DragLocks: Handling Temporal Ambiguities in Direct Manipulation Video Navigation

Thorsten Karrer **Moritz Wittenhagen** **Jan Borchers**
RWTH Aachen University
52074 Aachen, Germany
{karrer, wittenhagen, borchers}@cs.rwth-aachen.de

ABSTRACT

Direct manipulation video navigation (DMVN) systems allow to navigate inside video scenes by spatially manipulating objects in the video. Problems arise when dealing with temporal ambiguities where a time span is projected onto a single point in image space, e.g., when objects stop moving. Existing DMVN systems deal with these cases by either disabling navigation on the paused object or by allowing jumps in the timeline. Both of these workarounds are undesirable as they introduce inconsistency or provoke loss of context. We analyze current practices regarding temporal ambiguities and introduce two new methods to visualize and navigate object pauses. User tests show that the new approaches are better suited for navigation in scenes containing temporal ambiguities and are rated higher in terms of user satisfaction.

Author Keywords
Video navigation; direct manipulation; accessing pauses.

ACM Classification Keywords
H.5.2 Information Interfaces and Presentation: User Interfaces. - Interaction styles.

INTRODUCTION

Navigating in videos, i.e., accessing a certain frame inside a length of video material for viewing, still frame extraction, or editing, has traditionally been performed using indirect controls such as the timeline slider or direct entry of frame numbers and time codes. Recently, several approaches have been proposed that allow navigation by interacting with the content of the video directly. With these direct manipulation video navigation (DMVN) systems, objects in the video can be grabbed and dragged along their motion trajectories, causing the video to advance or reverse accordingly [3, 4, 6, 7].

While DMVN systems have been shown to significantly increase efficiency for certain in-scene navigation tasks [3, 6], the interaction fails in the presence of certain object motion patterns. Figure 1, e.g., shows an object that halts its movement for a couple of frames. When dragging the object across the screen, those frames either have to be skipped—losing the context of what else happens in the scene—or one has to give up on the direct mapping between the positions of the cursor and the object. Clearly, similar problems occur whenever an object occupies the same position in the scene at multiple times; the object's trajectory exhibits *temporal ambiguities*. Three different cases of motion patterns related to this problem can be identified:

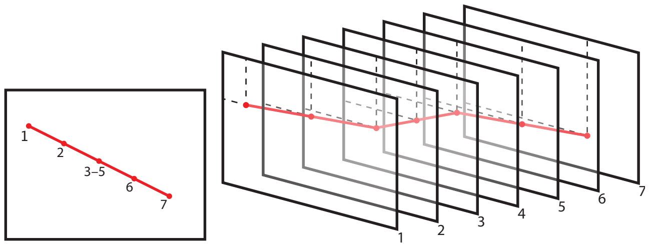

Figure 1. If an object occupies the same position in multiple frames, e.g., while pausing, the trajectory through these positions collapses to a single point. The frames then cannot be accessed using DMVN techniques.

Recurring movements, e.g., the rotating hands of a clock or a ball bouncing up and down. Here, problems arise mainly because it is difficult to know in which 'iteration' of the movement the object is at the moment. When the trajectory forms a cusp, it is unclear if the video is to be advanced forward or backward in time when dragging an object away from the cusp; to counter this problem, some DMVN systems enforce directional continuity constraints [3, 7].

Self-intersecting trajectories, e.g., a looping roller coaster or a car heading straight towards the camera. Similar to the case of recurring movements, the mapping problems can be mitigated by employing temporal or arc-length continuity constraints during the navigation [3, 6]. If the object is at a position where the trajectory self-crosses many times it may still be confusing to the user.

Pauses, e.g., a model walking down the catwalk, striking a pose and holding it for a few seconds before moving on. Ideally, when interacting with an object, a DMVN technique should allow access to those parts of the video where the object is pausing: Assume, for example, that we drag the model into the pose and then want to navigate to a frame *inside that* *pause* where the lighting is perfect or the model assumes a certain facial expression. This is a situation in which most strategies of existing DMVN systems fail. Because the pause only occupies a single pixel on the movement trajectory (cf. Fig.2b) it is skipped entirely. Thus, users either miss everything that is happening in the video while the object stops, possibly failing to notice the existence of a pause at all, or they have to switch to another means of navigation, like the timeline slider (cf. Fig.2a), to access the video frames in the pause. Although pauses structure a scene temporally—which should help with navigation—, DMVN systems cannot leverage this advantage but are hindered by them.

In the remainder of the paper, we analyze existing approaches to deal with temporal ambiguities and propose two different solutions that allow pause navigation without relinquishing control of the dragged object. We then report on a controlled experiment, where we evaluated the effectiveness and usability of our techniques compared to the existing approaches.

TEMPORAL AMBIGUITIES IN CURRENT DMVN SYSTEMS

The existing literature discusses recurring movements and self-intersecting trajectories together with techniques to suppress undesired jumps in the video while navigating. Pauses have been neglected so far. The current approaches rely on modifying the distance measure d that is used to determine the next frame f to be displayed during the interaction:

$$f(p,T) = [\operatorname*{argmin}_{t \in T}(d(p,t))]_f$$

where p is the screen position the user is dragging the object to and $T \subseteq F \times P$ is the object's trajectory consisting of tuples (t_f, t_p) of a frame number and a position.

Two classes of distance measures have been proposed: *purely spatial* distance measures depending only on t_p, and *spatio-temporal* distance measures depending on both t_p and t_f.

The most straightforward implementation of a purely spatial distance measure is found with Goldman et al. [4].

$$d_1(p,t) \cong \|p - t_p\|$$

While dragging an object, the system will always display the frame where the object's position is closest to the mouse cursor. The advantage of this technique is that objects can be very quickly repositioned along their trajectory. However, the playback position may freely jump discontinuously over the length of the video. Time ranges where the object pauses are not accessible so users have to resort to using the timeline slider to navigate inside a pause. In fact, the pauses are skipped completely and can only be detected by sudden changes in the rest of the frame when dragging across the pause position.

Dragicevic et al. [3] propose a spatial distance measure that reduces the frequency of jumps and ensures directional continuity at cusp points:

$$d_2(p,t) \cong \|p - t_p\| + \|\text{arclen}(t_p, o_p)\| + k_D$$

with o_p being the object's current position and $k_D > 0$ being added whenever the arc-length changes signs from the last step. In their paper, they specifically discuss the problems of recurring movements at cusp points and self-intersecting trajectories, and they formulate directional continuity and arc-length continuity as two requirements for DMVN systems. Due to d being a purely spatial metric, however, their system also skips object pauses.

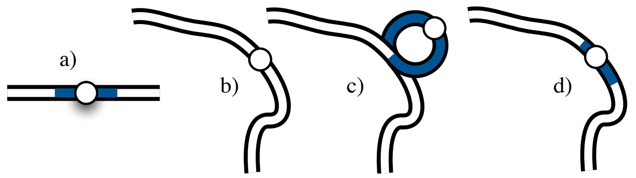

Figure 2. Schematic of the spatial extent of a pause (blue) for a) *timeline slider*: No special handling of pauses, all frames are distributed evenly. b) *DMVN*: Pause has no extent. c) *DMVN with embedded timeline*: Pause is distributed over a part of the trajectory. d) *DMVN with loop*: The trajectory form is altered to contain a loop representing the pause.

Karrer et al. [6] discuss the need to disambiguate situations of recurring movements and to avoid jumps in the video during navigation. They propose a spatio-temporal distance function that includes the current frame number o_f.

$$d_3(p,t) \cong \|(p_x - t_{p_x}, p_y - t_{p_y}, o_f - t_f)^T\|$$

This approach works reasonably well when it comes to self-intersecting or recurring movement trajectories (but not cusps), and it allows to easily spot pauses; the dragged object will 'stick' to the beginning of the pause. Dragging the object further, the spatial distance component outweighs the temporal one and the object will 'snap' to a position some time after the pause. While accessing the frames after the pause is possible by backtracking along the trajectory, navigating the contents of the pause is not possible.

Kimber et al. [7] propose a similar spatio-temporal distance measure that also ensures directional continuity at cusps.

$$d_4(p,t) \cong c_\theta \cdot \|p - t_p\| + \|o_f - t_f\| + k_D$$

with k_D defined as above. Otherwise, this approach behaves like d_3 but includes a time-dependent term c_θ that makes the 'snapping' across pauses happen more easily the longer the dragging takes.

We can see that spatio-temporal distance measures mitigate the problem of jumps during the interaction and at least allow to recognize object pauses. At the same time, they do not help to navigate the pauses, and they introduce a number of additional problems on their own: these compound measures have to be weighted according to the spatial and temporal sampling rates of the video—there is no universal answer to the question "how many pixel worth is one frame?".

PROPOSED SOLUTIONS

Outside the domain of DMVNs, solutions to a similar class of problems have been proposed for some time in the context of the general domain of direct manipulation control elements like sliders and selectors. Accessing elements in a list without mapping each element to its own pixel on a slider has been investigated among others by [1] and [2]. This problem setting is akin to navigating object pauses in DMVN in so far as the time range of a pause is mapped in its entirety to a single pixel, too, and thus cannot be navigated. Our approach thus borrows from these techniques: instead of changing the distance measure to navigate pauses, we locally extend the geometry of the trajectories—a technique inspired by the Popup-Vernier [2]. Alternatively, we dynamically change the mapping along the trajectory depending on the location of the interaction; this idea is akin to the Alphaslider [1].

Loop

Our first technique expands the single spatial point of a pause to a *loop* on the trajectory (Fig. 2c). Thus, the pause can now be navigated in one stroke, without having to re-home to the timeline slider. The distinct shape of a loop is a recognizable pattern that gives users a good hint of its special meaning. The user can navigate to the beginning (or the end respectively) of the pause by dragging the object to the base of the loop. Frames inside the pause are equally distributed on the loop and are now accessible as if on a curved timeline slider.

Navigation inside the loop can be performed at different temporal granularities depending on the radius of the dragging gesture; this is similar to the micrometer interface [1] or the mobile zoom slider [5]. Finally, the loop ensures that the rest of the trajectory is unmodified, guaranteeing direct manipulation of the object on the rest of the trajectory. To avoid the need of traversing the loop when the user intends to navigate to a time after the pause, the loop could be deactivated via a *quasimode* [8], e.g., by holding down the control key.

Embedded Timeline

For our second technique, we change the mapping along a short extent of the trajectory around the pause location, so that this part controls the time of the video instead of the location of the object (Fig. 2d). All time points on this part of the trajectory are redistributed equally. The embedded timeline thus does not change the form of the trajectory but changes its meaning in a local area around the pause. Expanding the pause into the spatial domain guarantees that we can navigate the pause in one stroke without having to leave the object of interest. Conserving the trajectory shape allows navigation to a point after the pause while retaining of what happens during a pause. This comes at the cost of decoupling navigation from the object on that part of the trajectory that is used for the expansion. With this technique there is no way to identify the pause from the shape of the trajectory alone; thus, we set the color of the embedded timeline to be different from that of the rest of the trajectory. For longer pauses, this technique will show the same resolution problems as the timeline slider, making it hard to navigate to a specific frame inside the pause.

Both techniques work best if used with either a spatio-temporal distance measure or a spatial distance measure that ensures arc-length continuity. Although pauses would be accessible even with a metric like d_1, the extended geometry could very easily be skipped involuntarily during the interaction.

NAVIGATION AROUND PAUSES

We identified four different target areas on a trajectory in regard to navigation around pauses: *Before the pause*, at the *edge of the pause*, *in the pause*, and *after the pause*. In the following, we describe these cases for forward navigation. Navigation *before the pause* (cf. Fig 1, frames 1–2) is equivalent to not having a pause at all and thus is already covered by the standard DMVN approaches. The *edge of a pause* (cf. Fig 1, frame 3) is the point were an object stops moving. It is accessible easily when using spatio-temporal distance measures, because of the 'sticking', and when using the loop, because the pause is visually distinguishable. Navigating to the edge is harder when using purely spatial distance measures or the embedded timeline. Loop and embedded timeline both support navigation *in a pause* (cf. Fig 1, frames 3–5). We expect the loop to perform better for longer pauses because the embedded timeline will suffer from resolution issues. Existing DMVN systems—regardless of the choice of distance measure—require the use of the timeline slider to navigate in the pause since the pause has no spatial extent. Navigation to the area *after the pause* (cf. Fig 1, frames 6–7) requires crossing the pause. This is easiest for purely spatial distance measures, only requiring to drag the object to the target position. Spatio-temporal distance measures may require the help of the timeline slider to cross the pause. Both loop and embedded timeline allow crossing the pause easily; if the quasimode mentioned above is not implemented the loop requires the traversal of the extended shape of the path.

EXPERIMENT

We conducted an experiment to investigate navigation performance for *purely spatial* and *spatio-temporal* distance measures (d_1 and d_3) and the *loop* and *embedded timeline* techniques. In our implementation of loop and embedded timeline, we also used d_3 as distance measure. In the loop condition, there was no quasimode and the loops were always visible. We tested these four techniques in six different experiments: navigation to a point on the *edge*, *inside*, and *after* a pause, each in separate versions for long (>25s) and short (<5s) pauses. For each of the six experiments, we used a different section of a longer video showing two people playing a board game. Trajectories were created using optical flow fields as described in [6]. We detected a pause when an object was not moving for more than 0.2s.

Participants used a Wacom Cintiq 12WX and a stylus as an input device. They were introduced to DMVNs before the experiment and had time to familiarize themselves with each of the four techniques on a separate video. For each experiment, subjects were given a simple navigation task ("Navigate to the frame where the right player rolls a six."). After each experiment, people were asked to rank the four techniques in order of their personal preference. To avoid learning effects, participants watched each video scene twice before performing the task. At the end, we performed a short structured interview with the participant.

The study design was within-subjects; subjects performed the task in each experiment using all techniques. The order of techniques was randomized for each participant using a balanced 4x4 latin square. We measured the navigation time in each trial, defined as the time between acquisition of the object to lifting the stylus in a 0.5s range of the target

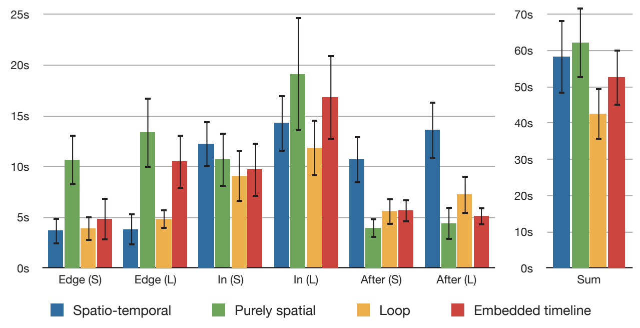

Figure 3. Summary of navigation times. Left: Means of navigation time and 95% confidence intervals for all techniques and experiments. Navigation to the *edge*, *in*, and *after* short (S) and long (L) pauses. Right: Means of the sum of navigation times for each technique.

time. We analyzed the measurements with a repeated measures ANOVA using Greenhouse-Geisser correction and, in case of significance, post-hoc analysis employing paired t-tests with Bonferroni correction. For nonparametric data, we used a Friedman test and Wilcoxon signed-rank tests.

28 users (four female) between 21 and 36 years were recruited from our university's campus. Free snacks and drinks were offered during the study and we raffled an Amazon gift card.

Results

Before analyzing the data, we removed two outliers where a participant misunderstood the task. Figure 3 shows a summary of the recorded navigation times. When looking at the sum of navigation times for each technique, subjects were fastest when using loop ($M = 42.1s$, $SD = 17.2s$), followed by embedded timeline ($M = 50.5s$, $SD = 18.8s$), spatio-temporal ($M = 58.5s$, $SD = 24.9s$) and purely spatial ($M = 62s$, $SD = 23.7s$). Differences were strongly significant between loop and purely spatial ($p < .001$), and significant for loop and spatio-temporal ($p < .05$).

Navigation to the edge of a pause was significantly faster for the spatio-temporal and loop conditions than for the purely spatial condition ($p < .001$). Embedded timeline was only faster than purely spatial for short pauses ($p < .05$).

When navigating to a point after a pause the purely spatial, loop, and embedded timeline conditions were significantly faster than the spatio-temporal condition ($p < .001$). Additionally, purely spatial and embedded timeline performed significantly better than loop for long pauses ($p < .05$).

The data revealed that our participants strongly preferred loop over all other techniques ($p < .001$). Also, the embedded timeline technique was preferred over the spatio-temporal condition ($p < .001$).

During the interview, when asked if they ever felt that they did not know where they were in the video with any of the techniques, one participant named loop, two embedded timeline, four the spatio-temporal and 12 the purely spatial technique, indicating that, indeed, the purely spatial navigation can provoke a loss of context. When being asked if they could keep a good overview over what was happening in the video with any of the techniques, the results were consistent (loop: 17, embedded timeline: 8, spatio-temporal: 4, purely spatial: 4).

CONCLUSION AND FUTURE WORK

The results indicate that both the spatio-temporal and the purely spatial conditions do not perform well overall in regard to pauses. Embedded timeline works well for short pauses but is problematic in long pauses. Loop seems to be a viable alternative for all tested situations and was the preferred technique by our users. For future work we plan looking into techniques that deal with the limitations of the work presented here. Limitations we identified are overlapping loops or embedded timelines on paths containing frequent pauses and the added overhead of having to navigate several pauses even when navigating to a time after all pauses. For the latter, using a quasimode to dynamically disable the loop would be helpful to quickly get across pauses. Further work includes a similar analysis of recurring movements and combining the conclusions with the ones presented here. Also, we plan to further analyse pauses, especially of varying lengths, and variations of the techniques presented here, e.g. different visualizations.

ACKNOWLEDGMENTS

This work was partially funded by the German B-IT Foundation and by the German Government through its UMIC Excellence Cluster for Ultra-High Speed Mobile Information and Communication and its HumTec Human Technology Center at RWTH Aachen University.

REFERENCES

1. Ahlberg, C., and Shneiderman, B. The Alphaslider: A rapid and compact selector. In *Proc. of CHI'94* (1993).
2. Ayatsuka, Y., Rekimoto, J., and Matsuoka, S. Popup vernier: a tool for sub-pixel-pitch dragging with smooth mode transition. In *Proc. of UIST'98* (1998).
3. Dragicevic, P., Ramos, G., Bibliowitcz, J., Nowrouzezahrai, D., Balakrishnan, R., and Singh, K. Video browsing by direct manipulation. In *Proc. of CHI'08* (2008).
4. Goldman, D., Gonterman, C., and Curless, B. Video object annotation, navigation, and composition. In *Proc. of UIST'08* (2008).
5. Hürst, W., and Götz, G. Interface designs for pen-based mobile video browsing. In *Proc. of DIS'08*, ACM Press (2008).
6. Karrer, T., Weiss, M., Lee, E., and Borcers, J. DRAGON: a direct manipulation interface for frame-accurate in-scene video navigation. In *Proc. of CHI'08* (2008).
7. Kimber, D., Dunnigan, T., Girgensohn, A., Shipman, F., Turner, T., and Yang, T. Trailblazing: Video Playback Control by Direct Object Manipulation. In *Proc. of ICME'07, IEEE* (2007).
8. Raskin, J. *The humane interface*. New directions for designing interactive systems. Addison-Wesley, 2000.