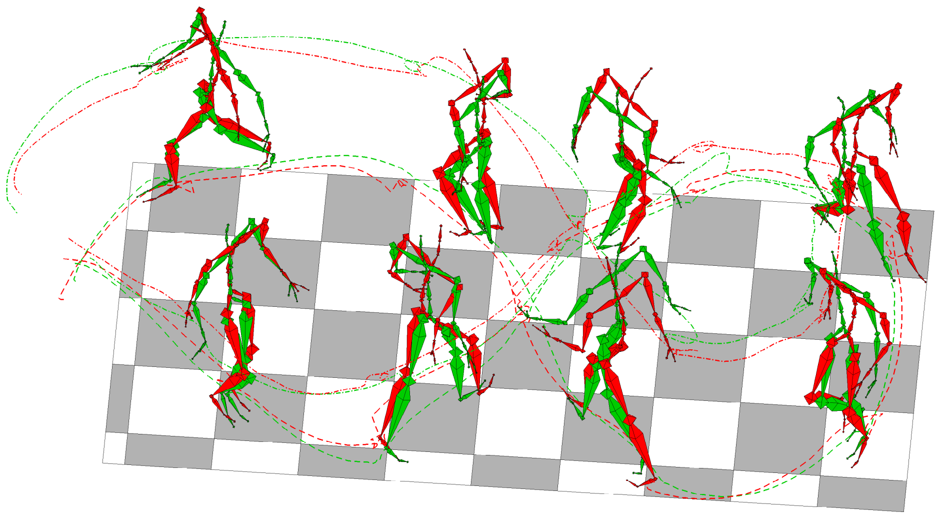


Figure 2.10: Result of reconstructing a motion consisting of 21 steps with the window-based approach. The original motion (green) with our reconstruction (red) are shown in form of some sample frames and the trajectories of a virtual marker on the left foot and the right hand.

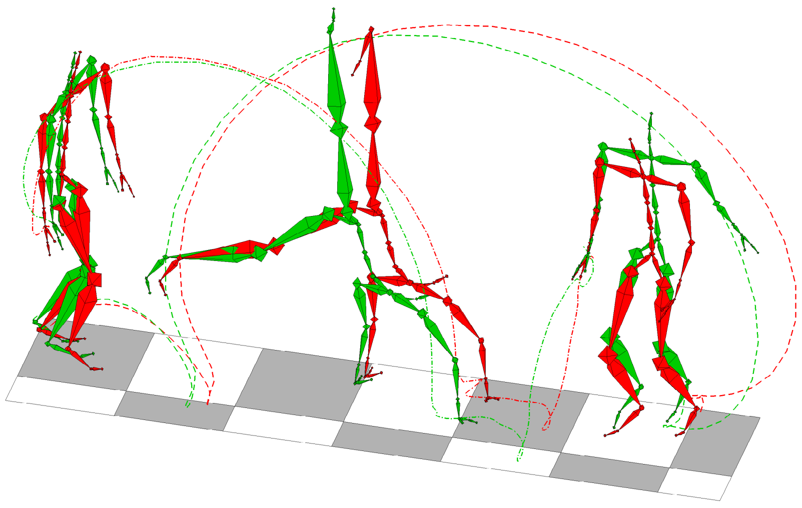


Figure 2.11: Reconstruction of a cartwheel sequence. The original motion (green) with our reconstruction (red) are shown in form of some sample frames and the trajectories of a virtual marker on the left foot and the right hand.

	Walking		**Cartwheel**	
Regarded joints	D_{pv}	D_E	D_{pv}	D_E
footL	15.23	12.16	21.22	15.32
footR	17.63	12.02	40.34	15.83
handL	14.83	11.44	25.35	15.57
handR	14.75	10.13	50.42	17.63
footL, footR	15.32	8.18	26.50	15.11
footL, handL	23.41	5.55	24.81	15.37
footL, handR	14.47	10.18	41.49	16.19
footL, footR, handL	17.26	14.55	25.22	15.39
footL, handL, handR	14.50	10.64	23.45	16.59
footL, footR, handL, handR	14.82	10.15	29.69	15.29
footL, footR, handL, handR, shoulderR	14.98	10.45	29.62	15.29
footL, footR, handL, handR, kneeL, kneeR	14.54	10.67	25.27	15.26

Table 2.1: Average reconstruction errors for sample motions using our *Multi-linear Motion Model* (MMM). Errors are given using the novel distance measure defined in Section 2.4 summing over all joints (denoted by D_{pv}), and the commonly used L_2-distance calculated over the joint angles (denoted by D_E).

knowledge about the input motion that gives rise to the low-dimensional control signal, for example in computer games or in sport training and rehabilitation. For sport training one could for example build a multi-linear motion model with motions of professional athletes as well as beginners (and arbitrary intermediate levels). With such a model in hand, we could not only numerically rate a new user's skill level, but also reconstruct and visualize his performance just on the basis of a low-dimensional control signal. For reconstructing motions for which there is no a priori knowledge available, our approach could be combined with motion classification techniques such as *motion templates* [MR06]. This idea has also already been investigated by Krüger [Krü11].

The requirements of multi-linear models, however, render them pretty much useless in a realistic (real-time) motion capture scenario. Here, we do in general not have any a-priori knowledge about the input motion. Moreover, the model is inherently too inflexible for reconstructing arbitrary motions: As mentioned earlier, a single tensor can only represent a single class of very related motions (e.g., walking motions). Such a class of motions is even further restricted by the fact that all motions to be stored in the same tensor have to be in temporal correspondence (achieved by dynamic time warping in a pre-processing step [KTW07, KTMW08]), which basically forbids any variation in the length of these motions. To be more precise: Original motions may differ in speed and thus in duration (as the dynamic time warping will take care of this), but not in "semantic length", e.g., their number of foot steps. This is why generously speaking of a tensor of *walking motions* actually conceals some serious restrictions (related to basically every approach that seeks to interpolate entire motions rather than single poses): Motions in such a tensor must at least have the same number of steps and start with the same foot. Unfortunately and obviously worse, this also holds for the motion to be reconstructed with this tensor, which is why we have to rely on a pre-classification.

2.6.1 Optimization problem revisited

In Section 2.3.3, we briefly discussed the intuitive interpolation interface and the inherent variable reduction provided by the multi-linear framework. It has to be stated, however, that the same interface can be transferred to the linear model by defining

$$\lambda_{asr} := \lambda_A^a \cdot \lambda_S^s \cdot \lambda_R^r \tag{2.25}$$

(using the same notation as in Section 2.3.3), and solving for the reduced parameter set $(\lambda_A, \lambda_S, \lambda_R)$ only. In fact, as far as motion synthesis is concerned, the multi-linear model only restricts the space of possible linear combinations (and hence synthesizable motions) compared to the full parameter optimization. Moreover, the linear model together with the parameter reduction defined in equation 2.25 does not only provide the same intuitive interpolation interface, it even requires less multiplications to produce the exact same outcome (even without taking the HOSVD into account). So why do we need multi-linear models?

Indeed, this question is quite justified. There is, however, one discipline where the multi-linear framework outperforms the linear one: compression. As was shown by Krüger et al. [KTW07, KTMW08], the conservation of semantic information enables higher compression rates. While the pure data reduction seems to be not that important in our case, the more appealing thing is the following: Opposed to the linear model, truncating the components of the multi-linear decomposition even allows us to further reduce the number of parameters needed for the synthesis of a new motion, and thus the number of variables we have to solve for. The idea is visualized in Figure 2.12: Instead of solving for $\{\lambda^a_A, \lambda^s_S, \lambda^r_R\} =: \Lambda$ and explicitly computing linear combinations of matrix rows, we directly solve for $\{x^{\check{a}}_A, x^{\check{s}}_S, x^{\check{r}}_R\} =: X$. Note that since all mode-matrices are orthogonal and hence square, the number of variables in Λ equals the number of variables X in the *uncompressed* case, but is lower when matrices are truncated. The relation between the different variable sets in the general case has already been mathematically expressed in equation 2.4.

In the reconstruction scenario (as opposed to the classification task), we are usually not interested in the actual values of the parameters that give rise to the synthesized motion. Thus, we usually do not care that the new variable set X is less descriptive and less intuitive than Λ. If we are, however, interested in the more descriptive solution, it has to be stated that while X-values can be in principle easily transformed into Λ-values, this requires solving an underdetermined system of linear equations, which in general has infinitely many solutions.

In conclusion, we must say that despite building a unified and simple framework with an intuitive interface for data (or dimension) reduction, (fine-grained) motion classification, interpolation-based motion synthesis, and motion reconstruction (with aforementioned restrictions), multi-linear motion models are inherently not flexible enough to fulfill general motion capture requirements. This is why we finally decided not to further pursue the multi-linear approach for motion reconstruction.

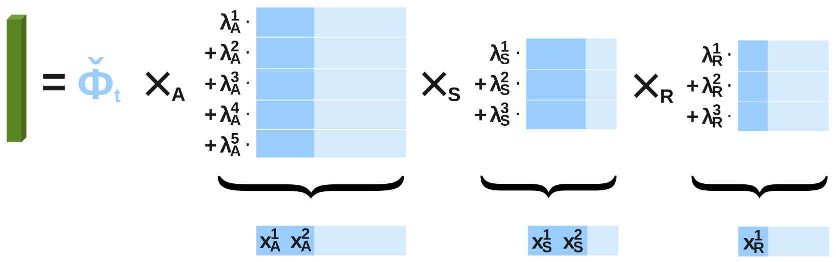


Figure 2.12: By truncating the core tensor Φ and respective mode-matrices, we can speed up the optimization not only by saving multiplications due to smaller-sized matrix factors, but also by further reducing the number of variables when solving for $x_A^{\check{a}}, x_S^{\check{s}}, x_R^{\check{r}}$ instead of $\lambda_A^a, \lambda_S^s, \lambda_R^r$. The resulting motion (visualized as green bar on the left side) then is an *approximation* of a linear combination of original motions. In this example, the tensor has three natural modes (A, S, R) with dimensions 5, 3, and 3, which were truncated to 2, 2, and 1 dimensions, respectively. $\check{\Phi}_t$ denotes the result of mode-multiplying the *truncated* core tensor Φ with all *truncated* matrices related to technical modes.

2.7 Outlook

A different approach to performance animation was taken by Chai and Hodgins [CH05]. In their seminal work, Chai and Hodgins present a complete data-driven real-time animation system for synthesizing motions based on low-dimensional control input obtained by tracking a small set of retro-reflective markers attached to an actor's body. While the general framework makes the system flexible and powerful, the type of control signal imposes various constraints on the recording environment. As has already been noted by the authors, their method should not only be suitable for sparse sets of optical markers, but also for other low-dimensional control signals, e.g., ones provided by inertial sensors. Unfortunately, as we have already pointed out [TKZW08], simply replacing the position-based control signal by accelerations does not yield any satisfying results. Furthermore, opposed to our naive assumption at that time, it is not possible to reliably estimate new control points in position space by using the position information from the previously reconstructed pose and double integrating the acceleration data for one time-step. Even though the position estimate bases on a very short time span between two frames, the problem of velocity and position drifts due to data noise, imperfect pose reconstruction, and inaccurate estimation of sensor orientations is not negligible. Nevertheless, we considered the general framework presented by Chai and Hodgins as very appealing for our application. The following work will describe the challenges introduced by our intention to replace a position-based control signal by an acceleration-based one, and how we successfully dealt with them.

3 Motion reconstruction based on sparse accelerometer data

3.1 Introduction

The increasing availability and demand of high-quality motion capture (mocap) data has become a driving force for the development of data-driven methods in computer animation. One major strand of research deals with the generation of plausible and visually appealing motion sequences by suitably modifying and combining already existing mocap material. In the synthesis step, task- and application-specific constraints are to be considered. Such constraints may be specified by textual descriptions [AFO03] or by low-dimensional control signals as supplied by recent game consoles [Nin11, Son11].

Chai and Hodgins [CH05] describe a data-driven scenario where a sparse set of video-based control signals is used for creating believable character animations. In their seminal work, the authors present a complete online animation system, where control data obtained by tracking 6–9 retro-reflective markers is used to construct a local model of the user's motion from a prerecorded set of mocap data. From this model, a high-dimensional, naturally-looking animation is synthesized that approximates the controller-specified constraints. One drawback of this approach is that the usage of retro-reflective markers and calibrated cameras to generate the control input imposes various constraints on the recording environment (e.g., illumination, volume, indoor). Furthermore, such systems are inconvenient with respect to setup and calibration, while being comparatively costly. Slyper and Hodgins [SH08b] describe a first system for retrieving upper-body mocap sequences using a small number of low-cost accelerometers as control input only.

The work described here, originally presented in the article *Motion Reconstruction Using Sparse Accelerometer Data* [TZK+11], builds upon, combines, and extends the approaches by Hodgins et al. discussed above. We introduce a complete data-driven system for generating plausible full-body motion streams; see Figure 3.1 for an overview. As control input, we employ four 3D accelerometers that are fixed next to the wrists and ankles of a user's body in a predefined way. Furthermore, motion priors are given in form of a knowledge base consisting of a large number of motion sequences, which have been recorded using marker-based mocap systems. In our approach, the knowledge base may be heterogeneous, containing motions of different types and styles performed by various actors. In a preprocessing step, we derive suitably simulated acceleration readings from the stored motion sequences, making them comparable with the sensor input. Furthermore, for later usage, the knowledge base is indexed using a kd-tree structure. At runtime, the sensor input is processed, frame-wise triggering a nearest-neighbor (NN) search. For the current input frame, the retrieved poses are used to update a data structure that points to entire motion subsequences in the knowledge base best explaining the controller input over the past frames. This data structure, which is an online-capable extension of the *lazy neighborhood graph* introduced by Krüger et al. [KTWZ10], is then used in the reconstruc-

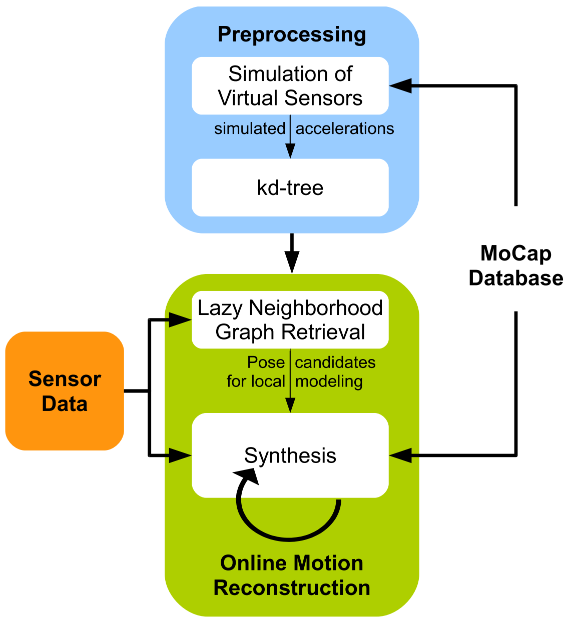

Preprocessing
Simulation of Virtual Sensors
simulated accelerations
kd-tree
MoCap Database
Sensor Data
Lazy Neighborhood Graph Retrieval
Pose candidates for local modeling
Synthesis
Online Motion Reconstruction

Figure 3.1: Overview of the animation system.

tion step to compute the current frame of the outputted animation. For the reconstruction, we introduce an optimization procedure that depends not only on the retrieved information, but also considers the temporal context as well as the forward-integrated control signals.

3.1.1 Main contributions

First, we introduce a novel online framework for reconstructing full-body motion streams based on very sparse accelerometer input. Slyper and Hodgins [SH08b] aim to reconstruct the upper-body motion using five accelerometers, whereas our method allows for full-body motion reconstruction with only four sensors that are fixed next to the wrists and ankles. The suitability of the number and placement of sensors is backed up by our experiments. In contrast to all existing methods for motion reconstruction from sparse accelerometer data, our method is the first that allows for synthesizing new motions from a given knowledge base. Our approach can flexibly deal with temporal and spatial variations—as opposed to previous methods that reconstruct a motion by choosing a prerecorded clip from a database [SH08b].

Furthermore, the database used in Slyper and Hodgins [SH08b] is small and contains only a restricted number of different motion clips. In contrast, our knowledge base is orders of magnitude larger and contains many different motions performed by different individuals in various styles. Because of the increased complexity and ambiguity, more sophisticated approaches regarding retrieval and motion synthesis are required. In contrast to Slyper and Hodgins, our reconstruction is frame-accurate where an optimal pose hypothesis is computed for each frame of the control input. As second contribution, we present an online variant of the lazy neighborhood graph previously introduced by Krüger et al. [KTWZ10]. Opposed to the original graph, our novel variant allows for a very efficient analysis of continuous motion streams having a speedup of more than one order of magnitude for the application presented in this work. Based on our novel approach, NN-based motion retrieval does not constitute a computational bottleneck any longer.

As a third main contribution, we elaborate on a novel prior model that minimizes reconstruction ambiguities for data-driven motion synthesis even in challenging cases and simultaneously accounts for temporal and spatial variations on the controller side and knowledge base side. Our proposed kernel regression-based pose prior is quite different from other approaches previously presented in the context of position-based reconstruction and synthesis [CH05, SL06, SKL07, LWB+10]. The main advantage of our approach lies in its generality: our algorithm even produces reasonable results if the poses retrieved by the NN-search belong to various logically distinct motions. This property is essential to our application as similar accelerometer (control input) readings may be associated with very different motion classes and thus different hypotheses. Novel motion and smoothness priors used in our work effectively guide the synthesis process towards a relatively smooth and plausible reconstruction. Please note that in previous work in the field of motion synthesis [CH05] ad hoc temporal priors that enforce smoothness by minimizing accelerations have been applied. In contrast, our approach is fully data-driven and adapts to variations that occur in particular when the directionality of a motion changes (e.g., at turning points of a locomotion).

3.2 Related work

There are many ways for capturing and recording human motions including mechanical, magnetic, optical, and inertial devices. Each motion capturing (mocap) technology has its own strengths and weaknesses with regard to accuracy, expressiveness, and operating expenses, see Maiocchi [Mai96], Moeslund [MHK06], or Wikipedia [Wik11] for an overview. For example, optical marker-based mocap systems typically provide high-quality motion data such as positional information as joint coordinates or rotational information as joint angles [Pha11, Vic11, Gia11]. However, requiring an array of calibrated high-resolution cameras as well as special garment equipment, such systems are not only cost intensive but also impose limiting constraints on the actor and the recording environment.

In recent years, low-cost inertial sensors, which can be easily attached to an actor's body or even fit in a shoe, have become popular in computer game and sports applications [SH08b, Nik11, Adi11, Nin11]. However, the inertial information obtained from such sensors, such as joint accelerations, angular velocities, or limb orientations, is often of low expressive power and affected by noise. To avoid drifts that often occur when using inertial sensors, various approaches based on sensor fusion have been proposed to improve and stabilize motion tracking. For example, in the Xsens system rotational drifts are avoided by incorporating magnetic field sensors [SRV10]. Vlasic et al. [VAV$^+$07] combine inertial sensors with ultrasonic distance sensors to compensate for relative positional drifts.

Another strategy for improving motion capturing is to include prior knowledge on kinematics or dynamics of the motion to be expected. Here, data-driven methods, as also employed in our work, have turned out to be a powerful approach generating such additional constraints. The (real-time) control of virtual characters using mocap data—also known as computer puppetry [SLSG01]—is one key challenge in the field of computer animation. Besides the use of high-dimensional optical systems, various controller-based systems have been described that allow for generating and reconstructing visually appealing motion sequences from low-dimensional sensor input. In its easiest form, as is also often done in commercial computer game applications, controller data may trigger certain actions. Low-dimensional sensor input is often used for specifying free parameters in model-based computer animation; see, for example, Badler et al [BHG93], Cooper et al. [CHP07], Dontcheva et al. [DYP03], and Oore et al. [OTH02]. Shiratori and Hodgins [SH08a] use inertial-based control data to specify a small number of free parameters in physically-based character animation. When high-dimensional data has to be generated using only low-dimensional control data, especially data-driven approaches show promising results. For example, Feng et al. [FKY08] use sparse control points and an example-based model to deform complex geometries. Another approach is to use the low-dimensional sensor input to retrieve suitable motion sequences from a database containing high-dimensional mocap sequences. For example, Slyper and Hodgins [SH08b] describe a system, where a small number of low-cost accelerometers are used to identify and playback prerecorded human upper-body motions. An extension to this work is sketched by Kelly et al. [KCHO10], where a motion database consisting solely of tennis motions is used to reconstruct the actions of a tennis player wearing six accelerometers. Such reuse of prerecorded human mocap data requires efficient retrieval of similar motions from databases [KPZ$^+$04, MRC05], as well as a good understanding of how motions have to be parametrized in order to yield smooth transitions between several retrieved motion clips [KG03]. Liu et al. [LWC$^+$11] recently presented a framework that is conceptually very similar to the performance animation system presented by Chai and Hodgins [CH05], outlined earlier. However, they formulate the motion reconstruction problem in a maximum a posteriori framework, utilizing a series of online local *dynamic* motion models, and the control input is provided by a small set of both inertial and ultrasonic sensors.

Our approach is also inspired by the animation system presented by Chai and Hodgins. Opposed to using optical markers and calibrated cameras, we use a sparse set of four 3D accelerometers to generate the control data. Also, we do not use a static graph structure quadratic in memory size, but instead employ a memory-efficient data structure that much better scales to larger datasets. Finally, opposed to the system presented by Slyper and

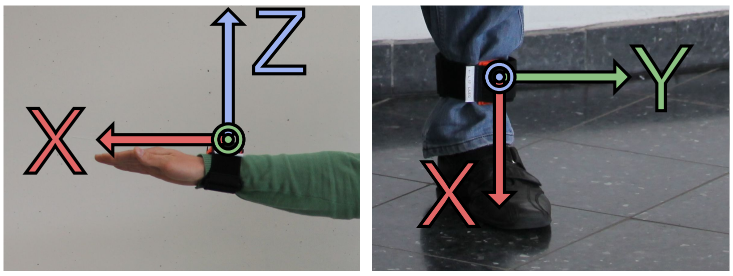


Figure 3.2: The four accelerometers are attached to the lower arms and lower legs using simple straps.

Hodgins [SH08b], our approach allows for handling moderate temporal and other variations that are not reflected well by the given database motions.

3.3 Control input and knowledge base

In this work, the control input is provided by the same devices that we used in the multi-linear framework presented earlier. Specifically, four Xsens MTx devices [Xse11] are attached to the lower arms and lower legs of an actor, next to the wrists and ankles, respectively. Despite the fact that these kind of sensors provide a lot of different information, including rate of turn, magnetic field and orientation (see Section 2.3.1), we here only use the calibrated readings of the devices' accelerometers. Thus, our findings can be applied to much smaller (and less expensive) sensors using accelerometers only. These calibrated readings are given in the unit $\mathrm{m/s^2}$ and are expressed with respect to the sensors' local coordinate systems.

In order to make the data originating from these sensors comparable with data originating from the knowledge base, the sensors have to be carefully aligned with the respective limbs they are fixed to. Figure 3.3 shows the ideal placement of the sensors, where the X-axis of the sensors coincides with the direction of the underlying bone, pointing away from the body's center. In case of the arms, we align the sensors such that their Z-axes are pointing upwards when the arms are stretched out and the palms are pointing downwards. The sensors at the legs are placed in a way that the Z-axes are pointing forward while being orthogonal to their related X-axis as well as to

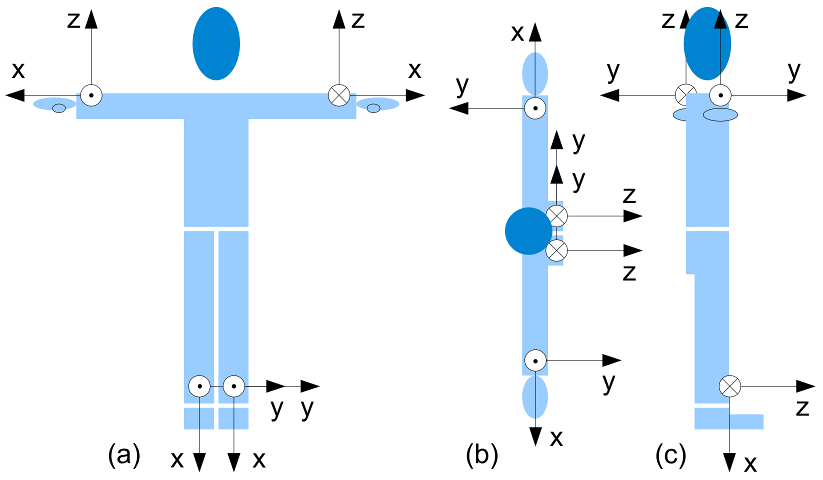


Figure 3.3: Schematic representation of the ideal sensor setup.

the rotation axis of the corresponding knee. Finally, the Y-axes are chosen to form right-handed coordinate systems with respect to the X- and Z-axes.

Obviously, the result of simply attaching the sensors with straps to the respective limbs (see Figure 3.2) will always diverge to some extent from the ideal placement shown in Figure 3.3. We found, however, that by fixing the devices with reasonable care, two of the three rotational degrees of freedom of all sensors are already very well-defined. That is, using reasonably large and tight straps, the four sensors do (due to their box shape) barely have any play regarding rotations about their Y- and Z-axes; see Figure 3.4 (a) and (b). As a result, the X-axis (the axis that coincides with the direction of the underlying bone) is very well-defined. Although the error caused by a deviation of sensor rotations about this axis is also often negligible, we performed the following simple calibration step in all our experiments to refine the orientations of the wrist sensors: We simply asked the actor to hold a t-pose (with palms facing to the ground) for a few seconds, and then compared the accelerometer readings against the known direction of gravity. In an ideal setup, the gravitational component would now entirely project onto the z-axes of the wrist sensors. Assuming that the X-axes are optimal, we can easily determine the rotational offset (with respect to the X-axis) between the actual and the ideal setup. Rather than physically moving the

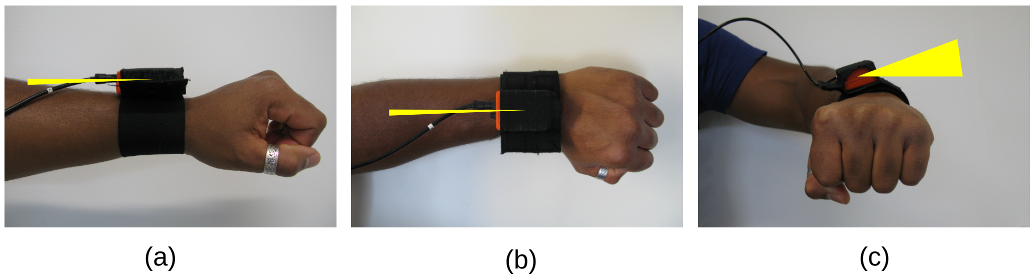


Figure 3.4: By carefully attaching the sensors to the lower arms using reasonably tight straps, we get near-ideal alignments with respect to rotation about the Y-axis (a) and Z-axis (b). We use the accelerometer readings in a static t-pose to calculate the rotational offset with respect to the X-axis (c) between the actual and ideal sensor placement.

sensors according to this offset (which would be very inconvenient), we used it to numerically correct all future measurements that were taken using the same setup. Regarding the sensors attached to the lower legs, a similar procedure could be used. Unfortunately, however, we can not use the t-pose to estimate the offset around the X-axis, as in this case the X-axis is parallel to the direction of gravity. While different calibration poses are possible and even more complex calibration procedures may be applied, for instance the one proposed by Slyper and Hodgins [SH08b], carefully fixing the sensors and only refining the orientations of the wrist sensors turned out to suffice in the context of our application.

In the following, we assume that our knowledge base consists of a sequence of poses indexed by the set $[1 : N] := \{1, \ldots, N\}$, with N denoting the total number of frames. Furthermore, we assume that each pose is given in joint angle representation denoted by $\vec{q}_n$, $n \in [1 : N]$. To obtain joint positions of a pose, forward kinematics need to be applied based on a given skeleton model, which contains information about the topology, the actor's bone lengths, as well as the degrees of freedom of each joint. In the following, we assume that all skeletons underlying the data of our knowledge base have the same topology. One key mechanism in our approach is the identification of suitable high-dimensional joint angle data by using low-dimensional accelerometer readings as query.

In this cross-modal retrieval scenario, we need to compare two different motion data representations of different dimensionalities. To bridge this gap, we simulate accelerometer readings for all motions in the knowledge base by computing the accelerations of *virtual sensors* that are placed on the limbs of the virtual actors in the same way as the real sensors are placed on the limbs of the real actors. After calculating the positions of these virtual sensors using forward kinematics, we compute their second time derivatives and obtain their accelerations relative to the global frame. Then, we simply add the acceleration component corresponding to gravity—which is inherently measured by each accelerometer—and transform the resulting quantity to the local coordinate systems of the virtual sensors. Helten et al. [HMT+11] present a systematic analysis of various feature representations (including local accelerations) in the context of a cross-modal retrieval scenario, where inertial-based query motions are used to retrieve high-quality optical mocap.

In addition to simulated sensor data, we pre-compute quantities that we later use in the synthesis step of our method, including the positions $\vec{x}_n$, velocities $\vec{v}_n$, and accelerations $\vec{a}_n$ of all joints. For normalization purposes, these quantities are given in the root coordinate system. All derivatives are approximated using a five-point stencil that has an approximation error of order $O(h^4)$. The respective formulas are

$$\vec{v}(t) = \frac{-\vec{x}(t+2h) + 8\vec{x}(t+h) - 8\vec{x}(t-h) + \vec{x}(t-2h)}{12h} \tag{3.1}$$

for the first derivative, and

$$\vec{a}(t) = \frac{-\vec{v}(t+2h) + 16\vec{v}(t+h) - 30\vec{v}(t) + 16\vec{v}(t-h) - \vec{v}(t-2h)}{12h^2} \tag{3.2}$$

for the second derivative.

Note that instead of using the original skeletons, forward kinematics for all motions (as well as synthesis) is performed on a *standard skeleton*, whose bone lengths are averaged across all skeletons represented in the knowledge base. We will, however, also present an analysis of the effect of varying actor sizes. For all our tests, we neglected the skeleton's foot and hand joints, resulting

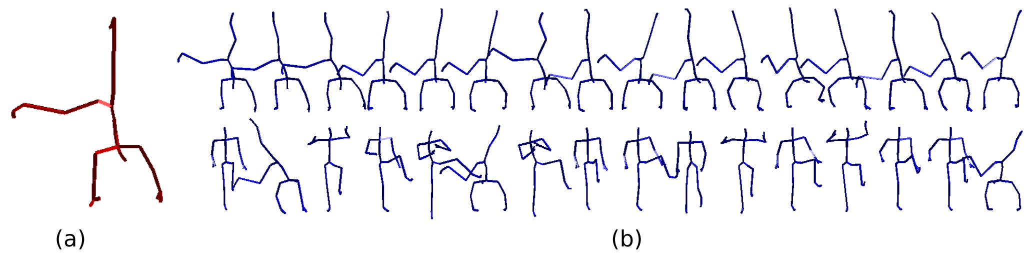


Figure 3.5: (a) The query frame, taken from a high dynamic motion (cartwheel). Note that the head is pointing down. (b) Top row: 16 nearest neighbors retrieved based on the *positions* of the wrist and ankle joints of the query frame. Bottom row: 16 nearest neighbors retrieved using the *accelerations* of the wrist and ankle joints of the query frame.

in a representation with 21 joints and a total of 43 rotational degrees of freedom.

The simulated sensor accelerations are denoted by $\vec{\alpha}_n$ and indexed using a kd-tree of dimension $4 \cdot 3 = 12$. At those low dimensions, Andoni and Indyk [AI08] state that kd-trees are well suited for fast nearest-neighbor searches. In our case, such fast nearest-neighbor searches are used to identify all poses in the knowledge base that are most similar to a given sensor reading; see Section 3.4.

3.4 Fast similarity search using acceleration data

3.4.1 Comparing accelerations

In our scenario, controller input is compared against prerecorded mocap sequences on the basis of 3D accelerations. A comparison of motion frames on the acceleration level is, however, much less descriptive than, for example, on the joint angle or 3D positional level as used by Chai and Hodgins [CH05]. Thus, this comparison may result in a large number of false positives, in particular when using a frame-wise retrieval procedure; see Figure 3.5.

In the multi-linear approach discussed earlier we were comparing global *co-*

ordinate accelerations of sensors (accelerations with respect to a fixed world coordinate system). In contrast to that, we are now focusing on accelerations as they are reported by the accelerometers. These accelerations, in the following referred to as *local* accelerations, are given in the local coordinate systems of the sensors and represent an overlay of the sensors' coordinate acceleration and acceleration due to gravity. Undoubtedly, global coordinate accelerations are more intuitive than *local* accelerations. The easier interpretability, however, comes at several costs: First of all, in order to obtain global coordinate accelerations from our sensors, we have to make use of magnetic field sensors and gyroscopes in addition to the accelerometers, which in turn forbids the use of very small and inexpensive devices. Second, by only using the global coordinate accelerations (and not *explicitly* making use of the integrated orientation information) we might even lose valuable information. To illustrate that, consider a sensor at rest: While the inherent measurement of the gravitational component present in local accelerations allows us to determine its orientation up to a single degree of freedom (the orientation about the vertical axis), its coordinate acceleration is zero in each direction and its orientation thus indistinguishable from every other orientation. On the other hand it must not be denied that separating coordinate acceleration from acceleration due to gravity may also disambiguate measurements.

Slyper and Hodgins [SH08b] address this inherently ill-posed problem by exploiting the temporal coherence of motions and querying fixed-length sequences of accelerometer readings. Temporal variations (e.g., motions performed at different speeds), however, are not handled in their approach. In our approach, we incorporate temporal coherence by using a data structure referred to as *Online Lazy Neighborhood Graph* (OLNG), which is an extension to the *Lazy Neighborhood Graph* (LNG) introduced by Krüger et al. [KTWZ10].

3.4.2 Lazy Neighborhood Graph (LNG)

As we have seen, it is almost impossible to find a reliable mapping from the low-dimensional space of accelerations (when observed at a single point in time) to the high-dimensional space of human poses. By comparing *sequences* of accelerations, similar to Slyper and Hodgins [SH08b], we can, however, heavily reduce the existing ambiguities. We thus now aim to identify sequences of accelerations in the knowledge base that match the control signal.

Basically, the LNG described by Krüger et al. [KTWZ10] can be directly applied to this task. The basic idea of the LNG can be summarized as follows: Given that every frame in the knowledge base has a unique index (with consecutive frames having consecutive indices), instead of searching for a motion sequence that *globally* matches the input signal (or a subsequence thereof), we can search for a sequence of consecutive, ascending indices whose corresponding frames *locally* match the input signal. More specifically, in our scenario we would proceed as follows: Say that our control input consists of accelerations $\vec{\alpha}^1, \vec{\alpha}^2, \ldots, \vec{\alpha}^M$, where M denotes the length of the compared acceleration sequences. First, for each sample $\vec{\alpha}^t$, with $1 \leq t \leq M$, we identify the K (locally) closest samples in the knowledge base using a fixed radius k-nearest-neighbors search; see Figure 3.6 (a). Second, we build a graph by treating each of the $M \cdot K$ samples as a node n_k^t, with $1 \leq t \leq M$ and $1 \leq k \leq K$, identifying it by its unique database index $i(n_k^t)$, and inserting edges between nodes that form *valid continuations*. While pointing out that different definitions of valid continuations are possible, Krüger et al. define them formally as follows (notation slightly adjusted): Let $i_1 := i(n_{k_1}^{t_1})$ and $i_2 := i(n_{k_2}^{t_2})$ be two indices representing nodes in this graph (and poses in the knowledge base). Then i_2 builds a valid continuation of i_1, if one of the following three conditions applies:

- $i_1 + 1 = i_2$ and $t_1 + 1 = t_2$,
- $i_1 + 1 = i_2$ and $t_1 = t_2$,
- $i_1 = i_2$ and $t_1 + 1 = t_2$.

Each case leads to the insertion of a directed edge from i_1 to i_2. Figure 3.6 (b) gives a more illustrative interpretation of this definition. Note that the second and third condition actually produce a time warping in the corresponding

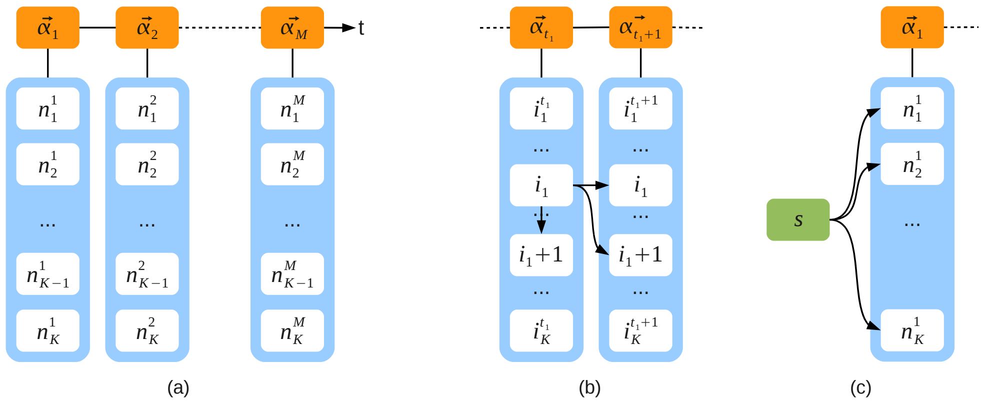


Figure 3.6: (a) The $M \cdot K$ nodes of the lazy neighborhood graph (visualized in white) are defined by the K locally nearest neighbors of M subsequent samples taken from the control signal (visualized in orange). (b) Edges are inserted between nodes that build *valid continuations*. The illustrated edges are based on the definition of *valid continuations* given by Krüger et al. [KTWZ10] (notation adjusted), with i_1 being the database index of a node $n_{k_1}^{t_1}$. (c) By adding an additional source node s (visualized in light green) and connecting it with all nodes n_k^1, we yield a single-source shortest-paths problem.

motion sequence, and that the three conditions essentially represent the three basic steps (diagonal, vertical, and horizontal) commonly used in traversing dynamic time warping (DTW) cost matrices; see Figure 3.10 (a). It can easily be seen, that the resulting graph is directed and acyclic. In addition, a topological ordering (which means, whenever there is an edge from x to y, the ordering visits x before y) is already given by construction.

Associating each node with costs proportional to the distance reported by the k-nearest-neighbor search[1], and paths with costs given by accumulating the costs of all related nodes, we now want to find paths of minimal costs that traverse the graph "from left to right", that is from $t = 1$ to $t = M$. These paths will be referred to as *global* or *maximum-length* paths in the following. By adding an additional source node s to the graph and connecting it to

[1]Krüger et al. formally assign these costs to (incoming) *edges* rather than to *nodes*. I, however, prefer to think of them as being assigned to nodes. Also, my definition is supposed to avoid confusion as I will later introduce an additional type of costs that naturally relates to edges.

all “left” nodes n_k^1, with $1 \leq k \leq K$ (compare Figure 3.6 (c)), this turns into a single-source shortest-paths problem, which due to the aforementioned properties of the resulting graph can be solved in linear time. Each global path then defines a global alignment between the input signal and the motion segment that is represented by the path.

Krüger et al. state that the paths found by this method “are equal to the paths found by subsequence dynamic time warping [Mül07] under the condition that all frames that are assigned to each other by subsequence DTW are in the neighborhood of the query motion.” Whether this condition is fulfilled or not does not only depend on the size K of the neighborhood and the length M of the regarded sequence, but also on the properties of the control signal. The more discriminative the control signal is, the less scattered are the retrieved neighbors in pose space (and at the same time also the corresponding indices in the graph), and the more valid continuations and hence paths exist in the resulting graph structure. While position-based control signals (like the ones discussed by Krüger et al.) in general provide the required discriminative power, the local neighborhoods of acceleration samples can be extremely scattered in pose space, as we have shown in Figure 3.5. As a consequence, in our scenario the resulting graph has in general less edges, which means that less global paths can be found. Often the graph does not even provide a single global path at all.

The most obvious way to compensate the scattering of indices is to increase the local neighborhoods, and indeed the neighborhoods in our scenario have to be significantly larger than when querying position-based features. However, increasing the neighborhoods too drastically will obviously slow down the whole process. Another way to compensate the scattering is to make M smaller and search for shorter paths. Note that we are justified to do this as in our scenario comparing sequences of accelerations instead of single time samples is just a means to an end: Finding a sequence of accelerations that matches our control signal makes the mapping into pose space more reliable, but in contrast to applications discussed by Krüger et al. we are by no means obliged to report sequences of a specific length. Ultimately, we are still searching for *poses* that match the actual performance at a single point in time, so there is no reason to restrict our search to paths of a certain length only. The definition of the LNG, however, requires us to fix M, and thus introduces a big problem: What is the right choice for M? Choosing it too large might cause that no (or too few[2]) global paths can be found, choosing it too small might prevent dissolving the discussed ambiguities. As the degree of scattering furthermore depends on the control signal itself—that is on the performed motion—what we actually want is M to be adaptive, or to only define an *upper bound* rather than a restriction to the path lengths. Unfortunately, this is not directly supported in the LNG. The only way to simulate adaptiveness of the window length in the LNG is to build a graph of maximum length and then perform several shortest-paths searches with different source nodes.

Another difference to applications of the LNG presented by Krüger et al. is that we are here dealing with a (sampled) *continuous* control signal and need to identify optimal subsequences in the knowledge base at each single point in time. This would require to build an LNG at each single point in time. Rebuilding the LNG for every new sensor reading, however, would be costly and unnecessary, since most of the data inside the graph structure can be reused. We now introduce an extension to the LNG that we refer to as *Online Lazy Neighborhood Graph* (OLNG). This data structure does not only enable very efficient updates but also allows us to efficiently extract paths of arbitrarily short and upper-bounded length.

3.4.3 Online Lazy Neighborhood Graph (OLNG)

We assume that the control input consists of a (sampled) continuous stream of sensor accelerations $(\ldots, \vec{\alpha}^{t-2}, \vec{\alpha}^{t-1}, \vec{\alpha}^{t}, \ldots)$, where $\vec{\alpha}^{t}$ denotes the accelerations at time $t \in \mathbb{Z}$, and that we want to find poses that match the actual performance at time t. For reasons discussed earlier we will exploit the temporal coherence of motions by querying sequences of accelerations instead of single time samples. In order to find these without introducing any latency into

[2]Remember that we want to identify similar poses in order to build local statistical models, which requires us to identify a certain number of paths.

our system, we consider the *last* M sensor readings $(\vec{\alpha}^{t-M+1}, \ldots, \vec{\alpha}^{t-1}, \vec{\alpha}^{t})$. In other words, we search for acceleration sequences in the knowledge base that match the sensor readings over the last M frames.

Let K be the number of (locally) nearest neighbors, and let S^t be the set of indices representing the K nearest neighbors of $\vec{\alpha}^t$, which can be efficiently computed using the kd-tree mentioned in Section 3.3. Our goal is now to identify a subset of S^t that is reliable enough to build a local statistical model of poses at time t. As discussed earlier, we consider a pose in S^t to be more reliable if a motion sequence of appropriate length containing this pose matches the control signal. A matching of sequences in turn can be found by identifying paths of consecutive indices in a data structure like the one defined by the LNG. As a consequence, in order to be able to numerically rate the reliability of the poses under consideration, we have to find for each node in S^t the path with minimal cost that leads to this node. Before outlining the procedure for updating the OLNG, I will first describe its overall structure and point out similarities and differences to the previously discussed LNG.

Similar to the LNG, nodes of the OLNG are defined by the frames reported by the k-nearest-neighbors search, and directed edges that encode temporal coherence between frames are inserted between nodes that build valid continuations. For building the OLNG, however, we use a slightly different definition of a valid continuation, enforcing *strict* monotonicity with respect to the column (time) index, which effectively limits the amount of possible motion warping. More precisely, we define $i_2 := i(n_{k_2}^{t_2})$ to be a valid continuation of $i_1 := i(n_{k_1}^{t_1})$, if one of the following three step size conditions applies:

- $i_1 + 1 = i_2$ and $t_1 + 1 = t_2$,
- $i_1 + 2 = i_2$ and $t_1 + 1 = t_2$,
- $i_1 + 1 = i_2$ and $t_1 + 2 = t_2$.

Figures 3.7 (b) and 3.10 (b) illustrate the three conditions.

Again, the edges allow for constructing paths, and each such path yields an index sequence, which in turn corresponds to a motion subsequence in the

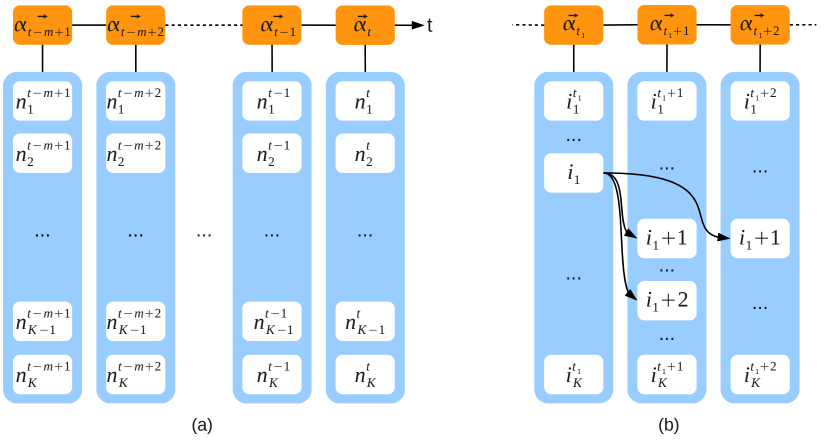


Figure 3.7: (a) The $M \cdot K$ nodes of the online lazy neighborhood graph (visualized in dark blue) are defined by the K locally nearest neighbors of the last M sensor readings. (b) As for the LNG, edges are inserted between nodes that build *valid continuations*. The illustrated edges are based on our definition of *valid continuations*, which differs from the one given by Krüger et al. [KTWZ10]. Here, i_1 is the database index of a node $n_{k_1}^{t_1}$. Note that one of the edges skips a column and thus has to be appropriately penalized in order not to favor shortcuts when comparing paths.

knowledge base. While costs for nodes are defined as in the LNG and likewise contribute to the total costs associated with a path, the structure of the OLNG requires us to introduce additional costs. These additional costs are in particular necessary since now we have to identify and compare paths of different lengths: While longer paths are preferred, they contain more nodes that add to their total cost than shorter paths, so shortness of paths must be appropriately penalized. This is done by adding an additional cost for each time sample within the considered time window that is not represented by the path. The quantity of this penalty cost has to be chosen such that it exceeds the maximum cost assigned to any node at the respective point in time, which is the cost assigned to the most distant (K-th) neighbor. In our implementation we are using twice the cost assigned to the corresponding K-th neighbor to penalize an unrepresented time point. For the same reason for which we have to penalize shortness of paths, we also have to introduce appropriate penalty costs for edges that skip columns (those created by the third step size condition; see Figure 3.7 (b)). Otherwise the resulting shortcuts would be favored in the shortest-paths search. A reasonable choice is to associate such an edge with the cost of one of the two nodes it is connecting. Edges that do not skip columns are defined to have zero costs. The total costs of a path are finally given by summing up

- the costs of all related *nodes*, measuring the local dissimilarities of acceleration samples,
- the costs of all related *edges*, avoiding a favoritism of shortcuts in the graph,
- and the costs penalizing overall shortness of paths in support of longer paths.

Based on these costs we can now easily rate and compare the poses in S^t, and only keep a subset for the construction of our local statistical pose model. Figure 3.8 illustrates the computation of path costs. Note that in this example nodes are identified by their costs rather than by their indices.

We now describe the procedure for efficiently updating the OLNG. Suppose that the OLNG has been constructed for the readings $(\vec{\alpha}^{t-M+1}, \dots, \vec{\alpha}^{t})$, and that for each node in S^T we have identified the path with lowest costs leading there. Now a new reading $\vec{\alpha}^{t+1}$ arrives. First, for $\vec{\alpha}^{t+1}$, the K nearest neighbors are retrieved (using the kd-tree) and stored in S^{t+1}. The OLNG is extended by adding nodes corresponding to these indices (forming a new last column). Furthermore, novel edges that end in the added nodes are introduced; see Figure 3.9 (b). These edges are chosen in such a way that they fulfill the step size and index conditions while extending previously constructed paths of minimal cost. Finally, the nodes corresponding to $\vec{\alpha}^{t-M+1}$ as well as the involved edges are removed to obtain the updated OLNG; see Figure 3.9 (d).

As the graph structure is built incrementally and not as a whole as proposed by Krüger et al. [KTWZ10], our implementation is suitable for online applications. There is no latency introduced by our OLNG, even at the

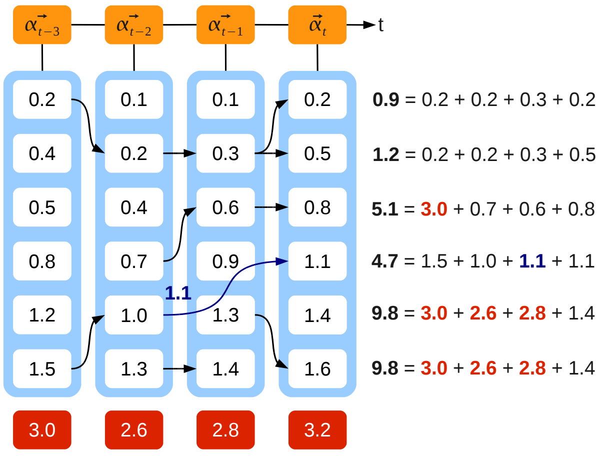


Figure 3.8: A toy example of size $M = 4$ and $K = 6$ illustrating the computation of path costs in the OLNG. The $M \cdot K = 24$ nodes are identified (as well as column-wise ordered) by their costs rather than by their indices. The costs of a path are given by summing up all costs assigned to nodes and edges along the path. An edge that skips a column is associated with the cost of the node it is pointing at, all other edges have zero costs. The red boxes indicate the costs that penalize unrepresented time points (or more general: shortness of paths), defined as twice the costs of the most distant neighbor.

beginning of a data stream. Moreover, the original “static” approach completely ignores all paths (motion segments) that start to evolve within the boundaries of a given frame window, regardless of their global performance. Due to its incremental nature, our approach detects and considers such paths directly as they appear. Hence, the window size M in our case only gives an upper bound on the length of retrieved motion segments without limiting them to that length, and can be seen as the *preferred* path length. Thus, in cases where no full-length matches can be found, shorter motion fragments are considered by our method.

In order to make the OLNG even more robust towards scattering of indices, we have considered additional valid continuations in our implementation. The main motivation for this was the finding that a single highly scattered (and possibly unrepresentative) local neighborhood can cause useful paths to

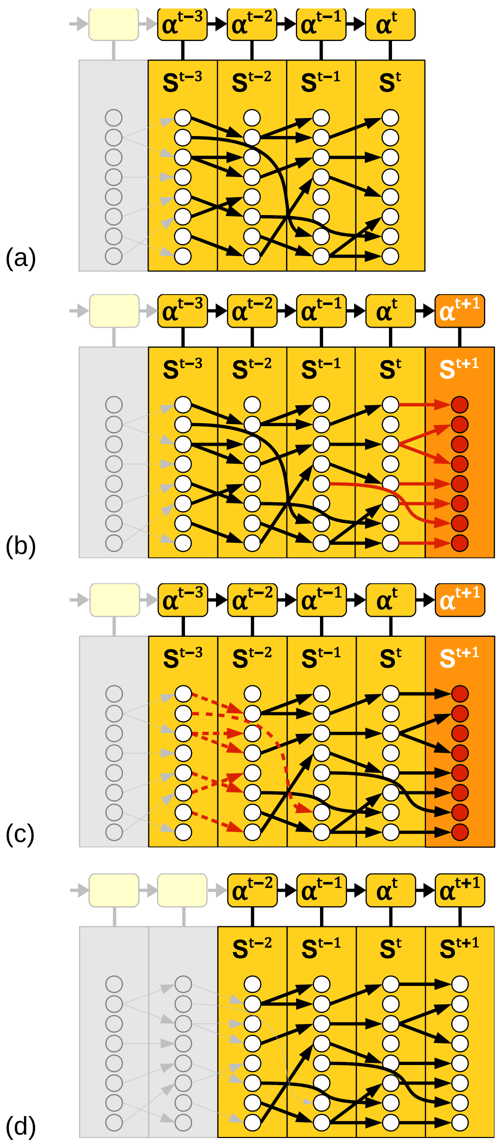


Figure 3.9: Online Lazy Neighborhood Graph (OLNG) with $M = 4$ and $K = 8$. Each vertical column corresponds to the K nearest neighbors (each neighbor indicated by a circle) of a sensor reading $\vec{\alpha}^{t-m+1}$, $m \in [1 : M]$. The edges encode temporal coherence between the nearest neighbors. The figure illustrates the implementation of the OLNG.

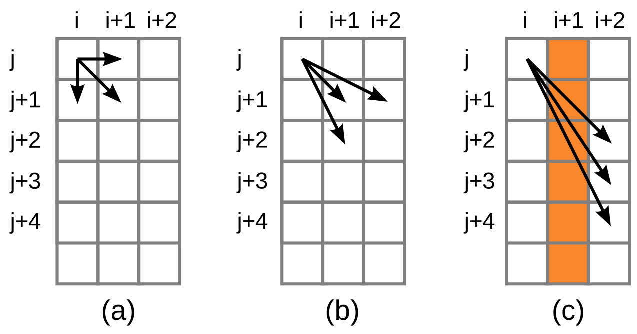


Figure 3.10: Visualization of the discussed step size conditions as their corresponding steps in a DTW cost matrix. (a) The step sizes used in the LNG. (b) The basic step sizes used in the OLNG. (c) Additional step sizes used in our implementation in order to bridge gaps caused by a single scattered neighborhood (visualized in orange).

be discarded. This is why we additionally defined $i_2 := i(n_{k_2}^{t_2})$ to be a valid continuation of $i_1 := i(n_{k_1}^{t_1})$ if one of the following three conditions applies:

- $i_1 + 2 = i_2$ and $t_1 + 2 = t_2$,
- $i_1 + 3 = i_2$ and $t_1 + 2 = t_2$,
- $i_1 + 4 = i_2$ and $t_1 + 2 = t_2$.

Note that appropriate costs have to be assigned to all the resulting edges. Figure 3.10 summarizes all discussed step size conditions by illustrating them as their corresponding steps for traversing a dynamic time warping (DTW) cost matrix.

In summary, the novel OLNG allows for extremely efficient retrieval of motion subsequences, which is of central importance for our online application. More precisely, by using the proposed implementation a speedup of more than one order of magnitude can be achieved for the examples presented in this work compared to tests with an implementation based on the static method. Generally speaking, this speedup is linear in the size of the sliding window M. Our retrieval procedure can handle moderate temporal variations and is extremely memory efficient: only the kd-tree of size $O(N)$ is stored and *one* OLNG of size $O(KM)$. Furthermore, each update step requires only $O(K \log N)$ operations, where the nearest neighborhood search determines the complexity. Opposed to previously introduced data structures of quadratic complexity, our approach scales well to large datasets consisting of millions of frames.

3.5 Motion reconstruction

The goal of our reconstruction approach is to closely approximate a performed motion. As our system is driven by a very low-dimensional control input, there is no way to directly infer complete high-dimensional motions. Thus, to eventually estimate plausible full-body results, the missing degrees of freedom need to be synthesized using the knowledge embedded in the database. While there exist many methods for synthesizing motions, the method of choice in most data-driven scenarios is to build a new motion based on similar ("neighboring") prerecorded motion clips or poses. We adopt this basic idea by using the online algorithm described in Section 3.4, which provides for each time-step t a set of K paths together with associated costs. Each path represents a motion subsequence in the knowledge base and points to a specific pose at time t, and the related cost is assumed to describe the dissimilarity of this pose to the actual performance at this point in time. In practice, due to the aforementioned properties of the control signal, most of the resulting paths are rather short and have high costs. In the following, we will discard these and only consider those $I \ll K$ paths having the lowest costs for building our local statistical model of poses at time t. We denote $C^t = \{c_1^t, \dots, c_I^t\}$ to be the costs of these I paths. Let furthermore $Q^t = \{\vec{q}_1^t, \dots, \vec{q}_I^t\}$ be the set of joint angle configurations of these poses, and $X^t = \{\vec{x}_1^t, \dots, \vec{x}_I^t\}$, $V^t = \{\vec{v}_1^t, \dots, \vec{v}_I^t\}$, and $A^t = \{\vec{a}_1^t, \dots, \vec{a}_I^t\}$ be the sets of positions, velocities, and accelerations of their joints, with respect to the root coordinate system. As these quantities were already computed in the preprocessing step, see Section 3.3, they can be easily obtained from the knowledge base at runtime. Finally, based on the costs $C^t = \{c_1^t, \dots, c_I^t\}$, we introduce *normalized weights* denoted by $W^t = \{w_1^t, \dots, w_I^t\}$, where the value of each weight w_i^t is given by

$$w_i^t = \frac{\max(C^t) - c_i^t}{\sum_{j=1}^{I}(\max(C^t) - c_j^t)}. \tag{3.3}$$

Now, we formulate motion reconstruction as an energy minimization problem. For each time-step we aim to find a pose $\vec{q}_{\text{best}}$ that optimally satisfies constraints imposed by the observation (measured acceleration data) while also being consistent with similar motion clips retrieved from the database. More precisely, our energy function to be minimized is based on two components where a data prior term enforces plausible reconstruction results and a control term is driven by the measured accelerometer data:

$$\vec{q}_{\text{best}} = \underset{\vec{q}}{\operatorname{argmin}}(w_{\text{prior}} \cdot E_{\text{prior}}(\vec{q}) + w_{\text{contr}} \cdot E_{\text{contr}}(\vec{q})). \tag{3.4}$$

Here, the two weights w_{prior} and w_{contr} are user-defined constants. In the following, we will take a closer look on the terms of this energy function. To this end, we assume that we have already reconstructed the motion up to time t. Now, at $t+1$, a new control input $\vec{\alpha}^{t+1}$ arrives from the sensors, which is used to update the OLNG. The most recent information we get from the OLNG are Q^{t+1} and W^{t+1}.

In the following sections, joint positions are predicted using short time integration. At this point we emphasize that despite this fact our method is *not* prone to error accumulation. The main reason for this is that the predicted joint positions are only used as an additional control input that helps to guide the synthesis in position space. Also, they are predicted for one frame into the future only. The overall motion, however, is mainly driven by the local models that are generated by the OLNG, which in turn is based on prerecorded database motions and continuously updated by the control signal itself.

3.5.1 Prior term

For human motions, similarity in a (low-dimensional) acceleration space does not automatically induce similarity on pose or joint velocity level. Thus, for large heterogeneous databases, motions with similar control signals tend to be scattered in both pose and velocity space. As our approach relies on such neighborhoods, using the control signal alone as objective function may yield artifacts such as jittering or degenerated poses. To avoid implausible results, a data-driven prior model that measures the a priori likelihood of a motion based on the motions given by the knowledge base is used. Our prior model consists of three different components: First, a *pose prior* characterizes the probability of a pose with respect to the distribution in pose space determined by database samples. Second, a *motion prior* measures the likelihood of a pose regarding the temporal evolution of a motion. Third, a *smoothness prior* reduces jerkiness. Based on this model a three-term energy function E_{prior} with user-defined weights w_{pose}, w_{motion}, and w_{smooth} is computed:

$$\begin{aligned} E_{\text{prior}}(\vec{q}) = {} & w_{\text{pose}} \cdot E_{\text{pose}}(\vec{q}) \\ & + w_{\text{motion}} \cdot E_{\text{motion}}(\vec{q}) \\ & + w_{\text{smooth}} \cdot E_{\text{smooth}}(\vec{q}) \end{aligned} \tag{3.5}$$

In contrast to existing approaches used in the context of motion synthesis [CH05], the term E_{prior} is effective also in cases where the retrieved poses Q^{t+1} belong to very different types of motion. Moreover, ad hoc smoothness heuristics are avoided by taking a data-driven approach.

Pose prior

The set of poses Q^{t+1} with corresponding weights W^{t+1} provided by the online algorithm are used to locally characterize the probability density in pose space. Instead of using a multivariate normal distribution model, as was done by Chai and Hodgins [CH05], we propose a kernel based approach to approximate the likelihood p_{pose} of a synthesized pose candidate $\vec{q}$:

$$p_{\text{pose}}(\vec{q}) \propto \sum_{i=1}^{I} w_i^{t+1} \cdot \mathcal{K}(|\vec{q}_i^{t+1} - \vec{q}|). \tag{3.6}$$

Here, $\mathcal{K}$ is a symmetric kernel function. Note that such a kernel-based representation is well suited to approximate arbitrary shaped probability density functions including multiple peaks, which is a desirable property not only for our application but also for data-driven motion synthesis in general. As for conventional unit integral kernel functions (e.g., Gaussians), p_{pose} is maximized for poses that are likely according to the samples included in the database. To this end, the prior needs to be reformulated as an expression suitable for energy minimization. In practice, a square root kernel is used to compute the energy term E_{pose}:

$$E_{\text{pose}}(\vec{q}) = \sum_{i=1}^{I} w_i^{t+1} \cdot \sqrt{|\vec{q}_i^{t+1} - \vec{q}|}. \tag{3.7}$$

The above expression yields results (regarding optimality) comparable to p_{pose} (see Figure 3.11) while—due to the choice of $\mathcal{K}$—not being prone to numerically vanishing gradients if $p_{\text{pose}} \approx 0$, which is desirable for gradient-based energy minimization techniques.

Motion prior

Besides being plausible on a pose level, the temporal evolution of a reconstructed motion should be consistent with motions observed in reality. More specifically, the movement of the joints should be directed in a believable way. The latter objective is achieved by employing a motion prior accounting for the joint velocities V^{t+1} and the joint accelerations A^{t+1} of the neighboring database poses included in Q^{t+1}. To be more precise, we estimate a probability density distribution for $\vec{x}^{t+1}$ (the true joint positions at time $t+1$) by computing the second-order Taylor expansion at the joint positions $\vec{x}^t$ (associated to $\vec{q^t}$) using V^{t+1} and A^{t+1}. For the i-th sample ($i \in \{1,..,n\}$),

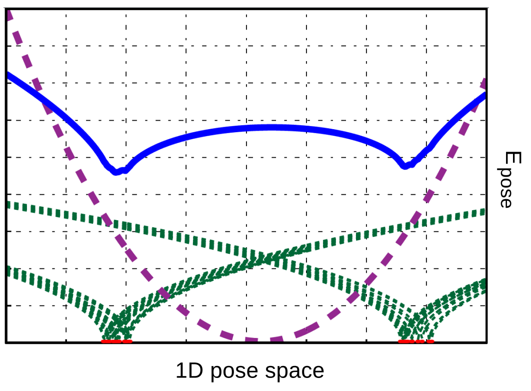


Figure 3.11: A simple example illustrating the effect of our kernel based approach in case of clustered data samples. Here, the green dashed lines indicate the kernel functions centered at the sample positions and the solid blue line represents our energy term E_{pose}. The dotted purple line symbolizes the energy function proposed by Chai and Hodgins [CH05]. Please note how the local sample density is determining the likelihood of a pose candidate in contrast to Chai and Hodgins: Clusters of samples induce distinctive local minima of E_{pose}.

the estimated positions $\vec{x'}_i^{t+1}$ are then given by

$$\vec{x'}_i^{t+1} = \vec{x}^t + \vec{v}_i^{t+1} \cdot \Delta t + \frac{1}{2}\vec{a}_i^{t+1} \cdot \Delta t^2. \tag{3.8}$$

To approximate the probability density based on the set $\{\vec{x'}_i^{t+1} | i \in [1 : I]\}$, we take a kernel-based approach very similar to p_{pose}. Hence, for the resulting energy term, with $\vec{x}$ denoting the joint positions of a pose candidate $\vec{q}$, one gets

$$E_{\text{motion}}(\vec{x}) = \sum_{i=1}^{I} w_i^{t+1} \cdot \sqrt{|\vec{x'}_i^{t+1} - \vec{x}|}. \tag{3.9}$$

Smoothness prior

Using prior and control terms for energy minimization already yields plausible results in many cases. However, as these two terms at most account for the last synthesized pose, high frequency jitter may occur. In contrast to most existing approaches that attempt to enforce smoothness by minimizing joint accelerations, we make direct use of the a priori knowledge provided by the database. A pose $\vec{q}$ (with joint positions $\vec{x}$) is assumed to be plausible, if the induced joint accelerations are consistent with the joint accelerations of neighboring database samples. Again, as for pose and motion priors, the likelihood of a pose candidate is measured by kernel based density estimation:

$$E_{\text{smooth}}(\vec{a}) = \sum_{i=1}^{I} w_i^{t+1} \cdot \Delta t \cdot \sqrt{|\vec{a}_i^{t+1} - \vec{a}|}, \tag{3.10}$$

with

$$\vec{a} = \Delta t^{-2} \cdot \left(\vec{x} - 2\vec{x}^t + \vec{x}^{t-1}\right). \tag{3.11}$$

3.5.2 Control term

Accelerations have already been used to retrieve motion subsequences (and thereby also poses) that are likely to be similar to the actual performed ones. As the subsequent motion synthesis is based on these poses, this step already provides a certain degree of implicit control and effectively restricts the space of possible outcomes. However, a direct use of these accelerations as control signal is not a viable choice as it provides not enough discriminatory power to guarantee a similarity in pose space, which is essential for a stable motion reconstruction. For exactly that reason, the control term is computed based on extremal joint positions that closely match the actual sensor positions.

Let $\langle\vec{y}\rangle$ be the projection of a vector $\vec{y}$ to the subspace formed by the components related to the joints that are next to the virtual sensors. Assuming proper positions $\vec{x}^t$ at frame t the probability density distribution of the next joint positions at $t+1$ is estimated by numerical integration of the equation of motion using V^t:

$$\tilde{\vec{x}}_i^{t+1} = \left\langle \vec{x}^t + \vec{v}_i^t \cdot \Delta t \right\rangle + \frac{1}{2}\hat{\vec{\alpha}}^t \cdot \Delta t^2. \tag{3.12}$$

Here, accelerations $\hat{\vec{\alpha}}^t$ are computed by transforming control signal readings $\vec{\alpha}^t$ to root frame coordinates using the local frames induced by the previously synthesized pose $\vec{q}^t$ and subtracting gravity. Assuming that the database includes motions similar to the one performed, we use $\{\vec{\tilde{x}}_i^{t+1} | i \in [1:I]\}$ to derive an energy term to be minimized:

$$E_{\mathrm{contr}}(\vec{x}) = \sum_{i=1}^{I} w_i^{t+1} \cdot \sqrt{|\vec{\tilde{x}}_i^{t+1} - \langle \vec{x} \rangle \,|}. \tag{3.13}$$

Using velocities from the database effectively avoids overshooting effects that would otherwise occur if, for example, no smooth transition between different poses can be synthesized.

Naturally, our approach is only approximate as no direct control in pose space is available and the quality of results depends on estimated properties (such as the current pose) and the motion clips included in the database. Moreover, root accelerations have not been explicitly considered, which makes the method less accurate in case of high-dynamic root movement. However, despite all these theoretical deficiencies, the proposed method works well in practice. The main reason is that, if a class of motions is included in the database, the reconstructed motion is mainly driven by this data and only adjusted by measured joint accelerations.

3.5.3 Energy minimization

We employ a gradient-descent-based method[3] to minimize the objective function (Equation 3.4) with respect to a pose that optimally satisfies our statistical and control constraints. Initializing energy minimization with the previously synthesized pose, the method usually quickly converges after few iterations. During optimization, the different user-defined weights included in Equation 3.4 were kept fixed at the following values: $w_{\mathrm{contr}} = 1$, $w_{\mathrm{prior}} = 5$, $w_{\mathrm{pose}} = 0.6$, $w_{\mathrm{motion}} = 0.2$, $w_{\mathrm{smooth}} = 0.2$. According to our experience, slightly changing these values does not substantially affect the overall quality of reconstruction results.

To decrease optimization costs and to improve robustness of the approach, this minimization is not performed in the high-dimensional pose domain. As originally proposed by Chai and Hodgins [CH05], a local linear model ap-

[3]The `lsqnonlin` function (large scale) of MATLAB was used.

proximation of the pose space is applied instead. Using a weighted PCA for dimensionality reduction we take full advantage of the pose weights W^t computed at each frame. Generally, there is a trade-off between accuracy and optimization speed. If a fast synthesis is essential, a lower-order PCA approximation will, while being less accurate, yield faster results. Preserving 99% of the original variance, the dimension of a pose reduced from 43 for the full pose representation to as few as 14 components on average while still producing visually satisfying results. Note that the nature of our control signal, in contrast to a position-based one, may cause scattered and in particular clustered neighborhoods in pose space and thus suppress strong dimensionality reduction.

3.6 Results

We have tested the effectiveness of our system with *simulated* as well as *real* sensor readings. As relating human perception to numerical distance measures is inherently difficult [TWC+09], the widely accepted average RMS error of joint positions (relative to the skeleton root frame) is used in the following for all numerical comparisons. Please note that the distance measure introduced in Section 2.4 performs all computations with respect to a global coordinate frame, we are here, however, not explicitly reconstructing the root motion. While we describe a very simple procedure for estimating the root motion in Section 3.6.4, synthesized motions may be still arbitrarily rotated about the vertical axis. The suggested distance measure thus can not be directly used for evaluating our system.

3.6.1 Tests based on real sensor readings

The treadmill experiment

One of the very first experiments we did was the following *treadmill experiment*. Its goal was to evaluate the OLNG search rather than the motion synthesis which hadn't been fully developed yet at this point. The nice

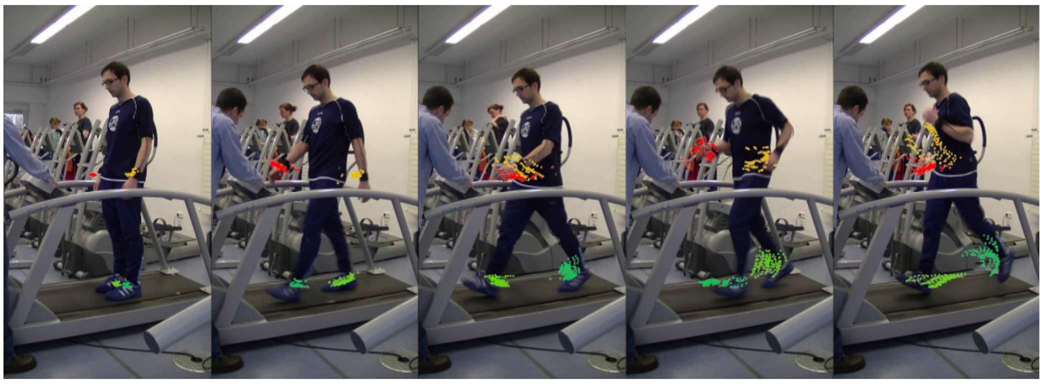


Figure 3.12: Snapshots taken from the video of the treadmill experiment. The video shows an overlay of the 128 most similar poses identified by the OLNG, and a recording of the actual performance. The 128 poses are visualized by the positions of their wrist and ankle joints, with red, orange, green, and turquoise representing the right wrist, left wrist, right ankle, and left ankle, respectively. The snapshots were taken at times of very different treadmill speeds.

thing about the treadmill environment was that it allowed us to very precisely control and smoothly vary the speed of the performance. One person equipped with four accelerometers (attached next to his wrists and ankles) simply had to adapt his motion to the speed of the treadmill, which was controlled by a second person. Starting at rest, the speed of the treadmill was continuously and slowly increased to 12 km/h, thereby smoothly changing the athlete's motion from standing to walking to jogging/running. While the used knowledge base contained walking and jogging motions (among other motion classes), the very fine variations and transitions between them were not directly represented. We used the OLNG based search to find for each frame of the performance the 128 most similar poses.

As we do not have ground-truth data of the athlete's performance, we evaluated the quality of the retrieved local models visually. The accompanying video shows an overlay of the 128 poses that were considered to be most similar to the actual performance by the OLNG—visualized by the positions of their wrist and ankle joints—and a video recording of the performance. Please note that due to a missing camera calibration the overlay had to be adjusted manually so that the two layers do not perfectly match. Figure 3.12 shows some snapshots of the video. It can be seen in the video that the very slow motion at the beginning of the performance is barely identified by the OLNG. This is because the measured acceleration is too small and thus not discriminative enough. The athlete is, however, still slowly walking when the walking pattern is finally captured and reflected by the local models. During normal and fast walking, the sets of poses are very bundled and very few frames show scattered poses. The transition from fast walking to slow jogging is immediately reflected by the retrieved local models and introduces only a little amount of scattering for a very short time. In the following, the retrieved poses reflect the actual performance very well until the maximum running speed of 12 km/h is achieved. All in all, we considered the results of this first experiment as very promising and motivating for our following work.

A question that might arise is whether the treadmill environment actually reflects a realistic scenario with respect to the sensor readings, as it obviously eliminates basically every forward motion of the athlete. As a consequence, in contrast to any general environment, the athlete's global position is even at high speeds barely moving on a treadmill. Since we are dealing with accelerations, the difference in sensor readings, however, is actually very small. In fact—what might not be obvious at first sight—it is only the acceleration of the treadmill that introduces a deviation, and its actual speed does not affect the sensor readings at all. In other words, it does not make any difference to the accelerometer readings whether the athlete is running (with constant speed) on a treadmill or on the street (neglecting any differences in the conditions of the ground). Since in our experiment the acceleration of the treadmill is very small compared to the actual accelerations of the sensors, it can be neglected and the environment can be seen as a realistic scenario.

Outdoor motion reconstruction

In a second test, we reconstructed motions of two different actors performed in an outdoor setting. As we again do not have ground-truth data to quantify

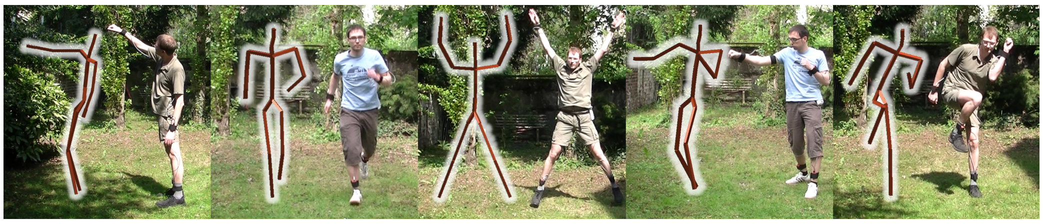


Figure 3.13: Frames taken from the video illustrating our outdoor motion reconstruction experiment. The motion synthesis was driven by four 3D accelerometers attached to the lower arms and lower legs of one of two actors. Reconstructed poses are always shown to the left of the original performance.

the reconstruction accuracy in this case, the results shown in the video—reconstruction alongside a video recording of the original performance—only provide a qualitative comparison. However, the main challenges are illustrated by the given examples. In addition, the outdoor setting clearly shows that, as opposed to optical systems, inertial-based devices as used in our scenario impose only very little constraints on the actor or the recording environment with regard to lighting conditions, recording volume, or setup.

Due to the time warping capabilities of the OLNG employed for finding close matches to a given query, our approach is not sensitive to moderate temporal variations. As a consequence, we are able to synthesize motions at speeds not explicitly covered by the knowledge base, as already indicated in the treadmill experiment. As it is inherently complicated to create temporal variation for arbitrary motions, we restricted an additional analysis to a sequence of localized jumping jack motions performed at different speeds. The results—the reconstruction errors with respect to ground-truth data, given for our method as well as a variant that avoided to time warp motions in order to match them—are summarized in Figure 3.14. Here, the lower errors obtained with our method clearly show the general advantage of the OLNG over a conventional linear search algorithm that does not account for time warped motions in the retrieval step.

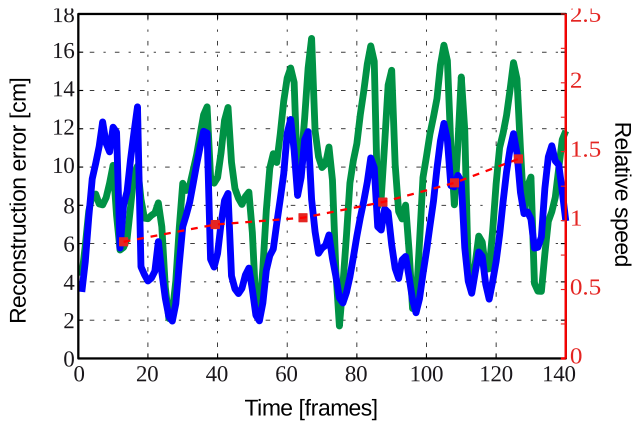


Figure 3.14: Smoothly varying the relative speed of a jumping jack motion over time. Here, the dashed red line indicates the relative speed and the red dots the "hand clapping" frame for six subsequent jumping jacks. Blue: Reconstruction error using our method. Green: Reconstruction error using a variant of our method that does not account for time warped motion sequences.

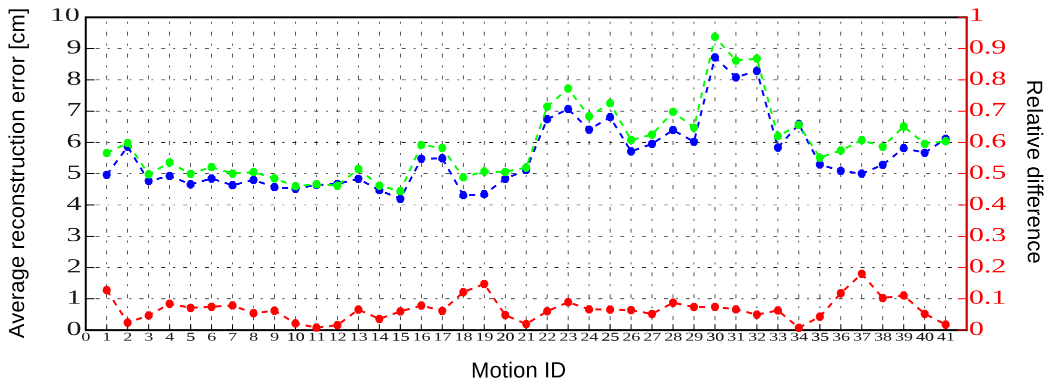


Figure 3.15: Reconstruction error for motions with recorded ground-truth. Blue: reconstruction errors using sensor data (MTx accelerations). Green: reconstruction errors using simulated accelerometer readings. Red: relative difference between both.

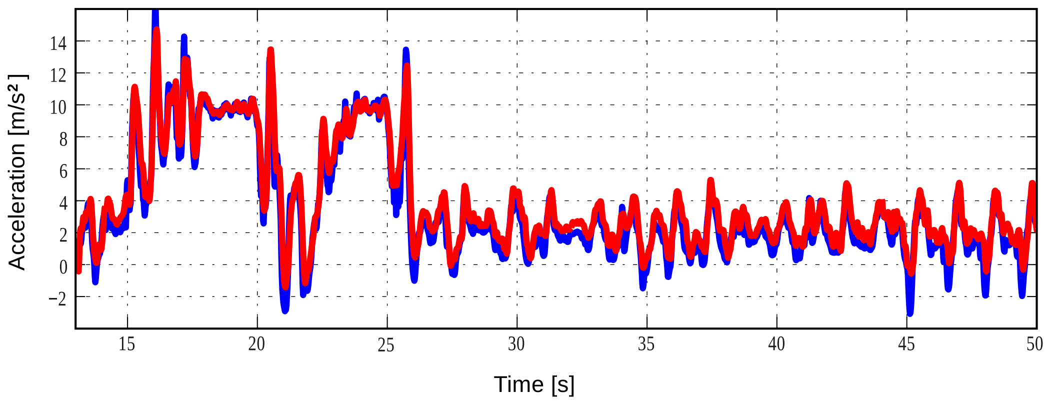


Figure 3.16: Comparison of simulated sensor data (red) and real accelerometer readings (blue) obtained from an MTx device attached to a user's left wrist.

Comparison to simulated accelerometer readings

In order to quantify differences between real and simulated sensor readings (computed from given MoCap data), we simultaneously captured a set of 41 motions (of one actor) using Xsens MTx sensors and an optical motion capture system (a 12-camera Vicon system). Then, we reconstructed all these motions on the basis of the actual sensor readings as well as of simulated ones. The average reconstruction errors for both scenarios are given in Figure 3.15. Although almost all reconstructions on the basis of simulated data numerically perform slightly better, the differences of the reconstruction errors are much smaller than the reconstruction error of our method per se. Also, both error curves are highly consistent in their overall course.

As demonstrated by the numerical ground-truth comparison, simulated sensors yield comparable results to real readings. This high similarity is furthermore underlined by Figure 3.16, where real sensor readings are compared directly to simulated ones. These observations enable us to use the large body of systematically recorded motions of the HDM05 database [MRC+07] for systematic evaluations of our method.

3.6.2 Tests based on simulated sensor readings

In this section, we report on a series of tests to evaluate how our proposed reconstruction behaves under the variation of several important aspects. To this end, we first take a closer look on how our reconstruction is affected by the size and diversity of the used knowledge base. Second, we elaborate on the influence of the window length M used in the OLNG. Third, we test how the size of the actor influences the reconstruction process, in particular when the actor to be reconstructed is much smaller or much larger than all the actors included in the knowledge base. Finally, we evaluate how the number and the placing of the sensors affects the quality of our reconstruction.

General scenarios used for testing

In the following experiments, the knowledge base consists of motion clips taken from the publicly available motion database HDM05 [MRC+07]. This database consists of various parts and sections in which different motion classes including locomotion, grabbing and depositing, and sports motions are performed. The motions inside the database were performed by five different actors, referred to by their initials (`bd`, `bk`, `dg`, `mm`, `tr`). In the following, we denote different knowledge bases by the same naming pattern that was used in the documentation of the HDM05 database to describe single motion files:

`HDM_{actor}_{part}-{scene}_{take}_{framerate}`.

In our case, we use asterisks serving as wildcards to represent any possible value of that field. Furthermore, if an "M" was added to the name, also copies that have been mirrored at the natural symmetry axis of the skeleton (inverting "left" and "right") were added. Analogously, a suffix "R" means that also the time-reversed counterparts of the motions were added to the knowledge base. For example, the knowledge base `HDM_bd_01-**_**_25M` represents all motion clips from Part 1 of the HDM05 database performed by the actor `bd`, together with their mirrored copies.

In all of the following experiments, the used control data was obtained by simulating virtual sensors, as described in Section 3.3, using a set of *test*

Scenario	Knowledge base
1	Contains *also* motions of actor to be reconstructed
2	Contains *only* motions of actor to be reconstructed
3	Contains *no* motions of actor to be reconstructed

Table 3.1: Summary of the three scenarios used for our evaluations.

motions also obtained from the HDM05 database. The actual test motion itself was never included in the knowledge base. Furthermore, we defined three different reconstruction *scenarios*, where the type of the scenario was determined by whether the actor of the considered test motion was included in the knowledge base or not; see Table 3.1 for an overview.

For some of the following experiments, we additionally defined three special types of knowledge bases matching the scenarios shown in Table 3.1.

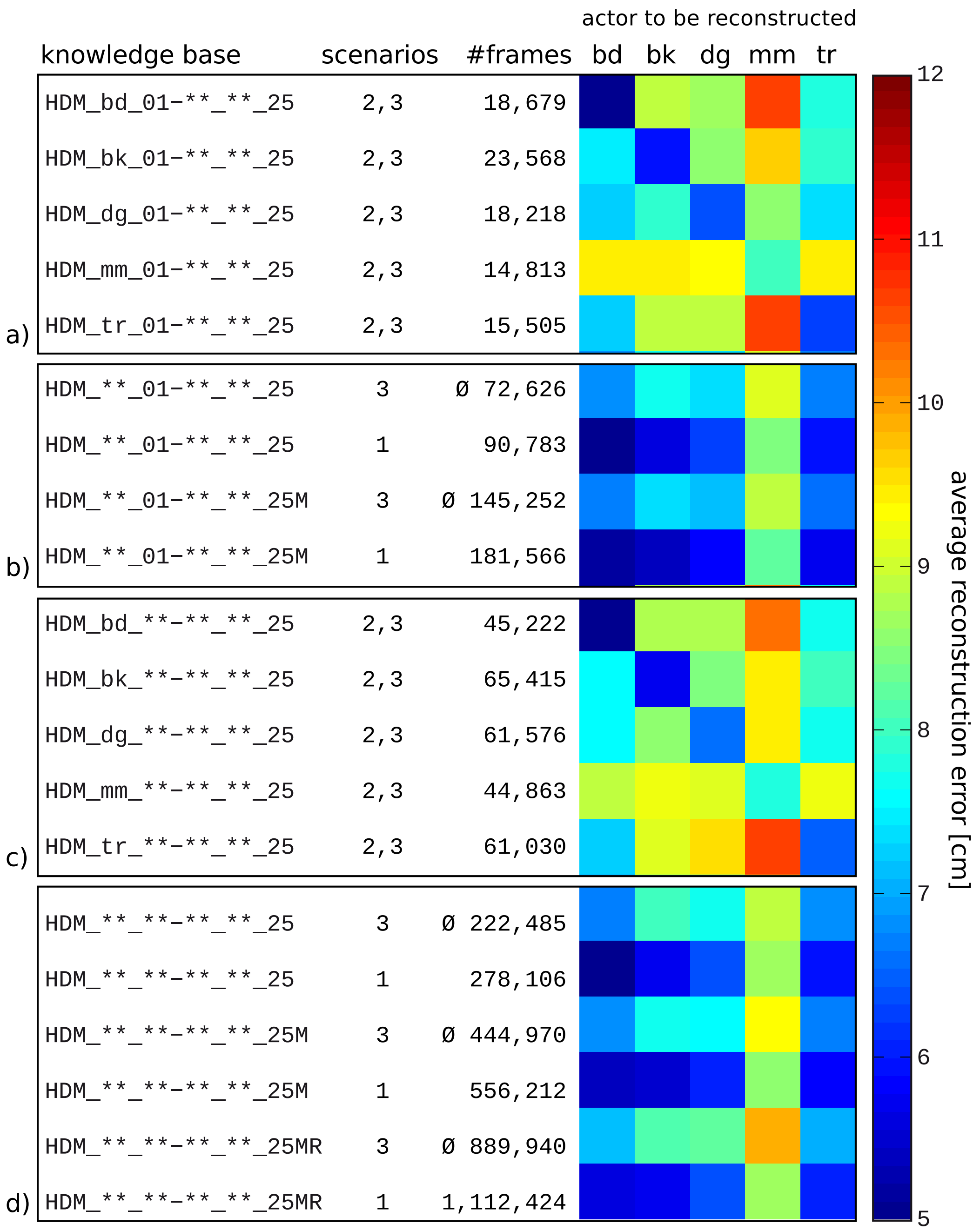
- DB_1: All motion clips of all five actors contained in the HDM05 database together with mirrored copies. In total this knowledge base comprises about 0.56 million frames (370 minutes of MoCap at 25 fps). Again, despite the fact that the actor to be reconstructed is always included in this knowledge base, the corresponding test motions are *never* included in the knowledge base.
- DB_2^i: A subset of DB_1 including only motion samples of the i^{th} ($i \in [1:5]$) subject, whose motion is reconstructed.
- DB_3^i: DB_1 without samples of the subject, whose motion is reconstructed ($\mathrm{DB}_3^i := \mathrm{DB}_1 \backslash \mathrm{DB}_2^i$).

Size and diversity of the knowledge base.

In a first experiment, we analyzed how the size and diversity of the knowledge base influences the quality of the reconstructed test motions. To this end, we created five sets of test motions (one for every actor), each containing six motion clips from Part 1 (locomotion) of the HDM05 database, deliberately chosen in such a way that every motion class described in Part 1 was covered. Additionally we composed a set of 20 different knowledge bases, largely differing in size and diversity and reflecting all previously described scenarios, and used them to reconstruct all test motions of all actors.

The reconstructed motions were then compared to the original test motions using the RMS error of joint positions. The results of this experiment are shown in Figure 3.17. The columns give the name of the knowledge base, the scenarios resulting from combining this knowledge base with the respective test motions, the size of the knowledge base (with average numbers of frames for scenario 3, where all motions of a certain actor had to be removed), and—for each actor separately—the color-coded averaged RMS errors of the reconstructed motions.

The first important observation of this experiment is that the reconstruction quality is noticeably better when motions of the actor to be reconstructed are contained in the used knowledge base (scenario 1 and 2), one, however, still obtains satisfying results if this is not the case (scenario 3). This can directly be seen by comparing subsequent rows in subfigures 3.17 (b) and 3.17 (d), as well as by the prominent blue diagonals in subfigures 3.17 (a) and 3.17 (c) representing scenario 2. As our method is very robust towards variations in actor sizes (see Section 3.6.2) and relatively robust towards moderate variations in speed (see Section 3.6.1), this is mainly due to the variations in style that exist between different actors. When examining the underlying motion data from the HDM05 database it becomes clear that this is especially true for actor `mm`, where one can observe large performance variations even within the same actor category. The second important observation is that the reconstruction quality only slightly decreases when knowledge bases become bigger and less homogeneous. This can be seen by comparing rows corresponding to same scenarios in subfigures 3.17 (b) and 3.17 (d), as well as by comparing the overall appearances of subfigures 3.17 (a) and 3.17 (c). Strictly speaking, one should distinguish between a mere increase of the size of the knowledge base (e.g., obtained by including mirrored motions) and the apparent increase of the diversity by including new motion classes. This is, however, quite difficult, as both attributes are strongly related in most practical scenarios. In particular, the results for actor `mm`, however, support our intuition that increasing the diversity has a higher influence on the results than a mere inflation of the knowledge base.



	knowledge base	scenarios	#frames
a)	HDM_bd_01-**_**_25	2,3	18,679
	HDM_bk_01-**_**_25	2,3	23,568
	HDM_dg_01-**_**_25	2,3	18,218
	HDM_mm_01-**_**_25	2,3	14,813
	HDM_tr_01-**_**_25	2,3	15,505
b)	HDM_**_01-**_**_25	3	Ø 72,626
	HDM_**_01-**_**_25	1	90,783
	HDM_**_01-**_**_25M	3	Ø 145,252
	HDM_**_01-**_**_25M	1	181,566
c)	HDM_bd_**-**_**_25	2,3	45,222
	HDM_bk_**-**_**_25	2,3	65,415
	HDM_dg_**-**_**_25	2,3	61,576
	HDM_mm_**-**_**_25	2,3	44,863
	HDM_tr_**-**_**_25	2,3	61,030
d)	HDM_**_**-**_**_25	3	Ø 222,485
	HDM_**_**-**_**_25	1	278,106
	HDM_**_**-**_**_25M	3	Ø 444,970
	HDM_**_**-**_**_25M	1	556,212
	HDM_**_**-**_**_25MR	3	Ø 889,940
	HDM_**_**-**_**_25MR	1	1,112,424

Figure 3.17: Average reconstruction error for a given knowledge base and a given actor to be reconstructed. In addition, the sizes of the knowledge bases (in terms of number of frames) as well as the effective scenarios are indicated (with blue diagonals in the first and third box representing Scenario 2).

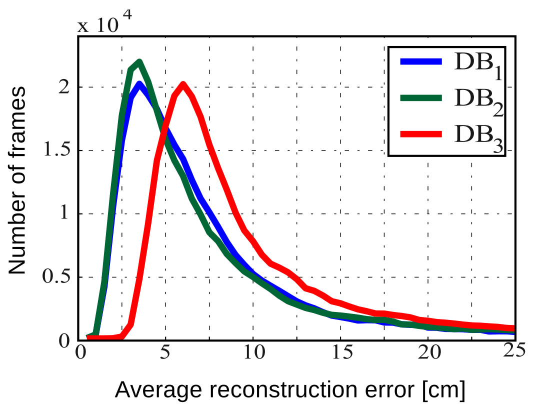


Figure 3.18: Histogram of the average RMS error for the different scenarios.

Since the averaged RMS error over all frames gives only a limited insight, we conducted another experiment in which we analyzed the distribution of the RMS error. To this end, we reconstructed *all* motions of the HDM05 database, using the knowledge bases DB_1, DB_2^i, $i \in [1:5]$, and DB_3^i, $i \in [1:5]$, reflecting the scenarios 1, 2, and 3. While in case of DB_1 all motions were reconstructed using the same knowledge base (except for the absence of the currently regarded test motion), in case of DB_2^i and DB_3^i only motions performed by the i^{th} actor were reconstructed, and the results were unified and summarized as DB_2 and DB_3 respectively. Here, for each scenario, the resulting per-frame RMS errors of all reconstructed motions were accumulated and plotted as histogram with a binning of 0.5 cm; see Figure 3.18. As indicated by the narrow peaks at relatively low error levels, reconstructed poses are very likely to be consistent with the original ones. Moreover, DB_1 and DB_2 give higher-quality results than DB_3, as has been expected.

Window size

As our motion synthesis highly depends on the quality of the local models identified by the OLNG, the size M of the window used to retrieve paths might be a critical parameter. However, as M only defines an upper bound to the lengths of paths rather than constraining them to a specific length (as in the algorithm described by Krüger et al. [KTWZ10]), its assignment is far less

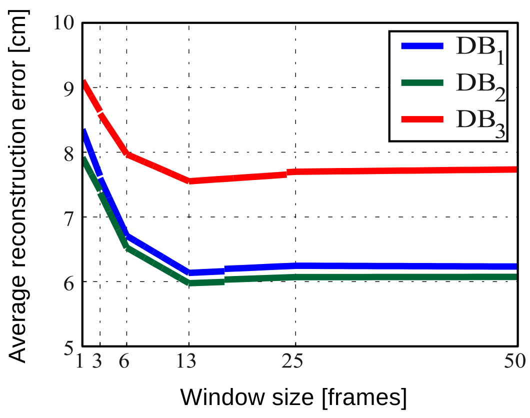


Figure 3.19: Influence of the window size M of the OLNG. All motions were sampled at 25 frames per second.

critical than in the original approach. Figure 3.19 shows the reconstruction results using different window sizes. As can be seen, the minimum reconstruction error was obtained when using a window size of 13 frames, which roughly corresponds to half a second at a sampling rate of 25 frames per second. The significantly larger errors at smaller window sizes indicate that sequences of a certain length effectively help to evaluate and disambiguate the local neighborhoods. The slight increase of the error when expanding the window to 25 frames can be explained by paths that stretch across the entire window of 25 frames, and thus are preferred over shorter ones, although they describe the control signal significantly worse in its second half (the last 13 frames). This result indicates that half a second is a viable choice for the window size, and that accelerations older than half a second may not be representative for the actual performance anymore. An easy way to take the "aging" of paths into account would be to weigh the costs of nodes proportional to their (temporal) distance to point of time we are actually interested in. The reason for the error not increasing more when further expanding the window size is simply that in our scenario almost no paths longer than a second are found.

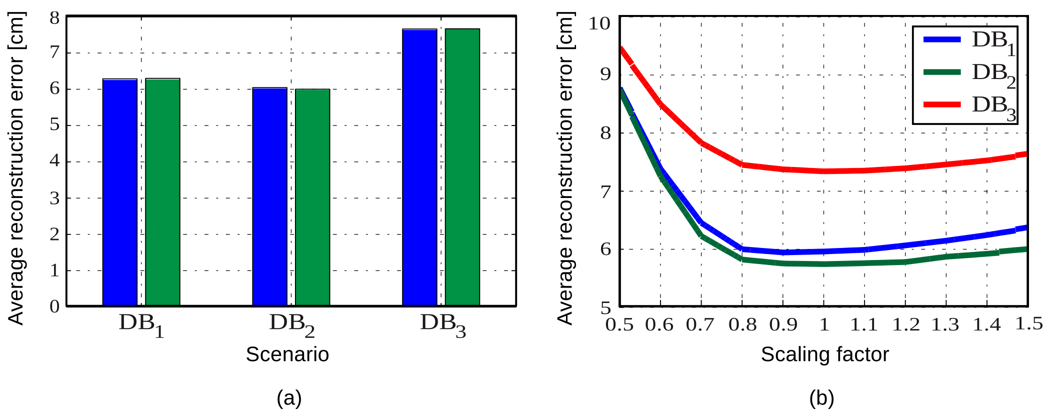


Figure 3.20: Dependency of the average reconstruction error on the size and proportions of actors: (a) Comparing the use of original proportions (blue) in the underlying knowledge base with our standard approach of using skeletons with averaged bone lengths (green). (b) Uniformly scaling the actor to be reconstructed, while using averaged skeletons in the underlying knowledge base. The scaling factor is indicated, ranging from 0.5 to 1.5.

Size and proportions of actors

To investigate how different sizes and proportions of actors affect the reconstruction results, we performed two tests. In a first experiment we built the knowledge bases (containing simulated sensor readings) using the original skeleton information of the five different individuals—with body heights ranging from roughly 170 cm to about 200 cm—included in the HDM05. As can be seen from Figure 3.20 (a), the reconstruction error is virtually unaffected by naturally occurring variations in actor sizes. Also, these numerical findings are supported by visually comparing the quality of reconstructed motions. For a second, more synthetic test, we got back to our standard practice, using knowledge bases built upon skeletons whose bone lengths were averaged across the different actors. Now, however, we systematically scaled the bone lengths of the actor to be reconstructed in the range $[0.5, 1.5]$ (hence modified the simulated sensor readings) while keeping the knowledge bases unchanged. To account for the fact that the used point cloud distance measure linearly depends on the scaling factor, the scaled bone lengths were transformed to their original size after reconstruction for the sake of comparison. Figure 3.20 (b) shows the average reconstruction errors in the three described scenarios plotted against the scaling factor. Again, this test indicates that the reconstruction is relatively robust regarding diversity in body height.

The prior model

As discussed in Section 3.5.1, the novel priors substantially improve the reconstruction quality compared to existing models previously presented in the context of motion synthesis. In order to validate this claim we made comparisons to methods that attempt to adapt existing prior models to our framework:

- Using a prior along the lines of Chai and Hodgins [CH05]: A local multivariate normal distribution was used to approximate the distribution of local neighbors in pose space and the distance of a synthesized pose was measured by its Mahalanobis distance to compute E_{contr}. Moreover an ad hoc smoothness prior replacing the original energy term E_{smooth} was employed that minimizes joint accelerations.

- Employing the Mahalanobis distance (instead of using the proposed kernel regression method) for each term of the prior E_{contr}, E_{motion}, and E_{smooth}.

The results, the average reconstruction error for each of our database scenarios summarized in Table 3.21 clearly evidence the benefit of using our motion priors together with kernel regression: The average reconstruction error decreases by about 30% for the new model.

Tests with different sensor setups

The average reconstruction error was analyzed for different sparse sensor configurations (refer to Table 3.2) including one to six sensors. This was done by both a histogram based approach, similar to Section 3.1 (see Figure 3.22 (a)), performed on the complete HDM05 database, and—for the sake of easier comparison—an evaluation of the subset of test motions described

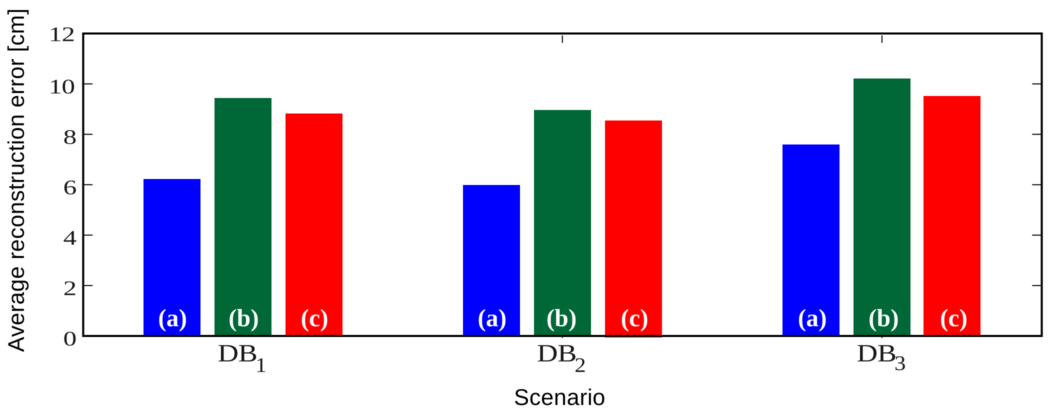
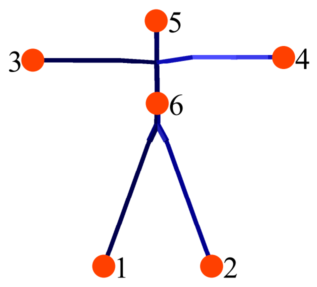


Figure 3.21: Comparing the average reconstruction error obtained with the new prior model (a) to approaches that attempt to adapt existing ones to our framework: (b) Replacing kernel regression in E_{contr}, E_{pose}, E_{motion}, and E_{smooth} by Mahalanobis distance; (c) Using multivariate normal distribution prior model for E_{contr} together with an ad hoc smoothness assumption.



Number of sensors	Used sensors
1	[3]
2	[2 3]
3	[1 2 3]
4	[1 2 3 4]
5	[1 2 3 4 5]
6	[1 2 3 4 5 6]

Table 3.2: Different sensor setups

earlier in this section (see Figure 3.22 (b)). Naturally, additional sensors tend to improve the reconstruction quality as less information needs to be inferred from the knowledge base. However, as demonstrated by our results, a large variety of motions can be well approximated with surprisingly few sensors. This is in particular true for motions that are performed similarly across different individuals (e.g., walking or running motions) where more than four sensors gave no substantial improvement. Of course, our reported results are empiric results with respect to the test motion database: Although our test databases taken from HDM05 contain a variety of motions, they contain rather few motions where there are different movements of the head

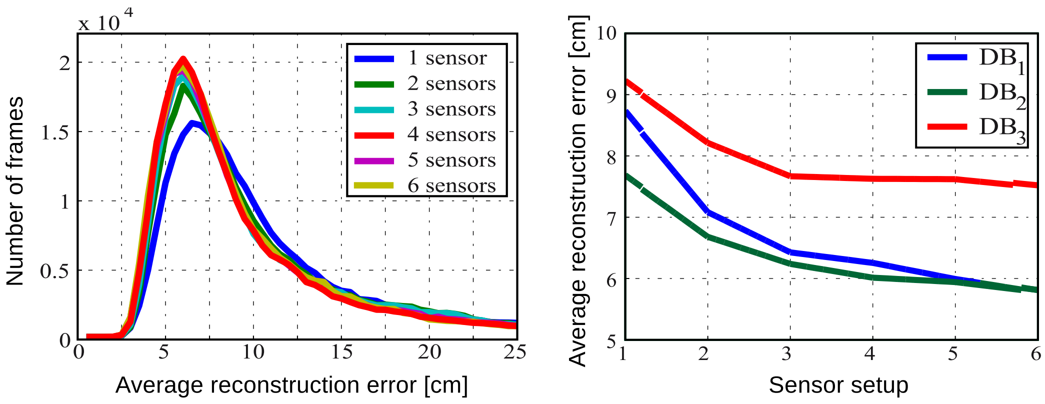


Figure 3.22: Dependency of the average reconstruction error on the sensor setup: (a) Histogram-based evaluation, performed on the complete HDM05 database. (b) Evaluation based on a selected set of test motions.

and torso for the same motions of the hands and feet (like sitting down a table without moving feet and hands versus other static poses, certain “belly dance” motions, etc.). On the other hand, even less than four sensors may produce reasonable results in certain cases, if the control signal is expressive enough to differentiate between different motion styles and if joint movement is highly correlated (such as for walking motions).

3.6.3 Runtime

The prototypic implementation of our method is computationally relatively costly, as a motion synthesis is required for motion reconstruction. Please note, however, that the search for similar motion segments in large databases is no longer a bottleneck, opposed to existing techniques. As can be seen in Table 3.3, optimization is the most time-consuming step of the whole pipeline. It takes about 380 milliseconds per-frame to reconstruct a motion based on a given stream of control data in our single-threaded MATLAB implementation. For all tests presented in this work the size of the neighborhood used for OLNG and priors/control was $K = 4096$ and $I = 256$, respectively.

Preprocessing	kd-tree construction	1 390 ms
Online reconstruction	kd-tree-based NN search	51 ms
	OLNG update	12 ms
	Energy minimization	380 ms

Table 3.3: Runtimes for the components of our reconstruction pipeline (using DB_1 with $N \approx 6 \cdot 10^5$, $K = 4096$, $I = 256$). The runtimes were measured using single-threaded MATLAB-Code on a Core i7 @ 3.07 GHz. While the runtime of the preprocessing is as given, the runtimes of the online motion reconstruction are averaged over all frames.

3.6.4 Synthesizing a plausible root motion

So far all poses were considered to be normalized with respect to skeleton root position and orientation. However, there are applications that aim to synthesize characters that freely move in space over time, which require a world frame representation of poses. As no information about the actual root movement is given, the required data needs to be synthesized from database samples. We found that using the weighted average of root motions of samples included in Q^t with weights W^t already yields acceptable results in cases where similar motion clips are included in the database. Although more sophisticated approaches are possible in principle, we believe that a substantially more accurate and robust estimate would require additional sensors measuring root orientation and global positions. A simple example indicating that the proposed method generates consistent root motions is given in Figure 3.23. Here, the estimate of the root velocity of the run-walk-run motion sequence presented in the video is shown. The different subsequent phases (continuously accelerating the running speed, turning into a short walking phase, turning into a running again) are clearly reflected by the root velocity. This example does not only show that our rather simple method for estimating root motions is of practical use, but does also demonstrate the capability of our approach to account for temporal variations of motions. In the underlying HDM05 database no range of running motions at various speeds have been captured but only the slight variations of a slow

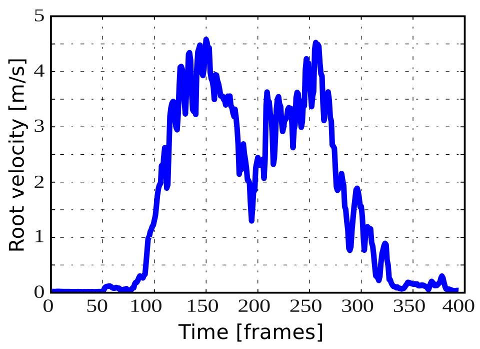


Figure 3.23: Estimated root speed of a run-walk-run motion

running.

3.6.5 Limitations

Since our framework largely relies on similarities in joint acceleration space, effective motion reconstruction is possible only if similar motion sequences induce similar acceleration sequences and vice versa. In other words, sensor readings need to be discriminative enough to differentiate between different motion classes across different subjects while still covering possible variations. If a motion is violating this assumption, a plausible reconstruction is not possible.
An obvious limitation of our method is that occasionally jumps between poses may occur. However, please note that this is a potential issue of any online method that attempts to reconstruct motions based on ambiguous data streams. Another general restriction of the method is that, due to missing positional and orientational information, root motion is only approximate, and that acceptable results are obtained only if motions very similar to the one to be reconstructed are included in the database. Finally, all currently publicly available mocap databases have been designed without our application in mind. As a result, no special care was taken in creating a skeleton representation whose joint frames are consistent with the actual motion. While this might be no issue for joint positions, it substantially affects the usability regarding our method.

3.7 Conclusion

We presented a novel data-driven framework together with a prototypical implementation for reconstructing high-dimensional full-body motions from low-dimensional accelerometer readings. Key components of this framework are a data structure that we referred to as *Online Lazy Neighborhood Graph* (OLNG) and a novel, fully data-driven prior model. We described how the OLNG can be applied for very efficiently retrieving similar poses and motion sequences from a large and heterogeneous motion capture database using a continuous stream of control inputs (accelerometer readings in our case). We demonstrated the robustness of our system in a comprehensive series of experiments. At this, we in particular showed that our novel prior model is able to minimize reconstruction ambiguities while simultaneously accounting for temporal and spatial variations in both the knowledge base and the control signal. Although the control input for our experiments was generated by inertial sensors, we want to emphasize again that we only made use of the integrated accelerometer readings. Thus, our results can be achieved with much smaller and less expensive devices. Our work can be seen as a proof of concept that, although acceleration data of motions contains less information than positional data, not only action recognition but also motion reconstruction is possible in many cases using the data of surprisingly few accelerometers only.

4 Motion reconstruction using a subsequence graph

4.1 Introduction

Optical motion capture systems rely on tracking the positions of a (generally large) number of markers attached to an actor. However, often markers are occluded by surrounding objects, interacting actors, or even by the actor himself. While current systems that commonly rely on a large number of calibrated cameras can handle marker occlusions pretty well up to a certain degree, they inevitably fail if the available information becomes too sparse. In this case, data-driven approaches can be used to fill in the missing information afterwards. In the following, I will present a conceptually very simple yet very effective data-driven approach that fills in the missing information using a small set of motions that are known to be similar to the original performance.

4.1.1 Related work

Various approaches have been suggested for reconstructing full-body motions from low-dimensional control signals. In this work, I have not only introduced two very different reconstruction frameworks myself, but I have also already discussed the state of the art in this field of research; see Section 3.2.

The problem in this chapter, however, is slightly different. While earlier presented approaches rely on a strategically designed control signal, we now have to deal with unintentionally sparse control input. Furthermore, we now deal with positional data rather than with acceleration data. The given task thus relates to the missing marker problem which is commonly encountered in marker-based mocap systems. Rudimentary missing marker recovery solutions, more or less based on interpolation techniques and kinematic information about the underlying skeleton, are provided by commercial mocap systems [Vic11, Pha11, Gia11]. These systems, however, fail if a significant percentage of markers is missing for an extended period of time. Kinematic information together with information about positions of neighboring markers in previous frames is also exploited by Herda et al. [HFP$^+$00]. Kalman filters have been employed to predict the trajectories of missing markers [DU03], but also inevitably fail if markers are missing for an extended period of time, or if they are missing entirely. Also data-driven methods have been proposed [LM06, LMFP10, LC10, BKZW11b, BKZW11a]. None of these, however, deal with extremely sparse measurements, e.g., measurements that provide only 10% of uncorrupted data.

Grochow et al. [GMHP04] describe a style-based inverse kinematics system, and outline how a global, nonlinear dimensionality reduction technique, a Gaussian Process Latent Variable Model [Law04], can be applied to the task of motion capture with missing markers. While the results shown in their accompanying video look very appealing, it is not clear how they are affected by large amounts of marker flickering (markers appearing and disappearing), and how much natural motion detail is lost due to the proposed optimization in low-dimensional pose space.

Conceptually, the method described here can be seen as a specific type of a motion graph, aiming to synthesize a new motion by appropriately rearranging existing clips. Thus, it in particular builds upon ideas presented by Kovar et al. [KGP02], Arikan and Forsyth [AF02], Leet et al. [LCR+02], and Arikan et al. [AFO03]. I furthermore found that some ideas are similar to work by Hsu et al. [HGP04] as well as Basu et al. [BSC05].

4.2 Control signal, knowledge base and skeleton representation

Control signal

The control signal in the here presented problem is in many respects very different from the control signals discussed in Chapters 2 and 3. First of all and most obvious, we are here dealing with positions rather than accelerations. As we have shown earlier, positions are much more discriminative than accelerations, so that this can be seen as an advantage. Unfortunately, it comes with a big handicap: While in the earlier presented frameworks we *deliberately* chose a sparse control signal (in order to reduce costs, setup time, and intrusiveness), the sparsity of the control input here is caused by problems that occurred during the motion capture session and is thus *unintentional.* In other words: We do not control the information that we can access. In practice this leads to the following problems:

- We may have very *local* information only (e.g., only markers on the actor's head).
- The structure and dimensionality of the control signal are constantly *changing* (as markers appear and disappear).
- We may have *gaps* in the control signal, i.e., frames with no information about the performance at all.

Please note that we didn't have to deal with any of these problems in the earlier presented approaches. Specifically, our control signal consists of a sparse set of 3D points that are given in a global reference frame. We do however assume that each of these 3D points is correctly *labeled*, i.e., assigned to a marker, and that we know for each marker its reference joint in the underlying skeleton topology together with its local positional offset. This allows us to regard our control signal as a set of T marker trajectories, with T being the total number of markers attached to the actor. Most of these trajectories will however be incomplete or even empty. From now on I will refer to the motion that produced the control signal (i.e., the motion we seek to reconstruct) as the *original performance*.

Knowledge base

In contrast to the mostly large and heterogeneous motion capture databases employed in the earlier presented approaches, our knowledge base here consists only of a small number of motions that are known to be similar to the original performance. These motions are taken from similar (successful) recordings in the same motion capture session, from reference shots that were done earlier, or even from sequences that were recorded afterwards with the sole purpose of supporting the reconstruction. So why do we then need to reconstruct the original motion at all? The reason is that in general we cannot reproduce the exact performance afterwards. After all, the problems we have to deal with (the sparsity of the control signal) are caused by very demanding (often outdoor) motion capture environments, usually involving interactions with objects and/or simultaneously captured persons. These interactions need to be spatially and temporally aligned. If such reference motions are recorded afterwards, this is done in less demanding (studio) environments. Due to the lower costs, such reference motions are also often performed by different persons, which means that our knowledge base might contain motions of various actors.

Skeleton representation

In our case, the skeleton is represented by a hierarchical (tree-like) structure of 93 so-called *bodies*, one of which is declared as *root*. The root, which is located in the center of the skeleton's pelvis, is the only body that does

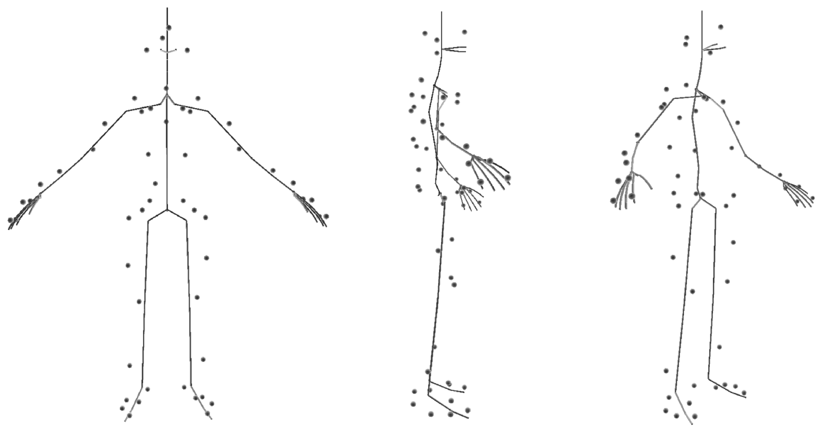


Figure 4.1: Skeleton representation and exemplary marker setup.

neither have a parent nor a spatial expansion (i.e., it has zero length). It is furthermore the only body whose degrees of freedom (dofs) are expressed in world coordinates. While the root has exactly three *rotational* and three *translational* dofs, all other bodies have between zero and three rotational and between zero and three translational (stretching) dofs, all of which are expressed in the coordinate frame of their respective parent body. In total, this adds up to 128 dofs for the complete skeleton representation. However, not all of these *intrinsic* dofs are exposed to the user or the animation system. In fact, an animation is driven by a smaller number of *extrinsic* dofs. While many of these directly map onto intrinsic ones, some affect more than one intrinsic dof. Examples for that are the *spine stretch* dof, that stretches all bones belonging to the spine at once, or the *finger curl* dof, that produces a simultaneous curl of all fingers (on the same hand). The exact mapping onto intrinsic dofs in these cases is determined by pre-defined weights. Motions are captured by (optically) tracking a set of roughly 50-60 passive markers attached to an actor, using a commercial mocap system [Gia11]. All motions were recorded at 60 fps. Figure 4.1 depicts the skeleton representation together with an exemplary marker setup.

4.3 Subsequence graphs

4.3.1 Objective and preliminary considerations

As we have seen, motion synthesis can be performed on windows of frames (cf. Chapter 2) or in a frame-wise manner, as is usually required in an online framework (cf. Chapter 3). While a frame-exact optimization in general is expected to be more flexible and yield higher accuracy, several reasons made me choose the window-based approach for the given problem: First, the problem is an offline task and there is no need to make it online-capable. Second, the control input is likely to be extremely sparse, very local (i.e., limited to a certain body part only) and may even contain gaps, which makes frame-exact reconstruction inherently very difficult. Third, the structure and dimensionality of the control input is constantly changing, which can easily introduce significant amounts of jitter in a frame-wise motion synthesis. Fourth, no matter how close the synthesized motion is to the control input in the end, in a movie production pipeline it will be edited afterwards by an artist anyway. Based on this, we can formulate our objective as follows: Given a sparse set of (labeled) marker trajectories as control input, and a knowledge base consisting of motion data that is known to resemble the original performance (or parts thereof), we want to synthesize a motion that best reflects the temporal and spatial characteristics of the control input, and in particular provides a suitable basis for subsequent (both manual and automatic) motion editing. This means that it is particularly important that the final animation has all the natural details that are contained in motion capture data but are difficult to create by hand. For this reason I decided to first avoid any local optimizations or interpolations that run the risk of washing out natural details, and instead synthesize a motion by stitching together original motion clips.

4.3.2 Outline of the approach

Based on the above considerations, a very simple approach can be outlined as follows:

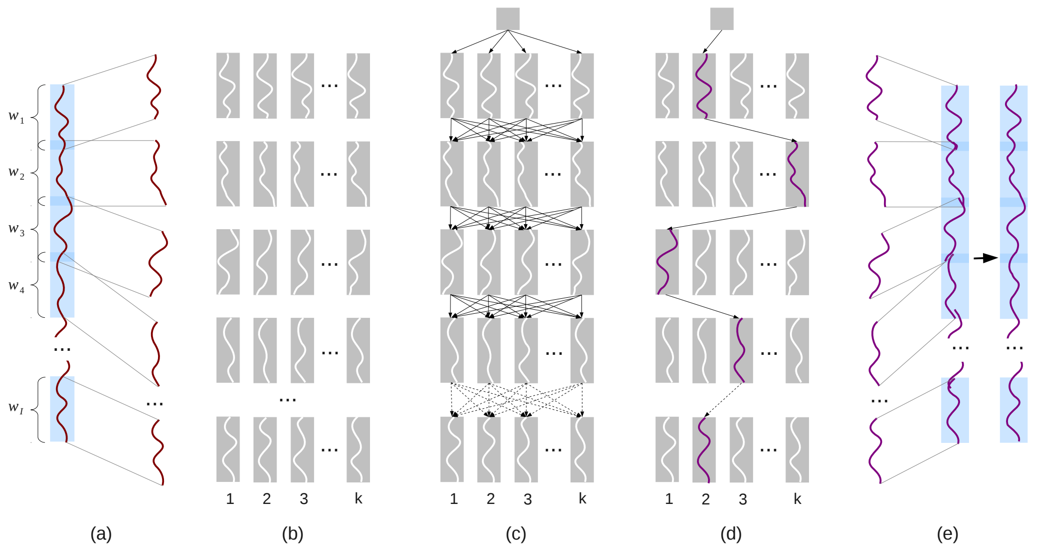


Figure 4.2: Outline of our approach to motion reconstruction using a subsequence graph (SSG).

1. Divide the control signal into equally-sized time windows.
2. For each window find the unique (temporally and spatially aligned) motion subsequence in the knowledge base that best matches the control signal within that window.
3. Concatenate all retrieved subsequences to get the resulting animation.

This approach, however, does not produce satisfying results. The main reason lies in the sparsity of the control signal, which causes subsequences to be reported that locally match the control signal but do not fit into the overall motion. As a result, subsequences cannot be meaningfully concatenated and transitions look very unnatural. Very simple modifications however already largely improve the outcome, and lead to the following procedure:

1. Divide the control signal into I windows w_i of size N frames, with consecutive windows w_i, w_{i+1} overlapping by o frames (Figure 4.2 (a)).
2. For each window w_i find the k (temporally and spatially aligned) motion subsequences in the knowledge base that best match the control signal within that window (Figure 4.2 (b)).

3. Build a graph structure (in the following referred to as *subsequence graph* (SSG)), where:
   - each of the $I \cdot k$ retrieved subsequences is a node,
   - subsequences belonging to consecutive time windows w_i, w_{i+1} are connected by a directed edge pointing from w_i to w_{i+1},
   - costs are assigned to nodes based on their distance to the control signal,
   - costs are assigned to edges based on the similarity of the two corresponding subsequences within the overlapping region (transition costs),
   - a source node is inserted that connects to the k subsequences that belong to the very first time window.

   Note that by construction the subsequence graph is (similar to the LNG and OLNG presented in Section 3.4) topologically ordered, directed, and acyclic, and thus shortest paths can be found in linear time. The structure of the subsequence graph is illustrated in Figure 4.2 (c).

4. Find the path with lowest costs that starts at the source node and ends at a subsequence that corresponds to the very last time window w_I (Figure 4.2 (d)).

5. Concatenate all subsequences that belong to this path by cross-fading them within their overlapping regions (Figure 4.2 (e)).

4.3.3 Subsequence retrieval

For constructing the subsequence graph we need to identify for each window of the control signal a set of similar motion subsequences in the knowledge base. While this is done using *subsequence dynamic time warping* (subsequence DTW) similar to the method described by Müller in Section 4.4 of his book *Information Retrieval for Music and Motion* [Mül07], I want to point out a small but important detail regarding my implementation. For details about the general procedure please refer to Müller's book.

While it identifies optimal warping paths with respect to their total costs, the algorithm for retrieving subsequences from a knowledge base as described by Müller favors *short* paths. It is thus biased towards decelerating motions in the knowledge base in order to align them with the control signal. In my implementation I counterbalanced this preference by introducing *local weights* in the computation of the accumulated cost matrix, similar to the variation of DTW Müller describes in Section 4.2.2 of his book. Due to differences in the initialization of this matrix, its structure in the subsequence retrieval scenario, however, slightly differs from that in classical DTW, which is why weights need special attention. Specifically, when using the classical DTW steps (horizontal, vertical and diagonal, cf. Figure 3.10 (a)) in a subsequence retrieval scenario, the diagonal *and* the vertical (the one that produces a deceleration in the retrieved subsequence) step need to be appropriately penalized by local weights. I will now formalize and concretize this statement following the notation used by Müller: Let M be the number of frames in the knowledge base, N the number of frames in our query sequence (a window taken from the control signal), $C \in \mathbb{R}^{N\times M}$ the (local) cost matrix, and $n \in [1 \dots N]$ and $m \in [1 \dots M]$ row and column indices, respectively. The accumulated cost matrix $D \in \mathbb{R}^{N\times M}$ for retrieval of subsequences without preferring any alignment is then recursively defined:

$$D(n,m) = \begin{cases} C(n,m) & \text{if } n = 1 \\ \sum_{k=1}^{n} C(k,m) & \text{if } m = 1 \\ \min \begin{cases} D(n-1,m-1) + 2 \cdot C(n,m) \\ D(n-1,m) + 2 \cdot C(n,m) \\ D(n,m-1) + C(n,m) \end{cases} & \text{otherwise} \end{cases} \tag{4.1}$$

In my implementation I furthermore chose the step size conditions to constrain the slope of warping paths to the bounds $\frac{1}{2}$ and 2 (cf. Figure 3.10 (b)).

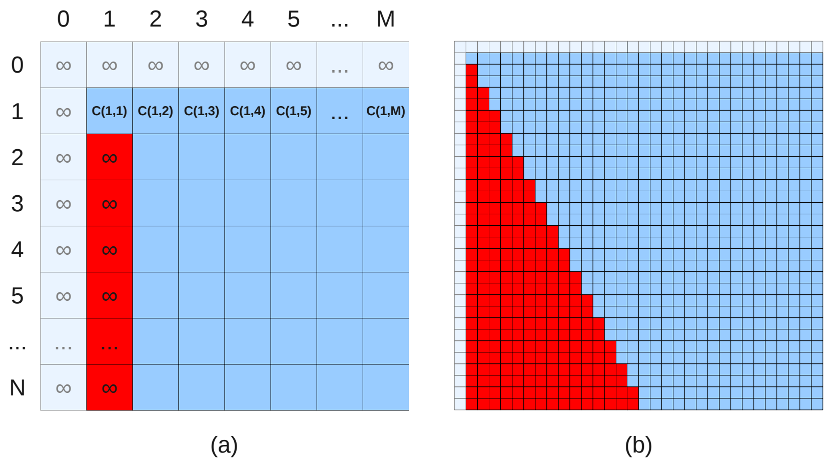


Figure 4.3: (a) Initialization of the accumulated cost matrix for subsequence DTW using step size conditions that constrain the slope of warping paths to the bounds $\frac{1}{2}$ and 2 (cf. Figure 3.10 (b)). (b) Defining $D(n,m) = \infty$ if $m = 1$ and $n > 1$ will propagate infinity values to the red area and effectively restrict the search for optimal warping paths to valid regions.

With these step size conditions, the accumulated cost matrix computes as:

$$D(n,m) = \begin{cases} C(n,m) & \text{if } n = 1 \\ \infty & \text{if } m = 1 \text{ and } n > 1 \\ \min \begin{cases} D(n-1,m-1) + C(n,m) \\ D(n-1,m-2) + C(n,m) \\ D(n-2,m-1) + 2 \cdot C(n,m) \end{cases} & \text{otherwise} \end{cases} \tag{4.2}$$

By additionally defining $D(n,m) := \infty$ if $n = 0$ or $m = 0$, all boundary conditions are handled. Figure 4.3 gives a more illustrative interpretation of the initialization of D, representing the first two cases in Equation 4.2 together with the boundary definition. Given this accumulated cost matrix D, the k best subsequences are retrieved following the procedure described by Müller, also discarding subsequences that only differ by a slight shift from already retrieved ones. Please note that the introduced weights also have to be considered when backtracking the optimal warping paths in D.

Distance measure

The (local) cost matrix C that is needed for the computation of the accumulated cost matrix D is obtained by evaluating a local cost or distance measure comparing each frame of the control signal to each frame of the knowledge base. Two things make it difficult to find an effective as well as efficient distance measure: First, marker positions are given in a global world coordinate frame, which means that they are not directly comparable, and there is no obvious way to define a local reference system. Second, most markers are only occasionally visible, which means that the available information and thus the dimensionality of the control signal varies over time. This makes it basically impossible to define a consistent (pre-computable) feature set among frames, and thus forbids the use of efficient data structures like, for example, a kd-tree.[1]

For these reasons I chose the point cloud distance measure proposed by Kovar et al. [KGP02], a variation of which I had already used in the multilinear framework for comparing point clouds of accelerations. It defines the distance between two point clouds as the minimal (weighted) sum of squared distances between corresponding points, given that a rigid 2D transformation consisting of a rotation about the vertical axis and a translation in the ground plane may be applied to one of the point clouds. As a result from this definition, this distance measure is not only invariant under horizontal translation and vertical rotation, but also provides us with the optimal transformation that aligns one point cloud with the other. Furthermore, it flexibly adapts to the variations in the dimensionality of the control signal.

Based on this distance measure, each entry of the local cost matrix computes as:

$$C(n,m) = \min_{\theta,x_0,z_0} \left(\sum_j \lambda_j \| p_j^n - T_{\theta,x_0,z_0} p_j'^m \|^2 \right), \tag{4.3}$$

where p_j^n is a point in the cloud defined at frame n of the control signal, $p_j'^m$

[1] Baumann et al. [BKZW11a] build a new kd-tree for each motion cleaning process using a subset of *viable* markers. In our case, however, the control signal is in general too sparse to provide reliable marker subsets. Also, by not restricting us to pre-computable feature sets, we can make full use of *all* available information.

is a corresponding point in the cloud defined at frame m of the knowledge base, λ_j is a weight assigned to the pair $(p_j^n, p_j'^m)$, and T_{θ,x_0,z_0} is a linear transformation that rotates a point by θ degrees about the (vertical) y-axis and then translates it by x_0, z_0. Similar to what Kovar et al. suggest in their work, I also defined point clouds on windows of frames, thereby effectively incorporating derivative information into the metric.[2] Specifically, I used a window of a quarter of a second (i.e., 15 frames at 60 fps), centered at the frame under consideration, with weights tapering off to both ends according to a Gaussian function.

4.3.4 Graph construction

Once the k best subsequences are identified for each window $w_i, 1 \leq i \leq I$, the subsequence graph (SSG) is built as outlined in Section 4.3.2. Construction of the SSG is very straightforward (see also Figure 4.2): All retrieved subsequences are considered as nodes, nodes belonging to subsequent time windows are connected by directed edges, and costs are assigned to both nodes and edges. The cost assigned to a node is directly given by its respective DTW distance (measuring the effort of aligning the control signal to the corresponding subsequence in the knowledge base), the cost assigned to an edge is computed based on the similarity of the two involved subsequences within their overlapping region. The latter is used as an indicator of how well two subsequences can be concatenated and is also referred to as *transition cost*. More precisely, the transition costs $c_t(S, S')$ of two subsequences S and S' are defined as:

$$c_t(S, S') = \sum_j \left(\| T_{\theta,x_0,z_0} p_j - T'_{\theta,x_0,z_0} p'_j \|^2 \right), \tag{4.4}$$

where p and p' are 3D points in the overlapping regions of the subsequences S and S', and T and T' are transformations that optimally align the two subsequences with their corresponding windows of the control signal. For

[2]Please note that these windows are different from the (possibly overlapping) windows used to subdivide the control signal.

the computation of T and T' we again use the point cloud distance measure defined in Equation 4.3, with $\lambda_j := 1$ for all j. Please note that we do not compute optimal alignments between subsequences, but instead compute for each subsequence its optimal alignment with the control signal. This ensures that the resulting motion is not only locally but also globally aligned with the control signal.

By adding an additional source node that connects to all subsequences corresponding to the very first time window w_1, we turn our motion reconstruction task into a shortest path problem. Now we just have to search for the path with lowest accumulated cost that leads from the source node to a subsequence corresponding to the very last time window w_I. The costs of a path are given by summing up all costs of related nodes and edges, and an additional parameter is introduced to control the relative importance of both. As mentioned earlier, due to properties of the SSG, an optimal path can be found in linear time (linear in the number of nodes). Note, however, that the number of nodes is fairly small (in fact, generally orders of magnitude smaller than in the OLNG presented in Chapter 3), and the time needed for the path search is negligible compared to the computation of the subsequences.

4.3.5 Motion synthesis

Given a path in the SSG, all that is left to do to yield a reconstruction of the original performance is to appropriately concatenate the identified subsequences. This is done by simply cross-fading their exposed dofs within their overlapping regions according to

$$q_{i,i+1}(t) = \alpha(t) \cdot q_i(t) + (1 - \alpha(t)) \cdot q_{i+1}(t). \tag{4.5}$$

Here, q_i and q_{i+1} are the dof values of two subsequences S_i and S_{i+1} to be cross-faded, $q_{i,i+1}$ denotes the dof values of the cross-fading result, t indicates the frame index relative to the beginning of the overlapping region (i.e., $1 \leq t \leq o$ within this region), and $\alpha(t)$ is a C^1-continuous transition function with $\alpha(t) = 1$ for $t \leq 1$ and $\alpha(t) = 0$ for $t \geq o$.

Gaps

As mentioned earlier, the control signal may contain gaps, i.e., time windows without any information about the original performance at all. Most of these gaps, however, will be very small (often even consist of a single frame only), and do not need any special attention. In fact, our implementation simply ignores all gaps that are smaller than half the size of the windows used to subdivide the control signal. Such small gaps are then automatically "bridged" by a single subsequence or a concatenation of two of them. Larger gaps, however, have to be treated in a special way during the construction of the SSG. The basic idea to do this is as follows: If a (large) gap is encountered,

1. find the k best subsequences that *end* immediately before as well as the k best subsequences that *start* immediately after the gap,
2. for each of these $2k$ subsequences determine a warping factor based on their warping paths[3],
3. use these warping factors together with the knowledge base to extend the subsequences such that they (temporally) span the entire gap[4],
4. add the extended subsequences as nodes (together with appropriate edges and related costs) to the subsequence graph.

The resulting graph is a valid SSG and can be used for motion reconstruction as described earlier.

4.4 Results

The following *exemplary* evaluation of the described framework is based on *synthetic* data, which means that I simulated sparse control signals by discarding available information from clean motion data. Specifically, our

[3]The warping factor is simply a number that defines the amount of acceleration or deceleration. In our implementation, using the slope-constraining step size conditions described in Section 4.3.3, the warping factor is a number between $\frac{1}{2}$ and 2.

[4]Subsequences "behind" the gap have to be extended backward in time.

knowledge base consisted of about 75 seconds (roughly 4500 frames at 60 fps) of locomotion data (including walking, jogging, and running motions, as well as respective transitions), and control signals of different sparsity were simulated based on a motion of about 13 seconds (roughly 780 frames at 60 fps). The motion that was used to simulate the control signal was never included in the knowledge base. In fact, the framework ensures that if the original motion was included in the knowledge base, the synthesized motion would always be an identical copy of it, no matter how sparse the control signal was chosen.

In a first test, I used the trajectories of four markers, which were attached to the hands and feet of an actor, as control signal. Figure 4.4 (a) shows the corresponding local cost matrix, overlayed with white warping paths representing subsequences retrieved by the subsequence search, and red warping paths representing subsequences that belong to the optimal path and are thus part of the reconstructed motion. In this example, the size of each time window was set to $N = 20$ frames ($\frac{1}{3}$ second), the overlap between subsequent time windows to $o = 10$ frames ($\frac{1}{6}$ second), and the number of subsequences per time window to $k = 8$. The resulting motion is shown in the accompanying video.

In a second test, I used the same marker trajectories as control signal, but introduced gaps by discarding all marker data in randomly chosen time intervals of varying lengths. Figure 4.4 (b) shows the local cost matrix, with gaps in the control signal visualized as dark blue bands. In this example, the smallest gap was 5 frames wide, and the largest gap was 80 frames wide. The accompanying video shows the motion synthesized based on the depicted cost matrix. For this example, I used the same settings ($N = 20, o = 10, k = 8$) as in the previously described case without gaps.

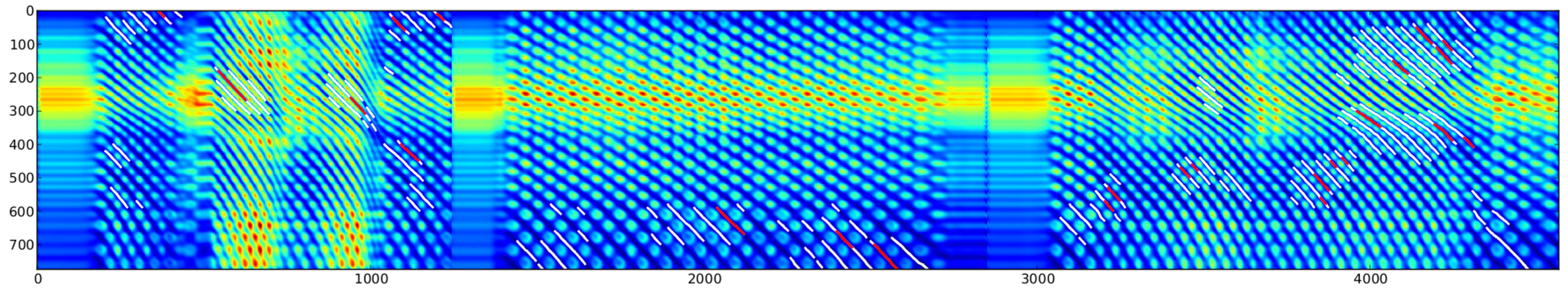
I performed more tests in order to evaluate the framework regarding its effectiveness with respect to not only very sparse but also very local data. In one of them I used the sparsest possible control input, consisting of a single marker only, which was attached to a foot. The local cost matrix together with warping paths is depicted in Figure 4.5 (a), the synthesized motion is again shown in the video. A similar result was obtained by using the trajectory of a single marker attached to a hand as control input. It has to be stated, however, that the trajectories of feet and hands are likely the most descriptive ones for this kind of locomotion, and that we in general cannot rely on having a such descriptive control input. Figure 4.5 (b) shows the local cost matrix for a less descriptive control signal, defined by a small set of markers attached to the head and upper back of the actor only. The blurriness of the matrix illustrates the lack of discriminatory power: The local distance measure largely fails to discriminate poses. Although the resulting animation clearly reflects these issues, the outcome, which is shown in the video, is still plausible. The result was obtained choosing $N = 15, o = 6, k = 16$, and additionally increasing the transition weights to get smoother transitions.

I was also able to run a test on real production data. In the given scene, which was shot in a very demanding outdoor setting, the actor's body was so heavily occluded by surrounding objects that only a few markers on his head and his upper back could be tracked.[5] Furthermore, all these markers were constantly appearing and disappearing. The knowledge base consisted of reference motions that had been recorded afterwards (by the same actor) in the mocap studio. The animation produced by my subsequence graph framework was generally considered as a very suitable basis for subsequent motion editing.

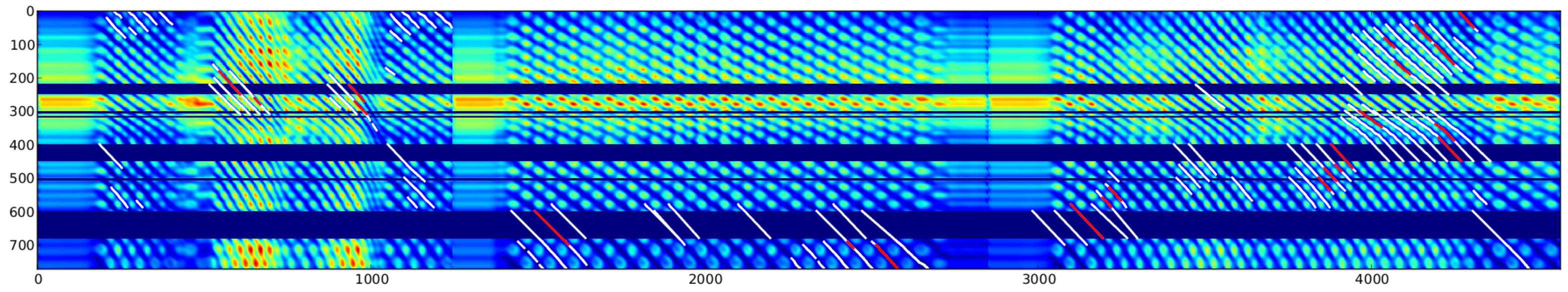
4.5 Conclusion and future work

I presented a conceptually very simple yet very effective method for reconstructing full-body motions from extremely sparse, position-based control signals. Although the resulting animations do not meet the quality requirements of a movie production, they provide a suitable basis for subsequent motion editing. Since existing motion clips are only played back (and cross-faded), the results strongly depend on the motions in the knowledge base,

[5]While the structure of the control signal in the previously described experiment was intentionally chosen to resemble the structure of the control signal in the production shot, the latter was much more repetitive and thus actually less demanding than the synthetic data.

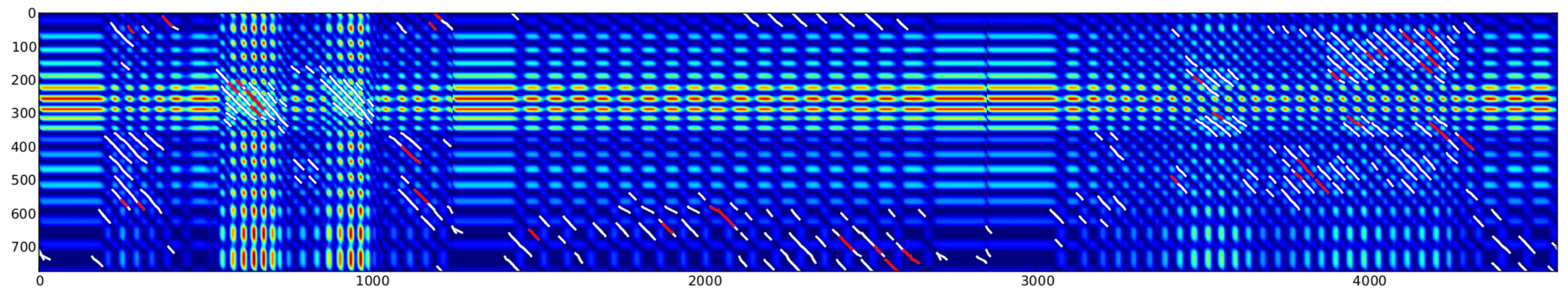


(a) Using the trajectories of four markers attached to the hands and feet of the actor as control signal.



(b) Introducing gaps into the control signal.

Figure 4.4: The vertical axis gives the frame of the control signal, the horizontal axis the frame of the knowledge base (with all motions in the knowledge base "horizontally stacked"). Blue encodes small distances (very similar), red encodes high distances (very unsimilar). White paths represent the best (time-warped) subsequences found for each time window, red paths represent the subsequences that are part of the synthesized motion. The dark blue bands in (b) represent gaps in the control signal.


(a) Using the trajectory of a single marker attached to a foot of the actor as control signal.

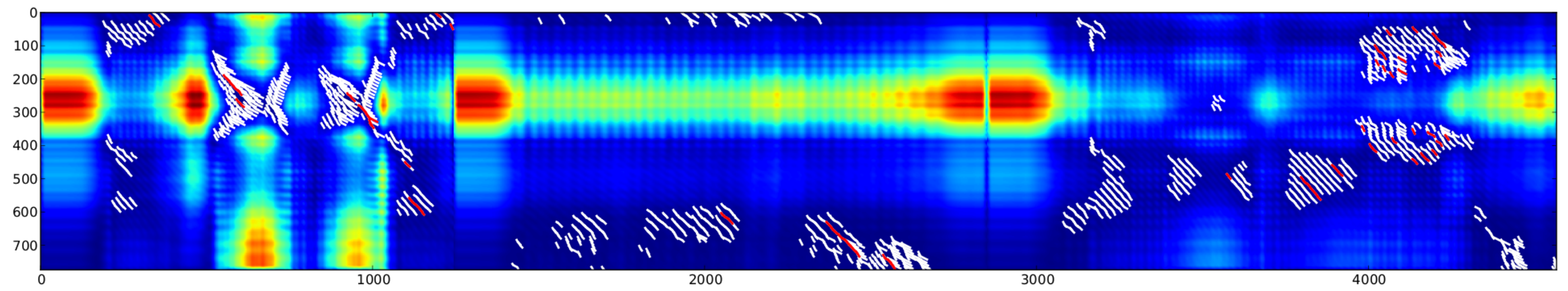

(b) Using a small set of markers attached to the head and back of the actor as control signal.

Figure 4.5: The vertical axis gives the frame of the control signal, the horizontal axis the frame of the knowledge base (with all motions in the knowledge base "horizontally stacked"). Blue encodes small distances (very similar), red encodes high distances (very unsimilar). White paths represent the best (time-warped) subsequences found for each time window, red paths represent the subsequences that are part of the synthesized motion. The blurriness of (b) compared to (a) indicates that the trajectory of a foot is much more discriminative for locomotion than the motion of the head or back.

and high accuracy (in terms of matching the control signal) can generally only be achieved by adopting or additionally applying other techniques.

One way to achieve such accuracy is to apply an inverse kinematics (IK) solver to the reconstructed motion in a post-processing step. In fact, I have already performed first tests using a Space-Time IK (STIK) solver, which simultaneously solved for all frames in the entire motion. Specifically, this solver tried to make the motion match the control signal (using soft constraints) while globally minimizing its joint accelerations and the difference to the original motion (which is, in our case, the motion produced in the reconstruction step). The accompanying video also shows a result of this combined reconstruction effort.

The simple motion blending scheme used in my implementation sometimes causes footskating artifacts. Thus, a footskate cleanup, as for example described by Kovar et al. [KSG02], would be another useful post-process.

Unfortunately, I was not able to perform a comprehensive systematic evaluation of the described method. Instead I presented an exemplary evaluation. In order to refine the method, however, I consider a more thorough analysis as an important part of future work.

I can think of several improvements to the method. For example, I presume that there are better ways to segment the control signal. Instead of fragmenting it into equally-sized windows, one could for instance cut it at extremal points of its acceleration curve. Early experiments also indicated that adaptive window sizes can be dynamically determined during the computation of the global cost matrix. The gap filling is based on a very simple, linear motion extrapolation. For large gaps, one idea would be to add additional nodes (subsequences) to the graph, candidates for which could be easily retrieved by searching for subsequences similar to the extrapolated ones (or even subsequences thereof).

Another idea would be to use the outcome of the described reconstruction framework as (additional) input for a second data-driven synthesis step. The outcome could then be seen as a first guess of the motion, which, for example, would allow for frame-wise nearest-neighbor searches in high-dimensional pose-space (or arbitrary, lower-dimensional subspaces thereof).

Basically, I consider the conceptual simplicity of the described method appealing. The fact that only very few and intuitive parameters have to be specified in order to quickly obtain a reconstruction result also seems to be appreciated by potential users in a production environment. As discussed earlier, however, for producing high-quality motions additional techniques have to be applied. While first steps towards this have already been taken, it is part of future work to investigate how different strategies can be optimally combined.

5 Conclusion and future work

In this thesis I presented three different data-driven approaches for reconstructing human motions based on very low-dimensional control signals.

I first discussed a multi-linear framework, where motion sequences were arranged in high-dimensional tensors and decomposed using a higher-order singular value decomposition (HOSVD) to yield an intuitive interface for synthesizing new motions. The control input for this framework was provided by a small set of inertial sensors. By simulating the readings of *virtual* sensors for synthesized motions, we were able to formulate motion reconstruction as an offline optimization problem, aiming to find a linear combination of original motions whose simulated sensor readings best match the actual sensor readings. Given that a pre-classification of the control signal was available, this approach allowed us to create naturally looking animations driven by an extremely sparse control signal. Due to the need for such a pre-classification and several other discussed limitations, the multi-linear framework, however, was not able to meet the requirements of a more general and less restricted motion capture scenario. Although several extensions to the implementation of the multi-linear framework are possible, I consider some of its limitations as too fundamental for the task of motion reconstruction and thus do not plan to pursue this particular approach any further.

The second approach that I presented in this thesis was inspired by the performance animation system described by Chai and Hodgins [CH05]. Here, our control input was provided by a small set of accelerometers, and motion reconstruction was formulated as an *online* optimization problem, at each point in time trying to maximize the likelihood of the synthesized pose with respect to a local statistical model of poses that was learned at runtime. The main challenge in replacing the position-based control signal used in the original system by an acceleration-based one was to find a reliable mapping from the low-dimensional space of accelerations to the high-dimensional space of poses. I addressed this problem by introducing a dynamic data structure called *online lazy neighborhood graph* (OLNG), that very efficiently exploits temporal coherence of motions in order to disambiguate the mapping. The OLNG is a very general technique that can be used in basically every online retrieval scenario where matching sequences have to be identified based on a continuous control signal. Krüger et al. [KZBW11] for instance applied the OLNG for retrieving short motion sequences from a motion capture database based on a stream of skeletal joint angle data. These sequences were then used to enhance an existing animation. Motion sequences retrieved from a knowledge base in general do not only provide a priori knowledge about their history, but also contain empirical information about their possible future evolution. Thus, the OLNG also directly allows for real-time motion *anticipation*, which I expect to be useful in a motion capture scenario that has to deal with severe occlusions, or also for collision avoidance in robotics applications. As another contribution, I presented a novel, fully data-driven prior model that effectively minimized reconstruction ambiguities even in challenging cases while simultaneously accounting for both temporal and spatial variations. This model also lays the foundation for a different animation task: Although technically very different from the approach suggested by Pullen and Bregler [PB02], it can be used for synthesizing specified missing degrees of freedom in key-frame animations (motion completion), or for enhancing key-frame animations with details of motions extracted from a database (motion texturing). In fact, it has already been successfully applied recently to both disciplines [BKZW11a, KZBW11, BKZW11b]. A particularly appealing topic of future research would also be to add a *physics*-prior to the optimization model. Such a prior would allow us to exploit yet another aspect of available prior knowledge, and to eventually constrain the outcome to a physically valid motion. For most applications it will be essential to have a real-time capable implementation, which is not given at this point. Overall, I consider the presented ideas a valuable contribution towards robust, (cost-)efficient, and non-intrusive reconstruction of full-body motions, which various disciplines in (bio-)medical engineering, sports science, and game development may benefit from.

Finally, I described a method for reconstructing motions on the basis of very sparse marker trajectories. In particular, I introduced a novel motion graph structure that I referred to as *subsequence graph* (SSG), and showed how it can be used to synthesize naturally looking full-body motions that globally match an extremely sparse, constantly changing, and possibly very local control input. Using a conceptually very simple motion prediction scheme, the SSG was also able to produce meaningful animations in the presence of gaps in the control signal, i.e., time windows in which we did not have any information about the original performance at all. Although the resulting animations did not fulfill movie quality requirements, they were generally regarded as a very solid basis for subsequent motion editing. Given the incompleteness of the measurements, such a basis was furthermore considered to be much more cumbersome and time-consuming to obtain by hand. How to optimally combine the practicability and robustness of subsequence graphs with the flexibility and accuracy of (local) motion optimization techniques is a topic of future research.

Bibliography

[Adi11] ADIDAS: *Adidas MiCoach Homepage*. http://www.adidas.com/us/micoach/, Accessed December 31th, 2011. http://www.adidas.com/us/micoach/. Version: 2011

[AF02] ARIKAN, Okan ; FORSYTH, D. A.: Interactive motion generation from examples. In: *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*. San Antonio, Texas : ACM Press, 2002. – ISBN 1–58113–521–1, S. 483–490

[AFO03] ARIKAN, Okan ; FORSYTH, David A. ; O'BRIEN, James F.: Motion synthesis from annotations. In: *ACM Trans. Graph.* 22 (2003), Nr. 3, S. 402–408

[AI08] ANDONI, Alexandr ; INDYK, Piotr: Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In: *Commun. ACM* 51 (2008), Nr. 1, S. 117–122. http://dx.doi.org/10.1145/1327452.1327494. – DOI 10.1145/1327452.1327494. – ISSN 0001–0782

[BHG93] BADLER, Norman I. ; HOLLICK, Michael J. ; GRANIERI, John P.: Real-Time Control of a Virtual Human Using Minimal Sensors. In: *Presence: Teleoperators and Virtual Environments* (1993), Nr. 1, S. 82–86

[BKZW11a] BAUMANN, Jan ; KRÜGER, Björn ; ZINKE, Arno ; WEBER, Andreas: Data-Driven Completion of Motion Capture Data.

In: *Workshop on Virtual Reality Interaction and Physical Simulation (VRIPHYS)*, Eurographics Association, Dezember 2011

[BKZW11b] Baumann, Jan ; Krüger, Björn ; Zinke, Arno ; Weber, Andreas: Filling Long-Time Gaps of Motion Capture Data. In: *SCA'11: Poster Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2011

[BSC05] Basu, Suddha ; Shanbhag, Shrinath ; Chandran, Sharat: Search and transitioning for motion captured sequences. In: *Proceedings of the ACM symposium on Virtual reality software and technology*. New York, NY, USA : ACM, 2005 (VRST '05). – ISBN 1–59593–098–1, 220–223

[BSP+04] Barbič, Jernej ; Safonova, Alla ; Pan, Jia-Yu ; Faloutsos, Christos ; Hodgins, Jessica K. ; Pollard, Nancy S.: Segmenting motion capture data into distinct behaviors. In: Heidrich, Wolfgang (Hrsg.) ; Balakrishnan, Ravin (Hrsg.): *Proceedings of the Graphics Interface 2004 Conference*, Canadian Human-Computer Communications Society, 2004. – ISBN 1–56881–227–2, 185–194

[Car04] Carnegie Mellon University Graphics Lab: *CMU Motion Capture Database*. http://mocap.cs.cmu.edu. Version: 2004. – mocap.cs.cmu.edu

[CH05] Chai, Jinxiang ; Hodgins, Jessica K.: Performance animation from low-dimensional control signals. In: *ACM Trans. Graph.* 24 (2005), July, 686–696. http://dx.doi.org/http://doi.acm.org/10.1145/1073204.1073248. – DOI http://doi.acm.org/10.1145/1073204.1073248. – ISSN 0730–0301

[CHP07] Cooper, Seth ; Hertzmann, Aaron ; Popović, Zoran: Active learning for real-time motion controllers. In: *ACM Trans. Graph.* 26 (2007), Nr. 3, S. 5. http://dx.doi.

org/http://doi.acm.org/10.1145/1276377.1276384. – DOI http://doi.acm.org/10.1145/1276377.1276384. – ISSN 0730–0301

[DU03] Dorfmüller-Ulhaas, Klaus: Robust Optical User Motion Tracking Using a Kalman Filter / Universitätsbibliothek der Universität Augsburg. Universitätsstr. 22, 86159 Augsburg, 2003. – Forschungsbericht

[DYP03] Dontcheva, Mira ; Yngve, Gary ; Popović, Zoran: Layered acting for character animation. In: *ACM Trans. Graph.* 22 (2003), July, 409–416. http://dx.doi.org/http://doi.acm.org/10.1145/882262.882285. – DOI http://doi.acm.org/10.1145/882262.882285. – ISSN 0730–0301

[FF05] Forbes, Kate ; Fiume, Eugene: An efficient search algorithm for motion data using weighted PCA. In: *Proc. 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ACM Press, 2005. – ISBN 1–7695–2270–X, S. 67–76

[FKY08] Feng, Wei-Wen ; Kim, Byung-Uck ; Yu, Yizhou: Real-time data driven deformation using kernel canonical correlation analysis. In: *ACM Trans. Graph.* 27 (2008), August, 91:1–91:9. http://dx.doi.org/http://doi.acm.org/10.1145/1360612.1360690. – DOI http://doi.acm.org/10.1145/1360612.1360690. – ISSN 0730–0301

[GBT04] Glardon, Pascal ; Boulic, Ronan ; Thalmann, Daniel: PCA-Based Walking Engine Using Motion Capture Data. In: *CGI '04: Proceedings of the Computer Graphics International (CGI'04)*. Washington, DC, USA : IEEE Computer Society, 2004. – ISBN 0–7695–2171–1, S. 292–298

[Gia11] GiantStudios: *Giant Studios Homepage*. http://www.

giantstudios.com/, Accessed December 25th, 2011. http://www.giantstudios.com/. Version: 2011

[GMHP04] Grochow, Keith ; Martin, Steven L. ; Hertzmann, Aaron ; Popović, Zoran: Style-based inverse kinematics. In: *ACM Transactions on Graphics* 23 (2004), Nr. 3, 522–531. http://doi.acm.org/10.1145/1015706.1015755. – ISSN 0730–0301. – SIGGRAPH 2004

[Gol09] Golla, Tim: *Multilineare Bewegungssynthese anhand niedrigdimensionaler Vorgaben*, Rheinische Friedrich-Wilhelms-Universität Bonn, Diploma Thesis, 2009. – (Engl. translation: Multi-linear Motion Synthesis Based on Low-Dimensional User Specifications)

[HFP+00] Herda, L. ; Fua, P. ; Plänkers, R. ; D. ; Boulic, R. ; Thalmann, D.: Skeleton-Based Motion Capture for Robust Reconstruction of Human Motion. In: *Computer Animation*. Philadelphia, PA, may 2000

[HG07] Heck, Rachel ; Gleicher, Michael: Parametric motion graphs. In: *I3D '07: Proceedings of the 2007 symposium on Interactive 3D graphics and games*. New York, NY, USA : ACM Press, 2007. – ISBN 978–1–59593–628–8, S. 129–136

[HGP04] Hsu, Eugene ; Gentry, Sommer ; Popović, Jovan: Example-based control of human motion. In: *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*. Aire-la-Ville, Switzerland, Switzerland : Eurographics Association, 2004. – ISBN 3–905673–14–2, S. 69–77

[HMT+11] Helten, Thomas ; Müller, Meinard ; Tautges, Jochen ; Weber, Andreas ; Seidel, Hans-Peter: In: *Proceedings of the 33rd Annual Symposium of the German Association for Pattern Recognition (DAGM)*, Springer-Verlag Berlin Heidelberg, 9 2011, S. 61–70

[KB09] Kolda, Tamara G. ; Bader, Brett W.: Tensor Decompositions and Applications. In: *SIAM Review* 51 (2009), September, Nr. 3, S. 455–500. http://dx.doi.org/10.1137/07070111X. – DOI 10.1137/07070111X

[KCHO10] Kelly, Philip ; Conaire, Ciarán Ó ; Hodgins, Jessica ; O'Conner, Noel E.: Human motion reconstruction using wearable accelerometers (Poster). In: *ACM SIGGRAPH / Eurographics Symposium on Computer Animation (SCA)*, 2010

[KG03] Kovar, Lucas ; Gleicher, Michael: Flexible Automatic Motion Blending with Registration Curves. In: Breen, D. (Hrsg.) ; Lin, M. (Hrsg.): *Eurographics/SIGGRAPH Symposium on Computer Animation*, Eurographics Association, 2003, 214-224

[KGP02] Kovar, Lucas ; Gleicher, Michael ; Pighin, Frédéric: Motion Graphs. In: *ACM Transactions on Graphics* 21 (2002), Nr. 3, 473–482. http://doi.acm.org/10.1145/566654.566605. – ISSN 0730–0301. – SIGGRAPH 2002

[KPZ+04] Keogh, Eamonn ; Palpanas, Themistoklis ; Zordan, Victor B. ; Gunopulos, Dimitrios ; Cardle, Marc: Indexing large human-motion databases. In: *VLDB '04: Proceedings of the Thirtieth international conference on Very large data bases*, VLDB Endowment, 2004. – ISBN 0–12–088469–0, S. 780–791

[Krü11] Krüger, Björn: *Synthesizing Human Motions*, Rheinische Friedrich-Wilhelms-Universität Bonn, Diss., 2011

[KSG02] Kovar, Lucas ; Schreiner, John ; Gleicher, Michael: Footskate cleanup for motion capture editing. In: *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*. New York, NY, USA : ACM, 2002. – ISBN 1–58113–573–4, S. 97–104

[KTMW08] Krüger, Björn ; Tautges, Jochen ; Müller, M. ; Weber, Andreas: Multi-Mode Tensor Representation of Motion Data.

In: *Journal of Virtual Reality and Broadcasting* 5 (2008), Juli, Nr. 5. – ISSN 1860–2037

[KTW07] Krüger, Björn ; Tautges, Jochen ; Weber, Andreas: Multi-Mode Representation of Motion Data. In: *The 2nd International Conference on Computer Graphics Theory and Applications (GRAPP 2007), Volume AS/IE*, INSTICC Press, März 2007. – ISBN 978–972–8865–72–6, S. 21–29

[KTWZ10] Krüger, Björn ; Tautges, Jochen ; Weber, Andreas ; Zinke, Arno: Fast Local and Global Similarity Searches in Large Motion Capture Databases. In: *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Aire-la-Ville, Switzerland, Switzerland : Eurographics Association, Juli 2010 (SCA '10), 1–10

[KZBW11] Krüger, Björn ; Zinke, Arno ; Baumann, Jan ; Weber, Andreas: Data-Driven Texturing of Human Motions. In: *ACM SIGGRAPH ASIA 2011: Posters*, 2011

[Law04] Lawrence, Neil D.: Gaussian process latent variable models for visualisation of high dimensional data. In: *In NIPS*, 2004, S. 2004

[LC10] Lou, Hui ; Chai, Jinxiang: Example-Based Human Motion Denoising. In: *IEEE Transactions on Visualization and Computer Graphics* 16 (2010), S. 870–879. `http://dx.doi.org/http://doi.ieeecomputersociety.org/10.1109/TVCG.2010.23`. – DOI http://doi.ieeecomputersociety.org/10.1109/TVCG.2010.23. – ISSN 1077–2626

[LCR$^+$02] Lee, Jehee ; Chai, Jinxiang ; Reitsma, Paul S. A. ; Hodgins, Jessica K. ; Pollard, Nancy S.: Interactive control of avatars animated with human motion data. In: *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics* *and interactive techniques.* San Antonio, Texas : ACM Press, 2002. – ISBN 1–58113–521–1, S. 491–500

[LM06] Liu, G. ; McMillan, L.: Estimation of missing markers in human motion capture. In: *The Visual Computer* 22 (2006), Nr. 9, S. 721–728

[LMFP10] Li, Lei ; McCann, James ; Faloutsos, Christos ; Pollard, Nancy: BoLeRO: A principled Technique for Including Bone Length Constraints in Motion Capture Occlusion Filling. In: *The ACM SIGGRAPH / Eurographics Symposium on Computer Animation (SCA 2010)* (2010)

[LMV00] Lathauwer, Lieven D. ; Moor, Bart D. ; Vandewalle, Joos: A multilinear singular value decomposition. In: *SIAM J. Matrix Anal. Appl* 21 (2000), S. 1253–1278

[LWB+10] Lee, Yongjoon ; Wampler, Kevin ; Bernstein, Gilbert ; Popović, Jovan ; Popović, Zoran: Motion fields for interactive character locomotion. In: *ACM Trans. Graph.* 29 (2010), December, 138:1–138:8. http://dx.doi.org/http://doi.acm.org/10.1145/1882261.1866160. – DOI http://doi.acm.org/10.1145/1882261.1866160. – ISSN 0730–0301

[LWC+11] Liu, Huajun ; Wei, Xiaolin ; Chai, Jinxiang ; Ha, Inwoo ; Rhee, Taehyun: Realtime human motion control with a small number of inertial sensors. In: *Symposium on Interactive 3D Graphics and Games.* New York, NY, USA : ACM, 2011 (I3D '11). – ISBN 978–1–4503–0565–5, 133–140

[LZWM05] Liu, Guodong ; Zhang, Jingdan ; Wang, Wei ; McMillan, Leonard: A system for analyzing and indexing human-motion databases. In: *Proc. 2005 ACM SIGMOD Intl. Conf. on Management of Data*, ACM Press, 2005. – ISBN 1–59593–060–4, S. 924–926

[Mai96] MAIOCCHI, Roberto: *3-D character animation using motion capture.* Upper Saddle River, NJ, USA : Prentice-Hall, Inc., 1996. – 10–39 S. – ISBN 0–13–518309–X

[MHK06] MOESLUND, T. B. ; HILTON, A. ; KRÜGER, V.: A survey of advances in vision-based human motion capture and analysis. In: *Comput. Vis. Image Underst.* 104 (2006), Nr. 2, S. 90–126

[MK06] MUKAI, Tomohiko ; KURIYAMA, Shigeru: Multilinear Motion Synthesis Using Geostatistics. In: *ACM SIGGRAPH / Eurographics Symposium on Computer Animation - Posters and Demos*, 2006, S. 21–22

[MLC10] MIN, Jianyuan ; LIU, Huajun ; CHAI, Jinxiang: Synthesis and editing of personalized stylistic human motion. In: *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games.* New York, NY, USA : ACM, 2010 (I3D '10). – ISBN 978–1–60558–939–8, 39–46

[MP07] MCCANN, James ; POLLARD, Nancy: Responsive Characters from Motion Fragments. In: *ACM Transactions on Graphics* 26 (2007), Nr. 3. – SIGGRAPH 2007

[MR06] MÜLLER, Meinard ; RÖDER, Tido: Motion Templates for Automatic Classification and Retrieval of Motion Capture Data. In: *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ACM Press, 2006, S. 137–146

[MRC05] MÜLLER, Meinard ; RÖDER, Tido ; CLAUSEN, Michael: Efficient content-based retrieval of motion capture data. In: *ACM Trans. Graph.* 24 (2005), July, 677–685. `http://dx.doi.org/http://doi.acm.org/10.1145/1073204.1073247`. – DOI http://doi.acm.org/10.1145/1073204.1073247. – ISSN 0730–0301

[MRC+07] Müller, M. ; Röder, T. ; Clausen, M. ; Eberhardt, B. ; Krüger, B. ; Weber, A.: Documentation Mocap Database HDM05 / Universität Bonn. 2007 (CG-2007-2). – Forschungsbericht. – ISSN 1610–8892. – www.mpi-inf.mpg.de/resources/HDM05

[Mül07] Müller, Meinard: *Information Retrieval for Music and Motion*. Springer, 2007. – ISBN 978–3–540–74047–6

[Nik11] Nike: *Nike Homepage*. http://www.nike.com, Accessed December 28th, 2011. http://www.nike.com. Version: 2011

[Nin11] Nintendo: *Nintendo Homepage*. http://www.nintendo.com/wii, Accessed December 28th, 2011. http://www.nintendo.com/wii. Version: 2011

[OBHK05] Ormoneit, Dirk ; Black, Michael J. ; Hastie, Trevor ; Kjellström, Hedvig: Representing cyclic human motion using functional analysis. In: *Image Vision Comput.* 23 (2005), Nr. 14, S. 1264–1276

[OTH02] Oore, Sageev ; Terzopoulos, Demetri ; Hinton, Geoffrey: A Desktop Input Device and Interface for Interactive 3D Character Animation. In: *Proceedings of Graphics Interface 2002 (GI'02)*, 2002, S. 133–140

[PB02] Pullen, Katherine ; Bregler, Christoph: Motion capture assisted animation: texturing and synthesis. In: *ACM Trans. Graph.* 21 (2002), July, 501–508. http://dx.doi.org/http://doi.acm.org/10.1145/566654.566608. – DOI http://doi.acm.org/10.1145/566654.566608. – ISSN 0730–0301

[Pha11] PhaseSpace: *PhaseSpace Motion Capture*. http://www.phasespace.com, Accessed December 28th, 2011. http://www.phasespace.com. Version: 2011

[RCO05] Rovshan, Kalanov ; Cho, Jieun ; Ohya, Jun: D-12-79 A Study of Synthesizing New Human Motions from Sampled Motions Using Tensor Decomposition. In: *Proceedings of the IEICE General Conference* 2005 (2005), Nr. 2, 229. http://ci.nii.ac.jp/naid/110004746409/en/

[SH07] Safonova, Alla ; Hodgins, Jessica K.: Construction and Optimal Search of Interpolated Motion Graphs. In: *ACM Transactions on Graphics* 26 (2007), Nr. 3. – SIGGRAPH 2007

[SH08a] Shiratori, Takaaki ; Hodgins, Jessica K.: Accelerometer-based user interfaces for the control of a physically simulated character. In: *ACM Trans. Graph.* 27 (2008), December, 123:1–123:9. http://dx.doi.org/http://doi.acm.org/10.1145/1409060.1409076. – DOI http://doi.acm.org/10.1145/1409060.1409076. – ISSN 0730–0301

[SH08b] Slyper, Ronit ; Hodgins, Jessica: Action Capture with Accelerometers. In: *Proceedings of the 2008 ACM/Eurographics Symposium on Computer Animation*, 2008

[SHP04] Safonova, Alla ; Hodgins, Jessica K. ; Pollard, Nancy S.: Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. In: *ACM Transactions on Graphics* 23 (2004), Nr. 3, 514–521. http://doi.acm.org/10.1145/1015706.1015754. – ISSN 0730–0301. – SIGGRAPH 2004

[SKL07] Sok, Kwang W. ; Kim, Manmyung ; Lee, Jehee: Simulating Biped Behaviors from Human Motion Data. In: *ACM Transactions on Graphics (SIGGRAPH 2007)* 26 (2007), jul, Nr. 3, S. Article 107

[SL06] Shin, Hyun J. ; Lee, Jehee: Motion synthesis and editing in low-dimensional spaces: Research Articles. In: *Com-*

put. Animat. Virtual Worlds 17 (2006), July, 219–227. http://dx.doi.org/http://dx.doi.org/10.1002/cav.v17:3/4. – DOI http://dx.doi.org/10.1002/cav.v17:3/4. – ISSN 1546–4261

[SLSG01] Shin, Hyun J. ; Lee, Jehee ; Shin, Sung Y. ; Gleicher, Michael: Computer puppetry: An importance-based approach. In: *ACM Trans. Graph.* 20 (2001), April, 67–94. http://dx.doi.org/http://doi.acm.org/10.1145/502122.502123. – DOI http://doi.acm.org/10.1145/502122.502123. – ISSN 0730–0301

[SO06] Shin, Hyun J. ; Oh, Hyun S.: Fat graphs: constructing an interactive character with continuous controls. In: *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation.* Aire-la-Ville, Switzerland : Eurographics Association, 2006. – ISBN 3–905673–34–7, S. 291–298

[Son11] Sony: *Playstation Move Website.* http://us.playstation.com/ps3/playstation-move/, Accessed Dec 28th, 2011. http://us.playstation.com/ps3/playstation-move/. Version: 2011

[SRV10] Schepers, H. M. ; Roetenberg, Daniel ; Veltink, Peter H.: Ambulatory human motion tracking by fusion of inertial and magnetic sensing with adaptive actuation. In: *Med Biol Eng Comput* 48 (2010), Nr. 1, S. 27–37

[Tau07] Tautges, Jochen: *Multilineare Repräsentation von Bewegungsdaten*, Rheinische Friedrich-Wilhelms-Universität Bonn, Diploma Thesis, 2007. – (Engl. translation: Multi-linear Representation of Motion Data)

[TBv04] Thorne, Matthew ; Burke, David ; van de Panne, Michiel: Motion Doodles: An Interface for Sketching Character Motion. In: *ACM SIGGRAPH 2004* (2004)

[TKZW08] Tautges, Jochen ; Krüger, Björn ; Zinke, Arno ; Weber, Andreas: Reconstruction of Human Motions Using Few Sensors. In: Schumann, M. (Hrsg.) ; Kuhlen, T. (Hrsg.): *Virtuelle und Erweiterte Realität: 5. Workshop der GI-Fachgruppe VR/AR*. Shaker Verlag, September 2008. – ISBN 978–3–8322–7572–3, S. 1–12

[Tro02] Troje, Nikolaus F.: Decomposing biological motion: A framework for analysis and synthesis of human gait patterns. In: *Journal of Vision* 2 (2002), Nr. 5, S. 371–387

[TWC+09] Tournier, M. ; Wu, X. ; Courty, N. ; Arnaud, E. ; Revéret, L.: Motion Compression using Principal Geodesics Analysis. In: *Computer Graphics Forum* 28 (2009), Nr. 2, S. 355–364. – EUROGRAPHCS 2009

[TZK+11] Tautges, Jochen ; Zinke, Arno ; Krüger, Björn ; Baumann, Jan ; Weber, Andreas ; Helten, Thomas ; Müller, Meinard ; Seidel, Hans-Peter ; Eberhardt, Bernd: Motion Reconstruction Using Sparse Accelerometer Data. In: *ACM Trans. Graph.* 30 (2011), Mai, Nr. 3, 18:1–18:12. http://dx.doi.org/10.1145//1966394.1966397. – DOI 10.1145//1966394.1966397. – ISSN 0730–0301

[Vas02] Vasilescu, M. Alex O.: Human motion signatures: Analysis, synthesis, recognition. In: *Proc. Int. Conf. on Pattern Recognition*. Quebec City, Canada, 2002, S. 456–460

[VAV+07] Vlasic, Daniel ; Adelsberger, Rolf ; Vannucci, Giovanni ; Barnwell, John ; Gross, Markus ; Matusik, Wojciech ; Popović, Jovan: Practical motion capture in everyday surroundings. In: *ACM Trans. Graph.* 26 (2007), July. http://dx.doi.org/http://doi.acm.org/10.1145/1276377.1276421. – DOI http://doi.acm.org/10.1145/1276377.1276421. – ISSN 0730–0301

[VBPP05] Vlasic, Daniel ; Brand, Matthew ; Pfister, Hanspeter ; Popović, Jovan: Face transfer with multilinear models. In: *ACM Trans. Graph.* 24 (2005), Nr. 3, S. 426–433. http://dx.doi.org/http://doi.acm.org/10.1145/1073204.1073209. – DOI http://doi.acm.org/10.1145/1073204.1073209. – ISSN 0730–0301. – SIGGRAPH 2005

[Vic11] Vicon: *Motion Capture Systems from Vicon*. http://www.vicon.com, Accessed December 28th, 2011. http://www.vicon.com. Version: 2011

[Wik11] Wikipedia: *Motion capture*. http://en.wikipedia.org/wiki/Motion_capture, Accessed December 28th, 2011. http://en.wikipedia.org/wiki/Motion_capture. Version: 2011

[Xse11] Xsens: *3D Motion Tracking*. http://www.xsens.com, Accessed December 28th, 2011. http://www.xsens.com. Version: 2011