DRAGIMATION: Direct Manipulation Keyframe Timing for Performance-based Animation

Benjamin Walther-Franks
Universität Bremen, Germany

Marc Herrlich
Universität Bremen, Germany

Thorsten Karrer
RWTH Aachen, Germany

Moritz Wittenhagen
RWTH Aachen, Germany

Roland Schröder-Kroll
Universität Bremen, Germany

Rainer Malaka
Universität Bremen, Germany

Jan Borchers
RWTH Aachen, Germany

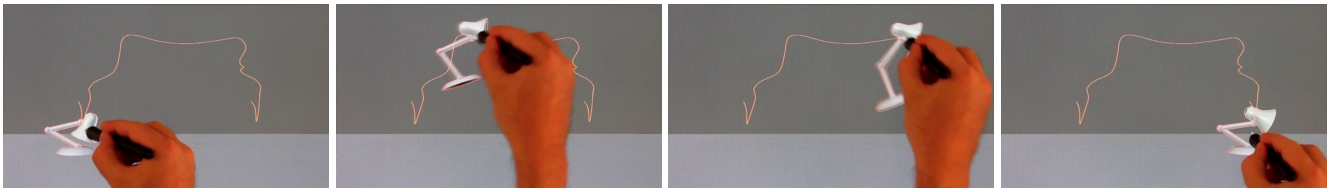

Figure 1: Using *Dragimation* to time a jump animation. The timing of the dragging motion is recorded and applied to the keyframes.

ABSTRACT

Getting the timing and dynamics right is key to creating believable and interesting animations. However, using traditional keyframe animation techniques, timing is a tedious and abstract process. In this paper we present *Dragimation*, a novel technique for interactive performative timing of keyframe animations. It is inspired by direct manipulation techniques for video navigation that leverage the natural sense of timing all of us possess. We conducted a user study with 27 participants including professional animators as well as novices, in which we compared our approach to two other interactive timing techniques, timeline scrubbing and sketch-based timing. Dragimation is comparable regarding objective error measurements to the sketch-based approach and significantly better than scrubbing and is the overall preferred technique by our test users.

Index Terms: H.5.1 [Information Interfaces and Presentation]: Multimedia Information Systems—Animations; H.5.2 [Information Interfaces and Presentation]: User Interfaces—Interaction styles

1 INTRODUCTION

Creating action is at the heart of character animation. Since the early days of animation, action has been shaped through a sequence of key poses with extreme positions of the character. Animators control the speed of the motion by determining how many frames are between these key poses. Fewer frames result in faster transitions between key poses, more frames in slower transitions.

In classical animation, creating the intermediate frames, or *in-betweening*, used to be the work of the animator. In computer animation, in-betweening happens automatically. While this method gives full temporal and spatial control over the animation, it requires the animator to think in abstract units, usually frames, to define the timing for a motion. This abstract notion of dynamics does not come naturally, and animators spend a long time perfecting this art. At the same time, humans do have a good intuitive grasp of timing—anybody could quickly indicate a motion they have in mind with gestures, or by voicing a rhythm.

Graphics Interface Conference 2012
28-30 May, Toronto, Ontario, Canada
Copyright held by authors. Permission granted to CHCCS/SCDHM to publish in print form, and ACM to publish electronically.

Realtime animation methods set about this goal differently, recording real world movement to define virtual motion and timing. Motion capture enables actors to define motion and dynamics of digital characters via body expression, facial expression or gestures. Computer puppetry, or performance animation, is a form of motion capture that emphasizes instant visual feedback [22], supporting actors by letting them adjust their performance based on the feedback. Next to keyframe animation and physical simulation, motion capture has become established as means of animation in the film, television, and computer game industries. Yet the benefits of capturing expressive and spontaneous acting come at the cost of a lack of precise control. Furthermore, motion capture constitutes a workflow completely different to keyframe animation, and capture data is not easily integrated into a keyframe animation process.

Research has proposed to combine the best of both worlds by applying realtime and non-realtime approaches to different parts of the animation process. Performance-based timing—also called performance timing—is an approach in which the animator first sets key poses, then acts out the timing for these poses in realtime. This can be done by drawing the trajectory of the target motion at a certain pace with a stylus and applying the sketch timing to the scene [24], or by scrubbing the timeline [20]. Performance timing techniques thus combine the precision of keyframe-based spatial control with the natural sense of timing of the animator.

In this paper, we establish *Dragimation* as a new direct manipulation technique to control timing in performance-based animation. Dragimation takes the idea of dragging a feature "through time", and develops it as a new way to act out the timing of key poses. With this, we adapt a solution for a problem in the domain of video navigation to solve a different problem in the context of animation timing. We claim that the directness of Dragimation improves the quality of timing and user satisfaction in performance-based animation timing tasks. We substantiate this claim by comparing Dragimation to drawing timings by *Sketching* and to the timeline-based *Scrubbing* technique of prior works in a user study. Both objective measurements and subjective user rankings are highly in favor of Dragimation for performance timing. It is equal to sketching in producing timings, and both are more suited to the task than scrubbing. Dragimation is also the clear winner of a subjective assessment regarding ease of use, mental load, and overall user preference.

2 Dragimation: Performance Timing by Dragging

Timing can be defined as modifying a piece of spatio-temporal data in a way that, while its spatial component stays fixed, its temporal component can be adjusted freely. This generalized problem is not exclusive to animation; it poses itself in the same manner for other fields such as video editing or video navigation. For the video-related problem a number of solutions have been proposed, some of which are applicable to the animation case as well. Direct manipulation video navigation (DMVN) systems [4, 7, 9, 10, 21] in particular allow users to move through the video with arbitrary pacing while being guided by the spatial motion of one or more objects in the scene. The DMVN concept of spatially guided interactive time control can also be used for defining a timing. It can specifically solve problems of current timeline-based [20] and sketch-based [24, 23] performance timing approaches.

The timeline-based approach [20] makes use of the well-known one-dimensional visualization of time as a bar. The reconfiguration of motion timing created by moving the playhead across this bar is recorded and determines a new timing. This horizontal *Scrubbing* input motion usually has no relation to the object motion, making the timing task cognitively challenging.

With the *Sketching* technique [24, 23], the user draws a curve at an arbitrary position of the main view with the desired pacing. This curve may or may not resemble the motion path of the object. During this procedure, feedback is limited to the curve appearing under the pen; the scene to be animated remains static. When the pen is lifted, a matching algorithm tries to map the curve to the motion path based on local features. This mapping does often not reflect the intention of the animator, especially when the drawn curve's shape differs from that of the motion path, which is a known limitation of the technique. It has then to be adjusted manually by adding and removing feature correspondences. A command applies the timing of the drawn curve to the object. Only then, the new timing is reflected in the scene. This is the first instance of content feedback the animator gets. The manual matching pass required for the sketch-based technique creates a significant break in the interaction, adding a layer of cognitive indirection.

We bring the concept of DMVN to the context of performance-based animation timing with a technique we call *Dragimation*. With Dragimation, the animator drags the object to be animated along its motion path through the scene. This is a direct manipulation interaction; feedback is immediately given, allowing closed-loop performance adjustments. After lifting the pen, the animation is already fully re-timed. This approach has two advantages: Firstly, it reduces the global matching problem between two representations of a path—the one carefully laid out in the animation and the one drawn by the user for re-timing—to a local problem in which at any point during the process an input position simply has to be projected onto the animation path of the object. This local problem has been studied in the area of DMVNs and can be solved in a variety of ways. Such mappings require a close coupling of input motion to object motion, strengthening the directness of the interaction. Secondly, the resulting animation is already visible even during the interactive re-timing by the motion of the object itself, as with performance timing via a timeline. This removes a layer of cognitive indirection that especially novices have difficulties with.

3 Related Work

Research has been exploring alternative animation paradigms for quite some time. Computer puppetry, often also referred to as performance animation, combines motion capture with instant visual feedback [22]. It endeavors to bring spontaneity and natural expressiveness back to computer animation. The motivation often stated in this context is to make the process of bringing virtual scenes to life more accessible, especially to novices. While this can involve virtual reality setups [1] or custom-built animatronics puppets [5], motion capture setups can also be achieved with smaller optically tracked widgets [3] or accelerometers [18]. Even less demanding in terms of hardware are 2D motion capture setups using continuous input from the mouse [13, 17] or multi-touch devices [16, 11, 27]. Discrete control actions can also be recorded in realtime to interactively drive an animation by triggering actions, much like controlling a video game [13, 26, 28]. Also aimed at novice animators are sketch-based animation interfaces that combine realtime with non-realtime manipulation [2] or map input sketch gestures to atomic motion sequences from a motion library [25].

Usually performance animation aims at a high correlation between input space and animation space, since this is the most natural and easily graspable mapping. Confining motion to interpolations between sets of spatial keyframes can aid the capture process by focussing on predefined spatial arrangements[8]. A generalization is to map any animation parameter to the axes of an input modality such as a pen tablet, and let animators control the animation by moving in this space over time [12]. However, this comes at the cost of abstraction that must be learned. Performance animation techniques can involve physical models to drive the motion of secondary (not manipulated) features [19] or to form the basis of an input-output mapping [13, 28].

Creating motion and timing it can be seen as two distinct tasks. Performance timing applies realtime control only to the timing of an animation. It requires the actual motion to have been defined previously, either with realtime or non-realtime tools. The manner in which the acted timing is captured and applied depends on the technique. The literature proposes two realtime techniques for timing animations. With the timeline technique, the animator interactively records a timing by scrubbing along the timeline [20]. The propagation through time created by input movement is recorded in realtime and applied as the new timing, turning the timeline from a browsing into a recording tool. The strength of this technique is interactive feedback: when the playhead is moved to a certain time, the view updates to the spatial configuration at that point in time, putting it in line with computer puppetry techniques. However, there is no spatial relation between the linear control and the animated motion. A sketch-based approach lets the animator give a feature's timing by mimicking its motion via sketching [24]. The sketched path is then matched to the original trajectory. The animation is timed so that the timing matches the speed at which the pen moved while sketching—slowly sketched parts of the path create a slow timing, faster pen movements a faster timing. The strength of this technique is its support for close spatial correspondence of input to output, giving the actor a sense of the whole motion, rather than just propagation through time. However, the automatic matching between sketched path and feature trajectory is often not as the animator desired, requiring manual matching of path features. After a successful match the animator gets feedback on the result. Dragimation combines the strengths of both techniques into one, using direct manipulation and interactive feedback to give the animator a sense of space and time. The local, realtime input-output mapping eschews any extra post-hoc matching.

While they are not the focus of this work, non-realtime (re)timing methods provide a more parametric approach, e.g., by employing physical models to re-calculate realistic motions given timing parameters [15].

Direct manipulation of objects in a scene has been studied in the context of navigating through a video scene. These direct manipulation video navigation systems (DMVNs) focus on how to neglect the time domain when interacting with a scene that is already timed. Usually, these systems show the motion path taken by an object, and provide a way to navigate along this path by dragging the object to the desired point. Trailblazing [10] employs direct manipulation in a video surveillance setting to interact with objects in the video or on a floor plan. DRAGON [9] and DimP [4] propose a more gen-

eral use of direct manipulation browsing for video analysis, e.g., of sport videos. Goldman et al. showed the feasibility of this approach for a variety of video editing tasks [7]. Recently, Shar and Narayanan even used direct manipulation for video editing, retiming a segmented part of the video against a still background or the rest of the scene [21]. Dragimation brings this principle of direct manipulation time control to the computer animation domain, as we will describe next.

4 Implementation

In the following we describe Sketching, Scrubbing and Dragimation in detail before explaining the retiming step common to all three techniques. All three techniques assume an already created motion, which can be arbitrarily timed. We employ the terminology of McCann et al., who define the (re)timing process as a mapping from the result to the original: output motion (the final animation) is created from the input motion (defined by the initial set of key poses) by mapping the playback time (the user's time or realtime) to the source time (defined by the original timing) [15].

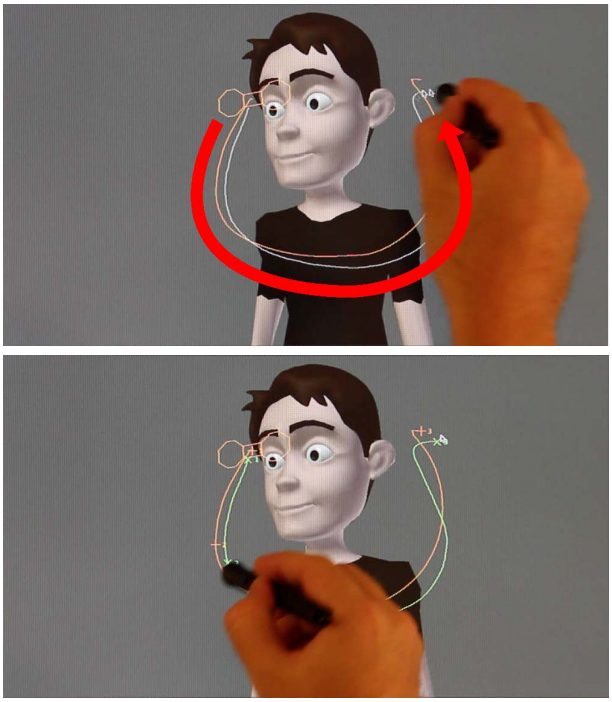

Figure 2: The two stages of Sketching the timing: 1) draw path, 2) check path matching.

4.1 Sketching

Sketching implements a sketch-based technique for performance-based keyframe timing [24]. It builds on a static representation of a feature's motion, its motion path. This can be generated by sampling the feature location at regular intervals of the animation. To retime, the user mimics the motion path of the feature to be retimed by sketching a similar path with a pen on a tablet input device. Motion path and sketched path are then matched (semi-)automatically and the timing of sketch path samples determines how initial keyframes are retimed.

The sketched path is recorded as a list of triples (x, y, t) with two spatial and one temporal dimension. The system then determines salient points on both curves by finding local minima and maxima in both spatial dimensions. Thus detected minima and maxima are filtered further by a threshold. These salient points divide both curves into segments, and are used to match the two curves: for every keyframe, the normalized arc length position along a curve segment of the original motion curve is used to calculate the recorded playback time of the corresponding point on the sketched curve. If the algorithm does not find a good match between salient points on both curves, the salient points can be edited manually.

We added improvements in part suggested by Terra & Metoyer's user study [23]. To improve the manual matching process, the source curve changes color depending on the success of the matching: if the number of salient points matches, the curve turns green, otherwise red. We also numbered the salient points to make it easier to determine where on the path they lie.

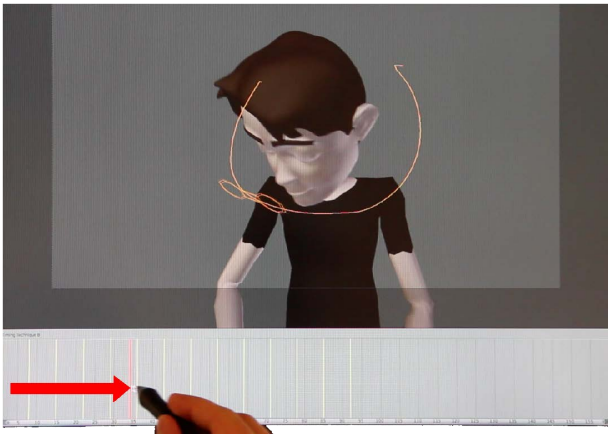

Figure 3: Scrubbing uses the timeline as a recording device.

4.2 Scrubbing

Scrubbing implements the timeline-based technique originally presented as a plugin to the 3D creative suite Maya [20]: The user slides or scrubs along the timeline with a cursor to define the new timing.

A linear spatial mapping relates cursor position to source time. The view is updated when a new source time is visited, giving interactive feedback, and the playback (real) time that has passed since scrubbing was initiated is saved with the current source time. Applying the recorded time mapping retimes the frame range scrubbed. Keyframes after the retimed range are shifted accordingly, while keyframes before are left untouched (see section 4.4).

We made three modifications and extensions beyond the original technique: First, since we assume that the sequence of key poses in time should remain the same, we constrain movement of the playhead to only forward in time, i.e., a movement from left to right, on the timeline. This means that the function mapping playback time to source time can never have a negative derivative. Second, since we want to be independent of the original time resolution, we allow sub-frame scrubbing. Thus, resolution is limited only by input device resolution and the dimensions of the timeline and frame range it depicts, most of which can be adjusted to meet the resolution desired. Third, contrary to the original implementation, we also correct timing curve tangency, as described at the end of this section.

4.3 Dragimation

Dragimation was developed based on direct manipulation video navigation (DMVN). It combines the spatial relation of user input to the resulting motion with the interactive feedback of direct manipulation interaction. Like Sketching, it makes use of a spatial

representation of a feature's motion over time, its motion path (see section 4.1). In order to retime an animation, the animator picks a feature and drags along its motion path through time. The animation is updated according to the current feature position, giving the animator an immediate feedback on his actions. When releasing the drag, the animation is retimed to represent the timing that the animator has just acted out by dragging.

We assume that the user has set a view so that the trajectory is maximally parallel to the view plane, i.e., with no segments of the path at a large angle to the view plane. This needs to be guaranteed to assure a near-constant 2D input to 3D motion ratio. As with Scrubbing, we also assume that the correct sequence of key poses is already set, so source time only moves forward, which is the same as stating that the arc length position of the feature along the trajectory always increases. We also require movement along the curve independent of the original timing. This means that the smallest unit of the motion path should not be determined by frames, but by the spatial resolution of the input. To ensure this, we generate the motion path by sampling the object over time at regular spatial rather than temporal intervals.

The simplest metric for determining the current position along the curve is to find a point on the curve with the shortest distance to the input cursor. However, with complex paths or fast movements this can result in unwanted jumps along the curve. For timing it is critical that an animator can create fluid movements along the curve. We thus add Dragicevic et al.'s *arc length continuity* [4] that extends the distance metric to include the change in arc length as a third dimension, reproduced here for convenience:

$$D = \sqrt{(p_x - C_x)^2 + (p_y - C_y)^2 + (k \cdot \overline{C_a C})^2} \qquad (1)$$

where p_x and p_y are the coordinates of the pointer on the screen, C_x and C_y are the coordinates of any point C on the curve projection, and $\overline{C_a C}$ is the arc-length distance between the currently active point C_a and C on the projected curve [4]. The scalar $k \geq 0$ allows to weight the arc-length continuity component. While [4] recommends a value of $k \approx 1$ for good results in video navigation, this produced too much "slur" or lag for the performance timing task, especially with fast animations. $k \approx 0.5$ yielded a good tradeoff between a highly interactive response and smoothly following the path.

We only allow travel along the curve in one direction by searching the segment of the curve defined by the currently active point C_a on the curve and the far end of the curve. This direction constraint guarantees *directional continuity*, so that we do not need to consider it in the metric. An important requirement for direct manipulation techniques is that they are responsive. Dragicevic et al. suggest optimizations to ensure interactive rates. However, we did not find this necessary for our implementation, as interactivity was possible even with longer motion paths. In order to create a true dragging action, we only start time traversal when the cursor is in proximity of the feature, rather than enabling a user to click anywhere on the curve. Recording the timing works similar to Scrubbing, with the source time being determined for every cursor movement and the tuple (source time, playback time) saved for the retiming process described below.

4.4 Applying the Recorded Timing

Scrubbing and Dragimation record a relation between playback time and source time that we store in a list of tuples (source time, playback time) we call a *time map*. For the Sketching tool, we create a time map by matching the paths based on salient features using the procedure proposed by Terra & Metoyer [24]. Thus, unifying the time mapping allows us to streamline the retiming process, making it the same for all three techniques.

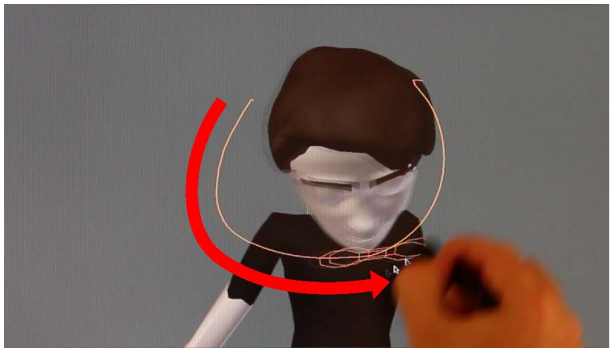

Figure 4: Timing a head turn with Dragimation.

Since the time map initially only covers the frame range visited during interactive timing and the recorded playback (or real) time is not yet aligned with the animation time, we perform some post-processing on the time map to make it ready for look-up:

- We align the two time domains by offsetting the playback time component of each tuple by the source time of the first recorded tuple.
- We cover frames before the retimed range by adding a tuple (t_0^{src}, t_0^{src}) to the beginning of the list, where t_0^{src} is the first frame of the source time.
- We cover frames after the retimed range by appending a tuple (t_1^{src}, t_1^{pb}) to the end of the list, where t_1^{src} is the last frame of the source time range and t_1^{pb} is the last frame of the playback time range (the same value adjusted by the temporal contraction or dilation of the retimed time range).

To retime the keyframes, we look up the source time mapped to each keyframe's playback time and set it as the keyframe's new time. If there is no tuple in the list with a keyframe's playback time, we interpolate linearly between the nearest value before and the nearest after the frame. Since we constrain movement to only forward in time, we never record more than one playback time for a source time (the derivative of the mapping function is never negative). In order to maintain the spatial configuration of the motion, the last step is to scale the keyframe bezier handles along the time axis based on the retimed keys [24].

5 Evaluation

To evaluate Dragimation, we conducted a study comparing it to the established Scrubbing and Sketching tools for timing animations. For each technique, we measured objective quality of timings generated and the subjective satisfaction of participants. Our study design is based on the setup with which Terra & Metoyer compared their sketch-based performance timing to keyframe animation [23]. We constructed a setup with three realtime tools instead of their one realtime tool vs. one non-realtime tool setup, since we wanted to evaluate all three performance timing techniques. Although we could have used keyframing as a baseline, we omitted comparing performance-based timing to non-performance timing anew since this has already been explored in the literature and we wanted to keep session times within a feasible scope.

The entire workflow in which performance timing would be used has three steps:

1. Create a set of key poses with the standard tools. The timing of these key poses need not be determined yet, placing them in a sequence over time with arbitrary spacing suffices.

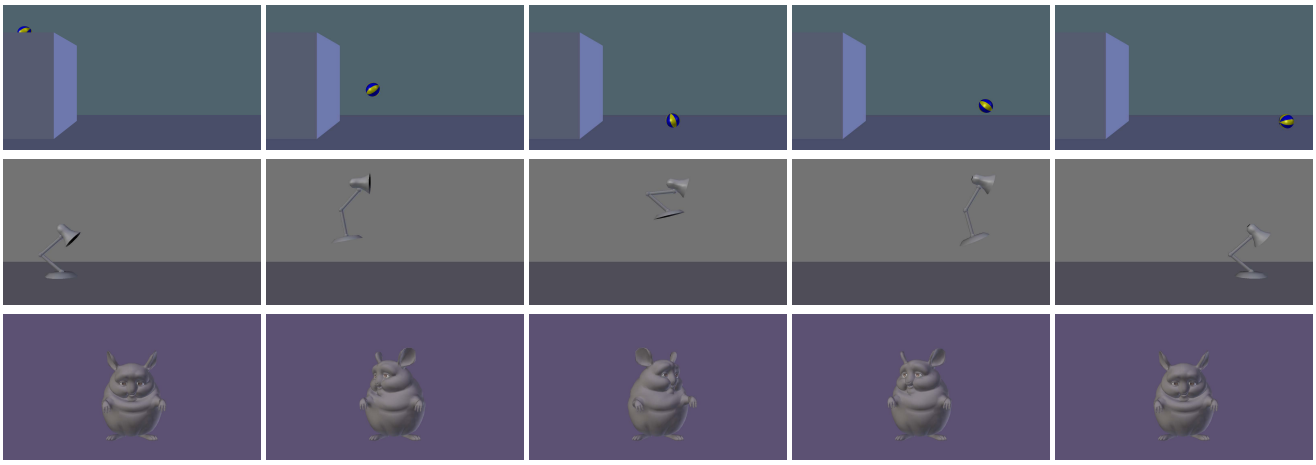

Figure 5: The three scenes used for the timing task in our user study. Bouncing ball, jumping lamp character, chinchilla character looking from side to side.

2. Perform the timing using one of the three realtime techniques. The recorded timing is applied to the temporal layout of the keyframes. Repeat this step until the desired timing is reached.

3. Further adjust key poses, key times and interpolation with the standard tools if necessary.

Since we are investigating only the actual realtime tools, we assumed the keyframe layout to be given, skipping step 1, and omitted the post-editing process, leaving out step 3. The study thus focused on the realtime part of the performance timing workflow, step 2.

5.1 Task

In order to have a comparable measure of how well a tool can be used for timing, we follow Terra & Metoyer's design of timing to a reference. For a set of sample animations, we defined a target timing that subjects were asked to imitate as good as possible.

We selected four sample scenes for the timing tasks, one for the tutorial and three for the main study. The criteria considered were speed of the animation, spatial complexity of the motion, and overall animation length. Further, animations were selected to be representative of typical animation tasks. We also wanted to vary the type of character or object central to the scene—humanoid, object, or anthropomorphic object. The three scenes were a spatially complex bouncing ball animation of medium length and speed, a lamp character in a fairly complex short jumping animation with anticipation and follow-through, and an animal character slowly looking from side to side (Fig. 5). The tutorial scene featured a jumping humanoid character and was spatially less complex and of shorter length. For the study user interface, the initial keyframes of each animation were distributed over time at equal spacing.

5.2 Design

The study used a within-subjects design, with each participant testing all three techniques on the same scenes. First, the experimenter explained and demonstrated the use of each timing tool. Participants were then given sufficient time to explore each technique with a tutorial task until they felt comfortable using it. The main part of the study was done in three blocks, testing all three techniques on each scene. Since the tasks were quite different in nature for each scene, we judged the learning effect between scenes as negligible. We thus kept the presentation order of the scenes constant, while the order of techniques was counter-balanced based on a Latin square. For each tool, participants had 10 trials to approximate the goal timing as closely as possible, resulting in 90 trials per user. A trial consisted of using the tool once and viewing the resulting timing. After 10 trials with one tool, the system automatically switched to the next tool, until all three techniques had been used for the current scene. We recorded the resulting animation and duration for each trial.

After each scene, participants were asked to rank the three techniques according to *precision* ("With which technique did you feel you achieved the most exact result, closest to the reference timing?"), *ease of use* ("How cumbersome did it seem to you to create a timing with each technique?"), and *mental load* ("During the task, how often did you have to consciously remember how a technique works?"), plus an *overall ranking* on which tool they most preferred for this scene. They were asked to rank the techniques based only on usage with the scene they had just retimed. After all three scenes, participants were asked to comment on the learnability of each technique using the learnability sub-scale of the System Usability Scale questionnaire [14]. This was followed up by an interview in which subjects were asked to remark on any positive or negative impressions, and to give a comparison to other timing techniques they were familiar with, if any. Finally, participants filled in a form on demographics and prior experience.

5.3 Apparatus and Interface

Participants sat at a table, using an interactive pen display (Wacom Cintiq UX). Interactive displays were found to be the optimal device for performance timing in previous work [23], and it is a reasonable assumption that animators have such a device at their disposal. Before the experiment, they were asked to adjust angle and position of the Cintiq as well as the height of the chair to a comfortable position. A reference monitor playing a rendered video with the goal timing of the current scene in an endless loop was placed next to the interactive display. The user interface of the Cintiq consisted of a 3D view of the scene with the feature to be retimed and its motion path highlighted (Fig. 6). For the Scrubbing technique, a timeline was displayed beneath the 3D view, for Sketching and Dragimation this was left blank. For the Sketching technique, the bar below the 3D view featured a large button labeled "Apply". A complete trial using Sketching consisted of performing the timing once, editing the resulting curve if necessary, and manually issuing the Apply command. For Scrubbing and Dragimation, a complete trial consisted of performing the timing once, which was then au-

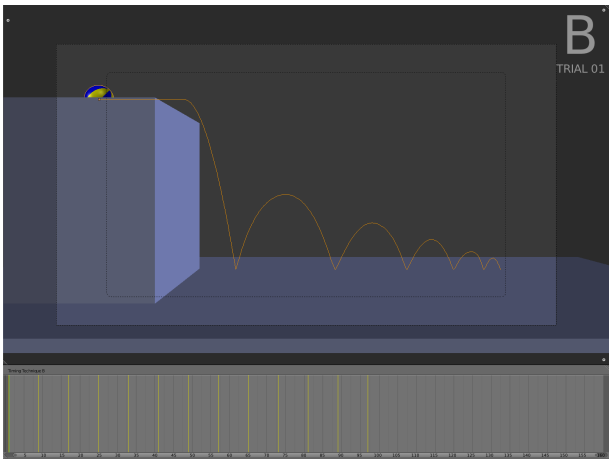

Figure 6: The interface used in our study. The motion path is displayed as an orange curve. The timeline was only displayed for the scrub technique, for the other two it was left blank. Current technique and trial were displayed in the top right corner. Techniques were coded with the letters A (Sketching), B (Scrubbing) and C (Dragimation).

tomatically applied on lifting the pen from the screen. After the timing was applied, the resulting animation was played back in an endless loop for review. A tap with the pen ended the playback and reset the scene and timing to the initial setup. An alphanumeric display in the top right corner of the 3D view displayed the current tool and trial.

5.4 Hypothesis

We claimed that the directness of the interaction and the closed-loop feedback that Dragimation provides has a positive impact on both the objective performance of the animator and their subjective preference compared to Sketching and Scrubbing.

5.5 Participants

27 subjects aged 22–43 (average=28.4) participated in our study, 7 of which were female. All were right-handed. 11 were experienced in computer animation (more than 3 years of experience), 6 considered themselves intermediate (between 0.5 and 3 years of experience) and 10 were novices to animation (less than 0.5 years of experience).

5.6 Results

5.6.1 Precision and Time

To determine quality of timing, we employ the measure proposed by Terra & Metoyer: Per trial, we calculate the offset between target and achieved time for each keyframe. Since the three animations have a different number of keyframes, we calculate the average error over all keyframes for each trial, giving us a single value for proximity to the target timing. In order to account for a learning effect, we pick the trial with the lowest error from each set of 10 trials per scene per technique. After cleaning the data from outliers (video data corroborates that one subject did not follow the task of timing to a target but rather created a completely different timing, largely ignoring the reference video), the mean errors in frames are 3.85 (SD 3.51) for Sketching, 4.09 (SD 2.30) for Scrubbing and 3.58 (SD 2.03) for Dragimation (Fig. 7, note that the error bars are not an indicator for statistical significance). A Friedman test shows a significant difference ($p = 0.007$) for technique. Pairwise Wilcoxon tests show Sketching ($p = 0.017$) and Dragimation

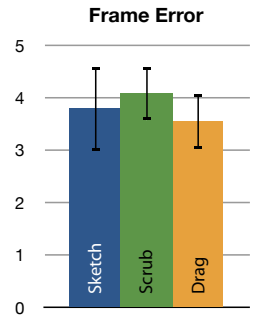

Figure 7: Comparison of overall frame error means. Error bars indicate 95% confidence interval.

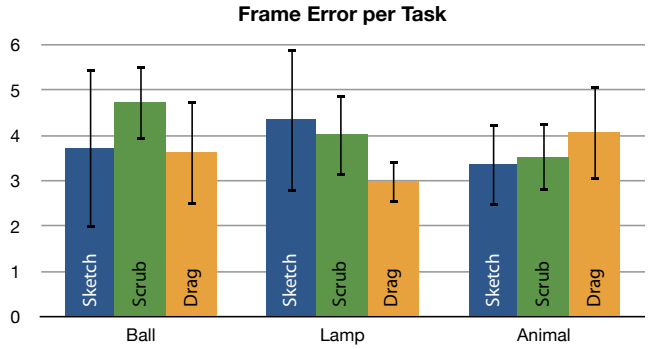

Figure 8: Comparison of per task frame error means. Error bars indicate 95% confidence interval.

($p = 0.033$) to have significantly lower error than Scrubbing. A Mann-Whitney test comparing expert and novices did not reveal any significant effect.

With an average of 12.9 (SD 4.4) and 14.2 (SD 4.9) seconds per trial respectively, Scrubbing and Dragimation were approximately equally fast to use, while the timing process with Sketching took roughly twice as long with 27.4 (SD 7.6) seconds on average. This can be attributed to the fact that the sketch-based technique is not fully realtime, due to its manual feature editing. In the scope of the whole animation process, this time difference is negligible, and we will thus not base any further distinction between the techniques on task completion time. However, it is highly likely that these values are far below the time required to fulfill the task with standard keyframe placement.

5.6.2 User Observation

Video footage from the experiment illuminates issues with the setup and individual techniques. The task of timing to a video presented at a nearby monitor seemed to create an artificial situation that obstructed the interactive feedback of Scrubbing and Dragimation. In many cases, subjects watched the reference video while performing the timing on the interactive display. While this worked for Sketching, neglecting the visualization impaired using Scrubbing and Dragimation, since these rely on their interactive feedback. This is an artificial condition as animators will usually not time to a reference, but create a desired timing they have in mind, which was the price we had to pay in order to have a good quantitative measure for timing precision. We summarize the issues observed for each technique below.

With *Sketching*, the path matching algorithm often resulted in the

need to manually edit feature points. This was a task many participants had difficulties with, since the correct total number and relative position along paths is essential for an optimal mapping. Some participants adopted strategies of sketching the path at a smaller scale away from the target curve, and a few did not mimic the trajectory at all but created an abstract path that they matched nearly entirely manually.

When *Scrubbing* the timing, subjects were nearly always confused by the mapping between timeline and feature motion. There was a further problem with interactive feedback as some subjects looked at the timeline when performing the scrub, rather than watching the viewport.

An issue occurring with *Dragimation* was that hand and pen occluded the viewport, a problem inherent to direct manipulation techniques on interactive screens. Furthermore, while the algorithm locally matching input to path seems to be well suited for the smooth arc trajectory of Task 2, it could run into problems when confronted with the sharp cusps of the motion path in Task 3. While dragging over a sharp cusp, the feature could "get stuck" when the target path motion was only followed lazily or short-cut. It then jumped along the path in unwanted jerks, potentially distorting the desired timing.

5.6.3 Subjective Assessment

The rankings of the three techniques that participants gave provide results clearly in favor of Dragimation (Fig. 9). A Friedman test reveals a highly significant ($p < 0.001$) effect for technique for the qualities precision, ease of use, mental load and overall preference. Pairwise Wilcoxon tests show Dragimation to be ranked significantly higher than Scrubbing regarding precision ($p < 0.001$), ease of use ($p = 0.009$), mental load ($p = 0.001$) and overall preference ($p < 0.001$). They also show Dragimation to be ranked significantly higher than Sketching regarding ease of use ($p < 0.001$), mental load ($p = 0.003$) and overall preference ($p = 0.003$). Again, a Mann-Whitney test comparing experts and novices did not reveal any significant effect for group across all four qualities, thus experts did not significantly diverge from novices in their assessment.

The learnability scores achieved (on a scale from 0 to 20) were 12.6 (SD 5.7) for Sketching, 14.8 (SD 5.0) for Scrubbing, and 17.1 (SD 3.45) for Dragimation (Fig. 10), with a highly significant effect for technique (Friedman test, $p < 0.001$). Pairwise Wilcoxon tests showed Scrubbing to score significantly higher than Sketching ($p = 0.005$) and Dragimation to score significantly higher than Scrubbing ($p = 0.024$).

In the interviews, participants almost equivocally judged all three techniques as very intuitive, easy, and quick to use. While most did not enjoy the manual editing often necessary with Sketching, some appreciated it as a means to control the performance mapping. Subjects complained about the non-spatial mapping of Scrubbing, which made it difficult to judge which input would lead to which timing. Dragimation was often cited to be the most intuitive tool. Many participants stated they could imagine each technique to have its use for certain application scenarios, although there was no consensus on which was best for what type of task. When asked for a comparison with keyframing tools, the performance timing approach as such was judged to be less precise than keyframe animation, but more suited to create natural, spontaneous timing. It was also thought to be much faster and less cumbersome than the keyframe-based method, and many participants predicted significant productivity improvements. Many participants also stated that they could well imagine using such tools for prototyping an animation timing, and tweaking details afterwards with standard keyframe tools.

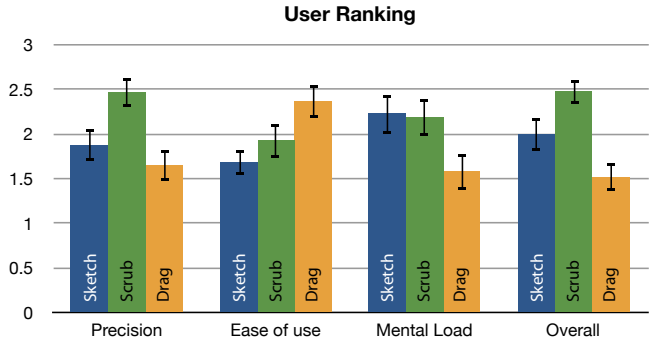

Figure 9: Comparison of user ranking means. Precision, mental load, overall: lower is better. Ease of use: higher is better. Error bars indicate 95% confidence interval.

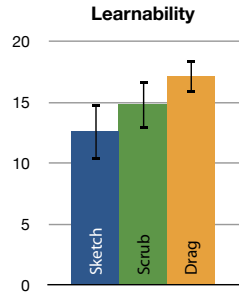

Figure 10: Comparison of learnability score means. Error bars indicate 95% confidence interval.

5.6.4 Summary of Results

The results show that Dragimation lets animators produce timings equally well as Sketching, and better than Scrubbing. This proves our hypothesis that Dragimation has a positive influence on the objective performance of timing regarding the timeline-based Scrubbing technique. We cannot claim this regarding Dragimation vs. Sketching since their objective performance seems to be about equal. Dragimation was judged to be more precise than Scrubbing, and not less precise than Sketching. All subjects found Dragimation easier to use than Sketching and Scrubbing, with less mental load than the other tools. Dragimation ranked top among all three tools in overall user preference. Dragimation was also thought easier to learn than Scrubbing, which in turn was judged easier to learn than Sketching. This proves our second hypothesis that Dragimation has a positive influence on the subjective preference of techniques regarding ease of use, mental load, overall preference and learnability. The interview comments further support these findings. They also show that many participants, including professional animators, could well imagine benefits from using performance timing tools in a keyframe animation process, supporting the workflow we suggested. Since both objective ability and subjective preference are very important for creative tools, we see these results as highly in favor of our more direct, in-the-loop timing technique.

6 Known Limitations

All three performance timing techniques studied suffer from problems already mentioned in the literature: The length of an animation that can be timed is limited. They cannot produce timings that are too fast for humans to recreate, such as high frequency motion, although linearly scaling playback speed could improve this. And

they need a maximally constant input-motion ratio for optimal use, i.e., the motion path must be mostly parallel to the view plane [23].

While Scrubbing can also be used to time non-spatial phenomena such as color change, Sketching and Dragimation are limited to timing motion. This can be approached by making such attributes controllable through spatial handles. Dragimation in particular is further affected by the physical forces influencing the animator's acting. When following the motion path, pen, fingers, hand and arm are subject to real world inertia and other physical restrictions, limiting timing possibilities. While this is also the case for Sketching, its editable mapping provides more flexibility. Thus one could "outwit" the system by, e.g., drawing a curve to retime the motion at a cusp, then making sure that the path is nevertheless mapped as desired via manual feature editing. However, transferring the animator's inertia to the timing can also be seen as a feature. After all, reproducing real world phenomena is what motion capture is all about. In any case, the laws of physics are as much part of the natural panache of humans as experience and intuition. What Dragimation does suffer from is occlusion and reach problems typical for direct manipulation on interactive screens [6]. A possible solution is to provide two duplicate views of the animation, one mainly for control and the other mainly for visualization, at the cost of directness. The problem of cusps in the curve snagging the dragged feature can be counteracted by either improving the proximity metric, or smoothing the curve, e.g., with a Gaussian filter.

7 Conclusion

We presented *Dragimation* as a new method for performance-based timing of keyframe animations. It was inspired by recent developments in direct manipulation video navigation techniques. We proposed that the close spatial correspondence between input and output and the interactive feedback of direct manipulation make it better suited for the performance timing task, for which intuitive mappings and natural interaction are essential. A user study with 27 participants of varying experience with animation comparing Dragimation to a *Sketching*-based and a timeline-based *Scrubbing* technique supports this claim. Dragimation and Sketching achieved significantly more precise results than Scrubbing in a timing-to-reference task. Dragimation was significantly higher ranked than both other techniques in a subjective assessment regarding ease of use, mental load, overall preference and learnability. We identified some deficiencies of the Dragimation technique and suggested ways to address them.

With this work, we intend to make animation timing more accessible by offering an improvement of directness and user satisfaction for timing tools. In a professional setting, animators could use performance-based techniques to develop an initial timing that they then refine with traditional tools, or others involved in the animation process but untrained in animation tools could use them as a means to better express and communicate their timing ideas. This is supported by the tenor of our interviews with professional animators and novices who both expect significant productivity enhancements from using performance timing in an animation workflow. It indicates that such realtime tools are a valid option for beginners and more experienced animators alike.

References

[1] J.-F. Balaguer and E. Gobbetti. Sketching 3D Animations. *Computer Graphics Forum*, 14(3):241–258, 1995.

[2] R. C. Davis, B. Colwell, and J. A. Landay. K-sketch: a 'kinetic' sketch pad for novice animators. In *Proc. CHI '08*, pages 413–422. ACM, 2008.

[3] M. Dontcheva, G. Yngve, and Z. Popović. Layered acting for character animation. *ACM Trans. Graph.*, 22(3):409–416, 2003.

[4] P. Dragicevic, G. Ramos, J. Bibliowitcz, D. Nowrouzezahrai, R. Balakrishnan, and K. Singh. Video browsing by direct manipulation. In *Proc. CHI '08*, pages 237–246. ACM, 2008.

[5] C. Esposito, W. B. Paley, and J. Ong. Of mice and monkeys: a specialized input device for virtual body animation. In *Proc. SI3D '95*, pages 109–213. ACM, 1995.

[6] C. Forlines, D. Vogel, and R. Balakrishnan. HybridPointing: fluid switching between absolute and relative pointing with a direct input device. In *Proc. UIST '06*, pages 211–220. ACM, 2006.

[7] D. B. Goldman, C. Gonterman, B. Curless, D. Salesin, and S. M. Seitz. Video object annotation, navigation, and composition. In *Proc. UIST '08*, pages 3–12. ACM, 2008.

[8] T. Igarashi, T. Moscovich, and J. F. Hughes. Spatial keyframing for performance-driven animation. In *Proc. SCA '05*, pages 107–115. ACM, 2005.

[9] T. Karrer, M. Weiss, E. Lee, and J. Borchers. DRAGON: a direct manipulation interface for frame-accurate in-scene video navigation. In *Proc. CHI '08*, pages 247–250. ACM, 2008.

[10] D. Kimber, T. Dunnigan, A. Girgensohn, F. Shipman, T. Turner, and T. Yang. Trailblazing: Video Playback Control by Direct Object Manipulation. In *Proc. ICME '07*, pages 1015–1018. IEEE, 2007.

[11] M. Kipp and Q. Nguyen. Multitouch puppetry: creating coordinated 3D motion for an articulated arm. In *Proc. ITS '10*, pages 147–156. ACM, 2010.

[12] H. Kouda, I. Kanaya, and K. Sato. Motion Stroke-A Tablet-Based Interface for Motion Design Tool Using Drawing. In *Proc. HCI Int. '09*, pages 821–829. Springer, 2009.

[13] J. Laszlo, M. van de Panne, and E. Fiume. Interactive control for physically-based animation. In *Proc. SIGGRAPH '00*, pages 201–208. ACM, 2000.

[14] J. Lewis and J. Sauro. The Factor Structure of the System Usability Scale Human Centered Design. In *Human Centered Design*, volume 5619 of *LNCS*, chapter 12, pages 94–103. Springer, 2009.

[15] J. McCann, N. S. Pollard, and S. Srinivasa. Physics-based motion retiming. In *Proc. SCA '06*, pages 205–214. Eurographics Association, 2006.

[16] T. Moscovich, T. Igarashi, J. Rekimoto, K. Fukuchi, and J. F. Hughes. A Multi-finger Interface for Performance Animation of Deformable Drawings. In *Proc. UIST '05*. ACM, 2005.

[17] M. Neff, I. Albrecht, and H.-P. Seidel. Layered Performance Animation with Correlation Maps. In *Proc. EG '07*, pages 675–684. Eurographics Association, 2007.

[18] S. Oore, D. Terzopoulos, and G. Hinton. A Desktop Input Device and Interface for Interactive 3D Character Animation. In *Proc. GI '02*, pages 133–140, 2002.

[19] S. Oore, D. Terzopoulos, and G. Hinton. Local Physical Models for Interactive Character Animation. *Computer Graphics Forum*, 21:337–346, 2002.

[20] J. Sampath. Nuke. http://sjagannathan.tripod.com/nuke, Last accessed: 2011/09/23, 1999.

[21] R. Shah and P. J. Narayanan. Trajectory based Video Object Manipulation. In *Proc. ICME '11*. IEEE, 2011.

[22] D. J. Sturman. Computer Puppetry. *Computer Graphics in Entertainment*, 1998.

[23] S. C. Terra and R. A. Metoyer. A performance-based technique for timing keyframe animations. *Graph. Models*, 69:89–105, 2007.

[24] S. C. L. Terra and R. A. Metoyer. Performance timing for keyframe animation. In *Proc. SCA '04*, pages 253–258. Eurographics Association, 2004.

[25] M. Thorne, D. Burke, and M. van de Panne. Motion doodles: an interface for sketching character motion. *ACM Trans. Graph.*, 23(3):424–431, 2004.

[26] B. Walther-Franks. Performance Animation of 3D Content on Multitouch Interactive Surfaces. In *Proc. ITS '10*, pages 343–346. ACM, 2010.

[27] B. Walther-Franks, M. Herrlich, and R. Malaka. A Multi-Touch System for 3D Modelling and Animation. In *Proc. SG '11*. Springer, 2011.

[28] P. Zhao and M. van de Panne. User interfaces for interactive control of physics-based 3D characters. In *Proc. I3D '05*, pages 87–94. ACM, 2005.