

MERL – A MITSUBISHI ELECTRIC RESEARCH LABORATORY
<http://www.merl.com>

Style Machines

Matthew Brand
MERL
201 Broadway
Cambridge, MA 02139
brand@merl.com

Aaron Hertzmann
NYU Media Research Laboratory
719 Broadway
New York, NY 1003
hertzman@mrl.nyu.edu

TR-2000-14 April 2000

Abstract

We approach the problem of stylistic motion synthesis by learning motion patterns from a highly varied set of motion capture sequences. Each sequence may have a distinct choreography, performed in a distinct style. Learning identifies common choreographic elements across sequences, the different styles in which each element is performed, and a small number of stylistic degrees of freedom which span the many variations in the dataset. The learned model can synthesize novel motion data in any interpolation or extrapolation of styles. For example, it can convert novice ballet motions into the more graceful modern dance of an expert. The model can also be driven by video, by scripts, or even by noise to generate new choreography and synthesize virtual motion-capture in many styles.

In Proceedings of SIGGRAPH 2000, July 23-28, 2000. New Orleans, Louisiana, USA.

This work may not be copied or reproduced in whole or in part for any commercial purpose. Permission to copy in whole or in part without payment of fee is granted for nonprofit educational and research purposes provided that all such whole or partial copies include the following: a notice that such copying is by permission of Mitsubishi Electric Information Technology Center America; an acknowledgment of the authors and individual contributions to the work; and all applicable portions of the copyright notice. Copying, reproduction, or republishing for any other purpose shall require a license with payment of fee to Mitsubishi Electric Information Technology Center America. All rights reserved.

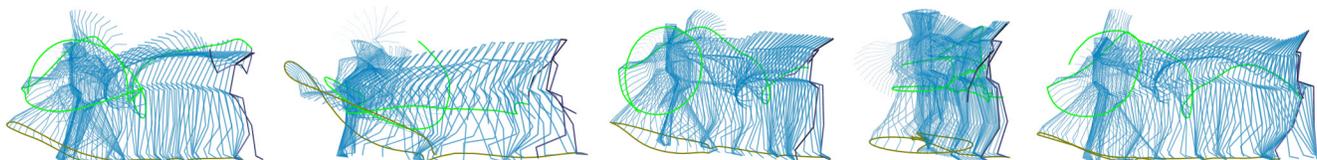
Copyright © Mitsubishi Electric Information Technology Center America, 2000
201 Broadway, Cambridge, Massachusetts 02139

Submitted December 1999, revised and release April 2000.

Style machines

Matthew Brand
Mitsubishi Electric Research Laboratory

Aaron Hertzmann
NYU Media Research Laboratory



A pirouette and promenade in five synthetic styles drawn from a space that contains ballet, modern dance, and different body types. The choreography is also synthetic. Streamers show the trajectory of the left hand and foot.

Abstract

We approach the problem of stylistic motion synthesis by learning motion patterns from a highly varied set of motion capture sequences. Each sequence may have a distinct choreography, performed in a distinct style. Learning identifies common choreographic elements across sequences, the different styles in which each element is performed, and a small number of stylistic degrees of freedom which span the many variations in the dataset. The learned model can synthesize novel motion data in any interpolation or extrapolation of styles. For example, it can convert novice ballet motions into the more graceful modern dance of an expert. The model can also be driven by video, by scripts, or even by noise to generate new choreography and synthesize virtual motion-capture in many styles.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation; I.2.9 [Artificial Intelligence]: Robotics—Kinematics and Dynamics; G.3 [Mathematics of Computing]: Probability and Statistics—Time series analysis; E.4 [Data]: Coding and Information Theory—Data compaction and compression; J.5 [Computer Applications]: Arts and Humanities—Performing Arts

Keywords: animation, behavior simulation, character behavior.

1 Introduction

It is natural to think of walking, running, strutting, trudging, sashaying, etc., as stylistic variations on a basic motor theme. From a directorial point of view, the style of a motion often conveys more meaning than the underlying motion itself. Yet existing animation tools provide little or no high-level control over the style of an animation.

In this paper we introduce the style machine—a statistical model that can generate new motion sequences in a broad range of styles, just by adjusting a small number of stylistic knobs (parameters). Style machines support synthesis and resynthesis in new styles, as well as style identification of existing data. They can be driven by many kinds of inputs, including computer vision, scripts, and noise. Our key result is a method for learning a style machine, including the number and nature of its stylistic knobs, from data. We use style machines to model highly nonlinear and nontrivial behaviors such as ballet and modern dance, working with very long unsegmented motion-capture sequences and using the learned model to generate new choreography and to improve a novice’s dancing.

Style machines make it easy to generate long motion sequences containing many different actions and transitions. They can offer a broad range of stylistic degrees of freedom; in this paper we show early results manipulating gender, weight distribution, grace, energy, and formal dance styles. Moreover, style machines can be learned from relatively modest collections of existing motion-capture; as such they present a highly generative and flexible alternative to motion libraries.

Potential uses include: **Generation:** Beginning with a modest amount of motion capture, an animator can train and use the resulting style machine to generate large amounts of motion data with new orderings of actions. **Casts of thousands:** Random walks in the machine can produce thousands of unique, plausible motion choreographies, each of which can be synthesized as motion data in a unique style. **Improvement:** Motion capture from unskilled performers can be resynthesized in the style of an expert athlete or dancer. **Retargetting:** Motion capture data can be resynthesized in a new mood, gender, energy level, body type, etc. **Acquisition:** Style machines can be driven by computer vision, data-gloves, even impoverished sensors such as the computer mouse, and thus offer a low-cost, low-expertise alternative to motion-capture.

2 Related work

Much recent effort has addressed the problem of editing and reuse of existing animation. A common approach is to provide interactive animation tools for motion editing, with the goal of capturing the style of the existing motion, while editing the content. Gleicher [11] provides a low-level interactive motion editing tool that searches for a new motion that meets some new constraints while minimizing the distance to the old motion. A related optimization method is also used to adapt a motion to new characters [12]. Lee et al. [15] provide an interactive multiresolution motion editor for fast, fine-scale control of the motion. Most editing

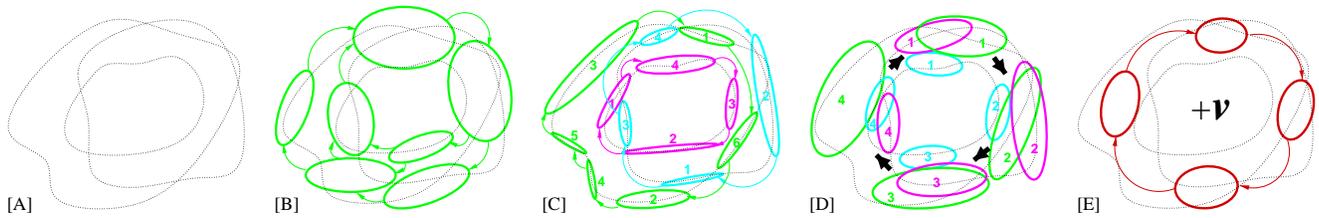


Figure 1: Schematic illustrating the effects of cross-entropy minimization. [A]. Three simple walk cycles projected onto 2-space. Each data point represents the body pose observed at a given time. [B]. In conventional learning, one fits a single model to all the data (ellipses indicate state-specific isoprobability contours; arcs indicate allowable transitions). But here learning is overwhelmed by variation among the sequences, and fails to discover the essential structure of the walk cycle. [C]. Individually estimated models are hopelessly overfit to their individual sequences, and will not generalize to new data. In addition, they divide up the cycle differently, and thus cannot be blended or compared. [D]. Cross-entropy minimized models are constrained to have similar qualitative structure and to identify similar phases of the cycle. [E]. The generic model abstracts all the information in the style-specific models; various settings of the style variable \boldsymbol{v} will recover all the specific models plus any interpolation or extrapolation of them.

systems produce results that may violate the laws of mechanics; Popović and Witkin [16] describe a method for editing motion in a reduced-dimensionality space in order to edit motions while maintaining physical validity. Such a method would be a useful complement to the techniques presented here.

An alternative approach is to provide more global animation controls. Signal processing systems, such as described by Bruderlin and Williams [7] and Unuma et al. [20], provide frequency-domain controls for editing the style of a motion. Witkin and Popović [22] blend between existing motions to provide a combination of motion styles. Rose et al. [18] use radial basis functions to interpolate between and extrapolate around a set of aligned and labeled example motions (e.g., happy/sad and young/old walk cycles), then use kinematic solvers to smoothly string together these motions. Similar functionality falls out of our framework.

Although such interactive systems provide fine control, they rely on the labors of skilled animators to produce compelling and convincing results. Furthermore, it is generally difficult to produce a new motion that is substantially different from the existing motions, in style or in content (e.g., to convert by hand a ponderous walk to a jaunty dance, etc.)

The above signal-processing methods also require that the example motions be time-warped; in other words, that sequential correspondences can be found between each component of each motion. Unfortunately, it is rarely the case that any set of complex motions will have this property. Style machines automatically compute flexible many-to-many correspondences between sequences using fast dynamic programming algorithms.

Our work unites two themes that have separate research histories in motion analysis: estimation of dynamical (in the sense of time-evolving) models from examples, and style and content separation. Howe et al. [14] analyze motion from video using a mixture-of-Gaussians model. Grzeszczuk et al. [13] learn control and physical systems from physical simulation. Several authors have used hidden Markov Models to analyze and synthesize motion. Bregler [6] and Brand [5] use HMMs to recognize and analyze motion from video sequences. Brand [4] analyzes and resynthesizes animation of human speech from example audio and video. With regard to styles, Wilson and Bobick [21] use parametric HMMs, in which motion recognition models are learned from user-labeled styles. Tenenbaum and Freeman [19, 10] separate style from content in general domains under a bilinear model, thereby modeling factors that have individually linear but cooperatively multiplicative effects on the output, e.g., the effects of lighting and pose in images of faces.

These style/content models depend on large sets of hand-labeled and hand-aligned samples (often exponential in the number of stylistic degrees of freedom DOFs) plus an explicit statement of what

the stylistic DOFs are. We now introduce methods for extracting this information directly and automatically from modest amounts of data.

3 Learning stylistic state-space models

We seek a model of human motion from which we can generate novel choreography in a variety of styles. Rather than attempt to engineer such a model, we will attempt to learn it—to extract from data a function that approximates the data-generating mechanism.

We cast this as an unsupervised learning problem, in which the goal is to acquire a generative model that captures the data’s essential structure (traces of the data-generating mechanism) and discards its accidental properties (particulars of the specific sample). Accidental properties include noise and the bias of the sample. Essential structure can also be divided into two components, which we will call structure and style. For example, walking, running, strutting, etc., are all stylistic variations on bipedal locomotion, a dynamical system with particularly simple temporal structure—a deterministic loop.

It is up to the modeler to make the structure/style distinction. State-space representations are very useful here: We take the *structure* of bipedal locomotion to be a small set of dynamically-significant qualitative states along with the rules that govern changes of state. We take *style* to be variations in the mapping from qualitative states to quantitative observations. For example, shifting one’s weight load onto the right leg is a dynamically-significant state common to all forms of bipedal locomotion, but it will look quite different in running, trudging, etc.

An appropriate state-space model for time-series data is the hidden Markov model (HMM). An HMM is a probabilistic finite-state machine consisting of a set of discrete states, state-to-state transition probabilities, and state-to-signal emission probabilities—in this paper, each state has a Gaussian distribution over a small space of full-body poses and motions. (See §A for a concise HMM tutorial; see [17] for a detailed tutorial.) We will add to the HMM a multidimensional style variable \boldsymbol{v} that can be used to vary its parameters, and call the result a *stylistic* HMM (SHMM), or time-series **style machine**. (See §B for formal definitions.) The SHMM defines a space of HMMs; fixing the parameter \boldsymbol{v} yields a unique HMM.

Here we show how to separate structure, style, and accidental properties in a dataset by minimizing entropies in the SHMM. The main advantages of separating style from structure is that we wind up with simpler, more generative models for both, and we can do so with significantly less data than required for the general learning setting. Our framework is fully unsupervised and automatically identifies the number and nature of the stylistic degrees of freedom (often much fewer than the number of variations in the dataset).

The discovered degrees of freedom lend themselves to some intuitive operations that are very useful for synthesis: style mixtures, exaggerations, extrapolations, and even analogies.

3.1 Generic and style-specific models

We begin with a family of training samples. By “family” we mean that all the samples have some generic data-generating mechanism in common, e.g., the motor program for dancing. Each sample may instantiate a different variation. The samples need not be aligned, e.g., the ordering, timing, and appearance of actions may vary from sample to sample. Our modeling goal is to extract a single parameterized model which covers the generic behavior in the entire family of training samples, and which can easily be made to model an individual style, combination of styles, or extrapolation of styles, just by choosing an appropriate setting of the style variable \mathbf{v} .

Learning involves the simultaneous estimation of a generic model and a set of style-specific models with three objectives: (1) each model should fit its sample(s) well; (2) each specific model should be close to the generic model; and (3) the generic model should be as simple as possible, thereby maximizing probability of correct generalization to new data. These constraints have an information-theoretic expression in eqn. 1. In the next section we will explain how the last two constraints interact to produce a third desirable property: The style-specific models can be expressed as small variations on the generic model, and the space of such variations can be captured with just a few parameters.

We first describe the use of style machines as applied to pure signal data. Details specific to working with motion-capture data are described in §4.

3.2 Estimation by entropy minimization

In learning we minimize a sum of entropies—which measure the ambiguity in a probability distribution—and cross-entropies—which measure the divergence between distributions. The principle of minimum entropy, advocated in various forms by [23, 2, 8], seeks the simplest model that explains the data, or, equivalently, the most complex model whose parameter estimates are fully supported by the data. This maximizes the information extracted from the training data and boosts the odds of generalizing correctly beyond it.

The learning objective has three components, corresponding to the constraints listed above:

1. The cross-entropy between the model distribution and statistics extracted from the data measures the model’s misfit of the data.
2. The cross-entropy between the generic and a specific model measures inconsistencies in their analysis of the data.
3. The entropy of the generic model measures ambiguity and lack of structure in its analysis of the data.

Minimizing #1 makes the model faithful to the data. Minimizing #2 essentially maximizes the overlap between the generic and specific models and congruence (similarity of support) between their hidden states. This means that the models “behave” similarly and their hidden states have similar “meanings.” For example, in a dataset of bipedal motion sequences, all the style-specific HMMs should converge to similar finite-state machine models of the locomotion cycle, and corresponding states in each HMM to refer to qualitatively similar poses and motions in the locomotion cycle. E.g., the n th state in each model is tuned to the poses in which the body’s weight shifts onto the right leg, regardless of the style of motion (see figure 1). Minimizing #3 optimizes the predictiveness of the model by making sure that it gives the clearest and most concise picture of the data, with each hidden state explaining a clearly delineated phenomenon in the data.



Figure 2: Flattening and alignment of Gaussians by minimization of entropy and cross-entropy, respectively. Gaussian distributions are visualized as ellipsoid iso-probability contours.

Putting this all together gives the following learning objective function

$$\theta^* = \arg \min_{\theta} \underbrace{H(\omega)}_{\text{data entropy}} + \underbrace{D(\omega||\theta)}_{\text{misfit}} + \underbrace{H(\theta)}_{\text{3:model entropy}} + \underbrace{D(\theta^*||\theta)}_{\text{2:incongruence}} + \dots \quad (1)$$

-log posterior

1:-log likelihood -log prior

where θ is a vector of model parameters; ω is a vector of expected sufficient statistics describing the data \mathbf{X} ; θ^* parameterizes a reference model (e.g., the generic); $H(\cdot)$ is an entropy measure; and $D(\cdot)$ is a cross entropy measure.

Eqn. 1 can also be formulated as a Bayesian posterior $P(\theta|\omega) \propto P(\omega|\theta)P(\theta)$ with likelihood function $P(\mathbf{X}|\theta) \propto e^{-H(\omega)-D(\omega||\theta)}$ and a prior $P(\theta) \propto e^{-H(\theta)-D(\theta^*||\theta)}$. The data entropy term, not mentioned above, arises in the normalization of the likelihood function; it measures ambiguity in the data-descriptive statistics that are calculated *vis-à-vis* the model.

3.3 Effects of the prior

It is worth examining the prior because this is what will give the final model θ^* its special style-spanning and generative properties.

The prior term $e^{-H(\theta)}$ expresses our belief in the parsimony principle—a model should give a maximally concise and minimally uncertain explanation of the structure in its training set. This is an optimal bias for extracting as much information as possible from the data [3]. We apply this prior to the generic model. The prior has an interesting effect on the SHMM’s emission distributions over pose and velocity: It gradually removes dimensions of variation, because flattening a distribution is the most effective way to reduce its volume and therefore its entropy (see figure 2).

The prior term $e^{-D(\theta^*||\theta)}$ keeps style models close and congruent to the generic model, so that corresponding hidden states in two models have similar behavior. In practice we assess this prior only on the emission distributions of the specific models, where it has the effect of keeping the variation-dependent emission distributions clustered tightly around the generic emission distribution. Consequently it minimizes distance between corresponding states in the models, not between the entire models. We also add a term $-T' \cdot D(\theta^*||\theta)$ that allows us to vary the strength of the cross-entropy prior in the course of optimization.

By constraining generic and style-specific Gaussians to overlap, and constraining both to be narrow, we cause the distribution of state-specific Gaussians across styles to have a small number of degrees of freedom. Intuitively, if two Gaussians are narrow and overlap, then they must be aligned in the directions of their narrowness (e.g., two overlapping disks in 3-space must be co-planar). Figure 2 illustrates. The more dimensions in which the overlapping Gaussians are flat, the fewer degrees of freedom they have relative to each other. Consequently, as style-specific models are drawn toward the generic model during training, all the models settle into a parameter subspace (see figure 3). Within this subspace, all the variation between the style-specific models can be described with just a few parameters. We can then identify those degrees of freedom by solving for a smooth low-dimensional manifold that contains the parameterizations of all the style-specific models. Our

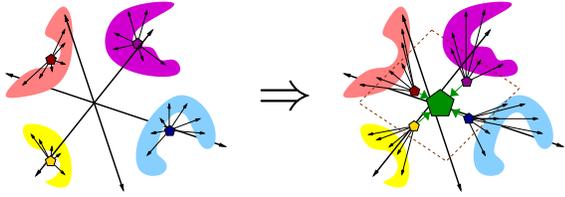


Figure 3: Schematic of style DOF discovery. LEFT: Without cross-entropy constraints, style-specific models (pentagons) are drawn to their data (clouds), and typically span all dimensions of parameter space. RIGHT: When also drawn to a generic model, they settle into a parameter subspace (indicated by the dashed plane).

experiments showed that a linear subspace usually provides a good low-dimensional parameterization of the dataset’s stylistic degrees of freedom. The subspace is easily obtained from a principal components analysis (PCA) of a set of vectors, each representing one model’s parameters.

It is often useful to extend the prior with additional functions of θ . For example, adding $-T \cdot H(\theta)$ and varying T gives deterministic annealing, an optimization strategy that forces the system to explore the error surface at many scales as $T \downarrow 0$ instead of myopically converging to the nearest local optimum (see §C for equations).

3.4 Optimization

In learning we hope to simultaneously segment the data into motion primitives, match similar primitives executed in different styles, and estimate the structure and parameters for minimally ambiguous, maximally generative models. Entropic estimation [2] gives us a framework for solving this partially discrete optimization problem by embedding it in a high-dimensional continuous space via entropies. It also gives us numerical methods in the form of maximum a posteriori (MAP) entropy-optimizing parameter estimators. These attempt to find a best data-generating model, by gradually extinguishing excess model parameters that are not well-supported by the data. This solves the discrete optimization problem by causing a diffuse distribution over all possible segmentations to collapse onto a single segmentation.

Optimization proceeds via Expectation-Maximization (EM) [1, 17], a fast and powerful fixpoint algorithm that guarantees convergence to a local likelihood optimum from any initialization. The estimators we give in §D modify EM to do cross-entropy optimization and annealing. Annealing strengthens EM’s guarantee to quasi-global optimality—global MAP optimality with probability approaching 1 as the annealing schedule lengthens—a necessary assurance due to the number of combinatorial optimization problems that are being solved simultaneously: segmentation, labeling, alignment, model selection, and parameter estimation.

The full algorithm is:

1. Initialize a generic model and one style-specific model for each motion sequence.
2. EM loop until convergence:
 - (a) E step: Compute expected sufficient statistics ω of each motion sequence relative to its model.
 - (b) M step: (generic): Calculate maximum *a posteriori* parameter values $\hat{\theta}^*$ with the minimum-entropy prior, using E-step statistics from the entire training set.
 - (c) M step: (specific): Calculate maximum *a posteriori* parameter values $\hat{\theta}$ with the minimum-cross-entropy prior, only using E-step statistics from the current sequence.

- (d) Adjust the temperature (see below for schedules).

3. Find a subspace that spans the parameter variations between models. E.g., calculate a PCA of the differences between the generic and each style-specific model.

Initialization can be random because full annealing will obliterate initial conditions. If one can encode useful hints in the initial model, then the EM loop should use partial annealing by starting at a lower temperature.

HMMs have a useful property that saves us the trouble of hand-segmenting and/or labelling the training data: The actions in any particular training sequence may be squashed and stretched in time, oddly ordered, and repeated; in the course of learning, the basic HMM dynamic programming algorithms will find an optimal segmentation and labelling of each sequence. Our cross-entropy prior simply adds the constraint that similarly-labeled frames exhibit similar behavior (but not necessarily appearance) across sequences. Figure 4 illustrates with the induced state machine and labelling of four similar but incongruent sequences, and an induced state machine that captures all their choreographic elements and transitions.

4 Working with Motion Capture

As in any machine-learning application, one can make the problem harder or easier depending on how the data is represented to the algorithm. Learning algorithms look for the most *statistically* salient patterns of variation the data. For motion capture, these may not be the patterns that humans find *perceptually* and *expressively* salient. Thus we want to preprocess the data to highlight sources of variation that “tell the story” of a dance, such as leg-motions and compensatory body motions, and suppress irrelevant sources of variation, such as inconsistent marker placements and world coordinates between sequences (which would otherwise be modeled as stylistic variations). Other sources of variation, such as inter-sequence variations in body shapes, need to be scaled down so that they do not dominate style space. We now describe methods for converting raw marker data into a suitable representation for learning motion.

4.1 Data Gathering and Preprocessing

We first gathered human motion capture data from a variety of sources (see acknowledgements in §8). The data consists of the 3D positions of physical markers placed on human actors, acquired over short intervals in motion capture studios. Each data source provided data with a different arrangement of markers over the body. We defined a reduced 20 marker arrangement, such that all of the markers in the input sequences could be converted by combining and deleting extra markers. (Note that the missing markers can be recovered from synthesized data later by remapping the style machines to the original input marker data.) We also doubled the size of the data set by mirroring, and resampled all sequences to 60Hz.

Captured and synthetic motion capture data in the figures and animations show the motions of markers connected by a fake skeleton. The “bones” of this skeleton have no algorithmic value; they are added for illustration purposes only to make the markers easier to follow.

The next step is to convert marker data into joint angles plus limb lengths, global position and global orientation. The coccyx (near the base of the back) is used as the root of the kinematic tree. Joint angles alone are used for training. Joint angles are by nature periodic (for example, ranging from 0 to 2π); because training assumes that the input signal lies in the infinite domain of \mathbf{R}^n , we took some pain to choose a joint angle parameterization without

discontinuities (such as a jump from 2π to 0) in the training data.¹ However, we were not able to fully eliminate all discontinuities. (This is partially due to some perversities in the input data such as inverted knee bends.)

Conversion to joint angles removes information about which articulations cause greatest changes in body pose. To restore this information, we scale joint angle variables to make statistically salient those articulations that most vary the pose, measured in the data set by a procedure similar to that described by Gleicher [11]. To reduce the dependence on individual body shape, the mean pose is subtracted from each sequence. Finally, noise and dimensionality are reduced via PCA; we typically use ten or fewer significant dimensions of data variation for training.

4.2 Training

Models are initialized with a state transition matrix $P_{j \rightarrow i}$ that has probabilities declining exponentially off the diagonal; the Gaussians are initialized randomly or centered on every n th frame of the sequence. These initial conditions save the learning algorithm the gratuitous trouble of selecting from among a factorial number of permutationally equivalent models, differing only in the ordering of their states.

We train with annealing, setting the temperature T high and making it decay exponentially toward zero. This forces the estimators to explore the error surface at many scales before committing to a particular region of parameter space. In the high-temperature phase, we set the cross-entropy temperature T' to zero, to force the variation models to stay near the generic model. At high temperatures, any accidental commitments made in the initialization are largely obliterated. As the generic temperature declines, we briefly heat up the cross-entropy temperature, allowing the style-specific models to venture off to find datapoints not well explained by the generic model. We then drive both temperatures to zero and let the estimators converge to an entropy minimum.

These temperature schedules are hints that guide the optimization: (1) Find global structure; (2) offload local variation to the specific models; (3) then simplify (compress) all models as much as possible.

The result of training is a collection of models and for each model, a distribution γ over its hidden states, where $\gamma_{t,i}(\mathbf{y}) = p(\text{state } i \text{ explains frame } t, \text{ given all the information in the sequence } \mathbf{y})$. Typically this distribution has zero or near-zero entropy, meaning that γ has collapsed to a single state sequence that explains the data. γ (or the sequence of most probable states) encodes the *content* of the data; as we show below, applying either one to a different style-specific model causes that content to be resynthesized in a different style.

We use γ to remap each model's emission distributions to joint angles and angular velocities, scaled according to the importance of each joint. This information is needed for synthesis. Remapping means re-estimating emission parameters to observe a time-series that is synchronous with the training data.

4.3 Making New Styles

We encode a style-specific HMM in a vector by concatenating its state means μ_i , square-root covariances ($\mathbf{K}_{ij} / \sqrt{|\mathbf{K}_{ij}|}$, for $i \leq j$), and state dwell times (on average, how long a model stays in one state before transitioning out). New styles can be created by interpolation and extrapolation within this space. The dimensionality of the space is reduced by PCA, treating each HMM as a single observation and the generic HMM as the origin. The PCA gives us a

subspace of models whose axes are intrinsic degrees of variation across the styles in the training set. Typically, only a few stylistic DOFs are needed to span the many variations in a training set, and these become the dimensions of the style variable \mathbf{v} . One interpolates between any styles in the training set by varying \mathbf{v} between the coordinates of their models in the style subspace, then reconstituting a style-specific HMM from the resulting parameter vector. Of course, it is more interesting to extrapolate, by going outside the convex hull of the training styles, a theme that is explored below in §5.

4.4 Analyzing New Data

To obtain the style coordinates of a novel motion sequence \mathbf{y} , we begin with a copy of the generic model (or of a style-specific model which assigns \mathbf{y} high likelihood), then retrain that model on \mathbf{y} , using cross-entropy constraints with respect to the original generic model. Projection of the resulting parameters onto the style manifold gives the style coordinates. We also obtain the sample's state occupancy matrix γ . As mentioned before, this summarizes the content of the motion sequence \mathbf{y} and is the key to synthesis, described below.

4.5 Synthesizing Virtual Motion Data

Given a new value of the style variable \mathbf{v} and a state sequence $S(\mathbf{y}) \in \gamma(\mathbf{y})$ encoding the content of \mathbf{y} , one may resynthesize \mathbf{y} in the new style \mathbf{y}' by calculating the maximum-likelihood path $\arg \max_{\mathbf{y}'} p(\mathbf{y}' | \mathbf{v}, S(\mathbf{y}))$. Brand [4] describes a method for calculating the maximum-likelihood sample in $O(T)$ time for T time-steps. §F generalizes and improves on this result, so that all the information in γ is used.

This resulting path is an inherently smooth curve that varies even if the system dwells in the same hidden state for several frames, because of the velocity constraints on each frame. Motion discontinuities in the synthesized samples are possible if the difference in velocities between successive states is large relative to the frame (sampling) rate. The preprocessing steps are then reversed to produce virtual motion-capture data as the final output.

Some actions take longer in different styles; as we move from style to style, this is accommodated by scaling dwell times of the state sequence to match those of the new style. This is one of many ways of making time flexible; another is to incorporate dwell times directly into the emission distributions and then synthesize a list of varying-sized time-steps by which to clock the synthesized motion-capture frames.

In addition to resynthesizing existing motion-capture in new styles, it is possible to generate entirely new motion data directly from the model itself. Learning automatically extracts motion primitives and motion cycles from the data (see figure 4), which take the form of state sub-sequences. By cutting and pasting these state sequences, we can sequence new choreography. If a model's state machine has an arc between states in two consecutively scheduled motion primitives, the model will automatically modify the end and beginning of the two primitives to transition smoothly. Otherwise, we must find a path through the state machine between the two primitives and insert all the states on that path. An interesting effect can also be achieved by doing a random walk on the state machine, which generates random but plausible choreography.

5 Examples

We collected a set of **bipedal locomotion time-series** from a variety of sources. These motion-capture sequences feature a variety of different kinds of motion, body types, postures, and marker placements. We converted all motion data to use a common set of markers on a prototypical body. (If we do not normalize the body,

¹For cyclic domains, one would ideally use von Mises' distribution, essentially a Gaussian wrapped around a circle, but we cannot because no analytic variance estimator is known.

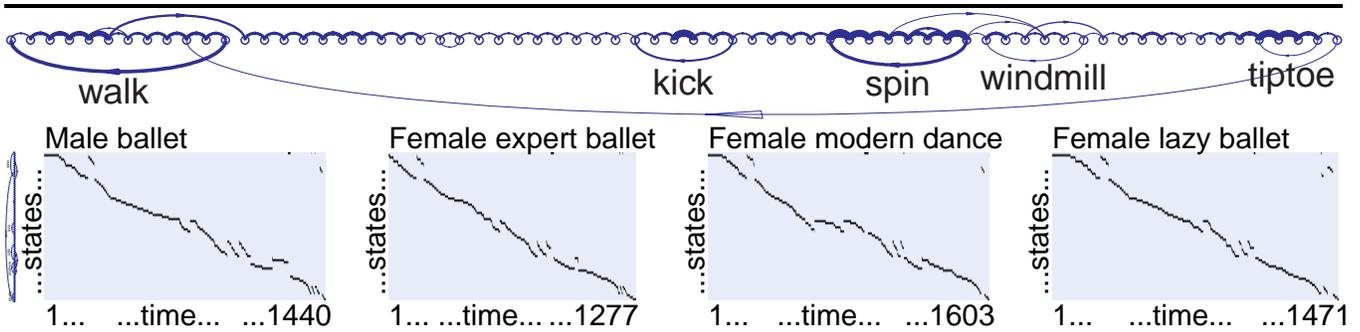


Figure 4: TOP: State machine learned from four dance sequences totalling 6000+ frames. Very low-probability arcs have been removed for clarity. Motion cycles have been labeled; other primitives are contained in linear sequences. BOTTOM: Occupancy matrices (constructed while learning) indicate how each sequence was segmented and labeled. Note the variations in timing, ordering, and cycles between sequences.

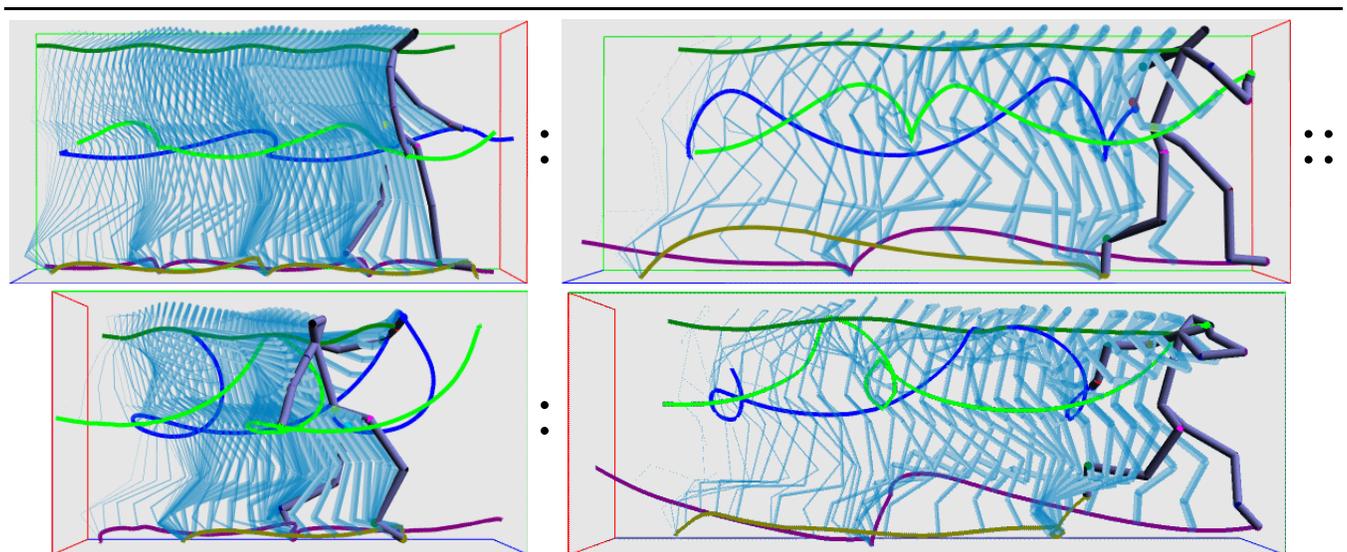


Figure 5: Completion of the analogy walking:running::strutting:X via synthesis of stylistic motion. Stick figures show every 5 frames; streamers show the trajectories of the extremities. X extrapolates both the energetic arm swing of strutting and the power stride of running.

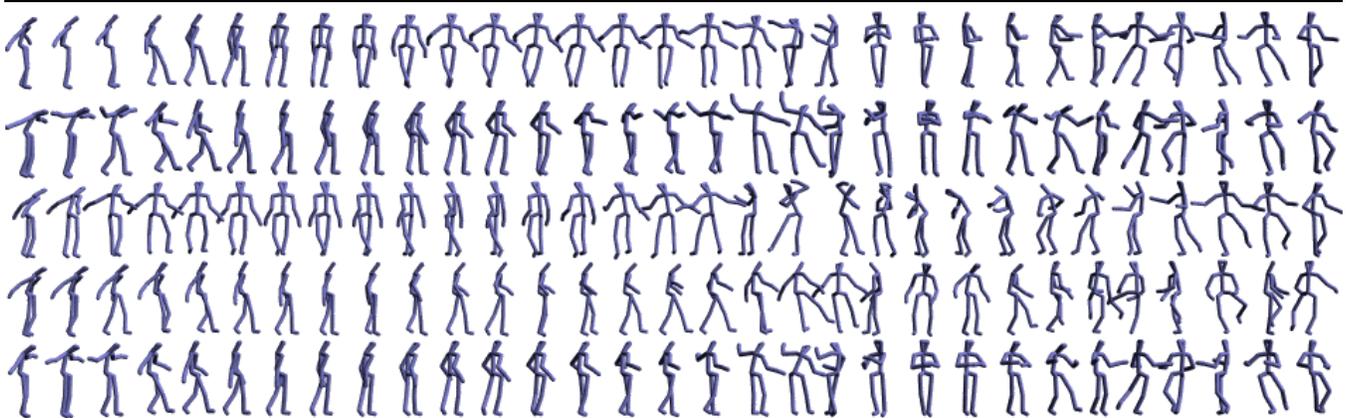


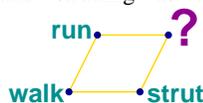
Figure 6: Five motion sequences synthesized from the same choreography, but in different styles (one per row). The actions, aligned vertically, are tiptoeing, turning, kicking, and spinning. The odd body geometry reflects marker placements in the training motion-capture.

the algorithm typically identifies variations in body geometry as the principal stylistic DOFs.)

As a hint to the algorithm, we first trained an HMM on a very low-dimensional representation of the data. Entropic estimation yielded a model which was essentially a phase-diagram of the locomotive cycle. This was used as an initialization for the full SHMM training. The SHMM was lightly annealed, so it was not constrained to use this hint, but the final generic model did retain some of the information in the initialization.

The PCA of the resulting style-specific models revealed that 3 stylistic degrees of freedom explained 93% of the variation between the 10 models. The most significant stylistic DOF appears to be global pose, e.g., one tilts forward for running, back for funny-walking. It also contains information about the speed of motion. Style DOF #2 controls balance and gender; varying it modifies the hips, the distance of the footfall to the midline, and the compensating swing of the arms. Finally, style DOF #3 can be characterized as the amount of swagger and energy in the motion; increasing it yields sequences that look more and more high-spirited. Extrapolating beyond the hull of specific models yields well-behaved motion with the expected properties. E.g., we can double the amount of swagger, or tilt a walk forward into a slouch. We demonstrate with analogies.

Analogies are a particularly interesting form of extrapolation. Given the analogical problem walking:running::strutting:X, we can solve for X in terms of the style coordinates $X = \text{strutting} + (\text{running} - \text{walking})$, which is equivalent to completing the parallelogram having the style coordinates for walking, running, and strutting as three of its corners.



The resulting synthesized sequence (figure 5) looks like a fast-advancing form of skanking (a rather high-energy pop dance style). Similarly, the analogy walking:running::cat-walking:X gives something that looks like how a model might skip/run down a catwalk in a fashion show.

Now we turn to examples that cannot be handled by existing time-warping and signal processing methods.

In a more complicated example, the system was trained on four **performances by classically trained dancers** (man, woman-ballet, woman-modern-dance, woman-lazy-ballet) of 50-70 seconds duration each, with roughly 20 different moves. The performances all have similar choreographies but vary in the timing, ordering, and style of moves. A 75-state model took roughly 20 minutes to train on 6000+ frames, using interpreted Matlab code on a single CPU of a 400MHz AlphaServer. Parameter extinction left a 69-state SHMM with roughly 3500 parameters. Figure 4 shows that the system has discovered roughly equivalent qualitative structure in all the dances. The figure also shows a flowchart of the “choreography” discovered in learning.

We then took a 1600-frame sequence of a novice dancer attempting similar choreography, but with little success, getting moves wrong and wrongly ordered, losing the beat, and occasionally stumbling. We resynthesized this in a masculine-modern style, obtaining notable improvements in the grace and recognizability of the dance. This is shown in the accompanying video.

We then **generated new choreography** by doing a random walk on the state machine. We used the resulting state sequence to synthesize new motion-capture in a variety of styles: $\frac{3}{2}$ ballet- $\frac{1}{2}$ languid; modern+male, etc. These are shown in the video. Figure 6 illustrates how different the results are by showing poses from aligned time-slices in the different synthesized performances.

Finally, we demonstrate **driving style machines from video**. The essence of our technique is the generation of stylistically var-

ied motion capture from HMM state sequences (or distributions over states). In the examples above, we obtained state sequences from existing motion capture or random walks on the HMM state machine. In fact, such state sequences can be calculated from arbitrary signals: We can use Brand’s shadow puppetry technique [5] to infer state sequences and/or 3D body pose and velocity from video image sequences. This means that one can create animations by acting out a motion in front of a camera, then use style machines to map someone else’s (e.g. an expert’s) style onto one’s choreography. In the accompanying video we show some vision-driven motion-capture and stylistic variations thereon.

6 Discussion

Our unsupervised framework automates many of the dreariest tasks in motion-capture editing and analysis: The data needn’t be segmented, annotated, or aligned, nor must it contain any explicit statement of the theme or the stylistic degrees of freedom (DOFs). All these things are discovered in learning. In addition, the algorithms automatically segment the data, identify primitive motion cycles, learn transitions between primitives, and identify the stylistic DOFs that make primitives look quite different in different motion-capture sequences.

This approach treats animation as a pure data-modeling and inference task: There is no prior kinematic or dynamic model; no representation of bones, masses, or gravity; no prior annotation or segmentation of the motion-capture data into primitives or styles. Everything needed for generating animation is learned directly from the data.

However, the user isn’t forced to stay “data-pure.” We expect that our methods can be easily coupled with other constraints; the quadratic synthesis objective function and/or its linear gradient (eqn. 21) can be used as penalty terms in larger optimizations that incorporate user-specified constraints on kinematics, dynamics, foot placement, etc. That we have *not* done so in this paper and video should make clear the potential of raw inference.

Our method generalizes reasonably well off of its small training set, but like all data-driven approaches, it will fail (gracefully) if given problems that look like nothing in the training set. We are currently exploring a variety of strategies for incrementally learning new motions as more data comes in.

An important open question is the choice of temperature schedules, in which we see a trade-off between learning time and quality of the model. The results can be sensitive to the time-courses of T and T' and we have no theoretical results about how to choose optimal schedules.

Although we have concentrated on motion-capture time-series, the style machine framework is quite general and could be applied to a variety of data types and underlying models. For example, one could model a variety of textures with mixture models, learn the stylistic DOFs, then synthesize extrapolated textures.

7 Summary

Style machines are generative probabilistic models that can synthesize data in a broad variety of styles, interpolating and extrapolating stylistic variations learned from a training set. We have introduced a cross-entropy optimization framework that makes it possible learn style machines from a sparse sampling of unlabeled style examples. We then showed how to apply style machines to full-body motion-capture data, and demonstrated three kinds of applications: resynthesizing existing motion-capture in new styles; synthesizing new choreographies and stylized motion data therefrom; and synthesizing stylized motion from video. Finally, we showed style machines doing something that every dance student has wished for: Superim-

posing the motor skills of an expert dancer on the choreography of a novice.

8 Acknowledgments

The datasets used to train these models were made available by Bill Freeman, Michael Gleicher, Zoran Popovic, Adaptive Optics, Biovision, Kinetix, and some anonymous sources. Special thanks to Bill Freeman, who choreographed and collected several dance sequences especially for the purpose of style/content analysis. Egon Pasztor assisted with converting motion capture file formats, and Jonathan Yedidia helped to define the joint angle parameterization.

References

- [1] L. Baum. An inequality and associated maximization technique in statistical estimation of probabilistic functions of Markov processes. *Inequalities*, 3:1–8, 1972.
- [2] M. Brand. Pattern discovery via entropy minimization. In D. Heckerman and C. Whittaker, editors, *Artificial Intelligence and Statistics #7*. Morgan Kaufmann., January 1999.
- [3] M. Brand. Exploring variational structure by cross-entropy optimization. In P. Langley, editor, *Proceedings, International Conference on Machine Learning*, 2000.
- [4] M. Brand. Voice puppetry. *Proceedings of SIGGRAPH 99*, pages 21–28, August 1999.
- [5] M. Brand. Shadow puppetry. *Proceedings of ICCV 99*, September 1999.
- [6] C. Bregler. Learning and recognizing human dynamics in video sequences. *Proceedings of CVPR 97*, 1997.
- [7] A. Bruderlin and L. Williams. Motion signal processing. *Proceedings of SIGGRAPH 95*, pages 97–104, August 1995.
- [8] J. Buhmann. Empirical risk approximation: An induction principle for unsupervised learning. Technical Report IAI-TR-98-3, Institut für Informatik III, Universität Bonn, 1998., 1998.
- [9] R. M. Corless, G. H. Gonnet, D. E. G. Hare, D. J. Jeffrey, and D. E. Knuth. On the Lambert W function. *Advances in Computational Mathematics*, 5:329–359, 1996.
- [10] W. T. Freeman and J. B. Tenenbaum. Learning bilinear models for two-factor problems in vision. In *Proceedings, Conf. on Computer Vision and Pattern Recognition*, pages 554–560, San Juan, PR, 1997.
- [11] M. Gleicher. Motion editing with spacetime constraints. *1997 Symposium on Interactive 3D Graphics*, pages 139–148, April 1997.
- [12] M. Gleicher. Retargeting motion to new characters. *Proceedings of SIGGRAPH 98*, pages 33–42, July 1998.
- [13] R. Grzeszczuk, D. Terzopoulos, and G. Hinton. Neuroanimator: Fast neural network emulation and control of physics-based models. *Proceedings of SIGGRAPH 98*, pages 9–20, July 1998.
- [14] N. R. Howe, M. E. Leventon, and W. T. Freeman. Bayesian reconstruction of 3d human motion from single-camera video. In S. Solla, T. Leend, and K. Muller, editors, *Advances in Neural Information Processing Systems*, volume 10. MIT Press, 2000.
- [15] J. Lee and S. Y. Shin. A hierarchical approach to interactive motion editing for human-like figures. *Proceedings of SIGGRAPH 99*, pages 39–48, August 1999.
- [16] Z. Popović and A. Witkin. Physically based motion transformation. *Proceedings of SIGGRAPH 99*, pages 11–20, August 1999.
- [17] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, Feb. 1989.
- [18] C. Rose, M. F. Cohen, and B. Bodenheimer. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics & Applications*, 18(5):32–40, September - October 1998.
- [19] J. B. Tenenbaum and W. T. Freeman. Separating style and content. In M. Mozer, M. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, pages 662–668. MIT Press, 1997.
- [20] M. Unuma, K. Anjo, and R. Takeuchi. Fourier principles for emotion-based human figure animation. *Proceedings of SIGGRAPH 95*, pages 91–96, August 1995.
- [21] A. Wilson and A. Bobick. Parametric hidden markov models for gesture recognition. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 21(9), 1999.
- [22] A. Witkin and Z. Popović. Motion warping. *Proceedings of SIGGRAPH 95*, pages 105–108, August 1995.
- [23] S. C. Zhu, Y. Wu, and D. Mumford. Minimax entropy principle and its applications to texture modeling. *Neural Computation*, 9(8), 1997.

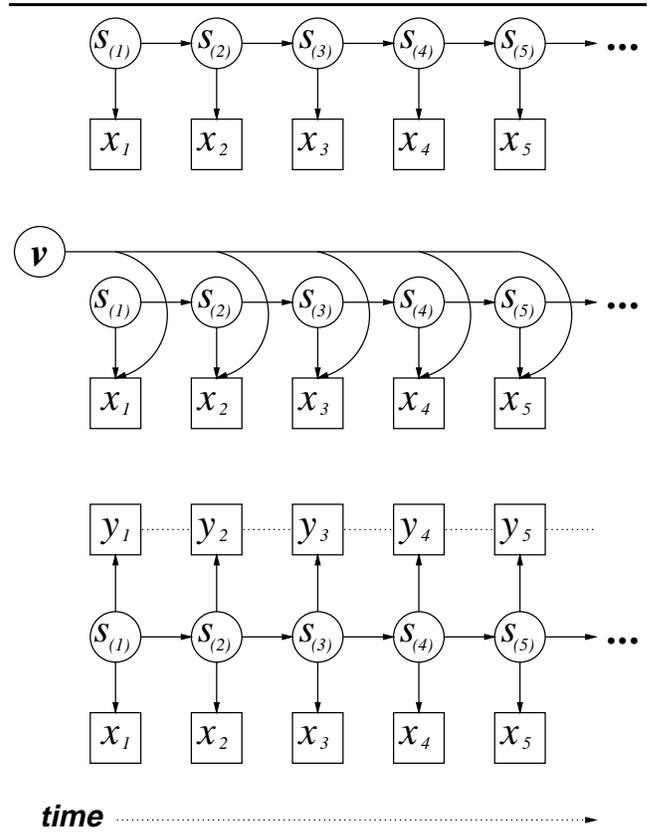


Figure 7: Graphical models of an HMM (top), SHMM (middle), and path map (bottom). The observed signal \mathbf{x}_t is explained by a discrete-valued hidden state variable $\mathbf{s}_{(t)}$ which changes over time, and in the case of SHMMs, a vector-valued style variable \mathbf{v} . Both \mathbf{v} and $\mathbf{s}_{(t)}$ are hidden and must be inferred probabilistically. Arcs indicate conditional dependencies between the variables, which take the form of parameterized compatibility functions. In this paper we give rules for learning (inferring all the parameters associated with the arcs), analysis (inferring \mathbf{v} and $\mathbf{s}_{(t)}$), and synthesis of novel but consistent behaviors (inferring a most likely \mathbf{y}_t for arbitrary settings of \mathbf{v} and $\mathbf{s}_{(t)}$).

A Hidden Markov models

An HMM is a probability distribution over time-series. Its dependency structure is diagrammed in figure 7. It is specified by $\theta = \{\mathcal{S}, P_i, P_{j \rightarrow i}, p_i(\mathbf{x})\}$ where

- $\mathcal{S} = \{s_1, \dots, s_N\}$ is the set of discrete states;
- stochastic matrix $P_{j \rightarrow i}$ gives the probability of transitioning from state j to state i ;
- stochastic vector P_i is the probability of a sequence beginning in state i ;
- emission probability $p_i(\mathbf{x})$ is the probability of observing \mathbf{x} while in state i , typically a Gaussian $p_i(\mathbf{x}) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_i, \mathbf{K}_i) = e^{-(\mathbf{x} - \boldsymbol{\mu}_i)^\top \mathbf{K}_i^{-1} (\mathbf{x} - \boldsymbol{\mu}_i) / 2} / \sqrt{(2\pi)^d |\mathbf{K}_i|}$ with mean $\boldsymbol{\mu}_i$ and covariance \mathbf{K}_i .

We cover HMM essentials here; see [17] for a more detailed tutorial. It is useful to think of a (continuous-valued) time-series \mathbf{X} as a path through configuration space. An HMM is a state-space

model, meaning that it divides this configuration space into regions, each of which is more or less “owned” by a particular hidden state, according to its emission probability distribution. The likelihood of a path $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_T\}$ with respect to particular sequence of hidden states $S = \{s_{(1)}, s_{(2)}, \dots, s_{(T)}\}$ is the probability of each point on the path with respect to the current hidden state ($\prod_{t=1}^T p_{s_{(t)}}(\mathbf{x}_t)$), times the probability of the state sequence itself, which is the product of all its state transitions ($P_{s_{(1)}} \prod_{t=2}^T P_{s_{(t-1)} \rightarrow s_{(t)}}$). When this is summed over all possible hidden state sequences, one obtains the likelihood of the path with respect to the entire HMM:

$$p(\mathbf{X}|\boldsymbol{\theta}) = \sum_{S \in \mathcal{S}^T} \left[P_{s_{(1)}} p_{s_{(1)}}(\mathbf{x}_1) \prod_{t=2}^T P_{s_{(t-1)} \rightarrow s_{(t)}} p_{s_{(t)}}(\mathbf{x}_t) \right] \quad (2)$$

A maximum-likelihood HMM may be estimated from data \mathbf{X} via alternating steps of Expectation—computing a distribution over the hidden states—and maximization—computing locally optimal parameter values with respect to that distribution. The E-step contains a dynamic programming recursion for eqn. 2 that saves the trouble of summing over the exponential number of state sequences in \mathcal{S}^T :

$$p(\mathbf{X}|\boldsymbol{\theta}) = \sum_i \alpha_{T,i} \quad (3)$$

$$\alpha_{t,i} = p_i(\mathbf{x}_t) \sum_j \alpha_{t-1,j} P_{j \rightarrow i}; \quad \alpha_{1,i} = P_i P_i(\mathbf{x}_1) \quad (4)$$

α is called the forward variable; a similar recursion gives the backward variable β :

$$\beta_{t,i} = \sum_j \beta_{t+1,j} p_j(\mathbf{x}_{t+1}) P_{i \rightarrow j}; \quad \beta_{T,i} = 1 \quad (5)$$

In the E-step the variables α, β are used to calculate the expected sufficient statistics $\boldsymbol{\omega} = \{C, \gamma\}$ that form the basis of new parameter estimates. These statistics tally the *expected* number of times the HMM transitioned from one state to another

$$C_{j \rightarrow i} = \sum_{t=2}^T \alpha_{t-1,j} P_{j \rightarrow i} p_i(\mathbf{x}_t) \beta_{t,i} / P(\mathbf{X}|\boldsymbol{\theta}), \quad (6)$$

and the probability that the HMM was in hidden state s_i when observing datapoint \mathbf{x}_t

$$\gamma_{t,i} = \alpha_{t,i} \beta_{t,i} / \sum_i \alpha_{t,i} \beta_{t,i}. \quad (7)$$

These statistics are optimal with respect to all the information in the entire sequence and in the model, due to the forward and backward recursions. In the M-step, one calculates maximum likelihood parameter estimates which are simply normalizations of $\boldsymbol{\omega}$:

$$\hat{P}_{i \rightarrow j} = C_{i \rightarrow j} / \sum_i C_{i \rightarrow j} \quad (8)$$

$$\hat{\boldsymbol{\mu}}_i = \sum_t \gamma_{t,i} \mathbf{x}_t / \sum_t \gamma_{t,i} \quad (9)$$

$$\hat{\mathbf{K}}_i = \sum_t \gamma_{t,i} (\mathbf{x}_t - \hat{\boldsymbol{\mu}}_i)(\mathbf{x}_t - \hat{\boldsymbol{\mu}}_i)^\top / \sum_t \gamma_{t,i} \quad (10)$$

After training, eqns. 9 and 10 can be used to remap the model to any synchronized time-series.

In §D we replace these with more powerful entropy-optimizing estimates.

B Stylistic hidden Markov models

A stylistic hidden Markov model (SHMM) is an HMM whose parameters are functionally dependent on a style variable \mathbf{v} (see figure 7). For simplicity of exposition, here we will only develop the case where the emission probability functions $p_i(\mathbf{x}_i)$ are Gaussians whose means and covariances are varied by \mathbf{v} . In that case the SHMM is specified by $\boldsymbol{\theta} = \{S, P_i, P_{j \rightarrow i}, \boldsymbol{\mu}_i, \mathbf{K}_i, \mathbf{U}_i, \mathbf{W}_i, \mathbf{v}\}$ where

- mean vector $\boldsymbol{\mu}_i$, covariance matrix \mathbf{K}_i , variation matrices $\mathbf{U}_i, \mathbf{W}_i$ and style vector \mathbf{v} parameterize the multivariate Gaussian probability $p_i(\mathbf{x}_i)$ of observing a datapoint \mathbf{x}_i while in state i :

$$p(\mathbf{x}_i | s_i) = \mathcal{N}(\mathbf{x}_i; \boldsymbol{\mu}_i + \mathbf{U}_i \mathbf{v}, \mathbf{K}_i + \mathbf{W}_i \mathbf{v}).$$

where the stylized covariance matrix $\mathbf{K}_i + \mathbf{W}_i \mathbf{v}$ is kept positive definite by mapping its eigenvalues to their absolute values (if necessary).

The parameters $\{P_i, P_{j \rightarrow i}, \boldsymbol{\mu}_i, \mathbf{K}_i\}$ are obtained from data via entropic estimation; $\{\mathbf{U}_i, \mathbf{W}_i\}$ are the dominant eigenvectors obtained in the post-training PCA of the style-specific models; and \mathbf{v} can be estimated from data and/or varied by the user. If we fix the value of \mathbf{v} , then the model becomes a standard discrete-state, Gaussian-output HMM. We call the $\mathbf{v} = \mathbf{0}$ case the *generic HMM*.

A simpler version of this model has been treated before in a supervised context by [21]; in their work, only the means vary, and one must specify by hand the structure of the model’s transition function $P_{j \rightarrow i}$, the number of dimensions of stylistic variation $\dim(\mathbf{v})$, and the value of \mathbf{v} for every training sequence. Our framework learns all of this automatically without supervision, and generalizes to a wide variety of graphical models.

C Entropies and cross-entropies

The first two terms of our objective function (eqn. 1) are essentially the likelihood function, which measures the fit of the model to the data:

$$H(\boldsymbol{\omega}) + D(\boldsymbol{\omega} || \boldsymbol{\theta}) = -\mathcal{L}(\mathbf{X}|\boldsymbol{\theta}) = -\log P(\mathbf{X}|\boldsymbol{\theta}) \quad (11)$$

The remaining terms measure the fit of the model to our beliefs. Their precise forms are derived from the likelihood function. For multinomials with parameters $\boldsymbol{\theta} = \{\theta_1, \dots, \theta_d\}$,

$$H(\boldsymbol{\theta}) = -\sum_i \theta_i \log \theta_i, \quad (12)$$

$$D(\boldsymbol{\theta}^* || \boldsymbol{\theta}) = \sum_i \theta_i^* \log(\theta_i^* / \theta_i). \quad (13)$$

For d -dimensional Gaussians of mean $\boldsymbol{\mu}$, and covariance \mathbf{K} ,

$$H(\boldsymbol{\theta}) = \frac{1}{2} [d \log 2\pi e + \log |\mathbf{K}|], \quad (14)$$

$$D(\boldsymbol{\theta}^* || \boldsymbol{\theta}) = \frac{1}{2} [\log |\mathbf{K}| - \log |\mathbf{K}^*| + \sum_{ij} (\mathbf{K}^{-1})_{ij} ((\mathbf{K}^*)^{-1})_{ij} + (\boldsymbol{\mu} - \boldsymbol{\mu}^*)^\top \mathbf{K}^{-1} (\boldsymbol{\mu} - \boldsymbol{\mu}^*) - d]. \quad (15)$$

The SHMM likelihood function is composed of multinomials and Gaussians by multiplication (for any particular setting of the hidden states). When working with such composite distributions, we optimize the sum of the components’ entropies, which gives us a measure of model coding length, and typically bounds the (usually uncalculable) entropy of the composite model. As entropy declines the bounds become tight.

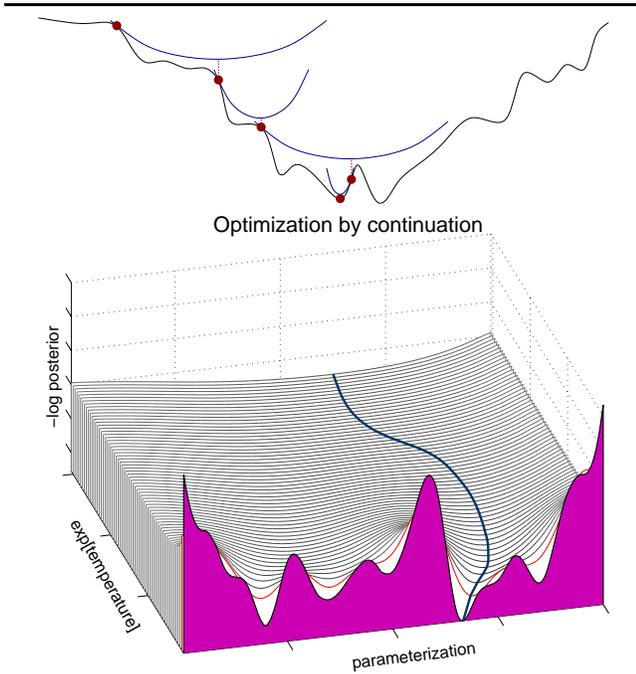


Figure 8: TOP: Expectation maximization finds a local optimum by repeatedly constructing a convex bound that touches the objective function at the current parameter estimate (E-step; blue), then calculating optimal parameter settings in the bound (M-step; red). In this figure the objective function is shown as energy = $-\log$ posterior probability; the optimum is the lowest point on the curve. BOTTOM: Annealing adds a probabilistic guarantee of finding a global optimum, by defining a smooth blend between model-fitting—the hard optimization problem symbolized by the foremost curve—and maximizing entropy—an easy problem represented by the hindmost curve—then tracking the optimum across the blend.

D Estimators

The optimal Gaussian parameter settings for minimizing cross-entropy *vis-à-vis* datapoints \mathbf{x}_i and a reference Gaussian parameterized by mean $\boldsymbol{\mu}^\bullet$ and covariance \mathbf{K}^\bullet are

$$\hat{\boldsymbol{\mu}} = \frac{\sum_i^N \mathbf{x}_i + Z' \boldsymbol{\mu}^\bullet}{N + Z'}, \quad (16)$$

$$\hat{\mathbf{K}} = \frac{\sum_i^N (\mathbf{x}_i - \hat{\boldsymbol{\mu}})(\mathbf{x}_i - \hat{\boldsymbol{\mu}})^\top + Z'((\hat{\boldsymbol{\mu}} - \boldsymbol{\mu}^\bullet)(\hat{\boldsymbol{\mu}} - \boldsymbol{\mu}^\bullet)^\top + \mathbf{K}^\bullet)}{N + Z + Z'} \quad (17)$$

The optimal multinomial parameter settings *vis-à-vis* event counts $\boldsymbol{\omega}$ and reference multinomial distribution parameterized by probabilities $\boldsymbol{\kappa}^\bullet$ are given by the fixpoint

$$\hat{P}_{j \rightarrow i} = \exp \left[W \left(\frac{-(\omega_{j \rightarrow i} + Z' \kappa_{j \rightarrow i}^\bullet)}{Z e^{\lambda/Z} - 1} \right) + \lambda/Z - 1 \right], \quad (18)$$

$$\hat{\lambda} = \frac{1}{M} \sum_i^M \left(\frac{(\omega_{j \rightarrow i} + Z' \kappa_{j \rightarrow i}^\bullet)}{P_{j \rightarrow i}} + Z \log P_{j \rightarrow i} + Z \right) \quad (19)$$

where W is defined $W(x)e^{W(x)} = x$ [9]. The factors Z, Z' vary the strength of the entropy and cross-entropy priors in annealing. Derivations will appear in a technical report available from <http://www.merl.com>.

These estimators comprise the maximization step illustrated in figure 8.

E Path maps

A path map is a statistical model that supports predictions about the time-series behavior of a target system from observations of a cue system. A path map is essentially two HMMs that share a backbone of hidden states whose transition function is derived from the target system (see figure 7). The output HMM is characterized by per-state Gaussian emission distributions over both target configurations and velocities. Given a path through cue configuration space, one calculates a distribution γ over the hidden states as in §A, eqn. 7. From this distribution one calculates an optimal path through target configuration space using the equations in §F.

F Synthesis

Here we reprise and improve on Brand's [4] solution for likeliest motion sequence given a matrix of state occupancy probabilities γ . As the entropy of γ declines, the distribution over all possible motion sequences becomes

$$\lim_{H(\gamma) \downarrow 0} p_{\theta}(\mathbf{Y} | \gamma) = e^{-\frac{1}{2} \sum_t \sum_i \gamma_{t,i} \tilde{\mathbf{y}}_{t,i}^\top \mathbf{K}_i^{-1} \tilde{\mathbf{y}}_{t,i} + c}, \quad (20)$$

where the vector $\tilde{\mathbf{y}}_{t,i} \doteq [\mathbf{y}_t - \boldsymbol{\mu}_i, (\mathbf{y}_t - \mathbf{y}_{t-1}) - \dot{\boldsymbol{\mu}}_i]^\top$ is the target position and velocity at time t minus the mean of state i , and c is a constant. The means $\boldsymbol{\mu}_i$ and covariances \mathbf{K}_i are from the synthesizing model's set of Gaussian emission probabilities $p_{s(t)}(\mathbf{y}_t, \dot{\mathbf{y}}_t) \doteq \mathcal{N}([\mathbf{y}_t, \dot{\mathbf{y}}_t - \mathbf{y}_{t-1}]; [\boldsymbol{\mu}_{s(t)}, \dot{\boldsymbol{\mu}}_{s(t)}], \mathbf{K}_{s(t)})$. Breaking each inverse covariance into four submatrices $\mathbf{K}_j^{-1} \doteq \begin{bmatrix} A_j & B_j \\ C_j & D_j \end{bmatrix}$, we can obtain the maximum likelihood trajectory \mathbf{Y}^* (most likely motion sequence) by solving the weighted system of linear equations

$$\forall_{t=2}^{T-1} \forall_{j,i} \begin{bmatrix} -\gamma_{t-1,j}(B_j + D_j) \\ \gamma_{t-1,j}(A_j + B_j + C_j + D_j) + \gamma_{t,i} D_i \\ -\gamma_{t,i}(C_i + D_i) \end{bmatrix}^\top \begin{bmatrix} \mathbf{y}_{t-1} \\ \mathbf{y}_t \\ \mathbf{y}_{t+1} \end{bmatrix} = \gamma_{t-1,j} F_j - \gamma_{t,i} E_i, \quad (21)$$

where $E_j \doteq [C_j \ D_j] [\boldsymbol{\mu}_j \ \dot{\boldsymbol{\mu}}_j]^\top$, $F_j \doteq [A_j \ B_j] [\boldsymbol{\mu}_j \ \dot{\boldsymbol{\mu}}_j]^\top + E_j$ and the endpoints are obtained by dropping the appropriate terms from equation 21:

$$\forall_j \gamma_{j,1} \begin{bmatrix} D_j \\ -C_j - D_j \end{bmatrix}^\top \begin{bmatrix} \mathbf{y}_0 \\ \mathbf{y}_1 \end{bmatrix} = -\gamma_{j,1} E_j \quad (22)$$

$$\forall_j \gamma_{j,T} \begin{bmatrix} -B_j - D_j \\ A_j + B_j + C_j + D_j \end{bmatrix}^\top \begin{bmatrix} \mathbf{y}_{T-1} \\ \mathbf{y}_T \end{bmatrix} = \gamma_{j,T} F_j \quad (23)$$

This generalizes Brand's geodesic [4] to use all the information in the occupancy matrix γ , rather than just a state sequence.

The least-squares solution of this $\mathbf{L}\mathbf{Y} = \mathbf{R}$ system can be calculated in $O(T)$ time because \mathbf{L} is block-tridiagonal.

We introduce a further improvement in synthesis: If we set \mathbf{Y} to the training data in eqn. 21, then we can solve for the set of Gaussian means $\mathbf{M} = \{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \dots\}$ that minimizes the weighted squared-error in reconstructing that data from its state sequence. To do so, we factor the r.h.s. of eqn. 21 into $\mathbf{L}\mathbf{Y} = \mathbf{R} = \mathbf{G}\mathbf{M}$ where \mathbf{G} is an indicator matrix built from the sequence of most probable states. The solution for \mathbf{M} via the calculation $\mathbf{M} = \mathbf{L}\mathbf{Y}\mathbf{G}^{-1}$ tends to be of enormous dimension and ill-conditioned, so we precondition to make the problem well-behaved: $\mathbf{M} = (\mathbf{Q}^\top \mathbf{Y})(\mathbf{Q}^\top \mathbf{Q})^{-1}$, where $\mathbf{Q} = \mathbf{L}\mathbf{G}^{-1}$. One caution: There is a tension between perfectly fitting the training data and generalizing well to new problems; unless we have a very large amount of data, the minimum-entropy setting of the means will do better than the minimum squared error setting.