To appear in ACM TOG 34(6).

Generalizing Wave Gestures from Sparse Examples for Real-time Character Control

Helge Rhodin[1], James Tompkin[2], Kwang In Kim[3], Edilson de Aguiar[4],
Hanspeter Pfister[2], Hans-Peter Seidel[1], Christian Theobalt[1]
[1]MPI for Informatics, [2]Harvard Paulson SEAS, [3]Lancaster University, [4]Federal University of Espirito Santo

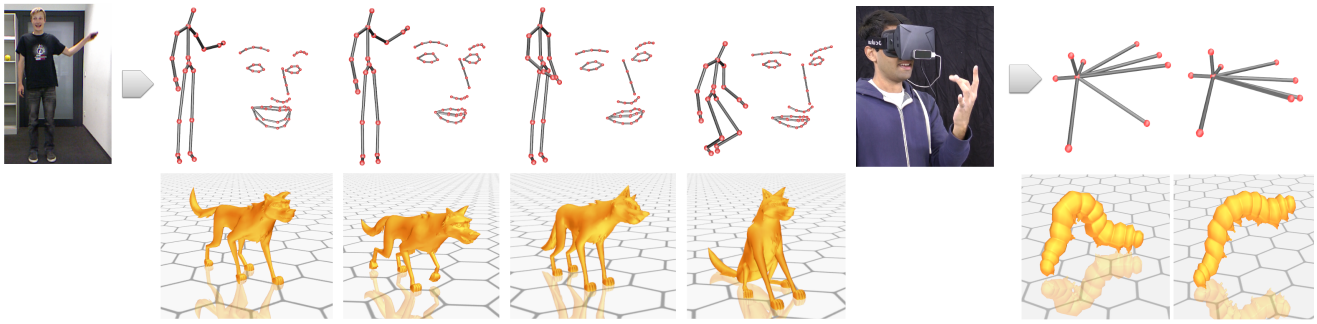

Figure 1: Top: *Skeletal, facial, and hand motions are tracked by off-the-shelf sensors. From sparse user-defined examples, our method reliably separates and extrapolates simultaneously-performed gestures to control characters in games or VR.* Bottom left: *Dog happy walk, neutral run, shaking, and sitting.* Bottom right: *Caterpillar crawl variations, controlled by varying amplitude, phase, and frequency of input gestures.*

Abstract

Motion-tracked real-time character control is important for games and VR, but current solutions are limited: retargeting is hard for non-human characters, with locomotion bound to the sensing volume; and pose mappings are ambiguous with difficult dynamic motion control. We robustly estimate wave properties — *amplitude, frequency, and phase* — for a set of interactively-defined gestures by mapping user motions to a low-dimensional independent representation. The mapping separates simultaneous or intersecting gestures, and extrapolates gesture variations from single training examples. For animations such as locomotion, wave properties map naturally to stride length, step frequency, and progression, and allow smooth transitions from standing, to walking, to running. Interpolating out-of-phase locomotions is hard, e.g., quadruped legs between walks and runs switch phase, so we introduce a new time-interpolation scheme to reduce artifacts. These improvements to real-time motion-tracked character control are important for common cyclic animations. We validate this in a user study, and show versatility to apply to part- and full-body motions across a variety of sensors.

CR Categories: I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction techniques I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation;

Keywords: Virtual character control, motion mapping, dynamics.

1 Introduction

Real-time character control is important for games and virtual worlds. Recent motion controllers using low-cost body and hand skeletal trackers (Microsoft Kinect, Leap Motion) are particularly promising for virtual reality (VR), where typical interfaces may be impractical due to impaired real-world vision. The possible mappings between human motion and character motion span a spectrum of control: Retargeting motion controllers [Gleicher 1998] map human and character bodies 'one-to-one' at the bone level. While expressive, character motion is consequently restricted to the sensing volume and to the ability of the user to perform complex motions, with mappings hard to generalize to non-human characters. At the other end of the spectrum, gestural control-action pairs [Johnson et al. 1999; Raptis et al. 2011] trigger character motions from a database, but there is no control of motion style variation through extrapolation.

Recent techniques [Seol et al. 2013; Rhodin et al. 2014] lie in-between, by mapping poses to character animations. This forfeits retargeting, but expands control flexibility and expressivity over gesture-action pairs, and is a good fit for games. However, there are problems: pure pose mappings are under-constrained for complex character motion, and so velocity and acceleration features add constraints. These are sensitive to noise, and may result in jerky or stilted animations. Further, a user moving his legs faster will induce a *faster playing* walk animation, when the desire is for a progression of dynamism from walking to running. While in principle this could be accomplished with multiple maps, in practice this is difficult and better extrapolation of control tempo is needed.

Inspired by Fourier domain representations [Unuma et al. 1995; Shiratori and Hodgins 2008], we robustly estimate instantaneous wave properties — *amplitude, frequency, and phase (AFP)* — of high-dimensional user pose motions. Our versatile method allows interactive part- or full-body control motion definition via diverse capture sensors. Key to our approach is the separation of simultaneously-performed control motions into independent low-dimensional waves. We learn this mapping interactively from a single example per control motion, such that wave properties can be estimated by windowed Fourier analysis and extrapolated to control motion variations. With a mapping learned, instantaneous real-time estimates are still challenging as future user input is unknown. We overcome this by exploiting the temporal dependence of phase and frequency.

Cyclic animations are common, and our wave gestures are often well suited to cyclic control tasks, e.g., for walking locomotion, frequency maps to number of steps per second, amplitude to step size, and phase to walk progression. This allows control over variations in style which would be complex for existing pose mapping approaches, such as a slow wide steps or fast narrow steps. Non-cyclic linear motions are also needed, e.g., sitting down or raising an arm. Seol et al. [2013] classify input motions and blend different pose

mappings to improve cyclic motion control. We extend this idea by allowing superpositions of both linear pose mappings and cyclic motion mappings, all within a motion graph. Further, our approach extends to additional input modes with shared parameter spaces, e.g., a face tracker measuring users mood to control character emotion.

We generate new animations by interpolating wave properties within parametric animation database spaces. Our general interpolation scheme operates on meshes, is independent of a character rig or skeleton, supports arbitrary character shapes and topologies, and enables foot sliding prevention. For locomotion, often quadrupeds switch the phase of individual limbs between walks to runs. We propose to reduce classical artifacts by aligning limbs individually by time warping, and interpolating limb timing differences separately from limb positions. This minimizes the size of the character animation database needed for smooth dynamic state changes.

We show that wave gestures are more robust to noise and operate over a larger temporal domain than commonly-used velocity and acceleration features [Seol et al. 2013; Rhodin et al. 2014], and that they successfully decorrelate ambiguous control motions, which is not possible with existing approaches [Shiratori and Hodgins 2008]. Further, with ten participants in three game-like quantitative tasks and in qualitative questionnaires, we discover that our wave gesture control is more intuitive and more accurate for animation tempo control than a gamepad, but is more physically demanding. For applications, we interviewed three animation professionals, who found potential as a 'blocking' tool for early-stage content creation, for live performance, and for games.

At a system level, we provide key features when compared to existing methods (Tab. 1). Our contributions to the literature are:

- A technique to robustly and accurately estimate amplitude, frequency, and phase of simultaneous gestures in real time, generalized from a single user-defined reference motion.
- An interpolation method for motions with out-of-phase sub-motions that cannot be aligned by traditional time-warping.
- A live animation system, which couples wave gesture to parametric motion graphs and layers different input modalities.

2 Related work

Direct puppetry. The transfer of user performance onto virtual characters via motion tracking control has been used successfully in animation for many years. Sturman [1998] reviews early work in character control, describing pedals, gloves, joysticks, and body suits to empower multiple users to orchestrate control of an animated character, similar to classical puppetry [Oore et al. 2002]. Layering time-sequential performances allows for rich animations even from mouse and keyboard input [Dontcheva et al. 2003; Neff et al. 2007; Martin and Neff 2012]. Acting out animations proves useful for animation timing [Terra and Metoyer 2004], but control becomes difficult for complex characters and motions.

Physical simulation. Direct manipulation of selected character degrees of freedom (DOF) can be eased by adding physical simulation. Coros et al. [2012] couple mouse input via rest state adaptation in an elastic physical simulation, and Laszlo et al. [2000] couple mouse and keyboard input with physical simulation of bipedal motion through proportional derivative controllers. These works can be more broadly applied with a system for robust input generalization from a single reference control motion, which is our focus.

Retargeting. Gleicher [1998] showed that user motion can be retargeted onto new bipedal characters with different body proportions through a set of position objectives and motion frequency constraints. Shin et al. [2001] added importance sampling through the proximity of end effectors to the environment. Retargeting can generalize to characters of different topology by duplicating and mirroring [Hecker et al. 2008]. Ishigaki et al. [2009] applied retargeting to games: user motion is blended in real time with example animations, depending on user intention, and environmental and physical constraints. These works rely on expensive optical motion capture.

For low-cost systems, Chai et al. [2005] tracked sparse 2D marker positions in stereo video, and achieved 3D pose reconstruction by constraining motion to a local linear model. Lee et al. [2002] transition a motion graph by performance according to silhouette features from video. Microsoft Kinect brought real-time control of virtual characters to casual users through retargeting skeletal motion [Vögele et al. 2012], facial expression transfer [Weise et al. 2011], skeleton-based deformations of objects [Chen et al. 2012], and mapping rigid object motions into virtual worlds [Held et al. 2012]. As these methods afford direct control, they do not apply to non-human topologies, and cannot control existing dissimilar animations — for retargeting, input and output motions are similar by construction.

Data driven. We can remove the similarity requirement with pre-authored character animations, and create indirect mappings between user and character by learning their dependency from examples. Most common are *pose mappings*, which map the tracked user pose, frame-by-frame, to a corresponding character pose. Pose mappings have been constructed through linear transfer of interpolations weights [Bregler et al. 2002], non-linear interpolation through a hierarchical mesh shape space [Baran et al. 2009], Gaussian process latent variable model (GPLVM) [Lawrence 2004; Yamane et al. 2010], and linear maps to rigged characters [Dontcheva et al. 2003; Seol et al. 2013] and mesh characters [Rhodin et al. 2014; Celikcan et al. 2014]. Example input motions are usually performed by the user, which is simple with automatic guidance [Rhodin et al. 2014]. Example character animations are artist created.

To improve control of arbitrary characters, Seol et al. [2013] classify user input poses in two: 'simple' motions, where character features map to human features directly and a linear map is applied, and 'complex' motions, where features does not (e.g., many pairs of legs) and so a nearest-neighbor lookup (NN) finds the closest animation frame from a pre-defined coupling. Both outputs are then blended. This setup generalizes via the linear map from the authored character animations as new live motions are performed. However, the NN map does not generalize, which leads to stilted animation when given new live motion variations, with no extrapolation to provide control over motion style variations.

Pose mapping techniques allow real-time control of characters of non-human shape and topology, as example-driven mappings decouple input and output motion style. However, the output animation quality and detail is limited by the ambiguity of pose mappings. Our goal is to overcome these ambiguities by generalizing *properties* of motions from sparse examples.

Action control. These methods trigger (non-human) character actions by detecting user intention. Actions are commonly detected by dynamic time warping and classification of predefined motions [Raptis et al. 2011] and by similarity of inferred dynamical movement primitives [Ijspeert et al. 2013]. Such high-level control enables *sympathetic interfaces*, e.g., animation through a sensor equipped plush doll [Johnson et al. 1999], and automatic maintenance of the emotional character state to stay *in character* [Tomlinson et al. 2002]. While our approach considers user intention, we focus on estimating, generalizing, and mapping properties of motion, which provides a much finer level of control.

Fourier representations. These capture motion properties in frequency bands and have been used to create animation variations [Pullen and Bregler 2000], to blend animations, and to alter motion

Feature	Gleicher [1998]	Shiratori et al. [2008]	Rhodin et al. [2014]	Seol et al. [2013]	Ishigaki et al. [2009]	Oore et al. [2002]	Ours
Easy to use AFP	○	●	○	○	○	○	●
Control of motion dynamics	○	●	○	○	●	○	●
Extensible motion graph / intentions	○	◑	○	◑	●	○	●
Control of fast and slow motions	●	●	○	○	●	●	●
Superposition control	●	○	●	●	●	●	●
Direct control of skeleton DOF	●	○	○	●	●	●	○
Interactive user-defined control motions	○	○	●	●	○	○	●
No body part or DOF assignment required	◑	○	●	○	◑	○	●
Independent of rig	○	●	●	○	○	○	●
Non-biped topology	○	◑	●	●	○	●	●
Robust to user size and shape	●	●	◑	○	●	●	●
No database or predefined controller required	●	○	○	○	○	●	○
Diverse and multiple tracker capable	○	○	●	◑	○	●	●
Robust to low quality input device	○	●	●	●	○	○	●
High dimensional tracking input	●	○	●	●	●	○	●
Real time	●	●	●	●	●	●	●
Low control delay	●	○	●	◑	●	●	◑
Non-bipedal complex motion transitions	○	○	○	○	○	●	●
Foot-sliding prevention (biped & quadruped)	◑	◑	○	○	◑	○	●
Physical realism	○	●	○	○	●	○	○
Live animation (game, performance)	○	●	●	●	●	○	●
Prototyping / blocking animation	●	●	◑	◑	◑	●	●
Professional 'final' animation	●	○	○	○	◑	●	○

Table 1: ● / ◑ / ○ *: full / partial / no support; Feature table comparison to state of the art real-time character control methods. For our task of versatile character control with application to games, our method is often a better fit than existing methods, though the system does require a pre-existing animation database.*

style [Unuma et al. 1995]. Akhter et al. [2012] propose a bilinear spatio-temporal basis that describes oscillations around a set of example shapes. Our work could be viewed as a particular form of Fourier decomposition of high-dimensional *input* motions.

An inspiring step in this direction is the method of Shiratori and Hodgins [2008], where amplitude, phase, and frequency of low-dimensional accelerometer sensors are mapped to a physically-simulated character. The method of Lockwood and Singh [2012] classifies *finger walking* motions from touchpad input by contact features, e.g., frequency, into gross motion classes.

We generalize this idea to handle high-dimensional input motions, continuous output (e.g., speed vs. classification into slow or fast), user-defined control motions with arbitrary periodic trajectories, and simultaneously-performed motions.

For output, we synthesize animations from a parametric motion graph [Rose et al. 1998; Heck and Gleicher 2007; Casas et al. 2012]. Table 1 relates the most relevant real-time character control methods.

3 Method overview

Our goal is to generalize wave properties of motions from sparse examples for real-time character control. Three steps are required: *authoring*, *control definition*, and *live control* (Fig. 2). Pseudocodes for our approach are included in a supplemental document.

First, in the authoring step (§4), example character animations are arranged to form *parametric motion classes* as nodes in a motion graph, as would be typical for a game. Second, in a control definition step (§5), a *reference control motion* is specified for each motion class, from which we learn a mapping per node (§6). These motions would typically be predefined by a game designer, but as

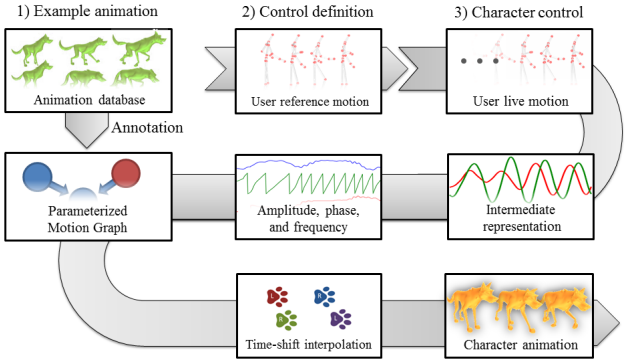

Figure 2: *Pipeline:* 1) *An animation database is created by an artist.* 2) *Control definition: the user interactively performs one reference motion for each parametric motion class.* 3) *Live character control: the virtual character is controlled by estimateing AFP parameters from independent intermediate representations, with animations synthesized by a new time-shift interpolation.*

this is computationally fast it also allows for interactive end-user definition. Third, the user performs *live motions* for real-time character control (§7). Simultaneously-performed motions are separated, and variations in motion AFP (amplitude, frequency, and phase) are extrapolated to generate new motion style variations. Animations are synthesized by database interpolation; for quadrupeds, we introduce a separate time and pose interpolation of individual limbs which improves animation quality and reduces foot-skating artifacts (§7.2).

4 Parametrized character representation

To create a semantic connection between user control motion variation and character animation, we arrange example animations into a vector space of parameters [Rose et al. 1998; Heck and Gleicher 2007]. We use AFP parameters to generate live animations by interpolating the estimated and annotated AFP parameters. For instance, the user varying their leg height in a mimicking walk could be transferred to a dog character's stride length by annotating the database animations of standing and walking with amplitude 0 and 1, respectively. In a different application, amplitude could map to step height, for instance, for a horse dressage game character. The frequency parameter could trigger a transition from walk to run by the user speeding up the control motion, or could parametrize a shaking animation with varying centrifugal force. Depending on the tracker, these user motions can range from single finger movements, to facial expressions, up to full body motions. Additional annotation dimensions such as emotion (e.g., *happy*, *sad*) and heading rate (e.g., *left*, *right*) are also possible depending on the tracker.

Formally, an animation Y of frame length T is a sequence of individual meshes $\mathbf{y}_t$, so $Y = [\mathbf{y}_1, \cdots, \mathbf{y}_T]$. Each $\mathbf{y}_t$ is a list of mesh vertex positions, i.e., no rig is required and any animation creation system can be used. T is an animation-specific variable, as different Y have different lengths. To build a vector space, each Y in a database $\mathcal{Y}$ is assigned parameters $\theta_Y = (a, f)$ to represent variations in frequency f (e.g., *fast, slow*) and amplitude a (e.g., *large, small*). As we focus on cyclic animations, the time index that specifies the current frame t of the example animation is parametrized in the cyclical domain $[0, 2\pi)$, invariant to the animation length T, by phase $\varphi := 2\pi t/T$. Y then exists in a two-dimensional vector space, with one dimension per parameter. These annotated database animations form parametric motion classes, which we combine into a motion graph (Fig. 3).

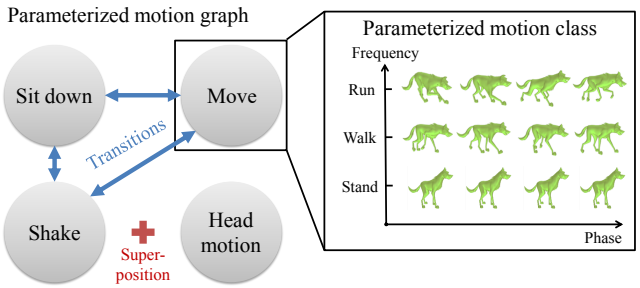

Figure 3: *Parametrized motion graph for the dog character. Each node represents a parametrized motion class that synthesizes character animation from gesture AFP parameters. Edges mark transitions which are initiated by gesture activation. We also support superposition of secondary actions such as head motion which are additive.*

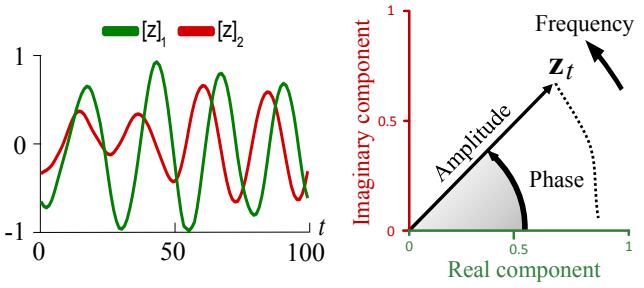

Figure 4: Left: *Our intermediate representation as time-varying signal.* Right: *A polar plot with the intermediate representation as phase and amplitude. The frequency of sequence Z, and the current instantaneous sample $\mathbf{z}_t$, are represented by the dotted line.*

5 Reference control motion definition

Each parametric motion class requires one reference control motion X as a sequence $X = [\mathbf{x}_1, \cdots, \mathbf{x}_T]$ of poses as point positions. This could come from any tracking system. Each motion is defined by performance at an arbitrary steady speed for one period at maximum amplitude, which provides a kind of 'physical normalization' of amplitude to $[0, 1]$. For instance, one cycle of a mimicking walk, where the legs are raised as high as possible. Practically, the start frame $\mathbf{x}_1$ is marked by pressing a remote control. The end frame is automatically detected for cyclic motions by finding the pose most similar to the start pose which is at least $2/3$ periods away from the start. For non-cyclic motion classes, i.e., a dog sitting down, we require rest and extreme poses to be marked with a remote press, e.g., lowering the arm from horizontal to vertical position.

Although we require only a single control motion example, we are able to generalize or extrapolate to variations in the live animation step by analyzing the input motion for differences in AFP. Further, no manual mapping (e.g., [Shiratori and Hodgins 2008]) or degree-of-freedom tagging (e.g., [Seol et al. 2013, §4.1]) is needed as it is automatically inferred by the regression method below, which makes it possible to define reference control motions in seconds.

6 User-to-character motion mapping

The goal of the live step is to map the stream V of user poses $\mathbf{x}_t$ to the parametrized character representation according to the defined set of reference control motions $\mathcal{X}$. The live step accomplishes two tasks: separation (disambiguation) of simultaneously performed control motions for each reference motion X (§6.1), and estimation of AFP motion parameters θ_X for character animation (§6.2).

6.1 Separation of simultaneous gestures

Simultaneous input motions primarily occur in two situations: 1) For superposition effects, e.g., dog shaking while walking, with two or more simultaneous control motions; 2) At transitions between graph nodes like walking and jumping, future control motions may be started while current motions are gradually stopped, which we refer to as *intersecting* motions. Direct estimation of input AFP leads to interference between motions (Fig. 11).

Instead, we separate high-dimensional input V into independent intermediate representations $Z_X = [\mathbf{z}_1, ..., \mathbf{z}_T]$ for each reference control motion X by linear pose mappings $\Phi_X : \mathbf{x} \rightarrow z$. Inspired by previous work in low dimensional circular embeddings [Lee and Elgammal 2004] and frequency band decompositions [Akhter et al. 2010], we design Z_X as a complex sine wave (Fig. 4):

$$\mathbf{z}_t = a_t \begin{pmatrix} \cos(\varphi_t) \\ \sin(\varphi_t) \end{pmatrix}. \quad (1)$$

Z_X forms a curve in polar coordinates which evolves counter-clockwise as t increases. It is a low dimensional abstraction of input pose $\mathbf{x}_t$ which encodes phase as angle φ_t, amplitude as pointer magnitude a_t, and frequency f_t as change of phase over time.

Mappings Φ_X are learned by pairing X to one period of the complex sine wave $Z_X = [\mathbf{z}_1, \cdots, \mathbf{z}_T]$, with $\varphi_t = \varphi_{t-1} + 2\pi/T$, where T is set by $\mathbf{x}_1$ and $\mathbf{x}_T$. This matches the performance of one period of motion at maximum amplitude during control definition, $a_t = 1$, which sets the range of available amplitudes in the live motion control to $[0, 1]$. The initial phase φ_1 is set to the frame that is farthest from the mean.

To separate simultaneous control motions, we enforce zero output for the remaining control motions by using them as negative examples and setting $\mathbf{z}_t = 0$. This forces Φ to depend on properties of the input that are unique to the reference. For instance, the two motions of bending a single finger and of bending all fingers at once can be distinguished and mapped to different motion classes without any labeling of body parts (Fig. 11).

Each map Φ_X is inferred by linear Gaussian process regression [Rasmussen and Williams 2006] and is parametrized by a matrix M that is given as the mode of the posterior distribution:

$$p(M|X, Z_X) \propto \exp\left(-||MX - Z_X||_F^2 - \sigma_n||M||_F^2\right), \quad (2)$$

where $||.||_F$ is the Frobenius norm and σ_n is a regularization parameter. The linearity of Φ_X offers good extrapolation from the training sequence in that an amplified input motion leads to a proportionally larger amplitude of Z_X, and an increase in motion speed leads to an increase of the frequency. This makes these parameters intuitively controllable. While, in general, more flexible non-linear maps can be adopted if necessary, the linear map proved to be sufficient in our experiments and was preferred over non-linear alternatives, as it is fast and it does not require tuning of additional hyperparameters.

Non-cyclic linear motions. These have no notion of frequency, amplitude, and phase. Instead, we form a separate one-dimensional space z that represents the progression by $\varphi = 2\pi z, z \in [0, 1]$. During learning, the reference intermediate representation is defined as $z_t = t/T$ to map progression in the control motion linearly to φ.

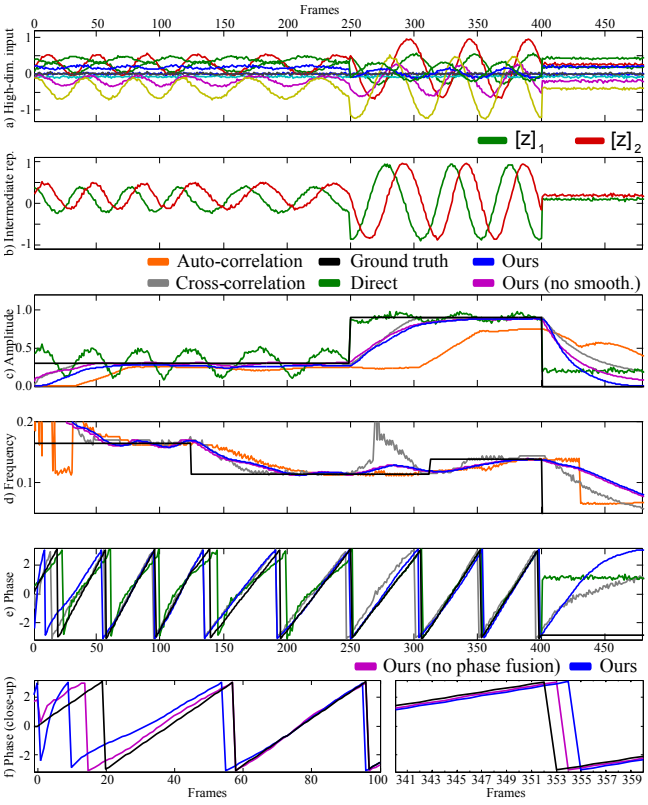

Figure 5: *AFP parameter estimation on a synthetic sequence with ground truth:* a) *User input motion, displayed as trajectories over time,* b) *mapped onto the intermediate wave representation,* c–e) *Resulting amplitude, frequency, and phase estimates. Shiratori and Hodgins [2008] use cross-correlation (gray) for phase, which is generally noisier, and auto-correlation (orange) for frequency, which has delay of one period. Also direct estimates (green) of amplitude by $\|z\|$ and phase by atan2$([z]_2, [z]_1)$ are erroneous (first half of the motion).* f) *Our phase fusion compensates phase discontinuities (first* 15 *frames) and preserves the signal otherwise (frames* > 60*).*

6.2 Live estimate of motion properties

For notation ease, we explain how AFP motion properties are estimated for a single reference motion, and so we drop subscript X. We apply Gabor filtering, a variant of windowed Fourier analysis which has optimal time-frequency resolution [Feichtinger and Strohmer 1998]. Gabor functions are sinusoids modulated by Gaussians $N(x;\mu,\sigma)$, where x is time, and μ and σ are the Gaussian center and standard deviation, respectively:

$$g(x;\mu,f) \Leftarrow \begin{pmatrix}\cos(2\pi f x)\\ \sin(2\pi f x)\end{pmatrix} N(x;\mu,\lambda/f). \tag{3}$$

We find the Gabor function that best fits (maximum response), and we adopt its phase, amplitude, and frequency as the instantaneous estimates $\hat{a}_t$, $\hat{f}_t$, and $\hat{\varphi}_t$. The Gabor response is the complex inner product $r_f = \langle [g(t-\tau;\mu,f),\ldots,g(t;\mu,f)], [z_{t-\tau},\ldots,z_t]\rangle$. We filter a history of $\tau = 150$ frames (5 seconds) of Z with a series of 50 Gabor functions with wavelengths $1/f \in [5, 150]$ and mean μ fixed to the most recent frame t.

Cross-correlation is one natural alternative (e.g., [Shiratori and Hodgins 2008]); however, we chose Gabor filtering as phase has an analytic form $\hat{\varphi}_t = \text{atan2}([r_f]_2, [r_f]_1)$ which led to higher accuracy in our experiments (Fig. 5; §8) vs. the required discrete phase sampling for cross-correlation.

Previous methods smooth the input signal temporally to overcome tracking noise and errors from user imprecision, which is essential for estimating velocity and acceleration. However, deciding smoothing window width is difficult: a large window strongly damps estimates, but small windows preserve high frequency noise. In our case, the Gabor window size adapts to the input motion speed: For slow motions, the window is large and high frequency noise is effectively ignored; for fast motions, a small window preserves rapid changes. The robustness-response trade-off is exposed by λ. We choose $\lambda = 2/3$, which smooths the response over most of one period (Fig. 5, c & d). This is not a hard delay: response is immediate but small, increasing in magnitude over time.

Noise detection. In preliminary experiments, the estimated amplitude was undesirably high for low signal-to-noise ratios, leading to unintended character actions. To reduce the amplitude in these cases, we exploit that noise corrupts the sinusoidal form of Z. Intuitively, given the best fit Gabor function, if Z is still not close to this perfect sinusoid shape, then the input is likely to be dominated by noise.

A good measure for how close Z is to a perfect sinusoid is the quotient of maximum Gabor response, r_f, and the total energy, n_f, apparent over the corresponding Gaussian window, with energy $n_f = \langle [N(t-\tau;\mu,\lambda/f),\ldots,N(t;\mu,\lambda/f)], [|z_{t-\tau}|,\ldots,|z_t|]\rangle$. If Z is not sinusoidal, i.e., $r_f/n_f < 5/6$, then the estimated amplitude, $\hat{a}$, is linearly damped to $a = \hat{a}2(s-1/3)$. We show in our experiments (see video) that this scaling effectively reduces the amplitude if the signal cannot be uniquely assigned to the control motion.

Discontinuity compensation. As future input is unknown in our live setting, the Gabor function (and its Gaussian window) is one sided: it is not smooth as it has a sharp edge. Hence, strong noise from partial tracking failure is possible, as are multiple local maxima within the filter response due to fast input motion frequency switches. These can lead to strong discontinuities in the estimated phase and frequency. To compensate for drastic changes, discovered frequency $\hat{f}_t$ and amplitude $\hat{a}_t$ parameters are smoothed over time to f_t and a_t by a small Gaussian of $\sigma = 200$ms, respectively. This is different from smoothing the input poses as high frequencies are preserved, and is closer in spirit to ease-in and ease-out effects.

To stabilize phase, we exploit the temporal dependency of frequency and phase. The instantaneous phase estimate $\hat{\varphi}_t$ is fused with its previous estimate φ_{t-1} and phase speed $2\pi f_t$:

$$\varphi_t = \frac{\hat{\varphi}_t + \gamma(2\pi f_t + \varphi_{t-1})}{1+\gamma}, \tag{4}$$

where $\gamma = 10$ was empirically set to balance integrated and estimated phase, and computation is in the circular domain $\varphi_t \mod 2\pi$. One could integrate φ_t directly from f_t and φ_{t-1} (i.e., $\gamma = \infty$), but this decouples the phase of the user control motion from the character animation, and so leads to less control. The fusion effect is visualized in Fig. 5 e–f.

Under-constrained input. For very simple input motions which show variation in a single dimension, such as raising and lowering an arm periodically, the mapping to the two-dimensional intermediate representation is under-constrained. We adopt a heuristic on the uncertainty of the prediction: The predictive variance ϑ_t corresponding to the input $\mathbf{x}_t$ is estimated as the average pair-wise distance of the $n = 10$ reference intermediate representation frames that were assigned to the n nearest neighbors of $\mathbf{x}_t$ in the control definition step. This is a good estimator when the given input $\mathbf{x}_t$ is close to the training data [Kwon et al. 2015], as in our reference/live setup.

The estimated variance reduces the influence of the potential unreliable component: we average variances over the Gabor filter window and weight the influence of the real and imaginary component of Z by their reciprocal, respectively. For complex motions both dimensions have full weight, as their variances are similarly low, which increases robustness to noise.

7 Live character animation

Given a user motion stream $V = [\ldots, \mathbf{x}_{t-1}, \mathbf{x}_t]$ observed until t, parameters $\theta_X := (a, f, \varphi)$ are estimated for all control motions $X \in \mathcal{X}$ as described in Section 6.2. Here, we explain how θ_X is used to initiate transitions in the motion graph (§7.1) and to synthesize the actual animation of the character (§7.2).

7.1 Motion graph and motion transitions

We connect multiple parametrized classes into a motion graph, with edges as transitions between classes (Fig. 3, §7.1). By graph construction, nodes with an edge distance greater than two are independent, which increases gesture scalability. We trigger transitions along an edge by varying the activation of simultaneously-performed control motions (cf. Ishigaki et al. [2009] with sequential but not simultaneous distinction). As simultaneous gestures are made independent by mapping to separate intermediate representations, we simply use the estimated amplitude a of θ_X to activate gesture X.

Transitions are initiated by increasing the amplitude α of the target node beyond 0.2 (maximum is 1). A transition is successful if the amplitude of the source node control motion is reduced to below 0.2. We abort the transition if the target activation sinks below 0.1. During the transition, we blend linearly over a fixed time window of half a second between the source and target. For non-cyclical motions, the progression parameter φ of θ_X is used instead of a as it measures the distance to the rest state. If desired, unrealistic transitions such as stopping during a jump could be prevented by restricting transitions to specific points [Kovar et al. 2002] or windows [Heck and Gleicher 2007].

7.2 Time-shift animation interpolation

Transitions and superpositions of multiple motion classes are synthesized by motion blending. To synthesize animations within a single motion class X with parameters equal to the most recent estimates θ_X, we build upon the radial basis function method of Rose et al. [1998] and interpolate the nearest database animations with interpolation weights w_Y for each animation sequence Y set inversely proportional to the parameter distance $\|\theta_X - \theta_Y\|_W$, where W normalizes each dimension to $[0, 1]$. The timing of the animation, i.e., the time index into $Y = [\mathbf{y}_1, \ldots, \mathbf{y}_T]$, is given by the inferred phase φ of θ_X.

We use linear derivative time warping [Keogh and Pazzani 2001] to temporally align database animations during authoring. Even after temporal alignment, the naive interpolation of different quadruped locomotion states can cause strong artifacts, such as a leg stuck halfway during a transition between walks and runs. This is because the leg actually switches phase: it moves in the opposite direction in the run than in the walk (Fig. 6). This cannot be solved by improving global alignment methods, nor by using different character representation such as skeletons, because the problem is inherent to the motion. While rarer, this issue is still possible in bipeds if arms and legs move synchronously and then asynchronously across motions.

This problem is related to the timing and interpolation of upper and lower body motions [Ashraf and Wong 2000; Ha and Han 2008], and to *asynchronous time warping* for horse gait transitions, where

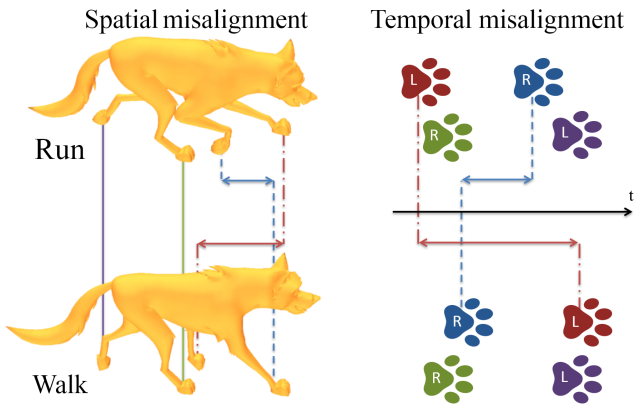

Figure 6: *The quadruped leg interpolation problem. Interpolating between walk and run animations is difficult for many quadrupeds.* Left: *While the back legs align well spatially between the walk and run frames, the front legs switch phase.* Right: *Global linear time-warping cannot fix this temporal alignment problem as the order of foot placements switches (here, on the left-hand side).*

legs are blended separately and timing differences are compensated gradually [Huang et al. 2013; Sung 2013]. However, these methods only work for transitions between two motions of boned characters, and not our more general mesh character case where transition length is unknown, and where we may interpolate between more than two animations (our most sophisticated example blends seven animations in a 4D space). We also experimented with dynamic time-warping of segments instead of linear time-warping. However, this led to unrealistic changes in the dynamics of the motion.

We solve the problem with separate timelines for individual limbs that can shift for alignment, and by separating time and shape (mesh) interpolation. During database construction, the character mesh is segmented into torso and limb segments. The longest animation sequence is selected as reference and all other sequences are aligned by linear time warping, individually for each limb segment. This provides a phase offset $\nabla\varphi_{Y,i}$ for each motion Y and limb index i. During runtime, the time offsets (phase offsets) are interpolated separately for each i according to weights w_Y in the circular domain:

$$\varphi_i = \varphi + \sum_{Y \in \mathcal{Y}} w_Y \nabla\varphi_{Y,i}. \tag{5}$$

As each example motion Y has the same weight w_Y across all segments, the temporal dependencies between limbs are maintained implicitly if possible and are interpolated if they contradict (e.g., the order of foot plants). Only after this temporal interpolation is the mesh interpolation performed at phases φ_i, with a differential coordinate reconstruction method which prevents seams between segments and includes global motion. Additional details on the time and mesh interpolation are explained in the supplemental document.

7.3 Extensions

Secondary animation. Secondary control motions are superimposed onto the primary character animation via mesh deformations. The deformation is computed as the difference between the mesh animation controlled by the secondary control motion and the character rest pose. One example of this is lifting the torso of the dinosaur by raising the user's head, superimposed onto a simultaneously-controlled walk, or bending of the caterpillar body during turns caused by the user physically turning. Superimposing AFP mappings is also possible, e.g., shaking the dog's torso while walking.

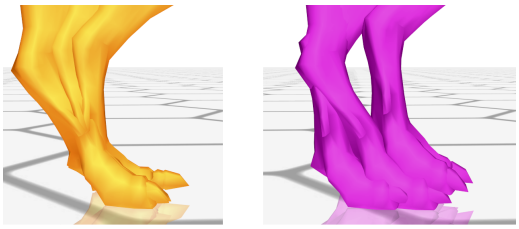

Figure 7: *Overlay of three frames, spaced 10 frames apart, of the dinosaur's paw during a live-controlled walk.* Left: *The application of the foot-plant constraint effectively prevents foot-sliding.* Right: *In the unconstrained case, the foot slides back and forth.*

Table 2: *Evaluation character databases, numerating the motion classes with characteristics and number of example animations. Parameter annotations are listed in the supplemental document.*

Character	Dog	Caterpillar
Features	Transition, superposition, complex motion	Transition
Params.	Emotions, amplitude, frequency	Amplitude
Motion classes	Walk, sit, shake, look, wave, scratch, jaw	Walk, crawl, look, jump
# Animations	7 + 1 + 2 + 1 + 1 + 1 + 1	4 + 1 + 1 + 1
# DB frames	130	626

Character	Horse	Humanoid	Dinosaur
Features	Rich class control	Smooth locomotion	Transition, superposition
Params.	Emotions, frequency	Amplitude, frequency	Amplitude
Motion classes	Walk	Walk	Walk, jump, bend, stretch
# Animations	7	6	4
# DB frames	160	165	91

Foot-sliding cleanup. At coarsely sampled regions of the parameter space, the mesh interpolation can deviate from the global character motion, leading to foot sliding or no foot-ground contact. To correct this, we label database animation frames that show ground contact; then, during synthesis, we use the approach of Lee et al. [2010] and perform a weighted vote (contact=1, no contact=0) between all database motions with weights w_Y and per foot time indices from the time-shift interpolation (§7.2). The activation threshold is set to 0.9 in our experiments. During reconstruction, the constrained feet are pinned to the ground by additional vertex position constraints in our differential coordinate solver (see supplemental). This procedure is simple compared to more complex methods like MeshIK [Sumner et al. 2005], and effectively prevents sliding (Fig. 7). Importantly, the foot constraint interpolation benefits significantly from our time shift solution to the quadruped leg interpolation problem.

Emotion and direction control. We estimate user emotion with a face tracker, which is an additional parameter dimension e for character control. Further, the body orientation of the user with respect to the input device is transferred to control the character rate of turn β. This second additional parameter dimension improves the quality of animations such as the long body of a caterpillar turning.

8 Experiments

Our results are best observed in our supplemental video, with a second video providing additional comparisons to existing techniques. We test our method on 5 characters (Table 2): dog, caterpillar, horse, human, and the dinosaur of Seol et al. [2013]. Please see the supplemental document for more details on the experimental setup. The experiments were performed on a Xeon CPU E5-1620-3.6GHz at 30 FPS for models of 10k faces.

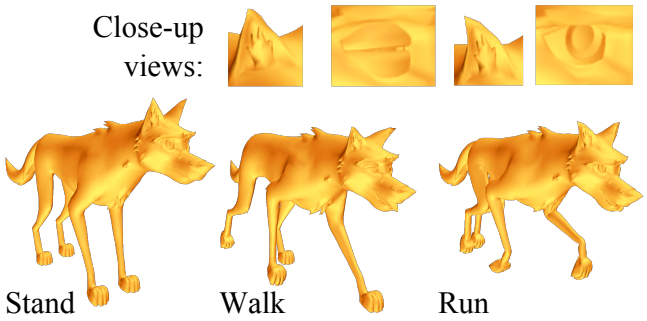

Figure 8: *Frames from a dog animation generated by mimicking a walk. The close-up views highlight that temporally and spatially localized details and dynamics of the original animation are preserved by our system, such as an eye blink and flapping of the ear.*

8.1 Character animation quality

Versatile input devices. Characters are animated with different user body, hand, and face control motions, captured by non-intrusive sensors that are suitable for VR applications (Fig. 1). The body is tracked as 20 3D joint positions by Microsoft's Kinect and the hand as 9 3D fingertip and palm positions by Leap Motion. We classify facial expressions by Intraface [Xiong and De la Torre 2013] as a continuous value e between sad -1, neutral 0, and happy $+1$.

Intuitive control. Our mapping is able to generalize and distinguish between various user-defined control motions. Our users preferred human mimicking motions (swinging arms synchronously and asynchronously for walking and jumping, Fig. 10) and abstract mimicking of character style (arm undulation for crawling, Fig. 10).

Generalizing control. Figure 8 shows control of the dog locomotion class (Fig. 3). From a single user-defined reference control motion of an on-the-spot run, and from three character animations, we can generalize to variations in frequency and amplitude affecting character stride length and step speed, and dynamics style changes from stand, to walk, and to run. Details such as eye blinks and speed-dependent ear wiggles are preserved from the artist animation. The parallel coordinates graph in Figure 9 also shows that our mapping is able to reach the majority of a four-dimensional parameter space, including character emotion.

Robust instantaneous estimates. For the same dog sequence, the importance of each filtering component is shown in our video. In addition, the quality of estimates is compared to ground truth on a synthetic arm waving input motion with additive white Gaussian noise (SNR = 93.8). Our method quickly estimates changes in control speed and amplitude, gracefully smooths over discontinuities, and is more accurate compared to baseline methods (Fig. 5).

One key advantage of our method is that it is reliable for a very large range of input motion speeds. We show this with the human character, which has database animations of slow, medium and fast walks, as well as medium and fast runs. Our method smoothly cycles through slow walks of one quarter steps per second, up to a fast run of two steps per second.

Parametrized time-shift interpolation. With our time-shift approach, the artifacts with interpolating out-of-phase limb motions are significantly reduced, allowing effective foot-sliding prevention for bipeds and quadrupeds when varying between motion styles such as walk and run and different step sizes (Fig. 7). This is visible in all our quadruped locomotion examples and in a specific side-by-side comparison in the video.

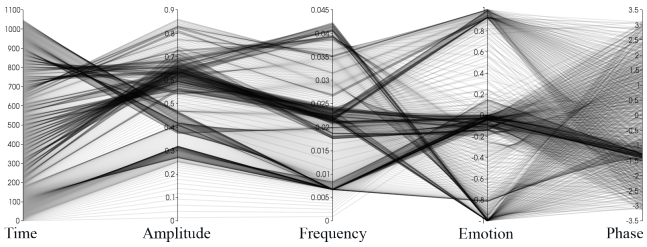

Figure 9: *Parallel coordinate plot of the parameter space covered during the horse animation, where each line from left to right corresponds to one parameter configuration. Amplitude ranges from 0 almost to 1; frequency shows clusters around walk (0.025) and gallop (0.04) with transitions; emotion covers the full range from -1 (sad) to +1 (happy). Phase cycles between $-\pi$ and π. This demonstrates that our control scheme reaches wide parameter spaces.*

Motion graph and control superposition. Interpolating within a parametric motion class (stand to walk to run, Fig. 1) is caused by changing motion speed and amplitude. Transitions across classes (crawl to jump, Fig. 10) are caused by change of control motion. Secondary actions such as head motion and body bending are superimposed. The final dog example in our video combines everything: hand and face trackers for both cyclic and non-cyclic motions with wave-based and linear control. We extrapolate control within a shared locomotion and emotion space, plus linear control over head rotation, pawing at the dirt, shaking, mouth motion, begging, and sitting. This control example shows many of our advantages, most notably the independence of control motions, e.g., bending of the first finger is used in three separate control motions without interference from combinations with other fingers.

8.2 Comparison to related work

Table 1 compares the most related character control methods.

AFP estimation (Shiratori et al. [2008]). A natural baseline for AFP estimation is auto- and cross-correlation as proposed by Shiratori et al. [2008]. In our experiments, normalized cross-correlation between the reference control motion and input motion performs similarly in terms of delay and estimated values to Gabor filtering on our intermediate representation for either an independent control motion or for simultaneously-performed control motions which are spatially separated (e.g., left and right arm motion, Fig. 5c–e). However, frequency and phase estimates were less reliable as the signal needs to be convolved for a discrete set of phase-frequency combinations. In contrast, the Gabor filter gives phase analytically and only requires to sample the frequency dimension. Shiratori et al. [2008] use auto-correlation to measure the periodicity of a signal. It requires two motion periods for comparison, hence, introduces a lag of one period compared to cross-correlation and our approach. Moreover, it was less reliable in our experiments (Fig. 5c-d). Overall, cross-correlation is an alternative to Gabor filtering, but was discarded due to its drawbacks.

Our main contribution to the AFP estimation problem is the separation of simultaneously-performed motions into independent intermediate representations. We demonstrate its importance with the hand-controlled dog, where talking, shaking and scratching the paw are controlled by rotating the thumb and shaking the whole hand. Direct frequency decomposition methods (e.g., [Unuma et al. 1995]) do not consider prior knowledge of a particular reference control motion, and would lead to permanent undesired control and superposition. Directly applying cross-correlation to the input motion (e.g., [Shiratori and Hodgins 2008]) activates all character motions simultaneously when only one of the corresponding input motions

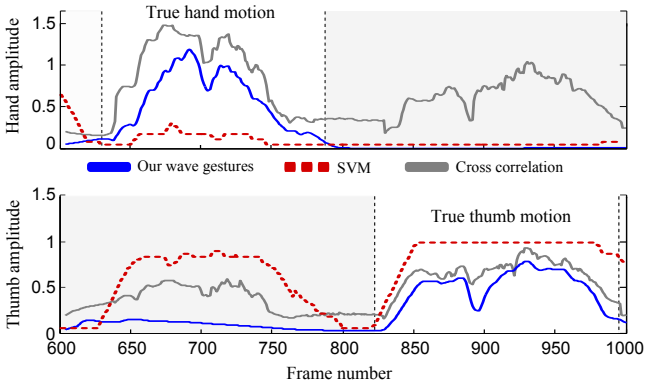

Figure 11: *Analysis of control independence. Our Gabor filtering of independent wave representations separates rotation of the whole hand (*top*) from thumb rotation (*bottom*), with high amplitude in true motion areas and low otherwise. Cross-correlation (e.g., [Shiratori and Hodgins 2008]) suffers from strong interference with high amplitude in all areas of motion. SVM classification (averaged over 30 frames as per [Seol et al. 2013]) falsely detects no hand motion, and also detects thumb motion during hand motion.*

is performed, as all involve motion of the thumb (Fig. 11). This is effectively prevented by our separation method, as only very subtle interference is visible. We shown this in the supplemental video for simultaneous and intersecting control motions.

Gesture activation classification (Seol et al. [2013]). We compare our method to SVM classification on the task of detecting the active control gesture. We train a Gaussian-kernel SVM on position, velocity, and acceleration features and average the activation of each gesture over 30 frames, as proposed by Seol et al. [2013]. For control motions which are spatially separated (e.g., left and right arm control of caterpillar), SVM and our method are equally robust, with a slightly lower delay for SVM. However, for hand input where the same finger is used during multiple gestures, SVM falsely detects thumb motion instead of hand rotation, and also fails to detect the hand rotation at all, because instantaneous velocity and acceleration features do not distinguish the performed quick hand motions reliably (Fig. 11). In contrast, our method separates control motions correctly in this challenging case with very little interference.

Pose mappings. To see the effect of our motion mapping in contrast to existing pose mappings, we compare against a shared GPLVM with 15 latent dimensions (similar to Yamane et al. [2010]), a latent volume nearest neighbor (NN) pose mapping with 15 neighbors (similar to the mapping of Seol et al. [2013] for cyclic motions), and a linear pose mapping (similar to Rhodin et al. [2014] and Seol et al. [2013]). We train all systems with an arm swing control motion of 32 frames, corresponded to horse walk and horse stand animations.

Our method is able to reproduce all details of the horse animations and extrapolates to smaller step lengths by reducing the control arm swing extent. No other compared method is able to generalize this change in control: The shared GPLVM method shows very strong jitter. The NN lookup cannot reproduce the temporal evolution of the original animation exactly due to input noise and user imprecision, and also cannot generalize the transition from walk to stand. The linear map suffers from less jitter, but the animation detail is reduced. Moreover, at high control motion speed, it exhibits unnatural exaggerated rotations in the hooves, as the mapping was learned from a single example motion at fixed velocity and the used velocity features do not generalize to significantly different input speeds. None of these artifacts occur in our generated animation.

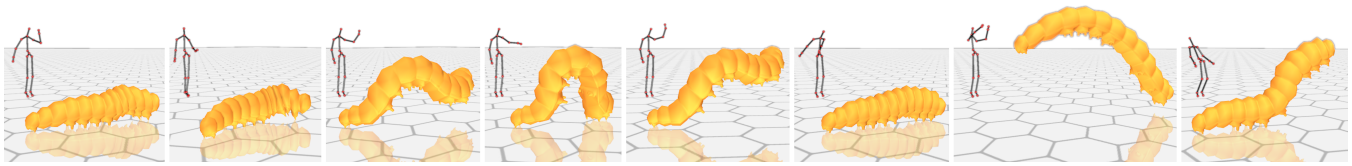

Figure 10: *Animation of the caterpillar character, controlled by the user* (black skeleton)*:* Column 1: *Crawl by swinging the arms asynchronously.* Column 2: *Bending by turning user body.* Column 3-5: *Separate walk styles with large style variations by waving the left arm at different amplitudes.* Column 6-7: *Jump by swinging arms synchronously.* Column 8: *Rising the head by bending.*

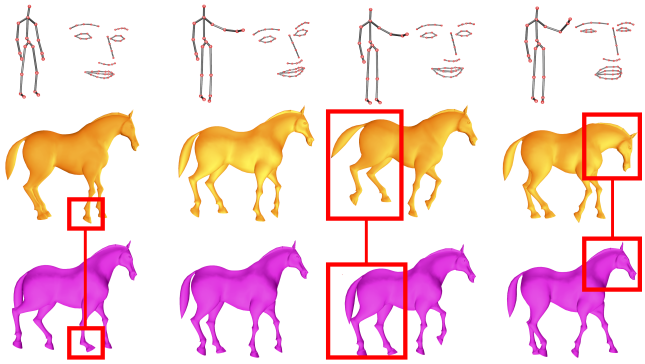

Figure 12: *Comparison with [Rhodin et al. 2014].* Top: *User body and face pose, used for both techniques.* Middle: *Our animation using the parametrized horse motion class; from left to right: stand, trot, happy gallop, and sad gallop.* Bottom: *Rhodin et al. shows distortions in the stand and is not able to generalize to a gallop when speeding up (column 3) nor to different moods (column 4).*

Pose mappings — Seol et al. [2013]. We directly compare to the dinosaur animation presented by Seol et al. [2013]. Cyclical motions are classified and mapped by NN maps, which are limited as they do not generalize to motions of varying amplitudes. Our method improves the control of cyclic motions as it is able to recover very slow and very fast motions, with independent control of dinosaur step length and step speed, all generalized from a single user motion training example. Moreover, we provide more controls (jumping and tail wiggle) within the same control sequence due to independence of controls. Finally, our control is more robust, and so the synthesized cyclic dinosaur walk contains less temporal jitter and no foot sliding.

Pose mappings — Rhodin et al. [2014]. Compared on horse locomotion (Fig. 12), our method provides improved control over motion style and emotion by mapping to a parametrized locomotion class of stand, trot, and gallop animations in happy, neutral, and sad emotion variations (seven animation examples). Neither the richness of control nor the extrapolation from a single example control motion is possible with the linear mapping used by Rhodin et al. [2014].

8.3 User evaluation

We studied wave gestures with 10 participants. For fair comparison, we predefined control motions for all participants, as per a typical game setting. In a pilot study, we rejected NN mapping (similar to Seol et al. [2013]) as this did not generalize well to different user body proportions nor to variations in user-specific control motion characteristics. Direct pose mapping (similar to Rhodin et al. [2014]) worked well for control; but output animations look stilted, with foot skating, floating, and temporal jitter. Specifically, very slow or fast control motions that differ strongly from the reference control motion lead to unpleasant artifacts (see video, model comparison section). Thus, we chose the familiar gamepad as a baseline. Amplitude and frequency map to left/right analog triggers, heading to left thumb stick, and motion transitions to face buttons. Progression is obtained by integrating frequency.

We tested three game-like tasks: 1) Follow a curved path, to test world locomotion control; 2) Transition between motion classes at specific world points, to tests action or event response; 3) Control horse step length and frequency to match a reference, to test precise control (see video, 03:38). The path has width $4\times$ character width and 10 turns with average maximum curvature $0.26\times$ path width. After training of one minute for each task, all subjects were able to solve all tasks with both methods. Figure 13 shows box plots with outlier rejection computed using ROUT at $Q = 1\%$, P values computed using paired Student's t-test analysis, and significance shown at the 95% confidence level.

In summary, wave gestures are a competent alternative to a gamepad for our tasks. With the same control motions, all 10 participants were able to complete all tasks. This shows robustness to different body shapes and input motion variations, including to children (03:38–04:03, bottom left). In detail, for task 1, our method is slightly worse than the gamepad, straying from the path 0.6 times on average per participant and taking on average 4 seconds to get back on track, though this is somewhat expected as directional control is simple with the familiar gamepad. That said, users found it easier to adapt character speed to narrow or wide curves with wave gestures. Task 2 was comparable with both methods, but gamepad character animation looks less convincing with many abrupt changes in direction, amplitude, and frequency (video, 03:32–03:47, bottom right). For task 3, wave gestures were significantly more accurate (p-value = 3.95×10^{-2}), with frequency also showing improvements (p-value = 5.18×10^{-2}). We believe this is because frequency control is intuitive if performed with cyclic motions, as is change in stride length through amplitude control. We strengthened this belief in a post-task questionnaire, with wave gestures rated significantly more intuitive for stride length and step frequency control.

Users identified two significant limitations which apply broadly to gesture control: greater control delay vs. gamepad, and higher physical demand. Should a motion be uncomfortable, one benefit of our approach is that different control motions can be interactively defined in just a few seconds, with no required limb or part association. Our approach also allows different motion trackers to be swapped in easily as, after character authoring, the only input we require is 2D or 3D skeleton points. For example, we track the hand (see video), which is more suitable for longer control sessions or desk work.

One aspect that is untested in this study is the superposition of multiple motions, which is hard to map to a gamepad but is easily solved with wave gestures. The lack of more negative significant differences may be surprising given the familiarity of gamepads, though it is clear that both schemes have strengths and weaknesses.

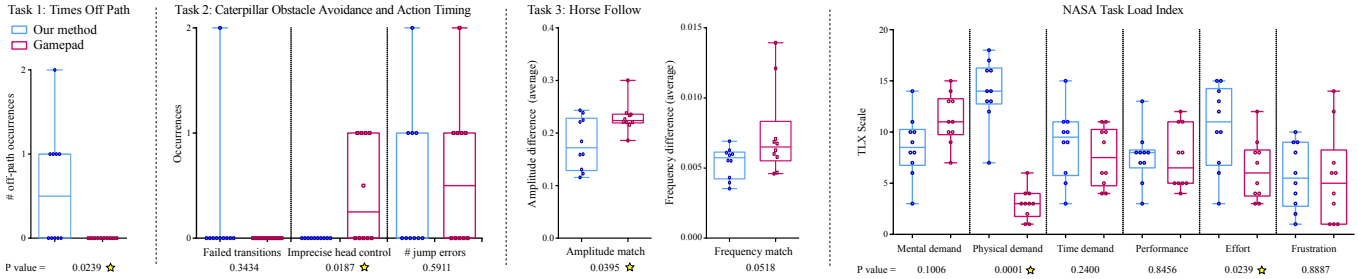

Figure 13: *Box and whisker plots for our task and questionnaire evaluation of our approach against a familiar gamepad controller. Two-tailed P values are provided from paired Student's t-test analysis, which significant at the 95% confidence level ($P < 0.05$) marked with a star.*

8.4 Expert animation practitioners

We invited three professionals to asses our system: a live animator, an offline animator, and a game animation middleware developer.

Live animator. Our trained performance animator was very enthusiastic about our system. They imagined a large potential for character control on stage through the mixing of pre-authored animations. However, the stage has demanding standards: while our approach is robust, slight delays when transitioning can cause inexperienced users to repeat actions, and the user must learn to trust this behavior in performance. She requested additional database animations, such as foot scratching and jumping; our system scales well to these additional animations by the motion graph.

Offline animator. Our offline animator saw the largest potential for our system at the prototyping stages of the animation pipeline, where quick and easy generation of animations would help communicate with the director. Further, they saw potential during content creation as a blocking tool for creating an initial animation from a storyboard, which is then later refined offline in the standard animation pipeline.

Game animation middleware developer. Our developer suggested that our approach fits many game requirements, notably the user flexibility, speed, and robustness. Our authoring pipeline is very similar, with games typically using animation blend trees (synonymous with parametric motion classes) and state machines (a simplified motion graph). Beyond real-time control, he saw a benefit for games companies who buy off-the-shelf motion databases, where our technique would allow animators to create new sequences in the style of the original database but with expanded variety, particularly for quadrupeds where existing data is rarer.

9 Limitations and discussion

Any system is a point in a design space with trade-offs. For real-time character control system, these are typically expressiveness, learnability, flexibility, and robustness. Our choices focus on improving the last three of these attributes. However, we limit expressiveness, in contrast to retargeted 'one-to-one' mapping approaches, with a need for authored character animations. Having said that, our approach is a good fit for games and virtual worlds. One way to overcome the general expressiveness problem is to allow a gesture to switch into a retargeted animation mode as a node in the motion graph, e.g., when appropriate for direct interaction with an object. For environmental context triggering, our activation variables work similarly to Ishigaki et al. [2009], so moving in the world could trigger motion control changes.

One might suspect that our 2D intermediate representation is too drastic a reduction in dimensionality. However, the intrinsic dimensionality of individual motions is low, and a drastic reduction is actually desirable: 1) To combat 'noise', as user shape and coordination skills vary greatly (e.g., children), with tracking inaccuracy also affecting the result; 2) Each parametric motion class has one reference control motion, and control through variations of this motion are inherently similar. A drastic reduction is also sufficient: If we attempt to reconstruct original control motions from our wave representation by the inverse linear map Φ^{+}, we measure reconstruction error as $\approx 0.3\times$ the standard deviation (over all dimensions) of the original signal. Empirically, our examples show that we compete with or exceed the flexibility, ease of control, and animation quality of alternative high-dimensional pose mapping approaches. In principle, the intermediate representation and filtering could also be extended to capture multiple harmonic frequency bands, which would allow multiple frequency motion detection, but this complicates user control and is harder to understand for novice users.

The applied Gabor filtering infers AFP from ≈one period of motion, which limits the ability to control speed and amplitude of character motions within a single period. Control can become difficult at very abrupt changes such as quickly transitioning to high jumps. There is a fundamental trade-off between robustness and responsiveness; though different filtering techniques might adapt to this specific case, e.g., cubature Kalman filter [Arasaratnam and Haykin 2009]. If desired, physical constraints could be incorporated to restrict unrealistic motions, such as stopping during jumps or flight phases.

Our approach to distinguishing gestures is robust as competing reference control motions are used as negative examples in training. We show scalability to many different control motions in our video; however, we have difficulty distinguishing gestures which are *very* similar, e.g., hand waving in a straight line vs. on an arc. Any control gestures which are more than two edges apart in a motion graph are independent, so 'scalability' must also consider *how many edges are typically needed per node?* For a game, this is governed by the available control DOFs. A typical gamepad has 10-12 binary buttons, two linear triggers, and two 2D linear thumbsticks. In our example of the dog controlled by the hand and face, we show independent linear or cyclic control of four locomotion, one emotion, and six action parameters. Under this comparison, we fare comparably: we have less binary states, but we offer more expression per DOF. While the scalability problem is general to gestural approaches, we somewhat relieve these restrictions through the graph design.

The question of intuitive gestures for arbitrary characters is open-ended, involving issues such as learnability, comfort, and user preference. We demonstrate a flexible approach which allows interactive mapping of different controls learned with a single reference control motion from the user. Our user study shows that people of many shapes and sizes could quickly adapt to using the system with good accuracy.

Our approach is applicable to characters created by various animation tools because it is always possible to export any animation

format to our mesh representation (e.g., rig or deformation cages), but it is harder to take our synthesized mesh animation and recover rig parameters to continue animation with traditional pipelines. In principle, our core mapping is independent of character representation; however, demonstrating a mapping to rigged characters remains future work.

Currently, we do not extrapolate outside the animation database. However, this would be possible by embedding into a latent space with extrapolation capabilities, e.g., non-linear Gaussian process latent variable model (GPLVM) embeddings [Lee and Elgammal 2004; Levine et al. 2012] and multi-dimensional scaling [Shin and Lee 2006; Cashman and Hormann 2012]. Our focus is instead on robust input generalization — the output of our algorithm could be used as input to these techniques to drive animation synthesis.

10 Conclusion

We present an approach to decompose robustly and in real-time high-dimensional input motions into wave parameters of amplitude, frequency, and phase for a set of control motions. We interactively learn a mapping from single reference examples of a user defined control motion to an intermediate 2D sinusoid representation, which lets us generalize variations of wave parameters during live motion. This provides intuitive control variation, particularly for cycles, and produces higher-quality character animation than competing approaches. For instance, when interpolating within a parametrized database, simply increasing the frequency of a gesture enables natural transitions from walking to running. Our approach applies to arbitrary characters, and so for quadrupeds, we solve the locomotion interpolation problem with a time-shifted approach that separates temporal alignment from pose interpolation and partially decouples character segments (e.g., limbs). In a user study, we verified that our system was intuitive to learn and operate, and applies well to different users, control motions, and motion trackers. As such, it had the potential to be useful for situations where traditional controllers are inappropriate, particularly for game and VR applications.

Acknowledgements

We thank Gabi Kussani, our professional Hohnsteiner puppeteer, our animators Gottfried Mentor and Cynthia Collins, Yeongho Seol for his correspondence and his dinosaur character, Gregorio Palmas and Hendrik Strobert for visualization help, and Michael Neff, Takaaki Shiratori, Kiran Varanasi, Simon Pilgrim, and all reviewers for their valuable discussion and feedback. This research was partially funded by the ERC Starting Grant project CapReal (335545). Kwang In Kim thanks EPSRC EP/M00533X/1. James Tompkin and Hanspeter Pfister thank NSF CGV-1110955.

References

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

AKHTER, I., SIMON, T., KHAN, S., MATTHEWS, I., AND SHEIKH, Y. 2012. Bilinear spatiotemporal basis models. *ACM TOG 31*, 2, 1–12.

ARASARATNAM, I., AND HAYKIN, S. 2009. Cubature Kalman filters. *IEEE Trans. Automatic Control 54*, 6, 1254–1269.

ASHRAF, G., AND WONG, K. C. 2000. Generating Consistent Motion Transition via Decoupled Framespace Interpolation. *CGF 19*, 3, 447–456.

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

BREGLER, C., LOEB, L., CHUANG, E., AND DESHPANDE, H. 2002. Turning to the masters: Motion capturing cartoons. *ACM TOG (Proc. SIGGRAPH) 21*, 3, 399–407.

CASAS, D., TEJERA, M., GUILLEMAUT, J.-Y., AND HILTON, A. 2012. 4D parametric motion graphs for interactive animation. In *Proc. I3D*, 103–110.

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

CELIKCAN, U., YAZ, I. O., AND CAPIN, T. 2014. Example-based retargeting of human motion to arbitrary mesh models. *CGF 34*, 1, 216–227.

CHAI, J., AND HODGINS, J. K. 2005. Performance animation from low-dimensional control signals. *ACM TOG (Proc. SIGGRAPH) 24*, 686–696.

CHEN, J., IZADI, S., AND FITZGIBBON, A. 2012. KinÊtre: animating the world with the human body. In *Proc. UIST*, 435–444.

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

DONTCHEVA, M., YNGVE, G., AND POPOVIĆ, Z. 2003. Layered acting for character animation. *ACM TOG (Proc. SIGGRAPH)*, 409–416.

FEICHTINGER, H. G., AND STROHMER, T. 1998. *Gabor analysis and algorithms: Theory and applications*. Springer Science & Business Media.

GLEICHER, M. 1998. Retargetting motion to new characters. In *Proc. SIGGRAPH*, ACM, 33–42.

HA, D., AND HAN, J. 2008. Motion synthesis with decoupled parameterization. *Vis. Comput. 24*, 7-9, 587–594.

HECK, R., AND GLEICHER, M. 2007. Parametric motion graphs. In *Proc. I3D*, 129–136.

HECKER, C., RAABE, B., ENSLOW, R. W., DEWEESE, J., MAYNARD, J., AND VAN PROOIJEN, K. 2008. Real-time motion retargeting to highly varied user-created morphologies. *ACM TOG (Proc. SIGGRAPH) 27*, 3, 27:1–27:11.

HELD, R., GUPTA, A., CURLESS, B., AND AGRAWALA, M. 2012. 3d puppetry: A kinect-based interface for 3d animation. *Proc. UIST*, 423–434.

HUANG, T.-C., HUANG, Y.-J., AND LIN, W.-C. 2013. Real-time horse gait synthesis. *Computer Animation and Virtual Worlds 24*, 2, 87–95.

IJSPEERT, A. J., NAKANISHI, J., HOFFMANN, H., PASTOR, P., AND SCHAAL, S. 2013. Dynamical movement primitives: Learning attractor models for motor behaviors. *Neural Comput. 25*, 2, 328–373.

ISHIGAKI, S., WHITE, T., ZORDAN, V. B., AND LIU, C. K. 2009. Performance-based control interface for character animation. *ACM TOG (Proc. SIGGRAPH) 28*, 3, 61:1–61:8.

JOHNSON, M. P., WILSON, A., BLUMBERG, B., KLINE, C., AND BOBICK, A. 1999. Sympathetic interfaces: Using a plush toy to direct synthetic characters. In *Proc. CHI*, 152–158.

KEOGH, E., AND PAZZANI, M. 2001. Derivative dynamic time warping. In *Proc. SIAM SDM*, 5–7.

KOVAR, L., GLEICHER, M., AND PIGHIN, F. 2002. Motion graphs. *ACM TOG (Proc. SIGGRAPH)*, 473–482.

KWON, Y., KIM, K. I., TOMPKIN, J., KIM, J. H., AND THEOBALT, C. 2015. Efficient learning of image super-resolution and compression artifact removal with semi-local Gaussian processes. *IEEE TPAMI 37*, 9, 1792–1805.

LASZLO, J., VAN DE PANNE, M., AND FIUME, E. 2000. Interactive control for physically-based animation. *Proc. SIGGRAPH*, 201–208.

LAWRENCE, N. D. 2004. Gaussian process latent variable models for visualisation of high dimensional data. *Proc. NIPS*, 329–336.

LEE, C., AND ELGAMMAL, A. 2004. Gait style and gait content: bilinear models for gait recognition using gait re-sampling. *Proc. FG*, 147–152.

LEE, J., CHAI, J., REITSMA, P. S., HODGINS, J. K., AND POLLARD, N. S. 2002. Interactive control of avatars animated with human motion data. In *ACM TOG (Proc. SIGGRAPH)*, vol. 21, ACM, 491–500.

LEE, Y., WAMPLER, K., BERNSTEIN, G., POPOVIĆ, J., AND POPOVIĆ, Z. 2010. Motion fields for interactive character locomotion. *ACM TOG (Proc. SIGGRAPH Asia) 29*, 6, 138:1–138:8.

LEVINE, S., WANG, J. M., HARAUX, A., POPOVIĆ, Z., AND KOLTUN, V. 2012. Continuous character control with low-dimensional embeddings. *ACM TOG (Proc. SIGGRAPH) 31*, 4, 1–10.

LOCKWOOD, N., AND SINGH, K. 2012. Finger walking: motion editing with contact-based hand performance. In *Proc. SCA*, 43–52.

MARTIN, T., AND NEFF, M. 2012. Interactive quadruped animation. In *Proc. Motion in Games*, 208–219.

NEFF, M., ALBRECHT, I., AND SEIDEL, H.-P. 2007. Layered performance animation with correlation maps. *CGF 26*, 3, 675–684.

OORE, S., TERZOPOULOS, D., AND HINTON, G. 2002. A desktop input device and interface for interactive 3D character animation. *Proc. Graphics Interface*, 133–140.

PULLEN, K., AND BREGLER, C. 2000. Animating by multi-level sampling. In *Proc. CA*, 36–42.

RAPTIS, M., KIROVSKI, D., AND HOPPE, H. 2011. Real-time classification of dance gestures from skeleton animation. In *Proc. SCA*, 147–156.

RASMUSSEN, C. E., AND WILLIAMS, C. K. I. 2006. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA.

RHODIN, H., TOMPKIN, J., KIM, K. I., VARANASI, K., SEIDEL, H.-P., AND THEOBALT, C. 2014. Interactive motion mapping for real-time character control. *CGF (Proc. Eurographics) 33*, 2.

ROSE, C., COHEN, M. F., AND BODENHEIMER, B. 1998. Verbs and adverbs: multidimensional motion interpolation. *IEEE CG&A 18*, 5, 32–40.

SEOL, Y., O'SULLIVAN, C., AND LEE, J. 2013. Creature features: online motion puppetry for non-human characters. In *Proc. SCA*, 213–221.

SHIN, H. J., AND LEE, J. 2006. Motion synthesis and editing in low-dimensional spaces. *Comput. Animat. Virtual Worlds 17*, 3-4, 219–227.

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

SHIRATORI, T., AND HODGINS, J. K. 2008. Accelerometer-based user interfaces for the control of a physically simulated character. *ACM TOG (Proc. SIGGRAPH Asia) 27*, 5, 123:1–123:9.

STURMAN, D. J. 1998. Computer puppetry. *IEEE CG&A 18*, 1, 38–45.

SUMNER, R. W., ZWICKER, M., GOTSMAN, C., AND POPOVIĆ, J. 2005. Mesh-based inverse kinematics. *ACM TOG (Proc. SIGGRAPH) 24*, 3, 488–495.

SUNG, M. 2013. Fast motion synthesis of quadrupedal animals using a minimum amount of motion capture data. *ETRI Journal 35*, 6, 1029–1037.

TERRA, S. C. L., AND METOYER, R. A. 2004. Performance timing for keyframe animation. In *Proc. SCA*, 253–258.

TOMLINSON, B., DOWNIE, M., BERLIN, M., GRAY, J., LYONS, D., COCHRAN, J., AND BLUMBERG, B. 2002. Leashing the alphawolves: Mixing user direction with autonomous emotion in a pack of semi-autonomous virtual characters. In *Proc. SCA*, 7–14.

UNUMA, M., ANJYO, K., AND TAKEUCHI, R. 1995. Fourier principles for emotion-based human figure animation. In *Proc. SIGGRAPH*, 91–96.

VÖGELE, A., HERMANN, M., KRÜGER, B., AND KLEIN, R. 2012. Interactive steering of mesh animations. In *Proc. SCA*, 53–58.

WEISE, T., BOUAZIZ, S., LI, H., AND PAULY, M. 2011. Realtime performance-based facial animation. *ACM TOG (Proc. SIGGRAPH) 30*, 4.

XIONG, X., AND DE LA TORRE, F. 2013. Supervised descent method and its applications to face alignment. In *Proc. CVPR*, 532–539.

YAMANE, K., ARIKI, Y., AND HODGINS, J. 2010. Animating non-humanoid characters with human motion data. In *Proc. SCA*, 169–178.