Realtime Style Transfer for Unlabeled Heterogeneous Human Motion

Shihong Xia[1] Congyi Wang[1] Jinxiang Chai[2]* Jessica Hodgins[3]
[1]Institute of Computing Technology, CAS [2]Texas A&M University [3]Carnegie Mellon University

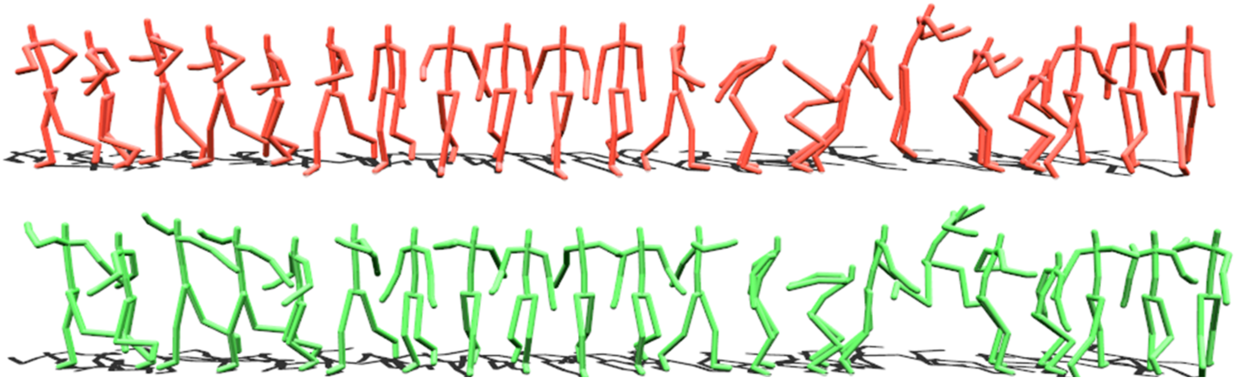

Figure 1: *Our realtime style generation system automatically transforms an unlabeled, heterogeneous motion sequence into a new style. (top) the input motion in the "neutral" style; (bottom) the output animation in the "proud" style. Note the more energetic arm motions and jump in the stylized motion.*

Abstract

This paper presents a novel solution for realtime generation of stylistic human motion that automatically transforms unlabeled, heterogeneous motion data into new styles. The key idea of our approach is an *online learning* algorithm that automatically constructs a series of local mixtures of autoregressive models (MAR) to capture the complex relationships between styles of motion. We construct local MAR models on the fly by searching for the closest examples of each input pose in the database. Once the model parameters are estimated from the training data, the model adapts the current pose with simple linear transformations. In addition, we introduce an efficient local regression model to predict the timings of synthesized poses in the output style. We demonstrate the power of our approach by transferring stylistic human motion for a wide variety of actions, including walking, running, punching, kicking, jumping and transitions between those behaviors. Our method achieves superior performance in a comparison against alternative methods. We have also performed experiments to evaluate the generalization ability of our data-driven model as well as the key components of our system.

I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—animation;

Keywords: Character animation, realtime style transfer, online local regression, data-driven motion synthesis

1 Introduction

Stylistic variations in motion that suggest a particular personality, mood or role are essential for storytelling and for bringing characters to life. For everyday actions such as locomotion, the style or emotion of a motion might convey more meaning than the underlying motion itself. For example, the anger conveyed by "he stalked across the room" is likely more significant than the act of crossing the room. Thus far, one of the most successful solutions to this problem is to model style differences between example motions and use the learned model to transfer motion from one style to another one (*e.g.*, [Amaya et al. 1996; Hsu et al. 2005]).

Hsu and colleagues [2005] introduced a linear time-invariant (LTI) model to encode style differences between example motions and achieved realtime performance for transferring the input motion from one style to another. Despite the progress made over the past decade, creating appropriate data-driven models for online style transfer remains challenging for two reasons.

First, a lifelike human character must possess a rich repertoire of activities and display a wide range of variations for a given action. This task inevitably requires data-driven models that can scale to large and heterogeneous motion data sets. However, most existing techniques are challenged when applied to input motion that contains heterogeneous behaviors such as *walking⇒running⇒jumping* because they assume that the relationship between the input and output styles can be approximated by a global linear model, such as the LTI model described by Hsu and colleagues [2005]. Global linear models are often appropriate for homogenous data sets (e.g., walking), but might not be suitable for complex heterogeneous motion data.

A second challenge is that previous work on style transfer often requires that the input motion is labeled in terms of behavior and style attributes before transferring it to a new style. However, automatically labeling the input motion in terms of behavior and style attributes remains difficult, particularly for online applications. The problem becomes even worse when the input motion cannot be classified into a single predefined behavior or style. For example, a "proud" walk performed by an old man might

*Contact author: jchai@cs.tamu.edu

display a hybrid style of “proud” and “old” and therefore cannot be classified into either “proud” or “old”.

This paper presents a novel solution that addresses both challenges (Figure 1). The key idea of our approach is an *online learning* algorithm that automatically builds a series of local mixtures of autoregressive (MAR) models to describe the differences between the input and output styles. We construct local regression models on the fly by searching for the closest examples of each input pose in the training database. Once the model parameters are estimated from the training data, the model transforms the input poses into the output style with simple linear transformations. A new local model is created to transform each successive pose. The local model avoids the problem of finding an appropriate structure for a global model, which would necessarily be complex and highly nonlinear for heterogeneous motion data.

We further extend the local regression model to mixtures of local regression models to automatically handle unlabeled input motion data. To achieve this goal, we annotated the training data in terms of behavior and style attributes in the preprocessing step. At run time, we construct local autoregressive models that correspond to every combination of predefined behaviors and styles and optimally predict the output poses by interpolating/extrapolating the prediction results obtained by each of local model.

We demonstrate the power and effectiveness of our method on a wide variety of human movements, including a wide range of walking data as well as heterogeneous actions containing walking, running, punching, kicking, jumping and transitions between those behaviors. We show that our method advances the state of the art by comparing it against alternative methods, including LTI models [Hsu et al. 2005] and Gaussian process models [Ikemoto et al. 2009]. The evaluation shows that our method obtains consistently better results than alternative methods for all types of test data. More significantly, our method can handle unlabeled, heterogeneous input motion sequences, a capability that has not been demonstrated in any previous work. Finally, we assess the generalization ability of our model and perform experiments to evaluate the key components of our system.

1.1 Contributions

Our approach to realtime style synthesis and control includes a number of technical contributions:

- A novel online local MAR model that automatically transforms an unlabeled heterogeneous motion data into different styles, a capability that has not been demonstrated in previous work.
- A simple yet effective online regression model that models the temporal differences between the input and output styles.
- An extension of our spatial-temporal model to handle input motion that differs significantly from the training motions. This extension significantly improves the generalization ability of our model.
- A style interpolation scheme that allows for realtime control of output styles by blending the parameters of distinctive styles on the fly.

2 Background

We construct a data-driven model to represent spatial-temporal differences between styles of motion and use it to transfer input motion data from one style to another. We therefore focus our discussion on stylistic motion generation and data-driven motion modeling.

Stylistic motion generation. One solution to stylistic motion synthesis is style translation [Amaya et al. 1996; Hsu et al. 2005; Shapiro et al. 2006; Ikemoto et al. 2009], which models style by examining differences between example motions and uses the extracted styles to transfer the input motion from one style to another one. Hsu and colleagues [2005] encoded the transformation between two motions with the same behavior but different styles as a linear time-invariant system. Once this system has been identified, the transformation can be applied to a new motion faster than real time, producing a corresponding motion in the learned style. Ikemoto and colleagues [2009] introduced a nonparametric model of the transformation between motion sequences using Gaussian process models of kinematics and dynamics and applied them to edit the input motion into a desired style.

Researchers have also explored data-driven approaches that build statistical motion models for interpreting motion variations caused by a single factor (*e.g.*, style) [Brand and Hertzmann 2000] or multiple factors (*e.g.*, gait, identity, and state) [Wang et al. 2007; Min et al. 2010]. For example, Brand and Hertzmann [2000] modeled the space of content and style as a parameterized set of hidden Markov models. Wang and his colleagues [2007] introduced Gaussian Process latent model with a multifactor kernel to capture stylistic variations of human motion. Min and his colleagues [2010] recently extended the idea to modeling both “style” and “identity” of human motion. They used a large corpus of preregistered motion data to construct a multilinear motion model to explicitly model both “style” and “identity” variations in the same action.

Our method is most closely related to the work of Hsu and colleagues [2005]. Both methods aim to model the relationship between styles of motion using example motion data and use the learned model to rapidly transform the input motion into different styles. Our motion model, however, is significantly different from theirs because we model the style differences using a series of local mixtures of autoregressive models rather than the linear time-invariant model adopted in their system. One advantage of our online local models is the ability to handle unlabeled, heterogeneous motion data. Section 7.2 shows that our model produces more accurate results than their approach and can handle unlabeled motion data.

Data-driven motion modeling. Our work builds upon a significant body of recent work that utilizes prerecorded data for human motion modeling. One solution is weighted interpolations of motion examples [Rose et al. 1998; Kovar and Gleicher 2004]. Motion interpolation registers a set of structurally similar but distinctive motion examples and then parameterizes them in an abstract space defined for motion control. Given the control parameters’s new values, the sequences can be smoothly blended with appropriate kernel functions such as radial basis functions. Statistical models provides an appealing alternative for human motion modeling because they are compact and can be used to generate an infinite number of motions that are not in the prerecorded motion data. Recently, a wide variety of statistical motion models have been developed and their applications include motion synthesis [Chai and Hodgins 2007; Lau et al. 2009; Min and Chai 2012], inverse kinematics [Grochow et al. 2004], and performance-based animation [Chai and Hodgins 2005].

However, none of these approaches focuses on modeling spatial-temporal differences between styles of motion, a goal targeted by our paper. In addition, our model is significantly different because we use a series of local mixtures of autoregressive models to model the transformation between the input and output

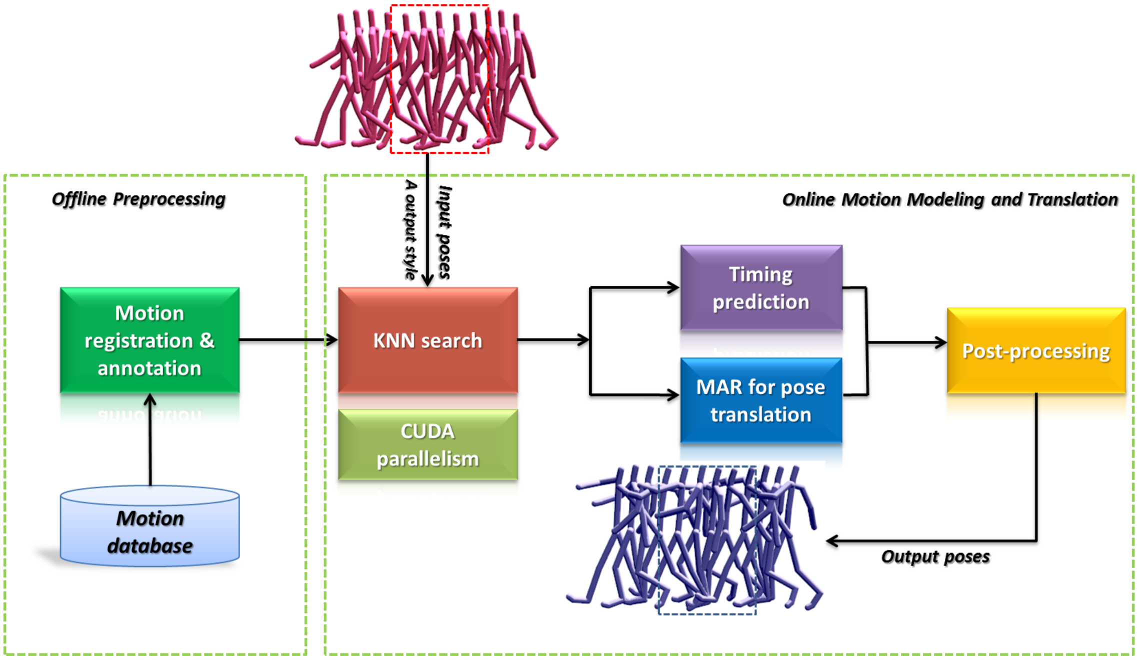

Figure 2: *System overview.*

styles.

3 Overview

Our goal is to transform input motion data into a sequence of frames in the output style. We assume that the input frames arrive one at a time and are processed in an online manner. We also assume that the input frames do not have labels for action or style attributes. Our system contains three major components (Figure 2):

Motion registration and annotation. Our method is data-driven because it utilizes the prior knowledge embedded in a prerecorded motion database to do style transformation. We therefore discuss how to preprocess the prerecorded motion data for data-driven style modeling. For each action in the database, we register all of the motion examples against each other and annotate them in terms of "action" and "style" attributes. In addition, all the database examples are annotated with contact information and this information allows us to remove foot sliding artifacts in the output animation.

Stylistic motion modeling and generation. The key challenge here is to model the spatial-temporal relationship between styles of motion from the preprocessed motion database. The problem is challenging because the transformation is often complex and highly nonlinear and the input motion is often unlabeled. Our key idea is to use an *online learning* algorithm to automatically construct a series of local regression models to approximate the spatial-temporal transformation at each frame. Specifically, we construct local regression models on the fly from the closest examples of each input pose in the database. We further extend the local regression models to mixture of local regression models in order to handle the unlabeled input motion. In addition, we discuss how to extend our style translation method to handle input motion data that differs significantly from the training motions.

Postprocessing. The synthesized motion often violates environmental contact constraints. We introduce an efficient classification algorithm to automatically detect footplant constraints in the input motion. We utilize the detected footplant constraints to remove foot sliding artifacts in output animation.

We describe these components in more detail in the next three sections.

4 Motion Registration and Annotation

We performed a series of offline captures to create a large and heterogeneous human stylistic motion database of about 11 minutes using a Vicon optical motion capture system [Vicon 2015] with eighteen 120 Hz cameras. The database consists of a wide variety of human actions, including walking, running, jumping, kicking, punching and transitions between those behaviors. For each action, we recorded motion examples corresponding to eight distinctive styles: neutral, proud, angry, depressed, strutting, childlike, old, and sexy. Except for the root joint, all joint angles are converted to Cartesian parameters with the exponential-map parameterization [Grassia 1998]. This representation ensures proper manipulation of the joint-angle quantities required for style translation.

We register all of the motion examples corresponding to the same action against each other because they are structurally similar. For example, we can pick a "neutral" walk as a reference motion and use it to register all of the walking motions in the database with appropriate time warping functions. In our implementation, we register the motion examples in a translation- and rotation-invariant way by decoupling each pose from its translation in the ground plane and the rotation of its hips about the vertical axis [Kovar and Gleicher 2003].

We annotate all of the registered motion examples in terms of "action" and "style" attributes. Motion annotation is important because it allows us not only to model heterogeneous motion data but also to transform an input motion without knowing its action and/or style. In our experiments, style and action annotations were achieved without any manual intervention because we instructed the subject to perform each motion with a particular action and style during the motion capture session.

In addition, we annotate the motion examples with contact information. Contact annotations were achieved by annotating the canonical timeline (*i.e.*, the timeline of the reference motion) of each action. For example, annotating the "walking" examples is achieved by annotating four contact instances on the canonical timeline ("left foot down", "left foot up," "right foot down", and "right foot up"). In our implementation, we encode the contact information of each frame using a binary feature vector. Each bit represents a particular type of contact event, *e.g.,* the left foot plants on the ground. Each frame requires only a 2-bit label (left foot and right foot). Contact annotations enable us to automatically enforce environmental contact constraints in the style translation process, thereby eliminating noticeable visual artifacts such as foot sliding in the output animation. The entire process for annotating contacts takes several minutes for a 11 minutes motion capture database because we only need to annotate the canonical timeline of each action.

5 Stylistic Motion Modeling and Translation

We now describe how to utilize the preprocessed motion database to translate an input motion into a sequence of frames in the output style. Section 5.1 introduces the online regression models that represent the key idea in our approach. We assume that the input motion data are labeled in terms of action and style attributes. In Section 5.2, we extend the method to handle unlabeled input motion data. We then discuss how to further extend our data-driven model to handle input motion data that differs significantly from the training motions in Section 5.3. Section 5.4 introduces our style interpolation scheme that allows for realtime style control in the output animation.

5.1 Online Regression Models

We present an *online learning* algorithm [Aha 1997; Chai and Hodgins 2005] to automatically construct a series of local regression models to approximate the spatial-temporal relationship between the input and output styles. The online learning algorithm postpones all computation until an explicit request for information (*e.g.*, prediction) is received. To predict the output style at the current frame, we first search the preprocessed database for examples that are close to the current input frame. These examples are then used as training data to learn a regression model for predicting the pose and timing of an output frame. Once its model parameters are estimated from the training data, the model translates the current pose with simple linear transformations. A new regression model is created to translate each successive pose. Therefore, our regression model is time-varying because its model parameters are functions of time.

KNN search. Constructing a time-varying regression model requires identifying the K nearest neighbors (KNN) in the preprocessed database for each input pose. We perform KNN searches only on the database examples that are annotated with the same style and action as the input frame. We compute the distance between the input pose and a database example based on the poses formed over two sliding windows of frames of a user-defined length s. The use of the temporal window effectively incorporates derivative information into the distance metric. In our experiment, s is set to 5 frames or 1/24 second.

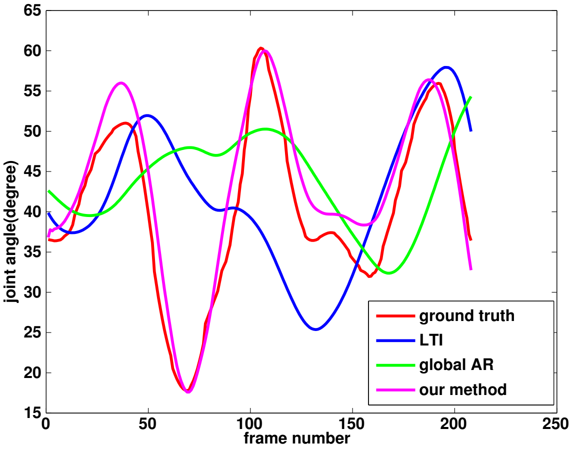

Figure 3: *Comparing our online local model against LTI and global autoregressive model (GAR): the comparison curves show the values of one DOF (right humerus) from all three methods and ground truth data over time. The evaluation is based on cross validation described in Section 7.2.*

We compute the distance between two skeletal poses in terms of a point cloud driven by the skeleton [Kovar et al. 2002]. Mathematically, we calculate the minimal weighted sum of squared distances given that an arbitrary rigid 2D transformation may be applied to the second point cloud:

$$\min_{\theta,x_0,z_0} \sum_i w_i \|\mathbf{p}_i - \mathbf{T}_{\theta,x_0,z_0}\mathbf{p}'_i\|^2, \tag{1}$$

where $\mathbf{p}_i$ and $\mathbf{p}'_i$ are 3D positions corresponding to the query and database poses. The linear transformation $\mathbf{T}_{\theta,x_0,z_0}$ rotates a point $\mathbf{p}'_i$ about the y (vertical) axis by θ degrees and then translates it by (x_0,z_0) on the floor plane. The weight w_i may be chosen both to assign more importance to certain joints and to taper off towards the end of the window. When the skeleton structure of the input motion data is different from the skeletal model of our training data, we perform a motion retargeting step for the input motion using the method described in [Monzani et al. 2000].

Pose translation. We now describe how to use the closest examples of the current frame to construct an efficient online regression model for style translation. Our model constrains the input-output mapping to a specific form; namely, it assumes that each degree of freedom of the skeletal pose is independent of the others. This idea is motivated by the success of previous work on style translation [Amaya et al. 1996; Hsu et al. 2005] that processes each degree of freedom of input motion data separately.

For each degree of freedom, we adopt a time-varying autoregressive model to explain the relationship between the input and output styles:

$$y_t = \mathbf{u}_t^T \beta_t + \varepsilon_t, \tag{2}$$

where x_t and y_t are the input and output variables at frame t. These equations describe the mapping between an input $\mathbf{u}_t = [x_{t-1}, y_{t-1}, x_t, 1]^T$ and an output y_t. The model parameters $\beta_t = [a_t, b_t, c_t, d_t]^T$ determine the mapping between the input and the output at the current frame. $\varepsilon_t \backsim N(0,\sigma)$ is a white noise term that accounts for measurement errors.

Given the K closest examples for the current pose, we estimate the model parameters using a linear least squares regression. We introduce a regularization term, $\lambda\|\beta_t\|^2$, to reduce ambiguity. Once

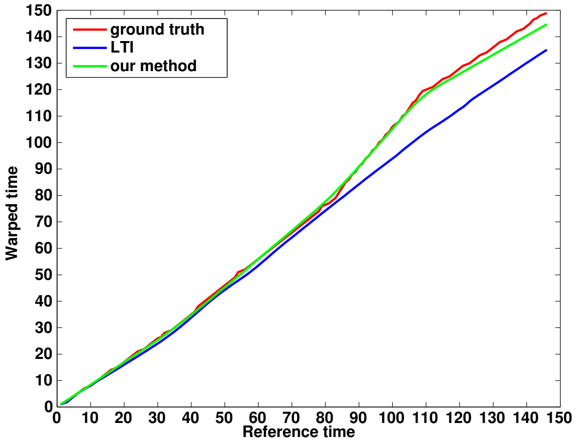

Figure 4: *Comparing our method against LTI for timing prediction: the comparison curves show the time warping functions from our method, LTI and ground truth data. The evaluation is based on cross validation described in Section 7.2.*

the model parameters are estimated, we can use the model to translate the current pose into the output pose with the simple linear transformations described in Equation (2). The local model avoids the problem of finding an appropriate structure for a global model, which would necessarily be complex and highly nonlinear for heterogeneous actions. Instead, we assume that a series of local autoregressive models are sufficient to approximate a complex, nonlinear function for style translation. Figure 3 shows the advantage of our method over two alternative methods: LTI [Hsu et al. 2005] and a global autoregressive model.

We represent the input poses in terms of the original timeline of the input motion data while the output poses are defined in the canonical timeline of the reference motion data in the preprocessed database. The output poses therefore are synthesized in the canonical timeline of reference motion data and we must predict the timing for the output poses.

Timing prediction. Timing prediction is crucial to style translation because different styles of the same action often differ significantly in speed. Our task here is to predict the time difference $w(n) = t(n) - t(n-1)$ between the current and previous poses in the output style. As with the pose prediction, we apply K-nearest neighbor interpolation to predict the timing of the current pose in the output style. For the current pose, we first compute its distance to K closest examples in the database and then assign each of closest examples a weight based on their distance to the current pose. These weights are then applied to predict the timing difference of the current pose in the output style by interpolating the corresponding timing differences of the closest examples in the database. We choose Gaussian functions as our interpolation kernel. The larger the distances, the smaller the weights. We normalize the weights to ensure they sum to one.

A remaining issue for timing prediction is how to choose an appropriate distance metric for KNN searches. Our distance metric considers both the input and output styles. Note that before timing prediction, we have already synthesized the current pose in the output style. We define the distance metric based on a combination of four features, including the current pose and pose velocity of the input frame, the synthesized pose of the output style, and the synthesized pose velocity of the output style in the canonical timeline. Figure 4 shows a cross validation result on our method and LTI (please refer to the details in Section 7.2).

5.2 Handling Unlabeled Motion Data

We now extend the online local regression model described in Section 5.1 to handle unlabeled input motion. We use the annotation information in the training data to achieve this goal. Briefly, we annotate the training data in terms of behavior and style attributes in the preprocessing step. At run time, we construct local autoregressive models corresponding to every combination of predefined behaviors and styles and predict the output poses by interpolating/extrapolating the prediction results obtained by each of local model in a probabilistic framework of mixture of regression models.

Mixture of regression models [Wong and Li 2000] provides a way to model the regression function when it arises from a number of a priori known classes $j = 1,...,c$ with unknown proportion λ_j. We model the complex relationship between the input and output styles using the following MAR model:

$$f(y_t|\mathbf{u}_t) = \sum_{j=1}^{c} \lambda_j \phi(y_t|\mathbf{u}_t^T \beta_t^j, \sigma_j^2). \qquad (3)$$

These equations describe the relationship between an input $\mathbf{u}_t = [x_{t-1}, y_{t-1}, x_t, 1]^T$ and an output pose y_t, where $\lambda_j, j = 1,...,c$ are the mixing proportions, each of which is a probability (a real number between 0 and 1, inclusive), all of which sum to 1. β_t^j and σ_j are the autoregressive model parameters and variances of the j-th class. Note that $\phi(\cdot|\mu,\tau^2)$ denotes the normal density with mean μ and variance τ^2.

We define the number of a priori known classes based on the annotation information in the preprocessed database. Each frame in the training data belongs to a particular class, which is associated with a particular combination of action and style. When an explicit request for prediction is received, we search the K nearest neighbors in each class $j = 1,..,c$ and use them to construct local autoregressive models $y_t = \mathbf{u}_t^T \beta_t^j + \varepsilon_t^j, j = 1,...,c$ corresponding to each class using the least square regression technique described in Section 5.1.

The proportion weights λ_j are prior probabilities of a particular class j. We compute the prior probability of a particular class j based on the average distance between the current frame and its K nearest neighbors:

$$\lambda_j \propto \exp(-\frac{D_j}{\delta \min_j D_j}), \quad j = 1,...,c, \qquad (4)$$

where D_j is the average distance between the current frame and its K nearest neighbors. The kernel width δ is experimentally set to 0.7. We normalize the proportion weights λ_j so that they sum to 1.

Once we learn the mixture of autoregressive models, we can use it to predict the pose in the output style. This prediction is achieved by calculating the expectation of the random variable y_t:

$$\begin{aligned} E(y_t) &= \int y_t f(y_t|\mathbf{u}_t) \mathrm{d}y_t \\ &= \int y_t \textstyle\sum_{j=1}^{c} \lambda_j \phi(y_t|\mathbf{u}_t^T \beta_t^j, \sigma_j^2) \mathrm{d}y_t \\ &= \textstyle\sum_{j=1}^{c} \lambda_j \int y_t \phi(y_t|\mathbf{u}_t^T \beta_t^j, \sigma_j^2) \mathrm{d}y_t \\ &= \textstyle\sum_{j=1}^{c} \lambda_j \mathbf{u}_t^T \beta_t^j. \end{aligned} \qquad (5)$$

The last derivation step is based on the fact that the expectation of a normal distribution $\phi(\cdot|\mu,\tau^2)$ is its mean μ. The same idea is used to predict the timing of each output pose.

We apply a similar idea to translate partially labeled motion data into the output styles. For example, if the input motion data is “neutral”, we can search “neutral” examples corresponding to each action (*e.g.*, neutral walking or neutral running) and use K nearest neighbors in each action to learn a mixture of autoregressive models. The output style is synthesized by appropriately interpolating the prediction poses corresponding to each class. In addition, to handle unlabeled/partially labeled motion data, we also extend our timing prediction model to a mixture of KNN interpolation models in the same way as the MAR model.

5.3 Autoregressive Model with Bias

When the input motion is significantly different from the motions in the database, the system may output motions with artifacts. The ideal solution, of course, is to expand the training set, but that is often impossible. Here we describe how to extend our autoregressive model by blending the original input into the output of the prediction model when such events occur.

The key idea of our autoregressive model with bias is to introduce a confidence interval $(\mu - \kappa\sigma, \mu + \kappa\sigma)$ for a regression model, where κ defines the confidence interval, μ is the mean of the training inputs, and σ is the standard deviation of the training inputs. If the input variable x_t falls inside the interval, we assume the trained regression model is reliable and can be used to accurately predict the output style. When the input variable x_t lies outside of the interval (*e.g.*, $x_t \leq \mu - \kappa\sigma$), we deem the prediction not reliable. This observation leads us to introduce a bias term Δx to move the input variables inside the interval so that the prediction will be reliable:

$$\begin{aligned} x_t &= x_t - \Delta x \\ x_{t-1} &= x_{t-1} - \Delta x, \end{aligned} \tag{6}$$

where $\Delta x = x_t - \mu \pm \kappa\sigma$ represents the bias between the input variable x_t and the boundary of the confidence interval $\mu \pm \kappa\sigma$. We include the same bias term into the input variable in the previous frame x_{t-1} in order to preserve derivative information in the input motion. However, introducing the bias term to the input variables x_t and x_{t-1} causes a loss of the original styles in the input variable. To address the issue, we introduce a corresponding bias term $\Delta y = \rho\Delta x$ to the output variables:

$$\begin{aligned} y_t &= y_t - \rho\Delta x \\ y_{t-1} &= y_{t-1} - \rho\Delta x, \end{aligned} \tag{7}$$

where ρ is a value between 0 and 1 indicating the preference of style translation and content preservation. When ρ is set to “0”, we focus on the translation of the style. If the user prioritizes the preservation of the content, she can set ρ to a higher value (*e.g.*, “1”). In practice, the user can choose an appropriate value for ρ to balance the trade-off between content preservation and style translation. In Section 7.4, we evaluate both the importance of the bias term and the influence of different values of ρ on the resulting animation.

5.4 Style Interpolation and Control

Style control is important to stylistic motion synthesis because the emotions or roles of a character can be quite different and often change over time. Our system allows the user to control the style of the output animation at run time by blending the parameters of distinctive styles. For example, we can translate a “neutral” walk into an “angry”-“strutting” walk by appropriately interpolating the two output styles: “angry” and “strutting”. In our implementation, we represent the output style as a non-negative combination of styles in the database. This representation enables us to define the output style using a continuous vector $\mathbf{s}(t) = [s_1(t), s_2(t), \ldots, s_8(t)]^T$ that stacks the blending weights of each style. The user can control the interpolation weights to interpolate two or three distinctive styles (see the accompanying video). Figure 5 shows our visualization interfaces for interpolating distinct styles.

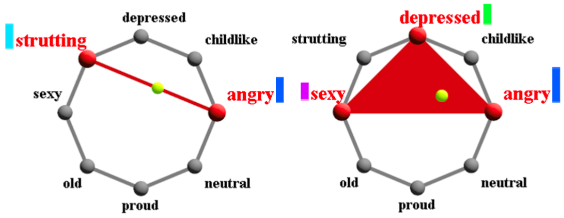

Figure 5: Online style control: (left) interface for interpolating two distinctive styles; (right) interface for interpolating three distinctive styles. The user can select which vertices to interpolate as well as their interpolation weights at run time. Note that each polygon vertex defines a distinctive style. The red vertices are interpolation vertices. The yellow bars show the scales of interpolation weights.

6 Postprocessing

Given a good training set, our model outputs high-quality motions, but it may violate kinematic constraints imposed by the environment. The most visible artifact is footskate, which can be corrected by existing methods if the footplants are annotated [Kovar et al. 2002]. We introduce a simple yet effective classifier to automatically annotate the footplant constraints in input/output motion data.

We apply KNN classification techniques to automatically label the contact information of the input motion data. To annotate each input frame, we first search for the K nearest neighbors in the database. In the preprocessing step, each database pose was annotated using classes defined by binary footplant annotations, where “1” indicates that footplant is “on” and “0” means “off”. We compute the distance using the metric described in Equation (1), except that we now set the weights of upper body joints to 0. We evaluate the likelihood of footplant constraints being “on” or “off” by weighing each instance’s class based on the inverse of its distance to the query pose. We reduce classification noise by applying Gaussian filters to smooth the likelihood values. The system assumes that the footplant constraint is “on” when the likelihood of footplant constraints being “on” is higher than the likelihood of being “off”. If there is contact between the character and the environment (*e.g.*, the left foot must be planted on the ground), we apply an inverse kinematics technique to ensure that the contact point on the character is located on the corresponding contact plane.

7 Results

We demonstrate the power and effectiveness of our method on a wide variety of human movements (Section 7.1). In Section 7.2, we compare our method against alternative methods for online style transfer, including LTI models [Hsu et al. 2005] and Gaussian process models [Ikemoto et al. 2009]. Section 7.3 evaluates the generalization ability of our model. Finally, we perform experiments to evaluate the key components of our model in terms of the importance of local mixtures of autoregressive models, the benefit of the bias term and with/without the use of labeling (Section 7.4). Our results are best seen in the accompanying main video as well as the evaluation video.

Action	Length (frames)
A	49461
B	7835
C	6293
D	7125
E	6970
F	2145
Total	79829

Table 1: *Details of our stylistic motion database. A: walking with different step sizes, speeds and turning angles; B: running with different speeds and turning angles; C: jumping; D: punching; E: kicking; F: transition actions.*

Table 1 shows the details of our stylistic motion database. Each action listed in the table contains eight styles, and the total length of the database is about 11 minutes (79829 frames). The input to our style translation system is very flexible and can be in various forms. We have tested our system on input motion data captured by Vicon [2015], animation generated by motion synthesis techniques (*e.g.*, motion graphs [Kovar et al. 2002]), and key-framed animation created by an artist.

Our system is appealing to online style transfer because it has very few parameters to be tuned. Specifically, the entire system has three key parameters. For all the results reported in the paper, the number of nearest neighbors (K) is set to 40, the bias value (ρ) is set to 0.7 and the regularization weight (λ) is set to 0.1.

Computation times. Our system achieves real-time performance on a PC with Intel(R) Xeon(R) processor E3-1240 3.40GHz and NVIDIA graphic card GTX 780T (3GB). Table 2 summarizes computation times per frame for all the key components of our system. Our KNN search process is based on the entire prerecorded motion database. In the current implementation, we speed up the KNN searching process and the pose regression process with GPU acceleration while the rest components of the system are implemented on CPU. The system requires constant memory at run-time. For the current size of our training database, the memory usage for GPU and CPU is about 200Mb and 90Mb, respectively.

7.1 Test on Real Data

We have tested our method on heterogeneous motion data containing walking, running, jumping, kicking, punching and their transitions and on walking with a wide range of motion variations. In addition, we show that our system allows the user to control output style of human motion on the fly by blending the parameters of distinctive styles. For all the results shown here, we assume that both the content and style of the input motion data are not known.

Heterogeneous actions. The goal of this experiment is to evaluate the performance of the system when handling heterogeneous motion data. Figure 1 shows the results for a test motion sequence with heterogeneous actions, including running, walking, jumping and walking. We also transformed an input motion sequence (walking, punching, kicking, jumping to walking) into different styles. The results in the accompanying video show that the system produces high-quality animation in the output styles.

Homogeneous actions. We applied various styles to new walking sequences with different speeds, step sizes, turning angles, and subjects. Our first evaluation used a walking motion ("walk along a circle") generated by motion graphs [Kovar et al. 2002]. Next, we translated a walking motion containing big steps, sharp turning, and speeding up into various styles. We also tested our system on walking data obtained from different human subjects downloaded from the CMU mocap database. Our system yielded consistently good results that exhibited transfer of large-scale content and fine detail.

Component	Time (ms)	Ratio (%)
KNN search	7.31	40.4%
MAR for spatial-temporal regression	9.7	53.60
Post-processing	1.09	6%
Total	18.1	100%

Table 2: *Computation times for each component of our system.*

Online style interpolation and control. In this experiment, we translated a "neutral" walking motion into various mixed styles such as "old-proud" walking and "depressed-angry-sexy" walking. The accompanying video shows the user can continuously control the output style at run time. The test motion data, which was generated by motion graphs [Kovar et al. 2002], consists of a walking motion with random changes in speed and direction.

7.2 Evaluation and Comparison

In this experiment, we compare our method against alternative methods for style transfer, including LTI models [Hsu et al. 2005] and Gaussian process (GP) regression models [Ikemoto et al. 2009]. We assume that both the behavior and style of test motion data are known in advance as such labeling information is required by both alternative methods.

7.2.1 Comparison Against Linear Time-invariant Model

We have compared our method against the style translation method proposed previously [Hsu et al. 2005]. We implemented the style translation algorithm described by Hsu and colleagues [2005]. Briefly, we applied iterative motion warping (IMW) to automatically compute dense correspondences between example motions and applied LTI models constructed from the preprocessed training examples to translate input motion data into various styles. In addition, we used the post-processing techniques described in their paper to improve the quality of the output animation. Similar to Hsu and colleagues [2005], our method also used IMW to register training examples.

Our evaluation is based on leave-one-out cross-validation on both homogeneous and heterogeneous data sets. Specifically, we used a single motion sequence from the data sets as the validation data and the remaining sequences as the training data. This test is repeated such that each sequence in the whole data set is used once as validation data. We computed the synthesis errors for both pose and timing. We measured the pose error as the average 3D joint position discrepancy between the synthesized poses and the ground truth poses. The temporal error was computed as the average temporal difference between the synthesized timings and the ground truth timings. Figure 6 shows the cross validation results for both methods.

Leave-one-out evaluation on homogeneous data set. We have evaluated the performance of both methods on two different data sets of walking. We first tested the methods on the MIT walking data used by Hsu and colleagues [2005]. We downloaded three cycles of walking data ("normal" and "crouch") and used them to perform a leave-one-out evaluation for both methods. Next, we compared our method against LTI on our own data sets. Specifically, we tested both methods on translating walking data

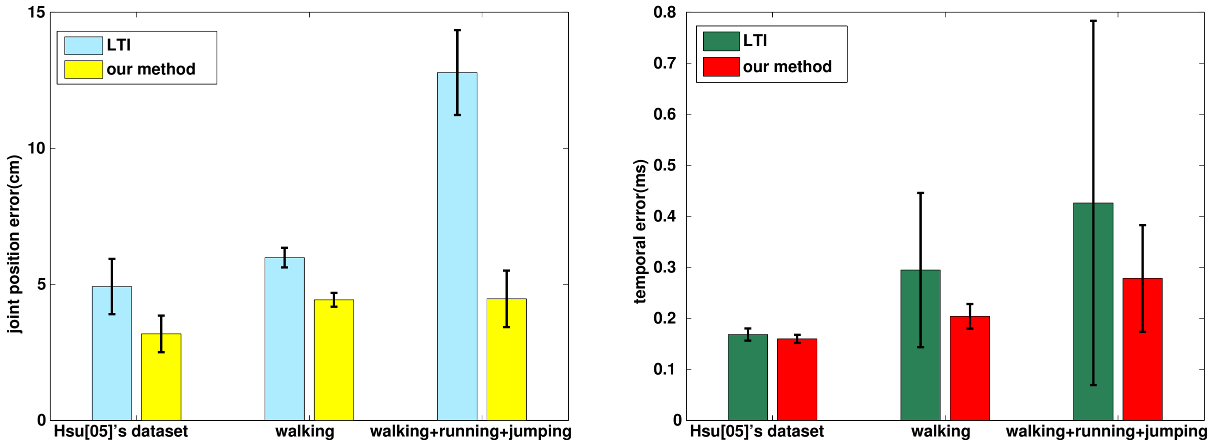

Figure 6: *Leave-one-out cross-validation on our method and LTI: (left) comparisons of pose synthesis errors; (right) comparisons of timing prediction errors. The evaluation is based on three different data sets: the MIT data set, our walking data set, and our heterogeneous data set of walking, running, and jumping.*

from the "neutral" style into the "proud" style via cross-validation. Figure 6 shows that our method produces more accurate synthesis results for both pose and timing.

Leave-one-out evaluation on heterogeneous data set. A major benefit of our method is in the ability to handle heterogeneous input motion data. This experiment shows the ability of both methods to translate heterogeneous motion data into a different style. We evaluated both methods on heterogeneous motion data from walking, running and jumping. Our method produces much more accurate results than LTI (see the accompanying video).

Comparison on real data. We have also applied both methods to translate real heterogeneous motion data into various styles. The evaluation video shows the comparison results for two sequences: *running⇒walking* and *running⇒walking⇒jumping⇒walking*. Our method produces high-quality animation for both sequences while LTI fails to obtain satisfactory results for either sequence.

7.2.2 Comparison Against Gaussian Process Regression

We compared our method with a Gaussian process regression model [Ikemoto et al. 2009]. In particular, Ikemoto and colleagues [2009] used Gaussian process regression to learn the transformation between the input and output motion sequences to solve the generalized motion editing problem. In addition, they used a stabilized predictor to handle input data that is not in the training database. For comparison, we adopted a similar idea for solving the style translation problem. Specifically, we applied the GP algorithm to model the motion differences between the input and output styles and used the learned GP regression model as well as the stabilized predictor to synthesize stylized output animation. Our GP implementation is based on the GPML toolbox [Rasmussen and Nickisch 2010].

We have tested both methods on homogeneous and heterogeneous motions via leave-one-out evaluation. For the first experiment, the training examples are one pair of "proud"/"neutral" walking data (236 frames and 241 frames, respectively). As shown in the evaluation video, both methods produce high-quality output animation. We also tested on heterogeneous data sets including walking, running, jumping, and transition actions from jumping to walking. The evaluation video shows that GP regression fails to produce satisfactory results, while our method does. One reason for this difference in performance is that Gaussian process regression scales poorly with large quantities of training data from heterogeneous data sets. GP regression is well suited for their task of artistic motion editing because they do not have large quantities of training data.

Type	LHom	LHet	UHom	UHet
Hsu et al.	G	P	N/A	N/A
Ikemoto et al.	G	P	N/A	N/A
Our method	G	G	G	G

Table 3: *Summary of the comparison against alternative methods. LHom: labeled homogeneous input motion; UHom: unlabeled homogeneous input motion; LHet: labeled heterogeneous input motion; UHet: unlabeled heterogeneous input motion; P: poor results; G: good results; N/A: results are not available because the algorithm cannot handle unlabeled input motion data.*

7.2.3 Comparison Summary

We now summarize our two comparisons (see Table 3). Both LTI [Hsu et al. 2005] and GP regression [Ikemoto et al. 2009] assume that input motion is labeled in advance. Therefore, they cannot effectively handle "unlabeled" input motion sequences (homogeneous or heterogeneous). For heterogeneous input motion data ("labeled" or "unlabeled"), neither method can achieve satisfactory results. The evaluation shows that our method obtains consistently better results than the alternative methods for all of our test data. More significantly, our method can handle "unlabeled" and/or "heterogeneous" input motion sequences, a capability that was not been demonstrated in any previous method.

7.3 Evaluation on Motion Generalization

One advantage of our model is the ability to handle motion data that differs from the prerecorded motion data. We tested our system on "dinosaur" walking downloaded from the CMU mocap database and "robotic" locomotion keyframed by an artist (see the accompanying video). Note that both motion sequences differ stylistically from motions in the database. We assume both motion sequences are unlabeled in terms of style and action attributes. The accompanying video shows that the system generates high-quality animation in various styles.

To further evaluate the generalization ability of our model, we tested two more motions that differ from those in the training database. The first one is a motion obtained from the CMU mocap database called “exaggerated walking”. It was used as an input motion with the style unlabeled and the action labeled as “walking”. Figure 7(a) shows the closest poses in the database for a few sample frames of the input motion. As shown in the evaluation video, the synthesized motion not only successfully changes the style but also preserves the “exaggerated” features of the original motion. Another test motion is also from the CMU mocap database, called “mummy walking”, had the style unlabeled and the action labeled as “walking”. Figure 7(b) shows the nearest neighbors in the database for a few sample frames. Those poses are not similar, indicating that the input motion is very different from the training data. Our method still yields a satisfactory stylistic motion, as shown in the evaluation video. The “mummy” indeed appears “proud”,“sexy”, and “depressed”.

In our last motion generalization evaluation, we attempted cross-action style translation. The evaluation video shows our results for using a “walking” database to transfer styles for “running” and “jumping” and using a “running” database to transfer styles for “walking” and “jumping”. Figure 7(c) and Figure 7(d) show the three nearest neighbors of some sample frames for “walking” and “jumping” sequences when the training data is “running”. This evaluation shows that our method can still generate styles even when the input action is not in the training data.

7.4 Component Evaluations

In this section, we evaluate the key components of our style modeling and synthesis process. Our results are seen in the accompanying evaluation video.

Importance of the MAR model. To justify the importance of local mixture models, we compared our model against classic local regression models, including Nearest Neighbors (NN), KNN interpolation (KNN), and a local autoregressive model. The evaluation video shows the comparison against alternative local regression models on two input motion sequences obtained from the CMU mocap database. The first one is called “run and leap”, and the second one is called “Mickey walking”. We process them with both action and style unlabeled. Neither motion is the training database in those styles, although the database does contain other runs, leaps and walks. We used the same number of nearest neighbors for all methods. As shown in the accompanying evaluation video, our method produces superior results when compared to the other local models.

Benefit of the bias term. The bias term is crucial for input motion data that are outside the space explored in the training data. To demonstrate the importance of the bias term to our model, we compared the synthesis results using the MAR model with and without the bias term. The accompanying video shows a side-by-side comparison between the two. Without the bias term, the system failed to produce high-quality output animation. With the bias term, the output animation can not only transfer the style but also preserve the content of the original motion, thereby producing satisfactory results.

Evaluation of different bias values. We have evaluated the influence of different bias values on the output animation. In this experiment, we set the bias values to 0.0, 0.3, 0.7, and 1.0, respectively. The input sequence is a “marching” motion from the CMU mocap database, with the action labeled as “walking” and the style unlabeled. We transferred the input motion into a “proud” style. The accompanying video shows synthesized stylistic motions corresponding to different bias values. Setting the bias value to 0.0 focuses on style translation; the output animation therefore loses some fine details of the original “marching” motion. As we gradually increase the bias value, the output animation exhibits more of the details of the input motion, especially for the lower body. For all of the results shown in the paper, we set the bias value to 0.7, which keeps a good balance between style translation and detail preservation.

With/without labeling. One unique property of our system is its capability to work with unlabeled, heterogeneous motion data. This experiment evaluates the performances of our method on different labeling conditions. In particular, we tested our algorithm on an input motion sequence with four different labeling conditions: “action labeled”, “style labeled”, “both labeled” and “neither labeled”. The evaluation video shows a side-by-side comparison on a test motion sequence containing running, walking, jumping, and walking, as well as their transitions. For each labeling condition, the system produces slightly different yet consistently good output animation.

8 Conclusion

In this paper, we have developed a realtime data-driven animation system for stylistic motion synthesis and control. The main contribution of our work is to introduce a time-varying mixture of autoregressive models for representing the complex relationship between the input and output styles. In addition, we have developed an efficient online local regression model to predict the timing of synthesized poses. Our system is appealing for online style translation and control because it runs in real time, produces high-quality animation in various styles, works well for unlabeled, heterogeneous motion data, and can handle motions that are significantly different from the training data sets.

Our system was designed to add various styles or emotions to existing animation. Therefore, users should not expect to use our system to generate animation from scratch. One way to address this limitation is to combine our technique with existing motion synthesis techniques. In our experiments, we have combined our style transformation technique with motion graphs to synthesize both motion and style for output animation (*e.g.*, walking proudly while following a circle). In the future, we would like to continue this effort and integrate our technique into existing motion synthesis systems (*e.g.*, [Min and Chai 2012]) or performance animation systems (*e.g.*, [Wei et al. 2012]). Such applications would significantly improve the utility of an animation system because the users now not only can control the content of an output animation but also its emotions or styles.

Thus far, we have tested our system on eight distinct styles and heterogeneous actions including walking, running, jumping, kicking, and punching. In the future, we would like to include more actions and more styles in the training database. We believe that some elements of our new representation could be applied to many applications in human motion analysis and synthesis. For example, our online local mixture models could be extended to recognizing actions and styles from input motion data obtained from Kinect cameras. Therefore, one immediate direction for future work is to investigate the applications of our data-driven models in human motion analysis and synthesis.

Acknowledgement

This material is based upon work partially supported by the National Science Foundation under Grants No. IIS-1065384 and IIS-1055046, National Natural Science Foundation of China

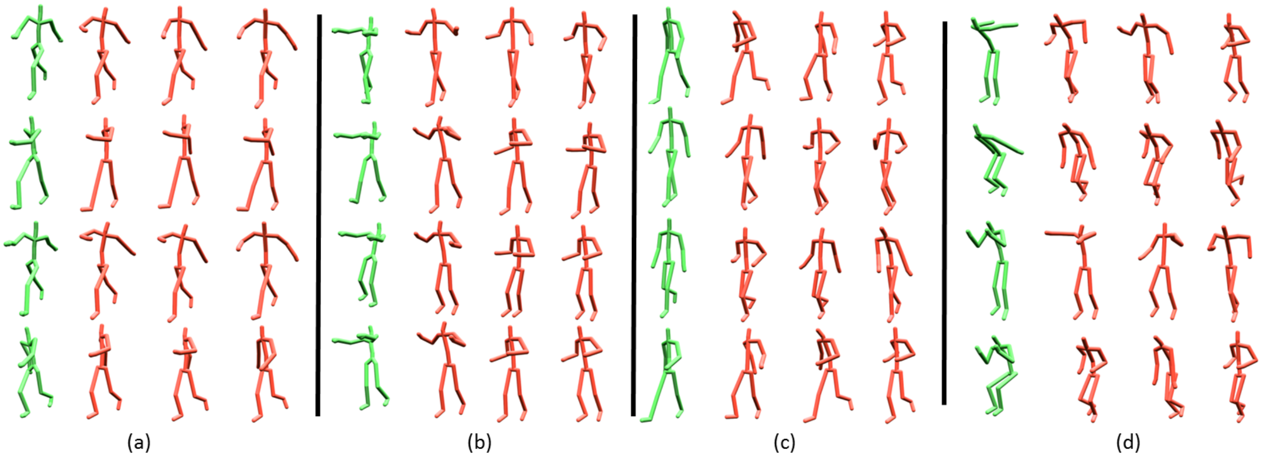

Figure 7: *Nearest neighbors of input motions (in ascending order). Green represents sample poses of an input motion and red shows the three closest neighbors from the database. The input motions from left to right are "exaggerated walking", "mummy walking", "neutral walking", and "neutral jumping". Note that the training database for "neutral walking" and "neutral jumping" sequences only contained running data.*

(No.61173055), and National Key Technology R&D Program of China (No.2013BAK03B07).

References

Aha, D. W. 1997. Editorial, Special issue on lazy learning. In *Artificial Intelligence Review*. 11(1-5):1–6.

Amaya, K., Bruderlin, A., and Calvert, T. 1996. Emotion from motion. In *Proceedings of Graphics Interface 1996*, 222–229.

Brand, M., and Hertzmann, A. 2000. Style machines. In *Proceedings of ACM SIGGRAPH 2000*, 183–192.

Chai, J., and Hodgins, J. 2005. Performance animation from low-dimensional control signals. *ACM Transactions on Graphics 24*, 3, 686–696.

Chai, J., and Hodgins, J. 2007. Constraint-based motion optimization using a statistical dynamic model. *ACM Transactions on Graphics 26*, 3, Article No. 8.

Grassia, F. S. 1998. Practical parameterization of rotations using the exponential map. *Journal of Graphics Tools 3*, 3, 29–48.

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 Popović, 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.

Kovar, L., and Gleicher, M. 2003. Registration curves. In *ACM SIGGRAPH/EUROGRAPH Symposium on Computer Animation*. 214–224.

Kovar, L., and Gleicher, M. 2004. Automated extraction and parameterization of motions in large data sets. *ACM Transactions on Graphics 23*, 3 (Aug.), 559–568.

Kovar, L., Gleicher, M., and Pighin, F. 2002. Motion graphs. *ACM Transactions on Graphics 21*, 3 (July), 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, Article No. 171.

Min, J., and Chai, J. 2012. Motion graphs++: A compact generative model for semantic motion analysis and synthesis. *ACM Transactions on Graphics 31*, 6, 153:1–153:12.

Min, J., Liu, H., and Chai, J. 2010. Synthesis and editing of personalized stylistic human motion. In *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, 39–46.

Monzani, J., Baerlocher, P., Boulic, R., and Thalmann, D. 2000. Using an intermediate skeleton and inverse kinematics for motion retargeting. *Computer Graphics Forum 19*, 3, 11–19.

Rasmussen, C. E., and Nickisch, H. 2010. Gaussian processes for machine learning (gpml) toolbox. *Journal of Machine Learning Research 11*, 3011–3015.

Rose, C., Cohen, M. F., and Bodenheimer, B. 1998. Verbs and adverbs: Multidimensional motion interpolation. In *IEEE Computer Graphics and Applications*. 18(5):32–40.

Shapiro, A., Cao, Y., and Faloutsos, P. 2006. Style components. In *Proceedings of Graphics Interface 2006*, 33–39.

Vicon, 2015. http://www.vicon.com.

Wang, J. M., Fleet, D. J., and Hertzmann, A. 2007. Multifactor gaussian process models for style-content separation. *Proceedings of the 24th International Conference on Machine Learning*. 975-982.

Wei, X., Zhang, P., and Chai, J. 2012. Accurate realtime full-body motion capture using a single depth camera. *ACM Transactions on Graphics 31*, 6 (Nov.), 188:1–188:12.

Wong, C. S., and Li, W. K. 2000. On a mixture autoregressive model. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 62*, 1, 95–115.