What Did My Users Experience? Discovering Visual Stimuli on Graphical User Interfaces of the Web

RAPHAEL MENGES, Semanux GmbH, Germany
STEFFEN STAAB, University of Stuttgart, Germany & University of Southampton, UK
CHRISTOPH SCHAEFER, Tobii AB, Sweden
TINA WALBER, Tobii AB, Sweden
CHANDAN KUMAR, Fraunhofer Institute for Industrial Engineering, Germany

Main tasks of usability experts for Web sites comprise the analysis of user interaction behavior on graphical user interfaces, the discovery of issues, and the derivation of improvements to the interface. The analysis of user interaction behavior and corresponding discovery of issues are made difficult by modern Web interfaces that incorporate dynamic interface elements and that orchestrate complex reactions to user responses. We propose a semi-automated approach for discovering visual stimuli, which capture summarized views of the interface as encountered by users during interaction. Discovered visual stimuli allow for meaningful aggregations of user interactions based on what users encountered on the interface such that the analysis by usability experts can relate the interface views with user interactions correctly and identify arising issues. We provide WebVSD as an implementation of the approach and perform a set of evaluations with real-world Web sites that show the accuracy of proposed methods in isolation and in the tool chain, as well as case studies and a survey of usability experts indicating the usefulness of the suggested approach.

CCS Concepts: • **Human-centered computing** → **Systems and tools for interaction design**; *Usability testing*; • **Information systems** → *Data extraction and integration.*

Additional Key Words and Phrases: usability analysis, dynamic interface, behavioral study, visual change, stimulus shot, visual stimulus, eye tracking, user experience, computer vision

ACM Reference Format:
Raphael Menges, Steffen Staab, Christoph Schaefer, Tina Walber, and Chandan Kumar. 2025. What Did My Users Experience? Discovering Visual Stimuli on Graphical User Interfaces of the Web. In *ACM Transactions on the Web*. ACM, New York, NY, USA, 64 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

User experience (UX) has emerged as a pivotal factor in gauging the quality of a product or service, focusing on how effectively a product performs for end-users. While there is no universally accepted definition of UX, various perspectives converge on the idea that it integrates the physical and technical aspects of a product with the cognitive processes that occur during user interaction [45, 60, 78, 89]. This emphasis on emotional impact and overall satisfaction underscores the importance of UX in product development.

Usability, an integral component of user experience, is the most commonly employed paradigm in the product design industry. Usability studies focus on the effectiveness, efficiency, and satisfaction with which users can achieve specific goals with a product or system [78]. More specifically, usability studies play a crucial role in optimizing user interactions on graphical user interfaces of the Web. Clickstream analysis [81, 97, 98] or A/B testing [8, 87] are popular methods for usability studies that aim at understanding *how often* users interact with *which individual elements*. However, these methods often lack the ability to contextualize user engagement, making it challenging for

To appear in ACM Transactions on the Web, Accepted January 2025.

usability experts to understand the specific circumstances prompting user responses to various element.

In contrast, we aim at supporting usability experts to analyze user interaction behavior targeting a more comprehensive understanding of the what user encountered during interaction [36, 77, 80]. This lifts the scope of analysis from how the users interact with individual elements in the interface to the question how the users interact with the arrangements of all the elements that the interface is composed of. For example, the same button element might draw users' attention and trigger action next to a slowly evolving video, but might be overlooked next to a fast-paced video. Therefore, usability experts need to understand how the arrangements of elements in interfaces stimulate user responses, such as mouse or eye movements, mouse clicks, touch, or key pressings, to discover shortcomings of these interfaces.

1.1 From Analysing Usability of Static Views to Analysing Dynamic User Interfaces

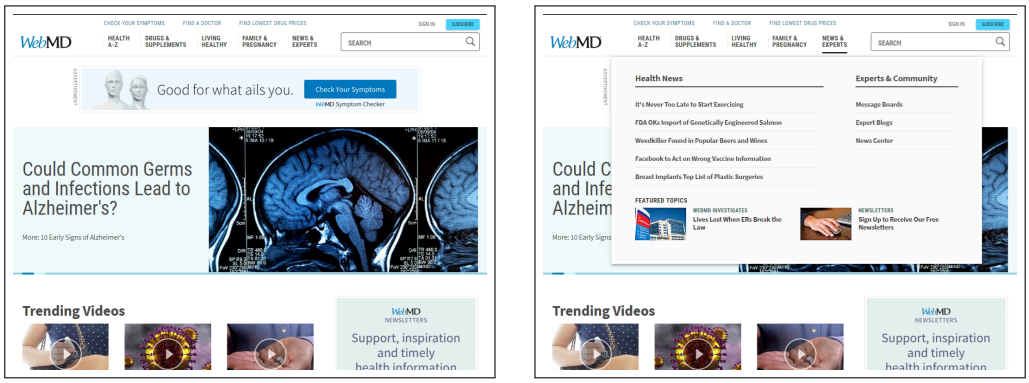
Commonly, usability experts define *areas of interest* (AOIs) [48] on user interfaces to aggregate and analyze interaction data from multiple users. For instance, a usability expert might want to analyze eye gazes or mouse movements within an AOI to answer questions like "In which part of the menu have users fixated their eye gaze most often?," "Have users understood where in the photo carousel they might click?," or "Has the title text or the image attracted more attention by the users?" When multiple AOIs are marked on an interface, the sequence of visited AOIs can be noted to answer questions that concern the user engagement, e. g., "Did users first look at the product or the price tag?"

Traditionally, graphical user interfaces on the Web have comprised views with a static set of elements and the views have not changed their appearance during a user session [14, 18, 35, 54, 83]. Then, each view may be considered a *stimulus* that triggers various responses of users, e. g., eye gaze paths or mouse traces, which may be recorded for each such stimulus. A usability expert may easily mark AOIs on these stimuli and analyze and aggregate recorded observations at the level of one or several AOIs or a whole stimulus providing her with a wealth of implicit user feedback to judge the suitability of a user interface design.

In modern Web interfaces, however, users manipulate and respond to passive as well as to active content [70]. We refer to such interfaces as *dynamic user interfaces*. Dynamic user interfaces may extend menus and choreograph photo carousels, adjust dynamically to clicking, hovering, and scrolling triggered by the user, or exhibit other complex rendering and visual effects [11]. See Figure 1 for an example of an expandable menu on the front page of the WebMD Web site. User interface designers may use a plenitude of styling attributes per element like `visibility`, `opacity`, `display`, `transform`, and `z-index` to overlay content and dynamically control the visuals of an interface via scripts. The actual visibility of an element is not only controlled by the attributes of the element itself, but it also depends on the configuration of other elements on the user interface, which may overlay the element or themselves become hidden by the element.

These characteristics of dynamic user interfaces make it difficult to align the various interactions of multiple users with well-defined stimuli such that the usability expert would easily understand how the users responded to which dynamic states of an interface. Usability experts so far have tackled this challenge conducting analyses on dynamic user interfaces employing one or several of the following three paradigms:

(1) *Video and Interaction Recordings.* This method records each user session in a video, which is temporally aligned with the recording of the user's interactions [1, 29, 67, 92]. Thus, arbitrary dynamics of an interface are captured. However, due to differences between users' behaviors, the recordings of various users cannot be aligned and aggregated. Therefore, a usability



(a) Front page of the WebMD site after the initial loading before any user interaction.

(b) The same page when a user hovers with the mouse cursor over the menu entry "NEWS AND EXPERTS."

Fig. 1. The appearance and functionality of a dynamic user interface can dramatically change automatically over the course of time or at user interaction.

expert must inspect the recordings of the screen contents and the user interactions one-user-at-a-time to understand all the effects of the design, making the analysis time-consuming and tedious and often overwhelming. Overall, this approach does not scale to the considerable number of users required in usability analyses [37].

(2) *Introspection and Interaction Recordings.* This method interprets the code that defines an user interface. In theory, the recording of changes in the internal structure of an interface allows for reproducing the stimuli. However, it can be *(i)*, difficult to keep up-to-date with the interface technologies, *(ii)*, difficult to judge the effect of structural changes on the visuals of an interface, *(iii)*, result in a plenitude of recorded states, which hinder an aggregation of the interaction recordings from multiple users, and *(iv)*, limited to interfaces that allow for introspection. The introspection method has been applied in the context of Web [18, 44, 54] and Android applications [30–32, 63, 105], yet it is not suitable for analyses of dynamic user interfaces with study participants in general.

(3) *Virtual Screenshot and Interaction Recordings.* This method treats each view of an interface, e. g., per loading of a Web page, as a stimulus [27, 41, 82, 86, 92]. This method has been extended by us through stitching consecutive screenshots resulting from the scrolling of a static Web page identified by its URL [69]. By such means, recordings of user interactions may be aligned to fewer virtual screenshots facilitating analysis by the usability expert who needs to define AOIs on fewer visual stimuli and must inspect fewer stimulus-response recordings. The disadvantage of this method is that the screenshots cannot generally reflect dynamic content changes. Dynamics in interfaces that are not related to solely scrolling of the entire page body cannot be properly aligned with corresponding user interactions.

Virtual screenshots and video recordings constitute two extreme points as to what they assume should be the basis for analyzing stimulus-response behavior. At one end virtual screenshots, which represent possibly dynamic user interfaces assume that each user interface view, e. g., corresponding to one URL, constitutes a single stimulus. This assumption, however, fails to capture all the dynamics. At the other end, video recordings assume that each video frame constitutes a stimulus overwhelming the usability expert with too many stimulus-response pairs. Therefore, we argue for a middle ground between the two extremes that exhibits the advantages of both,

while avoiding their disadvantages. While introspection methods also target such a middle ground, the difficulties we analyzed above call out for replacing introspection methods by an alternative approach.

Since usability experts are interested in the systematic responses of users to stimuli provided by a user interface, let us scrutinize what constitutes a stimulus. In general, a *stimulus* is any kind of sensory information that is received by an agent and is capable of evoking a response. In psychology, a stimulus is an independent variable defined and manipulated by an experimenter [42]. In usability analysis, the user interface sets the stimuli and the challenge lies in discovering which stimuli triggered which response — or lack of response.[1]

Given the purpose of analyzing the usability of graphical user interfaces, we are most interested in users' responses to *visual stimuli*. If the visual stimuli as independent variables are not carefully defined by an experimenter, but serendipitously by the user interface dynamics, how can we define and represent them and delineate one from the other?

We suggest to define a *visual stimulus* according to what users encountered *visually* on the screen and to delineate two states of screen displays into two different visual stimuli if users would note a remarkable difference between the two. Thus, a visual stimulus can be represented by a set of frames that are equivalent according to a human observer. While this definition allows for the intrusion of human subjectivity, we will later find that our annotation protocol leads to high inter-annotator agreement.

Corresponding to this definition of visual stimulus, *visual stimulus discovery* is defined as the task that automatically groups visually similar states, be they contiguous or disjointed within a single user session or resulting from different users' sessions.

1.2 Performing Usability Analysis with Visual Stimuli

Following their discovery, visual stimuli from dynamic user interfaces can be used as a tool for analyses by usability experts. The aim of this research work has been the development of a method that discovers suitable virtual stimuli in the recordings of user engagement on dynamic interfaces and its embedding in an overarching framework for usability analysis with visual stimuli. The usability expert employs our framework, whose core parts are formalized in Section 3, according to the following phases (also cf. Figure 2):

Phase 1: Setup. The usability expert sets up a user study. Interactions of study participants and videos of their screen appearances are automatically recorded and aligned (cf. Section 4).

Phase 2: Bootstrapping. In the bootstrapping phase, contiguous video frames are semi-automatically labeled either as belonging to one *stimulus shot*, i. e., a partial recording corresponding to one visual stimulus, or representing the boundary between two adjacent stimulus shots (cf. Section 5.1).

Phase 3: Training. In the training phase, a visual change classifier learns to mimic the usability expert decisions as to whether two contiguous frames should be considered as belonging to the same or different visual stimuli (cf. Section 5.2 and its evaluation in Section 5.3).

Phase 4: Application. In the application phase, the learned classifier is applied to video recordings in order to *(i)*, group contiguous video frames from one user session into stimulus shots and to *(ii)*, cluster stimulus shots from several user sessions that correspond to likewise user interactions (cf. Section 6 and its evaluation in Section 7).

[1] In typical applications of A/B testing, the stimulus is carefully manipulated by incrementally changing the user interface. However, such incremental changes are not applicable for all usability analysis questions and not during the whole life cycle of a software application.

Phase 5: Analysis. In the usability analysis phase, the usability expert models her analyses by defining AOIs on visual stimuli, visualizing results on visual stimuli and querying for aggregated measures on AOIs and on visual stimuli (cf. Section 8.1). We have surveyed the suitability of visual stimuli discovery for inclusion in the workflow of usability experts (cf. Section 8.2).

Phases 1 and 5 are basically identical to corresponding phases when using the virtual screenshots methods, which usability experts are already accustomed to in their current work routine. Phase 2 is a labor-intensive labeling process. This labeling process can be skipped if no domain-specific training is required, though domain-specific training may substantially improve the quality of visual stimulus discovery (cf. Section 5.3). We have developed a semi-automated annotation approach that significantly streamlines the labeling work (cf. Section 5.1). Phases 3 and 4 are fully automatic and correspond to the core method development of our work. We call our specific implementation of the visual stimuli discovery in this work *WebVSD*.

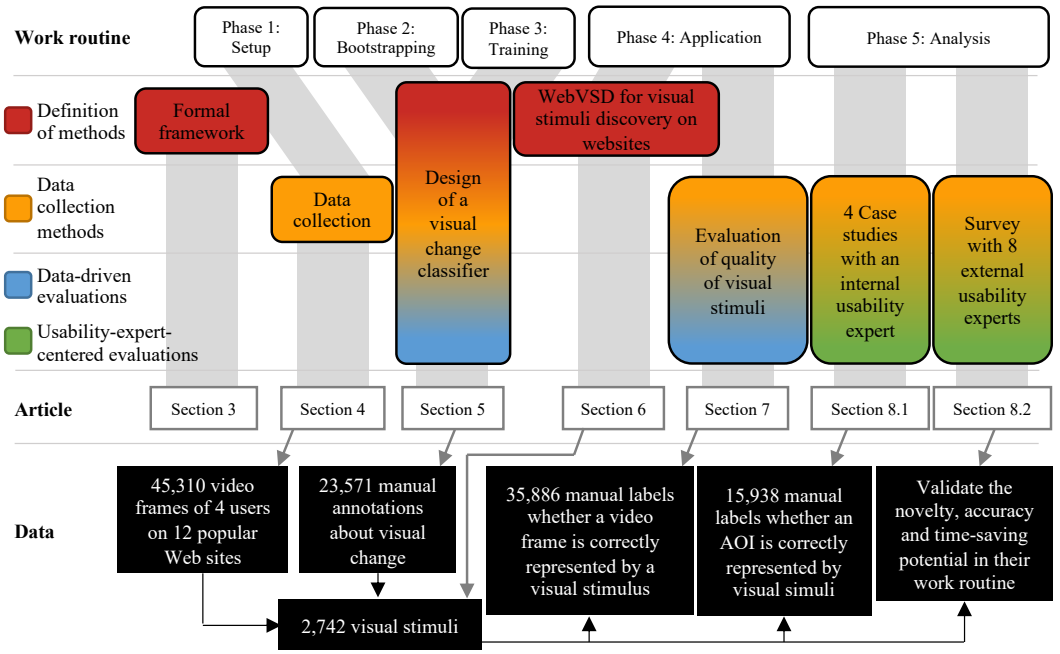


Fig. 2. Graph that shows the relations between the phases in usability analysis and the sections of this article.

1.3 Research Methodology Pursued in this Article

Research presented in this article pursues a complex undertaking. First, we define the new challenge of discovering visual stimuli that is rooted in the real-world needs of usability analysts. Second, we propose methods to address that challenge. From a research methodology point of view we then need to answer two corresponding research questions: *(i), are methods for visual stimuli discovery suitable to help usability analysts?* And, *(ii), is our method for visual stimuli discovery the right method?* These two questions cannot be completely disentangled. Responding to question *(ii)*, we have performed data-driven evaluations and comparison of methods for discovering visual stimuli to address the issue of *technical accuracy*. However, such an evaluation and comparison would

be void, if we did not show its usefulness, too. Responding to question *(i)*, we need to provide indications of the *usefulness* of visual stimuli discovery in usability-expert-centered evaluations, which is only possible, if we already have a corresponding method for visual stimuli discovery of sufficient quality.

Evaluating Technical Accuracy. We have borrowed evaluation methodology commonly applied in shot detection when evaluating visual stimuli discovery in Section 5.3 and Section 7. As is also indicated in Figure 2, we have created a dataset of substantial size, and we evaluate them with several complementary measures.

Indicators for the Usefulness of Visual Stimuli Discovery. Evaluating usefulness of visual stimuli discovery is more tricky. There is no suitable open-source usability analysis suite which we could have augmented by visual stimuli discovery to evaluate its usefulness. Also, it is out of scope of this research contribution to engineer such a comprehensive usability analysis suite that would integrate visual stimuli discovery for full-fledged investigation. Therefore, we have decided to evaluate usefulness by the following two schemes that provide complementary supporting indicators (cf. Section 8).

The first evaluation aims at describing indicators for usefulness when a usability expert applies visual stimuli discovery to real-world Web sites (cf. Section 8.1). The reader may note that such usefulness will not only depend on the Web site appearance and dynamics, but also on the objectives of the Web site operator and the task that a user aims to accomplish. For instance, a car registration site should allow the user to easily and efficiently complete the registration, while an online shop might rather want to entice the user to browse for and buy products. Given the huge combinatorics of possible appearances, dynamics, operator objectives and user tasks, our aim for evaluation cannot be the investigation of a representative sample as exploring the notion of representativeness itself would already exceed the scope of this paper by far. However, with four Web sites and questions involving AOIs that usability experts are typically interested in, we claim we can cover substantial ground to derive reasonable indicators for usefulness of the proposed methods.

While in the first evaluation concerning usefulness, we rely on our own experience and knowledge about what usability experts are typically interested in, in the second evaluation we survey independent usability experts for their opinion about the usefulness of our method (cf. Section 8.2). To this end, we have presented usability experts with an interactive online survey that allowed the usability experts to understand the method as applied in a typical work context in order to judge its usefulness. The experts provide us with rich feedback about the usefulness of our method in improving their work routines.

Ethical Considerations of the Dataset Collection, Dataset Annotation, and Evaluations. All human-related studies as part of the data collection, data annotation, and evaluations have been performed while the first, second, and last authors, who supervised the studies, were at the University of Koblenz, Germany. The university did not have an ethical commitee at that time. However, we had prepared user consents according to best practices in HCI-studies, explaining each study to the participant in detail. Every participant who participated in the study has signed such a consent.

1.4 Contributions of this Article

To summarize, this work comprises the following contributions:

- We describe the novel challenge of *visual stimuli discovery* on dynamic graphical user interfaces.
- We define a *methodology for acquiring data* for visual stimuli discovery and apply it on real-world Web sites with WebVSD.

- We *define a framework and methods* for visual stimuli discovery.
- We *evaluate* the methods and the framework using *various measures related to technical accuracy*.
- We *evaluate* the framework with regard to *usefulness by several case studies as well as by a survey* of usability experts.[2]
- We provide WebVSD as *open-source software* to the research community including the tools to create a visual stimuli discovery framework and to evaluate it.[3]
- We provide a *rich data set* of user sessions on dynamic Web sites including all annotations and outputs from the execution of the visual stimuli discovery.[4]

2 RELATED WORK

The primary goal of this work is to enhance the efficiency of UX experts in their tasks. In this regard, it partly aligns with various other approaches aimed at automating usability and UX studies that share a common vision [2, 34, 88]. For example, the utilization of visualization [15], machine learning [2, 34], and numerous other AI techniques [88] to automate UX processes. Compared to these automation tools and methods, it is essential to highlight that our approach prioritizes accurately depicting user interactions with an interface. As a result, our approach is more specifically comparable to specific techniques for visual stimuli discovery to improve usability analysis with dynamic interfaces. In the following sections, we delve deeper into related work concerning the understanding of user sessions on dynamic states of interfaces for usability analysis. First, we review methods of introspection that interprets the code to render the interface on a screen. Second, we review approaches of virtual screenshots that consider the interface as it is a stimulus to the user on a screen. Finally, we discuss the methodology of shot and scene detection, as we employ that methodology for defining our approach to visual stimuli discovery.

2.1 Introspection and Interaction Recordings

We call methods that interpret the interface-defining code *introspection methods*. Introspection methods rely on the fact that interfaces are defined by code in languages and corresponding libraries which have been specifically designed for the purpose of interface definition (HTML and CSS, Microsoft WinUI with XAML, Apple Cocoa with Objective-C, Java Swing, Qt with C++). The interface-defining code entails all information about elements in an interface and their attributes like type (e. g., text, button, slider, or image), position (e. g., x-y-coordinate on screen), size (e. g., width and height), color (e. g., font color for a text or area color of a button), and interactivity (e. g., mouse hover, mouse click). Sometimes this code is available for parsing (like on the Web with HTML and CSS) or often the code can be retrieved via accessibility APIs like the IAccessible2,[5] the Microsoft System.Windows.Automation framework,[6] or the Assistive Technology Service Provider Interface.[7] We elaborate on related work that uses introspection for segmenting contents in an interface, associating interactions with an interface, crawling of interface designs, and enabling the usability analysis on interfaces. Moreover, we provide insights from our past work on introspection of interfaces.

[2] https://github.com/raphaelmenges/vsd-expert-survey
[3] https://github.com/raphaelmenges/visual-stimuli-discovery/tree/master/code
[4] https://zenodo.org/record/5031618
[5] https://accessibility.linuxfoundation.org/a11yspecs/ia2/docs/html
[6] https://docs.microsoft.com/en-us/dotnet/api/system.windows.automation?view=net-5.0
[7] https://en.wikipedia.org/wiki/Assistive_Technology_Service_Provider_Interface

Segmenting Interface Contents. Introspection has been used to segment interfaces into multiple blocks. The blocks may be defined to contain elements that look similar or offer a certain functionality (e. g., message list, item in a shop). One might imagine to apply a segmentation iteratively to the interface during recording and compare the segments over time to understand changes in an interface. Cai et al. [18] described one of the most popular segmentation methods for Web pages. Their algorithm employed a top-down approach by parsing the document tree and applying heuristics to identify elements that might separate blocks along horizontal or vertical axes. Even though the algorithm was called "Vision-based Page Segmentation" (VIPS), Cai et al. did not consider the rendering of the Web page but instead interpreted styling rules like `borderColor` via introspection. There had been attempts to improve the algorithm by adding more rules to the heuristics [3] and going beyond introspection by applying image processing on the rendered interface [28]. However, no overlapping elements can be considered by these methods and dynamics in interfaces have not been regarded, yet.

Associating Interactions with an Interface. Introspection has also been used to associate interactions like mouse clicks and eye gaze data with an interface. Burg et al. [17] proposed a tool to support Web developers in understanding visual changes of elements on Web pages. The tool allowed for automatically tracking changes of a selected element and it revealed the code snippets that caused the visual change. However, a prior selection of elements is not feasible for a usability study because a usability expert might not anticipate which parts of the interface will cause problems during user interactions. Lamberti et al. [58] described a method to aggregate and display the intensity of user attention Web pages with element-aware heatmaps. Before recording interaction, they simulated mouse clicks and structural changes on the document tree, which were rendered in the viewport. The resulting visuals were stored on a server-side recording tool. During the interaction recording, they gathered dynamic changes in the document tree at user events or via frequent polling of the document tree. However, neither did they clarify how the changes in the visibility of elements were recognized, nor did their evaluation cover dynamic interfaces. Hienert et al. [47] processed a Web page at its textual level and mapped eye gaze data to the fixated words. They estimated which words have been read more often by users and which parts of the text on a Web page were ignored by users. They took a Web page document as input, rendered it with a Web browser, and queried for the coordinates of the words on the rendering of the Web page. Then, they stored for each word the number of fixations that fell onto it. To compensate for slight layout changes and animations, they merged the same words among different users within a context of 50 characters in the text before or after. They did only consider textual contents. Neither did they consider the formatting of the text nor did they consider any other elements on the page. There have been further works to associate interaction data with single elements on a Web page [10]. However, none of the methods create visual stimuli from dynamic interfaces that allow aligning the interactions by multiple users.

Crawling the Design of Interfaces. Kumar et al. [54] presented "Webzeitgeist," a framework that allowed for querying visual properties of Web pages and elements represented in the document tree and stylesheet. The framework consisted of a database that stored the document tree, styling attributes, a rendered image, and computer vision features from the rendered image of each Web page. While the database provided a rendering of each Web page, the contents of the document tree itself were not checked for visibility in the rendered image. Dynamics that occurred during interaction by users were explicitly excluded from the crawling process. Recent methods for understanding the design of interfaces by using huge data sets using deep learning [21, 49] have similar limitations in regard to dynamics.

Synthesis of Viewport-depending Layouts. With the arrival of novel devices of different sizes and shapes, the diversity of screens has increased. The adaption of user interfaces to these different screen sizes requires further effort for interface designers and usability experts. Several approach partly automate the generation of different layouts [50, 53, 66] using introspection. So far, however, these approaches do not automatically adapt the full dynamics of user interfaces.

Methods and Tools for Usability Analysis. Another approach to capture the interaction with Web pages is the recording of user actions at document tree level, called *session replay*. One can record the document tree on the client side, e. g., using the Mozilla Firefox Web browser and the WebReplay[8] functionality. However, the technique was designed to support Web developers in analyzing the behavior of individual elements when a single user interacts with them. Alternatively, there are frameworks like LogRocket[9] or mouseflow[10] that require a script to be integrated into a Web page, yet can record the sessions of every user who accesses the Web site. But the integration of a script limits the approach to Web sites to which a usability expert has administrative access. In general, session replays do not automatically create visual stimuli from the recordings that could be aligned between multiple users.

Authors' Prior Work. We have developed a Web browser that can be controlled entirely by eye gaze [68]. In this Web browser, we augmented the interactive elements of each Web page with gaze-sensitive icons, such that a user could select an interactive element conveniently with eye gaze. We provided the user with an appropriate interaction mode after the selection of an interactive element, e. g., an eye typing keyboard when the user selected a text input field. We developed an efficient introspection approach using a Mutation Observer in JavaScript to gather the necessary knowledge about interactive elements in real-time. The Mutation Observer informed us about changes in the document tree of a Web page related to interactive elements like hyperlinks, text fields, or videos. We showed that the adaptation in our Web browser significantly improved the performance and usability of eye-gaze-based interaction in comparison to approaches that emulate mouse and keyboard events using eye gaze. While our past work [68] made heavy use of introspection, it also showed us the limitations of introspection. We found that on some Web pages it was difficult to understand the visibility of interactive elements. For example, on the front page of Google search multiple text input fields were stacked one over the other. At least one of the text input fields was used for displaying suggestions in a gray font, while the other text input field expected the actual user input. The order of rendering was defined by the `z-index` property. Because of the frequent use of Google search, we accommodated such idiosyncrasy by implementing a Google search-specific heuristic in our Web browser. However, such heuristics does not generalize to complex visibility configurations. Moreover, visual stimuli discovery must go far beyond the introspection of interactive elements and rather consider all elements that are displayed on a Web page.

Takeaway. To summarize, introspection has been successfully applied in the past to track interaction on single, static elements of an interface. However in recent years, the means for defining interfaces in general and especially Web pages have become more and more complex with the introduction of standards like HTML 5[11] and CSS 3.[12] Developers can implement a specific look and behavior on an interface in various manners. The multitude of variations on how to implement

[8] https://developer.mozilla.org/en-US/docs/Mozilla/Projects/WebReplay
[9] https://logrocket.com
[10] https://mouseflow.com
[11] https://html.spec.whatwg.org
[12] https://www.w3.org/Style/CSS/current-work.en.html

content, design, and interactivity makes it difficult to cover and understand all aspects of an interface with an introspection-based recording. Investigating such methods on current, real-world Web applications we found that: *(i)*, they are difficult to adapt to changing standards in interface-defining languages and accessibility APIs, *(ii)*, it remains difficult to understand the implications of structural changes to the visuals on an interface as relevant to a usability expert, and *(iii)*, it remains difficult to aggregate a very large number of structural states into meaningful visual stimuli to which the interactions by multiple users can be meaningfully associated. Therefore, we had to reject the idea that the recording of changes in the internal structure of an interface allows for reproducing useful visual stimuli that cover the ensemble of many elements.

2.2 Virtual Screenshot and Interaction Recordings

Interfaces are traditionally displayed on a two-dimensional screen, like a computer monitor or a smartphone. A screen consists of a pixel matrix, in which each pixel emits portions of red, green, and blue light. Depending on the interface to be displayed, the portions of emitted light are adjusted throughout the pixel matrix. The composition of the pixel lights from the matrix are perceived by a human viewer as interface. These pixels can be composed to virtual screenshots that comprise the complete interface, which is possibly larger than what can be displayed to the user at once on a screen. Virtual screenshots can be used to overlay them with eye gaze paths or mouse traces from multiple users for usability analysis. In this regard, we relate this paper to work from visual analytics of eye gaze data, reverse engineering of interfaces from pixels, and methods and tools for the usability analysis, based on the interpretation of pixels. Moreover, we provide insights from our past work on virtual screenshots.

Analyzing Eye Gaze Data Visually. Visual analytics is a research field that aims to process complex data such that a human can visually interpret it, i. e., using virtual screenshots. Kurzhals et al. [56] presented a visual analytics method for the analysis of eyegaze data from head-mounted eye-tracking devices based on automatic image comparison and clustering. First, they cropped thumbnails of the foveated region from the video recording at every fixation. Second, they aggregated the thumbnails of subsequent video frames into a segment until the similarity fell below a preset threshold. Third, they computed a similarity matrix between all segments across the recordings using SIFT-feature histograms and color histograms. Fourth, they clustered the segments according to their similarity using unsupervised spectral clustering. Then, analysts could annotate the segment clusters semantically and might combine them toward areas of interest. Their method was designed to work on the video recording of physical objects in a real environment. Similar works employed deep neural networks to identify objects in the foveated region of an ego-centric video stream [7] to automatically mark areas of interest in the video stream. However, these methods do not include the surroundings of fixations in the virtual screenshots. Therefore, an interface would be not captured as a whole because neither spatial relations are preserved nor scroll offsets are compensated. However, a usability expert needs to work on a representation of an interface that looks similar to what has been experienced by the users. Only then a usability expert can interpret the attention in the context of all content that has been visible to the user, including the possibility to observe that some content was ignored by users.

Reverse Engineering of Interfaces. Methods for reverse engineering of interfaces attempted to recognize elements through interpretation of pixels and template matching alone [6, 20, 35, 75, 76, 101]. Recently, deep learning has emerged as a promising method to detect and identify the elements of an interface from screenshots. Zhang et al. used screenshots as input to deep learning methods to recognize interface elements in iPhone applications to make them accessible with a screen reader [104]. They have built an extensive data set of iPhone app screens and combined

structural information and human annotation to label interface elements on each screenshot. There are further works that employ deep learning to generate vector shapes from interface screenshots to make designs editable [90], to generate code from interface screenshots [9, 22, 25], to draw mockups that accelerate the development process [74], or to enable visual design search moving beyond text-based search queries [14]. All of these methods expect interfaces to originate from similar toolkits and required layouts without overlaps. None of the methods combine multiple screenshots to produce virtual screenshots that can align the interactions by multiple users.

Methods and Tools for Usability Analysis. Deka et al. recorded and crawled interactions on Android app interfaces [30–32]. They captured screenshots every time pixel values changed on the screen. They did not compose virtual screenshots nor estimate the changes in the interface with regard to a usability analysis. Simko and Vrba [83] presented a method to support usability analysis of app-prototype interfaces on a smartphone with eye tracking. In their approach a usability expert defined *reoccurring scenes* on the interface before the study. Each reoccurring scene was represented with one screenshot. During the study, they recorded the interactions with the app-prototype interface on a smartphone using a camcorder. For each frame from the video recording of the camcorder, they cropped the screen content and assigned the frame to one of the predefined reoccurring scenes. They employed scale-invariant feature transform (SIFT) and structural similarity (SSIM) features to map the frames to the reoccurring scenes and heuristically set a threshold for the mapping process. Their method did not discover scenes automatically but expected all scenes to be predefined by a usability expert. Simko and Vrba only evaluated a banking app prototype interface with very little dynamics in their experiment and each scroll offset was regarded as unique scene.

Feiz et. al [40] have identified the classification of visual change from screenshots as interesting problem, too. They aimed to improve the automatic crawling and testing of mobile applications with reliable comparison of screenshots and efficient grouping screenshots. They let crowd workers interact with 1,110 iPhone applications and collected 77,655 unique screenshots. Another set of crowd workers annotated the assignment of individual screenshots from each application to clusters of screenshots that originate from the same screen. They employed an company-internal QA team to check and improve on the annotations, which implies a high quality of data set. From this annotated data set, they trained two models: *(i)*, A screen similarity model that combines an UI object detector [104] with a transformer model to recognize instances of the same screen from a collection of screenshots from the same application.*(ii)*, A screen transition model using a Siamese network architecture to identify the dissimilarity between screenshots and the transition to three types of system events, namely virtual keyboard, dialog boxes, and scrolling of the screen.
Their evaluation results for *(i)*, the screen similarity model with an F_1 score of 0.83 and *(ii)*, for the screen transition model with an F_1 score of 0.71 are comparable to our results (cf. Table 4). The authors published the guideline for annotators as additional material,[13] but to the best of our knowledge neither their collected data set nor their models. Their guideline for annotators is composed of on-hand examples in iPhone-specific UI designs. Feiz et. al also state that their dataset rarely includes scrolling activity, which is very different from our work. Last, they do not compose a visual stimuli from the clustered screenshots. Thus, interaction data cannot be straightforward aggregated for an analysis by an usability expert.

In contrast, many tools show eye gaze data of multiple users as heatmap on a virtual screenshot, like sticky.ai [1], eyezag [41], CoolTool [27], or realeye [86]. Tobii Pro Studio [92], a tool by the market leader of dedicated eye-tracking devices Tobii AB, also offers eye gaze mapping on a recording of a user session on a Web site. However, Tobii explicitly recommends watching the

[13] https://dl.acm.org/action/downloadSupplement?doi=10.1145%2F3490099.3511109&file=guidelines.pdf

interaction with dynamic elements on a video recording instead of the virtual screenshots, because the dynamics are not covered in their generated virtual screenshots:

> "...if the participants access a drop-down menu on the website [,] the viewing patterns on that menu will be recorded and aggregated on the screenshot of the webpage, but the menu itself won't be visible on that screenshot."[14]

Since more than 10 years ago there are approaches to apply object tracking algorithms to identify and track AOIs in video recordings, especially in the context of eye gaze tracking using head-mounted cameras [13, 93]. Until now they have been neither implemented in usability tools nor have been broadly used by usability experts.

Authors' Prior Work. We have proposed a method [69] to aggregate mouse and gaze data from multiple users onto a single stitched screenshot that served as a virtual screenshot. We took screenshots of the browser viewport at predefined time intervals during a user session, cropped viewport-relative elements from the screenshot, stitched the screenshots according to the scroll offset, and finally composed the screenshots together while depicting each viewport-relative element only once. While our method accounted for user interaction through scrolling and dynamic loading of content on the bottom of the interface through JavaScript, further dynamics of an interface like interactive navigation menus or photo carousels were not addressed. Nevertheless, we employ the idea to separate the treatment of viewport-relative elements from the rest of the interface also in this paper.

Takeaway. To summarize, virtual screenshots do not properly handle the dynamics of interface. Only the previous works in visual analytics [7, 56] include extensive dynamics. However, their methods do not aim to provide a complete images of the interface as required for a comprehensive usability analysis. We aim to build on the concept of comprehensive virtual screenshots with the visual stimuli discovery, yet handle complex dynamics in interfaces.

2.3 Detecting Shots and Scenes in Video Recordings

Since our approach delineates visual stimuli from a video recording, resulting in a series of stimulus shots, we briefly survey related work on shot and scene detection in video recordings. Scene detection is a well-established field of research that aims to cluster different shots within a video or movie into scenes of coherent environment, time, actors, and story [72]. A video can be cut into a series of scenes, which itself can be separated into shots. A shot is the recording by one camera from a specific position and angle. A scene is composed based on a coherent environment and set of actors. There are cases in which two or more shots are interwoven, e. g., through frequent switches between shots that serve the dramatic purpose. Thus, methods for automatic shot detection and clustering of shots and scenes constitute research relevant for addressing the problem of stimuli discovery. Simko and Vrba [83] applied scene detection methods for visual stimuli discovery. They classified frames from a video recording of the interface into before-known visual stimuli. Instead, we want to investigate which features from shot and scene detection are useful for our purpose of delineating visually coherent states of an interface into visual stimuli that are unknown before the analysis.

Takeaway. Popular features for shot and scene detection like face detection, audio processing, and subtitle analysis [33] do not apply to our use case. However, various computer vision features that are used to detect changes in lighting, setting, and environment in videos might signify visual

[14] https://www.tobiipro.com/learn-and-support/learn/steps-in-an-eye-tracking-study/data/Aggregating-eye-tracking-data-across-several-participants-in-web-recordings, accessed on 31st May 2021.

Table 1. Comparison with related work. ●, ◗, and ❍ encode "yes", "partly", and "no", respectively.

	Reference (↑ year)	Domain	Data Set	Dynamics	Overlapping Contents	Preserves Context	Code-based	Pattern Matching	Deep Learning	Computer-vision Features
Introspection and Interaction Rec.	Cai et al. [18]	Web	140 crawled pages	❍	❍	●	●	❍	❍	None
	Beymer and Russell [10]	Web	1 page	❍	❍	●	●	❍	❍	None
	Kumar et al. [54]	Web	> 100,000 crawled screens	❍	❍	●	●	❍	❍	GIST, color, edges
	Burg et al. [17]	Web	1 page + 2 apps	●	❍	❍	●	❍	❍	None
	Cormier et al. [28]	Web	50 crawled pages	❍	❍	●	❍	❍	❍	Edges
	Lamberti et al. [58]	Web	4 pages	◗	❍	●	●	❍	❍	None
	Krosnick et al. [53]	Web	None	❍	❍	●	●	❍	❍	None
	Hienert et al. [47]	Web	143 pages	❍	❍	❍	●	❍	❍	None
	Chen et al. [21]	Mobile	68,702 crawled screens	❍	❍	●	●	❍	●	SSIM, color, OCR
	Huang et al. [49]	Mobile	3,802 sketches	❍	❍	●	❍	❍	●	None
	Menges et al. [68]	Web	None	●	❍	❍	●	❍	❍	None
	Jiang et al. [50]	Flexible	None	❍	❍	●	●	❍	❍	None
	Lukes et al. [66]	Flexible	9 pages at 20 different sizes	❍	●	●	●	❍	❍	None
Virtual Screenshot and Interaction Rec.	Yeh et al. [101]	Desktop	> 50,000 screens from books	●	❍	●	❍	●	❍	SIFT, OCR
	Dixon and Fogarty [35]	Desktop	None	●	❍	●	❍	●	❍	None
	Brône et al. [13]	World	None	●	❍	❍	❍	❍	❍	SURF
	Chang et al. [20]	Desktop	6 screens	●	●	●	●	●	❍	None
	Toyama et al. [93]	World	438 images	●	❍	❍	❍	❍	❍	SIFT
	Banovic et al. [6]	Desktop	34 tutorial videos	●	❍	●	❍	●	❍	None
	Kurzhals and Weiskopf [57]	World	6 images + 90 seconds video	●	❍	❍	❍	❍	❍	SIFT, color histograms
	Nguyen and Csallner [75]	Mobile	472 screens	❍	❍	●	●	●	❍	Edges, contours, OCR
	Barz and Sonntag [7]	World	None	●	❍	❍	❍	❍	❍	None
	Deka et al. [32]	Mobile	> 18, 000 crawled screens	●	❍	●	●	❍	●	None
	Deka et al. [30]	Mobile	72,219 (partly crawled) screens	●	❍	●	●	❍	●	None
	Deka et al. [31]	Mobile	15–50 users on 10 apps	●	❍	●	●	❍	●	None
	Menges et al. [69]	Web	4 pages	◗	◗	●	●	❍	❍	None
	Nguyen et al. [76]	Mobile	Unclear	❍	❍	●	●	❍	●	None
	Swearngin et al. [90]	Mobile	203 drawings	❍	❍	●	❍	❍	❍	Edges
	Beltramelli [9]	Mobile	Synthetic screens	❍	❍	●	❍	❍	●	None
	Chen et al. [22]	Mobile	85,277 crawled screens	❍	❍	●	❍	❍	●	None
	Simko and Vrba [83]	Mobile	30,660 frames	❍	❍	●	❍	❍	❍	SIFT + SSIM
	Chen et al. [25]	Mobile	46,202 crawled screens	❍	❍	●	❍	❍	●	None
	Moran et al. [74]	Mobile	191,300 crawled screens	❍	❍	●	●	❍	●	Edges, morph., contours, OCR
	Zhang et al. [104]	Mobile	80,945 crawled screens	❍	❍	●	●	❍	●	Color, OCR
	Bunian et al. [14]	Mobile	4,543 screens	❍	❍	●	❍	❍	●	None
	Feiz et al. [40]	Mobile	77,655 screens	●	◗	●	●	❍	●	None
	Our method	**Web**	**45,310 frames (12 pages)**	●	●	●	●	❍	❍	**53 features**

changes which are relevant our approach. Multiple approaches that successfully employed color histograms [94], edge-features [102], optical flow [46], and SIFT features [73] in the purpose of shot and scene detection in videos constitute candidates that we empirically investigate and compare in this paper.

We provide a comprehensive overview how existing approaches relate to our work in Table 1. The table provides for each work specifics about their domain and data set, whether the presented methods cover the various challenges in visual stimuli discovery (dynamics in the interface, overlapping contents in the interface, or preservation of context between elements in the interface), and techniques applied by the authors (code introspection, pixel-pattern-matching, deep learning on images, or traditional computer-vision features). Papers are grouped by general approach to the problem (introspection and interaction recordings and virtual screenshots and interaction recordings) and arranged in ascending order by year or publication.

3 FORMAL FRAMEWORK OF VISUAL STIMULI DISCOVERY

In this section, we give a high-level overview of our method and an abstract formalization of its major components. We aim to support the usability analysis of dynamic interfaces which exhibit rich interaction behaviors, choreographing dynamic elements with changing content on user responses. For such usability analysis, visual stimuli cannot be defined beforehand. We propose a framework to discover visual stimuli (semi-) automatically. Figure 3 depicts our framework of visual stimuli discovery in its application phase. In the example, three uses interact with the same interface. We record a video of the screen contents during their interaction. These video recordings are first split into stimulus shots per user and then the stimulus shots are clustered across the users into clusters of stimulus shots. These clusters are then represented by single images, which can be worked on by a usability expert and which we call visual stimuli. The remainder of this section formalizes the depicted framework from left to right:

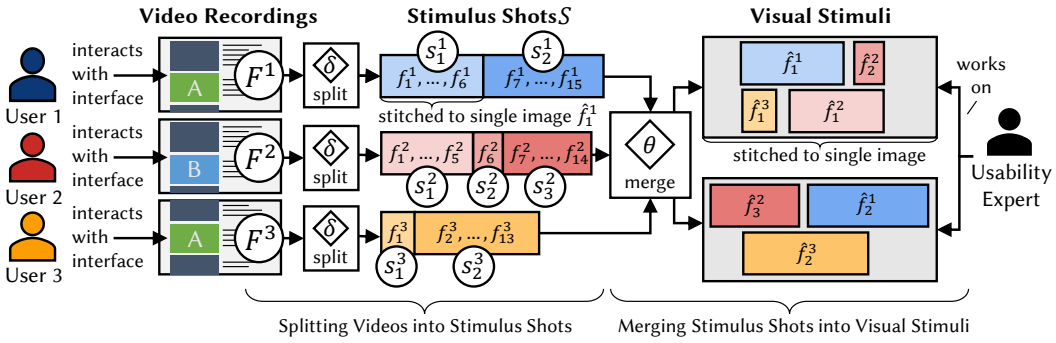


Fig. 3. Application phase for visual stimuli discovery considering three user sessions.

Splitting Videos into Stimulus Shots. We define the video recording F^i of a user session i to consist of a sequence of frames $F^i = (f_1^i, f_2^i, \ldots, f_{n_i}^i)$. For this, we define a discrete classifier δ that detects visual changes between two frames f_a, f_b. The aim is that $\delta(f_a, f_b)$ returns 0 if the two frames are visually so similar that they should be considered to belong to one visual stimulus, and 1 otherwise.

We partition a complete video recording F^i of a user session i into a totally ordered set of stimulus shots $S^i = (s_1^i, \ldots, s_j^i)$, where for every $s_j^i \in S^i$ there are k and l such that

$$s_j^i = (f_k^i, \ldots, f_{k+l}^i \mid \forall m \in [k, k+l-1] : \delta(f_m^i, f_{m+1}^i) = 0 \ \wedge \ \delta(f_{k-1}^i, f_k^i) = \delta(f_{k+l}^i, f_{k+l+1}^i) = 1).$$

For the first and the last stimulus shot, we apply a slightly simplified formalization that drops the condition $\delta(f^i_{k-1}, f^i_k) = 1$ and $\delta(f^i_{k+l}, f^i_{k+l+1}) = 1$, respectively. Then, we define $\hat{f}^i_j$ as a stitch of all frames in s^i_j. We define the set of all stimulus shots from all video recordings as $S = \cup_i S^i$.

Merging Stimulus Shots into Visual Stimuli. Our goal is to merge visually similar stimulus shots within and between user sessions toward a visually coherent visual stimulus. Thus, we perform agglomerative clustering on the stimulus shots S across all users using a discrete distance function θ. We define θ between two stitches $\hat{f}_a$, $\hat{f}_b$ as

$$\theta(\hat{f}_a, \hat{f}_b) = (1 - \delta(\hat{f}_a, \hat{f}_b)) \cdot A(\hat{f}_a, \hat{f}_b).$$

If both stitches are visually different, δ is 1 and θ becomes 0. If both stitches are visually similar, δ is 0 and θ becomes A. The function A computes the area of overlap between the two stitched frames $\hat{f}_a$ and $\hat{f}_b$ and is measured in the number of pixels. The function serves as a similarity score between two stitched frames. It allows us to cluster stimuli shots that cover bigger portions of an interface. The output of the clustering is a set of stimulus shot clusters. We stitch the stitched frames of the stimulus shots of each cluster into a single stitched image, which we call a visual stimulus. A usability expert then works on the visual stimuli.

4 RECORDING A DATA SET OF USERS SESSIONS ON DYNAMIC INTERFACES

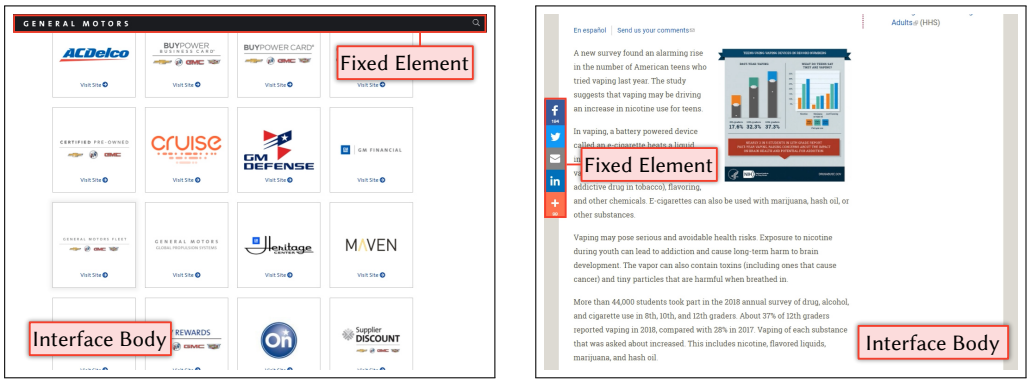
We have created a data set from the recording of study participant interactions with dynamic interfaces, which corresponds to **Phase 1: Setup**. We decided to use popular, rich, and complex Web sites as representatives for dynamic interfaces. The data set is segmented into sessions. We have recorded a video and a datacast for each session. The video contains a recording of the interface viewport on the screen as it has been displayed to a participant. The datacast conveys information as extracted from the introspection of the interface, alongside the interaction recordings. The data set has been published on the Zenodo platform[4] under a CC0 public domain license.[15]

Creating a Recorder. We have developed a logger application based on the Qt [26] framework and its integrated Chromium-based [4] Web engine. The video recording was captured with five frames per second from the Qt environment via FFmpeg [5], using VP9 [91] encoding.[16] The direct capture from the Qt environment allowed us to exclude the mouse cursor from the video. We have logged mouse cursor movements, mouse clicks, scroll events, and eye gaze data as interaction recording. Scrollbars have been hidden for the recording, as they would have disturbed the features. We identified navigation menus or advertisement banners that stay on the same viewport position while a user scrolls, aka viewport-relative elements or *fixed elements* [69], by their `position` property either set to `fixed` or `sticky`. Moreover, we checked the visibility of fixed elements by *(i)*, checking for the property `display` to be different from `none` *(ii)*, checking for the property `visibility` to be different from `hidden` and checking for the property `opacity` to have a value higher than 0.4. Thereafter we queried for the position and extents of the fixed elements through `getBoundingClientRect()`. This procedure was triggered every 50 milliseconds during a recording. We are referring to pixels that do not belong to a fixed element as *interface body*. Each pixel of a frame either belongs to the interface body or to a fixed element. See Figure 4 for examples.

The display had a size of 24 in. and a resolution of 1,680 × 1,050 pixels. The Web browser interface has been limited to basic browsing controls like an address bar, back and forwards navigation,

[15] https://creativecommons.org/share-your-work/public-domain/cc0

[16] We have observed that the video encoder skipped in total 1,270 single frames (2,7% of recording time) during the recording. The frame loss was distributed uniformly across the duration of all sessions. This loss has no impact on the evaluations presented in this work.



(a) The Web site of General Motors has a fixed element that stays on top of the viewport at scrolling.

(b) The Web site of NIH has a fixed element that provides links to their social media accounts.

Fig. 4. Pixels of Web sites in a viewport are either assigned to the interface body or to a fixed element.

recording facilities, and a viewport to the Web page with a resolution of 1,024 × 768 pixels, similar to [28]. The eye gaze data was recorded with a Tobii 4C eye-tracking device, which captures eye gaze at a frequency of 90 Hz. See Figure 5 for the setup.

Defining the Recording Procedure. We have chosen twelve Web sites using English language from four different categories, inspired by the Alexa Top 50[17] categories. The category "Shopping" consists of the sites "walmart.com" (Walmart), "amazon.com" (Amazon), and "store.steampowered.com" (Steam). The category "News" comprises "reddit.com/r/pics/top/?t=month" (Reddit), "edition.cnn.com" (CNN),

[17]https://www.alexa.com/topsites

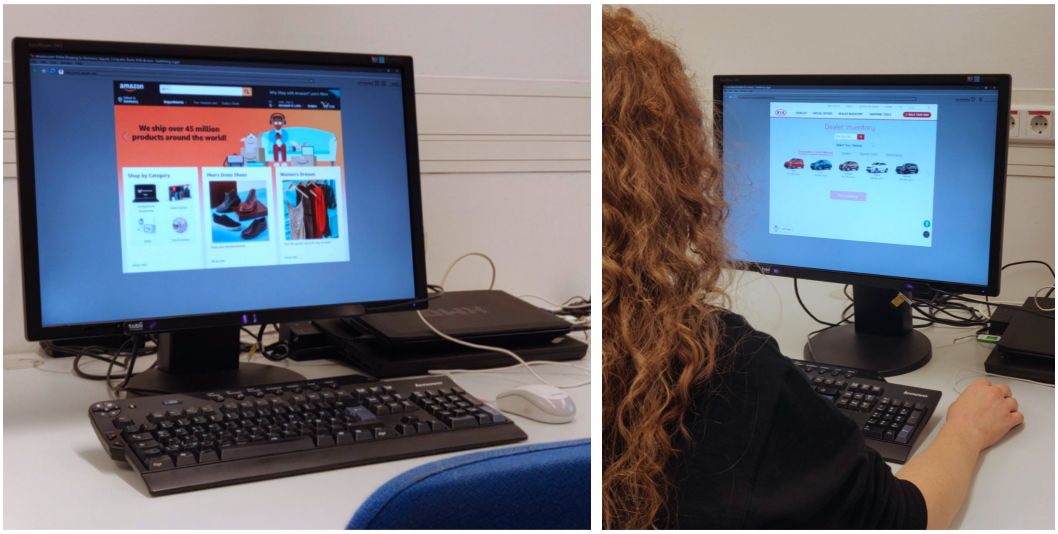


Fig. 5. The recording setup consisted of a monitor with remote eye-tracking device mounted below the screen and mouse and keyboard for input. The photo on the right was taken during the recording.

and “theguardian.com/international” (Guardian). The category “Health” includes “nih.gov” (NIH), “webmd.com” (WebMD), and “mayoclinic.org” (MayoClinic). The category “Cars” consists of the sites “gm.com” (General Motors), “nissanusa.com” (Nissan), and “kia.com/us/en/home” (Kia). We have defined a protocol for the participants that instructed them to explore each page of each site we included thoroughly and trigger dynamic behavior, i. e., hover over elements and menus with the mouse, read descriptions, and click through photos of a photo carousel. For each site, the participants started the interaction on the landing page. Then the participants were given the task to find a specific hyperlink on the landing page and navigate to the next page. On the next page, they were asked to explore the page and choose a hyperlink of their own choice as per their interest. The data set was not designed to uncover usability flaws on these interfaces, as there is no contextual task given to the participants on each site. Instead, we aimed to create a data set that would comprise many real-world dynamics a usability expert may be confronted with within a behavioral study. See Appendix A for details about the protocol.

Participants in the Recording. Four male participants (age = 30 ± 2.35 years) were invited for the data set recordings. We decided to limit the recording to four participants due to the annotation effort for the sessions of each participant. All four participants are researchers in computer science and experienced Web users. They participated voluntarily in the data set recording. The strategy to invite experienced Web users was motivated by the nature of sophisticated tasks, e. g., to explore the dynamics of pages thoroughly, however, not to leave the page accidentally through the activation of an outgoing hyperlink. Many menus require a mouse click for expansion, while other interface elements lead to another site if clicked. Only experienced Web users can make reasonable predictions of these design choices.

Results of the Recording. We recorded 155 minutes of Web browsing in about 1.23 GB of video recordings and datacasts. All videos together contain a total of 45,310 frames. We have recorded 10,742 scrolls, 111,058 mouse movements, 893 clicks and 837,716 eye gaze points. The participants have browsed 172 URLs, which means on average 3.56 URLs per session. This reflects our protocol, considering that some participants clicked by accident on outgoing hyperlinks. The fixed elements as encoded in the datacasts make up about 15.36% of the pixels in the video recordings. In the remainder of the paper, we refer to the participants whom we have observed for data collection as “users.”

5 DESIGNING A VISUAL CHANGE CLASSIFIER

The visual change classifier δ is required in the framework of visual stimuli discovery for both splitting and merging of video recordings toward visual stimuli. When coming up with a visual change classifier, the following two research questions must be answered:

> **Research Question 1 (RQ1)**: How can we formalize the decision model of a usability expert about visual change?
>
> **Research Question 2 (RQ2)**: Which computer vision features and classifiers are best suited to decide about visual change?

We first define a decision process from the point of view of a usability expert. Then we apply this decision process to the collected data and perform cross annotator validation, addressing RQ1. Considering the collected and annotated data, we introduce computer vision features from shot and scene detection literature and discuss different options for classifiers using these features. Then we evaluate the proposed features and classifiers, addressing RQ2.

5.1 Annotating Visual Change in the Collected Data

We have designed a decision process that reflects the point of view of a usability expert whose objective is to aggregate interactions on distinctive visual states of an interface, as known, e. g., from tools like Tobii Pro Studio [92] or EYEVIDO lab [38]. This corresponds to **Phase 2: Bootstrapping**. See Figure 6 for the decision process.

For annotation of the visual change according to the decision process, we prepare the collected data as depicted in Figure 7. For each (a) consecutive pair of frames from the video (e. g., frames n-1 and n, frames n and n+1, ...), we (b) crop the fixed elements from the frames by accessing the DOM, as described in Section 4. In the figure this results in one region that belongs to the interface body referred to as `html/body` and one region that belongs to a fixed element referred to as `html/body/nav`. The regions are treated separately in the following — similarly to the enhanced representation method [69]. Then, (c) each pair of regions is treated as an *observation*. We assume that scrolling should not affect the determination of visual change, because we later stitch together two frames from different scroll offsets. Therefore, we transform the regions from the interface body to match in their scroll offset and we crop only the overlapping image portions for further consideration. See Appendix B for details about our approach for estimating the scroll offset. Each observation is then (d) manually annotated for visual change ("yes"/"no").

We have developed a tool as part of WebVSD[18] that provides a graphical interface to walk through the observations of a user session and let a usability expert annotate the visual change following the described procedure. See Figure 8 for a screenshot of the tool as used in the annotation. We believe that the decision model about visual change cannot be applied in the scope of a crowd working annotation, as it requires a deep understanding of which visual changes usability experts are interested in. Therefore, the first author of this paper (an experienced researcher in usability analysis) performed the task of annotating the data set over 16 hours within three days. We automatically skipped the annotation of observations that perfectly match every pixel, as the images are visually identical and would unnecessarily consume annotation effort. This applies to 62,283 out of the total of 86,791 observations. Thus, 23,571 observations have been annotated, from which 4,446 observations have been annotated as visually different.

To verify the objectivity of our decision process, we also let two students annotate a subset of the data set. The students were not involved in the development of the visual stimuli discovery. They were introduced to the task with an explanation of the decision process on a sheet of paper showing Figure 6 from page 19 and an in-person explanation of the annotation on one exemplary user session. They annotated a subset of 12 user sessions out of the total of 48 user sessions, i. e., the shopping-related Web sites of the first user, the news-related Web sites of the second user, the health-related Web sites of the third user, and the car-related Web sites of the fourth user. It took them about three hours, each. We report a Fleiss' Kappa score of 0.71 in comparing the annotations, which is to be considered as a substantial agreement according to Landis and Koch [59]. Thus, we successfully formalized a decision model of a usability export about visual change, resolving RQ1.

5.2 Detecting Visual Change using Computer Vision

The goal of detecting visual change with computer vision is to reproduce the human-based decision process from Figure 6 with the video recordings as the sole input. For this, we process our dataset, extract computer vision features, and develop classifiers to detect visual changes based on the investigated computer vision features. This corresponds to **Phase 3: Training**.

[18] https://github.com/raphaelmenges/visual-stimuli-discovery/tree/master/code#trainer

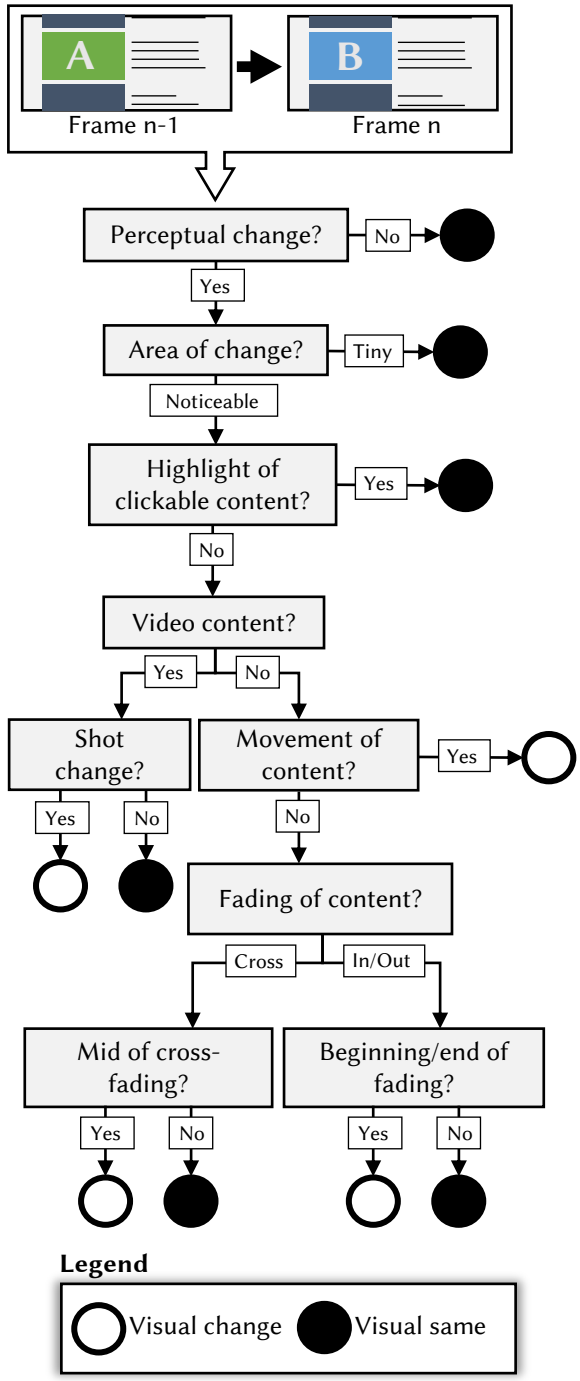


Fig. 6. Decision process for the annotation of visual change.

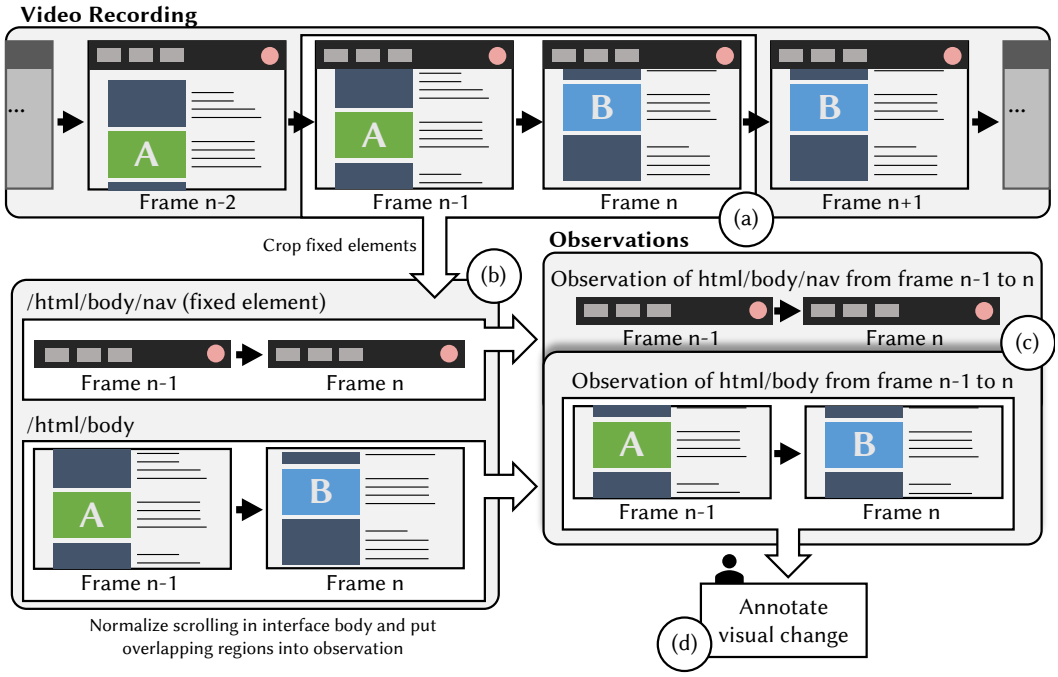


Fig. 7. Observations are extracted from contiguous frames in a video recording and annotated for visual change by a usability expert. We take (a) a pair of consecutive frames from the video recording and (b) crop the fixed elements. Moreover, we normalize the scroll offset in the interface body and consider the overlap in the following for the observation in (c). Last, (d) we label the visual change in each observation.

Preprocessing of the Data. We have removed 937 observations whose image overlap was below or equal to 32 pixels in either width or height. We consider the overlap of below or equal 32 pixels as not relevant for the training of the visual change classifier.

Features for Visual Change Detection. A variety of features to compare video frames is applicable [33]. We compute in total 53 computer vision features categorized into seven different types, deemed relevant to detect visual changes in an interface. See Table 2 for a listing of the features. Value-based features consider the difference of pixel values between two images either via aggregation of absolute value differences or via counting of changed pixels. This type of low-level feature accounts for every kind of visible change in an interface on a local scale. Histogram-based features consider the change in color distribution between two images on a global scale. This type of feature might support recognition of overall changes (e. g., dimming of background) on the interface, while local changes might go unnoticed. Edge-based features rely on changes in the visibility of edges between two images. Edges are an important aspect of human vision. Therefore, they may provide a strong indication of visual changes on an interface. Signal-based features like peak-signal-to-noise ratio (PSNR) and the mean structural similarity index [99] (MSSIM) are popular to measure image quality differences between two images. Optical-flow-based features consider the movement of content in video frames. This type of feature might be of interest for moving content like photo carousels and scrollable regions like chat windows. SIFT-based features could match descriptors of remarkable points between two images. This complex feature supports checking for changing contents at a layout level. Some interfaces predominately contain textual content. We detect text

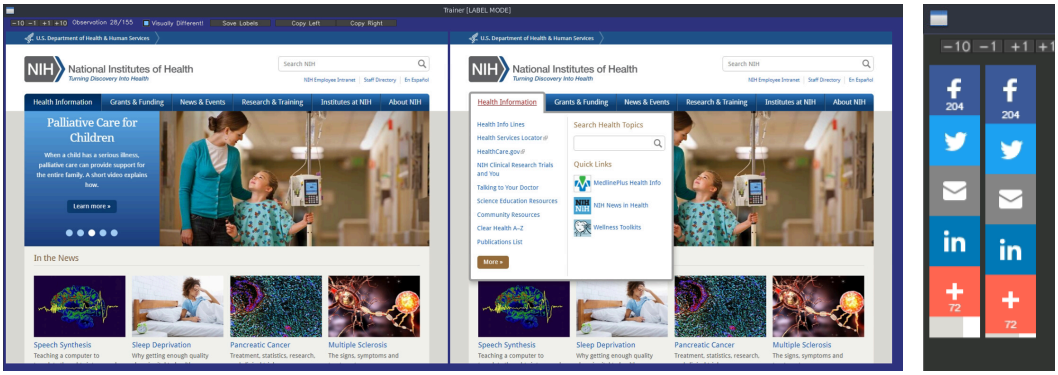


Fig. 8. Screenshots of the tool used to annotate visual change in each observation. On the left is a screenshot that shows the observation of the interface body in frame 110 and frame 111 on NIH in the session of the first user. The observation should be annotated as visually different, due to the open menu in the second frame. On the right is a screenshot that shows the observation of a fixed element in frame 317 and frame 318 on NIH in the session of the first user. The observation should be annotated as visual same because the visual change is not of relevance for usability analysis due to the rather small area of change.

using the Tesseract 4.0 optical character recognition library [84] and derive text-based features. We use the recognized texts both for bag-of-words and for character-level n-gram (with n = 3 [101]) similarity estimations. See Appendix C for more details about the features. We have computed the features using OpenCV [12]. Moreover, we have normalized all features independently with the min-max method. The min-max method normalizes all values of each feature in the training data between zero and one and applies the same transformation for the test data.

Choice of Classifiers for Visual Change Detection. Despite the trend toward deep learning in interface understanding [23, 100], we employ traditional classifiers in this work. This decision has been taken due to the limited amount of data we have available, as it has to be annotated by usability experts. Moreover, there exist well-received and ready-to-use automatic video shot segmentation tools [52]. One of the best scoring tools is the Multimedia Knowledge and Social Media Analytics Laboratory (MKLab) [71], which we have applied to our data set. The tool uses a score function, based on local (SURF) and global (color histograms) features, to compare consecutive frames for transitions. Applied to our data set, the MKLab tool detects only 175 shot separations through abrupt transitions, one dissolve transition, and 49 wipe transitions. In comparison, we have manually annotated 4,446 shot separations throughout the data set. Thus, we argue that existing video shot segmentation tools which look for explicit transitions cannot be applied to the problem of visual stimuli discovery and we need to investigate choices in features and classifiers for the detection of visual change as important to a usability expert.

We have evaluated a logistic regression classifier, a support vector classifier (SVC), and a random forest classifier. The SVC uses a radial basis function kernel and balanced class weighting using the synthetic minority over-sampling technique [61]. The random forest classifier makes use of 100 decision trees and entropy as a measurement for purity. We have also evaluated different tree counts, yet variations did not yield an improvement in classification. Analogously to the SVC, a balanced class weighting has been applied. Additionally, we have implemented a baseline classifier. For every training set, a threshold of the feature `count_bgr` is optimized regarding an average weighted F_1 score. The threshold determined in the `count_bgr` feature is then used to classify the

Table 2. Computer vision features we have evaluated to estimate visual change. Postfixes b, g, and r stand for the blue, green, and red color channel, respectively. Postfixes h, s, and l stand for hue, saturation, and lightness. The postfixes of the SIFT-based matches describe the applied threshold on the SIFT-feature similarity score. The postfix spatial stands for an additional check for a similar image coordinate. See Appendix C for details.

Type	Value-based		Histogram-based
Feature	*Aggregation*	*Count*	*Correlation*
Variations	agg_bgr/b/g/r	count_bgr/b/g/r	corr_b/g/r
	agg_h/s/l	count_h/s/l	corr_h/s/l
	agg_gray	count_gray	corr_gray

Type	Edge-based	Signal-based	
Feature	*Change Fraction*	*PSNR*	*MSSIM*
Variations	change_fraction	psnr	mssim_b/g/r

Type	Optical-flow-based		SIFT-based
Feature	*Angle*	*Magnitude*	*Match*
Variations	angle_mean/std	mag_max/mean/std	match/_0/4/16/64/256/512
			match_spatial
			match_dist_min/max/mean/std

Type	Text-based	
Feature	*Bag of Words*	*n-Grams*
Variations	diff_words_count	n_grams_match_count/match_ratio/jaccard
	unique_term_count	n_grams_min_count/max_count
		n_grams_vocabulary_size

test data. The baseline can be intuitively described as a threshold of the number of pixels that are different between two images.

5.3 Evaluating Visual Change Classifiers

After introducing our evaluation setup, we explore two usage scenarios for the visual change classifiers. In the first usage scenario, we perform training and testing with sessions on the same Web site, representing highly similar interfaces. This is a scenario in which a usability expert would annotate the video recording of one user on a specific Web site and then apply the trained classifier to separate the interaction recordings of all other users. In the second usage scenario, we perform training and testing with sessions across Web sites, representing diverse interfaces with different layouts and contents. This is a scenario in which a classifier would be trained independently from the task at hand of the usability expert. This usage scenario accounts for visual changes that are general to many Web sites. We have used Python 3 with scikit-learn [79] for both usage scenarios.

Evaluation Setup. Evaluating visual change discovery, we borrow methodology from shot detection. Typical evaluations of shot detection methods are tested on 6 [24, 106] to 10 [95] movies. We have evaluated the technical accuracy on our recordings from 12 real-world Web sites visited by 4 users, resulting in 48 user sessions. Our technical evaluation relies on the manual annotation

of visual frames and visual stimuli by one of the authors. However, the annotations process has been verified by a cross-validation with 2 students who labeled 12 out of 48 user sessions for visual change.

The targets of the classifiers are the two classes of "visual change" and its absence, called "visual same." We report the F_1 scores per class. Additionally, we report the measures of coverage and overflow for shot detection [96]. The measure of *coverage* indicates how much a stimulus shot from the ground truth is represented by the most overlapping computed stimulus shot. For example, see Figure 9 (a). The ground-truth stimulus shot $\tilde{s}_n$ has most overlap with the computed stimulus shot s_2, which represents 3 out of 7 frames of $\tilde{s}_n$. Thus, the coverage of $\tilde{s}_n$ is $3/7 \approx 0.43$. A value close to one is better, a value close to zero is worse. The measure of *overflow* indicates how much the neighboring stimulus shots of each ground-truth stimulus shot are overlapped with computed stimulus shots intersecting with the ground-truth stimulus shot.

For example, see Figure 9 (b). The ground-truth stimulus shot $\tilde{s}_n$ neighbors with $\tilde{s}_{n-1}$ and $\tilde{s}_{n+1}$. Moreover, $\tilde{s}_n$ is represented by the computed stimulus shots s_2 and s_3. The overflow computes how much the computed stimulus shots representing $\tilde{s}_n$ overlap with the the neighbors of $\tilde{s}_n$. Out of 3 frames in $\tilde{s}_{n-1}$, 1 frame overlaps with s_2. Out of 3 frames in $\tilde{s}_{n+1}$, 3 frames overlap with s_3. Thus, the overflow of $\tilde{s}_n$ is $(1+3)/(3+3) = 0.\overline{6}$. A value close to zero is better, a value close to one is worse. We calculate coverage and overflow by first taking the labels of visual change as boundaries of the ground-truth stimulus shots. Each classifier is then applied to compute visual changes, which are then taken as boundaries of computed stimulus shots.

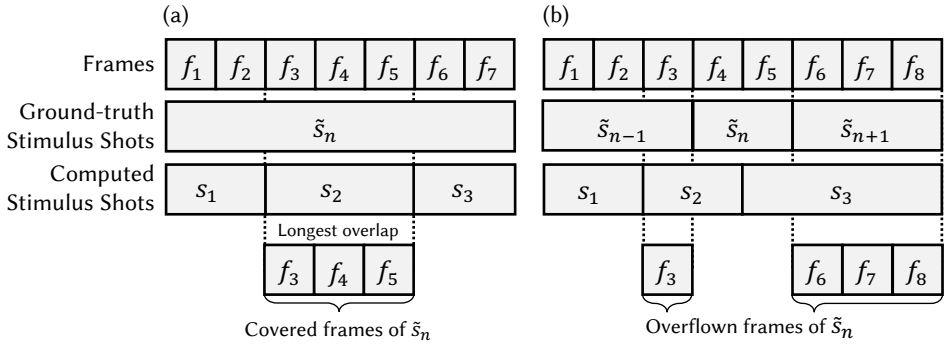


Fig. 9. Intuition of (a) coverage and (b) overflow measures. Inspired by Figure 1 of Vendrig and Worring [96].

Usage Scenario 1: Within Site. In this setting, we have performed evaluations when working on a single Web site, where the scenario is that the annotation of one user session serves as training data the other three user as test data. This corresponds to 25% of the data from one Web site for training and 75% of the data from the same Web site for testing. Rotating the one user session that serves as training material, we thus apply a four-fold cross-validation. We think this scenario represents a realistic usage of WebVSD by one usability expert. For a more conventional data split using more data for training than for testing, see the second usage scenario. Initially, we have performed an importance analysis of the features with scikit-learn, which uses a forest of decision trees to determine the relative importance of each feature in the decisions.[19] The results are listed in Table 3.

[19]https://scikit-learn.org/stable/auto_examples/ensemble/plot_forest_importances.html

Table 3. Feature importances when considering one user session as training data and three user sessions as test data.

Rank	Feature	Importance	Rank	Feature	Importance
1.	change_fraction (edge)	9.5% ± 5.2%	9.	psnr (signal)	3.5% ± 3.4%
2.	mssim_r (signal)	5.5% ± 4.2%	10.	agg_g (value)	3.4% ± 2.1%
3.	mssim_g (signal)	5.2% ± 3.5%	11.	agg_bgr (value)	3.2% ± 2.2%
4.	match_256 (SIFT)	4.3% ± 3.8%	12.	agg_r (value)	2.8% ± 1.7%
5.	match_spatial (SIFT)	4.2% ± 3.6%	13.	agg_gray (value)	2.7% ± 1.5%
6.	mssim_b (signal)	4.2% ± 3.1%	14.	agg_h (value)	2.5% ± 2.4%
7.	count_b (value)	4.1% ± 3.2%	15.	match_64 (SIFT)	2.3% ± 1.5%
8.	count_l (value)	3.5% ± 1.7%	16.	mag_max (optical flow)	2.2% ± 2.5%

We observe that the most important features are edge-based, signal-based, SIFT-based, and value-based. The features from optical flow, histograms, and text recognition are not important. Considering only the important features, we have performed a four-fold cross-validation of the bespoken classifiers. The results are listed in Table 4. The random forest classifier outperforms the SVC and the logistic regression, why their results are omitted from the table.

The F_1 scores for visual same and visual change are overall better for the random forest classifier than for the baseline classifier. The F_1 score for detecting visual same is for all Web sites above 90 percent, even reaching over 95 percent in eight out of twelve Web sites. The F_1 scores for the visual change are mixed. For sites like NIH, MayoClinic, and Nissan the F_1 score is above 90 percent. For other sites like CNN (60%), Reddit (74%), Guardian (79%), and WebMD (73%) the F_1 score is below 80 percent. The classification of visual change seems to be worse on news-related sites in general. In detail, on the recall is better on CNN, Reddit and Guardian (81%, 82%, 87%) than the precision (51%, 69%, 74%). We think a higher recall is more favourable than a higher precision, because a false-positive classification does only lead to more stimulus shots which may lead to a higher count of visual stimuli. In contrast, a false-negative leads to merging visually different frames, which would mean an incorrect synchronization of interaction data with the visual stimulus. The WebMD site from the health category has an low F_1 score for visual change in comparison to the other sites from the health category. In detail, the precision is 75% and the recall is 71%. After manually checking which visual changes are not recognized, we found that visual changes between the bright background and white menu do not cause significant changes regarding most features.

The baseline classifier often performs best in the overflow measure. However, one must look at both coverage and overflow in combination. For example, for the NIH site, the baseline classifier produces many false-positive shot boundaries. Thus, the overflow is zero (which is very good) but so is the coverage (which is bad). The video would be divided into far too many stimulus shots, resulting in an over-segmentation.

Usage Scenario 2: Across Sites. This scenario mimics a setting where a usability expert does not train the tool itself, but rather relies on the tool being pretrained on other domains. Per category, we use the user sessions from the three other categories as training data and the user sessions from the category itself as testing data. This corresponds to 75% of the data for training and 25% of the data for testing. Analogously to the first scenario, we performed an importance analysis of the features with scikit-learn. The results are listed in Table 5.

The results are similar to those from the first usage scenario, with a slightly different weighting. The edge-based feature is even more important, while again the signal, value, and SIFT-based features contribute to most classifications. Similar to the first usage scenario, the features from optical flow, histograms, and text recognition are not important. Considering only the important

Table 4. Classifiers of visual change when considering only value-based, edge-based, signal-based, and SIFT-based features. We use a four-fold cross validation. One user session acts as training data, and three user sessions act as test data. We report F_1 scores and the shot-detection measures of coverage and overflow for the interface body. *R. Forest* is a random forest classifier. Best classifier result per Web site is printed in bold font. ↑ denotes that a higher value is better. ↓ denotes that a lower value is better.

Shopping Sites	Walmart		Amazon		Steam	
Classifier	*R. Forest*	*Baseline*	*R. Forest*	*Baseline*	*R. Forest*	*Baseline*
↑ Visual Same [F_1]	**94% ± 02%**	88% ± 02%	**95% ± 01%**	91% ± 01%	**92% ± 00%**	72% ± 03%
↑ Visual Change [F_1]	**88% ± 03%**	79% ± 02%	**85% ± 03%**	76% ± 01%	**81% ± 02%**	58% ± 01%
↑ Agg. Coverage	**0.96 ± 0.04**	0.81 ± 0.02	**0.96 ± 0.02**	0.87 ± 0.04	**0.91 ± 0.03**	0.82 ± 0.03
↓ Agg. Overflow	0.17 ± 0.07	**0.04 ± 0.03**	0.12 ± 0.04	**0.11 ± 0.03**	**0.08 ± 0.06**	0.11 ± 0.05
News Sites	**Reddit**		**CNN**		**Guardian**	
Classifier	*R. Forest*	*Baseline*	*R. Forest*	*Baseline*	*R. Forest*	*Baseline*
↑ Visual Same [F_1]	**96% ± 01%**	84% ± 03%	**93% ± 03%**	76% ± 07%	**97% ± 00%**	80% ± 02%
↑ Visual Change [F_1]	**74% ± 04%**	45% ± 05%	**60% ± 09%**	33% ± 01%	**79% ± 03%**	44% ± 03%
↑ Agg. Coverage	**0.96 ± 0.03**	0.75 ± 0.02	**0.83 ± 0.09**	0.53 ± 0.05	**0.87 ± 0.02**	0.51 ± 0.02
↓ Agg. Overflow	0.10 ± 0.05	**0.00 ± 0.00**	0.30 ± 0.20	**0.08 ± 0.13**	0.07 ± 0.05	**0.00 ± 0.00**
Health Sites	**NIH**		**WebMD**		**MayoClinic**	
Classifier	*R. Forest*	*Baseline*	*R. Forest*	*Baseline*	*R. Forest*	*Baseline*
↑ Visual Same [F_1]	**97% ± 01%**	87% ± 02%	**93% ± 01%**	86% ± 02%	**98% ± 00%**	94% ± 01%
↑ Visual Change [F_1]	**92% ± 03%**	72% ± 01%	**73% ± 02%**	63% ± 04%	**90% ± 01%**	79% ± 01%
↑ Agg. Coverage	**0.97 ± 0.02**	0.68 ± 0.09	**0.90 ± 0.03**	0.82 ± 0.04	**0.96 ± 0.02**	0.77 ± 0.02
↓ Agg. Overflow	0.11 ± 0.05	**0.00 ± 0.00**	0.13 ± 0.05	**0.07 ± 0.02**	0.06 ± 0.01	**0.05 ± 0.03**
Car Sites	**General Motors**		**Nissan**		**Kia**	
Classifier	*R. Forest*	*Baseline*	*R. Forest*	*Baseline*	*R. Forest*	*Baseline*
↑ Visual Same [F_1]	**97% ± 00%**	83% ± 03%	**97% ± 00%**	93% ± 01%	**97% ± 00%**	93% ± 01%
↑ Visual Change [F_1]	**82% ± 01%**	40% ± 03%	**90% ± 01%**	79% ± 02%	**84% ± 03%**	48% ± 01%
↑ Agg. Coverage	**0.91 ± 0.03**	0.69 ± 0.10	**0.96 ± 0.03**	0.79 ± 0.04	**0.90 ± 0.05**	0.80 ± 0.03
↓ Agg. Overflow	0.24 ± 0.07	**0.21 ± 0.19**	0.13 ± 0.06	**0.10 ± 0.10**	0.14 ± 0.10	**0.04 ± 0.07**

Table 5. Feature importances when considering the user sessions of three categories as training data and the user sessions of one category as test data.

Rank	Feature	Importance	Rank	Feature	Importance
1.	`change_fraction` (edge)	13.8% ± 1.7%	9.	`count_b` (value)	2.7% ± 0.5%
2.	`psnr` (signal)	9.6% ± 1.2%	10.	`agg_h` (value)	2.5% ± 0.2%
3.	`agg_b` (value)	5.1% ± 0.6%	11.	`match_256` (SIFT)	2.4% ± 0.3%
4.	`agg_g` (value)	5.0% ± 0.6%	12.	`agg_s` (value)	2.4% ± 0.6%
5.	`agg_gray` (value)	4.9% ± 0.4%	13.	`count_bgr` (value)	2.2% ± 0.4%
6.	`agg_l` (value)	4.3% ± 0.6%	14.	`count_s` (value)	2.1% ± 0.6%
7.	`agg_r` (value)	3.6% ± 0.3%	15.	`count_g` (value)	2.1% ± 0.4%
8.	`agg_bgr` (value)	3.3% ± 0.5%	16.	`count_g` (value)	2.0% ± 0.5%

features, we have performed a four-fold cross-validation of the bespoken classifiers. The results are listed in Table 6. The random forest classifier outperforms the logistic regression, therefore results from the latter are omitted from the table.

Table 6. Classifier of visual change with user sessions across categories when considering only value-based, edge-based, signal-based, and SIFT-based features. The user sessions of three categories act as training data, and the user sessions of one category acts as test data. *SVC* is a support vector classifier, *R. F.* is a random forest classifier. *B.* is the baseline classifier. Best classifier result per category is printed in bold font. ↑ denotes that a higher value is better. ↓ denotes that a lower value is better.

Category	Shopping			News			Health			Cars		
Classifier	*SVC*	*R. F.*	*B.*	*SVC*	*R. F.*	*B.*	*SVC*	*R. F.*	*B.*	*SVC*	*R. F.*	*B.*
↑ Visual Same [F_1]	**93%**	**93%**	76%	92%	**94%**	82%	**96%**	**96%**	87%	**97%**	96%	88%
↑ Visual Change [F_1]	83%	**84%**	65%	54%	**64%**	39%	84%	**87%**	63%	**85%**	82%	54%
↑ Agg. Coverage	**0.95**	0.93	0.74	**0.93**	0.91	0.62	0.95	**0.96**	0.78	**0.93**	0.90	0.68
↓ Agg. Overflow	0.11	0.04	**0.02**	0.27	0.20	**0.05**	0.10	0.09	**0.06**	0.17	0.14	**0.06**

The F_1 scores for visual same are similar to the first usage scenario. For both SVC and the random forest classifiers, the F_1 score is over 90 percent. However, for visual change, the classification is partly worse than in the first usage scenario. While the F_1 score of visual change on Reddit and Guardian are 74 percent and 79 percent respectively, the corresponding F_1 score of news is for the second usage scenario at best 64 percent.

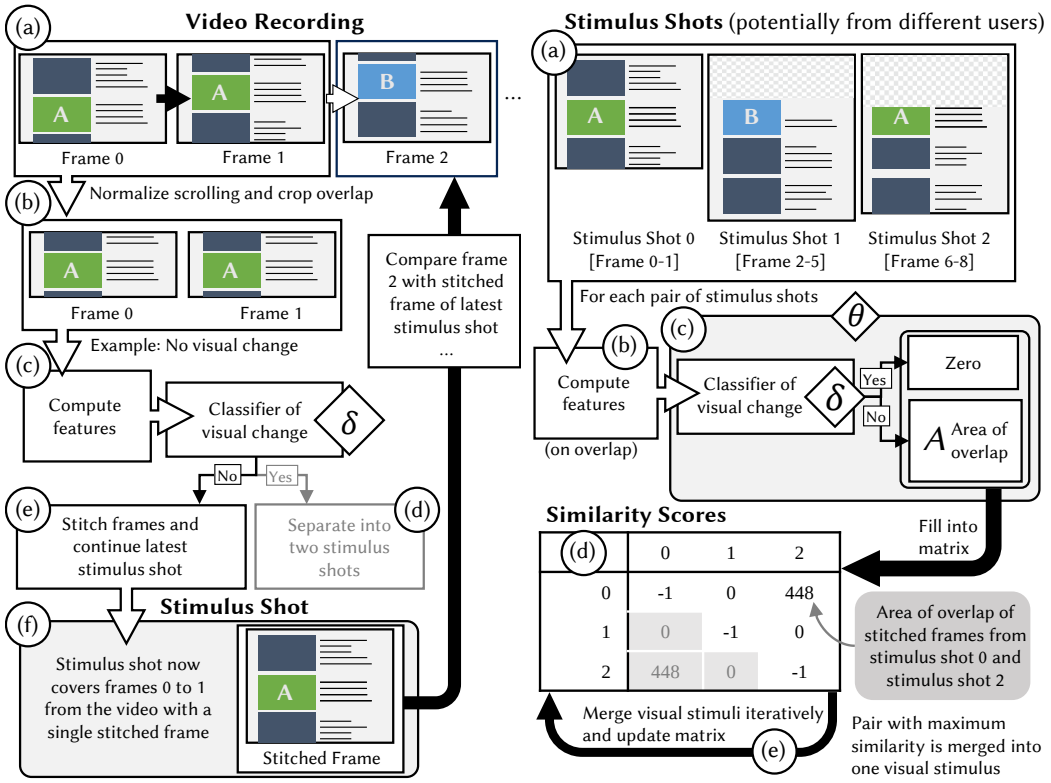
Takeaway about the Detection of Visual Change. The important features are edge-based, signal-based, SIFT-based, and value-based across both usage scenarios. We speculate that there may have been insufficient movement on the interfaces to trigger optical flow. Histograms appear to be less useful for our purpose, as there are usually no global changes in color distribution caused by dynamics in interfaces. The text-based features have been promising, yet it might be that dynamics do not introduce enough new text to be detected in comparison to the already existing text on an interface. The optical character recognition seems also to be very sensitive to fading animations. This takeaway addresses RQ2 in regard to which computer vision features are best suited to decide about visual change.

The baseline classifier can be consistently outperformed by SVC and random forest classifiers. SVC and the random forest classifiers are both viable choices for visual change classification, especially in the second usage scenario. This takeaway addresses RQ2 in regard to which classifiers are best suited to decide about visual change. Nevertheless, the performance of visual change classification varies from interface to interface, as visible in the results from the first usage scenario. In the following application and evaluation, we therefore employ the first usage scenario. This allows for deep insights into what better or worse performance in visual change classification means to the visual stimuli discovery.

6 WEBVSD: DISCOVERING VISUAL STIMULI ON WEB SITES

We define how to discover visual stimuli on Web sites given the visual change classifier from the previous section as our method implementation WebVSD, which corresponds to **Phase 4: Application**. Similar to the annotation, pixels from the interface body and the fixed elements are treated separately in the visual stimuli discovery. This is motivated by our previous work [69], in which we demonstrated how fixed elements can be overlaid on stitched screenshots from the interface body for a more efficient usability analysis.

First, we split the video recordings into stimulus shots. See Figure 10a for an illustration of the splitting procedure in visual stimuli discovery. (a) Initially, the first frame is considered the current stimulus shot. Then, we take the next frame, (b) normalize scrolling, and (c) compute the features



(a) Splitting a video recording into stimulus shots. (b) Merging stimulus shots into visual stimuli.

Fig. 10. Application phase of the visual stimuli discovery by example. Initially, all video recordings are split into stimulus shots. Then, the stimulus shots are merged across user sessions toward visual stimuli.

on the overlap of both frames. If the visual change classifier detects a based on the features, (d) the current stimulus shot is closed and a new stimulus shot is started with the currently processed frame. If the visual change classifier does detect no change, (e) the currently processed frame is put into the current stimulus shot. (f) The frames of the stimulus shot are stitched and considered for the next frame in the detection of visual change. After the splitting, each video recording of user sessions is partitioned into a series of stimulus shots.

We have decided to not stitch the frames at the very end of the splitting procedure. Instead, we stitch the frames on the fly during the splitting procedure. This allows us to compare each frame from the video recording not only with the previous frame but with the so-far stitched frame of the current stimulus shot. The stitched frame can contain areas from frames before the previous frame, such that the visual change classification has more information to decide whether the current frame belongs to the current stimulus shot or not. Therefore, this approach further improves the outcome of the splitting procedure.

Second, we perform agglomerative clustering with single linkage on the set of all stimulus shots. Analogously to the splitting process, the stitched images of stimulus shots that are found to be similar during clustering are stitched together and treated as a single stimulus shot in the further course of clustering. In the end, visually coherent stimulus shots have been merged into

Table 7. Visual change classifier that has been employed to execute the framework of visual stimuli discovery. The classifier has been trained with session of the first user on each Web site, according to the first usage scenario. ↑ denotes that a higher value is better. ↓ denotes that a lower value is better.

Web Site	Walmart	Amazon	Steam	Reddit	CNN	Guardian
↑ Vis. Same [F_1]*	92%	93%	93%	97%	89%	97%
↑ Vis. Change [F_1]*	85%	82%	80%	78%	50%	81%
↑ Agg. Coverage*	0.91	0.95	0.97	0.95	0.78	0.88
↓ Agg. Overflow*	0.06	0.07	0.13	0.06	0.07	0.11

Web Site	NIH	WebMD	MayoClinic	GM	Nissan	Kia
↑ Vis. Same [F_1]*	95%	94%	98%	97%	98%	98%
↑ Vis. Change [F_1]*	88%	78%	91%	84%	93%	87%
↑ Agg. Coverage*	0.95	0.94	0.99	0.95	0.98	0.95
↓ Agg. Overflow*	0.08	0.14	0.05	0.11	0.15	0.15

visual stimuli. See Figure 10b for a graphical illustration of the merging procedure in visual stimuli discovery. (a) We consider each pair of stimulus shots and (b) compute features on the overlap of their stitched frames. The features are fed into the classifier of visual change. (c) If a visual change is detected between the stitched frames of a pair of stimulus shots, a score of zero is assigned as their similarity score. If there is no visual change, the pixel area of overlap of the stitched frames is assigned as their similarity score. (d) All similarity scores are entered into a matrix that stores pairwise similarities between entries. Initially, all entries belong to pairs of stimulus shots. When a pair of stimulus shots are merged, the merged stimulus is considered instead, and the stitched frame of the merged stimulus shots is used for the computation of further similarity scores. (e) We iteratively take the entry with the highest score and merge the corresponding pair of stimulus shots. The procedure does stop at a predefined threshold of similarity that no entry in the matrix exceeds. The remaining stimulus shots, regardless of whether they have been merged with other stimulus shots or not, are considered as visual stimuli, represented through their stitched frames.

Many Web sites contain animations of dynamic elements like in- and out-fading menus or moving photos in photo carousels. The duration of the animations is usually below one second and triggers a visual change classifier more than once during the animation. We argue that any stimulus shot that lasts below one second should not contribute to the discovery process, as those stimulus shots would produce over-segmented results. Therefore, we merge stimulus shots below one second to the closest stimulus shot in time that is at least one second long without stitching the pixels of the shorter stimulus shot.

We have implemented WebVSD[3] in C++, using OpenCV [12] and the Shogun machine learning library [85]. We have applied WebVSD to our data set. Specifically, we have chosen the classifiers from the first usage scenario trained with the video recording and labels of the first user. See Table 7 for details about the performance of the classifiers. Applying WebVSD to our data set, we discover 2,742 visual stimuli for the twelve Web sites across the four user sessions each. See Appendix D for an example of the visual stimuli discovered by WebVSD. Examples about edge-cases in visual change classification are depicted in Appendix E, which is part of the quality evaluation of WebVSD.

7 ASSESSING THE QUALITY OF VISUAL STIMULI

The goal of the visual stimuli discovery is to reduce the information overload in usability analyses while accurately representing the recorded user sessions. This leads us to the following two research questions:

> **Research Question 3 (RQ3)**: How much can the visual stimuli discovery reduce the information overload in comparison to video recordings?
>
> **Research Question 4 (RQ4)**: How accurately do the visual stimuli represent the screen contents for a subsequent usability analysis?

We compare the number of pixels in the visual stimuli to the number of pixels in the video recordings to measure information overload, addressing the RQ3. This measure provides a reasonable indication about how much data a usability expert has to scan when analyzing the user sessions. The measure can be computed from the data alone. In contrast, the accuracy in representing the user sessions in the visual stimuli cannot be computed from the data alone, as the decision process of visual change must be taken into account. Therefore, we need to annotate for each discovered visual stimulus whether it represents the video recording correctly, addressing RQ4. This assessment verifies the outcome of **Phase 4: Application**.

7.1 Annotating the Discovered Visual Stimuli

We annotate for every frame of the video recordings whether it is correctly represented by one of the discovered visual stimuli according to our definition of visual change in Figure 6 on page 19.

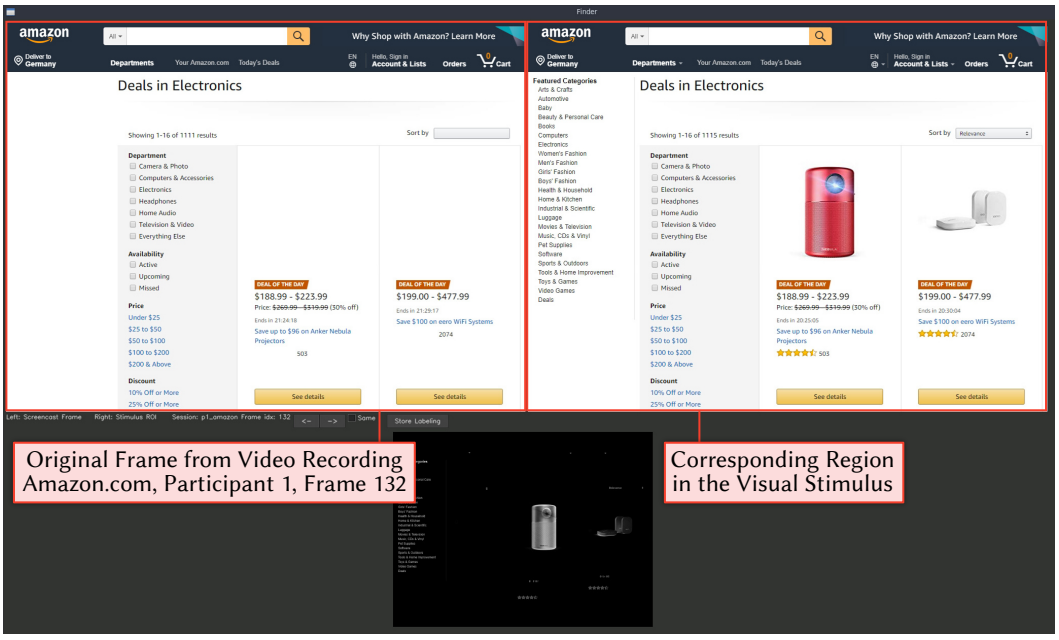


Fig. 11. Screenshot of our tool to check for a frame whether it is correctly represented in a discovered visual stimulus. In addition to a side-by-side comparison on the top, we also display a difference-image on the bottom. In the example given, the product pictures are missing from the frame, whereas the product pictures are visible in the visual stimulus. Therefore, we have labeled the representation of the frame as incorrect.

Table 8. Quality of the visual stimuli from the interface body. Some frames are entirely covered by fixed elements and do not account for this assessment. The count of annotated frames is therefore stated as *# Annotated Frames*. The number of discovered visual stimuli on these frames is denoted as *# Visual Stimuli*. The *Pixel Ratio* is computed by considering dividing the number of pixels from the visual stimuli by the number of pixels from the frames that display the interface body. The number of ratio of *Correct Frames* frames shows how many frames have been labeled as correctly represented by their visual stimulus. ↑ denotes that a higher value is better. ↓ denotes that a lower value is better.

Web Site	Walmart	Amazon	Steam	Reddit	CNN	Guardian
# Annotated Frames	3,511	4,553	5,466	2,078	3,433	3,725
# Visual Stimuli	135	189	186	37	60	32
↓ Pixel Ratio	5.79%	6.55%	4.84%	5.06%	3.72%	2.91%
↑ Correct Frames	96.89%	95.46%	81.77%	97.23%	99.13%	66.31%

Web Site	NIH	WebMD	MayoClinic	GM	Nissan	Kia
# Annotated Frames	2,623	3,736	2,904	2,688	4,507	2,152
# Visual Stimuli	30	100	38	39	82	22
↓ Pixel Ratio	1.96%	4.14%	2.32%	3.09%	3.67%	2.16%
↑ Correct Frames	99.88%	85.01%	96.5%	95.97%	88.79%	99.1%

We have created a dedicated tool to perform the annotation in WebVSD.[20] The tool has a two-column interface, where the left column displays a frame from the video recording and the right column displays the visual stimulus by which the frame is represented. See Figure 11 for a screenshot of the tool. If the pixels between the frame and the visual stimulus match perfectly, the frame is automatically labeled as represented correctly. See Appendix E for examples from the annotation. We perform the annotation and assessment separately for the interface body and the fixed elements in the following.

7.2 Assessing the Quality of Visual Stimuli from the Interface Body

We assess the quality of visual stimuli by estimating their potential reduction in visual overload and the accuracy in representing the screen contents from the user sessions. The results of the quality assessment are listed in Table 8. The ratio between the number of pixels in the visual stimuli and the number of pixels in the video recordings is on macro average 3.85% ± 1.47%, indicating that the visual stimuli contain tremendously fewer pixels than the video recordings, addressing RQ3. We argue that this indicates a reduced information overload in usability analysis. In total, 41,376 frames have been annotated with regard to the interface body. The first author of this work has labeled 35,886 frames manually. 5,490 frames have been matched perfectly with their visual stimuli and were automatically labeled as correctly represented. The annotation results state that on macro average 92% ± 10% frames are correctly represented by their visual stimuli, addressing RQ4. However, the results differ between the Web sites. For eight of the sites, the scores are well above ninety percent (Walmart with 96.89%, Amazon with 95.46%, Reddit with 97.23%, CNN with 99.13%, NIH with 99.88%, MayoClinic with 96.5%, GM with 95.97%, and Kia with 99.1%). However, for Steam with 81.77%, Guardian with 66.31%, WebMD with 85.01%, and Nissan with 88.79% the number of correct frames are below ninety percent. On the Steam site, we found that the order of game tiles and the order of screenshots of games have been randomized across user sessions.

[20] https://github.com/raphaelmenges/visual-stimuli-discovery/tree/master/code#finder

Thus, many frames were wrongly merged into visual stimuli with a similar-looking layout while containing similar pictures at different positions of the layout. See Figure 32 on page 63 for an example. The Guardian site changed articles and respective photos on the front page during the six hours of data set recording, yet, the frames were still merged into visual stimuli. See Figure 29 on page 61 for an example. The WebMD site displayed personalized advertisements in each user session. See Figure 33 on page 63 for an example. Furthermore, some elements on the WebMD site were slightly moving up and down during loading the page, which impacted the entire layout of the page over time. The Nissan site displayed photo carousels with only a slightly different look, which the visual change classifier struggles to distinguish. See Figure 31 on page 62 for an example. There is some noise in the results across all sites introduced by animated advertisement banners and users who highlighted text spans with the cursor.

7.3 Assessing the Quality of Visual Stimuli from Fixed Elements

We found that the visual stimuli discovery created far too many visual stimuli on fixed elements. See Table 9 for the count of visual stimuli discovered on the interface body against the count of visual stimuli discovered on fixed elements. WebVSD has discovered more visual stimuli on fixed elements than on the interface body on eight Web sites, while fixed elements are only 15.36% of the pixels in the video recordings. The visual stimuli discovery clearly fails for fixed elements, why we did not take the futile effort to annotate the quality of visual stimuli from fixed elements. Instead, we provide two examples of the visual stimuli discovery fails for fixed elements.

Figure 12 shows the top menu on the Walmart site, which is a fixed element. The top menu has a rectangular shape and does not change its shape over time. Therefore we expect a single visual stimulus to be discovered from it in our WebVSD. But the background of the top menu is transparent, such that the contents from the interface body are visible behind the top menu. As the top menu stays on a fixed position, the portion of the interface body that is visible through the top menu changes during a user session because of animations and scrolling by the user. This changes the appearance of the top menu dramatically over time and causes WebVSD to discover 105 visual stimuli. Figure 13 shows a chat avatar on the Kia site, which is a fixed element. A chat avatar has a non-rectangular shape and changes its vertical position in a hovering animation over time. The animation and the transparency make WebVSD discover 669 visual stimuli.

We discuss the limitation in discovering visual stimuli from fixed elements in Section 9 and suggest approaches for handling fixed elements properly in future works. Similar issues of discovering too many visual stimuli may arise on highly personalized Web sites in e-commerce that look different to every user. However, the aim of the visual stimuli discovery is to provide a basis of visually similar stimuli on which the interaction data of multiple users can be analyzed in combination. Therefore, highly personalized Web sites are not in the scope of the targeted interfaces and are thus not included in the data set.

8 EVALUATING THE USEFULNESS OF AUTOMATICALLY DISCOVERED VISUAL STIMULI

While Section 5 and Section 7 provide quantitative measurements of technical correctness, the question remains open whether the visual stimuli discovery is useful for usability experts. Therefore we take the point of view of a usability expert employing WebVSD in her workflow. We come up with two research questions in this regard:

> **Research Question 5 (RQ5)**: Can visual stimuli represent the interface in a manner that enables subsequent usability analysis for a usability expert?

Table 9. Count of visual stimuli per Web site from the interface body and from fixed elements.

Web Site	Walmart	Amazon	Steam	Reddit	CNN	Guardian
Total Count	349	266	186	151	275	123
Interface Body	135	189	186	37	60	32
Fixed Elements	214	77	0	114	215	91

Web Site	NIH	WebMD	MayoClinic	GM	Nissan	Kia
Total Count	171	164	38	143	185	691
Interface Body	30	100	38	39	82	22
Fixed Elements	141	64	0	104	103	669

Fig. 12. Visual stimuli discovered from the top menu on the Walmart site, which is a fixed element. We indicate the file path to the visual stimulus in the data set below each visual stimulus.

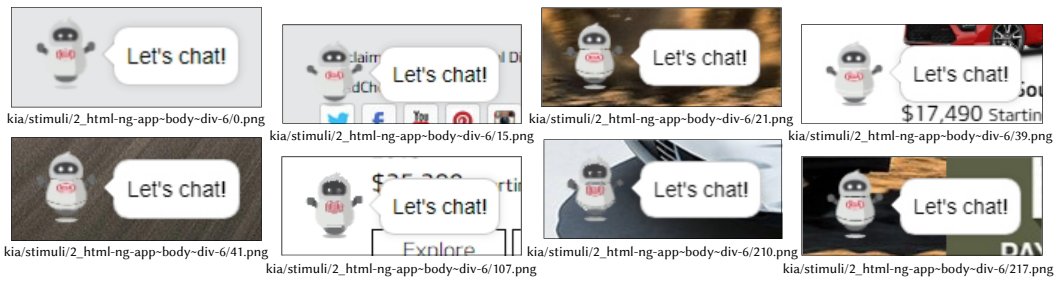


Fig. 13. Visual stimuli discovered from a chat robot on the Kia site, which is a fixed element. We indicate the file path to the visual stimulus in the data set below each visual stimulus.

> **Research Question 6 (RQ6)**: Can visual stimuli make the workflow of a usability expert more efficient?

Addressing both research questions RQ5 and RQ6, we analyze four case studies in detail and report the results of an interactive survey with experts. This evaluation corresponds to **Phase 5: Analysis**.

8.1 Case Studies about using Automatically Discovered Visual Stimuli

We aim to investigate how the common practice of AOI marking might benefit or might be jeopardized by tools that exploit visual stimulus discovery. It seems methodologically infeasible to come up with representative samples of Web sites that a usability expert has to study and representative samples of AOIs that a usability expert would define. Therefore, we cannot and do not want to claim that we can provide a comprehensive data set with complete gold standard annotations. However, we are interested in practical observations that we derive from the application of visual stimuli discover.

Moreover, usability experts perform the analysis of interfaces in professional software suites like Tobii Pro Studio [92], RealEye.io [86], or EYEVIDO Lab [38]. These software suites integrate the recording of user sessions on Web sites, the data processing, and the analysis into a single workflow. The tight integration of the workflow is necessary due to the amount and specificity of the collected data. There are yet no standard formats to store eye gaze data or interface representations. However, this proprietary handling of data renders it impractical to plug the method of visual stimuli discovery into existing software suites. Therefore, we perform the case studies outside a software suite and we have written custom tools for annotation.

We first explain the design of our case studies. As part of each case study, we mimic plausible behaviour of a usability expert by annotating video recordings with typical AOIs. Based on these annotations, we annotate video recordings and visual stimuli of the case studies about the occurrence of the respective AOI in terms of precision and recall.

Choice of the AOIs. We have decided to perform case studies for a variety of Web sites and AOIs from our data set that are chosen according to the following three heuristics H1 – H3, which ensure that corresponding AOIs cannot be found trivially:

H1: The AOI has occurred in every user session on the respective Web site.
H2: The AOI contains an interactive element that users have interacted with.
H3: The AOI covers contents that have a dynamic appearance or a complex visibility configuration.

We have chosen four AOIs, each analyzed in one case study. See Figure 14 for images of the AOIs. In the following we shortly introduce and motivate each AOI:

(a) A tile on CNN. The tile links to the review of the movie "Captain Marvel." The tile was on the second page of the CNN site the users have browsed to (H1). The tile has been gazed at and hovered with the mouse cursor by the users (H2). At page load, the tile changes its appearance over time as itself and its styling is loaded (H3).

(b) The top menu on Guardian. The top menu was available on all pages of the Guardian site (H1). Users individually expanded the top menu, hovered with the mouse cursor over the entries, and collapsed the top menu. In addition, they were advised to click on the "Sports" entry, according to the protocol as listed in Appendix A (H2). Hovering with the mouse cursor, enhancing, and collapsing dynamically also changed the appearance and visibility of the top menu (H3).

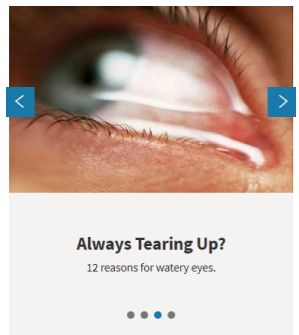
(c) The footer menu on Walmart. The footer menu was on the bottom of all pages of the Walmart site (H1). Users gazed through the entries of the footer menu and were asked to click on "Toys" under "In The Spotlight," according to the protocol as listed in Appendix A (H2). The footer menu itself is not dynamic but often overlaid by another menu that is aligned to the left side of the viewport. This makes the visibility configuration of the footer menu complicated (H3). See Figure 17 for a situation in which the footer menu is overlaid by the menu from the left.

(d) A carousel slide on WebMD. The carousel was on the second page of the WebMD site the users have browsed to and every user was confronted with that slide in the carousel (H1). Users could

(a) Tile on CNN

(b) Top Menu on Guardian

(c) Footer Menu on Walmart (black frame added for readability)



(d) Carousel Slide on WebMD

Fig. 14. AOIs that have been chosen for the case studies.

navigate through multiple slides in the carousel by clicking on the arrow icons (H2). Additionally, the slides changed automatically after a short period (H3).

It is to be noted that out of the twelve Web sites we have chosen sites for which the visual change classifier has diverse performance. For some Web sites like Walmart and CNN, the visual change classifier performs well. For other Web sites like Guardian and WebMD, the performance of the visual change classifier is worse in comparison. Thus, we want to find out how much these differences in the performance of the visual change classifier would affect the usefulness for a usability expert.

Annotating the Video Recordings for the AOIs. For each case study, we annotated in which frames of the video recordings the AOI is displayed as ground truth. The annotation employs the decision process of visual change from Figure 6 on page 19. If the AOI was only partly displayed because it was occluded by other elements or cropped by the viewport, it was still labeled as displayed because it would be still useful to be analyzed in usability analysis. We have developed a tool in WebVSD[21] for the annotation process. See Figure 15 for a screenshot taken during the annotation. The annotation was performed by the first author of this paper. He annotated in total 15, 938 frames

[21] https://github.com/raphaelmenges/visual-stimuli-discovery/tree/master/code#evaluator

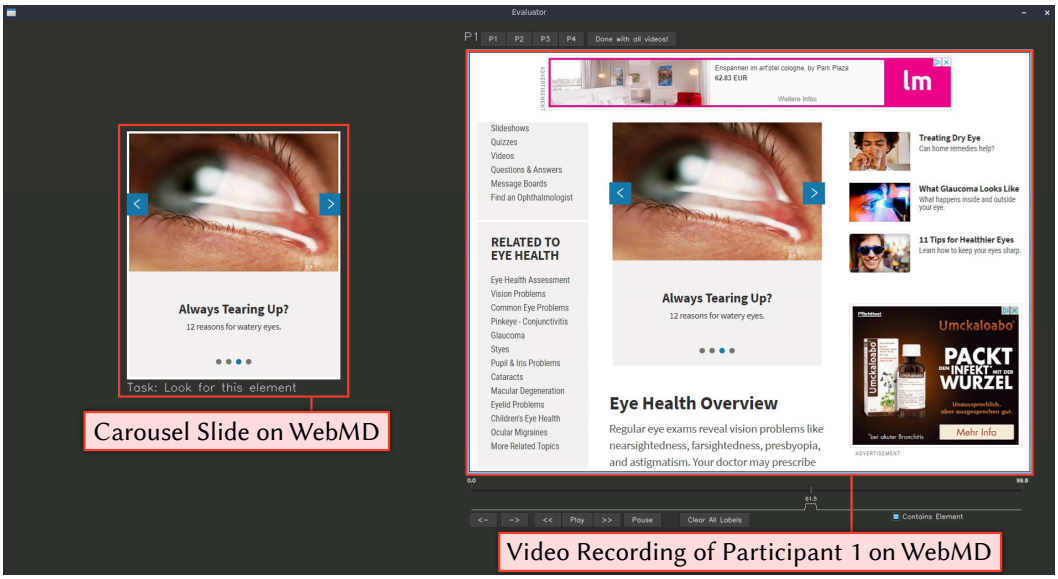


Fig. 15. We annotated each frame from the video recordings of a Web site whether it displays the AOI or not.

across the user sessions on four Web sites. He did also annotate the visual change in Section 5.1, where the decision process in general and his annotations specifically were cross-validated by students. Thus, we did not perform additional cross-validation for the annotation here. About 7.3% of the frames displayed one of the AOIs. Details about the annotation can be found in Table 10.

Annotating the Visual Stimuli for the AOIs. The annotation of the visual stimuli happened in two steps. First, we annotated for each case study in which visual stimuli the corresponding AOI is displayed. Similar to the annotation of the video recordings, AOIs which were partly visible were still labeled as displayed. The tool used for annotating the video recordings could be also used to annotate the visual stimuli.[8.1] See Figure 16 for a screenshot taken during the annotation. Furthermore, see Figure 17 for examples of visual stimuli from the annotation. Second, for each visual stimulus that displayed the AOI, we looked up the frames it was created from during the visual stimuli discovery. We annotated these frames according to they actually display the AOI. In comparison to the previous annotation of the video recordings, we have not only labeled whether the AOI is displayed or not. Instead, we decided to use three labels. "Correct Frames" are frames in the visual stimuli that correctly display the AOI of the case study. "Incorrect Frames" are frames in

Table 10. We have annotated each frame from the video recordings of a Web site whether it displays the AOI or does not display the AOI. If the AOI was partly occluded through other elements or cropped by the viewport, it has been still labeled as displayed in the frame.

Area of Interest	Tile on CNN	Top Menu on Guardian	Footer Menu on Walmart	Carousel Slide on WebMD
# Annotated Frames	3855	3823	4524	3736
# Frames with AOI	316	235	476	134
# Frames without AOI	3539	3588	4048	3602

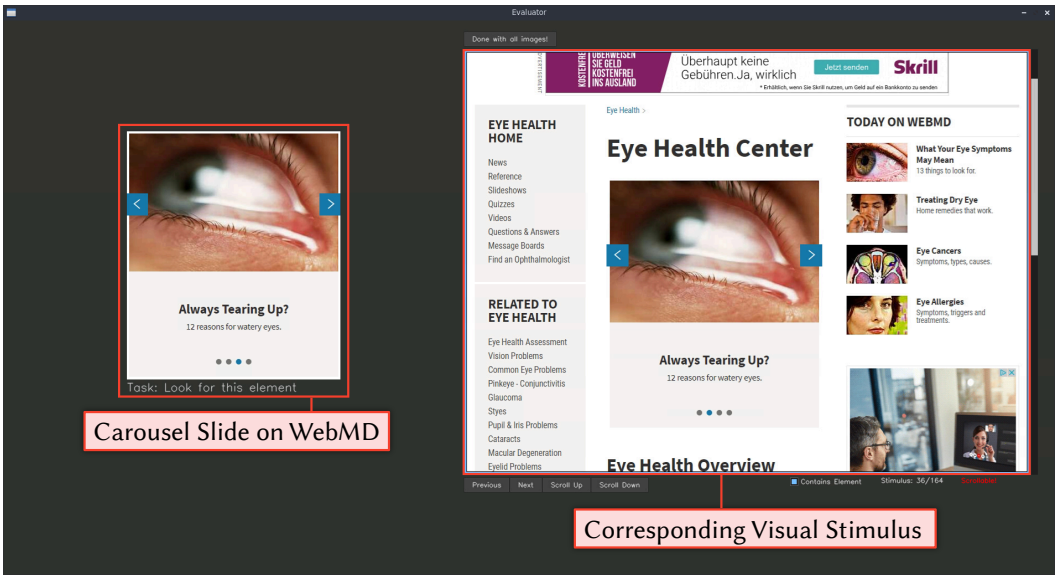


Fig. 16. We annotated each visual stimulus of a Web site whether it displays the AOI or not.

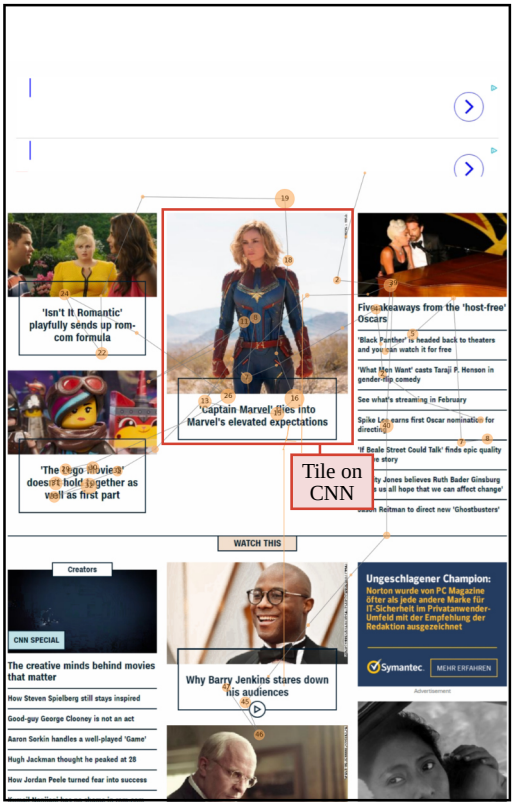
the visual stimuli that do not display the AOI of the case study. "Stitched Frames" are frames in the visual stimuli that are irrelevant in regard to the AOI, as they are portions of the Web page that are above or below the AOI and have been stitched accordingly. We added the third label of "Stitched Frames" to correctly compute the measures of precision and recall in the next paragraph. We have developed another tool for this annotation in WebVSD,[22] see Figure 18 for a screenshot. Analogously to the other annotations, we employed the decision process of visual change from Figure 6 on page 19. The annotations were performed by the first author of this paper. Details about the annotation can be found in Table 11.

Evaluating the Case Studies. Considering the annotation of video recordings and visual stimuli, we compute the precision and recall for each case study in correctly displaying the AOI. The precision is computed as the number of correct frames divided by the sum of correct frames and incorrect frames. The recall is computed as the number of correct frames divided by the number of frames from the video recordings that display the AOI. See Table 12. The table also shows the performance of the visual change classifiers as originally stated in Table 7 on page 28. This allows the reader a straightforward estimation of the influence of the performance of the visual change classifiers on the case studies. We discuss for each case study the results in the following.

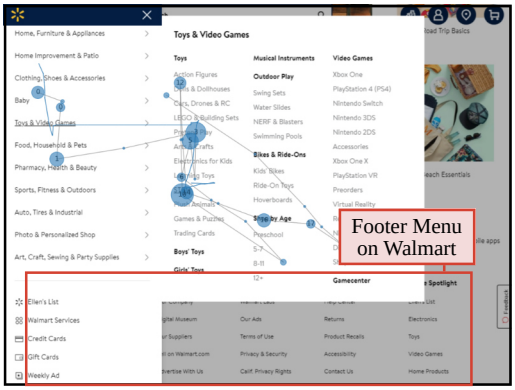
(a) A tile on CNN. The precision has a value of 93% and the recall is perfect. The deficit in the precision is caused by frames during loading of the page that do not contain the tile but are merged into visual stimuli that contain the tile. The corresponding visual change classifier shows low scores of visual change detection and coverage. Yet, the performance does not harm the results of the case study substantially.

(b) The top menu on Guardian. The precision has a value of 97% and the recall is perfect. A minor deficit in precision is because of a few frames in which the top menu has already been collapsed, but the frames are still merged into visual stimuli that display an extended menu. The very good results in the case study compared with the comparably worse performance of the corresponding

[22] https://github.com/raphaelmenges/visual-stimuli-discovery/tree/master/code#preciser



cnn/stimuli/0_html/plots/18-scanpath_mouse.png
(the AOI has been found on 9 more visual stimuli)



walmart/stimuli/2_html~body~div-0~div~div~div-0~div-0~div-header-spark-menu/plots/49-scanpath_mouse.png
(the AOI has been found on14 more visual stimuli)

guardian/stimuli/1_html/plots/
2-scanpath_mouse.png
(the AOI has been found on 3 more visual stimuli)

webmd/stimuli/1_html/plots/40-scanpath_mouse.png
(the AOI has been found on 4 more visual stimuli)

Fig. 17. One exemplary visual stimulus from each of the four case studies that display the respective AOI. We have marked the AOI on each visual stimulus with a red box. We plot eye gaze data as scanpath and mouse cursor movements as lines onto the visual stimulus for this illustration. The path to the displayed stimuli in the data set is given beneath the visual stimulus.

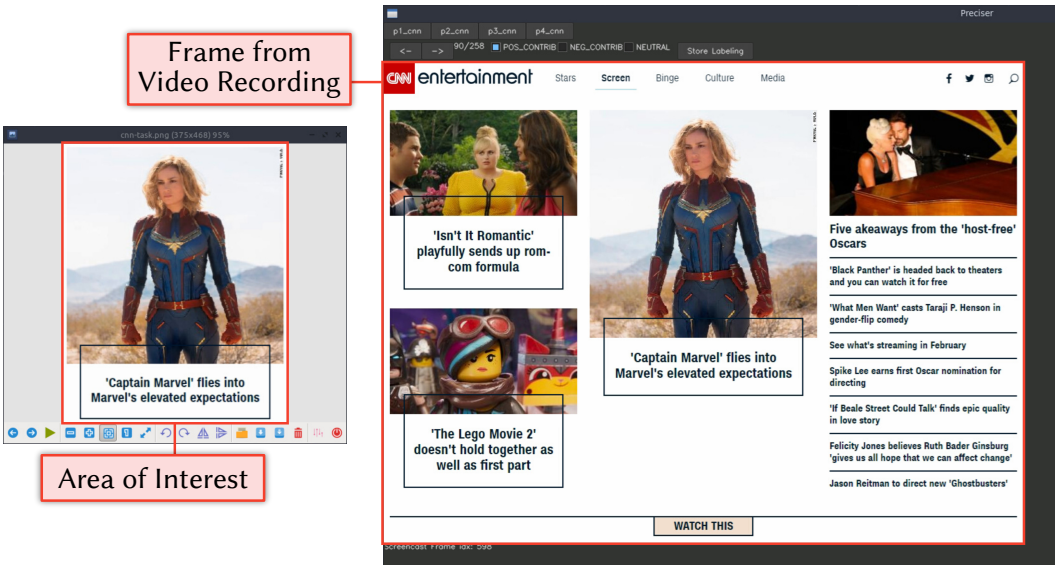


Fig. 18. Screenshot of our tool we used to annotate whether a visual stimulus correctly represents an AOI. On the left, we display the AOI in an image viewer. Here, it is the tile on CNN. On the right, our interface loads for each visual stimulus that is labeled to contain the AOI the frames from the video recordings it has been composed of. For each of these frames, we annotate whether the frame actually displays the AOI or does not display the AOI.

visual change classifier is due to the task at hand. The quality assessment in the previous section looks at the entire frames of the video recordings, whereas the case study only focuses on the top menu. On the Guardian site, the visual change classifier often does not detect a change due to a different photo behind the top menu. Thus, the visual stimuli discovery merges the top menu into

Table 11. We have annotated each visual stimuli and all individual frames they are composed from whether they display the AOI or not display the AOI. If the AOI was partly occluded through other elements or cropped by the viewport, the AOI has been still counted as displayed. It is to note that we also annotated visual stimuli from fixed elements. However, the only case study in which fixed elements contained the AOIs have been on Walmart, where the footer menu was sometimes covered by the fixed side menu (see Figure 17). Although the visual stimuli discovery in general fails, it worked well for the Walmart case study.

Area of Interest	Tile on CNN	Top Menu on Guardian	Footer Menu on Walmart	Carousel Slide on WebMD
# Annotated Visual Stimuli	275	123	349	164
# Visual Stimuli with AOI	10	4	15	5
# Visual Stimuli without AOI	265	119	334	159
# Annotated Frames	708	243	859	324
# Correct Frames	330	235	464	106
# Incorrect Frames	26	8	0	147
# Stitched Frames	352	0	395	71

fewer visual stimuli. This affects the quality evaluation but does not harm the working in the case study.

(c) The footer menu on Walmart. The precision is perfect and the recall has a value of 97%. These very good results correspond to the well-performing visual change classifier used in the visual stimuli discovery.

(d) The carousel slide on WebMD. The precision has a value of 42% and the recall has a value of 80%. We found that other carousel slides and expansions from the top menu have been mixed into the visual stimuli. This corresponds with the poor classification of visual change on the WebMD site.

The quality assessment of visual stimuli discovery in the previous section considers everything on a Web page. When interested in an AOI however, some errors will affect the AOI on that page, but other errors will only affect other AOIs on that page. Nevertheless, a better-performing visual change classifier also means a more precise analysis, as the case studies show. Especially when a well-performing visual change classifier can be provided, the workflow of a usability expert could be more efficient due to much less effort in marking AOIs.

We argue that the overall outcome shows that visual stimuli can represent the interface suitably for a usability expert, addressing RQ 5. Similar to the enhanced representation [69], the marking of AOIs would be faster on the visual stimuli than on the video recording, addressing RQ 6.

8.2 Preliminary Feedback on the Usefulness of Visual Stimuli Discovery

We complement the observations and experiences collected in the case studies with an online survey targeted at usability experts.[2] The purpose of the survey is to gather feedback by external usability experts in regard to our research questions RQ5 and RQ6. First, we describe the design of the survey. Then, we present and discuss the results of the survey.

Designing a Survey about Visual Stimuli Discovery. The survey consisted of eleven pages. On the first two pages, we asked for the demographics and expertise of the participant. We explained what we define as a dynamic interface on page three. Then, we had three AOIs from the case studies, namely the footer menu on the Walmart site, the carousel on the WebMD site and the top menu on the Guardian site. We decided to not include the AOI on the tile from CNN because, *(i)*, the tile does not change upon user interaction and, *(ii)*, we wanted to shorten the survey to reduce the number of drop-outs. For each scenario, we had two pages in the survey. First, we pointed to the AOI and

Table 12. Evaluation of the case studies based on the annotation from Table 10 and Table 11. *Precision* is calculated as *# Correct Frames* divided (*# Correct Frames* + *# Incorrect Frames*). *Recall* is calculated as *# Correct Frames* divided *# Frames with AOI*. The values below are repeated from Table 7 from page 28 to allow a comprehensive reading. They signify the performance of the visual change classifier that was used in the visual stimuli discovery. ↑ denotes that a higher value is better. ↓ denotes that a lower value is better.

Area of Interest	Tile on CNN	Top Menu on Guardian	Footer Menu on Walmart	Carousel Slide on WebMD
↑ Precision	93%	97%	100%	42%
↑ Recall	100%	100%	97%	80%
↑ Vis. Same [F_1]*	89%	97%	92%	94%
↑ Vis. Change [F_1]*	50%	81%	85%	78%
↑ Agg. Coverage*	0.78	0.88	0.91	0.94
↓ Agg. Overflow*	0.07	0.11	0.06	0.06

displayed the corresponding four user sessions of the data set as an embedded video recording. The participants were asked to skim through the video recordings and tell us how they would tackle the given analysis task for 20 users with their current workflow. Second, we had a gallery with the visual stimuli that displayed the AOI. It were the same visual stimuli that we annotated for the case studies and generated by visual stimuli discovery as reported in Section 6. The participants could zoom and pan the visual stimuli and display the eye gaze paths or mouse traces over the visual stimuli. We asked the participants how accurately they believe an analysis would be when using our visual stimuli and whether they could save time in comparison to their current workflow. After the three case studies, we asked for general feedback about proposed method on the tenth page and handled the submission of the survey on the last page. See Appendix F for details about how we have integrated video recordings and visual stimuli interactively into the survey by developing a custom survey framework with JavaScript.

Results of the Survey. We asked for participation via personal and E-mail-list invites; and public postings in respective LinkedIn groups. We received eight responses, submitted by five female and three male participants (age = 35.9 ± 8.0 years). This is a common sample size for a survey with domain experts in usability [55, 57]. The professions of the participants ranged from three researchers (1, 2, and 7 years of experience), over a product owner (5 years), a usability expert (5 years), a sales associate (1 year), a consultant (23 years), to the head of an eye-tracking laboratory (8 years). Four of the participants have conducted more than 10 usability studies, three have conducted 4 – 10 studies and one participant has conducted 1 – 3 studies. Conducting Web usability studies was the most common among participants, as seven votes were received for Web pages, five for mobile apps, four for desktop applications, and three for advertisements. All participants make use of eye gaze data for usability analysis, seven consider mouse clicks, and five touch input. Eye gaze is commonly analyzed with AOIs on the visual stimulus (among all participants), additionally, six participants also utilize heatmaps.

Given the three case studies, we have asked the participants how they would tackle the given analysis task for 20 users with their current workflow. We phrased the question in such way to make the experts consider the most scalable solutions for an analysis task. They have mentioned a variety of methods, i. e., four participants would use a video recording, two mention screenshots, four suggest a composed screenshot (like the enhanced representation), two would do a video recording with a camcorder, three would apply the "thinking-aloud" method, and one mentions a Live-DOM representation (recording of events and replay on the live Web site). Regarding the proposed method, we asked the participants for each scenario about the perceived accuracy of visual stimuli ("I could work with the discovered stimuli as accurate as with the method in my current workflow.") and whether using them could save them time in analysis ("I could save time when working on the discovered stimuli instead with the method in my current workflow."), each on a five-point Likert scale with ratings from from "Strongly Disagree" to "Strongly Agree". The results are shown in Figure 19. The accuracy is rated high for the footer menu on the Walmart site and the top menu on the Guardian site. However, the carousel slide on the WebMD site received a few negative votes. This might be caused by visual stimuli that show the fading process of the slides, yet are still considered to display the slide-of-interest by our definition of visual change from Figure 6 on page 19.

The general feedback about is shown in Figure 20 and demonstrates that experts would like to use the proposed method for usability analysis, addressing RQ 5. The acceptance of the semi-automatic mode emphasizes that our scenario of labeling one user session manually for visual change to train the visual change classifier appears to be realistic. We also asked the participants' opinions on the novelty of the proposed method. All participants believe the method is novel, half of them state it

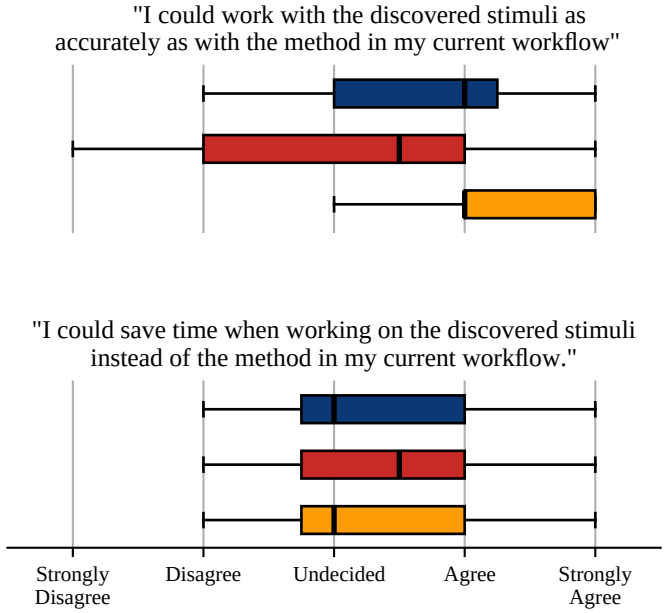


Fig. 19. Ratings of the accuracy and potential for time saving with proposed method per scenario. ■ Footer Menu on Walmart. ■ Carousel Slide on WebMD. ■ Top Menu on Guardian.

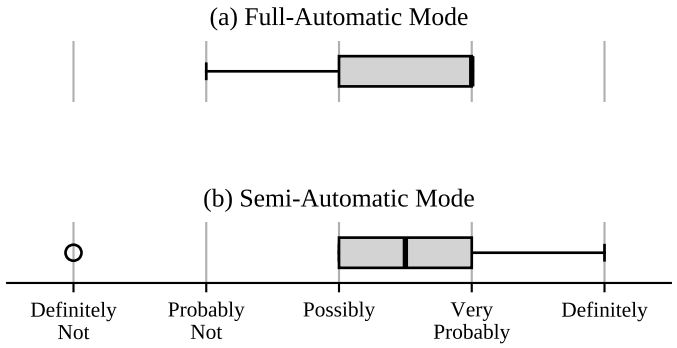


Fig. 20. Feedback whether usability experts would employ the visual stimuli discovery in their workflow. We distinguish between (a) a full-automatic mode or (b) a semi-automatic mode. For the latter, a usability expert would have to manually label one user session, to train the visual change classifier.

is very novel, and half of them categorize it as relatively novel compared to the existing methods they use.

The experts perceived the visual stimuli to be as accurate as video recordings and stated that the visual stimuli would help them in saving time and effort, addressing RQ 6. The experts were very welcoming about a method of visual stimuli discovery that can support their workflow. One participant left feedback that broadens the use-case beyond the aggregation of eye gaze and mouse data. According to their feedback, dynamics are “especially challenging when synchronizing with

additional measurements like GSR [Galvanic skin resistance], NIRS [Near-infrared spectroscopy], EEG [Electroencephalography] etc."

8.3 Threats to Validity of Preliminary Feedback

In the survey targeted at usability experts presented in the previous Section 8.2, we recognize several potential threats to validity that may affect the interpretation of our results. Below, we address these concerns in detail.

Participant Recruitment and Selection Bias. While participants were recruited through a variety of channels, including personal, professional, and social platforms such as LinkedIn, the recruitment process introduces the possibility of selection bias. Although we did not exclusively select participants with personal connections, the presence of familiar contacts may have influenced the objectivity of the responses. The challenge of finding usability experts for studies of this nature often limits the recruitment process, as is evidenced by similar studies in the field with limited numbers of participants, e. g., comparable papers [16, 55, 62] involve 0, 3, and 5 experts, respectively. Our usability expert-centered evaluation incorporates the feedback of 8 experts. However, we acknowledge that this selection method could still lead to biased feedback. Future research should aim to minimize this bias by using more neutral recruitment strategies, such as through independent agencies or third-party platforms.

Participant Demographics and Expertise. The demographics and expertise of the participants also present a potential threat to validity. While participants held diverse professional roles—such as researchers, product owners, and consultants—all self-identified as having prior experience in UX studies, particularly in the use of eye-tracking data, which is essential for evaluating visual stimuli. Despite this, only one participant explicitly identified as a usability expert, which could limit the generalizability of the results to the broader community of usability practitioners. This mismatch between the participant pool and the intended target group (usability experts) may have impacted the depth and applicability of the feedback. Future studies should aim to recruit participants with more direct alignment to the target group to improve the relevance and accuracy of the evaluation results.

Design and Construct Validity. The design of the questionnaire itself introduces another potential bias. Referring to the evaluated approach as "our method" and presenting participants with a novel solution without direct comparisons to established baselines could have led to a more favorable evaluation based on the novelty rather than the actual utility of the approach. However, we encouraged participants to assess the method in comparison to their familiar workflows, asking them to evaluate how the proposed approach would enhance or integrate into their existing practices. In this way, participants' current practices served as an implicit baseline. Nonetheless, the absence of a direct comparison with alternative methods remains a limitation. Future studies may adopt more explicit comparative methodologies, such as incorporating baseline comparisons or conducting blinded evaluations, to ensure further objective feedback.

Limitations of Subjective Evaluations. The subjective nature of participant feedback in this study introduces an inherent threat to validity, as individual preferences and biases may influence responses. While the survey aimed to gather qualitative insights into the usability of visual stimuli discovery, we acknowledge that subjective evaluations can be influenced by a variety of factors, including personal familiarity with the authors' work or the novelty of the method itself. In future research, it will be important to explore strategies to reduce these biases, such as using objective performance metrics alongside qualitative feedback. However, we also recognize that

when evaluating novel functionalities, as presented in this paper, objective comparisons to baseline approaches may be inherently limited.

Despite above-mentioned limitations, the feedback gathered in this small-scale study offers valuable qualitative insights into the potential integration of visual stimuli discovery into usability expert workflows. We acknowledge that future studies should address these threats to validity by broadening the participant pool, refining the recruitment and questionnaire design, and incorporating more objective evaluation methods. These steps will further strengthen the generalizability and robustness of the findings.

9 LIMITATIONS AND FUTURE WORK

In this work, we have discussed a sophisticated method of visual stimuli discovery to address various complex issues in usability analysis. However, there is a huge design space for techniques and parameters involved in the realization of our method, and hence there are several limitations that also present opportunities for interesting future directions as following.

Choosing Web Sites for a Data Set. Our data set that was recorded on real-world Web sites has been challenging with respect to layouts (e. g., Steam), contents (e. g., TheGuardian), and advertisements (e. g., CNN, WebMD) which differ across user sessions. In the setup of a real-world usability evaluation we can expect the Web sites to be adjusted in layout, content, and advertisements such that they either appear similar to all users or differ only in specific aspects (A/B testing).

Recording Setup of the Web Sites. We chose to set the viewport size in our data set collection to $1{,}024 \times 768$ pixels, similar to [28]. This resolution is feasible to trigger responsive Web pages to display their tablet-layout[23] or their desktop-layout.[24] Nevertheless, we regard that many users on desktop and laptop computers experience Web sites nowadays on much bigger screens and mobile users on much smaller screens. Whereas many desktop-layouts of Web sites fill up the available space with a static color [43], mobile-layouts usually relocate the elements of Web pages to fit the available screen estate. Thus we argue that our data set comprises quite the upper limited of complexity in layout and dynamcis that most users face in the Web.

Exploring more Classifiers. In this work, we applied a selection of computer vision features using a random forest classifier. In the future, we can investigate different approaches to visual change classification, e. g., using a convolutional neural network. We could interpret the visual change detection as an image classification task, by feeding both frames to be compared into the convolutional neural network and output the decision whether there is a visual change between both frames. However, a convolutional neural network requires a huge amount of data for the training phase. This conflicts with the decision process of visual change, which must be performed by a usability expert. However, we imagine a synthetic data set that is comprised of crops from virtual screenshots from interfaces, whose visual change label could be assigned without manual intervention. The synthetic data set could be used to pre-train the convolutional neural network, while a data set with manual annotation might then be used to tune the weights of the final network.

Reducing the Annotation Effort. Updates to the interface in new releases or consideration of other interfaces require annotation of visual changes for a well performing classifier, which is a major effort for the usability expert. We argue that the annotation effort for a usability expert can be further reduced by suggesting incidents of visual change in a video recording. For this, we might employ a visual change classifier that has been trained on a plenitude of interfaces. We

[23] https://www.altamira.ai/blog/common-screen-sizes-for-responsive-web-design

[24] https://www.hobo-web.co.uk/best-screen-size or https://www.browserstack.com/guide/ideal-screen-sizes-for-responsive-design

have already reported the performance of a classifier that is trained on interfaces from entirely different categories than the interface in the test data, see Table 6 on page 26. The results may not be sufficient for direct application of the classifier, yet may be used to support a usability expert in reducing the annotation effort.

Understanding the Effects of User-Profiles on Visual Stimuli Discovery. The user experience on a Web site can significantly vary depending on the user's profile, including their specific goals and the Web site's functionality. In cases where a Web site is highly personalized, such as one with a curated newsfeed, it becomes challenging to create uniform page-level visual stimuli to gather interaction data from multiple users simultaneously. Instead, it is necessary to identify component-level visual stimuli, like individual news items, and analyze how users interact with these specific components. Our method for discovering visual stimuli is already effective for Web sites that present similar layouts to all users and provides a foundation for further developing approaches that can operate at a more granular, component-based level.

Handling the Fixed Elements. One of the main limitations of our method implementation of WebVSD is that the fixed elements could not be handled efficiently. Our heuristic was that fixed elements are rectangular and static, however, transparent backgrounds, sub-menus, and animated chatbots make the fixed elements non-rectangular and often highly dynamic. To deal with this, we still believe that fixed elements should be treated separately from the interface body in the visual stimuli discovery. Otherwise, it would be difficult to understand whether a visual change is caused through changes in the interface itself or through different scrolling behaviors of contents. In the future, we imagine that a computer vision-based recognition of fixed elements might be more successful in handling non-rectangular and dynamic fixed elements. This would also lift the requirement to access the DOM at all, making WebVSD agnostic to interface definitions.

Improving the Merging of Stimulus Shots. We imagine that the approach for merging the stimulus shots into visual stimuli in WebVSD can be further improved. As of now, pixels from video frames, whether they are in the interface body or in an fixed element, are represented in exactly one visual stimulus. However, it might be more useful for a meaningful aggregation of eye gaze data to have a region represented in more than one visual stimulus. In addition, the similarity score for the decision of which stimulus shots to merge is based on the overlap in their pixel area. This can be changed to another metric or a machine learning approach could be employed to decide about the merging order of stimulus shots into visual stimuli.

Integrating the Interaction Recordings. Another idea is to incorporate the data from interaction recordings, i. e., eye gaze and mouse data, into the process of visual stimuli discovery. As of now, the visual stimuli are only used to analyze eye gaze and mouse data as overlay. However, data about interactions can also signify visual changes. For example, most visual changes do happen after a user has moved the mouse over a hoverable element or a user clicked on a menu icon. We might consider the interactions as an addition to the video recording and perform a visual change classification in parallel on that data, e. g., by feeding the interaction data also into a convolutional neural network.

Moreover, it could be interesting to a usability expert which visual changes are seen by a user and which visual changes are ignored by a user. This can be achieved by using the eye gaze data to divide a frame into a foveated region and a peripheral region. Features of visual change might be weighted differently in each region.

Visual Changes through Scrolling. Some interfaces will remain difficult to be processed such that eye gaze data can be aggregated across user sessions in a meaningful way. Especially product

(a) 2700 pixels scroll offset (b) 3150 pixels scroll offset (c) 3900 pixels scroll offset

Fig. 21. The animation on the product page of the Apple iPhone XS is controlled by the scrolling of a user.

presentations in Web stores that make use of advanced animations at scroll events. A prominent example is the product page of the Apple iPhone XS.[25] The page interprets the scroll offset as a time indicator for a viewport-relative animation. A user can scroll to play the animation, stop scrolling to pause the animation, and scroll up to revert the animation. See Figure 21 for an example of the Web page at different scroll offsets. Therefore, a time-wise synchronization between users remains difficult due to the scrolling-driven playback.

The Future of Usability Analysis. We believe that the usefulness of the visual stimuli discovery will further benefit from the integration into existing analytical tools and development suites used by usability experts. Moreover, our method is not limited to the benefit of usability experts, but psychologists can also take advantage of the method in setting up experiments that investigate rich dynamic stimulus-response scenarios.

In the future, to deal with the increasing number of Web interfaces, evolving design trends, and interaction patterns, we would need more efficient methods for usability analysis. We envision an ecosystem that considers the interface code, the rendering of the interface during a user session, interaction data, and prior decisions of usability experts on similar interfaces, to automatically judge the usability of an interface and even to report the issues in the design to the respective developers. Such an ecosystem would require a deeper understanding of the elements on an interface and across interfaces. Especially compound elements that are crafted from multiple elements but appear as a consistent element to a user are required to be identified and tracked. Those compound elements appear to a user as photo carousels, expandable menus, product views, geographical maps, or chat widgets. We suggest detecting those compound elements across the discovered states of the interface and attempt to understand their internal state-machine, e. g., the photos in a carousel or the options in a menu. This will help to find the compound elements across views of an interface, as menus or product views might be consistent throughout the entire interface. Finally, we might be able to divide a visual stimulus into portions of reoccurring layouts across views of an interface and specific contents in an interface, or even only a single state, which would allow an even more insightful aggregation of interaction recordings. In the end, this level of understanding of an interface could allow us to make the behavioral analysis on highly personalized interfaces like social media services more efficient and feasible for a larger audience.

10 CONCLUSION

Determining visual stimuli of an interface is an important aspect to allow for more efficient usability studies with a sufficient number of users. In this regard, we presented a framework to

[25] https://web.archive.org/web/20190104023835/https://www.apple.com/iphone-xs, accessed on 3rd June 2021.

discover visual stimuli in video recordings of user sessions on dynamic interfaces. We introduced the detection of visual changes in video recordings of user sessions on dynamic interfaces. We examined various features from scene detection literature that are relevant in indicating visual changes and successfully trained classifiers for recognizing visual changes on video recordings of real-world Web sites. The quality assessment on our data set signified that the visual stimuli discovery may reduce the visual overload of usability experts while preserving an accurate representation of the user sessions regarding the interface body. We found that our method implementation WebVSD does not work optimally for fixed elements, but suggested methods to overcome this limitation in the future. Moreover, we estimated the usefulness in several case studies that replicated the task of a usability expert. We corroborated these findings through a small-scale survey, gathering early feedback. Looking ahead, towards the deployment of the solution, we anticipate broadening the study to involve a larger participant pool. This expansion will enable a more thorough assessment, offering deeper insights into the efficacy and usability of the proposed solution across a wider user demographic.

ACKNOWLEDGMENTS

The authors acknowledge the financial support by the Federal Ministry of Education and Research of Germany under the project numbers 01IS17095B and 01IS20030B. Moreover, the work was partially supported by the MICME (Multimodal Interaction and Collaboration In Medical Environment) project, funded by the Federal Ministry of Education and Research of Germany under the ZIM initiative with the project number KK5057901LF0. We are grateful for the test participants who participated in recording the data set, the student assistant who labeled a subset of the data to verify the decision process about visual change, and the experts who took part in our online surveys.

STATEMENT OF PREVIOUS RESEARCH

Much of the groundwork for this paper was completed during the doctoral research of the primary author. In comparison to the thesis, the article has undergone substantial expansion in the introduction, providing a more detailed explanation of motivation, defining the visual stimuli discovery in depth and introducing the phases of usability analysis in relation to our framework, now also depicted in Figure 2. Moreover, we have newly included a comprehensive overview and comparison of our approach with more related work in Table 1. Last, we have significantly enhanced the discussion of limitations and future work in this article. Besides reporting about the work in the thesis, the work has not been published before.

REFERENCES

[1] Tobii AB. 2018. *Sticky.* https://www.sticky.ai

[2] Abdallah MH Abbas, Khairil Imran Ghauth, and Choo-Yee Ting. 2022. User experience design using machine learning: a systematic review. *IEEE Access* 10 (2022), 51501–51514.

[3] M. Elgin Akpınar and Yeliz Yesılada. 2013. Vision Based Page Segmentation Algorithm: Extended and Perceived Success. In *Current Trends in Web Engineering*, Quan Z. Sheng and Jesper Kjeldskov (Eds.). Springer International Publishing, Cham, 238–252. https://doi.org/10.1007/978-3-319-04244-2_22

[4] The Chromium Authors. 2019. The Chromium Projects. https://www.chromium.org

[5] The FFmpeg Authors. 2019. FFmpeg. https://ffmpeg.org

[6] Nikola Banovic, Tovi Grossman, Justin Matejka, and George Fitzmaurice. 2012. Waken: Reverse Engineering Usage Information and Interface Structure from Software Videos. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology* (Cambridge, Massachusetts, USA) *(UIST '12)*. ACM, New York, NY, USA, 83–92. https://doi.org/10.1145/2380116.2380129

[7] Michael Barz and Daniel Sonntag. 2016. Gaze-Guided Object Classification Using Deep Neural Networks for Attention-Based Computing. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing:*

Adjunct (Heidelberg, Germany) *(UbiComp '16)*. Association for Computing Machinery, New York, NY, USA, 253–256. https://doi.org/10.1145/2968219.2971389

[8] Sanjay Batra and Ram R Bishu. 2007. Web usability and evaluation: issues and concerns. In *International Conference on Usability and Internationalization*. Springer, 243–249.

[9] Tony Beltramelli. 2018. Pix2code: Generating Code from a Graphical User Interface Screenshot. In *Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems* (Paris, France) *(EICS '18)*. Association for Computing Machinery, New York, NY, USA, Article 3, 6 pages. https://doi.org/10.1145/3220134.3220135

[10] David Beymer and Daniel M. Russell. 2005. WebGazeAnalyzer: A System for Capturing and Analyzing Web Reading Behavior Using Eye Gaze. In *CHI '05 Extended Abstracts on Human Factors in Computing Systems* (Portland, OR, USA) *(CHI EA '05)*. ACM, New York, NY, USA, 1913–1916. https://doi.org/10.1145/1056808.1057055

[11] Tanja Blascheck, Kuno Kurzhals, Michael Raschke, Michael Burch, Daniel Weiskopf, and Thomes Ertl. 2017. Visualization of Eye Tracking Data: A Taxonomy and Survey. *Computer Graphics Forum* 36, 8 (2017), 260–284. https://doi.org/10.1111/cgf.13079

[12] Gary Bradski. 2000. The OpenCV Library. *Dr. Dobb's Journal of Software Tools* (2000).

[13] Geert Brône, Bert Oben, and Toon Goedemé. 2011. Towards a More Effective Method for Analyzing Mobile Eye-Tracking Data: Integrating Gaze Data with Object Recognition Algorithms. In *Proceedings of the 1st International Workshop on Pervasive Eye Tracking Mobile Eye-Based Interaction* (Beijing, China) *(PETMEI '11)*. Association for Computing Machinery, New York, NY, USA, 53–56. https://doi.org/10.1145/2029956.2029971

[14] Sara Bunian, Kai Li, Chaima Jemmali, Casper Harteveld, Yun Fu, and Magy Seif Seif El-Nasr. 2021. VINS: Visual Search for Mobile User Interface Design. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 423, 14 pages. https://doi.org/10.1145/3411764.3445762

[15] Paolo Buono, Danilo Caivano, Maria Francesca Costabile, Giuseppe Desolda, and Rosa Lanzilotti. 2019. Towards the detection of ux smells: the support of visualizations. *IEEE Access* 8 (2019), 6901–6914.

[16] Michael Burch, Andreas Kull, and Daniel Weiskopf. 2013. AOI Rivers for Visualizing Dynamic Eye Gaze Frequencies. *Computer Graphics Forum* 32, 3pt3 (2013), 281–290. https://doi.org/10.1111/cgf.12115 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.12115

[17] Brian Burg, Andrew J. Ko, and Michael D. Ernst. 2015. Explaining Visual Changes in Web Interfaces. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software Technology* (Charlotte, NC, USA) *(UIST '15)*. ACM, New York, NY, USA, 259–268. https://doi.org/10.1145/2807442.2807473

[18] Deng Cai, Shipeng Yu, Ji-Rong Wen, and Wei-Ying Ma. 2003. Extracting Content Structure for Web Pages Based on Visual Representation. In *Proceedings of the 5th Asia-Pacific Web Conference on Web Technologies and Applications* (Xian, China) *(APWeb'03)*. Springer-Verlag, Berlin, Heidelberg, 406–417. https://doi.org/10.1007/3-540-36901-5_42

[19] John F. Canny. 1986. A Computational Approach to Edge Detection. *IEEE Trans. Pattern Anal. Mach. Intell.* 8, 6 (June 1986), 679–698. https://doi.org/10.1109/TPAMI.1986.4767851

[20] Tsung-Hsiang Chang, Tom Yeh, and Rob Miller. 2011. Associating the Visual Representation of User Interfaces with Their Internal Structures and Metadata. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology* (Santa Barbara, California, USA) *(UIST '11)*. Association for Computing Machinery, New York, NY, USA, 245–256. https://doi.org/10.1145/2047196.2047228

[21] Chunyang Chen, Sidong Feng, Zhenchang Xing, Linda Liu, Shengdong Zhao, and Jinshui Wang. 2019. Gallery D.C.: Design Search and Knowledge Discovery through Auto-Created GUI Component Gallery. *Proc. ACM Hum.-Comput. Interact.* 3, CSCW, Article 180 (Nov. 2019), 22 pages. https://doi.org/10.1145/3359282

[22] Chunyang Chen, Ting Su, Guozhu Meng, Zhenchang Xing, and Yang Liu. 2018. From UI Design Image to GUI Skeleton: A Neural Machine Translator to Bootstrap Mobile GUI Implementation. In *Proceedings of the 40th International Conference on Software Engineering* (Gothenburg, Sweden) *(ICSE '18)*. Association for Computing Machinery, New York, NY, USA, 665–676. https://doi.org/10.1145/3180155.3180240

[23] Jieshan Chen, Mulong Xie, Zhenchang Xing, Chunyang Chen, Xiwei Xu, Liming Zhu, and Guoqiang Li. 2020. Object Detection for Graphical User Interface: Old Fashioned or Deep Learning or a Combination?. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (Virtual Event, USA) *(ESEC/FSE 2020)*. Association for Computing Machinery, New York, NY, USA, 1202–1214. https://doi.org/10.1145/3368089.3409691

[24] Liang-Hua Chen, Yu-Chun Lai, and Hong-Yuan Mark Liao. 2008. Movie scene segmentation using background information. *Pattern Recognition* 41, 3 (Jan. 2008), 1056–1065. https://doi.org/10.1016/j.patcog.2007.07.024

[25] Sen Chen, Lingling Fan, Ting Su, Lei Ma, Yang Liu, and Lihua Xu. 2019. Automated Cross-Platform GUI Code Generation for Mobile Apps. In *AI4Mobile 2019 - 2019 IEEE 1st International Workshop on Artificial Intelligence for Mobile (AI4Mobile 2019 - 2019 IEEE 1st International Workshop on Artificial Intelligence for Mobile)*, Yang Liu, Minhui Xue, Lei Ma, and Li Li (Eds.). Institute of Electrical and Electronics Engineers Inc., United States, 13–16.

https://doi.org/10.1109/AI4Mobile.2019.8672718 Funding Information: We appreciate the reviewers' constructive feedback. This work is partially supported by NSFC Grant 61502170, NTU Research Grant NGF-2017-03-033 and NRF Grant CRDCG2017-S04. Funding Information: ACKNOWLEDGMENTS We appreciate the reviewers' constructive feedback. This work is partially supported by NSFC Grant 61502170, NTU Research Grant NGF-2017-03-033 and NRF Grant CRDCG2017-S04.; 1st IEEE International Workshop on Artificial Intelligence for Mobile, AI4Mobile 2019 ; Conference date: 24-02-2019.

[26] The Qt Company. 2019. Cross-platform software development for embedded & desktop. https://www.qt.io

[27] CoolTool. 2018. *CoolTool.* https://cooltool.com

[28] Michael Cormier, Karyn Moffatt, Robin Cohen, and Richard Mann. 2016. Purely Vision-based Segmentation of Web Pages for Assistive Technology. *Comput. Vis. Image Underst.* 148, C (July 2016), 46–66. https://doi.org/10.1016/j.cviu.2016.02.007

[29] Laura Cowen, Linden Js Ball, and Judy Delin. 2002. An eye movement analysis of web page usability. In *People and computers XVI-memorable yet invisible*. Springer, 317–335.

[30] Biplab Deka, Zifeng Huang, Chad Franzen, Joshua Hibschman, Daniel Afergan, Yang Li, Jeffrey Nichols, and Ranjitha Kumar. 2017. Rico: A Mobile App Dataset for Building Data-Driven Design Applications. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology* (Quebec City, QC, Canada) *(UIST '17)*. ACM, New York, NY, USA, 845–854. https://doi.org/10.1145/3126594.3126651

[31] Biplab Deka, Zifeng Huang, Chad Franzen, Jeffrey Nichols, Yang Li, and Ranjitha Kumar. 2017. ZIPT: Zero-Integration Performance Testing of Mobile App Designs. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology* (Québec City, QC, Canada) *(UIST '17)*. Association for Computing Machinery, New York, NY, USA, 727–736. https://doi.org/10.1145/3126594.3126647

[32] Biplab Deka, Zifeng Huang, and Ranjitha Kumar. 2016. ERICA: Interaction Mining Mobile Apps. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology* (Tokyo, Japan) *(UIST '16)*. Association for Computing Machinery, New York, NY, USA, 767–776. https://doi.org/10.1145/2984511.2984581

[33] Manfred Del Fabro and Laszlo Böszörmenyi. 2013. State-of-the-art and future challenges in video scene detection: a survey. *Multimedia Systems* 19, 5 (Oct. 2013), 427–454. https://doi.org/10.1007/s00530-013-0306-4

[34] Giuseppe Desolda, Andrea Esposito, Rosa Lanzilotti, and Maria F Costabile. 2021. Detecting emotions through machine learning for automatic UX evaluation. In *Human-Computer Interaction–INTERACT 2021: 18th IFIP TC 13 International Conference, Bari, Italy, August 30–September 3, 2021, Proceedings, Part III 18*. Springer, 270–279.

[35] Morgan Dixon and James Fogarty. 2010. Prefab: Implementing Advanced Behaviors Using Pixel-based Reverse Engineering of Interface Structure. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Atlanta, Georgia, USA) *(CHI '10)*. ACM, New York, NY, USA, 1525–1534. https://doi.org/10.1145/1753326.1753554

[36] Claudia Ehmke and Stephanie Wilson. 2007. Identifying web usability problems from eyetracking data. (2007).

[37] Sukru Eraslan, Yeliz Yesilada, and Simon Harper. 2016. Eye Tracking Scanpath Analysis on Web Pages: How Many Users?. In *Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications* (Charleston, South Carolina) *(ETRA '16)*. ACM, New York, NY, USA, 103–110. https://doi.org/10.1145/2857491.2857519

[38] EYEVIDO GmbH. 2021. *EYEVIDO Lab.* https://eyevido.de/en/how-does-eyevido-lab-works

[39] Gunnar Farnebäck. 2003. Two-frame Motion Estimation Based on Polynomial Expansion. In *Proceedings of the 13th Scandinavian Conference on Image Analysis* (Halmstad, Sweden) *(SCIA'03)*. Springer-Verlag, Berlin, Heidelberg, 363–370. http://dl.acm.org/citation.cfm?id=1763974.1764031

[40] Shirin Feiz, Jason Wu, Xiaoyi Zhang, Amanda Swearngin, Titus Barik, and Jeffrey Nichols. 2022. Understanding Screen Relationships from Screenshots of Smartphone Applications. In *27th International Conference on Intelligent User Interfaces* (Helsinki, Finland) *(IUI '22)*. Association for Computing Machinery, New York, NY, USA, 447–458. https://doi.org/10.1145/3490099.3511109

[41] Eyezag GbR. 2018. *Eyezag.* https://eyezag.de

[42] James J Gibson. 1960. The concept of the stimulus in psychology. *American psychologist* 15, 11 (1960), 694.

[43] Kim Golombisky and Rebecca Hagen. 2013. *White space is not your enemy: a beginner's guide to communicating visually through graphic, Web and multimedia design.* Routledge.

[44] Julián Grigera, Alejandra Garrido, José Matías Rivero, and Gustavo Rossi. 2017. Automatic detection of usability smells in web applications. *International Journal of Human-Computer Studies* 97 (2017), 129–148. https://doi.org/10.1016/j.ijhcs.2016.09.009

[45] Marc Hassenzahl and Noam Tractinsky. 2006. User experience-a research agenda. *Behaviour & information technology* 25, 2 (2006), 91–97.

[46] Alexander G. Haupmann and Michael J. Witbrock. 1998. Story Segmentation and Detection of Commercials in Broadcast News Video. In *Proceedings of the Advances in Digital Libraries Conference (ADL '98)*. IEEE Computer Society, Washington, DC, USA, 168–. http://dl.acm.org/citation.cfm?id=582987.785930

[47] Daniel Hienert, Dagmar Kern, Matthew Mitsui, Chirag Shah, and Nicholas J. Belkin. 2019. Reading Protocol: Understanding What Has Been Read in Interactive Information Retrieval Tasks. In *Proceedings of the 2019 Conference on Human Information Interaction and Retrieval* (Glasgow, Scotland UK) *(CHIIR '19)*. Association for Computing Machinery, New York, NY, USA, 73–81. https://doi.org/10.1145/3295750.3298921

[48] Kenneth Holmqvist, Marcus Nyström, Richard Andersson, Richard Dewhurst, Halszka Jarodzka, and Joost Van de Weijer. 2011. *Eye tracking: A comprehensive guide to methods and measures*. OUP Oxford.

[49] Forrest Huang, John F. Canny, and Jeffrey Nichols. 2019. *Swire: Sketch-Based User Interface Retrieval*. Association for Computing Machinery, New York, NY, USA, 1–10. https://doi.org/10.1145/3290605.3300334

[50] Yue Jiang, Wolfgang Stuerzlinger, and Christof Lutteroth. 2021. ReverseORC: Reverse Engineering of Resizable User Interface Layouts with OR-Constraints. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 316, 18 pages. https://doi.org/10.1145/3411764.3445043

[51] George H. Joblove and Donald Greenberg. 1978. Color Spaces for Computer Graphics. *SIGGRAPH Comput. Graph.* 12, 3 (Aug. 1978), 20–25. https://doi.org/10.1145/965139.807362

[52] Rodrigo M. Kishi, Tiago H. Trojahn, and Rudinei Goularte. 2016. An Evaluation of Readily Usable Automatic Video Shot Segmentation Techniques. In *Proceedings of the 22nd Brazilian Symposium on Multimedia and the Web* (Teresina, Piaui; State, Brazil) *(Webmedia '16)*. ACM, New York, NY, USA, 199–202. https://doi.org/10.1145/2976796.2988174

[53] Rebecca Krosnick, Sang Won Lee, Walter S. Laseck, and Steve Onev. 2018. Expresso: Building Responsive Interfaces with Keyframes. In *2018 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. 39–47. https://doi.org/10.1109/VLHCC.2018.8506516

[54] Ranjitha Kumar, Arvind Satyanarayan, Cesar Torres, Maxine Lim, Salman Ahmad, Scott R. Klemmer, and Jerry O. Talton. 2013. Webzeitgeist: Design Mining the Web. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Paris, France) *(CHI '13)*. ACM, New York, NY, USA, 3083–3092. https://doi.org/10.1145/2470654.2466420

[55] Kuno Kurzhals, Marcel Hlawatsch, Florian Heimerl, Michael Burch, Thomas Ertl, and Daniel Weiskopf. 2016. Gaze Stripes: Image-Based Visualization of Eye Tracking Data. *IEEE Transactions on Visualization and Computer Graphics* 22, 1 (Jan. 2016), 1005–1014. https://doi.org/10.1109/TVCG.2015.2468091

[56] Kuno Kurzhals, Marcel Hlawatsch, Christof Seeger, and Daniel Weiskopf. 2017. Visual Analytics for Mobile Eye Tracking. *IEEE Transactions on Visualization and Computer Graphics* 23, 1 (Jan. 2017), 301–310. https://doi.org/10.1109/TVCG.2016.2598695

[57] Kuno Kurzhals and Daniel Weiskopf. 2013. Space-Time Visual Analytics of Eye-Tracking Data for Dynamic Stimuli. *IEEE Transactions on Visualization and Computer Graphics* 19, 12 (Dec. 2013), 2129–2138. https://doi.org/10.1109/TVCG.2013.194

[58] Fabrizio Lamberti, Gianluca Paravati, Valentina Gatteschi, and Alberto Cannavo. 2017. Supporting Web Analytics by Aggregating User Interaction Data From Heterogeneous Devices Using Viewport-DOM-Based Heat Maps. *IEEE Transactions on Industrial Informatics* 13, 4 (2017), 1989–1999. https://doi.org/10.1109/TII.2017.2658663

[59] J. Richard Landis and Gary G. Koch. 1977. The measurement of observer agreement for categorical data. *Biometrics* 33 1 (1977), 159–74.

[60] Effie Lai-Chong Law, Virpi Roto, Marc Hassenzahl, Arnold POS Vermeeren, and Joke Kort. 2009. Understanding, scoping and defining user experience: a survey approach. In *Proceedings of the SIGCHI conference on human factors in computing systems*. 719–728.

[61] Guillaume Lemaître, Fernando Nogueira, and Christos K. Aridas. 2017. Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning. *Journal of Machine Learning Research* 18, 17 (2017), 1–5. http://jmlr.org/papers/v18/16-365

[62] Nils Lichtenberg, Raphael Menges, Vladimir Ageev, Ajay Abisheck Paul George, Pascal Heimer, Diana Imhof, and Kai Lawonn. 2018. Analyzing Residue Surface Proximity to Interpret Molecular Dynamics. *Computer Graphics Forum* (2018). https://doi.org/10.1111/cgf.13427

[63] Thomas F. Liu, Mark Craft, Jason Situ, Ersin Yumer, Radomir Mech, and Ranjitha Kumar. 2018. Learning Design Semantics for Mobile Apps. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology* (Berlin, Germany) *(UIST '18)*. ACM, New York, NY, USA, 569–579. https://doi.org/10.1145/3242587.3242650

[64] Margaret Livingstone. 2002. *Vision and art : the biology of seeing*. Abrams, Harry N., New York, N.Y, 46–67.

[65] David G. Lowe. 2004. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision* 60, 2 (1 Nov. 2004), 91–110. https://doi.org/10.1023/B:VISI.0000029664.99615.94

[66] Dylan Lukes, John Sarracino, Cora Coleman, Hila Peleg, Sorin Lerner, and Nadia Polikarpova. 2021. Synthesis of Web Layouts from Examples. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (Athens, Greece) *(ESEC/FSE 2021)*. Association for Computing Machinery, New York, NY, USA, 651–663. https://doi.org/10.1145/3468264.3468533

[67] Miles Macleod and Ralph Rengger. 1993. The Development of DRUM: A Software Tool for Video-assisted Usability. *People and Computers VIII* 7 (1993), 293.

[68] Raphael Menges, Chandan Kumar, and Steffen Staab. 2019. Improving User Experience of Eye Tracking-Based Interaction: Introspecting and Adapting Interfaces. *ACM Trans. Comput.-Hum. Interact.* 26, 6, Article 37 (Nov. 2019), 46 pages. https://doi.org/10.1145/3338844 Accepted May 2019.

[69] Raphael Menges, Hanadi Tamimi, Chandan Kumar, Tina Walber, Christoph Schaefer, and Steffen Staab. 2018. Enhanced Representation of Web Pages for Usability Analysis with Eye Tracking. In *Proceedings of the 2018 ACM Symposium on Eye Tracking Research & Applications* (Warsaw, Poland) *(ETRA '18)*. ACM, New York, NY, USA, Article 18, 9 pages. https://doi.org/10.1145/3204493.3204535

[70] Ali Mesbah, Arie van Deursen, and Stefan Lenselink. 2012. Crawling Ajax-Based Web Applications Through Dynamic Analysis of User Interface State Changes. *ACM Trans. Web* 6, 1, Article 3 (March 2012), 30 pages. https://doi.org/10.1145/2109205.2109208

[71] Vasileios Mezaris, Evlampios Apostolidis, and Alexandros Pournaras. [n.d.]. Video Shot and Scene Segmentation - MKLab. https://mklab.iti.gr/results/video-shot-and-scene-segmentation

[72] Dalibor Mitrovic, Stefan Hartlieb, Matthias Zeppelzauer, and Christian Breiteneder. 2010. Scene Segmentation in Artistic Archive Documentaries. In *HCI in Work and Learning, Life and Leisure*. Springer-Verlag Berlin Heidelberg„ 6389, 400–410. http://publik.tuwien.ac.at/files/PubDat_188968.pdf Vortrag: 6th Symposium of the Workgroup Human-Computer Interaction and Usability Engineering, USAB 2010, Klagenfurt; 2010-11-04 – 2010-11-05.

[73] Dalibor Mitrović, Stefan Hartlieb, Matthias Zeppelzauer, and Maia Zaharieva. 2010. Scene Segmentation in Artistic Archive Documentaries. In *HCI in Work and Learning, Life and Leisure*, Gerhard Leitner, Martin Hitz, and Andreas Holzinger (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 400–410.

[74] Kevin Moran, Carlos Bernal-Cárdenas, Michael Curcio, Richard Bonett, and Denys Poshyvanyk. 2020. Machine Learning-Based Prototyping of Graphical User Interfaces for Mobile Apps. *IEEE Transactions on Software Engineering* 46, 2 (2020), 196–221. https://doi.org/10.1109/TSE.2018.2844788

[75] Tuan Anh Nguyen and Christoph Csallner. 2015. Reverse Engineering Mobile Application User Interfaces with REMAUI. In *Proceedings of the 30th IEEE/ACM International Conference on Automated Software Engineering* (Lincoln, Nebraska) *(ASE '15)*. IEEE Press, 248–259. https://doi.org/10.1109/ASE.2015.32

[76] Tam The Nguyen, Phong Minh Vu, Hung Viet Pham, and Tung Thanh Nguyen. 2018. Deep Learning UI Design Patterns of Mobile Apps. In *Proceedings of the 40th International Conference on Software Engineering: New Ideas and Emerging Results* (Gothenburg, Sweden) *(ICSE-NIER '18)*. Association for Computing Machinery, New York, NY, USA, 65–68. https://doi.org/10.1145/3183399.3183422

[77] Jakob Nielsen and Kara Pernice. 2010. *Eyetracking web usability*. New Riders.

[78] Jakub Štěpán Novák, Jan Masner, Petr Benda, Pavel Šimek, and Vojtěch Merunka. 2023. Eye tracking, usability, and user experience: A systematic review. *International Journal of Human–Computer Interaction* (2023), 1–17.

[79] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, and et al. 2011. Scikit-Learn: Machine Learning in Python. *J. Mach. Learn. Res.* 12, null (Nov. 2011), 2825–2830.

[80] Alex Poole and Linden J Ball. 2006. Eye tracking in HCI and usability research. In *Encyclopedia of human computer interaction*. IGI global, 211–219.

[81] Sylvain Senecal, Pawel J Kalczynski, and Jacques Nantel. 2005. Consumers' decision-making process and their online shopping behavior: a clickstream analysis. *Journal of Business research* 58, 11 (2005), 1599–1608.

[82] Naziha Shekh.Khalil, Ecem Dogruer, Abdulmohimen K. O. Elosta, Sukru Eraslan, Yeliz Yesilada, and Simon Harper. 2020. EyeCrowdata: Towards a Web-Based Crowdsourcing Platform for Web-Related Eye-Tracking Data. In *ACM Symposium on Eye Tracking Research and Applications* (Stuttgart, Germany) *(ETRA '20 Adjunct)*. Association for Computing Machinery, New York, NY, USA, Article 31, 6 pages. https://doi.org/10.1145/3379157.3391304

[83] Jakub Simko and Jakub Vrba. 2019. Screen Recording Segmentation to Scenes for Eye-Tracking Analysis. *Multimedia Tools Appl.* 78, 2 (Jan. 2019), 2401–2425. https://doi.org/10.1007/s11042-018-6369-7

[84] Ray Smith. 2007. An Overview of the Tesseract OCR Engine. In *Proceedings of the Ninth International Conference on Document Analysis and Recognition - Volume 02 (ICDAR '07)*. IEEE Computer Society, Washington, DC, USA, 629–633. http://dl.acm.org/citation.cfm?id=1304596.1304846

[85] Sören Sonnenburg, Gunnar Rätsch, Sebastian Henschel, Christian Widmer, Jonas Behr, Alexander Zien, Fabio De Bona, Alexander Binder, Christian Gehl, and Vojtech Franc. 2010. The SHOGUN Machine Learning Toolbox. *Journal of Machine Learning Research* 11 (2010), 1799–1802. https://dl.acm.org/citation.cfm?id=1859911

[86] RealEye sp. z o. o. 2018. *RealEye*. https://www.realeye.io

[87] Maximilian Speicher, Andreas Both, and Martin Gaedke. 2014. Ensuring web interface quality through usability-based split testing. In *International conference on web engineering*. Springer, 93–110.

[88] Åsne Stige, Efpraxia D Zamani, Patrick Mikalef, and Yuzhen Zhu. 2023. Artificial intelligence (AI) for user experience (UX) design: a systematic literature review and future research agenda. *Information Technology & People* (2023).

[89] Jonathan Strohl, Christian Gonzalez, Jacob Sauser, Soodeh Montazeri, and Brian Griepentrog. 2015. Creating forms and disclosures that work: using eye tracking to improve the user experience. In *Universal Access in Human-Computer Interaction. Access to Today's Technologies: 9th International Conference, UAHCI 2015, Held as Part of HCI International 2015, Los Angeles, CA, USA, August 2-7, 2015, Proceedings, Part I 9*. Springer, 121–131.

[90] Amanda Swearngin, Mira Dontcheva, Wilmot Li, Joel Brandt, Morgan Dixon, and Andrew J. Ko. 2018. Rewire: Interface Design Assistance from Examples. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3173574.3174078

[91] The WebM Project. 2019. The WebM Project. https://www.webmproject.org/code

[92] Tobii AB. 2020. *Tobii Pro Lab User's Manual*. Version 1.138.1.

[93] Takumi Toyama, Thomas Kieninger, Faisal Shafait, and Andreas Dengel. 2012. Gaze Guided Object Recognition Using a Head-Mounted Eye Tracker. In *Proceedings of the Symposium on Eye Tracking Research and Applications* (Santa Barbara, California) *(ETRA '12)*. Association for Computing Machinery, New York, NY, USA, 91–98. https://doi.org/10.1145/2168556.2168570

[94] Ba Tu Truong, Svetha Venkatesh, and Chitra Dorai. 2003. Scene extraction in motion pictures. *IEEE Transactions on Circuits and Systems for Video Technology* 13, 1 (Jan. 2003), 5–15. https://doi.org/10.1109/TCSVT.2002.808084

[95] Tu Truong, Svetha Venkatesh, and Chitra Dorai. 2003. Scene extraction in motion pictures. *Circuits and Systems for Video Technology, IEEE Transactions on* 13 (02 2003), 5 – 15. https://doi.org/10.1109/TCSVT.2002.808084

[96] Jeroen Vendrig and Marcel Worring. 2002. Systematic Evaluation of Logical Story Unit Segmentation. *Trans. Multi.* 4, 4 (Dec. 2002), 492–499. https://doi.org/10.1109/TMM.2002.802021

[97] Gang Wang, Tristan Konolige, Christo Wilson, Xiao Wang, Haitao Zheng, and Ben Y Zhao. 2013. You are how you click: Clickstream analysis for sybil detection. In *22nd USENIX Security Symposium (USENIX Security 13)*. 241–256.

[98] Gang Wang, Xinyi Zhang, Shiliang Tang, Haitao Zheng, and Ben Y Zhao. 2016. Unsupervised clickstream clustering for user behavior analysis. In *Proceedings of the 2016 CHI conference on human factors in computing systems*. 225–236.

[99] Zhou Wang, Alan Bovik, Hamid Sheikh, and Eero Simoncelli. 2004. Image Quality Assessment: From Error Visibility to Structural Similarity. *Trans. Img. Proc.* 13, 4 (April 2004), 600–612. https://doi.org/10.1109/TIP.2003.819861

[100] Thomas D. White, Gordon Fraser, and Guy J. Brown. 2019. Improving Random GUI Testing with Image-Based Widget Detection. In *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis* (Beijing, China) *(ISSTA 2019)*. Association for Computing Machinery, New York, NY, USA, 307–317. https://doi.org/10.1145/3293882.3330551

[101] Tom Yeh, Tsung-Hsiang Chang, and Robert C. Miller. 2009. Sikuli: Using GUI Screenshots for Search and Automation. In *Proceedings of the 22Nd Annual ACM Symposium on User Interface Software and Technology* (Victoria, BC, Canada) *(UIST '09)*. ACM, New York, NY, USA, 183–192. https://doi.org/10.1145/1622176.1622213

[102] Ramin Zabih, Justin Miller, and Kevin Mai. 1995. A Feature-based Algorithm for Detecting and Classifying Scene Breaks. In *Proceedings of the Third ACM International Conference on Multimedia* (San Francisco, California, USA) *(MULTIMEDIA '95)*. ACM, New York, NY, USA, 189–200. https://doi.org/10.1145/217279.215266

[103] Herbert Zettl. 2002. *Essentials of Applied Media Aesthetics*. Springer US, Boston, MA, 11–38. https://doi.org/10.1007/978-1-4615-1119-9_2

[104] Xiaoyi Zhang, Lilian de Greef, Amanda Swearngin, Samuel White, Kyle Murray, Lisa Yu, Qi Shan, Jeffrey Nichols, Jason Wu, Chris Fleizach, Aaron Everitt, and Jeffrey P Bigham. 2021. Screen Recognition: Creating Accessibility Metadata for Mobile Applications from Pixels. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 275, 15 pages. https://doi.org/10.1145/3411764.3445186

[105] Xiaoyi Zhang, Anne Spencer Ross, and James Fogarty. 2018. Robust Annotation of Mobile Application Interfaces in Methods for Accessibility Repair and Enhancement. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology* (Berlin, Germany) *(UIST '18)*. ACM, New York, NY, USA, 609–621. https://doi.org/10.1145/3242587.3242616

[106] Songhao Zhu and Yuncai Liu. 2008. A novel scheme for video scenes segmentation and semantic representation. In *2008 IEEE International Conference on Multimedia and Expo*. 1289–1292. https://doi.org/10.1109/ICME.2008.4607678

A PROTOCOL FOR DATA SET RECORDING

This data set shall contain visual dynamics on Web pages. On each of the following Web sites, explore each visited page thoroughly. Hover over elements and menus, read descriptions, and navigate through carousels. You may also rest for some seconds and observe automatically changing slides. Try not to click on a hyperlink other than described in the tasks below.

A.1 Shopping

(1) https://www.walmart.com (click on the burger menu in the top-left corner and on carousel navigations)
   (a) Click under “In The Spotlight” on “Toys,” in the bottom of the page
   (b) Click on a toy of choice
(2) https://www.amazon.com (hover the departments and click carousel navigations, click on “quick look”)
   (a) Click on “Deals in Electronics”
   (b) Click on an item of choice.
(3) https://store.steampowered.com (click on the carousel navigations)
   (a) Click on “Free Games”
   (b) Click on a game of choice

A.2 News

(4) https://www.reddit.com/r/pics/top/?t=month (hover over user names in post replies)
   (a) Click on the post “His adventure is just beginning”
   (b) Go back to the overview and click on another post of choice
(5) https://edition.cnn.com (click on the burger menu in the top right corner)
   (a) Click under “entertainment” on “Screen” in the bottom of the page
   (b) Click on a review of choice
(6) https://www.theguardian.com/international (click on “More” in the menu bar)
   (a) Click on “Sport” in the top menu
   (b) Click on an article of choice

A.3 Health

(7) https://www.nih.gov (hover over the menu bar)
   (a) Click on “Health Information”
   (b) Click on an article of choice
(8) https://www.webmd.com (hover over the menu bar)
   (a) Click on “Eye Health” in the bottom of the page
   (b) Click on a “RELATED TO EYE HEALTH” topic of choice
(9) https://www.mayoclinic.org (click the burger menu on top)
   (a) Click on “Mayo Clinic’s campus in Arizona” in the bottom of the page
   (b) Click on a further hyperlink of choice

A.4 Cars

(10) https://www.gm.com (click on “OUR STORIES” in the menu bar, hover over the images)
   (a) Click on “Our Brands”
   (b) Click on a brand of choice
(11) https://www.nissanusa.com (click on the menu entries on top, click in the car carousel on the categories)
   (a) Click under “THE FUTURE OF NISSAN INTELLIGENT MOBILITY” on “Learn More”
   (b) Click a further hyperlink of choice
(12) https://www.kia.com/us/en/home (click in the car carousel on the categories)
   (a) Click under “Technology” on “Performance” in the bottom of the page
   (b) Click a further hyperlink of choice

B ESTIMATION OF SCROLL OFFSET

Across all Web sites, we have faced difficulties in aligning the scroll offset as recorded in the datacasts with the scroll offset that is visible in the video recordings. We could not find an affine transformation of the series of scroll offsets, e. g., adding a fixed delay to the timestamps, which would align the scroll offsets correctly. The non-synchronicity of reported scroll offsets and the visual scroll offset might be caused by the parallel execution of rendering, input processing, and script execution in modern Web engines. However, an incorrect measure of the scroll offset may trigger the visual change classifier falsely. Therefore, we have implemented a method to compute the scroll offset on a video recording using computer vision features. In our method, we match features between consecutive frames and estimate a transformation that would explain the offset of the key points of the matching features between both frames. We interpret that transformation as the scroll offset.

Initially, we divide each frame in a 4×3 grid and compute in total 1,000 ORB [12] keypoints with descriptors on the Y channel (brightness information in the VP9 codec, see Section C for details). We exclude areas covered by fixed elements from the computations. Moreover, there might be elements on an interface, like buttons with a similar design, that occur multiple times. These elements can create highly similar descriptors, which would make a mapping between the descriptors of two frames ambiguous. Thus, we match the descriptors of a frame with each other and remove those pairs of descriptors that have a hamming distance of lower than five.

After having features for each frame, we match the features between two consecutive frames. We consider a feature pair with a hamming distance smaller or equal to ten as a match. We compute a homography with the RANSAC algorithm[26] on the matches, estimating the most probable transformation between the key points in the consecutive frames. We interpret the vertical translation component of that transformation as scrolling, which we use to adjust the scroll offset originally reported by JavaScript. Furthermore, we reset the scroll offset when a `document.hidden` event has been reported by JavaScript, as this event indicates that the Web browser is about to load a different Web page.

In the following, we show how scroll offsets differ between the datacast and our method. The x-axis represents the frame indices from the video recordings. The y-axis represents the scroll offset in pixels. The yellow line denotes the scroll offset as reported by the Qt WebEngine [26]. The red dotted line denotes the scroll offset as reported by a JavaScript callback. The blue area (instead of a line for readability reasons) denotes the scroll offset as computed with our method. In our experience, the method works well most of the time and outperforms the methods relying solely on the Qt WebEngine or JavaScript callback. In case an incorrect estimation of the scroll offset is propagated to visual stimuli, we have counted them as incorrect in the quality evaluation in Section 7.

[26] https://docs.opencv.org/master/d1/de0/tutorial_py_feature_homography.html

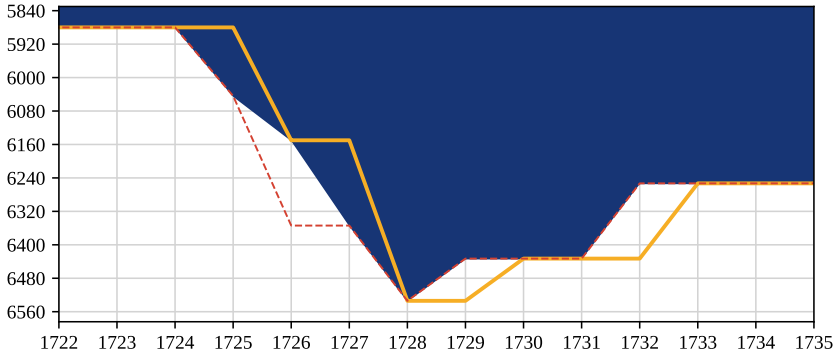


Fig. 22. A scroll sequence by the fourth user on Amazon. General misalignment of scrolling as reported by the Qt WebEngine and JavaScript is visible.

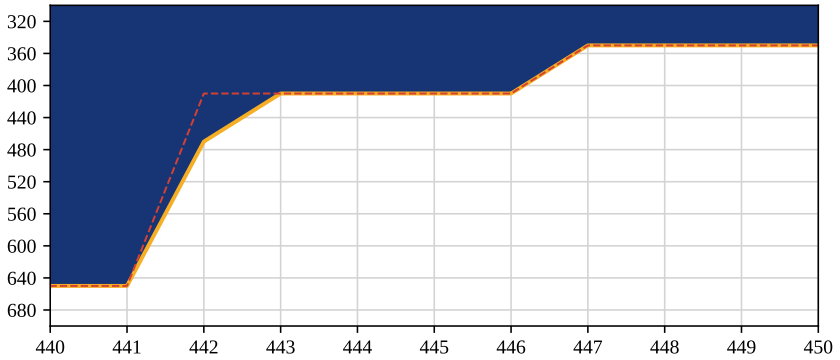


Fig. 23. A scroll sequence by the third user on NIH. Here, our method agrees with the scrolling reported by the Qt WebEngine, but not with the scrolling reported by JavaScript.

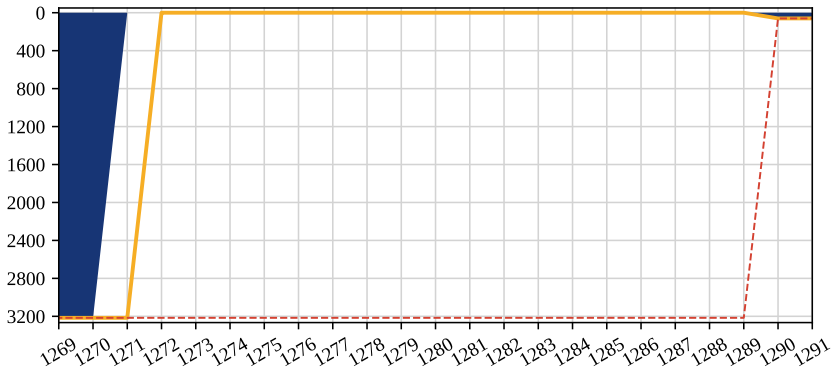


Fig. 24. A scroll sequence by the fourth user on Amazon. On frame 1,271, a new Web page begins to load. The Qt WebEngine notices the change already one frame later. The JavaScript callback reports about the scrolling only after the Web page had been loaded and the user scrolled.

C FEATURES FOR VISUAL CHANGE CLASSIFICATION

In this work, we propose to use computer vision features to detect visual changes on video recordings of Web browsing sessions. We suggest 53 features as listed in Table 2 on page 22. In the following, we provide details about the features and variations of features, alongside the color spaces in which we calculate the features or variations, respectively. The features are designed to describe the difference between two image pairs, i. e., to detect a visual change from one frame in the video recording to the next frame in the video recording. The names of the features as listed in Table 2 and referred to in the work are given in parentheses.

Color Spaces. An image from a video frame can be interpreted as a rectangular matrix of pixels. Each pixel holds information about its color. There are various color spaces to define the color. In our work, we employ the RGB, the HSL, and the YUV color spaces.

The RGB color space is the most common in computer vision. Each pixel holds brightness information about its red, green, and blue brightness. Almost all monitor panels nowadays consist of corresponding red, green, and blue subpixels.

The HSL color space is an alternative representation of the RGB color space and is more aligned with human perception of colors [51, 103]. It defines color as values of hue, saturation, and lightness.

The YUV color space is a color space that was used for analog TV broadcasts. The color is divided into a grayscale signal, the Y component, that can be straightforwardly interpreted by black-and-white TVs. The other two channels U and V contain the chroma signal. Those two channels can be interpreted by color TVs to enrich the picture with colors.

All video recordings in our data set (Section 4) have been compressed using the VP9 codec. The VP9 codec[27] uses the YUV color space to store the color information of the video frames. It to stores the grayscale signal with full resolution in the Y channel, while downsampling the pixel resolution of a video frame for the chroma signal of the U and V channel. This is motivated by the fact that human perception is less sensitive to compression in grayscale – which defines the contrast of an image – than for compression of chroma [64].

We take each video frame in the YUV color space and convert the colors to the RGB color space and the HSL color space. Moreover, we take the Y channel directly as a grayscale image. Certain features have been computed for multiple color spaces. We indicate the utilized color space in the description and the name of each feature. For example, the name `agg_h/s/l` describes the three variations of a feature that aggregates color values (`agg`). The three variations in `agg_h/s/l` are the aggregation in the hue (`h`), saturation (`s`), and lightness (`l`) channels.

Value-based Features. Given an image pair, we can directly interpret the color values from the pixels. When there is a visual change between two images, some values in the pixel matrix change. We compare each pixel in one image with the pixel at the same coordinate in the other image and sum up the differences. We suggest two approaches for the summation of differences, resulting in two features that have eight variations each. First, we can sum up the value difference between corresponding pixels for various color spaces (`agg_bgr/b/g/r`, `agg_h/s/l`, and `agg_gray`). Second, we can count how many corresponding pixels have different values, regardless of the exact difference in the value. Again, this can be performed across various color spaces (`count_bgr/b/g/r`, `count_h/s/l`, and `count_gray`).

Histogram-based Features. Histograms represent the distributions of values in the color channels of an image. Changes in the color distribution might indicate shifts in what is depicted on an interface, e. g., when a menu is expanded, or a viewport-filling gallery is shown. Given an image

[27] https://www.webmproject.org/vp9/levels

pair, we compute histograms with 16 uniform bins for each image, considering the RGB and HSL color space and the images as grayscale. Then, we compute the correlation between the corresponding histograms of both images (`corr_b/g/r`, `corr_h/s/l`, and `corr_gray`).

Edge-based Features. Another popular method in image processing and recognition is to estimate discontinuities in images, for which human perception is especially sensitive. The edge change fraction [102] counts the pixels of edges that are either introduced or removed when transitioning from the preceding image to the succeeding. Given an image pair, the maximum count of those incoming edges and outgoing edges is considered as a feature (`change_fraction`). We extract edges from the images with a Canny filter [19] on the grayscale image with a lower threshold of 64 and an upper threshold of 192. We set the dilation kernel to search for incoming and outgoing edges to five pixels diameter.

Signal-based Features. Signal-based features are used in image processing to measure the quality differences between the two images. The features attempt to approximate the human perception of reconstruction quality in image compression. This is useful to estimate the effect of compression on image or video material and might be also useful to detect a visual change between two images. Given an image pair, we consider one image as the original image and the other image as the compressed image.

One signal-based feature we use is the peak signal-to-noise ratio (PSNR) on grayscale images. It considers the maximum intensity within the original image and the compressed image and relates the maximum intensity to the mean square error between the original image and the compressed image. We compute this feature for the grayscale version of the image pair (`psnr`).

Another signal-based feature we use is the structural similarity (SSIM) [99] index. The SSIM index compares pairwise windows from the original image and the compressed image. The comparison is performed with the combination of three characteristics of an image: luminance, contrast, and structure. The result is a decimal value in a range between -1 and 1. A value of 0 indicates no structural similarity. A value of 1 indicates perfect structural similarity. A negative value speaks for locally inverted image structures, yet is rather uncommon. The SSIM index can be calculated across an entire image, resulting in an SSIM map. We calculate the SSIM map using a sliding Gaussian window of size 11×11 pixels. The average of the SSIM map is reported as mean structural similarity (MSSIM). We calculate the MSSIM for the blue, green, and red channels of the RGB color space (`mssim_b/g/r`).

Optical-flow-based Features. Objects in a video are recorded as a set of adjacent pixels. Optical flow assumes that between two contiguous video frames the set of pixels constituting an object either stays in one position or moves as a whole. Following this assumption, optical flow allows for recognizing moving objects. Given a pair of images, we compute a dense optical flow [39] on the grayscale version of the image pair. We report the mean and standard deviation of the angle (`angle_mean/std`) and the magnitude (`mag_max/mean/std`) of the detected movement.

SIFT-based Features. The scale-invariant feature transform (SIFT) [65] algorithm is used to detect and describe local features in images. Given an image pair, we let OpenCV [12] choose 500 key points on the grayscale version of each image and compute a descriptor per keypoint. Then, we match the descriptors from both images using the Euclidean norm. We count the found matches and filter the matching descriptors with values of 0, 4, 16, 64, 256, and 512 as a threshold (`match/_-0/4/16/64/256/512`). Matching keypoint pairs with a descriptor distance of zero are rare, whereas a set of matches at a descriptor distance of a value that is at least 500 contains almost all matches. We also check how many descriptor matches are also spatially close to each other on the images with a radius of less or equal to three pixels (`match_spatial`). The combination of descriptor matches

and spatial matches – analogously to optical flow – can reveal moving objects in an interface. Furthermore, we report about minimum, maximum, mean, and standard deviation of the descriptor distances (`match_dist_min/max/mean/std`).

Text-based Features. Interfaces often display substantial amounts of text. Frequently, dynamic interfaces present additional text on the fly, e. g., sub-categories in an online shop or pop-ups that provide details about a product. For example, when expanding the menu of a music shopping Web site, the words "Phil Collins" may appear. Given an image pair, we detect text using the Tesseract 4.0[28] optical character recognition library. Tesseract employs an LSTM neural network architecture to detect text on images. We use the official training data for English as provided by the developers to train the neural network.[29] After the application of Tesseract on our data, we filter the detected text with a confidence value above 0.5 and erase non-ASCII characters.

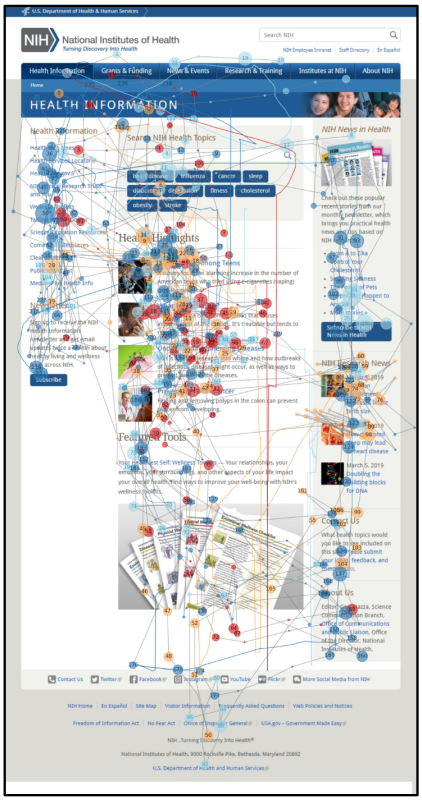
We use a bag-of-words representation for comparing two sets of recognized words from the two images. It is to be noticed that one word can occur multiple times in a set. If we recognize two times "car", the word "car" will be present two times in the set of recognized words. We exclude words of less than three characters. We match the words from the two images pairwise and report the number of terms that have no matching partner in the other image (`diff_words_count`. As another feature, we count the unique words in both images. For this, we take the two sets of words from the two images and erase duplicates in each set. Then, we erase words that occur in both images. Finally, we take the number of words left from both images (`unique_term_count`).

Another popular approach to compare texts is a character n-gram. Given a string, n-grams represent the string as the bag of all (possibly overlapping) substrings of length n. A string of length k has k-n+1 substrings of this kind. When considering n = 3 and the text "click here," the n-grams would be "cli," "lic," and so forth. The tokens cover characters across words because space itself is considered as a character. Similar to Sikuli [101], a tool that uses image processing on the screen contents to recognize interface elements, we define n = 3 in our experiments. We consider the set of n-grams per image and find the matches. Then, we report features like the count of matches of n-grams between the two images (`n_grams_match_count`), the amount of n-grams (`n_grams_min_count/max_count`), the ratio of the count of matches against the minimum number of unique n-grams from both images (`n_grams_match_ratio`), the Jaccard similarity of the n-grams (`n_grams_jaccard`), and the vocabulary size of both images combined (`n_grams_vocabulary_size`).

[28] https://github.com/tesseract-ocr/tesseract
[29] https://github.com/tesseract-ocr/tessdata

D DISCOVERED VISUAL STIMULI ON NIH.GOV

In the following, we show all thirty visual stimuli that we discovered in the video recordings of four users on NIH.gov, excluding fixed elements. Gaze data and mouse cursor movements by the four users have been added as overlays. We plot gaze data as scanpath and mouse cursor movements as lines. Each user is represented by a unique color. We provide the file path to the image in the data set below each image.



nih\stimuli\1_html\10.png

nih\stimuli\1_html\0.png

nih\stimuli\1_html\1.png

nih\stimuli\1_html\13.png

Fig. 25. Four visual stimuli discovered in the user sessions on NIH. They demonstrate the aggregation of a lot of interaction data by multiple users.

nih\stimuli\1_html\2.png

nih\stimuli\1_html\3.png

nih\stimuli\1_html\4.png

nih\stimuli\1_html\6.png

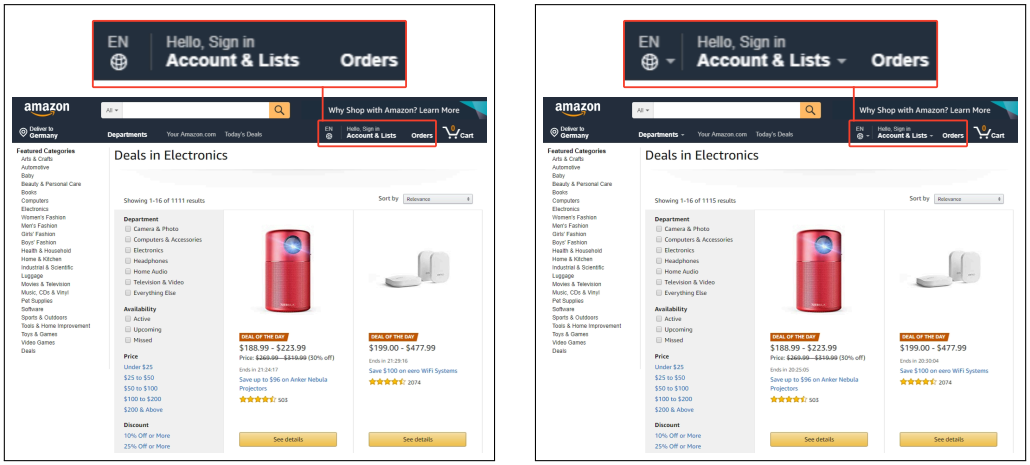
nih\stimuli\1_html\7.png

nih\stimuli\1_html\5.png

Fig. 26. Six visual stimuli discovered in the user sessions on NIH. They show how the users have interacted with the pop-up-menus on the front page.

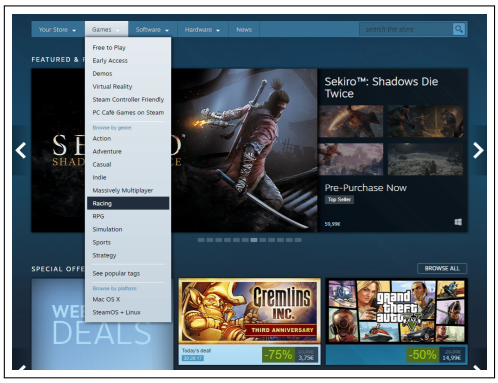
E EXAMPLES FROM THE QUALITY EVALUATION

The assessment in Section 7 shows that visual stimuli can reduce the amount of information that a usability expert has to examine substantially while reflecting the contents of the video recordings correctly. However, our implementation of visual stimuli discovery does not produce perfect results, i. e., sometimes the visual change classifier fails in detecting a visual change. Following, we show examples from the annotation process for the quality assessment. On the left, we display the original frame from the video recording. On the right, we display the region from the visual stimulus that should represent the frame on the left. We explain for each example how we decided about the correctness in the representation of the frame in its respective visual stimulus.

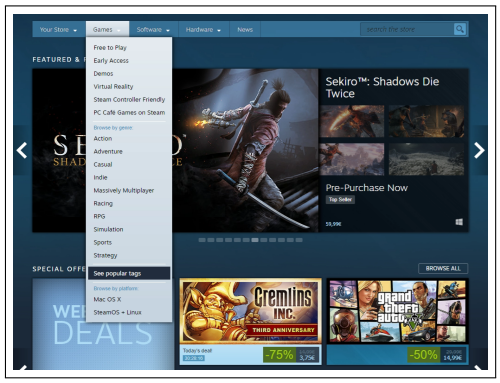


(a) Frame from the video (b) Corresponding region in the visual stimulus

Fig. 27. On the left is frame 136 from the video recording of the first user on Amazon. On the right is the visual stimulus that represents that frame, cropped to the corresponding region. The frame has been marked as correctly represented by the visual stimulus. On the frame from the video recording a few caret symbols are missing. The corresponding part has been magnified and added on top of the respective image. This behaviour falls under *"Area of change?"* as *"Tiny"* in the decision process from Figure 6 on page 19, which signifies no visual change.

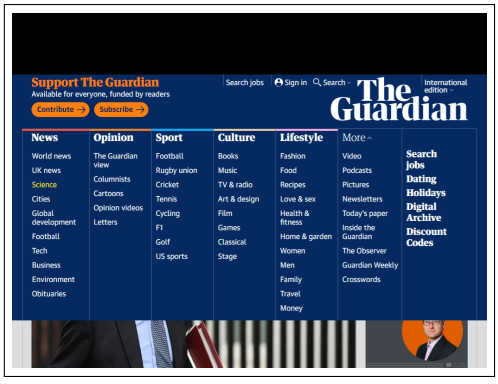


(a) Frame from the video

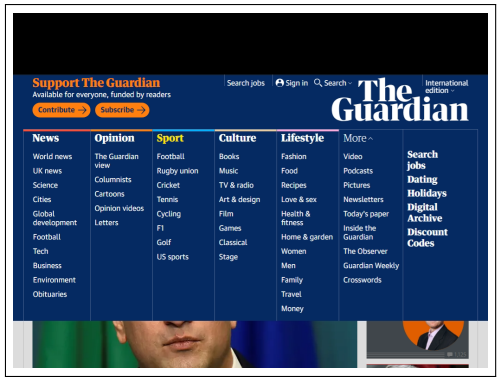


(b) Corresponding region in the visual stimulus

Fig. 28. On the left is frame 587 from the video recording of the second user on Steam. On the right is the visual stimulus that represents that frame, cropped to the corresponding region. The frame has been marked as correctly represented by the visual stimulus. A different sub-menu entry has been highlighted because of a mouse over. This falls under the category of *"Highlight of clickable content?"* in the decision process from Figure 6 on page 19, which signifies no visual change.

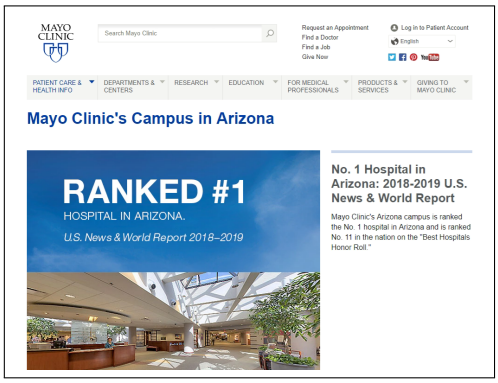


(a) Frame from the video

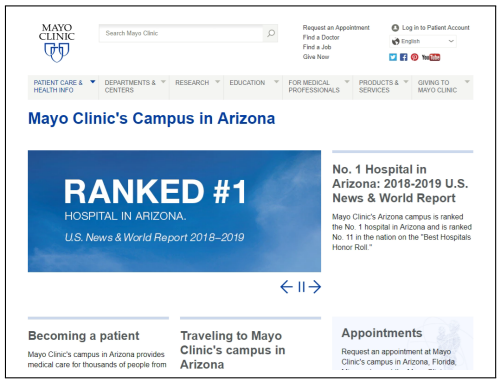


(b) Corresponding region in the visual stimulus

Fig. 29. On the left is frame 174 from the video recording of the first user on Guardian. On the right is the visual stimulus that represents that frame, cropped to the corresponding region. The frame has been marked as not correctly represented by the visual stimulus, because of the different photos behind the menu.

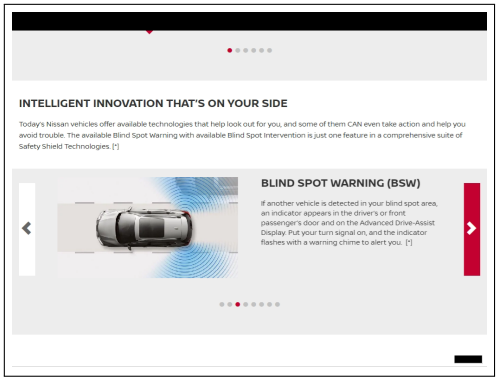


(a) Frame from the video

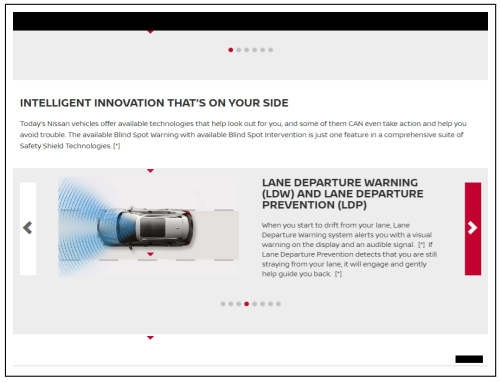


(b) Corresponding region in the visual stimulus

Fig. 30. On the left is frame 287 from the video recording of the first user on MayoClinic. On the right is the visual stimulus that represents that frame, cropped to the corresponding region. The frame has been marked as not correctly represented by the visual stimulus, because of the missing carousel photo in the visual stimulus.

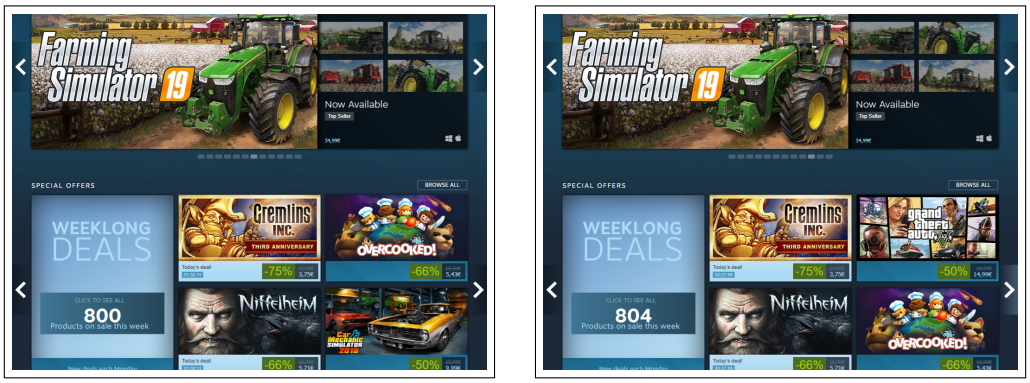


(a) Frame from the video



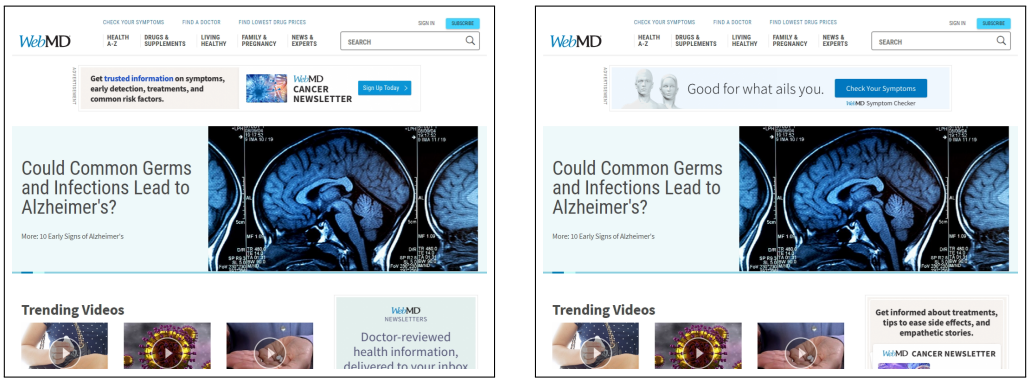
(b) Corresponding region in the visual stimulus

Fig. 31. On the left is frame 494 from the video recording of the first user on Nissan. On the right is the visual stimulus that represents that frame, cropped to the corresponding region. The frame has been marked as not correctly represented by the visual stimulus, because of the different slides in the carousel.



(a) Frame from the video (b) Corresponding region in the visual stimulus

Fig. 32. On the left is frame 83 from the video recording of the first user on Steam. On the right is the visual stimulus that represents that frame, cropped to the corresponding region. The frame has been marked as not correctly represented by the visual stimulus because the game tiles are different.



(a) Frame from the video (b) Corresponding region in the visual stimulus

Fig. 33. On the left is frame 0 from the video recording of the first user on WebMD. On the right is the visual stimulus that represents that frame, cropped to the corresponding region. The frame has been marked as not correctly represented by the visual stimulus, because of the different ad in the lower-right corner.

F EXPERT SURVEY

We have designed a survey to assess the visual stimuli discovery with usability experts. We have published the complete survey including the questions, results and analysis thereof for archival purposes on GitHub.[2] The primary goal of the survey was to integrate video recordings of users browsing the dynamic Web pages and to display the corresponding visual stimuli interactively. In this way, the usability experts could get an insight into the individual analysis tasks and provide us with feedback about how they would perform those tasks and whether the visual stimuli discovery can improve their workflow.

Displaying the video recordings and the visual stimuli in the online survey have been challenging. We did not want to rely on an external hosting service for the video recordings. External hosting services may perform excessive compression of the videos, have issues in streaming performance, and introduce restricted controls over the video playback. Therefore, we have hosted the survey and all data on our Web server. This allowed us to add extended controls over the video recordings. A participant could either control an individual video or use the buttons below the videos to control all videos at once, i. e., to play, pause, reset, or skim the four videos. See Figure 34 for a screenshot about how we presented the video recordings to the usability experts. Besides the video recordings, we have shown the corresponding visual stimuli to the participants. The height of the visual stimuli often exceeds the available screen space, why we could not use a standard picture gallery implementation. Therefore, we have implemented a gallery widget that allows a participant to choose from the generated visual stimuli, control the level of magnification, pan the visual stimulus via drag-and-drop, and toggle an overlay with mouse traces and eye gaze path on and off.

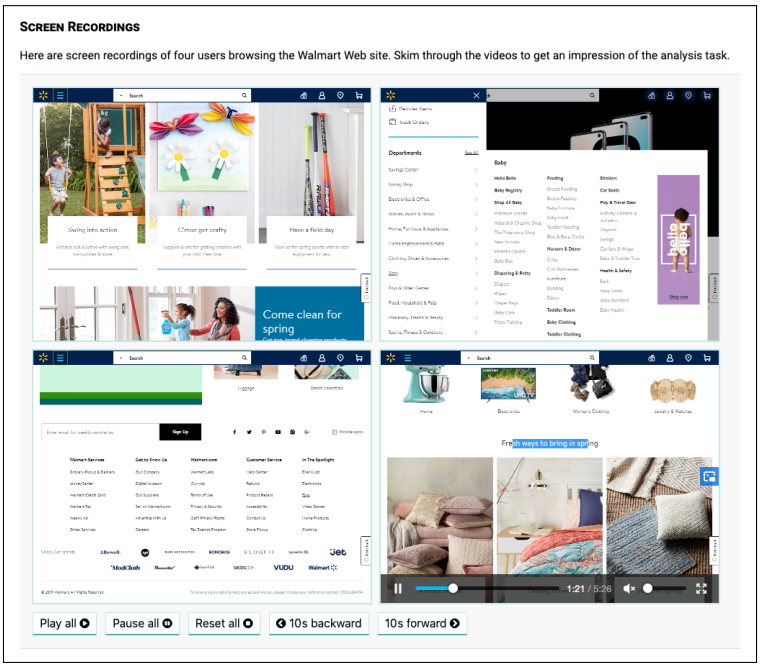


Fig. 34. We display the video recordings as the users have experienced the Web site on the screen. The screenshot shows the display of the four video recordings of the users on the Walmart Web site. A participant could play the video recordings individually or all at once.