Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2010)
M. Otaduy and Z. Popovic (Editors)

Animating Non-Humanoid Characters with Human Motion Data

Katsu Yamane[1,2], Yuka Ariki[1,3], and Jessica Hodgins[2,1]

[1]Disney Research, Pittsburgh, USA
[2]Carnegie Mellon University, USA
[3]Nara Institute of Science and Technology, Japan

Abstract

This paper presents a method for generating animations of non-humanoid characters from human motion capture data. Characters considered in this work have proportion and/or topology significantly different from humans, but are expected to convey expressions and emotions through body language that are understandable to human viewers. Keyframing is most commonly used to animate such characters. Our method provides an alternative for animating non-humanoid characters that leverages motion data from a human subject performing in the style of the target character. The method consists of a statistical mapping function learned from a small set of corresponding key poses, and a physics-based optimization process to improve the physical realism. We demonstrate our approach on three characters and a variety of motions with emotional expressions.

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

1. Introduction

This paper presents a method for generating whole-body skeletal animations of non-humanoid characters from human motion capture data. Examples of such characters and snapshots of their motions are shown in Figure 1 along with the human motions from which the animations are synthesized. Such characters are often inspired by animals or artificial objects, and their limb lengths, proportions and even topology may be significantly different from humans. At the same time, the characters are expected to be anthropomorphic, i.e., convey expressions through body language understandable to human viewers, rather than moving as real animals.

Keyframing has been almost the only technique available to animate such characters. Although data-driven techniques using human motion capture data are popular for human animation, most of them do not work for non-humanoid characters because of the large differences between the skeletons and motion styles of the actor and the character. Capturing motions of the animal does not help solve the problem because animals cannot take directions as human actors can. Another possible approach is physical simulation, but it is very difficult to build controllers that generate plausible and stylistic motions.

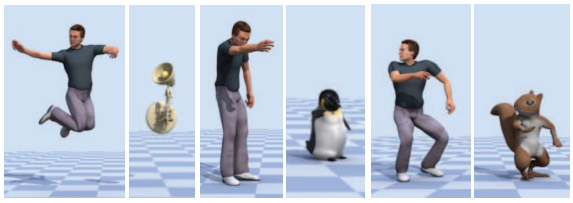

Figure 1: *Non-humanoid characters animated using human motion capture data.*

To create the motion of a non-humanoid character, we first capture motions of a human subject acting in the style of the target character. The subject then selects a few key poses from the captured motion sequence and creates corresponding character poses on a 3D graphics software system. The remaining steps can be completed automatically with little user interaction. The key poses are used to build a statistical model for mapping a human pose to a character pose. We can generate a sequence of poses by mapping every frame

DOI: 10.2312/SCA/SCA10/169-178

of the motion capture sequence using the mapping function. Finally, an optimization process adjusts the fine details of the motion, such as contact constraints and physical realism. We evaluate our approach by comparing to principal component analysis, nearest neighbors, and Gaussian processes, and verify that our method produces more plausible results.

Compared to keyframe animation, our method significantly reduces the time and cost required to create animations of non-humanoid characters. In our experiment, our method uses two hours for a motion capture session, 18 hours for selecting and creating key poses, and 70 minutes of computation time to generate 18 animations (7 minutes in total) of three characters, while an animator can spend weeks to create the same amount of animation by keyframing.

This paper is organized as follows: after reviewing the related work in Section 2, we present the overview of our method in Section 3. Sections 4 and 5 describe the two main components of our approach: the statistical model for mapping human poses to characters, and the dynamics optimization. Finally, we show the results in Section 6, followed by a discussion of limitations and future work in Section 7.

2. Related Work

While a number of algorithmic techniques have been developed for animating human characters, most of them are not applicable to non-humanoid characters because they assume that the target character has human-like proportions and topology [LS99, PW99, CK00, SLGS01], An exception is the work by Gleicher [Gle98], where he extended his motion retargetting technique to non-humanoid characters by explicitly specifying the correspondence of body parts in the original and new characters. Hecker et al. [HRMvP08] described an online system for game applications that can map motions in a motion library to any character created by players. Although the mapping algorithm is very powerful and flexible once a motion library is created, the animator has to annotate the motions in detail.

Baran et al. [BVGP09] developed a method for transferring deformation between two meshes, possibly with different topologies. A simple linear mapping function can generate plausible meshes for wide range of poses because of their rotation-invariant representation of meshes. Although this algorithm is more powerful than ours in the sense that it handles the character's mesh directly, working with the skeleton makes it easier to consider the dynamics and contact constraints.

In theory, Simulation- and physics-based techniques can handle any skeleton model and common tasks such as locomotion and balancing [WK88, LP02, JYL09, MZS09]. However, they are typically not suitable for synthesizing complex behaviors with specific styles due to the difficulty in developing a wide variety of controllers for characters of different morphologies.

Learning from artists' input to transfer styles between characters has been studied in Ikemoto et al. [IAF09]. They use artists' input to learn a mapping function based on Gaussian processes from a captured motion to a different character's motion. Their method requires that another animation sequence, edited from the original motion capture data, is provided to learn the mapping function. Such input gives much richer information about the correspondence than the isolated key poses used in our work, but it is more difficult to edit a full motion sequence using commonly available software. Bregler et al. [BLCD02] also developed a method that transfers a 2D cartoon style to different characters.

Our method employs a statistical model called shared Gaussian process latent variable models (shared GPLVM) [ETL07] to map a human pose to a character pose. Shon et al. [SGHR05] used a shared GPLVM to map human motion to a humanoid robot with many fewer degrees of freedom. Urtasun et al. [UFG*08] developed a method to incorporate explicit prior knowledge into GPLVM, allowing synthesis of transitions between different behaviors and with spacetime constraints. Grochow et al. [GMHP04] used another extension of GPLVM (scaled GPLVM) to bias the inverse kinematics computation to a specific style. These techniques use multiple sequences of motions to learn the models. In our work, we use a small set of key poses, rather than sequences, to learn a mapping function that covers a wide range of behaviors. We believe that it is much easier for actors and animators to create accurate character poses than to create appealing motion sequences, and that the dynamics, or velocity information, can best come from the actor's captured motion.

3. Overview

Figure 2 shows an overview of our animation synthesis process. The rectangular blocks indicate manual operations, while the rounded rectangles are automatic operations.

We first capture motions of a trained actor or actress performing in the style of the target character. We provide instructions about the capability and characteristics of the character and then rely on the actor's talent to portray how the character would act in a particular situation.

The actor then selects a few key poses among the captured motion sequences. The poses should be selected so that they cover and are representative of the space of the poses that appear in the captured motions. The last task for the actor is to create a character pose corresponding to each of the selected key poses. If necessary, an animator can operate a 3D graphics software system to manipulate the character's skeleton. This process is difficult to automate because the actor often has to make intelligent decisions to, for example, realize the same contact states on characters with completely different limb lengths. The actor may also want to add poses that are not possible for the human body, such as an extreme

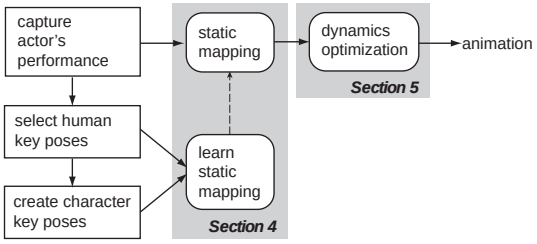

Figure 2: *Overview of the system. Rectangular blocks indicate manual operations and rounded rectangles are processed automatically.*

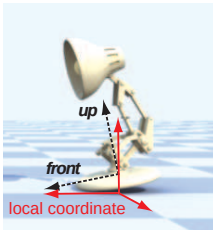

Figure 3: *The local coordinate frame (shown in solid, red line) for representing the feature point positions.*

back bend for a character that is much more flexible than humans.

The key poses implicitly define the correspondence between the body parts of the human and character models, even if the character's body has a different topology. The remaining two steps can be completed automatically without any user interaction. First we build a statistical model to map the human pose in each frame of the captured motion data to a character pose using the given key poses (Section 4). We then obtain the global transformation of the poses by matching the linear and angular momenta of the character motion to that of the human motion (Section 5). In many cases, there are still a number of visual artifacts in the motion such as contact points penetrating the floor or floating in the air. We therefore fine tune the motion by correcting the contact point positions and improving the physical realism through an optimization process taking into account the dynamics of the character.

4. Static Mapping

We employ a statistical method called shared Gaussian latent variable model (shared GPLVM) [ETL07, SGHR05] to learn a static mapping function from a human pose to a character pose. Shared GPLVM is suitable for our problem because human poses and corresponding character poses will likely have some underlying nonlinear relationship. Moreover, shared GPLVM gives a probability distribution over the character poses, which can potentially be used for adjusting the character pose to satisfy other constraints.

Shared GPLVM is an extension of GPLVM [Law03], which models the nonlinear mapping from a low-dimensional space (latent space) to an observation space. Shared GPLVM extends GPLVM by allowing multiple observation spaces sharing a common latent space. The main objective of using shared GPLVM in prior work is to limit the output space with ambiguity due to, for example, monocular video [ERT*08]. Although our problem does not involve ambiguity, we adopt shared GPLVM because we only have a sparse set of corresponding key poses. We expect that there is a common causal structure between human and character motions. In addition, it is known that a wide variety of human motions are confined to a relatively low-dimensional space [SHP04]. A model with a shared latent space would be an effective way to discover and model the space that represents that underlying structure.

Our mapping problem involves two observation spaces: the D_Y-dimensional human pose space and the D_Z-dimensional character pose space. These spaces are associated with a D_X-dimensional latent space. In contrast to the existing techniques that use time-series data for learning a model, the main challenge in our problem is that the given samples are very sparse compared to the complexity of the human and character models.

4.1. Motion Representation

There are several options to represent poses of human and character models. In our implementation, we use the Cartesian positions of multiple feature points on the human and character bodies, as done in some previous work [Ari06]. For the human model, we use motion capture markers because marker sets are usually designed so that they can well represent human poses. Similarly, we define a set of virtual markers for the character model by placing three markers on each link of the skeleton, and use their positions to represent character poses.

The Cartesian positions must be converted to a local coordinate frame to make them invariant to global transformations. In this paper, we assume that the height and roll/pitch angles are important features of a pose, and therefore only cancel out the horizontal position and yaw angle. For this purpose, we determine a local coordinate frame to represent the feature point positions.

The local coordinate is determined based on the root position and orientation as follows (Figure 3). We assume that two local vectors are defined for the root joint: the *front* and *up* vectors that point in the front and up directions of the model. The position of the local coordinate is simply the projection of the root location to a horizontal plane with a constant height. The z axis of the local coordinate points in

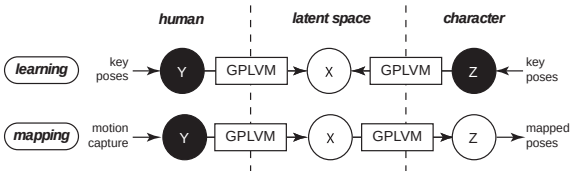

Figure 4: *Outline of the learning and mapping processes. The inputs are drawn with black background.*

the vertical direction. The x axis faces the heading direction of the root joint, which is found by first obtaining the single-axis rotation to make the *up* vector vertical, and then applying the same rotation to the *front* vector. The y axis is chosen to form a right-hand system.

For each key pose i, we form the observation vectors y_i and z_i by concatenating the local-coordinate Cartesian position vectors of the feature points of the human and character models, respectively. We then collect the vectors for all key poses to form observation matrices Y and Z. We denote the latent coordinates associated with the observations by X.

4.2. Learning and Mapping

The learning and mapping processes are outlined in Figure 4, where the inputs are drawn with a black background. In the learning process, the parameters of the GPLVMs and the latent coordinates for each key pose are obtained by maximizing the likelihood of generating the given pair of key poses. In the mapping process, we obtain the latent coordinates for each motion capture frame that maximize the likelihood of generating the given human pose. The latent coordinates are then used to calculate the character pose using GPLVM.

An issue in shared GPLVM is how to determine the dimension of the latent space. We employ several criteria as detailed in Section 6.2 for this purpose.

4.2.1. Learning

A GPLVM [Law03] parameterizes the nonlinear mapping function from the latent space to observation space by a kernel matrix. The (i,j) element of the kernel matrix K represents the similarity between two data points in the latent space x_i and x_j, and is calculated by

$$K_{ij} = k(x_i,x_j) = \theta_1 \exp\left\{-\frac{\theta_2}{2}||x_i - x_j||^2\right\} + \theta_3 + \beta^{-1}\delta_{ij} \tag{1}$$

where $\Phi = \{\theta_1,\theta_2,\theta_3,\beta\}$ are the model parameters and δ represents the delta function. We denote the parameters of the mapping functions from latent space to human pose by Φ_Y and from latent space to character pose by Φ_Z.

Assuming a zero-mean Gaussian process prior on the functions that generates the observations from a point in the latent space, the likelihoods of generating the given observations are formulated as

$$P(Y|X,\Phi_Y) = \frac{1}{\sqrt{(2\pi)^{ND_Y}|K_Y|^{D_Y}}} \exp\left\{-\frac{1}{2}\sum_{k=1}^{D_Y} y_k^T K_Y^{-1} y_k\right\}$$

$$P(Z|X,\Phi_Z) = \frac{1}{\sqrt{(2\pi)^{ND_Z}|K_Z|^{D_Z}}} \exp\left\{-\frac{1}{2}\sum_{k=1}^{D_Z} z_k^T K_Z^{-1} z_k\right\}$$

where K_Y and K_Z are the kernel matrices calculated using Eq.(1) with Φ_Y and Φ_Z respectively, and y_k and z_k denote the k-th dimension of the observation matrices Y and Z respectively. Using these likelihoods and priors for Φ_y, Φ_z and X, we can calculate the joint likelihood as

$$\begin{aligned} P_{GP}(Y,Z|X,\Phi_Y,\Phi_Z) &= P(Y|X,\Phi_Y)P(Z|X,\Phi_Z) \\ &\quad \times P(\Phi_Y)P(\Phi_Z)P(X). \end{aligned} \tag{2}$$

Learning shared GPLVM is essentially an optimization process to obtain the model parameters Φ_Y, Φ_Z and latent coordinates X that maximize the joint likelihood. The latent coordinates are initialized using Kernel Canonical Correlation Analysis (CCA) [Aka01].

After the model parameters Φ_Z are learned, we can obtain the probability distribution of the character pose for given latent coordinates x by

$$\bar{z}(x) = \mu_Z + Z^T K_Z^{-1} k(x) \tag{3}$$

$$\sigma_Z^2(x) = k(x,x) - k(x)^T K_Z^{-1} k(x) \tag{4}$$

where $\bar{z}$ and σ_Z^2 are the mean and variance of the distribution respectively, μ_Z is the mean of the observations, and $k(x)$ is a vector whose i-th element is $k_i(x) = k(x,x_i)$.

4.2.2. Mapping

The mapping process starts by obtaining the latent coordinates that correspond to a new human pose using a method combining nearest neighbor search and optimization [ETL07]. For a new human pose y_{new}, we search for the key pose y_i with the smallest Euclidean distance to y_{new}. We then use the latent coordinates associated with y_i as the initial value for the gradient-based optimization process to obtain the latent coordinates $\hat{x}$ that maximize the likelihood of generating y_{new}, i.e.,

$$\hat{x} = \arg\max_x P(y_{new}|x,Y,X,\Phi_Y). \tag{5}$$

The optimization process converged in all examples we have tested. We use the latent coordinates $\hat{x}$ to obtain the distribution of the character pose using Eqs.(3) and (4).

5. Dynamics Optimization

The sequence of poses obtained so far does not include the global horizontal movement. It also does not preserve the contact constraints in the original human motion because they are not considered in the static mapping function.

Dynamics optimization is performed in three steps to solve these issues. We first determine the global transformation of the character based on the linear and angular momenta of the original human motion. We then correct the contact point positions based on the contact information. Finally, we improve the physical plausibility by solving an optimization problem based on the equations of motion of the character, a penalty-based contact force model, and the probability distribution given by the static mapping function.

5.1. Global Transformation

We determine the global transformation (position and orientation) of the character so that the linear and angular momenta of the character match those obtained by scaling the momenta in the human motion. We assume a global coordinate system whose z axis points in the vertical direction and x and y axes are chosen to form a right-hand system.

The goal of this step is to determine the linear and angular velocities, v and ω, of the local coordinate frame defined in Section 4.1. Let us denote the linear and angular momenta of the character at frame i in the result of the static mapping by $P_c(i)$ and $L_c(i)$ respectively. If the local coordinate moves at $v(i)$ and $\omega(i)$, the momenta would change to

$$\hat{P}_c(i) = P_c(i) + m_c v(i) + \omega(i) \times p(i) \quad (6)$$

$$\hat{L}_c(i) = L_c(i) + I_c(i)\omega(i) \quad (7)$$

where m_c is the total mass of the character, $p(i)$ is the the whole-body center of mass position represented in the local coordinate, and $I_c(i)$ is the moments of inertia of the character around the local coordinate's origin. Evaluating these equations requires the inertial parameters of individual links of the character model, which can be specified manually or automatically from the density and volume of the links.

We determine $v(i)$ and $\omega(i)$ so that $\hat{P}_c(i)$ and $\hat{L}_c(i)$ match the linear and angular momenta in the original human motion capture data, $P_h(i)$ and $L_h(i)$, after applying appropriate scaling to address the difference in kinematics and dynamics parameters. The method used to obtain the scaling parameters will be discussed in the next paragraph. Given the scaled linear and angular momenta $\hat{P}_h(i)$ and $\hat{L}_h(i)$, we can obtain $v(i)$ and $\omega(i)$ by solving a linear equation

$$\begin{pmatrix} m_c E & -[p(i)\times] \\ 0 & I_c(i) \end{pmatrix} \begin{pmatrix} v(i) \\ \omega(i) \end{pmatrix} = \begin{pmatrix} \hat{P}_h(i) - \hat{P}_c(i) \\ \hat{L}_h(i) - \hat{L}_c(i) \end{pmatrix} \quad (8)$$

where E is the 3×3 identity matrix. We integrate $v(i)$ and $\omega(i)$ to obtain the position and orientation of the local coordinate in the next frame. In our implementation, we only consider the horizontal transformation, i.e., linear velocity in the x and y directions and the angular velocity around the z axis because the other translation and rotation degrees of freedom are preserved in the key poses used for learning, and therefore appear in the static mapping results. We extract the appropriate rows and columns from Eq.(8) to remove the irrelevant variables.

The scaling factors are obtained from size, mass, and inertia ratios between the human and character models. Mass ratio is $s_m = m_c/m_h$ where m_h is the total mass of the human model. The inertia ratio consists of three values corresponding to the three rotational axes in the global coordinate. To calculate the inertia ratio, we obtain the moments of inertia of the human model around its local coordinate, $I_h(i)$. We then use the ratio of the diagonal elements $(s_{ix}\ s_{iy}\ s_{iz})^T = (I_{cxx}/I_{hxx}\ \ I_{cyy}/I_{hyy}\ \ I_{czz}/I_{hzz})^T$ as the inertia ratio. The size ratio also consists of three values representing the ratios in depth (along x axis of the local coordinate), width (y axis), and height (z axis). Because we cannot assume any topological correspondence between the human and character models, we calculate the average feature point velocity for each model when every degree of freedom is rotated at a unit velocity one by one. The size scale is then obtained from the velocities v_h for the human model and v_c for the character model as $(s_{dx}\ s_{dy}\ s_{dz})^T = (v_{cx}/v_{hx}\ \ v_{cy}/v_{hy}\ \ v_{cz}/v_{hz})^T$. Using these ratios, the scaled momenta are obtained as $\hat{P}_{h*} = s_m s_{d*} P_{h*}$, $\hat{L}_{h*} = s_{i*} L_{h*}$ where $* = \{x, y, z\}$.

5.2. Contact Point Adjustment

We then adjust the poses so that the points in contact stay at the same position on the floor, using the contact states in the original human motion. We assume that a corresponding human contact point is given for each of the potential contact points on the character. Potential contact points are typically chosen from the toes and heels, although other points may be added if other parts of the body are in contact. In our current system we manually determine the contact and flight phases of each point, although some automatic algorithms [IAF05] or additional contact sensors may be employed.

Once the contact and flight phases are determined for each contact point, we calculate the corrected position. For each contact phase, we calculate the average position during the contact phase and use its projection to the floor as the corrected position. To prevent discontinuities due to the correction, we also modify the contact point positions while the character is in flight phase by smoothly interpolating the position corrections at the end of the preceding contact phase Δc_0 and at the beginning of the following one Δc_1 as

$$\hat{c}(t) = c(t) + (1 - w(t))\Delta c_0 + w(t)\Delta c_1 \quad (9)$$

where c and $\hat{c}$ are the original and modified positions respectively, and $w(t)$ is a weighting function that smoothly transitions from 0 to 1 as the time t moves from the start time t_0 of the flight phase to the end time t_1. In our implementation, we use $w(t) = h^2(3 - 2h)$ where $h = (t - t_0)/(t_1 - t_0)$.

5.3. Optimizing the Physical Realism

Finally, we improve the physical realism by adjusting the vertical motion of the root so that the motion is consistent with the gravity and a penalty-based contact model.

We represent the position displacement from the original motion along a single axis by a set of N weighted radial basis functions (RBFs). In this paper, we use a Gaussian for RBFs, in which case the displacement Δz is calculated as

$$\Delta z(t) = \sum_{i=1}^{N} w_i \phi_i(t),\ \phi_i(t) = \exp\left\{-\frac{(t-T_i)^2}{\sigma^2}\right\} \quad (10)$$

where T_i is the center of the i-th Gaussian function and σ is the standard deviation of the Gaussian functions. In our implementation, we place the RBFs with a constant interval along the time axis and set σ to be twice that of the interval. We denote the vector composed by the RBF weights as $w = (w_i\ w_2\ \ldots w_N)^T$.

The purpose of the optimization is to obtain the weights w that optimize the three criteria: (1) preserve the original motion as much as possible, (2) maximize the physical realism, and (3) maximize the likelihood with respect to the distribution output by the mapping function. Accordingly, the cost function to minimize is

$$Z = \frac{1}{2} w^T w + k_1 Z_p + k_2 Z_m \quad (11)$$

where the first term of the right hand side tries to keep the weights small, and the second and third terms address the latter two criteria of the optimization. Parameters k_1 and k_2 are user-defined positive constants.

Z_p is used to maximize the physical realism and given by

$$Z_p = \frac{1}{2}\left\{(F-\hat{F})^T(F-\hat{F}) + (N-\hat{N})^T(N-\hat{N})\right\} \quad (12)$$

where F and N are the total external force and moment required to perform the motion, and $\hat{F}$ and $\hat{N}$ are the external force and moment from the contact forces.

We can calculate F and N by performing the standard inverse dynamics calculation (such as [LWP80]) and extracting the 6-axis force and moment at the root joint.

We calculate $\hat{F}$ and $\hat{N}$ from the positions and velocities of the contact points on the character used in Section 5.2, based on a penalty-based contact model. The normal contact force at a point whose height from the floor is z ($z < 0$ if penetrating) is given by

$$f_n(z,\dot{z}) = \frac{1}{2}k_P\left(\sqrt{z^2 + \frac{4f_0^2}{k_P^2}} - z\right) - k_D g(z)\dot{z} \quad (13)$$

$$g(z) = \begin{cases} 1 - \frac{1}{2}\exp(kz) & (z < 0) \\ \frac{1}{2}\exp(-kz) & (0 \le z) \end{cases}$$

where the first and second terms of Eq.(13) correspond to the spring and damper forces respectively. When $\dot{z} = 0$, the asymptote of Eq.(13) is $f_n = -k_P z$ for $z \to -\infty$ and $f_n = 0$ for $z \to +\infty$, which is the behavior of the standard linear spring contact model with spring coefficient k_P. The formulation adopted here smoothly connects the two functions to produce a continuous force across the state space. The constant parameter f_0 denotes the residual contact force at $z = 0$ and indicates the amount of error from the linear spring contact model. The second term of Eq.(13) acts as a linear damper, except that the activation function $g(z)$ continuously reduces the force when the penetration depth is small or the point is above the floor. The spring and damping coefficients are generally chosen so that the ground penetration does not cause visual artifacts.

The friction force f_t is formulated as

$$\begin{aligned} f_t(r,\dot{r}) &= \mu f_n \hat{F}_t \\ \hat{F}_t &= h(f_{t0})F_{t0} \\ f_{t0} &= ||F_{t0}||,\ F_{t0} = -k_{tP}(r-\hat{r}) - k_{tD}\dot{r} \\ h(f_{t0}) &= \begin{cases} \frac{1-\exp(-k_t f_{t0})}{f_{t0}} & (\varepsilon < f_{t0}) \\ k_t & (f_{t0} < \varepsilon) \end{cases} \end{aligned} \quad (14)$$

where r is a two-dimensional vector representing the contact point position on the floor, $\hat{r}$ is nominal position of the contact point, μ is the friction coefficient, k_t, k_{tP} and k_{tD} are user-specified positive constants, and ε is a small positive constant. Friction force is usually formulated as $\mu f_n F_{t0}/f_{t0}$, which is a vector with magnitude μf_n and direction F_{t0}. To solve the singularity problem at $f_{t0} = 0$, we have introduced the function $h(f_{t0})$ that approaches $1/f_{t0}$ as $f_{t0} \to \infty$ and some finite value k_t as $f_{t0} \to 0$. The optimization is generally insensitive to the parameters used in Eq.(14).

The last term of Eq.(11), Z_m, represents the negative log-likelihood of the current pose, i.e.,

$$Z_m = -\sum_i \log P(z_i) \quad (15)$$

where i denotes the frame number and z_i is the position vector in the observation space formed from the feature points positions of the character at frame i. Function $P(z)$ gives the likelihood of generating a given vector z from the distribution given by Eqs.(3) and (4).

6. Results

We prepared three characters for the tests: *lamp*, *penguin*, and *squirrel* (Figure 5). The *lamp* character is an example of character inspired by an artificial object but yet able to perform human-like expressions using the arm and lamp shade as body and face. The completely different topology and locomotion style from humans make it difficult to animate the character. The *penguin* character has human-like topology but the limbs are extremely short with limited mobility. Although it still does biped walking, its locomotion style is also very different from humans because of its extremely short legs. The *squirrel* character has human-like topology

Figure 5: *Three characters used for the experiment: lamp, penguin and squirrel.*

but may also walk on four legs. The tail is occasionally animated during the key pose creation process, but we do not extensively animate the tail in the present work.

The software system consists of three components.

- An in-house C++ code library for reading motion capture data and key poses, converting them to feature point data, computing the inverse kinematics, and evaluating Z_p of the cost function.
- A publicly available MATLAB implementation of the learning and mapping functions of shared GPLVM [Law03].
- MATLAB code for evaluating Z_m of the cost function and performing the optimization using the MATLAB function *lsqnonlin*.

The parameters used in the examples are as follows:

- Eq.(11): $k_1 = 1 \times 10^{-5}$, $k_2 = 1$
- Eq.(13): $k_P = 1 \times 10^4$, $k_D = 1$, $f_0 = 1$, $k = 20$
- Eq.(14): $\mu = 1$, $k_{tP} = 0$, $k_{tD} = 100$, $k_t = 20$, $\varepsilon = 1 \times 10^{-6}$

6.1. Manual Tasks

We recorded the motions of a professional actor expressing six emotions (anger, disgust, fear, happiness, sadness and surprise) for each of the three characters. Before the motion capture session, we showed a picture of each character and verbally explained the kinematic properties (e.g., no or extremely short legs, may walk on four legs). The capture session lasted about two hours.

The actor and an animator worked together with a 3D graphics software system (Maya) to select key poses from the motion capture data and create corresponding poses for the characters. Some of the key poses are shown in the supplemental movie. It took 18 hours in total to select and create 115 key poses from 18 motion sequences, which averaged approximately 9 minutes per pose. The average interval between key poses in the motions is 3.6 seconds, which is obviously much longer than the interval in standard keyframe animations. Table 1 summarizes the statistics of the data obtained through this process.

Table 1: *Statistics of the measured and created data. Each column shows the duration of the sequence in seconds (left) and the number of key poses selected from each sequence (right).*

emotion	lamp		penguin		squirrel	
anger	18.6	16	14.3	3	22.9	14
disgust	34.4	1	23.3	7	20.0	3
fear	29.7	2	25.2	4	28.5	18
happiness	20.1	7	23.7	11	25.8	9
sadness	19.5	3	29.6	4	26.7	3
surprise	10.7	1	26.1	4	19.2	5
total	133	30	142	33	143	52

6.2. Static Mapping

We trained a shared GPLVM for each character using the key poses created by the actor and animator. An issue in using GPLVM is how to determine the dimension of the latent space, D_X. We use two criteria to determine D_X.

The first criteria is the error between the given and mapped character key poses. The error improved by increasing D_X up to 15 but did not improve much more at $D_X > 15$. We also found that 70 iterations is enough for optimizing the model parameters.

Another desired property is that the character motion becomes continuous when the human motion is continuous. Figure 6 shows the first two dimensions of the trajectories in the latent space when a human motion capture sequence (happy lamp) is input to the models when we used 2, 15 and 30 dimensional spaces. The 2-dimensional space, often used in the literature, is obviously not enough to describe the wide range of postures found in our data set. Although a 15-dimensional space is enough to reconstruct the training data, the trajectory in the latent space is still jerky and results in unnatural jumps in the character motion. We therefore use a 30-dimensional space for all examples.

The first and second columns of Figure 7 show snapshots of the original human poses and the results of static mapping. Note that the character stays above a fixed point on the floor because the horizontal movement has been removed from the key poses before learning the mapping function.

6.3. Dynamics Optimization

The third to fifth columns of Figure 7 depict the results of each step in the dynamics optimization process. First, a horizontal movement is attached to the mapped poses as shown in the third column. The contact point positions are corrected using the contact state information in the human motion as shown in the fourth column. Finally, an optimization process takes place to add physical realism to the motion and the result shown in the last column is obtained.

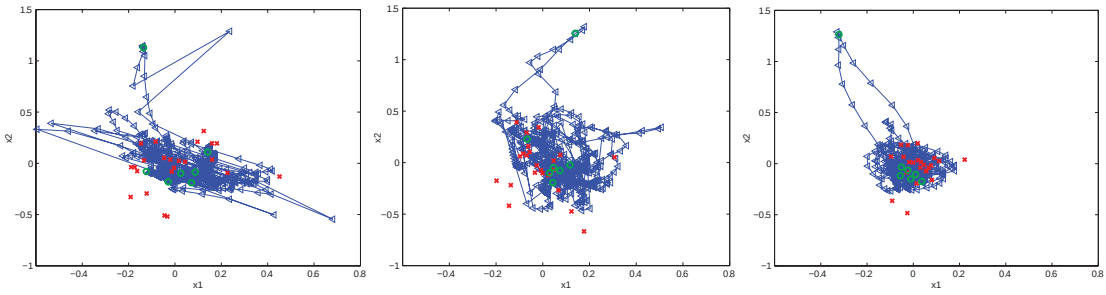

Figure 6: *Trajectories in the latent space when a human motion capture sequence is input to the model, projected to the first two dimensions of the latent space. From left to right: 2-, 15- and 30-dimensional latent spaces. The trajectory is represented by a line with triangles that denote the latent coordinates of individual frames. The (green) circles represent the key poses from the same motion sequence as the trajectory, and (red) crosses are the key poses from other motions.*

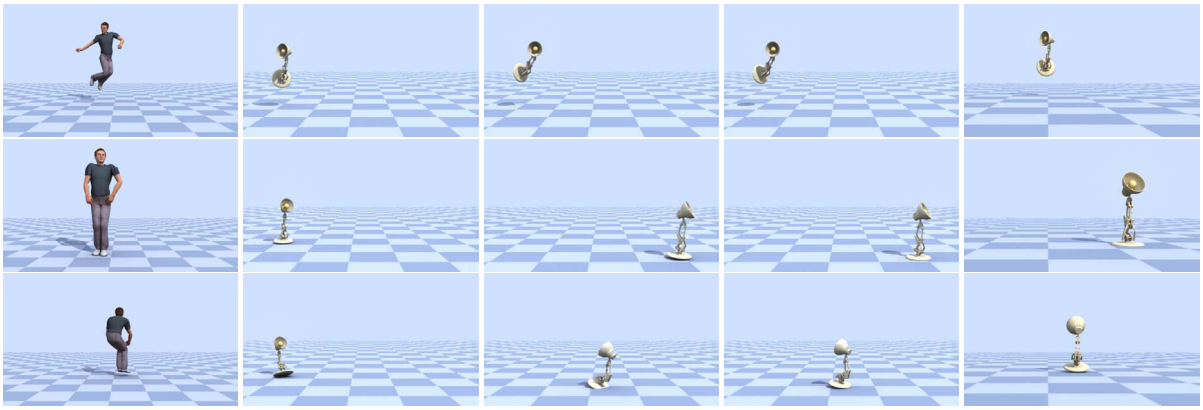

Figure 7: *Results from each step. From left to right: original human motion, after the static mapping, after adding horizontal movement based on the momentum, after correcting the contact point positions, and after adding physical realism.*

We show some of the numerical results during this process. Figure 8 shows the scaled human momentum and character momentum in the forward direction. The two graphs in Figure 9 show the required contact force (calculated by inverse dynamics) and the contact force from the contact model. The left and right graphs denote the forces before and after the optimization. The optimization algorithm tries to match the two forces by modifying the trajectory, which changes both the required and actual contact forces.

6.4. Examples

We show additional examples of the synthesized animations in the supplemental movie.

We compare shared GPLVM with the following mapping techniques:

- Principal component analysis and linear mapping (PCA): We obtain the 30-dimensional spaces that contain human and character key poses using principal component analysis, and then obtain a linear mapping between the two low-dimensional spaces.
- Linear interpolation of nearest neighbors (NN): We find the N nearest key poses using the Cartesian distance of the vector composed of human marker data, and obtain the weighted sum of the character feature point data where the weight is inversely proportional to the distance. We have tested with $N = 3$ (NN3) and $N = 10$ (NN10).
- Gaussian Process (GP): We obtain the Gaussian process model that maximizes the likelihood of generating the character key poses from the human key poses. We then use the mean of the distribution obtained by each motion capture data frame as the output of the model.

We use the happy lamp example and the mapped motions are processed by the dynamics optimization algorithm to obtain the final results.

Simple models (PCA and NN) cannot effectively model the nonlinear relationship between the human and character pose spaces with sparse examples. NN works better especially with 10 neighbors but does not generalize well

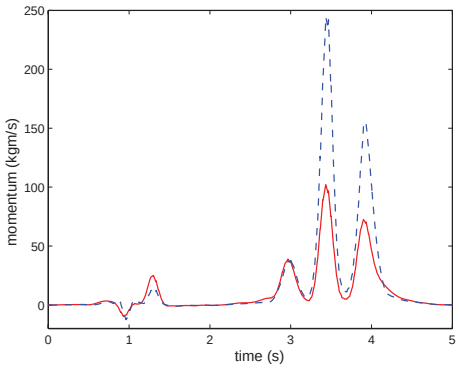

Figure 8: *Example of linear momentum in the forward direction; dashed (blue): scaled human momentum, solid (red): character's momentum after adding the horizontal movement.*

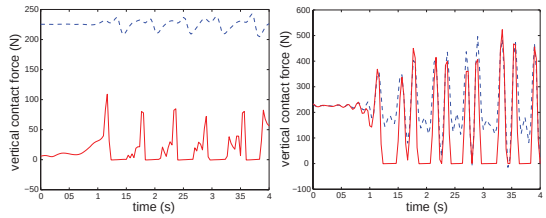

Figure 9: *Example of vertical contact force before (left) and after (right) optimization; dashed (blue): required contact force from inverse dynamics, solid (red): actual contact force from contact model.*

enough. The GP model produces results similar to the shared GPLVM, but slightly less dynamic. This issue could be fixed by applying the dynamics optimization to many DOFs but it may significantly modify the actor's original intention.

We also test the effect of the number of key poses by using GP and shared GPLVM learned from fewer key poses. We use the lamp model and remove the seven key poses from the happy motion for learning, and synthesize the happy lamp motion. Neither models can predict the extreme poses during the first high jump, which is expected because we do not have any example close to that pose. However, the shared GPLVM seem to have better generalization capability in terms of jump heights and head directions as demonstrated in the movie.

We also show an example generated by the retargeting function of Maya. Although the motion preserves some of the characteristics of the human motion, the fine details, especially the pose-specific correspondences, are lost.

Finally, the video shows several examples of the synthesized motions. The first three clips show some of the emotional motions to demonstrate that the original expression is successfully transferred to character motions. The last three clips show dance motions of the three models. The lamp and penguin dances are synthesized using only the key poses shown in Table 1, while we provided six more key poses to synthesize the squirrel dance.

7. Discussion

The basic idea of this approach is to leverage the creativity of actors in imagining how non-humanoid characters should move, in contrast to the current standard animation procedure where animators have creative control via 3D graphics software. Utilizing human performance can potentially drastically reduce the time required for creating animations because human performances are completed in real time. The use of motion capture also opens up a second pool of talent as actors are skilled at using their bodies to tell a story or convey an emotion. The problem with non-humanoid characters is that we have to map human motion to characters with significantly different proportions or topologies, which is the main technical contribution of this paper.

We tested our algorithm on three characters with different levels of similarity to humans. Although the squirrel model is the closest to humans in terms of the proportions and mobility, motions of the lamp and penguin models look more plausible and expressive. We hypothesize that this result occurs because the set of key poses for the squirrel model is not sufficient to cover its wide range of motion, and because viewers do not expect as much expression from the simple bodies of the lamp and penguin.

Our current system has several limitations. First, we assume that the key poses cover a sufficiently wide variety of poses that the character may take, relying on human intuition to select an appropriate set of key poses. In our experiments, the actor provided additional key poses for frames that looked particularly bad in the mapping results generated by the initial set of key poses. It might also be possible to provide an interface that indicates less covered areas using, for example, the variation map shown in Figure 6, or clustering the human poses in the motion capture data.

Our system currently does not consider the geometry of the character. This issue can be partially addressed during the key pose creation step by adjusting the character's poses so that they do not cause collisions in the body. However, we cannot avoid collisions if they happen between the key poses or with external objects, as shown in some of the examples.

Another limitation is that we cannot animate limbs that do not exist in the human body, such as the squirrel's tail. This issue can be handled by attaching a physical, controllable limb to the subject, or by manipulating the additional limb in the key pose creation step.

A few directions remain for future work. Combining this approach with a motion planning algorithm would be essential to avoid collisions within the body or the environment

because non-humanoid characters often have different geometry. We could take advantage of the probability distribution given by the mapping function by, for example, modifying less confident poses for collision avoidance.

Observing the process of creating key poses revealed some possible interfaces for the task. For example, the animator always started by matching the orientation of the character's root joint to that of the human. Another common operation was to match the direction of the faces. These operations can be easily automated and potentially speed up the key pose creation process. Determining poses of limbs and trunk, on the other hand, seems to require high-level reasoning that is difficult to automate.

The current algorithm is not realtime due to the optimization process. For appliactions that does not require physical consistency, we can omit the last step in Section 5 and synthesize motions in realtime because the first two steps are sufficiently fast. Realtime synthesis would open up some interesting applications such as interactive performance of non-humanoid characters by teleoperation. Currently such performances are animated by selecting from prerecorded motion sequences and therefore the variety of responses is limited. A realtime version of our system would allow much more flexible interaction.

Acknowledgements

Yuka Ariki was at Disney Research, Pittsburgh when she did the work. The authors would like to thank the following individuals for their help: Joel Ripka performed the three characters for the motion capture session and created the key poses; Justin Macey managed the motion capture session and cleaned up the data; Moshe Mahler created the character models, helped with the key pose creation process, and rendered the final animations and movie. Also thanks to Leon Sigal for advices on statistical models.

References

[Aka01] AKAHO S.: A kernel method for canonical correlation analysis. In *International Meeting of Psychometric Society* (2001).

[Ari06] ARIKAN O.: Compression of motion capture databases. *ACM Transactions on Graphics 25*, 3 (2006), 890–897.

[BLCD02] BREGLER C., LOEB L., CHUANG E., DESHPANDE H.: Turning to the masters: Motion capturing cartoons. *ACM Transactions on Graphics 21*, 3 (2002), 399–407.

[BVGP09] BARAN I., VLASIC D., GRINSPUN E., POPOVIĆ J.: Semantic deformation transfer. *ACM Transactions on Graphics 28*, 3 (2009), 36.

[CK00] CHOI K., KO H.: Online Motion Retargetting. *The Journal of Visualization and Computer Animation 11* (2000), 223–235.

[ERT*08] EK C. H., RIHAN J., TORR P. H. S., ROGEZ G., LAWRENCE N. D.: Ambiguity modeling in latent spaces. In *Workshop on Machine Learning and Multimodal Interaction* (2008), pp. 62–73.

[ETL07] EK C. H., TORR P. H. S., LAWRENCE N. D.: Gaussian process latent variable models for human pose estimation. In *Workshop on Machine Learning and Multimodal Interaction* (2007), pp. 132–143.

[Gle98] GLEICHER M.: Retargetting Motion to New Characters. In *Proceedings of SIGGRAPH '98* (1998), pp. 33–42.

[GMHP04] GROCHOW K., MARTIN S., HERTZMANN A., POPOVIĆ Z.: Style-based inverse kinematics. *ACM Transactions on Graphics 23*, 3 (2004), 522–531.

[HRMvP08] HECKER C., RAABE B., MAYNARD J., VAN PROOIJEN K.: Real-time motion retargeting to highly varied user-created morphologies. *ACM Transactions on Graphics 27*, 3 (2008), 27.

[IAF05] IKEMOTO L., ARIKAN O., FORSYTH D.: Knowing when to put your foot down. In *Proceedings of Symposium on Interactive 3D Graphics and Games* (2005).

[IAF09] IKEMOTO L., ARIKAN O., FORSYTH D.: Generalizing motion edits with gaussian processes. *ACM Transactions on Graphics 28*, 1 (2009), 1.

[JYL09] JAIN S., YE Y., LIU C.: Optimization-based interactive motion synthesis. *ACM Transactions on Graphics 28*, 1 (2009), 10.

[Law03] LAWRENCE N. D.: Gaussian process latent variable models for visualisation of high dimensional data. In *Advances in Neural Information Processing Systems (NIPS)* (2003).

[LP02] LIU C. K., POPOVIĆ Z.: Synthesis of Complex Dynamic Character Motion from Simple Animations. *ACM Transactions on Graphics 21*, 3 (2002), 408–416.

[LS99] LEE J., SHIN S.: A Hierarchical Approach to Interactive Motion Editing for Human-like Figures. In *Proceedings of ACM SIGGRAPH '99* (Los Angeles, CA, 1999), pp. 39–48.

[LWP80] LUH J., WALKER M., PAUL R.: Resolved Acceleration Control of Mechanical Manipulators. *IEEE Transactions on Automatic Control 25*, 3 (1980), 468–474.

[MZS09] MACCHIETTO A., ZORDAN V., SHELTON C.: Momentum control for balance. *ACM Transactions on Graphics 28*, 3 (2009).

[PW99] POPOVIĆ Z., WITKIN A.: Physically Based Motion Transformation. In *Proceedings of SIGGRAPH '99* (Los Angeles, CA, 1999), pp. 11–20.

[SGHR05] SHON A. P., GROCHOW K., HERTZMANN A., RAO R. P. N.: Learning shared latent structure for image synthesis and robotic imitation. In *Advances in Neural Information Processing Systems (NIPS)* (2005).

[SHP04] SAFONOVA A., HODGINS J., POLLARD N.: Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. *ACM Transactions on Graphics 23*, 3 (2004), 514–521.

[SLGS01] SHIN H., LEE J., GLEICHER M., SHIN S.: Computer puppetry: an importance-based approach. *ACM Transactions on Graphics 20*, 2 (2001), 67–94.

[UFG*08] URTASUN R., FLEET D., GEIGER A., POPOVIC J., DARRELL T., LAWRENCE N.: Topologically-constrained latent variable models. In *Proceedings of the 25th International Conference on Machine Learning* (2008), pp. 1080–1087.

[WK88] WITKIN A., KASS M.: Spacetime constraints. *ACM SIGGRAPH Computer Graphics 22*, 4 (1988), 159–168.