Improving Early Navigation in Time-Lapse Video with Spread Loading

Carl Gutwin, Michael van der Kamp, Md. Sami Uddin, Kevin Stanley, Ian Stavness, and Sally Vail
University of Saskatchewan / Agriculture and Agri-Food Canada
Saskatoon, SK
firstname.lastname@usask.ca; sally.vail@canada.ca

Figure 1: Three views of a time-lapse sequence illustrating spread-loading. Left: 5-month overview at $\frac{1}{4}$-size (all frames loaded). Middle: 3-day $\frac{1}{2}$-size view (green bars indicate loading pattern). Right: 2-hour full-size view (nine frames loaded).

ABSTRACT

Time-lapse videos are often navigated by scrubbing with a slider. When networks are slow or images are large, however, even thumbnail versions load so slowly that scrubbing is limited to the start of the video. We developed a frame-loading technique called spread-loading that enables scrubbing regardless of delivery rate. Spread-loading orders frame delivery to maximize coverage of the entire sequence; this provides a temporal overview of the entire video that can be fully navigated at any time during delivery. The overview initially has a coarse temporal resolution, becoming finer-grained with each new frame. We compared spread-loading with traditional linear loading in a study where participants were asked to find specific episodes in a long time-lapse sequence, using three views with increasing levels of detail. Results show that participants found target episodes significantly and substantially faster with spread-loading, regardless of whether they could click to change the load point. Users rated spread-loading as requiring less effort, and strongly preferred the new technique.

CCS CONCEPTS

• Human-centered computing → Web-based interaction.

KEYWORDS

Video navigation, temporal overviews, scrubbing

ACM Reference Format:
1 INTRODUCTION

Time-lapse videos are video sequences where frames are captured at a much lower frequency than the playback rate, leading to the illusion that time is moving faster than normal. The ubiquity of inexpensive cameras and networks means that time-lapse video is becoming common – including security footage, life-logging recordings, wildlife “trail cam” videos, or records of agricultural test plots (e.g., Figure 1).

One of the main interactive tasks that occurs when working with time-lapse video involves exploring the video for episodes of interest: e.g., looking for the appearance of an animal on a wildlife camera, or finding a particular stage in a plant’s development (such as flowering). However, navigating in time-lapse video can be difficult compared to other kinds of linear content. The standard technique for supporting exploratory navigation is scrubbing, where users drag a linear controller such as a seek bar back and forth to find episodes of interest. Scrubbing, however, only works well when all of the video’s frames are available – and when the frames must be delivered over a network, the loading time for the video can be slow enough to make scrubbing infeasible (because only the beginning of the video is available). Many video players allow users to click on the seek bar to change the video’s loading point – but this requires that users know where to click. In order to navigate effectively, users need to have an overview understanding of the video.

Although an experienced user may simply remember where different episodes appear, users who are less familiar with the content need a more concrete overview to guide their navigation. For example, many video players load a smaller version of the video first, and then let the user navigate using the overview rather than waiting for the full sequence. Video overviews are effective because the user’s navigation decisions are often not dependent on a full-resolution representation of the sequence. That is, a user can often make early navigation decisions from the overview – such as deciding on a portion of the video to load in more detail – without waiting for the entire video to arrive.

A standard method of providing an overview is to load a series of key frames that appear above the seek bar. This approach can work well when the episodes of interest can be visible in a thumbnail – but in most time-lapse videos, many visual elements of the scene stay the same over the course of the video. This means that key-frame overviews are often too small to be useful for the user to make navigation decisions. In addition, some tasks require a more detailed version of the overview to make a navigation decision.

In this paper we present a novel solution to the problem of early navigation in video, called spread-loading. Based on the idea that scrubbing can provide a quick temporal overview of the entire video, we change the order in which frames are loaded to optimize scrubbing even when frames load slowly. Whereas the standard frame-loading strategy loads frames linearly from the current loading point, spread-loading maximally distributes the location of the next frame to load: it starts by loading the middle, first, and last frames, then the frames at the center of every remaining gap (Figures 1 and 5). At any point during the video’s delivery, there is always a complete-as-possible overview that can be scrubbed from the start to the end of the video sequence. The overview initially has a very coarse temporal resolution, becoming finer-grained with the arrival of every new frame. The goal of spread-loading is to let users make early navigation decisions – decisions that could mean that most of the video does not need to be delivered at all.

We implemented linear loading and spread-loading in a web-based video player, and compared the two techniques in a within-participants experiment (N=16). Participants were asked to find specific episodes in a summer-long time-lapse sequence of a crop growing in a field. Participants had to first find the approximate location of a target episode using a quarter-size video (where a selection chose a three-day period), then refine their navigation selection in a half-size video (selecting about two hours), then switch to a full-size video that had enough detail to identify the target episodes. We simulated a low-bandwidth streaming network, and used targets that are meaningful to the analysis of the crop in the video (e.g., the start of flowering). We asked participants to find targets as quickly and accurately as possible, using both a linear and a spread-loading technique. In addition, we tested versions of each technique where users could click to change the loading point.

Our results show that spread-loading allowed participants to complete their tasks significantly and substantially faster than linear loading. This advantage for spread loading occurred regardless of whether users were allowed to change the load point. Performance with spread loading was almost 30 seconds faster per target than linear loading. Clicking to change the load point did not significantly improve the performance of linear loading. In addition, NASA-TLX effort scores significantly favored spread-loading, and fourteen of sixteen participants preferred the new method.

Spread-loading provides a temporal overview that lets users make early navigation decisions, even when overviews load slowly (e.g., with remote networks or Tor-style routing, or when a more detailed overview is needed in order to make subtle judgements). Spread-loading maximizes the amount of information about the entire video, at any point during the video’s delivery, making scrubbing more effective regardless of the speed of the network or the size of the video. We focus on time-lapse video, but the underlying approach could benefit other types of video as well.
2 RELATED WORK ON VIDEO NAVIGATION

Streaming video allows the user to start viewing video frames (e.g., start playback) before the entire video file has been downloaded, and is now the standard method for viewing video on the Internet. A wide variety of research has been carried out on the technical aspects of streaming, from video transmission techniques and protocols, to multicasting strategies and video encoding techniques (see [2, 20, 38] for summaries). Substantial research has also been carried out to improve user tools for browsing, navigating, and working with video streams (e.g., [3, 10, 14, 22, 30, 32]). Researchers have investigated issues such as providing finer-grained interaction with video [29, 30], navigation through direct manipulation of objects in the video [10], dealing with multiple related videos in video libraries [28], and, of particular importance to our work, providing video summaries and overviews.

Overviews are valuable in many visual tasks because they provide a high-level representation of an entire document or dataset, allowing users to explore the data and make decisions without using the full-resolution version [23]. Video is one type of data where overviews are useful – however, video overviews are different than other types because video is a linear dynamic medium where a large number of individual images (frames) are typically viewed in sequence. This means that overviews and summaries must deal with the temporal nature of a video as well as its visual and spatial information. A number of summarization techniques have been proposed, each supporting different kinds of navigation.

Early work on video summarization involved techniques for creating condensed versions of a video [28], and features for compressed playback, extraction of text (e.g., from slides in a presentation), and visual representations of key frames (e.g., in a ‘film-strip’ format) [21]. Controls for scrubbing were added to some video players to allow quick navigation to a particular scene [21]. Much of this work was focused on retrieval and navigation of local video, which loads quickly; the introduction of network streaming technologies highlighted the need for overviews because of the delay in delivering the entire video file to the viewer [20].

Several types of overview are possible for streaming videos, each able to support different types of navigation decisions and user interactions.

*Extracted structure*. A number of techniques extract information from the video or audio track (or the interaction history of other users) to determine a high-level structure such as a table of contents, a list of topics discussed, or scene list (e.g., [12]). These techniques are often used for summarization and navigation of meetings or presentations. The extracted information is typically displayed and accessed in a separate window or panel beside the video [32]. Other representations are also possible: for example, the Video Skimmer [21] and Smart Video Player [7] provided a row of thumbnails showing important frames that are extracted based on a summarization algorithm (e.g., scene changes or shot boundaries). One problem with this approach is that there is not enough space in a video player to show a large number of these keyframes. Researchers have proposed various solutions to this problem: for example, some systems use a larger space and present the key frames in a storyboard, comic-book [6, 35], or magazine [19] format; others use methods of compressing the information about frames into the available space. The idea of compressing keyframes

---

**Figure 2: Thumbnail overviews in YouTube and Netflix.**

Tapestries system creates a static linear collage using images of characters and events from key frames, allowing composition of more elements than would be possible in a storyboard [4]; and a technique presented by Rav-Acha and colleagues [31] multiplexes the video summary to overlay time-separated events onto a single frame.

*Annotation-based overviews*. A user can also make navigation decisions based on timeline annotations that represent video content or other users’ actions. For example, the VANNA system shows annotations about important events on the video timeline [11]; other tools show a histogram of users’ actions in video-based learning environments [19, 33, 39] or artificial landmarks in the seek bar to improve memorability [36]. SceneSkim [27] shows several tracks including captions, scripts, and plot summaries.

*Dynamic scrubbable thumbnail overviews*. Thumbnail versions of a video’s key frames have long been used as a way to summarize the video content [21, 25]. There are several different ways that thumbnails can be extracted and displayed for streaming video. Matejka and colleagues [23] recognized that users often want to navigate beyond the current load point in a streaming video, but they need a representation of the video in order to do so. Their Swift system sent a thumbnail overview across the network first, so that users could make navigation decisions such as where to click in the main video. This approach is now standard practice on most networked video players (e.g., Figure 2); the thumbnail view pops up and "plays" as the user moves their mouse over the seek bar, providing access to the entire video regardless of the download progress. As described below, however, thumbnail overviews need to match the user’s navigation task with the spatial and temporal resolution of the video.

*Static thumbnail overviews*. Early systems such as Video Skimmer [21] and Smart Video Player [7] provided a row of thumbnails showing important frames that are extracted based on a summarization algorithm (e.g., scene changes or shot boundaries). One problem with this approach is that there is not enough space in a video player to show a large number of these keyframes. Researchers have proposed various solutions to this problem: for example, some systems use a larger space and present the key frames in a storyboard, comic-book [6, 35], or magazine [19] format; others use methods of compressing the information about frames into the available space. The idea of compressing keyframes
to the width of a single pixel has been explored in the Color Browser, which colors lines of pixels in the overview according to the dominant color of each frame [3]; other systems use a slit-scan approach to support change awareness [26, 34].

Hybrid approaches. A few projects have combined features from dynamic and static overviews. The Swifter system [24] address the limits on the temporal resolution of a traditional dynamic overview – that is, fixed-length overviews have a low sampling rate for longer videos, meaning that scenes can flash by too quickly or be missed entirely during scrubbing. Swifter’s approach is to provide a full-screen paged view that can accommodate more thumbnails [24]. A similar approach is taken by the Panopticon system [16], which also presents a large grid of keyframes along with interaction techniques for scrubbing and exploring scenes related to the keyframes. In addition, some systems make the summarization process interactive - such as ElasticPlay, which provides user control over the summarization process based on the current “time budget” for watching a portion of the video [17].

3 DESIGN FRAMEWORK AND SPREAD-LOADING

When designing an overview technique for streaming video, several issues must be taken into account in order to best support the tasks that the user will need to carry out. Here we summarize factors that are relevant to the design of an overview technique for time-lapse video, covering issues of loading rate, quality requirements, ordering, and presentation. We then describe the idea of spread-loading and present an overview technique that uses the idea.

Image resolution. Video frames can be downsampled to reduce the size of the thumbnail overview. However, size reduction reduces image detail, and different navigation tasks will require different levels of detail in an overview. Quality requirements are dependent on the task and on the visual characteristics of the video: for example, finding a scene that involves a large visual change (e.g., from an indoor to an outdoor background) may be possible with a very small overview, but finding an event that is less visually obvious (e.g., whether flowering has started in the crop shown in Figure 3) requires a more detailed overview image.

Frame selection and temporal resolution. Video overviews use only a small subset of frames, and there are several ways of determining which frames should be included. A simple approach is to use uniform sampling (i.e., take every nth frame), where the sampling rate changes the temporal resolution of the overview. This introduces a trade-off between the coverage and the size of the overview: if the overview is a fixed length (e.g., 200 frames) then longer videos will have more space between each thumbnail (potentially missing important scenes); if the sampling rate is held constant (e.g., use every 50th frame), then the overview varies in size based on the length of the video (e.g., for our 150,000-frame time-lapse sequences, a 1:50 sampling rate implies an overview of 3000 frames). Several researchers have considered this problem and proposed non-uniform sampling techniques. For example, Bennet and colleagues describe several sampling techniques that aim to provide an optimal set of frames for a video summary based on content; their techniques can also be used to create visual effects in the summary such as motion trails for objects that move through the scene [5]. Similarly, Joshi and colleagues present a technique for optimal frame selection to provide a given speedup while also maintaining smooth camera motion in the summary [18].

Network loading rate. We focus on situations where the user wishes to start navigating right away on selecting a video, using the video overview and scrubbing. The ability of a system to deliver an overview within 2-4 seconds (allowing near-immediate interaction) depends on the number of frames in the overview, the size of the frames, and the network speed. To illustrate the trade-offs, Figure 4 shows loading times for a 200-frame overview with various frame sizes from 6-14KB and network speeds from 1-10Mbps. For reference, 6KB is needed for a single 64x36 JPEG-encoded image taken from the image set used in our study; 9KB is needed for a 128x72 image, and 14KB for a 150x84 image (video formats require less space on average, but these sizes approximate MEG I-frames which can be decoded independently [15]). We note that all of these frame sizes are likely too small for any analysis tasks involving detailed examination and analysis of the time-lapse images; for larger images, the situation in Figure 4 only becomes worse.

Streaming the overview. Even though a thumbnail overview is much smaller than the entire video, it is advantageous to allow streaming of the overview in order to allow early navigation actions before the full overview is received. This is the situation investigated in the study below.
Order in which frames are sent. When streaming, frames can be delivered in different orders. Traditional loading always only sends frames in increasing order, never going back to fill gaps; spread-loading changes this strategy to maximally fill in every gap and provide evenly-distributed coverage of the source video sequence.

Adaptation to network conditions. Since network bandwidth varies with traffic patterns, a fixed-size overview that is appropriate for some conditions may be unsuitable when traffic is high and available bandwidth drops. Some techniques dynamically adapt to network conditions by measuring latency, delivery time, and loss rate, and then changing the quality of the video to better match the network’s current resources. For example, several services use HTTP Adaptive Streaming [1] for the actual video, and the same technique could be used to adapt delivery of overviews.

Static or dynamic UI representation. A main UI-level decision involves whether to display the overview dynamically (e.g., a popup over the scrubbing cursor) or statically (e.g., a filmstrip or grid of images). For time-lapse video, we believe that dynamic representations are a better fit, because the self-similarity of the frames means that animating the frames through scrubbing allows users to see differences over time. Other video types with larger visual changes might be better suited by a static representation such as a grid of thumbnails; these can avoid the flashing and visual ‘popping’ that can occur in a dynamic representation.

Changing the load point. A second client-side issue is whether the user can interact with the in-progress overview to change delivery parameters. Most importantly, the user may wish to see the overview first for a particular region of the video – for example, they may know that the scene they want to find is near the end of the video, so they click to move the load point of the overview. We call this “click-seeking;” it is already standard for the main video, but can also be applied to slow-loading overviews.

Spread-Loading for Video Overviews

Spread-loading changes the load order of frames in a streaming video overview. Typical overviews load linearly (from frame 1 to frame n), but this poorly supports scrubbing through the entire video when the overview does not load immediately. Spread-loading, in contrast, loads frames in the order in which they will be of use for navigation. Which frames are most useful for early navigation depends on the content and the user’s tasks – for time-lapse video, providing coverage of the full time range of the video is likely to best support navigation actions.

Once the relative importance of overview frames is determined, spread-loading attempts to provide maximal coverage of the video sequence regardless of the number of overview frames that have been loaded. The spread-loading algorithm is simple: load the first and last frames, then load the frame that fills the largest remaining gap in the overview (see Figure 5). This method loads frames in several passes of increasing granularity, with each pass adding detail to the overview – much like progressive image-loading that first shows a coarse view of the entire image and then gradually refines it.

The algorithm applies even if the selection of “most important” frames is different. For example, the important frames in a narrative film might be the first frames of each scene; but once these are determined, the spread-loading algorithm continues, selecting the nearest important frame to the center point of each gap. Spread-loading can also be used with click-seeking. Changing the load point could change the loading order in several ways: e.g., immediately loading the frame nearest to the click location, loading more densely around the click, or starting each each granularity pass at the new load point, and then filling other gaps normally.

In summary, spread-loading is a better way to order the delivery of frames in a video overview, so as to preserve the user’s ability to scrub through the entire sequence regardless of how much of the overview has been delivered.

4 COMPARISON STUDY – METHODS

To evaluate the effectiveness of spread-loading for early navigation, we carried out a laboratory study that asked participants to find specific episodes in four-month time-lapse sequences of crop test plots (e.g., Figure 1), using four interfaces that combined two factors – the loading approach (spread-loading or standard linear loading), and whether...
or not the user could click to change the load point (click-seeking). Different time-lapse videos (~100,000 frames, taken 1/min. during daylight hours) were used with each interface (counterbalanced so that each UI evenly used each video).

Study Conditions
We tested all four conditions from our two factors (loading order and availability of click-seeking).

**Loading Order (Linear vs. Spread)**. Linear loading delivers frames sequentially from the current load point, and is the standard method seen in current video players. Spread-loading delivers frames as described in the previous section (i.e., first and last frames, then whichever frame is at the midpoint of the next largest gap).

**Click-Seeking (On vs. Off)**. As described above, click-seeking allows users to click to change the load point. For both loading strategies, changing the load point meant that frames started loading from the user’s click, and continued based on the constraints of the loading strategy. For linear, frames are loaded sequentially from the new load point until the end of the video is reached (or the user clicks again), and the load point then moves back to the leftmost gap. For spread-loading, we used a simple version of the approach described above: clicking meant that the next frame would fill the gap to the right of the new load point, but then continue to fill the other gaps as normal.

Participants and Apparatus
Sixteen participants (4 female, 12 male), ages 17-45 (mean 28.1), were recruited from a local university. All were familiar with desktop systems and streaming video. The study took 60 minutes, and participants were given a $10 honorarium.

Study software was written in Javascript using the P5 graphics library (p5js.org), and ran in a web browser with a local Node.js back end that both served the web page and delivered the video frames. All frames were stored on the local disk, and network delay was simulated (see below). The study used the Chrome browser maximized on a 19-inch 1080p monitor, and used a Windows 10 PC with an Intel Core-i7 processor and onboard Intel graphics. All input was received through the standard mouse and keyboard, and the system recorded all performance data. Questionnaires used a separate web page.

*Simulated Low-Bandwidth Network*. All UI conditions used a local webserver; therefore, we simulated a slow network in the Node.js server. When a request for an image was received, it was queued for a set time delay, and then delivered normally. Latency approximated a slow mobile network (small images: 125ms/image (8fps); medium images, 335ms/image (3fps); full-size images, 625ms/image (1.6fps)).

![Figure 6: From top: small, medium, and high-resolution tabs. Green bars in timeline indicate which frames are loaded. Filmstrip overview is below timeline.](image)

**Study Procedure and Tasks**
Participants were given an overview of the study, and then given an informed consent form and a demographic questionnaire. We randomly assigned participants to an order group (using a Latin Square design) and introduced them to the task and the first interface that they would be using. Participants then completed two practice tasks using a different time-lapse dataset, and then six test tasks with each UI.
For each task, participants clicked a “Start” button to begin the trial, and were shown a target frame on another monitor (visible throughout the trial); their job was to locate that scene in the time-lapse sequence using the current interface (Fig. 6). Targets were chosen to involve important events in the agricultural development of the crop in the images – for example, the point at which plants emerge, grow large enough to touch the next row, start flowering, produce pods, or “lodge” (bend over late in the season). Most of the events were in the first half of the sequences, but we always included one near the end as well.

For all tasks, the UI showed three tabs corresponding to three different image resolutions: small (about $\frac{1}{3}$ resolution), medium (about $\frac{1}{3}$ resolution), and full resolution. All frames were scaled to 1280x760 for display on the screen. Participants could scrub through the sequence by moving the cursor left or right; if not all frames were loaded, then scrubbing showed the closest frame to the mouse cursor. Participants could also change the load point (if available) by clicking the left mouse button with the cursor on the timeline. In all views, a thumbnail filmstrip was shown below the timeline as a reference (64x48 images, no load delay).

Users could switch to a higher-resolution view by pressing the right button – this would load a subset of the sequence in the higher-resolution view, centered at the user’s click location (see Figure 6). The small view always showed an overview of the entire sequence (four months); the medium view showed about three days; and the large view showed about two hours. All views showed 129 frames, sampled evenly across the desired time period. To move back to a lower-resolution view, users could click on the tab for that view or press the Escape key. Once the user had navigated the large view to a frame that was within 30 frames of the target in either direction, a red bar appeared in the timeline, and the user clicked on a “Finished” button to end the trial (similar blue markers were provided in the other views to show that the user was close to the target). There was a 90-second limit on each trial (note that the small overview always loaded in less than 20 seconds).

Loading of the frames for the small view began as soon as the target frame was shown to the participant.

Study Design and Hypotheses

The study used a repeated-measures design with two factors: Load Technique (Linear or Spread) and Click-Seeking (On or Off). Order of the two factors was counterbalanced, but with both Click-seeking conditions always done within the same Load Technique. Each of the four interfaces was therefore seen equally in each position of the order. We collected dependent measures about performance and subjective experience: task completion rate (i.e., rate of successful completion within the 90-second limit), task completion time (from appearance of the target to clicking on the “Finished” button), the number of clicks in each tab of the interface (small, medium, and large views), the amount of time spent scrubbing in each tab, and the total time in each tab. For perception of effort, the dependent measures were the responses to the TLX-style questionnaire.

Our main hypotheses were that participants would:

- H1: complete tasks faster using spread-loading compared to linear loading;
- H2: complete tasks faster with click-seeking;
- H3: rate spread-loading as requiring less effort;
- H4: rate click-to-seek as requiring less effort;
- H5: prefer spread-loading and click-seeking.

Activity measures at each level were used to explore differences in completion time, perceived effort, and preferences.

5 RESULTS

No data were removed as outliers. We report effect sizes for significant RM-ANOVA results as general eta-squared $\eta^2$ (considering .01 small, .06 medium, and >.14 large [8]). For all followup tests involving multiple comparisons, the Holm correction was used.

Task Completion Rate and Time

Task completion rate. RM-ANOVA showed a significant main effect of Load Technique on completion rate ($F_{1,15}=71.67$, $p<.001$, $\eta^2=0.44$), and a main effect of Click-seeking ($F_{1,15}=5.34$, $p<.05$, $\eta^2=0.065$); but no interaction ($F_{1,15}=2.96$, $p=0.11$). As seen in Figure 7, more tasks were completed with Spread-Loading (mean 88%) than with Linear Loading (mean 50.8%).

Completion time. Results for completion time mirror those for completion rate. RM-ANOVA showed significant main effects for both Load Technique ($F_{1,15}=40.0$, $p<.001$, $\eta^2=0.45$) and Click-seeking ($F_{1,15}=7.22$, $p<.001$, $\eta^2=0.046$), again with no interaction ($F_{1,15}=0.75$, $p=0.40$). As seen in Figure 7, Spread-Loading was substantially faster than Linear (44.2s vs. 69.4s).

Time at each level of detail. To compare the time spent at each level (low, medium, and high) we carried out an RM-ANOVA to look for interaction between Level of Detail and our main factors. RM-ANOVA showed a significant interaction between Load Technique and Level ($F_{2,30}=40.0$, $p<.001$, $\eta^2=0.09$), but not between Click-seeking and Level ($F_{2,30}=92$, $p=.41$, $\eta^2=0.45$). As shown in Figure 8, users of linear loading spent substantially more time in the medium-resolution view, which was not the case for spread-loading.

Navigation Actions

We measured navigation actions in two ways, at each level of detail (see Figures 9 and 10): first, the number of mouse clicks on the interface (in the Click-Seeking interfaces, clicking
moved the load point; in all interfaces it also moved the display point as close as possible to the location of the click; and second, the amount of time spent scrubbing the video.

Number of mouse clicks To determine whether different interfaces led to different clicking behaviour, we carried out a 2x2x3 RM-ANOVA (Load Technique x Click-Seeking x Level of Detail). The ANOVA showed significant main effects on the number of click actions for all three factors: for Load Technique, $F_{1,15}=22.7$, $p<.001$, $\eta^2=0.12$; for Click-Seeking, $F_{1,15}=28.0$, $p<.001$, $\eta^2=0.19$; and for Level of Detail, $F_{2,30}=19.5$, $p<.001$, $\eta^2=0.13$. (A main effect of Click-Seeking was expected given that it involves the mouse).

However, there were also interactions between each pair of factors, as well as a three-way interaction (all $p<.001$). As shown in Figure 9, the interactions primarily arise from the large differences between the number of navigation actions in the Linear-Click condition: people clicked more than four times as often in this interface than in the others, and clicked particularly often in the medium-detail and full-size views. The availability of click-seeking, however, did not always lead to substantially more clicks – as shown in Figure 9, the number of clicks was not substantially higher when click-seeking was turned on in the Spread interface.

Time spent scrubbing RM-ANOVA showed effects on scrubbing time for all three factors: for Load Technique, $F_{1,15}=18.5$, $p<.001$, $\eta^2=0.11$; for Click-Seeking, $F_{1,15}=27.1$, $p<.001$, $\eta^2=0.07$; and for Level of Detail, $F_{2,30}=11.3$, $p<.001$, $\eta^2=0.20$. There were also interactions between Load Technique and Click-Seeking ($F_{1,15}=5.81$, $p<.05$, $\eta^2=0.02$), and between Load Technique and Level ($F_{2,30}=25.0$, $p<.001$, $\eta^2=0.10$). As shown in Figure 10, both Linear conditions had substantially higher scrubbing time in the low and medium views (particularly with click-seeking), whereas scrub time for Spread was similar across detail levels regardless of click-seeking.

Perceived Effort

We used the Aligned Rank Transform [37] with the ARTool package in R to enable analysis of the NASA-TLX responses using RM-ANOVA. Results are shown in Table 1 and Figure 11. For all TLX measures, there were strong main effects of Load Technique, with Spread loading seen as requiring less effort (mental, physical, and temporal), allowing greater success, and causing less frustration, than Linear loading.
were all significant. At the end of the study session we asked participants which they preferred overall. Participants strongly preferred click-seeking (faster: 15 of 16; more accurate: 15 of 16; better overall: 14 of 16). Chi-squared tests also strongly preferred click-seeking (faster: 15 of 16; more accurate: 15 of 16; better overall: 14 of 16). Only one measure showed an effect of Click-Seeking (having click-seeking on was seen as allowing greater success). There were no other likely candidates still to be loaded.

Table 1: TLX RM-ANOVA results (ART-transformed [37])

<table>
<thead>
<tr>
<th>Load Technique</th>
<th>Click-Seek</th>
<th>Interaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mental</td>
<td>F_{1,45}=15.0, p&lt;.001</td>
<td>F_{1,45}=1.52, p=.22</td>
</tr>
<tr>
<td>Physical</td>
<td>F_{1,45}=12.3, p&lt;.005</td>
<td>F_{1,45}=1.76, p=.19</td>
</tr>
<tr>
<td>Temporal</td>
<td>F_{1,45}=11.6, p&lt;.005</td>
<td>F_{1,45}=0.04, p=.90</td>
</tr>
<tr>
<td>Success</td>
<td>F_{1,45}=41.5, p&lt;.001</td>
<td>F_{1,45}=3.15, p=.08</td>
</tr>
<tr>
<td>Difficulty</td>
<td>F_{1,45}=11.7, p&lt;.005</td>
<td>F_{1,45}=1.95, p=.17</td>
</tr>
<tr>
<td>Frustration</td>
<td>F_{1,45}=33.3, p&lt;.001</td>
<td>F_{1,45}=1.29, p=.26</td>
</tr>
</tbody>
</table>

Preferences

At the end of the study session we asked participants which condition they felt was faster, which was more accurate, and which they preferred overall. Participants strongly preferred spread-loading to linear (15 of 16 rated it as faster, 13 of 16 as more accurate, and 14 of 16 as better overall), and also strongly preferred click-seeking (faster: 15 of 16; more accurate: 15 of 16; better overall: 14 of 16). Chi-squared tests were all significant.

6 DISCUSSION

Our evaluation of spread-loading provides several findings:

- Participants were significantly and substantially faster when using spread-loading compared to linear loading;
- Participants used click-seeking far less with spread-loading than they did with linear loading;
- Participants spent significantly less time scrubbing the video with spread-loading than with linear loading;
- Participants rated spread-loading as requiring significantly less effort than linear loading, and strongly preferred both spread-loading and click-seeking.

In the following sections, we consider possible explanations for these results, look at how the idea of spread-loading can be generalized to other contexts, and discuss limitations and directions for future research in this area.

Explanations for Results

Faster navigation with spread-loading. Finding target scenes was on average almost 30 seconds faster with spread-loading than with linear loading. There are several possible reasons for this difference. In some cases, spread-loading allowed people to come close to the target frame early in the loading process and make an early navigation decision – this was the intended benefit of the technique. However, the difference between the loading techniques was larger than expected, particularly given that all but one of the targets were in the first half of the video (which should lessen any advantage of spread-loading). Our observations suggest that people may have delayed their navigation decisions with linear loading in order to see if there were other possible candidates yet to come. Spread-loading, in contrast, better illustrates the temporal ‘big picture’ of the time-lapse sequence. Thus, being able to see the state of the crop development across whole summer may have given participants confidence that there were no other likely candidates still to be loaded.

Click-seeking primarily used with linear loading. Users clicked about 46 times with Linear+Click, compared to about 17 times with Spread+Click. We believe that participants were clicking in the Linear UI to try and gain a better understanding of what was happening at different points in the sequence – something that spread-loading already does naturally. In addition, clicking in the spread-loading UI has substantially less effect than it does with linear loading – it only changes the start point of each granularity pass, and so only changes the loading of a few frames in each pass. In contrast, click-seeking with linear loading can dramatically speed up access to a particular part of the timeline. We note that with Linear+Click, participants could have approximated spread-loading, by clicking at regular intervals to gain a quick overview – but they did not appear to do so (and click-seeking overall did not speed up the task).

Amount of time spent scrubbing. Participants spent more time scrubbing the video with linear loading – 12 seconds on average vs 8.2 seconds for spread-loading. We believe that the main reason is simply that people spent more time looking for the target with linear loading, and since looking through the video requires scrubbing, they spent more time scrubbing as well. Our observations suggest that people scrubbed back and forth almost continuously to look for a target – suggesting that they may have been using a rough pattern-matching strategy rather than carefully inspecting specific regions of the overview.

Subjective effort ratings. Overall, the TLX ratings followed the performance data, and it is likely that participants’ better completion times with spread-loading led them to rate it...
as lower effort and higher success than linear loading. One result of note is that click-seeking did not substantially improve effort ratings for linear loading (except for perceived success). This was surprising – we believed that click-seeking would address some of the limitations of linear loading, but it did not, either in performance or perceived effort.

Overall preferences. Participants strongly preferred spread-loading, which was expected given the clear performance benefits of this technique. However, it was interesting to see that most participants also preferred click-seeking, even though it had little effect on performance. Even with spread-loading, click-seeking may give users a sense of control in situations where a slow network makes navigation difficult.

Generalization

The value of spread-loading is directly related to requirements for image size and overview density, and the network speed. The need for large overviews with substantial visual detail and slow rural networks are characteristic of the agricultural research projects that motivated our work – but several other combinations of these factors will also lead to advantages for spread-loading. There is a point, however, at which linear and spread-loading techniques converge – because they behave the same once all of the frames are loaded. Nevertheless, it is worth considering spread-loading as the default delivery order for video overviews – assuming that the goal is to give a high-level understanding of the video content, then giving access to the full time span of the video is likely still a good idea. When overviews load quickly, there is no difference between spread and linear loading; but when network traffic is high, the technique can provide a noticeable advantage for early navigation.

We believe that spread-loading can also be adapted from time-lapse contexts to other types of video. The main difference is to determine what are the important frames that should be loaded first – we used uniform sampling, but as described earlier, there are several non-uniform techniques that provide an optimal set of frames based on the content of the video (e.g., [5, 31]). In future work we will explore the use of these alternate frame-selection techniques with other video types – it may be that the substantial changes between frames in narrative videos will make scrubbing more difficult, but some research suggests that people can successfully pattern-match even in a rapid serial presentation of images [9]. In addition, refinements in frame selection and loading order could also benefit time-lapse sequences as well – for example, if there are more events of interest in the first half of a video, spread-loading could prioritize the loading of frames from this region.

Finally, although we used sequences of individual pictures as frames in our study, spread-loading can also work with standard video formats (which are already configured to allow random access to specific frames). For example, MPEG formats encode individual frames either as I-frames (which contain all the information needed to decode the image), and P- and B-frames (which are encoded relative to preceding or following frames). I-frames are used as reference points for reconstructing other frames, and are therefore "clean random access points" into the video [15]. Typically, every 15th frame of a video is an I-frame. Spread-loading would therefore work with streaming video formats by sending only I-frames until the temporal resolution of the overview (i.e., the gap size) is reached; at this point, P- and B-frames would be sent to fill the gaps.

Limitations and Future Work

There are several ways that our research can be extended in future studies. First, we did not consider static overview representations such as Swifter [24] or Panopticon [16], and it would be interesting to see whether the technique can be applied to a grid of thumbnails. Second, we plan to test spread-loading with other types of video in addition to time-lapse sequences – such as narrative movies or life-logging records [13], with the adaptations described above. To explore spread-loading in a variety of real-world contexts, we are currently building a mobile video player that has spread-loading as the default overview mechanism. Third, we plan to investigate how spread-loading supports other navigation tasks: for example, we believe that spread-loading should provide good support for revisitation of remembered scenes. Fourth, future studies will compare (and possibly extend) spread-loading to other summarization techniques.

7 CONCLUSIONS

Video navigation is often accomplished through scrubbing; but when networks are slow or images are large, an overview or a low-resolution version of a video may load slowly, limiting the usefulness of scrubbing. We introduced a novel frame-loading technique called spread-loading that enables scrubbing regardless of how many frames have been delivered. Spread-loading orders frame delivery to maximize coverage of the entire sequence, and always provides a temporal overview of the entire video that can be navigated at any time during delivery. Our comparison study showed that people were able to find target episodes significantly faster with spread-loading, regardless of whether they could click to change the load point. Users rated spread-loading as requiring less effort, and strongly preferred the new technique. Our work shows that adapting delivery order to navigation needs can have a large effect on performance and usability.

ACKNOWLEDGMENTS

Supported by the Canada First Research Excellence Fund and the Plant Phenotyping and Imaging Research Centre.
REFERENCES


