Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2007)
D. Metaxas and J. Popovic (Editors)

Guided Time Warping for Motion Editing

Eugene Hsu Marco da Silva Jovan Popović

Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology

Abstract

Time warping allows users to modify timing without affecting poses. It has many applications in animation systems for motion editing, such as refining motions to meet new timing constraints or modifying the acting of animated characters. However, time warping typically requires many manual adjustments to achieve the desired results. We present a technique which simplifies this process by allowing time warps to be guided by a provided reference motion. Given few timing constraints, it computes a warp that both satisfies these constraints and maximizes local timing similarities to the reference. The algorithm is fast enough to incorporate into standard animation workflows. We apply the technique to two common tasks: preserving the natural timing of motions under new time constraints and modifying the timing of motions for stylistic effects.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism

1. Introduction

A motion that at first appears perfect, whether it is keyframed, captured, or synthesized, will often require further adjustment when it is combined with other elements in a scene. Modern animation systems provide a number of tools that transform motions to meet new requirements. The *time warp* is one such tool that allows users to adjust the timing of animated characters without affecting their poses. Its importance follows from the fact that poses often must remain fixed in later stages of the animation pipeline, since they provide the interface through which artists collaborate. Any significant spatial changes to a character often necessitate equally significant changes to the environment, lighting, or camera placement. Timing is one of the few aspects that can be changed without significantly disrupting workflow.

Typical time warps, however, require significant manual intervention. After describing high-level synchronization or duration requirements using keytimes, current animation tools also require users to adjust a curve to achieve the desired effect. These manual steps are further complicated by the fact that changes to timing, unlike changes to poses, inherently cannot provide instant visual feedback. Manual time warping typically requires alternating between tuning the time-warp curve and reviewing the results. This process is laborious even for the most talented animator.

SCA 2007, San Diego, California, August 04 - 05, 2007

Our guided time warping algorithm aims to simplify this task. As with traditional techniques, users begin by specifying rough timing constraints using keytimes. Instead of tedious manual refinement, users can select a reference motion that exhibits the desired timing properties. The algorithm then computes a time-warp curve that satisfies the constraints and mimics the timing properties of the reference.

Internally, our technique solves a discrete optimization that balances the goals of preserving the original motion and mimicking the timing of the provided reference. The constraints of the optimization emphasize content preservation to ensure that the results do not circumvent the intent of the user. The objective function uses a local measure of timing similarity that guides the solution towards the reference without requiring registered motions.

We demonstrate the applicability of guided time warping to several common timing tasks. It is often necessary, for instance, to adjust the duration of a motion to meet time constraints, synchronize it with other scene elements, or match external signals. Existing animation systems only offer spline-based time warps which generally require significant tuning to achieve natural results. Here, guided time warping can use the motion as its own reference to preserve its natural timing properties without needing additional adjustments. This is illustrated in Figure 1.

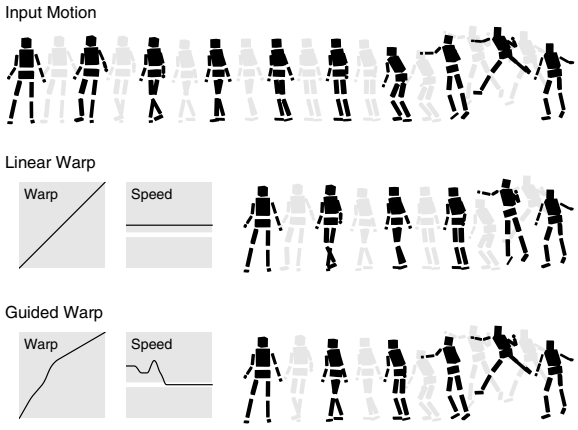

Figure 1: *Time warps are often applied to modify the durations of motions.* TOP: *In the provided input motion, a character turns, pauses, and jumps.* MIDDLE: *To shorten this motion, spline-based warping techniques perform a linear warp that speeds up all parts of the motion equally. This yields an implausibly quick jump.* BOTTOM: *Our technique guides the warp towards a more natural result without requiring additional constraints.*

Careful adjustments of timing can also be used to affect critical qualities of animation. Animators routinely use time warps to change factors such as anticipation, weight, emotion, and style. Such tasks require a great deal of skill and patience. Guided time warping can be used to propagate carefully tuned timing modifications to other portions of an animation. For instance, an animator can edit a single gait cycle and use our technique to replicate its timing modifications to an entire set of walking motions.

We present our technique as a new motion editing tool that addresses one of the most important aspects of animation: timing. The application of time warps can achieve surprisingly flexible edits without disrupting carefully tuned spatial relationships. Guided time warping can help animators achieve these results more easily and has the potential to augment existing motion synthesis techniques.

2. Related Work

The usefulness of the time warp operation is underscored by its almost universal appearance in professional animation tools such as Autodesk Maya, 3D Studio Max, and Avid SOFTIMAGE|XSI. In addition, time warping can be found in many video editing and compositing tools including Adobe After Effects and Apple Final Cut Pro. Within these tools, time warps are used routinely to create remarkable results. However, the methods available in these products are basic variants of the spline-based technique proposed by Witkin and Popović [WP95] and thus require a significant amount of skill and patience.

In the research community, the importance of timing in animation has long been appreciated. Lasseter's seminal work describes how even the slightest timing differences can greatly affect the perception of action and intention [Las87]. More recently, a number of interactive techniques have been proposed to address the timing problem. Performance-driven animation techniques allow the user to act out animations using various interfaces (e.g., [DYP03, TBvdP04]). Terra and Metoyer present a technique that is targeted specifically towards time warping [TM04].

Acting out a motion is a more intuitive way to achieve proper timing, but it still requires time and skill. McCann and colleagues [MPS06] describe a technique to compute physically accurate time warps. Our technique aims for greater generality by relying on data. By doing so, it can capture qualities of animation that may be difficult or impossible to describe using notions of physical consistency or optimality. Furthermore, many animations draw their appeal from their stylized interpretations of realism, which can not be described using physical techniques.

Motion data has been used successfully for many problems in computer animation. Nonparametric methods assemble short segments of motion data to generate results that match user specifications, such as desired paths on the ground or scripts of actions to perform [KGP02, LCR*02, AF02, AFO03]. Of particular relevance to our work are methods that detect repeatable actions [GVdG00] or record timing variations for a particular action [SSSE00]. The timing problem is only partially addressed by these techniques because they leave the original timing intact at all but a predetermined set of jump points.

Parametric methods employ different models of time. Blending and morphing methods (e.g., [BW95, RCB98]) extrapolate timing using linear combinations of time warps. Their flexibility is limited by the availability of multiple registered examples of particular actions. Statistical parameterizations add flexibility, but generally involve complex model estimation procedures for carefully constructed examples [HPP05] or large data sets [BH00, LWS02]. By focusing exclusively on timing, guided time warping avoids the need for explicit registration and computationally expensive learning procedures.

We drew motivation for our work from the speech domain, which has employed time warps to compress speech signals for the purpose of rapid review (e.g., [CWS98]). Such techniques perform nonlinear time warps to preserve comprehensibility, showing that far higher playback rates can be achieved than with linear warps. Unfortunately, they would be difficult to apply directly to our task because of their reliance on specific knowledge about speech understanding. The current understanding of motion in computer graphics does not provide the same guidelines.

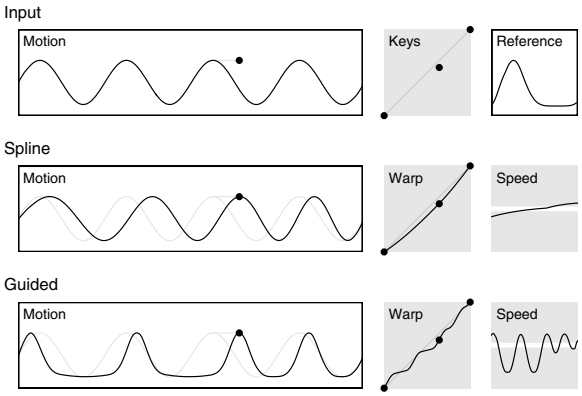

Figure 2: TOP: *A user provides an input motion, a set of keytimes, and a reference motion.* MIDDLE: *A spline-based technique computes a smooth warp that satisfies the constraints but ignores the reference motion.* BOTTOM: *Guided time warping satisfies the constraints and approximates the timing properties of the reference motion. A spline-based technique would require significant manual refinement to achieve the same results.*

3. Method

The guided time warping problem can be stated as follows. A user provides an *input motion*, *keytimes*, and a *reference motion*. Our technique aims to compute a time warp that interpolates the keytimes, yielding an *output motion* that is similar to both the input and the reference, as shown in Figure 2. These two conflicting goals are reconciled by using different definitions of similarity.

For similarity to the input motion, we enforce constraints on the time warp to ensure that it preserves the content of the input motion independently of temporal transformations. This is vital to ensure that our technique does not undermine the intent of the animator. Our constraint formulation is inspired by dynamic time warping. We realize these constraints in a discrete formulation (§3.1).

For similarity to the reference, we define an objective that evaluates the *local* similarity of a candidate output motion to the reference. The input and reference motions will often contain different sequences of actions, precluding the use of standard correspondence techniques. Instead, we pursue local *timing* similarities, in contrast to the global *content* similarities achieved by dynamic time warping. A key intuition in our work is the application of a local score function that captures the timing properties of the reference motion (§3.2).

The objective function, subject to the previously mentioned constraints, is minimized by mapping the problem to a constrained path search. This yields an efficient dynamic programming solution (§3.3). Simple postprocessing operations are then applied to achieve the desired results (§3.4).

3.1. Constraints

Time warps can only reproduce poses from the input motion, but their general application can still yield results that look vastly different. An animator may desire such effects, but producing them automatically without explicit instructions to do so can lead to unexpected and undesirable results. For instance, a general time warp could be used to lengthen a walk motion by repeating gait cycles. In earlier prototyping, this may be acceptable. But when environments are considered, such a change could mean the difference between approaching an obstacle and walking straight through it. Our technique realizes these concerns by enforcing constraints that ensure that the content of the input is preserved.

We borrow our constraint formulation from dynamic time warping [RJ93]. This algorithm is commonly used in computer animation to find global correspondences between motions (e.g., [RCB98]), but it can also be viewed as a time-invariant similarity measure. The latter perspective offers a convenient and proven way to define preservation of input content: the result of the any time warp must be *identical* to the input motion under the allowed transformations. Since time warps do not change poses, this can be performed by simply enforcing the constraints of dynamic time warping within our optimization.

In the context of our problem, the constraints of dynamic time warping can be stated as follows. It requires *complete* time warps, in that they subjectively contain the entirety of the input motion. In other words, large blocks of motion should not be deleted, although it is acceptable to speed through them quickly. It also requires *monotonic* time warps, in that they do not loop or reverse time.

We enforce these constraints in a discrete time compression, which we define as follows. Given a sequence of input frames $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, any time warp can be specified by a subsequence of its frame indices. The identity time warp is defined by the complete subsequence $\{1, \ldots, n\}$. Any subsequence of the identity time warp yields a monotonic time warp, as its indices must be strictly increasing. Any monotonic time warp is also complete if it includes 1 and n, and satisfies the property that no two adjacent elements differ by more than some integer s. The latter constraint forces the speed of the output to not exceed s times the original speed; this prevents the time warp from skipping large blocks of frames. These concepts are illustrated in Figure 3.

Time compressions are only one of many possible time warps. However, we can reduce other desired warps to compression by suitable transformations. Time modifications and expansions are reformulated as compressions by upsampling the input. Backward time warps can be converted to forward time warps by reversing the input. Finally, time warps with multiple keytimes can be split into subproblems of one of the aforementioned cases. As such, we will limit our subsequent discussion to the case of compression.

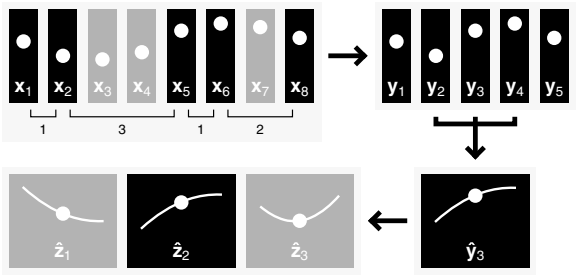

Figure 3: *An input motion is compressed by selecting a subsequence of its frames. Our constraints require that the first and last frames are retained and that no gap between selected frames exceeds a certain amount. The resulting output is scored by computing local velocity and acceleration estimates and comparing them to equivalently computed values in the provided reference.*

3.2. Objective

We aim to compute the quality of a candidate time warp with respect to a provided reference motion. As mentioned before, finding a global correspondence between the input motion and the reference motion is generally neither possible nor desirable.

Our technique uses a measure of local similarity to the reference motion. It stems from the following intuition: a pose is an indication of activity, and that activity is likely to be executed with a certain timing. For example, a sitting pose changes more slowly than a jumping pose. The input motion may already exhibit such canonical timings, but time warps induce changes to the velocity and acceleration of each pose. Certain poses, such as ones from walking, exhibit natural variations in velocity and acceleration. Other poses, such as ones from jumping, exhibit nearly constant acceleration. At a minimum, our objective should capture these variations.

Our solution uses a local score function that computes whether the velocities and accelerations induced by the time warp resemble those that appear in the reference motion. In the context of discrete compression, this can be more formally defined as follows. Suppose a candidate time warp yields an output of $\{\mathbf{y}_1,\ldots,\mathbf{y}_m\}$. Using finite differences, we compute feature vectors for each frame that encode pose, velocity, and acceleration. The local score of a frame is then defined as its distance to the nearest equivalently computed feature vector from the reference motion. The value of the objective function is defined as a sum of local scores:

$$\sum_{i=2}^{m-1} f(\mathbf{y}_{i-1},\mathbf{y}_i,\mathbf{y}_{i+1}). \tag{1}$$

To compute meaningful velocities and accelerations, we must first choose a proper representation of poses. We follow the approach of Arikan and Forsyth [AF02] and compute feature vectors for joint *positions* in the torso coordinate system of the character. This transformation allows frames to be represented independently of global position and orientation, thus increasing the generality of the local timing models.

The function $f(\mathbf{y}_{i-1},\mathbf{y}_i,\mathbf{y}_{i+1})$ is evaluated as follows. We use the adjacent frames to compute finite-difference approximations of $\dot{\mathbf{y}}_i$ and $\ddot{\mathbf{y}}_i$. These terms are assembled into a feature vector $\hat{\mathbf{y}}_i = [\mathbf{y}_i, \dot{\mathbf{y}}_i, \ddot{\mathbf{y}}_i]$. We perform a k-nearest-neighbor query on equivalently computed feature vectors for the reference $\hat{\mathbf{z}}_j$. The value of f is then defined as the average Euclidean distance between the feature vector and its nearest neighbors:

$$f(\mathbf{y}_{i-1},\mathbf{y}_i,\mathbf{y}_{i+1}) = \frac{1}{k}\sum_{\hat{\mathbf{z}}_j\in N_i} \left\|\hat{\mathbf{y}}_i - \hat{\mathbf{z}}_j\right\|. \tag{2}$$

Here, N_i is the set of k-nearest-neighbors to $\hat{\mathbf{y}}_i$. For the typical case of smaller reference motions, this can be performed with a linear search. For larger reference motions, we use an approximate nearest-neighbor data structure [AMN*98].

One concern that arises here is that of temporal aliasing. Our technique shares this issue with the dynamic time warping algorithm, as it also employs a discrete signal representation. However, we must take greater precautions, as aliasing artifacts can be exacerbated by the application of finite-difference estimators. We initially experimented with multiscale representations of the input motion to account for variable sampling, but found in the end that prefiltering with a uniform low-pass filter was sufficient.

Our decision to avoid higher-order derivatives was motivated by the need to balance the flexibility and generality of the local timing model. Higher-order derivatives would more accurately reflect the local properties of a given reference frame, but they would also be more sensitive to slight differences and provide less meaningful feature vectors. Ultimately, our choice was made by a combination of intuition and experimentation.

Before arriving at the current solution, we first tried a purely kinematic approach that attempted to preserve velocities and accelerations of the input motion. We found that this strategy works fairly well when the goal is to match the timing of the input motion as much as possible. However, it can not be applied to more interesting cases in which the input and reference differ.

We also considered a number of parametric statistical models for the conditional probabilities of velocity and acceleration given a certain pose. We found that, in practical use, the small sizes of the provided reference motions made it difficult to produce reliable estimates without overfitting. While various forms of priors and dimensionality reduction could be applied, we opted for the nearest-neighbor approach because of its relative simplicity and successful application towards a number of difficult modeling problems in graphics (e.g., [FTP03]) and other domains.

3.3. Optimization

Given the constraints and objective function described in the previous sections, we can define a globally optimal optimization procedure by transforming the problem into a constrained shortest path search. Note that, as with dynamic time warping, we define global optimality with respect to a discrete formulation.

We construct vertices that correspond to *pairs* of input frames that may occur together in the warped output. This allows us to represent the local terms of the objective function as edge weights. Formally, we denote each vertex by an ordered pair of frames $(\mathbf{x}_i, \mathbf{x}_j)$, where $i < j$ and $j - i \leq s$. We define directed edges connecting vertices $(\mathbf{x}_i, \mathbf{x}_j)$ to $(\mathbf{x}_j, \mathbf{x}_k)$ with weight $f(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k)$. This yields a directed acyclic graph with ns vertices and ns^2 edges.

Any path from $(\mathbf{x}_1, \mathbf{x}_i)$ to $(\mathbf{x}_j, \mathbf{x}_n)$ yields a time warp that satisfies the desired constraints, and the total weight of such a path is precisely the value of our objective. For example, suppose the input consists of frames $\{\mathbf{x}_1, \ldots, \mathbf{x}_8\}$. Then a time warp $\{1,2,5,6,8\}$ corresponds to the path $(\mathbf{x}_1, \mathbf{x}_2) \rightarrow (\mathbf{x}_2, \mathbf{x}_5) \rightarrow (\mathbf{x}_5, \mathbf{x}_6) \rightarrow (\mathbf{x}_6, \mathbf{x}_8)$. The edge weights sum to the value of the objective: $f(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_5) + f(\mathbf{x}_2, \mathbf{x}_5, \mathbf{x}_6) + f(\mathbf{x}_5, \mathbf{x}_6, \mathbf{x}_8)$. This example is shown in Figure 3.

A simple shortest path search will yield the optimal time warp of the input if no other constraints are given. Since the graph is directed and acyclic, such a search is linear in the size of the input motion: $O(ns^2)$. However, in most cases, we wish to find the optimal time warp given a target duration m (although we show one example in our results in which this is not the case). This condition can be satisified by constraining the number of edges in the shortest path search [Sai68]. Specifically, we want the shortest path from $(\mathbf{x}_1, \mathbf{x}_i)$ to $(\mathbf{x}_j, \mathbf{x}_n)$ with precisely $(m-2)$ edges.

The algorithm exploits the fact that the shortest path with p edges must contain a shortest path of $(p-1)$ edges. This is formalized by the following recurrence:

$$c_1(\mathbf{x}_i, \mathbf{x}_j) = 0, \tag{3}$$

$$c_p(\mathbf{x}_j, \mathbf{x}_k) = \min_i \, c_{p-1}(\mathbf{x}_i, \mathbf{x}_j) + f(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k), \tag{4}$$

where $c_p(\mathbf{x}_i, \mathbf{x}_j)$ is the cost of the shortest p-edge path to vertex $(\mathbf{x}_i, \mathbf{x}_j)$. The minimization proceeds by dynamic programming: first compute all 1-edge shortest paths, extend those paths to yield 2-edge shortest paths, and so on. The optimal warp can be recovered by backtracking from the vertex that minimizes $c_{m-2}(\mathbf{x}_j, \mathbf{x}_n)$.

Equivalently, this optimization can be viewed as a *standard* shortest path search on a directed acyclic graph with vertices $\{(\mathbf{x}_i, \mathbf{x}_j)\} \times \{1, \ldots, m-2\}$ and directed edges $(\mathbf{x}_i, \mathbf{x}_j, p) \rightarrow (\mathbf{x}_j, \mathbf{x}_k, p+1)$. Here, the $\times$ operator denotes the Cartesian set product. From this perspective, the time complexity of $O(mns^2)$ directly follows. Note that this is asymptotically identical to dynamic time warping.

So far, we have only described the case when the the first and last frames of the output are required to match the first and last frames of the input, respectively. Intermediate constraints may arise when more than two keytimes are specified. These can be enforced by requiring the path to pass through specific vertices $(\mathbf{x}_i, \mathbf{x}_j)$ at specific points p during the optimization procedure, which essentially restarts the optimization with a modified initialization.

This latter observation can be exploited to achieve a significant, although heuristic, speed improvement to our technique: we first solve a warp on a subsampled version of the input motion (say, at 5 Hz instead of 30 Hz). The low-resolution solution can then be used to set intermediate keytimes for a full-resolution search. In practice, we found that this optimization could significantly improve runtimes with minimal degradation of quality.

3.4. Postprocessing

Our technique produces a discrete approximation to the optimal time warp. A direct playback of the selected frames will often yield jumpy results. We resolve this issue by applying a moving average filter to the warp, which is then used to resample the joint angles of the input motion using quaternion slerp interpolation.

Time warps, regardless of how they are generated, can modify the frequency content of the input. For instance, compressions of time can cause a barely perceivable sway to become a nervous jitter. Conversely, expansions of time can yield overly smoothed results. Such results are sometimes desirable, but they can be distracting in many applications.

A simple solution to the compression issue is to apply a uniform smoothing to the output motion, but doing so also dampens important details such as environmental contacts. Such contacts should ideally be identified and incorporated into a smoothing process. Unfortunately, this is problematic for motion-capture data, as robust contact detection is difficult and prone to error.

We instead chose to extend the simple solution by applying a local averaging operation to each frame using a window size equal to the amount of compression. This approach exploits the observation that undesirable high-frequency details are only introduced during higher compressions and adapts the amount of smoothing accordingly. While it can still cause certain details to be lost, we found that it produces good results in practice.

The expansion issue is more difficult to resolve, as restoring high frequencies involves hallucinating information absent from the original motion. Existing techniques could be applied to restore detail and texture (e.g., [PB02]). We chose to present our results without additional modification because we did not find such artifacts to be very distracting.

4. Results

We demonstrate the application of our technique to editing longer input motions rather than interpolating closely spaced keytimes. This emulates the typical use of keyframing tools for time warping: animators first set keytimes to meet duration or synchronization requirements and then add intermediate keytimes to achieve the desired results. In these cases, guided time warping can be used to create a detailed time warp without further user intervention. We envision this to be the primary use of our technique.

All of our evaluations are performed on motion capture data downsampled to 30 Hz. We believe our technique to be equally applicable towards high-quality keyframe animation, but we unfortunately did not have access to such data sets. Given the difficulty of presenting our results in figure form, we instead refer to specific examples shown in our accompanying video as V1–V8.

Our first set of results (V1–V5) emphasize how our technique can change the durations of input motions while preserving their own natural timing properties. We also show results (V6–V8) in which we modify the timing properies for different stylistic effects.

4.1. Preservation

Time warping is commonly applied to modify the duration of motions while preserving the natural timing properties of input motion as much as possible. As described previously, our technique can be applied to this task by providing the input motion as its own reference. In our video, we compare our preservation results to those achieved by linear warping. Since duration changes are only specified with two keytimes, this emulates the behavior of spline-based solutions such as motion warping [WP95] when given the same task.

Our first evaluations (V1–V3) test a simple base case for our technique: silence removal. This commonly used approach for audio can be applied to motions by detecting periods of inactivity. However, a naive implementation of such a technique would be difficult to tune, as it is rare that motions are perfectly static. In contrast, our technique smoothly speeds through low-activity periods in the motion.

These evaluations highlight another desirable aspect of our technique: it manages to preserve the timing of ballistic activities (jumping and falling) without any prior knowledge of torques, constraints, or gravity. Instead, our objective function realizes that accelerations during such motions are practically constant and thus penalizes any temporal deviations. In fact, when presented with the task of shortening a more challenging motion with only walking and jumping (V4), the walking portions are sped up extensively while the jumping portions are preserved with framewise accuracy. In contrast, a linear scaling of the time axis yields slightly slower locomotion, but implausible jumps. These results are mirrored for expansions of the time axis (V5).

4.2. Modification

Sometimes time warps are used for the express purpose of deviating from natural timing. To apply our technique for such tasks, the user simply has to provide a reference motion with the desired timing properties. Note that our present formulation requires the reference motion to have the same skeleton as the input.

Our first modification evaluation (V6) shows a limited form of style modification to a weak boxing sequence. By providing an aggressive sequence of punches as a reference, our method warps the original input to provide the desired effect while maintaining its duration. Hsu and colleagues [HPP05] present a similar evaluation which relies on the availability of matched pairs of motions. Guided time warping has no such requirement: the provided input and reference motions can have a different sequence of punches. Although their technique can modify poses as well, there are benefits to restricting modifications to the time axis, as evidenced by our application of the computed time warp to video footage taken during the capture session. This is shown in Figure 4 and in our accompanying video.

While it may be possible to provide canned reference motions, it is often the case that the only motion available is the one to be modified. In these situations, animators typically must resort to manual warping. Our technique can applied to simplify this task. We captured a standard locomotion sequence and adjusted a small segment of it to look more like a limp (V7). This was a slow process, and repeating it for the rest of the motion would have been very tedious. Instead, we used guided time warping to propagate the timing properties of the modified segment to the remainder of the clip. One important note about this result is the somewhat unnatural upper body movement in both the reference and the output. This is an unfortunate artifact of any time warping algorithm, since poses in a true limp differ from those in a normal walk. In spite of this, we chose to include this example because it highlights the ability of our technique to work with very short reference motions.

McCann and colleagues [MPS06] demonstrated a physical approach to solve a similar task to the previous example using a modification of gravity. While this may appear to be simpler than our technique, it requires estimating physical parameters which are often unavailable for animated characters. Furthermore, it is often desirable to have motions that defy physical laws. We demonstrate one such application by automatically emphasizing jumps using slow motion (V8). For this example, we performed a manual time warp to a brief portion of the input motion and used it as a reference. The result shows the remainder of the jumps emphasized in a similar fashion, despite the fact that the jumps in the input differ significantly from those in the reference. For this evaluation, we did not care how long the output was, so we ran an unconstrained version of our shortest path search (§3.2).

Trial	n	m	r	s	Objective	Optimization	Heuristic
V1	493	280	493	8	1.3 s	2.7 s	0.17 s
V2	316	180	316	10	1.7 s	1.4 s	0.11 s
V3	378	180	378	8	1.9 s	1.4 s	0.14 s
V4	486	320	486	8	1.0 s	2.6 s	0.20 s
V5	401	150	101	4	0.3 s	0.7 s	0.06 s
V6	601	301	301	8	1.3 s	3.8 s	0.23 s
V7	573	143	45	8	0.9 s	1.8 s	0.14 s
V8	1524	590	91	8	2.5 s	0.1 s	—

Table 1: *Parameters and running times for our evaluations, measured on a 2.4 GHz Pentium 4 PC. The fast optimization time for V8 is due to its unconstrained optimization.*

4.3. Performance

Details of our evaluations are given in Table 1. We used $k = 3$ nearest neighbors for our local score function. The values n, m, and r are frame counts for the input, output, and reference motions, respectively. The value s is the frame skip limit, as defined in §3.1. These frame counts are given *after* reduction to compression. The time expansion for V5, for instance, was implemented by first upsampling a 101-frame input to 401 frames and then compressing it using the original 101-frame input as the reference. A similar tactic was employed for V6, V7, and V8, as they required both compressions and expansions of the time axis.

We provide separate timings for the objective evaluation (§3.2) and optimization (§3.3) components of our technique. The former was implemented in C++ using publicly available software [AMN*98], and the latter was implemented in MATLAB. The cited timings use our full optimization procedure. We also applied the heuristic from §3.3 by downsampling our motions by a factor of eight. This yields much faster optimization times with little effect on quality. However, it does not change objective evaluation times because it eventually performs a pruned search at full resolution.

5. Conclusion

Time warping is a fundamental motion editing operation that can be used for a variety of applications. However, time warps must be carefully constructed to ensure that the desired results are achieved. Guided time warping provides a simple alternative to tedious manual specification that can fit into existing animation workflows.

One limitation of our technique, as presented, is that it can sometimes be uncompromisingly automatic. After keytimes and a reference motion are provided, an animator has little control over the shape of the resulting warp curve. More keytimes could be specified, but it may also be desirable to add smoothness and tangent constraints. These would be difficult to encode into guided time warping due to its local objective. A possible solution might employ motion interpolation methods (e.g., [KG03]): by blending between guided warps and spline warps of the same motion, a user effectively has continuous control over the influence of the reference.

Figure 4: *Guided time warping is not only useful for warping animated sequences, but other types of signals as well. We used the output of our technique to warp video sequences taken during our motion capture session. Here, a weak boxing sequence is made to look more aggressive.*

Our results show the preservation of plausible jumps, but it is important to emphasize that our technique does not explicitly maintain physical consistency. When this is a priority, other techniques should be applied [MPS06]. A benefit using a data model is its capability to handle non-physical animations. Our focus on animated characters suggests the use of our technique for cartoons and special effects. However, a broader view of non-physical animation encompasses things such as camera movement, motion graphics, video footage, and so on. In future work, we hope to expand upon possible applications of our technique in these areas.

Ultimately, all time warps are limited by the fact that they can not create or modify actual postures. No amount of temporal adjustment can convert a slow walk into a fast sprint, let alone convert a poorly constructed motion into a good one. As such, we view our technique as complementary to motion synthesis methods in two regards. From an application perspective, guided time warping is intended for use in later stages of the animation pipeline, when full resynthesis is often impractical. From a technical perspective, our technique can add new capabilities to existing synthesis techniques by providing a more flexible representation of timing.

Acknowledgements

We thank the members of the MIT CSAIL Computer Graphics Group for valuable advice and assistance. This work was supported in part by the Singapore-MIT GAMBIT Game Lab and software donations from Autodesk.

References

[AF02] ARIKAN O., FORSYTH D. A.: Interactive motion generation from examples. *ACM Transactions on Graphics 21*, 3 (July 2002), 483–490.

[AFO03] ARIKAN O., FORSYTH D. A., O'BRIEN J. F.: Motion synthesis from annotations. *ACM Transactions on Graphics 22*, 3 (July 2003), 402–408.

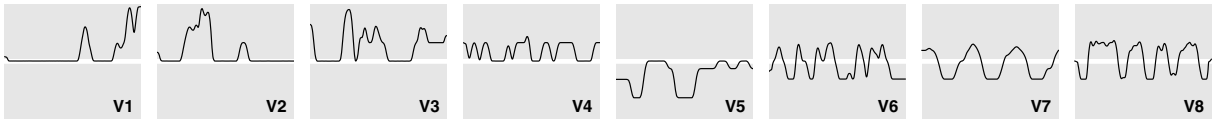

Figure 5: *Derivatives of time warps for our video evaluations, shown on logarithmic scales. The center lines in each plot correspond to unit time, and the limits correspond to tenfold slow-motion and fast-forward.*

[AMN*98] ARYA S., MOUNT D. M., NETANYAHU N. S., SILVERMAN R., WU A. Y.: An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM 45*, 6 (1998), 891–923.

[BH00] BRAND M., HERTZMANN A.: Style machines. In *Proceedings of ACM SIGGRAPH 2000* (July 2000), Annual Conference Series, ACM SIGGRAPH, pp. 183–192.

[BW95] BRUDERLIN A., WILLIAMS L.: Motion signal processing. In *Computer Graphics (Proceedings of SIGGRAPH 95)* (Aug. 1995), Annual Conference Series, ACM SIGGRAPH, pp. 97–104.

[CWS98] COVELL M., WITHGOTT M., SLANEY M.: Mach1: Nonuniform time-scale modification of speech. In *Proceedings of 1998 IEEE International Conference on Acoustics, Speech, and Signal Processing* (1998), pp. 328–338.

[DYP03] DONTCHEVA M., YNGVE G., POPOVIĆ Z.: Layered acting for character animation. *ACM Transactions on Graphics 22*, 3 (July 2003), 409–416.

[FTP03] FREEMAN W. T., TENENBAUM J. B., PASZTOR E.: Learning style translation for the lines of a drawing. *ACM Transactions on Graphics 22*, 1 (Jan. 2003), 33–46.

[GVdG00] GOMES J., VELHO L., DA SILVA F. W., GOLDENSTEIN S. K.: Motion processing using variable harmonic components. In *Computer Animation 2000* (May 2000), pp. 62–70.

[HPP05] HSU E., PULLI K., POPOVIĆ J.: Style translation for human motion. *ACM Transactions on Graphics 24*, 3 (Aug. 2005), 1082–1089.

[KG03] KOVAR L., GLEICHER M.: Flexible automatic motion blending with registration curves. In *Symposium on Computer Animation (SCA)* (Aug. 2003), ACM Press, pp. 214–224.

[KGP02] KOVAR L., GLEICHER M., PIGHIN F.: Motion graphs. *ACM Transactions on Graphics 21*, 3 (July 2002), 473–482.

[Las87] LASSETER J.: Principles of traditional animation applied to 3d computer animation. In *Computer Graphics (Proceedings of SIGGRAPH 87)* (July 1987), vol. 21, pp. 35–44.

[LCR*02] LEE J., CHAI J., REITSMA P. S. A., HODGINS J. K., POLLARD N. S.: Interactive control of avatars animated with human motion data. *ACM Transactions on Graphics 21*, 3 (July 2002), 491–500.

[LWS02] LI Y., WANG T., SHUM H.-Y.: Motion texture: A two-level statistical model for character motion synthesis. *ACM Transactions on Graphics 21*, 3 (July 2002), 465–472.

[MPS06] MCCANN J., POLLARD N. S., SRINIVASA S.: Physics-based motion retiming. In *2006 ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (Sept. 2006), pp. 205–214.

[PB02] PULLEN K., BREGLER C.: Motion capture assisted animation: Texturing and synthesis. *ACM Transactions on Graphics 21*, 3 (July 2002), 501–508.

[RCB98] ROSE C., COHEN M. F., BODENHEIMER B.: Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics and Applications 18*, 5 (1998), 32–40.

[RJ93] RABINER L. R., JUANG B.-H.: *Fundamentals of Speech Recognition*. Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[Sai68] SAIGAL R.: A constrained shortest route problem. *Operations Research 16*, 1 (Jan.–Feb. 1968), 205–209.

[SSSE00] SCHÖDL A., SZELISKI R., SALESIN D. H., ESSA I.: Video textures. In *Proceedings of ACM SIGGRAPH 2000* (July 2000), Annual Conference Series, pp. 489–498.

[TBvdP04] THORNE M., BURKE D., VAN DE PANNE M.: Motion doodles: an interface for sketching character motion. *ACM Transactions on Graphics 23*, 3 (Aug. 2004), 424–431.

[TM04] TERRA S. C. L., METOYER R. A.: Performance timing for keyframe animation. In *2004 ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (July 2004), pp. 253–258.

[WP95] WITKIN A., POPOVIĆ Z.: Motion warping. In *Computer Graphics (Proceedings of SIGGRAPH 95)* (Aug. 1995), Annual Conference Series, ACM SIGGRAPH, pp. 105–108.