The definitive version was published in ACM Trans. Graph. 30:3(18:1-18:12) May 2011
http://doi.acm.org/10.1145//1966394.1966397

Motion Reconstruction Using Sparse Accelerometer Data

JOCHEN TAUTGES
Universität Bonn
and
ARNO ZINKE
GfaR mbH, Bonn
and
BJÖRN KRÜGER and JAN BAUMANN and ANDREAS WEBER
Universität Bonn
and
THOMAS HELTEN and MEINARD MÜLLER and HANS-PETER SEIDEL
Universität des Saarlandes and MPI Informatik
and
BERND EBERHARDT
HdM Stuttgart

The development of methods and tools for the generation of visually appealing motion sequences using pre-recorded motion capture data has become an important research area in computer animation. In particular, data-driven approaches have been used for reconstructing high-dimensional motion sequences from low-dimensional control signals. In this paper, we contribute to this strand of research by introducing a novel framework for generating full-body animations controlled by only four 3D accelerometers that are attached to the extremities of a human actor. Our approach relies on a knowledge base that consists of a large number of motion clips obtained from marker-based motion capturing. Based on the sparse accelerometer input a cross-domain retrieval procedure is applied to build up a lazy neighborhood graph in an online fashion. This graph structure points to suitable motion fragments in the knowledge base, which are then used in the reconstruction step. Supported by a kd-tree index structure, our procedure scales to even large datasets consisting of millions of frames. Our combined approach allows for reconstructing visually plausible continuous motion streams even in the presence of moderate tempo variations which may not be directly reflected by the given knowledge base.

Jochen Tautges and Thomas Helten were financially supported by grants from *Deutsche Forschungsgemeinschaft* (WE 1945/5-1 and MU 2686/3-1).
E-mail: {tautges, zinke, kruegerb, baumannj, weber}@cs.uni-bonn.de, {thelten, meinard, hpseidel}@mpi-inf.mpg.de, eberhardt@hdm-stuttgart.de

DOI 10.1145/XXXXXXX.YYYYYYY
http://doi.acm.org/10.1145/XXXXXXX.YYYYYYY

Categories and Subject Descriptors: I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism; I.3.6 [**Computer Graphics**]: Methodology and Techniques; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms: Motion Capture

Additional Key Words and Phrases: Motion reconstruction, acceleration data, online control, motion retrieval

ACM Reference Format:

Tautges, J. et al. 2011. Motion Reconstruction Using Sparse Accelerometer Data. ACM Trans. Graph. XX, X, Article XXXX (MONTH 2011), 12 pages.
DOI = 10.1145/XXXXXXX.XXXXXXX
http://doi.acm.org/10.1145/XXXXXXX.XXXXXXX

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 [Arikan et al. 2003] or by low-dimensional control signals as supplied by recent game consoles [Nintendo 2010]. In [Chai and Hodgins 2005], a data-driven scenario is described where a sparse set of video-based control signals is used for creating believable character animations. In their seminal work, Chai and Hodgins 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 pre-recorded 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 mark-

ers 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. In recent years, inexpensive inertial-based sensors have been used for controlling animated avatars in video games. Slyper and Hodgins [2008] describe a first system for retrieving upper-body mocap sequences using a small number of low-cost accelerometers as control input.

The work described in this paper 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 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 heterogenous 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 pose-wise nearest neighbor 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 in [Krüger et al. 2010], is then used in the reconstruction 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.

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 [2008] 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—opposed to previous methods that reconstruct a motion by choosing a prerecorded clip from a database [Slyper and Hodgins 2008]. Furthermore, the database used in Slyper and Hodgins [2008] 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. Opposed 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 a lazy-neighborhood graph previously introduced in [Krüger et al. 2010]. 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.

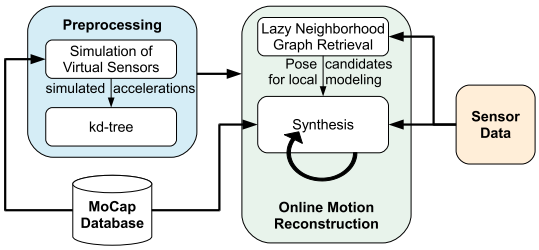


Fig. 1: *Overview of the animation system.*

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 (e. g. [Chai and Hodgins 2005]) and synthesis ([Shin and Lee 2006; Sok et al. 2007; Lee et al. 2010]). 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 (controller input) readings may be associated to very different motion classes and thus different hypotheses. Novel motion and smoothness priors used in our paper effectively guide the synthesis process towards a relatively smooth and plausible reconstruction. Please note that in previous work in the field of motion synthesis [Chai and Hodgins 2005] 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).

1.2 Paper organization

The remainder of this paper is organized as follows. In Section 2, we discuss previous work related to our approach. In Section 3, we describe the involved data types (control input, knowledge base) and discuss indexing issues. The online lazy neighborhood graph is introduced in Section 4. The optimization procedure underlying the motion reconstruction step is then explained in detail in Section 5. Finally, experiments and evaluation are described in Section 6, and we conclude by discussing limitations and future work in Section 7.

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 1996; Moeslund et al. 2006; Wikipedia 2010] 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 [PhaseSpace 2010; Vicon 2010]. 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 [Nike 2010; Nintendo 2010; Slyper and Hodgins 2008]. 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 magnet field sensors [Schepers et al. 2010]. In [Vlasic et al. 2007], the authors 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 this paper, 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 [Shin et al. 2001]—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 on the basis of 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, e. g., [Badler et al. 1993; Cooper et al. 2007; Dontcheva et al. 2003; Oore et al. 2002]. In [Shiratori and Hodgins 2008], inertial-based control data is used to specify a small number of free parameters in physically-based character animation. When high-dimension data has to be generated using only low-dimensional control data, especially data-driven approaches show promising results. For example Feng et al. [2008] 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 [2008] describe a system, where a small number of low-cost accelerometers are used to identify and playback pre-recorded human upper-body motions. An extension to this work is sketched in [Kelly et al. 2010], 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 pre-recoded human mocap data requires efficient retrieval of similar motions from databases [Keogh et al. 2004; Müller et al. 2005], as well as a good understanding of how motions have to be parametrized in order to yield smooth transitions between several retrieved motion clips [Kovar and Gleicher 2003].

Our approach is inspired by the animation system presented by Chai and Hodgins [2005], outlined earlier. Opposed to using optical markers and calibrated cameras, we use a sparse set of four 3D accelerometers to generate the control data. Also, we don't 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 [Slyper and Hodgins 2008], our approach allows for handling moderate temporal and other variations that are not reflected well by the given database motions.

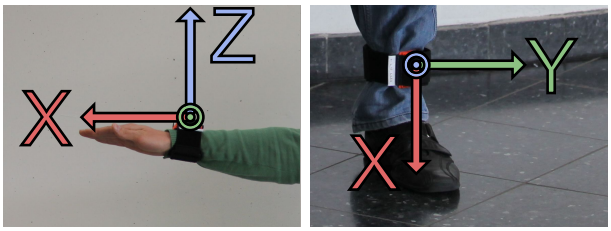


Fig. 2: *The four accelerometers are attached to the lower arms and lower legs using simple straps.*

3. CONTROL INPUT AND KNOWLEDGE BASE

In this paper we use four Xsens MTx devices [Xsens 2010] which provide the control input to our reconstruction framework. The sensors are attached to the lower arms and lower legs 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, we only use the calibrated readings from the devices' accelerometers. Thus our findings can be applied to much smaller—and cheaper—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 2 shows the 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 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. Note that also sensor calibration procedures as proposed by [Slyper and Hodgins 2008] may be applied. However, carefully fitting the sensorshas 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 $\boldsymbol{q}_n$, $n \in [1:N]$. 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 use *virtual sensors* for motions of the knowledge base to simulate accelerometer readings obtained from controller input. These virtual sensors are placed on the limbs in the same way as the real sensors. 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, and transform the resulting quantity to the local coordinate

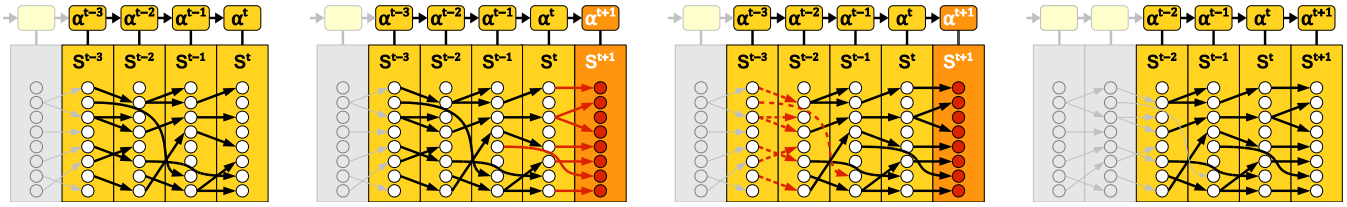


Fig. 3: *Online Lazy Neighborhood Graph (OLNG) with $M = 4$ and $K = 8$. Each vertical column corresponds to the K nearest neighbors (each neighbor indicated by a circle) of a sensor reading $\boldsymbol{\alpha}^{t-m+1}$, $m \in [1 : M]$. The edges encode temporal coherence between the nearest neighbors. The figure illustrates the implementation of the OLNG.*

systems of the virtual sensors. In addition to simulated sensor data, we pre-compute quantities that we later use in the synthesis step of our method, including the positions $\boldsymbol{x}_n$, velocities $\boldsymbol{v}_n$, and accelerations $\boldsymbol{a}_n$ of all joints. For normalization purposes, these quantities are given in the root coordinate system. 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. For all our tests, we neglected the skeleton's foot and hand joints, resulting in a representation with 21 joints and a total of 43 rotational degrees of freedom.

The simulated sensor accelerations are denoted by $\boldsymbol{\alpha}_n$ and indexed using a kd-tree of dimension $4 \cdot 3 = 12$. At those low dimensions, Andoni and Indyk [2008] 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 4.

4. ONLINE LAZY NEIGHBORHOOD GRAPH

In our scenario, controller input is compared with mocap sequences of the knowledge base using 3D accelerations. However, note that a comparison on the acceleration level is much less descriptive than, e. g., on the joint angle or 3D positional level as used in [Chai and Hodgins 2005]. This may result in a large number of false positives, in particular when using a framewise retrieval procedure. Therefore, Slyper and Hodgins [2008] use the temporal coherence of motions by querying fixed-length sequences of accelerometer readings. However, temporal variations (e. g. motions performed at different speeds) 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). The OLNG structure used in this work is similar to the *lazy neighborhood graph* (LNG) introduced in [Krüger et al. 2010], but there are some significant improvements over the original method. In particular, we show how the LNG can be built up incrementally making its construction efficient and online capable. We assume that the control input consists of a (sampled) continuous stream of sensor accelerations $(\ldots, \boldsymbol{\alpha}^{t-2}, \boldsymbol{\alpha}^{t-1}, \boldsymbol{\alpha}^{t}, \ldots)$, where $\boldsymbol{\alpha}^t$ denotes the current frame at time $t \in \mathbb{Z}$. Fixing a number K of nearest neighbors, let $S^t \subset [1 : N]$ be the set of size K consisting of the frame indices of the K nearest neighbors of $\boldsymbol{\alpha}^t$. Note that S^t can be computed efficiently using the kd-tree mentioned in Section 3. We now consider the last M sensor readings $(\boldsymbol{\alpha}^{t-M+1}, \ldots, \boldsymbol{\alpha}^{t})$ for a fixed number $M \in \mathbb{N}$. Then, the nodes of the OLNG can be represented by an $M \times K$ array, where the m^{th} column, $m \in [1 : M]$, corresponds to the sorted set S^{t-m+1}, see Figure 3 (a) for an illustration. Now, the directed edges of the OLNG, which are strictly monotonous with respect to the column index, encode temporal coherence between the retrieved indices stored in the columns. In particular, each edge enforces monotonicity while respecting a maximum bound between connected indices. The edges allow for constructing paths within the $M \times K$ array. Each such path yields an index sequence, which in turn corresponds to a motion subsequence within the knowledge base. The main observation is that each such subsequence is similar to the sensor input $(\boldsymbol{\alpha}^{t-M+1}, \ldots, \boldsymbol{\alpha}^{t})$. The construction of the paths is based on the following principles. First, by imposing suitable step size and index conditions[1] for the edges, temporal variations can be modeled by the paths. Second, for a given path, a cost is associated that depends on the frame similarities (between retrieved subsequence frames and query frames), the step sizes of the edges (skipped columns are penalized), and the length of the path (longer paths are preferred). Third, only K paths are stored. More precisely, for each node in the last column corresponding to S^t, a path of minimal cost ending in this node is stored. For further details on the original LNG, we refer to [Krüger et al. 2010].

Since we want to reconstruct motions from a continuous stream of sensor readings, we have to identify optimal subsequences inside the knowledge base for every point in time. Rebuilding the LNG for every new sensor reading would be costly and unnecessary, since most of the data inside the graph structure can be reused. We now introduce a novel procedure that allows for efficiently updating the OLNG. Suppose that the OLNG has been constructed for the readings $(\boldsymbol{\alpha}^{t-M+1}, \ldots, \boldsymbol{\alpha}^{t})$ and that a new reading $\boldsymbol{\alpha}^{t+1}$ arrives. First, for $\boldsymbol{\alpha}^{t+1}$, 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 (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 $\boldsymbol{\alpha}^{t-M+1}$ as well as the involved edges are removed to obtain the updated OLNG, see Figure 3 (d).

As the graph structure is built incrementally and not as a whole as in [Krüger et al. 2010], our implementation is suitable for online applications. There is no latency introduced by our OLNG, even at the 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, regard-

[1] As for the mentioned stepsize conditions, however, it should be noted, that we differ from the original paper: We found that steps enforcing strict monotonicity in time while simultaneously avoiding temporal warpings by a factor greater than two perform better than the proposed steps under the given conditions.

less of their global performance. Due to its incremental nature our approach detects and considers such paths directly as they appear. Hence, in contrast to [Krüger et al. 2010], 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. Thus, in cases where no full-length matches can be found, shorter motion fragments are considered by our method.

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 which are of quadratic complexity, our approach scales well to large datasets consisting of millions of frames.

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 by 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") pre-recorded motion clips or poses. We adopt this basic idea by using the online algorithm described in Section 4 which provides for each time step t a set of K paths together with associated costs. In practice, most of the resulting paths are rather short and have high costs. In the following, we only consider those $I \ll K$ paths having the least costs. We denote $C^t = \{c_1^t, \ldots, c_I^t\}$ to be the costs of these I paths. Furthermore, let $Q^t = \{\boldsymbol{q}_1^t, \ldots, \boldsymbol{q}_I^t\}$ be the set of joint angle configurations given by the last frames of all these paths. Analogously let $X^t = \{\boldsymbol{x}_1^t, \ldots, \boldsymbol{x}_I^t\}$ be the positions, $V^t = \{\boldsymbol{v}_1^t, \ldots, \boldsymbol{v}_I^t\}$ be the velocities, and $A^t = \{\boldsymbol{a}_1^t, \ldots, \boldsymbol{a}_I^t\}$ be the accelerations of the joints with respect to the root coordinate system. As these quantities were already computed in the preprocessing step, see Section 3, they can be easily obtained from the knowledge base at runtime. Finally, based on the costs $C^t = \{c_1^t, \ldots, c_I^t\}$, we introduce *normalized weights* denoted by $W^t = \{w_1^t, \ldots, w_I^t\}$, where the value of each weight w_i^t is given by

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

Now, we formulate the motion reconstruction as an energy minimization problem. For each time step we aim to find a pose $\boldsymbol{q}_{\text{best}}$ 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:

$$\boldsymbol{q}_{\text{best}} = \underset{\boldsymbol{q}}{\operatorname{argmin}}(w_{\text{prior}} \cdot E_{\text{prior}}(\boldsymbol{q}) + w_{\text{contr}} \cdot E_{\text{contr}}(\boldsymbol{q})). \tag{2}$$

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 $\boldsymbol{\alpha}^{t+1}$ 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 because of the following two reasons. First, we only predict the joint positions for one frame into the future. Second, Q^t is continuously updated by the lazy neighborhood graph, which is based on the pre-recored database motions.

5.1 The 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:

$$\begin{aligned} E_{\text{prior}}(\boldsymbol{q}) = {} & w_{\text{pose}} \cdot E_{\text{pose}}(\boldsymbol{q}) \\ & + w_{\text{motion}} \cdot E_{\text{motion}}(\boldsymbol{q}) \\ & + w_{\text{smooth}} \cdot E_{\text{smooth}}(\boldsymbol{q}) \end{aligned} \tag{3}$$

In contrast to existing approaches used in the context of motion synthesis [Chai and Hodgins 2005], 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.

5.1.1 *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 used by Chai and Hodgins [2005], we propose a kernel based approach to approximate the likelihood p_{pose} of a synthesized pose candidate $\boldsymbol{q}$:

$$p_{\text{pose}}(\boldsymbol{q}) \propto \sum_{i=1}^{I} w_i^{t+1} \cdot \mathcal{K}(|\boldsymbol{q}_i^{t+1} - \boldsymbol{q}|). \tag{4}$$

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 in-

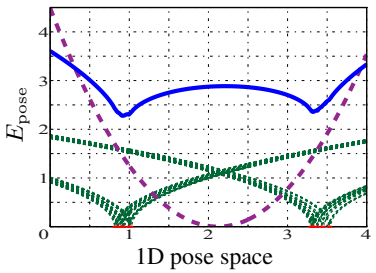


Fig. 4: *A simple example illustrating the effect of our kernel based approach in case of clustered data samples. Here, the green dashed lines indicate the kernel functions centered at the sample positions and the solid blue line represents our energy term E_{pose}. The dotted purple line symbolizes the energy function proposed by Chai and Hodgins [2005]. Please note how the local sample density is determining the likelihood of a pose candidate in contrast to Chai and Hodgins [2005]: Clusters of samples induce distinctive local minima of E_{pose}.*

tegral 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 re-formulated as an expression suitable for energy minimization. In practice, a square root kernel is used to compute the energy term E_{pose}:

$$E_{\text{pose}}(\boldsymbol{q}) = \sum_{i=1}^{I} w_i^{t+1} \cdot \sqrt{|\boldsymbol{q}_i^{t+1} - \boldsymbol{q}|}. \tag{5}$$

The above expression yields results (regarding optimality) comparable to p_{pose} (see Figure 4) 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.

5.1.2 *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 $\boldsymbol{x}^{t+1}$ (the true joint positions at time $t+1$) by computing the second order Taylor expansion at the joint positions $\boldsymbol{x}^t$ (associated to $\boldsymbol{q}^t$) using V^{t+1} and A^{t+1}. For the i-th sample ($i \in \{1, .., n\}$) the estimated positions $\boldsymbol{x'}_i^{t+1}$ are then given by:

$$\boldsymbol{x'}_i^{t+1} = \boldsymbol{x}^t + \boldsymbol{v}_i^{t+1} \cdot \Delta t + \frac{1}{2}\boldsymbol{a}_i^{t+1} \cdot \Delta t^2. \tag{6}$$

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

$$E_{\text{motion}}(\boldsymbol{x}) = \sum_{i=1}^{I} w_i^{t+1} \cdot \sqrt{|\boldsymbol{x'}_i^{t+1} - \boldsymbol{x}|}. \tag{7}$$

5.1.3 *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 which attempt to enforce smoothness by minimizing joint accelerations, we make direct use of the a-priori knowledge provided by the database. A pose $\boldsymbol{q}$ (with joint positions $\boldsymbol{x}$) 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:

$$E_{\text{smooth}}(\boldsymbol{a}) = \sum_{i=1}^{I} w_i^{t+1} \cdot \Delta t \cdot \sqrt{|\boldsymbol{a}_i^{t+1} - \boldsymbol{a}|} \tag{8}$$

with

$$\boldsymbol{a} = \Delta t^{-2} \cdot \left(\boldsymbol{x} - 2\boldsymbol{x}^t + \boldsymbol{x}^{t-1}\right). \tag{9}$$

5.2 The 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 $\langle \boldsymbol{y} \rangle$ be the projection of a vector $\boldsymbol{y}$ to the subspace formed by the components related to those joints which are next to the virtual sensors. Assuming proper positions $\boldsymbol{x}^t$ 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:

$$\tilde{\boldsymbol{x}}_i^{t+1} = \left\langle \boldsymbol{x}^t + \boldsymbol{v}_i^t \cdot \Delta t \right\rangle + \frac{1}{2}\hat{\boldsymbol{\alpha}}^t \cdot \Delta t^2. \tag{10}$$

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

$$E_{\text{contr}}(\boldsymbol{x}) = \sum_{i=1}^{I} w_i^{t+1} \cdot \sqrt{|\tilde{\boldsymbol{x}}_i^{t+1} - \langle \boldsymbol{x} \rangle|}. \tag{11}$$

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 make 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.

5.3 Energy Minimization

We employ a gradient descent based method[2] to minimize the objective function (2) 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 2 were kept fixed at the following values: $w_{\text{contr}} = 1$, $w_{\text{prior}} = 5$, $w_{\text{pose}} = 0.6$, $w_{\text{motion}} = 0.2$, $w_{\text{smooth}} = 0.2$. 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 [2005] a local linear model approximation 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.

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 [Tournier et al. 2009], the widely accepted average RMS error of joint positions (relative to the skeleton root frame) is used in the following for all numerical comparisons.

6.1 Tests Based on Real Sensor Readings

In a first test, we reconstructed motions of two different actors performed in an outdoor setting. As we do not have ground truth data to quantify 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, opposed to optical systems, the 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 it is inherently complicated to create temporal variation for arbitrary motions, we restricted our 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 13. Here, the lower errors obtained with our method

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

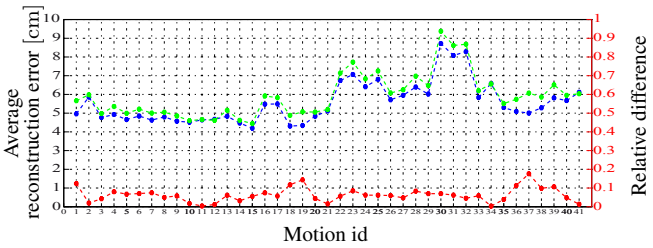


Fig. 6: *Reconstruction error for motions with recorded ground truth.* ***Blue****: reconstruction errors using sensor data (MTx accelerations).* ***Green****: reconstruction errors using simulated accelerations.* ***Red****: relative difference between both.*

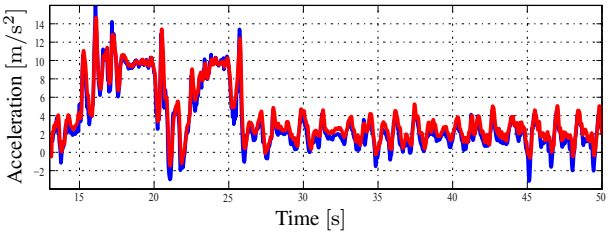


Fig. 7: *Comparison of simulated sensor data* ***(red)*** *and real accelerometer readings* ***(blue)*** *obtained from an MTx device attached to a user's left wrist.*

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.

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) by 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 6. 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 7 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 [Müller et al. 2007] for systematic evaluations of our method.

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.

Fig. 5: *Using accelerations obtained by four 3D accelerometers attached to a human actor as control input, a full-body animation is reconstructed using motion fragments retrieved from a knowledge base consisting of motion capture data.*

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 I. : *Summary of the three scenarios used for our evaluations.*

6.2.1 *General scenarios used for testing.* In the following experiments the knowledge base consists of motion clips taken from the publicly available motion database HDM05, see [Müller et al. 2007]. The 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 data base 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 using a set of *test 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 I for an overview.

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

—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 ($\mathrm{DB}_3^i := \mathrm{DB}_1 \backslash \mathrm{DB}_2^i$).

6.2.2 *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 8. 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 8 (b) and 8 (d), as well as by the prominent blue diagonals in subfigures 8 (a) and 8 (c) representing scenario 2. As our method is very robust towards variations in actor sizes (see 6.2.4) and relatively robust towards moderate variations in speed (see 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 8 (b) and 8 (d), as well as by comparing the overall appearances of subfigures 8 (a) and 8 (c). Strictly speaking one should distinguish between a mere increase

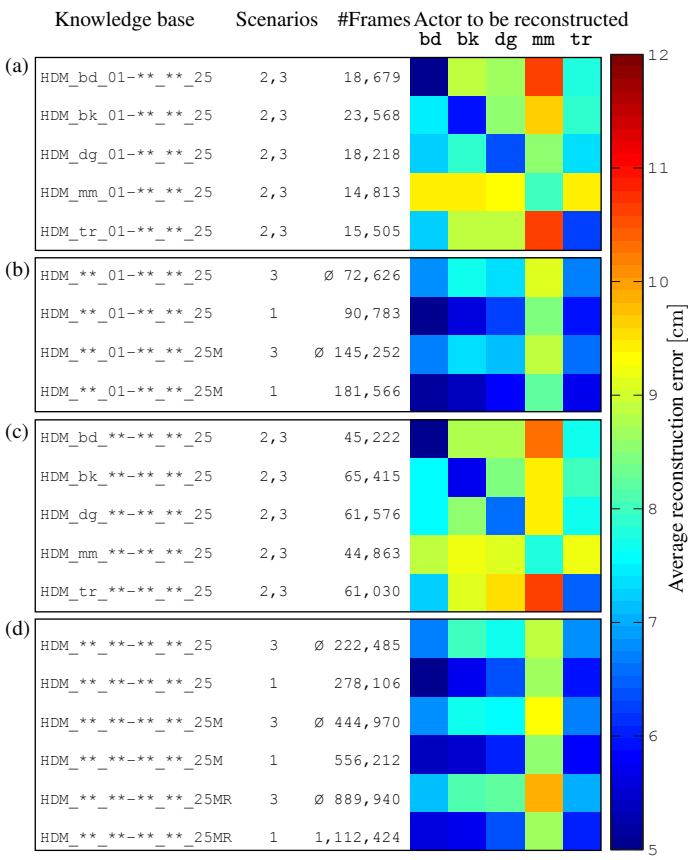


Fig. 8: *Average reconstruction error for a given knowledge base and a given actor to be reconstructed. In addition, the sizes of the knowledge bases (in terms of number of frames) as well as the effective scenarios are indicated (with blue diagonals in the first and third box representing Scenario 2).*

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.

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 $0.5\,\mathrm{cm}$, see Figure 9 (a). 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.

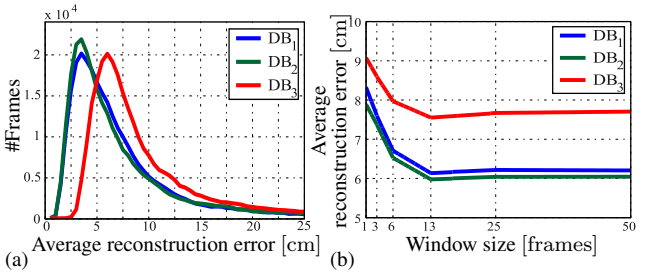


Fig. 9: ***(a)*** *Histogram of the average RMS error for the different scenarios,* ***(b)*** *Influence of the window size M of the OLNG.*

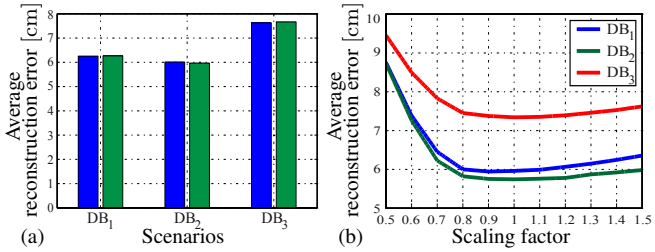


Fig. 10: *Dependency of the average reconstruction error on the size and proportions of actors:* ***(a)*** *Comparing the use of original proportions (blue) in the underlying knowledge base with our standard approach of using skeletons with averaged bone lengths (green).* ***(b)*** *Uniformly scaling 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.*

6.2.3 *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 [Krüger et al. 2010], its assignment is far less critical than in the original approach. Figure 9 (b) shows the reconstruction results using different window sizes.

6.2.4 *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 HDM05. As can be seen from Figure 10 (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 10 (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.

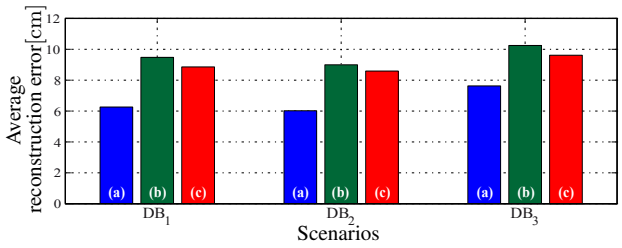


Fig. 11: *Comparing the average reconstruction error obtained with the new prior model (a) to approaches that attempt to adapt existing ones to our framework: (b) Replacing kernel regression in E_{contr}, E_{pose}, E_{motion} and E_{smooth} by Mahalanobis distance; (c) Using multivariate normal distribution prior model for E_{contr} together with an ad-hoc smoothness assumption.*

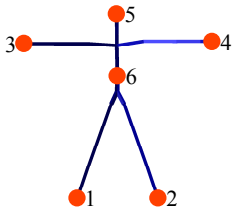
6.2.5 *The prior model.* As discussed in Section 5.1 the novel priors substantially improve the reconstruction quality compared to existing models previously presented in the context of motion synthesis. In order to validate this claim we made comparisons to methods that attempt to adapt existing prior models to our framework:

—Using a prior along the lines of Chai and Hodgins [2005]: 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 11 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.

6.2.6 *Tests with different sensor setups.* The average reconstruction error was analyzed for different sparse sensor configurations (cf. Table II) including one to six sensors. This was done by both a histogram based approach, similar to Section I (see Figure 12 (a)) performed on the complete HDM05 data base, and—for the sake of easier comparison—an evaluation of the subset of test motions described earlier in this section (see Figure 12 (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 data base: Although our test data bases taken from HDM05 contain a variety of motions, they contain rather few motions where there are different movements of the head and torso for the same motions of the hands and feet (like sitting down a table without moving feet and hands vs. 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).



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 II. : *Different sensor setups*

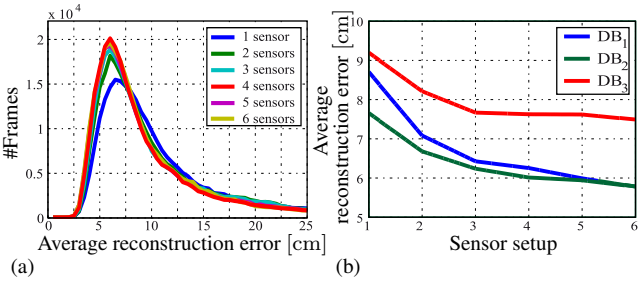


Fig. 12: *Dependency of the average reconstruction error on the sensor setup:* ***(a)*** *Histogram-based evaluation, performed on the complete HDM05 database* ***(b)*** *Evaluation based on a selected set of test motions*

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

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 data bases is no longer a bottleneck, opposed to existing techniques. As can be seen in Table III, 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.

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.

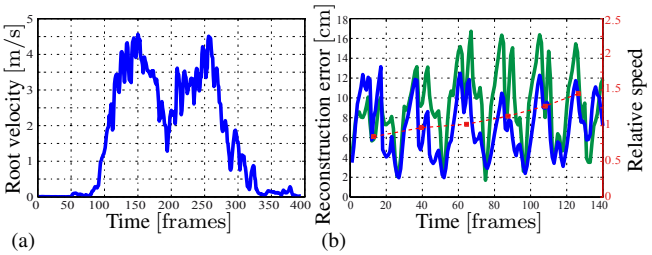


Fig. 13: ***(a)*** *Estimated root speed of a run-walk-run motion.* ***(b)*** *Smoothly varying the relative speed of a jumping jack motion over time. Here, the dashed red line indicates the relative speed and the red dots the "hand clapping" frame for six subsequent jumping jacks.* ***Blue****: Reconstruction error using our method.* ***Green****: Reconstruction error using a variant of our method that does not account for time warped motion sequences.*

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 for generating consistent root motions is given in Figure 13. 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 data base no range of running motions at various speeds have been captured but only the slight variations of a slow running.

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.

7. CONCLUSION AND FUTURE WORK

Although acceleration data of motions contains less information than positional data we have shown that not only action recognition but also motion reconstruction is possible in many cases using the data of surprisingly few accelerometers.

As by exploiting pure accelerometer data only the use of very small and cheap sensors is possible, and as also our chosen sensor positioning is highly practical, we will build such devices in future work and exploit them for various tasks requiring motion puppetry. In this context, it will be essential to allow for real-time reconstructions. Our current implementation that is based on single threaded MATLAB code is not fast enough (on current standard PCs) to meet this goal, but also not too far away: We presume that code optimizations (and porting code to C++) combined with parallelizations exploiting the trend to multi-core CPUs will achieve this goal in the near future.

The novel data-driven prior model for motion synthesis based on large, heterogeneous databases lays also the foundation for other animation tasks: Although technically very different from the approach suggested by Pullen and Bregler [2002] it can be used as well for synthesizing specified missing degrees of freedom in key-frame animations, but also for "texturing" key-frame animations with details of motions extracted from the databases. It will be another topic of future research to extend our techniques in these directions.

Acknowledgements

The authors would like to thank Anne Sevenich for speaking the audio track of the supplemental video. This research was supported in part by *Deutsche Forschungsgemeinschaft* under grants WE 1945/5-1 and MU 2686/3-1.

REFERENCES

ANDONI, A. AND INDYK, P. 2008. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM 51,* 1, 117–122.

ARIKAN, O., FORSYTH, D. A., AND O'BRIEN, J. F. 2003. Motion synthesis from annotations. *ACM Trans. Graph. 22,* 3, 402–408.

BADLER, N. I., HOLLICK, M. J., AND GRANIERI, J. P. 1993. Real-time control of a virtual human using minimal sensors. *Presence: Teleoperators and Virtual Environments* 1, 82–86.

CHAI, J. AND HODGINS, J. K. 2005. Performance animation from low-dimensional control signals. *ACM Trans. Graph. 24,* 3, 686–696.

COOPER, S., HERTZMANN, A., AND POPOVIĆ, Z. 2007. Active learning for real-time motion controllers. *ACM Trans. Graph. 26,* 3, 5.

DONTCHEVA, M., YNGVE, G., AND POPOVIĆ, Z. 2003. Layered acting for character animation. *ACM Trans. Graph. 22*, 409–416.

FENG, W.-W., KIM, B.-U., AND YU, Y. 2008. Real-time data driven deformation using kernel canonical correlation analysis. *ACM Trans. Graph. 27*, 91:1–91:9.

KELLY, P., CONAIRE, C. O., HODGINS, J., AND O'CONNER, N. E. 2010. Human motion reconstruction using wearable accelerometers (poster). In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation (SCA).*

KEOGH, E., PALPANAS, T., ZORDAN, V. B., GUNOPULOS, D., AND CARDLE, M. 2004. Indexing large human-motion databases. In *VLDB '04: Proceedings of the Thirtieth international conference on Very large data bases*. VLDB Endowment, 780–791.

KOVAR, L. AND GLEICHER, M. 2003. Flexible automatic motion blending with registration curves. In *Eurographics/SIGGRAPH Symposium on*

Computer Animation, D. Breen and M. Lin, Eds. Eurographics Association, 214–224.

KRÜGER, B., TAUTGES, J., WEBER, A., AND ZINKE, A. 2010. Fast local and global similarity searches in large motion capture databases. *Eurographics / ACM SIGGRAPH Symposium on Computer Animation*, 1–10.

LEE, Y., WAMPLER, K., BERNSTEIN, G., POPOVIĆ, J., AND POPOVIĆ, Z. 2010. Motion fields for interactive character locomotion. *ACM Trans. Graph. 29*, 138:1–138:8.

MAIOCCHI, R. 1996. *3-D character animation using motion capture*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

MOESLUND, T. B., HILTON, A., AND KRÜGER, V. 2006. A survey of advances in vision-based human motion capture and analysis. *Comput. Vis. Image Underst. 104,* 2, 90–126.

MÜLLER, M., RÖDER, T., AND CLAUSEN, M. 2005. Efficient content-based retrieval of motion capture data. *ACM Trans. Graph. 24*, 677–685.

MÜLLER, M., RÖDER, T., CLAUSEN, M., EBERHARDT, B., KRÜGER, B., AND WEBER, A. 2007. Documentation: Mocap Database HDM05. Computer Graphics Technical Report CG-2007-2, Universität Bonn. jun. `http://www.mpi-inf.mpg.de/resources/HDM05`.

NIKE. 2010. Nike homepage. `http://www.nike.com`, Accessed March 10th, 2010.

NINTENDO. 2010. Nintendo homepage. `http://www.nintendo.com`, Accessed March 10th, 2010.

OORE, S., TERZOPOULOS, D., AND HINTON, G. 2002. A desktop input device and interface for interactive 3d character animation. In *Proceedings of Graphics Interface 2002 (GI'02)*. 133–140.

PHASESPACE. 2010. PhaseSpace motion capture. `http://www.phasespace.com`, Accessed March 10th, 2010.

PULLEN, K. AND BREGLER, C. 2002. Motion capture assisted animation: texturing and synthesis. *ACM Trans. Graph. 21*, 501–508.

SCHEPERS, H. M., ROETENBERG, D., AND VELTINK, P. H. 2010. Ambulatory human motion tracking by fusion of inertial and magnetic sensing with adaptive actuation. *Med Biol Eng Comput 48,* 1, 27–37.

SHIN, H. J. AND LEE, J. 2006. Motion synthesis and editing in low-dimensional spaces: Research articles. *Comput. Animat. Virtual Worlds 17*, 219–227.

SHIN, H. J., LEE, J., SHIN, S. Y., AND GLEICHER, M. 2001. Computer puppetry: An importance-based approach. *ACM Trans. Graph. 20*, 67–94.

SHIRATORI, T. AND HODGINS, J. K. 2008. Accelerometer-based user interfaces for the control of a physically simulated character. *ACM Trans. Graph. 27*, 123:1–123:9.

SLYPER, R. AND HODGINS, J. 2008. Action capture with accelerometers. In *Proceedings of the 2008 ACM/Eurographics Symposium on Computer Animation*.

SOK, K. W., KIM, M., AND LEE, J. 2007. Simulating biped behaviors from human motion data. *ACM Transactions on Graphics (SIGGRAPH 2007) 26,* 3 (jul), Article 107.

TOURNIER, M., WU, X., COURTY, N., ARNAUD, E., AND REVÉRET, L. 2009. Motion compression using principal geodesics analysis. *Computer Graphics Forum 28,* 2, 355–364. EUROGRAPHCS 2009.

VICON. 2010. Motion capture systems from vicon. `http://www.vicon.com`, Accessed March 10th, 2010.

VLASIC, D., ADELSBERGER, R., VANNUCCI, G., BARNWELL, J., GROSS, M., MATUSIK, W., AND POPOVIĆ, J. 2007. Practical motion capture in everyday surroundings. *ACM Trans. Graph. 26*.

WIKIPEDIA. 2010. Motion capture. `http://en.wikipedia.org/wiki/Motion_capture`, Accessed March 10th, 2010.

XSENS. 2010. 3D motion tracking. `http://www.xsens.com`, Accessed March 10th, 2010.