Style-Based Inverse Kinematics

Keith Grochow[1] Steven L. Martin[1] Aaron Hertzmann[2] Zoran Popović[1]

[1]University of Washington [2]University of Toronto

Abstract

This paper presents an inverse kinematics system based on a learned model of human poses. Given a set of constraints, our system can produce the most likely pose satisfying those constraints, in real-time. Training the model on different input data leads to different styles of IK. The model is represented as a probability distribution over the space of all possible poses. This means that our IK system can generate any pose, but prefers poses that are most similar to the space of poses in the training data. We represent the probability with a novel model called a Scaled Gaussian Process Latent Variable Model. The parameters of the model are all learned automatically; no manual tuning is required for the learning component of the system. We additionally describe a novel procedure for interpolating between styles.

Our style-based IK can replace conventional IK, wherever it is used in computer animation and computer vision. We demonstrate our system in the context of a number of applications: interactive character posing, trajectory keyframing, real-time motion capture with missing markers, and posing from a 2D image.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation; I.2.9 [Artificial Intelligence]: Robotics—Kinematics and Dynamics; G.3 [Artificial Intelligence]: Learning

Keywords: Character animation, Inverse Kinematics, motion style, machine learning, Gaussian Processes, non-linear dimensionality reduction, style interpolation

1 Introduction

Inverse kinematics (IK), the process of computing the pose of a human body from a set of constraints, is widely used in computer animation. However, the problem is inherently underdetermined: for example, for given positions of the hands and feet of a character, there are many possible character poses that satisfy the constraints. Even though many poses are *possible*, some poses are *more likely* than others — an actor asked to reach forward with his arm will most likely reach with his whole body, rather than keeping the rest of the body limp. In general, the likelihood of poses depends on the body shape and style of the individual person, and designing this likelihood function by hand for every person would be a difficult or impossible task. Current metrics in use by IK systems (such as distance to some default pose, minimum mass displacement between poses, or kinetic energy) do not accurately represent the space of natural poses. Moreover, these systems attempt to represent all styles with a single metric.

In this paper, we present an IK system based on learning from previously-observed poses. We pose IK as maximization of an objective function that describes how desirable the pose is — the optimization can satisfy *any* constraints for which a feasible solution exists, but the objective function specifies how desirable each pose is. In order for this system to be useful, there are a number of important requirements that the objective function should satisfy. First, it should accurately represent the space of poses represented by the training data. This means that it should prefer poses that are "similar" to the training data, using some automatic measure of similarity. Second, it should be possible to optimize the objective function in real-time — even if the set of training poses is very large. Third, it should work well when there is very little data, or data that does not have much redundancy (a case that leads to overfitting problems for many models). Finally, the objective function should not require manual "tuning parameters;" for example, the similarity measure should be learned automatically. In practice, we also require that the objective function be smooth, in order to provide a good space of motions, and to enable continuous optimization.

The main idea of our approach is to represent this objective function over poses as a Probability Distribution Function (PDF) which describes the "likelihood" function over poses. Given training poses, we can learn all parameters of this PDF by the standard approach of maximizing the likelihood of the training data. In order to meet the requirements of real-time IK, we represent the PDF over poses using a novel model called as a Scaled Gaussian Process Latent Variable Model (SGPLVM), based on recent work by Lawrence [2004]. All parameters of the SGPLVM are learned automatically from the training data, the SGPLVM works well with small data sets, and we show how the objective function can be optimized for new poses in real-time IK applications. We additionally describe a novel method for interpolating between styles.

Our style-based IK can replace conventional IK, wherever it is used. We demonstrate our system in the context of a number of applications:

- **Interactive character posing,** in which a user specifies a single pose based on a few constraints;
- **Trajectory keyframing,** in which a user quickly creates an animation by keyframing the trajectories a few points on the body;
- **Real-time motion capture with missing markers,** in which 3D poses are computed from incomplete marker measurements; and
- **Posing from a 2D image,** in which a few 2D projection constraints are used to quickly estimate a 3D pose from an image.

The main limitation of our style-based IK system is that it requires suitable training data to be available; if the training data does not match the desired poses well, then more constraints will be needed. Moreover, our system does not explicitly model dynamics, or constraints from the original motion capture. However, we have found that, even with a generic training data set (such as walking or calibration poses), the style-based IK produces much more natural poses than existing approaches.

email: keithg@cs.washington.edu, steve0@cs.berkeley.edu, hertzman@dgp.toronto.edu, zoran@cs.washington.edu. Steve Martin is now at University of California at Berkeley.

2 Related work

The basic IK problem of finding the character pose that satisfies constraints is well studied, e.g., [Bodenheimer et al. 1997; Girard and Maciejewski 1985; Welman 1993]. The problem is almost always underdetermined, meaning that many poses satisfy the constraints. This is the case even with motion capture processing where constraints frequently disappear due to occlusion. Unfortunately, most poses that satisfy constraints will appear unnatural. In the absence of an adequate model of poses, IK systems employed in industry use very simple models of IK, e.g., performing IK only on individual limbs (as in Alias Maya), or measuring similarity to an arbitrary "reference pose." [Yamane and Nakamura 2003; Zhao and Badler 1998]. This leaves an animator with the task of specifying significantly more constraints than necessary.

Over the years, researchers have devised a number of techniques to restrict the animated character to stay within the space of natural poses. One approach is to draw from biomechanics and kinesiology, by measuring the contribution of individual joints to a task [Gullapalli et al. 1996], by minimizing energy consumption [Grassia 2000], or mass displacement from some default pose [Popović and Witkin 1999]. In general, describing styles of body poses is quite difficult this way, and many dynamic styles do not have a simple biomechanical interprepration.

A related problem is to create realistic animations from examples. One approach is to warp an existing animation [Bruderlin and Williams 1995; Witkin and Popović 1995] or to interpolate between sequences [Rose et al. 1998]. Many authors have described systems for producing new sequences of movements from examples, either by direct copying and blending of poses [Arikan and Forsyth 2002; Arikan et al. 2003; Kovar et al. 2002; Lee et al. 2002; Pullen and Bregler 2002] or by learning a likelihood function over sequences [Brand and Hertzmann 2000; Li et al. 2002]. These methods create animations from high-level constraints (such as approximate target trajectories or keyframes on the root position). In constrast, we describe a real-time IK system with fine-grained kinematic control. A novel feature of our system is the ability to satisfy arbitrary user-specified constraints in real-time, while maintaining the style of the training data. In general, methods based on direct copying and blending are conceptually simpler, but do not provide a principled way to create new poses or satisfy new kinematic constraints.

Our work builds on previous example-based IK systems [ElKoura and Singh 2003; Kovar and Gleicher 2004; Rose III et al. 2001; Wiley and Hahn 1997]. Previous work in this area has been limited to interpolating poses in highly-constrained spaces, such as reaching motions. This interpolation framework can be very fast in practice and is well suited to environments where the constraints are known in advance (e.g., that only the hand position will be constrained). Unfortunately, these methods require that all examples have the same constraints as the target pose; furthermore, interpolation does not scale well with the number of constraints (e.g., the number of examples required for Radial Basis Functions increases exponentially in the input dimension [Bishop 1995]). More importantly, interpolation provides a weak model of human poses: poses that do not interpolate or extrapolate the data cannot be created, and all interpolations of the data are considered equally valid (including interpolations between very dissimilar poses that have similar constraints, and extreme extrapolations). In constrast, our PDF-based system can produce full-body poses to satify *any* constraints (that have feasible solutions), but *prefers* poses that are most similar to the training poses. Furthemore, interpolation-based systems require a significant amount of parameter tuning, in order to specify the constraint space and the similarity function between poses; our system learns all parameters of the probability model automatically.

Video motion capture using models learned from motion capture data is an active area of research [Brand 1999; Grauman et al. 2003; Howe et al. 2000; Ramanan and Forsyth 2004; Rosales and Sclaroff 2002; Sidenbladh et al. 2002]. These systems are similar to our own in that a model is learned from motion capture data, and then used to prefer more likely interpretations of input video. Our system is different, however, in that we focus on new, interactive graphics applications and real-time synthesis. We suspect that the SGPLVM model proposed in our paper may also be advantageous for computer vision applications.

A related problem in computer vision is to estimate the pose of a character, given known correspondences between 2D images and the 3D character (e.g., [Taylor 2000]). Existing systems typically require correspondences to be specified for every handle, user guidance to remove ambiguities, or multiple frames of a sequence. Our system can estimate 3D poses from 2D constraints from just a few point correspondences, although it does require suitable training data to be available.

A few authors have proposed methods for style interpolation in motion analysis and synthesis. Rose et al. [1998] interpolate motion sequences with the same sequences of moves to change the styles of those movements. Wilson and Bobick [1999] learn a space of Hidden Markov Models (HMMs) for hand gestures in which the spacing is specified in advance, and Brand and Hertzmann [2000] learn HMMs and a style-space describing human motion sequences. All of these methods rely on some estimate of correspondence between the different training sequences. Correspondence can be quite cumbersome to formulate and creates undesirable constraints on the problem. For example, the above HMM approaches assume that all styles have the same number of states and the same state transition likelihoods. In contrast, we take a simpler approach: we learn a separate PDF for each style, and then generate new styles by interpolation of the PDFs in the log-domain. This approach is very easy to formulate and to apply, and, in our experience, works quite well. One disadvantage, however, is that our method does not share information between styles during learning.

3 Overview

The main idea of our work is to learn a probability distribution function (PDF) over character poses from motion data, and then use this to select new poses during IK. We represent each pose with a 42-dimensional vector $\mathbf{q}$, which consists of joint angles, and the position and orientation of the root of the kinematic chain. Our approach consists of the following steps:

Feature vectors. In order to provide meaningful features for IK, we convert each pose vector to a feature representation $\mathbf{y}$ that represents the character pose and velocity in a local coordinate frame. Each motion capture pose $\mathbf{q}_i$ has a corresponding *feature vector* $\mathbf{y}_i$, where i is an index over the training poses. These features include joint angles, velocity, and vertical orientation, and are described in detail in Section 4.

SGPLVM learning. We model the likelihood of motion capture poses using a novel model called a Scaled Gaussian Process Latent Variable Model (SGPLVM). Given the features $\{\mathbf{y}_i\}$ a set of motion capture poses, we learn the parameters of an SGPLVM, as described in Section 5. The SGPLVM defines a low-dimensional representation of the original data: every pose $\mathbf{q}_i$ has a corresponding vector $\mathbf{x}_i$, usually in a 3-dimensional space. The low-dimensional space of $\mathbf{x}_i$ values is called the *latent space*. In the learning process, we estimate the $\{\mathbf{x}_i\}$ parameters for each input pose, along with the parameters of the SGPLVM model (denoted α, β, γ, and $\{w_k\}$). This learning process entails numerical optimization of an objective function L_{GP}. The likelihood of new poses is then described by the original poses and the model parameters. In order to keep the

model efficient, the algorithm selects a subset of the original poses to keep, called the *active set*.

Pose synthesis. To generate new poses, we optimize an objective function $L_{IK}(\mathbf{x}, \mathbf{y}(\mathbf{q}))$, which is derived from the SGPLVM model. This function describes the likelihood of new poses, given the original poses and the learned model parameters. For each new pose, we also optimize the low-dimensional vector $\mathbf{x}$. Several different applications are supported, as described in Section 7.

4 Character model

In this section, we define the parameterization we use for characters, as well as the features that we use for learning. We describe the 3D pose of a character with a vector $\mathbf{q}$ that consists of the global position and orientation of the root of the kinematic chain, plus all of the joint angles in the body. The root orientation is represented as a quaternion, and the joint angles are represented as exponential maps. The joint parameterizations are rotated so that the space of natural motions does not include singularities in the parameterization.

For each pose, we additionally define a corresponding D-dimensional *feature vector* $\mathbf{y}$. This feature vector selects the features of character poses that we wish the learning algorithm to be sensitive to. This vector includes the following features:

- **Joint angles**: All of the joint angles from $\mathbf{q}$ are included. We omit the global position and orientation, as we do not want the learning to be sensitive to them.
- **Vertical orientation**: We include a feature that measures the global orientation of the character with respect to the "up direction," (along the Z-axis) defined as follows. Let $\mathbf{R}$ be a rotation matrix that maps a vector in the character's local coordinate frame to the world coordinate frame. We take the three canonical basis vectors in the local coordinate frame, rotate them by this matrix, and take their Z-components, to get an estimate to the degree that the character is leaning forward and to the side. This reduces to simply taking the third row of $\mathbf{R}$.
- **Velocity and acceleration**: In animations, we would like the new pose to be sensitive to the pose in the previous time frame. Hence, we use velocity and acceleration vectors for each of the above features. For a feature vector at time t, the velociy and acceleration are given by $\mathbf{y}_t - \mathbf{y}_{t-1}$ and $\mathbf{y}_t - 2\mathbf{y}_{t-1} + \mathbf{y}_{t-2}$, respectively.

The features for a pose may be computed from the current frame and the previous frame. We write this as a function $\mathbf{y}(\mathbf{q})$. We omit the previous frames from the notation, as they are always constant in our applications. All vectors in this paper are column vectors.

5 Learning a model of poses

In this section, we describe the Scaled Gaussian Process Latent Variable Model (SGPLVM), and a procedure for learning the model parameters from training poses. The model is based on the Gaussian Process (GP) model, which describes the mapping from $\mathbf{x}$ values to $\mathbf{y}$ values. GPs for interpolation were introduced by O'Hagan [1978], Neal [1996] and Williams and Rasmussen [1996]. For a detailed tutorial on GPs, see [MacKay 1998]. We additionally build upon the Gaussian Process Latent Variable Model, recently poposed by Lawrence [2004]. Although the mathematical background for GPs is somewhat involved, the implementation is straightforward.

Kernel function. Before describing the learning algorithm, we first define the parameters of the GP model. A GP model describes the mapping between $\mathbf{x}$ values and $\mathbf{y}$ values: given some training data $\{\mathbf{x}_i, \mathbf{y}_i\}$, the GP predicts the likelihood of a new $\mathbf{y}$ given a new $\mathbf{x}$. A key ingredient of the GP model is the definition of a *kernel function* that measures the similarity between two points $\mathbf{x}$ and $\mathbf{x}'$ in the input space:

$$k(\mathbf{x}, \mathbf{x}') = \alpha \exp\left(-\frac{\gamma}{2}||\mathbf{x} - \mathbf{x}'||^2\right) + \delta_{\mathbf{x},\mathbf{x}'}\beta^{-1} \quad (1)$$

The variable $\delta_{\mathbf{x},\mathbf{x}'}$ is 1 when $\mathbf{x}$ and $\mathbf{x}'$ are the same point, and 0 otherwise, so that $k(\mathbf{x}, \mathbf{x}) = \alpha + \beta^{-1}$ and the $\delta_{\mathbf{x},\mathbf{x}'}$ term vanishes whenever the similarity is measured between two distinct variables. The kernel function tells us how correlated two data values $\mathbf{y}$ and $\mathbf{y}'$ are, based on their corresponding $\mathbf{x}$ and $\mathbf{x}'$ values. The parameter γ tells us the "spread" of the similarity function, α tells us how correlated pairs of points are in general, and β tells us how much noise there is in predictions. For a set of N input vectors $\{\mathbf{x}_i\}$, we define the $N \times N$ *kernel matrix* $\mathbf{K}$, in which $\mathbf{K}_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j)$.

The different data dimensions have different intrinsic scales (or, equivalently, different levels of variance): a small change in global rotation of the character affects the pose much more than a small change in the wrist angle; similarly, orientations vary much more than their velocities. Hence, we will need to estimate a separate scaling w_k for each dimension. This scaling is collected in a diagonal matrix $\mathbf{W} = \mathrm{diag}(w_1, ..., w_D)$; this matrix is used to rescale features as $\mathbf{Wy}$.

Learning. We now describe the process of learning an SGPLVM, from a set of N training data points $\{\mathbf{y}_i\}$. We first compute the mean of the training set: $\mu = \sum \mathbf{y}_i / N$. We then collect the k-th component of every feature vector into a vector $\mathbf{Y}_k$ and subtract the means (so that $\mathbf{Y}_k = [\mathbf{y}_{1,k} - \mu_k, ..., \mathbf{y}_{N,k} - \mu_k]^T$). The SGPLVM model parameters are learned by minimizing the following objective function:

$$L_{GP} = \frac{D}{2}\ln|\mathbf{K}| + \frac{1}{2}\sum_k w_k^2 \mathbf{Y}_k^T \mathbf{K}^{-1} \mathbf{Y}_k + \frac{1}{2}\sum_i ||\mathbf{x}_i||^2 + \ln\frac{\alpha\beta\gamma}{\prod_k w_k^N} \quad (2)$$

with respect to the unknowns $\{\mathbf{x}_i\}, \alpha, \beta, \gamma$ and $\{w_k\}$. This objective function is derived from the Gaussian Process model (Appendix A). Formally, L_{GP} is the negative log-posterior of the model parameters. Once we have optimized these parameters, the SGPLVM provides a likelihood function for use in real-time IK, based on the training data and the model parameters.

Intuitively, minimizing this objective function arranges the $\mathbf{x}_i$ values in the latent space so that similar poses are nearby and the dissimilar poses are far apart, and learns the smoothness of the space of poses. More generally, we are trying to adjust all unknown parameters so that the kernel matrix $\mathbf{K}$ matches the correlations in the original $\mathbf{y}$'s (Appendix A). Learning in the SGPLVM model generalizes conventional PCA [Lawrence 2004], which corresponds to fixing $w_k = 1$, $\beta^{-1} = 0$, and using a linear kernel. As described below, the SGPLVM also generalizes Radial Basis Function (RBF) interpolation, providing a method for learning all RBF parameters and for constrained pose optimization.

The simplest way to minimize L_{GP} is with numerical optimization methods such as L-BFGS [Nocedal and Wright 1999]. However, in order for the real-time system to be efficient, we would like to discard some of the training data; the training points that are kept are called the *active set*. Once we have optimized the unknowns, we use a heuristic [Lawrence et al. 2003] to determine the active set. Moreover, the optimization itself may be inefficient for large datasets, and so we use a heuristic optimization based on Lawrence's [2004] in order to efficiently learn the model parameters and to select the active set. This algorithm alternates between

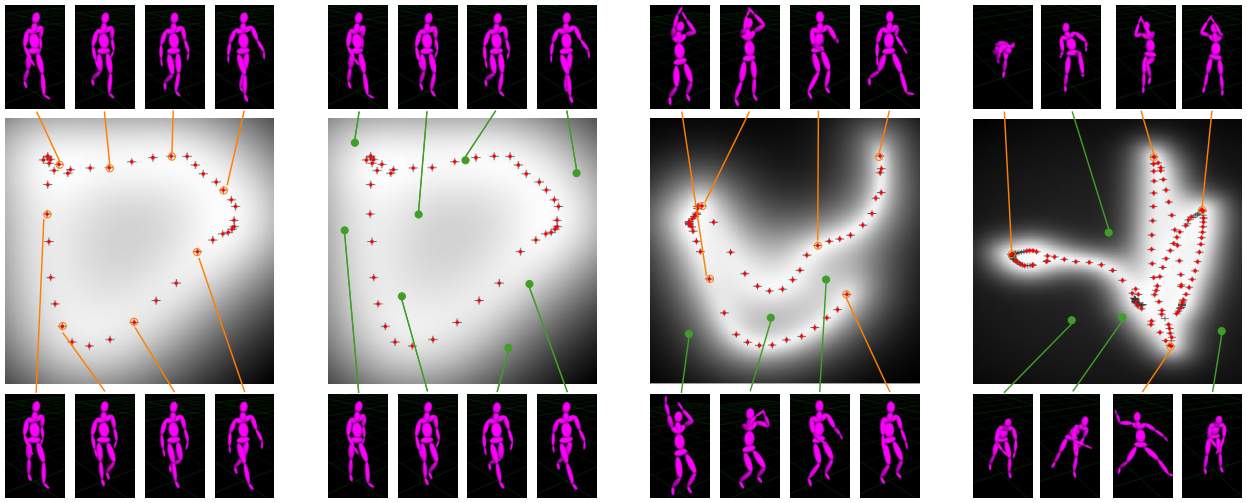


Figure 1: SGPLVM latent spaces learned from different motion capture sequences: a walk cycle, a jump shot, and a baseball pitch. ***Points:*** The learning process estimates a 2D position $\mathbf{x}$ associated with every training pose; plus signs (+) indicate positions of the original training points in the 2D space. Red points indicate training poses included in the training set. ***Poses:*** Some of the original poses are shown along with the plots, connected to their 2D positions by orange lines. Additionally, some novel poses are shown, connected by green lines to their positions in the 2D plot. Note that the new poses extrapolate from the original poses in a sensible way, and that the original poses have been arranged so that similar poses are nearby in the 2D space. ***Likelihood plot:*** The grayscale plot visualizes $-\frac{D}{2}\ln\sigma^2(\mathbf{x}) - \frac{1}{2}||\mathbf{x}||^2$ for each position $\mathbf{x}$. This component of the inverse kinematics likelihood L_{IK} measures how "good" $\mathbf{x}$ is. Observe that points are more likely if they lie near or between similar training poses.

optimizing the model parameters, optimizing the latent variables, and selecting the active set. These algorithms and their tradeoffs are described in Appendix B. We require that the user specify the size M of the active set, although this could also be specified in terms of an error tolerance. Choosing a larger active set yields a better model, whereas a smaller active set will lead to faster performance during both learning and synthesis.

New poses. Once the parameters have been learned, we have a general-purpose probability distribution for new poses. The objective function for a new pose parameterized by $\mathbf{x}$ and $\mathbf{y}$ is:

$$L_{IK}(\mathbf{x},\mathbf{y}) = \frac{||\mathbf{W}(\mathbf{y}-\mathbf{f}(\mathbf{x}))||^2}{2\sigma^2(\mathbf{x})} + \frac{D}{2}\ln\sigma^2(\mathbf{x}) + \frac{1}{2}||\mathbf{x}||^2 \qquad (3)$$

where

$$\mathbf{f}(\mathbf{x}) = \mu + \overline{\mathbf{Y}}^T\overline{\mathbf{K}}^{-1}\mathbf{k}(\mathbf{x}) \qquad (4)$$

$$\sigma^2(\mathbf{x}) = k(\mathbf{x},\mathbf{x}) - \mathbf{k}(\mathbf{x})^T\overline{\mathbf{K}}^{-1}\mathbf{k}(\mathbf{x}) \qquad (5)$$

$$= \alpha + \beta^{-1} - \sum_{1\le i,j\le M} (\overline{\mathbf{K}}^{-1})_{ij}k(\mathbf{x},\mathbf{x}_i)k(\mathbf{x},\mathbf{x}_j) \qquad (6)$$

and $\overline{\mathbf{K}}$ is the kernel matrix for the active set, $\overline{\mathbf{Y}} = [\mathbf{y}_1 - \mu, ..., \mathbf{y}_M - \mu]^T$ is the matrix of active set points (mean-subtracted), and $\mathbf{k}(\mathbf{x})$ is a vector in which the i-th entry contains $k(\mathbf{x},\mathbf{x}_i)$, i.e., the similarity between $\mathbf{x}$ and the i-th point in the active set. The vector $\mathbf{f}(\mathbf{x})$ is the pose that the model would predict for a given $\mathbf{x}$; this is equivalent to RBF interpolation of the training poses. The variance $\sigma^2(\mathbf{x})$ indicates the uncertainty of this prediction; the certainty is greatest near the training data. The derivation of L_{IK} is given in Appendix A.

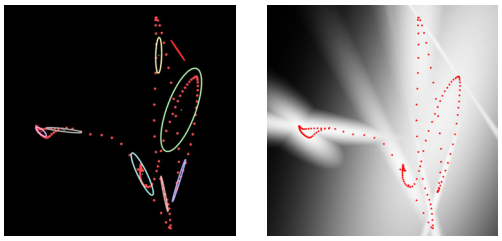
The objective function L_{IK} can be interpreted as follows. Optimization of a $(\mathbf{x},\mathbf{y})$ pair tries to simultaneously keep the $\mathbf{y}$ close to the corresponding prediction $\mathbf{f}(\mathbf{x})$ (due to the $||\mathbf{W}(\mathbf{y}-\mathbf{f}(\mathbf{x}))||^2$ term), while keeping the $\mathbf{x}$ value close to the training data (due to the $\ln\sigma^2(\mathbf{x})$ term), since this is where the prediction is most reliable. The $\frac{1}{2}||\mathbf{x}||^2$ term has very little effect on this process, and is included mainly for consistency with learning.

6 Pose synthesis

We now describe novel algorithms for performing IK with SGPLVMs. Given a set of motion capture poses $\{\mathbf{q}_i\}$, we compute the corresponding feature vectors $\mathbf{y}_i$ (as described in Section 4), and then learn an SGPLVM from them as described in the previous section. Learning gives us a latent space coordinate $\mathbf{x}_i$ for each pose $\mathbf{y}_i$, as well as the parameters of the SGPLVM (α, β, γ, and $\{w_k\}$). In Figure 1, we show SGPLVM likelihood functions learned from different training sequences. These visualizations illustrate the power of the SGPLVM to learn a good arrangement of the training poses in the latent space, while also learning a smooth likelihood function near the spaces occupied by the data. Note that the PDF is not simply a matter of, for example, Gaussian distributions centered at each training data point, since the spaces inbetween data points are more likely than spaces equidistant but outside of the training data. The objective function is smooth but multimodal.

Overfitting is a significant problem for many popular PDF models, particularly for small datasets without redundancy (such as the ones shown here). The SGPLVM avoids overfitting and yields smooth objective functions both for large and for small data sets (the technical reason for this is that it marginalizes over the space of model representations [MacKay 1998], which properly takes into account uncertainty in the model). In Figure 2, we compare with another common PDF model, a mixtures-of-Gaussians (MoG) model [Bishop 1995; Redner and Walker 1984], which exhibits problems with both overfitting and local minima during learning[1]. In addi-

[1]The MoG model is similar to what has been used previously for learning in motion capture. Roughly speaking, both the SHMM [Brand and Hertzmann 2000] and SLDS [Li et al. 2002] reduce to MoGs in synthesis, if we



Gaussian components Log-likelihood

Figure 2: Mixtures-of-Gaussians (MoG). We applied conventional PCA to reduce the baseball pitch data to 2D, then fit an MoG model with EM. Although it assigns highest probability near the data set, the log-likelihood exhibits a number of undesirable artifacts, such as long-and-skinny Gaussians which assign very high probabilities to very small regions and create a very bumpy objective function. In contrast, the likelihood functions shown in Figure 1 are much smoother and more appropriate for the data. In general, we find that 10D PCA is required to yield a reasonable model, and MoG artifacts are much worse in higher dimensions.

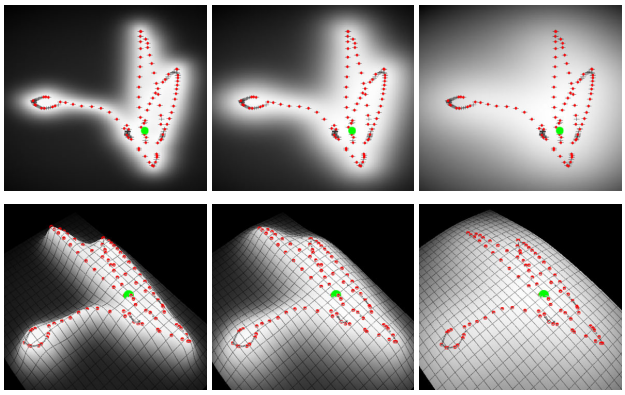


Figure 3: Annealing SGPLVMs. ***Top row:*** The left-most plot shows the "unannealed" original model, trained on the baseball pitch. The plot on the right shows the model retrained with noisy data. The middle plot shows an interpolation between the parameters of the outer models. ***Bottom row:*** The same plots visualized in 3D.

tion, using an MoG requires dimension reduction (such as PCA) as a preprocess, both of which have parameters that need to be tuned. There are principled ways to estimate these parameters, but they are difficult to work with in practice. We have been able to get reasonable results using MoGs on small data-sets, but only with the help of heuristics and manual tweaking of model parameters.

6.1 Synthesis

New poses $\mathbf{q}$ are created by optimizing $L_{IK}(\mathbf{x}, \mathbf{y}(\mathbf{q}))$ with respect to the unknowns $\mathbf{x}$ and $\mathbf{q}$. Examples of learned models are illustrated in Figure 1. There are a number of different scenarios for synthesizing poses; we first describe these cases and how to state them as optimization problems. Optimization techniques are described in Section 6.2.

The general setting for pose synthesis is to optimize $\mathbf{q}$ given some constraints. In order to get a good estimate for $\mathbf{q}$, we also must estimate an associated $\mathbf{x}$. The general problem statement is:

$$\arg\min_{\mathbf{x},\mathbf{q}} L_{IK}(\mathbf{x}, \mathbf{y}(\mathbf{q})) \tag{7}$$

$$\text{s.t. } C(\mathbf{q}) = 0 \tag{8}$$

for some constraints $C(\mathbf{q}) = 0$.

The most common case is when only a set of handle constraints $C(\mathbf{q}) = 0$ are specified; these handle constraints may come from a user in an interactive session, or from a mocap system.

Our system also provides a 2D visualization of the latent space, and allows the user to drag the mouse in this window, in order to view the space of poses in this model. Each point in the window corresponds to a specific value of $\mathbf{x}$; we compute the corresponding pose by maximizing L_{IK} with respect to $\mathbf{q}$. A third case occurs when the user specifies handle constraints and then drags the mouse in the latent space. In this case, $\mathbf{q}$ is optimized during dragging. This provides an alternative way for the user to find a point in the space that works well with the given constraints.

6.1.1 Model smoothing

Our method produces an objective function that is, locally, very smooth, and thus well-suited for local optimization methods. However, distributions over likely poses must necessarily have many local minima, and a gradient-based numerical optimizer can easily get trapped in a poor minima when optimizing L_{IK}. We now describe a new procedure for smoothing an SGPLVM model that can be used in an annealing-like procedure, in which we search in smoother versions of the model before the final optimization. Given training data and a learned SGPLVM, our goal is to create smoothed (or "annealed") versions of this SGPLVM. We have found that the simplest annealing strategy of scaling the individual model parameters (for example, halving the value of β) does not work well, since the scales of the three α, β, and γ parameters are closely intertwined.

Instead, we use the following strategy to produce a smoother model. We first learn a normal (unannealed) SGPLVM as described in Section 5. We then create a noisy version of the training set, by adding zero-mean Gaussian noise to all of the $\{\mathbf{y}_i\}$ values in the active set. We then learn new values α, β, and γ using the same algorithm as before, but while holding $\{\mathbf{x}_i\}$ and $\{w_k\}$ fixed. This gives us new "annealed" parameters α', β', γ'. The variance of the noise added to data determines how smooth the model becomes. Given this annealed model, we can generate a range of models by linear interpolation between the parameters of the normal SGPLVM and the annealed SGPLVM. An example of this range of annealed models is shown in Figure 3.

6.2 Real-time optimization algorithm

Our system optimizes L_{IK} using gradient-based optimization methods; we have experimented with Sequential Quadratic Programming (SQP) and L-BFGS [Nocedal and Wright 1999]. SQP allows the use of hard constraints on the pose. However, hard constraints can only be used for underconstrained IK, otherwise the system quickly becomes infeasible and the solver fails. The more general solution we use is to convert the constraints into soft constraints by adding a term $||C(\mathbf{q})||^2$ to the objective function with a large weight. A more desirable approach would be to enforce hard constraints as much as possible, but convert some constraints to soft constraints when necessary [Yamane and Nakamura 2003].

Because the L_{IK} objective is rarely unimodal, we use an annealing-like scheme to prevent the pose synthesis algorithm from getting stuck in local minima. During the learning phase, we precompute an annealed model as described in the previous section. In our tests, we set the noise variance to .05 for smaller data sets and

view a single frame of a sequence in isolation. The SHMM's entropic prior helps smooth the model, but at the expense of overly-smooth motions.

0.1 for larger data sets. During synthesis, we first run a few steps of optimization using the smoothed model (α', β', γ'), as described in the previous section. We then run additional steps on an intermediate model, with parameters interpolated as $\frac{1}{\sqrt{2}}\alpha + (1-\frac{1}{\sqrt{2}})\alpha'$. The same interpolation is applied to β and γ. We then finish the optimization with respect to the original model (α, β, γ). During interactive editing, there may not be enough time to fully optimize between dragging steps, in which case the optimization is only updated with respect to the smoothest model; in this case, the finer models are only used when dragging stops.

6.3 Style interpolation

We now describe a simple new approach to interpolating between two styles represented by SGPLVMs. Our goal is to generate a new style-specific SGPLVM that interpolates two existing SGPLVMs L_{IK0} and L_{IK1}. Given an interpolation parameter s, the new objective function is:

$$L_s(\mathbf{x}_0, \mathbf{x}_1, \mathbf{y}(\mathbf{q})) = (1-s)L_{IK0}(\mathbf{x}_0, \mathbf{y}(\mathbf{q})) + sL_{IK1}(\mathbf{x}_1, \mathbf{y}(\mathbf{q})) \quad (9)$$

Generating new poses entails optimizing L_s with respect to the pose $\mathbf{q}$ as well a latent variables $\mathbf{x}_0$ and $\mathbf{x}_1$ (one for each of the original styles).

We can place this interpolation scheme in the context of the following novel method for interpolating style-specific PDFs. Given two or more pose styles — represented by PDFs over possible poses — our goal is to produce a new PDF representing a style that is "in between" the input poses. Given two PDFs over poses $p(\mathbf{y}|\theta_0)$ and $p(\mathbf{y}|\theta_1)$, where θ_0 and θ_1 describe the parameters of these styles, and an interpolation parameter s, we form the interpolated style PDF as

$$p_s(\mathbf{y}) \propto \exp((1-s)\ln p(\mathbf{y}|\theta_0) + s\ln p(\mathbf{y}|\theta_1)) \quad (10)$$

New poses are created by maximizing $p_s(\mathbf{y}(\mathbf{q}))$. In the SGPLVM case, we have $\ln p(\mathbf{y}|\theta_0) = -L_{IK0}$ and $\ln p(\mathbf{y}|\theta_0) = -L_{IK1}$. We discuss the motivation for this approach in Appendix C.

7 Applications

In order to explore the effectiveness of the style-based IK, we tested it on a few applications: interactive character posing, trajectory keyframing, realtime motion capture with missing markers, and determining human pose from 2D image correspondences. Examples of all these applications are shown in the accompanying video.

7.1 Interactive character posing

One of the most basic — and powerful — applications of our system is for interactive character posing, in which an animator can interactively define a character pose by moving handle constraints in real-time. In our experience, posing this way is substantially faster and more intuitive than posing without an objective function.

7.2 Trajectory keyframing

We developed a test animation system aimed at rapid-prototyping of character animations. In this system, the animator creates an animation by constraining a small set of points on the character. Each constrained point is controlled by modifying a trajectory curve. The animation is played back in realtime so that the animator can immediately view the effects of path modifications on the resulting motion. Since the animator constrains only a minimal set of points, the rest of the pose for each time frame is automatically synthesized using style-based IK. The user can use different styles for different parts of the animation, by smoothly blending from one style to another. An example of creating a motion by keyframing is shown in Figure 4, using three keyframed markers.

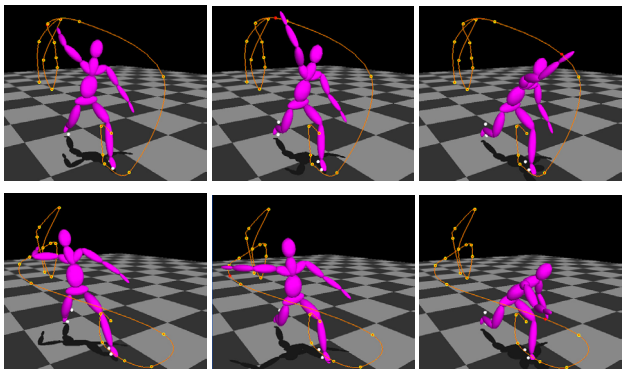


Figure 4: Trajectory keyframing, using a style learned from the baseball pitch data. ***Top row:*** A baseball pitch. ***Bottom row:*** A side-arm pitch. In each case, the feet and one arm were keyframed; no other constraints were used. The side-arm contains poses very different from those in the original data.

7.3 Real-time motion capture with missing markers

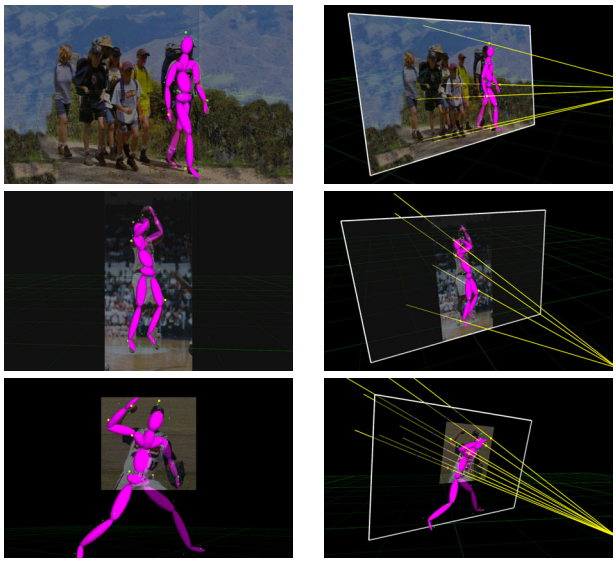
In optical motion capture systems, the tracked markers often disappear due to occlusion, resulting in inaccurate reconstructions and noticeable glitches. Existing joint reconstruction methods quickly fail if several markers go missing, or they are missing for an extended period of time. Furthermore, once the a set of missing markers reappears, it is hard to relabel each one of them so that they correspond to the correct points on the body.

We designed a real-time motion reconstruction system based on style-based IK that fills in missing markers. We learn the style from the initial portion of the motion capture sequence, and use that style to estimate the character pose. In our experiments, this approach can faithfully reconstruct poses even with more than 50% of the markers missing.

We expect that our method could be used to provide a metric for marker matching as well. Of course, the effectiveness of style-based IK degrades if the new motion diverges from the learned style. This could potentially be addressed by incrementally relearning the style as the new pose samples are processed.

7.4 Posing from 2D images

We can also use our IK system to reconstruct the most likely pose from a 2D image of a person. Given a photograph of a person, a user interactively specifies 2D projections (i.e., image coordinates) of a few character handles. For example, the user might specify the location of the hands and feet. Each of these 2D positions establishes a constraint that the selected handle project to the 2D position indicated by the user, or, in other words, that the 3D handle lie on the line containing the camera center and the projected position. The 3D pose is then estimated by minimizing L_{IK} subject to these 2D constraints. With only three or four established correspondences between the 2D image points and character handles, we can reconstruct the most likely pose; with a little additional effort, the pose can be fine-tuned. Several examples are shown in Figure 5. In the baseball example (bottom row of the figure) the system obtains a plausible pose from six projection constraints, but the depth of



Frontal view Side view

Figure 5: 3D posing from a 2D image. Yellow circles in the front view correspond to user-placed 2D constraints; these 2D constraints appear as "line constraints" from a side view.

the right hand does not match the image. This could be fixed by one more constraint, e.g., from another viewpoint or from temporal coherence.

8 Discussion and future work

We have presented an inverse kinematics system based on a learned probability model of human poses. Given a set of arbitrary algebraic constraints, our system can produce the most likely pose satisfying those constraints, in real-time. We demonstrated this system in the context of several applications, and we expect that style-based IK can be used effectively for any problem where it is necessary to restrict the space of valid poses, including problems in computer vision as well as animation. For example, the SGPLVM could be used as a replacement for PCA and for RBFs in example-based animation methods.

Additionally, there are a number of potential applications for games, in which it is necessary that the motions of character both look realistic and satisfy very specific constraints (e.g., catching a ball or reaching a base) in real-time. This would require not only real-time posing, but, potentially, some sort of planning ahead. We are encouraged by the fact that a leading game developer licensed an early version of our system for the purpose of rapid content development.

There are some limitations in our system that could be addressed in future work. For example, our system does not model dynamics, and does not take into account the constraints that produced the original motion capture. It would also be interesting to incorporate style-based IK more closely into an animation pipeline. For example, our approach may be thought of as automating the process of "rigging," i.e., determining high-level controls for a character. In a production environment, a rigging designer might want to design some of the character controls in a specific way, while using an automatic procedure for other controls. It would also be useful to have a more principled method for balancing hard and soft constraints in real-time, perhaps similar to [Yamane and Nakamura 2003], because too many hard constraints can prevent the problem from having any feasible solution.

There are many possible improvements to the SGPLVM learning algorithm, such as experimenting with other kernels, or selecting kernels automatically based on the data set. Additionally, the current optimization algorithm employs some heuristics for convenience and speed; it would be desirable to have a more principled and efficient method for optimization. We find that the annealing heuristic for real-time synthesis requires some tuning, and it would be desirable to find a better procedure for real-time optimization.

Acknowledgements

Many thanks to Neil Lawrence for detailed discussions and for placing his source code online. We are indebted to Colin Zheng for creating the 2D posing application, and to Jia-Chu Wu for for last-minute image and video production. David Hsu and Eugene Hsu implemented the first prototypes of this system. This work was supported in part by UW Animation Research Labs, NSF grants EIA-0121326, CCR-0092970, IIS-0113007, CCR-0098005, an NSERC Discovery Grant, the Connaught fund, Alfred P. Sloan Fellowship, Electronic Arts, Sony, and Microsoft Research.

A Background on Gaussian Processes

In this section, we briefly describe the likelihood function used in this paper. Gaussian Processes (GPs) for learning were originally developed in the context of classification and regression problems [Neal 1996; O'Hagan 1978; Williams and Rasmussen 1996]. For detailed background on Gaussian Processes, see [MacKay 1998].

Scaled Gaussian Processes. The general setting for regression is as follows: we are given a collection of training pairs $\{\mathbf{x}_i, \mathbf{y}_i\}$, where each element $\mathbf{x}_i$ and $\mathbf{y}_i$ is a vector, and we wish to learn a mapping $\mathbf{y} = f(\mathbf{x})$. Typically, this is done by least-squared fitting of a parametric function, such as a B-spline basis or a neural network. This fitting procedure is sensitive to a number of important choices, e.g., the number of basis functions and smoothness/regularization assumptions; if these choices are not made carefully, over- or under-fitting results. However, from a Bayesian point of view, we should never estimate a specific function f during regression. Instead, we should marginalize over all possible choices of f when computing new points — in doing so, we can avoid overfitting and underfitting, and can additionally learn the smoothness parameters and noise parameters. Remarkably, it turns out that, for a wide variety of types of function f (including polynomials, splines, single-hidden-layer neural networks, and Gaussian RBFs), marginalization over all possible values of f yields a Gaussian Process model of the data. For a GP model of a single output dimension k, the likelihood of the outputs given the inputs is:

$$p(\{\mathbf{y}_{i,k}\}|\{\mathbf{x}_i\}, \alpha, \beta, \gamma) = \frac{1}{\sqrt{(2\pi)^N |\mathbf{K}|}} \exp(-\frac{1}{2}\mathbf{Y}_k^T \mathbf{K}^{-1} \mathbf{Y}_k) \quad (11)$$

using the variables defined in Section 5.

In this paper, we generalize GP models to account for different variances in the output dimensions, by introducing scaling parameters w_k for each output dimension. This is equivalent to defining a separate kernel function $k(\mathbf{x}, \mathbf{x}')/w_k^2$ for each output dimension[2]; plugging this into the GP likelihood for dimension k yields:

$$p(\{\mathbf{y}_{i,k}\}|\{\mathbf{x}_i\}, \alpha, \beta, \gamma, w_k) = \frac{w_k^N}{\sqrt{(2\pi)^N |\mathbf{K}|}} \exp(-\frac{1}{2} w_k^2 \mathbf{Y}_k^T \mathbf{K}^{-1} \mathbf{Y}_k) \quad (12)$$

[2] Alternatively, we can derive this model as a Warped GP [Snelson et al. 2004], in which the warping function rescales the features as $w_k \mathbf{Y}_k$

The complete joint likelihood of all data dimensions is $p(\{\mathbf{y}_i\}|\{\mathbf{x}_i\},\alpha,\beta,\gamma,\{w_k\}) = \prod_k p(\{\mathbf{y}_{i,k}\}|\{\mathbf{x}_i\},\alpha,\beta,\gamma,w_k)$.

SGPLVMs. The Scaled Gaussian Process Latent Variable Model (SGPLVM) is a general technique for learning PDFs, based on recent work Lawrence [2004]. Given a set of data points $\{\mathbf{y}_i\}$, we model the likelihood of these points with a scaled GP as above, in which the corresponding values $\{\mathbf{x}_i\}$ are initially unknown — we must now learn the $\mathbf{x}_i$ as well as the model parameters. We also place priors on the unknowns: $p(\mathbf{x}) = \mathcal{N}(0;\mathbf{I})$, $p(\alpha,\beta,\gamma) \propto \alpha^{-1}\beta^{-1}\gamma^{-1}$.

In order to learn the SGPLVM from training data $\{\mathbf{y}_i\}$, we need to maximize the posterior $p(\{\mathbf{x}_i\},\alpha,\beta,\gamma,\{w_k\}|\{\mathbf{y}_i\})$. This is equivalent to minimizing the negative log-posterior

$$\begin{aligned} L_{GP} &= -\ln p(\{\mathbf{x}_i\},\alpha,\beta,\gamma,\{w_k\}|\{\mathbf{y}_i\}) \qquad (13)\\ &= -\ln p(\{\mathbf{y}_i\}|\{\mathbf{x}_i\},\alpha,\beta,\gamma,\{w_k\})(\prod_i p(\mathbf{x}_i))p(\alpha,\beta,\gamma)\\ &= \frac{D}{2}\ln|\mathbf{K}| + \frac{1}{2}\sum_k w_k^2 \mathbf{Y}_k^T \mathbf{K}^{-1}\mathbf{Y}_k + \frac{1}{2}\sum_i ||\mathbf{x}_i||^2 + \ln\frac{\alpha\beta\gamma}{\prod_k w_k^N} \end{aligned}$$

with respect to the unknowns (constant terms have been dropped from these expressions).

One way to interpret this objective function as follows. Suppose we ignore the priors $p(\mathbf{x})$ and $p(\alpha,\beta,\gamma)$, and just optimize L_{GP} with respect to an $\mathbf{x}_i$ value. The optima should occur when $\frac{\partial L_{GP}}{\partial \mathbf{x}_i} = \frac{\partial L_{GP}}{\partial \mathbf{K}}\frac{\partial \mathbf{K}}{\partial \mathbf{x}_i} = 0$. One condition for this to occur is $\frac{\partial L_{GP}}{\partial \mathbf{K}} = 0$; similarly, this would make L_{GP} optimal with respect to all $\{\mathbf{x}_i\}$ values and the α, β, and γ parameters. If we solve $\frac{\partial L_{GP}}{\partial \mathbf{K}} = 0$ (see Equation 15), we obtain a system of equations of the form $\mathbf{K} = \mathbf{W}\mathbf{Y}\mathbf{Y}^T\mathbf{W}^T/D$, or

$$k(\mathbf{x}_i,\mathbf{x}_j) = (\mathbf{W}(\mathbf{y}_i-\mu))^T(\mathbf{W}(\mathbf{y}_j-\mu))/D \qquad (14)$$

The right-hand side of this expression will be large when the two poses are very similar, and negative when they are very different. This means that we try to arrange the $\mathbf{x}$'s so that $\mathbf{x}_i$ and $\mathbf{x}_j$ are nearby if and only if $\mathbf{y}_i$ and $\mathbf{y}_j$ are similar. More generally, the kernel matrix should match the covariance matrix of the original data rescaled by $\mathbf{W}/\sqrt{D}$. The prior terms $p(\mathbf{x})$ and $p(\alpha,\beta,\gamma)$ help prevent overfitting on small training sets.

Once the parameters have been learned, we have a general-purpose probability distribution for new poses. In order to define this probability, we augment the data with a new pose $(\mathbf{x},\mathbf{y})$, in which one or both of $(\mathbf{x},\mathbf{y})$ are unknown. Adding this new pose to L_{GP}, rearranging terms, and dropping constants yields the log-posterior L_{IK} (Equation 3).

B Learning algorithm

We tested two different algorithms for optimizing L_{GP}. The first directly optimizes the objective function, and then selects an active set (i.e., a reduced set of example poses) from the training data. The second is a heuristic described below. Based on preliminary tests, it appears that there are a few tradeoffs between the two algorithms. The heuristic algorithm is much faster, but more tied to the initialization for small data sets, often producing $\mathbf{x}$ values that are very close to the PCA initialization. The full optimization algorithm produces better arrangements of the latent space $\mathbf{x}$ values — especially for larger data sets — but may require higher latent dimensionality (3D instead of 2D in our tests). However, because the full optimization optimizes all points, it can get by with less active set points, making it more efficient at run-time. Nonetheless, both algorithms work well, and we used the heuristic algorithm for all examples shown in this paper and the video.

Active set selection. We first outline the greedy algorithm for selecting the active set, given a learned model. The active set initially contains one training pose. Then the algorithm repeatedly determines which of the points not in the active set has the highest prediction variance $\sigma^2(\mathbf{x})$ (Equation 5). This point is added to the active set, and the algorithm repeats until there are M points in the active set (where M is a limit predetermined by a user). For efficiency, the variances are computed incrementally as described by Lawrence et al. [2003].

Heuristic optimization algorithm. For all examples in this paper, we used the following procedure for optimizing L_{GP}, based on the one proposed by Lawrence [2004], but modified[3] to learn $\{w_k\}$. The algorithm alternates between updating the active set, and the following steps: First, the algorithm optimizes the model parameters, α, β, and γ by numerical optimization of L_{GP} (Equation 2); however, L_{GP} is modified so that only the active set are included in L_{GP}. Next, the algorithm optimizes the latent variables $\mathbf{x}_i$ for points that are not included in the active set; this is done by numerical optimization of L_{IK} (Equation 3). Finally, the scaling is updated by closed-form optimization of L_{GP} with respect to $\{w_k\}$. Numerical optimization is performed using the Scaled Conjugate Gradients algorithm, although other search algorithms could also be used. After each of these steps, the active set is recomputed.

The algorithm may be summarized as follows. See [Lawrence 2004] for further details.

function LEARNSGPLVM($\{\mathbf{y}_i\}$)
  initialize $\alpha \leftarrow 1, \beta \leftarrow 1, \gamma \leftarrow 1, \{w_k\} \leftarrow \{1\}$
  initialize $\{\mathbf{x}\}$ with conventional PCA applied to $\{\mathbf{y}_i\}$
  for $T = 1$ **to** *NumSteps* **do**:
    Select new active set
    Minimize L_{GP} (over the active set) with respect to α, β, γ
    Select new active set
    for each point i not in the active set **do**
      Minimize $L_{IK}(\mathbf{x}_i,\mathbf{y}_i)$ with respect to $\mathbf{x}_i$.
    end for
    Select new active set
    for each data dimension d **do**
      $w_k \leftarrow \sqrt{M/(\overline{\mathbf{Y}}_k^T\overline{\mathbf{K}}^{-1}\overline{\mathbf{Y}}_k)}$
    end for
  end for
  return $\{\mathbf{x}_i\},\alpha,\beta,\gamma,\{w_k\}$

Parameters. The active set size and latent dimensionality trade-off run-time speed versus quality. We typically used 50 active set points for small data sets and 100 for large data sets. Using a long walking sequence (of about 500 frames) as training, 100 active set points and a 3-dimensional latent space gave 23 frames-per-second synthesis on a 2.8 GHz Pentium 4; increasing the active set size slows performance without noticably improving quality. We found that, in all cases, a 3D latent space gave as good or better quality than a 2D latent space. We use higher dimensionality when multiple distinct motions were included in the training set.

C Style interpolation

Although we have no formal justification for our interpolation method in Section 6.3 (e.g., as maximizing a known likelihood function), we can motivate it as follows. In general, there is no reason to believe the interpolation of two objective functions gives a reasonable interpolation of their styles. For example, suppose we

[3] We adapted the source code available from `http://www.dcs.shef.ac.uk/~neil/gplvm/`

represent styles as Gaussian distributions $p(\mathbf{y}|\theta_0) = \mathcal{N}(\mathbf{y}|\mu_0;\sigma^2)$ and $p(\mathbf{y}|\theta_1) = \mathcal{N}(\mathbf{y}|\mu_1;\sigma^2)$ where μ_0 and μ_1 are the means of the Gaussians, and σ^2 is the variance. If we simply interpolate these PDFs, i.e., $p_s(\mathbf{y}) = -(1-s)\exp(-||\mathbf{y}-\mu_0||^2/\sigma^2) - s\exp(-||\mathbf{y}-\mu_1||^2/2\sigma^2)$, then the interpolated function is *not* Gaussian — for most values of s, it has two minima (near μ_0 and μ_1). Howver, using the log-space interpolation scheme, we get an intuitive result: the interpolated style $p_s(\mathbf{y})$ is also a Gaussian, with mean $(1-s)\mu_0 + s\mu_1$, and variance σ^2. In other words, the mean linearly interpolates the means of the input Gaussians, and the variance is unchanged. A similarly-intuitive interpolation results when the Gaussians have different covariances. While analyzing the SGPLVM case is more difficult, we find that in practice this scheme works quite well. Moreover, it should be straightforward to interpolate any two likelihood models (e.g., interpolate an SGPLVM with an MoG), which would be difficult to achieve otherwise.

D Gradients

The gradients of L_{IK} and L_{GP} may be computed with the help of the following derivatives, along with the chain rule:

$$\frac{\partial L_{GP}}{\partial \mathbf{K}} = \mathbf{K}^{-1}\mathbf{W}\mathbf{Y}\mathbf{Y}^T\mathbf{W}^T\mathbf{K}^{-1} - D\mathbf{K}^{-1} \quad (15)$$

$$\frac{\partial L_{IK}}{\partial \mathbf{y}} = (\mathbf{W}^T\mathbf{W}(\mathbf{y}-\mathbf{f}(\mathbf{x})))/\sigma^2(\mathbf{x}) \quad (16)$$

$$\frac{\partial L_{IK}}{\partial \mathbf{x}} = -\frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}}^T \mathbf{W}^T\mathbf{W}(\mathbf{y}-\mathbf{f}(\mathbf{x}))/\sigma^2(\mathbf{x}) + \frac{\partial \sigma^2(\mathbf{x})}{\partial \mathbf{x}}\left[D - \frac{||\mathbf{W}(\mathbf{y}-\mathbf{f}(\mathbf{x}))||^2}{\sigma^2(\mathbf{x})}\right]/(2\sigma^2(\mathbf{x})) + \mathbf{x} \quad (17)$$

$$\frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} = \overline{\mathbf{Y}}^T\overline{\mathbf{K}}^{-1}\frac{\partial \mathbf{k}(\mathbf{x})}{\partial \mathbf{x}} \quad (18)$$

$$\frac{\partial \sigma^2(\mathbf{x})}{\partial \mathbf{x}} = -2\mathbf{k}(\mathbf{x})^T\overline{\mathbf{K}}^{-1}\frac{\partial \mathbf{k}(\mathbf{x})}{\partial \mathbf{x}} \quad (19)$$

$$\frac{\partial k(\mathbf{x},\mathbf{x}')}{\partial \mathbf{x}} = -\gamma(\mathbf{x}-\mathbf{x}')k(\mathbf{x},\mathbf{x}') \quad (20)$$

$$\frac{\partial k(\mathbf{x},\mathbf{x}')}{\partial \alpha} = \exp\left(-\frac{\gamma}{2}||\mathbf{x}-\mathbf{x}'||^2\right) \quad (21)$$

$$\frac{\partial k(\mathbf{x},\mathbf{x}')}{\partial \beta} = \delta_{\mathbf{x},\mathbf{x}'} \quad (22)$$

$$\frac{\partial k(\mathbf{x},\mathbf{x}')}{\partial \gamma} = -\frac{1}{2}||\mathbf{x}-\mathbf{x}'||^2 k(\mathbf{x},\mathbf{x}') \quad (23)$$

where $\mathbf{Y} = [\mathbf{y}_1 - \mu, ..., \mathbf{y}_N - \mu]^T$ is a matrix containing the mean-subtracted training data.

References

ARIKAN, O., AND FORSYTH, D. A. 2002. Synthesizing Constrained Motions from Examples. *ACM Transactions on Graphics 21*, 3 (July), 483–490. (Proc. of ACM SIGGRAPH 2002).

ARIKAN, O., FORSYTH, D. A., AND O'BRIEN, J. F. 2003. Motion Synthesis From Annotations. *ACM Transactions on Graphics 22*, 3 (July), 402–408. (Proc. SIGGRAPH 2003).

BISHOP, C. M. 1995. *Neural Networks for Pattern Recognition*. Oxford University Press.

BODENHEIMER, B., ROSE, C., ROSENTHAL, S., AND PELLA, J. 1997. The process of motion capture – dealing with the data. In *Computer Animation and Simulation '97*, Springer-Verlag Wien New York, D. Thalmann and M. van de Panne, Eds., Eurographics, 3–18.

BRAND, M., AND HERTZMANN, A. 2000. Style machines. *Proceedings of SIGGRAPH 2000* (July), 183–192.

BRAND, M. 1999. Shadow Puppetry. In *Proc. ICCV*, vol. 2, 1237–1244.

BRUDERLIN, A., AND WILLIAMS, L. 1995. Motion signal processing. *Proceedings of SIGGRAPH 95* (Aug.), 97–104.

ELKOURA, G., AND SINGH, K. 2003. Handrix: Animating the Human Hand. *Proc. SCA*, 110–119.

GIRARD, M., AND MACIEJEWSKI, A. A. 1985. Computational Modeling for the Computer Animation of Legged Figures. In *Computer Graphics (Proc. of SIGGRAPH 85)*, vol. 19, 263–270.

GRASSIA, F. S. 2000. *Believable Automatically Synthesized Motion by Knowledge-Enhanced Motion Transformation*. PhD thesis, CMU Computer Science.

GRAUMAN, K., SHAKHNAROVICH, G., AND DARRELL, T. 2003. Inferring 3D Structure with a Statistical Image-Based Shape Model. In *Proc. ICCV*, 641–648.

GULLAPALLI, V., GELFAND, J. J., AND LANE, S. H. 1996. Synergy-based learning of hybrid position/force control for redundant manipulators. In *Proceedings of IEEE Robotics and Automation Conference*, 3526–3531.

HOWE, N. R., LEVENTON, M. E., AND FREEMAN, W. T. 2000. Bayesian Reconstructions of 3D Human Motion from Single-Camera Video. In *Proc. NIPS 12*, 820–826.

KOVAR, L., AND GLEICHER, M. 2004. Automated Extraction and Parameterization of Motions in Large Data Sets. *ACM Transactions on Graphics 23*, 3 (Aug.). In these proceedings.

KOVAR, L., GLEICHER, M., AND PIGHIN, F. 2002. Motion Graphs. *ACM Transactions on Graphics 21*, 3 (July), 473–482. (Proc. SIGGRAPH 2002).

LAWRENCE, N., SEEGER, M., AND HERBRICH, R. 2003. Fast Sparse Gaussian Process Methods: The Informative Vector Machine. *Proc. NIPS 15*, 609–616.

LAWRENCE, N. D. 2004. Gaussian Process Latent Variable Models for Visualisation of High Dimensional Data. *Proc. NIPS 16*.

LEE, J., CHAI, J., REITSMA, P. S. A., HODGINS, J. K., AND POLLARD, N. S. 2002. Interactive Control of Avatars Animated With Human Motion Data. *ACM Transactions on Graphics 21*, 3 (July), 491–500. (Proc. SIGGRAPH 2002).

LI, Y., WANG, T., AND SHUM, H.-Y. 2002. Motion Texture: A Two-Level Statistical Model for Character Motion Synthesis. *ACM Transactions on Graphics 21*, 3 (July), 465–472. (Proc. SIGGRAPH 2002).

MACKAY, D. J. C. 1998. Introduction to Gaussian processes. In *Neural Networks and Machine Learning*, C. M. Bishop, Ed., NATO ASI Series. Kluwer Academic Press, 133–166.

NEAL, R. M. 1996. *Bayesian Learning for Neural Networks*. Lecture Notes in Statistics No. 118. Springer-Verlag.

NOCEDAL, J., AND WRIGHT, S. J. 1999. *Numerical Optimization*. Springer-Verlag.

O'HAGAN, A. 1978. Curve Fitting and Optimal Design for Prediction. *J. of the Royal Statistical Society, ser. B 40*, 1–42.

POPOVIĆ, Z., AND WITKIN, A. P. 1999. Physically Based Motion Transformation. *Proceedings of SIGGRAPH 99* (Aug.), 11–20.

PULLEN, K., AND BREGLER, C. 2002. Motion Capture Assisted Animation: Texturing and Synthesis. *ACM Transactions on Graphics 21*, 3 (July), 501–508. (Proc. of ACM SIGGRAPH 2002).

RAMANAN, D., AND FORSYTH, D. A. 2004. Automatic annotation of everyday movements. In *Proc. NIPS 16*.

REDNER, R. A., AND WALKER, H. F. 1984. Mixture Densities, Maximum Likelihood and the EM Algorithm. *SIAM Review 26*, 2 (Apr.), 195–202.

ROSALES, R., AND SCLAROFF, S. 2002. Learning Body Pose Via Specialized Maps. In *Proc. NIPS 14*, 1263–1270.

ROSE, C., COHEN, M. F., AND BODENHEIMER, B. 1998. Verbs and Adverbs: Multidimensional Motion Interpolation. *IEEE Computer Graphics & Applications 18*, 5, 32–40.

ROSE III, C. F., SLOAN, P.-P. J., AND COHEN, M. F. 2001. Artist-Directed Inverse-Kinematics Using Radial Basis Function Interpolation. *Computer Graphics Forum 20*, 3, 239–250.

SIDENBLADH, H., BLACK, M. J., AND SIGAL, L. 2002. Implicit probabilistic models of human motion for synthesis and tracking. In *Proc. ECCV, LNCS 2353*, vol. 1, 784–800.

SNELSON, E., RASMUSSEN, C. E., AND GHAHRAMANI, Z. 2004. Warped Gaussian Processes. *Proc. NIPS 16*.

TAYLOR, C. J. 2000. Reconstruction of Articulated Objects from Point Correspondences in a Single Image. In *Proc. CVPR*, 677–684.

WELMAN, C. 1993. *Inverse Kinematics and Geometric Constraints for Articulated Figure Manipulation*. PhD thesis, Simon Fraser University.

WILEY, D. J., AND HAHN, J. K. 1997. Interpolation Synthesis of Articulated Figure Motion. *IEEE Computer Graphics & Applications 17*, 6 (Nov.), 39–45.

WILLIAMS, C. K. I., AND RASMUSSEN, C. E. 1996. Gaussian Processes for Regression. *Proc. NIPS 8*, 514–520.

WILSON, A. D., AND BOBICK, A. F. 1999. Parametric Hidden Markov Models for Gesture Recognition. *IEEE Trans. PAMI 21*, 9 (Sept.), 884–900.

WITKIN, A., AND POPOVIĆ, Z. 1995. Motion Warping. *Proceedings of SIGGRAPH 95* (Aug.), 105–108.

YAMANE, K., AND NAKAMURA, Y. 2003. Natural motion animation through constraining and deconstraining at will. *IEEE Transactions on Visualization and Computer Graphics 9*, 3 (July), 352–360.

ZHAO, L., AND BADLER, N. 1998. Gesticulation Behaviors for Virtual Humans. In *Pacific Graphics '98*, 161–168.