Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2012), pp. 1–6
P. Kry and J. Lee (Editors)

Interactive Steering of Mesh Animations

Anna Vögele, Max Hermann, Björn Krüger and Reinhard Klein

Bonn University, Institute of Computer Science II, Computer Graphics

Abstract

Creating geometrically detailed mesh animations is an involved and resource-intensive process in digital content creation. In this work we present a method to rapidly combine available sparse motion capture data with existing mesh sequences to produce a large variety of new animations. The key idea is to model shape changes correlated to the pose of the animated object via a part-based statistical shape model. We observe that compact linear models suffice for a segmentation into nearly rigid parts. The same segmentation further guides the parameterization of the pose which is learned in conjunction with the marker movement. Besides the inherent high geometric detail, further benefits of the presented method arise from its robustness against errors in segmentation and pose parameterization. Due to efficiency of both learning and synthesis phase, our model allows to interactively steer virtual avatars based on few markers extracted from video data or input devices like the Kinect sensor.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.7]: Three-Dimensional Graphics and Realism—[Animation]

Keywords: motion transfer, motion capture, mesh animation, morphable part model

1. Introduction

The creation of detailed mesh animations is a time consuming task, even for skilled animators. We introduce a system which allows even untrained users to generate new mesh animations based on an example input mesh sequence and sparse marker data. An illustration of a characteristic result is given in Fig. 1.

Inspired by previous work in the field of modifying mesh sequences we found that, commonly, sophisticated input is required to produce variations of a mesh animation, either in terms of additional mesh animations or registered high quality scans or detailed manual user input. This motivates the question as to what is the sparsest user input that still allows for concerted creation of novel animations. This topic has been investigated in a different context from several specific angles: Tena et al. [TDM11] and Weise et al. [WBLP11] concentrate on facial animation while Huang et al. [HZY*11] focus on hand deformations. These works are complemented by the presented motion transfer system to steer articulated full body movement.

Aside from direct motion transfer our method is also capable of semantic deformation transfer [BVGP09]. By adapting the marker input carefully to the example mesh, differing bone length and movement styles can be compensated. Furthermore, since parts are treated individually, articulated parts of the input can be mapped arbitrarily to mesh parts as in the example shown in Fig. 1.

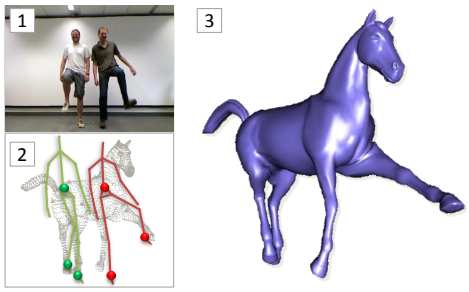

Figure 1: *Based on Kinect input (1,2) a novel animation of an existing mesh sequence is created (3). Three markers per person are sufficient to steer the horse legs.*

In summary, our method meets the following requirements for an interactive motion transfer system: 1) The motion style of the MoCap input is preserved. 2) Synthesis of poses not contained in the training sequence is possible. 3) Faithful reproduction of characteristic shape deformations from mesh examples. 4) Analysis is efficient (within minutes) and synthesis is interactive.

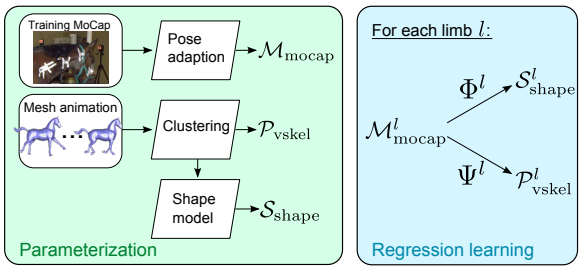

Figure 2: *Analysis phase of presented method (Sec. 4).*

2. Related Work

Working with mesh sequences has recently become an active field of research with two major trends. Trend one is to perform high level editing of mesh sequences, e.g. by transferring the global pose to different models [SP04, BVGP09], combining this with mesh editing approaches [XZY*07, FB11] or modifying the sequence in an abstract 2-D parameterization [CH12]. A second trend are data-driven deformation and enveloping approaches, e.g. in cloth simulation [KG08], anatomy-based animation [WPP07] and most recently in wrinkle synthesis of human hands [HZY*11].

Data-driven deformation. In [HZY*11], high quality 3D scans of human hands serve as training examples for wrinkle deformation with respect to hand poses. Huang et al. perform non-linear regression between a sparse set of control points and normal displacement maps derived from the scans. By restricting the influence region of each control point, a geometrically local deformation model is trained. Wang et al. [WPP07] propose to replace traditional linear blend skinning by shape spaces learned from examples. Contrary to our approach, correct kinematic skeletons for the examples are available, and the focus of [WPP07] is on quality, as opposed to robustness or sparse marker input.

Editing of mesh sequences. Deformation transfer [SP04, BSPG06] is a popular editing method for meshes, extended to mesh animations by Xu et al. [XZY*07]. The general approach is based on deformation gradients representing triangle rotations between compatible meshes. Applying deformation gradients of one mesh animation to a different mesh, Pose transfer is facilitated. However, changing the style of motion is not possible.

Automatic rigging/skinning. Automatic skeleton rigging and skinning approaches such as [JT05, BP07] also take additional shape information into account, although no shape models are employed. Example meshes are solely used to train blend skinning approaches, either by fitting skeletons to or by deriving blending weights from meshes. In [BP07], a kinematic skeleton is fitted robustly into a single static mesh which can in turn be animated. On the other hand, De Aguiar et al. [dATTS08] show how to fit a kinematic skeleton to a mesh animation. In [JT05] a large number of virtual bones

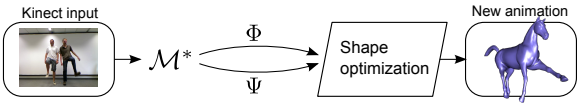

Figure 3: *Synthesis phase of presented method (Sec. 5).*

with skinning weights is derived from clustering triangle rotations, leading to a hardware efficient representation.

3. Overview of our method

We come up with a part-based model that distinguishes the mesh into limb and body parts. This separation allows a concise and robust parameterization of shape (per part) as well as pose (as relationship between parts). Pose and shape analysis produce low dimensional parameterizations for each part which in turn are connected via regression functions. The final synthesis step is formulated as a shape optimization problem that moderates between pose and shape predictions. The whole approach decomposes into an analysis phase and a synthesis phase, illustrated in Figures 2 and 3.

Distinction between *limbs* and *body* of a given model makes sense due to the following observation. Suitable input marker sets are required to provide information unique to articulated poses. These are typically given by positions of feet and hands (humans), hooves, wings or ears (animals) in relation to a reference (e.g. hip or spine) within the given body. In the context of searching human motion databases, Krüger et al. [KTWZ10] have successfully restricted to such information as well.

4. Learning a combined model for pose and shape

The combined model of pose and shape has to relate marker positions to mesh pose and shape. To this end suitable representations of marker input and mesh pose are required. A linear shape model computed on nearly rigid clusters of the mesh delivers a compact set of shape parameters (Sec. 4.4). Marker input, pose and shape of each limb l are linked by regression functions Φ^l and Ψ^l as

$$\mathcal{M}^l_{\text{mocap}} \xrightarrow{\Phi^l} \mathcal{P}^l_{\text{vskel}} \quad \text{and} \quad \mathcal{M}^l_{\text{mocap}} \xrightarrow{\Psi^l} \mathcal{S}^l_{\text{shape}} \qquad (1)$$

where $\mathcal{M}^l_{\text{mocap}}$ and $\mathcal{P}^l_{\text{vskel}}$ are the parameter sets for pose and $\mathcal{S}^l_{\text{shape}}$ for shape, respectively (Sec. 4.3).

Input to the training phase are m marker and n mesh vertex trajectories given as $m_i : \{1,\dots,F_M\} \to \mathbb{R}^3$ for $i = 1,\dots,m$ and $v_j : \{1,\dots,F_N\} \to \mathbb{R}^3$ with $j = 1,\dots,n$ over F_M MoCap and F_N mesh animation frames. Additionally, the user selects a corresponding vertex for each marker which results in a correspondence map $\tau\colon \{1,\dots,m\} \to \{1,\dots,n\}$.

4.1. Pose representation and parameterization

Pose parameters $\mathcal{M}^l_{\text{mocap}}$ are derived from the marker graph $G = (M, E_G)$ consisting of the used markers $M = \{1,\dots,m\}$

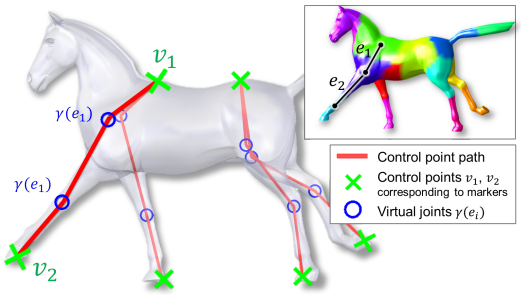

Figure 4: *Control points and clustering for the horse model.*

and edges $E_G \subset M^2$, roughly describing limb and body topology. The body is identified with a reference edge $e_{\text{ref}} \in E_G$, whereas limbs are simple paths in G. Considering the sparsest MoCap setting with one marker per end-effector, each limb path p_G will consist of a single edge (m_1, m_2) between two markers. Limb orientation is later parameterized relative to e_{ref}. For humans and animals e_{ref} is usually given by markers placed near the backbone, but in general, any nearly rigid edge can serve as reference edge.

On the mesh, a virtual skeleton graph $H = (C, E_H)$ is defined capturing adjacency of the mesh clusters and driving the parameterization $\mathcal{P}^l_{\text{vskel}}$. Nodes in $C = \{1, \ldots, k\}$ represent clusters while $E_H \subset H^2$ contains undirected edges for adjacent clusters. Since the mesh is dissected into (nearly) rigid parts the clustering reflects, to some extent, the bone and joint semantics of common animation skeletons. To parameterize the orientation between two clusters, *virtual joints* J are introduced at the barycenters of the cluster intersections. This is accomplished by attaching a virtual joint, $\gamma : E_H \to J$, to each edge.

Limbs in H are characterized by extracting a limb path from the marker graph G as follows. For each limb path $p_G = (m_1, m_2)$ in G we define a set of *control points*

$$p^l_{\text{ctrl}} = (v_1, \gamma(e_1), \ldots, \gamma(e_{|p^l_{\text{ctrl}}-2|}), v_2) \tag{2}$$

where vertices $v_1 = \tau(m_1)$ and $v_2 = \tau(m_2)$ correspond to markers m_1 and m_2. Intermediate virtual joints $\gamma(e_i)$ are defined by edges e_i along the shortest path in H between the clusters containing v_1 and v_2, see Fig. 4 for an illustration.

Pose parameterization. The pose parameters in $\mathcal{M}^l_{\text{mocap}}$ and $\mathcal{P}^l_{\text{vskel}}$ are outlined in Table 1. Rotation angles and edge lengths are derived from each limb path p_G within the marker graph. Based on the denser set of control points (2) in the virtual skeleton, unit quaternions representing local rotations can be computed. The direction of movement of the associated markers m_i and vertices v_i, computed via forward differences $d_{m_i}(f) = \frac{m_i(f+1) - m_i(f)}{\|m_i(f+1) - m_i(f)\|}$ and similarly for d_{v_i}, is considered for both parameter sets. Velocity would provide an alternative parameter but is less robust due to different sampling rates and movement styles of mesh animation and marker input.

Parameters for marker graph $\mathcal{M}^l_{\text{mocap}}$	
α_1	Angle between the e_{ref} and the first edge in p_G.
α_j	Angles between successive edges in p_G (for $j > 1$).
$\|e_j\|$	Lengths of edges in p_G.
d_{m_i}	Direction of movement of marker m_i.
Parameters for virtual skeleton $\mathcal{P}^l_{\text{vskel}}$	
q_1	Quaternion rotating between the mesh vertices corresponding to e_{ref} and the first edge $(v_1, \gamma(e_1))$ in p^l_{ctrl}.
q_j	Quaternion rotating between successive edges in p^l_{ctrl}.
d_{v_j}	Direction of movement of control point v_j.

Table 1: *Pose parameterization*

For $\mathcal{M}^l_{\text{mocap}}$ it is important to take edge lengths $\|e_j\|$ into account since we do not require any markers near real joint positions. Thus, the edges in the marker graph do not correspond to otherwise employed bones in animation skeletons. Accordingly, a change in length of these edges is a strong indicator for a possible bend of in-between joints which are available in the virtual skeleton.

4.2. Pose adaption

As implied, we do not expect the settings of marker input and and mesh model to be the same in an anatomical sense, nor do we require an overall equivalence of proportions between the two. Also, poses of animation input do not necessarily occur in the original mesh sequence. Contrarily, within a certain range, they are the basis on which new motion styles will be trained. This requires sensible pose adaption as preparation for further computations. To begin with, a best-frame fit between mesh sequence and input marker sequence is key to successful training. A mapping

$$f^*_l : \{1, \ldots, F_M\} \to \{1, \ldots, F_N\} \tag{3}$$

fits the former setting to the latter according to rotation angle conformity. That is angles θ_m, θ_v between (m_1, m_2) and (v_1, v_2) on each limb path are considered and we minimize

$$f^*_l(j) := \min_i \left(|\Delta_m(j) - \Delta_v(i)| + \frac{\sigma}{2}(1 - \text{sgn}(\Delta_m \Delta_v)) \right) \tag{4}$$

where $\Delta_m(f) = \theta_m(f+1) - \theta_m(f)$, Δ_v accordingly and $\sigma = \text{stdev}(\theta_m)$. Furthermore, the total and local ranges of motion between the settings will severely differ as exemplified by comparison of leg rotation angles in different equine gaits. To allow reasonable motion transfer, the variances within both sets need to agree. As a matter of fact, so should the variances of other corresponding parameter sets such as the lengths of graph edges in G and M. Meeting both above conditions calls for inverse kinematics to restore relative positions correctly.

4.3. Pose and shape regression

Training the relationship between parameterized pose information of the original input $\mathcal{M}^l_{\text{mocap}}$ and corresponding

pose parameters $\mathcal{P}^l_{\text{vskel}}$ will achieve meaningful pose transfer. Multivariate multiple regression serves well as a training mechanism for Φ^l in (1) and supports interpolation between trained motion parameters [Ren02]. Posed as a multivariate least squares problem

$$\min_{\Phi^l} \left\| \Phi^l \cdot \begin{bmatrix} \mathcal{M}^l_{\text{mocap}} \\ 1 \end{bmatrix} - \mathcal{P}^l_{\text{vskel}} \right\|^2_{\text{Frob}} \tag{5}$$

this means optimizing a prediction $\Phi^l \in \mathbb{R}^{s \times s'}$ with s, s' the respective dimensions of the parameter sets. The underlying statistical relationship between the same parameter input $\mathcal{M}_{\text{mocap}}$ and the part-shape information $\mathcal{S}^l_{\text{shape}}$ establishes the *shape* characteristics Ψ^l of the output animation in the same fashion.

4.4. Part-Shape model

A separate linear shape model [BV99] is established for each cluster of the mesh. Given n cluster vertices with index set $\{i_1, \ldots, i_n\}$ as a matrix

$$X = \begin{bmatrix} v_{i_1}(1) & \cdots & v_{i_1}(F_N) \\ \vdots & \ddots & \vdots \\ v_{i_n}(1) & \cdots & v_{i_n}(F_N) \end{bmatrix} = [\mathbf{x}_1 \cdots \mathbf{x}_{F_N}] \in \mathbb{R}^{3n \times F_N},$$

a set of eigenshapes V is derived via eigendecomposition of the scatter matrix $\frac{1}{F_N - 1}(X - \overline{X})(X - \overline{X})^T = V \Sigma^2 V^T$ with $\Sigma^2 = \text{diag}(\sigma_1^2, \ldots, \sigma_{F_N}^2)$. This leads to a linear shape model

$$\hat{\mathbf{x}} = \overline{\mathbf{x}} + V\lambda \tag{6}$$

where $\overline{\mathbf{x}} = \sum_i^{F_N} \mathbf{x}_i$ is the mean shape and $\lambda = (\lambda_1, \ldots, \lambda_k)^T$ are shape parameters. The Gaussian model assumption underlying (6) implies that λ_i should be normally distributed according to the shape variances σ_i^2.

Before the shape parameters $\mathcal{S}_{\text{shape}}$ can be derived, the mesh sequence has to be brought into a common coordinate system. This is done by rigidly aligning all mesh frames against an (arbitrarily chosen) reference frame . Afterwards, principal modes V of shape variation are computed. Due to clustering into nearly rigid parts, the shape variation of a single part can be compactly described by few linear modes. In our experiments we observe that the first two modes explain almost always more than 95 percent of shape variability. Projection of an aligned shape x into the space spanned by the first k modes $V_{1\ldots k}$ results in k shape coefficients:

$$s = (s_1, \ldots, s_k)^T = V_{1\ldots k}^T (\mathbf{x} - \overline{\mathbf{x}}), \quad s \in \mathcal{S}_{\text{shape}}$$

4.5. Clustering

Developing a part-based shape model depends upon a suitable method to derive cluster parts from a given mesh model. In our case, a clustering into near-rigid parts was performed by the compression method proposed by Sattler et al. [SSK05], comprising clustered PCA. This accommodates shape models of separate mesh parts with respect to trajectory resemblance. The inset in Fig. 4 shows an example clustering for the horse sequence.

For later rigid alignment and shape optimization we extend the disjoint partitioning from the clustering algorithm by overlaps between adjacent clusters. Vertices in the overlaps serve as constraints in mentioned optimizations connecting adjacent cluster shapes and orientations. For this purpose it is appropriate to simply extend a disjoint partitioning by adjacent vertices along cluster intersections.

5. Synthesis of new animations

After a short training phase, the conditioned pose and shape regression functions Φ and Ψ can be applied to new marker input. Synthesis starts by computing the parameterization $\mathcal{M}^*$ from the new marker input. Subsequently, $\Phi(\mathcal{M}^*)$ gives the correlated pose of the virtual skeleton, in terms of quaternions, whereas $\Psi(\mathcal{M}^*)$ yields the according shape parameters for the limbs.

Greedy alignment.

Shape optimized.

Shape and pose parameters can be conflicting such that direct assembly of synthesized shapes rotated by predicted quaternions leads to unpleasant artifacts as shown in the inset. These issues are addressed in an iterative optimization process. Its final goal is to find a rigid alignment for all clusters as well as shape parameters that produce an artifact free mesh respecting pose and shape predictions. Simultaneous optimization of both rigid alignment and shape parameters leads to a non-linear problem. Similar to [XZY*07,BBW*11] we approach the solution by an alternating least squares method. It efficiently solves for a rigid alignment (Sec. 5.1) keeping the shape parameters fixed and vice versa. Executing the shape optimization for a few iterations will return a consistent alignment and shape parameters for all clusters (Sec. 5.2). After blending multiple occuring clusters (Sec. 5.3) the final output mesh is assembled.

5.1. Greedy rigid alignment

Rotation and translation have to be estimated simultaneously for all mesh clusters. Avoiding an intricate global solution we greedily align pairwise adjacent clusters according to their overlap. Starting with the largest cluster, the adjacent cluster with most overlapping vertices is chosen. The two clusters are rigidly aligned and merged into a super-cluster for further alignment. Repeating this greedy procedure will finally return a single super-cluster with all mesh clusters consistently aligned. Keeping track of rotations and trans-

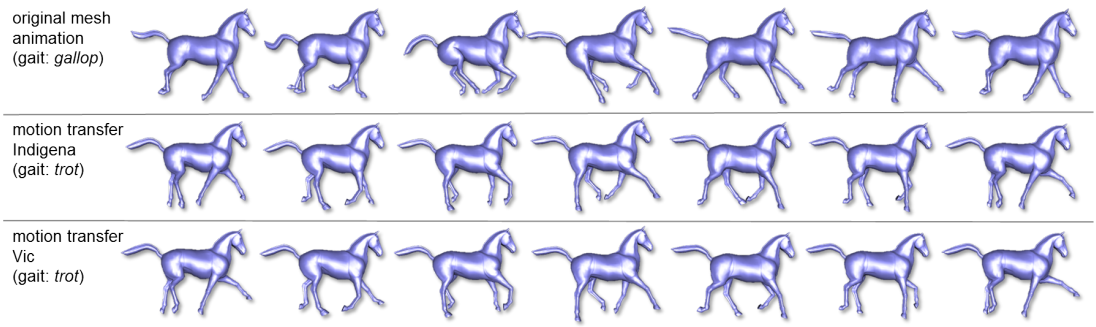

Figure 5: *Motion transfer from MoCap data of Indigena dataset to Sumner horse sequence. For comparison, frames with similar pose in right fore leg are aligned column wise.* (First row:) *Original Sumner horse sequence.* (Second row:) *Motion transfer synthesized on Indigena trot sequence.* (Third row:) *Motion transfer synthesized on Vic trot sequence. Note that the trot sequences are similar to each other but differ severely in motion style from the gallop of the original mesh animation.*

lations in this process a rigid transformation for each mesh cluster is found.

5.2. Shape optimization

Given a set of prescribed vertex coordinates $\{x_i\}$, $x_i \in \mathbb{R}^3$ we optimize the shape $\hat{\mathbf{x}} \in \mathbb{R}^{3n}$ to match $\hat{x}_i = x_i$. As a further constraint we want the shape to be "plausible" that is force it to stay close to the Gaussian shape model. We use the formulation of Berner et al. [BBW*11, BBW*12] and express the shape optimization as the following minimization

$$\lambda^* := \min_{\lambda} \sum_{i \in C} \|x_i - \hat{x}_i(\lambda)\|^2 + \beta \cdot \sum_{i=1}^{k} \frac{\|\lambda_i - \lambda_i^+\|^2}{\sigma_i^2} \qquad (7)$$

which yields optimized shape parameters λ^*. The first term assures that the fixed vertices x_i in the overlap to adjacent clusters are matched while the second term regularizes the predicted shape weighted with a user parameter β. We introduce prior knowledge about "plausible" shape parameters here via λ_i^+ and penalize deviations from these. This allows the limbs to adhere to the shape predicted by $\Psi(\mathcal{M}^*)$ while body clusters are initialized with the mean shape prior $\lambda_i^+ = 0$.

Eq. (7) leads to an over-determined linear system $A\lambda = b$ of $3k$ equations which is solved efficiently in a least squares fashion. Note that the number of fixed vertices k in the cluster overlaps is only a small fraction of the mesh vertices.

5.3. Blending of shared clusters

So far we omitted the fact that a mesh cluster can be contained in several limbs. This is a typical situation in coarse clustering where the limbs meet at a common point (e.g. the hip in a human model). We call this *shared clusters*. For the optimization process we simply duplicate shared clusters in the virtual skeleton graph and treat them independently. This is important, since the shape parameters can be conflicting for poses very different from the ones in the mesh animation.

Blending a cluster shared by m limbs is realized through blend functions w_i, one for each limb. At any boundary vertex v to a specific limb l weights are $w_l(v) = 1$ and $w_i(v) = 0$ for $i \neq l$. Based on minimum geodesic distance $d_l(v)$ from vertex v to the boundary vertices of limb l, smooth linear interpolation weights are given by

$$w_l(v) = \frac{\sum_{i=1}^{m} d_i(v) - d_l(v)}{\sum_{i=1}^{m} d_i(v)} \qquad (8)$$

Note that the blending functions do not depend on the synthesis parameters and can be pre-computed.

6. Results

We implemented a Matlab prototype of our motion transfer system which, though not optimized, synthesizes several frames per second.

Motion transfer of quadrupeds. Our method manages the translation of motion attributes tracked from quadruped locomotion to a given mesh sequence. Results of training equine gait samples (trot) on a sequence of horse mesh frames displaying a different gait (canter) are shown in Fig. 5. As expected, they expose characteristic differences in overall pose as well as local attributes according to the pose dissimilarities of the featured gaits. As a matter of fact, these examples also display slight shape distortion within certain parts of the model, e.g. hooves. These were exhibited by original mesh samples in mild form and are exaggerated by restriction to linear regression as training method. Resorting to kernel CCA [FKY08] for training purposes should help suppress such phenomena. Synthesis on a motion sequence performed by a different individual subsequent to the above training was conducted in addition. The results constitute the third row of Fig. 5 and display gait style characteristics of the second individual compared to the first.

Motion transfer of bipeds. Similarly, a biped bunny mesh was trained to perform a straightforward walk learned on an example sequence of human locomotion. Alternative synthesis on a different walking sequence involving a curve

results in the correct change of walking direction and posture in the succeeding animation, please see the accompanying video. Within this context, the semantic quality of the transfer is apparent since the skeleton proportions of selected mesh and motion capture input differ considerably. All motion capture sequences were taken from the HDM05 database [MRC*07].

Interactive steering of animations by Kinect input. Since creating mesh animations from motion capture data has proved to be feasible, steering mesh animations by user interaction can be tackled as well. Kinect input of human locomotion - performed jointly by two actors - were transferred to the mesh model setup. Results are shown in Fig. 1 and the accompanying video.

7. Conclusion and Future Work

The work at hand describes an efficient way towards more effortless creation of new digital content from existing material. The key idea is combining available mesh sequences with a variety of motion sensing input. In particular, since motion sensing input devices are emerging at consumer level, the method points to new paradigms in the field. We presented a method for effective motion transfer from sparse marker input data to mesh sequences. Mindful pose adaption and shape optimization achieves plausible results in different scenarios. We demonstrated that the proposed method comes with a variety of applications such as semantic deformation transfer, interactive steering of mesh animations and motion style transfer to bi- and quadrupeds.

Current limitations are foot skating artifacts and lack off ground contact, which can e.g. be modeled as hard constraints in the synthesis step. So far, extrapolation quality depends on the particular input sequences. Considering more complex examples are expected to improve results, but will probably require non-linear methods in the analysis phase.

Future work will focus on generalizing the essential idea of transferring sparse input signals to part-based pose and shape models. Investigating alternative input signals, e.g. audio, is one such generalization. Employing more abstract shape spaces which model features of given data on a higher semantic level is another. The latter could involve modeling motion properties of more complex nature.

Acknowledgements. This work was partially supported by Deutsche Forschungsgesellschaft within the priority program SPP1335. We thank Rebeka Zsoldos for supporting us with horse motion capture data, Robert Sumner for the horse mesh and the Blender Foundation for the bunny sequence under a Creative Commons Attribution 3.0 License.

References

[BBW*11] BERNER A., BURGHARD O., WAND M., MITRA N. J., KLEIN R., SEIDEL H.-P.: *A Morphable Part Model for Shape Manipulation*. Tech. rep., MPI Informatik, 2011. 4, 5

[BBW*12] BERNER A., BURGHARD O., WAND M., MITRA N. J., KLEIN R., SEIDEL H.-P.: Morphable components for modeling from shape collections. *ACM Trans. Graphics* (2012). (in preparation). 5

[BP07] BARAN I., POPOVIĆ J.: Automatic rigging and animation of 3d characters. *SIGGRAPH 26*, 3 (2007). 2

[BSPG06] BOTSCH M., SUMNER R., PAULY M., GROSS M.: Deformation transfer for detail-preserving surface editing. *Vision, Modeling and Visualization (VMV)* (2006), 357–364. 2

[BV99] BLANZ V., VETTER T.: A morphable model for the synthesis of 3d faces. *SIGGRAPH* (1999), 187–194. 4

[BVGP09] BARAN I., VLASIC D., GRINSPUN E., POPOVIĆ J.: Semantic deformation transfer. *SIGGRAPH 28*, 3 (2009), 36:1–36:6. 1, 2

[CH12] CASHMAN T. J., HORMANN K.: A continuous, editable representation for deforming mesh sequences with separate signals for time, pose and shape. *Eurographics* (2012), 735–744. 2

[dATTS08] DE AGUIAR E., THEOBALT C., THRUN S., SEIDEL H.-P.: Automatic Conversion of Mesh Animations into Skeleton-based Animations. *Eurographics 27* (4 2008). 2

[FB11] FRÖHLICH S., BOTSCH M.: Example-driven deformations based on discrete shells. *Computer Graphics Forum 30*, 8 (2011), 2246–2257. 2

[FKY08] FENG W.-W., KIM B.-U., YU Y.: Real-time data driven deformation using kernel canonical correlation analysis. *SIGGRAPH 27*, 3 (2008), 91:1–91:9. 5

[HZY*11] HUANG H., ZHAO L., YIN K., QI Y., YU Y., TONG X.: Controllable hand deformation from sparse examples with rich details. In *SCA* (2011), pp. 73–82. 1, 2

[JT05] JAMES D. L., TWIGG C. D.: Skinning mesh animations. *SIGGRAPH 24*, 3 (2005). 2

[KG08] KIRCHER S., GARLAND M.: Free-form motion processing. *SIGGRAPH 27*, 2 (2008), 12:1–12:13. 2

[KTWZ10] KRÜGER B., TAUTGES J., WEBER A., ZINKE A.: Fast local and global similarity searches in large motion capture databases. In *SCA* (July 2010), pp. 1–10. 2

[MRC*07] MÜLLER M., RÖDER T., CLAUSEN M., EBERHARDT B., KRÜGER B., WEBER A.: *Documentation Mocap Database HDM05*. Tech. rep., Universität Bonn, June 2007. 6

[Ren02] RENCHER A. C.: *Methods of Multivariate Analysis*. John Wiley and Sons, 2002. 4

[SP04] SUMNER R. W., POPOVIĆ J.: Deformation transfer for triangle meshes. *SIGGRAPH 23*, 3 (2004), 399–405. 2

[SSK05] SATTLER M., SARLETTE R., KLEIN R.: Simple and efficient compression of animation sequences. *SCA* (2005). 4

[TDM11] TENA J. R., DE LA TORRE F., MATTHEWS I.: Interactive region-based linear 3d face models. *SIGGRAPH 30* (2011), 76:1–76:10. 1

[WBLP11] WEISE T., BOUAZIZ S., LI H., PAULY M.: Realtime performance-based facial animation. *SIGGRAPH 30*, 4 (July 2011), 77:1–77:10. 1

[WPP07] WANG R. Y., PULLI K., POPOVIĆ J.: Real-time enveloping with rotational regression. *SIGGRAPH 26*, 3 (July 2007). 2

[XZY*07] XU W., ZHOU K., YU Y., TAN Q., PENG Q., GUO B.: Gradient domain editing of deforming mesh sequences. *SIGGRAPH* (2007), 84–94. 2, 4