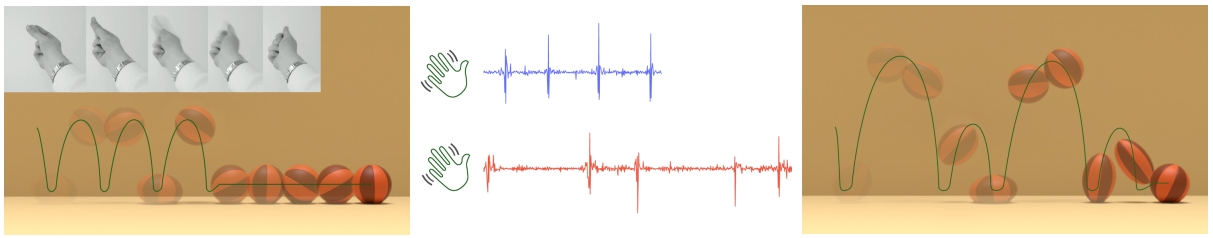
Project Report

Animation Retiming from Gestures

Marie Bienvenu
IMA Master, Sorbonne University, Paris, France



LIX, Polytechnique
Supervised by Damien Rohmer and Pascal Guehl

August 23, 2024

Acknowledgements

This internship was made possible thanks to the support of several people, to whom I'd like to express my sincere gratitude.

First of all, I'd like to express my sincere thanks to Damien Rohmer and Pascal Guehl, for their guidance throughout the internship. Their explanations and advice enabled me to carry out genuine research work, from start to finish.

Next, I'd like to thank the whole team at Dada! Animation, for their time and kindness, and especially Quentin, Clémence and Jonathan.

I'd also like to thank the entire VISTA research team, for their warm welcome and general help: Julia, Paul, Rodrigue, Pooran, Marie-Paul, Pierre, Rodrigo, Théo, as well as the members of the GéomeriX team, coffee break accomplices: Mathieu, Théo, Adrien, Xi, Julie and Robin.

Finally, a big thank you to Tara and Xavier, my office mates, for their company and eternal good humor.

Contents

1. Introduction

1.1 General context

Producing 3D animation for film, advertising or video games is a time-consuming and costly task. Currently, the most time-consuming stage of production is the animation phase, during which skilled animators describe the movement of scene elements over time. This stage takes place after the screenplay has been written, characters, objects and sets have been designed, and the skeletons of the elements to be animated have been put in place, and before the lights have been set up, the images rendered and the final colors composed (see Figure ??). It is therefore at the heart of a complex process, and acts as a bottleneck during production. Thus, optimizing it would represent a significant gain.

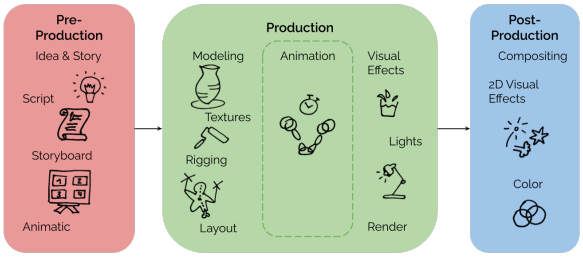


Figure 1.1: The 3D feature film pipeline splits into three major stages: pre-production (left), production (middle) and post-production(right). At the center lies the animation step, where skilled animators bring the characters to life.

When animating a 3D scene, the animator defines the *animation curves*, which describe the variation of each of the degrees of freedom of the *controllers* (the skeleton elements or proxy objects) of the objects and characters over time. This definition is done iteratively, generally starting with the setting of key poses, i.e. the definition of the value of each controller at a precise instant. Intermediate poses are then created, in order to perfect the interpolation until the desired result is achieved (see 1.2).

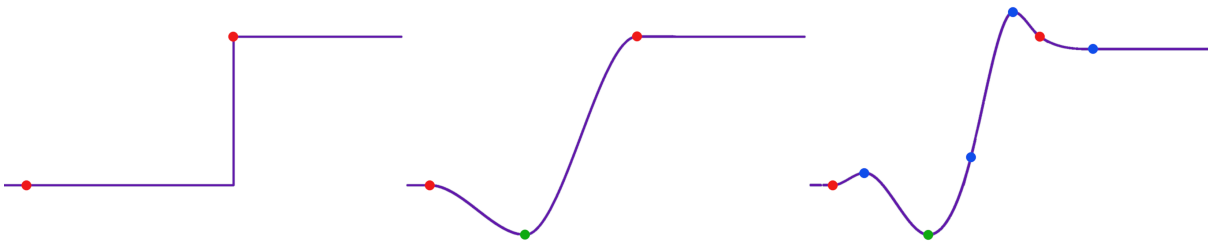


Figure 1.2: An animation curve may take several forms along the artistic process. At first, it is only the constant interpolation between key poses (left). Next, it progresses into a smooth interpolation, after adding intermediate poses called breakdowns (middle). The final, polished curve has many additional keys, such as anticipation, overshoots, and others (right).

At each stage, the animator has the animation validated by a supervisor, as the more detailed the animation, the more difficult it would be to change direction. These review sessions are also an opportunity to discuss any additions to be made for the next version. During these sessions, the supervisor may ask for modifications, to refine the animation in time (notions of rhythm) or space (legibility of poses). These modifications are generally

expressed by simple intuitive gestures performed on the animation in progress, as well as sound effects, and convey a high-level notion of the modifications that the animator must interpret and implement.

While much graphics research has focused on generating animation from scratch, typically using motion capture (MOCAP) or via physics-based simulation, the use of revision sessions to automatically edit an existing animation remains largely neglected. Indeed, analyzing such revision sessions to calculate an appropriate modification to the animation curve is a difficult problem. Each supervisor or animator may have his or her own way of expressing the desired modification using gestures and sound effects. So we can't simply rely on predefined gesture and sound templates or existing databases from which the correspondence between gesture or sound and modification could be learned. Furthermore, the modification applied to the animation must always preserve the qualities of the 3D animation to be represented.

This internship fell within the scope of this very broad problem, which will also be the subject of a thesis in the future. In order to carry out a complete project, it was decided, in conjunction with my tutors, to focus on modifications expressed through gestures.

ANR Animation Conductor. This work has been supported by the ANR *Animation Conductor* project, funded by the French National Research Agency (project ID: ANR-23-CE33-0014[1]), which is coordinated by the Dada ! Animation[2] studio, in partnership with the two research laboratories LIX (Laboratoire d'Informatique de l'Ecole Polytechnique) and IRISA (Institut de Recherche en Informatique et Systèmes Aléatoires). It started in September 2023 and will last 42 months. "*This project aims at the development of new methods for modifying animations of virtual shapes and characters from human voices and gestures, mimicking, sounding, exaggerating, or correcting the pre-choreographed motion. These methods will focus on the expressive control of the temporal aspect of events and deformations and will target the modification of existing animations: re-synchronization, adaptation of amplitudes, rhythms, and patterns; as well as the extrapolation to the synthesis of new ones. Several modalities will be studied as complementary approaches: sound recordings, gestures acquired by videos and by virtual reality tools; before proposing a methodology taking advantage of multimodal inputs. The project aims at concrete applications for the production of animated films with the integration of computer graphics animators and professional engineers at the core of the definition and progress of the project, as well as through open-source professional prototyping.*"

1.2 Structure of the internship

The work carried out during this internship was the subject of a publication submission to the MIG 2024[3] conference (ACM SIGGRAPH Motion, Interaction, and GAMES 2024), available in the appendix. Its content, which summarizes all the scientific contributions made during the internship, are detailed in Chapters 3 to 5. Other avenues explored during the internship, especially prior to the decision to publish, are presented in Chapter 6, and various technical elements, linked to the collaboration with industrial partner Dada ! Animation, in Chapter 7. Finally, results are discussed in Chapter 8.

[1] https://anr.fr/Project-ANR-23-CE33-0014
[2] https://www.dada-animation.com/en/
[3] https://sgmig.hosting.acm.org/mig-2024/

2. Related work

2.1 Full-body motion capture

Various approaches have been proposed in research to synthesize animation from human inputs, starting with classical motion capture (MOCAP) [Gleicher(1999), Sharma et al.(2019)] providing a 1-to-1 correspondence between sensor and character movement. These methods, such as *Layered acting* [Dontcheva et al.(2003)], *KinÊtre* [Chen et al.(2012)], and *Creature Features* [Seol et al.(2013)], aim to provide a direct correspondence between the animator's movements and the character's joints (see Figure 2.1). For this type of device, the animator takes on the skin of the character to be animated, which leads to restrictions on the morphology of the characters that can be animated, as well as on the difficulty of the gymnastics of the desired motions. In the case of *Layered acting*, this restriction on morphology is mitigated by a system of coupling between the different elements of the character's skeleton, and the different degrees of freedom of the mime; thus, it is possible to modify the animation of the eight legs of a spider with a single gesture. The system is based on a correlation calculation that is independent of temporal phase shift. In the case of *KinÊtre*, characters have no skeleton; a mesh deformation graph is created on the fly and then *attached* to the animator, so that any movement by the animator is reflected in the character. This makes it possible to animate arbitrary objects, such as chairs or wardrobes; however, as human morphology does not allow for the correct mimicry of a quadruped, it will take two people to animate a horse race. Finally, in the case of *Creature Features*, animation is based on the existence of a bank of pre-existing unit animations, associated with specific mimes; coupling the new mime with known mimes then enables the desired animation to be synthesized using the bank. This approach allows us to animate any character with a bank, but for a variety of gestures limited by the bank's content.

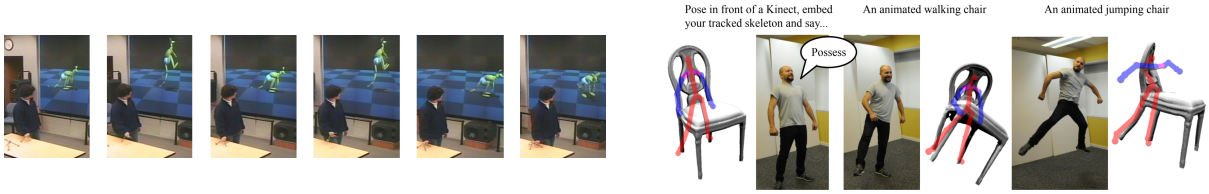


Figure 2.1: Both *Layered acting* (left) and *KinÊtre* (right) rely on motion capture devices, and transfer, in some way, the user motion to the object without modification.

These methods, while focusing more on animation creation than modification, offer interesting approaches; however, they have restrictions that are prohibitive in the context of our project, which aims to modify animations with an arbitrary degree of freedom on objects and characters of arbitrary morphology, without the need to mimic the desired animation.

Techniques such as those presented in *Time-Tunnel* [Zhou et al.(2024a)] and *Reframe* [Zhou et al.(2024b)] focus on the modification of animation already created, via motion capture in the case of *Reframe*. The modification is made possible by the creation of a virtual reality interface, based on precise gestures enabling spatial and temporal editing of the animation (see Figure 2.2). These approaches depart from our goal, as they don't

see gestures as high-level cues but rather as tools for directly manipulating poses and rhythm.

Figure 2.2: *Time-Tunnel* proposes a VR interface making animation editing simpler, by removing the usual separation between time and space.

2.2 Hand-based authoring and editing

Some approaches, such as *HandAvatar* [Jiang et al.(2023)] and *AgileFingers* [Lin et al.(2024)], rely specifically on hand gestures and poses to synthesize complex animations. They propose couplings between the various parts of the hands and fingers, and the joints of the characters to be animated; then, the user's gesture is transmitted to the character by optimization methods. More general approaches to gesture mapping 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, offering a comprehensive taxonomy of gestures, actions and desired effects following a survey of professionals. Their system, dedicated to VR, does not require direct mimicry of a controller's degrees of freedom, unlike Motion Capture systems; on the contrary, it encourages general modifications of spatial amplitude or temporality (see Figure 2.3). However, this degree of generality is accompanied by a loss of finesse, as demonstrated by the animations produced using this system. In the context of a real, professional-quality production, *MagicalHands* would be useful for animating physical phenomena - smoke cones, clouds, etc. - but would probably struggle to animate characters.

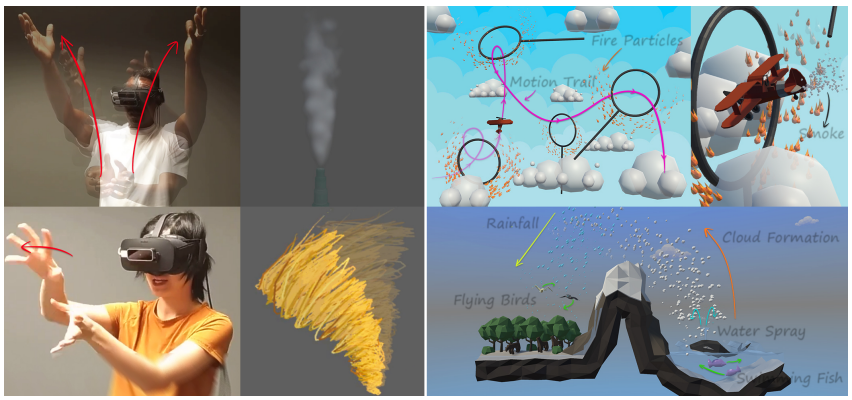


Figure 2.3: *MagicalHands* encourages intuitive editing of general spatio-temporal attributes through natural gestures.

Spatial Motion Doodles offers an intuitive way to create animations through motion doodling, allowing animators to draw trajectories for characters to follow (see Figure 2.4). This approach involves combining existing unit animations - jumping, walking, running in the example of the publication - to synthesize the animation desired by the animator, whether in terms of trajectory, rhythm or weight. These properties are maintained thanks to a classification system for the desired qualities, set against certain measurable properties of the gesture, such as curvature, twist and so on. This system approaches the notion of gesture semantics in a very interesting way, but is not applicable to all scenarios, since it relies on a previously created bank of animations.

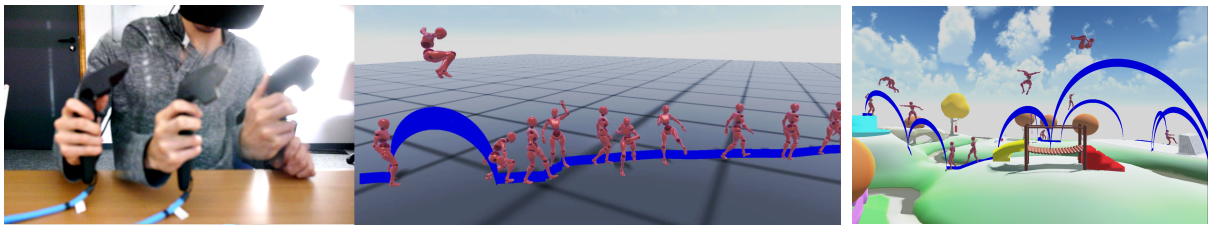


Figure 2.4: *Spatial Motion Doodles* offers a play-like experience able to synthesize various gymnastics from simple gestures.

2.3 Sketch-based approaches

Another approach is to use drawn paths, as shown by *Space-time sketching* [Guay et al.(2015)], which allows animators to draw movement trajectories directly on the animation timeline. Building on the notion of the line of action - a line that alone can characterize the energy and intention of a pose - well known to animators, *Space-time sketching* proposes the dynamic line of action, an evolution in time of the line of action that allows a complete animation to be synthesized with just a few strokes (see Figure 2.5). As it stands, this approach can only animate monopeds or flying animals, but the addition of secondary action lines would extend this limit to more complex characters. Furthermore, it identifies three major movement axes: the path, where the whole body follows the same path, the bounce, characterized by critical points, and the roll, which presents a loop; it is therefore difficult to apply to arbitrarily complex movements, which cannot always be seen as combinations of these three axes.

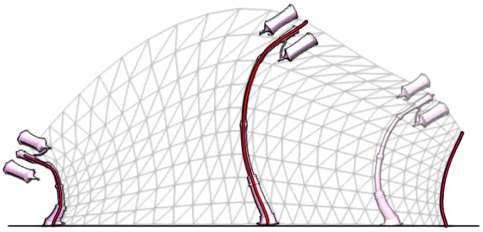


Figure 2.5: *Space-time sketching* introduces the Dynamic Line of Action, a surface representing the essence of an animation in both space and time.

Based on this technique, and taking it a step further, *SketchiMo* [Choi et al.(2016)] offers the possibility of modifying existing animations using sketches of lines of action (sometimes called body lines) and trajectories, and widens the field of possibilities to include bipeds making complex dance movements or obstacle courses, all while maintaining quality animation (see Figure 2.6). *Tangent-space optimization for interactive animation control* [Ciccone et al.(2019)] uses motion trajectories and curves to propose an animation modification technique that simplifies the classic interface, decoupling space and time.

Figure 2.6: *SketchiMo* allows the user to edit both trajectories and body lines with just a few strokes in an appealing interface.

On the other hand, *Authoring Motion Cycles* focuses on the creation of new motion sequences by defining cycles, i.e. animations that can be played in a smooth loop. Based on the repeated performance of cyclic motion via mouse drawing, *Authoring Motion Cycles* proposes a method which aligns the iterations in order to synthesize a noiseless average cycle. *Authoring Motion Cycles* then goes a step further, proposing a cycle visualization and editing tool that can be integrated into third-party software, enabling spatial and temporal manipulation of cycles (see Figure 2.7). These are explicit drawings, devoid of high-level interpretation, enabling the synthesis and modification of arbitrarily complex animations on characters of arbitrary morphology, all with remarkably fine control. Also based on animator performance, Terra and Metoyer's work [Terra and Metoyer(2004), Terra and Metoyer(2007)] and later *Dragimation* [Walther-Franks et al.(2012)] synchronize the animation of key poses with a drawing made on the fly (see figure 2.7). Their approach is to use the sketch as a rhythm cue, but retain existing trajectories, thus maintaining a level of quality; they're close to what we want to achieve via gestures, but choose fast methods to maintain an interactive response time, whereas we'll allow ourselves greedier methods in order to support more complex cases.

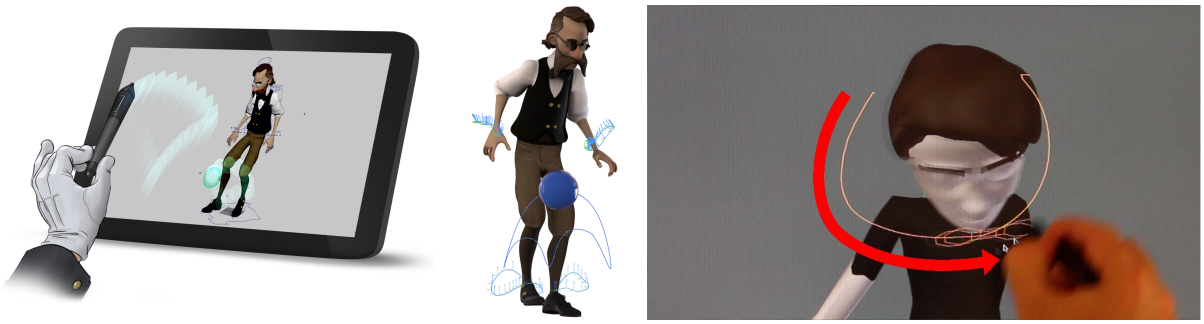


Figure 2.7: On the left, *Authoring Motion Cycles* records multiples sketched performances of a cycle to create a clean animation. On the right, *Dragimation* retimes an existing animation using a performed sketch with similar shape.

Also based on sketches, *Monster Mash* [Dvorožňák et al.(2020)] lets you create new characters and their animations through drawings. In this fully autonomous approach, 3D objects are inflated from drawings that make depth planes explicit and then, without an animation skeleton, animated by performing on screen the trajectory of control points directly on the shape's surface (see Figure 2.8.) This approach allows for objects with arbitrary geometry, but places a restriction on the animations, which must not alter the

order of the different planes: a leg located behind the body cannot pass in front of it. These are therefore planar animations.

2.4 Deep Learning

Finally, artificial intelligence has recently opened the door to a new wave of approaches. Recent advances in deep learning and virtual reality have further pushed back the boundaries of animation synthesis and editing. The playAbility[1] software, for example, uses facial gestures to control virtual scenes in video games, particularly for disabled users. *Generating smooth human motion from sparse tracking inputs using diffusion models* [Du et al.(2023)] shows how AI can fill gaps in motion capture data, creating realistic animations even from limited inputs. 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 realistic animations. However, these methods often rely on predefined gestures or extensive databases for learning, which limits their adaptability to unique user inputs.

Figure 2.8: *Monster mash* inflates drawings then lets the user animate them using sketched trajectories.

2.5 Sound-based editing

Beyond gestures and drawings, sound inputs have also been studied in relation to character animation via natural language for lip-syncing [Zhou et al.(2018)], or music for dancing characters [Kim et al.(2003), Li et al.(2021)]. These works fall outside of our scope, as we decided to focus on gesture-based retiming.

[1] https://www.playability.gg/

3. Overview

3.1 Contributions

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 feet 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 individual 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 himself performing gestures initially synchronized with the preexisting 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 (see Figure 3.1).

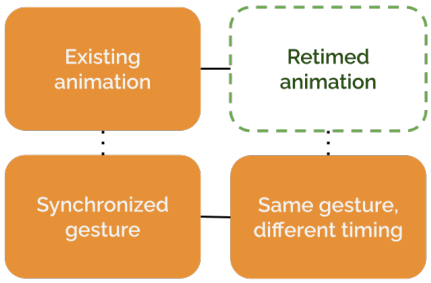


Figure 3.1: Inputs (solid orange boxes) and output (dashed green box) of our system. Some data are similar in nature but different in timing (solid lines), some are of different nature but synchronized (dotted lines).

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 (see Figure 3.2). 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 a single hand, or two hands gestures.

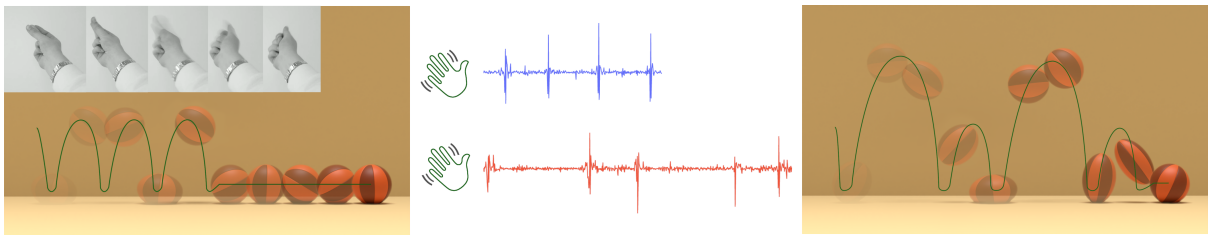


Figure 3.2: Given an input animation (left), our method can use the notion of timing expressed by impulse-like gestures (middle) to automatically compute a new retimed animation (right). The top gesture in blue is expressed by the user in synchronicity with the input animation, while the down one in red expresses the new expected timing.

3.2 Method and video analysis

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 $I^{\text{in}}(t)$ 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 he conveys the key-events via impulse-like gestures, i.e. clear change of velocity in its motion during a short amount of time. The second video input $I^{\text{out}}(t)$, $0 < 0 < T^{\text{out}}$ 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 $I^{\text{out}}(t)$.

Our method works in two steps: first, we find a correspondence between I^{in} and I^{out} (see Chapter 4), and second, we deform C^{in} into C^{out} (see Chapter 5). This correspondence is computed from the impulse-based gestures from the supervisor. To remain agnostic to the nature of the gesture, we propose to detect the times corresponding to the impulses by 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 $\mathcal{O}_p(I^{\text{in}}(t))$ the optical flow of the video I^{in} at pixel p, 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:

$$s^{\text{in}}(t) = \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{1}{\#P}\sum_{p\in\#P}\|\mathcal{O}_p(I^{\text{in}}(t))\|\right), \tag{3.1}$$

and similarly for $s^{\text{out}}(t)$ with $I^{\text{out}}(t)$. The impulse gestures of the supervisor corresponds to short acceleration, and are therefore corresponding to local peaks in both s^{in} and s^{out} signals, as shown in Figure 3.3.

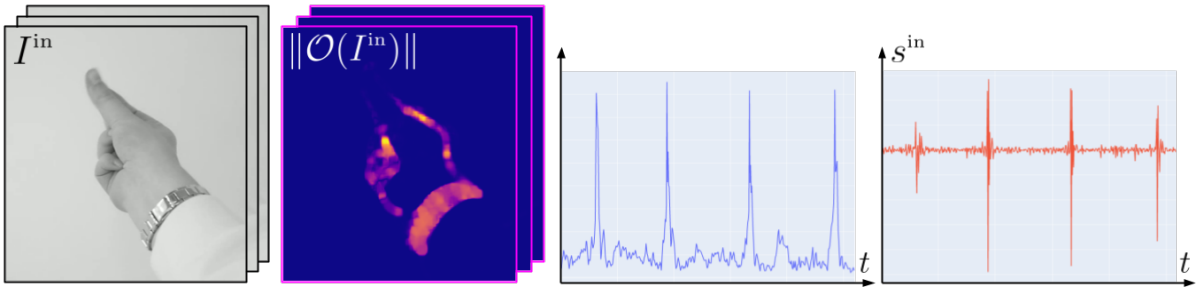


Figure 3.3: Given an *impulse-based* input video I^{in} (left), we first compute the magnitude of its optical flow (center left), then average it over space to get a scalar signal (center right). Finally, this scalar signal is derived over time to get the impulse signal s^{in} (right).

3.3 Artistic care

As mentioned earlier, the animator's work is part of a long creative process, requiring a high degree of anticipation. Every animation is therefore preceded by a discussion on the constraints of the shot - script, setting, shot length - as well as its intentions - level of energy and nature of the character's emotions in particular. From a theatrical point of view, each scene is part of an act; each scene must contribute something to the work, such as advancing the plot, or allowing the spectator to breathe. What should the spectator understand or learn from this shot? Where should they look? All these questions need to be answered before drawing the animation.

On top of all this, the animation itself must be of the highest quality: legible, plausible, pleasant to look at - in short, lively. All this comes from the expertise of the animator, who, through the meticulous description of each curve, will breathe life into the machine. In particular, he will keep in mind certain general rules, known as the 12 principles of animation and introduced in *The illusion of life* [Johnston and Thomas(1981)], which provide tools for creating lively animation. Although they date back to the era of hand-drawn animation, these principles are still extremely useful and respected today for computer animation. They include exaggeration, timing, anticipation, stretch and squash, arcs, appeal, and several others (see Figure 3.4).

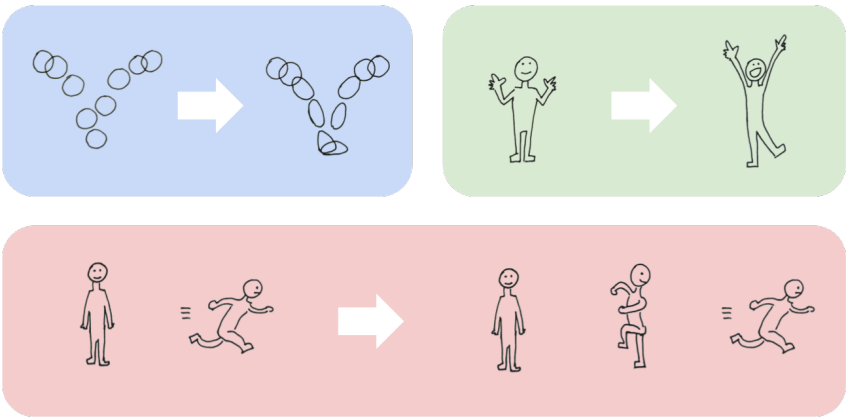


Figure 3.4: The 12 animation principles include squash and stretch (top left), exaggeration (top right), and anticipation (bottom), among others.

As our aim is to use expert gestural cues to modify existing quality animations, preserving these principles, and *a fortiori* the life in the animation, will have to be the subject of particular care when modifying animations. It's not only about matching curves and applying temporal warping; it's also a question of preserving the animator's choices and intentions for each curve, so as to be certain that what inhabits them - what we'll later call **semantics** - has been preserved. This topic will be detailed in Chapter 5.

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

$$C(x,y) = \left| s^{\text{in}}(t_x) - s^{\text{out}}(t_y) \right| \ . \tag{4.1}$$

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

$$E(P) = \sum_{k \in \mathcal{I}_P} C(x_k, y_k), \tag{4.2}$$

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:

- it provides fundamentally a sequential solution, which therefore handles the notion of successive orders between the peaks of the two signals.

- 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 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 present first why we can consider that such peaks can be defined by a set of indices $i^\star$ that are subset of $\mathcal{I}_P$ in 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. 4.1. 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 time correspondence where the peaks are located. We call $i^\star$ the path index associated to such coordinates that corresponds

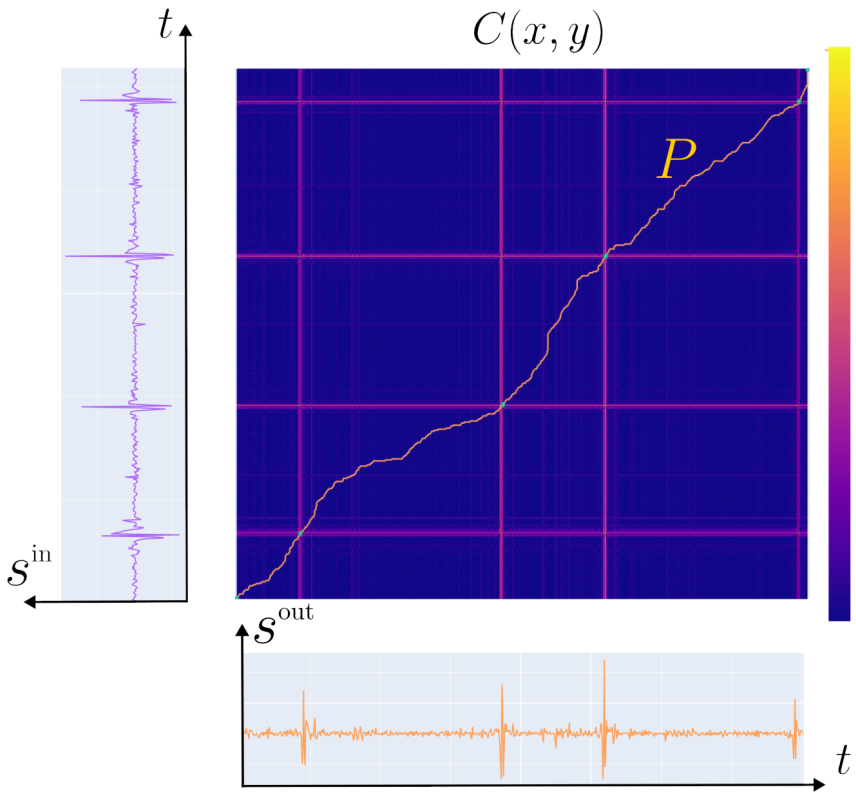


Figure 4.1: The cost map C computed from the values of s^{in} and s^{out} with high cost values in red/yellow, and the path P minimizing the cost computed from Dynamic Time Warping.

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 $C_{|x_0y_0}$ such that

$$C_{|x_0y_0}(x,y) = \begin{cases} C(x,y) & \text{if } x \neq x_0 \text{ and } y \neq y_0 \\ +\infty & \text{otherwise} \end{cases} \tag{4.3}$$

and $P_{|x_0y_0}$ the alternative optimal path obtained with DTW on the modified cost map $C_{|x_0y_0}$, 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

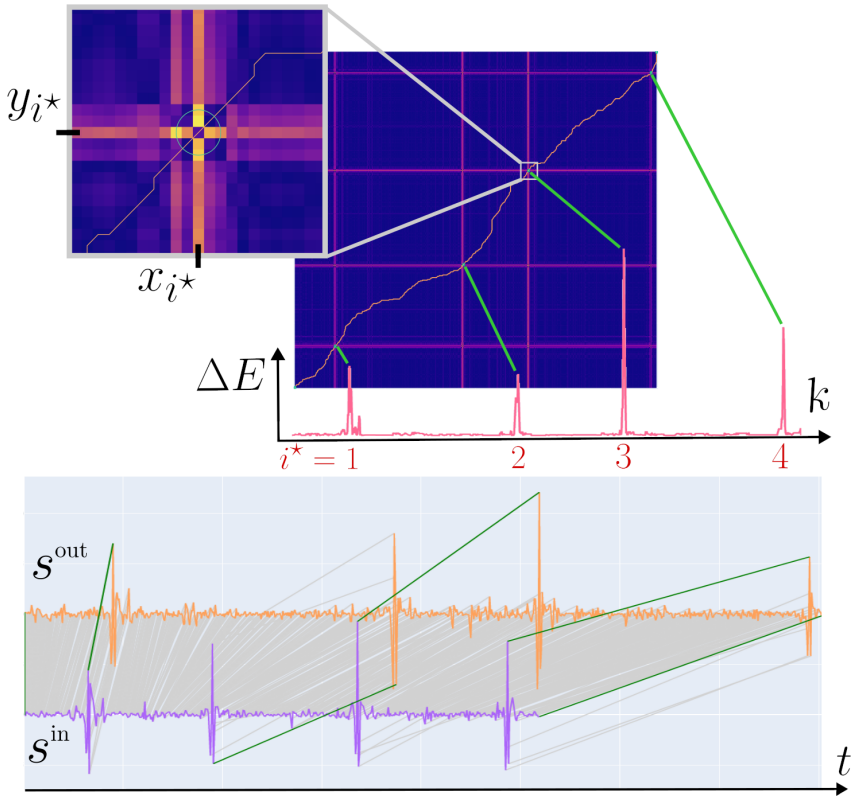


Figure 4.2: Top: The cost map with a close-up on the peak region defining $(x_i^\star, y_i^\star)$ at the central low-cost value surrounded by high-costs barriers. The red graph illustrates the value of ΔE along the indices of the path P and shows clear peak values. Bottom: The green lines show the final correspondence between s^{in} and s^{out}. The light gray lines are the dense correspondence depicted by P.

in evaluating the two following quantities

$$\begin{aligned} \Delta E(P,k) &= E(P_{|x_k y_k}) - E(P) \\ \mathcal{D}(P,k) &= \text{Area}(P_{|x_k y_k}\,,\ P)/(T^{\text{in}}\,T^{\text{out}})\ , \end{aligned} \tag{4.4}$$

where $\text{Area}(P_{|x_k y_k}\,,\ P)$ 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

$$\begin{cases} \Delta E(P, i^\star) > 2, \text{ and} \\ \mathcal{D}(P, i^\star) < 0.1\% \ . \end{cases} \tag{4.5}$$

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. In the case where these criteria are true more than once in less than a quarter of second, we keep a single coordinates associated with the maximal value of ΔE in such interval in order to have a unique peak time.

As illustrated in Fig. 4.2, 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 its 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. (4.5), as illustrated in Figure 4.3. Indeed, removing one pixel along a long barrier of high cost will not significantly impact the alternative path that will continue to pass through the neighborhood of the forbidden pixel.

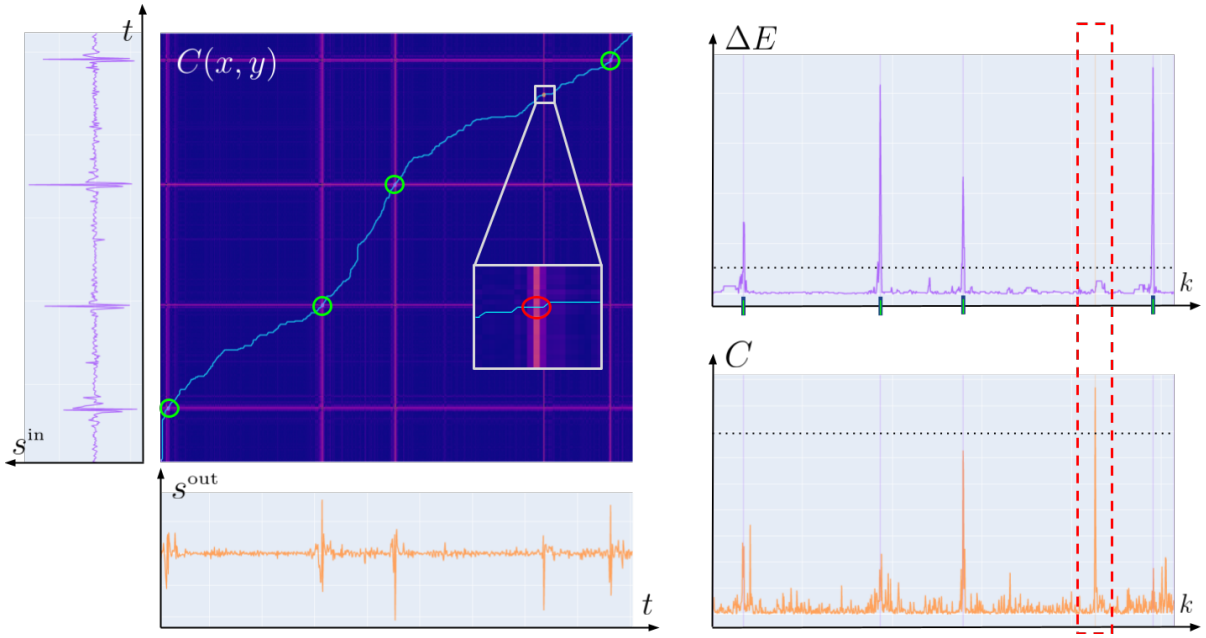


Figure 4.3: On the left: the cost map associated to signals with different number of peaks. The *outlier* pixel (red) is detected using the criteria (top right) as well as cost values (bottom right). In green, the peaks detected by the algorithm.

Furthermore, the local orientation of the cost barrier can be evaluated via gradient $\nabla C(x_i, y_i)$, such that an 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 re-iterate 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}, and then re-iterate the algorithm. In practice, we consider a copy of the time-interval starting between the last two detected peaks in s^{in} and the end of the signal, and smoothly blended with the original signal in order to create a coherent reference.

5. Semantic preserving animation retiming

In this section, we present our approach to applying retiming to generate the resulting animation C^{out} while preserving the semantics of the animation curves, in order to preserve as much of the animator's original intent as possible. In particular, we're going to look at the treatment of derivatives, which dictates the interpolation made between keys and therefore have a huge impact on the animation's dynamics. These derivatives impact the application of some of the 12 principles of animation, such as ease in/ease out, and more generally what animators call *spacing*, a fundamental concept in animation (see Figure 5.1).

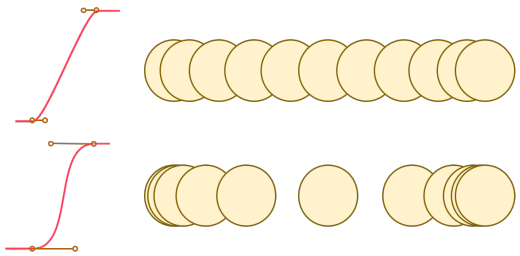


Figure 5.1: On the left, the animation curve managing the position of the ball over time ; on the right, the corresponding animation. *Spacing* is defined as the distance an object covers in one frame, in image space. It can be constant over time (top), or vary (bottom), depending on the derivatives (orange handles on the left).

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 impulses an objective time scaling factor defined as piecewise constant:

$$\lambda_i^{\text{pc}} = \frac{y_{(i+1)^\star} - y_{i^\star}}{x_{(i+1)^\star} - x_{i^\star}}, \tag{5.1}$$

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

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

$$\mathcal{E}_1 = \sum_i \left(\lambda_i^+ - \lambda_i^{\text{pc}}\right)^2 + \left(\lambda_i^- - \lambda_{i-1}^{\text{pc}}\right)^2 . \tag{5.2}$$

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

$$\mathcal{E}_2 = \sum_{i \in \mathcal{S}} \left(\lambda_i^+ - \lambda_i^-\right)^2 \ . \tag{5.3}$$

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

$$\mathcal{E}_3 = \sum_{i \leq N_{\text{peaks}}-1} \left(\lambda_i^+ - \lambda_{i+1}^-\right)^2 \ . \tag{5.4}$$

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 $[\lambda_i^+, \lambda_{i+1}^-]$.

6. Other ideas

During a review session, the supervisor uses intuitive gestures to express the changes to be made to an animation. These gestures can express notions of amplitude, timing, and many others; they can apply to all or part of the animation, whether in time - affecting only a subset of frames - or in space - affecting only certain objects in the scene, or even only certain attributes of these objects. All these possibilities make our task a complex one. In this section, we'll outline some of the ideas we've put forward during the internship to help us answer some of these questions.

6.1 Correlation between gesture and animation

The question of gesture analysis raises, among others, the following two questions:

(1) What data must be extracted from the gesture to understand it? Position, speed, acceleration, other? What about two-handed gestures?

(2) What curves in the scene does the gesture relate to? A whole body, a single limb, in position, in rotation?

The first means being able to extract various time-dependent descriptors $s_1(t), ..., s_n(t)$ from the video of the gesture, such as the position of each hand, their spacing, speed, acceleration, or others, and being able to choose which are relevant. If there are several of these, the question arises of how to merge them to produce a single key signal.

The second involves extracting all the animation curves from the 3D scene, i.e. each attribute - position, rotation, scaling, etc. - of each controller - second phalanx of the right index finger, ankle, left eyebrow - of each character or object. Then, for each of these curves, calculate time-dependent descriptors $r_1(t), ..., r_m(t)$: derivative, second derivative, or others. And finally, compare all this with the gesture to determine which curves need to be modified, and in what way - amplitude, timing, spacing, etc.

The difficulty with this method is that it requires us to compare temporal signals from different spaces - gesture, animation - but now living in the same common descriptor space, in order to determine their relevance.

The basic tool for comparing temporal signals is distance in the mathematical sense: we could calculate, for each pair of signals, the average of their term-to-term difference (see Eq. 6.1). However, such a measure is not very robust to noise. To overcome this, we could interpret our signals as the realization of random variables S_i, R_j, and calculate their correlation coefficient (see Eq. 6.2).

$$dist(s_i, r_j) = \frac{1}{\Delta T} \sum_t |s_i(t) - r_j(t)| \quad (6.1) \qquad \rho(s_i, r_j) = \frac{\mathbb{E}(S_i R_j) - \mathbb{E}(S_i)\mathbb{E}(R_j)}{\sigma_{S_i} \sigma_{R_j}} \quad (6.2)$$

This coefficient varies between -1 and 1 and quantifies the amount of information that a value on one of the signals provides about the value of the other signal, at the same time. 0 means that the two signals are independent.

In our case, we realized after a few experiments that ρ values were difficult to interpret and extremely variable in real-life situations. What's more, given that we wanted to be able to use impulse gestures to edit any animation curve, even if it had nothing to do - apart from the presence of keys at the right moment - with the gesture (i.e. be free of the mime constraint), this line of thinking didn't make much sense. We therefore set this

part aside, to concentrate on matching gestures. In the final proposal, a configuration file specifies which gesture descriptor to associate with which animation curves. The gesture descriptor is systematically the one described in Chapter 3, i.e. the derivative of the mean amplitude of the optical flow (see Equation 3.1).

The literature search also gave rise to the idea of using Cross Correlation Analysis, which, among other things, enables signals to be merged to obtain a target signal, in order to merge several gesture descriptors. However, after a few unsuccessful experiments, this idea too was shelved.

6.2 Dynamic Value Warping

We're interested here in the case where the supervisor's gesture expresses a change in the amplitude of the animation, in addition to (or instead of) the timing change.

Building on our success with Dynamic Time Warping, we came up with the idea of creating a "Dynamic Value Warping" (DVW) module. Instead of ordering the points of two signals in time, and then comparing them in value, we could order the points in value, and compare them in time:

$$C(x, y) = |s_{\text{in}}^{-1}(v_x) - s_{\text{out}}^{-1}(v_y)| \tag{6.3}$$

where v_x if the x-th ordered value of s^{in}, and v_y the y-th ordered value of s^{out}.

This means that the algorithm will strive to create vertical pairs instead of horizontal ones in curve space. This approach, very similar to DTW, creates 2D maps that are in fact very different. Indeed, our signals contain noise, which strongly affects this map by swapping columns or rows, whereas for DTW, it just biased the pixel values.

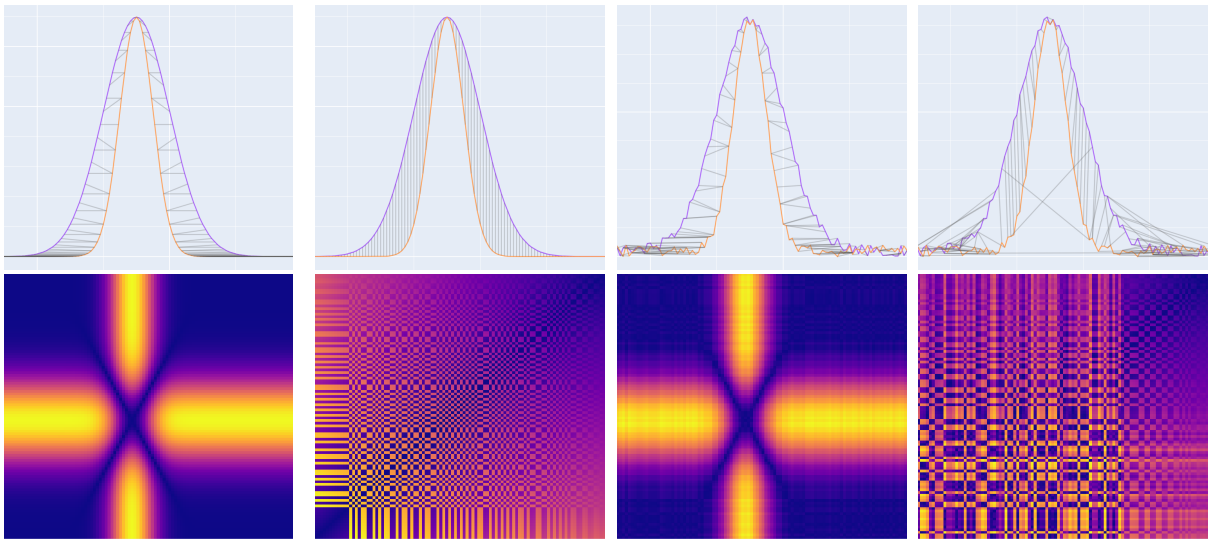


Figure 6.1: Top: optimal correspondence found; bottom: matching 2D map. From left to right: DTW applied to clean gaussian curves, DVW applied to the same curves, DTW applied to noisy gaussian curves, DVW applied to the same curves. As we can see, DTW still manages to find a correct correspondence despite noise, whereas DVW on the opposite gets very affected by noise.

This instability, proven by simple tests on synthetic data (see Figure 6.1), made it impossible to deploy this technique on real-world data.

In our final proposal, another, much simpler method was devised to take account of amplitude changes: the creation of a piece-wise linear temporal scaling function g. At its vertices, which coincide in time with output signal peak time, its value is a ratio between the input and output signals:

$$g(t_{y^*}) = \frac{\sum_{i=-w}^{w} s^{\text{out}}(t_{y^*+w})}{\sum_{i=-w}^{w} s^{\text{in}}(t_{x^*+w})} \tag{6.4}$$

where (t_{x^*}, t_{y^*}) is a peak time correspondence between s^{in} and s^{out} as defined in Chapter 4, and w a user-defined window size.

Unfortunately, time has run out to provide examples of applications of this method.

7. Technical choices

7.1 Collaboration with an industrial partner

As mentioned earlier, this internship is part of a larger project involving an industrial partner, *Dada! Animation* (Dada). Dada is a small Parisian animation studio with a penchant for experimentation, with projects in VR or linked to AI, for example. I had the opportunity to visit their studios and meet their teams, in particular the animator and the technical director. These meetings were an opportunity to make the link between their needs, which were at the root of our project, and our scientific intuitions. My gap year at Gobelins in 3D animation was very useful in bridging the gap between the artistic concepts at the heart of the animator's profession, full of jargon derived from English, and the scientific concepts to which they relate, which animators don't always master.

In addition, my discussions with the technical director highlighted certain technical constraints that I had to respect during the development of my project, so that it could be applied in a production context. One such constraint, for example, is the fact that not all productions use the same software: Blender, Autodesk Maya, Autodesk 3ds Max and so on. It is virtually impossible to switch from one of these programs to another, as they also differ in the way they define rigs[1]. My project therefore needed to clearly redefine all its inputs, so that it could be ported to any of these programs. This redefinition took the form of a "protocol" defining the 3D scene, controller, animation curve and all operations performed on these objects, the development of which was discussed iteratively with Dada's technical team.

7.2 Implementation and architecture

This project was developed entirely in Python, for reasons of prototyping speed and compatibility with 3D animation software. The existence of a Visual Studio Code extension dedicated to Blender scripting, and the fact that Blender is free to use, led us to choose Blender as our 3D software for setting up examples.

As far as video recording was concerned, we quickly realized that recording impulsive gestures - finger snaps, abrupt gestures - required a frame rate higher than the usual 30 fps. We therefore recorded all our videos in 60 fps; for simplicity's sake, the animations were also produced in 60 fps (although studio animation is generally done in 24 or 30 fps).

Finally, the scope of the project, and the number of computing bricks to be created in order to achieve a complete pipeline, made it essential to devise an object-oriented architecture, which I designed and built from scratch. A schematic of this architecture is available in Figure 7.1. Particular attention was also paid to the creation of test files, example scripts and clear documentation. User-defined thresholds and other specifications were separated into dedicated configuration files. Overall, the architecture was built with modularity and maintainability in mind.

[1] rig: program that governs the reaction of the skeleton and skin of a character (or object) to the animation of its controllers.

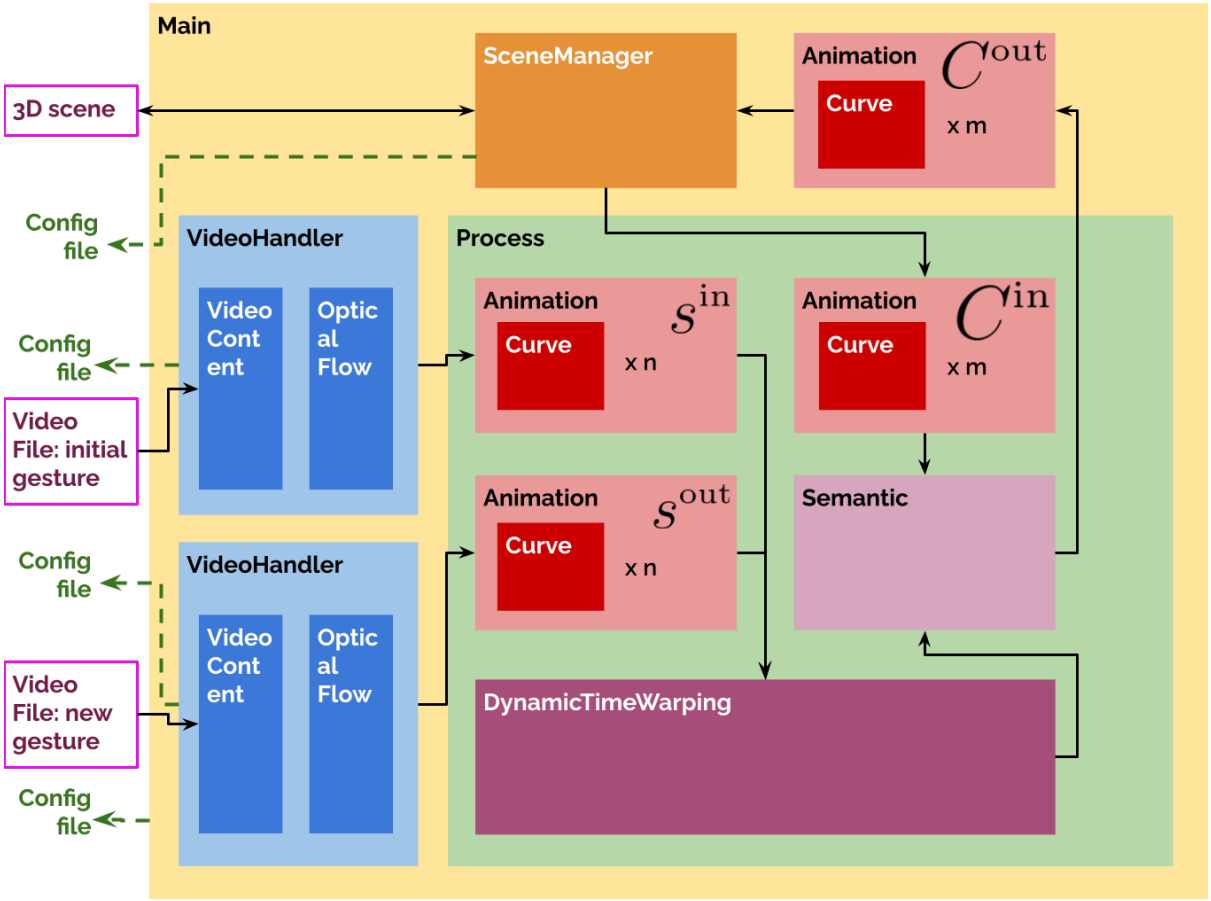


Figure 7.1: Simplified block representation of the project's object-oriented architecture. Solid blocks represents classes, and solid arrows represent the relationship between them. Dotted blocks represent data, and dotted arrows represent the relationship to a dedicated configuration file. The *SceneManager* class handles the live communication with the 3D scene opened in Blender. The *VideoHandler* class handles the loading and processing of the video files from the camera.

8. 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 squatch and stretch effect. The animation curve is composed of the translation of the sphere in time and the time-varying scaling parameter. Fig. 3.2 presents the retiming obtained on the sphere, and the associated curves are depicted in Fig. 8.1. 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.

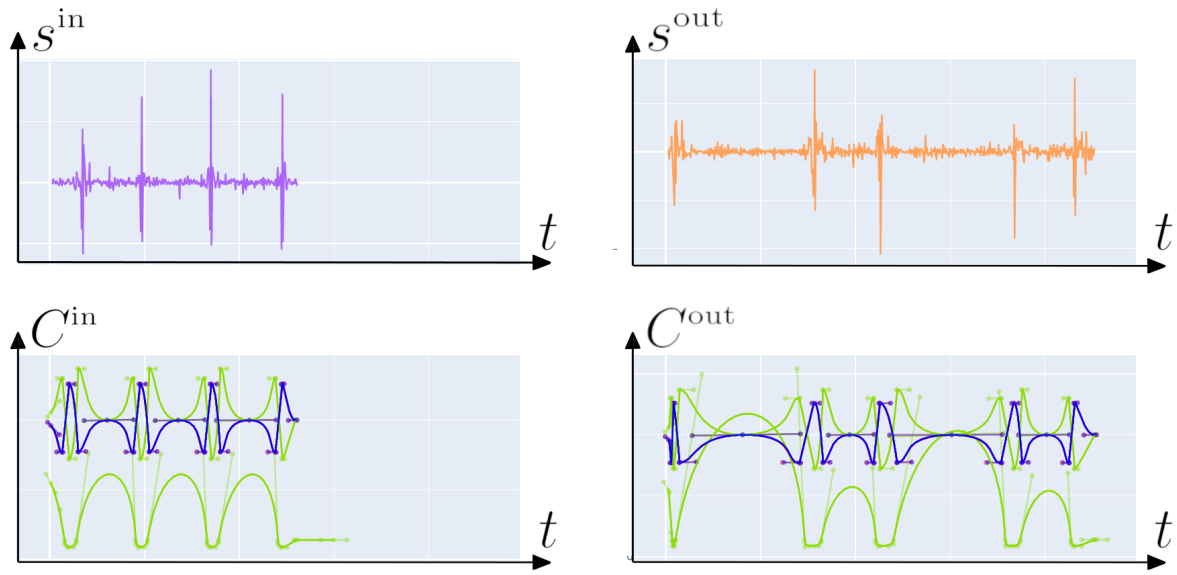


Figure 8.1: Bouncing ball animation retiming curves corresponding to the example in Fig. 3.2. Translation and scaling are shown respectively by the green and blue curves.

The effect of the semantic preservation is illustrated in Fig. 8.2 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 firework animation in Fig. 8.3-left where both

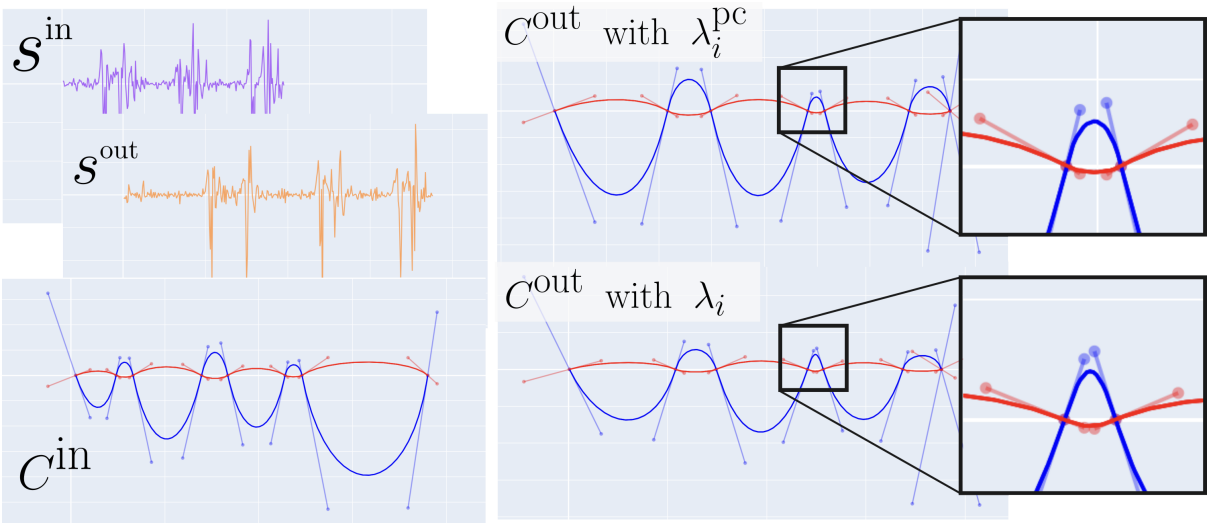


Figure 8.2: Effect of the semantic preserving time scaling applied to a smooth animation curve. The close-up view illustrates how our optimization helps to preserve the sharper curve of the second maxima already present in the initial animation and is associated with a dynamic change of direction.

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.

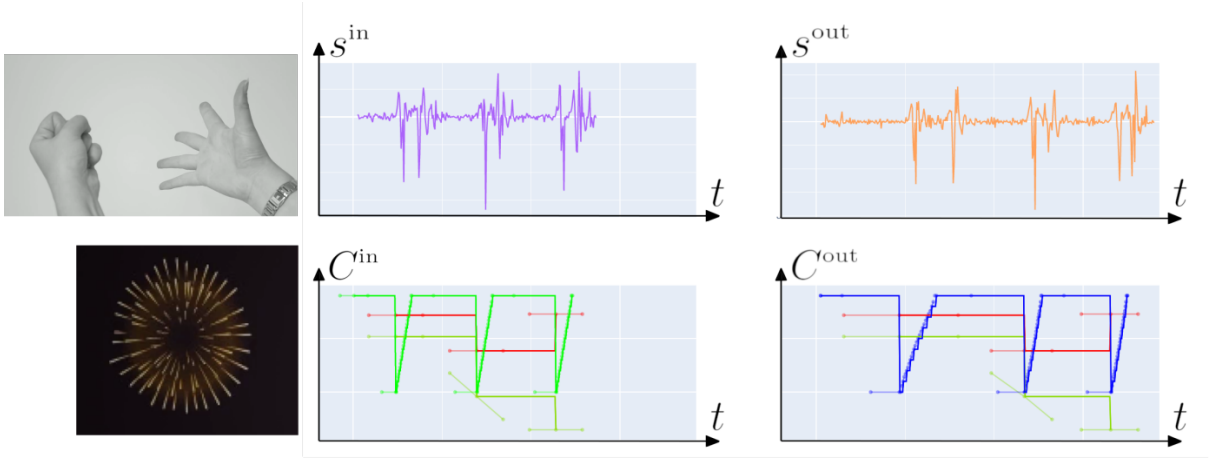


Figure 8.3: Retiming of a firework with two hands opening and closing;

We finally present a more complex application scenario with a walking character in Fig. 8.4-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 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.

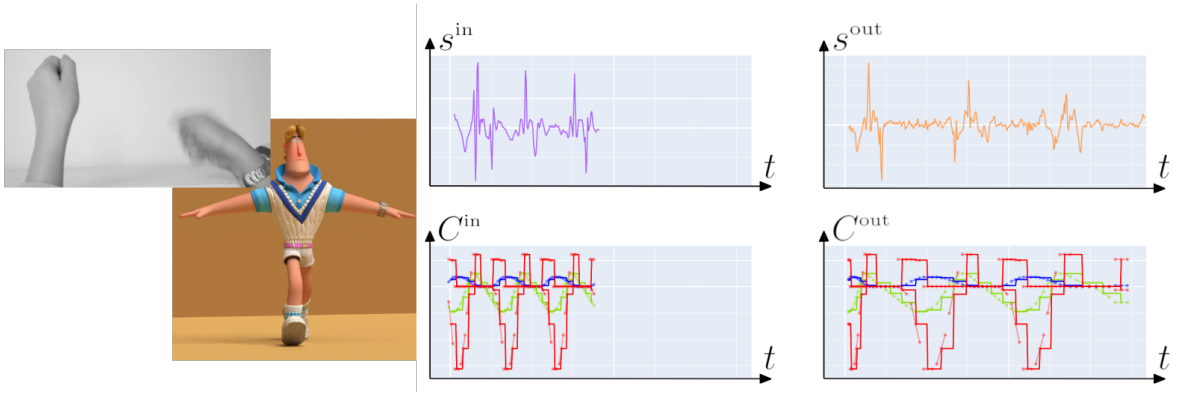


Figure 8.4: Retimed walking cycle with two hands hitting a table. The curves only show the right-hand signal.

We observed experimentally that this method is able to handle a large range of motion without assuming any a-priori on the gesture, granted that it illustrate a set of impulses. The typical failure cases are met when the impulse gestures are too close in time, or when the motion is not fast enough, thus bluring the cost map.

9. Conclusion

9.1 Recapitulative and possible improvements

We described 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 is intended to be explored and validated with real-world CG animations and animators' feedback.

Regarding the possible improvements, 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 signal(s), potentially through the computation of high-level features, to the candidate animation curves (see Chapter 6 for more explanations). 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; this topic will be covered in the forthcoming thesis.

9.2 Personal experience

This end-of-study internship was a decisive step in my academic and professional career, which has been oriented towards the arts and sciences for many years. It was an opportunity to put to good use my skills in images, acquired during this Master's year, as well as my knowledge of the world of 3D and animation, outlined during my internship at BUF[1] Compagnie's R&D department (VFX) two years ago, then extended during my gap year at Gobelins[2] last year. In the future, once I've completed my studies at Télécom Paris and obtained my diploma from Ecole Polytechnique, my ambition is to do a thesis in an animation studio. This would affirm my desire to invest in research for the creative industries, which are full of challenges to be solved and overcome.

Courtesy of @Dada!Animation

[1] https://buf.com/
[2] https://www.gobelins.fr/

Bibliography

[Arora et al.(2019)] 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).

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

[Choi et al.(2016)] 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.

[Ciccone et al.(2019)] 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.

[Dontcheva et al.(2003)] Mira Dontcheva, Gary Yngve, and Zoran Popović. 2003. Layered acting for character animation. In *ACM SIGGRAPH*. 409–416.

[Du et al.(2023)] 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.

[Dvorožňák et al.(2020)] 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.

[Garcia et al.(2019)] 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).

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

[Guay et al.(2015)] 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.

[Jiang et al.(2023)] 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.

[Johnston and Thomas(1981)] Ollie Johnston and Frank Thomas. 1981. *The illusion of life: Disney animation*. Disney Editions New York.

[Kim et al.(2003)] 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.

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

[Lin et al.(2024)] 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.

[Seol et al.(2013)] 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.

[Sharma et al.(2019)] 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.

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

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

[Walther-Franks et al.(2012)] 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.

[Zhan et al.(2023)] 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.

[Zhou et al.(2024a)] 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.

[Zhou et al.(2024b)] 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.

[Zhou et al.(2018)] 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.

Appendices

Expressive Animation Retiming from Impulsed-Based Gestures

Anonymous Author(s)
Submission Id: 3131

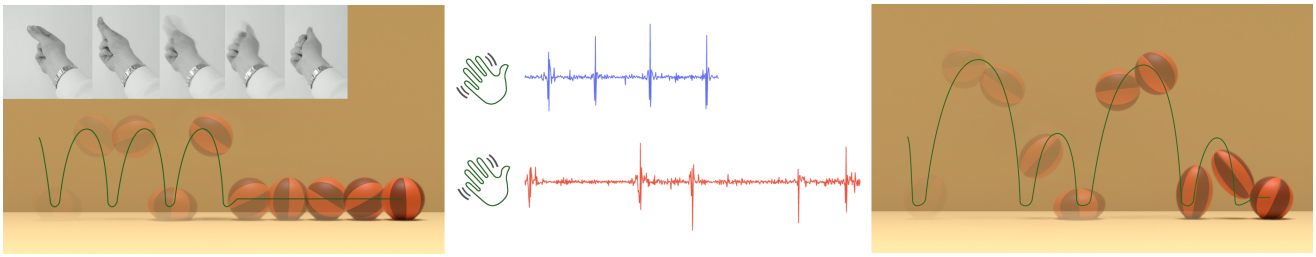


Figure 1: Given an input animation (left), our method can use the notion of timing expressed by impulse-like gestures (middle) to automatically compute a new retimed animation (right). The top gesture in blue is expressed by the user in synchronicity with the input animation, while the down one in red 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:
Anonymous Author(s). 2024. Expressive Animation Retiming from Impulsed-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/XXXXXXX.XXXXXXX

MIG '24, November 21–23, 2024, Arlington, VA

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

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 its 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 animatior 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 its own way to express the modification he wants using its 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 individual 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 himself 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 a 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. These methods, such as *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. Similarly, *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. These methods, such as those discussed in *Creature Features* [Seol et al. 2013], *AgileFingers* [Lin et al. 2024], and *HandAvatar* [Jiang et al. 2023], 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. While the *Performance-based timing* approaches from [Terra and Metoyer 2004, 2007] are close to what we want, contrary to them we focus on retiming with differences between an input and output animation curve, handling different numbers of detected peaks, and for this, unlike them, we choose a more greedy method to explore the possibilities of our approach. Also for *MagicalHands* [Arora et al. 2019], while interesting for its more general mapping from gesture to physical animation, we don't want a capture system, nor VR, just the hands, and managing different types of gestures.

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 TimeWarping.

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 $I^{\text{in}}(t)$ 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 he convey the key-events via impulse-like gestures, i.e. clear change of velocity in its motion during a short amount of time. The second video input $I^{\text{out}}(t)$, $0 < 0 < T^{\text{out}}$ 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 $I^{\text{out}}(t)$.

Our method works in two steps. First, in finding a correspondance between I^{in} and I^{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 $O_p(I^{\text{in}}(t))$ the optical flow of the video I^{in} at pixel p, 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

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

and similarily for $s^{\text{out}}(t)$ with $I^{\text{out}}(t)$. The impulse gestures of the supervisor corresponds to short acceleration, and are therefore corresponding to 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

- $(x,y) \rightarrow (x+1,y)$, 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.
- $(x,y) \rightarrow (x+1,y+1)$, diagonal step, corresponds to advancing time in both the input and output animation. There is no local retiming.
- $(x,y) \rightarrow (x,y+1)$, 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 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 present first why we can consider that such peaks can be defined by a set of indices $i^\star$ that are subset of $\mathcal{I}_P$ in providing a geometrical interpretation of the structure of the 2D cost map, and then describe our formal approach for extracting the relevant peak-times.

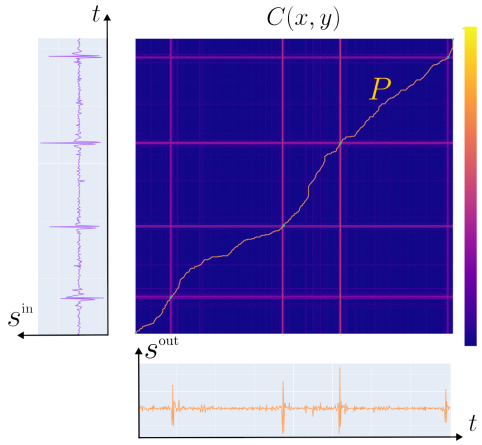


Figure 2: The cost map C computed from the values of s^{in} and s^{out} with high cost values in red/yellow, and the path P minimizing the cost computed from Dynamic Time Warping.

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 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.

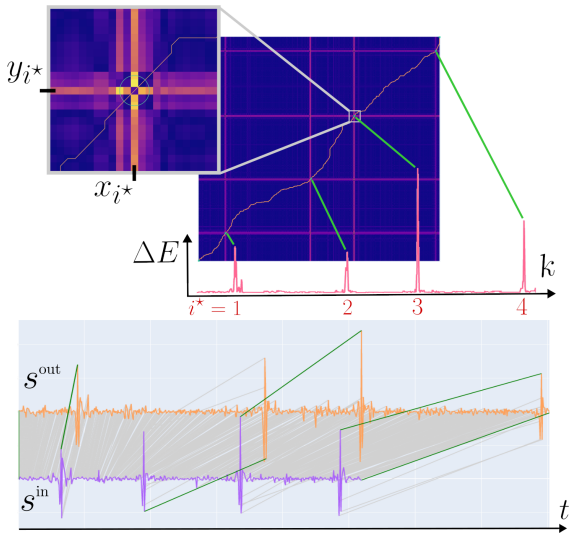


Figure 3: Top: The cost map with a close-up on the peak region defining $(x_i^\star, y_i^\star)$ at the central low-cost value surrounded by high-costs barriers. The red graph illustrates the value of ΔE along the indices of the path P and shows clear peak values. Bottom: The green lines show the final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

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

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

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

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

where $\text{Area}(P_{|x_k\, y_k}\,,\ P)$ 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. In the case where these criteria are true more than once in less than a quarter of second, we keep a single coordinates 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 its 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 that will continue to pass through the neighborhood of the forbidden pixel. Furthermore, the local orientation of the cost barrier can be evaluated via gradient $\nabla C(x_i, y_i)$, such that an horizontal direction indicates that there is more peaks in the input signal than in the output signal, and a 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 detected peaks in s^{in} and the end of the signal, and smoothly blended with the original signal in order 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 $i \in [1, N_{\text{peaks}} - 1]$.

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 $[\lambda_i^+, \lambda_{i+1}^-]$.

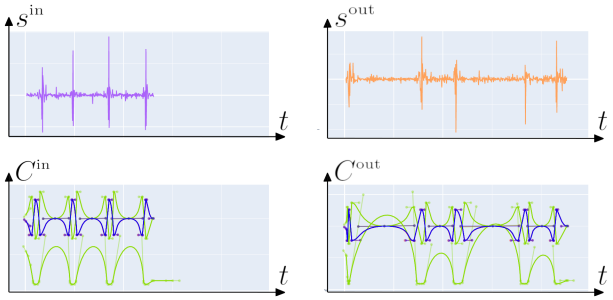


Figure 4: Bouncing ball animation retiming curves corresponding to the example in Fig. 1. Translation and scaling are shown respectively by the green and blue curves.

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 squatch 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

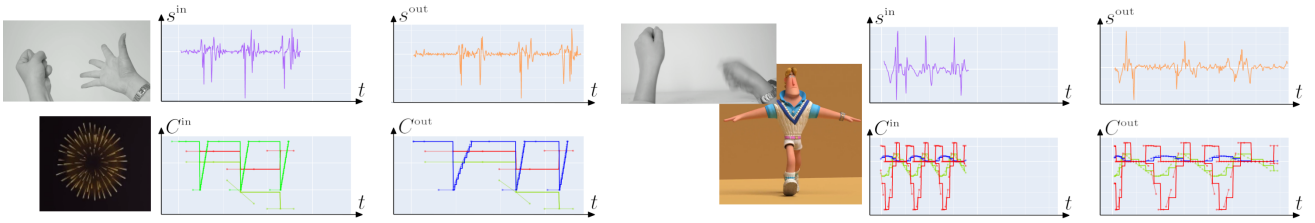


Figure 5: Left: Retiming of a firework with two hands opening and closing; Right: Retimed walking cycle with two hands hitting a table. The curves only show the right-hand signal.

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.

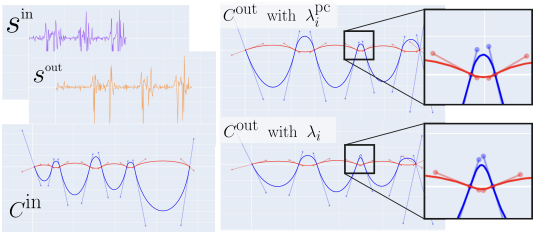


Figure 6: Effect of the semantic preserving time scaling applied to a smooth animation curve. The close-up view illustrates how our optimization helps to preserve the sharper curve of the second maxima already present in the initial animation and is associated with a dynamic change of direction.

Another scenario is illustrated by the firework 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 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.

We observed experimentally that this method is able to handle a large range of motion without assuming any a-priori on the gesture, granted that it illustrate a set of impulses. The typical failure cases are met when the impulse gestures are too close in time, or when the motion is not fast enough, thus bluring the cost map.

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 signal(s), potentially through the computation of high-level features, to the candidate animation curves. 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.

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.

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.

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.

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.

Received 19 July 2024; revised 6 September 2024; accepted 6 September 2024