Expressive Animation Retiming from Impulse-Based Gestures

Marie Bienvenu, Pascal Guehl, Quentin Auger, Damien Rohmer

Expressive Animation Retiming from Impulse-Based Gestures

Marie Bienvenu
LIX, École Polytechnique/CNRS, IP Paris
France

Pascal Guehl
LIX, École Polytechnique/CNRS, IP Paris
France

Quentin Auger
Dada! Animation
France

Damien Rohmer
LIX, École Polytechnique/CNRS, IP Paris
France

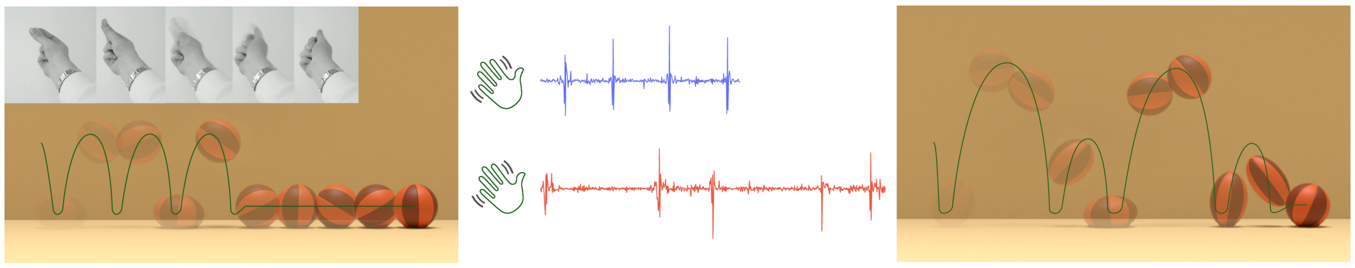

Figure 1: Given an input animation (left), our method can use the notion of timing expressed by impulse-like gestures (middle) to automatically compute a new retimed animation (right). The top gesture in blue is expressed by the user in synchronicity with the input animation, while the down one in red expresses the new expected timing.

Abstract

We present a method for retiming existing 3D animations able to handle seamlessly arbitrary impulse-like user gestures, thus enabling expressive video-based control inspired from common review sessions used in animation studios. The approach works in recording two videos with fast, impulse-based, gestures: one synchronized with the existing 3D animation and another featuring a new time sequence. We then propose an automatic generation of a modified 3D animation retimed to match the sequences of the second video. To this end, we introduce a robust and automatic method relying on Dynamic Time Warping able to compute the sequential correspondence between the timings of the impulse gestures. The method can adapt to various individual gestures without requiring dedicated learning, and can take into account the semantic integrity of the original 3D animation after retiming.

CCS Concepts

• **Human-centered computing** → *Interaction design*; *Interaction techniques*; • **Computing methodologies** → *Animation*.

Keywords

Animation, Authoring, Motion Editing, Gesture

ACM Reference Format:
Marie Bienvenu, Pascal Guehl, Quentin Auger, and Damien Rohmer. 2024. Expressive Animation Retiming from Impulse-Based Gestures . In *Proceedings of 17th annual ACM SIGGRAPH conference on Motion, Interaction and Games (MIG '24).* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3677388.3696323

Author's version published at
MIG '24, November 21–23, 2024, Arlington, VA
2024. ACM ISBN 979-8-4007-1090-2/24/11
https://doi.org/10.1145/3677388.3696323

1 Introduction

A major time-consuming task of 3D animated production is the generation of animated shapes, those animations are defined via the so-called *animation curves* depicting the spatial variation of the degrees of freedom (aka controllers) of the shape in time. In professional pipeline, such animations are almost never created in a single shot, but rather iteratively refined by the animator himself, or with their supervisor along *review sessions.* During these sessions the supervisor may ask modifications over an existing animation in order to refine it in time or magnitude. These modifications are typically expressed by simple mimicry gestures made over the running animation and convey a high-level notion of modifications that the animator must interpret and implement.

While a large body of research works in Graphics have focused on the generation of animation from scratch, typically using motion capture (MOCAP) or via physics-based simulation, using review-like sessions to automatically edit an existing animation remains largely overlooked. Indeed, analyzing such review sessions to compute an appropriate modification on the animation curve is a challenging problem. Each supervisor or animation may have their own way to express the modification they want using their gestures. As such, we cannot simply rely on specific pre-defined template gestures or existing databases where the mapping between gesture and modification could be learned from. Moreover, the modification applied to the animation must still preserve the nature of the 3D animation to be represented.

In this work, we will consider the case of 3D animation featuring key events at precise time, and we will call these as *impulse-based* *animation*. These animations include, for instance, spheres that bounce off walls, fireworks that are thrown and explode, or walking characters whose feets touch the ground at specific times. We then propose a method able to automatically compute a re-timing of such animation given a recorded video of the gesture, while seamlessly adapting to various individuals and/or gesture type as long as these convey the underlying notion of such impulses. We consider the following process. In a first step, the user video-records themselves performing gestures initially synchronized with the pre-existing 3D animation. In a second step, the user records another video while performing similar gestures with different timings. Our goal is to automatically compute a modified 3D animation that matches the new timing proposed in the second video.

The key technical contribution of our approach is to propose a robust method able to find optimal correspondence between impulses expressed in the two recorded videos of the user gesture. This correspondence must take into account the sequential ordering of the impulses, while being independent of the type of gesture and the number of impulses. To this end, we propose to rely on the notion of dynamic time warping allowing to express optimal correspondence between the two sequences. Once the correspondence is found, we propose a dedicated re-timing method which preserves the semantic of the initial 3D animation. We present our method on a set of shapes modified using either single hand or two hands gestures.

2 Related work

Different approaches have been proposed in research to synthesize animation from human inputs, starting from classical Motion Capture (MOCAP) [Gleicher 1999; Sharma et al. 2019] providing a 1-to-1 mapping between the sensor and the character joint. Such methods, like *Layered acting* [Dontcheva et al. 2003] and *KinÊtre* [Chen et al. 2012], focus on providing a direct mapping between the animator's movements and the character's joints. However, these methods are not always practical in real animation studio settings where animators need to review and edit their work without being encumbered by wearable devices, and switch back to animation software. In contrast, we focus on a "seamless integration" to avoid disrupting the animator's workflow.

More general gestures mapping approaches include works such as *MagicalHands* [Arora et al. 2019] and *Spatial Motion Doodles* [Garcia et al. 2019]. *MagicalHands* provides a versatile system for gesture mapping, although it is limited in real production interaction and editing scenarios. *Spatial Motion Doodles* offers an intuitive way to create animations through motion doodling, allowing animators to draw trajectories that characters will follow. Another approach involves the use of drawn sketches, as demonstrated in *Space-time sketching* [Guay et al. 2015], which allows animators to draw motion paths directly onto the animation timeline. *Authoring Motion Cycles* [Ciccone et al. 2017] focuses on creating new motion sequences by defining cyclic movements that fit within a predefined trajectory. Likewise, *Monster Mash* [Dvorožňák et al. 2020] enables the creation of new animations by sketching, but it is primarily designed for generating new content rather than editing existing animations. These methods are useful for creating entirely new animations, but are less effective for modifying existing ones to enhance the artistic vision of animators and supervisors.

Gesture-based approaches, such as those used in *HandAvatar* [Jiang et al. 2023], map human gestures to character actions. These methods often face the problem of ambiguity in gesture interpretation and require decisions about how to translate gestures into specific animation commands. *PlayAbility* software, for example, uses facial gestures for controlling virtual scenes in video games, especially for users with disabilities. A few inspirational works attempt to have more general mappings between the human body and a character [Chen et al. 2012; Dontcheva et al. 2003], but they focused mostly on character posing rather than on animation authoring. Puppetry-based approaches link specific gestures to control particular characters. Such methods, like those discussed in *Creature Features* [Seol et al. 2013], *AgileFingers* [Lin et al. 2024], *HandAvatar* [Jiang et al. 2023], and other performance interface approaches for interactive 3d character animation ([Oore et al. 2002] [Yin and Pai 2003] [Neff et al. 2007] [Lockwood and Singh 2012] [Lockwood and Singh 2016]), are often too character-specific and do not generalize well to different animators' gestures or to different characters. Additionally, *tangent-space optimization for interactive animation control* [Ciccone et al. 2019] offers precise adjustments for character movements, enhancing the animator's ability to refine animations.

Beyond gestures, sound inputs have also been studied in relation to character animation via natural language for lip synchronization [Zhou et al. 2018], or to music for dancing characters [Kim et al. 2003; Li et al. 2021]. Recent advancements in deep learning and virtual reality have further pushed the boundaries of animation synthesis and editing. Techniques such as those presented in *TimeTunnel* [Zhou et al. 2024a], *TimeTunnel Live* [Zhou et al. 2024c], and *Reframe* [Zhou et al. 2024b], focus on integrating spatial and temporal motion editing within virtual reality environments. However, these methods often rely on predefined gestures or extensive databases for training, which limits their adaptability to unique user inputs. Generating smooth human motion from sparse tracking inputs using diffusion models [Du et al. 2023] showcases how AI can fill in gaps in motion capture data, creating realistic animations even from limited inputs.

Performance-based approaches to keyframe animation timing have also been explored, as seen in works by Terra and Metoyer [Terra and Metoyer 2004, 2007] trying to track features in animation curves such as peaks, and *Dragimation* [Walther-Franks et al. 2012]. These approaches generally create animations from scratch, focusing on generating the *timings*, rather than retiming existing animations to match new user-defined sequences. *Dynamic Time Warping* approaches have been proposed by [Heck et al. 2006] and [Hsu et al. 2007] to control and editing character motion with retiming. Although the *Performance-based timing* methods presented in [Terra and Metoyer 2004, 2007] and the use of *Dynamic Time Warping* from [Hsu et al. 2007] are similar to our goal, our approach focuses on retiming by considering the differences between input and output animation curves, accommodating variations in the number of detected peaks. In contrast to their work, we adopt a more greedy method to explore the potential of our approach. Similarly, while *MagicalHands* [Arora et al. 2019] offers an intriguing general mapping from gestures to physical animation, our aim differs, as we seek to manage hand gestures without using a capture system or VR.

Additionally, sketch-based motion editing methods such as SketchiMo [Choi et al. 2016] offer intuitive and rapid adjustments for articulated characters' movements but do not address the need for preserving the semantic integrity of the original animation. The exploration of multimodal image synthesis and editing using generative AI [Zhan et al. 2023] demonstrates the potential of combining AI with traditional animation techniques to achieve lifelike animations. While these methods show great promise, they often lack robust sequential correspondence between user gestures and animation timings, a feature our method addresses with Dynamic Time Warping.

3 Method overview

We consider the three following inputs. First a reference 3D *impulse-based animation* corresponding to the current state of the animation made by an artist. The animation is represented in a generic way by one or more animation curves $C^{\text{in}}(t)$, where $0 < t < T^{\text{in}}$ represents the time, and C^{in} can represent one or more trajectories of the degrees of freedom of the animation. We note that the notion of *impulse-based animation* explained in the introduction indicates that the animation features key events at precise time, but does not imply specific criteria on the animation curves themselves that can remain smooth all along the animated sequence. We further consider a set of two supervisor video inputs. The first video input $V^{\text{in}}(t)$ represents the supervisor performing a gesture synchronized with the reference animation $C^{\text{in}}(t)$ during the time $t \in [0, T_{\text{in}}]$. The gesture itself performed by the supervisor can be arbitrary and does not necessarily mimic the shape trajectory as a MOCAP system would require, but it is assumed that they convey key-events via impulse-like gestures, i.e. clear changes of velocity in the motion during a short amount of time. The second video input $V^{\text{out}}(t)$, $0 < 0 < T^{\text{out}}$ represents the supervisor performing similar *impulse-based gestures* but with different timing. Our goal is to generate as output, a new animation $C^{\text{out}}(t)$, obtained in deforming the initial curves $C^{\text{in}}(t)$, but matching the timing of the second video input $V^{\text{out}}(t)$.

Our method works in two steps. First, in finding a correspondance between V^{in} and V^{out}, and second in deforming C^{in} to C^{out}. This correspondance is computed from the impulse-based gestures from the supervisor. To remain agnostic to the nature of gesture, we propose to detect the times corresponding to the impulses a generic change of apparent velocity rather than a dedicated analysis the images of the video. We propose to rely on a simpler scalar signal derived from the optical flow of the video. Let us call $O_p(V^{\text{in}}(t))$ the optical flow of the video V^{in} at pixel p and time t, and $\#P$ the number of pixels in the image. We define the scalar signal $s^{\text{in}}(t)$ as the average of the derivative of the norm of the optical flow over all pixels as

$$s^{\text{in}}(t) = \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{1}{\#P}\sum_{p\in\#P}\left\|O_p(V^{\text{in}}(t))\right\|\right), \tag{1}$$

and similarly for $s^{\text{out}}(t)$ with $V^{\text{out}}(t)$. The impulse gestures of the supervisor correspond to short accelerations and are therefore reflected by local peaks in both s^{in} and s^{out} signals.

4 Robust time correspondence between impulse signals

In this part, we propose a method able to find a sequential correspondence of times associated to peaks in the signal s^{in} with the times associated to peaks in the signal s^{out}. The main challenge is to handle such correspondence despite the noise associated with these two signals computed from videos, and without precise assumption of the shape and magnitude of these peaks.

A naive approach would be to attempt to segment each signal separately into peaks in detecting local maxima, before associating each of them in order. However, local maxima detection on a generic noisy signal is an ill-posed problem, and such approach would therefore rely heavily on user-defined thresholds. Instead, we propose the use of a method that fundamentally relies on a global sequential correspondence between the input and output signal.

4.1 Dense correspondence with Dynamic Time Warping

4.1.1 Correspondence as discrete path coordinates. We denote the discrete time samples of the signal s^{in} as t_x where the integer index x is such that $1 < x < N^{\text{in}}$ and $t_x = (x-1)/(N^{\text{in}}-1)\,T^{\text{in}}$, and similarly for s^{out} with the integer index y such that $1 < y < N^{\text{out}}$ and $t_y = (y-1)/(N^{\text{out}}-1)\,T^{\text{out}}$. A dense correspondence between the two signals can be represented a sequence of coordinate pairs that starts at $(1,1)$, ends at $(N^{\text{in}}, N^{\text{out}})$, and such that the pair following the index (x,y) is either $(x+1,y)$, $(x+1,y+1)$, or $(x,y+1)$. We call such a sequence a *path* P. Considering the space of all indices as a 2D map where $(1,1)$ is at the bottom left, and $(N^{\text{in}}, N^{\text{out}})$ is at the top right, a single step along the trajectory of such path can be interpreted geometrically as the following

- $(x,y) \to (x+1,y)$, horizontal step, corresponds to advancing time in the input animation while the time is stalled in the output animation. The associated re-timing would act as a local acceleration.
- $(x,y) \to (x+1,y+1)$, diagonal step, corresponds to advancing time in both the input and output animation. There is no local retiming.
- $(x,y) \to (x,y+1)$, vertical step, corresponds to stalled time in the input animation while advancing in the output animation. The associated re-timing would act as a local time compression.

4.1.2 Optimal path computation. To evaluate a notion of path quality, we introduce the discrete pairwise signal difference C called *cost* as the 2D map such as

$$C(x,y) = \left|s^{\text{in}}(t_x) - s^{\text{out}}(t_y)\right| \ . \tag{2}$$

A path P defined by the sequence (x_k, y_k) is associated to a global path score

$$E(P) = \sum_{k\in\mathcal{I}_P} C(x_k, y_k), \tag{3}$$

where $\mathcal{I}_P$ is a set of indices along the path P.

We propose to compute an optimal path minimizing $E(P)$ using the Dynamic Time Warping (DTW) algorithm. Beyond its efficient computation, DTW has the following advantages: First, DTW provides fundamentally a sequential solution, which therefore handles the notion of successive orders between the peaks of the two signals. Second, it provides a solution which is globally optimal without relying on a user-defined threshold, as such the correspondence between the peaks remains, up to a large extent, robust to the noise of the video-related signal, and agnostic to the magnitude of the

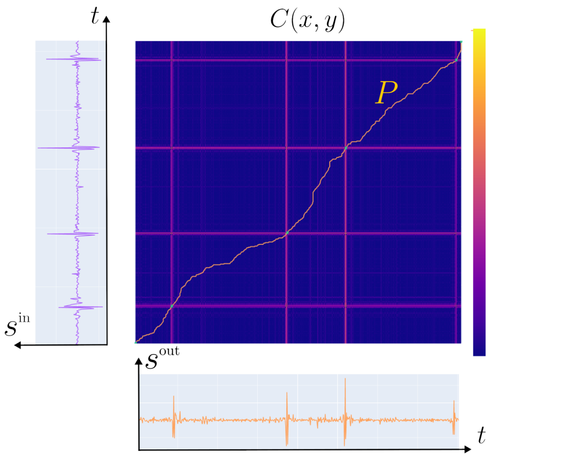

Figure 2: The cost map C computed from the values of s^{in} and s^{out} with high cost values in red/yellow, and the path P minimizing the cost computed from Dynamic Time Warping.

peaks. However, the dense path computed by DTW is not fully satisfying to compute a dense re-timing as is. Indeed, while the correspondence between the main peaks is well handled, the remaining correspondences between the other instants of the signals can slightly vary depending on noise, which would lead to spurious time extension, or contractions if applied directly.

4.2 Robust sparse peak-times extraction

Given a path P optimizing the global path score $E(P)$, we aim at computing the precise time coordinates $(x_{i^\star}, y_{i^\star})$ where two peaks are in correspondence in the input and output signal, i.e. $s^{\text{in}}(t_{x_{i^\star}})$ corresponds to the peak in $s^{\text{out}}(t_{y_{i^\star}})$. In the following, we consider the assumption where the same number of peaks N_{peaks} are present in the two signals. We first present why we can consider that such peaks can be defined by a set of indices $i^\star$ that are a subset of $\mathcal{I}_P$ by providing a geometrical interpretation of the structure of the 2D cost map, and then describe our formal approach for extracting the relevant peak-times.

4.2.1 Geometrical interpretation of a path. Let us provide a first intuitive description of the structure associated with the 2D map C as illustrated in Fig. 2. The latter is typically composed of a set of thin horizontal and vertical bars corresponding to high cost values. These straight bars represent time coordinates associating a peak in one signal to a low value in the other – a vertical bar corresponds to a peak in the input signal, while a horizontal one is a peak in the output signal. At the cross between an horizontal and a vertical high cost lies a few coordinates with lower cost value which corresponds to a link between two peaks in the two signals. The region is typically very small as it corresponds to the very few frames where the peaks between the two signals are synchronized.

When computing a path with DTW, the horizontal and vertical bars act as *barriers* to the path trajectory. In contrast, the thin region at the cross between such bars can be seen as narrow passages where the paths is "forced" to pass to avoid the high cost of the barriers. Our approach consists in leveraging the precision of such narrow passages computed by DTW to robustly define the specific

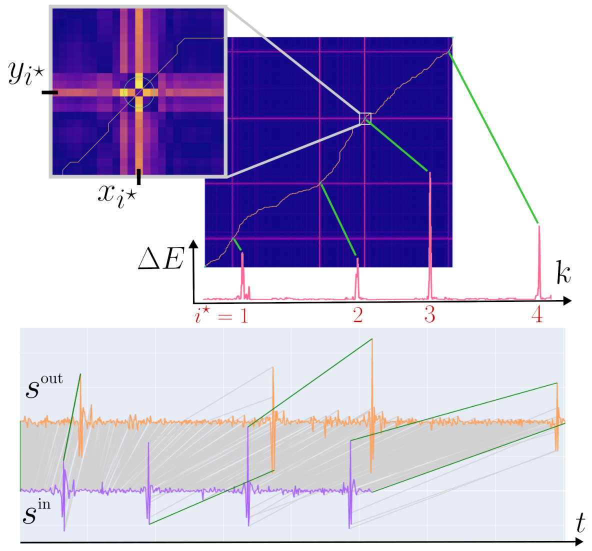

Figure 3: Top: The cost map with a close-up on the peak region defining $(x_i^\star, y_i^\star)$ at the central low-cost value surrounded by high-costs barriers. The red graph illustrates the value of ΔE along the indices of the path P and shows clear peak values. Bottom: The green lines show the final correspondence between s^{in} and s^{out}. The light gray lines are the dense correspondences depicted by P.

time correspondence where the peaks are located. We call $i^\star$ the path index associated to such coordinates that corresponds to the sparse times when the peaks in $s^{\text{in}}(t_{x_{i^\star}})$ corresponds to the peak in $s^{\text{out}}(t_{y_{i^\star}})$. Between two consecutive peaks, the path computed by DTW passes through regions on the cost map associated with low values spreading over a larger area. Such a path is less constrained and can vary depending on small noise variations. We therefore disregard this part of the path trajectory and enforce a consistent diagonal segment between two consecutive peaks to ensure regular time correspondence, and therefore a smooth final homogeneous retiming.

4.2.2 Peak extraction method. Our robust computation of the indices $i^\star$ rely on the core idea that such coordinates are critical passages for the trajectory of the path P computed by the DTW algorithm. Computing an alternative path that cannot pass by such coordinates should therefore be associated with a higher path score, while still remaining close enough to the ideal path P.

To compute such characteristics, we introduce a modified cost map $C_{|x_0 y_0}$ such that

$$C_{|x_0 y_0}(x, y) = \begin{cases} C(x, y) & \text{if } x \neq x_0 \text{ and } y \neq y_0 \\ +\infty & \text{otherwise} \end{cases} \quad (4)$$

and $P_{|x_0 y_0}$ the alternative optimal path obtained with DTW on the modified cost map $C_{|x_0 y_0}$, thus forced to avoid the coordinate (x_0, y_0).

For every index $k \in \mathcal{I}_P$ at coordinate (x_k, y_k) along the path P, our algorithm works in evaluating the two following quantities

$$\begin{aligned} \Delta E(P, k) &= E(P_{|x_k y_k}) - E(P) \\ \mathcal{D}(P, k) &= \text{Area}(P_{|x_k y_k}, \ P)/(T^{\text{in}} \, T^{\text{out}}) \ , \end{aligned} \quad (5)$$

where Area($P_{|x_k y_k}$, P) represents the positive area between the two curves delimited by $P_{|x_k y_k}$ and P. We then consider that the index $i^\star$ at coordinates $(x_{i^\star}, y_{i^\star})$ matches a corresponding peak if

$$\begin{cases} \Delta E(P, i^\star) > 2, \text{ and} \\ \mathcal{D}(P, i^\star) < 0.1\% \,. \end{cases} \tag{6}$$

The first criteria indicates that the path cost is significantly impacted by the absence of the peak at $(x_{i^\star}, y_{i^\star})$, while the second condition ensure that the alternative pass would remain close to the optimal one. The thresholds were found empirically. In the case where these criteria are true more than once in less than a quarter of second, we keep a single coordinate associated with the maximal value of ΔE in such interval in order to have a unique peak time.

As illustrated in Fig. 3, this method allows to extract the position of the correlated peaks very clearly in comparison to a more direct analysis of each input signal separately. Moreover, although the two criteria are associated with two user-thresholds, these are acting on global properties over the optimal path, and therefore are less sensitive to local values and noise of the signal.

4.3 Extension to different number of peaks

In real case application, the supervisor may express a different number of impulses in their input and output signals. In such a case, the path followed by the DTW algorithm will not only pass through the narrow passages associated to corresponding peaks, but will also necessarily pass through high cost regions, i.e. vertical or horizontal barrieres, that are associating a peak in one signal to a low value in the other.

These passages can be automatically detected as they are associated to pixel (x_i, y_i) with high cost value $C(x_i, y_i) > \lambda$, while not satisfying the criteria proposed in Eq. (6). Indeed, removing one pixel along a long barrier of high cost will not significantly impact the alternative path, which will continue to pass through the neighborhood of the forbidden pixel. Furthermore, the local orientation of the cost barrier can be evaluated via the gradient vector $\nabla C(x_i, y_i)$ such that a horizontal direction indicates that there is more peaks in the input signal than in the output signal, and conversely for a vertical direction.

In the case where more peaks are present in the input signal, we simply truncate the time of s^{in} to match the last peak and reiterate the algorithm to extract the relevant peak-correspondence. In the case where more peaks are present in the output signal, we propose to duplicate the last part of the input signal s^{in} as well as its corresponding animation C^{in}. In practice, we consider a copy of the time interval starting between the last two peaks detected in s^{in} and the end of the signal, and smoothly blend it with the original signal to create a coherent reference.

5 Semantic preserving animation retiming

In this section, we present how to apply the retiming in order to generate the resulting animation C^{out}. Let us consider the peak times $(x_{i^\star}, y_{i^\star})$ introduced previously that represent corresponding gesture impulses. We can define in the interval between such impulse an objective time scaling factor defined as piecewise constant

$$\lambda_i^{\text{pc}} = \frac{y_{(i+1)^\star} - y_{i^\star}}{x_{(i+1)^\star} - x_{i^\star}}, \tag{7}$$

for $i \in [1, N_{\text{peaks}} - 1]$.

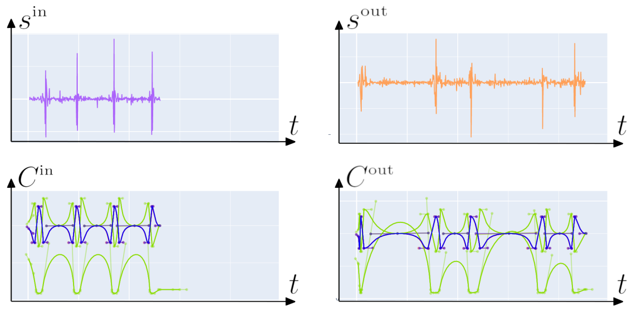

Figure 4: Bouncing ball animation retiming curves corresponding to the example in Fig. 1. Translation and scaling are shown respectively by the green and blue curves.

A straightforward method to retime the animation would involve directly applying such piecewise constant time warp to generate $C^{\text{out}}(t) := C^{\text{in}}(\lambda_i^{\text{pc}}(t))$, for $t \in [x_{i^\star}, x_{(i+1)^\star}]$. However, this approach would fail to maintain the semantics of the input animation as it would introduce tangential discontinuities even if the input animation is smooth. Instead, we propose to consider natively that the animation curves used in production are handled via key-frames, i.e. a value and derivative information at a specific time. In classical animation, impulses often correspond to keyframe positions. We take into account this first element in snapping our detected peak times to the closest keyframe time. This helps to consolidate the fact that this detected peak time is significant in the animation, and also improves the precise timing in case where the conductor doesn't perform his gesture in perfect synchronization with C^{in}.

We propose a least square formulation taking into account the underlying semantic of the animation curve depicted in C^{in} to find optimal scaling factors λ_i expressed, respectively on the left and right side of $(x_{i^\star}, y_{i^\star})$ as λ_i^- and λ_i^+. Our first energy term represents the attachment to the objective piecewise-constant values

$$\mathcal{E}_1 = \sum_i \left(\lambda_i^+ - \lambda_i^{\text{pc}}\right)^2 + \left(\lambda_i^- - \lambda_{i-1}^{\text{pc}}\right)^2 \,. \tag{8}$$

The second energy term expresses the preservation of smooth derivatives on the left and right side of the keyframe in the case where this constraint exists in the input animation. Calling $\mathcal{S}$ the set of keyframe indices where the derivatives are smooth in C^{in}, we define

$$\mathcal{E}_2 = \sum_{i \in \mathcal{S}} \left(\lambda_i^+ - \lambda_i^-\right)^2 \,. \tag{9}$$

The last term expresses the preservation of the scaling in the interval between two keyframes, thus preserving a symmetric curve if it is the case in the initial animation, as

$$\mathcal{E}_3 = \sum_{i \leq N_{\text{peaks}} - 1} \left(\lambda_i^+ - \lambda_{i+1}^-\right)^2 \,. \tag{10}$$

The optimal set of values for λ_i^+ and λ_i^- are obtained as the one minimizing the combination $\mathcal{E} = w_1\,\mathcal{E}_1 + w_2\,\mathcal{E}_2 + w_3\,\mathcal{E}_3$, where w_0, w_1, w_2 are user defined parameters. The final output animation is obtained as $C^{\text{out}}(t) := C^{\text{in}}(\lambda_i(t))$, where $\lambda_i(t)$ is obtained as the linear interpolation between the values $[\lambda_i^+, \lambda_{i+1}^-]$.

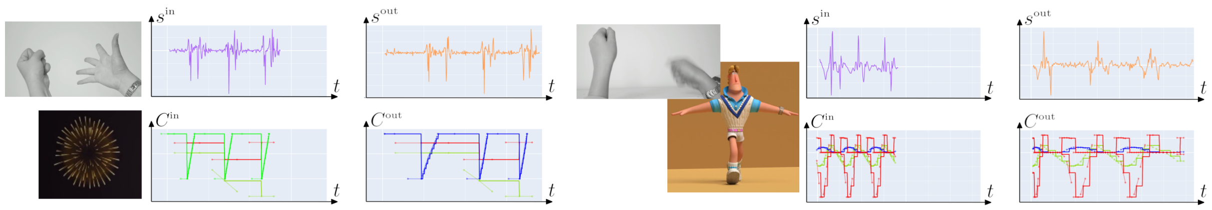

Figure 5: Left: Retiming of a firework with two hands opening and closing; Right: Retimed walking cycle with two hands hitting a table. The curves only show the right-hand signal.

6 Results and Discussion

We show the result obtained using our approach on three different scenarios featuring a bouncing ball, fireworks, and a walking character. The animated results are proposed in the accompanying video. The input videos were captured using a standard camera, Lumix DMC-G80, in Full HD 1920x1080 at 60 fps. To limit the need for video processing that is out of the scope of our work, we recorded our gestures in a clean environment with a focus on the user's hands. The computation of the optical flow was performed with OpenCV, and the rest of our retiming algorithm was implemented using Python, which typically takes 5 minutes to compute without any optimization. The 3D virtual scenes were created using Blender.

The first example is a bouncing sphere featuring squash-and-stretch effect. The animation curve is composed of the translation of the sphere in time and the time-varying scaling parameter. Fig. 1 presents the retiming obtained on the sphere, and the associated curves are depicted in Fig. 4. The gesture was performed with fingers snapping, and the impulse is expressed by the quick snap that is initially synchronized with the bouncing impact of the sphere. The number of impulses is higher on the new curve, and the final bounce in the output is obtained as a retimed version of the last bounce of the input animation. Note that our method is agnostic to the type of animation, the change of magnitude of the ball is automatically obtained when scaling the derivative values, and the squash-and-stretch effect adapts automatically to the retiming as being a part of the animation curve.

The effect of the semantic preservation is illustrated in Fig. 6 on a modified animation of the sphere sliding on the floor associated to smooth derivative constraints. Our optimized time scale values λ_i help maintain the appearance of the trajectory, thereby better preserving the dynamics of the original animation after retiming.

Another scenario is illustrated by the fireworks animation in Fig. 5-left where both hands are used to independently retime different-colored fireworks. In this case, we split the analysis of the video into a left and right parts to treat the two hands as two separate signals, one attached to the degrees of freedom of the yellow fireworks and the other to the red fireworks'. The user gesture is a sudden opening followed by a sudden closing of the hand, whose timing are synchronized with the appearance and disappearance of the fireworks. The edited animation curves include the firework position in space, as well as the channel representing the sprite texture animation, which features a discontinuous behavior.

We finally present a more complex application scenario with a walking character in Fig. 5-right. In this case, the user gesture is performed as brief hits with clenched wrists on a surface using both left and right hand. Compared to the previous ones, the scene is

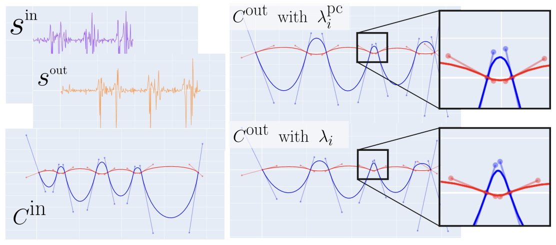

Figure 6: Semantic preserving time scaling applied to a smooth animation curve. The close-up view illustrates how our optimization helps to preserve the sharper curve of the second maxima already present in the initial animation and is associated with a dynamic change of direction.

closer to a production scene, as it features a fully-rigged character performing a much more complex motion. This example illustrates the particular case of motion cycles, which is also supported by our method granted the animation and gestures include "padding" repetitions before and after the cycle of interest, to mitigate edge effects. In this particular case, the degrees of freedom retimed by the user's gesture include the position, rotation and roll of both feet.

7 Conclusion and Future Work

We described in this short paper a method for retiming an existing animation using expressive gestures. Our method is based on a robust correspondence between two signals featuring impulses to perform piecewise-linear time scaling. Additionally, it addresses specific constraints in animation production, such as key-framing with smooth or split derivatives. Since our method targets retiming rather than full synthesis, it is highly adaptable to various types of 3D animations and user gestures, provided they include clear impulses.

The examples presented in this work, such as the bouncing sphere with squash-and-stretch, are simple but are also considered as the fundamental of animation principles from which more complex scenes are derived. We thus see this approach as a first step toward a more comprehensive and generic set of tools able to modify existing animations, which we intend to explore and validate with real-world CG animations and animators' feedback. We further propose the following extension in future work: First, we only considered a single signal from the gesture when looking at impulses. However, user gestures can also convey the notion of magnitude, direction and frequency, which would be interesting to investigate. Secondly, in our examples, the attachment of the input signal to the degrees of freedom of the animation curves was simple and done manually. For complex characters with numerous degrees of freedom, an automatic attachment could be implemented by correlating the gesture signals to the candidate animation curves, potentially through the computation of high-level features. Finally, additional expressive modality, such as the use of sound, would also be an interesting input to consider and could complement and/or dis-ambiguate the use of gestures to express the desired retiming.

Acknowledgments

This work is supported by the *Animation Conductor* project funded by the French National Research Agency ANR-23-CE33-0014.

References

Rahul Arora, Rubaiat Habib Kazi, Danny M. Kaufman, Wilmot Li, and Karan Pratap Singh. 2019. MagicalHands: Mid-Air Hand Gestures for Animating in VR. *ACM UIST* (2019).

Jiawen Chen, Shahram Izadi, and Andrew Fitzgibbon. 2012. KinÊtre: animating the world with the human body. In *ACM UIST*. 435–444.

Byungkuk Choi, Roger Blanco i Ribera, John P Lewis, Yeongho Seol, Seokpyo Hong, Haegwang Eom, Sunjin Jung, and Junyong Noh. 2016. SketchiMo: sketch-based motion editing for articulated characters. *ACM SIGGRAPH, Transactions on Graphics* 35, 4 (2016), 1–12.

Loïc Ciccone, Martin Guay, Maurizio Nitti, and Robert W Sumner. 2017. Authoring motion cycles. In *Symposium on Computer Animation*. 1–9.

Loïc Ciccone, Cengiz Öztireli, and Robert W. Sumner. 2019. Tangent-space optimization for interactive animation control. *ACM Trans. Graph.* 38, 4 (2019), 101:1–101:10.

Mira Dontcheva, Gary Yngve, and Zoran Popović. 2003. Layered acting for character animation. In *ACM SIGGRAPH*. 409–416.

Yuming Du, Robin Kips, Albert Pumarola, Sebastian Starke, Ali K. Thabet, and Artsiom Sanakoyeu. 2023. Avatars Grow Legs: Generating Smooth Human Motion from Sparse Tracking Inputs with Diffusion Model. In *CVPR*. 481–490.

Marek Dvorožňák, Daniel Sỳkora, Cassidy Curtis, Brian Curless, Olga Sorkine-Hornung, and David Salesin. 2020. Monster mash: a single-view approach to casual 3D modeling and animation. *ACM Transactions on Graphics (TOG)* 39, 6 (2020), 1–12.

Maxime Garcia, Rémi Ronfard, and Marie-Paule Cani. 2019. Spatial Motion Doodles: Sketching Animation in VR Using Hand Gestures and Laban Motion Analysis. *Motion, Interaction and Games* (2019).

Michael Gleicher. 1999. Animation from observation: Motion capture and motion editing. *ACM SIGGRAPH* 33, 4 (1999), 51–54.

Martin Guay, Rémi Ronfard, Michael Gleicher, and Marie-Paule Cani. 2015. Space-time sketching of character animation. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 1–10.

Rachel Heck, Lucas Kovar, and Michael Gleicher. 2006. Splicing upper-body actions with locomotion. In *Computer Graphics Forum*, Vol. 25. Wiley Online Library, 459–466.

Eugene Hsu, Marco da Silva, Jovan Popovic, et al. 2007. Guided time warping for motion editing. (2007).

Yu Jiang, Zhipeng Li, Mufei He, David Lindlbauer, and Yukang Yan. 2023. Handavatar: Embodying non-humanoid virtual avatars through hands. In *CHI Conference on Human Factors in Computing Systems*. 1–17.

Tae-hoon Kim, Sang Il Park, and Sung Yong Shin. 2003. Rhythmic-motion synthesis based on motion-beat analysis. *ACM Transactions on Graphics (TOG)* 22, 3 (2003), 392–401.

Ruilong Li, Shan Yang, David A Ross, and Angjoo Kanazawa. 2021. Ai choreographer: Music conditioned 3d dance generation with aist++. In *ICCV*. 13401–13412.

Yue Lin, Yudong Huang, David Yip, and Zeyu Wang. 2024. AgileFingers: Authoring AR Character Animation Through Hierarchical and Embodied Hand Gestures. In *Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*. 1015–1016.

Noah Lockwood and Karan Singh. 2012. Fingerwalking: motion editing with contact-based hand performance. In *Proceedings of the 11th ACM SIGGRAPH/Eurographics conference on Computer Animation*. 43–52.

Noah Lockwood and Karan Singh. 2016. Gestural motion editing using mobile devices. In *Proceedings of the 9th International Conference on Motion in Games*. 25–30.

Michael Neff, Irene Albrecht, and Hans-Peter Seidel. 2007. Layered performance animation with correlation maps. In *Computer Graphics Forum*, Vol. 26. Wiley Online Library, 675–684.

Sageev Oore, Demetri Terzopoulos, and Geoffrey Hinton. 2002. A desktop input device and interface for interactive 3d character animation. In *Graphics Interface*, Vol. 2. 133–140.

Yeongho Seol, Carol O'Sullivan, and Jehee Lee. 2013. Creature features: online motion puppetry for non-human characters. In *Symposium on Computer Animation*. 213–221.

Shubham Sharma, Shubhankar Verma, Mohit Kumar, and Lavanya Sharma. 2019. Use of motion capture in 3D animation: motion capture systems, challenges, and recent trends. In *International conference on machine learning, big data, cloud and parallel computing (comitcon)*. IEEE, 289–294.

Sílvio César Lizana Terra and Ronald A Metoyer. 2004. Performance timing for keyframe animation. In *Symposium on Computer animation*. 253–258.

Sílvio César Lizana Terra and Ronald Anthony Metoyer. 2007. A performance-based technique for timing keyframe animations. *Graphical Models* 69, 2 (2007), 89–105.

Benjamin Walther-Franks, Marc Herrlich, Thorsten Karrer, Moritz Wittenhagen, Roland Schröder-Kroll, Rainer Malaka, and Jan Borchers. 2012. Dragimation: direct manipulation keyframe timing for performance-based animation. In *Graphics Interface*. 101–108.

KangKang Yin and Dinesh K Pai. 2003. FootSee: an interactive animation system.. In *Symposium on Computer Animation*. 329–338.

Fangneng Zhan, Yingchen Yu, Rongliang Wu, Jiahui Zhang, Shijian Lu, Lingjie Liu, Adam Kortylewski, Christian Theobalt, and Eric P. Xing. 2023. Multimodal Image Synthesis and Editing: The Generative AI Era. *IEEE TPAMI* 45, 12 (2023), 15098–15119.

Qian Zhou, David Ledo, George Fitzmaurice, and Fraser Anderson. 2024a. TimeTunnel: Integrating Spatial and Temporal Motion Editing for Character Animation in Virtual Reality. In *CHI*. 1–17.

Qian Zhou, Aniruddha Prithul, Hans Kellner, Brian Pene, David Ledo, Sebastian Herrera Urrutia, Hilmar Koch, George Fitzmaurice, and Fraser Anderson. 2024b. Reframe: Recording and Editing Character Motion in Virtual Reality. In *ACM SIGGRAPH 2024 Immersive Pavilion*. 1–2.

Qian Zhou, Aniruddha Prithul, Hans Kellner, Brian Pene, David Ledo, Sebastian Herrera Urrutia, Hilmar Koch, George W. Fitzmaurice, and Fraser Anderson. 2024c. TimeTunnel Live: Recording and Editing Character Motion in Virtual Reality. In *ACM CHI*. 425:1–425:4.

Yang Zhou, Zhan Xu, Chris Landreth, Evangelos Kalogerakis, Subhransu Maji, and Karan Singh. 2018. Visemenet: Audio-driven animator-centric speech animation. *ACM SIGGRAPH, Transactions on Graphics* 37, 4 (2018), 1–10.