# Interactive Search in Video: Navigation With Flick Gestures vs. Seeker-Bars

Klaus Schoeffmann, Marco A. Hudelist, Bonifaz Kaufmann, Kevin Chromik

Klagenfurt University, Universitaetsstr. 65-67, 9020 Klagenfurt, Austria
{ks, marco}@itec.aau.at, bonifaz.kaufmann@gmail.com, kchromik@edu.aau.at

**Abstract.** On touch-based devices such as smartphones and tablets users are accustomed to browse through lists and collections by using flick gestures. For video navigation, however, mobile touch devices still use the seeker-bar interaction concept. In this paper, we evaluate the performance of a flick gesture-based video player in direct comparison to a default video player with seeker-bar navigation for the purpose of interactive search in video. We have developed a special video player on a tablet device and performed a user study with 16 users with two different types of interactive search: target/known-item search and scene counting. Our results show that the flick-based video player is less performant than the default video player in terms of search time, but more efficient in finding target scenes and the preferred interface by the vast majority of tested users.

## 1 Introduction

When users want to navigate in a video they typically use a seeker-bar and scrub to the corresponding temporal position in the video sequence. Almost all video players and video interaction tools use this navigational interaction concept for random access in video [13, 14]. This works fine as long as the video duration is rather small and, hence, the required scrubbing accuracy is rather low. However, if the video has a length of a few hours, scrubbing a seeker-bar becomes inconvenient, especially on small displays, since even small movements result in large temporal jumps (e.g., several minutes instead of seconds). Many researchers have focused on how to improve navigation behavior and performance for such situations and proposed improved navigation tools; an overview is given in Section 2. However, in practice these sophisticated navigation means are unfortunately not available to the majority of users, since their devices typically only feature a default video player. Therefore, many users often simply switch to fast-forward and reverse mode when they want to find a specific scene in a long video (e.g., for *known-item search* tasks [10, 11]), or use the available seeker-bar for navigation.

With mobile touch devices such as smartphones and tablets, a very natural and intuitive way of enabling fast-forward and reverse functionality would be a *flick* gesture: The user touches the screen with his/her finger and pushes the video forward or backward. Once pushed to either direction the playback rate suddenly changes (to fast-forward or reverse, depending on the pushed direction) and then linearly returns to normal, except the user performs another flick gesture. Otherwise, if the user flicks again, the fast-forward or reverse speed increases even more. Flicking is a very well-known gesture that is available on most (if not all) smartphones and tablets for browsing lists and collections, and we can assume that users are very accustomed to this type of interaction. However, currently it is not used for controlling the video playback rate, i.e., for interactive search and navigation in video, which is exactly the focus of this work.

The idea of navigating in video through flicking was already investigated in [8]. Unfortunately, from their evaluation we cannot directly deduce whether flick-based navigation in video works worse or better for interactive search tasks than seeker-bar navigation. Their interaction models – called *dynamic/static flicking* and *dynamic/static panning* – were designed for stylus-based interaction on a PDA and did not consider the actual speed of the flick gesture. Instead, they used the distance of the flick gesture as granularity for navigation in video. Moreover, in their evaluation the authors did not compare flick-based navigation to seeker-bar navigation but evaluated flicking against panning and could not find any statistical differences, neither for task performance nor for subjective ratings.

In this paper we investigate the performance of flick gesture-based video navigation for interactive search tasks in videos. For that purpose we have implemented a special video player on an Apple iPad device and performed a user study with 16 users and two different types of tasks: (i) known-item/target search and (ii) specific scene counting. We evaluate the achievable performance of a flick gesture-based player in direct comparison to the one achievable with a common video player using seeker-bar interaction. Our evaluation considers several aspects: (i) achievable search time, (ii) retrieval performance, and (iii) user experience and preference. Our results show that for the majority of tested tasks/videos, users are faster with the common video player but less efficient in finding target scenes than with the flick gesture-based player. At the same time they find the flick gesture-based player more convenient and the better interface than the default video player.

## 2 Related Work

In the last two decades, several proposals for improving video content navigation were made. An overview of recent work in this area can be found in [13, 14]. Here, we only summarize works proposed for mobile devices with touch screen interaction, a topic addressed by only a few authors.

Hürst et al. proposed the *Mobile ZoomSlider* interface [7] for stylus-based navigation on handheld devices. The basic idea is a virtual seeker-bar that can be used at any screen position. The vertical click position is utilized as a parameter for navigation resolution. Moving the stylus left or right results in backward or forward navigation and the vertical position of the drag operation defines how fast the navigation action is performed. The same interaction concept has been proposed for video content navigation on PDAs and smartphones in a follow-up work [8]. The authors further suggested circular navigation gestures for stylus-based navigation in video and presented a few different interaction concepts in [6]. However, unfortunately in the evaluation no direct comparison to seeker-bar navigation was performed.

Karrer et al. [9] proposed the *PocketDRAGON* interface for touch-based video content navigation on mobile devices. Instead of an overlaid seeker-bar, they suggest direct manipulation of objects in the scene. A similar concept was already proposed by Dragicevic for video navigation on desktop PCs in an earlier work [2]. In their work, motion tracking is performed and object motion is used as basis for the dragging operation along a motion trajectory. Additionally to this object-based navigation mode, which rather improves the navigation accuracy and typically does not allow quicker navigation over longer segments in the video, they also support two-finger gestures. A horizontal wipe gesture with two fingers allows to jump to the previous or next scene in the video. However, as no evaluation has been performed by the authors, it remains unclear how well this navigation concept supports navigation tasks in videos.

Huber et al. [4] focused on improving navigation in e-learning videos rather than entertaining videos and proposed the *Wipe'n'Watch* interface. Instead of a seeker-bar, they suggest to use wipe gestures for touch-based navigation on smartphones. Their interface that operates in portrait mode is subdivided into two areas: (i) the upper area shows the actual video content and (ii) the lower area shows an overview of all available keyframes, i.e., available slides that act as direct access points. Their work also targets inter-video navigation, similarly to the idea of the *RotorBrowser*, proposed by De Rooij et al. for desktop use [1]. Hence, vertical wiping allows to jump between semantically similar segments among videos; e.g., topically related segments. The availability of such related segments is indicated with an arrow in the upper right corner of the interface. However, in their evaluation they did not directly compare their approach to seeker-bar navigation. Therefore, it is not clear if this interaction concept is superior than a seeker-bar.

In a recent work [12], we did already investigate navigation performance in video with a multi-touch navigation model that uses different navigation granularity according to the vertical position the wipe gesture is performed. Our results showed that uses like this kind of navigation and – for short videos – were even faster than with the default video player.

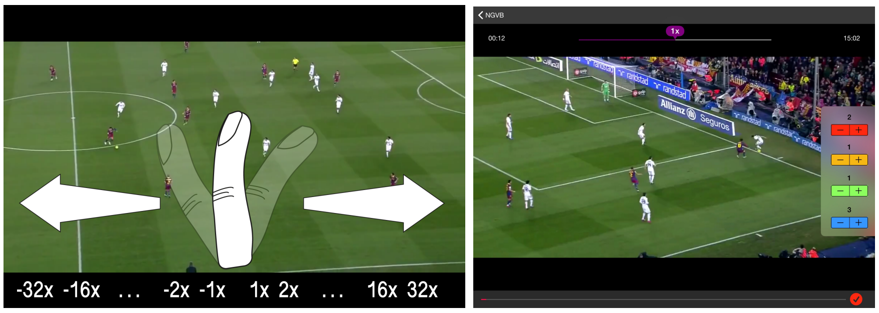
In [5], we proposed the *Keyframe Navigation Tree* interface for touch-based video content navigation on tablets. The main component of its navigation concept are three scrollable horizontal stripes with very compact keyframes, which can significantly outperform seeker-bar navigation at known-item search tasks.

To best of our knowledge, no one has however investigated the navigation performance of a video player using flick gestures, in direct comparison to seeker-bar navigation.

## 3 Video Navigation With Flick Gestures

To evaluate the performance of video browsing/navigation with flick gestures, we developed a new video navigation tool, which we call *flick player*.

### 3.1 Interaction Concept



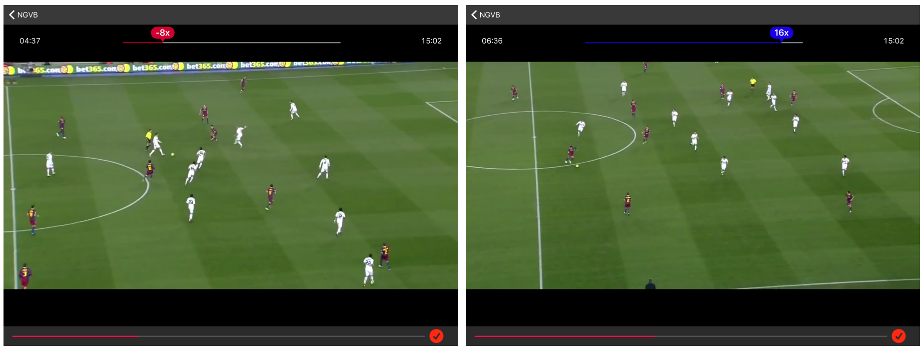
(a) Interaction concept. (b) Interface for *counting tasks*.

**Fig. 1.** The *flick player* allows users to push the video forward and backward, i.e., switch to fast-forward and reverse mode by a simple flick gesture. Playback speed can vary between -32x and 32x. Without any further interaction the player returns to single playback speed (1x), according to a linear easing function.

The interaction concept of the flick player is based on the same idea proposed in [8]: instead of providing a seeker-bar for navigation, we allow the user to flick over the screen with his/her finger (see Figure 1(a)). More precisely, when the user performs a flick gesture to the right the video player switches to fast-forward mode with a playback speed that corresponds to the velocity of the flick gesture (up to a maximum of 32x). If the user performs only one flick gesture, the playback speed will automatically return to normal playback rate (1x) after a while. We use a linear *easing function* for that purpose, which has been selected after several experiments with different easing functions and early test users. If, however, another flick gesture is performed by the user, the momentum of the gesture will be added to the current value of the playback speed. Hence, the fast-forward or reverse rate will be increased even more, until the maximum of 32x is reached. We use the same interaction concept for reverse mode, which can be enabled via a flick gesture to the left.

### 3.2 Implementation Details and Issues

We implemented the flick gesture-based interaction concept on an Apple iPad device with a 9.7-inch screen. Our implementation does always show the current playback rate with a colored line in a dedicated visualization area at the top of the screen. At normal playback rate (1x) a purple bubble is shown in the middle of the line (see top of Figure 1(b)). If the user switches to fast-forward, the bubble becomes blue and moves to the right. Similarly, in reverse mode the bubble becomes red and moves to the left (see Figure 2). Additionally, our implementation shows the current playback time (top left), the duration of the video (top right), and the current position in the video with a non-interactive time-line visualization (bottom).



(a) Interface for *target search* tasks (reverse mode). (b) Interface for *target search* tasks (fast-forward mode).

**Fig. 2.** Our implementation of the flick player also shows the current playback speed (top) as well as the current position in the video (bottom).

Even though we used a recent edition of a tablet device (an Apple iPad Air), its performance was not good enough to decode and play HD video at a speed of 32 times, even not with several optimizations and special video encodings with very small GOP (group-of-pictures) sizes. The only working solution we found was to encode 10 different versions of each video file, each with a different "encoded playback speed" (-16x,-8x,-4x,-2x,-1x,1x,2x,4x,8x, and 16x) and change the real playback rate of the actual video player only in the range of 0.5x to 2x. For example, with the video file version encoded at 8x speed, we can simulate any actual playback rate from 4x to 16x, although the video player uses only a real playback rate from 0.5x to 2x. Hence, in our implementation we use three simultaneous video players that are loaded with three "adjacent instances" (e.g., 4x, 8x, and 16x), where only the video player instance in the middle of this group is currently visible (e.g., with the 8x file version and a real playback rate of 1x). When the user would perform a flick gesture to the right, the flick player would increase the playback rate accordingly to the velocity. If the real playback rate of the video player reaches 2x (actual playback rate of 16x), our tool would switch to the next video player loaded with the 16x speed file version and use it at 1x real playback rate (same actual playback rate). With this instance we can further increase the real playback speed from 1x up to 2x, and simulate an actual playback rate of up to 32x.

This idea is consequently used for both directions (faster and slower playback) and implemented in a way that at any time, all three video player instances are loaded with the correct file version. Our implementation allows a very smooth transition from one video player instance to another, where the current video player is brought to the background and the other one is brought to the foreground, such that the user will not notice this "implementation trick".

## 4 Evaluation

To evaluate the performance of the flick player in direct comparison to a default video player, we conducted a user study with 16 participants (10 men, 6 women; age: 25.38 $\pm$ 3.2 years) that had to solve interactive search tasks in videos. All participants were experienced smartphone users, i.e., had been actively using a smartphone for at least one year. Each participant performed search tasks in four different videos for two different kind of search tasks (see Table 1). For *target search* tasks (also known as known-item search tasks) we presented the target scene of interest to the user and requested him/her to find it as fast as possible with the corresponding tool (flick player or default video player). For *scene counting* tasks we requested the user to find several instances of specific scenes as fast as possible, and increase counter values by using available stepper buttons in the right part of the interface, as shown in Figure 1(b). Such specific scenes were goals, corners, throw-ins, and free-kicks in the soccer videos, and the appearance of specific answers (A, B, C, and D) in the "Who Wants to Be A Millionaire" videos. We used a maximum of four different kinds of such scenes for scene counting tasks, since our interface provides only four counter buttons.

Each participant tested both interfaces (flick player and default video player) with different instances of test videos (however with the same genre). For the selection of tasks and interfaces we followed a latin-square principle to avoid any familiarization effects. For example, if for target search with the flick player the video file *1_news* was used, *2_news* was used for target search with the default video player, and vice versa.

In the following we perform statistical analysis on the collected log data and evaluate the performance of both interfaces in terms of run-time, retrieval efficiency, and user ratings.

### 4.1 Target Search Tasks

**Search-Time** A paired-samples t-test was used to determine whether there was a statistically significant mean difference between the interfaces regarding search time for documentary videos. Data are mean seconds $\pm$ standard deviation, unless otherwise stated. One outlier was detected that was more than 1.5 box-lengths away from the edge of the box in a boxplot. Inspection of its value did not reveal it to be extreme and it was kept in the analysis. The assumption of

**Table 1.** Videos used for the evaluations.

| Video | Genre | Duration (hh:mm:ss) | Search Task |
|---|---|---|---|
| 1_documentary<br>2_documentary | Documentary about nature/animals | 00:50:04<br>00:44:32 | Target Search |
| 1_news<br>2_news | News show | 00:53:14<br>00:53:30 | Target Search |
| 1_football<br>2_football | Excerpt of a soccer match | 00:15:02<br>00:15:00 | Scene Counting |
| 1_gameshow<br>2_gameshow | Excerpt of "Who Wants To Be A Millionaire" | 00:13:38<br>00:15:01 | Scene Counting |

normality was not violated, as assessed by Shapiro-Wilk's test ($p = .08$). The default player ($67.813 \pm 30.259$ secs) was statistically significant faster than the flick player ($128.813 \pm 93.293$ secs) with $t(15) = 3.026, p = .05, d = .756$.

The same test was performed for news videos. No outlier was detected in this case. The assumption of normality was not violated, as assessed by Shapiro-Wilk's test ($p = .835$). The test revealed that the default player ($81.313 \pm 31.33$ secs) was again statistically significant faster than the flick player ($142.313 \pm 60.643$ secs) with $t(15) = 4.608, p = .0005, d = 1.152$.

## 4.2 Scene Counting Tasks

In terms of scene counting we analyzed the data in different aspects.

**Search-Time** In a first step, we investigated whether the required search times to complete a search task for a video type ("Who Wants To Be A Millionaire", soccer) differed between the interfaces. We concentrated first on the "Who Wants To Be A Millionaire" videos. Four outliers were detected, which were more than 1.5 box-lengths from the edge of the box in a boxplot. Inspection of their values did not reveal them to be extreme and it they were kept in the analysis. As normality was violated in Shapiro-Wilk's test ($p = .05$) we used Wilcoxon signed-rank test in this case. A statistically significant median difference was detected between the default video player (median=105 secs) and the flick player (median=148 secs), $z = -2.638, p = .05$.

In case of the soccer videos, two outliers were detected, which were more than 1.5 box-lengths from the edge of the box in a boxplot. Inspection of their values did not reveal them to be extreme and it they were kept in the analysis. Normality was violated in Shapiro-Wilk's test ($p = .05$). Therefore, we again used the Wilcoxon signed-rank test to detect differences. This time, however, no statistically significant median difference could be detect between the default video player (median=219 secs) and the flick player (median=261 secs).

**Retrieval Efficiency** As a next step, we investigated if there are differences in the number of found scenes between the interfaces. In case of the "Who Wants To

Be A Millionaire" videos no outliers were detected, but normality was violated in Shapiro-Wilk's test. Therefore, a Wilcoxon signed-rank test was performed. However, no statistically significant median difference was measured between the video player (median=13 instances) and the flick player (median=14 instances).

When we looked at the data of the soccer videos, a Shapiro-Wilk's test revealed that the assumption of normality was not violated ($p = .406$). Therefore, we decided to use a paired-samples t-test. No outliers were detected. The test showed a statistical significant difference between the default video player (mean=6.938 instances) and the flick player (mean=8.188 instances), $t(15) = 2.825, p = .05, d = .706$.

## 4.3 Questionnaires

In the questionnaires we asked our participants to perform a subjective rating of each aspect of the interfaces, according to the NASA-TLX (Task-Load-Index) method [3]: (i) how mentally demanding was the interaction, (ii) how physically demanding was it, (iii) how hurried users felt when solving the task with the interface, (iv) how hard they had to work to accomplish their level of performance, (v) how insecure, stressed or annoyed they felt when using the interface, and (vi) how successful they felt in accomplishing the tasks. They could choose a value between 0 and 10, where 0 in all cases but for question (iv) represented "very low" and 10 represented "very high". In contrast, for question (vi) 0 indicated "perfect" and 10 indicated "failure".

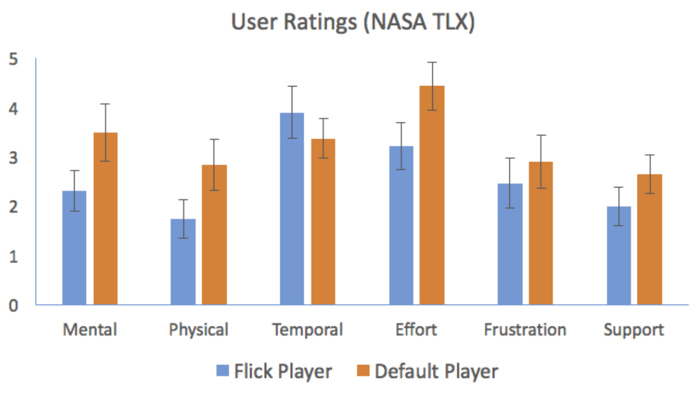


**Fig. 3.** Perceived workload ratings (according to NASA Task-Load-Index [3], with Likert-scale 0-10) for both interfaces (error bars: ± s.e. of the mean)

To determine whether there are statistical significant differences between the interfaces we performed a Wilcoxon signed-rank test for each question. The flick player scored statistically significantly better for questions (i) $z = 2.145, p = .05$,

(ii) $z = 2.419, p = .05$, and (iv) $z = 2.621, p = .05$. In contrast, the default video player performed significantly better for question (vi) $z = 2.224, p = .05$. In all other cases the difference was not statistically significant.

### 4.4 Preferred Interface

We further asked the users to rate both interfaces on a range between 1 and 6, where 1 is best and 6 is worst. The average rating of all 16 users was $2.5 \pm 0.18$ for the default video player, and $1.75 \pm 0.17$ for the flick player, which is statistically significant better. Out of 16 users, 11 users (68.75%) gave a better rating to the flick player, three gave the same rating as to the default video player, and only two gave a worse rating. When asked about the reasons for their ratings, many users mentioned that with the default video player they did not like that they always had to keep their finger on the seeker-bar and move it while watching the content. They much more preferred to just push the video and then concentrate on the playback. Similarly, most users mentioned that they like that less physical interaction is required for the flick player.

## 5 Discussion and Conclusions

We found that users perceived the flick player as less mentally demanding and less physically demanding than the default video player, and felt that the flick player requires less effort for interactive video search tasks in general. More than 68% of the participants rated the flick player as the better interface for interactive video search. At the same time, however, they think that the default video player with a common seeker-bar provides better support for such navigational search tasks in video content.

The evaluations of the run-time and the retrieval efficiency revealed that for three out of four video files users were significantly faster with the default video player for both kinds of tested search tasks (target search and scene counting). However, in terms of retrieval efficiency the flick player performed significantly better, at least for one of the two tested videos (cf. Table 1 and Section 4.2).

Figure 4 shows navigation plots of four different users, which reveal the navigation behavior in the same video file for the two different interfaces over time. In the right-top part of the figure we can see the reason why many users were slower with the flick player: many users did not make use of available high-speed playback rates (e.g., 16x and 32x), instead they mainly used playback rates of 1x to 4x and, hence, required a lot of time for the scene counting task.

With the default video player (left two plots) users could quite quickly navigate over the video and easily count the target scenes by constantly scrubbing the seeker-bar at moderately high speed. Only those users that made use of the available high playback rates of the flick player, such as the one in the bottom right of Figure 4, could achieve a similarly high – or even better – run-time performance.

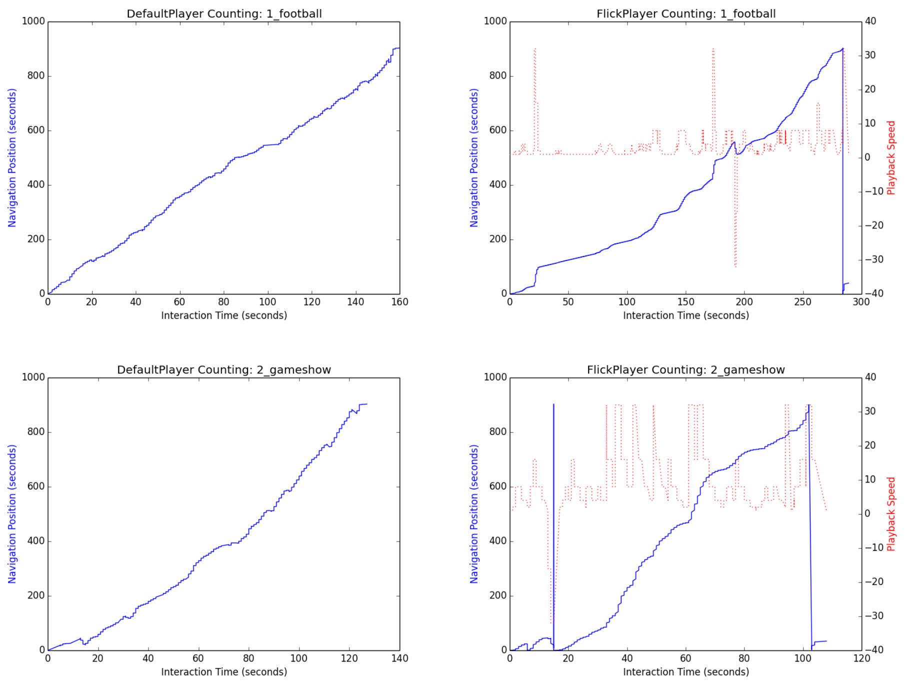


**Fig. 4.** Navigation behavior of four different users for scene counting in two different videos with the default video player (left) and the flick player (right).

The same is true for the navigation behavior at target search tasks (Figure 5), where only users with a high amount of 32x playback rate could outperform users of the default video player (bottom-right in the figure). The top-left part in the figure shows another interesting observation: this user obviously had a rough clue about the temporal position of the target scene in the video, for any reason whatsoever, and navigated very quickly to the last quarter of the video where he/she looked around more carefully. Such a navigation behavior is currently unfortunately not supported by the interaction concept of the flick player.

From these findings we can conclude that there is high potential for video navigation with flick gestures and users seem to welcome such an alternative navigation model (see Figure 3). Also, a flick-based video player can enable higher interactive retrieval efficiency, but with the current interaction model cannot outperform the default video player with seeker-bar navigation in terms of search time. This means that additional studies should be performed, where seeker-bar navigation is used in combination with flick-gestures and/or the flick gesture flexibility is further improved.

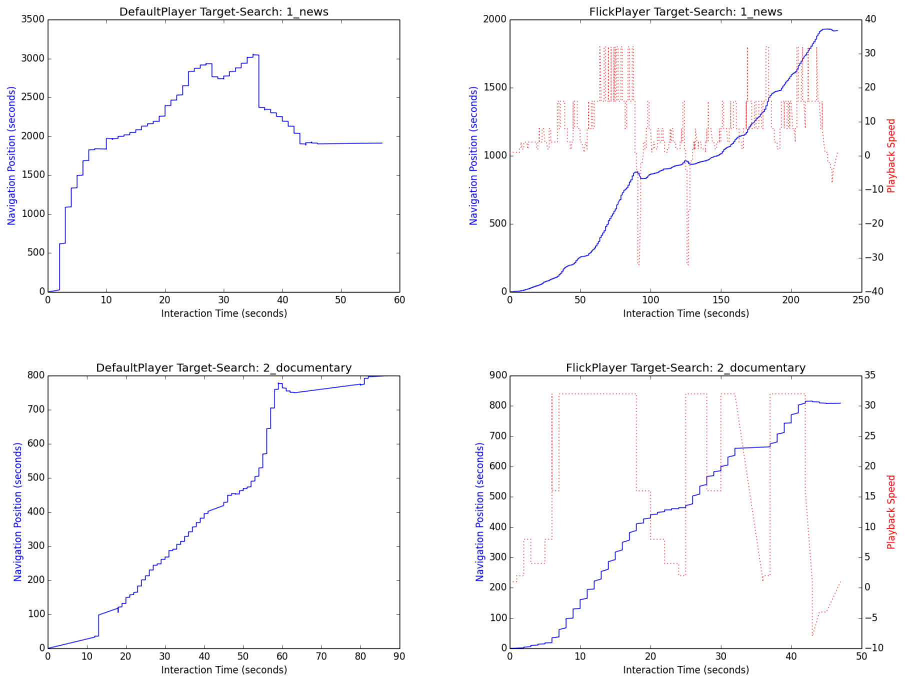


**Fig. 5.** Navigation behavior of four different users for target search in two different videos with the default video player (left) and the flick player (right).

## Acknowledgments

The work was funded by the Federal Ministry for Transport, Innovation and Technology (bmvit) and Austrian Science Fund (FWF): TRP 273-N15, supported by Lakeside Labs GmbH, Klagenfurt, Austria and funded by the European Regional Development Fund and the Carinthian Economic Promotion Fund (KWF) under grant 20214/26336/38165.

## References

1. O. de Rooij, C. G. M. Snoek, and M. Worring. Mediamill: semantic video search using the rotorbrowser. In *Proc. of the ACM Int. Conf. on Image and video retrieval*, pages 649–649. ACM Press, 2007.
2. P. Dragicevic, G. Ramos, J. Bibliowitcz, D. Nowrouzezahrai, R. Balakrishnan, and K. Singh. Video browsing by direct manipulation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 237–246, New York, NY, USA, 2008. ACM.
3. S. Hart and L. Staveland. Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research. In *Human mental workload*, pages 139–183. P.A. Hancock and N. Meshkati (Eds.), Elsevier, 1988.

4. J. Huber, J. Steimle, R. Lissermann, S. Olberding, and M. Mühlhäuser. Wipe'n'watch: spatial interaction techniques for interrelated video collections on mobile devices. In *Proceedings of the 24th BCS Interaction Specialist Group Conference*, BCS '10, pages 423–427, Swinton, UK, UK, 2010. British Computer Society.
5. M. Hudelist, K. Schoeffmann, and Q. Xu. Improving interactive known-item search in video with the keyframe navigation tree. In X. He, S. Luo, D. Tao, C. Xu, J. Yang, and M. Hasan, editors, *MultiMedia Modeling*, volume 8935 of *LNCS*, pages 306–317. Springer International Publishing, 2015.
6. W. Hürst and G. Götz. Interface designs for pen-based mobile video browsing. In *Proceedings of the 7th ACM Conference on Designing Interactive Systems*, DIS '08, pages 395–404, New York, NY, USA, 2008. ACM.
7. W. Hürst, G. Götz, and M. Welte. Interactive video browsing on mobile devices. In *Proceedings of the 15th international conference on Multimedia*, MULTIMEDIA '07, pages 247–256, New York, NY, USA, 2007. ACM.
8. W. Hürst and K. Meier. Interfaces for timeline-based mobile video browsing. In *Proceedings of the 16th ACM Int. Conf. on Multimedia*, pages 469–478. ACM, 2008.
9. T. Karrer, M. Wittenhagen, and J. Borchers. Pocketdragon: a direct manipulation video navigation interface for mobile devices. In *Proceedings of the 11th Int. Conf. on Human-Computer Interaction with Mobile Devices and Services*, MobileHCI '09, pages 47:1–47:3, New York, NY, USA, 2009. ACM.
10. P. Over, G. Awad, M. Michel, J. Fiscus, G. Sanders, W. Kraaij, A. F. Smeaton, and G. Quéenot. Trecvid 2013 – an overview of the goals, tasks, data, evaluation mechanisms and metrics. In *Proceedings of TRECVID 2013*. NIST, USA, 2013.
11. K. Schoeffmann. A user-centric media retrieval competition: The video browser showdown 2012-2014. *MultiMedia, IEEE*, 21(4):8–13, Oct 2014.
12. K. Schoeffmann, K. Chromik, and L. Boeszoermenyi. Video navigation on tablets with multi-touch gestures. In *Multimedia and Expo Workshops (ICMEW), 2014 IEEE International Conference on*, pages 1–6, July 2014.
13. K. Schoeffmann, F. Hopfgartner, O. Marques, L. Boeszoermenyi, and J. M. Jose. Video browsing interfaces and applications: a review. *SPIE Reviews*, 1(1):018004, 2010.
14. K. Schoeffmann, M. A. Hudelist, and J. Huber. Video interaction tools: A survey of recent work. *ACM Computing Surveys*, pages 1–36, 2015. accepted for publication.