Motion Graphs++: a Compact Generative Model for Semantic Motion Analysis and Synthesis

Jianyuan Min
Texas A&M University

Jinxiang Chai*
Texas A&M University

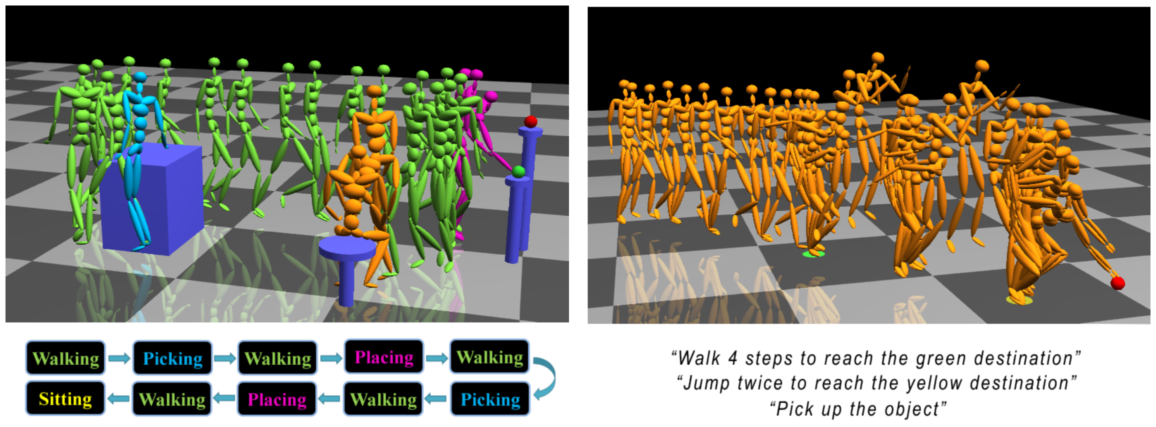

Figure 1: *Semantic motion analysis (left) and synthesis (right) with our generative statistical model.*

Abstract

This paper introduces a new generative statistical model that allows for human motion analysis and synthesis at both semantic and kinematic levels. Our key idea is to decouple complex variations of human movements into *finite structural variations* and *continuous style variations* and encode them with a concatenation of morphable functional models. This allows us to model not only a rich repertoire of behaviors but also an infinite number of style variations within the same action. Our models are appealing for motion analysis and synthesis because they are *highly structured*, *contact aware*, and *semantic embedding*. We have constructed a compact generative motion model from a huge and heterogeneous motion database (about two hours mocap data and more than 15 different actions). We have demonstrated the power and effectiveness of our models by exploring a wide variety of applications, ranging from automatic motion segmentation, recognition, and annotation, and online/offline motion synthesis at both kinematics and behavior levels to semantic motion editing. We show the superiority of our model by comparing it with alternative methods.

Keywords: character animation, generative motion models, semantic motion analysis and synthesis, motion planning

Links: DL PDF

*e-mail: jianyuan|jchai@cs.tamu.edu

ACM Reference Format
Min, J., Chai, J. 2012. Motion Graphs++: a Compact Generative Model for Semantic Motion Analysis and Synthesis. *ACM Trans. Graph. 31* 6, Article 153 (November 2012), 12 pages.
DOI = 10.1145/2366145.2366172 http://doi.acm.org/10.1145/2366145.2366172.

1 Introduction

This paper focuses on constructing a generative motion model to create a rich repertoire of behaviors for virtual humans. Thus far, one of the most successful solutions to this problem is to build *generative statistical models* from prerecorded motion data. Generative statistical models are appealing for motion analysis and synthesis because they are compact, they have strong generalization ability to create motions that are not in prerecorded motion data, and they can generate an infinite number of motion variations with a small number of hidden variables. Despite the progress made over the last decade, creating appropriate generative statistical models for human motion synthesis and control remains challenging for a number of key reasons.

Scalability. A responsive lifelike human character must possess a rich repertoire of activities and display a wide range of styles within the same action. This inevitably requires generative statistical models to scale up well to huge and heterogeneous motion datasets. However, current generative statistical models (e.g., [Chai and Hodgins 2007; Lau et al. 2009; Min et al. 2009]) are often behavior-specific and focus on modeling detailed style variations within the same action. They have not demonstrated that they can model a rich repertoire of behaviors and a wide range of style variations at the same time.

Semantic embedding. Current approaches often fail to encode semantic information into generative models. For example, they cannot identify when and where to pick up an object and cannot count how many walking steps are in output animation. This prohibits the user from creating and controlling an animation at the semantic level (e.g., "walk five steps and pick up the object"). However, for many applications, users are more interested in which actions to perform than how to animate the character at the kinematics level.

Contact awareness. Current generative models are not contact aware because they encode little or no information about environmental contact information such as "left foot on ground." As a result, generated motions often violate environmental contact constraints unless the user explicitly specifies the timings and positions of contact constraints across the entire motion. This is always undesirable because it leads directly to noticeable visual artifacts (e.g., foot sliding and ground penetration) in output animation.

Structure preserving. While there is an infinite number of motion variations within the same action, global structures of the same action are always finite. For example, walking emerges as a sequence of alternating double and single-stance phases, whereas running alternates between single-stance and flight. Current generative statistical models, however, are mainly focused on modeling spatial-temporal variations within a small temporal window rather than global structures of human actions, and thus face great risk of destroying motion structures in the generalization process.

This paper introduces a generative motion model that addresses all the aforementioned challenges. One key idea is to decouple complex variations of human movements into *finite structural variations* and *continuous style variations* and encode them separately into a finite graph of morphable functional models. Each node in the graph stores a morphable function $\mathbf{X}_i = M(\mathbf{s}_i)$, where the parameters $\mathbf{s}_i$ provide a continuous, compact representation for allowable style variations corresponding to a particular motion primitive $\mathbf{X}_i$. Each edge is associated with a probability distribution function $pr(\mathbf{s}_i|\mathbf{s}_j)$ that defines possible transitions from one motion primitive to another one. These morphable models can be strung together to create a rich repertoire of activities in the fully continuous configuration space of a human character. Another key idea of our modeling process is to embed semantics and contact information into generative statistical models. This is a nontrivial task because it requires automatic annotation of an *infinite* number of motions generated by generative statistical models.

Our generative motion models are compact and amenable for motion analysis and synthesis (Figure 1). We have constructed a compact generative model from a huge and heterogeneous motion database (about two hours mocap data and more than 15 different actions). We have demonstrated the power and effectiveness of our models by exploring a wide variety of applications, ranging from automatic motion segmentation, classification, and annotation, online/offline motion synthesis and control, performance-based animation control to semantic motion editing. To the best of our knowledge, this is the *first* generative statistical model that allows for motion analysis and synthesis at a semantic level.

Our work is made possible by a number of technical contributions:

- A generative statistical motion model that addresses all the aforementioned challenges. This also requires us to build probabilistic transition models that ensure seamless transitions from one morphable functional model to another.
- A Maximum A Posteriori (MAP) framework for analyzing, synthesizing, controlling, and editing human motion at both kinematic and semantic levels.
- A novel motion planning technique that combines the power of graph walks, probabilistic sampling, and gradient-based optimization for motion analysis and synthesis.
- A new motion analysis process that allows for automatic motion segmentation, classification, and annotation at a semantic level. In addition, we introduce a new motion editing process that allows the user to conveniently edit an input motion sequence at a semantic level. This is nontrivial as previous motion editing tools are mainly focused on kinematic motion editing.

2 Background

Our work builds upon a significant body of recent work on constructing generative statistical models for human motion analysis and synthesis. Generative statistical motion models are often represented as a set of mathematical functions, which describe human movement using a small number of hidden parameters and their associated probability distributions. Thus far, a wide variety of generative statistical models have been developed and their applications include motion analysis and synthesis, inverse kinematics, and motion style interpolation and transfer. In general, previous generative statistical models can be categorized into variants of Hidden Markov Models (HMMs) [Molina Tanco and Hilton 2000; Bowden 2000; Brand and Hertzmann 2000], statistical dynamic models for modeling spatial-temporal variations within a small temporal window [Li et al. 2002; Hsu et al. 2005; Chai and Hodgins 2007; Lau et al. 2009; Ye and Liu 2010; Wei et al. 2011], and low-dimensional statistical models for human poses [Grochow et al. 2004; Chai and Hodgins 2005]. Our models advance the state of the art in human motion modeling by addressing the four aforementioned challenges in generative statistical modeling. We show for the first time that generative statistical models can be applied to analyze and synthesize a rich repertoire of human activities at both kinematic and semantic levels.

Our generative motion model builds on the success of deformable motion models developed by Min et al. [2009]. However, their models, while powerful, are inherently limited to modeling spatial-temporal variation in structurally similar motions and thus are not appropriate to analyze and synthesize a rich repertoire of behaviors. We significantly extend their idea to heterogeneous data sets by concatenating distinctive morphable functions via graph walks and probabilistic sampling. This extension is non-trivial because it leads us to a new contact-aware and semantic embedding generative model for human motion analysis and synthesis, a novel GP-based transition model that ensures seamless transitions from one morphable model to another one, a new motion planning technique that combines the power of graph walks, probabilistic sampling, and gradient-based optimization, and a number of new applications such as semantic motion analysis, synthesis and editing.

Our approach models complex human movement using a finite directed graph of morphable motion models. Though the general idea of using a directed graph for human motion modeling is not new and dates back to pioneering work on motion graphs [Kovar et al. 2002; Arikan and Forsyth 2002; Lee et al. 2002], our approach is different from motion graphs as well as their various extensions [Arikan et al. 2003; Gleicher et al. 2003; Lau and Kuffner 2005; Safonova and Hodgins 2007; Shum et al. 2008; Lee et al. 2010] in that our models are generative and concatenate *continuous* morphable functions rather than *discrete* poses from prerecorded motion data. As a result, our models are more compact and allow for more accurate motion control than motion graphs (see comparison results in Section 5.5). In addition, unlike motion graphs, our models are also amenable to motion analysis such as motion segmentation, classification, and annotation. More recently, Lee and colleagues [2010] extended the idea of motion graphs by concatenating continuously interpolated poses from prerecorded motion data. However, similar to motion graphs, their models are not generative because they need to retain all prerecorded motion data for motion interpolations and concatenations. They are also not highly structured and semantic embedding, and thus prohibit the user from synthesizing and controlling human motion at the semantic level.

Our models are related to recent work on using interpolations for motion synthesis and control [Rose et al. 1998; Kovar and Gleicher 2004; Mukai and Kuriyama 2005; Shin and Oh 2006; Heck and Gleicher 2007; Treuille et al. 2007]. Motion interpolations regis-

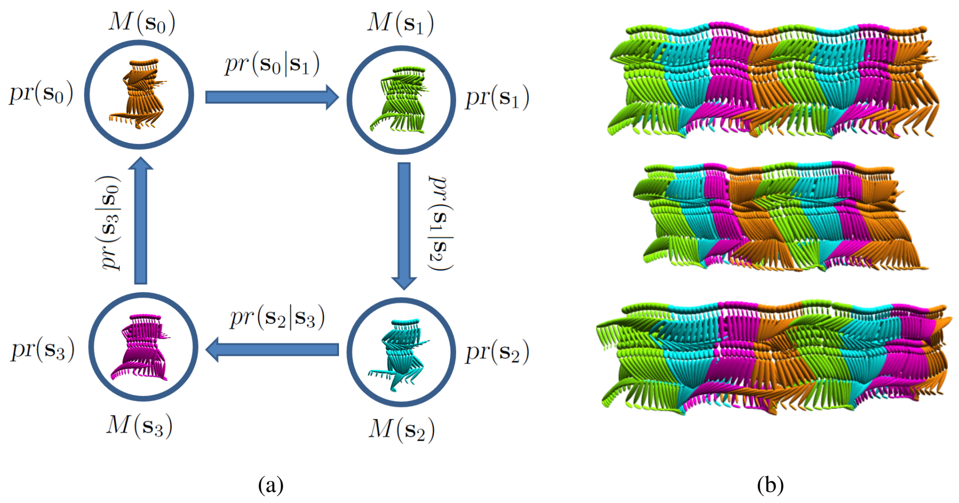

Figure 2: *A morphable graphs model for running: (a) we decompose running motions into four distinctive motion primitives, including "left stance", "flight 1", "right stance", and "flight 2" and model their transitions with a directed graph of four nodes. Each graph node $i \in V$ stores a generative statistical model, including a morphable function $\mathbf{X}_i = M(\mathbf{s}_i)$ and a* prior *distribution function* $\text{pr}(\mathbf{s}_i)$*, to model continuous style variations with a small number of morphable parameters $\mathbf{s}_i$. An arc $(i, j) \in A$, which represents an allowable transition from primitive i to primitive j, stores a probability distribution function $\mathbf{pr}(\mathbf{s}_j|\mathbf{s}_i)$ over the morphable parameters $\mathbf{s}_j$ at node j, given the morphable parameters $\mathbf{s}_i$ at node i. (b) An infinite number of running styles can be generated by our morphable graphs model using random motion generation algorithm described in Section 4.1.*

ter a set of structurally similar motion examples and parameterize them in an abstract space defined for motion interpolations. Among all the systems, our approach is most similar to parametric motion graphs by Heck and Gleicher [2007], which extended motion interpolation techniques to heterogeneous datasets by constructing allowable transitions between different types of interpolated actions. Similar to motion graphs, their models are not generative and are not amenable to motion analysis and editing. Our transition model is also different from theirs because it is based on Gaussian processes rather than local interpolations adopted in their work. In addition, instead of using sparse data interpolations for motion control, we formulate the motion synthesis and control problem in a Maximum A Posteriori framework and develop a hybrid optimization procedure to find an optimal solution. This enables more accurate and flexible animation control (see comparison results in Section 5.5).

Our work is also relevant to recent efforts on developing a highly structured representation for large unstructured motion capture datasets. One notable example is motion motif graphs developed by Beaudoin and colleagues [2008]. The motion-motif graph applies motion segmentation and clustering techniques to unstructured motion data and builds a graph-based representation for the compressed motion data. However, their representation is mainly designed for analyzing and visualizing the contents and connectivity of prerecorded motion datasets rather than motion synthesis and control targeted in this paper. It is not clear how their representation can be extended to enable accurate and flexible motion control at both the kinematic and semantic levels. In addition, their models also have the same limitations as previous generative statistical models.

3 Morphable Graphs Modeling

While natural human movement displays a wide range of style variations within the same action, high-level structures of human actions are always finite. For example, walking emerges as a sequence of alternating double and single-stance phases, whereas running alternates between single-stance and flight. This observation leads us to develop an efficient representation for human movement.

In our new representation, we decompose each action into distinctive motion primitives and encode transitions between motion primitives using a directed graph. For example, we decompose running motions into four distinctive motion primitives, including "left stance," "flight 1," "right stance," and "flight 2," and encode their transitions with a directed graph of four nodes (see Figure 2(a)), where each node corresponds to a distinctive motion primitive. Finite graph structures are appropriate to model high-level structures of human movement because in practice there is a finite number of distinctive motion primitives. Each motion primitive, however, might display an infinite number of stylistic variations such as continuous speed variations. We choose to use generative statistical models to represent continuous style variations within each primitive because they can model an infinite number of motion variations using a small number of parameters. In this paper, we call generative motion models associated with each motion primitive as *morphable motion primitives* and a directed graph of *morphable motion primitive* as *morphable graphs*.

We model complex human movements using a finite directed graph of *morphable motion primitives*: $G = (V, A)$. A node $i \in V$ in the graph corresponds to a *morphable motion primitive*, including a morphable function $\mathbf{X}_i = M(\mathbf{s}_i)$ and a *prior* distribution function $pr(\mathbf{s}_i)$ over the morphable parameters $\mathbf{s}_i$. An arc $(i, j) \in A$, which represents an allowable transition from node i to j, stores a probability distribution function $\mathbf{pr}(\mathbf{s}_j|\mathbf{s}_i)$ over the morphable parameters $\mathbf{s}_j$ at node j, given the morphable parameters $\mathbf{s}_i$ at node i. Another unique property of our generative model is that each *morphable motion primitive* is embedded with environmental contact information and semantics.

We outline the modeling process as follows:

- We first identify important keyframes (e.g., left foot strike) for each motion example in the database and use them to decompose each motion sequence into distinctive motion segments. Motion segments are automatically placed into the same motion primitives if they share the same set of keyframes.

- We apply functional data analysis techniques to all motion segments associated with the same motion primitive and construct a morphable function to compactly represent continuous style variations within the same motion primitive: $\mathbf{X}_i = \mathbf{M}(\mathbf{s}_i)$, where the vector $\mathbf{s}_i$ is a low-dimensional vector for the i-th motion primitive. For each primitive, we also model a prior distribution function $pr(\mathbf{s}_i)$ over the morphable vector $\mathbf{s}_i$ to ensure generated motions are close to prerecorded motion data.

- We organize all the *morphable motion primitives* into a finite directed graph $G = (V, A)$. Each node $i \in V$ in the graph stores one morphable motion primitive. A directed edge *(i,j)* is added to A if there is a valid transition from one motion primitive i to another one *j* in prerecorded motion data. We construct transition distribution functions $\mathbf{pr}(\mathbf{s}_j|\mathbf{s}_i)$ using Gaussian Processes (GP) to model possible transitions between adjacent *morphable motion primitives*.

- We annotate each morphable motion primitive to embed environmental contact information and semantic knowledge into the generative models.

We describe details of each step in the rest of this section.

3.1 Preprocessing Motion Data

Our motion data preprocessing step aims to decompose unstructured motion data into a set of contact-consistent, structurally similar motion segments that are suitable for functional data analysis.

Motion segmentation. Similar to motion interpolations [Rose et al. 1998], our preprocessing step first identifies a set of "keyframes" in prerecorded motion examples, which are used to segment motion examples into distinctive motion primitives. In our application, the keyframes often correspond to instances when contact state transitions occur (e.g., left foot strike) or instances with highest visual content changes such as "punching," "picking," and "placing." This ensures that all the motion segments within the same motion primitive are contact consistent and structurally similar. Contact transition frames are automatically detected by using a similar method adopted in [Lee et al. 2002]. To detect frames that correspond to highest visual content changes, we select a few motion examples for each action and annotate all the keyframe poses in each of the selected examples. The system then automatically search similar keyframe poses in the rest motion examples by using the distance metric similar to [Kovar et al. 2002]. We segment each sequence at the keyframes and group all the motion segments in terms of the keyframes. Motion segments that share the same starting and ending keyframes are placed into the same group (i.e. motion primitive).

Motion registration and decomposition. For each motion primitive, we register all the motion segments against each other and decompose them into two functional data sets which are suitable for generative statistical modeling. Briefly, we pick one segment as a reference motion and use it to register the rest of segments with appropriate time warping functions. We register 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 up axis [Kovar and Gleicher 2003]. We employ dynamic time warping techniques to register all motion segments within the same motion primitive. Next, we warp each motion segment to a new motion segment in a canonical timeline defined by the reference motion using their corresponding time warping functions. This step allows us to decompose all the motion segments within the same motion primitive into two functional data sets: *warped motion segments* and *time warping functions*. Both data sets are defined in the canonical timeline and therefore are suitable for functional statistical analysis.

3.2 Modeling Morphable Motion Primitives

We now discuss how to apply generative statistical modeling techniques to the preprocessed motion data to construct a compact morphable function that operates in the fully-continuous configuration space of the character. Our analysis algorithm builds upon functional data analysis described in [Min et al. 2009].

We apply functional PCA to all the warped motion segments associated with each motion primitive. As a result, we can model a motion segment defined in a canonical timeline by the reference motion using a mean motion segment $\mathbf{p}_0$ and a weighted combination of eigen motion segments $\mathbf{p}_m, m = 1..., M$:

$$P(\vec{\alpha}) \quad = \quad \mathbf{p}_0 + [\mathbf{p}_1 ... \mathbf{p}_M]\vec{\alpha}, \tag{1}$$

where the vector $\vec{\alpha} = [\alpha_1, \alpha_2, ..., \alpha_M]^T$ represents the eigen weights and the vectors $\mathbf{p}_m, m = 1, ..., M$ are a set of orthogonal modes to model geometric variations across the entire motion sequence.

Similarly, we apply functional PCA to all the time warping functions associated with each motion primitive to build a low-dimensional model for time warping functions. To preserve the property of time warping functions, which are always positive and strictly monotonic everywhere, we transform the precomputed time warping functions $w(t)$ into a new space $z(t)$: $z(t) = \ln(w(t) - w(t-1)), t = 1, ...T$ and apply the functional PCA to the transformed time warping functions in the new space. Finally, we transform the function back to the original space and obtain the following low-dimensional representation for time warping functions:

$$H(t; \vec{\gamma}) \quad = \quad \textstyle\sum_{i=1}^{t} \exp(b_0(i) + [b_1(i)...b_K(i)]\vec{\gamma}), \tag{2}$$

where the vector $\vec{\gamma}$ is the combination weights to represent a time warping function in a low-dimensional space and the scalar $b_k(i)$ represents the i-th component of the k-th eigen vector $\mathbf{b}_k, k = 1, ..., K$.

After combining Equation (1) and (2), we obtain the following morphable function to model spatial-temporal variation of a motion primitive:

$$\begin{aligned} \mathbf{X} &= M(\mathbf{s}) \\ &= P(\vec{\alpha}) \otimes H(\vec{\gamma}), \end{aligned} \tag{3}$$

where the morphable vector $\mathbf{s}$ stacks the vector $\vec{\alpha}$ and $\vec{\gamma}$. The operator $\otimes$ warps a motion segment $P(\vec{\alpha})$ in the canonical time line with a time warping function $H(\vec{\gamma})$.

We learn a joint probability distribution function $pr(\mathbf{s})$ based on the *warped motion segments* and the precomputed *time warping functions* from prerecorded data. We model the prior distribution $pr(\mathbf{s})$ with a Gaussian mixture model (GMM). The parameters of the Gaussian mixture model are automatically estimated using an Expectation-maximization algorithm [Bishop 1996]. The joint probability distribution constrains the motion generated by the morphable motion to stay close to the prerecorded motion examples.

3.3 Transitions Between Morphable Primitives

Now each node $i \in V$ in the graph contains a continuous morphable function $M(\mathbf{s}_i)$ and a prior distribution function $pr(\mathbf{s}_i)$ over the morphable parameters . Our next challenge is to model possible transitions for every arc $(i, j) \in A$ in the graph G. This will allow us to concatenate morphable primitives to form a rich repertoire of human activities. The problem is challenging because of finite samples in an infinite space.

Our idea is to utilize the prerecorded motion data to learn a probability distribution function $pr(\mathbf{s}_j|\mathbf{s}_i)$ for predicting the morphable parameters at one node $\mathbf{s}_j$ from another $\mathbf{s}_i$. A high value for the distribution function $pr(\mathbf{s}_j|\mathbf{s}_i)$ means a good transition from one motion segment $M(\mathbf{s}_i)$ to another one $M(\mathbf{s}_j)$. We propose to use a Gaussian Process (GP) model to model a probabilistic distribution function $pr(\mathbf{s}_j|\mathbf{s}_i)$. We choose the Gaussian process because GP can efficiently model nonlinear properties of transition functions and its learning process involves very few manual tuning parameters. GP and its invariants have recently been applied to many problems in computer graphics, including nonlinear dimensionality reduction for human poses [Grochow et al. 2004], motion editing [Ikemoto et al. 2009], and motion synthesis [Wei et al. 2011].

In our application, GP learns a prior distribution function over the morphable parameters $\mathbf{s}_j$ at node j, given the morphable parameters $\mathbf{s}_i$ at node i and training data $\mathtt{D}$. We write this probability as

$$pr(\mathbf{s}_j|\mathbf{s}_i, \mathtt{D}). \tag{4}$$

Because we use Gaussian process, the probability distribution function defined in Equation (4) is a multidimensional Gaussian distribution function:

$$pr(\mathbf{s}_j|\mathbf{s}_i, \mathtt{D}) \quad = \quad \mathcal{N}(\mu(\mathbf{s}_i), \Sigma(\mathbf{s}_i)), \tag{5}$$

where both mean μ and covariance matrix Σ are functions of the morphable parameters $\mathbf{s}_i$ (for details, see [Rasmussen and Williams 2006]). If the covariance has a small determinant, then the Gaussian distribution has a narrow peak, indicating high confidence in the prediction; similarly, a large determinant indicates low confidence.

There are several publicly available implementations of Gaussian process. We used the Gaussian process library developed by Lawrence [2009]. One limitation of GP learning is that memory requirements and computational demands grow as the square and cube respectively of the size of training data. We address this limitation by adopting sparse approximation strategies for Gaussian process regression [Quinonero-Candela and Rasmussen 2005].

3.4 Annotating Morphable Graphs

This section discusses how to embed environmental contact information and semantic knowledge into morphable graphs models.

Contact annotation. In general, annotating a generative statistical model with environmental contact information is difficult because doing so requires accurate annotation of an *infinite* number of motion instances generated by the model. Our generative model naturally supports environmental contact annotation because motions generated by the same morphable motion primitive are always contact-consistent and structurally similar. We annotate our morphable graphs model by annotating the canonical timeline of every morphable primitive. The advantage to annotating the canonical timeline is that annotation labor is irrelevant to the number of database examples. For example, a morphable primitive for walking can be easily annotated by labeling its four contact instances on the canonical timeline (“left heel down,” “left toe up,” “right heel down’,’ “right toe up”).

We encode contact information in each frame using a binary feature vector. Each bit represents a particular type of contact events, e.g., left toe plants on the ground. For locomotion such as walking, jumping, and running, each frame requires only a 4-bit label (left heel, left toe, right heel, and right toe). Extra bits (e.g., left hip and right hip, left knee and right knee, left hand and right hand) are needed to encode contact-rich motions such as “sitting,” “kneeling down,” and “climbing up.” Contact annotation enables us to automatically enforce environmental contact constraints in a motion generalization process, thereby eliminating noticeable visual artifacts such as foot sliding and ground penetration in output animation.

Semantics annotation. Semantics annotation is used to describe high-level behaviors of human movements. In our system, we use them to describe motion primitives in the graph. To accommodate a rich repertoire of activities, the annotations should be flexible. In other words, the user should be able to use an arbitrary vocabulary suited to the motions being annotated. The vocabulary chosen for the annotations defines the level of control of the synthesis.

We first annotate qualitative features of motions. The database of motions used for our examples consisted of approximately two hours of “everyday” activities. The vocabulary that we chose to annotate this database consisted of: *standing, walking, running, jumping, stepping stones, backward walking, kneeling down, kneeling up, picking, placing, sitting, standing up, left punching, right punching, climbing up, and climbing down*. Our annotation vocabulary reflects our database. A different choice of vocabulary would be appropriate for different collections. Some of these annotations can co-occur: “picking an object while kneeling down.”

A more accurate description of motions at the task level often requires appropriate annotation of quantitative properties of human actions. For example, a task-level description of “picking” often needs to identify *when* and *where* to pick up an object. This causes problems for the annotations because both the timings and the hand locations of “picking” motion instances are not constant but vary with morphable parameters. To address this issue, we choose to annotate the “timings” on the canonical timelines of morphable primitives and describe the “locations” based on their local coordinates rather than global coordinates. Now both the canonical timeline and the local coordinates are constant; therefore, their semantic information can be conveniently embedded into morphable primitives. More generally, important quantitative features of motions are often defined by kinematic functions of character poses at multiple key instances. Annotating such features requires encoding both kinematic functions and the timing of key instances. Similarly, annotating semantics on the canonical timeline allows us to significantly reduce annotation labor. In our experiment, the entire process for annotating both contacts and semantics takes about 30 minutes for the morphable graphs model constructed from a two-hours motion capture database with 16 distinctive behaviors.

In our implementation, we create a mapping table to embed both qualitative and quantitative features into the morphable graphs models. The mapping table is critical to semantic motion synthesis and analysis. On one hand, semantic motion synthesis can retrieve the mapping table to convert semantic control commands into low-level animation constraints. For example, verbs such as “walking” and “running” are mapped to corresponding morphable primitives (i.e. graph nodes) and adverbs such as “five steps” are mapped to the total number of graph nodes visited. On the other hand, they can be used in motion analysis process to extract the semantics embedded in unknown motion sequences.

4 Applications

One key advantage of our generative models is that they are amenable for motion analysis and synthesis at both the kinematic and sematic level. Section 4.1 discusses how to apply our model to generate natural-looking human movements. We then describe its applications in online motion control (Section 4.2), offline animation design (Section 4.3), and semantic motion analysis and editing

(Section 4.4).

4.1 Random Motion Generation

The morphable graphs model decomposes overall motion variations into *finite structural variations* from *continuous style variations*. Accordingly, we develop a two-step approach to synthesize a random motion sequence.

- **Structural variation synthesis.** Structural variation is critical for creating a rich repertoire of human activities. We synthesize structural variations by sequentially concatenating morphable nodes via graph walks. Since every graph node stores a distinctive morphable primitive, a graph walk produces a path of morphable nodes, denoted as $n_0 \to ... \to n_T$.
- **Style variation synthesis.** Once we obtain a sequence of morphable nodes, we can use them to create a wide range of style variations via probabilistic sampling techniques. Briefly, we first generate a motion segment $\mathbf{X}_0 = M(\mathbf{s}_{n_0})$ by randomly drawing a sample for the morphable parameters $\mathbf{s}_{n_0}$ based on the prior distribution *pr*$(\mathbf{s}_{n_0})$ stored in the first node n_0. We synthesize subsequent motion segments $\mathbf{X}_i = \mathbf{M}(\mathbf{s}_{n_i}), i = 1, ..., T$ by recursively drawing a random sample for the morphable parameters $\mathbf{s}_{n_i}$ based on the transition distribution *pr*$(\mathbf{s}_{n_i}|\mathbf{s}_{n_{i-1}})$ as well as the morphable parameter $\mathbf{s}_{n_{i-1}}$ at the previous graph node n_{i-1}.

To transition from one motion segment to another one, e.g., $\mathbf{X}_{i-1} \to \mathbf{X}_i$, we rotate and translate every motion segment so that each segment starts from where the previous one ended. Modeling the possible transitions with a distribution function prevents generating unnatural transitions and therefore significantly reduces the discontinuities around the transition points. In practice, there might still be some discontinuities around the transition points. However, these discontinuities are often very small and we can distribute them within a smoothing window around the discontinuity. We address this issue using motion blending techniques present by Arikan and Forsyth [2002]. Although blending avoids jerkiness in the transitions, it might violate environmental contact constraints. Our process eliminates the artifact by automatically enforcing contact constraints embedded in the morphable graphs model. Figure 2(b) shows three sequences of running motions randomly generated by our motion generation process.

4.2 Online Motion Control

Our model naturally supports online motion control. Motions are planned on a node-by-node basis using a hybrid motion planning process that combines the power of graph walks, probabilistic sampling, and gradient-based optimization. Given the current morphable node, the online planner evaluates the state resulting from each of many possible actions that follows the current node. This planning process is recursively run to look one or more morphable nodes into the future.

We formulate the online motion control problem in a Maximum A Posteriori (MAP) framework. Given the current state of human characters, $\mathbf{M}(\mathbf{s}_{n_0}), n_0 \in V$, the current control commands specified by the user $\mathbf{c}_{ol}$, and the interaction 3D environment $\mathbf{e}$, we determine an optimal path of the morphable nodes $n_1 \to ... \to n_L$ as well as the optimal values for corresponding morphable parameters $\mathbf{s}_{n_1}, ..., \mathbf{s}_{n_L}$ by solving the following MAP problem:

$$\max_{\{n_1,...,n_L\},\{\mathbf{s}_{n_1},...,\mathbf{s}_{n_L}\}} pr(\mathbf{s}_{n_1}, ..., \mathbf{s}_{n_L}|\mathbf{s}_{n_0}, \mathbf{c}_{ol}, \mathbf{e}), \quad (6)$$

where L is the number of morphable nodes the planning process looks into the future.

According to Bayes rule, the above posteriori distribution function can be decomposed into the following three terms:

$$\textstyle\prod_{i=1}^{L} pr(\mathbf{s}_{n_i}|\mathbf{s}_{n_{i-1}}) pr(\mathbf{e}|\mathbf{s}_{n_1}, ..., \mathbf{s}_{n_L}) pr(\mathbf{c}_{ol}|\mathbf{s}_{n_1}, ..., \mathbf{s}_{n_L}), \quad (7)$$

where the first term is the *transition* term defined in Equation (5), which models the prior distribution that transitions from one morphable node ($\mathbf{s}_{n_{i-1}}$) to another one ($\mathbf{s}_{n_i}$). The second term is the *contact-awareness* term which penalizes the inconsistency between the generated motion and the environmental contacts $\mathbf{e}$. The third term is the *control* term which measures how well the generated motion satisfies the user's control inputs $\mathbf{c}_{ol}$. In the following we focus our discussion on the *contact-awareness* term and the *control* term since the *transition* term is already defined in Equation (5).

Contact-awareness term. One appealing property of our model is that every frame in generated motions is automatically annotated with contact information. The *contact-awareness* term ensures that synthesized motions are always consistent with environmental contacts. More specifically, for every frame in the concatenated motion pieces $\mathbf{M}(\mathbf{s}_{n_i}), i = 1, ..., L$, we first retrieve the contact information encoded in the corresponding morphable function. If there is a contact between the character and the environment (e.g., the left foot must be planted on the ground), we measure the distance between the synthesized contact point on the character and the corresponding contact plane. We assume gaussian distributions with a standard deviation σ_{en} for the *contact-awareness* term. In our experiment, we set a low value to σ_{en} because maintaining correct environmental contacts is very important to the perceptual quality of the synthesized motion.

Control term. Our model enables the user to accurately control a character at both the kinematic and semantic level. Low-level kinematic control is important because it allows the user to accurately control stylistic variations of particular actions. Our online motion control framework is very flexible and supports any kinematic control inputs. The current system allows the user to control an animation by selecting any point on the character and specifying a path for the selected point to follow. The user could also direct the character by defining the high-level control knobs such as turning angles, step sizes, and locomotion speeds.

Assuming gaussian noise with a standard deviation of σ_{ol} for the user's control inputs $\mathbf{c}_{ol}$, we can define the likelihood of the *control* term for kinematics control as follows:

$$pr(\mathbf{c}_{ol}|\mathbf{s}_{n_1}, ..., \mathbf{s}_{n_L}) \propto \exp\left(\frac{-\|\mathbf{f}(M(\mathbf{s}_{n_1}), ..., M(\mathbf{s}_{n_L})) - \mathbf{c}_{ol}\|^2}{2\sigma_{ol}^2}\right), \quad (8)$$

where the function $\mathbf{f}$ is a forward kinematic function that map the "hypothesized" motion to "hypothesized" control knob values. The standard deviation σ_{ol} indicates the user's confidence in the specified control inputs and is important for the tradeoff between the control accuracy and the quality of the generalized motion. The higher the standard deviation; the lower the confidence.

Semantic motion control focuses on which actions to perform without becoming entangled in low-level kinematic details. To achieve motion control at the semantic level, we use semantic knowledge encoded in the mapping table to convert semantic control commands into appropriate low-level animation constraints, including *graph node constraints* and *kinematic constraints*. For example, verbs such as "backward walking" are mapped to their corresponding graph nodes. Adverbs such as "picking up an object at a particular location" are mapped to corresponding graph nodes ("picking") as well as kinematic functions retrieved from the mapping table. The mapped kinematic constraints are used to evaluate the goodness of the generated motions in a way similar to those used in kinematic motion control (see Equation (8)). The mapped graph nodes are compared against the "synthesized" graph nodes to measure the goodness of the "synthesized" motion.

Realtime motion planning. Online motion control requires solving the MAP estimation problem in Equation (7). This is a nontrivial task because it involves optimizing a nonsmooth, nonconvex function over both discrete variables, $n_1, ..., n_L$, and continuous variables, $\mathbf{s}_{n_1}, ..., \mathbf{s}_{n_L}$. In addition, the planning algorithm should be fast enough that the control interface appears responsive and the user remains engaged in the animation task.

One idea is to combine graph walks and probabilistic sampling techniques to search an optimal solution to the MAP estimation problem. Briefly, we enumerate all possible paths starting from the current node, randomly draw a number of independent samples for each path based on the prior distributions $\prod_{i=1}^{L} pr(\mathbf{s}_{n_i}|\mathbf{s}_{n_{i-1}})$, and evaluate the "goodness" of each sample based on the sum of the *control* term and the *contact-awareness* term. One limitation of probabilistic sampling is that there is no guarantee the sampled solutions can precisely match the control inputs and environmental contact constraints.

To address this issue, we propose a hybrid motion planning algorithm that combines graph walks and probabilistic sampling with gradient-based optimization. We first apply graph walks and probabilistic sampling techniques to find a "close" solution and then employ gradient-based optimization to fine tune the solution in the vicinity of the "close" solution. To achieve this, we use the "close" solution as a reference to fix the path of graph nodes and refine the values of the morphable parameters by solving the following continuous optimization problem:

$$\min_{\{\mathbf{s}_{n_{1:L}}\}} -\ln pr(\mathbf{e}|\mathbf{s}_{n_1}, ..., \mathbf{s}_{n_L}) - \ln pr(\mathbf{c}_{ol}|\mathbf{s}_{n_1}, ..., \mathbf{s}_{n_L}). \tag{9}$$

We analytically evaluate the Jacobian terms of the objective function and run a gradient-based optimization with the Levenberg-Marquardt algorithm [Lourakis 2009]. The exact number of samples for probabilistic sampling depends heavily on the complexity of the graph and control inputs, but we found 300 samples to be more than enough for all of the cases we tried. The hybrid planning algorithm not only alleviates the local minima problem but also significantly improves the accuracy and efficiency of online motion control process. The current online motion synthesis system runs in real time (*42* frames per second) on a machine with Intel i7 2.80GHZ CPU.

4.3 Offline Motion Design

Our generative model is also well suited for offline animation design. Similar to online motion control, we formulate the offline animation design problem in a MAP estimation framework. Given user-defined constraints $\mathbf{c}_{off}$ as well as environmental contacts $\mathbf{e}$, we select the most likely path of morphable nodes $n_1, ..., n_T$ and compute the optimal morphable parameters $\mathbf{s}_{n_1}, ..., \mathbf{s}_{n_T}$ corresponding to each node:

$$\max_{T,\{n_{1:T}\},\{\mathbf{s}_{n_{1:T}}\}} pr(\mathbf{s}_{n_1}, ..., \mathbf{s}_{n_T}|\mathbf{c}_{off}, \mathbf{e}), \tag{10}$$

where $n_i \in V$ is the index for the i-th morphable node in the synthesized motion and T is the total number of the morphable nodes. However, unlike online motion control, the number of the morphable nodes (T) in offline motion synthesis is often unknown and needs to be directly estimated from the motion synthesis process.

Again, we decompose the posteriori distribution function in Equation (10) into three terms and estimate the most likely motion sequence $\mathbf{M}(\mathbf{s}_{n_1}) \rightarrow ... \rightarrow \mathbf{M}(\mathbf{s}_{n_T})$ with the hybrid motion planning algorithm described in Section 4.2. Similarly, our animation design system allows for accurate and precise motion control at both the semantic and kinematic level.

Continuous style editing. Another nice property of our system is to allow the user to continuously edit stylistic variations of synthesized motions with direct manipulation interfaces. This involves continuously updating the morphable parameters based on user constraints, while fixing the path on the graph. The user can incrementally edit the synthesized motion by dragging any character point at any frame, modifying the path of any point on the character over a period of time, or adjusting the values of high-level control knobs such as walking directions, speeds, and step sizes. In addition, we can edit the motion by adjusting positions or sizes of interaction objects. For example, to edit the styles of the action "sitting on a chair", the user can directly manipulate the location or height of the chair.

Again, we apply the negative log to the posteriori distribution function and convert the MAP problem into an energy minimization problem:

$$\begin{aligned}\min_{\{\mathbf{s}_{n_{1:T}}\}} -\textstyle\sum_{i=1}^{T} \ln pr(\mathbf{s}_{n_i}|\mathbf{s}_{n_{i-1}}) - \ln pr(\mathbf{c}_{edit}|\mathbf{s}_{n_1}, ..., \mathbf{s}_{n_T}) \\ - \ln pr(\mathbf{e}|\mathbf{s}_{n_1}, ..., \mathbf{s}_{n_T}).\end{aligned} \tag{11}$$

Given a fixed path of morphable nodes, $n_1 \rightarrow ... \rightarrow n_T$, we employ gradient-based optimization [Lourakis 2009] to incrementally modify the morphable parameters $\mathbf{s}_{n_1}, ..., \mathbf{s}_{n_T}$ based on the updated constraints $\mathbf{c}_{edit}$ on the fly.

4.4 Semantic Motion Analysis and Editing

A primary difference between generative statistical models and alternative data-driven methods such as motion graphs and motion interpolations is that they are amenable to motion analysis. The goal of our motion analysis process herein is to identify both the morphable primitives $\{n_1, ..., n_T\}$ and the corresponding morphable parameters $\{\mathbf{s}_{n_1}, ..., \mathbf{s}_{n_T}\}$ associated with an "observed" motion sequence $\mathbf{z}_{input}$. Again, this can be formulated as the following MAP estimation problem:

$$\max_{T,\{n_{1:n}\},\{\mathbf{s}_{n_{1:T}}\}} pr(\mathbf{s}_{n_1}, ..., \mathbf{s}_{n_T}|\mathbf{z}_{input}), \tag{12}$$

where $n_i \in V$ is the index for the i-th morphable node in the input motion and T is the total number of the morphable nodes.

Our motion analysis process adopts the optimization scheme similar to motion synthesis and control. We modify the *control* term in such a way that measures the difference between the "synthesized" motion, $\mathbf{M}(\mathbf{s}_{n_1}) \rightarrow ... \rightarrow \mathbf{M}(\mathbf{s}_{n_T})$, and the "observed" motion data, $\mathbf{z}_{input}$. In addition, we drop off the *contact-awareness* term because 3D environments of the "observed" motion sequence are often unknown. We optimize the motion in the same way as we do with online motion control. Given the current morphable node, the motion analysis process evaluates the state resulting from each of many possible actions that follows the current node. This is applied recursively to look a small number of morphable nodes into the future.

Once we synthesize the most likely motion sequence to match the "observed" motion data, we can automatically annotate the "observed" motion data with semantic knowledge and contact information embedded in the "synthesized" motion sequence. This "analysis-by-synthesis" strategy allows us not only to segment the "observed" motion data into multiple motion primitives but also to classify and annotate each motion primitive with our semantic vocabulary. For example, the system can count how many steps are in an "observed" walking sequence and identify when and where to pick up an object in a "picking" sequence. The level of semantics extracted from input motion depends on the vocabulary chosen for the annotations. In addition, automatic contact annotations can be achieved within the same analysis framework.

Semantic motion editing. Unlike kinematic motion editing, semantic motion editing focuses on modifying important qualitative and quantitative properties rather than low-level kinematic details of input motion data. Before semantic motion editing, we first apply semantic motion analysis to extract semantics embedded in input motion data. We modify the extracted semantics and use them to edit the input motion data in a way similar to offline animation design. The level of semantic motion editing is dependent on the vocabulary chosen for the annotations. For example, we can modify qualitative properties of input motion data by adding and/or deleting particular verbs such as "walking." We can also edit the motion at the semantic level by modifying quantitative properties of the data such as "the number of walking steps," "when and where to pick up an object," and "the location of a sitting chair." For instance, we can apply semantic motion editing to modify the number of walking steps, speed up/slow down the walking, or replace walking with other behaviors.

5 Results

We have demonstrated the power and effectiveness of our model on a wide variety of online and offline applications, including semantic motion analysis, editing and synthesis, online motion control, performance-based animation control, and offline motion design. To the best of our knowledge, this is the *first* generative statistical model that has demonstrated motion analysis, synthesis and editing at a semantic level. In addition, we have demonstrated the superiority of our model over several baseline models such as motion graphs and motion interpolations. We have also assessed the generalizability of our motion model via user studies. Lastly, we have evaluated the performance of our system in terms of the importance of GP transition models and the number of samples used for probabilistic motion planning. Our results are best seen in video form.

Our model is compact as the memory consumption of the constructed model (about 16.2Mb) is significantly lower than the original training data (about 248.3Mb). It also scales up well to large and heterogeneous datasets. The original datasets contain $1,049,669$ frames (145.79 minutes) and consist of 16 distinctive actions, including *standing, walking, running, two-feet jumping, stepping-stone jumping, sitting down, standing up, climbing up, climbing down, left punching, right punching, picking, placing, kneeling down, kneeling up, backward walking*, and their transitions.

5.1 Semantic Motion Analysis, Editing and Synthesis

Semantic motion analysis. This experiment demonstrates the application of our generative motion model to automatic motion segmentation, classification, and annotation (see Figure 1(left)). The test motion sequence includes four distinctive human actions ("walking," "picking," "placing," and "sitting") as well as their transitions. Our system automatically decomposes the whole motion sequence into motion segments corresponding to distinctive actions and then classify each motion segment using our annotation vocabulary. In addition, our system can also annotate the input motion sequence using semantic information embedded in our model, for example, counting the number of walking steps and identifying when and where to pick up and place an object.

Semantic motion editing. After we extract semantic information embedded in the input motion data with our motion analysis process, we can edit the motion at the semantic level, for example, replacing the "walking" with the "backward walking," inserting new actions "climbing up" and "climbing down" to pick up an object placed on an extremely high place, and inserting new actions "kneeling down" and "kneeling up" to pick up an object on the ground. The accompanying video shows the results obtained by the semantic motion editing process.

Semantic motion synthesis. Semantic motion synthesis allows the user to accurately and interactively control a human character by simply issuing high-level control commands such as "walking four steps to reach a point (A)," "jumping twice to a point (B)," and "picking up the object at (C)" (see Figure 1(right)). Animation and control of human characters at the semantic level is particularly important for novice users or autonomous agents, as it allows them to focus on high-level tasks rather than low-level kinematic details.

5.2 Online Motion Control

The online motion control system offers precise, realtime control over human characters with any kinematic constraints. We have demonstrated the flexibility and effectiveness of our online motion control system in a number of realtime applications.

Speed control and seamless behavior transitions. The system allows the user to control the speed of the character without worrying about transitions between different behaviors. For example, gradually speeding up the character automatically invokes transitions from "slow walking" to "fast walking," and then to "running." The character can also automatically switch to an appropriate jumping motion in order to avoid obstacles while matching the speed specified by the user.

Following stepping stones. In this example, the character is asked to follow the stepping stones in front of her. The user does not need to specify any contact constraints throughout the whole motion control process. The stepping stones are arranged to correspond to different step sizes, terrain heights, and turning angles.

Punching control. This experiment shows that the user can control punching actions in realtime by directly moving the target point in 3D space.

Direction control. This application demonstrates that the user can accurately control the direction of a walking character with very little latency. The accompanying video shows that the character can make a sharp turn, e.g., -180-degree or +180-degree, within the three steps.

5.3 Performance-based Animation Control

Performance-based animation is particularly appealing to generating detailed stylistic motions for human characters. The accompanying video shows that the user can synthesize and control a wide variety of stylistic walking motions, including "goosestep," "lame walking," "sneaky walking," and "crab walking," via performance-based control interfaces. To achieve this, we attach several retro-markers on the subject and reconstruct the 3D trajectories of the markers using vision-based tracking techniques similar to techniques described in [Chai and Hodgins 2005]. The system then automatically generates realistic animation that precisely matches input trajectory constraints.

5.4 Offline Motion Design

This experiment demonstrates that a novice can use a combination of *offline motion synthesis* and *continuous style editing* techniques to incrementally create natural-looking animation for human characters. In the accompanying video, we show the whole process for

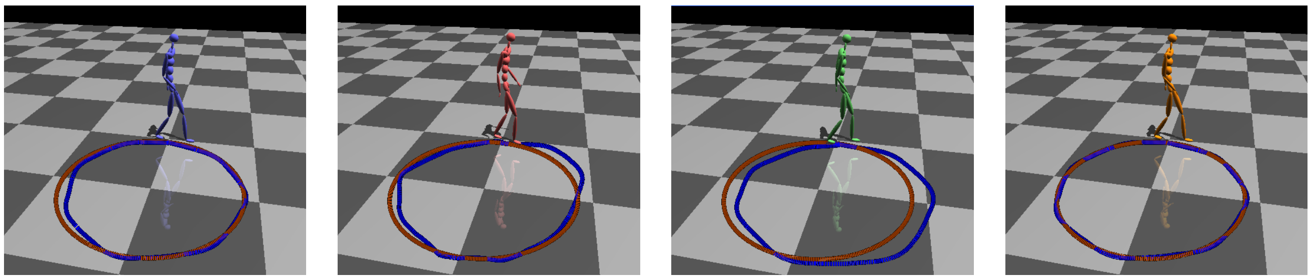

Figure 3: *Comparisons against motion graphs and motion interpolations. The results generated by motion graphs, motion interpolations I, motion interpolations II and our method are shown in left to right. Note that we visualize user-specified paths and synthesized paths in blue and maroon, respectively.*

creating a long animation sequence in a large and complex environment, including "normal walking," "sneaky walking," "jumping on stepping stones," and "sitting on a chair." For user's convenience, various motion control and editing tools have been developed, including action selection, path control, keyframe editing, and environmental object manipulation.

5.5 Comparisons Against Alternative Methods

We have evaluated the effectiveness of our algorithm by comparing with motion graphs and motion interpolations. Our comparison focuses on only motion synthesis and control because neither motion graphs nor motion interpolations is suitable for motion analysis. We evaluate the methods based on three criteria: *control accuracy and flexibility*, *motion quality* and *memory requirement*. The evaluation is based on four specific applications: "following the path along a circle (A)," "stylized walking with performance interfaces (B)," "walking and turning (C)," and "reaching the destination as soon as possible (D)."

Our motion graphs implementation is based on the technique described in [Kovar et al. 2002]. And we use the method described in [Kovar and Gleicher 2004] to do motion interpolations. Note that current motion interpolation techniques (e.g., [Kovar and Gleicher 2004] and [Mukai and Kuriyama 2005]) are often focused on interpolations between motion examples rather than transitions from one example to another one. We, therefore, implement two versions of motion interpolations algorithms. "Motion interpolation I" uses motion blending techniques described in [Rose et al. 1998] to create motion transitions. "Motion interpolation II" considers the transition smoothness from one segment to another one. More specifically, when we compute the distance between the current motion segment and a database motion segment for interpolations, we consider not only their distance in the control space (e.g., "walking direction") but also how well the database motion segment can be concatenated to the previously synthesized motion segment. The system interpolates the database motion examples that can only be smoothly concatenated to the previous synthesized motion segment. For both interpolation techniques, we apply inverse kinematics techniques to remove foot sliding artifacts.

Control accuracy and flexibility. An ideal motion synthesis system should allow the users to accurately control an animated character at different levels in order to accommodate the users with different skill levels. The experiment clearly shows that our method allows for more accurate motion control than both motion graphs and motion interpolations at both the kinematics and behavior levels. Table 1 reports the control errors obtained from all the three methods. For example, for kinematics control command (A) "following the path along a circle" shown in Figure 3, our control error is 2 cm per frame and this is much smaller than those from motion

Motion types	A	B	C	D_1	D_2
Motion graphs	11	24	8.21	4.7 ± 1.1	8.4 ± 1.0
Interp. I	14	22	4.48	N/A	N/A
Interp. II	31	31	7.56	N/A	N/A
Our method	2	6	3.24	3.1 ± 0.7	6.9 ± 0.6

Table 1: *Comparisons of control accuracy. The control error units for motion A, B, C, D_1, and D_2 are cm per frame, cm per frame, degrees per turn, seconds and (seconds), respectively.*

graphs (11 cm per frame) and both interpolation methods (14 cm per frame and 31 cm per frame). We have observed that motion interpolations often produce large control errors when continuous constraints (e.g., following a path or performance interfaces) are used for animation control.

For high-level motion control (D) "reaching the destination as soon as possible," we fix the starting point of the character and then randomly sample 20 destination points at a specific distance from the starting point. Column D_1 and D_2 in Table 1 report the mean and standard deviation of response times corresponding to the distance of 10 and 20 meters, respectively. As expected, our method produces a motion that reaches the final destination much faster than motion graphs. Note that both motion interpolations methods cannot achieve the goal specified by the user because they lack a planning scheme for high-level behavior control.

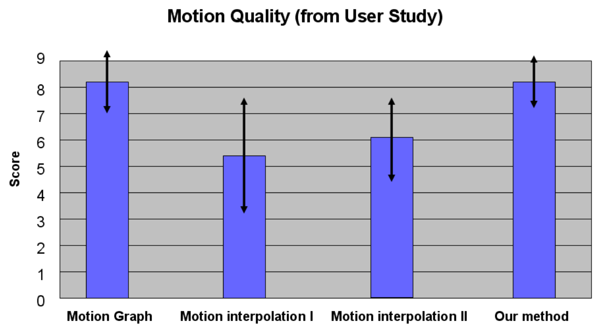

Figure 4: *Comparisons of motion quality. We asked the users to give a score (1-9) of how realistic synthesized motions are from each method. This graph shows the results of these scores, including means and standard deviations for motions generated by each method.*

Motion quality. We have also compared the quality of synthesized motions via user studies. We tested fourteen users, including seven

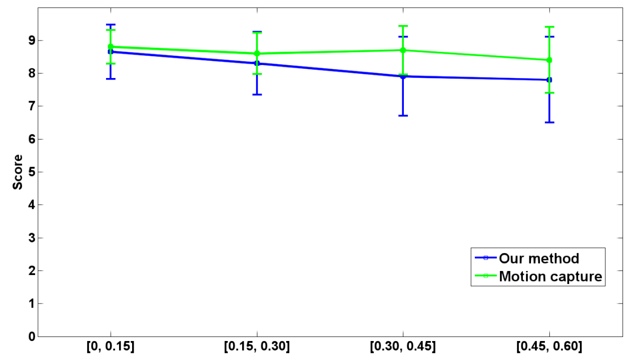

Figure 5: *Evaluation on motion generalization via user studies. The graph compares the quality of synthesized motions against that of ground truth motion capture data. The evaluation is based on animation with different degrees of user constraints variations (terrain heights). We randomly generate* 30 *terrain heights within each of terrain-height ranges: [0, 0.15], [0.15, 0.30], [0.30, 0.45], and [0.45, 0.60] (in meter); we then generate a motion for each specific terrain height and evaluate its motion quality via user study.*

males and seven females. Each user has either little or no previous experience with 3D animation. We rendered animation on a stick figure (See Figure 3). We randomly organized 20 animation clips generated by all the four algorithms and presented them on a projection screen in a small conference room to the users. Participants were instructed to provide a score of how realistic synthesized motions are using a rating scale that ranged from 0 ("least realistic") through 9 ("most realistic"). We report mean scores and standard deviations for motions generated by each method in Figure 4. The user studies show that motions generated by our method are highly realistic and in the same level as those from motion graphs. The motions generated by both interpolations are given lower scores, mainly because of poor transitions between interpolated motions. Among the two interpolation techniques, "motion interpolation II" produces more natural looking animations because it considers transition smoothness in motion interpolations.

Memory size. Computers or game consoles, particularly mobile devices, have a limited amount of physical memory and are often constrained in the amount of bandwidth available. A data-driven model for an animated character should be as compact as possible in order to minimize the consumption of memory and bandwidth. Here, we compare memory usage of each model. Our model is much more compact than motion graphs and motion interpolations. Our model consumes only about 16Mb memory for the whole motion database used in our experiment. In contrast, motion interpolations consume about 250Mb memory for the same database because it needs to retain all the original data for interpolation. Among the three models, motion graphs model consumes the largest amount of memory because it stores both the database poses and the edge connections between all the poses. The motion graphs model we constructed uses only motion capture data of walking, running, and their transitions, and the whole graph consumes about 137Mb memory.

5.6 Evaluation on Motion Generalization

One advantage of our motion models is their ability to generate motions that are not in a prerecorded motion database. We have studied the generalizability of our motion model via user studies. Here we focus our evaluation on one particular animation task "walking on uneven terrain" by assessing the quality of synthesized motions corresponding to terrain heights ranging between 0 to 0.6 meters. Specifically, we randomly generate 30 terrain heights within each of terrain-height ranges: [0, 0.15], [0.15, 0.30], [0.30, 0.45], and [0.45, 0.60] (in meter). We then generate a motion for each specific terrain height and evaluate its motion quality via user study. For each range of terrain heights, we also pick 30 random motion sequences from the prerecorded motion database. The mean scores and standard variations are reported in Figure 5. The user study result shows that the quality of synthesized motions is comparable to that of motion capture data. And increasing the degree of user constraints variation does not dramatically affect the quality of generalized motions. For reference, the percentage of database motions corresponding to each of terrain-height ranges [0, 0.15], [0.15, 0.30], [0.30, 0.45], and [0.45, 0.60] is about 70%, 12%, 10%, and 8%, respectively.

5.7 More Evaluations

In addition, we have done experiments to evaluate the performance of our system in terms of the importance of GP transition models and the number of samples for probabilistic motion planning.

Number of samples. The companying video shows a side-by-side comparison for motions generated with a different number of samples required for probabilistic motion planning. We have observed that control errors decreases as the number of samples increases. However, more samples also mean more expensive CPU time. The number of samples depends on the tradeoff between control accuracy and computational time. In our experiment, we found 300 samples are often sufficient to allow for accurate motion control while still keeping the system running at interactive frame rates.

Importance of GP transition models. In the case of "without GP transition models," we exclude GP transition models for selecting and concatenating morphable motions. More specifically, given a path of morphable nodes from motion planning, we randomly sample the morphable parameters and select motions clips based on how well the synthesized motions match user-specified constraints and how smooth the concatenated motions are at transition frames. Next, we use motion blending techniques to create transitions from one motion segment to another one. Lastly, we apply inverse kinematics to remove foot sliding artifacts. The comparison video clearly shows that GP transition models improve the realism of synthesized motions, particularly around transition frames. For a fair comparison, both motions follow the same path of morphable nodes and use the same number of samples (300) for probabilistic sampling.

6 Conclusion and Discussion

We have introduced a generative statistical model to analyze and synthesize a rich repertoire of human activities. Our models are appealing for motion analysis and synthesis because they are *compact*, *highly structured*, *contact aware*, *semantic embedding*, and *scalable* to huge and heterogeneous datasets. We have demonstrated the power and effectiveness of our models by exploring a wide variety of exciting applications, ranging from automatic motion segmentation, recognition, and annotation, and online/offline motion synthesis in both kinematics and behavior level to semantic motion editing.

Unlike current generative statistical models, motions generated by our model are always highly realistic. One reason is that morphable graphs models are highly structured: they not only preserve local structures within each motion primitive but also model natural transitions between different motion primitives. Our model is also contact aware and embedded with contact information, thereby removing unpleasant visual artifacts often present in the motion generalization process.

Our morphable graphs model decomposes each human action into distinctive motion primitives and encodes structural variation using a graph of motion primitives. As a result, traversing a different graph path produces a motion containing a different high-level structure. We have demonstrated structure variation caused by 16 distinctive actions and their transitions. We have also demonstrated that our system can generate a wide range of style variations, including functional variations (e.g., locomotion speeds, step sizes, uneven terrains, and turning angles) and stylistic variations such as "goosestep," "sneaky walking," and "lame walking." This is because our morphable motion primitives are capable of modeling an infinite number of style variations associated with each motion primitive.

Our model is appealing for motion control because it allows the user to accurately control a realistic human character at different levels. For example, a novice can focus on which actions to perform without concerning styles or kinematic details of synthesized motions. A more skillful user can precisely control spatial-temporal variations of complex human actions with detailed kinematic constraints. Motion control at the semantic level is particularly enticing for building intelligent and autonomous characters because we can focus on our attention on behavior planning rather than kinematics details. One of the immediate directions for future work is, therefore, to investigate the application of our algorithm in generating autonomous agents or intelligent crowds.

One limitation of our algorithm is that it is data-driven and therefore cannot generate new motions that cannot be represented by the morphable graphs model. For instance, the current morphable graphs model cannot generate a motion "walking with head scratching" because similar patterns are not included in the current database. One possibility to address this limitation is to capture or keyframe a motion "walking with head scratching," register the motion with the generated motion using the low-body motion data, and transfer the upper-body pattern "head scratching" to the generated motion.

Acknowledgement

This work was supported by the National Science Foundation under Grants No. IIS-1055046 and IIS-1065384.

References

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

ARIKAN, O., FORSYTH, D. A., AND O'BRIEN, J. F. 2003. Motion synthesis from annotations. In *ACM Transactions on Graphics*. 22(3):402–408.

BEAUDOIN, P., COROS, S., VAN DE PANNE, M., AND POULIN, P. 2008. Motion-motif graphs. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA'08, 117–126.

BISHOP, C. 1996. *Neural Network for Pattern Recognition*. Cambridge University Press.

BOWDEN, R. 2000. Learning statistical models of human motion. In *IEEE Workshop on Human Modelling, Analysis and Synthesis, CVPR2000*.

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. In *ACM Transactions on Graphics*. 24(3):686–696.

CHAI, J., AND HODGINS, J. 2007. Constraint-based motion optimization using a statistical dynamic model. In *ACM Transactions on Graphics*. 26(3):Article No.8.

GLEICHER, M., SHIN, H. J., KOVAR, L., AND JEPSEN, A. 2003. Snap-together motion: assembling run-time animations. In *Proceedings of the 2003 symposium on Interactive 3D graphics*, I3D'03, 181–188.

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

HECK, R., AND GLEICHER, M. 2007. Parametric motion graphs. In *Proceedings of the 2007 symposium on Interactive 3D graphics and games*. 129–136.

HSU, E., PULLI, K., AND POPOVIĆ, J. 2005. Style translation for human motion. In *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, 1–12.

KOVAR, L., AND GLEICHER, M. 2003. Flexible automatic motion blending with 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. In *ACM Transactions on Graphics*. 23(3):559–568.

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

LAU, M., AND KUFFNER, J. J. 2005. Behavior planning for character animation. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA' 05, 271–280.

LAU, M., BAR-JOSEPH, Z., AND KUFFNER, J. 2009. Modeling spatial and temporal variation in motion data. In *ACM Transactions on Graphics*. 28(5): Article No. 171.

LAWRENCE, N. D., 2009. Gaussian process software. http://www.cs.manchester.ac.uk/neill/gp/.

LEE, J., CHAI, J., REITSMA, P., HODGINS, J., AND POLLARD, N. 2002. Interactive control of avatars animated with human motion data. In *ACM Transactions on Graphics*. 21(3):491–500.

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

LI, Y., WANG, T., AND SHUM, H.-Y. 2002. Motion texture: A two-level statistical model for character synthesis. In *ACM Transactions on Graphics*. 21(3):465–472.

LOURAKIS, M. I. A., 2009. levmar: Levenberg-Marquardt nonlinear least squares algorithms in C/C++.

MIN, J., CHEN, Y.-L., AND CHAI, J. 2009. Interactive generation of human animation with deformable motion models. *ACM Transactions on Graphics*. 29(1): article No. 9.

MOLINA TANCO, L., AND HILTON, A. 2000. Realistic synthesis of novel human movements from a database of motion capture examples. In *Proceedings of the Workshop on Human Motion*. 137-142.

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

QUINONERO-CANDELA, J., AND RASMUSSEN, C. E. 2005. A unifying view of sparse approximate gaussian process regression. *Journal of Machine Learning Research*. 6: 1935–1959.

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

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.

SAFONOVA, A., AND HODGINS, J. K. 2007. Construction and optimal search of interpolated motion graphs. In *ACM Transactions on Graphics*. 26(3): Article No. 106.

SHIN, H. J., AND OH, H. S. 2006. Fat graphs: constructing an interactive character with continuous controls. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '06, 291–298.

SHUM, H. P. H., KOMURA, T., SHIRAISHI, M., AND YAMAZAKI, S. 2008. Interaction patches for multi-character animation. *ACM Transactions on Graphics*. 27(5): Article No. 114.

TREUILLE, A., LEE, Y., AND POPOVIĆ, Z. 2007. Near-optimal character animation with continuous control. *ACM Transaction on Graphics*. 26(3): Article No. 7.

WEI, X., MIN, J., AND CHAI, J. 2011. Physically valid statistical models for human motion generation. *ACM Trans. Graph. 30*, 19:1–19:10.

YE, Y., AND LIU, K. 2010. Synthesis of responsive motion using a dynamic model. *Computer Graphics Forum (Proceedings of Eurographics)*. 29(2): 555-562.