Tutorial Mismatches: Investigating the Frictions due to Interface Differences when Following Software Video Tutorials

Raphaël Perraud, Aurélien Tabard, Sylvain Malacria

Tutorial Mismatches: Investigating the Frictions due to Interface Differences when Following Software Video Tutorials

Raphaël Perraud
Univ. Lille, Inria, CNRS, Centrale Lille, UMR 9189 CRIStAL
Lille, France
raphael.perraud@inria.fr

Aurélien Tabard*
Univ Lyon, UCBL, CNRS, INSA Lyon, LIRIS, UMR5205
Villeurbanne, France
aurelien.tabard@univ-lyon1.fr

Sylvain Malacria
Univ. Lille, Inria, CNRS, Centrale Lille, UMR 9189 CRIStAL
Lille, France
sylvain.malacria@inria.fr

ABSTRACT

Video tutorials are the main medium to learn novel software skills. However, the User Interface (UI) presented in a video tutorial may differ from the learner's UI because of customizations or differences in software versions. We investigate the frictions resulting from such differences on a learners' ability to reproduce a task demonstrated in a video tutorial. Through a morphological analysis, we first identify 13 types of "interface differences" that differ in terms of availability, reachability and spatial location of features in the interface. To better assess the frictions resulting from each of these differences, we then conduct a laboratory study with 26 participants instructed to reproduce a vector graphics editing task. Our results highlight interesting UI comparison behaviors, and illustrate various approaches employed to visually locate features.

CCS CONCEPTS

• **Human-centered computing → Human computer interaction (HCI)**; *Empirical studies in HCI*; Graphical user interfaces.

KEYWORDS

software learning, video tutorial, interface distance

ACM Reference Format:
Raphaël Perraud, Aurélien Tabard, and Sylvain Malacria. 2024. Tutorial Mismatches: Investigating the Frictions due to Interface Differences when Following Software Video Tutorials. In *Designing Interactive Systems Conference (DIS '24), July 1–5, 2024, IT University of Copenhagen, Denmark.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3643834.3661511

1 INTRODUCTION

Video tutorials have become one of the main medium to learn new skills [49], with platforms such as Youtube or TikTok playing an important role in their online diffusion [6, 11]. And millions of users are now turning to such tutorials when they need to develop new software skills [39]. Their popularity can be explained by the procedural nature of the tutorials [96]. By sharing detailed steps in context, they support more efficient learning compared to conceptual knowledge sharing [24, 88].

However, differences between the User Interface (UI) presented in video tutorials and the interface available to users can hinder their ability to follow procedures. These differences stem from software versions [70], differences in Operating System (OS), software customizations, or "modular" software in which components can be configured and exposed on demand. Moreover, since video tutorials are time-consuming to create, they are rarely updated. Therefore, they run the risk of having an increasing number of differences with learners' set-ups, as they are produced at a given instant, with specific configurations/customizations, plugins, and languages.

These differences between learner's and tutorial's UIs may lead to *frictions* when a user tries to reproduce procedural instructions of a video tutorial to learn a software. These *frictions* can manifest as challenges in locating and accessing features or commands, misunderstanding visual representations, or experiencing confusion when navigating the interface. Additionally, they may arise from differences in terminology, layout, or functionality between the tutorial and the learner's interface, impeding the learning process.

These frictions can be attributed to a lack of *congruence* between the tutorial and the workspace, a concept crafted by Tversky et al. [83] to describe how "the structure and content of the external representation [here, a video tutorial] should correspond to the desired structure and content of the internal representation [here, the interface to be manipulated]". Tversky et al. showed that lack of congruence is detrimental to learning [83]. For video tutorials, when learners work with different interfaces than the ones used for producing the tutorials [82], they have to undergo an additional *user interface translation* step [72] in order to compensate for the differences that may exist between the two interfaces [44]. And as users often tackle software learning in a task-oriented approach [34, 43, 45, 73], this will either lead to poorer learning performances or to lengthy manual adaptations to align the two interfaces.

In this paper, we first present a morphological analysis in which we identify 13 types of "interface differences". These differences relate to 1) whether a feature used in the tutorial is *available* in the interface or not, and 2) when available, whether it is *directly reachable* or not, or if it requires extra *workspace configuration*. To clarify the impact of these differences, we present a laboratory study conducted with 26 participants instructed to reproduce vector graphics editing tasks. Our results highlight interesting UI comparison behaviors and illustrate various approaches employed to visually locate features in a UI. Finally, we reflect on the best strategies to reduce the *frictions* created by differences in interfaces and inform the design of future video tutorial systems.

*Also with Univ. Lille, Inria, CNRS, Centrale Lille, UMR 9189 CRIStAL.

Authors' version
DIS '24, July 1–5, 2024, IT University of Copenhagen, Denmark

https://doi.org/10.1145/3643834.3661511

2 BACKGROUND

Video tutorials are a popular method to develop new skills [39]. The growth of video content on social networks (Youtube, Instagram, TikTok, Facebook, etc.) has increased the visibility and reach of such tutorials [18, 51]. According to recent estimates, "how-tos" of various sorts, from make-up to software, could account for 1.5% of Youtube 10 billions videos [57]. Despite some drawback in terms of indexation, navigation, and search, video tutorials offer a number of benefits that explain their popularity. Videos enable demonstrations for tasks that are difficult to explain using words [16], they also show context more clearly and tend to be more exhaustive in their description than text-based tutorials.

In learning sciences, procedural knowledge sharing is known to better support learning compared to conceptual knowledge sharing [24, 88]. And video tutorials are procedural in nature [96]. As a medium, they also foster a more pronounced engagement from learners [22, 32]. This means that learners allocate more time to video tutorials than to other types of teaching resources [12, 42, 78]. And more broadly, engagement is often associated to a better learning experience and/or performance [5].

2.1 Benefits of videos for software learning

When it comes to learning software, video tutorials convey dynamics such as user interaction and its effects much better than static resources, which makes them much more popular than text-based tutorials among learners [53, 89]. The video tutorial creation process and its now established conventions (screen given a larger place, audio explanation, shortcuts displayed, video of the teacher often in a corner, and limited editing) also lead to "over-the-shoulder" learning [84]. Moreover, Guo et al. noted that the strong emphasis on visual content makes it easier to overcome languages barriers [31]. These characteristics make videos a media well suited for software learning [37, 86, 87].

2.2 Challenges of video tutorials for software learning

Video tutorials nonetheless suffer from some limitations. One challenge lies in finding the relevant tutorial among an increasing offer. Several projects aim to offer a broader view of available videos, in order to help users choose the most relevant one [27, 28, 48, 91]. But volume is not the only problem. Phrasing *what to learn* is often a challenge. Indeed, users tend to formulate *task-oriented* (their goals) search queries rather than *tool-oriented* ones [28, 39, 45]. However, such goals are not well captured in video meta-data, especially for small tasks part of a larger workflow.

Therefore, once users pick a tutorial, ensuring its relevance often involves getting an overview by navigating through the timeline [23, 41, 66, 92]. Systems such as Video Lens [56], Panopticon [35], LectureScape [40], CodeTube [69] or SceneSkim [65] facilitate skimming and navigation through videos using transcripts or metadata to analyze the content of the tutorial.

Even when a tutorial is thematically relevant, and covers the right task and tools, further complications can arise due to a lack of similarity between the interface in the video and the interface on the learner's computer. Software is malleable [14] and subject to changes, leading to variations in user configurations due to application or OS updates, customization of the interface, localization, themes, etc. While experienced users may handle such differences, in a learning context they create *frictions* we investigate in this article. The literature on the psychology of learning has shown that a difference between what instructions show and what learners have available is detrimental to learning [83]. The idea that *"the structure and content of the external representation should correspond to the desired structure and content of the internal representation"* is what Tversky et al. call the *congruence* principle [83]. Studied initially with learning animations, it also applies to instructional hands-on video demonstrations [9].

2.3 Overcoming interface differences

The problems related to differences between interfaces is well established in the HCI community and has been studied in various settings. It is in the context of transfer learning that the problems of interface differences have been studied most, as the difference can be very important, such as between Gimp and Adobe Photoshop [72]. While domain knowledge from one tool can be useful to conduct tasks in another, differences have an impact on performance. Raissi et al. have shown that even when users alternate between different interfaces frequently, small differences or changes in interfaces can still produce significant drops in performance [71]. To tackle this issue, *Blocks-to-CAD* proposes a cross-application bridge concept for learning new software that gradually changes a familiar application into another one which uses similar interaction paradigms [47]. For its part, *Show-me-how* focuses on better supporting transfer learning between similar software by offering UI translation capabilities [72]. It also proposes in-app search tools that are capable of understanding the terms used in various applications to redirect to the proper feature. Another approach would be to leverage users' awareness of available feature in the interface [25] or their vocabulary [17, 26] in order to be able to overgo by themselves the interface differences.

However, very little has been done to understand the difficulties involved in differentiating between two interfaces of the same application. Alvina et al. [3] studied how some interface concepts could help users overcome issues that arise from interaction paradigm differences (e.g. across mobile and desktop), but this was a design-driven exploration focused on interaction paradigms. We are rather interested in understanding the frictions due to differences in versions or customizations between a tutorial and the interface users have at hand, since they can generate depreciation or a lack of relevance [62, 93].

3 FRAMING THE INTERFACE DIFFERENCES

Software interfaces exhibit a high degree of customization, modularity, and adaptability, which can modify various aspects of the commands comprising an interface. The primary alterations a command can undergo pertain to its representation (the visual instance of the command within the interface), and its integration within the broader application/GUI, defined by its *Reachability* and *Availability*. Changes influencing a command's representation, such as updates to its visual iconography, are deemed infrequent in occurrence. Consequently, we focus on modifications related to the

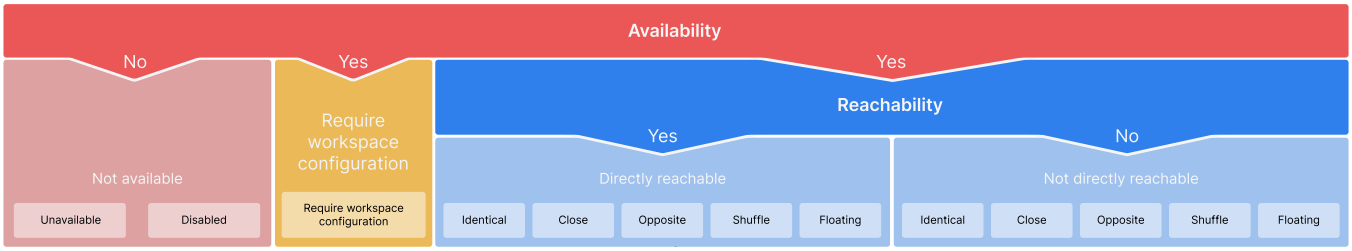

Figure 1: Conceptual model of the dimensions (Reachability and Availability) on which differences can be observed when comparing interfaces through 13 types of differences.

Reachability and *Availability* within the GUI. In this aim, we deliberately omit considerations of *Reachability* modifications attributable to differences in input modalities such as keyboard shortcuts. Our emphasis remains directed towards examining frictions stemming from differences that impact the GUI, that may alter three of the main common challenges that users face when learning a new software: understanding how to perform a task, awareness of tools and features, and locating tools and features [30]. We have therefore defined the concepts of *Reachability* and *Availability* as follows:

Reachability. The *Reachability* of a feature is dependent of its application hierarchy and its interface hierarchy. This former refers to the containerization of commands into semantic entities (e.g. the command relative to align an object to the left border is contained into an "Alignment" parent window, itself contained by "Text"). The latter refers then to the spatial arrangement (location, size, rotation and layer level of the *Representation*) of the command on the interface. The *Reachability* of a command is dependent of the input modalities that can invoke the same functionality through another way (such as keyboard shortcuts).

Availability. The *Availability* of a command in a software defines if an enabled instance of the command exists in the current state of the application. The variance of the *Availability* of a functionality will be mostly due to application versioning or third-party add-ons.

In this paper, we focus on the frictions related to the variations of the *Reachability* and the *Availability* of commands between an interface used in an instructional video and the learner's interface. We have characterized the differences of *Reachability* according to several levels of magnitude of spatial displacement between the interfaces. This magnitude is defined by the spatial distance of a same component between two states of an interface, and by the amount of user actions required to arrange it identically between the two interfaces (see figure 1 for a breakdown of differences). We have therefore defined two categories of differences of *Reachability*.

Features Directly reachable (**DR**) on screen, whose parent component is present and visible on both interfaces (e.g. the "Brush" tool in Affinity Designer's toolbar). Features that are not Not directly reachable (**NDR**) on-screen, whose parent component is present on the screen but not active (e.g. the parameters of the Brush tool, visible by selecting the tool to make those appear on the "contextual bar" located at the top of the workspace). The Directly reachable (**DR**) and Not directly reachable (**NDR**) categories are each divided into five magnitudes of spatial distances (see figure 2):

(1) **identical**: at the exact same location on both interfaces
(2) **close**: the feature is embedded in the same parent container but in another location (displaced within the same panel)
(3) **floating**: the feature is in a floating window, hovering the workspace, and is not related to a parent container
(4) **opposite**: the feature is located on the opposite side of the display in a similar parent container (displaced into a panel which contains other components of similar hierarchy)
(5) **shuffle**: interface component locations are shuffled, the spatial arrangement being heavily modified without modifying its hierarchy

The case Requiring workspace configuration (**RWC**) for a feature to become reachable is apart: the functionality is in fact not reachable by default in the interface, but *available* within the software if activated, for example the parent window from the "Windows" application menu.

Finally, the *Availability* of a feature can vary. It can be available in the application but *disabled* in the current state of the interface (e.g. features that depend on selection in the workspace, or hardware configuration). Or it can simply being Not available (**NA**) (e.g. it requires the addition of a third-part plugin or a different version of the software). Therefore, the following questions arise according to the *Reachability* or *Availability* difference of a feature that can induce frictions to a learner:

RQ1. Which interface difference creates the most friction?
RQ2. What behavior do users adopt to recover the missing feature?
RQ3. In order to minimize *Reachability* differences, should the spatial proximity between the location of a functionality in the tutorial and the workspace be favored over direct reachability on both interfaces?

In the following, we will refer to these conditions by their abbreviations ($\mathbf{DR}_{\text{INSTANCE}}$, $\mathbf{NDR}_{\text{INSTANCE}}$, $\mathbf{NA}_{\text{INSTANCE}}$ and **RWC**).

4 USER EVALUATION

We conducted an in-lab user study to investigate the frictions generated by interface differences. In this study, we chose to use Affinity Designer 2 as experimental test bench for several reasons. First, it is a vector graphics editing software, a type of software for which

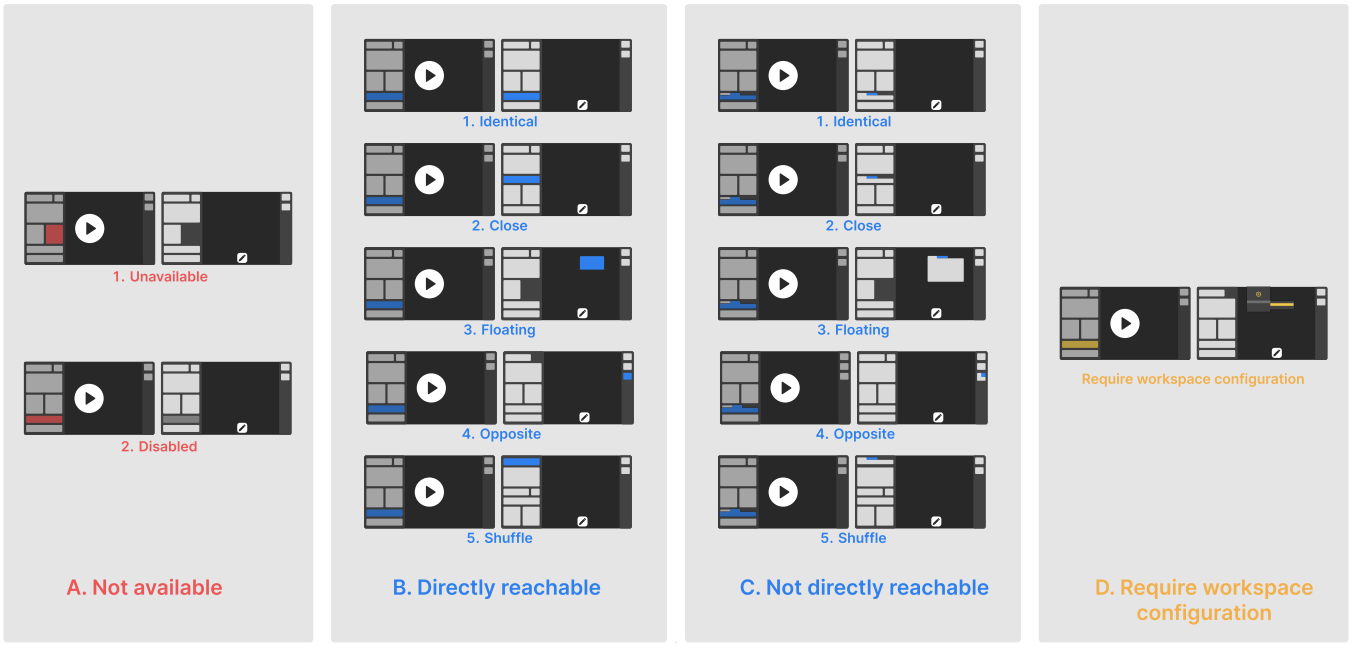

Figure 2: Characterization of the 13 types of differences when comparing a Video Interface to a User Interface.

video tutorials are frequent. Second, it is offering many opportunities to manipulate the interface since its GUI relies on a modular layout in which independent graphical panels can be grouped, detached or displaced. Third, it is not the industry standard, minimizing the risks of recruiting participants that would be strongly familiar with it.

The study consisted in 13 independent tasks to test the 11 levels of *Reachability* and the two different states of *Availability*. Each of the independent tasks consisted in following a dedicated instructional video that demonstrated how to select, activate and apply a feature of the software. These instructional videos were specifically created for the need of this experiment, each illustrating the use of one feature that required to be activated (which also required interacting with the parent container when the feature is not directly reachable) and applied (that is set a unique parameter as demonstrated in the video, e.g. the angle of a rotation) on a specific object of the workspace . These videos were short (9 seconds on average) and had no audio track.

4.1 Experimental setup

We used a dual-screen desktop-like setup, with instructional videos displayed in full-screen on the leftmost display, and the Affinity Designer UI displayed in full-screen on the rightmost display. As we used the same features for both **DR** and **NDR** conditions, we used two different Affinity workspace setups with different windows and panels arrangements for the UI in order to avoid potential learning effects. An experimenter was present to reset the workspace and interface between tasks. After each task completion, an iPad was provided to participants for them to evaluate the following statements through 7-point Likert items:

- **(L1)** *I already knew where the tool/functionality was located on my interface before watching the video.*
- **(L2)** *I located this tool/functionality easily on my interface.*
- **(L3)** *The location of the functionality in the video interface helped me to locate the tool/functionality on my interface.*
- **(L4)** *Once I had located (and revealed) the tool/functionality, I could easily activate it.*
- **(L5)** *Once I had activated the tool/functionality, I could easily apply the tool/functionality in the workspace.*
- **(L6)** *I consider this functionality is coherently integrated with the rest of my user interface.*
- **(L7)** *I consider this functionality is coherently integrated with the rest of the video interface.*
- **(L8)** *The tool/functionality was already visible on the screen before I perform any action on the interface.*
- **(L9)** *If the tool/functionality was not visible on the screen, it was easy to reveal it.* (conditional)

Note that **(L9)** was conditionally displayed if the answer to the question on visibility **(L8)** was negative. Animated gifs explaining specific terms (locate, activate, apply) were displayed next to the Likert item questions to facilitate their interpretation.

4.2 Procedure

Participants were invited to seat at a desk and to first answer demographic questions, as well as indicate their familiarity with video tutorials and vector graphics software. Afterward, an eye tracker was calibrated on the display where participants would manipulate the User Interface (UI Display). A brief demonstration of the Affinity Designer's interface followed, showing how to select and manipulate objects in the workspace, apply basic effects, and draw basic shapes. Participants understanding of these basic principles was

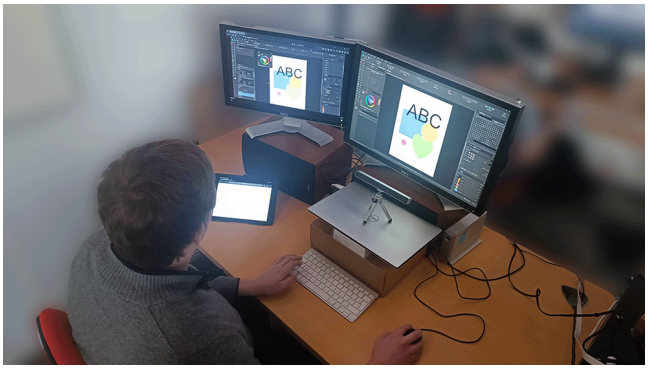

Figure 3: Illustration of the experimental setup.

then verified by asking them to draw and modify a rectangle. Next, they progressed through the 13 instructional videos (presented in random order), with the instruction to replicate the demonstrated tasks as closely as possible. Participants were free to interact with the videos and the interface. There was no restriction on video watching. Each task had a 5-minutes time limit, the experimenter interrupting the participants if they had not completed the task by that time. Participants could also give up on the task anytime if they decided they would not be able to complete it in less than 5 minutes. After that, a new instructional video was loaded automatically, and participants could begin the next task at their discretion. At the end of the experiment, an optional debriefing session was scheduled for participants wishing to look back on the experiment and discuss the strategies they used to resolve the frictions they encountered.

4.3 Participants and apparatus

The experiment was conducted on a desktop PC computer running Affinity Designer 2 under Windows 11. The video display was a Dell UltraSharp 2007WFP of 44x27.5cm (1650x1050px), while the UI display was a Dell 2408WFP of 52x32.5cm (1920x1200px). Participants interacted with an external keyboard and mouse, while an EyeTribe eye tracker recorded their eye movements at 60Hz. User interactions with the instructional video as well as mouse events on the UI display were logged using a Python script using Pyinput [64]. The post-task completion questionnaire was displayed and answered to on an iPad (6th generation), displaying a survey implemented in Quasar [1], a Vue.js framework, synchronized with the computer via socket communication. We recruited 26 participants aged 23 to 50 (M=32, SD=7.6). All but two participants were new to Affinity Designer. Four participants reported never watching video tutorials, and five to be unfamiliar with vector graphics software.

4.4 Experimental design

This study aims to assess the frictions resulting from varying degrees of feature displacement affecting *Reachability* and different interface states affecting *Availability*. We evaluated the five instances of Directly reachable (**DR**) and Not directly reachable (**NDR**) differences outlined in Section 3, along with the Requiring workspace configuration (**RWC**) and two instances of *Availability* difference independently.

Tasks for Directly reachable (**DR**) and Not directly reachable (**NDR**) conditions each utilized the same feature presented in different interface setups, chosen for similar mental workload and interaction levels: rotating an element, applying visual effects, text styles, indentation, and opening a snapshot. These features were selected for their lack of prominent visual elements that could serve as interface landmarks (unique color, recognizable sign [54, 85] and potentially skew the search and location process.

Our primary dependent variable is interface difference, comprising of 13 types of differences presented separately to participants across 13 instructional videos.

In summary, we conducted a within-subject design experiment with 26 participants, resulting in 338 records and completed questionnaires, with an average completion time of 33 minutes.

5 RESULTS

5.1 Data processing and analysis approach

We collected data from mouse and keyboard inputs, interactions with the web video player using DOM events and gaze data from the eye tracker. We also conducted a video analysis from our screen recordings, and illustrate some of our observations with participant's verbatim to clarify their thinking. As mouse pointer coordination with gaze lead has been largely depicted in the literature [8, 19, 33, 58, 61, 74, 80], we considered both gaze and mouse movements in our analysis related to reaching a target and locating user's attention. We structured our results by themes presented in our research questions to triangulate our findings and conclusions.

For time analysis, we conducted one-way repeated measures ANOVAs, with Greenhouse-Geisser corrections applied to the degrees of freedom when sphericity was violated. For pairwise comparisons, we used a Bonferroni correction in which measured p-values are multiplied by the number of comparisons, keeping 0.05 as significance threshold.

We analyzed Likert items by first converting each answer to its equivalent on a $(-3, 3)$ integer scale, and then using Friedman tests, followed by paired Wilcoxon signed-rank tests with Bonferroni correction for pairwise comparisons (except for **(L9)** where Kruskall-Wallis and a post-hoc Dunn's test were used due to independent groups). In the following, graphical representations aggregate data by participant and by task and depict 95% bootstrapped confidence intervals using SciPy package [2] for all plots.

5.2 Overall tutorial completion

workspace configuration and floating conditions hindered tutorial completion the most. On average, participants successfully completed 89.8% of the 11 tasks where the target feature was *reachable*, either Directly reachable (**DR**), Not directly reachable (**NDR**) or Requiring workspace configuration (**RWC**). However, three conditions had relatively low success rates: **RWC** (61.5%), which required workspace reconfiguration, as well as $\mathbf{DR}_{\text{FLOATING}}$ (80.8%) and $\mathbf{NDR}_{\text{FLOATING}}$ (73%), which positioned the target feature in a floating window. Other *Reachability* conditions had success rates above 92.3%. Unsurprisingly, *Availability* conditions had a much lower success rate of 25% (15.4% and 34.6% for $\mathbf{NA}_{\text{UNAVAILABLE}}$ and $\mathbf{NA}_{\text{DISABLED}}$ respectively), easily explained by the fact that it required to rely on an alternative workflow not demonstrated in the

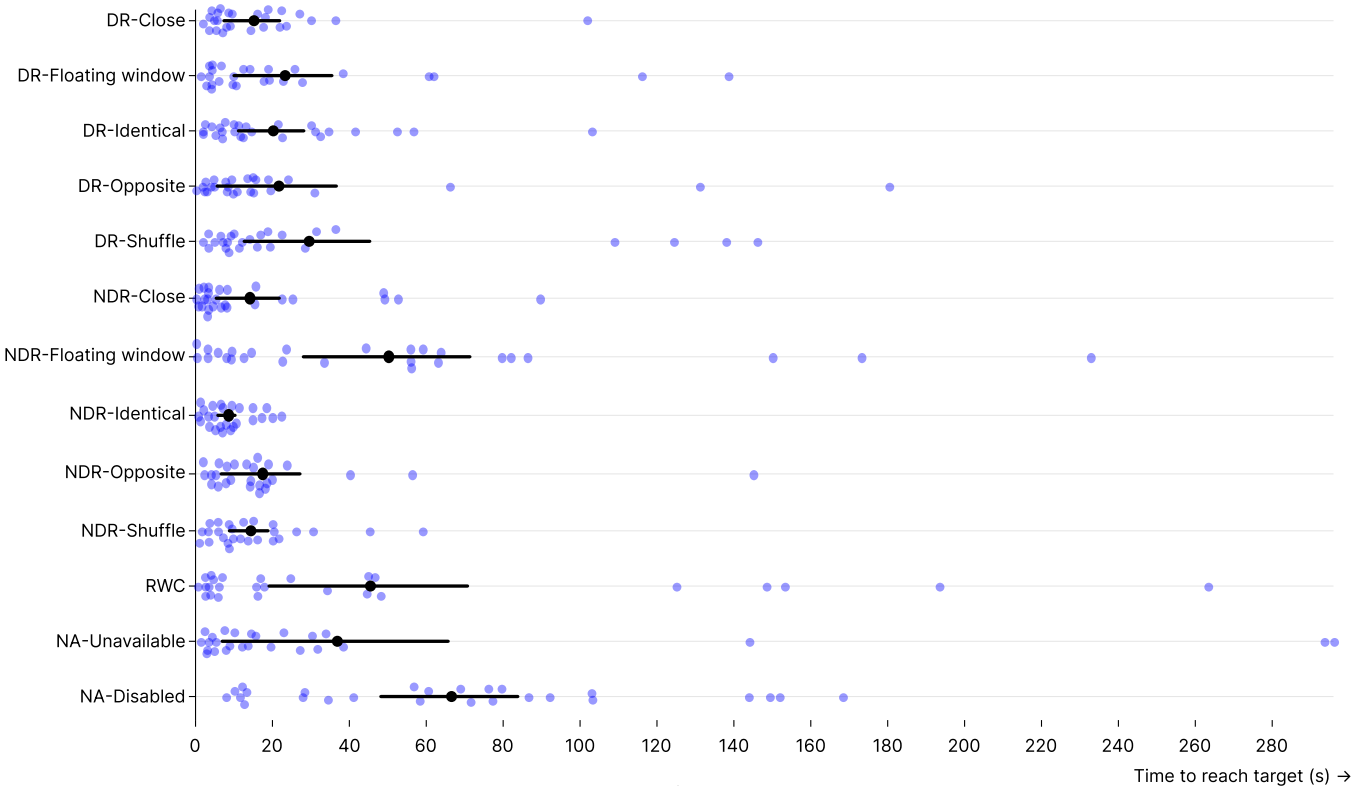

Figure 4: Time to visually locate target features by type of interface difference

instructional video. Results obtained for $\mathbf{NA}_{\text{UNAVAILABLE}}$ is explained by the fact that 4 participants more experienced with vectors graphics software were able to use other features to achieve a similar result and complete the task.

5.3 Locating the target feature in the UI

5.3.1 Time to visually locate the feature. The time taken to locate the feature visually was longer in the $\mathbf{NA}_{\text{DISABLED}}$ condition than in most other conditions. We estimate the time to locate the target feature T_{locate} as the time between when the feature was played in the tutorial and when the participant's gaze fixed, or pointer clicked, in the vicinity of the target feature in the user interface, based on the average diagonal length of a feature panel (Color, Swatches, Stroke, etc. also called *windows* in Affinity designer) of the interface when displayed on the Dell 2408WFP, which was 3.9cm. When the participant could not find the feature, we counted the time from when the feature was played in the tutorial to the point of abandonment or the end of the 5-minutes time limit, depending on the case. We found a significant effect ($F_{7.91,197.8} = 7.35, p < .001$) of Task on T_{locate}. Unsurprisingly, $\mathbf{NA}_{\text{DISABLED}}$ (67.0s) took significantly longer (since participants either abandoned or waited the 5-minutes limit) than all others conditions, except $\mathbf{NDR}_{\text{FLOATING}}$ (51.7s) and **RWC** (47.4s). For its part, $\mathbf{NDR}_{\text{FLOATING}}$ was found significantly different from $\mathbf{NDR}_{\text{IDENTICAL}}$ (8.6s) and $\mathbf{NDR}_{\text{CLOSE}}$ (14.8s). We did not find any other significant difference.

5.3.2 Perceived easiness to locate the target feature. Visually locating the target feature was perceived as more difficult when unavailable, or displayed in a floating window and not directly reachable. Indeed, we found a significant effect ($\chi^2(11, n = 26) = 127.7, p < 0.001$) of Task on the reported easiness to locate the target feature in the interface (**L2**). Pairwise comparisons revealed that participants perceived more difficulties in the $\mathbf{NDR}_{\text{FLOATING}}$ task (Md= -2.0, SD=1.8) than in all others, except $\mathbf{DR}_{\text{FLOATING}}$ and $\mathbf{NA}_{\text{UNAVAILABLE}}$. This difficulty to locate the target feature in the floating window is effectively illustrated by P22: *"[I was] looking for it in side panels, but sometimes the component was floating here."* Pairwise comparisons also revealed more difficulties in the **RWC** task (Md= -1.0, SD=2.1) than in the $\mathbf{DR}_{\text{IDENTICAL}}$, $\mathbf{NDR}_{\text{CLOSE}}$, $\mathbf{NDR}_{\text{IDENTICAL}}$, $\mathbf{NDR}_{\text{OPPOSITE}}$ and both **NA** conditions. Unsurprisingly, participants also expressed more difficulties during the $\mathbf{NA}_{\text{UNAVAILABLE}}$ task (Md= -3.0, SD=1.4) than during all others, except $\mathbf{NDR}_{\text{FLOATING}}$.

We also found a significant effect ($\chi^2(11, n = 26) = 161.1, p < 0.001$) of Task on the reported usefulness of the feature location in the video to locate it in the User Interface (**L3**). Pairwise comparisons showed that the location in the video helped more participants to locate the feature in the $\mathbf{DR}_{\text{IDENTICAL}}$ task (Md= 3.0, SD=1.5) than in $\mathbf{DR}_{\text{FLOATING}}$, $\mathbf{DR}_{\text{OPPOSITE}}$, $\mathbf{DR}_{\text{SHUFFLE}}$, $\mathbf{NDR}_{\text{FLOATING}}$, $\mathbf{NDR}_{\text{OPPOSITE}}$, **RWC** and $\mathbf{NA}_{\text{UNAVAILABLE}}$. For its part, the video helped more participants in the $\mathbf{NDR}_{\text{IDENTICAL}}$ task (Md= 3.0, SD=1.4) than all DIRECTLY REACHABLE (**DR**) except $\mathbf{DR}_{\text{IDENTICAL}}$, but also than $\mathbf{NDR}_{\text{FLOATING}}$, $\mathbf{NDR}_{\text{OPPOSITE}}$, **RWC** and $\mathbf{NA}_{\text{AVAILABLE}}$. Participants also reported

leveraging the video more during the **NA**$_{\text{DISABLED}}$ task (Md= 2.0, SD=1.6) than during the FLOATING tasks, **NDR**$_{\text{OPPOSITE}}$, **RWC** and **NA**$_{\text{UNAVAILABLE}}$. Interestingly, participants found the video more helpful to locate the feature in the **NDR**$_{\text{SHUFFLE}}$ task than for the FLOATING tasks, **RWC** and **NA**$_{\text{UNAVAILABLE}}$, confirming overall difficulties to locate the feature when positioned in floating windows.

Finally, we must note a moderately strong positive correlation (Pearson's correlation coefficient $r(24) = 0.65$, $p < 0.001$) between the expectation of the feature location based on its location in the tutorial (**L3)** and the perceived ease of locating it in their UI (**L2)**.

5.3.3 Perception of the immediate visibility of the feature. Floating windows hindered immediate visibility of the feature. We found a significant effect ($\chi^2(11, n = 26) = 147.0$, $p < 0.001$) of Task on the perception of the feature being already visible on screen before to have to interact with the interface (**L8)**. More precisely, pairwise comparisons revealed that participants found **RWC** (Md= -3.0, SD=0.5) and **NA**$_{\text{UNAVAILABLE}}$ (Md= -3.0, SD=0.5) to be significantly different from all other tasks. They also revealed participants found the feature less "already visible" in the **NDR**$_{\text{FLOATING}}$ task (Md= -1.0, SD=2.0) than in the **DR**$_{\text{CLOSE}}$, **DR**$_{\text{IDENTICAL}}$, **DR**$_{\text{OPPOSITE}}$ and **NA**$_{\text{DISABLED}}$ tasks. While this result may not be surprising, since the command is not directly reachable in the **NDR**$_{\text{FLOATING}}$ task but directly reachable in the others, participants' comments once again emphasized the difficulty to consider a floating window as likely to contain the target feature, as noted by P12: *"Oh no, what a trap! [It was] in the middle of my face and I would never have seen it. I mean I didn't know where it was whereas it was visible on the screen, it was totally visible. Well, it's a small tab [...] but it was totally visible"*. Lastly, participants found the feature also less visible "by default" in the **NDR**$_{\text{SHUFFLE}}$ task (Md= 1.0, SD=1.9) than in the **DR**$_{\text{CLOSE}}$, **RWC** and **NA**$_{\text{UNAVAILABLE}}$. Finally, we must note that except **NDR**$_{\text{FLOATING}}$, all NOT DIRECTLY REACHABLE (**NDR**) tasks were rated positively (*Mds* >= 1.0) suggesting that participants tend to consider the parent component rather than the feature itself.

5.4 Quantifying the frictions through steps to apply the feature

We then delved into the steps required to reproduce the tutorial once the feature was located. As previous work has already investigated the problems associated with the interaction between the instructional video and the User Interface [7, 38, 68, 95], we did not study these aspects and instead focused on the steps related to manipulating the learner's interface.

5.4.1 Participant's approaches. Participants expressed difficulties in localizing the feature on their screen, at a higher level, notably because the stacked tabs that did not help them to localize the feature directly, e.g. *"because the colors [of the interface] are the same [...] so I could miss those panels. The other thing is sometimes you can just see the first one"* (P1). Some participants who where not used to customized interfaces indicated looking for something which could reveal more features such as the *"little toothed wheel"* (P19) or a search bar to filter the feature displayed within the interface: *"I looked in settings because the tab wasn't there, [...] I looked in "Search" when I could"* (P15). Others participants more experienced with customizable software mentioned the hierarchy structure as helping to scan the interface in order to find the target feature: *"So I don't think there was any geographical proximity, because there wasn't, but on the other hand I looked for a similar structure with the same text."* (P8) Some also noted that the interfaces presented to them during the experiment lacked a logical structure, which may have hindered their analysis of the interface: *"And that problem wouldn't be there unless there was a feature I never use or, vaguely. I'll remember that 'ah yes! There's a palette that does that. [...] I'd have a sort of logical organization. Yeah, palettes grouped by activity theme, I think. So, the one in the video, I know how to find it because I can see very well what activity it is. Is it more related to text? I'll look for it in my own palette, where I'd put it."* (P14)

5.4.2 Perceived easiness to apply the feature on the workspace. Participants perceived that if a feature was not visible, it was also challenging to reveal it on their UI (**L9)**. We only collected this item in Likert scales if participants initially reported negative visibility of the feature, resulting in an imbalanced dataset. We found a significant effect (H(11) = 61.8, p < 0.001) of Task on the reported easiness to reveal the target feature on the interface (**L9)**. Pairwise comparisons showed that participants found the **NDR**$_{\text{IDENTICAL}}$ task (Md= 3.0, SD=0.5) to be significantly easier from **NDR**$_{\text{FLOATING}}$, but interestingly, also from **DR**$_{\text{FLOATING}}$. Pairwise comparisons also showed that the **RWC** task (Md= -1.0, SD=2.1) was harder to reveal than in the **NDR**$_{\text{IDENTICAL}}$. Lastly, participants found the feature harder to reveal in the **NA**$_{\text{UNAVAILABLE}}$ task (Md= -3.0, SD=1.2) than in the **NDR**$_{\text{CLOSE}}$, **NDR**$_{\text{IDENTICAL}}$, **NDR**$_{\text{OPPOSITE}}$ and **NDR**$_{\text{SHUFFLE}}$.

Part of the challenge to them was to understand the source of interface differences and determine the appropriate action to apply to the layout: *"I was just looking for it... Maybe I didn't find it; or I didn't understand how to make it appear instead."* (P16)

Participants reported difficulty to activate the feature only for NOT AVAILABLE (**NA**) tasks. We found a significant effect ($\chi^2(11, n = 26) = 121.0$, $p < 0.001$) of Task on the reported easiness to activate the target feature in the interface (**L4)**. More precisely, pairwise comparisons revealed that participants found **NA**$_{\text{UNAVAILABLE}}$ (Md= -3.0, SD=2.2) to be significantly different from all other tasks. Pairwise comparisons also revealed more difficulties in activating the **NA**$_{\text{DISABLED}}$ task (Md= 2.0, SD=2.2) than in the **DR**$_{\text{IDENTICAL}}$, **NDR**$_{\text{CLOSE}}$ and **NDR**$_{\text{IDENTICAL}}$.

We also found a significant effect ($\chi^2(11, n = 26) = 145.7, p < 0.001$) of Task on the reported easiness to apply the target feature in the interface (**L5)**. Once again, pairwise comparisons revealed that participants found **NA**$_{\text{UNAVAILABLE}}$ (Md= -3.0, SD=2.1) to be significantly different from all other tasks. Pairwise comparisons also revealed more difficulties in activating the **NA**$_{\text{DISABLED}}$ task (Md= -0.5, SD=2.1) than in the **DR**$_{\text{CLOSE}}$ and **NDR**$_{\text{IDENTICAL}}$., **DR**$_{\text{OPPOSITE}}$, **NDR**$_{\text{CLOSE}}$ and all NOT DIRECTLY REACHABLE (**NDR**) except **NDR**$_{\text{FLOATING}}$.

5.5 Workarounds adopted to localize and activate the feature

5.5.1 Instructional video usage. Participants did few backjumps in the video when the target feature was available. The median number of backjumps, representing the number of time users reverted in the video, ranged from 1 to 3 across tasks. All NOT DIRECTLY REACHABLE (**NDR**) tasks required 1.0 backjumps (SD<=1.0), as well as **DR**$_{\text{CLOSE}}$, **DR**$_{\text{FLOATING}}$, **DR**$_{\text{OPPOSITE}}$ (Md=1.0, SD<=1.0). **DR**$_{\text{IDENTICAL}}$

and $\mathbf{DR}_{\text{SHUFFLE}}$ revealed a median of 2.0 for both (SD=2.0). However, sequences $\mathbf{NA}_{\text{DISABLED}}$ and $\mathbf{NA}_{\text{UNAVAILABLE}}$ exhibited a higher median number of backjumps of 3, indicating more frequent reviewing of the tutorial. We found that participants spent different amount of time watching the video while it was playing, and as this measurement is correlated to the duration of the video, we present these results proportionally to the duration of the video. As the count of video backjumps indicated, only $\mathbf{NA}_{\text{UNAVAILABLE}}$ and $\mathbf{NA}_{\text{DISABLED}}$ show a watching time close or higher than 300%. Longer use of video was also observed for $\mathbf{DR}_{\text{SHUFFLE}}$ (180.0%) while the other tasks only required watching the video one and a half time at most.

5.5.2 Comparing interfaces. Difficulties to visually locate the target feature led to more back-and-forths between displays. We noted that participants performed on average less than 4.5 back-and-forths between displays for most tasks, except for $\mathbf{NDR}_{\text{FLOATING}}$ (10.1), **RWC** (14.4), $\mathbf{NA}_{\text{UNAVAILABLE}}$ (17.2), $\mathbf{NA}_{\text{DISABLED}}$ (14.6) where they needed to compare interfaces more often. Surprisingly, $\mathbf{DR}_{\text{CLOSE}}$ (4.3) required more back-and-forths than other DIRECTLY REACHABLE (**DR**) tasks which required less than 3.9 on average. Consequently, participants spent more time watching the video interface while the video was paused on $\mathbf{NDR}_{\text{FLOATING}}$ (22.9s), **RWC** (30.5s), $\mathbf{NA}_{\text{UNAVAILABLE}}$ (51.5s) and $\mathbf{NA}_{\text{DISABLED}}$ (53.1s) while they spent up to 12.8s on other tasks. Some participants mentioned that even though the previous tasks did not require to use the tool demonstrated in the video, they already had a clue to where to find it because they paid attention to other features that comprise their interface while looking for previous features, alluding to a potential *incidental learning* of the interface [52, 79]: *"I'm going to scan instead, because there are things that were there at one point. But then they're on the other side, so I mainly looked at what I have done on the tasks before."* (P19) At this stage, users may expect the feature to be available in the interface, and begin to compare interfaces to identify the differences where they think having seen the feature.

5.5.3 Behaviors and strategies for locating functionality. After having watched the tutorial video, participants primarily focused on analyzing specific areas of their interface and exploring it to locate the required feature. Participants' feedback confirmed that they first transposed the location of the feature in the video directly on their interface, and then relied on visual cues such as icons, shapes, before higher levels elements: *"I go first to the same location that I saw [in the video]. I think it's immediate that you do that. So I want to locate, to see if it's familiar. [...] And if I don't find it, then I try also to see if I can find the icons, but if that fast or quick task doesn't work I go through to see exactly what is the title [of the component]"* (P1).

When searching was unsuccessful, search strategies seem to differ according to participants' expertise with customizable interfaces. Inexperienced participants returned more frequently to the video interface, engaging in systematic interface comparisons to find correspondences, proceeding by visual reproduction based on visual cues such as "cogwheel" or "text styles": *"And then, when I couldn't find what I was looking for, I had to look a second time and then I was telling myself: 'Well, we're going to try and display maybe a toolbox that is not activated'"* (P20). Other participants analyzed the layout to retrieve hierarchy information and compare interfaces, like P1: *"So this structure is kind of the same. I mean you have kind of panels, so you can easily identify where the panels are, where you have the two lines over there with the titles or headings. So I go through the headings, see if there is one that I can find".*

Participants sometimes also relied on their existing domain knowledge from similar interfaces, expecting it to be useful in this situation: *"I first look at what makes sense to me. When it comes to object's position, I'm used to the Unity editor, so I first look to the right because it seems logical that it should be there"* (P6). These participants have built a mental model of the interface of a typical software (e.g. Adobe Illustrator or Unity), and when searching for a feature, they expect other applications, Affinity Designer in this case, to be also built according to a similar scheme that they can leverage, in *"a sort of logical organization. [...] with palettes grouped by activity theme"* (P14).

Finally, participants also looked for built-in features to update the interface, *"to reset the tools in a kind of "default configuration" or something like that"* (P16), filter lists of features or highlight components by enabling/disabling them when they couldn't find them visually: *"I went into Windows, and I removed it. For example, I said to myself that it's not there, and I have to make it appear. Then I realized that it was already there because it was checked. So I looked for where it had disappeared and then I saw where it had changed"* (P2). 10 of our 26 participants reported this technique as the most valuable, often referring to habits from using other customizable software like *"Adobe products such as After Effects because there is a lot more panels [to activate] here"* (P18). That being said, this method appeared to be considered as a last resort if previous visual searches were unsuccessful, as one participant explained: *"if it wasn't present in the interface, in this case it was a bit more of a visual search because I did a double confirmation round to see if I'd forgotten it, and once confirmed, my first reflex was to try to go to Windows to find it"* (P12). However, participants inexperienced with customizable interfaces were reluctant to do so because of an uncertainty about undoing these modifications: *"I tell myself every time: I hope that I'm not going to break it [the interface], I'm not going to break it because I'm afraid, it's always tricky. But I'm not at all used to this kind of... it's very unsettling."* (P5)

5.6 Why do participants encounter difficulties?

5.6.1 Need for spatial reorientation. Participants expressed expectations regarding the structure of the interface. Some participants had a preconceived layout in mind, and confined their visual search to areas where they expected to find a specific functionality: *"At the beginning of the first task, I spent 4 minutes thinking 'Why isn't it here? I mean, how is that possible? How can I make it appear here?' I didn't even think to look there [on the opposite side of the interface]"* (P2). Indeed, participants expected familiar layout patterns and associated means of reaching features from other applications to be replicated: *"For text, especially text, I never use them in the windows. It's never on the interface, it's always above. When I click on the text, there's a sort of ribbon above it."* (P22) This transfer learning hindered their search process, leading them to ignore entire parts of the interface, even though a number of features were directly reachable: *"And that's why I didn't look at the floating windows. Because literally, cognitively, I couldn't even see it at first"* (P21). Thus, while the complete spatial reorganization of the interface was confusing for participants during their visual search, requiring some

reorientation time[76], some were able to detect that the OPPOSITE conditions simply proposed a "mirrored" interface and to adapt their search accordingly: *"I think there was a mirror at one point, like the right side is on the left and vice versa, it was still okay because you could quickly see that it was just flipped"* (P20).

5.6.2 Finding help. None of the participants used the built-in application help. This may be because they did not even know where to find it within the application, as noted by P2: *"I did not consider that menus, for example when I was lost, could provide help. I felt it was adversarial. And I couldn't figure out how to get out of this adversarial mode. I mean, was there a backup? A mechanism that could have allowed me to... My impression is that the interface is poorly designed"*. In fact, the observed differences between interfaces created uncertainty for some participants: *"I was not sure I had made the right transformation [...] because I found something that looked similar, but wasn't quite right"* (P8).

5.6.3 Visual saliency of features. Finally, once participants acknowledged the feature was not at the same location as in the video, they had to scan the interface relying on visual landmarks that may help them to localize it. The lack of elements that could act as visual anchors such as a *"visual tab name or a symbol"* (P12) then complicated their search. Added to this are problems linked to the visual theme of the interface (contrast, monochrome colors) which make it difficult to distinguish elements from one another. Understanding which feature was selected during these short instructional videos was also a challenge for some participants. Indeed, as expressed by P9, *"even just following the cursor movement is really hard"*, which makes it more difficult to understand the actions. This was exacerbated by the back-and-forth movements between the two screens, that could confuse their visual cues for situating themselves between the two interfaces: *"Well, I have discovered with the time that even if I enjoy two screens, depending on the size, I prefer to have both UIs in the same because this movement of my head just makes me lose reference of things"* (P1).

6 DISCUSSION

Despite the large corpus of research on video tutorials, we do not have a clear understanding of the impact of interface differences when learning a new software, and how it may affect user's capability to develop software skills. In this paper, we explored 13 cases of interface differences varying in terms of *Reachability* or *Availability* between a video tutorial and the interface of users. In the following section, we discuss our key results, their generalizability, and outline design directions to minimize the frictions coming from interface differences.

6.1 Lessons learned from our study

Takeaway 1: *The usefulness of tutorials is correlated with the availability of the features presented.* Tasks involving a difference in *Availability* have shown the highest failure rate. This suggests that users' ability to reproduce a tutorial strongly depends on the features demonstrated being readily present in user's interface, and suggests that the congruence principle [83] still applies in the context of comparing UIs when reproducing a video tutorial. Although none of the participants of the experiment used the built-in help of the application, a behavior already observed in other studies [39, 68], those who completed the **NA** or **RWC** tasks had to explore the interface deeper to enable the missing UI components or find alternatives.

Design recommendation 1: Tutorial designers may indicate the features used in their demonstrations, as well as compatibility with different versions of the application used. Recommendation systems (e.g. [91]) could leverage it to display similar workflows within the application, by inferring high-level tasks from feature listed, and recommend videos to learn alternative workflows in order to assist users when stuck because of a missing feature. It might also be useful to indicate workflows in similar applications, as already proposed by Ramesh et al [72].

Takeaway 2: *Users prefer active search when they have to locate a feature.* Overall, participants spent more time inspecting and analyzing their interface than watching the video tutorial. These results are in line with previous observations on learner's approaches when seeking help [39]. When searching for a feature, their gaze did rapid back-and-forth to compare their interface to the one in the video. If unable to locate the feature quickly this way, participants initiated a second review of the tutorial to check whether they missed something. Still, their first approach was to activate interface elements, filter displayed components or navigate menus before going back to a more careful analysis of the video's interface. This is further visible with participants who where not able to complete the tasks, who may *"click everywhere"* (P19) until they give up, expecting to fortuitously reveal the missing feature from a drawer, a menu or a search bar. This suggests that when following a tutorial, users still prefer a *"trial-and-error"* strategy [55] to an extensive exploration of the tutorial.

Design recommendation 2: would benefit from a system that corrects "false-feedforward errors" [46] when they try to replicate a tutorial. Exposing and highlighting features in the user interface as they are triggered in a tutorial would help learners to locate a feature. It could be done using static or dynamic highlighting that could improve rapid feature identification. That being said, such highlighting should be carefully designed as excessive help might hinder spatial learning of the UI [77].

Takeaway 3: *Users rely on semantic grouping to locate a feature in their interface.* Several participants expressed difficulties to locate features in an interface that is not structured according to a logic they understand. They expect an interface segmented by "activity zones" and to search by "categories" across these zones of their interface. Even without prior knowledge of the interface, they are expecting it to have a structure they can rely on to build a mental model [75] to navigate through zones until they find the closest one to the target feature. Some participants explicitly considered the interface as divided in semantic groups, and expected it to be ordered in some logical manner, even if it is not theirs, to make sense of how the spatial arrangement of the components (such as their position, orientation, or size). In desktop configurations, users tend to visually inspect the main containers, such as the top and left toolbars [36]. Therefore, tasks proposing a layout where the parent component of the target feature was not part of a larger context of the interface layout (e.g. in a floating window or in a window to be activated by configuring the workspace) proved

to be the most difficult of the tasks in the *Reachability* condition. Altogether, these observations confirm previous work conducted on spatial perception in the context of adaptive interfaces, where a so-called *"negative"* difference between interfaces is detrimental to the localization of features within them [21]. Note, however, that the principle of "semantic groups that can be dynamically created by the proximity principle" [21] is not reversible. Indeed, components found outside any higher-level context, such as those presented to participants in the $\mathbf{DR}_{\text{FLOATING}}$ and $\mathbf{NDR}_{\text{FLOATING}}$ tasks, are simply ignored at first, as suggested by longer task completion times for the floating conditions.

Design recommendation 3: When a feature is activated in a tutorial, it would be beneficial for the learner to always show its conceptual hierarchy (within the application) and spatial hierarchy (within the interface).

Takeaway 4: *Interface comparison approaches depend on overall familiarity with customizable interfaces.* When looking for a target feature in their interface, participants often tried at first to leverage the location in the video, as confirmed by the moderately strong correlation between **(L3)** and **(L2)**. When participants did not find the feature during the first inspection of the interface, they employed on of the two following approaches: 1) exploration and visual analysis of the interface; 2) less frequently, new review of the video interface, followed by a comparison of the two interfaces. This visual comparison mostly relies on visuals landmarks [85] such as side panels, window/tab names, icons and shapes. We observed that depending on user's experience with customizable interfaces, the interface comparison seems to be done at different levels. Users with prior knowledge are known to exhibit more diverse viewing strategies when it comes to learning [50], which also seems to be the case when comparing different software interfaces. Indeed, less experienced participants compare interfaces using a spatially-based approach, expecting features to be at specific places. Those more experienced with modular and customizable interfaces rely on a comparison at a higher level, comparing "categories" such as windows, tabs, or layout structure.

Design recommendation 4: Tutorials interface should use, when possible, a stable and easily accessible command hierarchy, reset to the default configuration at the beginning of the video, and show how to set up properly the interface to enable the required features for the tutorial so inexperienced users can configure their interface in the same way. Indeed, users with prior knowledge of customizable interfaces are able to perform spatial remapping [76] between interfaces if a logical structure is given. This would also support "over-the-shoulder" learning [84], such which is particularly efficient to discover unknown features, new practices and workflows from other peers [59, 60].

Takeaway 5: *Experienced participants do cross-application transfer learning when searching a feature within an interface.* Several participants mentioned expecting a particular interface structure, or a logic that they could leverage to group features. This logical structure is based on their past experiences with other customizable interfaces [94], associating specific regions of the interface for specific types of functionality. They were expecting a default configuration with windows anchored in side panels, and contextual functionalities in the top panel. This approach allowed them to adapt more easily when the feature was missing from the screen, given that interface components can be moved and activated/deactivated, but it also generated spatial disorientation which hindered their visual search.

Design recommendation 5: If tutorials came with a list of features used, as expressed in our recommendation 1, a system such as Show-me-How [72] could help users benefit from their domain knowledge and avoid detrimental experiences due to transfer learning [71]. Future work can be inspired by cross-device learnability research [3, 13, 20] to design systems that can help learners to find a feature within two different interfaces.

Takeaway 6: *Direct reachability or spatial proximity, which one to choose?* As seen in Takeaway 4, even though participants tend to leverage the location of elements in the video, our results do not suggest that spatial proximity should be preferred over a direct reachability in order to minimize overall interface distance. Instead, user's knowledge of customizable interfaces should be considered. Indeed, inexperienced users seem to expect interfaces to be identical or as close as possible; spatial arrangement of features may help them to localize features within the interface. Users already experienced with modular interfaces reported that they infer the location of features from semantic groups [81], and once identified, rely on them to find a feature they do not know yet. While for inexperienced users, spatial proximity between interfaces should be a priority, it is not particularly helpful for more experienced users. For experienced users, promoting direct reachability should improve their efficiency to find the target feature, since they rely on a mental image of their interface [85] rather than a specific location. Future work should confirm this hypothesis by studying how participants leverage reachability and spatial proximity, depending on their experience with customizable interfaces.

Design recommendation 6: Spatial proximity should be favored for inexperienced users (focusing on improving the learnability of an interface), while direct reachability should be favored for experienced ones (focusing on performance improvement). In addition, this would require to have models capable of reliably assess user expertise [29] (in this case, mostly UI, command, and task expertise) in order to tailor instructional videos to user's level.

6.2 Generalizability of results

Our experimental design was task-centric, influenced by prior work that showed that users approach software learning with a specific goal in mind instead of an exploration-centric approach [4, 34, 39, 43, 73]. Nonetheless, users may also look for video tutorials to learn the interface itself, to develop software customization knowledge or for personal experiences [53]. Future work should investigate such types of video tutorial consumption.

Although we focused on one vector graphics software, there is no reason that the insights we have learned could not be applied to different types of feature-rich software that support similar customization capabilities and are frequently updated over time. Nowadays, many software offer a modular, theme-based interface that users can customize. As such software are prone to frequent updates that may change their interface organization and hierarchy from one version to the next, users of audio/photo/video editing, music composition, 3D modeling software or IDE may also

encounter frictions resulting from differences similar to those we identified. Professional software such as Enterprise Resource Planning tools (ERP) are also known for their high modularity, arbitrary updates, and difference across user profiles. Studying friction with tutorials for such tools would likely highlight more issues.

Regarding our study set-up, three participants explained during the debriefing interview that at some point in the experiment, they noticed that the target features were not in their expected location, and started to anticipate this behavior. However, developing an anticipatory behavior was difficult, since the target feature was located in the vicinity of where it was in the tutorial in 5 tasks out of 13 ($\mathbf{DR}_{\text{IDENTICAL}}$, $\mathbf{NDR}_{\text{IDENTICAL}}$, $\mathbf{DR}_{\text{CLOSE}}$, $\mathbf{NDR}_{\text{CLOSE}}$ and $\mathbf{NA}_{\text{DISABLED}}$). This left little opportunity to develop adverse anticipatory behaviors. This is further confirmed by the moderately strong positive correlation we observed in responses to **(L3)** and **(L2)**. Even if participants adopted such a behavior, it would have to be after several tasks therefore not impacting all measures. In addition, any potential benefit would possibly be compensated by loss in one of the above-listed 5 conditions. Finally, our informal inspection of gaze fixations did not suggest that participants adopted a strong behavior anticipating the command to be at a different location. Altogether, we believe that participants did not avoid the location from the video, when looking at their own interface.

7 CONCLUSION

In this work, we characterized 13 types of *interface differences* due to variations of the *Reachability* and *Availability* of features that build a user interface. We observed 26 participants facing these different types of *interface differences* to investigate the frictions they might encounter in locating and applying features when instructed to reproduce a video tutorial illustrating a task in a customizable vector graphics software. Our study contributes to better understanding how users proceed to compare interfaces when the feature demonstrated in the tutorial is not located on their interface at the same location that in the video interface. We depict the behaviors they have adopted and discuss implications for mitigating *interfaces differences* between a tutorial's interface and the learner's one.

Our study presented participants with a limited case study, featuring only one instructional video per task and lacking the diverse array of options available in real-world scenarios, such as "over-the-shoulder" learning [84] or community-based information sources [48, 90]. However, insights gleaned from post-experiment interviews revealed that participants are reluctant to switch tutorials once initiated, as the potential benefits may not outweigh the time already invested. Prior research has underscored the advantages of adaptive interfaces in training systems [15], emphasizing the importance of tailoring committed to a specific tutorial, rather than inundating them with alternative options fraught with challenges like vocabulary mismatch [4, 39] or information overload inherent to video tutorials [63, 66, 69]. Further research building upon this work toward adaptive interfaces [10, 67] could evaluate the cost-benefit of adapting tutorial interfaces to user interfaces and develop models of interface distances tailored to learner objectives, whether task-centric or interface-exploration oriented.

8 ACKNOWLEDGMENTS

This work was supported by the Agence Nationale de la Recherche project Discovery (ANR-19-CE33-0006) and the région Hauts-de-France (France).

REFERENCES

[1] [n. d.]. Quasar Framework - Build High-Performance VueJS User Interfaces in Record Time. https://quasar.dev/.
[2] [n. d.]. Scipy.Stats.Bootstrap — SciPy v1.12.0 Manual. https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.bootstrap.html.
[3] Jessalyn Alvina, Andrea Bunt, Parmit K. Chilana, Sylvain Malacria, and Joanna McGrenere. 2020. Where Is That Feature?: Designing for Cross-Device Software Learnability. In *Proceedings of the 2020 ACM Designing Interactive Systems Conference*. ACM, Eindhoven Netherlands, 1103–1115. https://doi.org/10.1145/3357236.3395506
[4] Oscar D. Andrade, Nathaniel Bean, and David G. Novick. 2009. The Macro-Structure of Use of Help. In *Proceedings of the 27th ACM International Conference on Design of Communication - SIGDOC '09*. ACM Press, Bloomington, Indiana, USA, 143. https://doi.org/10.1145/1621995.1622022
[5] James J. Appleton, Sandra L. Christenson, and Michael J. Furlong. 2008. Student Engagement with School: Critical Conceptual and Methodological Issues of the Construct. *Psychology in the Schools* 45, 5 (2008), 369–386. https://doi.org/10.1002/pits.20303
[6] Sara Atske. 2018. Many Turn to YouTube for Children's Content, News, How-To Lessons.
[7] Lingfeng Bao, Zhenchang Xing, Xin Xia, and David Lo. 2019. VT-Revolution: Interactive Programming Video Tutorial Authoring and Watching System. *IEEE Transactions on Software Engineering* 45, 8 (Aug. 2019), 823–838. https://doi.org/10.1109/TSE.2018.2802916
[8] Jennifer K Bertrand and Craig S Chapman. 2023. Dynamics of Eye-Hand Coordination Are Flexibly Preserved in Eye-Cursor Coordination during an Online, Digital, Object Interaction Task. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. ACM, Hamburg Germany, 1–13. https://doi.org/10.1145/3544548.3580866
[9] Jean-Michel Boucheix, Perrine Gauthier, Jean-Baptiste Fontaine, and Sandrine Jaffeux. 2018. Mixed Camera Viewpoints Improve Learning Medical Hand Procedure from Video in Nurse Training? *Computers in Human Behavior* 89 (Dec. 2018), 418–429. https://doi.org/10.1016/j.chb.2018.01.017
[10] Sarah Bouzit, Gaelle Calvary, Joelle Coutaz, Denis Chene, Eric Petit, and Jean Vanderdonckt. 2017. The PDA-LPA Design Space for User Interface Adaptation. In *2017 11th International Conference on Research Challenges in Information Science (RCIS)*. IEEE, Brighton, United Kingdom, 353–364. https://doi.org/10.1109/RCIS.2017.7956559
[11] Cynthia J. Brame. 2016. Effective Educational Videos: Principles and Guidelines for Maximizing Student Learning from Video Content. *CBE—Life Sciences Education* 15, 4 (Dec. 2016), es6. https://doi.org/10.1187/cbe.16-03-0125
[12] Lori Breslow, David Pritchard, Jennifer DeBoer, Glenda Stump, Andrew Ho, and Daniel Seaton. 2013. Studying Learning in the Worldwide Classroom: Research into edX's First MOOC. *Research in Practice and Assessment* 8 (June 2013), 13–25.
[13] Frederik Brudy, Christian Holz, Roman Rädle, Chi-Jui Wu, Steven Houben, Clemens Nylandsted Klokmose, and Nicolai Marquardt. 2019. Cross-Device Taxonomy: Survey, Opportunities and Challenges of Interactions Spanning Across Multiple Devices. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–28. https://doi.org/10.1145/3290605.3300792
[14] Federico Cabitza and Carla Simone. 2017. Malleability in the Hands of End-Users. In *New Perspectives in End-User Development*, Fabio Paternò and Volker Wulf (Eds.). Springer International Publishing, Cham, 137–163. https://doi.org/10.1007/978-3-319-60291-2_7
[15] John M. Carroll and Caroline Carrithers. 1984. Training Wheels in a User Interface. *Commun. ACM* 27, 8 (Aug. 1984), 800–806. https://doi.org/10.1145/358198.358218
[16] Pei-Yu Chi, Sally Ahn, Amanda Ren, Mira Dontcheva, Wilmot Li, and Björn Hartmann. 2012. MixT: Automatic Generation of Step-by-Step Mixed Media Tutorials. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology - UIST '12*. ACM Press, Cambridge, Massachusetts, USA, 93. https://doi.org/10.1145/2380116.2380130
[17] Parmit K. Chilana, Amy J. Ko, and Jacob O. Wobbrock. 2012. LemonAid: Selection-Based Crowdsourced Contextual Help for Web Applications. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, Austin Texas USA, 1549–1558. https://doi.org/10.1145/2207676.2208620
[18] Pedro Cuesta-Valiño, Pablo Gutiérrez-Rodríguez, and Patricia Durán-Álamo. 2022. Why Do People Return to Video Platforms? Millennials and Centennials on TikTok. *Media and Communication* 10, 1 (Feb. 2022), 198–207.
[19] Shujie Deng, Jian Chang, Julie A. Kirkby, and Jian J. Zhang. 2016. Gaze–Mouse Coordinated Movements and Dependency with Coordination Demands in Tracing. *Behaviour & Information Technology* 35, 8 (Aug. 2016), 665–679. https:

//doi.org/10.1080/0144929X.2016.1181209

[20] Charles Denis and Laurent Karsenty. 2003. Inter-Usability of Multi-Device Systems – A Conceptual Framework. In *Multiple User Interfaces* (1 ed.), Ahmed Seffah and Homa Javahery (Eds.). Wiley, 373–385. https://doi.org/10.1002/0470091703.ch17

[21] Tilman Deuschel and Ted Scully. 2016. On the Importance of Spatial Perception for the Design of Adaptive User Interfaces. In *2016 IEEE 10th International Conference on Self-Adaptive and Self-Organizing Systems (SASO)*. IEEE, Augsburg, Germany, 70–79. https://doi.org/10.1109/SASO.2016.13

[22] M Dewan, Mahbub Murshed, and Fuhua Lin. 2019. Engagement Detection in Online Learning: A Review. *Smart Learning Environments* 6, 1 (2019), 1–20. https://doi.org/article/10.1186/s40561-018-0080-z

[23] Michael Ekstrand, Wei Li, Tovi Grossman, Justin Matejka, and George Fitzmaurice. 2011. Searching for Software Learning Resources Using Application Context. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*. ACM, Santa Barbara California USA, 195–204. https://doi.org/10.1145/2047196.2047220

[24] Anna Ertelt, Alexander Renkl, and Hans Spada. 2006. Making a Difference - Exploiting the Full Potential of Instructionally Designed on-Screen Videos. *ICLS 2006 - International Conference of the Learning Sciences, Proceedings* 1 (Jan. 2006), 154–160.

[25] Leah Findlater and Joanna McGrenere. 2010. Beyond Performance: Feature Awareness in Personalized Interfaces. *International Journal of Human-Computer Studies* 68, 3 (March 2010), 121–137. https://doi.org/10.1016/j.ijhcs.2009.10.002

[26] Adam Fourney, Ben Lafreniere, Richard Mann, and Michael Terry. 2012. "Then Click Ok!": Extracting References to Interface Elements in Online Documentation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, Austin Texas USA, 35–38. https://doi.org/10.1145/2207676.2207682

[27] C. Ailie Fraser, Mira Dontcheva, Holger Winnemöller, Sheryl Ehrlich, and Scott Klemmer. 2016. DiscoverySpace: Suggesting Actions in Complex Software. In *Proceedings of the 2016 ACM Conference on Designing Interactive Systems*. ACM, Brisbane QLD Australia, 1221–1232. https://doi.org/10.1145/2901790.2901849

[28] C. Ailie Fraser, Tricia J. Ngoon, Mira Dontcheva, and Scott Klemmer. 2019. RePlay: Contextually Presenting Learning Videos Across Software Applications. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, Glasgow Scotland Uk, 1–13. https://doi.org/10.1145/3290605.3300527

[29] Tovi Grossman and George Fitzmaurice. 2015. An Investigation of Metrics for the In Situ Detection of Software Expertise. *Human–Computer Interaction* 30, 1 (Jan. 2015), 64–102. https://doi.org/10.1080/07370024.2014.881668

[30] Tovi Grossman, George Fitzmaurice, and Ramtin Attar. 2009. A Survey of Software Learnability: Metrics, Methodologies and Guidelines. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, Boston MA USA, 649–658. https://doi.org/10.1145/1518701.1518803

[31] Philip J. Guo. 2018. Non-Native English Speakers Learning Computer Programming: Barriers, Desires, and Design Opportunities. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, Montreal QC Canada, 1–14. https://doi.org/10.1145/3173574.3173970

[32] Philip J. Guo, Juho Kim, and Rob Rubin. 2014. How Video Production Affects Student Engagement: An Empirical Study of MOOC Videos. In *Proceedings of the First ACM Conference on Learning @ Scale Conference*. ACM, Atlanta Georgia USA, 41–50. https://doi.org/10.1145/2556325.2566239

[33] Jeff Huang, Ryen White, and Georg Buscher. 2012. User See, User Point: Gaze and Cursor Alignment in Web Search. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, Austin Texas USA, 1341–1350. https://doi.org/10.1145/2207676.2208591

[34] Nathaniel Hudson, Benjamin Lafreniere, Parmit K. Chilana, and Tovi Grossman. 2018. Investigating How Online Help and Learning Resources Support Children's Use of 3D Design Software. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, Montreal QC Canada, 1–14. https://doi.org/10.1145/3173574.3173831

[35] Dan Jackson, James Nicholson, Gerrit Stoeckigt, Rebecca Wrobel, Anja Thieme, and Patrick Olivier. 2013. Panopticon: A Parallel Video Overview System. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*. ACM, St. Andrews Scotland, United Kingdom, 123–130. https://doi.org/10.1145/2501988.2502038

[36] Yue Jiang, Luis A. Leiva, Hamed Rezazadegan Tavakoli, Paul R. B. Houssel, Julia Kylmälä, and Antti Oulasvirta. 2023. UEyes: Understanding Visual Saliency across User Interface Types. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23)*. Association for Computing Machinery, New York, NY, USA, 1–21. https://doi.org/10.1145/3544548.3581096

[37] Verena Käfer, Daniel Kulesz, and Stefan Wagner. 2017. What Is the Best Way For Developers to Learn New Software Tools? An Empirical Comparison Between a Text and a Video Tutorial. (2017). https://doi.org/10.48550/ARXIV.1704.00074

[38] Kandarp Khandwala and Philip J. Guo. 2018. Codemotion: Expanding the Design Space of Learner Interactions with Computer Programming Tutorial Videos. In *Proceedings of the Fifth Annual ACM Conference on Learning at Scale*. ACM, London United Kingdom, 1–10. https://doi.org/10.1145/3231644.3231652

[39] Kimia Kiani, George Cui, Andrea Bunt, Joanna McGrenere, and Parmit K. Chilana. 2019. Beyond "One-Size-Fits-All": Understanding the Diversity in How Software Newcomers Discover and Make Use of Help Resources. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, Glasgow Scotland Uk, 1–14. https://doi.org/10.1145/3290605.3300570

[40] Juho Kim, Philip J. Guo, Carrie J. Cai, Shang-Wen (Daniel) Li, Krzysztof Z. Gajos, and Robert C. Miller. 2014. Data-Driven Interaction Techniques for Improving Navigation of Educational Videos. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*. ACM, Honolulu Hawaii USA, 563–572. https://doi.org/10.1145/2642918.2647389

[41] Juho Kim, Phu Tran Nguyen, Sarah Weir, Philip J. Guo, Robert C. Miller, and Krzysztof Z. Gajos. 2014. Crowdsourcing Step-by-Step Information Extraction to Enhance Existing How-to Videos. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, Toronto Ontario Canada, 4017–4026. https://doi.org/10.1145/2556288.2556986

[42] René F. Kizilcec, Chris Piech, and Emily Schneider. 2013. Deconstructing Disengagement: Analyzing Learner Subpopulations in Massive Open Online Courses. In *Proceedings of the Third International Conference on Learning Analytics and Knowledge*. ACM, Leuven Belgium, 170–179. https://doi.org/10.1145/2460296.2460330

[43] Ben Lafreniere, Andrea Bunt, Matthew Lount, and Michael Terry. 2013. Understanding the Roles and Uses of Web Tutorials. *Proceedings of the International AAAI Conference on Web and Social Media* 7, 1 (2013), 303–310. https://doi.org/10.1609/icwsm.v7i1.14413

[44] Ben Lafreniere, Andrea Bunt, Matthew Lount, Michael Terry, and Donald Cowan. 2012. *Looks Cool, I'll Try This Later!": Understanding the Faces and Uses of Online Tutorials*. Technical Report 2012-01. University of Waterloo, Waterloo, CA.

[45] Benjamin Lafreniere, Andrea Bunt, and Michael Terry. 2014. Task-Centric Interfaces for Feature-Rich Software. In *Proceedings of the 26th Australian Computer-Human Interaction Conference on Designing Futures: The Future of Design*. ACM, Sydney New South Wales Australia, 49–58. https://doi.org/10.1145/2686612.2686620

[46] Benjamin Lafreniere, Parmit K. Chilana, Adam Fourney, and Michael A. Terry. 2015. *These Aren't the Commands You're Looking For*: Addressing False Feedforward in Feature-Rich Software. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. ACM, Charlotte NC USA, 619–628. https://doi.org/10.1145/2807442.2807482

[47] Ben Lafreniere and Tovi Grossman. 2018. Blocks-to-CAD: A Cross-Application Bridge from Minecraft to 3D Modeling. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*. ACM, Berlin Germany, 637–648. https://doi.org/10.1145/3242587.3242602

[48] Benjamin Lafreniere, Tovi Grossman, and George Fitzmaurice. 2013. Community Enhanced Tutorials: Improving Tutorials with Multiple Demonstrations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, Paris France, 1779–1788. https://doi.org/10.1145/2470654.2466235

[49] Patricia G. Lange. 2018. Informal Learning on YouTube. In *The International Encyclopedia of Media Literacy* (1 ed.), Renee Hobbs and Paul Mihailidis (Eds.). Wiley, 1–11. https://doi.org/10.1002/9781118978238.ieml0090

[50] Liang-Yi Li. 2019. Effect of Prior Knowledge on Attitudes, Behavior, and Learning Performance in Video Lecture Viewing. *International Journal of Human–Computer Interaction* 35, 4-5 (March 2019), 415–426. https://doi.org/10.1080/10447318.2018.1543086

[51] Gao-fu Liu, Peng-chao Gao, Yu-chun Li, and Zhuo-ping Zhang. 2019. Research on the Influence of Social Media Short Video Marketing on Consumer Brand Attitude. In *Proceedings of the 2019 5th International Conference on Social Science and Higher Education (ICSSHE 2019)*. Atlantis Press, Xiamen, China. https://doi.org/10.2991/icsshe-19.2019.192

[52] Wanyu Liu, Gilles Bailly, and Andrew Howes. 2017. Effects of Frequency Distribution on Linear Menu Performance. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. ACM, Denver Colorado USA, 1307–1312. https://doi.org/10.1145/3025453.3025707

[53] Laura MacLeod, Margaret-Anne Storey, and Andreas Bergen. 2015. Code, Camera, Action: How Software Developers Document and Share Program Knowledge Using YouTube. In *2015 IEEE 23rd International Conference on Program Comprehension*. IEEE, Florence, Italy, 104–114. https://doi.org/10.1109/ICPC.2015.19

[54] Aristides Mairena, Sami Uddin, and Carl Gutwin. 2022. Accidental Landmarks: How Showing (and Removing) Emphasis in a 2D Visualization Affected Retrieval and Revisitation. In *Graphics Interface 2022*.

[55] Damien Masson, Jo Vermeulen, George Fitzmaurice, and Justin Matejka. 2022. Supercharging Trial-and-Error for Learning Complex Software Applications. In *CHI Conference on Human Factors in Computing Systems*. ACM, New Orleans LA USA, 1–13. https://doi.org/10.1145/3491102.3501895

[56] Justin Matejka, Tovi Grossman, and George Fitzmaurice. 2014. Video Lens: Rapid Playback and Exploration of Large Video Collections and Associated Metadata. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*. ACM, Honolulu Hawaii USA, 541–550. https://doi.org/10.1145/2642918.2647366

[57] Ryan McGrady, Kevin Zheng, Rebecca Curran, Jason Baumgartner, and Ethan Zuckerman. 2023. Dialing for Videos: A Random Sample of YouTube. *Journal of Quantitative Description: Digital Media* 3 (Dec. 2023). https://doi.org/10.51685/jqd.2023.022

[58] Alexandre Milisavljevic, Kevin Hamard, Coralie Petermann, Bernard Gosselin, Karine Doré-Mazars, and Matei Mancas. 2018. Eye and Mouse Coordination During Task: From Behaviour to Prediction:. In *Proceedings of the 13th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*. SCITEPRESS - Science and Technology Publications, Funchal, Madeira, Portugal, 86–93. https://doi.org/10.5220/0006618800860093

[59] Emerson Murphy-Hill, Da Young Lee, Gail C. Murphy, and Joanna McGrenere. 2015. How Do Users Discover New Tools in Software Development and Beyond? *Computer Supported Cooperative Work (CSCW)* 24, 5 (Oct. 2015), 389–422. https://doi.org/10.1007/s10606-015-9230-9

[60] Emerson Murphy-Hill and Gail C. Murphy. 2011. Peer Interaction Effectively, yet Infrequently, Enables Programmers to Discover New Tools. In *Proceedings of the ACM 2011 Conference on Computer Supported Cooperative Work - CSCW '11*. ACM Press, Hangzhou, China, 405. https://doi.org/10.1145/1958824.1958888

[61] Vidhya Navalpakkam, LaDawn Jentzsch, Rory Sayres, Sujith Ravi, Amr Ahmed, and Alex Smola. 2013. Measurement and Modeling of Eye-Mouse Behavior in the Presence of Nonlinear Page Layouts. In *Proceedings of the 22nd International Conference on World Wide Web*. ACM, Rio de Janeiro Brazil, 953–964. https://doi.org/10.1145/2488388.2488471

[62] Manziba Akanda Nishi and Kostadin Damevski. 2020. Automatically Identifying Valid API Versions for Software Development Tutorials on the Web. *Journal of Software: Evolution and Process* 32, 4 (April 2020). https://doi.org/10.1002/smr.2227

[63] David G. Novick, Oscar D. Andrade, and Nathaniel Bean. 2009. The Micro-Structure of Use of Help. In *Proceedings of the 27th ACM International Conference on Design of Communication - SIGDOC '09*. ACM Press, Bloomington, Indiana, USA, 97. https://doi.org/10.1145/1621995.1622014

[64] Moses Palmér. [n. d.]. Pynput: Monitor and Control User Input Devices.

[65] Amy Pavel, Dan B. Goldman, Björn Hartmann, and Maneesh Agrawala. 2015. SceneSkim: Searching and Browsing Movies Using Synchronized Captions, Scripts and Plot Summaries. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. ACM, Charlotte NC USA, 181–190. https://doi.org/10.1145/2807442.2807502

[66] Amy Pavel, Colorado Reed, Björn Hartmann, and Maneesh Agrawala. 2014. Video Digests: A Browsable, Skimmable Format for Informational Lecture Videos. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*. ACM, Honolulu Hawaii USA, 573–582. https://doi.org/10.1145/2642918.2647400

[67] Tim F. Paymans, Jasper Lindenberg, and Mark Neerincx. 2004. Usability Trade-Offs for Adaptive User Interfaces: Ease of Use and Learnability. In *Proceedings of the 9th International Conference on Intelligent User Interface - IUI '04*. ACM Press, Funchal, Madeira, Portugal, 301. https://doi.org/10.1145/964442.964512

[68] Suporn Pongnumkul, Mira Dontcheva, Wilmot Li, Jue Wang, Lubomir Bourdev, Shai Avidan, and Michael F. Cohen. 2011. Pause-and-Play: Automatically Linking Screencast Video Tutorials with Applications. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology - UIST '11*. ACM Press, Santa Barbara, California, USA, 135. https://doi.org/10.1145/2047196.2047213

[69] Luca Ponzanelli, Gabriele Bavota, Andrea Mocci, Massimiliano Di Penta, Rocco Oliveto, Mir Hasan, Barbara Russo, Sonia Haiduc, and Michele Lanza. 2016. Too Long; Didn't Watch!: Extracting Relevant Fragments from Software Development Video Tutorials. In *Proceedings of the 38th International Conference on Software Engineering*. ACM, Austin Texas, 261–272. https://doi.org/10.1145/2884781.2884824

[70] Jon Porter. 2020. Facebook's Old Web Design Will Disappear in September. https://www.theverge.com/2020/8/21/21395079/facebook-new-design-default-september-classic-interface-disappearing.

[71] Reyhaneh Raissi, Evanthia Dimara, Jacquelyn H. Berry, Wayne D. Gray, and Gilles Bailly. 2020. Retroactive Transfer Phenomena in Alternating User Interfaces. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. ACM, Honolulu HI USA, 1–14. https://doi.org/10.1145/3313831.3376538

[72] Vidya Ramesh, Charlie Hsu, Maneesh Agrawala, and Björn Hartmann. 2011. ShowMeHow: Translating User Interface Instructions between Applications. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*. ACM, Santa Barbara California USA, 127–134. https://doi.org/10.1145/2047196.2047212

[73] John Rieman. 1996. A Field Study of Exploratory Learning Strategies. *ACM Transactions on Computer-Human Interaction* 3, 3 (Sept. 1996), 189–218. https://doi.org/10.1145/234526.234527

[74] Kerry Rodden and Xin Fu. 2007. Exploring How Mouse Movements Relate to Eye Movements on Web Search Results Pages. In *Web Information Seeking and Interaction*, Vol. Workshop on Web Information Seeking and Interaction. Amsterdam The Netherlands, 29–32.

[75] John W. Satzinger and Lorne Olfman. 1998. User Interface Consistency across End-User Applications: The Effects on Mental Models. *Journal of Management Information Systems* 14, 4 (March 1998), 167–193. https://doi.org/10.1080/07421222.1998.11518190

[76] Joey Scarr, Andy Cockburn, Carl Gutwin, and Sylvain Malacria. 2013. Testing the Robustness and Performance of Spatially Consistent Interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. Association for Computing Machinery, New York, NY, USA, 3139–3148. https://doi.org/10.1145/2470654.2466430

[77] Joey Scarr, Carl Gutwin, Andy Cockburn, and Andrea Bunt. 2015. StencilMaps and EphemeralMaps: Spatially Stable Interfaces That Highlight Command Subsets. *Behaviour & Information Technology* 34, 11 (Nov. 2015), 1092–1106. https://doi.org/10.1080/0144929X.2015.1046927

[78] Daniel T. Seaton, Yoav Bergner, Isaac Chuang, Piotr Mitros, and David E. Pritchard. 2014. Who Does What in a Massive Open Online Course? *Commun. ACM* 57, 4 (April 2014), 58–65. https://doi.org/10.1145/2500876

[79] Dennis Shelton and Robert C. Newhouse. 1981. Incidental Learning in a Paired-Associate Task. *The Journal of Experimental Education* 50, 1 (Sept. 1981), 36–38. https://doi.org/10.1080/00220973.1981.11011798

[80] Hamed R. Tavakoli, Fawad Ahmed, Ali Borji, and Jorma Laaksonen. 2017. Saliency Revisited: Analysis of Mouse Movements Versus Fixations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1774–1782.

[81] Dejan Todorovic. 2008. Gestalt Principles. *Scholarpedia* 3, 12 (Dec. 2008), 5345. https://doi.org/10.4249/scholarpedia.5345

[82] Cristen Torrey, Elizabeth F. Churchill, and David W. McDonald. 2009. Learning How: The Search for Craft Knowledge on the Internet. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, Boston MA USA, 1371–1380. https://doi.org/10.1145/1518701.1518908

[83] Barbara Tversky, Julie Bauer Morrison, and Mireille Betrancourt. 2002. Animation: Can It Facilitate? *International Journal of Human-Computer Studies* 57, 4 (Oct. 2002), 247–262. https://doi.org/10.1006/ijhc.2002.1017

[84] Michael B. Twidale. 2005. Over the Shoulder Learning: Supporting Brief Informal Learning. *Computer Supported Cooperative Work (CSCW)* 14, 6 (Dec. 2005), 505–547. https://doi.org/10.1007/s10606-005-9007-7

[85] Sami Uddin and Carl Gutwin. 2021. The Image of the Interface: How People Use Landmarks to Develop Spatial Memory of Commands in Graphical Interfaces. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. ACM, Yokohama Japan, 1–17. https://doi.org/10.1145/3411764.3445050

[86] Hans Van Der Meij and Jan Van Der Meij. 2014. A Comparison of Paper-Based and Video Tutorials for Software Learning. *Computers & Education* 78 (Sept. 2014), 150–159. https://doi.org/10.1016/j.compedu.2014.06.003

[87] Hans Van Der Meij, Jan Van Der Meij, Tessa Voerman, and Evert Duipmans. 2018. Supporting Motivation, Task Performance and Retention in Video Tutorials for Software Training. *Educational Technology Research and Development* 66, 3 (June 2018), 597–614. https://doi.org/10.1007/s11423-017-9560-z

[88] J. Van Der Meij and H. Van Der Meij. 2015. A Test of the Design of a Video Tutorial for Software Training. *Journal of Computer Assisted Learning* 31, 2 (April 2015), 116–132. https://doi.org/10.1111/jcal.12082

[89] Anna Vasilchenko, Adriana Wilde, Stephen Snow, Madeline Balaam, and Marie Devlin. 2018. Video Coursework: Opportunity and Challenge for HCI Education. In *Proceedings of the 2018 International Conference on Advanced Visual Interfaces*. ACM, Castiglione della Pescaia Grosseto Italy, 1–3. https://doi.org/10.1145/3206505.3206596

[90] Laton Vermette, Shruti Dembla, April Y. Wang, Joanna McGrenere, and Parmit K. Chilana. 2017. Social CheatSheet: An Interactive Community-Curated Information Overlay for Web Applications. *Proceedings of the ACM on Human-Computer Interaction* 1, CSCW (Dec. 2017), 1–19. https://doi.org/10.1145/3134737

[91] Xu Wang, Benjamin Lafreniere, and Tovi Grossman. 2018. Leveraging Community-Generated Videos and Command Logs to Classify and Recommend Software Workflows. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, Montreal QC Canada, 1–13. https://doi.org/10.1145/3173574.3173859

[92] Sarah A. Weir. 2014. Learnersourcing Subgoal Labels for How-to Videos. In *CHI '14 Extended Abstracts on Human Factors in Computing Systems*. ACM, Toronto Ontario Canada, 945–950. https://doi.org/10.1145/2559206.2579416

[93] Di Wu, Xiao-Yuan Jing, Hongyu Zhang, Yuming Zhou, and Baowen Xu. 2021. Leveraging Stack Overflow to Detect Relevant Tutorial Fragments of APIs. In *2021 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, Honolulu, HI, USA, 119–130. https://doi.org/10.1109/SANER50967.2021.00020

[94] Siriporn Yamnill and Gary N. McLean. 2001. Theories Supporting Transfer of Training. *Human Resource Development Quarterly* 12, 2 (2001), 195–208. https://doi.org/10.1002/hrdq.7

[95] Saelyne Yang, Jisu Yim, Aitolkyn Baigutanova, Seoyoung Kim, Minsuk Chang, and Juho Kim. 2022. SoftVideo: Improving the Learning Experience of Software Tutorial Videos with Collective Interaction Data. In *27th International Conference on Intelligent User Interfaces*. ACM, Helsinki Finland, 646–660. https://doi.org/10.1145/3490099.3511106

[96] Jeffrey M. Zacks and Barbara Tversky. 2003. Structuring Information Interfaces for Procedural Learning. *Journal of Experimental Psychology: Applied* 9, 2 (2003), 88–100. https://doi.org/10.1037/1076-898X.9.2.88