Continuous Character Control with Low-Dimensional Embeddings

Sergey Levine[1] Jack M. Wang[1] Alexis Haraux[1] Zoran Popović[2] Vladlen Koltun[1]

[1]Stanford University [2] University of Washington

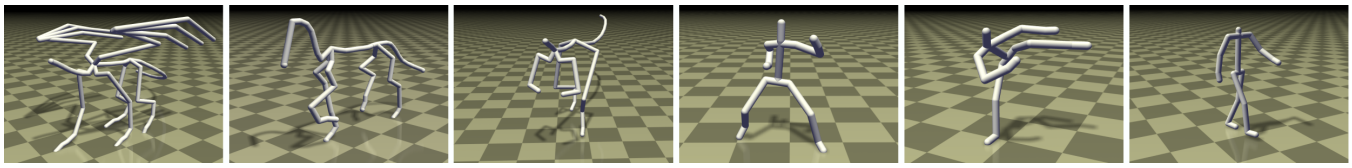

Figure 1: *Character controllers created using our approach: animals, karate punching and kicking, and directional walking.*

Abstract

Interactive, task-guided character controllers must be agile and responsive to user input, while retaining the flexibility to be readily authored and modified by the designer. Central to a method's ease of use is its capacity to synthesize character motion for novel situations without requiring excessive data or programming effort. In this work, we present a technique that animates characters performing user-specified tasks by using a probabilistic motion model, which is trained on a small number of artist-provided animation clips. The method uses a low-dimensional space learned from the example motions to continuously control the character's pose to accomplish the desired task. By controlling the character through a reduced space, our method can discover new transitions, tractably precompute a control policy, and avoid low quality poses.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation;

Keywords: character animation, data-driven animation, nonlinear dimensionality reduction, Gaussian processes, optimal control

Links: DL PDF

1 Introduction

Humans can learn new motion skills from just a few demonstrations: when a student observes an instructor perform a karate punch, he can emulate its style and repurpose it for striking in different directions. Such generalization is a challenge for character animation techniques, which often require the user to "spell out" the desired behavior with a comprehensive set of motions. Recent years have seen significant advances in example-based kinematic character control, but such methods derive their agility from the ability to efficiently navigate large example datasets. Constructing such large datasets requires considerable time and effort, especially for artist-animated characters or difficult acrobatic motions. While stock motion libraries can be used for animating generic characters, many applications demand unique motion styles and require their own datasets.

In this paper, we propose a new approach to interactive character animation that constructs agile, continuous kinematic controllers from a small number of example motion clips. Unlike previous methods, our approach does not require the user to provide an extensive set of motions that are rearranged to accomplish the desired task, but instead generalizes a few example clips into a continuous space of stylistically consistent motions. By navigating this space in real time, our method produces controllers that accomplish user-specified tasks with motions that resemble the provided clips. Since the learned space is continuous, the controller can create a continuously varying stream of motion and respond instantaneously to changes in user input.

To learn low-dimensional motion spaces, we use a novel variant of the Gaussian process latent variable model (GPLVM). Standard GPLVMs often learn embeddings that lack the dense connections necessary for generating transitions and variations, making them unreliable for control tasks. We augment the GPLVM with a novel connectivity prior that ensures that the embedding incorporates rich interactions between the example motions, allowing the controller to discover new transitions through the motion space that are not present in the examples. Our experiments demonstrate that such a prior is essential for creating agile, responsive characters.

Controlling the entire high-dimensional pose of the character with conventional optimal control methods is generally intractable, but we can exploit the compact pose representation induced by the learned low-dimensional space. Our controller interactively navigates this space using a precomputed policy, and the corresponding full body poses form a complete animation sequence that fulfills the desired task. In this way, our method is able to avoid the curse of dimensionality while preserving the ability to continuously vary the character's pose in response to user input.

At runtime, we use the precomputed policy to efficiently synthesize interactive motions. Our model also allows common runtime animation transformation operations, such as inverse kinematics and foot skate cleanup, to be interpreted as probabilistic inference and applied automatically without fine-tuning parameters, blending intervals, or joint weights. This allows the character to react naturally to the environment while fulfilling the user-specified task.

The main contribution of this work is a new approach for continuous kinematic control of interactive characters that uses a low-dimensional space of motions to generalize the example clips. This enables agile, continuous controllers to be built from small datasets, including artist-created motion clips, without requiring numerous examples of transitions and variations. The approach uses a novel

variant of the Gaussian process latent variable model that ensures that the learned space contains rich connections between the example clips. Such connections are critical for finding new transitions and for navigating the low-dimensional space in an agile manner. To exploit the discovered transitions, we augment a nonparametric dynamic programming algorithm with an adaptive refinement technique that seeks out new points that correspond to useful poses in the learned space. We evaluate our approach on a variety of characters and motion skills.

2 Related Work

A number of methods for generalizing from example data for interactive, real-time character animation have focused on interpolating motions along user-specified parameters. Such methods include the pioneering verbs and adverbs system [Rose et al. 1998], as well as techniques that employ dynamic models [Hsu et al. 2005] and statistical interpolation schemes [Mukai and Kuriyama 2005]. Unlike these methods, our approach is unsupervised and does not require task-specific interpolation parameters to be specified by the user.

Prior unsupervised methods created motion variations using graphical models [Lau et al. 2009] and used probabilistic techniques and dimensionality reduction to generate new motions that satisfy user constraints or edits [Grochow et al. 2004; Shin and Lee 2006; Chai and Hodgins 2007; Urtasun et al. 2008; Min et al. 2009; Ikemoto et al. 2009; Wei et al. 2011]. Since our goal is to animate interactive characters using optimal control, we are interested in models that represent motion with low-dimensional spaces, where optimal control methods are tractable. The Gaussian process latent variable model (GPLVM) is a nonlinear generalization of principal component analysis that learns such a space [Lawrence 2005]. Prior methods have applied the GPLVM to motion data [Grochow et al. 2004; Urtasun et al. 2005; Wang et al. 2008; Ye and Liu 2010] for tasks such as human tracking in video and constrained motion synthesis. However, no prior method has used the model for interactive, task-driven control, and prior applications have largely been limited to modeling homogeneous datasets such as forward walking. When motions from different tasks are used to train a GPLVM, they are often pushed far apart in the latent space. Responsive control requires a model that can construct enough transitions to link all usable parts of the data. Prior efforts to address this limitation constrain the latent space topology based on the dataset [Wang et al. 2007] or explicitly encourage similar poses to be embedded near each other [Lawrence and Quiñonero Candela 2006; Urtasun et al. 2008]. However, significant human intervention is often required to adapt such models for a specific dataset, and no prior method explicitly ensures good latent space connectivity. We address this issue with a novel connectivity prior that ensures that paths exist in the latent space to connect all example clips. This enables our model to handle even heterogeneous datasets that include kicks, punches, and sidesteps.

Full body kinematic control is a challenging problem, due to the high dimensionality of the character's configuration and the difficulty of avoiding low quality motions. High-quality kinematic controllers have been created for tasks such as boxing [Lee and Lee 2006] and locomotion [McCann and Pollard 2007; Treuille et al. 2007; Lo and Zwicker 2008; Lee et al. 2009], using a discrete representation of motion data called a motion graph, which uses a graph structure to describe how clips from an example library can be reordered into new motions [Arikan and Forsyth 2002; Kovar et al. 2002; Lee et al. 2002]. However, while graphs are well suited for representing large datasets, smaller datasets that lack extensive transitions and variations may be inadequate to induce an expressive discrete graph. Our dimensionality reduction technique, on the other hand, is specifically designed for generalizing from small

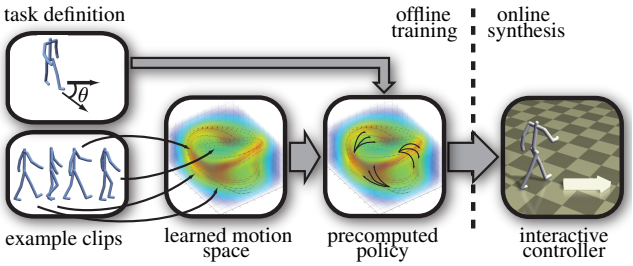

Figure 2: *In the training stage, our method first uses the example clips to learn a low-dimensional space, each point of which corresponds to a pose. Dynamic programming is then used to precompute a policy for navigating this space in order to fulfill a user-specified task. The policy can be used to animate an interactive character that continuously reacts to user input.*

datasets. Furthermore, our method controls the character continuously, and produces a continuous variety of poses. This allows it to respond immediately to user input and changes in the task.

Lee et al. [2010] proposed a kinematic continuous control scheme that operates directly on full body poses. However, such a method requires enough data to adequately fill out the high dimensional space of poses. By representing the data with a reduced space, our method avoids the need to control full body poses directly and can operate on much smaller datasets. Other prior work has proposed to improve the agility of graph-based methods by anticipating user inputs [McCann and Pollard 2007] and by increasing the number of graph edges with interpolation or physics-based optimization [Shin and Oh 2006; Zhao and Safonova 2009; Ren et al. 2010]. Methods that increase the number of edges in motion graphs can work with datasets that contain fewer example transitions, since the synthesized edges can discover transitions that are not provided. However, such approaches are still limited to making decisions at discrete points in time, and only at graph nodes, which always correspond to frames in the example clips. By controlling the character continuously, our method can be responsive to a continuous stream of user inputs without needing to choose discrete decision points.

While prior automated methods tend to work best with larger datasets, the dominant animation method in modern games and virtual worlds – sometimes called a "move tree" – uses small, custom-authored datasets. In this approach, a hand-authored state machine selects and blends carefully aligned motion clips. This requires significant programming and animation effort, but results in high quality controllers that use small amounts of very carefully authored data [Johansen 2009]. Unlike these approaches, our method is automatic and does not rely on the motions exhibiting special structure. This makes it possible to apply our approach to "raw" motion capture, and avoids the need for extensive programming effort. Our tasks are specified with intuitive reward functions that indicate success or failure, and optimal control is used to maximize the rewards.

3 Overview

The proposed character animation method, outlined in Figure 2, accepts as input a set of example clips representing the types and styles of motions that should be used, as well as a specification of the task that the controller should accomplish. The task is specified in terms of a set of task parameters, such as the desired walking direction or punching target, as well as a reward function that defines how desirable each pose and parameter value is. For example, the reward in a walking task might be high for moving in the desired direction. In a karate punching task, poses that place the hand at the

target position with a large velocity might be rewarded.

The kinematic character controller is constructed automatically in two stages. First, we use a modified Gaussian process latent variable model (GPLVM) to learn a low-dimensional space that describes the continuous range of motions that are stylistically consistent with the user examples. While the GPLVM is a powerful nonlinear dimensionality reduction method, it is prone to embed different motions far apart in the latent space, making it unsuitable for control. We use a novel connectivity prior to ensure that the learned space has rich connections between example motions, which enables the learned space to be navigated in an agile manner. Dense connectivity has long been recognized as a desirable feature in motion graphs [Kovar et al. 2002], and our proposed GPLVM connectivity prior provides an effective way to enforce analogously dense connectivity within a continuous nonlinear embedding, without the constraint of a discrete graph structure.

In the second stage, nonparametric approximate dynamic programming is used to precompute a near-optimal policy that constructs a continuous sequence of character poses in real time to maximize the user-specified reward function and fulfill the user's task. Although controlling the entire high-dimensional pose of the character with conventional optimal control methods is intractable due to the curse of dimensionality [Bertsekas 2001], we can efficiently precompute a near-optimal policy to navigate the learned low-dimensional space, which has a fixed dimensionality independent from the size of each pose. Our policy is computed on the low-dimensional space, and the learned mapping from this space to full body poses is used to reconstruct the animation sequence. Since the learned space describes a continuous range of poses, we preserve the ability to continuously modify the character's pose in response to changes in the task parameters (such as user input), while avoiding the curse of dimensionality.

Once the policy is learned, it can be used to synthesize interactive motions in real time that react rapidly to user input and accomplish the desired task. The probabilistic model further provides us with a principled probabilistic interpretation of common animation transformation operations, including inverse kinematics and foot skate cleanup. These operations can be framed as constrained maximum a posteriori (MAP) inference in the probabilistic model, and can be used to automatically modify the synthesized animation to smoothly satisfy additional constraints, such as hand or foot placement and collisions with the environment.

4 Motion Model

In order to learn control policies that navigate the space of character motions, we use a generative model that provides us with a compact, low-dimensional representation of high quality motions that resemble the provided examples. By using a probabilistic model, we can also evaluate the quality of the new motions and avoid low quality poses. Our model is based on the Gaussian process latent variable model (GPLVM) [Lawrence 2005], which generates poses as nonlinear functions over a learned low-dimensional latent space. We employ a novel connectivity prior to ensure that the latent space is well connected and suitable for control. The combination of dimensionality reduction and connectivity allows agile control policies to be learned even from small amounts of data.

4.1 Likelihood and Dynamics Terms

We represent the mapping from low-dimensional latent points $\mathbf{x}$ to $d_{\mathbf{Y}}$-dimensional pose vectors $\mathbf{y}$ as Gaussian processes (GPs), with each channel of $\mathbf{y}$ transformed by a diagonal scaling matrix $\mathbf{W}$ to account for the different amounts of variance in different joints [Grochow et al. 2004]. Under the GP, the log likelihood of the data $\ln p(\mathbf{Y}|\mathbf{X}, \bar{\alpha}, \mathbf{W})$ is proportional to

$$\mathcal{L}_{\mathbf{Y}} = -\frac{1}{2}\mathrm{tr}\left(\mathbf{K}_{\mathbf{Y}}^{-1}\mathbf{Y}\mathbf{W}^2\mathbf{Y}^{\mathrm{T}}\right) - \frac{d_{\mathbf{Y}}}{2}\ln|\mathbf{K}_{\mathbf{Y}}| + N\ln|\mathbf{W}|, \quad (1)$$

where $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_N]^{\mathrm{T}}$ and $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]^{\mathrm{T}}$. The matrix $\mathbf{K}_{\mathbf{Y}}$ is the covariance of the GP, with each entry given by the GP's kernel function. A common choice is the radial basis function (RBF) kernel, with learned hyperparameters $\bar{\alpha} = [\alpha_1, \alpha_2, \alpha_3]$:

$$k_{\mathrm{rbf}}(\mathbf{x}_i, \mathbf{x}_j; \bar{\alpha}) = \alpha_1 \exp\left(-\frac{\alpha_2}{2}||\mathbf{x}_i - \mathbf{x}_j||^2\right) + \alpha_3\delta_{ij}.$$

Under the RBF kernel, correlations between data points fall off smoothly as their latent coordinate distance increases.

When using the GPLVM with motion data, we must carefully consider the global position and orientation of the character's root. Prior methods often ignored this issue by removing root motion completely [Lawrence 2005], but root motion is essential for generating complete animations. Some prior methods used relative root motion [Wang et al. 2008], but adding relative root motion to $\mathbf{Y}$ has an unpleasant side effect: when we stay at the same latent position, the pose remains the same, but the root will move. This creates foot skating artifacts and results in inaccurate root velocities. Instead, we remove the root from $\mathbf{Y}$ but include it in a matrix of pose velocities $\dot{\mathbf{Y}} = [\dot{\mathbf{y}}_1, \ldots, \dot{\mathbf{y}}_{N-1}]^{\mathrm{T}}$. We model pose velocities as a function of transitions in the latent space, so that each $\dot{\mathbf{y}}_i$ is conditioned on two consecutive points in the latent space. The function is modeled as another GP, with scaling matrix $\mathbf{W}_{\dot{\mathbf{Y}}}$, kernel matrix $\mathbf{K}_{\dot{\mathbf{Y}}}$, and log likelihood $\mathcal{L}_{\dot{\mathbf{Y}}} \propto \ln p(\dot{\mathbf{Y}}|\mathbf{X}, \bar{\beta}, \mathbf{W}_{\dot{\mathbf{Y}}})$ in the form of Equation (1). Since we know that the velocity should be zero if $\mathbf{x}_t = \mathbf{x}_{t-1}$, we include this information in the GP by using a product kernel

$$k_{\dot{\mathbf{y}}}([\mathbf{x}_i, \mathbf{x}_{i-1}], [\mathbf{x}_j, \mathbf{x}_{j-1}]; \bar{\beta}) = \\ \beta_1\dot{\mathbf{x}}_i^{\mathrm{T}}\dot{\mathbf{x}}_j \exp\left(-\frac{\beta_2}{2}||\dot{\mathbf{x}}_i - \dot{\mathbf{x}}_j||^2 - \frac{\beta_3}{2}||\mathbf{x}_i - \mathbf{x}_j||^2\right) + \beta_4\delta_{ij},$$

where $\dot{\mathbf{x}}_i = \mathbf{x}_i - \mathbf{x}_{i-1}$ and $\bar{\beta} = [\beta_1, \beta_2, \beta_3, \beta_4]$. This creates a GP that is nonlinear in the velocities, while still behaving like a linear function for small values of $\dot{\mathbf{x}}_i$, so that small movements in the latent space correspond to small root velocities.

We found that including the velocities of all joints in $\dot{\mathbf{Y}}$ improved the quality of the embedding, by allowing it to take dynamic aspects of the motion into account, though only the velocities of the root are used during synthesis. We represent joint rotations with quaternions, with an additional angle to represent the rotation of the root about the vertical axis. This angle, as well as the horizontal position of the root, is only present in the velocity GP.

In addition to these reconstruction terms, we place a prior on the distribution of latent positions that captures our belief about the structure of the data. The prior consists of two parts: a dynamics term $\Phi_D(\mathbf{X})$, which captures the property that the example poses are generated consecutively by a dynamical process, and a novel connectivity term $\Phi_C(\mathbf{X})$, which captures the belief that the examples should interact with one another in the latent space, rather than being pushed apart. This connectivity term is essential for producing an embedding in which all parts of the data are reachable, and therefore usable by the controller. Since we would like to maximize both terms, our prior is a product of the form $p(\mathbf{X}) \propto \exp\left(\Phi_D(\mathbf{X})\right)\exp\left(\Phi_C(\mathbf{X})\right)$. The dynamics term is defined following the Gaussian process dynamical model (GPDM) [Wang et al. 2008], which assumes that latent points are generated by an autoregressive GP. We employ a second order model that

maps the pair $[\mathbf{x}_{i-1}, \dot{\mathbf{x}}_{i-1}]$ to $\mathbf{x}_i$, which has the log prior density

$$\Phi_D(\mathbf{X}) = -\frac{1}{2}\text{tr}\left(\mathbf{K}_\mathbf{X}^{-1}\mathbf{X}_{3:N}\mathbf{X}_{3:N}^\text{T}\right) - \frac{d_\mathbf{X}}{2}\ln|\mathbf{K}_\mathbf{X}| + \ln p(\mathbf{x}_1, \mathbf{x}_2), \tag{2}$$

where $\mathbf{X}_{3:N}$ denotes all rows except the first and second in each trajectory, $p(\mathbf{x}_1, \mathbf{x}_2)$ is a Gaussian prior on the first two frames, and the covariance $(\mathbf{K}_\mathbf{X})_{ij} = k_\text{rbf}([\mathbf{x}_{i-1}, \mathbf{x}_{i-2}], [\mathbf{x}_{j-1}, \mathbf{x}_{j-2}]; \bar{\gamma})$ is over all rows except the last.[1] We found that the second order model helps to avoid rapid changes in direction during synthesis. Following Lawrence [2006], we fix kernel hyperparameters for the dynamics to $\bar{\gamma} = [0.005, 0.5, 1e^{-4}]$ in all of our examples.

4.2 The Connectivity Prior

Models based on the GPLVM/GPDM tend to embed different motion trajectories far apart (even if similar poses exist), leading to poor pose reconstruction across trajectories [Wang et al. 2007]. This is because the GPLVM places dissimilar poses far apart, but makes no effort to place similar poses close together [Lawrence and Quiñonero Candela 2006]. This is a serious problem for control, because the agility of the controller depends on its ability to rapidly reach a variety of poses. If two portions of the example data are too far apart, one or the other will be not be used by the controller.

We propose a novel prior on the latent points $\mathbf{X}$ that encodes our preference for well-connected embeddings, and does not depend on the training poses. Specifically, we model the degree of connectivity in $\mathbf{X}$ using *graph diffusion kernels*, which are closely related to random walk processes on graphs [Kondor and Vert 2004].

Letting G denote a complete weighted graph with N vertices corresponding to latent variables $\mathbf{x}_1, \ldots, \mathbf{x}_N$, we define a random walk process on G such that the edge weights corresponds to the probabilities of single-step transitions between the connected vertices. We set $w(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|^{-p}$, where the value p controls how much we prefer short jumps over long ones. As p is increased, long series of short transitions become much more likely than a few long ones. We found that a value of $p = 4$ worked well. We consider $\mathbf{X}$ to be well-connected if no point in the embedding is unlikely to be reached from some other point under this random walk.

The connectedness of $\mathbf{X}$ can be determined using the diffusion kernel on G, which can be computed from the negative normalized graph Laplacian $\mathbf{H} = -\mathbf{T}^{-1/2}\mathbf{L}\mathbf{T}^{-1/2}$, where $\mathbf{T}$ is a diagonal matrix such that $\mathbf{T}_{ii} = \sum_j w(\mathbf{x}_i, \mathbf{x}_j)$ and

$$\mathbf{L}_{ij} = \begin{cases} \sum_k w(\mathbf{x}_i, \mathbf{x}_k) & \text{if } i = j \\ -w(\mathbf{x}_i, \mathbf{x}_j) & \text{otherwise} \end{cases}.$$

Given $\mathbf{H}$, the graph diffusion kernel $\mathbf{K}^d$ is obtained by the matrix exponential $\mathbf{K}^d = \exp(\beta\mathbf{H})$, where β is a diffusion rate.

$\mathbf{K}^d_{ij}$ gives the probability of a continuous time random walker starting from vertex i being found at vertex j [Kondor and Vert 2004]. Higher diffusion rates β serve to lower the diffusion distance between points connected by many short jumps, while points connected by a few long jumps remain separated due to our choice of transition probability. We empirically set $\beta = 1000$, and define the connectivity term as

$$\Phi_C(\mathbf{X}) = w_c \sum_{ij} \ln \mathbf{K}^d_{ij}, \tag{3}$$

[1] Here we assume a single trajectory for compactness, see Wang et al. [2008] for details regarding concatenation of multiple trajectories.

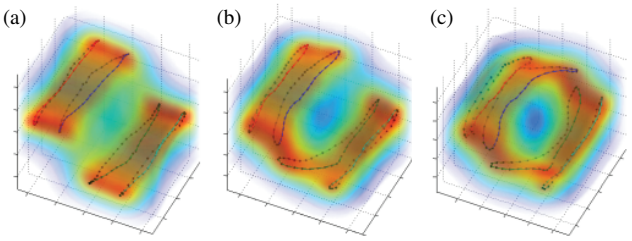

Figure 3: *Walking embeddings learned (a) without the connectivity term, (b) with $w_c = 0.1$, and (c) with $w_c = 1.0$. The prior encourages connections between the clips. Warmer colors indicate lower reconstruction variance.*

where $w_c = 0.1$ is a weight on the connectivity prior. This term corresponds to a "soft minimum" over the diffusion kernel values, and effectively penalizes the embedding proportional to its most disconnected pair of points. By itself, $\Phi_C(\mathbf{X})$ encourages all points to be connected by paths that consist of a number of short jumps, which leads to evenly spaced embeddings. When combined with the likelihood and the dynamics terms, we obtain a model that prefers embeddings where many trajectories interact frequently and all points can be reached from all other points, as shown in Figure 3.

4.3 Model Learning

Learning requires estimating the low-dimensional latent coordinates $\mathbf{X}$ and hyperparameters $\bar{\alpha}, \bar{\beta}, \mathbf{W}$, and $\mathbf{W}_{\dot{\mathbf{y}}}$ from the data matrices $\mathbf{Y}$ and $\dot{\mathbf{Y}}$. This is done by maximizing the log posterior

$$\ln p(\mathbf{X}, \bar{\alpha}, \bar{\beta}, \mathbf{W}, \mathbf{W}_{\dot{\mathbf{Y}}} | \mathbf{Y}, \dot{\mathbf{Y}}) \propto \mathcal{L}_\mathbf{Y} + \mathcal{L}_{\dot{\mathbf{Y}}} + \Phi_D(\mathbf{X}) + \Phi_C(\mathbf{X}) + \ln p(\bar{\alpha}) + \ln p(\bar{\beta}). \tag{4}$$

The likelihoods $\mathcal{L}_\mathbf{Y}$ and $\mathcal{L}_{\dot{\mathbf{Y}}}$ are pose and velocity reconstruction terms defined by Equation (1), which encourage the solution to be consistent with data, while $\Phi_D(\mathbf{X})$ and $\Phi_C(\mathbf{X})$ encourage smooth and well-connected embeddings, and are defined by Equations (2) and (3). As in previous work, the hyperparameter priors were set to $\ln p(\bar{\alpha}) = -\sum_i \ln \alpha_i$ and $\ln p(\bar{\beta}) = -\sum_i \ln \beta_i$ [Wang et al. 2008]. To learn the model, the log posterior was maximized using the LBFGS algorithm. The gradients of the likelihood, including the connectivity term, are discussed in Appendix B.

4.4 Pose Synthesis

Given a learned model $\Gamma = \{\mathbf{Y}, \mathbf{X}, \bar{\alpha}, \bar{\beta}, \mathbf{W}\}$, the prediction distribution over new poses given a corresponding latent coordinate $p(\mathbf{y}|\mathbf{x}, \Gamma)$ is Gaussian, with mean and covariance given by

$$g_\mathbf{y}(\mathbf{x}) = \mathbf{W}\mathbf{Y}^\text{T}\mathbf{K}_\mathbf{Y}^{-1}\mathbf{k}(\mathbf{x}) + \mathbf{b},$$
$$g^\sigma_\mathbf{y}(\mathbf{x}) = \mathbf{W}^2\left(k_\text{rbf}(\mathbf{x}, \mathbf{x}) - \mathbf{k}(\mathbf{x})^\text{T}\mathbf{K}_\mathbf{Y}^{-1}\mathbf{k}(\mathbf{x})\right),$$

where $\mathbf{k}(\mathbf{x})$ is an $N \times 1$ vector with i-th element $k_\text{rbf}(\mathbf{x}, \mathbf{x}_i)$ and $\mathbf{b}$ is a $d_\mathbf{Y} \times 1$ bias term equal to the mean of the original data. Given a latent point $\mathbf{x}$, $g_\mathbf{y}(\mathbf{x})$ represents the most likely corresponding pose. The variance $g^\sigma_\mathbf{y}(\mathbf{x})$ provides a confidence measure for the reconstructed pose: if $\mathbf{x}$ is far from the training data, it will often have a high variance in the prediction distribution. We take advantage of this confidence measure to learn control policies that avoid low-quality motions. The means $(g_{\dot{\mathbf{y}}}, f_\mathbf{x})$ and covariances $(g^\sigma_{\dot{\mathbf{y}}}, f^\sigma_\mathbf{x})$ of the prediction distributions for the velocity and dynamics functions can be written in a similar fashion. The dynamics distribution has previously been used to simulate the latent dynamical process,

producing motion samples [Wang et al. 2008]. In the next section, we show how we can instead produce goal-directed motions using optimal control in the latent space.

5 Control

In the second training stage, we precompute a near-optimal policy that navigates the learned latent space in order to accomplish a user-specified task. At runtime, this policy chooses transitions in the latent space at every time step, in response to the task (which includes user input) and the current latent position. Since points in the latent space correspond to poses, the policy produces character motions that accomplish the desired task in real time. The user specifies the task with a set of task parameters Θ, as well as a reward function $\hat{\mathcal{R}}$, which can depend on pose, velocity, and parameter value. In a walking task, Θ might be the desired walking direction, and $\hat{\mathcal{R}}$ might be high for moving in that direction. The policy would choose latent space transitions that maximize $\hat{\mathcal{R}}$, producing an animation that walks in the chosen direction.

5.1 Optimal Control

We model the control task as a Markov decision process (MDP). In an MDP, an agent chooses actions $\hat{a}$ to transition between states $\hat{s}$ in order to maximize the long-term sum of rewards. In the full body control task, states are pose-velocity-parameter tuples $\hat{s} = (\mathbf{y}, \dot{\mathbf{y}}, \theta)$, actions are changes in pose, and the reward function $\hat{\mathcal{R}}(\hat{s}_t, \hat{s}_{t+1})$ is given by the user. Velocities are included in the state to ensure smoothness and handle tasks with velocity-dependent rewards, such as martial arts. The optimal policy $\hat{\pi}^\star$ maximizes the expected sum of rewards over all time, with future rewards discounted by γ at each time step [Bertsekas 2001]:

$$\hat{\pi}^\star = \arg\max_{\hat{\pi}} \sum_{t=0}^{\infty} \gamma^t E[\hat{\mathcal{R}}(\hat{s}_t, \hat{s}_{t+1})|\hat{\pi}].$$

Optimal control in such a high dimensional space is intractable, because the difficulty of solving an MDP increases exponentially with the dimensionality of the state space [Bertsekas 2001]. Fortunately, the GPLVM latent space provides us with a low-dimensional alternative that still covers a variety of poses consistent with the user-provided data. We therefore learn a policy π on the reduced states $s = (\mathbf{x}, \dot{\mathbf{x}}, \theta)$, using the learned GP to reconstruct full body poses.

5.2 Reward Functions for Latent Space Control

Each state s_t maps to a full state $\hat{s}_t = (g_{\mathbf{y}}(\mathbf{x}_t), g_{\dot{\mathbf{y}}}(\mathbf{x}_t, \dot{\mathbf{x}}_t), \theta_t)$, where $g_{\mathbf{y}}(\mathbf{x}_t)$ and $g_{\dot{\mathbf{y}}}(\mathbf{x}_t, \dot{\mathbf{x}}_t)$ are the GP means defined in Section 4.4. We can use this mapping to project the reward function into the reduced space. Since not all latent points correspond to equally good poses, we add a quality term $\mathcal{R}_Q$. The full reward is then given by

$$\mathcal{R}(s_t, s_{t+1}) = \hat{\mathcal{R}}(\hat{s}_t, \hat{s}_{t+1}) + \mathcal{R}_Q(\mathbf{x}_t, \mathbf{x}_{t+1}).$$

We define the quality term using the variance of the reconstruction GP, which can act as a confidence score for the quality of the emitted pose [Grochow et al. 2004]. Similarly, the dynamics GP specifies a probability for each transition in the latent space, with the probability of a transition from $\mathbf{x}_t$ to $\mathbf{x}_{t+1}$ being proportional to $\exp(\frac{1}{f^\sigma_{\mathbf{x}}} ||\mathbf{x}_{t+1} - f_{\mathbf{x}}(\mathbf{x}_t)||^2)$. We define the quality term as

$$\mathcal{R}_Q(\mathbf{x}_t, \mathbf{x}_{t+1}) = -c_{\mathbf{x}} \frac{1}{f^\sigma_{\mathbf{x}}} ||\mathbf{x}_{t+1} - f_{\mathbf{x}}(\mathbf{x}_t)||^2 - c_{\mathbf{y}} g^\sigma_{\mathbf{y}}(\mathbf{x}_{t+1}).$$

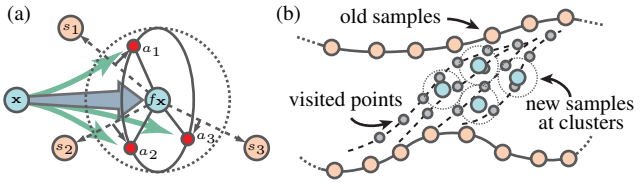

Figure 4: *(a) Actions are sampled in the direction of nearby value samples, on a ring orthogonal to the dynamics mean and radius proportional to the dynamics variance. (b) Refining samples are added by clustering visited points, weighted by Bellman error.*

The weight of the quality term is set by the constants $c_{\mathbf{x}} = 0.04$ and $c_{\mathbf{y}} = 0.04\mu_\sigma^{-1}$, where μ_σ is the average variance of the pose GP at the example points. While a variety of quality rewards can be constructed from the GP posterior distributions, our choice is motivated by the desire to avoid points with very high variance while minimizing the penalty for points near the data. Our penalty creates a low-cost basin in a region around the data, while heavily penalizing points that are very far away, effectively constraining the controller to high quality poses without seriously damaging its agility.

5.3 Actions

To control the character at interactive rates, we must select a discrete set of actions so that the policy can be evaluated efficiently at runtime. The quality term will penalize actions that are far from the mean of the GP dynamics. We therefore choose the mean itself and a number of points on a ring perpendicular to the mean direction, with a radius chosen so that $\exp(\frac{1}{f^\sigma_{\mathbf{x}}} ||\mathbf{x}_{t+1} - f_{\mathbf{x}}(\mathbf{x}_t)||^2) = \rho$. The value ρ provides a natural way to trade off agility for motion quality, since large transitions are more jerky. We use $\rho = 0.9$ in our implementation. Since the size of the ring depends on the dynamics variance $f^\sigma_{\mathbf{x}}(\mathbf{x}_t)$, the actions naturally become more aggressive further from the data, which prevents the controller from getting bogged down in high-variance, low-quality regions. The effect of the actions is to follow the GP dynamics but also gently "steer" the character in perpendicular directions as needed. To choose the particular points on the ring, we note that optimal actions tend to flow towards peaks in the MDP's *value function*. As discussed in the next section, we represent the value function with a nonparametric estimator, using a set of samples that include the example points. The peaks of our estimator lie at sample locations, so we assume that good actions will lie in the direction of nearby samples. We therefore place the samples on the ring in the direction of 16 nearby samples, as shown in Figure 4 (a).

5.4 Approximate Dynamic Programming

The state space of our MDP is continuous, so precomputing the optimal policy exactly is impossible. Instead, we use an approximate dynamic programming algorithm that estimates the MDP's value function using a nonparametric function estimator. The value function is the expected sum of rewards obtained by the optimal policy starting from a given state, and allows the optimal policy to be followed simply by choosing the actions that greedily maximize value. The value function is defined recursively as

$$V(s) = \max_a \mathcal{R}(s, a(s)) + \gamma V(a(s)), \tag{5}$$

where we abuse notation and use $a(s)$ to denote the state resulting from action a in state s. Recall that states are tuples $(\mathbf{x}, \dot{\mathbf{x}}, \theta)$, and actions are velocities in the latent space. In approximate dynamic

programming, V is represented with a function estimator $\tilde{V}$. We use a nonparameteric estimator based on [Ormoneit and Sen 2002]:

$$\tilde{V}(s) = \sum_{i=1}^{N} \phi(s, s_i) V(s). \tag{6}$$

The value at s is obtained by weighing samples at nearby states s_i using a kernel ϕ. Since the example poses are already known to be important for the task, we initially set the sample points to their latent positions $\mathbf{X}$, with task parameters sampled on a grid. $\tilde{V}$ allows us to approximately solve for the value at each sample state s_i by repeatedly applying the approximate version of the Bellman backup operator [Ormoneit and Sen 2002; Bertsekas 2001]:

$$V^{(k+1)}(s_i) = \max_a \mathcal{R}(s_i, a(s_i)) + \gamma \tilde{V}^{(k)}(a(s_i)). \tag{7}$$

For a state $(\mathbf{x}, \dot{\mathbf{x}}, \theta)$, we follow [Lee et al. 2010] and define the kernel as the product of a kernel over $(\mathbf{x}, \dot{\mathbf{x}})$ and task parameters θ. The $(\mathbf{x}, \dot{\mathbf{x}})$ kernel weighs the k nearest neighbors of $(\mathbf{x}, \dot{\mathbf{x}})$ proportionally to their inverse squared distance $w_i = \|(\mathbf{x}, \dot{\mathbf{x}}) - (\mathbf{x}, \dot{\mathbf{x}})_i\|^{-2}$, and the parameter kernel performs multilinear interpolation of adjacent grid points, which we denote by $\mathcal{I}(\theta, \theta_i)$. The final kernel is

$$\phi(s, s_i) = \mathcal{I}(\theta, \theta_i) \frac{w_i}{\sum_j w_j}.$$

At runtime, we simply choose the action that maximizes the one-step reward plus the value at the destination state, and display the pose given by $g_{\mathbf{y}}(\mathbf{x})$ for the current latent position $\mathbf{x}$:

$$\pi^\star(s) = \arg\max_a \left(\mathcal{R}(s, a(s)) + \gamma \tilde{V}(a(s)) \right).$$

The controller therefore only needs to store the values at each grid cell for each sample s_i.

5.5 Refining Samples with Bellman Error

Although the example points provide us with a good initial set of samples, values far from these samples may be inaccurate. We therefore employ an iterative refinement procedure to sample additional states and repair regions where $\tilde{V}$ provides a poor estimate of the value function. This allows us to learn a policy that effectively utilizes those regions of the latent space that are not near the example points but still correspond to useful motions. Bellman error quantifies the degree to which the approximate value function disagrees with Equation (5), and is defined as

$$\mathcal{E}(s) = \left(\tilde{V}(s) - \max_a \left[\mathcal{R}(s, a(s)) + \gamma \tilde{V}(a(s)) \right] \right)^2.$$

A good approximation will have low error at all states that the MDP will visit. If s is one of the samples, $\mathcal{E}(s)$ is zero by definition (Equation (7)), so we could reduce the error at a given state to zero by adding that state to our set of samples and repeating Equation (5) to convergence. We therefore perform several *refinement* steps, in which we first iterate Equation (7) to convergence, and then run the controller from a number of randomly selected starting locations to find states with high Bellman error. The latent positions and velocities corresponding to the visited states are then clustered to identify a small set of representative points, with each state weighted by its Bellman error. New samples are then added at the cluster centers, as shown in Figure 4.

We use expectation-maximization (EM) clustering with fixed cluster widths to obtain a uniform-density sampling of the visited regions. Uniform density is important because highly nonuniform samplings cause difficulty for nearest-neighbor based estimators. We heuristically estimate the local density of points by $\dot{\mathbf{x}}$, which corresponds to the spacing between temporally adjacent points, and set the cluster widths to $\frac{1}{16}$ of the average squared norm of its members' $\dot{\mathbf{x}}$ values at each iteration of EM. We perform four refinement steps, testing fifty 200-step trajectories at each step and increasing the number of samples by half the number of original example points. We used a parallel implementation of value iteration to speed up computation, which allowed all four refinement steps to be completed within a couple of hours for each controller.

6 Constraints and Controller Switching

In addition to producing agile controllers, our method provides principled probabilistic interpretations for common animation processing operations. Characters often interact with dynamic environments and accomplish secondary tasks: a character's pose might be modified to avoid intersecting a wall, foot skate cleanup might be used to avoid foot slipping artifacts, and the character might need to execute a task while holding an object or facing a particular direction. Such modifications are usually done in an application-specific manner with warping and inverse kinematics (IK). We can reformulate such operations as probabilistic inference, and apply them automatically during synthesis. We can also use this probabilistic framework to describe how the character should behave when the current controller is switched to a different one, which may be trained on different motion data to perform a different task.

6.1 User-Specified Pose Constraints

We represent user-specified warping tasks as distributions over poses. A natural way to specify a warping task is with a constraint $C(\mathbf{y}) = 0$, which induces a piecewise constant distribution

$$p_C(\mathbf{y}) \propto 1_{\{C(\mathbf{y})=0\}}.$$

In Section 4.4, we discussed how a point in the latent space corresponds to a Gaussian distribution over poses. Though we usually take the mean of this Gaussian, we can account for user-specified constraints by instead taking the most likely pose under the product of the GP distribution and the current constraints, given by

$$p(\mathbf{y}|\mathbf{x}) = p_{g_{\mathbf{y}}}(\mathbf{y}|\mathbf{x}) \prod_i p_{C\,i}(\mathbf{y}).$$

The most likely pose is obtained by maximizing $p_{g_{\mathbf{y}}}(\mathbf{y}|\mathbf{x})$ subject to the constraints imposed by $C_1, \ldots, C_n$. Since $p_{g_{\mathbf{y}}}(\mathbf{y}|\mathbf{x})$ is Gaussian with mean $g_{\mathbf{y}}(\mathbf{x})$ and diagonal covariance $g^\sigma_{\mathbf{y}}(\mathbf{x})$, the log probability of the pose is quadratic. Although we can solve this optimization in real time with standard nonlinear optimization techniques, many constraints can be linearized around the current pose. If only equality constraints are used, the linearized optimization is simply a linear system. In the case of inequality constraints, it is a quadratic program. In both cases, the optimization is given by

$$\max_{\mathbf{y}} ||\mathbf{y} - \mu||^2_\sigma \;\text{ s.t. } C_i(\mathbf{y}) = 0 \quad \forall\, i, \tag{8}$$

where $\mu = g_{\mathbf{y}}(\mathbf{x})$ denotes the mean and $\sigma = g^\sigma_{\mathbf{y}}(\mathbf{x})$ is the variance. Specific constraints, such as IK, are described in Appendix A.

6.2 Temporal Smoothness

Applying the constrained optimization independently to each frame may produce an animation that is not smooth, particularly if the constraints change suddenly. However, we can ensure temporal coherence by conditioning the pose distribution on the previous pose: instead of using $p_{g_{\mathbf{y}}}(\mathbf{y}_t|\mathbf{x}_t)$, we will instead use

	clips	frames	sec		clips	frames	sec
horse	3	71	2.4	run 1	2	75	2.5
dragon	3	138	4.6	run 2	4	220	7.3
dinosaur	3	188	6.3	punch 1	4	341	11.4
walk 1	2	135	4.5	punch 2	3	424	14.1
walk 2	4	266	8.9	punch 3	3	225	7.5
walk 3	4	259	8.5	kick	3	507	16.9

Figure 5: *The datasets used in our evaluation. Locomotion controllers used mirrored copies of the data for symmetry, which are not included in the totals. All clips are at 30 Hz.*

$p_{g_\mathbf{y}}(\mathbf{y}_t|\mathbf{y}_{t-1}, \mathbf{x}_t, \mathbf{x}_{t-1})$. The kernel function provides us with a covariance over any set of poses for which the latent coordinates are known, and we can compute a covariance over $\mathbf{y}_t$ and $\mathbf{y}_{t-1}$ as

$$\Sigma = k(\mathbf{X}_{t-1,t}, \mathbf{X}_{t-1,t}) - k(\mathbf{X}_{t-1,t}, \mathbf{X})\mathbf{K}^{-1}k(\mathbf{X}, \mathbf{X}_{t-1,t}),$$

where $\mathbf{X}_{t-1,t}$ is a matrix with $\mathbf{x}_{t-1}$ in the first row and $\mathbf{x}_t$ in the second.[2] Since $\mathbf{y}_t$ and $\mathbf{y}_{t-1}$ are jointly Gaussian, the conditional distribution is Gaussian, with the mean and variance given by

$$\mu = g_\mathbf{y}(\mathbf{x}_t) + \Sigma_{12}\Sigma_{22}^{-1}(\mathbf{y}_{t-1} - g_\mathbf{y}(\mathbf{x}_{t-1}))$$
$$\sigma = \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}$$

prior to scaling and biasing. The conditional mean smoothly returns the pose to $g_\mathbf{y}(\mathbf{x}_t)$ following a deviation, and the speed of recovery depends on the covariance between successive frames. To smoothly satisfy the constraints, we simply replace the mean and variance in Equation (8) with the conditional mean and variance given above.

6.3 Controller Switching

We can also use the conditional mean described in the previous section to smoothly switch from one controller to another, for example when the user chooses to change the current task or motion style. We must simply find the position $\mathbf{x}_t$ in the new controller's latent space that best resembles the current pose $\mathbf{y}_t$. This is done by optimizing $p(\mathbf{y}_t|\mathbf{x}_t)$ with respect to $\mathbf{x}_t$ under the new model, using the position of the closest example frame in the new model for initialization. This optimization is similar to the one used by Grochow et al. [2004], and can be performed at interactive rates. The mean pose corresponding to $\mathbf{x}_t$ may deviate from $\mathbf{y}_t$ if the new controller has different example motions, but the conditional mean provides a smooth sequence of poses that blend into the new controller. Naturally, this technique is only effective if the poses available to the new controller do not vary too drastically from the old one. Otherwise, a more sophisticated transition scheme may be required, such as the transition controllers described by [Lee et al. 2009].

7 Evaluation

We evaluated our method on directional locomotion and karate punches and kicks. In this evaluation, we analyze the results using both motion capture and keyframed motions as input, and evaluate the importance of the proposed connectivity prior by comparing controllers trained with and without this term. We also compare our controllers with a previous continuous control method by Lee et al. [2010] and a graph-based discrete controller based on prior work [Lee and Lee 2006; Treuille et al. 2007; McCann and Pollard 2007]. Finally, we analyze how the method behaves as the size of the dataset is increased. The accompanying video, together with an implementation of our method, is available from `http://graphics.stanford.edu/projects/cccIde`.

[2]For the smoothest results, noise should not be added to the diagonal of $k(\mathbf{X}_{t-1,t}, \mathbf{X}_{t-1,t})$, since the synthesized sequence will be noiseless.

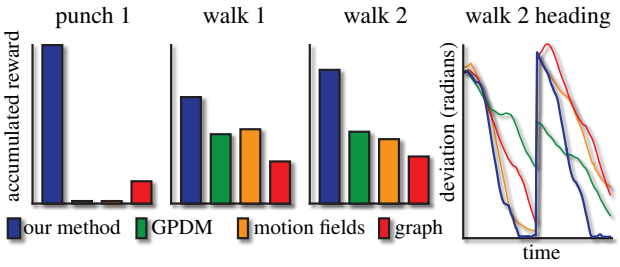

Figure 6: *Comparison between our method, our method with a standard GPDM without connectivity prior, motion fields, and a discrete graph-based controller. Bars compare total reward over 1000 steps, graph shows deviation from target heading over time.*

7.1 Controllers

Our controllers use either a directional or location-based task parameter space. Directional locomotion controllers use a one-dimensional task parameter space consisting of the desired movement direction in the coordinate frame of the character's root, while the martial arts controllers use a two-dimensional parameter space consisting of the horizontal position of the target, in polar coordinates around the character's coordinate frame.

The reward function for locomotion tasks increases exponentially as the difference between the current and desired velocity direction approaches zero, in order to produce a sharp peak at the desired heading. The reward function for martial arts tasks increases exponentially as the distance between the desired joint and the target approaches zero, and is multiplied by a threshold function to ensure the joint is traveling at sufficient velocity.

For each task, we constructed several controllers using various datasets. The set of controllers shown in the accompanying video is listed in Figure 5, along with the amount of data used for training. Locomotion controllers used four-dimensional latent spaces, while the martial arts spaces used five dimensions. All latent space figures show the first three dimensions. For walking and running controllers, the data was mirrored to ensure that both left and right turns are available. The table lists dataset sizes before mirroring. Notably, our method was able to produce compelling walking and running controllers from just two clips.

7.2 Nonhuman Characters

Example motions can be particularly scarce when they are produced by an artist, which is often the case for characters that cannot be motion captured, such as animals or fantasy creatures. To demonstrate that our method can produce controllers from small sets of such motions, we trained directional walking controllers from artist-created animations of a horse, a dragon, and a dinosaur. These examples also demonstrate that our method can seamlessly animate both bipedal and quadrupedal characters without special handling. The datasets for these controllers are listed in Figure 5, and the accompanying video presents example animations.

7.3 Ablation and Comparisons

To evaluate the connectivity term proposed in Section 4.2, we tested a set of controllers trained without this term, instead using the standard GPDM dynamics prior [Wang et al. 2008]. The results in Figure 6 compare the total reward (without the quality term) obtained by each controller on punching and walking tasks. Walking controllers trained without the connectivity prior performed drasti-

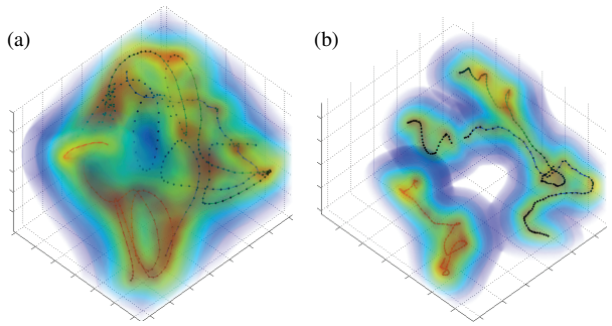

Figure 7: *Embeddings for the punching task (a) with and (b) without the connectivity term. When using the standard GPDM prior instead of the connectivity term, the embedding lacks the connections necessary for agile control.*

cally worse, while karate punching could not be performed at all. As shown in Figure 7, the standard GPDM could not embed the karate data with any meaningful connections, making it unsuitable for control. This indicates that the proposed connectivity term is essential for using such models for interactive control tasks.

Figure 6 also shows the rewards obtained by the motion fields method of Lee et al. [2010] and a discrete graph-based method modeled on prior work [Lee and Lee 2006; McCann and Pollard 2007; Treuille et al. 2007]. The reward for the discrete controller was integrated along graph edges for a fair comparison. Since motion fields do not use a probabilistic model to generalize the user-provided examples, they require substantially more data to produce compelling controllers. Although motion fields attained a higher reward on the locomotion task than the graph controller, the small dataset resulted in highly objectionable artifacts, as can be seen in the accompanying video. We found that motion fields could produce walking results comparable to our small 4.5 second controller with 21.3 seconds of data. Motion fields were unable to complete the punching task with the datasets we had available. The discrete graph-based method did not exhibit severe visual artifacts, but did not possess the agility of continuous methods, as indicated by the low reward it attained on the locomotion tasks.

7.4 Constrained Synthesis

We evaluated the constraints described in Section 6.1 and Appendix A on several controllers. All controllers in the accompanying video use the skate cleanup constraint. We also present a punching controller that uses a collision constraint to prevent the character's limbs from intersecting the target cylinder. In the accompanying video, the collision constraint can be seen to noticeably modify the character's pose to prevent intersections, and the character gracefully recovers after a collision without visible discontinuities due to the automatic probabilistic blending described in Section 6.2. Finally, we show several examples of controllers that use inverse kinematics constraints to alter the synthesized animation, including walking characters that keep their hands in their pockets or point a weapon at a target, and a punching character that is constrained to hold the target with the left hand. No additional data was used to produce these controllers. Images of the constraints are shown in Figure 8, and example animations are provided in the accompanying video.

7.5 Controller Switching

We evaluated the controller switching technique described in Section 6.3 by switching between walking and running, as well as

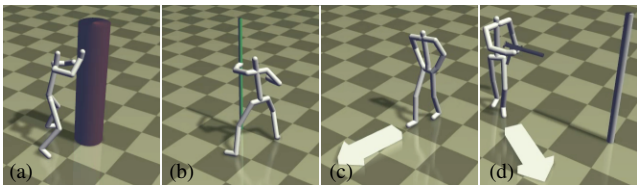

Figure 8: *We evaluated constrained synthesis with constraints that (a) prevent the limbs from intersecting a cylindrical target, (b) keep the left hand on the pole while punching, (c) keep the hands on the hips, and (d) cause the character to point a weapon at a target.*

punching and kicking. The controllers that were switched used entirely different datasets, although the motions had considerable qualitative similarity. As can be seen in the accompanying video, an interesting effect of the conditional mean blending technique from Section 6.2 is that tasks like running, which use faster motions, also exhibit quicker, more sudden transitions, while transitions into walking are more gradual, reflecting the statistical properties of walking motions. As discussed in Section 6.3, the effectiveness of this switching method depends on the similarity of the motions in the two controllers, and switching between highly dissimilar controllers may require more sophisticated machinery.

8 Discussion

We presented a method for animating interactively controlled characters using a small set of user-provided example motions. Our method continuously controls the character through a low-dimensional space learned by our probabilistic motion model. The learned motion space generalizes the examples, producing new transitions and variations. By controlling the character in a continuous reduced space, we avoid the curse of dimensionality while retaining the ability to continuously vary the character's pose in response to changes in the task and user input. As shown in the accompanying video, our method can handle complex tasks such as punching and kicking with just a few example motions, and can rapidly respond to changes in user input.

8.1 Limitations on Larger Datasets

A common concern with GPLVM-based methods is their capacity to handle large, heterogeneous datasets. We showed that effective controllers can be constructed from small sets of examples, but not all tasks can be described with such small datasets. With the aid of the connectivity prior, we could construct well-connected embeddings from our entire walking and karate datasets (69 and 62 seconds in length, respectively). The quality of the resulting controllers was comparable to those shown in the video, and the karate controller exhibited a wider variety of motions. However, the GPLVM is a nonparametric model, so the per-frame synthesis time scales linearly with the size of the training dataset. Our unoptimized, single-threaded implementation could therefore only sustain interactive animation rates on datasets under 30 seconds in length. Switching from a state value function to a state-action Q-function would improve runtime performance by a factor equal to the number of actions, since the poses resulting from each action would no longer need to be computed, and sparse GP approximation techniques could overcome the linear dependence on dataset size entirely by using a constant number of basis functions [Quiñonero Candela and Rasmussen 2005; Walder et al. 2008].

In addition to the synthesis cost, larger datasets also result in longer training times. Since evaluating the objective in Equation (4) requires inverting the kernel matrix, the training time scales cubically.

Sparse approximation techniques can overcome this limitation as well, and have been effective for training larger GPLVM models [Lawrence 2007].

Besides processing time, datasets with many heterogeneous motions also require greater care in selecting the latent space dimensionality. The model generalizes better when dimensionality is low, but if it is too low to represent the full variation in the data, the controller may exhibit artifacts as dissimilar motions are placed close together, or loss of agility as similar motions are placed far apart. Small or homogeneous datasets are insensitive to dimensionality, but large heterogeneous datasets require more care. Automatic methods for selecting latent space dimensionality can address this limitation [Geiger et al. 2009; Titsias and Lawrence 2010].

8.2 Future Work

One interesting avenue for future work is to incorporate additional domain knowledge into the model to improve generalization. For example, our model is unaware of many physical constraints on the character's motion, such as contacts and balance. While high-confidence poses appear physically plausible due to their similarity to the data, low-confidence poses may not. The quality term in the reward function prevents such regions from being visited, but a more physically-aware model may allow more powerful generalization in the future.

Our probabilistic model learns the reduced space without considering the control task. This enables the same model to be used with multiple controllers, but does not allow the dimensionality reduction process to emphasize task-relevant aspects of the data. An interesting direction for future work is to more tightly connect the two stages of our method, so that a more compact embedding could be learned by leveraging knowledge about the intended task and emphasizing those variations in the data that are most relevant.

Another exciting avenue for future work is to more effectively make use of the entire distributions over poses produced by our model for each point in the latent space. A hierarchical control scheme could use the latent space to perform high-level planning while performing fine-grained low-level control in the space of full body poses, subject to the distribution at the current latent position. For example, our model could be used to guide a low-level physics-based controller by providing it with a flexible objective that adjusts to the current configuration of the character, or to provide high-level planning for a manual manipulation controller to maneuver into position and manipulate an object. Another exciting direction for future work is to apply dimensionality reduction techniques not just to the space of motions, but also to the parameters of the task, in order to handle complex tasks with many parameters.

Acknowledgements

We thank Mixamo and Yongjoon Lee for providing motion clips, Philipp Krähenbühl for help with the rendering system, our motion capture participants for the martial arts motions, and the anonymous reviewers for their constructive comments. This work was conducted in conjunction with the Intel Science and Technology Center for Visual Computing and supported by NSF grant IIS-1017938. Sergey Levine was supported by NSF Graduate Research Fellowship DGE-0645962.

References

ARIKAN, O., AND FORSYTH, D. A. 2002. Interactive motion generation from examples. *ACM Transactions on Graphics 21*, 3, 483–490.

BERTSEKAS, D. P. 2001. *Dynamic Programming and Optimal Control*. Athena Scientific, Belmont, MA.

CHAI, J., AND HODGINS, J. K. 2007. Constraint-based motion optimization using a statistical dynamic model. *ACM Transactions on Graphics 26*, 3, 8:1–8:9.

GEIGER, A., URTASUN, R., AND DARRELL, T. 2009. Rank priors for continuous non-linear dimensionality reduction. In *Proc. CVPR*, IEEE, 880–887.

GROCHOW, K., MARTIN, S. L., HERTZMANN, A., AND POPOVIĆ, Z. 2004. Style-based inverse kinematics. *ACM Transactions on Graphics 23*, 3, 522–531.

HSU, E., PULLI, K., AND POPOVIC, J. 2005. Style translation for human motion. *ACM Transactions on Graphics 24*, 3, 1082–1089.

IKEMOTO, L., ARIKAN, O., AND FORSYTH, D. 2009. Generalizing motion edits with gaussian processes. *ACM Transactions on Graphics 28*, 1, 1:1–1:12.

JOHANSEN, R. S. 2009. *Automated Semi-Procedural Animation for Character Locomotion*. Master's thesis, Aarhus University.

KALBFLEISCH, J. D., AND LAWLESS, J. F. 1985. The analysis of panel markov data under a assumption. *Journal of the American Statistical Association 80*, 392, 863– 871.

KONDOR, R. I., AND VERT, J.-P. 2004. Diffusion kernels. In *Kernel Methods in Computational Biology*. The MIT Press, 171–192.

KOVAR, L., GLEICHER, M., AND PIGHIN, F. H. 2002. Motion graphs. *ACM Transactions on Graphics 21*, 3, 473–482.

LAU, M., BAR-JOSEPH, Z., AND KUFFNER, J. 2009. Modeling spatial and temporal variation in motion data. *ACM Transactions on Graphics 28*, 5, 171:1–171:10.

LAWRENCE, N. D., AND QUIÑONERO CANDELA, J. 2006. Local distance preservation in the GP-LVM through back constraints. In *Proc. ICML*, ACM, 513–520.

LAWRENCE, N. D. 2005. Probabilistic non-linear principal component analysis with Gaussian process latent variable models. *Journal of Machine Learning Research 6*, 1783–1816.

LAWRENCE, N. D. 2006. The Gaussian process latent variable model. Tech. rep., University of Sheffield.

LAWRENCE, N. D. 2007. Learning for larger datasets with the gaussian process latent variable model. *Journal of Machine Learning Research 2*, 243–250.

LEE, J., AND LEE, K. H. 2006. Precomputing avatar behavior from human motion data. *Graphical Models 68*, 2, 158–174.

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, 491–500.

LEE, Y., LEE, S. J., AND POPOVIĆ, Z. 2009. Compact character controllers. *ACM Transactions on Graphics 28*, 5, 169:1–169:8.

LEE, Y., WAMPLER, K., BERNSTEIN, G., POPOVIĆ, J., AND POPOVIĆ, Z. 2010. Motion fields for interactive character locomotion. *ACM Transactions on Graphics 29*, 6, 138:1–138:8.

LO, W.-Y., AND ZWICKER, M. 2008. Real-time planning for parameterized human motion. In *Symposium on Computer Animation*, ACM/Eurographics, 29–38.

MCCANN, J., AND POLLARD, N. 2007. Responsive characters from motion fragments. *ACM Transactions on Graphics 26*, 3, 6:1–6:8.

MIN, J., CHEN, Y.-L., AND CHAI, J. 2009. Interactive generation of human animation with deformable motion models. *ACM Transactions on Graphics 29*, 1.

MUKAI, T., AND KURIYAMA, S. 2005. Geostatistical motion interpolation. *ACM Transactions on Graphics 24*, 3, 1062–1070.

ORMONEIT, D., AND SEN, S. 2002. Kernel-based reinforcement learning. *Machine Learning 49*, 2-3, 161–178.

QUIÑONERO CANDELA, J., AND RASMUSSEN, C. E. 2005. A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research 6*, 1939–1959.

REN, C., ZHAO, L., AND SAFONOVA, A. 2010. Human motion synthesis with optimization-based graphs. *Computer Graphics Forum 29*, 2, 545–554.

ROSE, C., COHEN, M. F., AND BODENHEIMER, B. 1998. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics and Applications 18*, 5, 32–40.

SHIN, H. J., AND LEE, J. 2006. Motion synthesis and editing in low-dimensional spaces. *Journal of Visualization and Computer Animation 17*, 3-4, 219–227.

SHIN, H. J., AND OH, H. S. 2006. Fat Graphs: Constructing an interactive character with continuous controls. In *Symposium on Computer Animation*, ACM/Eurographics, 291–298.

TITSIAS, M. K., AND LAWRENCE, N. D. 2010. Bayesian gaussian process latent variable model. *Journal of Machine Learning Research 9*, 844–851.

TREUILLE, A., LEE, Y., AND POPOVIĆ, Z. 2007. Near-optimal character animation with continuous control. *ACM Transactions on Graphics 26*, 3, 7:1–7:8.

URTASUN, R., FLEET, D. J., HERTZMANN, A., AND FUA, P. 2005. Priors for people tracking from small training sets. In *Proc. ICCV*, IEEE, 403–410.

URTASUN, R., FLEET, D. J., GEIGER, A., POPOVIĆ, J., DARRELL, T. J., AND LAWRENCE, N. D. 2008. Topologically-constrained latent variable models. In *Proc. ICML*, ACM, 1080–1087.

WALDER, C., KIM, K. I., AND SCHÖLKOPF, B. 2008. Sparse multiscale Gaussian process regression. In *Proc. ICML*, ACM, 1112–1119.

WANG, J. M., FLEET, D. J., AND HERTZMANN, A. 2007. Multifactor Gaussian process models for style-content separation. In *Proc. ICML*, ACM, 975–982.

WANG, J. M., FLEET, D. J., AND HERTZMANN, A. 2008. Gaussian process dynamical models for human motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence 30*, 2, 283–298.

WEI, X. K., MIN, J., AND CHAI, J. 2011. Physically valid statistical models for human motion generation. *ACM Transactions on Graphics 30*, 3, 19.

YE, Y., AND LIU, C. K. 2010. Synthesis of responsive motion using a dynamic model. *Computer Graphics Forum 29*, 2, 555–562.

ZHAO, L., AND SAFONOVA, A. 2009. Achieving good connectivity in motion graphs. *Graphical Models 71*, 4, 139–152.

A Constraints

The basis of the constraints described in this appendix is inverse kinematics. In an inverse kinematics constraint, we wish to place joint i at a location $\mathbf{p}$. The form of the constraint is a quadratic penalty for deviation from this location:

$$C(\mathbf{y}_t) = \|\mathcal{F}(\mathbf{y}_t)_i - \mathbf{p}\|^2,$$

where $\mathcal{F}$ is the forward kinematics function that gives the position of joint i for the pose $\mathbf{y}_t$. The gradient of this constraint can be obtained with the chain rule using the Jacobian $\mathbf{J}$ of $\mathcal{F}$. We can also use the Jacobian to linearize the constraint as

$$\mathbf{J}\Delta\mathbf{y} = \mathcal{F}(\mathbf{y}_{t-1})_i - \mathbf{p},$$

where $\Delta\mathbf{y} = \mathbf{y}_t - \mathbf{y}_{t-1}$. When the constraint violation in the previous frame is small, the linearized form is sufficient. However, we found that even the nonlinear variant can be optimized sufficiently fast to allow the controller to run in real time at 30 Hz.

Foot skate cleanup can be achieved by using the IK constraint to fix the foot to remain in place during contacts. Previous methods have suggested using annotated example clips to detect foot contacts [Lee et al. 2010], which is compatible with our approach. To avoid annotating the training data, we instead use a simple heuristic to detect foot contacts and apply the constraint. We initiate a contact when the foot's height and velocity fall below fixed thresholds. This contact is broken only if its desired velocity (according to the conditional mean) has a positive vertical component and falls inside a vertical cone, indicating that the foot is being lifted.

We can also use the IK formulation to constrain the character's limbs from penetrating solid objects. We demonstrate this by constraining the punching controller to avoid penetrating a cylindrical target. The nonpenetration constraint is an IK constraint along the direction orthogonal to the surface at the point of contact. When the conditional mean penetrates the environment, we can approximate this constraint as an equality for fast linearization. When there is no penetration, the constraint is deactivated.

B Likelihood Gradients

In order to optimize the objective in Equation (4), we must compute its gradients with respect to $\mathbf{X}$ and the hyperparameters $\bar{\alpha}$, $\bar{\beta}$, $\mathbf{W}$, and $\mathbf{W}_{\dot{\mathbf{Y}}}$. The gradients of the GP terms $\mathcal{L}_{\mathbf{Y}}$, $\mathcal{L}_{\dot{\mathbf{Y}}}$, and $\Phi_D(\mathbf{X})$ are discussed in previous work [Grochow et al. 2004], and the gradients of the hyperparameter priors are straightforward to compute. The gradient of the connectivity term with respect to $\mathbf{X}$ is given by

$$\frac{\partial \Phi_C}{\partial \mathbf{X}} = w_c \sum_{ij} \frac{\partial \mathbf{K}^d_{ij}}{\partial \mathbf{X}} \frac{1}{\mathbf{K}^d_{ij}} = w_c \sum_{ij} \frac{\partial \left[\exp(\beta \mathbf{H})\right]_{ij}}{\partial \mathbf{X}} \frac{1}{\mathbf{K}^d_{ij}}$$

From [Kalbfleisch and Lawless 1985], the derivative of the matrix exponential requires the eigenvalues λ and eigenvectors $\mathbf{U}$ of the negative Laplacian $\mathbf{H}$, which we compute with a symmetric QR algorithm. The derivative is given by

$$\frac{\partial \left[\exp(\beta \mathbf{H})\right]}{\partial \mathbf{X}} = \mathbf{U}\mathbf{V}\mathbf{U}^{\mathrm{T}},$$

where $\mathbf{V}$ is defined as

$$\mathbf{V}_{ij} = \begin{cases} \mathbf{G}_{ij} \frac{e^{\beta\lambda_i} - e^{\beta\lambda_j}}{\lambda_i - \lambda_j} & \text{if } i \neq j \\ \beta \mathbf{G}_{ii} e^{\beta\lambda_i} & \text{if } i = j \end{cases},$$

and $\mathbf{G}$ is defined as

$$\mathbf{G} = \mathbf{U}^{\mathrm{T}} \frac{\partial \mathbf{H}}{\partial \mathbf{X}} \mathbf{U}.$$

The gradient of the Laplacian can be derived using the chain rule.