EUROGRAPHICS 2014 / B. Lévy and J. Kautz
(Guest Editors)

Volume 33 (2014), Number 2

Interactive Motion Mapping for Real-time Character Control

Helge Rhodin[1], James Tompkin[1,2], Kwang In Kim[1,3], Kiran Varanasi[1,4], Hans-Peter Seidel[1], Christian Theobalt[1]

[1]Max-Planck-Institute for Informatics, [2]Intel Visual Computing Institute, [3]Lancaster University, [4]Technicolor Research

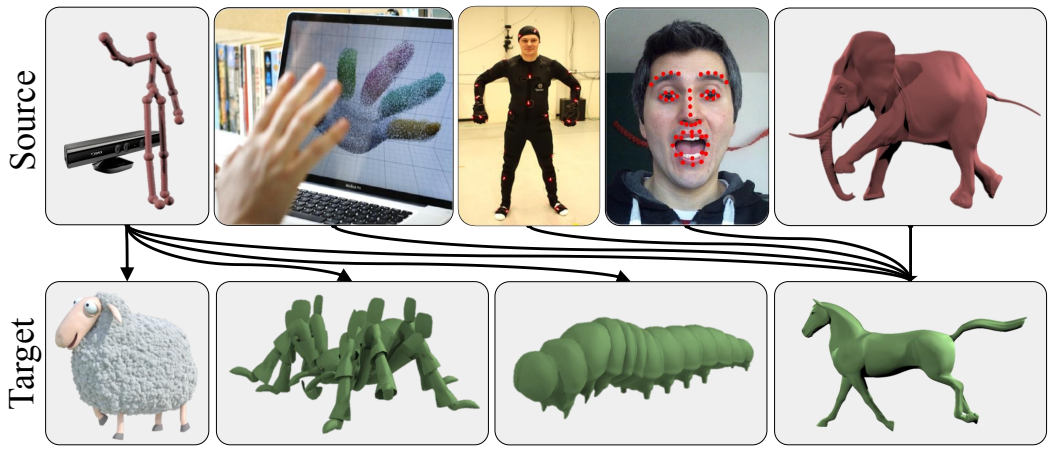

Figure 1: *Different motions are easily mapped between very different morphologies for fast and versatile real-time character control. We require only 4-8 interactively defined correspondences and support 3D point sequences and meshes as input.*

Abstract

It is now possible to capture the 3D motion of the human body on consumer hardware and to puppet in real time skeleton-based virtual characters. However, many characters do not have humanoid skeletons. Characters such as spiders and caterpillars do not have boned skeletons at all, and these characters have very different shapes and motions. In general, character control under arbitrary shape and motion transformations is unsolved - how might these motions be mapped? We control characters with a method which avoids the rigging-skinning pipeline — source and target characters do not have skeletons or rigs. We use interactively-defined sparse pose correspondences to learn a mapping between arbitrary 3D point source sequences and mesh target sequences. Then, we puppet the target character in real time. We demonstrate the versatility of our method through results on diverse virtual characters with different input motion controllers. Our method provides a fast, flexible, and intuitive interface for arbitrary motion mapping which provides new ways to control characters for real-time animation.

1. Introduction

New consumer input devices provide opportunities to interactively animate virtual characters. For example, the Microsoft Kinect controller outputs the 3D positions of human body skeletal joints in real time. With traditional skeleton-based animation pipelines, human joint data can be mapped to a virtual character so long as it has a similar skeleton to a human, thus providing character control. However, often non-humanoid characters need to be controlled. For instance, spiders can be parameterized by skeletons with different numbers of limbs. Characters like caterpillars have no obvious bone rigging and, even if a skeletal mapping could be defined, direct human skeletal control would be difficult. In these cases, mapping and retargetting of human motions would require time consuming manual work by animators. In general, retargetting has practical limitations, and the question of how to map arbitrary motions remains.

Existing character control algorithms cannot map arbitrary source and target motions, and new approaches need to fundamentally generalize. One approach is to deform

meshes by using joints as deformation constraints, e.g., to embody and move a chair [CIF12]. Another approach controls a quadrupedal target skeleton by mapping the motion to two humans, similar to a pantomime horse [VHKK12]. However, these approaches do not scale to target character shape and motions that are very different from human.

Solutions to the generalized motion retargetting problem need to map a source input space to the target output space of a virtual character, even if that character is structurally dissimilar to a human and even if its motion is not well represented by a skeleton. It should be possible to map an undulating arm motion to a crawling caterpillar motion, or to map a jockey saddle bounce motion to a horse gallop motion, all without explicitly specifying spatial correspondences.

We move towards this goal with a solution for interactive mapping and real-time character control. Our approach is *general* and abstracts from the classical skeleton-rig-based motion parameterization which is not suitable for mapping structurally different motion spaces. We take as input a *reference* source sequence of sparse 3D points or meshes, such as full body, hand, or face motion from any motion capture system. This approach bypasses time consuming and potentially error prone skeleton reconstruction. We also take as input a target character as a sequence of meshes, and no additional control rig is required. Thus, any animation type can be puppeted by our system, be it keyframe animation, physics-based simulation, or rig-based animation.

To achieve this, we represent both motions in dedicated feature spaces and learn a mapping between these intermediate representations, with bounds stemming from a *latent volume* which constrains the predicted poses. Latent volumes are bound from training motions which reduce the risk of causing artifacts if new *live* source motions for puppetry are unlike the reference source motions, and so they remove the need for the puppeteer to restrict their poses only to those correspondences defined. Further, this makes our approach robust, and mappings learned for one source can be successfully applied to other sources, such as in training/repetition scenarios or across actors.

Our approach is fast, flexible, and user friendly as our algorithm only requires the specification of 4-8 temporal correspondences between source and target sequences. We neither require spatial correspondences, e.g., limb mapping, nor correspondences to every pose in the target character motion. This enables an efficient *animate-correspond-synthesize* workflow: The user interactively corresponds their body poses to the target character, the mapping is learned in a few seconds, and then new animation is synthesized in real-time. As this cycle is very fast, users can iterate to find which source mappings are most intuitive and make on-the-fly control adjustments. This naturally leads to applications in simple motion and party games; however, our approach also gives new tools to animation artists. In conversations with animators, they found our system useful to 'give life' to existing animations through creative puppetry, for example, to produce variation when animating crowds or when adding timing nuances to a walk cycle through finger tracking. In figures and in a supplementary video, we show the performance and reliability of our method with a variety of real-time control examples with different source and target motions, such as mapping full body motion of a human to a caterpillar and mapping face motions onto a sheep.

We summarize our contributions:

- A real-time algorithm that can map between characters with different topology from sparse correspondences.
- A latent volume representation that efficiently exploits unlabeled data to allow robust character control.
- An automatic keyframe suggestion method to support the user during correspondence selection.

These create a robust and easy to set up system for interactive motion mapping of characters with arbitrarily different shapes and motions, which opens the door for more creative and intuitive real-time character animation.

2. Related Work

Creating a skeleton rig for a character to be animated in real time is a non-trivial and time-consuming task even for animators [GGP*00]. Skeleton-based character control with captured motion data requires the solution of a complex retargetting problem [Gle98], since the dimensions and topology of source and target skeletons may differ. Reusing motion behavior across characters is not straightforward, as motion retargetting produces awkward artifacts even when the characters are similar in geometry.

Recent research in animation has suggested solutions to simplify or bypass the skeleton-based character animation process. Hecker et al. map skeletal motions to user-created target characters with different morphologies [HRE*08]. New control rigs and control parameters are estimated by a particle-based inverse kinematics solver which makes the character move appropriately. Bharaj et al. automatically estimate a skeleton for multi-component characters, and describe a way of mapping source motion to the inferred target rig which succeeds as long as each target subchain can be assigned to an equivalent subchain in the source rig [BTST11]. However, both of these approaches still rely on skeletons and are hard to map to soft-bodied creatures.

Data-driven animation schemes parameterize a character motion space by example motions. Animation researchers have experimented with dimensionality reduction techniques for data reduction and for designing new motion controllers [AM01, KRFC09], and both linear mappings and non-linear approaches have been investigated. In linear frameworks, the movement space can be considered to be spanned by a set of basis shapes [TYAB01], or by a set of basis trajectories for each point [ASKK10]. In this new basis space, the character can be controlled by modifying the

influence of basis shapes or trajectories. Achter et al. propose a bilinear spatio-temporal basis that describes oscillations around a set of example shapes [ASK*12]. Some animation types cannot be well represented in a linear framework, and so these techniques might lead to a small latent control space with few meaningful dimensions. Non-linear dimensionality reduction techniques, such as Gaussian process latent variable models (GPLVM), kernel methods, or multi-dimensional scaling (MDS), have also been applied to animation parametrization [LWH*12]. For example, Cashman and Hormann project arbitrary motions onto a 2D control plane, obtained through MDS, to create new animations as paths within that latent space [CH12].

In contrast to these approaches, where each dimension in the latent space affects the surface deformation globally, direct local parameterizations in the source space are also feasible. James and Twigg represent general mesh deformations with a set of proxy bones and skinning weights [JT05]. Kavan et al. and Jacobson et al. extend this framework for fast and automatic computation [KSO10, JBK*12]. Dynamics can be integrated into the model by considering previous frames of motion. de Aguiar et al. use projection into a linear latent space in combination with a second-order linear dynamic system to simulate cloth motion [dASTH09], whereas Wang et al. use a non-linear system built through Gaussian process dynamic models [WFH08].

All of these data-driven methods propose parameterizations for single characters, but do not analyze their suitability for real-time motion mapping between very different characters. Feng et al. extract a set of control points from a single mesh animation and use kernel canonical correlation analysis (CCA) between control points and mesh to reproduce fine-scale surface details from the articulated movements of an underlying skeleton [FKY08]. In our context, since the animation is synthesized based only on control points from a single mesh, their motion transfer approach can only be applied when source and target characters are identical or similar in structure. Baran et al. map deformations of a source character to a target character by means of mesh deformation [BVGP09]. The method relies on a set of example temporal correspondences that encode the semantics of the mapping not usable for point based data as in many of our input sources. Zhou et al. extend the framework of Baran et al. to multi-component objects [ZXTD10]. One key difference of our approach is that we implicitly build a pose prior from all unlabeled samples in the target motion sequence, and not just for the given correspondences. We experimentally show the benefits of exploiting unlabeled data.

Our work also bears similarity to work on animation of deformable characters. For example, Coros et al. build controllers for purposeful motion of soft deformable characters via rest state adaptation in an elastic simulation [CMT*12]. The KinÊtre algorithm [CIF12] maps joint positions captured with Kinect onto a target character by means of a mesh deformation framework. Vögele et al. map the skeleton motions of two humans in real time to a quadripedal target skeleton, e.g., by mapping one skeleton to the front half of a horse, and one to the back [VHKK12]. In both these works, this mapping is limited to target characters with parts or subskeletons resembling a human skeleton; further, the motion mapping is an exact mapping between sources and target. In contrast, we can map different source motions to different target motions, even if source and target skeletons differ greatly, and even if no skeleton is available.

Puppeteering techniques allow direct control of a character with a dedicated input device [Stu98]. Dontcheva et al. obviate the common manual selection of corresponding degrees of freedom by CCA on a dense temporally aligned training sequence [DYP03]. Seol et al. puppet characters of different topology; however, detailed manual selection of features and a character rig are required [SOL13].

Yamane et al. describe an off-line approach to map the exact motions of a human to a target humanoid character, e.g., a hopping motion to make a Luxo lamp hop, or a toddle to control the walk of a penguin [YAH10]. With 30-50 correspondences hand-distributed at important poses, they build a mapping between source and target in a shared latent space obtained by GPLVM. As this mapping is noisy, they regularize with a post-process inverse kinematics solve with the target skeleton, as well as a further post-process to ensure foot planting. Our approach differs in many key aspects: 1) We do not require any pre-existing statisical relationship between source and target motions, as is required by their shared latent space. We demonstrate mappings between very different source and target motions. 2) We employ efficient linear models to enable real-time control. 3) Our method succeeds with as few as 4 correspondences, which are automatically suggested by our method and interactively refined in a simple interface. These key differences enable new real-time animation possibilities and more user-friendly control.

3. Method Overview

We wish to control a set of artist-created target motions — a character. First, we learn a high-dimensional oriented bounding box, or *latent volume*, which represents the space of controllable target poses (Fig. 4, left, and Sec. 5.1). This offline training computation takes about a minute. We wish to puppet the target motions using arbitrary source motions, and so the next step is for the user to interactively define a small number of pose correspondences (4 to 8) between example or *reference* source motions and the desired target motions (Fig. 4, middle, and Sec. 5.2). The reference source motions need only be performed once, and help is provided with automatic suggestions of appropriate poses. From these correspondences, we learn a mapping between reference source and target motions (Fig. 4, right, and Sec. 5.3), and this takes less than one second. This completes the learning stage.

In the synthesis phase, *live* source motions are inputted to

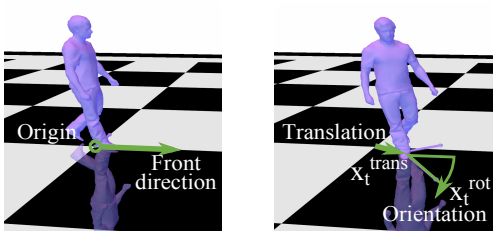

Figure 2: *Global translation $\mathbf{x}_t^{trans}$ and rotation $\mathbf{x}_t^{rot}$ of a general pose (*right*) extracted in relation to the rest pose* (left).

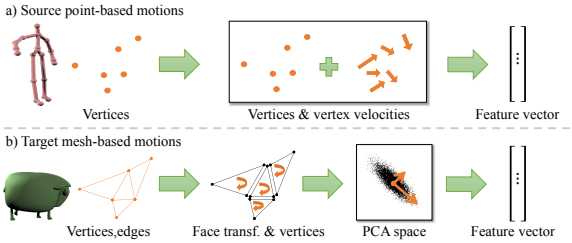

Figure 3: *a) Source point-based characters are represented as vertex positions and velocities. b) Target mesh characters are decomposed into vertex positions and face transformations to model rotations explicitly, and are projected into low-dimensional PCA space to reduce complexity.*

the learned motion transfer mapping to synthesize new motion sequences of the target character in real time (Fig. 6, and Sec. 6). Live motions are similar repetitions of the reference source motions used to define correspondence, and can vary in intensity, e.g., faster or slower, or arms raised less high, and can be performed simultaneously to cause motion combination or *superposition* effects. In most cases, the input reference and live source motions will both come from a tracking system such as Kinect, LeapMotion, or a face tracker, which creates a system for interactive motion mapping and real-time character control.

In all our diagrams, the input reference source motion used in training and the input live source used in synthesis are shown in red, the input target motion is in green, and the output synthesized motion is in blue.

4. Character Representation

Our source motion $(\mathbf{x}_1,...,\mathbf{x}_M)$ is a 3D point sequence with M frames, from tracking devices like Kinect or from vertices in a mesh sequence. Target character motions $(\mathbf{y}_1,...,\mathbf{y}_N)$ are mesh sequences. Our representation must separate global motion from character pose, e.g., to isolate a walk cycle from world movement, so that we can independently map these to different controls. We parametrize a motion into character pose feature vectors $(\mathbf{x}_1^{\text{pose}},...,\mathbf{x}_M^{\text{pose}})$, global translations $(\mathbf{x}_1^{\text{trans}},...,\mathbf{x}_M^{\text{trans}})$, and global rotations $(\mathbf{x}_1^{\text{rot}},...,\mathbf{x}_M^{\text{rot}})$.

4.1. Global Motion

For each frame in the source $\mathbf{x}_t$ (or target $\mathbf{y}_t$), we estimate the global position and orientation of the character on the ground plane as an offset to $\mathbf{x}_1$ by the least squares fit of $\mathbf{x}_t$ to $\mathbf{x}_1$ using orthogonal Procrustes analysis [Sor09] (Fig. 2). Global motion is represented by 3 degrees of freedom: a translation vector $\mathbf{x}_t^{\text{trans}} \in \mathbb{R}^2$ and a yaw rotation angle $\mathbf{x}_t^{\text{rot}} \in \mathbb{R}$.

4.2. Pose

Source Point-based Representation We represent character pose as the concatenated feature vector of 3D point positions and their velocities after compensating for global motion and mean centering. The velocities help disambiguate similar poses in simple source motions (e.g., raising vs. lowering limbs). For a pose $\mathbf{x}_t$ at time t, velocity vector $\dot{\mathbf{x}}_t = (\mathbf{x}_t - \mathbf{x}_{t-1})/\Delta t$, where Δt is the time between two frames. The source pose vector is $\mathbf{x}_t^{\text{pose}} = (\mathbf{x}_t^\top, w\dot{\mathbf{x}}_t^\top)^\top$, which is 6V in length for a character of V vertices. Factor w balances the contribution of static and dynamic information in the regression and is set to 0.1. For noisy vertex position input, we apply a small one-sided Gaussian filter to smooth temporally, with a 3-frame standard deviation (≈ 0.1 sec.).

Target Mesh Representation The simple point-based representation leads to strong distortions if used for target mesh characters. We remove distortions by exploiting information contained in the connectivity of the mesh. We extend the *deformation gradient* representation, which models surface deformation by combining per face transformations [SP04], with explicit vertex features (Fig. 3). We discuss the strength of this representation against alternatives in Section 7.1.

For each mesh face f we extract the affine transformation A_f in relation to the rest pose and decompose it into rotation and shear by polar decomposition. Rotations are compactly stored in axis-angle form and the six degrees of freedom of the symmetric shear matrix are linearized to a vector. To model the absolute position of potentially disconnected mesh components, which is not included in the original deformation gradient representation, we additionally store all vertex positions $\mathbf{v}_i$. Therefore, each pose vector $\mathbf{y}_t^{\text{pose}}$ is a concatenation of $3F$ rotation, $6F$ shear, and $3V$ point parameters for a character with F faces and V vertices.

Dimensionality Reduction We reduce the dimensionality of mesh representations to $D = 50$ by principal component analysis (PCA). D was chosen experimentally such that the dimensionality is drastically reduced while still preserving more than 99% of the original variance, and this improves the memory footprint and performance of the mapping for our real-time scenario. Henceforth, we refer to the resulting feature vector as $\mathbf{y}_t^{\text{pose}} \in \mathbb{R}^D$. The reconstruction from this

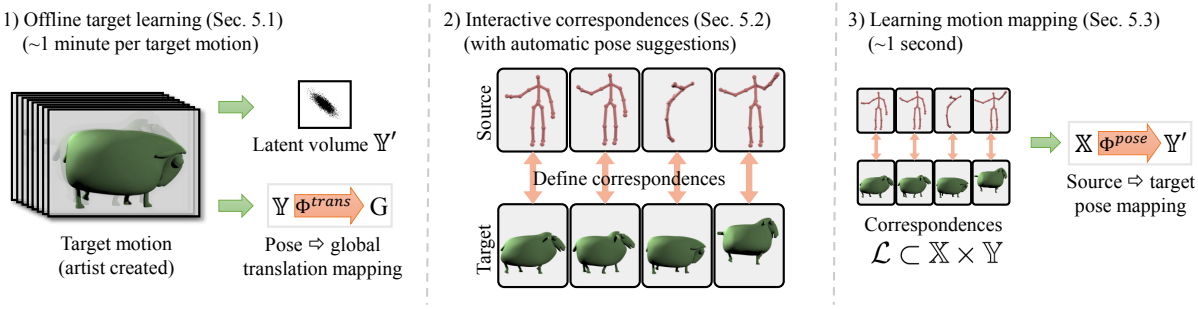

Figure 4: *Learning a motion mapping: 1) Offline, an artist creates an unlabeled target motion. We learn latent volume $\mathbb{Y}'$ and the dependency Φ^{trans} between changes in pose and global translation G. 2) Sparse key pose correspondences are automatically suggested and then interactively refined. 3) The map Φ^{pose} between source space $\mathbb{X}$ and target space $\mathbb{Y}$ is learned.*

low-dimensional feature vector is explained when we come to synthesize a new motion in Section 6.

5. Learning a Motion Mapping

We aim to learn a motion transfer function $\Phi : \mathbb{X} \rightarrow \mathbb{Y}$ between the space of source poses $\mathbb{X}$ and the space of target poses $\mathbb{Y}$ based on a sparse set of *labeled* pose correspondences $\mathcal{L} \subset \mathbb{X} \times \mathbb{Y}$.

This is an instance of the regression problem. In this context, previous works have successfully applied combinations of non-linear feature extraction (or data representation, e.g., GPLVM) and regression (e.g., kernel-based Gaussian process regression (GPR)) to offline character animation [YAH10, VHKK12, LWH*12]. However, our application poses new challenges that are not well addressed in existing systems: 1) only a limited number of labeled correspondences are given, and 2) character animation synthesis should be real-time. Fortunately, unlike typical regression, we have a set of *unlabeled* examples $\mathcal{U}_\mathbf{y} \subset \mathbb{Y}$ which is larger than the labeled training set of correspondences. We exploit this additional information on the potential variations to assist the regression process and estimate a *latent volume* $\mathbb{Y}'$ that effectively encodes and limits the predicted pose.

As such, our goals are three-fold (Fig. 4): 1) to learn a mapping Φ^{trans} between pose and global translation for the target sequence $(\mathbf{y}_1, ..., \mathbf{y}_N)$, and to learn the range of admissible target motion as a latent volume $\mathbb{Y}'$ (Sec. 5.1); 2) to generate pose correspondences $\mathcal{L}$ between reference source motions and target motions, in which we adopt Bayesian regression to suggest appropriate candidates (Sec. 5.2); and 3) from these correspondences, to learn a motion transfer function Φ^{pose} between source and target poses (Sec. 5.3).

5.1. Offline Target Learning

The offline learning includes two steps (Fig. 4, left): learning a mapping Φ^{trans} which links pose to global translation, and the computation of the latent volume $\mathbb{Y}'$.

Global Translation Map Global character motion $(\mathbf{y}_t^{\text{trans}}, \mathbf{y}_t^{\text{rot}})$ often depends on the change of pose $\dot{\mathbf{y}}_t^{\text{pose}}$, e.g., a pose in a run motion should cause more global translation than a pose in a walk motion. We learn a relationship between pose and translation with a linear map Φ^{trans} from the target pose velocities, which uses the heuristic that a quicker pose change leads to quicker locomotion:

$$\Phi^{\text{trans}} \Leftarrow \arg\min_{\mathbf{W}} \sum_{t=1,...,N} \|\mathbf{W}\,|\dot{\mathbf{y}}_t^{\text{pose}}| - \dot{\mathbf{y}}_t^{\text{trans}}\|^2 + \|\sigma_{\text{trans}} W\|^2, \tag{1}$$

where velocities are estimated by finite differencing, $|\mathbf{y}|$ is the coefficient-wise absolute value of vector $\mathbf{y}$, and σ_{trans} is the regularization parameter ($= 0.001$). We apply Φ^{trans} during synthesis to recover global translation, and separately apply rotations from the live source motion (Sec. 6).

Learning a Latent Volume We represent pose as a vector $\mathbf{y}_t^{\text{pose}}$, an element of $\mathbb{R}^D$. However, characters are physical objects whose pose variations are constrained by their bodies, and the plausible range of variation in $\mathbf{y}_t^{\text{pose}}$ is significantly smaller than $\mathbb{R}^D$. For instance, human pose is limited by joints, while caterpillar and crab poses are constrained by their skins and carapaces.

Traditionally, these constraints have been explicitly constructed through virtual skeletons with fine-tuned degrees of freedom and joint limits. In our case, we expect arbitrary source and target animations and, accordingly, we have no a priori knowledge about the admissible range of motions. Our approach is to exploit the unlabeled data $\mathcal{U}_\mathbf{y}$ for inferring implicitly the admissible variations through a high-dimensional oriented bounding box, or *latent volume*. By assuming that the distribution of poses is approximated by a Gaussian distribution, we find the principal axes of variation by decorrelating $\mathcal{U}_\mathbf{y}$ with PCA and estimating the range of motion to be the minimum interval I_c encompassing all data points in $\mathcal{U}_\mathbf{y}$ across PCA components c. The effectiveness of this operation is empirically demonstrated in Section 7.

5.2. Interactive Correspondence Selection

Given the latent volume of the target sequence, our next goal along the way to defining the pose motion transfer function Φ^{pose} is to label corresponding pose frame pairs $\mathcal{L} = \{(\mathbf{x}^{\text{pose}}_{l_1}, \mathbf{y}^{\text{pose}}_{l_1}), \ldots, (\mathbf{x}^{\text{pose}}_{l_n}, \mathbf{y}^{\text{pose}}_{l_n})\}$ between the reference source and target sequences, where each label element $(l_1, \ldots, l_n)$ contains indices for $\mathbf{x}^{\text{pose}}$ and $\mathbf{y}^{\text{pose}}$ in their respective frame ranges $[1, M]$ and $[1, N]$. From existing work, we might think that time-warping is a suitable method to align reference source and target character sequences by synchronising spatial correspondences; however, this assumes a priori temporal or spatial correlations. For instance, Vögele et al. use time-warping to align the legs of two humans to the front and rear limbs of a horse [VHKK12], but this 'limbed creature' spatial correlation assumption fails for arbitrary motions and characters.

In arbitrary cases, selecting good correspondences between reference source motions and target motions requires some skill as it is not always intuitive which source motions will lead to good control. To ease this process, we exploit a Bayesian regression model (detailed in Sec. 5.3) to provide assistance in two different use cases. As correspondences are defined and as $\mathcal{L}$ increases in size, this model provides us with the most probable corresponding pose $\mathbf{y}^{\text{pose}}_*$ for the current source pose $\mathbf{x}^{\text{pose}}$, where the probability density $P(\mathbf{y}^{\text{pose}}|\mathbf{x}^{\text{pose}})$ is inferred from the present set of correspondences $\mathcal{L}$ and where the variance corresponds to the uncertainty of the prediction. Therefore, we choose the predicted variance as a metric $q(\mathbf{x}^{\text{pose}})$ to suggest correspondences which will make mappings with good control.

Performance-based Often it is natural to perform the desired reference source motions, and so in this mode the user defines correspondences to target poses by performing the reference source motions and timing button presses. For instance, when capturing reference source motions with Kinect, our system uses a hand-held remote-controlled trigger which the user activates when their poses align to the desired target poses. To help the user, q is shown as a bar which increases in size and changes color from red to green when the performed pose is underrepresented by the present correspondence selection, i.e., when the performed pose explains much of the remaining variance (Fig. 5, bottom).

Automatic Correspondence Suggestion In other cases, the reference source motion is captured offline, or the user may wish to refine performance-based correspondences. For each target correspondence $\mathbf{y}^{\text{pose}}_l$, we suggest the 5 largest local maxima of $q\left(\mathbf{x}^{\text{pose}}_1, \ldots, \mathbf{x}^{\text{pose}}_M\right)$ from the reference source as candidates. Users can accept one of the suggestions, or are able to make adjustment with a pose time slider. After acceptance, the pose correspondence is added to $\mathcal{L}$, metric q is instantly recomputed for the updated $\mathcal{L}$, and new suggestions are proposed for the next correspondence (Fig. 5, top).

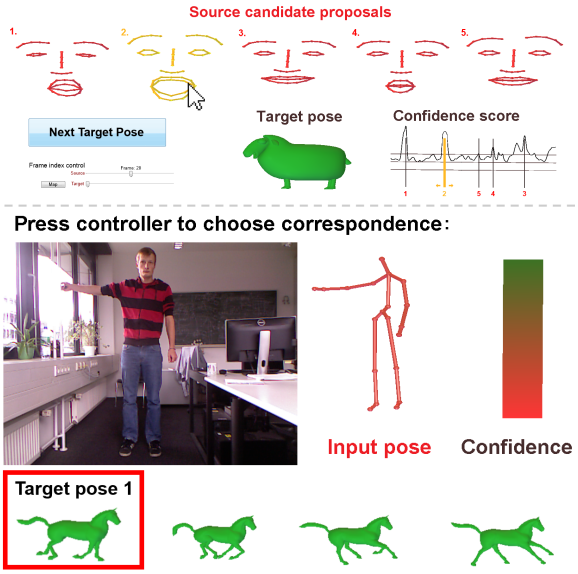

Figure 5: Top: *Automatic correspondence suggestion for mapping source face tracking data to a sheep character. The confidence plot score allows users to fine-tune the suggestions.* Bottom: *Performance-based correspondence definition, where the confidence of the current pose being a good controller is visualized to the user as a colored bar.*

5.3. Learning a Pose Mapping

Given our correspondences, the next step is to learn Φ^{pose}, the pose transfer function. We construct a linear map $\Phi^{\text{pose}}(\mathbf{x}^{\text{pose}}) = \mathbf{M}\mathbf{x}^{\text{pose}}$ that fits to a given set of labeled correspondences $\mathcal{L}$. Matrix $\mathbf{M}$ is of size $D \times$ *input pose dimension*. Adopting the standard Bayesian linear regression framework, we apply an identical isotropic Gaussian prior to each row of $\mathbf{M}$; $\text{row}(\mathbf{M}) \sim \mathcal{N}(\mathbf{0}, I)$ with $\mathcal{N}(\mathbf{m}, C)$ being a Gaussian distribution with mean vector $\mathbf{m}$ and covariance matrix C, and a Gaussian noise model:

$$\mathbf{y}^{\text{pose}} = f(\mathbf{x}^{\text{pose}}) + \varepsilon I, \tag{2}$$

where f is to be estimated and $\varepsilon \sim \mathcal{N}(0, \sigma^2)$. We set the noise value σ^2 to 0.05 for all our results; manual tuning may further improve results. We concatenate $\mathbf{x}^{\text{pose}}$ with 1 to model a constant offset between source and target.

Marginalizing the prior of $\mathbf{M}$ and the likelihood (Eq. 2) over $\mathbf{M}$, we obtain the predictive Gaussian distribution on the output target pose $\mathbf{y}^{\text{pose}}$ given test input $\mathbf{x}^{\text{pose}}_*$ [RW06]:

$$p(\mathbf{y}^{\text{pose}}|\mathbf{x}^{\text{pose}}_*, \mathcal{L}) = N(\sigma^{-2} Y X^\top A^{-\top} \mathbf{x}^{\text{pose}}_*, \\ \mathbf{x}^{\text{pose}\top}_* A^{-1} \mathbf{x}^{\text{pose}}_* I), \tag{3}$$

where $A = \sigma^{-2} X X^\top + I$ and matrices $X = (\mathbf{x}^{\text{pose}}_{l_1}, \ldots, \mathbf{x}^{\text{pose}}_{l_n})$, $Y = (\mathbf{y}^{\text{pose}}_{l_1}, \ldots, \mathbf{y}^{\text{pose}}_{l_n})$ are formed from $\mathcal{L}$. The map Φ^{pose} is then defined as the mode $\mathbf{y}^{\text{pose}}_* = \sigma^{-2} Y X^\top A^{-\top} \mathbf{x}^{\text{pose}}_*$ of the

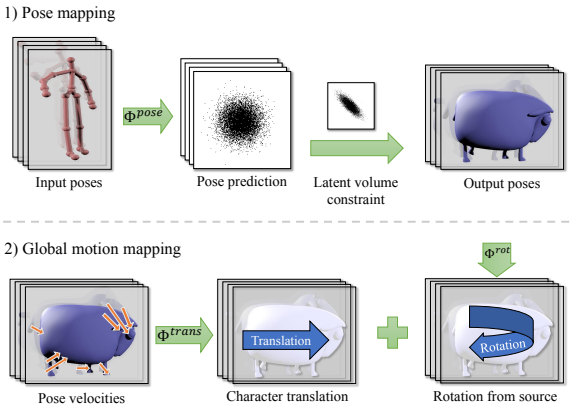

Figure 6: *Synthesis: 1) Given a new input pose, the distribution of most likely target poses is inferred through Φ^{pose} (Sec. 5.3) and constrained by the latent volume to obtain the final target character pose. 2) The global translation is inferred through Φ^{trans} from pose velocities (Sec. 5.1) and source rotation is mapped to output rotation by Φ^{rot} (Sec. 6).*

predictive distribution for test input $\mathbf{x}_*^{\text{pose}}$. Computing Φ^{pose} is equivalent to ridge regression. The Bayesian framework provides the variance $\mathbf{x}_*^{\text{pose}\top} A^{-1} \mathbf{x}_*^{\text{pose}}$ of the predictive distribution which corresponds to the uncertainty on the made prediction. This information is exploited as metric $q(\mathbf{x}_*^{\text{pose}})$ for suggesting label candidates as shown in Sec. 5.2. We apply Φ^{pose} to live source motions in the next section.

6. Synthesis

With pose mapping Φ^{pose}, we can now sequentially map endless live source motions $(\cdots, \chi_{t-1}, \chi_t, \cdots)$ to create new output motions $(\cdots, \gamma_{t-1}, \gamma_t, \cdots)$. Input χ and output γ are novel, and are not contained in the training data $(\mathbf{x}_1, ..., \mathbf{x}_M), (\mathbf{y}_1, ..., \mathbf{y}_N)$. As summarized in Fig. 6, synthesizing each output frame γ_t is performed in two steps: 1) Inferring the pose γ_t^{pose} by Φ^{pose} on χ_t^{pose}, constrained by the latent volume; and 2) Applying the global translation map and rotation map $\Phi^{\text{trans}}, \Phi^{\text{rot}}$ on the newly synthesized pose γ_t^{pose} to obtain global motion velocities $\dot{\gamma}_t^{\text{trans}}, \dot{\gamma}_t^{\text{rot}}$. Finally, these are integrated to yield $\gamma_t^{\text{trans}}, \gamma_t^{\text{rot}}$.

The latent volume constrains a new pose estimate γ_t^{pose} by projecting onto the PCA space and clipping each PCA component c to the respective interval I_c. The effect is that of a high dimensional bounding box in $\mathbb{R}^D$ whose sides are aligned with the principal axes of pose variation. This ensures that the outputs are strictly contained in the volume spanned by $\mathcal{U}_{\mathbf{y}}$, and unwanted deformations of the target character are prevented (see Sec. 7.1). Except for the latent volume bounding, all operations are linear and only depend on the most recent frames. This allows real-time control.

In principle, a rotation map Φ^{rot} could be learned in a similar way to Φ^{trans}; however, in practice this requires target sequences to contain sufficient examples of character rotation, and this is often not the case. Instead, the target rotation is controlled directly from source rotation χ_t^{rot} through $\Phi^{\text{rot}} : \chi^{\text{rot}} \rightarrow r\chi^{\text{rot}}$, with r manually adjusted to the agility of the character. Practically, this leads to intuitive control where rotating the body in front of Kinect also rotates the character.

6.1. Mesh Reconstruction

While the motion mapping is simple, we still need to return from our dedicated feature space. This is more complicated for mesh sequences as we decompose them into vertex positions $\mathbf{v}_i$ and per-face rotations and shears A_f, and project these into PCA space (Sec. 4). Thus, given a new pose γ_t^{pose} in feature space, first, the PCA coefficients are back-projected to obtain face transformations A_f and positional features $\mathbf{v}_i$. After the mapping procedure, the transformations of A_f and $\mathbf{v}_i$ might be inconsistent and might yield a disconnected surface. We reconstruct a coherent surface by solving a Poisson system similar to Sumner et al. [SP04] (see supplemental document for a more detailed explanation).

7. Evaluation

Our dataset consists of 8 different motions. The *CMUHuman* mocap data from the CMU Motion of Body Database, the *KinectHuman* sequences captured with a Kinect sensor, and the *LeapMotion* sequence from the Leap motion controller are all point-based sequences. *Face* contains 66 2D feature points tracked automatically from a video sequence. *Elephant*, *Horse*, *Caterpillar*, and *Sheep* are mesh sequences created by artists containing different motions. The data acquisition processes for *KinectHuman* and *LeapMotion* are real-time. Table 1 details these sequences and the number of correspondences between them.

The (hyper-)parameters of our algorithm are: temporal motion derivative weight $w = 0.1$ (Sec. 4), latent volume dimensionality $D = 50$ (Sec. 4.2), pose and translation regression regularization parameters $\sigma = 0.05, \sigma_{\text{trans}} = 0.001$ (Sec. 5.3), and per-character rotation angle r (Sec. 6). Except for r, which is a data-dependent fixed ratio, these parameters were tuned once and fixed throughout the entire set of experiments. Experiments were performed at 30-40fps on an Xeon 3.6Ghz Quad-Core for meshes of 3-16k triangles.

Figure 7 (left) shows how *KinectHuman-Handwave* is transfered to *Horse*. Although our algorithm was not explicitly directed to focus on the right hand in synthesizing the target, it successfully inferred the desired effect from only 4 correspondences. The hand waving speed controls the target motion speed. Any combination of source and target is possible — we replace *KinectHuman-Handwave* with *KinectHuman-Jockey* in Figure 7 (right), and now knee bounces control the target.

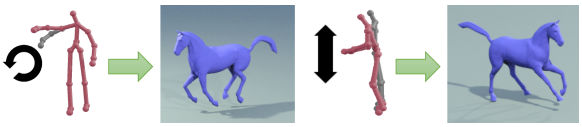

Figure 7: KinectHuman-Handwave *arm motions* (left) *and* KinectHuman-Jockey (right) *knee bend motions synthesize* Horse *gallops.*

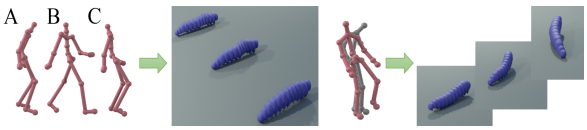

Figure 8: *Examples of motion transfer to a caterpillar.* Left: *Mapping learned from* CMU Human-Walk *style C and applied on different styles A,B,C varying in speed and step size, which displays the capability of style transfer.* Right: KinectHuman-Various *and the synthesized* Caterpillar.

Figure 8 (left) shows an example where a *CMUHuman* walk is transferred to a *Caterpillar* that has different topology in static shape and demonstrates completely distinct dynamics in motion. With only 4 labels, our algorithm successfully recovered the underlying correspondence in class. Further, this mapping can be directly applied to other source motions, and we puppet the caterpillar with stylized *slow*, *medium*, and *fast* walks.

Figure 8 (right) shows a more challenging scenario with walking, leaning, raising of the thorax, and jumping. This wider range of motions was covered by 8 interactively defined example correspondences. In Figure 11 and in the supplemental video, we show that the linearity of our mapping allows for realistic superposition of motions such as turn and crawl, head-lift and bend, and jump out of a head-lift pose.

In Figure 9, we show the transfer of facial motion between a human (*Face*) and a sheep (*Sheep-Face*) using only 6 example correspondences. The video shows correspondence refinement following automatic correspondence suggestion.

Figure 10 shows the last example in which a *KinectHuman* controls a *Sheep*. Local arm and global body motions of the source sequence are assigned to walking and jumping of the sheep based on 6 correspondences. Our algorithm well-generalizes these different classes of motions. In practical applications, a fine-detailed geometry (such as fleece) can be added to generate a visually pleasing character.

7.1. Alternative Approaches

The advantage of the combined shape space is shown in Figure 13 with a map from an *Elephant* to *Horse*. Raw point features lead to distortions when synthesizing rotational motions and the deformation gradient representation is agnostic

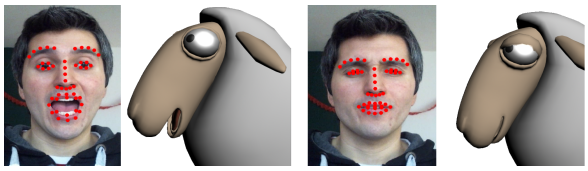

Figure 9: *Examples of facial motion transfer learned from 6 correspondences. The first and second columns:* Face *and the second and fourth columns: the synthesized* Sheep.

Table 1: *Target character frame length, walk cycles frame length, and the number of source-target pose correspondences for each sequence.* † *denotes Kinect and Elephant.*

Source char. Target char.	Kinect Caterp.	† Horse	Kinect Sheep	CMU Caterp.	Leap,Face Sheep	Mocap Lamp
Target frames	700	12	57	100	100	35
Target walk cycle length	20	12	18	150	n/a	n/a
# correspondences	8	4	6	4	6	8

to vertical translation. The latent volume and use of unlabeled data is compared to pure PCA in Figure 12. The gain from the remaining method components, such as the velocity features and regularization, are especially noticeable in motion; our supplemental video shows these gains alongside a comparison of our model to weighted nearest neighbor lookup and to the method of Yamane et al. [YAH10].

8. Discussion

Our final algorithm is a design that reflects the requirements imposed by the application scenario: interactive correspondence declaration, real-time mapping, and versatility to very different source and target sequences. Here, we briefly discuss how these choices were made.

Linear Regression The choice of a linear regressor over non-linear algorithms is not driven by the complexity but by the wish to succeed with sparse labels. With only 4-8 examples, the training and learning time of Gaussian process regression is comparable to linear regression but requires careful tuning of additional parameters. However, in our setting, the quality of outputs obtained from the linear regression was approximately equivalent to the output from non-linear ones.

Even for the linear regressor (with limited complexity), the problem is severely ill-posed as the number of correspondences is much smaller than the dimensionality of input and output data points. Exploiting a priori knowledge is essential for regularizing the regression process. Fortunately, unlike typical regression, we exploit *unlabeled* examples $\mathcal{U}_{\mathbb{Y}} \subset \mathbb{Y}$ to estimate a *latent volume* $\mathbb{Y}'$ that effectively encodes and limits the plausible motion range.

Latent Volume One of our main contributions is that we demonstrate that a simple orthogonal transform followed by

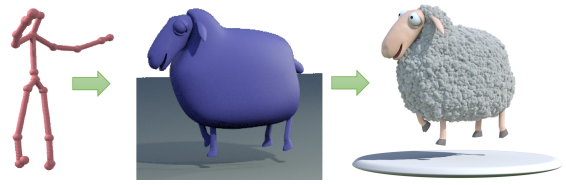

Figure 10: *With appropriate rendering, our system can create expressive characters. We integrate our motion transfer method into a professional animation-rendering pipeline to show transfer from sparse input to fine geometry.*

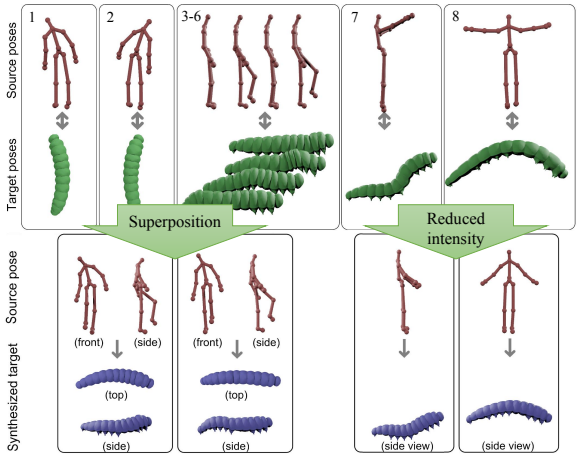

Figure 11: Top: *Correspondences labeled between joint positions and caterpillar mesh to control bending (1-2), crawl (3-6), head lift (7) and jump (8).* Bottom: *New pose synthesis of the caterpillar character showing intermediate and newly inferred superposition of, e.g., bend and crawl.*

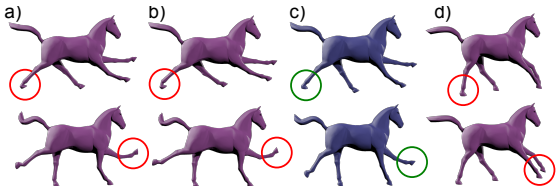

Figure 12: *Comparison of our method to related mapping techniques. The horse pose is synthesized from Kinect input through a) a linear map without unlabeled examples, b) in a PCA latent space of dimension 10, c) by our method, and d) our method without unlabeled examples. Artifacts are marked in red, successfully regularization in green.*

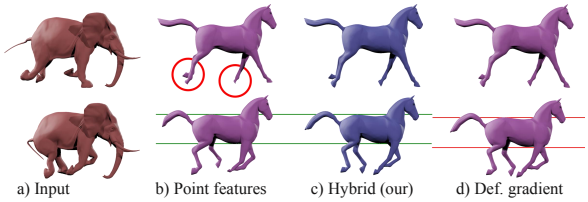

Figure 13: *Result of deformation transfer using a) point features that lead to shrinkage artifacts, b) our artifact free representation, and c) the deformation gradient representation which does not capture vertical translations.*

independent bounding of variables is effective for regularizing the regression process (see Sec. 5.1). Recently, automatic approaches have been proposed for implicitly inferring the constraints from the data using principal component analysis (PCA) or GPLVM.

In preliminary experiments, classical use of PCA for estimating constraints — PCA followed by significant dimensionality reduction — was insufficient for our purpose. Furthermore, flexible GPLVMs or sophisticated non-linear density estimators are too inefficient for interactive applications due to their high computational complexity in learning and prediction. Learning can take several hours even for low dimensional skeleton characters [LWH*12], and prediction has yet to be achieved in real time [YAH10].

Our latent volume representation is the result of the trade-off between performance and quality. In general, when properly regularized, sophisticated non-linear algorithms should lead to as good or better representations of plausible motion spaces; unfortunately, these algorithms cannot be used in real-time applications. Direct bounding by the sparse correspondences without considering the unlabeled data for decorrelation and support estimation is insufficient as it improperly restricts the admissible range and thereby leads to jerky animations (see supplemental video).

8.1. Limitations

Currently, our method yields too coarse a control to steer characters with broader dynamics. For example, only the basic facial expressions are transferred to the sheep in Figure 9. To enable this, regression and latent volume representation may need to resort to more flexible non-linear alternatives while retaining the computational efficiency for real-time applications. Applying the current method on clustered motion categories is one step in this direction.

During synthesis, we do not address collision detection, interaction of multiple components, or interaction with the environment. To achieve this, a post-processing similar to Yamane et al. would have to be extended to work in real time on mesh characters [YAH10].

Our algorithm is only applied to single source/target sequences. For complex puppetry applications, it might be helpful to allow multiple humans to control (different parts of) the same target. Conversely, a single human could control the movement of a swarm of small characters as a whole.

9. Conclusion

We have presented a data driven method for real-time control of virtual characters that offers fast and flexible interactive motion mapping. Our contribution is a mapping from source to target characters that ensures robustness by limiting the range of admissible poses through the use of a dedicated latent volume. Unlike other approaches, we interactively define only 4-8 correspondences between sequences, and so we must robustly learn this mapping by using unlabeled frames. This allows puppetry with many different motion sources such as body-, face-, and hand-tracking systems. Combined with the automatic correspondence suggestion, this leads to an animation system for quickly defining mappings between very different shapes and motions, and then intuitively controlling characters in real time.

Acknowledgements

We thank Gottfried Mentor and Wolfram Kampffmeyer for their character animations and helpful feedback and Robert Sumner and Jovan Popovic for the horse and elephant models. This work was partially supported by the ERC Starting Grant CapReal.

References

[AM01] ALEXA M., MÜLLER W.: Representing animations by principal components. *CGF (Proc. Eurographics) 19*, 4 (2001), 411–418. 2

[ASK*12] AKHTER I., SIMON T., KHAN S., MATTHEWS I., SHEIKH Y.: Bilinear spatiotemporal basis models. *ACM TOG 31*, 2 (2012), 1–12. 3

[ASKK10] AKHTER I., SHEIKH Y., KHAN S., KANADE T.: Trajectory space: a dual representation for nonrigid structure from motion. *IEEE TPAMI 33*, 7 (2010), 1442–1456. 2

[BTST11] BHARAJ G., THORMÄHLEN T., SEIDEL H.-P., THEOBALT C.: Automatically rigging multi-component characters. *CGF (Proc. Eurographics) 30*, 2 (2011), 755–764. 2

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

[CH12] CASHMAN T. J., HORMANN K.: A continuous, editable representation for deforming mesh sequences with separate signals for time, pose and shape. *CGF (Proc. Eurographics) 31*, 2pt4 (2012), 735–744. 3

[CIF12] CHEN J., IZADI S., FITZGIBBON A.: KinÊtre: animating the world with the human body. In *Proc. UIST* (2012), pp. 435–444. 2, 3

[CMT*12] COROS S., MARTIN S., THOMASZEWSKI B., SCHUMACHER C., SUMNER R., GROSS M.: Deformable objects alive! *ACM TOG (Proc. SIGGRAPH) 31*, 4 (2012), 69:1–69:9. 3

[dASTH09] DE AGUIAR E., SIGAL L., TREUILLE A., HODGINS J. K.: Stable spaces for real-time clothing. *ACM TOG (Proc. SIGGRAPH) 29*, 3 (2009), 106:1–106:9. 3

[DYP03] DONTCHEVA M., YNGVE G., POPOVIĆ Z.: Layered acting for character animation. *ACM TOG (Proc. SIGGRAPH)*, Siggraph (2003), 409. 3

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

[GGP*00] GLEICHER M., GRASSIA S., POPOVIC Z., ROSNTHAL S., THINGVOLD J.: Motion editing: Principles practice and promise. In *SIGGRAPH 2000 Course Notes 26* (2000). 2

[Gle98] GLEICHER M.: Retargeting motion to new characters. In *Proc. SIGGRAPH* (1998), pp. 33–42. 2

[HRE*08] HECKER C., RAABE B., ENSLOW R. W., DEWEESE J., MAYNARD J., VAN PROOIJEN K.: Real-time motion retargeting to highly varied user-created morphologies. *ACM TOG (Proc. SIGGRAPH) 27*, 3 (2008), 27:1–27:9. 2

[JBK*12] JACOBSON A., BARAN I., KAVAN L., POPOVIĆ J., SORKINE O.: Fast automatic skinning transformations. *ACM TOG (Proc. SIGGRAPH) 31*, 4 (2012), 77:1–77:10. 3

[JT05] JAMES D. L., TWIGG C. D.: Skinning mesh animations. *ACM TOG (Proc. SIGGRAPH) 24*, 3 (2005), 399–407. 3

[KRFC09] KRY P., REVERET L., FAURE F., CANI M.-P.: Modal Locomotion: Animating Virtual Characters with Natural Vibrations. *CGF (Proc. Eurographics) 28*, 2 (2009), 289–298. 2

[KSO10] KAVAN L., SLOAN P.-P., O'SULLIVAN C.: Fast and efficient skinning of animated meshes. *CGF (Proc. Eurographics) 29*, 2 (2010), 327–336. 3

[LWH*12] LEVINE S., WANG J. M., HARAUX A., POPOVIĆ Z., KOLTUN V.: Continuous character control with low-dimensional embeddings. *ACM TOG (Proc. SIGGRAPH) 31*, 4 (July 2012), 1–10. 3, 5, 9

[RW06] RASMUSSEN C., WILLIAMS C.: *Gaussian Processes for Machine Learning*. Adaptative computation and machine learning series. University Press Group Limited, 2006. 6

[SOL13] SEOL Y., O'SULLIVAN C., LEE J.: Creature features: online motion puppetry for non-human characters. In *Proc. SCA* (2013), SCA '13, ACM, pp. 213–221. 3

[Sor09] SORKINE O.: Least-squares rigid motion using SVD. *Technical notes, ETHZ* (2009). 4

[SP04] SUMNER R. W., POPOVIĆ J.: Deformation transfer for triangle meshes. *ACM TOG (Proc. SIGGRAPH) 23*, 3 (2004), 399–405. 4, 7

[Stu98] STURMAN D.: Computer puppetry. *Computer Graphics and Applications, IEEE*, February (1998). 3

[TYAB01] TORRESANI L., YANG D., ALEXANDER G., BREGLER C.: Tracking and modeling non-rigid objects with rank constraints. In *Proc. CVPR* (2001), pp. 493–500. 2

[VHKK12] VÖGELE A., HERMANN M., KRÜGER B., KLEIN R.: Interactive steering of mesh animations. In *Proc. SCA* (2012), pp. 53–58. 2, 3, 5, 6

[WFH08] WANG J. M., FLEET D. J., HERTZMANN A.: Gaussian process dynamical models for human motion. *IEEE TPAMI 30*, 2 (2008), 283–298. 3

[YAH10] YAMANE K., ARIKI Y., HODGINS J.: Animating non-humanoid characters with human motion data. In *Proc. SCA* (2010), pp. 169–178. 3, 5, 8, 9

[ZXTD10] ZHOU K., XU W., TONG Y., DESBRUN M.: Deformation transfer to multi-component objects. *CGF (Proc. Eurographics)* (2010), 319–325. 3