Generalizing Motion Edits with Gaussian Processes

LESLIE IKEMOTO  
University of California, Berkeley  
OKAN ARIKAN  
University of Texas, Austin  
and  
DAVID FORSYTH  
University of Illinois, Urbana-Champaign

One way that artists create compelling character animations is by manipulating details of a character’s motion. This process is expensive and repetitive. We show that we can make such motion editing more efficient by generalizing the edits an animator makes on short sequences of motion to other sequences. Our method predicts frames for the motion using Gaussian process models of kinematics and dynamics. These estimates are combined with probabilistic inference. Our method can be used to propagate edits from examples to an entire sequence for an existing character, and it can also be used to map a motion from a control character to a very different target character. The technique shows good generalization. For example, we show that an estimator, learned from a few seconds of edited example animation using our methods, generalizes well enough to edit minutes of character animation in a high-quality fashion. Learning is interactive: An animator who wants to improve the output can provide small, correcting examples and the system will produce improved estimates of motion. We make this interactive learning process efficient and natural with a fast, full-body IK system with novel features. Finally, we present data from interviews with professional character animators that indicate that generalizing and propagating animator edits can save artists significant time and work.

Categories and Subject Descriptors: I.3.7 [Computer Graphics] Three-Dimensional Graphics and Realism

General Terms: Algorithms

Additional Key Words and Phrases: Artist-guided content creation, controllable motion editing

ACM Reference Format:

1. INTRODUCTION

Expressive character animation is compelling because careful manipulation of subtle details of a motion can reveal a great deal about a character: from physical parameters like body weight, to movement style, to personality. Character animation can be difficult and expensive to produce because these details must be created and manipulated with great skill and care.

This article describes a system that uses generalization properties of a learning system to propagate changes created in short training sequences to a full animation. Such propagation is possible because motion generally has repetitive and predictable features.

To use the tool, the artist first selects a pre-existing animation clip and edits it. Our system creates a model of the edits. Then, given another animation clip, the system automatically generalizes and propagates the artist’s edits (Figure 1). Important properties of our system include the following:

—Incremental Model Refinement. The artist can change parts of the automatically generated sequence, and the system uses the changes to refine its internal model and produce a new sequence. The artist can then make additional changes which the system again uses to improve its model, thus improving the edited sequence. The ability to incrementally refine the model until it produces results the artist likes makes the system controllable. This means the artist’s interaction with the model is opportunistic: When our system generalizes more successfully than might be expected (e.g., see Figure 6), the artist can do less work.

—Generalization to New Characters. By editing the motion to fit a new character, the artist can use our method for retargeting. We...
show that our system works even for new characters that differ significantly in physical properties and movement styles from the original character. (In the rest of this article, we call the original character the source, and the new character the target.)

—Few Training Examples Needed. Examples are precious because an artist must create them by hand. Our system needs few of them (see Section 7). As we demonstrate in the supplemental video accompanying this article, our method needs significantly fewer examples than competing methods (see Section 2), because it combines both kinematic and dynamics estimates derived from example sequences.

—Smoothed Configuration and Acceleration Prediction. There are a variety of signals that could be transferred from a source motion to a target motion. If we were to transfer only a dynamical model, configurations might drift. If we were to transfer only configuration information, the motion might be jerky. If we were to transfer only acceleration information, configurations might again drift. Our method predicts both the configuration and the acceleration of the target and then constructs a sequence whose configurations and accelerations go close to predicted values. This is attractive because it imposes a consistency constraint on the reconstructed path. Furthermore, it means that if an artist edits a single frame in a sequence, the edit can propagate some way in time.

—Stabilized Predictions. Furthermore, our method has mechanisms that impose constraints less strongly if the predicted values are less certain. We stabilize our predictions by biasing the joint angles adopted by our target frames toward the joint angles of the source frames, with a stronger bias when predictions are less certain. This is attractive because it mitigates effects caused by poor predictions.

Our method works by merging the estimates from a function that generates poses for the new character and a function that estimates the dynamics. The estimation technique we use is known as Gaussian process regression, which is a supervised learning technique that develops a probabilistic model using input-output examples.

Our method benefits greatly from an intuitive motion editing system. We present a direct-manipulation interface for editing character animation that uses a fast, full-body inverse kinematics (IK) system for interactively posing the character. Our IK algorithm is at least as fast as current solutions, and the formulation fits well with our problem.

Finally, we interviewed professional character animators about their work practices. These interviews indicate that generalizing and propagating animator edits can save an artist significant time and work.

2. RELATED WORK

Our method is most closely related to the work of Hsu et al. [2005], who encode the transformation between two motions having matching content but differing style as a linear time-invariant system (for a reference, see Ljung [1999]). Once this system has been identified, the transformation can be applied to a new motion faster than real time, producing a corresponding motion in the learned style.

Our work shares goals with such research but there is an important difference. We use a nonparametric model of the transformation between motion sequences (Gaussian process regression), which means we can model transformations that are difficult for a linear time-invariant system. A particular attraction is better modelling of context-dependent transformations. For example, it may be difficult for Hsu et al.’s system to apply one type of transformation while the character is standing, and another while the character is walking. A second attraction is that our method requires relatively little training data, as we demonstrate in Section 7 and in our supplementary video. Hsu et al. report that their style transfer method requires 1–2 minutes of training data in a particular style, while our method requires only seconds.

Training a Gaussian process regression scales poorly with large quantities of training data. However, in our application we do not have large quantities of training data. For the generalized motion editing problem we consider in this work, the difference between seconds and minutes of training data is significant, because the artist must painstakingly pose the character to create examples. With the kernel we use, the method can produce only smooth sequences. We have not found this to be problematic, and it may generally be a desirable feature.

It is also possible to encode a complex mapping between a source motion and target motion by storing examples of the relationship between them. Hsu et al. [2004] showed they can use this technique to generate compelling animations of partner dance. Their method focuses primarily on motion control, and is not as well suited for the motion editing problem we consider here because their technique does not have a strategy for handling input motions that were not in the training set.

Our technique also draws from work in retargeting and computer puppetry, in which the motion of one character is adapted for another. The characters may, for example, differ in proportion [Gleicher 1998; Shin et al. 2001], in which case kinematic constraints should be identified and enforced. The characters may differ so much that an artist specifies corresponding poses [Bregler et al.
Other techniques adapt the motion of a person to drive an animated character [Chai and Hodgins 2005].

Retargeting and computer puppetry fall under the broader class of motion editing techniques, in which a motion is adapted to satisfy constraints. The constraints may be kinematic [Bruderlin and Williams 1995; Witkin and Popovic 1995], physical [Liu and Popovic 2002; Tak and Ko 2005; Shin et al. 2003; Popovic and Witkin 1999], stylistic [Pullen and Bregler 2002], emotional [Amaya et al. 1996], or a combination thereof. Chai and Hodgins [2007] learn a model of human dynamics from motion capture data, then use the model as a prior for constrained motion synthesis. Safonova and Hodgins [2007] solve the constrained motion synthesis problem by constructing and searching a special type of motion graph that supports interpolation of motion clips. Grochow et al. [2004] describe an inverse kinematics (IK) system that uses a version of a Gaussian process latent variable model [Lawrence 2004] to bias output poses towards natural poses.

Another important class of constraints comes from the environment. For example, a character’s feet are usually constrained to be stationary when planted [Kovar et al. 2002; Ikemoto et al. 2005].

All of these techniques, including ours, try to adapt the source motion while preserving content. Another broad class of techniques generates novel content. For example, rearranging clips from a collection of motion can produce novel, natural-looking sequences. Motion graphs encode valid rearrangements [Kovar et al. 2002; Arikan and Forsyth 2002; Lee et al. 2002]. While this technique produces novel content, other aspects, such as the style of the motion, are fixed to be the same as the collection.

Still another class of methods models both the content and style of motions. Perhaps the simplest example is motion blending [Rose et al. 1998; Kovar and Gleicher 2004]. Motion blending yields a parameterized blend space that varies over the differences in the blend targets. Careful choice of blend targets produces a blend space over, for example, content, style, or both. Other work models the space of content and style as a parameterized set of hidden Markov models [Brand and Hertzmann 2000], a set of linear dynamical systems [Li et al. 2002], or as the minimums of a parameterized energy function [Liu et al. 2005]. Modeling content and style is an attractive goal, but we must have sufficient data to estimate a dimensionality reduction [Brand and Hertzmann 2000; Li et al. 2002], or must perform an offline optimization over many parameters [Liu et al. 2005].

Other character animation methods use techniques based on Gaussian processes. Gaussian process regression is an elegant, non-parametric technique which yields a Gaussian probability distribution over the space of output vectors, given an input vector. Gaussian processes have a strong history of giving good results in predicting animations and kinematic configurations. The scaled Gaussian process latent variable model used in Grochow et al. [2004] learns the mapping between a low-dimensional latent space and a high-dimensional character pose space. In Wang et al. [2005], the Gaussian process dynamic model learns the mapping between a low-dimensional latent space and a high-dimensional character dynamics space. Mukai and Kuriyama used kriging, a technique closely related to Gaussian processes, for interpolating motion [Mukai and Kuriyama 2005]. Wang et al. separate style and content using multi-factor Gaussian process models [Wang et al. 2007]. Urtasun et al. use Gaussian processes to model human dynamics for tracking purposes [Urtasun et al. 2006], Lawrence and Moore visualize human animation with hierarchies of Gaussian process latent variable models [Lawrence and Moore 2007]. It would be interesting to consider how the hierarchies they describe could be applied to the problem we consider here. Gaussian processes can be seen as a complex scheme for identifying and blending relevant frames, and blending animations is remarkably effective (e.g., see Bruderlin and Williams [1995], Rose et al. [1998], Wiley and Hahn [1997]).

Generalization, that is, automatically producing natural-looking motion different from training motions is a crucial but notoriously difficult problem. As far as we know, all systems perform unreliably in this regard. One mitigating strategy in motion transfer is to use a heuristic to detect whether an input frame is outside the space of training data, then blend the output frame with the input [Hsu et al. 2005]. We use an analogous technique.

Our system incrementally builds a model of artist edits that is refined through additional examples. In a similar vein, Arikan et al. [2003] described a system for incrementally learning motion annotations.

It is possible to make animation more natural by creating an intuitive interface. Two recently described alternatives to using traditional keyframing programs are to act out the animation [Dontcheva et al. 2003] or to use spatial keyframing [Igarashi et al. 2005]. Moving the character by direct manipulation is also an intuitive way to edit motion [Gleicher 1997; Lee and Shin 1999]. Many commercial packages, like Autodesk’s Maya, support direct manipulation. One way to implement direct manipulation is with IK. Full-body IK is well studied (see e.g., Girard and Maciejewski [1985], Bodenheimer et al. [1997]). We present a novel formulation that is at least as fast as current solutions, and is especially suited to our problem.

3. METHOD

We describe a general, directly controllable method for learning the edits an artist makes to an animation. Our method can be used to edit an animation, or it can be applied after traditional retargeting techniques to yield a controllable retargeting system.

The artist first selects a pre-existing animation clip, then edits it to fit the target character. The artist may edit the target’s configuration in any frame, or manipulate timing between frames. The source and target clips are assumed to have the same number of frames, but the frames may appear with different timing. Our goal is to build a function that estimates a target animation from a set of features describing a source animation and the set of training examples the artist provided. This is a regression problem, and Section 4 describes our regression procedure. The features we use are described in Section 4.5. In Section 5, we describe our animation editing system, which features a direct-manipulation interface.

When the artist supplies another source animation, the system estimates the corresponding target animation using the technique described in Section 4. The artist can then inspect the new target animation. If the artist does not like any part of the animation, (s)he can fix the estimate. The system incorporates these fixes into the training data and produces a new estimate. Hence, the method is directly controllable because an artist can refine the system by providing feedback. After updating its internal model, the system produces a new target animation, which the artist again examines. Until the artist is satisfied, this estimate-inspect loop continues. Figure 2 shows a high-level schematic of the system.

The regression procedure we use (Section 4) assumes that similar input instances map to similar outputs. Conflicting training data may produce undesirable results (Figure 7). Our regression procedure can, however, estimate a noninjective transformation (i.e., two different inputs can map to the same output). See Section 4.2 for details.

4. REGRESSION

There are numerous cues to determine a target sequence from source information. There are large and characteristic accelerations that

ACM Transactions on Graphics, Vol. 28, No. 1, Article 1, Publication date: January 2009.
have important visual effects at some phases of movement. Important examples include: foot collisions with the ground and other contacts in general; accelerations that occur at the end of an arm swing during walking (Figure 3); the initial phase of fast ballistic movements like reaching or quick turns of the head. These phenomena could be captured by velocity predictions only if they occurred at a high sampling rate, and so we should predict acceleration explicitly. We could then integrate the accelerations to come up with a motion, but this procedure is likely to drift. This drift can be controlled by predicting kinematic configurations as well.

There is now a tension between freedom to author and efficiency. It might be possible predict blocks of target frames, allowing such parametric transformations as adding a constant offset to a particular joint angle for an entire sequence. Alternatively, we might predict individual target frames, allowing a greater range of deformations at the possible cost of more authoring effort. We incline toward the second approach, and our method regresses target configuration and acceleration at each sample from the source information. This means that, for example, to add a constant to a joint angle for an entire sequence we might need to supply many examples. However, some such temporally organized changes can be dealt with at the level of the authoring interface (i.e., the animator would select a block of source frames, add a constant to the angle, and label these as target frames). The advantage of pairing a regressing target from source frames independently like this is the very high level of control available. However, it is difficult to make crisp statements about what can and cannot be represented. For example, handling edits where the left arm of the source motion affects the right arm of the target character is rigged the same way as the source). Once we have configuration and acceleration predictions, we must obtain a sequence that is as consistent as possible with them all.

**Notation.** In general, we write $D$ to represent a set of training data (pairs of source and target frames), and $\theta$ to represent parameters of a learned predictor, subscribing to indicate which predictor. We write $x_i$ for features computed from the source sequence centered at the $i$th frame (Section 4.5), distinguishing on occasion between features used to predict kinematics ($x_i^{(k)})$ and those used to predict acceleration ($x_i^{(a)})$. Kinematic features encode the joint positions for a window of frames around the current frame and acceleration features encode a smoothed estimate of acceleration in joint position for the current frame. From these source features, we predict the target’s kinematics and acceleration encoded as joint angles (details in Section 4.5). Each of our predictors will give a probability distribution function over what it predicts. Each prediction is a random variable, and we will maintain the distinction between these random variables.

### 4.1 Learned Predictors

Our learned kinematics predictor gives a probability distribution function over the target kinematics at the $i$th frame, which we write as $Y_i$, given the source features at that frame ($x_i^{(k)}), the training set ($D), and parameters ($\theta_k). We write this probability distribution as

$$P_k(Y_i | x_i^{(k)}, D, \theta_k) = \frac{1}{Z_k(D)} \exp \left( -\frac{1}{2} (Y_i - \mu_k^{(i)})^T \Sigma_k^{(i)} (Y_i - \mu_k^{(i)}) \right).$$

The $l$ stands for learned. Our learned acceleration predictor gives a probability distribution function over the target accelerations at the $i$th frame, which we write as $\dot{Y}_i$, given the source features at that frame ($x_i^{(a)}), the training set ($D), and parameters ($\theta_a). We write this probability distribution as

$$P_a(\dot{Y}_i | x_i^{(a)}, D, \theta_a) = \frac{1}{Z_a(D)} \exp \left( -\frac{1}{2} (\dot{Y}_i - \dot{\mu}_a^{(i)})^T \Sigma_a^{(i)} (\dot{Y}_i - \dot{\mu}_a^{(i)}) \right).$$

We build the kinematic and dynamic predictors using Gaussian process regression. The technique is nonparametric because we need to have all of the training data (rather than just parameter estimates) to make predictions. For a good reference, see Rasmussen and Williams [2006a]. There are several publicly available implementations of Gaussian processes. We use the GPML package [Rasmussen and Williams 2006b]. Because we use Gaussian processes, the probability distribution functions defined in (1) and (2) are multidimensional Gaussian distributions

$$P_k(Y_i | x_i^{(k)}, D, \theta_k) = c_k \exp \left( (Y_i - \mu_k^{(i)})^T (\Sigma_k^{(i)})^{-1} (Y_i - \mu_k^{(i)}) \right),$$

$$P_a(\dot{Y}_i | x_i^{(a)}, D, \theta_a) = c_a \exp \left( (\dot{Y}_i - \dot{\mu}_a^{(i)})^T (\Sigma_a^{(i)})^{-1} (\dot{Y}_i - \dot{\mu}_a^{(i)}) \right),$$

where $\mu_k^{(i)}$ and $\Sigma_k^{(i)}$ are the mean and variance, respectively, of the predictive density for $Y_i$, and $\mu_a^{(i)}$ and $\Sigma_a^{(i)}$ are the mean and variance, respectively, of the predictive density for $\dot{Y}_i$; these are functions of the features, and come from the Gaussian process model. If the covariance has small determinant, then the Gaussian probability
distribution has a narrow peak, indicating high confidence in the prediction; similarly, a large determinant indicates low confidence. For some frames, $P_{i,k}$ and/or $P_{j,k}$ return an estimate with high variance. This means that the target sequence at that frame is weakly constrained by these models, and may result from ambiguous training data or from a source frame that lies far from the training data.

4.2 Stabilized Predictions

Low-confidence predictions are a nuisance because they may result in a poor target sequence. Hsu et al. describe a heuristic for handling this problem [Hsu et al. 2005]: When their system detects an input frame that lies outside the space of source configurations in their training set, they blend the estimated target frame with the source frame that lies closest to it.

We adopt a similar approach. For low-confidence predictions, we assume the predicted target’s joint angles should be reasonably close to the source’s joint angles. This assumes that source and target character are rigged in the same way; it might be necessary to use a method like Filmbox’s retargeting system if they are not.

Recall the target kinematic features at the $i$th frame is $Y_i$ and the target acceleration at the $i$th frame is $\ddot{Y}_i$. If we write $\dot{Y}_i$ for the vector of source joint angles and $\dot{\ddot{Y}}_i$ for the vector of source joint accelerations at frame $i$, the probability distribution functions are

$$P(Y_i | \dot{y}, \Sigma_{i}^{(k)}) = c_k \exp((Y_i - \dot{y})^T (\Sigma_{i}^{(k)})^{-1}(Y_i - \dot{y})),$$

$$P(\ddot{Y}_i | \ddot{y}, \Sigma_{i}^{(ra)}) = c_a \exp((\ddot{Y}_i - \ddot{y})^T (\Sigma_{i}^{(ra)})^{-1}(\ddot{Y}_i - \ddot{y})),$$

where $c_k$ and $c_a$ are normalizing constants and $\Sigma_{i}^{(k)}$ and $\Sigma_{i}^{(ra)}$ are the covariance matrices, which are chosen by the method of Section 4.3. The $r$ stands for regularizing.

We now wish to find the best sequence, given the available evidence. The predictions of kinematics and acceleration are independent. We stack the frame predictions into a vector $Y$, write the relevant features as $x$, and obtain

$$P(Y | x, \dot{y}, \theta_k) \propto P(Y | x, \dot{y}, \theta_k) P(Y | \dot{y}, \Sigma^{(k)})$$

$$\times P(\ddot{Y} | x, \ddot{y}, \theta_a) P(\ddot{Y} | \ddot{y}, \Sigma^{(ra)}),$$

where

$$\dot{Y} = \arg \max \sum \log (P(Y | x, \dot{y}, \theta_k) P(Y | \dot{y}, \Sigma^{(k)}) P(\ddot{Y} | x, \ddot{y}, \theta_a)) \times P(\ddot{Y} | \ddot{y}, \Sigma^{(ra)}),$$

subject to $L(Y) = \ddot{Y}$.

Now the frames are independent, so that

$$\log (P(Y | x, \dot{y}, \theta_k) P(Y | \dot{y}, \Sigma^{(k)}) P(\ddot{Y} | x, \ddot{y}, \theta_a))$$

$$= \sum \log (P(Y_i | x, \dot{y}, \theta_k) + P(Y_i | \dot{y}, \Sigma^{(k)})) + \log (P(\ddot{Y}_i | x, \ddot{y}, \theta_a) + P(\ddot{Y}_i | \ddot{y}, \Sigma^{(ra)})).$$

This gives a system of linear equations, obtained by stacking each frame vector into an overall vector (so that the $\dot{y}$ stack to become $\dot{y}$, and so on) and stacking covariance matrices into a matrix (so that the $\Sigma^{(k)}$ stack to become $\Sigma^{(ka)}$, and so on). Dropping the normalizing constants and substituting $LY$ for $\ddot{Y}$, we obtain

$$\ddot{Y} = \arg \max \sum (G^{(k)} + G^{(ka)} + G^{(ra)}).$$

where

$$G^{(k)} = (Y - \mu^{(k)})^T (\Sigma^{(k)})^{-1}(Y - \mu^{(k)})$$

$$G^{(ka)} = (Y - \dot{y})^T (\Sigma^{(ka)})^{-1}(Y - \dot{y})$$

$$G^{(ra)} = (LY - \mu^{(ra)})^T (\Sigma^{(ra)})^{-1}(LY - \mu^{(ra)})$$

The optimization function is quadratic in $Y$, the optimal motion sequence. We can solve for $Y$ by computing the partial derivative of the optimization function with respect to $Y$, then solving the sparse linear system that results using least squares.

Since we treat the features in $Y$ as independent of each other, the $\Sigma$ matrices are diagonal. To restrict the edits to particular joints (leaving other joints unmodified), we could modify the regularizing covariance matrices. Setting the $(i, i)$ entries of $\Sigma^{(k)}$ and $\Sigma^{(ra)}$ to values approaching zero would make the $i$th feature stay close to its source value. Joint limits are not explicitly encoded in this scheme.

Note that in areas of low confidence, different source features will always map to different target features because of our regularizing assumption that the target should stay close to the source. Therefore, in areas of low confidence, the mapping function we estimate is bijective, but in areas of high confidence, our method can estimate a noninjective mapping.

4.3 Weighting and Parameter Values

Target sequences are chosen to look like predictions from the learned Gaussian process, and also to transfer well from the source. These criteria must be weighted with respect to one another, and the choice of covariance matrices $\Sigma^{(k)}$ and $\Sigma^{(ra)}$ set their relative significance.

We would like to downweight the source when the Gaussian process
Our method can produce unpredictable results when input frames lie well outside the training data. The predictions will have high variance, so similar input frames can produce very different output frames. The top row of poses are frames of Jack skipping taken from roughly the same part of five different skipping cycles (i.e., they are not sequential frames). The middle row shows the frames predicted by the kinematics predictor. Each prediction corresponds directly to the input frame above it. The predictions are noticeably dissimilar because the input skipping frames lie well away from the walking frames used for training. However, because the kinematics predictor correctly tags these predictions with high variance, they are given low weight when the system computes the final output poses. The source frames are given high weight. Therefore, the final output poses (bottom row) look similar to the input frames and to each other.

The animation sequence that corresponds to this figure is in the video.

has low variance, and upweight it when the Gaussian process has high variance. Thus, we set the weights on the regularizing estimates to be inversely proportional to the weights on the Gaussian process estimates. Therefore, we set \( (\Sigma^{(rk)})^{-1} \) to be \( \Sigma^{(lk)} \) and \( (\Sigma^{(ra)})^{-1} \) to be \( \Sigma^{(la)} \).

An artist may wish the transferred sequence to respect configuration predictions more strongly than it does acceleration predictions. Similarly, the artist may trust Gaussian process predictions more strongly than simple transfer of joint angles (or vice versa). We should like to allow artists to weight configuration and acceleration predictions, as well as stabilizing terms with respect to one another. In our system, the artist sets four scalar weights that control how much the kinematics, dynamics, kinematics regularizer, and dynamics regularizer affect the solution, yielding

\[
\hat{Y} = \arg \max_Y \left( w_{lk} G^{(lk)} + w_{rk} G^{(rk)} + w_{la} G^{(la)} + w_{ra} G^{(ra)} \right).
\]  

(12)

The weights have fairly intuitive meaning. If \( w_{rk} \) and \( w_{ra} \) are set to zero, then the prediction depends only on the Gaussian processes regression. This choice of weights can be dangerous because the
estimates with high variance may be unreliable. Setting \(w_{\text{init}}\) and \(w_{\text{final}}\) to zero roughly yields the same result as traditional retargeting methods. Solving the optimization problem now involves solving a straightforward (sparse) linear system. In our experiments, we have found that tuning the weights typically requires 2–3 iterations. Since solving the sparse linear system is fast, these iterations can be done quickly. We expect that the weights could alternatively be set automatically.

Notice that frames are not independent in this formulation because of the smoothing effects related to \(L\) and \(D\). Another way to see this is to note that the matrix of the quadratic form we are minimizing has a block structure; it is not block diagonal. We are building a sequence whose frames lie close to the predicted frames and that also transfer from the source reasonably well, and whose accelerations lie close to predicted accelerations and also transfer from the source reasonably well.

4.4 Alternative Estimators

While we found Gaussian process estimators to work well, other estimators may exist. In particular, we tried using an alternative scattered data interpolation technique called moving least-squares. Moving least-squares has been shown to produce good results for other graphics applications such as surface reconstruction (see, e.g., Shen et al. [2004]). The moving least-squares formulation is

\[ W_p A x = W_p B, \]

where \(W_p\) is a diagonal weight matrix that depends on the source point we want to evaluate \(p\), \(A\) is the source training data, and \(B\) is the target training data. Moreover, \(x\) maps source vectors onto target vectors. We computed \(W\) using a Gaussian function where the mean of the Gaussian was \(p\). We tried various variance values by hand. We used the mean of the three highest weights as the confidence value. (Confidence can be considered a measure of inverse variance.)

This regression method gave us good kinematic predictions, and was also faster than using Gaussian processes. However, we found the variance values from Gaussian processes more reliable.

4.5 Features

Our predictors produce an estimate of target configuration, given a set of features \(x\). An important part of any regression process is choosing features that are likely to be good at predicting the required output.

Representing the source kinematics. Generally, we expect the target configuration to depend on the source configuration, with some information about motion to help with ambiguous cases (e.g., in typical human motion, there is a walking configuration which can look like a standing configuration). This suggests \(x^{(i)}\) should encode the configuration and acceleration at each frame of the source animation. Our feature consists of the skeletal configuration currently, 0.125 seconds ago, and 0.125 seconds in the future (we tried the alternative of using a window of frames centered around the relevant frame, but found not much difference in the results.) Including past and future frames like this exposes velocity and acceleration information, which are linear functions of these frames. We tried the alternative of using the skeletal configuration at the relevant frame only, but found that disambiguating frames with similar configuration and different acceleration yielded more reliable confidence estimates.

We represent the three skeletal configurations as joint positions in 3D space, but the regression can produce configurations in which the bones stretch/squash to noticeably incorrect lengths, and constraining the regressed estimate would create considerable unnecessary complexity. We encode the orientation of each bone relative to the orientation of its parent in the skeletal hierarchy. We have also tried encoding the orientation relative to the skeleton’s global orientation, which yielded mostly identical results.

Representing the target acceleration. The acceleration predictor outputs an estimate of the second derivative of the target’s skeletal configuration represented as joint angles parametrized by exponential maps [Grassia 1998]. It is possible to represent the configuration as joint positions in 3D space, but the regression can produce configurations in which the bones stretch/squash to noticeably incorrect lengths, and constraining the regressed estimate would create considerable unnecessary complexity. We encode the orientation of each bone relative to the orientation of its parent in the skeletal hierarchy. We have also tried encoding the orientation relative to the skeleton’s global orientation, which yielded mostly identical results.

Representing the target kinematics. The acceleration predictor outputs an estimate of the second derivative of the target’s skeletal configuration represented as joint angles parametrized by exponential maps, the second derivative of the root ground-plane position and vertical orientation relative to the root at the last frame, and the second derivative of the time elapsed since the last frame.

5. INTERACTIVE MOTION EDITING

For our method to be useful, it is beneficial to have a system that helps artists to quickly and naturally animate the target. The results of this animation process comprise the training data for our regression scheme.

Our system allows the user to modify a particular pose \(x(t)\) to obtain an edited pose \(y(t)\) using inverse kinematics, where \(t\) is the time of the pose that the user is modifying. For completeness, we briefly summarize our IK posing system used by the artist for creating his/her pose edits, which has some unique features.

Our system allows the user to change the joint locations as if they were independent. It also allows the user to move joints individually or as a group. By painting weights onto joints, the user controls how much a joint will be affected by mouse movement. The user then drags the mouse to edit the character’s configuration.

Denote \(p^g_i\) as the 3D position \(i\)th joint after the user manipulation. We first find the set of joint positions that are as close to the edited joint positions as possible while preserving bone lengths. This can be formulated as a constrained optimization

\[
\min_p \sum_i |p_i - p^g_i|^2 \quad \text{such that } |p_i - p_j| = d_{ij},
\]

where for every bone, \(d_{ij}\) is the length of the bone between joints \(i\) and \(j\). The objective function and the constraints are quadratic. We
solve for the joint positions \( p_i \) using Lagrange multipliers. Since the Hessian is sparse (and linear with respect to the degrees of freedom), we can solve this optimization efficiently with few iterations of Newton Raphson. Notice that we can easily constrain end effectors to be at a particular location by introducing additional quadratic constraints.

Now that we have computed joint positions in which the bone lengths are preserved, we can compute the joint angles. Unfortunately, joint angles cannot be uniquely determined from joint positions (because of the twist degree of freedom around the axis of the bone). Since we already have the joint angles before the user manipulates the joint positions, we can compute the minimum rotation required for each bone to align with the new joint positions (Figure 5).

This way of performing inverse kinematics is efficient: We perform this computation and compute the joint angles every time the user drags the mouse. From the edited pose \( y(t) \) and the original pose \( x(t) \), we obtain the a kinematic delta keyframe to construct the offset function \( d(t) \), where \( y(t) = d(t) + x(t) \). We also allow the user to drag a particular pose in time to modify the dynamics. This gives us the timing delta function \( k(t) \), where \( y(t) = d(t) + x(t + k(t)) \). This way, the user can directly author kinematic offsets as well as timing differences between the source and target animations interactively.

When the user edits a frame, the edits are propagated to a user-controlled number of adjacent frames. We take the difference between the edited frame and the original frame, then smoothly blend in the difference in the frames leading up to the edited frame, and smoothly blend out the difference in the subsequent frames using a linear blend.

6. ANIMATOR WORKLOAD

To examine our model and better understand how it might be used in practice, we conducted structured interviews [Beyer and Holzblatt 1998] with four professional animators (three from major movie studios, and one with over 20 years of experience). We asked them about their work practices, and about which aspects of editing character animation are challenging. From these interviews, we distilled two areas they deemed as important and difficult that our technique can help with.

—Degree of Expressiveness. Traditional retargeting techniques make the target character move like the source, but such faithfulness to the source motion may not be desirable. The motion may not fit the target character because the target character may have different physical properties, movement style, etc. Getting highly individualized, expressive motion usually requires hand-editing. Our technique can propagate an artist’s edits, so the artist can create expressive animation without having to create entire sequences by hand.

—Graphic Strength. An animator is often careful to produce frames that look good kinematically and dynamically. High kinematic quality is sometimes referred to as graphic strength. Producing graphically strong frames can be difficult because of the kinematic detail involved. Therefore, producing long animations with graphic strength can be hard. Our system can help the artist by propagating poses that are graphically strong.

Our results suggest that the combination of an intuitive editing tool and a system to generalize and propagate edits can reduce the work associated with both of these aspects (see Section 7). These indications are subjective, and in the future, we would like to further evaluate our system by observing practicing artists use it.

7. RESULTS

7.1 Training Details

Rather than predict all target degrees of freedom given all source degrees of freedom, we predict each target degree of freedom separately. There are a total of three hyperparameters for each of the two models (configuration and acceleration). To model the covariance, we use a radial basis function with an independent noise term, so the first two hyperparameters are for the radial basis function (the constant multiplicative term in front of the exponential, and the characteristic length scale), and the third is the scale of the noise term. For each model, we use the same values of all three hyperparameters for all of the target degrees of freedom, which suggests that their precise value is not particularly important. We use different values of these parameters for the configuration model and for the acceleration model.

We solved for the hyperparameters with the maximum likelihood function minimize in the GPML Matlab package [Rasmussen and Williams 2006b], which uses the conjugate gradient method. This function finds the hyperparameter values that minimize the negative log-marginal likelihood of the training data. We set it to iterate a maximum of 100 times. We initialize the RBF hyperparameters to 0 and initialize the noise hyperparameter to \( \log(\text{sqrt}(0.01)) \). We solve for new hyperparameters every time new edit data arrives, but we suspect that it could be possible reuse hyperparameters for new data. Estimating all six hyperparameters requires around 30 seconds.

The training sequences are all shown in their entirety in the video. We give the count of unmirrored frames used here, but we always mirror the training data to automatically double the amount of training data we have. For the ranger leaning back, we used 550 source-target frame pairs. For the ranger leaning back while walking but standing up straight, we use 550 and 74 frames, respectively. For Jack walking,
we used 432 frames. For Jack walking and skipping, we used 432 and 441 frames, respectively. For the giant walking, we used 535 frames. For Elvis walking, we used 448 frames. For Elvis walking and skipping, we used 448 and 350 frames, respectively.

We found the weighting parameters relatively easy to adjust because if a term is too heavily weighted, telltale signs manifest in the target animation. If the kinematic prediction weight is too high, the character will adopt unnatural poses on occasion. If the dynamic predictions are too heavily weighted, the character will tend to take on unnatural poses more often towards the end of a sequence. If the regularization terms are too heavily weighted, then the prediction will look too much like the source. The first time we used the system, it required three iterations to get a set of parameter values that produced satisfactory results, and we used these same values of the weighting parameters throughout the rest of the examples. This suggests our system is not sensitive to the choice of parameters.

7.2 Animations

Using our method, we created animations for a skeleton character, a space ranger, a giant, and Elvis. The goal of the animation edits was to produce highly caricatured motions for each character. Our technique was able to produce all of these different styles by automatically editing the same source motion capture clips. The motion capture database we used contains idling and different types of locomotion. We evaluated our results by inspection. We invite the reader to view our results in the accompanying video.

All of the artist-generated examples exhibited here were generated using the interactive motion editing system described in Section 5. We found it an intuitive way to fluidly edit animation.

The video demonstrates that characteristic properties of the example animations transfer well to the target motion. For example, we show that not only can we make the skeleton move in a highly caricatured fashion like the examples the artist provided, but that distinctive features of the training data manifest in the target animation. For example, the training data shows idiosyncratic marks, such as bowed legs. The artist we use for the video is a computer science graduate student, not a professional animator. The bowed legs and other idiosyncrasies are clearly visible in the target animation. In another example, the artist animates a giant character with a slow gait and short steps. These properties are preserved in the target animation we generated.

The work involved in editing frames is generally linear in the number of keyframes needed, which is usually proportional to the total number of frames. However, motion shows a great deal of repetition. Our method essentially encodes repeating structures in the animations so that the artist does not have to deal with every instance. Hence, the work involved in editing frames using our system can be considerably less than that in traditional editing methods. Because we need little training data, all of the training frames we used to generate each example are shown in the movie. For example, for each of the walking sequences, the artist needed to edit only three walking cycles.

As we demonstrate in the video, our method is directly controllable. The first part of the video features a space ranger character, who was animated directly from our existing motion capture database. We show that by providing a simple example, we can make him lean back while walking. By providing another example, we can remove the lean while he is standing. And by replacing the walking example with another, we can make him walk like a zombie (Figure 6).

Comparison with Hsu et al. [2005]. We prepared a supplemental video that compares our method with Hsu et al. [2005]. While the style transfer method and our method can both produce convincing human motion, one of the major advantages of our technique is that it requires significantly less training data.

Hsu et al. made publicly available the locomotion part of the motion corpus they used to demonstrate their technique. We excerpted clips from their “normal” walk to serve as source motions, then located corresponding target clips from their swaying, crouching, and jogging examples. To align corresponding source and target sequences, we used the iterative motion warping method described in their work. The crouching training data consisted of 3.25 seconds of source motion (or 389 frames at 120 Hz) paired with 422 frames of corresponding crouching. The swaying training data consisted of 2.21 seconds (269 frames) of source with 267 frames of corresponding swaying motion. The jogging training data contained a single walking cycle (1.21 seconds or 145 frames) paired with a single jogging cycle (107 frames).

Eugene Hsu provided us with his original implementation of the style transfer technique. We used his implementation to compare the style transfer method with ours. Starting with novel clips in the same “normal” style as the source motion, we generated corresponding target motions using the two methods. Hsu et al. report that their method requires a minimum of 1–2 minutes of training data (or 7200–14,400 frames at their capture rate of 120 Hz) to transfer a style successfully. Because we used significantly less training data, their system has difficulty fitting a model. Our technique can successfully transfer all three styles. We invite the reader to view the comparisons in the supplemental video.
7.3 Limitations

One limitation of our method is that Gaussian processes are not very scalable. For training, they demand \(O(n^3)\) time and \(O(n^2)\) storage, where \(n\) is the number of training frames. While there are methods like the informative vector machine [Lawrence and Platt 2004] which speed-up training considerably, such techniques may not actually be necessary for our problem. It generally takes a great deal of time to produce edits because they are done by hand. Therefore, we will usually face limited training data.

A second limitation is that we do not directly handle environmental contacts like footstrikes. Having a method that keeps contacts from slipping is desirable. Like many other techniques, we clean up contacts like foot strikes with an automated postprocessing step [Kovar et al. 2002; Ikemoto et al. 2005].

Third, our system assumes a single model of artist edits. In other words, our system cannot handle conflicting training data (Figure 7). This is because each of our predictors produces a single Gaussian distribution. We cannot model multimodality, which would be useful to capture conflicting artist edits, for example.

Fourth, our method can fail to generalize the artist’s edits to motions that are very different from the ones in the training set (Figure 4). In Figure 4, the top row of input poses are all similar to each other. The middle row shows the frames predicted by the kinematics predictor. The predictions are noticeably dissimilar because the input frames are very different from the training frames. However, because the kinematics predictor correctly tags these predictions with high variance, they are given low weight when the system computes the final output poses. Therefore, the final output poses look similar to the input frames and to each other. We invite the reader to take a closer look at the rightmost input frame on the top row in the same figure. While the other input frames were taken from skipping cycles, this frame is from a walk-to-skip transition. Kinematically it is similar to a skipping frame, but its dynamics consist of some combination of walking and skipping. The training data is of walking, so the dynamics predictor’s result has less variance than the kinematics predictor’s. Therefore, the system discounts the kinematics prediction (in which Jack’s arm is stuffed inside his body), but blends in the dynamics prediction (not shown, in which Jack’s arm is partially raised), to produce the final frame on the bottom.

A fifth limitation is that, currently, our implementation of our regression system is not real time. Estimating the parameters for the Gaussian processes consumes around 30 seconds, and producing frame estimations from the Gaussian processes proceeds at 2–3 frames per second. We did not optimize our code for speed, so we assume that these times could be considerably improved by a careful implementation. However, even these running times are fairly acceptable for our application.

Lastly, our technique may not be able to learn highly abstracted types of motion edits like “Add more zip to this motion” or “Make this character move more like this person.” Such edits may rely on larger temporal extents or on other features that our method currently does not support. Encoding meta-data-like motion semantics and editing intent would probably help.

8. CONCLUSIONS AND FUTURE WORK

We have presented a method that significantly reduces the amount of time needed to animate a new character through motion reuse. Our technique provides a directly controllable way to propagate context-dependent animation edits. We invite the reader to examine animations we produced using our technique in the accompanying videos. In the future, we would like to explore whether our technique can handle environmental constraints, such as foot plants, so that postprocessing is not required.

ACKNOWLEDGMENTS

We are very grateful to E. Hsu for providing his implementation of Hsu et al. [2005] and for suggesting points of comparison between the two methods. We thank M. Agrawala for helpful discussions, especially about interviewing animators and user interface design. This research was generously supported by ONR MURI N00014-01-1-0890, and benefited greatly from helpful discussions with the Berkeley and UT-Austin graphics groups.

REFERENCES


Received August 2007; accepted August 2008