SOFTWARE TESTING, VERIFICATION AND RELIABILITY
Softw. Test. Verif. Reliab. 2021;e1756.
Published online in Wiley Online Library (wileyonlinelibrary.com). https://doi.org/10.1002/stvr.1756

SPECIAL ISSUE PAPER

Automated visual classification of DOM-based presentation failure reports for responsive web pages

Ibrahim Althomali[1], Gregory M. Kapfhammer[2] and Phil McMinn[1]

[1] *Department of Computer Science, University of Sheffield, Sheffield, UK*
[2] *Department of Computer Science, Allegheny College, Meadville, PA, United States*

SUMMARY

Since it is common for the users of a web page to access it through a wide variety of devices—including desktops, laptops, tablets and phones—web developers rely on responsive web design (RWD) principles and frameworks to create sites that are useful on all devices. A correctly implemented responsive web page adjusts its layout according to the viewport width of the device in use, thereby ensuring that its design suitably features the content. Since the use of complex RWD frameworks often leads to web pages with hard-to-detect responsive layout failures (RLFs), developers employ testing tools that generate reports of potential RLFs. Since testing tools for responsive web pages, like ReDeCheck, analyse a web page representation called the Document Object Model (DOM), they may inadvertently flag concerns that are not human visible, thereby requiring developers to manually confirm and classify each potential RLF as a true positive (TP), false positive (FP), or non-observable issue (NOI)—a process that is time consuming and error prone. The conference version of this paper presented Viser, a tool that automatically classified three types of RLFs reported by ReDeCheck. Since Viser was not designed to automatically confirm and classify two types of RLFs that ReDeCheck's DOM-based analysis could surface, this paper introduces Verve, a tool that automatically classifies all RLF types reported by ReDeCheck. Along with manipulating the opacity of HTML elements in a web page, as does Viser, the Verve tool also uses histogram-based image comparison to classify RLFs in web pages. Incorporating both the 25 web pages used in prior experiments and 20 new pages not previously considered, this paper's empirical study reveals that Verve's classification of all five types of RLFs frequently agrees with classifications produced manually by humans. The experiments also reveal that Verve took on average about 4 s to classify any of the RLFs among the 469 reported by ReDeCheck. Since this paper demonstrates that classifying an RLF as a TP, FP, or NOI with Verve, a publicly available tool, is less subjective and error prone than the same manual process done by a human web developer, we argue that it is well-suited for supporting the testing of complex responsive web pages.

Received 11 November 2019; Revised 16 June 2020; Accepted 23 August 2020

KEY WORDS: automated web testing; automated layout failure classification; empirical studies; responsive layout failures; responsive web design; web presentation failures

1. INTRODUCTION

Since there is a significant number and variety of web-enabled devices, including phones, tablets, laptops and desktops, web developers no longer maintain a single ‘mobile version’ of a web page alongside a standard desktop version [1], instead opting to fully accommodate all of these devices.

*Correspondence to: Phil McMinn, Department of Computer Science, The University of Sheffield, 211 Portobello, Sheffield, S1 4DP.
†E-mail: p.mcminn@sheffield.ac.uk

Responsive web design (RWD) principles and frameworks allow developers to build web pages that provide an equivalent user experience across varied devices [2]. When a web developer correctly uses an RWD framework, like Bootstrap [3] or Foundation [4], they can create a web page that dynamically adapts its layout by ‘responding’ to the viewport width of the browser running on a device. Rather than requiring users to pan around or zoom in on the content, a web page with a proper responsive design only requires a user to vertically scroll [5], an action that a human can easily complete by, for instance, flicking their finger on a mobile screen or track pad.

Even though RWD principles and frameworks help to address the challenges of dealing with different viewport widths, they can introduce new types of presentational layout failures—referred to as ‘responsive layout failures’ (RLFs) [6]—that can prevent the display of a web page's content. As the viewport width changes, web page elements can start to overlap one another, protrude from their containing elements, disappear off the edge of the viewable portion of the page, wrap incorrectly, or appear with incorrect layout at a few viewport widths. At best, these RLFs make for a poor presentation of the page, leading to lost credibility [7] and decreased user loyalty [8]; at worst, they can lead to critical parts of the web application being inaccessible or unusable [9].

In addition to the ‘Responsive Design Mode’ in the Firefox browser and ‘Device Mode’ in the Chrome browser, other tools exist to help developers check their responsive designs. For instance, viewport resizers (e.g. other studies [10-12]) automatically resize browsers to common viewport widths used by web-enabled devices, conveniently allowing developers to see how their content is rendered. Yet, all of these tools require a human to examine simulated page renderings to manually identify problems, suggesting the need for a tool that can automate this challenging and time-consuming process.

The ReDeCheck tool [13] helps a web developer to identify five representative types of RLFs like, for instance, the situation when web page elements that are separated at one viewport width later appear to collide with one another at a narrower viewport [6]. However, because ReDeCheck analyses the Document Object Model (DOM) representation of a web page [14], the potential RLFs that it detects may not, in practice, be observable to humans. Since other responsive web testing tools, like VFDetector [15], also use the DOM, this concern is not restricted to the use of ReDeCheck, but rather a fundamental limitation of all DOM-based tools. For example, while the bounding boxes of two elements may overlap on a page, their backgrounds may be transparent, and their respective content may thus appear as non-overlapping to a human viewer. The fact that web developers still had to manually confirm and classify the failure reports created by ReDeCheck motivated us to create, as part of the conference version of this paper, the Viser tool [16].

The idea behind both the Viser tool and the follow-on tool introduced in this journal paper is to automatically classify each RLF according to whether it is a *true positive* (TP), *false positive* (FP), or a *non-observable issue* (NOI). [1]Without a tool to perform this classification, a web developer must—in a process that may often be error prone and time consuming—manually classify each RLF reported by ReDeCheck. To save time and reduce subjectivity, our prior conference paper introduced the Viser tool for automatically classifying an RLF into one of these three categories by manipulating the opacity of the HTML elements referenced in ReDeCheck's report of potential RLFs [16]. Since Viser cannot classify responsive layout failures pertaining to either incorrect element wrapping or the appearance of visual disturbances at a small number of viewport widths, this paper introduces ‘Verve’ (Visual classifiEr for ResponsiVe tEsting), a tool that automatically classifies all five types of the potential RLFs detected by ReDeCheck's DOM-based approach. Along with extending Viser's opacity manipulation method to detect element wrapping failures, Verve employs a histogram-based image comparison method that classifies the reported failure involving layout mistakes at a small number of viewport widths. This paper's description and evaluation of Verve is an extension of a conference paper [16] that appeared at the 12th International Conference on Software Testing, Verification and Validation (ICST 2019). As a review, the contributions of the original ICST 2019 conference paper were as follows:

[1] Although the conference version of this paper stated that Viser ‘verified’ the responsive layout failures reported by ReDeCheck, this paper uses the verb ‘classify’ to better describe the behaviour of Viser and this paper's tool, Verve.

A technique, implemented in tool called Viser, that automates a previously manual approach to confirming and classifying three of the five (i.e. *element collision*, *element protrusion* and *viewport protrusion*) responsive layout failures reported by the DOM-based ReDeCheck.

Using 25 web pages, an empirical evaluation that compared the automated results produced by Viser to those arising from a previously published manual analysis, finding that

(a) Viser accurately classified the three types of potential responsive layout failures identified by ReDeCheck, automatically classifying all 117 from prior work.
(b) Viser can automate a time-consuming manual analysis and, importantly, eliminate its subjectivity. Compared to humans, Viser categorized 28 failures differently, including three that were, after a reconsideration, ultimately reclassified as false positives.
(c) Viser is fast to run, requiring no more than a few seconds to complete its automated analysis for three of the types of responsive layout failures reported by ReDeCheck.

While the empirical results from the conference version of this paper were positive—thereby suggesting that further work was warranted—they were fundamentally limited by the fact that Viser did not support the automated confirmation and classification of two types of RLFs reported by ReDeCheck. This journal version expands and deepens the technical details and empirical evaluation of the approach presented in the ICST 2019 paper. The additional contributions that this extended journal article makes over the original ICST 2019 conference paper are as follows:

Implemented in a tool called Verve, that is publicly available in a GitHub repository at github.com/verve-tool/verve, new algorithms for automatically classifying the *wrapping*and *small-range* failures reported by ReDeCheck. Notably, Verve's classifier for the small-range failures uses histogram-based image comparison methods to determine when a web page's layout likely has a human observable problem. Verve also has an enhanced capability to classify *viewport protrusion* in cases where Viser could not effectively do so. This means that Verve can automatically confirm and classify all of the potential responsive layout faults reported by ReDeCheck, making it more generally useful than Viser.

Using the 25 web pages from the study in the aforementioned conference paper, an evaluation of Verve's ability to automatically classify the element collision, element protrusion and viewport protrusion RLFs reported by ReDeCheck, revealing an 86.3% agreement with the previous manual classification. Since Verve's automated classification can take place at different 'inspection points', this paper also empirically determines that the minimum viewport width is the best one for ensuring Verve's agreement with the manual classification.

Again using the 25 web pages from the conference paper's study, an experiment that evaluates Verve's ability to classify the wrapping failures and small-range failures reported by ReDeCheck. The results reveal that, for wrapping failures, Verve agrees with the manual classification 78.6% of the time and, for small-range failures, the level of agreement is 98.5%.

To assess the generalizability of all the RLF classification techniques, as implemented in Verve, this paper also reports on experiments with 143 potential responsive layout failures from 20 new subject web pages. For the element collision, element protrusion and viewport protrusion RLFs and these new subjects, Verve's classification agrees with the manual one 91.8% of the time. While, with the new subjects, Verve's classification of wrapping failures achieved a 64.7% agreement with the manual one; the results for the classification of small-range failures show that Verve matches the manual classification 73.6% of the time.

To assess the overall efficiency of Verve for all 45 subjects and across a total of 469 presentation failures, we recorded the time taken to classify all of the RLF types reported by ReDeCheck, observing that the tool takes about 4 s to perform a classification.

The remainder of this paper is organized in the following fashion. First, Section 2 reviews the principles and practices of responsive web design, highlighting the role that testing tools like ReDeCheck play in the development of high-quality web pages. Section 3 then explains how Verve automatically classifies a responsive layout failure reported by ReDeCheck as being either a true positive, a false positive, or a non-observable issue. In particular, this section shows how Verve manipulates an HTML element's opacity and uses histogram-based image comparison to perform its classification. Section 4 introduces the methodology that we followed to study the efficiency and effectiveness of VERVE, ultimately answering five research questions and demonstrating the promise of the presented approach. Finally, Section 5 summarizes the related work, and Section 6 concludes the paper and suggests new ways to enhance VERVE, further improving the manner in which the tool supports the creation of high-quality responsive web pages.

2. BACKGROUND

The presentation layer of a web application consists of one or more web pages, which are rendered by a web browser on the basis of several resources. A developer first creates a Hypertext Markup Language (HTML) document that specifies the structure of the page's content and involves HTML elements like text, images, multimedia, forms and scripts [14]. Developers associate Cascading Style Sheets (CSS) with an HTML document to specify how a browser should graphically style the HTML elements when rendering the page. Rules in the CSS can style the size and position of elements and can control, for instance, whether the text within them should be rendered in bold face or italic [17]. A browser parses the elements in an HTML document, along with the CSS rules, to form the Document Object Model (DOM) of the page. The DOM is a tree-like data structure that represents the page's visual presentation [14]. A developer can query or modify the page's DOM (and consequently, its visual appearance) through the creation and use of scripts run by the browser. An HTML element's properties, such as its width or height, can be accessed by specifying an eXtensible Markup Language (XML) path expression, known as its XPath [18]. The final arrangement of HTML elements on a page, as rendered by the browser, is called its layout.

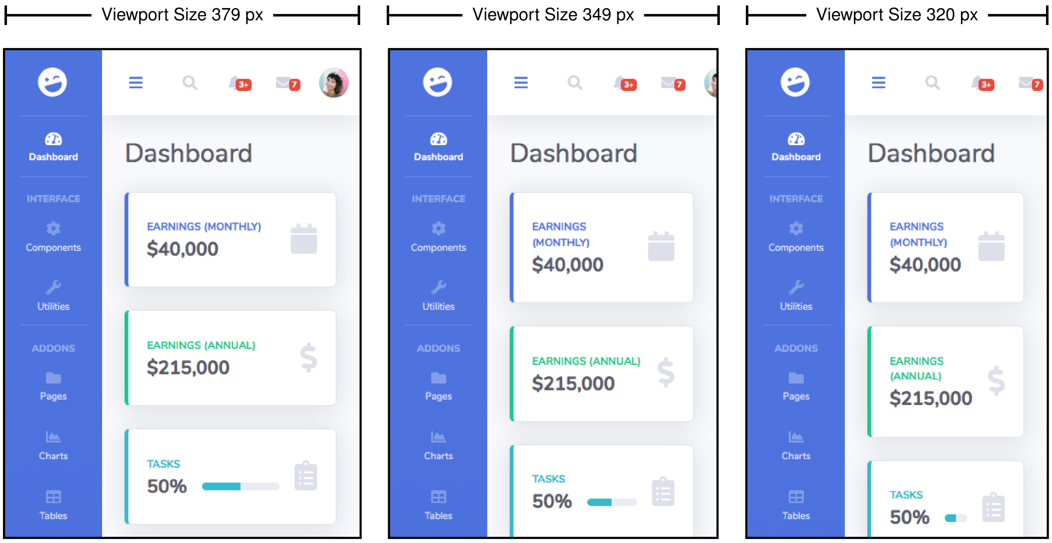
After overviewing the principles and practices of responsive web design, the remainder of this section first explains how testing tools like ReDECHECK automatically detect potential responsive layout failures. Since ReDECHECK works at the DOM level—and cannot ascertain whether the potential layout failure is visually detectable by a human—this paper then highlights the challenges associated with triaging non-observable issues, thereby setting the stage for VERVE, this paper's tool for automatically classifying a responsive layout failure as a TP, FP, or NOI.

2.1. Responsive web design

The responsive web design (RWD) paradigm [2] incorporates the concepts of fluid grids, flexible media and media queries, each of which support the web page design strategies for accommodating a range of viewport sizes. Often supported by frameworks such as Bootstrap [3] and Foundation [4], these concepts are implemented using HTML and CSS. Fluid grids allow the web browser to arrange a page's HTML elements into layouts that smoothly adjust according to the width of the device's viewport. Flexible media refer to images or video content that stretch or shrink in size depending on the screen space available to the browser. Finally, media queries allow developers to activate specific CSS rules depending on the viewport width of the user's device or the width of the web browser. For example, any CSS rules contained within the media query `@media(max-width:767px)` would be enabled if a user's device had a narrow screen width of 767 pixels or less, while `@media(min-width:1200px)` would trigger CSS rules when the page is viewed on the wide screen of a desktop computer that horizontally displays 1200 pixels or more.

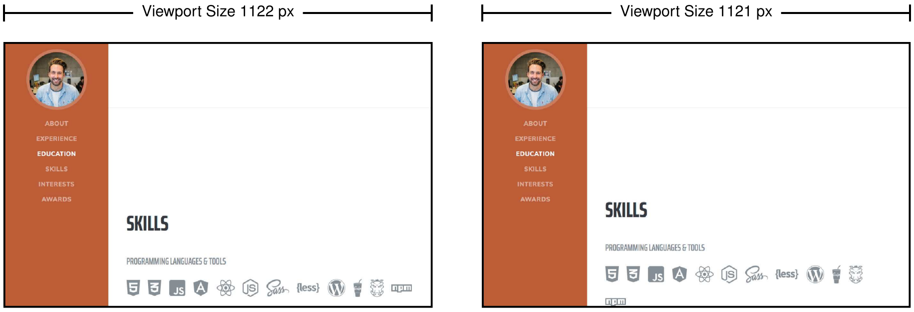
2.2. Testing to detect responsive layout failures

Even with the support of the RWD paradigm and its popular frameworks, web developers may accidentally introduce a wide variety of presentation failures [19]—including responsive layout failures (RLFs) [6]—such as the one shown in Figure 1. At a viewport width of 379 pixels (part (a)), although no problems are apparent, the DOM-based coordinates of the container of the top right profile image are protruding out of the main `body` element. At a narrower viewport width of 349 pixels (part (b)), horizontal space is more constricted, and the profile image is protruding off the right-hand edge of the page. At a smaller viewport width of 320 pixels (part (c)), the profile image is entirely missing



(A) NOI Viewport Protrusion (B) TP Viewport Protrusion (C) TP Viewport Protrusion

Figure 1. Three snapshots of the SB-Admin-2 web page that capture a viewport protrusion failure at the maximum of the reported failure range 320–379 pixels, in (a), and at the middle of the range in (b), and at the minimum of the range in (c), as reported by ReDECHECK and correctly classified by VERVE



(A) No Layout Failure (B) Wrapping Layout Failure

Figure 2. Two snapshots of the SB-Resume web page that capture its layout before a wrapping failure occurs, in (a), and a wrapping failure with the range of 1056–1121 pixels in (b), as reported by the ReDECHECK and correctly classified, without human intervention, as a true positive by the VERVE

from view and no longer accessible. Another example of an RLF is shown in Figure 2. At a viewport width of 1122 pixels (part (a)), the bottom right 12 icons of programming languages and tools are aligned in a row. While at the immediately narrower viewport of 1121 pixels (part (b)), the last icon wraps into a new row and is therefore flagged by ReDECHECK as an RLF.

One technique, implemented into the ReDECHECK tool [6], automatically detects some of the common RLFs in responsively designed web pages, including the following five types of failure. *Element collision* failures occur in a responsive design where the display space is sufficient to accommodate two HTML elements (Figure 3(a)), yet as the viewport becomes narrower, space between the elements tightens until they start to overlap one another (Figure 3(b)). Along with causing unsightly presentational effects, this can lead to the restriction or loss of a web page's functionality if HTML elements, such as important links or buttons, are obscured. *Element protrusion* failures occur when HTML elements 'pop' out of their containers due to reduced display space. At a wide viewport width (Figure 3(c)), the available display space allows the browser to render the element correctly within its container. However, as horizontal display space becomes smaller, the container starts to shrink. The containing element reaches its minimum size, which, for

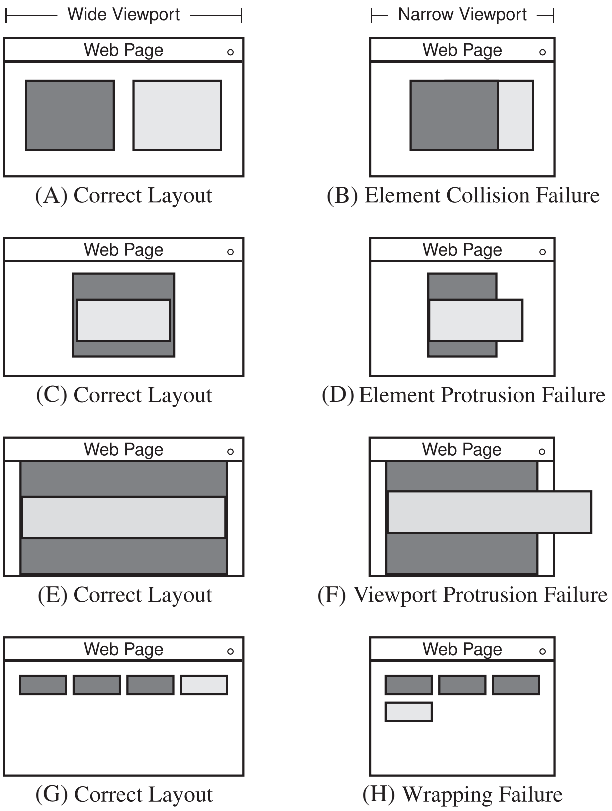


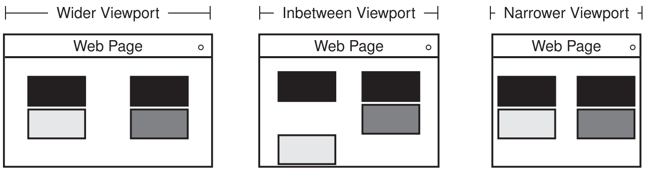
Figure 3. Wireframe examples of the *element collision*, *element protrusion*, *viewport protrusion* and *wrapping* responsive layout failures. The left-hand column furnishes the correct layout, while the right-hand column illustrates the layout failure that becomes evident as the viewport width narrows

instance, may be constrained by the text rendered within it. Eventually, the failure is evident when the containing element protrudes out of its container (Figure 3(d)).

Viewport protrusion is similar to element protrusion, except that an HTML element has protruded out of the viewport itself—that is, it has extended out of the `body` HTML element of the page (Figure 3(e) and 3(f)). Notably, Figure 1 shows real-world examples of viewport protrusion failures.

Wrapping failures occur when HTML elements that are part of a group that are supposed to appear together on a single line (Figure 3(g)), cannot fit together side by side, because the viewport is not wide enough to accommodate all of the elements. Wrapping failures are evident when one or more of the elements ‘wrap’ to form an additional line, separating them from the other elements (Figure 3(h)). Figure 2 illustrates a real-world example of a wrapping failure.

Finally, *small-range* failures are layouts that anomalously occur for only a small number of consecutive viewport widths. In contrast to the aforementioned RLFs, they do not necessarily occur due to the amount of available space for laying out elements tightening from a wider to a narrower viewport. Instead, they are often caused by mistakes in the CSS related to media queries. For instance, the media query ‘`@media (max-width: 768px) {...`’, and further media query as ‘`@media (min-width: 768px) {...`’ may be encoded by a developer. However, since the viewport ranges defined by both of these expressions are inclusive, both will be activated at the 768 pixel viewport width, potentially leading to strange layout effects. This is because two sets of rules will be activated when only one set was intended [20]. These types of failures are difficult for developers to spot, as they occur only in small sub-range(s) of the entire range of viewport widths in which the page may be viewed. Figure 4 gives an example of this: for the majority of viewport widths (i.e. parts (a) and (c)), four web page elements are displayed as a 2×2 grid. Yet, for a small number of viewports (i.e. part (b)), the bottom left element falls out of alignment with the others.



(A) Correct Layout (B) Small-Range Failure (C) Correct Layout

Figure 4. A wireframe example of a *small-range* responsive layout failure

2.3. *The ReDeCheck tool for testing responsive web pages*

The ReDeCheck (REsponsive DEsign CHECKer, pronounced 'Ready Check') tool, when run in 'common failure checking' mode, detects these RLFs by extracting a 'Responsive Layout Graph' (RLG) [13]. An RLG is a model of the responsive layout behaviour of a web page [21]. It represents, at different viewport widths, both the relative alignment of HTML elements with respect to one another (e.g. 'above', 'below' and 'contained') and which HTML elements are, and are not, set to be visible at each width. When constructing an RLG, ReDeCheck collates information by driving a desktop browser and rendering a web page at different viewport widths in a specified range. This viewport range typically starts with a narrow width, 320 pixels, akin to a mobile phone, and extends to a more spacious width of 1400 pixels, a viewport width corresponding to a web browser in use on a desktop computer. ReDeCheck extracts the DOM of the web page rendered at each viewport width and uses it to find the relative alignment of HTML elements when constructing the RLG.

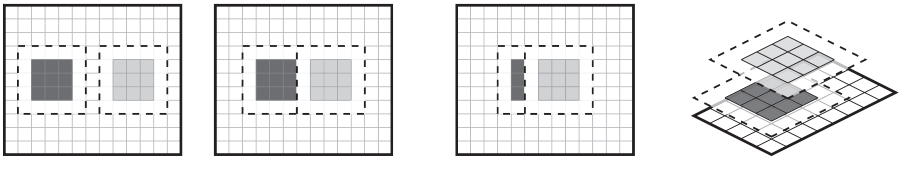
ReDeCheck uses the RLG to find potential layout failures, such as those involving element collisions, by checking for pairs of elements that were not overlapping at a particular viewport width, but then overlap at a narrower width [6]. Intuitively, ReDeCheck uses the layout at wider viewports to cross-check narrower widths. If pairs of elements were not overlapping or protruding at a particular viewport width but then do so as the viewport narrows, an RLF is likely to have manifested. This type of checking across viewport widths makes ReDeCheck less likely to report false negatives than if a developer was to use, for example, the Fighting Layout Bugs tool [22], which reports anomalies at a single viewport width. Tools of this nature report anomalies, such as overlapping elements, at a single viewport width, where they are more likely to be layout behaviour that a web developer intended-and not something for which an alert is warranted.

When ReDeCheck finds an RLF, it produces a report that states (i) the failure type (e.g. element collision or element protrusion); (ii) the viewport range of the RLF (i.e. the minimum and maximum viewport width for which the RLF was evident) and finally (iii) the XPaths of the HTML elements involved [13]. The HTML elements involved for protrusion failures are the protruding element and its container; while for collision failures, ReDeCheck reports the colliding elements. With wrapping failures, ReDeCheck reports the wrapped element along with the row elements with which it was originally aligned. Finally, small-range failures are characterized by some temporary and anomalous relative alignment of elements that suspiciously occurs at a few viewport widths. Here, the ReDeCheck tool reports the two elements for which relative alignment has changed. For the example in Figure 4, the report would highlight the light grey and dark grey HTML elements.

The next subsection summarizes results from prior empirical studies of ReDeCheck, pointing out that even though the automated tool improves the testing process, it may highlight certain responsive layout issues that, in practice, web developers do not normally focus on first.

2.4. *False positives and non-observable issues with DOM-based failure reports*

In a prior empirical study, ReDeCheck found RLFs in 16 of 26 web pages studied [6]. However, it produced 328 failure reports, of which only 197 were found to be true positives. This means that the reports output by ReDeCheck must be manually reviewed to see if the failure needs to be fixed. Importantly, this manual review process is time consuming and error prone, suggesting the need for an automated approach, like the Verve tool presented in this paper, that can efficiently and effectively determine if a potential RLF is a true positive, false positive, or non-observable issue.



(A) No Collision (B) Non-Observable Collision (C) Observable Collision (D) Layered View of Collision

Figure 5. The wireframes of two HTML elements, shown in light and dark grey, with a white border that is the same colour as the background of the page (part (a)). Parts (b) and (c) respectively depict elements with a non-observable collision and an observable collision. Finally, part (d) helps illustrate how VERVE manipulates opacity to perform automatic visual classification of a responsive layout failure in a web page

A particular problem experienced during the use of ReDECHECK is that, since its analysis is based on the DOM of the web pages it checks— an abstract representation of a web page—it cannot distinguish between issues that are observable in practice from those that are not. This particular class of reports are referred to by Walsh et al. [6] as 'non-observable issues'. Figure 5 highlights this problem, depicting two HTML elements in light and dark grey, with a white border that is the same colour as the background of the web page, as shown in part (a). Parts (b) and (c) reveal *non-observable* and *observable* collisions, respectively. In part (b), the two elements are technically colliding, but a person testing this web page is unlikely to see this as a problem because only the borders of the elements are overlapping—and they are the same colour as the background.

Figure 5(c) shows how an observable issue arises as a result of the dark grey element's content becoming obscured by that of the light grey one. As it does not take into account visual details beyond the size and coordinates of the elements concerned, ReDECHECK cannot distinguish between the two scenarios in parts (b) and (c) and so it reports them both. While the non-observable issues exemplified by part (b) of Figure 5 may be of interest to testers—as they are latent issues that could manifest in visible failures in different contexts—these RLFs are unlikely to be a high priority compared to the actual visual defect in part (c) of this figure. Yet, ReDECHECK offers no way to distinguish non-observable issues from true positives, thereby limiting its effectiveness.

The aim of VERVE, introduced in the next section, is to solve this problem by performing automated visual analysis of a web page with respect to failure reports, relieving a developer of having to manually complete this task, thereby making the process less time consuming and error prone. These two web testing tools work together, with ReDECHECK analysing a web page's DOM to detect potential RLFs and VERVE classifying each RLF as a true positive, false positive or non-observable issue, thereby ensuring that a web developer can focus on the most critical failures.

3. AUTOMATED VISUAL ANALYSIS TO CLASSIFY DOM-BASED FAILURE REPORTS

Since the isolated use of DOM-based tools, like ReDECHECK [13] and VFDetector [15], may report responsive layout failures that are either false positives or non-observable issues, this paper presents VERVE as a companion tool that can address this limitation. Figure 6 shows VERVE's approach to automatic visual classification of ReDECHECK's failure reports, comparing it with the series of manual steps that would otherwise be required. In comparison to the VISER tool presented in the conference version of this paper [16], VERVE is more general-purpose since it can automatically classify all five types of potential RLFs that ReDECHECK identifies.

After a developer runs ReDECHECK on a web page, it reports the RLFs that it detects, if any. Each report states the RLF type (e.g. element protrusion), the range of viewport widths for which the RLF was deemed to occur (in the form of a lower and an upper bound), and the XPaths of the HTML elements involved. If VERVE is not used, the developer must manually decide what to do with these reports. As the figure also shows, this involves loading up the web page; setting the viewport width of the browser to one within the reported range; manually identifying the elements and scrolling to the failure if necessary; and finally deciding if the RLF is a true positive or not.

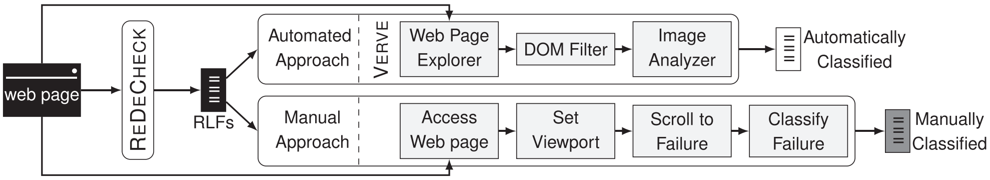


Figure 6. A high-level overview of the VERVE pipeline for the automatic visual classification of DOM-based layout failure reports produced by ReDeCheck. Along with the external input sources, this figure also shows a manual approach to confirmation and classification that requires a human web developer

As depicted in Figure 6, the VERVE tool automates these steps. The *web page explorer* opens the browser, sets the viewport width and locates the faulty elements. It first cross-checks ReDeCheck's result by examining the DOM in the *DOM filter* step. VERVE checks that each element reported has a physical size (i.e. its width and height are not zero) and they can be reached, if not initially present, by scrolling the web page. Specifically, VERVE checks the coordinates of the bounding box of each element. Negative coordinates are inherently unreachable by scrolling, while coordinates greater than the maximum position that can be scrolled to are also deemed to be off the page. The tool VERVE makes further checks at this stage specific to the different failure types discussed in the following sections. Any RLFs failing these inspections are reported as false positives. Otherwise, VERVE proceeds to the *Image Analyser* component for visual analysis of the failure.

The *image analyser* investigates specific regions of a web page, which we refer to as *areas of concern*(AOCs). AOCs are specific to each type of RLF. An AOC bounds a rectangle of the page pertaining to the elements involved in a layout failure where its graphical presence is suspected to have inadvertently overwritten other graphics or content on the page, or to have been written to the page out of position. The *image analyser* then attempts to determine if this is the case (i.e. the RLF produces visible, observable effects). For example, if the misplaced element has no content and is transparent, the RLF will not be detectable by a human using the web page and so a failure report produced by ReDeCheck will likely be of little concern to the page's developers.

The remainder of this section describes how VERVE identifies AOCs for different types of RLF and how the image analysis uses them to automatically classify the failure reports.

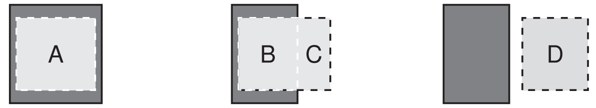
3.1. Element collision, element protrusion and viewport protrusion failures

Element collision, element protrusion and viewport protrusion failures all involve overlaps of two HTML elements, and as such, VERVE's image analysis follows a similar algorithmic process for classifying them. For protrusion failures, the *DOM filter* component of VERVE makes the additional check that the width and height of the protrusion are not zero (i.e. that the ‘protruding’ element is actually outside the bounds of its container). [2]

If this is the case, VERVE reports a false positive, else it proceeds to the *image analyser* phase. The DOM checks for collision failures verifying that the DOM coordinates of the reported elements do overlap. In the image analysis phase, VERVE identifies the AOCs relevant to the failure and tries to discern whether the failure is visible or not.

3.1.1. Identifying areas of concern. Figure 7 summarizes the ways in which two HTML elements can be arranged spatially with respect to one another. The two elements are depicted by dark grey and light grey boxes, respectively. The figure identifies three particular scenarios: ‘Contained’, where one element resides inside the bounds of another; ‘Overlapped’, where the two elements share some, but not all, of the same display space; and finally ‘Detached’, where the two elements are set completely apart from one another. The figure then shows how AOCs are determined for each type

[2] This can happen in practice due to discrepancies between the DOM extracted by ReDeCheck and VERVE since ReDeCheck uses JavaScript injected into the page to extract the DOM, while VERVE uses Selenium. We report on instances of this in our evaluation (Section 4.5), specifically in the answers to RQ1(a) and (b) given on pages 23 and 25.



	Layout Scenario		
	Contained	Overlapped	Detached
Element Collision	A	B	-
Element Protrusion	-	B, C	D
Viewport Protrusion	-	B, C	D

Figure 7. Identifying ‘areas of concern’ (AOCs) for the *element collision*, *element protrusion* and *viewport protrusion* RLFs and the three layout scenarios involving two distinct HTML elements, as depicted by the light grey and dark grey boxes. This figure uses a capital letter for an AOC in a specific RLF and scenario

of RLF with respect to each scenario. For *element collision*, the AOC is the portion of the secondary (light grey) element that is contained within the first (A in the fully contained scenario or B in the overlapped scenario). For *element protrusion*, there are two potential AOCs. The first is the overlapped portion (B), if it exists; and the second, the non-overlapping portion (C in the overlapped scenario, D in the detached scenario). The two portions are treated as separate AOCs to simplify the image analysis, which needs to take into account the fact that the foreground element is overlaid on different background elements. The same is true for *viewport protrusion*, except for that, in this case, the dark grey background element corresponds to the `body` element of the web page, which is the basic container for all web page elements.

3.1.2. Classification algorithm. Once the method detailed in the last subsection has identified an AOC, the image analysis tries to determine whether or not the HTML elements involved in the RLF—which are often stacked on top of one another—render different content in the same space or out of position. The approach works to ‘reveal’ the different layers of the AOC by removing the HTML elements concerned from the top level down to the background, systematically removing each element involved in the failure. (As an example, Figure 5(d) showed the stacking of elements involved in an element collision and the different ‘layers’ that are involved.) VERVE takes a snapshot image of the AOC at each layer and then compares them for differences. If there are any, then VERVE classifies the RLF as being visible (i.e. it is a true positive). If there are no differences, then it should not be a priority for developers and so the tool classifies the RLF as non-observable.

VERVE accesses different graphical layers in the display space by manipulating the `opacity` CSS property of HTML elements, thereby making it and its descendants invisible. We decided to use `opacity`, as opposed to removing elements completely, because removing elements can impact the layout of the remaining HTML elements on the page [17], which would potentially interfere with the classification of the RLF. By instead manipulating the element's opacity, the element is still ‘there’ as far as the layout is concerned, but the elements stacked below it are revealed for the purposes of taking a snapshot. This method also has the advantage of being browser independent.

A technical inconvenience arises when the AOC is larger than the portion of the web page currently viewable, due to the viewport size corresponding to the RLF, a situation that is common with *viewport protrusion* failures. Since the page's responsive design is likely to dictate that the failure no longer occurs, the viewport size cannot be increased to bring these elements back into view for snapshotting. In this scenario, VERVE scrolls the web page, taking snapshots of individual portions of the page and then ‘sewing’ the AOC together as necessary.

Nevertheless, it is not always possible to scroll and bring protruding elements into view. In these circumstances, VERVE performs what we refer to as a ‘best-effort’ approximation of the AOC by changing the page and altering the CSS properties of the offending elements in an attempt to ‘pull’ it into the visible portion of the page so that a snapshot can be taken. The previous version of this tool, VISER, was moderately successful at this through altering the `margin-left` property of the offending element [16].

However, some off-screen elements stubbornly refused to move with this method. We analysed these situations and made the best-effort analysis in VERVE more robust by (i) adjusting further

CSS properties, such as `margin-right`, if `margin-left` failed and (ii) labelling the modified property with the '`!important`' rule, so that VERVE's change could not be overridden later in the page's style sheet. This paper evaluates our improved best-effort analysis to these viewport protrusion failure scenarios as part of the empirical evaluation in Section 4.

Algorithm 1 furnishes the top-level algorithm for classifying collisions and protrusions. This is the image analysis component of the VERVE tool, as shown by Figure 6, for these particular types of RLFs. This component of VERVE finds the initial AOC to analyse, identified according to the principles illustrated in Figure 7. Further analysis is then performed by one or both of Algorithms 2 and 3, depending on the layout scenario. If one element contains the other, as with the *contained* scenario of Figure 7, or part of the other, as with the *overlapped* scenario, control passes to Algorithm 2. Depending on the scenario, the AOC is either A or B, as shown by Figure 7.

Algorithm 1 Classification of element collision and element/viewport protrusion failures

INPUT: Two HTML elements, $back$ and $front$, and the failure type, ft.
OUTPUT: TP if the RLF is deemed observable, NOI if it is not.

```
procedure CLASSIFYCOLLISIONSANDPROTRUSIONFAILURES($back, front, ft$)
    $scenario \leftarrow$ GETSCENARIO($back, front$)
    if $scenario$ = contained then
        $AOC \leftarrow$ GETCONTAINEDAOC($back, front$)                ▷ $AOC$ = A (Figure 7)
        return CONTAINEDAOCIMAGEANALYSIS($back, front, ft, AOC$)
    if $scenario$ = overlapped then
        $AOC \leftarrow$ GETCONTAINEDAOC($back, front$)                ▷ $AOC$ = B (Figure 7)
        return CONTAINEDAOCIMAGEANALYSIS($back, front, ft, AOC$)
    if $scenario$ = detached then
        $AOC \leftarrow$ GETDETACHEDAOC($back, front$)                 ▷ $AOC$ = D (Figure 7)
        return DETACHEDAOCIMAGEANALYSIS($front, AOC$)
```

Algorithm 2 Image analysis for contained AOCs

INPUT: Two HTML elements, $back$ and $front$, the failure type, ft, the AOC, AOC.
OUTPUT: TP if the RLF is deemed observable, NOI if it is not.

```
procedure CONTAINEDAOCIMAGEANALYSIS($back, front, ft, AOC$)
    $back \leftarrow$ MAKETRANSPARENT($back$)
    $front \leftarrow$ MAKETRANSPARENT($front$)
    $imgNoElements \leftarrow$ SNAPSHOT($AOC$)
    $back \leftarrow$ RESTORE($back$)
    $imgBack \leftarrow$ SNAPSHOT($AOC$)
    $front \leftarrow$ RESTORE($front$)
    $imgFront \leftarrow$ SNAPSHOT($AOC$)
    if $imgNoElements \neq imgBack \wedge imgNoElements \neq imgFront$ then
        if $ft$ = element_collision then
            return TP
        if $ft$ = element_protrusion $\vee$ $ft$ = viewport_protrusion then
            $AOC \leftarrow$ GETDETACHEDAOC($back, front$)             ▷ $AOC$ =C (Figure 7)
            return DETACHEDAOCIMAGEANALYSIS($front, AOC$)
    return NOI
```

Algorithm 3 Image analysis for detached AOCs

INPUT: An HTML element, $element$, and the AOC, AOC.
OUTPUT: TP if the RLF is deemed observable, NOI if it is not.

```
procedure DETACHEDAOCIMAGEANALYSIS($element, AOC$)
    $element \leftarrow$ MAKETRANSPARENT($element$)
    $imgNoElement \leftarrow$ SNAPSHOT($AOC$)
    $element \leftarrow$ RESTORE($element$)
    $imgElement \leftarrow$ SNAPSHOT($AOC$)
    if $imgNoElement \neq imgElement$ then
        return TP
    return NOI
```

Algorithm 2 takes the two HTML elements involved (i.e. the dark grey and light grey elements of Figure 7) and sets their opacity to 0%, ensuring that they are transparent via a call to MakeTRANSPARENT. This procedure modifies the opacity of the element dynamically using JavaScript injected into the page, that is, without having to reload it each time, a change is made. Three snapshots are then taken, first of the background (where both elements are transparent), which is saved in *imgNoElements*. Then, restoring *back* (i.e. the dark grey element) to its original opacity

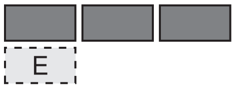


Figure 8. The area of concern (the element marked ‘E’) for a wrapping failure

level (using the Restore procedure), a further snapshot, *imgBack*, is taken. Finally, the foreground element is restored, and the algorithm takes another snapshot of the AOC and saves it to *imgFront*. These three images are then compared in line 9 of the algorithm. If there are differences and the failure type is an *element collision*, then the RLF is deemed to be visible, and the algorithm returns a true positive (TP), else the verdict is a non-observable issue (NOI). If there are differences for the other two failure types (i.e. *element protrusion* and *viewport protrusion*), then the AOC, when detached from the background element, must be analysed to see if the content has spilled outside its containing element. In this case, control passes to Algorithm 3 to handle the case of a detached area of concern, as the AOC is now known to be C, as per Figure 7.

In the detached scenario, Algorithm 3 may also be invoked directly from Algorithm 1, with the AOC being identified as D, as per Figure 7. The algorithm proceeds in a similar fashion to that of Algorithm 2, except there are only two layers to consider: that with the foreground element (i.e. the light grey element of Figure 7) present and that where it is transparent. The two snapshots are compared. If the images are different, then the algorithm returns a true positive (i.e. the RLF is visible), else the non-difference between the layers means that the RLF is non-observable.

As stated in Section 2, ReDeCheck reports, for each RLF, a viewport range that is the narrowest to the widest viewport width for which the RLF is manifested at the DOM level. Since Verve has a choice of the viewport at which it can visually inspect the RLF, we made this a configurable parameter of the tool. The default is to look at the narrowest viewport width (i.e. the ‘low end’ of the range) since RLFs are more likely to be noticeable at screen sizes with tighter layout constraints than at wider viewports that are less constrained for space. Since this assumption about the viewport inspection point may not generally hold, we investigate this parameter's configuration in Section 4.

3.2. *Wrapping failures*

For wrapping failures, the *DOM filter* phase of Verve involves an additional check of the DOM to determine if the wrapped element has vertical coordinates indicating that it is below the right-most element of the row from which ReDeCheck deems it to have wrapped. If this is not the case, then Verve reports a false positive, else Verve proceeds to the image analysis stage.

The *image analyser* component of Verve treats wrapping failures in the same way as the ‘detached’ case of Figure 7. The AOC is the area of the wrapped element as shown in Figure 8. The algorithm for wrapping failures (Algorithm 4) takes this AOC and passes it to the DetachedAOCImageAnalysis algorithm (Algorithm 3), which in turn will detect whether the wrapped element is visible or not (returning TP or NOI, respectively). For this type of RLF, cases where the wrapped element is non-visible correspond to those where the developer has inserted an invisible ‘spacer’ element intended to provide some vertical padding.

Algorithm 4 Classification of wrapping failures

INPUT: The wrapped element, $wrapped$.
OUTPUT: TP if the RLF is deemed observable, NOI if it is not.

```
procedure ClassifyWrappingFailures($wrapped$)
    $AOC \leftarrow$ getWrappingAOC($wrapped$)                ▷ $AOC$ = E (Figure 8)
    return DetachedAOCImageAnalysis($wrapped, AOC$)
```

3.3. *Small-range failures*

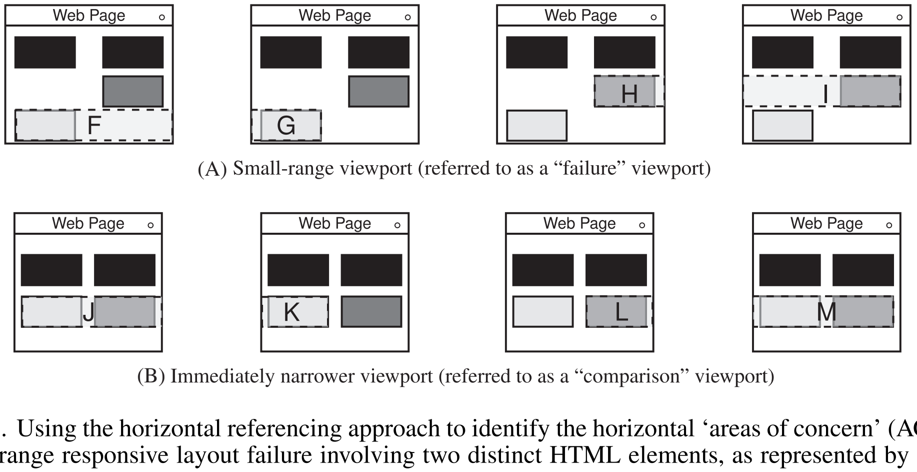
As noted in Section 2, small-range failures are altogether different from the other failure types. They are not caused as a result of tightening space, but rather mistakes in the CSS code related to media queries (e.g. off-by-one mistakes resulting in a media query being activated for a viewport width erroneously) or other emergent, intermittent effects resulting from the interaction of element CSS properties that conspire to make the layout look anomalous for particular viewport widths.

Because the visual effects that they cause are a result of general element misalignments whose positions are difficult to characterize in advance, small-range failures require a different treatment than that used for the previously discussed RLF types. VERVE therefore takes a different approach in which it attempts to measure the level of visual disturbance. If the visual difference is above some threshold for the small range, compared to the snapshots of the web page at viewport widths either side of the small range, VERVE flags the RLF as a true positive. If the visual difference is negligible (i.e. under experimentally determined thresholds), then VERVE flags the RLF as a false positive.

3.3.1. Identifying areas of concern for small-range failures.

As for other failure types, VERVE identifies regions of the web page (i.e. the areas of concern) for small-range failures, subjecting them to particular scrutiny. The difference with other failures is that VERVE not only identifies AOCs on the basis of the viewport containing the failure, but also on the viewports (i.e. with narrower and wider widths) at either side of the small range of viewports where the failure has been identified to occur by ReDECHECK. We refer to the viewports on either side of the failure as ‘comparison’ viewports, and any viewport in the small range where the failure is deemed to occur as a ‘failure viewport’. VERVE contrasts AOCs from the comparison viewports with their counterparts in a failure viewport and tries to automatically ascertain if there are visual differences.

The ReDECHECK tool reports small-range failures as pairs of elements whose relative alignment has changed for a small number of viewport widths (e.g. in Figure 4(b), the light grey element moves from the left of the dark grey element to its bottom left). Figure 9 depicts the scenario of Figure 4(b) with AOCs labelled. We present the small-range failure (part (a) of the figures, denoted the *failure viewport*) alongside the narrower viewport (part (b), denoted the *comparison viewport*) where the anomalous alignment is no longer evident. There are different ways in which AOCs can be obtained to try and capture the region of the visual disturbance as a result of the failure. The ‘horizontal referencing’ approach, illustrated by Figure 9, forms AOCs using the coordinates of the bounding boxes of each reported element at each viewport width, extended out horizontally either right or left to the edge of the page (e.g. F and G). This gives two AOCs for each element, four AOCs for each viewport, and a total of 12 AOCs for all three viewports. An alternative, illustrated by Figure 10, is to form the AOC from the snapshot by extending out from the reported element vertically to the top or bottom of the page (e.g. N and O). We implemented both the ‘horizontal referencing’ and a combined ‘horizontal-plus-vertical referencing’ approach into the VERVE tool and evaluate them in Section 4. The classification algorithm, described next, compares respective AOCs with one another using automated colour histogram analysis.

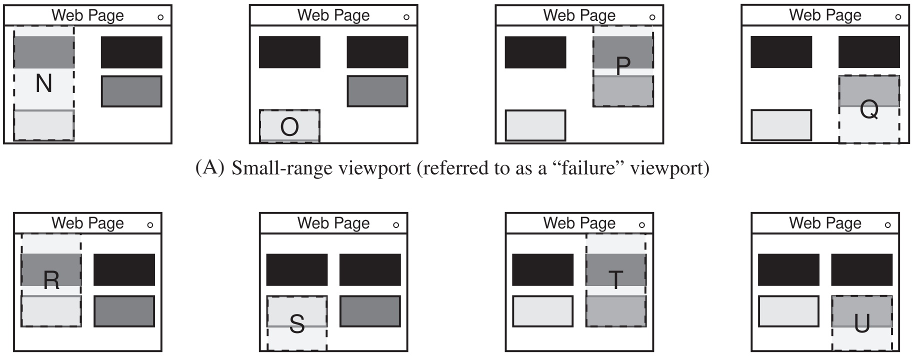


(A) Small-range viewport (referred to as a “failure” viewport)

(B) Immediately narrower viewport (referred to as a “comparison” viewport)

Figure 9. Using the horizontal referencing approach to identify the horizontal ‘areas of concern’ (AOCs) for a small-range responsive layout failure involving two distinct HTML elements, as represented by the light grey and dark grey boxes

DOI: 10.1002/stvr



(A) Small-range viewport (referred to as a “failure” viewport)

(B) Immediately narrower viewport (referred to as a “comparison” viewport)

Figure 10. Using the vertical referencing approach to identify the vertical ‘areas of concern’ (AOCs) for a small-range responsive layout failure involving two distinct HTML elements, represented by the light grey and dark grey boxes

3.3.2. Classification algorithm for small-range failures. The classification algorithm for small-range AOCs compares pairs of colour histograms for corresponding AOCs from a failure viewport and a comparison viewport (e.g. F and J from Figure 9). Algorithm 5 shows the steps for the combined ‘horizontal-plus-vertical referencing’ approach for identifying AOCs. The algorithm with ‘horizontal referencing’ AOC identification is the same but omits the steps involving identifying and using ‘vertical’ AOCs (e.g. lines 7–11 and 17–21).

Algorithm 5 Classification of small-range failures

INPUT: Two HTML elements, $element1$ and $element2$, involved in the failure; the failure viewport $failureVP$; the narrower comparison viewport $narrVP$; the wider comparison viewport $widerVP$; and finally, the colour histogram distance metric $metric$.
OUTPUT: TP if the RLF is deemed observable, FP otherwise.

```
 1: procedure CLASSIFYSMALLRANGEFAILURES($element1, element2, failureVP, narrVP, widerVP, metric$)
 2:     ▷ Get horizontal AOCs from failure viewport
 3:     $element1RightFailureAOC \leftarrow$ GETRIGHTAOC($element1, failureVP$)   ▷ $AOC =$ F (Figure 9)
 4:     $element1LeftFailureAOC \leftarrow$ GETLEFTAOC($element1, failureVP$)   ▷ $AOC =$ G (Figure 9)
 5:     $element2RightFailureAOC \leftarrow$ GETRIGHTAOC($element2, failureVP$)   ▷ $AOC =$ H (Figure 9)
 6:     $element2LeftFailureAOC \leftarrow$ GETLEFTAOC($element2, failureVP$)   ▷ $AOC =$ I (Figure 9)
 7:     ▷ Get vertical AOCs from failure viewport
 8:     $element1TopFailureAOC \leftarrow$ GETTOPAOC($element1, failureVP$)   ▷ $AOC =$ N (Figure 10)
 9:     $element1BottomFailureAOC \leftarrow$ GETBOTTOMAOC($element1, failureVP$)   ▷ $AOC =$ O (Figure 10)
10:     $element2TopFailureAOC \leftarrow$ GETTOPAOC($element2, failureVP$)   ▷ $AOC =$ P (Figure 10)
11:     $element2BottomFailureAOC \leftarrow$ GETBOTTOMAOC($element2, failureVP$)   ▷ $AOC =$ Q (Figure 10)
12:     ▷ Get horizontal AOCs from narrower (comparison) viewport
13:     $element1RightNarrAOC \leftarrow$ GETRIGHTAOC($element1, narrVP$)   ▷ $AOC =$ J (Figure 9)
14:     $element1LeftNarrAOC \leftarrow$ GETLEFTAOC($element1, narrVP$)   ▷ $AOC =$ K (Figure 9)
15:     $element2RightNarrAOC \leftarrow$ GETRIGHTAOC($element2, narrVP$)   ▷ $AOC =$ L (Figure 9)
16:     $element2LeftNarrAOC \leftarrow$ GETLEFTAOC($element2, narrVP$)   ▷ $AOC =$ M (Figure 9)
17:     ▷ Get vertical AOCs from narrower (comparison) viewport
18:     $element1TopNarrAOC \leftarrow$ GETTOPAOC($element1, narrVP$)   ▷ $AOC =$ R (Figure 10)
19:     $element1BottomNarrAOC \leftarrow$ GETBOTTOMAOC($element1, narrVP$)   ▷ $AOC =$ S (Figure 10)
20:     $element2TopNarrAOC \leftarrow$ GETTOPAOC($element2, narrVP$)   ▷ $AOC =$ T (Figure 10)
21:     $element2BottomNarrAOC \leftarrow$ GETBOTTOMAOC($element2, narrVP$)   ▷ $AOC =$ U (Figure 10)
22:     ▷ Get vertical and horizontal AOCs from wider (comparison) viewport
23:     $element1RightWiderAOC \leftarrow$ GETRIGHTAOC($element1, widerVP$)
24:     ...
25:     ▷ Match up the AOCs for colour histogram analysis
26:     $setsOfAOCs \leftarrow \{(element1LeftFailureAOC, element1LeftNarrAOC, element1LeftWiderAOC),$
27:     $(element1RightFailureAOC, element1RightNarrAOC, element1RightWiderAOC),$
28:     $\ldots\}$
29:     ▷ Compare the colour histograms for each set of AOCs
30:     for all ($failureAOC, narrAOC, widerAOC$) $\in setsOfAOCs$ do
31:         $histogramForFailureAOC \leftarrow$ GETCOLORHISTOGRAM(SNAPSHOT($failureAOC, failureVP$))
32:         $histogramForNarrAOC \leftarrow$ GETCOLORHISTOGRAM(SNAPSHOT($narrAOC, narrVP$))
33:         $histogramForWiderAOC \leftarrow$ GETCOLORHISTOGRAM(SNAPSHOT($widerAOC, widerVP$))
34:         $distanceNarrower \leftarrow$ GETDISTANCE($metric, histogramForFailureAOC, histogramForNarrAOC$)
35:         $distanceWider \leftarrow$ GETDISTANCE($metric, histogramForFailureAOC, histogramForWiderAOC$)
36:         $distance \leftarrow$ MIN($distanceNarrower, distanceWider$)
37:         if SATISFYTHRESHOLD($metric, distance$) then
38:             return TP
39:     return FP
```

The first part of the algorithm involves determining the AOCs (lines 2–24) and matching them up (lines 25–28) into sets across the three viewports involved. The comparison of these AOCs takes place in a loop over each set (lines 29–38). The algorithm obtains the image of each AOC, in the matched set—taken from a snapshot of the page at the appropriate viewport—and computes its colour histogram, a histogram of the number of pixels with a certain colour in the image (lines 31 and 33). The algorithm then uses a histogram metric to compute a 'distance' between the colour histogram of the image from the failure viewport against that of a narrower or wider comparison viewport (lines 34 and 35, respectively). The algorithm then chooses the smaller of the calculated narrower and wider comparison distances (line 36), the intuition being that the viewport with the smaller distance more likely better distinguishes the difference between the failure occurring or not (i.e. ignoring potential 'noise' caused by unrelated but intended layout changes programmed as part of a breakpoint in the CSS). If this distance is above a threshold (as determined by the SATISFYTHRESHOLD function on line 37 of the algorithm), VERVE deems there to be sufficient 'disturbance' to the presentation of the web page to report the failure as a true positive (line 38), else VERVE classifies it as a false positive (line 39).

For computing the colour histograms and each distance metric, VERVE uses the publicly available implementations in OpenCV [23]. Since different distance metrics are possible, we employed each of the six that are currently provided by OpenCV: *Bhattacharyya Distance*, *Chi-square*, *Alternative Chi-square*, *Correlation*, *Intersection* and *Kullback–Leibler Divergence* [24]. We determined the thresholds experimentally (see Section 4 for more details) and evaluated the suitability of each of these options for the subjects in this paper's empirical study, which we introduce and discuss next.

4. EMPIRICAL EVALUATION

To empirically investigate the effectiveness and efficiency of VERVE, we employed it to classify the RLFs found in the web pages used in the previous evaluation of ReDECHECK [6], while also evaluating it on a new set of subjects not used in prior work to study either ReDECHECK or VISER. This section give details of the research questions of the study in Section 4.2. First, however, it gives more details on the two sets of subjects that we used to conduct the experiments.

4.1. Subject web pages

We used two sets of web page subjects to evaluate VERVE. The first set, which we refer to throughout the paper as the *initial set* of subjects, is the same set of web pages used to evaluate ReDECHECK's effectiveness by Walsh et al. [6] and was used to evaluate our technique in the conference version of this paper [16]. For this extended paper, we further evaluated VERVE on 20 new subjects, which we refer to as the *additional set* of pages and describe in more detail next.

The initial set of subjects, made available by Walsh et al. [6], are listed in Table I(a) and comprises 25 web pages. We took these subjects, without modification, directly from the repository cited in Walsh et al.'s paper (github.com/redecheck/example-webpages). However, while the original set of web pages studied by Walsh et al. involved 26 subjects, we found that the tool previously used to download one subject's resources (*StumbleUpon*) had not worked correctly, and we were unable to reproduce it. Since this web page was no longer available in its original form and we could not reconstruct the subject's archive, we had to exclude it from this paper's experiments. This gave us, as part of the initial set, a total of 326 RLFs reported by ReDECHECK for use in the evaluation of VERVE. We used the manual classifications made by Walsh et al. [6], which are publicly available at redecheck.org/issta17.

The 25 web pages were previously selected in a random fashion by Walsh et al. through the use of the randomusefulwebsites.com service that was recently re-branded as discuvver.com. Although all of these web pages are responsive, they represent a wide variety of application domains and page layouts. For instance, while *Airbnb*, *ConsumerReports* and *Duolingo* are the web pages of major organizations, others such as *BugMeNot*, *PepFeed* and *RainyMood* are pages created by individuals

Table I. Experimental subject web pages.

Web site name	URL	Number of HTML elements	Number of CSS declarations
		(a) The initial set of web pages	
3-Minute-Journal	3minutejournal.com	80	5408
AccountKiller	accountkiller.com/en	344	4685
AirBnb	airbnb.com	1470	9964
BugMeNot	bugmenot.com	42	654
CloudConvert	cloudconvert.com	908	6494
Consumer-Reports	consumerreports.org	1079	8330
Covered-Calendar	coveredcalendar.com	148	8324
Days-Old	daysold.com	66	2800
Dictation	dictation.io	195	8290
Duolingo	duolingo.com	856	4150
Honey	joinjoney.com/install	461	7999
HotelWifiTest	hotelwifitest.com	359	6833
Mailinator	mailinator.com	280	8729
MidwayMeetup	midwaymeetup.com	86	4072
Ninite	ninite.com	642	4091
Pdf-Escape	pdfescape.com	180	2041
Pepfeed	pepfeed.com	343	7341
Pocket	getpocket.com	664	6416
RainyMood	rainymood.com	89	112
RunPee	runpee.com	438	14 788
TopDocumentary	topdocumentaryfilms.com	411	1521
UserSearch	usersearch.org	866	3590
WhatShouldIReadNext	whatshouldireadnext.com/search	112	2224
WillMyPhoneWork	willmyphonework.net	782	6572
ZeroDollarMovies	zerodollarmovies.com	247	11 228
Total		11 148	146 656
		(b) The additional set of web pages	
EatThisMuch	eatthismuch.com	807	12 318
Forvo	forvo.com	584	19 722
GMapStreetViewPlayer	brianfolts.com/driver	268	5617
RetailMeNot	retailmenot.com	1336	2823
HoursOf	hoursof.com	1258	9513
SB-Admin-2	startbootstrap.com/themes/sb-admin-2	360	6656
SB-Agency	startbootstrap.com/previews/agency	420	6303
SB-Business-Casual	startbootstrap.com/previews/business-casual	56	4438
SB-Clean-Blog	startbootstrap.com/previews/clean-blog	93	6160
SB-Coming-Soon	startbootstrap.com/previews/coming-soon	43	5969
SB-Creative	startbootstrap.com/previews/creative	135	6318
SB-Freelancer	startbootstrap.com/previews/freelancer	284	6064
SB-Grayscale	startbootstrap.com/previews/grayscale	116	6120
SB-Landing-Page	startbootstrap.com/previews/landing-page	130	6146
SB-New-Age	startbootstrap.com/previews/new-age	127	6649
SB-One-Page-Wonder	startbootstrap.com/previews/one-page-wonder	68	4424
SB-Resume	startbootstrap.com/previews/resume	176	5924
SB-Stylish-Portfolio	startbootstrap.com/previews/stylish-portfolio	143	6363
SimilarSites	similarsites.com	478	10 268
Tiiime	tiii.me	80	847
Total		6962	138 642

or small organizations. It is also worth noting that these web pages feature different layouts that vary according to, for instance, the number of rows and columns, with pages such as *ZeroDollarMovies* having images in a tight, grid-based layout and others such as *TopDocumentary*mixing both text and images on a simple grid. In summary, these web pages come from a wide variety of application domains, ensuring the representativeness of this paper's empirical results.

We further evaluated Verve with 20 new subject web pages, as shown in Table I(b), referred to as the *additional set* of subjects. These web pages did not previously feature in either the ReDeCheck study [6] or the original study presented in the conference version of this paper [16]. These additional subjects are a collection of seven real web pages and 13 'theme' web pages. Although the bootstrap-based web pages are themes that are not meant to be hosted as is, they demonstrate features for multiple types of web pages. For instance, among these, 13 themes are full-featured starter templates for web dashboards, advertising agencies and art portfolios. Furthermore, they are maintained in popular GitHub repositories publicly hosted in the 'Blackrock Digital' organization at github.com/blackrockdigital. These regularly updated themes are frequently forked and starred, suggesting that the responsive web development community sees them as useful templates for starting high-quality web sites. The popularity of these themes can be illustrated by noting that, for instance, the SB-Admin-2 subject was created by 14 contributors who made 19 releases to a project that has currently over 4400 forks and 7300 stars. We collected these 'theme' subjects by searching for responsive demonstration web pages that were available in public GitHub repositories. Collectively, the additional 20 web pages have a total of 143 RLFs reported by ReDeCheck.

In summary, the subject collection process gives us an overall total of 45 web pages with 469 RLFs to be automatically classified by Verve as part of our empirical evaluation.

4.2. Research questions

The experiments focus on answering the following five research questions. RQs1–3 evaluate Verve with the initial set of subjects due to Walsh et al. [6], previously used to evaluate ReDeCheck and Viser. RQ4 exclusively evaluates Verve with the new, additional set of subjects, while RQ5 concludes the research questions and features both sets of subjects.

RQ1: Element collision, element protrusion and viewport protrusion failures. (a)*Can*Verve*automatically classify failure reports for these types of failures and how does it compare to manual classification?* For this research question, we compare Verve's automated classification to the manual classification, using the tool's default setting of performing the image analysis at the narrowest viewport width reported for each RLF. (b)*Within the viewport range of these types of failure, what is the point of inspection that has the best chance of revealing a true positive?* For this research question, we determine if it is best to perform the image analysis at the narrowest viewport width for an RLF. We compare Verve's classification to the results of the manual classification process for three points in the RLF's viewport range: the minimum or narrowest (as investigated in RQ1a), the middle of the range, and the maximum width.

RQ2: Wrapping failures. *Can*Verve*automatically classify wrapping failure reports and how does it compare to manual classification?* To answer this research question, we compare Verve's classification results to the manual classification for wrapping failures.

RQ3: Small-range failures. *Can*Verve*automatically classify small-range failure reports, how does it compare to manual classification, and which colour histogram distance metric works the best?* For this research question, we compare Verve's results to the manual classifications for small-range failures, evaluating which colour histogram distance method is the most effective.

RQ4: Additional subjects. To assess the generalizability of Verve, we evaluate it with the additional set of subjects that were not previously used to study either ReDeCheck or Viser.

(a)*Does*Verve*effectively classify collision, element protrusion and viewport protrusion failure reports for the new responsive web pages?* To answer this question, we investigate the results from Verve at the minimum, middle and maximum viewports in the reported range for each RLF with the additional set of subjects.

(b)*Does*Verve*effectively classify wrapping failure reports for the new responsive web pages?*To answer this question, we analyse the results of Verve on the additional set of web pages for wrapping failures at the minimum, middle and maximum viewports in each failure's range.(c)*Does*Verve*effectively classify small-range failure reports for the new responsive web pages?* This question investigates the results of Verve on the additional set of web pages for both the horizontal referencing approach and the horizontal-plus-vertical referencing approach.

RQ5: Tool efficiency. *Does*Verve*efficiently classify the failure reports?* To determine if Verve operates fast enough to support its practical use in the testing of responsive web pages, we recorded the time the tool takes to perform failure report classification for all of the subjects.

4.3. Experimental methodology

To evaluate VERVE, we attempted to match the execution environment, as closely as is possible, to the setup for the original ReDECHECK experiments, thereby avoiding discrepancies in the results that might be due to differences in the experimental setup between the two evaluations. We therefore ran VERVE on an iMac with 8 GB of RAM, running OS version 10.13 and using version 46 of the Firefox browser. Similar to ReDECHECK, VERVE uses Selenium WebDriver [25] to interact with the web browser and to render web pages without scrollbars and at a fixed viewport height of 1000 pixels. To support small-range RLF classification, we integrated OpenCV [23] version 3.2 into VERVE, thus enabling the generation and comparison of colour histograms of the AOCS.

Throughout Section 4.5 as we provide answers to the research questions, we discuss instances where VERVE disagrees with the manual classification in three different categories: *subjective*, *obscured* and *misclassified* RLFs. For *subjective* and *obscured* disagreements, VERVE classified the RLF correctly, but human judgement of the RLF is potentially subjective (e.g. the visual discrepancy is not substantial, only amounting to a few pixels) or obscured because ReDECHECK reported (and VERVE subsequently analysed) different HTML elements to those actually causing the visual anomaly. *Misclassified* RLFs arise when either VERVE or the manual classification were certifiably incorrect. We discuss the reasons for each of these disagreements with each subject web page, outlining possible steps for future work to improve VERVE, if any steps are needed.

To answer RQ1a, we ran VERVE on each of the 117 RLFs identified on the initial set of web page subjects to reach an automatic classification. VERVE was configured to use the minimum viewport width reported for the range of each RLF concerned. We then checked whether VERVE agreed with the manual categorization of the RLF as decided in the original study by Walsh et al. [6]: true positive (TP, an observable failure), non-observable issue (NOI), or false positive (FP, no failure). FPs are failures reported by ReDECHECK that do not exhibit an issue visually in the design of the web page or in its internal DOM representation. We then calculated the percentage agreement of VERVE with the previous manual classification, investigating any differences in the categorization.

To answer RQ1b, we followed the same methodology as RQ1a, but ran VERVE using additional inspection points: the middle of the range reported for the RLF by ReDECHECK and the maximum point (i.e. the upper bound) of the range. While running the experiment for RQ1b, VERVE led us to the discovery of a defect in ReDECHECK. For 35 viewport protrusion RLFS, ReDECHECK incorrectly reported the upper bound of the viewport range for the failure. Rather than rendering a page at each possible viewport width to construct the RLG, ReDECHECK normally samples the entire range by rendering the page at intervals, performing a binary search between the last two sampled points to identify the precise viewport widths at which the relative alignment of two HTML elements, or the visibility of an individual element, changed. Since these RLFs were false positives, the incorrectly reported upper bound had a further, unintended effect on VERVE's behaviour. We found that ReDECHECK would produce the correct results if we changed its interval size to 1. After this reconfiguration, we re-ran ReDECHECK for the pages involving these particular RLFS.

To answer RQ2, we ran VERVE on each of the 14 wrapping failures detected in the initial set of 25 web pages. Given the failure viewport range of each responsive layout failure, we used VERVE to inspect and automatically classify the reported wrapping failures at the minimum, middle and maximum points of the range. As before, these automatic classifications were either TP, NOI or FP.

To answer RQ3, we used VERVE to automatically classify the 195 small-range failures reported with the initial set of web pages. Since small-range failures are restricted to a range of 1–5 viewports, only the minimum point is used to automatically classify the failure, using the viewports immediately narrower and wider than the small-range failure as comparison points, as discussed in Section 3.3. To establish the best histogram comparison measure, we used the full set of methods available in the OpenCV library: *Bhattacharyya Distance*, *Chi-square*, *Alternative Chi-square*, *Correlation*, *Intersection* and *Kullback–Leibler Divergence* [23]. For the *Bhattacharyya Distance*, *Kullback–Leibler Divergence*, *Chi-square* and *Alternative Chi-square* metrics, the lower the distance value, the better the match, with zero representing a perfect match. Since the images compared may have different sizes, some negative values arise when using *Kullback–Leibler Divergence*. With a perfect score of zero, the absolute value of this measure is used. The *Correlation* and *Intersection* metrics, however, use a higher-is-better score. *Correlation*'s score is bounded at one, while *Intersection*'s score is unbounded. So that we could easily and consistently compare the results using each metric, we wrote a wrapper function around *Correlation* and *Intersection* to convert its result to a lower-is-better score, as done by the other metrics. The wrapper for *Correlation* inverts its score, while the wrapper for *Intersection* normalizes its score and inverts its result.

In order for VERVE to conclude that it has detected 'enough' of a visual disturbance to classify a small-range failure as TP, we needed to set a TP threshold for each metric. To establish the thresholds, we did an initial run of VERVE to find the AOCs, create a colour histogram for each AOC, and output the *distance* between all pair of histograms. Throughout this process, a pair of histograms was sourced from the failure viewport and a comparison viewport, as explained in Algorithm 5. To automate and ensure the correctness of this process, we implemented a tool, called ThresholdFINDER, and used it to establish a TP threshold for each of the six measures, with the ultimate goal of maximizing accuracy. That is, the goal of ThresholdFINDER is to automatically find the thresholds that will maximize agreement with the manual classification. This tool takes the distances as input and uses them as candidate thresholds along with ±0.01 of each distance. Alternating through each candidate threshold, ThresholdFINDER automatically classifies each failure based on the candidate threshold. ThresholdFINDER's classifications are then compared to the manual classification to calculate an accuracy for each candidate threshold. When matching against the manual classification, ThresholdFINDER uses a balanced score for both classes, TP and FP. Ultimately, the tool reports the threshold with the maximum accuracy from the set of candidates. Whenever multiple candidate thresholds may achieve the same accuracy, ThresholdFINDER reports the threshold in the middle position of the identified set.

With VERVE using two alternative approaches to classify small-range failures (i.e. horizontal referencing and horizontal-plus-vertical referencing), we used ThresholdFINDER twice with the initial set of web pages, once for each approach. We refer to these thresholds that were determined using only the initial set of web pages as the *prospective* thresholds. To establish whether these thresholds are more generally applicable, the prospective thresholds are used by VERVE to automatically classify the small-range failure from the additional set of web pages as part of RQ4. While the prospective thresholds were determined using a subset of the total subjects, we used ThresholdFINDER again to determine a set of *retrospective*thresholds using all 45 web pages. As part of the discussion in Section 4.6, the retrospective thresholds are used to weigh in on the overall performance of VERVE when it uses the prospective thresholds.

To answer RQ4, we assessed how well the findings from the first three research questions extend to the 20 subject web pages in the additional set. Similar to the methodology for the previous research questions, we set VERVE to investigate three inspection points for each failure range and to automatically classify RLFs. For small-range failures, we used all six histogram methods and used the ThresholdFINDER-established thresholds from RQ3 to classify the failures from the subjects. Unlike the previous experiments, for this new set of web pages, we manually classified the failures produced by REDECHECK to compare with the classifications automatically produced by VERVE.

To control a validity threat arising from the lack of true positive small-range failures in the additional set of pages reported by REDECHECK, we manually injected faults into each page from the additional set to create small-range failures suitable for analysis as part of this research question. To complete this task, the first author manually selected a target element that seemed likely to cause a layout failure if displaced. The first author then made two changes to the page. First, they manually injected a single media query rule into each page, with the viewport range of 992–995 pixels wide. Second, they then added CSS rules nested within this query pertaining to the `margin-left` or `margin-top` property of the chosen element to cause it to be offset from its original position within the range of the media query. Both of these modifications were intended to produce a layout failure limited to a small number of viewport widths that REDECHECK would report as a small-range failure. We chose the range of 992–995 pixels since the majority of the additional set of subjects has a manually programmed breakpoint in the CSS at 992 pixels, making that position in the viewport range a realistic position where a small-range defect may occur in practice.

Finally, to answer RQ5, we ran VERVE for all 469 RLFs from our entire set of 45 web page subjects, examining each one at the lower bound of the range reported by ReDECHECK and recording the time taken for VERVE to run in each instance. We repeated this process 30 times for each RLF to obtain a reliable estimate of VERVE's running time and to minimize the chance effects that might be caused by, for example, the underlying operating system hosting the experiments.

VERVE's runtime includes a 200-ms added delay, used for all web pages and experiments in this paper, to allow the HTML elements to load and 'settle' into their final location and support, for instance, JavaScript-programmed transitional effects. This delay time is repeated whenever VERVE resizes the browser, scrolls the web page or changes the opacity of an element, meaning that multiple delays may be introduced throughout the entire classification process. We recorded VERVE's execution times with and without this delay. Importantly, we excluded the time to load the web page and resize the browser from our measurements as this cost is shared by any technique, whether manual, semi-automated or automated through the use of VERVE. All of the recorded times account for the overhead of finding the offending HTML elements, visually confirming and classifying the failure with VERVE, and writing to disk all the diagnostic images of the web pages.

4.4. Threats to validity

One threat to the validity of this paper's results is the extent to which they generalize to other web pages. However, care was taken to ensure that the overall set of subjects were diverse in terms of functionality and complexity, thus representing as wide a variety of web pages as possible. As discussed in Section 4.1, the initial set of subjects was drawn from the study by Walsh et al. [6], while the additional set contains subjects new to this paper. We selected the new subjects both by using a random URL generator and by browsing responsive theme templates that are freely available for public use. As Table I(a) and (b) show, the subjects vary considerably in complexity from 42 to 1470 HTML elements and from 112 to 19722 CSS declarations. The functionality and responsive layout of the chosen web pages also differ substantially; from SB-Admin-2, a template for back-end administrative portals; to DaysOld, providing calendar features; and Airbnb, supporting international e-commerce corporations. Since the set of additional subjects contained only false-positive reports and therefore did not contain any small-range failures, we manually injected code to construct them, as detailed in Section 4.3. Even though these failures are synthetic—and so may not be representative of all real-world small-range failures—we judge that they exemplify the concerns that practicing web developers have about this type of layout defect [20].

The validity of this paper's experiments hinges on accurately matching manual classifications of ReDECHECK's failure reports with the classifications automatically produced by VERVE. Since the manual results from the experimental evaluation of ReDECHECK [6] (involving the initial set of subjects) did not include the XPath of offending elements, the failures were manually matched using the available snapshots. These snapshots, combined with the type of failure, range and name of the web page, enabled us to confidently perform the matching. For the additional set of subjects, there was no pre-existing manual classification and so this was a task undertaken by the first author. Since the conclusions of the manual process are inherently subjective, we make our classifications and VERVE's results publicly available for independent evaluation at verve-tool.github.io. Importantly, we judge that many of the failures raised by ReDECHECK have very little subjectivity for these additional subjects and can be easily classified, thereby limiting this validity threat.

A further threat to validity is a defective implementation of the VERVE tool, which would compromise its classification of the responsive layout failures reported by ReDECHECK. To control this threat, we configured VERVE to keep a record of all the images used to evaluate each responsive layout failure. Furthermore, VERVE also maintained a record of the coordinates of each offending element. We consulted these records during the examination of all mismatched classifications, thus helping us to ensure that the prototype operated correctly. To further establish a confidence in the correctness of VERVE, we regularly performed additional manual and automated testing.

Furthermore, VERVE's use of the Firefox web browser and the Selenium testing tool is another validity threat. Yet, Firefox is a popular browser that is frequently used for RWD testing and therefore a good option for ensuring the representativeness of the results. Since this paper's experiments exclusively report on the use of VERVE on MacOS, we controlled this validity threat by trying the tool on other operating systems, ultimately confirming that its classifications are similar. We are also confident that Selenium did not compromise VERVE's correctness because, in addition to our error-free experiences when using Selenium, this web automation framework has a large and active user and developer community respectively committed to reporting and fixing defects. Similarly, defects in the OpenCV library are also a validity threat that may compromise this paper's experimental results. Even though OpenCV is a widely used library for image analysis, we addressed this concern by manually confirming OpenCV's analysis of select images and by following the best practices for its configuration and use (e.g. Bradski and Kaehler [24]).

Finally, since the timing results for RQ5 are subject to the interference of background operating system processes, we ran all of the experiments 30 times to mitigate the possibility of bias in our results. Importantly, to support the replication of this paper's experiments and to further control all the aforementioned validity threats, we made the VERVE tool, its documentation and the scripts needed to run the experiments all available at github.com/verve-tool/verve in a GitHub repository. To further support the confirmation of this paper's results, we have made a screenshot of every ReDECHECK RLF report, a summary of the manual classification results, and details about VERVE's automatic classification available at the verve-tool.github.iosite.

4.5. Experimental results

RQ1: Element collision, element protrusion and viewport protrusion failures

RQ1(a) *Can* VERVE *automatically classify failure reports for these types of failures and how does it compare to manual classification?* Table I gives the results from running VERVE on the outputs of ReDECHECK and their agreement with the manual classification performed by Walsh et al. [6]. For completeness, Table III(a) gives the full, broken down set of manually classified results from the original Walsh et al. study. For this research question, we focus on the results from the smallest viewport in the failures reported range, called the 'Minimum'. The numbers in parentheses correspond to the results obtained with the old version of the 'best effort' part of the analysis for viewport protrusion that was used in the technique from the conference version of this paper, as discussed in Section 3.1. We explain the results obtained with the latest version of VERVE used in this paper and then discuss the differences with the best effort viewport protrusion analysis. Table II shows that VERVE had an 86.3% overall agreement with the manual classification, with a further breakdown of this result by the RLF type. The 'Agreement with manual' section shows the ratio of failures for VERVE and manual classification, resulting in the reported percentages. The second number is the manual classification total (drawn from the totals in Table III(a)), while the first is the number of those failures that were categorized in the same manner by VERVE. At 93.5%, the best level of agreement between manual and VERVE is for the element collision failures.

Table II shows that there were 16 instances where VERVE's classification of an RLF did not agree with the outcome from manual analysis. We next discuss these 16 instances in three different categories introduced in Section 4.3, namely, *subjective*, *obscured* and *misclassified* RLFs. Nine RLFs fall into the *subjective* category. While these RLFs have a visual impact, the difference is so small that they are almost imperceptible to humans. Two of these cases involved changes to 2 pixels in width, yielding no real observable visual difference. While these RLFs are technically TPs, and were classified as such by VERVE, the manual analysis subjectively categorized them as NOIs. Future work needs to take these small differences into account when analysing RLFs.

Two further RLFs, evident in the ConsumerReports subject, were *obscured*. Two RLFs are TPs and were classified manually as such, yet VERVE reported them as NOIs. This was because ReDECHECK did not report the most specific elements involved in the failure. While VERVE's analysis was correct for the elements it was given by ReDECHECK, there was a noticeable visual effect detectable by humans. Since the manual analysis was not limited to the study of only the HTML elements reported by ReDECHECK, the effect of the RLF was easily spotted as a TP. Importantly, this difference is really a bug in ReDECHECK, rather than a problem with VERVE.

Table II. The results from using VERVE to classify the element collision, element protrusion and viewport protrusion responsive layout failures in the initial set of web pages after inspecting the 'Minimum' of the reported failure range. In this table 'TP', 'NOI' and 'FP' respectively denote a true positive, non-observable issue and false positive, as explained in Section 4.3. The numbers in parentheses correspond to the results obtained with the old version of the 'best effort' part of the analysis that was used in the technique presented in the conference version of this paper, as discussed in Section 3.1.

	Minimum									
	Element collision			**Element protrusion**			**Viewport protrusion**			
	TP	NOI	FP	TP	NOI	FP	TP	NOI	FP	**Tostal**
3-Minute-Journal	—	1/1	—	—	2/2	—	8/8	—	—	11
AccountKiller	—	—	—	—	—	—	—	—	—	—
AirBnb	—	1/1	—	—	1/4	3/—	(1) 2/—	(3) 2/4	—	9
BugMeNot	—	—	—	1/1	3/3	—	(1) 2/2	(1) —/—	—	6
CloudConvert	1/1	—	—	—	—	—	—	—	—	1
Consumer-Reports	1/—	6/7	—	1/1	3/3	—	9/9	3/3	—	23
Covered-Calendar	—	—	—	—	—	—	—	3/3	—	3
Days-Old	—	—	—	—	—	—	—	1/1	—	1
Dictation	—	—	—	—	—	—	—	1/1	—	1
Duolingo	1/—	—/1	—	—	—	—	2/2	2/2	—	5
Honey	—	—	—	—	8/8	—	—	2/2	—	10
HotelWifiTest	—	—	—	—	—	—	—/1	1/—	—	1
Mailinator	—	1/1	—	—	—	—	—	—	—	1
MidwayMeetup	1/1	—	—	—	1/1	—	(—) 1/—	(1) —/1	—	3
Ninite	—	—	—	—	—	—	—	—	—	—
Pdf-Escape	—	—	—	—/1	6/5	—	3/1	1/3	—	10
Pepfeed	4/4	3/3	—	—	2/2	—	1/1	1/1	—	11
Pocket	—	2/2	—	—	3/3	—	—	—	—	5
RainyMood	—	—	—	—	—	—	—	—	—	—
RunPee	—	—	—	—	—	—	—	—	—	—
TopDocumentary	—	7/7	—	—	4/4	—	—	—	—	11
UserSearch	—	1/1	—	—	—	—	—	—	—	1
WhatShouldIReadNext	—	—	—	—	—	—	—	2/2	—	2
WillMyPhoneWork	1/1	—	—	—	1/1	—	—	—	—	2
ZeroDollarMovies	—	—	—	—	—	—	—	—	—	—
Total	9	22	—	2	34	3	(25) 28	(22) 19	—	117
Agreement with manual	7/7	22/24	—	1/3	32/36	—	(21) 22/24	(19) 17/23	—	—
Agreement per failure type	93.5 %			84.6 %			(85.1) 83 %			—
Per inspection point	(87.2) 86.3 %									—

Two viewport protrusion RLFs were *misclassified* by VERVE. One viewport protrusion failure with *PDF-Escape* was classified by VERVE as an NOI but was manually classified as a TP. This is due to the `overflow` property of the protruding element being set as `hidden`. The protruding content could therefore not be 'seen' by VERVE. This result suggests the need for future work that modifies VERVE so that it manipulates the `overflow` property or tracks missing content from one viewport width to another. The final viewport protrusion involved the *Hotel WiFi Test* subject. For this page, content overflowing out of the viewport was classified by the VERVE tool as an NOI, although the manual analysis correctly categorized it as a TP. In this case, the human expert had to scroll horizontally to read the overflowing content, which did not line up correctly with elements in the web page's banner. With this RLF, VERVE did not detect any elements overwriting any other—the browser renders the overflowing content to a 'blank' area of the page—leading it to classify the issue as NOI. Yet, there were other side effects of this issue that *were* visually observable that VERVE does not currently account for: first, the horizontal size of the page had changed, meaning the user had to scroll to reach the content. Second, the overflowing content no longer aligned with the header of the web page. The inability of VERVE to detect these types of effects is something that we intend to address as part of a future work, as explained in Section 6.

Table III. The manual classification of RLFs. See Table 2 for a full description of the columns.

	Element collision			**Element protrusion**			**Viewport protrusion**			**Wrapping**			**Small-range**			
	TP	NOI	FP	TP	NOI	FP	TP	NOI	FP	TP	NOI	FP	TP	NOI	FP	**Total**
(a) Manual classification of RLFs from the initial set of web pages from a prior study [6]																
3-Minute-Journal	—	1	—	—	2	—	8	—	—	—	—	—	—	—	1	12
AccountKiller	—	—	—	—	—	—	—	—	—	2	—	—	147	—	5	154
AirBnb	—	1	—	—	4	—	—	4	—	2	—	—	—	—	2	13
BugMeNot	—	—	—	1	3	—	2	—	—	1	—	—	—	—	—	7
CloudConvert	1	—	—	—	—	—	—	—	—	—	—	—	1	—	—	2
Consumer-Reports	—	7	—	1	3	—	9	3	—	—	—	—	—	—	1	24
Covered-Calendar	—	—	—	—	—	—	—	3	—	2	—	—	—	—	—	5
Days-Old	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	1
Dictation	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	1
Duolingo	—	1	—	—	—	—	2	2	—	—	—	2	—	—	1	8
Honey	—	—	—	—	8	—	—	2	—	—	—	—	—	—	3	13
HotelWifiTest	—	—	—	—	—	—	1	—	—	—	—	—	—	—	2	3
Mailinator	—	1	—	—	—	—	—	—	—	—	—	—	—	—	2	3
MidwayMeetup	1	—	—	—	1	—	—	1	—	—	—	—	—	—	—	3
Ninite	—	—	—	—	—	—	—	—	—	1	—	1	—	—	—	2
Pdf-Escape	—	—	—	1	5	—	1	3	—	—	—	—	—	—	—	10
Pepfeed	4	3	—	—	2	—	1	1	—	1	—	—	2	—	14	28
Pocket	—	2	—	—	3	—	—	—	—	—	—	—	—	—	3	8
RainyMood	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
RunPee	—	—	—	—	—	—	—	—	—	—	—	1	—	—	5	6
TopDocumentary	—	7	—	—	4	—	—	—	—	—	—	—	—	—	2	13
UserSearch	—	1	—	—	—	—	—	—	—	1	—	—	—	—	—	2
WhatShouldIReadNext	—	—	—	—	—	—	—	2	—	—	—	—	—	—	—	2
WillMyPhoneWork	1	—	—	—	1	—	—	—	—	—	—	—	2	—	—	4
ZeroDollarMovies	—	—	—	—	—	—	—	—	—	—	—	—	—	—	2	2
Total	7	24	—	3	36	—	24	23	—	10	—	4	152	—	43	326
Total per failure type		31			39			47			14			195		—

Table III. Continued

	Element collision			**Element protrusion**			**Viewport protrusion**			**Wrapping**			**Small-range**			
	TP	NOI	FP	TP	NOI	FP	TP	NOI	FP	TP	NOI	FP	TP	NOI	FP	**Total**
(b) The manual classification of the additional set of web pages																
EatThisMuch	—	5	—	—	6	—	1	1	—	—	—	1	—	—	2	16
Forvo	—	—	—	—	3	—	—	—	—	2	—	2	—	—	29	36
GMapStreetViewPlayer	—	—	—	—	2	—	—	—	—	—	—	—	—	—	—	2
HoursOf	—	—	—	—	1	—	—	1	—	—	—	—	—	—	—	2
RetailMeNot	2	—	—	—	30	—	—	—	—	—	2	4	—	—	15	53
SB-Admin-2	—	—	—	—	—	—	1	—	—	—	—	—	—	—	1	2
SB-Agency	—	4	—	—	8	—	1	—	—	—	—	3	—	—	—	16
SB-Business-Casual	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
SB-Clean-Blog	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
SB-Coming-Soon	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
SB-Creative	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
SB-Freelancer	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
SB-Grayscale	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	1
SB-Landing-Page	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	1
SB-New-Age	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
SB-One-Page-Wonder	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
SB-Resume	—	1	—	—	—	—	—	—	—	2	—	—	—	—	—	3
SB-Stylish-Portfolio	—	—	—	—	—	—	—	—	—	—	—	—	—	—	4	4
SimilarSites	—	—	—	—	—	—	4	—	—	—	—	—	—	—	1	5
Tiiime	—	—	—	—	1	—	—	—	—	—	—	—	—	—	1	2
Total	2	10	—	—	51	—	8	2	—	4	2	11	—	—	53	143
Total per failure type		12			51			10			17			53		—

The final three element protrusion RLFs were *misclassified* by the manual analysis. VERVE found that these were FPs, since there was no protrusion at the DOM level. The manual analysis incorrectly reported these as NOIs, since there was no visual impact. Ultimately, we judge the root cause of this misclassification to be a defect in ReDeCheck's collection of DOM information when constructing the RLG, which relies on JavaScript injected into the web page, since the VERVE tool, which relies on Selenium, reported no protrusion.

Perhaps curiously, the level of agreement went down from 87.2% to 86.3% with the use of the 'improved' best effort analysis for viewport protrusion used in this paper. This can be seen when we compare the numbers for the 'old' best effort analysis that featured in the conference version of this paper, which appear in parentheses in Table II, with those for the improved analysis. The differences concern three failures associated with *Airbnb*, *BugMeNot* and *MidwayMeetup*. For *Airbnb* and *MidwayMeetup*, the best effort analysis originally failed, but serendipitously matched the manual classification of NOI. Yet, following improvements in this paper, VERVE classified the two failures as TPs. Each corresponds to a part of an image protruding out of the viewport, that, given the nature of the images concerned (e.g. for *MidwayMeetup*, part of a map), is difficult for a human eye to discern that something is actually 'missing' off the edge of the page. We categorize these disagreements, therefore, as *subjective*. Following the improved best effort analysis, VERVE's result for *BugMeNot* matched the manual analysis result, highlighting the benefits associated with this new approach.

Table IV. The results from using VERVE for element collision, element protrusion and viewport protrusion failures of the initial set of web pages after inspecting the 'Middle' of the reported failure range.

	Middle									
	Element collision			**Element protrusion**			**Viewport protrusion**			
	TP	NOI	FP	TP	NOI	FP	TP	NOI	FP	**Total**
3-Minute-Journal	—	1/1	—	—	2/2	—	6/8	2/—	—	11
AccountKiller	—	—	—	—	—	—	—	—	—	—
AirBnb	—	1/1	—	—	1/4	3/—	2/—	2/4	—	9
BugMeNot	—	—	—	1/1	3/3	—	(1) 2/2	(1) —/—	—	6
CloudConvert	1/1	—	—	—	—	—	—	—	—	1
Consumer-Reports	—	7/7	—	1/1	3/3	—	9/9	3/3	—	23
Covered-Calendar	—	—	—	—	—	—	—	3/3	—	3
Days-Old	—	—	—	—	—	—	—	1/1	—	1
Dictation	—	—	—	—	—	—	—	1/1	—	1
Duolingo	1/—	—/1	—	—	—	—	2/2	2/2	—	5
Honey	—	—	—	—	8/8	—	—	2/2	—	10
HotelWifiTest	—	—	—	—	—	—	—/1	1/—	—	1
Mailinator	—	1/1	—	—	—	—	—	—	—	1
MidwayMeetup	1/1	—	—	—	1/1	—	1/—	—/1	—	3
Ninite	—	—	—	—	—	—	—	—	—	—
Pdf-Escape	—	—	—	—/1	6/5	—	2/1	2/3	—	10
Pepfeed	4/4	3/3	—	—	2/2	—	—/1	2/1	—	11
Pocket	—	2/2	—	—	3/3	—	—	—	—	5
RainyMood	—	—	—	—	—	—	—	—	—	—
RunPee	—	—	—	—	—	—	—	—	—	—
TopDocumentary	—	7/7	—	—	4/4	—	—	—	—	11
UserSearch	—	1/1	—	—	—	—	—	—	—	1
WhatShouldIReadNext	—	—	—	—	—	—	—	2/2	—	2
WillMyPhoneWork	1/1	—	—	—	1/1	—	—	—	—	2
ZeroDollarMovies	—	—	—	—	—	—	—	—	—	—
Total	8	23	—	2	34	3	(23) 24	(24) 23	—	117
Agreement with manual	7/7	23/24	—	1/3	32/36	—	(17) 18/24	17/23	—	—
Agreement per failure type		96.8 %			84.6 %			(72.3) 74.5 %		—
Per inspection point					(82.9) 83.8%					—

Table V. The results from using VERVE for element collision, element protrusion and viewport protrusion failures of the initial set of web pages after inspecting the 'Maximum' of the reported failure range.

	Maximum									
	Element collision			Element protrusion			Viewport protrusion			
	TP	NOI	FP	TP	NOI	FP	TP	NOI	FP	**Total**
3-Minute-Journal	—	1/1	—	—	2/2	—	—/8	7/—	1/—	11
AccountKiller	—	—	—	—	—	—	—	—	—	—
AirBnb	—	1/1	—	—	1/4	3/—	2/—	2/4	—	9
BugMeNot	—	—	—	1/1	3/3	—	(1) 2/2	(1) —/—	—	6
CloudConvert	1/1	—	—	—	—	—	—	—	—	1
Consumer-Reports	—	7/7	—	1/1	3/3	—	8/9	4/3	—	23
Covered-Calendar	—	—	—	—	—	—	—	3/3	—	3
Days-Old	—	—	—	—	—	—	—	1/1	—	1
Dictation	—	—	—	—	—	—	—	1/1	—	1
Duolingo	1/—	—/1	—	—	—	—	2/2	2/2	—	5
Honey	—	—	—	—	8/8	—	—	2/2	—	10
HotelWifiTest	—	—	—	—	—	—	—/1	1/—	—	1
Mailinator	—	1/1	—	—	—	—	—	—	—	1
MidwayMeetup	1/1	—	—	—	1/1	—	—	1/1	—	3
Ninite	—	—	—	—	—	—	—	—	—	—
Pdf-Escape	—	—	—	—/1	6/5	—	2/1	2/3	—	10
Pepfeed	4/4	3/3	—	—	2/2	—	—/1	2/1	—	11
Pocket	—	2/2	—	—	3/3	—	—	—	—	5
RainyMood	—	—	—	—	—	—	—	—	—	—
RunPee	—	—	—	—	—	—	—	—	—	—
TopDocumentary	—	7/7	—	—	4/4	—	—	—	—	11
UserSearch	—	1/1	—	—	—	—	—	—	—	1
WhatShouldIReadNext	—	—	—	—	—	—	—	2/2	—	2
WillMyPhoneWork	1/1	—	—	—	1/1	—	—	—	—	2
ZeroDollarMovies	—	—	—	—	—	—	—	—	—	—
Total	8	23	—	2	34	3	(15) 16	(31) 30	1	117
Agreement with manual	7/7	23/24	—	1/3	32/36	—	(10) 11/24	18/23	—	—
Agreement per failure type		96.8 %			84.6 %			(59.6) 61.7 %		—
Per inspection point					(77.8) 78.6 %					—

Conclusion for RQ1(a): VERVE demonstrates high agreement 86.3% with manual classification when set to study the minimum point of the viewport range reported for each RLF.

RQ1(b)*Within the viewport range of these types of failure, what is the point of inspection that has the best chance of revealing a true positive?* Tables IV and V respectively show the results from when VERVE was configured to inspect the minimum and maximum point of the viewport range for each ReDECHECK-reported RLF. The results show that VERVE's classification can vary, depending on the chosen inspection point. VERVE is more likely to agree with manual inspection at an RLF's minimum viewport width. Compared to an agreement of 86.3% at the minimum width, the agreement for the middle and maximum point of the range drops to 83.8% and 78.6%, respectively. We next investigate the reason for the classification differences at each of these inspection points, referring to specific subjects to illustrate the key trade-offs in classification agreement.

The 'Middle' inspection point. Overall, there are four RLFs for which VERVE's classification did not agree with the manual analysis at the middle of the viewport range, for which there was an agreement at the minimum. Each RLF was a viewport protrusion; we now discuss these on a case-by-case basis. For the first two RLFs, the visibility of the failure varied depending on the viewport width chosen from the range reported by ReDECHECK. Thus, the change in classification reported by VERVE was correct for the RLFs involving *PDF-Escape* and PepFeed. The manual classification for these two RLFs is a true positive, which is accurate at the minimum viewport that ReDECHECK reports. However, as space expands, both RLFs become non-observable in the middle of the range. Thus, the manual analysis judgement does not hold for the entire range reported for each RLF, being correct at the minimum viewport width reported for the RLF, but incorrectly classified at the wider viewport widths. Importantly, VERVE can automatically detect the differences in observability. Both of the final two RLFs involve the *3-Minute Journal* subject. Notably, VERVE and the manual analysis agree that the RLF is a TP at the minimum viewport width. However, VERVE categorizes them as NOIs at the middle of the range. In this case, content overflows the viewport, which the tool does not properly detect, leading to a *misclassification* by the VERVE tool. This scenario is essentially identical to that which we experienced with the *Hotel WiFi Test* subject discussed as part of our answer to RQ1(a)—one which we intend to address as part of future work.

The 'Maximum' inspection point. There were seven RLFs for which VERVE's classification matched the manual classification at the minimum and middle inspection point, but not at the reported maximum viewport width. Again, each responsive layout failure was a viewport protrusion. In each case, the visibility of the RLF objectively changes, becoming an NOI at this inspection point. Similar to the first two differences identified for the middle inspection point, the manual analysis for these RLFs is a single judgement for the entire range. Hence, it does not take into account the change from visible to non-observable from narrower to wider viewport widths. Six of these RLFs are with the *3-Minute Journal* subject; the last RLF is from *ConsumerReports*.

Another notable result at the maximum point of inspection is a FP classification of a viewport protrusion failure for *3-Minute Journal*. When we investigated this issue, we found that based on the DOM coordinates as retrieved by VERVE and ReDECHECK, VERVE reported no protrusion, yet ReDECHECK reported the element as protruding by a single pixel. This is due in part to the different ways in which the two tools extract the DOM of the page—ReDECHECK uses JavaScript injected into the page to retrieve the DOM, while VERVE uses Selenium. We encountered a similar issue when answering RQ1(a), and we will investigate it as part of future work.

Following our improvements to the best effort analysis for viewport protrusion, the levels of agreements increase, since there was no disagreement for the failures described for RQ1a for *Airbnb* and *MidwayMeetup* at the middle and maximum inspection points, while the improved best effort analysis matched the manual classification for *BugMeNot*, as it did at the minimum inspection point.

Conclusion for RQ1(b): VERVE is more likely to agree with manual inspection at the minimum viewport width reported for each responsive layout failure. Nevertheless, inspection of larger viewports in the reported range is the only way to ensure the consistency of the classification throughout the range. The differences that are evident at wider inspection points can be explained by three phenomena: (1) the fact that a single verdict is produced for an RLF when its visibility may change throughout the viewport range for which it is reported; (2) the visibility/non-visibility of an RLF can be subjective as the viewport width changes; and (3) a small number of misclassified results by VERVE. It is important to note that the RLFs involved in the classification differences were exclusively viewport protrusion failures.

RQ2: Wrapping failures. For wrapping failures, Table VI shows the results from running the VERVE tool on wrapping RLF reports produced by ReDECHECK for the initial set of 25 web pages, and the agreement of these results with the manual classification shown by Table V(a). The results show that VERVE achieves agreement with the manual classification 78.6% of the time for all three inspection points. VERVE disagrees with the manual analysis for three failures. Two of these failures are with *Duolingo*. In these cases, the HTML elements are text-based links that form several rows in the footer of the page. Clearly, the developer intends these to wrap as the viewport size is reduced, potentially leaving one wrapped element on its own on a line. Therefore, the manual analysis classified these failures as false positives, yet VERVE reported them as true positives. The third failure is with *Ninite*. For this subject, the HTML elements concerned do not visually form a row, yet were identified as such at the DOM level. Again, the manual analysis resulted in a false-positive classification, yet the VERVE tool reported a true positive.

All three of the aforementioned cases are examples of *misclassification* by VERVE. In the case of *Duolingo*, it is difficult to judge the developer's intent, but VERVE could potentially be improved to detect these types of special cases, and with the *Ninite* example, further work could improve

Table VI. Results from Verve's use at three inspection points on wrapping failures in the initial set of pages.

	Wrapping									
	Minimum			**Middle**			**Maximum**			
	TP	NOI	FP	TP	NOI	FP	TP	NOI	FP	**Total**
1 3-Minute-Journal	—	—	—	—	—	—	—	—	—	—
AccountKiller	2/2	—	—	2/2	—	—	2/2	—	—	2
AirBnb	2/2	—	—	2/2	—	—	2/2	—	—	2
BugMeNot	1/1	—	—	1/1	—	—	1/1	—	—	1
CloudConvert	—	—	—	—	—	—	—	—	—	—
Consumer-Reports	—	—	—	—	—	—	—	—	—	—
Covered-Calendar	2/2	—	—	2/2	—	—	2/2	—	—	2
Days-Old	—	—	—	—	—	—	—	—	—	—
Dictation	—	—	—	—	—	—	—	—	—	—
Duolingo	2/—	—	—/2	2/—	—	—/2	2/—	—	—/2	2
Honey	—	—	—	—	—	—	—	—	—	—
HotelWifiTest	—	—	—	—	—	—	—	—	—	—
Mailinator	—	—	—	—	—	—	—	—	—	—
MidwayMeetup	—	—	—	—	—	—	—	—	—	—
Ninite	2/1	—	—/1	2/1	—	—/1	2/1	—	—/1	2
Pdf-Escape	—	—	—	—	—	—	—	—	—	—
Pepfeed	1/1	—	—	1/1	—	—	1/1	—	—	1
Pocket	—	—	—	—	—	—	—	—	—	—
RainyMood	—	—	—	—	—	—	—	—	—	—
RunPee	—	—	1/1	—	—	1/1	—	—	1/1	1
TopDocumentary	—	—	—	—	—	—	—	—	—	—
UserSearch	1/1	—	—	1/1	—	—	1/1	—	—	1
WhatShouldIReadNext	—	—	—	—	—	—	—	—	—	—
WillMyPhoneWork	—	—	—	—	—	—	—	—	—	—
ZeroDollarMovies	—	—	—	—	—	—	—	—	—	—
Total	13	—	1	13	—	1	13	—	1	14
Agreement with manual	10/10	—	1/4	10/10	—	1/4	10/10	—	1/4	—
Per inspection point		78.6%			78.6%			78.6%		—

ReDeCheck's detection of rows of elements as part of wrapping checks. The improvements that we implemented as part of an enhancement to Verve tool are discussed in Section 4.6.

Conclusion for RQ2: Verve is likely to consistently classify wrapping failures throughout the reported failure range. Therefore, a single point of inspection seems sufficient for automatic classification. Verve achieves 78.6% agreement with the manual analysis for wrapping failures with the initial set of subjects. Further work on both ReDeCheck and Verve is required to improve the analysis of the three cases where the manual analysis disagreed with Verve.

RQ3: Small-range failures We used six histogram comparison methods in our experiment: *Bhattacharyya Distance*, *Chi-square*, *Alternative Chi-square*, *Correlation*, *Intersection* and *Kullback–Leibler divergence*, which make up the full set of measurement methods made available by the OpenCV tool for the Java language [23]. As explained in Section 4.3, we used the ThresholdFinder tool to automatically determine the *prospective* threshold to maximize agreement with the manual classifications for the initial set of subjects above which Verve should report a result as a TP. We used these same values for the additional set of subjects as part of RQ4, which we deliberately excluded from the tuning process. The threshold numbers we obtained using this method are reported as the ϵ values in the table headings against each metric in Table VII.

Table VII reports the results when automatically classifying small-range failures using the horizontal referencing approach with each distance metric, comparing them with manual classification. The highest level of agreement was 97.4% with *Intersection*, which disagrees with manual classification for five RLFs. Of these five, two were TPs and three were FPs that were misclassifications by Verve. Yet, the two RLFs from *Cloudconvert* and *Will My Phone Work* are also duplicate

Table VII. The results from using VERVE featuring six histogram comparison measures for small-range failures of the initial set of web pages with the horizontal referencing approach described in Section 3.3.1. In this table, ϵ denotes the threshold value used for histogram comparison.

	Small-range Horizontal referencing												
	Bhattacharyya distance		**Chi-square**		**Alternative Chi-square**		**Correlation**		**Intersection**		**Kullback−Leibler divergence**		
	$\epsilon = 0.20$		$\epsilon = 0.11$		$\epsilon = 0.14$		$\epsilon = 0$		$\epsilon = 0.09$		$\epsilon = 0.24$		
	TP	FP	TP	FP	TP	FP	TP	FP	TP	FP	TP	FP	**Total**
3-Minute-Journal	—	1/1	—	1/1	—	1/1	—	1/1	—	1/1	—	1/1	1
AccountKiller	137/147	15/5	147/147	5/5	147/147	5/5	128/147	24/5	147/147	5/5	148/147	4/5	152
AirBnb	—	2/2	—	2/2	—	2/2	—	2/2	—	2/2	—	2/2	2
BugMeNot	—	—	—	—	—	—	—	—	—	—	—	—	—
CloudConvert	—/1	1/—	1/1	—	—/1	1/—	—/1	1/—	—/1	1/—	—/1	1/—	1
Consumer-Reports	—	1/1	—	1/1	—	1/1	—	1/1	—	1/1	—	1/1	1
Covered-Calendar	—	—	—	—	—	—	—	—	—	—	—	—	—
Days-Old	—	—	—	—	—	—	—	—	—	—	—	—	—
Dictation	—	—	—	—	—	—	—	—	—	—	—	—	—
Duolingo	—	1/1	—	1/1	—	1/1	—	1/1	—	1/1	—	1/1	1
Honey	1/—	2/3	3/—	—/3	3/—	—/3	3/—	—/3	2/—	1/3	3/—	—/3	3
HotelWifiTest	1/—	1/2	1/—	1/2	2/—	—/2	—	2/2	—	2/2	2/—	—/2	2
Mailinator	—	2/2	—	2/2	—	2/2	—	2/2	—	2/2	—	2/2	2
MidwayMeetup	—	—	—	—	—	—	—	—	—	—	—	—	—
Ninite	—	—	—	—	—	—	—	—	—	—	—	—	—
Pdf-Escape	—	—	—	—	—	—	—	—	—	—	—	—	—
Pepfeed	—/2	16/14	5/2	11/14	5/2	11/14	—/2	16/14	2/2	14/14	5/2	11/14	16
Pocket	—	3/3	—	3/3	—	3/3	—	3/3	—	3/3	—	3/3	3
RainyMood	—	—	—	—	—	—	—	—	—	—	—	—	—
RunPee	1/—	4/5	—	5/5	3/—	2/5	—	5/5	—	5/5	3/—	2/5	5
TopDocumentary	—	2/2	—	2/2	1/—	1/2	—	2/2	—	2/2	—	2/2	2
UserSearch	—	—	—	—	—	—	—	—	—	—	—	—	—
WhatShouldIReadNext	—	—	—	—	—	—	—	—	—	—	—	—	—
WillMyPhoneWork	2/2	—	2/2	—	2/2	—	2/2	—	—/2	2/—	2/2	—	2
ZeroDollarMovies	—	2	—	2	—	2	—	2	—	2	—	2	2
Total	142	53	159	36	163	32	133	62	151	44	163	32	195
Agreement with manual	139/152	40/43	152/152	36/43	151/152	31/43	130/152	40/43	149/152	41/43	151/152	31/43	—
Agreement per measure	91.8%		96.4%		93.3%		87.2%		97.4%		93.3%		—

Table VIII. The results from using VERVE featuring six histogram comparison measures for small-range failures of the initial set of web pages with the horizontal-plus-vertical referencing approach described in Section 3.3.1. In this table, ϵ denotes the threshold value used for histogram comparison.

	Small-range Horizontal-plus-vertical referencing												
	Bhattacharyya distance $\epsilon = 0.23$		**Chi-square** $\epsilon = 0.46$		**Alternative Chi-square** $\epsilon = 0.82$		**Correlation** $\epsilon = 0$		**Intersection** $\epsilon = 0.15$		**Kullback−Leibler divergence** $\epsilon = 1.55$		
	TP	FP	TP	FP	TP	FP	TP	FP	TP	FP	TP	FP	**Total**
3-Minute-Journal	—	1/1	—	1/1	—	1/1	—	1/1	—	1/1	—	1/1	1
AccountKiller	147/147	5/5	147/147	5/5	147/147	5/5	147/147	5/5	147/147	5/5	147/147	5/5	152
AirBnb	—	2/2	—	2/2	—	2/2	—	2/2	—	2/2	—	2/2	2
BugMeNot	—	—	—	—	—	—	—	—	—	—	—	—	—
CloudConvert	—/1	1/—	—/1	1/—	—/1	1/—	—/1	1/—	—/1	1/—	—/1	1/—	1
Consumer-Reports	—	1/1	—	1/1	—	1/1	—	1/1	—	1/1	—	1/1	1
Covered-Calendar	—	—	—	—	—	—	—	—	—	—	—	—	—
Days-Old	—	—	—	—	—	—	—	—	—	—	—	—	—
Dictation	—	—	—	—	—	—	—	—	—	—	—	—	—
Duolingo	—	1/1	—	1/1	—	1/1	—	1/1	—	1/1	—	1/1	1
Honey	—	3/3	2/—	1/3	1/—	2/3	3/—	—/3	—	3/3	—	3/3	3
HotelWifiTest	1/—	1/2	—	2/2	—	2/2	—	2/2	—	2/2	—	2/2	2
Mailinator	—	2/2	—	2/2	—	2/2	—	2/2	—	2/2	—	2/2	2
MidwayMeetup	—	—	—	—	—	—	—	—	—	—	—	—	—
Ninite	—	—	—	—	—	—	—	—	—	—	—	—	—
Pdf-Escape	—	—	—	—	—	—	—	—	—	—	—	—	—
Pepfeed	—/2	16/14	2/2	14/14	—/2	16/14	2/2	14/14	2/2	14/14	—/2	16/14	16
Pocket	—	3/3	—	3/3	—	3/3	—	3/3	—	3/3	—	3/3	3
RainyMood	—	—	—	—	—	—	—	—	—	—	—	—	—
RunPee	—	5/5	—	5/5	—	5/5	—	5/5	—	5/5	—	5/5	5
TopDocumentary	—	2/2	—	2/2	—	2/2	—	2/2	—	2/2	—	2/2	2
UserSearch	—	—	—	—	—	—	—	—	—	—	—	—	—
WhatShouldIReadNext	—	—	—	—	—	—	—	—	—	—	—	—	—
WillMyPhoneWork	2/2	—	—/2	2/—	—/2	2/—	2/2	—	—/2	2/—	1/2	1/—	2
ZeroDollarMovies	—	2	—	2	—	2	—	2	—	2	—	2	2
Total	150	45	151	44	148	47	154	41	149	46	148	47	195
Agreement with manual	149/152	42/43	149/152	41/43	147/152	42/43	151/152	40/43	149/152	43/43	148/152	43/43	—
Agreement per measure	97.9%		97.4%		96.9%		97.9%		98.5%		97.9%		—

reports of element collision failures that VERVE was, in fact, able to successfully classify with its element collision algorithm. Therefore, when small-range failures are also instances of other types of failures, it is better to use VERVE's other classification algorithms specifically tailored for those RLF types. As the results show, those algorithms are more reliable at classifying those RLFs, suggesting that the small-range colour histogram distance method should be reserved for situations when there is no duplicate report instance. Ultimately, this result highlights the benefits of having multiple ways to classify an RLF, as provided by VERVE.

Table VIII shows the results of VERVE for small-range failures using the horizontal-plus-vertical referencing approach, with the thresholds determined specifically for this approach reported as ϵ values in the table. Only a single failure, a duplicate collision report from *Cloudconvert* is misclassified by all six measures. The top performer, *Intersection*, has a 98.5% agreement and a total of three misclassifications. Ranking second with an agreement of 97.9% are the *Bhattacharyya Distance*, *Correlation*, and *Kullback–Leibler Divergence* measures, misclassifying a total of four failures each. Finally, the *Chi-square* measure had an agreement of 97.4%, misclassifying five responsive layout failures. With all measures achieving a high agreement with the manual classification, the lowest was *Alternative Chi-square* at 96.9%, with a total of six misclassifications.

Conclusion for RQ3: When configured to use the horizontal referencing approach, the VERVE tool can detect small-range failures with up to a 97.4% agreement with the manual classification using the *Intersection* distance metric. Moreover, with the horizontal-plus-vertical referencing approach, VERVE is able to detect small-range failures with 98.5% agreement using *Intersection*. These results suggest that *Intersection* is the best histogram comparison measure for use in VERVE.

Table IX. The results from using VERVE for element collision, element protrusion, and viewport protrusion failures of the additional set of web pages after inspecting the 'Minimum' of the reported failure range.

	Minimum									
	Element collision			**Element protrusion**			**Viewport protrusion**			
	TP	NOI	FP	TP	NOI	FP	TP	NOI	FP	**Total**
EatThisMuch	—	5/5	—	1/—	5/6	—	2/1	—/1	—	13
Forvo	—	—	—	—	3/3	—	—	—	—	3
GMapStreetViewPlayer	—	—	—	—	2/2	—	—	—	—	2
HoursOf	—	—	—	—	1/1	—	1/—	—/1	—	2
RetailMeNot	2/2	—	—	—	30/30	—	—	—	—	32
SB-Admin-2	—	—	—	—	—	—	1/1	—	—	1
SB-Agency	—	4/4	—	3/—	5/8	—	1/1	—	—	13
SB-Business-Casual	—	—	—	—	—	—	—	—	—	—
SB-Clean-Blog	—	—	—	—	—	—	—	—	—	—
SB-Coming-Soon	—	—	—	—	—	—	—	—	—	—
SB-Creative	—	—	—	—	—	—	—	—	—	—
SB-Freelancer	—	—	—	—	—	—	—	—	—	—
SB-Grayscale	—	—	—	—	—	—	1/1	—	—	1
SB–Landing-Page	—	—	—	—	—	—	—	—	—	—
SB-New-Age	—	—	—	—	—	—	—	—	—	—
SB-One-Page-Wonder	—	—	—	—	—	—	—	—	—	—
SB–Resume	—	1/1	—	—	—	—	—	—	—	1
SB-Stylish-Portfolio	—	—	—	—	—	—	—	—	—	—
SimilarSites	—	—	—	—	—	—	4/4	—	—	4
Tiiime	—	—	—	—	1/1	—	—	—	—	1
Total	2	10	—	4	47	—	10	—	—	73
Agreement with manual	2/2	10/10	—	—	47/51	—	8/8	0/2	—	—
Agreement per failure type		100%			92.2%			80%		—
Per inspection point					91.8%					—

RQ4: Additional subjects

RQ4(a) Does VERVE *effectively classify collision, element protrusion and viewport protrusion failure reports for the new responsive web pages?* Table IX furnishes the results from running VERVE on the element collision, element protrusion and viewport protrusion failures reported by REDECHECK for the 20 web pages in the additional set of subjects. This table also reports the agreement between the results of VERVE and the manual classification that we performed for the additional set. While this table gives the results from running VERVE at the minimum inspection point of the failure range, Tables X and XI respectively present the results from using the tool at the middle and maximum points. It is important to note that all of the manual classifications from Table III(b) are shown as the denominator of ratio values in all the result tables.

At the minimum and middle points of inspection, there were six failures in non-agreement, while at the maximum, there were an additional three mismatches. The six non-agreements at the minimum and middle inspection points were all automatically classified by VERVE as TPs, whereas in our manual classification we categorized them as NOIs. We conclude that all six are a *misclassification* by VERVE. Two were viewport protrusion failures from *EatThisMuch* and *HoursOf* and had only a few pixels changed that are not visible to the human eye. Three protrusion failures, from *SB-Agency*, were a *misclassification* due to a shortcoming of applying Algorithm 2 on element protrusion and viewport protrusion failures. Since this algorithm both does not make exceptions for minor changes involving a few pixels or the colour of an element and it does not consider other elements that may be involved, it fails for these cases. Specifically, a container with other elements that overlap by design with the reported protruding element may need a more sophisticated approach to visually classify them. We plan, as part of future work, to investigate if additional heuristics would improve or negatively influence, the overall results. Finally, since

Table X. The results from using VERVE for element collision, element protrusion and viewport protrusion failures of the additional set of web pages after inspecting the 'Middle' of the reported failure range.

	Middle									
	Element collision			**Element protrusion**			**Viewport protrusion**			
	TP	NOI	FP	TP	NOI	FP	TP	NOI	FP	**Total**
EatThisMuch	—	5/5	—	1/—	5/6	—	2/1	—/1	—	13
Forvo	—	—	—	—	3/3	—	—	—	—	3
GMapStreetViewPlayer	—	—	—	—	2/2	—	—	—	—	2
HoursOf	—	—	—	—	1/1	—	1/—	—/1	—	2
RetailMeNot	2/2	—	—	—	30/30	—	—	—	—	32
SB-Admin-2	—	—	—	—	—	—	1/1	—	—	1
SB-Agency	—	4/4	—	3/—	5/8	—	1/1	—	—	13
SB-Business-Casual	—	—	—	—	—	—	—	—	—	—
SB-Clean-Blog	—	—	—	—	—	—	—	—	—	—
SB-Coming-Soon	—	—	—	—	—	—	—	—	—	—
SB-Creative	—	—	—	—	—	—	—	—	—	—
SB-Freelancer	—	—	—	—	—	—	—	—	—	—
SB-Grayscale	—	—	—	—	—	—	1/1	—	—	1
SB-Landing-Page	—	—	—	—	—	—	—	—	—	—
SB-New-Age	—	—	—	—	—	—	—	—	—	—
SB-One-Page-Wonder	—	—	—	—	—	—	—	—	—	—
SB-Resume	—	1/1	—	—	—	—	—	—	—	1
SB-Stylish-Portfolio	—	—	—	—	—	—	—	—	—	—
SimilarSites	—	—	—	—	—	—	4/4	—	—	4
Tiiime	—	—	—	—	1/1	—	—	—	—	1
Total	2	10	—	4	47	—	10	—	—	73
Agreement with manual	2/2	10/10	—	—	47/51	—	8/8	0/2	—	—
Agreement per failure type		100%			92.2%			80%		—
Per inspection point					91.8%					—

Table XI. The results from using Verve for element collision, element protrusion and viewport protrusion failures of the additional set of web pages after inspecting the 'Maximum' of the reported failure range.

	Maximum									
	Element collision			**Element protrusion**			**Viewport protrusion**			
	TP	NOI	FP	TP	NOI	FP	TP	NOI	FP	**Total**
EatThisMuch	—	5/5	—	1/—	5/6	—	1/1	1/1	—	13
Forvo	—	—	—	—	3/3	—	—	—	—	3
GMapStreetViewPlayer	—	—	—	—	2/2	—	—	—	—	2
HoursOf	—	—	—	—	1/1	—	1/—	—/1	—	2
RetailMeNot	2/2	—	—	—	30/30	—	—	—	—	32
SB-Admin-2	—	—	—	—	—	—	—/1	1/—	—	1
SB-Agency	—	4/4	—	3/—	5/8	—	1/1	—	—	13
SB-Business-Casual	—	—	—	—	—	—	—	—	—	—
SB-Clean-Blog	—	—	—	—	—	—	—	—	—	—
SB-Coming-Soon	—	—	—	—	—	—	—	—	—	—
SB—Creative	—	—	—	—	—	—	—	—	—	—
SB-Freelancer	—	—	—	—	—	—	—	—	—	—
SB-Grayscale	—	—	—	—	—	—	—/1	1/—	—	1
SB-Landing-Page	—	—	—	—	—	—	—	—	—	—
SB-New-Age	—	—	—	—	—	—	—	—	—	—
SB-One-Page-Wonder	—	—	—	—	—	—	—	—	—	—
SB-Resume	—	1/1	—	—	—	—	—	—	—	1
SB-Stylish-Portfolio	—	—	—	—	—	—	—	—	—	—
SimilarSites	—	—	—	—	—	—	4/4	—	—	4
Tiiime	—	—	—	—	1/1	—	—	—	—	1
Total	2	10	—	4	47	—	7	3	—	73
Agreement with manual	2/2	10/10	—	—	47/51	—	5/8	0/2	—	—
Agreement per failure type		100%			92.2%			50%		—
Per inspection point					87.7%					—

EatThisMuch contains an HTML element that is floating instead of protruding, Verve's visual approach cannot correctly classify it as an FP.

For the maximum inspection point, there were three viewport failures in non-agreement with the manual classification. Verve classified these failures from *SB-Admin-2*, *EatThisMuch* and *SB-Grayscale* as NOI while the manual classification was TP. These failures are more visible at the minimum point of inspection and our findings from RQ1b are evident here. Figure 1 shows snapshots of the viewport failure from *SB-Admin-2*, illustrating how the failure is observable (TP) at the narrower inspection points and becomes non-observable (NOI) at a wider inspection point.

Conclusion for RQ4(a): For the element collision, element protrusion and viewport protrusion failures reported by ReDeCheck on the 20 pages in the additional set of subjects, Verve's automatic classification frequently matched the manual classification. Notably, it achieved 91.8% agreement with the manual approach, an increase from 86.3% with the initial set of subjects.

RQ4(b) *Does* Verve *effectively classify wrapping failure reports for the new responsive web pages?* All 17 wrapping failures reported for the additional 20 web pages were reported consistently by Verve at all three of the inspections points (i.e. minimum, middle and maximum), as shown by Table XII. This result supports the intuition that a single inspection point is sufficient for correctly classifying the wrapping failures. However, 11 of the 17 results did not agree with our manual classification. Although we manually classified them as FPs, Verve automatically classified them as TPs. Of the 11 non-agreements, four failures were never observed to form a row, although one element is visually below the others. Again, this is an issue with ReDeCheck that Verve cannot address based on visual information alone. However, Section 4.6 notes that we enhanced Verve so that it can also work at the DOM level, thereby overcoming this issue and improving these results.

Table XII. Using Verve at three inspection points on wrapping failures for the additional set of web pages.

	Wrapping									
	Minimum			**Middle**			**Maximum**			
	TP	NOI	FP	TP	NOI	FP	TP	NOI	FP	**Total**
EatThisMuch	1/—	—	—/1	1/—	—	—/1	1/—	—	—/1	1
Forvo	4/2	—	—/2	4/2	—	—/2	4/2	—	—/2	4
GMapStreetViewPlayer	—	—	—	—	—	—	—	—	—	—
HoursOf	—	—	—	—	—	—	—	—	—	—
RetailMeNot	4/—	2/2	—/4	4/—	2/2	—/4	4/—	2/2	—/4	6
SB-Admin-2	—	—	—	—	—	—	—	—	—	—
SB-Agency	3/—	—	—/3	3/—	—	—/3	3/—	—	—/3	3
SB-Business-Casual	—	—	—	—	—	—	—	—	—	—
SB-Clean-Blog	—	—	—	—	—	—	—	—	—	—
SB-Coming-Soon	—	—	—	—	—	—	—	—	—	—
SB-Creative	—	—	—	—	—	—	—	—	—	—
SB-Freelancer	—	—	—	—	—	—	—	—	—	—
SB-Grayscale	—	—	—	—	—	—	—	—	—	—
SB-Landing-Page	1/—	—	—/1	1/—	—	—/1	1/—	—	—/1	1
SB-New-Age	—	—	—	—	—	—	—	—	—	—
SB-One-Page-Wonder	—	—	—	—	—	—	—	—	—	—
SB-Resume	2/2	—	—	2/2	—	—	2/2	—	—	2
SB-Stylish-Portfolio	—	—	—	—	—	—	—	—	—	—
SimilarSites	—	—	—	—	—	—	—	—	—	—
Tiiime	—	—	—	—	—	—	—	—	—	—
Total	15	2	—	15	2	—	15	2	—	17
Agreement with manual	4/4	2/2	0/11	4/4	2/2	0/11	4/4	2/2	0/11	—
Per inspection point		35.3%			35.3%			35.3%		—

In the remainder of the response to this research question, we investigate the sources of the non-agreement with the manual classification. Notably, three of them were from *RetailMeNot*, and one was from *EatThisMuch*. Another two failures from *Forvo* and one more from *SB-Landing-Page* reported textual links that do, in fact, visually wrap but should not be classified as a failure, again suggesting that ReDeCheck's wrapping detection algorithm can be improved. An additional wrapping failure from *RetailMeNot* surfaces another reason for improving ReDeCheck. For this subject, the row-aligned elements do, in fact, change position—but this seems to be intended behaviour since there are developer-defined rules that make one of these elements no longer visible when the page is resized. Finally, three failures from *SB-Agency* report the wrapping of a social media icon from a set of three circular icons. Even though the awkward wrapping of icons would normally be a layout failure, this one includes three icons that wrap in an aesthetically pleasing fashion that leaves the layout largely unchanged, leading us to conclude that these are *subjective*.

Conclusion for RQ4(b): For wrapping failures, the agreement between the manual classification and the one reported by Verve dropped from 78.6% to 35.3% when moving from the initial to additional subjects. Yet, the root cause of this decrease in effectiveness seems to be inadequate detection by the ReDeCheck tool, an issue which we explain how to address in Section 4.6.

RQ4(c)*Does* Verve *effectively classify small-range failure reports for the new responsive web pages?*ReDeCheck reported a total of 53 small-range failures from the additional set that were all FP failures, suggesting the need for improvements to ReDeCheck's small-range detector. Therefore, to properly evaluate Verve on true positive reports, we manually injected small-range failures, creating another set of small-range reports raised by ReDeCheck. Our fault injection procedure, as described in Section 4.3, led to ReDeCheck detecting 96 further small-range failures. We manually classified all of these RLFs as TPs except for a single failure that did not pertain to the target element, which was an FP. This failure from *HoursOf* had an element positioned in the middle of its container outside the failure range, while inside the range was near the middle but not identified as such. Hence, that failure was an FP since no visual disturbance was observed.

First, we investigated the results of the additional set for subjects for which we did not inject failures. Table XIII gives the results from using VERVE on the small-range failures reported for the additional set of web pages using horizontal referencing. This table shows that *Correlation*obtained the highest level of agreement with a 94.3% match, in contrast to the conclusion for RQ3 where we found that *Intersection* was the best performer for the initial subjects. Yet, for the additional set of subjects, *Intersection* came in second place with 67.9% agreement, misclassifying 17 of the 53 failures as TPs. Moreover, all of these 17 were also misclassified using *Bhattacharyya Distance*, *Chi-square*, *Alternative Chi-square*, and *Kullback–Leibler Divergence*. Two of the 17 were also misclassified by *Correlation*. A closer examination of these 17 non-agreements revealed that 12 were reporting a single textual link that changed position in reference to other textual links and in reference of the container. Essentially, this change in position is a wrapping of the textual link that only happens for a small-range of four viewport widths. The 12 TPs misclassified by *Intersection* (as well as lower ranking measures) are perhaps best categorized as *subjective* while the additional four are a *misclassification* on the part of VERVE.

Table XIV furnishes the results from applying VERVE with the horizontal-plus-vertical referencing approach to the additional set of web pages. Consistent with our findings in RQ3, *Intersection*ranked the highest for this set of web pages, with 73.6% agreement with the manual classifications, misclassifying 14 out of 53 reported failures. All other measures similarly were in non-agreement, with the same 14 failures as the top performer. The next two measures jointly ranked second, *Alternative Chi-square* and *Correlation*, had a 71.7% agreement. Each misclassified one failure more than *Intersection*. The next jointly ranked measures, *Chi-square*and *Kullback–Leibler Divergence*, had a 66% agreement and misclassified a total of 18 failures. Finally, the lowest ranked measure, *Bhattacharyya Distance*, had a difference of six misclassifications compared to *Intersection*, with only a 60.4% agreement with the manual classifications.

In the remainder of the response to this research question, we investigate the set of small-range reports associated with the injected failures. Example snapshots of the *SB-Business-Casual*subject, featuring an injected small-range failure, are shown by Figure 11. This failure occurs at the range of viewport widths of 992–995 pixels, where the first author originally injected the fault. Parts (a) and (c) show the web page rendered at the 991 and 996 pixel viewport widths, respectively. These are the comparison viewports at either side of the failure viewport. At the narrower viewport of 991-pixels wide, shown in Figure 11(a), an image element of a barista in a coffee house is positioned above a block of text with a white background. At the wider viewport of 996 pixels, these two elements are overlapping by design. Part (b) of the figure shows the failure at the viewport width of 992 pixels, with these two elements positioned anomalously. Additional, unrelated changes to the layout start at the 992-pixel viewport width and remain as the viewport is widened. These include the header of the page where an expanded menu becomes part of the design, which also uses a larger logo. VERVE was able to correctly classify the failure as a TP using both the horizontal referencing approach and the horizontal-plus-vertical referencing approaches using all six different distance metrics.

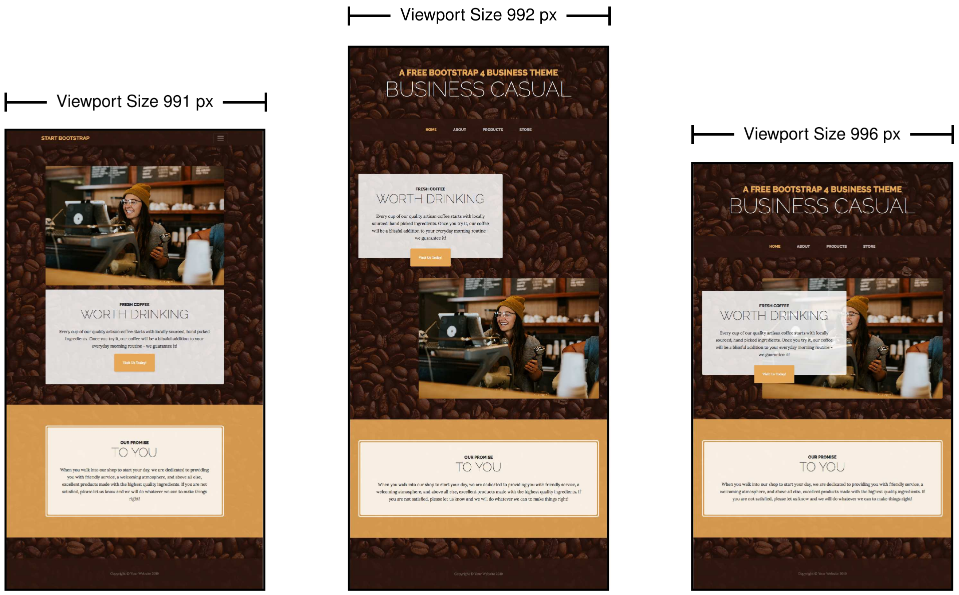
Table XV presents the results from applying VERVE to the set of synthetic faults using the horizontal referencing approach. Our analysis revealed that 27 failures were misclassified by VERVE as FPs for all six measures. The top performing measure, *Chi-square*, with 67.7% agreement, had an additional four misclassifications. A close second was *Alternative Chi-square*, which achieved 64.6% agreement, misclassifying three more than *Chi-square*. The *Kullback–Leibler Divergence* measure came in third with only 57.3% agreement. In fourth place, with only a 43.8% agreement, are *Bhattacharyya Distance* and *Correlation*. Finally, the poorest performing measure was *Intersection*, at 29.2% agreement. This result contradicts the *Intersection* measure's first place performance with the initial set and its position as the second ranked in the non-fault-injected additional set of web pages. We discuss these apparent contradictions in Section 4.6.

Table XIII. The results from using VERVE, featuring six histogram comparison measures for small-range failures of the additional set of web pages with the horizontal referencing approach. In this table, ϵ denotes the threshold value used for histogram comparison.

	Small-range Horizontal referencing												
	Bhattacharyya distance $\epsilon = 0.20$		**Chi-square** $\epsilon = 0.11$		**Alternative Chi-square** $\epsilon = 0.14$		**Correlation** $\epsilon = 0$		**Intersection** $\epsilon = 0.09$		**Kullback–Leibler divergence** $\epsilon = 0.24$		
	TP	FP	TP	FP	TP	FP	TP	FP	TP	FP	TP	FP	**Total**
EatThisMuch	—	2/2	—	2/2	—	2/2	—	2/2	—	2/2	—	2/2	2
Forvo	16/—	13/29	25/—	4/29	26/—	3/29	—	29/29	13/—	16/29	15/—	14/29	29
GMapStreetViewPlayer	—	—	—	—	—	—	—	—	—	—	—	—	—
HoursOf	—	—	—	—	—	—	—	—	—	—	—	—	—
RetailMeNot	5/—	10/15	6/—	9/15	9/—	6/15	3/—	12/15	4/—	11/15	12/—	3/15	15
SB-Admin-2	1/—	—/1	1/—	—/1	1/—	—/1	—	1/1	—	1/1	1/—	—/1	1
SB-Agency	—	—	—	—	—	—	—	—	—	—	—	—	—
SB-Business-Casual	—	—	—	—	—	—	—	—	—	—	—	—	—
SB-Clean-Blog	—	—	—	—	—	—	—	—	—	—	—	—	—
SB-Coming-Soon	—	—	—	—	—	—	—	—	—	—	—	—	—
SB-Creative	—	—	—	—	—	—	—	—	—	—	—	—	—
SB-Freelancer	—	—	—	—	—	—	—	—	—	—	—	—	—
SB-Grayscale	—	—	—	—	—	—	—	—	—	—	—	—	—
SB-Landing-Page	—	—	—	—	—	—	—	—	—	—	—	—	—
SB-New-Age	—	—	—	—	—	—	—	—	—	—	—	—	—
SB-One-Page-Wonder	—	—	—	—	—	—	—	—	—	—	—	—	—
SB-Resume	—	—	—	—	—	—	—	—	—	—	—	—	—
SB-Stylish-Portfolio	—	4/4	2/—	2/4	3/—	1/4	—	4/4	—	4/4	3/—	1/4	4
SimilarSites	—	1/1	—	1/1	—	1/1	—	1/1	—	1/1	—	1/1	1
Tiiime	—	1	—	1	—	1	—	1	—	1	—	1	1
Total	22	31	34	19	39	14	3	50	17	36	31	22	53
Agreement with manual	—	31/53	—	19/53	—	14/53	—	50/53	—	36/53	—	22/53	—
Agreement per measure	58.5%		35.8%		26.4%		94.3%		67.9%		41.5%		—

Table XIV. The results from using VERVE, featuring six histogram comparison measures for small-range failures of the additional set of web pages with the horizontal-plus-vertical referencing approach. In this table, ϵ denotes the threshold value used for histogram comparison.

	Small-range Horizontal-plus-vertical referencing												
	Bhattacharyya distance $\epsilon = 0.23$		**Chi-square** $\epsilon = 0.46$		**Alternative Chi-square** $\epsilon = 0.82$		**Correlation** $\epsilon = 0$		**Intersection** $\epsilon = 0.15$		**Kullback−Leibler divergence** $\epsilon = 1.55$		
	TP	FP	TP	FP	TP	FP	TP	FP	TP	FP	TP	FP	**Total**
EatThisMuch	—	2/2	—	2/2	—	2/2	—	2/2	—	2/2	—	2/2	2
Forvo	16/—	13/29	12/—	17/29	12/—	17/29	12/—	17/29	12/—	17/29	14/—	15/29	29
GMapStreetViewPlayer	—	—	—	—	—	—	—	—	—	—	—	—	—
HoursOf	—	—	—	—	—	—	—	—	—	—	—	—	—
RetailMeNot	4/—	11/15	2/—	13/15	2/—	13/15	3/—	12/15	2/—	13/15	4/—	11/15	15
SB-Admin-2	1/—	—/1	—	1/1	—	1/1	—	1/1	—	1/1	—	1/1	1
SB-Agency	—	—	—	—	—	—	—	—	—	—	—	—	—
SB-Business-Casual	—	—	—	—	—	—	—	—	—	—	—	—	—
SB-Clean-Blog	—	—	—	—	—	—	—	—	—	—	—	—	—
SB-Coming-Soon	—	—	—	—	—	—	—	—	—	—	—	—	—
SB-Creative	—	—	—	—	—	—	—	—	—	—	—	—	—
SB-Freelancer	—	—	—	—	—	—	—	—	—	—	—	—	—
SB-Grayscale	—	—	—	—	—	—	—	—	—	—	—	—	—
SB-Landing-Page	—	—	—	—	—	—	—	—	—	—	—	—	—
SB-New-Age	—	—	—	—	—	—	—	—	—	—	—	—	—
SB-One-Page-Wonder	—	—	—	—	—	—	—	—	—	—	—	—	—
SB-Resume	—	—	—	—	—	—	—	—	—	—	—	—	—
SB-Stylish-Portfolio	—	4/4	4/—	—/4	1/—	3/4	—	4/4	—	4/4	—	4/4	4
SimilarSites	—	1/1	—	1/1	—	1/1	—	1/1	—	1/1	—	1/1	1
Tiiime	—	1	—	1	—	1	—	1	—	1	—	1	1
Total	21	32	18	35	15	38	15	38	14	39	18	35	53
Agreement with manual	—	32/53	—	35/53	—	38/53	—	38/53	—	39/53	—	35/53	—
Agreement per measure	60.4%		66%		71.7%		71.7%		73.6%		66%		—



(A) Narrower Comparison Viewport (B) Small-range Layout Failure (C) Wider Comparison Viewport

Figure 11. Three snapshots of the *SB-Business-Casual* web page that capture its layout before a small-range failure occurs in (a), and a small-range failure with the range of 992–995 pixels in (b), and after the layout failure in (c), as reported by the ReDeCheck and correctly classified, without human intervention, as a true positive by the Verve tool using all six metrics and both approaches

Conclusion for RQ4(c): For small-range failures using the horizontal referencing approach, *Correlation* obtained the highest level of agreement, matching 94.3% of the time with our—notably all FP—manual classification of the additional set of web pages. However, using the horizontal-plus-vertical referencing approach, Verve had the highest agreement of 73.6% when using the *Intersection* measure. After injecting faults into the additional set, resulting in 95 TPs and 1 FP, the horizontal referencing approach achieved as high as 67.7% using *Chi-square*. Moreover, *Chi-square* achieved the highest agreement at 79.2% using the horizontal-plus-vertical referencing approach. Using the combined set of all web pages, Section 4.6 further explores the best performing comparison measure and small-range classification approach, deepening the conclusion for this research question and ultimately highlighting evidence suggesting that the Verve tool should, by default, use the horizontal-plus-vertical referencing approach with the *Chi-square* metric.

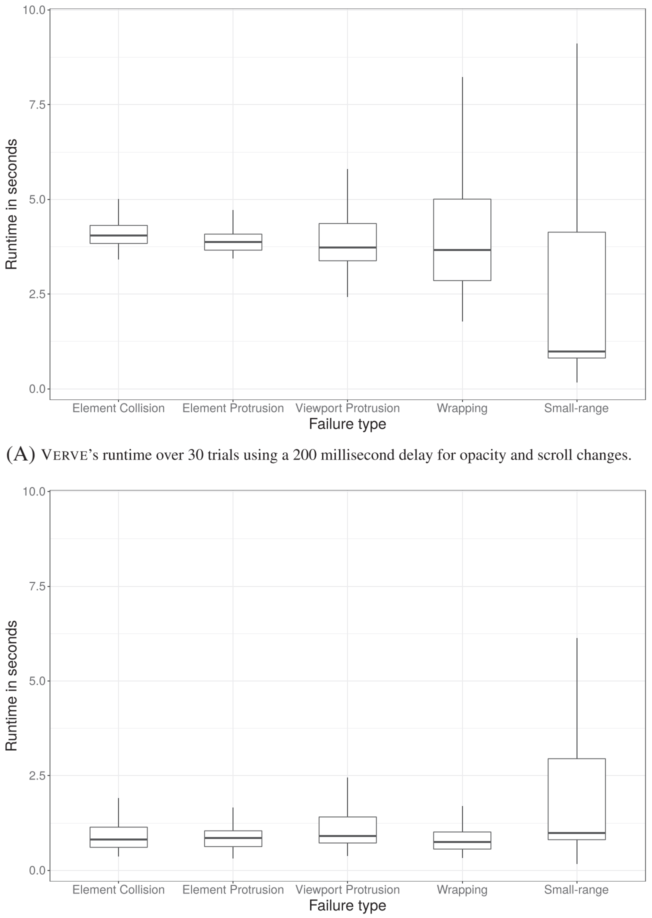
RQ5: Tool efficiency Verve took 4.08 s, on average, to automatically classify each responsive failure with a median value of 3.57 s. Figure 12(a) furnishes a box plot of the runtime for each type of failure with the 200-ms delay that Verve includes to allow web pages to 'settle' in layout following loading and resizing (for a more detailed discussion of this reasoning behind adding this delay, see the methodology for this research question in Section 4.2). The runtime for the element collision, element protrusion, viewport protrusion and wrapping failures had similar medians/means of 4.05/5.25, 3.88/3.98, 3.73/4.01 and 3.67/4.21 s, respectively, while small-range failures had a median value of 0.99 and a mean value of 3.92 s. As shown in Section 3, the algorithm for classifying this type of responsive layout failure is fundamentally different from the other four, and the main reason for the difference is the need to snapshot multiple AOCs across different viewports. To understand these execution times, it is important to note that we designed Verve to reuse snapshots where possible. The main reason for this is that many failures share the same viewports required for analysis. Therefore, avoiding the recapture of the same viewport can reduce the number of scrolling delays required. To bring this into perspective, the *Accountkiller* web page from the initial set of subjects had 147 small-range failure reports in the range of 476–480 pixels wide. Even though this causes a longer execution time for the first failure to request the snapshots

Table XV. The results from using VERVE, featuring six histogram comparison measures for small-range failures of the fault-injected additional set of web pages with the horizontal referencing approach explained in Section 3.3.1. In this table, ϵ denotes the threshold value used for histogram comparison.

	Small-range horizontal referencing												
	Bhattacharyya distance $\epsilon = 0.20$		**Chi-square** $\epsilon = 0.11$		**Alternative Chi-square** $\epsilon = 0.14$		**Correlation** $\epsilon = 0$		**Intersection** $\epsilon = 0.09$		**Kullback−Leibler divergence** $\epsilon = 0.24$		
	TP	FP	TP	FP	TP	FP	TP	FP	TP	FP	TP	FP	**Total**
EatThisMuch	3/3	—	3/3	—	3/3	—	3/3	—	3/3	—	3/3	—	3
Forvo	—/9	9/—	5/9	4/—	5/9	4/—	—/9	9/—	—/9	9/—	—/9	9/—	9
GMapStreetViewPlayer	4/4	—	4/4	—	4/4	—	2/4	2/—	4/4	—	4/4	—	4
HoursOf	—/2	3/1	—/2	3/1	—/2	3/1	—/2	3/1	—/2	3/1	—/2	3/1	3
RetailMeNot	—/8	8/—	—/8	8/—	—/8	8/—	—/8	8/—	—/8	8/—	—/8	8/—	8
SB-Admin-2	12/12	—	12/12	—	12/12	—	12/12	—	12/12	—	12/12	—	12
SB—Agency	5/5	—	5/5	—	5/5	—	5/5	—	—/5	5/—	5/5	—	5
SB-Business-Casual	1/1	—	1/1	—	1/1	—	1/1	—	1/1	—	1/1	—	1
SB-Clean-Blog	—/5	5/—	3/5	2/—	3/5	2/—	3/5	2/—	—/5	5/—	—/5	5/—	5
SB-Coming-Soon	3/3	—	3/3	—	3/3	—	3/3	—	3/3	—	3/3	—	3
SB-Creative	—/6	6/—	—/6	6/—	—/6	6/—	—/6	6/—	—/6	6/—	4/6	2/—	6
SB-Freelancer	7/7	—	7/7	—	7/7	—	7/7	—	—/7	7/—	7/7	—	7
SB-Grayscale	—/5	5/—	—/5	5/—	—/5	5/—	—/5	5/—	—/5	5/—	—/5	5/—	5
SB-Landing-Page	2/4	2/—	2/4	2/—	2/4	2/—	—/4	4/—	—/4	4/—	—/4	4/—	4
SB-New-Age	2/2	—	2/2	—	2/2	—	—/2	2/—	2/2	—	2/2	—	2
SB-One-Page-Wonder	—/3	3/—	3/3	—	3/3	—	—/3	3/—	—/3	3/—	2/3	1/—	3
SB-Resume	1/4	3/—	4/4	—	4/4	—	4/4	—	1/4	3/—	1/4	3/—	4
SB-Stylish-Portfolio	—/5	5/—	3/5	2/—	—/5	5/—	—/5	5/—	—/5	5/—	3/5	2/—	5
SimilarSites	—/6	6/—	6/6	—	6/6	—	—/6	6/—	—/6	6/—	6/6	—	6
Tiiime	1	—	1	—	1	—	1	—	1	—	1	—	1
Total	41	55	64	32	61	35	41	55	27	69	54	42	96
Agreement with manual	41/95	1/1	64/95	1/1	61/95	1/1	41/95	1/1	27/95	1/1	54/95	1/1	—
Agreement per failure type	43.8%		67.7%		64.6%		43.8%		29.2%		57.3%		—

Table XVI. The results from using VERVE, featuring six histogram comparison measures for small-range failures of the fault-injected additional set of pages with the horizontal-plus-vertical referencing approach explained in Section 3.3.1. In this table, ϵ denotes the threshold value used for histogram comparison.

	Small-range Horizontal-plus-vertical referencing												
	Bhattacharyya distance $\epsilon = 0.23$		**Chi-square** $\epsilon = 0.46$		**Alternative Chi-square** $\epsilon = 0.82$		**Correlation** $\epsilon = 0$		**Intersection** $\epsilon = 0.15$		**Kullback−Leibler divergence** $\epsilon = 1.55$		
	TP	FP	TP	FP	TP	FP	TP	FP	TP	FP	TP	FP	**Total**
EatThisMuch	3/3	—	3/3	—	—/3	3/—	3/3	—	3/3	—	3/3	—	3
Forvo	5/9	4/—	7/9	2/—	—/9	9/—	9/9	—	9/9	—	5/9	4/—	9
GMapStreetViewPlayer	2/4	2/—	—/4	4/—	—/4	4/—	2/4	2/—	2/4	2/—	3/4	1/—	4
HoursOf	—/2	3/1	—/2	3/1	—/2	3/1	—/2	3/1	—/2	3/1	—/2	3/1	3
RetailMeNot	6/8	2/—	6/8	2/—	—/8	8/—	—/8	8/—	—/8	8/—	—/8	8/—	8
SB-Admin-2	12/12	—	12/12	—	12/12	—	12/12	—	12/12	—	12/12	—	12
SB-Agency	5/5	—	5/5	—	5/5	—	5/5	—	5/5	—	5/5	—	5
SB-Business-Casual	1/1	—	1/1	—	1/1	—	1/1	—	1/1	—	1/1	—	1
SB-Clean-Blog	3/5	2/—	4/5	1/—	—/5	5/—	4/5	1/—	—/5	5/—	—/5	5/—	5
SB-Coming-Soon	3/3	—	3/3	—	3/3	—	3/3	—	3/3	—	3/3	—	3
SB-Creative	4/6	2/—	4/6	2/—	4/6	2/—	6/6	—	4/6	2/—	6/6	—	6
SB-Freelancer	7/7	—	7/7	—	7/7	—	7/7	—	5/7	2/—	6/7	1/—	7
SB-Grayscale	—/5	5/—	—/5	5/—	—/5	5/—	—/5	5/—	—/5	5/—	—/5	5/—	5
SB-Landing-Page	2/4	2/—	2/4	2/—	2/4	2/—	2/4	2/—	2/4	2/—	—/4	4/—	4
SB-New-Age	2/2	—	2/2	—	2/2	—	1/2	1/—	2/2	—	2/2	—	2
SB-One-Page-Wonder	2/3	1/—	3/3	—	3/3	—	—/3	3/—	—/3	3/—	—/3	3/—	3
SB-Resume	1/4	3/—	4/4	—	—/4	4/—	4/4	—	—/4	4/—	1/4	3/—	4
SB-Stylish-Portfolio	3/5	2/—	5/5	—	3/5	2/—	5/5	—	3/5	2/—	3/5	2/—	5
SimilarSites	—/6	6/—	6/6	—	—/6	6/—	—/6	6/—	—/6	6/—	—/6	6/—	6
Tiiime	1	—	1	—	1	—	1	—	1	—	1	—	1
Total	62	34	75	21	43	53	65	31	52	44	51	45	96
Agreement with manual	62/95	1/1	75/95	1/1	43/95	1/1	65/95	1/1	52/95	1/1	51/95	1/1	—
Agreement per failure type	65.6%		79.2%		45.8%		68.8%		55.2%		54.2%		—



(A) VERVE's runtime over 30 trials using a 200 millisecond delay for opacity and scroll changes.

(B) VERVE's runtime over 30 trials using no added delay.

Figure 12. VERVE's execution time in seconds across all of the 469 presentation failures and 30 trials using the horizontal-plus-vertical referencing approach. In these box plots, the bottom and top whiskers show the minimum and maximum data values excluding outliers, while the box itself represents the interquartile range and the bold middle line represents the median value

from this range, the time to retrieve a failure-specific AOC from the saved snapshots and the time to calculate all six histogram metrics are within the processing time of the relevant failure.

Table XVI presents the results from applying VERVE to the fault-injected web pages from the additional set while using one of the six histogram comparison measures and the horizontal-plus-vertical referencing approach. All six of the measures were in non-agreement for 12 out of the 96 failures in this set. The top performer for this set is *Chi-square* with a 79.2% agreement and a total of 20 failures in non-agreement. In second the place, *Correlation* has 68.8% agreement with 30 misclassifications. This comparison measure is followed by *Bhattacharyya Distance* with 65.6%, *Intersection* with 55.2%, *Kullback–Leibler Divergence* with 54.2% and *Alternative Chi-square* with 45.8% agreement. The most interesting result is that *Intersection* comes in fourth place even though it is the top performer for the subjects in the initial set and the non-fault-injected additional set.

Although the execution time of VERVE is normally stable, there are some subjects for which certain trials have execution times that are far outliers from the median value. For instance, when VERVE was configured to use the added delay setting, there was one trial for which it ran for 795.4 s when classifying an element collision failure reported for the *TopDocumentary* subject. Importantly, this was the only outlier in execution time across the 30 trials, as evident by the fact that the execution time for the other 29 trials was between 4.01 and 4.58 s. To ensure that the performance trends in the graphs are not skewed by these rare outliers that are likely due to the performance characteristics of the execution environment, Figure 12 presents box plots that elide the values that are far outliers. To exclude the outliers from the plots without removing any data points, we used the default option of the *ggplot2* package in the R language for statistical computation.

To further investigate the runtime of VERVE without the influence of the aforementioned delay, we temporarily disabled it and re-ran the timing experiments. Using this no-delay configuration, VERVE took, on average, 2.24 s to classify a reported RLF with a median value of 0.91 s. Notably, without the delay for opacity or scrolling, VERVE's performance is better on average. The opacity delay is used to ensure that the element is fully opaque or transparent when required. On the other hand, scrolling is used to move the visible portion of the page, dictated by the viewport height and width, to an AOC in order to capture a snapshot. The larger the AOC, in proportion to the viewport, the more scrolling is required to capture the images needed for analysis by VERVE and thus the greater number of delays that are needed in the default configuration of the tool, further explaining the noticeable decrease in average execution time in this setting.

Closer inspection of VERVE's automated classification results without the delay revealed good reasons for incorporating it in the first instance. We found that without the delay, VERVE classified three failures from *BugMeNot*, *Ninite* and *RetailMeNot* differently. This was for two reasons. First, VERVE's command to change an element's opacity is sometimes not fully effected through Selenium and the web browser before a snapshot is taken, resulting in an incorrect image and a subsequent inaccurate analysis. The second reason is that the DOM rectangle coordinates retrieved may vary if the position of the element has not 'settled' due to some JavaScript-programmed transitional effect.

Figure 12(b) furnishes a box plot of the runtime for each type of failure with no added delay. For element collision, element protrusion, viewport protrusion and wrapping failures, the median/mean values were 0.82/1.10, 0.85/1.01, 0.90/1.22 and 0.75/0.89 s, respectively. For small-range RLFs, the median was 0.99, and the mean was 3.29 s. Similar to the runs of VERVE with the added delay, the most notable difference between all five of the failure types is the runtime of small-range failures, which is fundamentally different than other approaches. Furthermore, the median value of 0.99 s for small-range remained the same with and without the added delay while the mean was largely un-effected by the removal of the delay. Noting that both (a) and (b) of the figure are using the same scale, the improvements to the runtime gained by removing the delay can be observed as the box plots are lower in the time scale and are generally more compact.

Conclusion for RQ5: On average, VERVE takes a mean of 4.08 s to classify an RLF. This includes the use of a 200-ms delay to allow web page elements to load and 'settle' into position. Without this delay, misclassifications are more likely. These results indicate that VERVE is practical, requiring developers to wait a very short amount of time for its classification results.

4.6. Discussion

The results from the empirical study with the initial set of subjects suggest that VERVE is a good automated alternative to the manual classification of reported presentation failures. Even though this finding carried into the additional set of subjects, there are open points for discussion and opportunities for improving VERVE, which this subsection examines in greater detail.

Performance of small-range classification approaches: The two ways that VERVE classifies small-range failures, namely, horizontal referencing and horizontal-plus-vertical referencing, ranked the colour histogram comparison measures differently across the initial set, additional set and the additional set with the manually injected faults. For the horizontal referencing approach, the top performing measure of the initial set was *Intersection* at 97.4% while the additional set ranked *Correlation* at 94.3%. Finally, the fault injected set of subjects ranked *Chi-square* as the top performer at 67.7%. For the horizontal-plus-vertical referencing approach, the top performing measure of the initial set was *Intersection* at 98.5% with all other comparison measures being close contenders. Furthermore, *Intersection* emerged at the top of the additional set at 73.6% agreement. Finally, in the fault-injected additional set *Chi-square* emerged as the top performer at 79.2%.

These results show that the isolated analysis of the three sets of subjects used in the study does not provide a clear picture of the superior small-range classification approach nor the superior histogram metric. Moreover, the previous section cannot determine if an alternative threshold would have performed better. To better elucidate these trade-offs, Table XVII brings together the findings of each set and aggregates the agreement for all pages. The table also furnishes both the *prospective* thresholds used in our experiments and the *retrospective* thresholds used to weigh in on possible agreement improvements. The overall top performing measure for all web pages using the *prospective* threshold was *Chi-square* for both the horizontal referencing approach and the horizontal-plus-vertical referencing approach. While the horizontal referencing approach achieved 79.1% agreement, the better performing approach, horizontal-plus-vertical referencing, achieved 87.5% agreement. Yet again, using the *retrospective* thresholds with horizontal referencing, *Chi-square* was the top performer with an agreement of 80.8%. As for the better performing approach, horizontal-plus-vertical referencing, it had a 87.8% agreement using the *Intersection* measure, with *Chi-square* being a close second as a top performer using both approaches. These results suggest that, by default, VERVE should use horizontal-plus-vertical referencing with *Chi-square*.

DOM-based improvement for classifying wrapping failures: The most notable drop in effectiveness between the initial and the additional set of subjects are those for the wrapping failures. Compared to the manual classifications, VERVE had 78.6% using the initial set while with the additional subjects had 35.3% agreement. The disagreements are due to failures that are actually visually acceptable—or false positives due to a flaw in ReDECHECK's algorithm that causes it to detect rows improperly.

A visually acceptable wrapping includes a text-based element in the footer of the page that wraps under other text-based elements or a wrapping of icons that takes on aesthetically pleasing formations when wrapped and does not harm the functionality of the web page (e.g. a centre-aligned row of three icons, where an element wraps, but the three icons are arranged into a triangle formation).

Improper row detection is caused by ReDECHECK incorrectly detecting that elements were horizontally aligned in a row, or flagging wrapping failures based on pairwise comparisons of the positions of individual elements. This is a method that is often too sensitive since it does not check, for example, that an element has wrapped beneath all other elements.

To overcome these issues, we updated VERVE's DOM-filter phase to ensure that (i) there are at least three elements in the original ‘row’ (as per the original ReDECHECK algorithm [6]); (ii) that the wrapped element is below all other row elements, and not just the last row element; and (iii) that all row elements are visible with a width and height greater than zero. Table XVIII shows the results when VERVE uses the upgraded DOM-filter phase, revealing an improvement in VERVE's agreement with the manual classification from 35.3% to 64.7% with the additional set of web pages.

Questioning observability: Whether an RLF is observable or not may not always be obvious, making final decisions, particularly manual classifications, somewhat subjective. Essentially, the task requires an observer to recognize a difference between what is visually expected and what is visually apparent. We found that the previously published manual classifications due to Walsh et al. [6] that we used in this paper were subject to some ‘exemptions’ based on the severity of a change. For instance, consider an element *A* that is overlapping the coordinates of an element *B*, with *n* pixels of element *A* overlapping *n* pixels of *B*. In this case, a human would decide whether the *n* pixels of overlap are negligible and if the overall aesthetics remain satisfactory. Both of these criteria are not easily defined and remain, to a great extent, subjective. Nevertheless, we aim to study them in future work. For example, it may be useful to measure the number of changed pixels, determine if a colour change is visible to the human eye, or introduce heuristics concerning AOC size.

It is also worth noting that the previously published manual classifications used in our experiments exhibit some biases that we were not aware of prior to using them. After a deeper examination, we found that some classifications were neither confined to the type of failure nor the XPaths reported. For instance, an element was reported as protruding out of its ancestor element, which was an NOI, but was manually classified as a TP because it was also protruding out of the parent element. We considered, therefore, reclassifying some of the RLFs. Although this would have been justifiable, we refrained from ‘tampering’ with the benchmark data in this way so as to not introduce any further sources of bias. Moreover, reclassification would not tackle the underlying subjective nature that is inherent in any manual classification of responsive layout failures in web pages.

Table XVII. The results from using VERVE's two alternative approaches to small-range classification, the horizontal referencing approach and the horizontal-plus-vertical referencing approach. This table re-presents the results from applying the *prospective* thresholds used in the study over the three sets: the initial set, the additional set and the fault-injected additional set. The table also newly presents the totals after aggregating the results of the three sets; it also presents the *retrospective* thresholds and the results from applying them.

				Agreement				
		Threshold values		**Initial set**	**Additional set**	**Fault-injected additional set**	**All sets combined**	
Approach	**Measure**	*prospective*	*retrospective*	Using *prospective (%)*	Using *prospective (%)*	Using *prospective (%)*	Using *prospective (%)*	Using *retrospective* (%)
Horizontal-plus-vertical	Intersection	0.15	0.08	98.5	73.6	55.2	82.6	87.8
Horizontal-plus-vertical	Chi-square	0.46	0.46	97.4	66.0	79.2	87.5	87.5
Horizontal-plus-vertical	Correlation	0.00	0.00	98.0	71.7	68.8	85.8	85.8
Horizontal-plus-vertical	Bhattacharyya distance	0.23	0.19	98.0	62.3	65.6	83.4	85.5
Horizontal-plus-vertical	Alternative Chi-square	0.82	0.30	96.9	71.7	45.8	78.8	84.6
Horizontal-plus-vertical	Kullback−Leibler divergence	1.55	0.48	98.0	66.0	54.2	80.8	84.0
Horizontal	Chi-square	0.11	0.05	96.4	35.9	67.7	79.1	80.8
Horizontal	Alternative Chi-square	0.14	0.05	93.3	26.4	64.6	75.0	78.8
Horizontal	Bhattacharyya distance	0.20	0.05	91.8	58.5	43.8	73.3	78.5
Horizontal	Intersection	0.09	0.02	97.4	67.9	29.2	73.8	76.5
Horizontal	Correlation	0.00	0.00	87.2	94.3	43.8	76.2	76.2
Horizontal	Kullback−Leibler divergence	0.24	0.01	93.3	41.5	57.3	75.3	75.9

Table XVIII. Results when using VERVE with an improved DOM-filter at three inspection points on wrapping failures for the additional set of web pages.

	Wrapping									
	Minimum			**Middle**			**Maximum**			
	TP	NOI	FP	TP	NOI	FP	TP	NOI	FP	**Total**
EatThisMuch	—	—	1/1	—	—	1/1	—	—	1/1	1
Forvo	4/2	—	—/2	4/2	—	—/2	4/2	—	—/2	4
GMapStreetViewPlayer	—	—	—	—	—	—	—	—	—	—
HoursOf	—	—	—	—	—	—	—	—	—	—
RetailMeNot	—	2/2	4/4	—	2/2	4/4	—	2/2	4/4	6
SB-Admin-2	—	—	—	—	—	—	—	—	—	—
SB-Agency	3/—	—	—/3	3/—	—	—/3	3/—	—	—/3	3
SB-Business-Casual	—	—	—	—	—	—	—	—	—	—
SB-Clean-Blog	—	—	—	—	—	—	—	—	—	—
SB-Coming-Soon	—	—	—	—	—	—	—	—	—	—
SB-Creative	—	—	—	—	—	—	—	—	—	—
SB-Freelancer	—	—	—	—	—	—	—	—	—	—
SB-Grayscale	—	—	—	—	—	—	—	—	—	—
SB-Landing-Page	1/—	—	—/1	1/—	—	—/1	1/—	—	—/1	1
SB-New-Age	—	—	—	—	—	—	—	—	—	—
SB-One-Page-Wonder	—	—	—	—	—	—	—	—	—	—
SB-Resume	2/2	—	—	2/2	—	—	2/2	—	—	2
SB-Stylish-Portfolio	—	—	—	—	—	—	—	—	—	—
SimilarSites	—	—	—	—	—	—	—	—	—	—
Tiiime	—	—	—	—	—	—	—	—	—	—
Total	10	2	5	10	2	5	10	2	5	17
Agreement with manual	4/4	2/2	5/11	4/4	2/2	5/11	4/4	2/2	5/11	—
Per inspection point		64.7%			64.7%			64.7%		—

Since all of the previous research that we reviewed in the area of testing web page presentation failures used the manual approach to visually confirming the reports from a prototype tool, the accuracy and consistency of the manual approach will influence, positively or negatively, the research outcomes. Although we did not experimentally study the output of other testing tools and different types of web page presentation failures, the results make it clear that there are benefits associated with the automated confirmation and classification of web page presentation failures.

Revealing the layers of a web page: Using the CSS `opacity` property is one way to verify presentation failures without making VERVE browser dependent. Another strategy is to manipulate the `visibility` property. However, descendant HTML elements can override the inheritance of this property, meaning that VERVE would have to traverse the DOM tree, potentially adding extra implementation complexity and execution time overhead. Moreover, a limitation of manipulating the `opacity` property emerged when a snapshot was taken before the element had become fully transparent. We discovered this to be the case for one particular viewport protrusion RLF where an `input` HTML element was only partially transparent when snapshots of the AOC were taken. This led us to insert an optional delay into VERVE so that it would wait for elements to become fully transparent. We discussed the effect of adding this delay on the time needed to run VERVE in RQ5, finding that VERVE was still practical for developers to run. With this evidence, multiple delays may be needed if the `visibility` property is used instead of `opacity` for each descendant element. Future work needs to confirm the viability of using the `visibility` property instead.

5. RELATED WORK

While, to our knowledge, there has been no prior research on the automatic visual classification of the presentation failures in responsive web pages, there is an extensive literature on web testing.

For instance, WebDIFF [26], CrossT [27], CrossCHECK [28] and X-PERT [29] are all cross-browser testing tools that use the DOM and/or screenshots to detect variations when a page is viewed on different browsers. It is also worth noting that, like the VERVE tool presented in this paper, both WebDIFF and CrossCHECK use image-based histograms and distance functions to perform tasks for web testing purposes. Similarly, tools such as WebSEE [30] and FieryEYE [31] use the prior version of a page as an oracle to detect presentation failures [30]. Unlike this paper, none of these tools combine the manipulation of HTML element opacity and the use of histogram-based image comparison to confirm and classify responsive layout failures in web pages. Finally, even though there are several prior approaches to web testing that support, for instance, the automated generation and replay of test cases by monitoring user behaviour (e.g. other studies [32-34]), none of them explicitly record the information necessary to enable the detection of layout failures on responsive web pages.

The ReDECHECK tool has two 'modes' [13]. In 'regression checking' mode, the tool targets regression issues by comparing the responsive layout of two versions of a web page [21,35]. In 'common failure checking' mode, ReDECHECK automatically detects the series of common responsive layout failures studied in this paper [6]. Like ReDECHECK, the VFDetector tool also finds responsive layout failures, but can further detect layout issues in pages triggered through human interactions [15]. Notably, tools like ReDECHECK and VFDetector focus on finding responsive layout failures by analysing the DOM of a web page, an approach that may incorrectly surface issues that a human could not reasonably observe. This means that, while both ReDECHECK and VFDetector automatically detect certain types of responsive layout failures, a developer must still manually inspect problems at multiple viewport widths to determine whether they are visible to humans—this is the challenging and error-prone task that VERVE effectively and automatically handles using methods including image opacity manipulation and comparison.

There are also several tools that support the verification of the layout properties of a web page. For instance, CASSIUS formalizes some of the semantics of CSS and supports automated reasoning about the behaviour of CSS style sheets [36]. The VizASSERT tool extends the formal model in CASSIUS, further supporting the automated verification of a web page's accessible layout [37]. Finally, the CORNIPICKLE tool verifies that a web page supports the layout properties specified by a tester [38,39]. Unlike VERVE, all these tools require some formal specification of web page layout. Notably, while these tools focus on automatically verifying layout properties, the presented tool confirms and classifies the responsive layout failures reported by tools like ReDECHECK.

There are many tools that support the design, implementation and testing of a web page's visual properties. For instance, SCRY is a reverse engineering tool that surfaces how changes in the underlying source code will influence a page's visual appearance [40]. Similarly, Liang et al. describe the SeeSS tool that visualizes how changes to a web site's CSS files will affect its layout, with the aim to support the manual detection of layout failures [41]. Moreover, the VISTA tool repairs the broken tests that focus on a web page's visual characteristics [42]. There are also a number of tools that seek to repair presentation failures in web pages. For instance, MFix repairs problems with the mobile-friendliness of a web page [43], while XFIX repairs cross-browser issues [44,45]. Additionally, IFIX [46] and its successor, IFIX^{++} [47], repair internationalization failures using search-based and clustering techniques, while CBRepair addresses the same types of failures but using constraint solving [48]. Finally, Li et al. proposed an approach that first analyses a person's interaction with a web page and then visualizes some presentation failures that might be evident to a human [9]. Since all these tools complement VERVE's focus on automatically confirming and classifying the layout failures reported by tools like ReDECHECK, together they constitute a full-featured strategy for improving the layout of responsively designed web pages.

There are many tools that support manual developer checking of responsive pages. For instance, multi-screenshot tools (e.g. others studies [10,11,49]) showcase a web page at a few common viewport widths, while others (e.g. others studies [12,50,51]) allow the tester to resize the viewport to a custom size. However, empirical studies by Walsh et al. point out that these methods often overlook layout failures that tools like ReDECHECK can automatically detect [6]. Also, in contrast to the graphical reports produced by tools like ReDECHECK, VFDetector and VERVE, all of the aforementioned developer tools have another limitation: the tester must inspect each screenshot, a process that is manual and often time consuming and error prone. Finally, while Fighting LAYOUT BUGS detects some types of layout failures [22], it only checks static layout properties and thus, unlike ReDECHECK and the VERVE tool presented in this paper, it is not applicable to the testing of responsive web pages.

Although this paper focuses on testing the responsive layout of a web page, it is worth pointing out that there are many prior approaches that concentrate on testing the visual properties and layout of either a desktop application or a mobile app. For instance, Memon and Soffa presented a regression testing technique for the graphical user interface (GUI) of a desktop application [52], while Brooks and Memon showed how to leverage user interaction profiles to guide GUI testing [53]. Furthermore, Hammoud et al. and Zaraket et al. proposed GUICop [54,55], a GUI testing approach that utilizes a specification language for capturing information about the layout and appearance of GUI components. This enables functional test cases to tolerate differences in the execution environment in which they are run—for example a change of screen resolution—while also checking properties of the GUI itself, such as the appearance and relative positioning of certain elements. The VERVE tool presented in this paper is also broadly similar to two prior tools for testing a graphical user interface, with both GUIDE and SIKULI highlighting the differences between GUI versions and the latter tool using OpenCV to perform this task [56,57]. Finally, in response to the rising prevalence of mobile apps, Morgado and Paiva and Amalfitano et al. developed testing techniques for Android apps, with the first of these papers identifying and testing recurring app behaviours [58] and the second focusing on ensuring the correctness of an app's layout when a mobile device's orientation changes [59]. Similar to this paper's approach, Moran et al. presented a technique that highlights the way in which a mobile app's layout and visual properties violate established design guidelines [60]. Since, like in the area of responsive web design, there are also many developer-focused tools for mobile app testing, Ardito et al. empirically compared two of the most prominent ones to confirm that the testing of an app's visual and layout properties is particularly challenging [61], as this paper argues is also the case for responsive web pages.

6. CONCLUSIONS AND FUTURE WORK

Even though responsive web design principles and frameworks enable the creation of web pages that display correctly on a wide variety of devices with differing viewport widths, developers may still introduce presentational problems in a web page. Even though the ReDECHECK tool automatically surfaces the responsive layout failures that are likely to exist in a web page, without additional tool support a human web developer must manually classify each RLF as being a true positive, false positive or a non-observable issue—a process that is often time consuming, subjective and error prone. The conference version of this paper introduced the VISER tool that could automatically perform this classification by manipulating the opacity of the HTML elements in a web page. While the empirical results from that paper highlighted the efficiency and effectiveness of VISER, the tool was limited because it could only classify the element collision, element protrusion and viewport protrusion layout failures reported by ReDECHECK.

Since VISER does not classify either the wrapping or small-range responsive layout failures, this paper presented ‘VERVE’ (Visual classifiEr for ResponsiVe tEsting), a tool that can automatically classify all five types of the RLFs reported by ReDECHECK's DOM-based approach. Along with extending VISER's opacity manipulation method to detect element wrapping failures, VERVE employs a histogram-based image comparison method that effectively classifies the small-range failures reported by ReDECHECK. Considering 20 new pages in addition to the 25 web pages from the conference paper's experiments, this paper reported on the results from a comprehensive study of VERVE's efficiency and effectiveness, revealing its classification of all five types of RLFs frequently agrees with the manual one produced by expert web developers. The experiments also showed that VERVE normally took about 4 s to classify an RLF among the 469 reported by ReDECHECK. Given that RLF classification with VERVE is less subjective and error prone than the same manual process performed by a human web developer, this paper's results suggest that the presented tool can support the testing of web pages that must responsively display at different viewport widths.

DOI: 10.1002/stvr

Given the demonstrated benefits of VERVE, future work will involve using the tool to visually confirm and classify the same layout failures when using different, for instance, runtime environments or browsers, thereby better supporting cross-browser testing. Although the empirical results demonstrate that VERVE integrates well with ReDECHECK, we ultimately discovered that some defects in ReDECHECK limit the functionality of the presented approach, thus motivating future work to ensure that VERVE integrates with other DOM-based tools for responsive web testing, like VFDetector [15]. We will also improve VERVE to resolve some of the other limitations highlighted by this paper's empirical results. For instance, since VERVE classifies an RLF as a true positive even if it only exhibits a visual disturbance of a few pixels, we plan to develop new approaches for highlighting those differences most noticeable and meaningful to humans. After integrating VERVE into an even more full-featured web development workflow, we plan to conduct additional experiments to evaluate its efficiency and effectiveness, always increasing the realism, complexity and number of web pages used as subjects. As we apply VERVE to more web pages, we will also run it with different browsers and operating systems, thereby further confirming its generalizability for web testing. Ultimately, we see both ReDECHECK and VERVE playing an important role in helping web developers to surface and classify responsive layout failures, triage and prioritize these defect reports, and automatically implement bug fixes for a defective web page.

ACKNOWLEDGEMENTS

We would like to thank Thomas Walsh for his assistance with the ReDECHECK tool and for help with preparing the subject web pages featured in this paper's empirical study.

Phil McMinn is supported in part by EPSRC grant EP/T015764/1.

REFERENCES

1. McMillen J. 10 Reasons your website needs to be mobile optimized. http://blog.teamtreehouse.com/10-reasons-website-needs-mobile-optimized
2. Marcotte E. *Responsive Web Design*. A Book Apart: New York, 2011.
3. Bootstrap responsive web design framework. https://getbootstrap.com/
4. Foundation responsive web design framework. http://foundation.zurb.com/
5. Creative Bloq: web design trends 2015–16: the long scroll. http://www.creativebloq.com/web-design/web-design-trends-2015-16-long-scroll-81516343
6. Walsh TA, Kapfhammer GM, McMinn P. Automated layout failure detection for responsive web pages without an explicit oracle. In *Proceedings of the International Conference on Software Testing and Analysis*. Santa Barbara, CA, USA, 2017; 192–202. https://dl.acm.org/doi/10.1145/3092703.3092712
7. Robins D, Holmes J. Aesthetics and credibility in web site design. *Information Processing & Management*. 2008; **44**(1): 386–399.
8. Cyr D, Head M, Ivanov A. Design aesthetics leading to M-loyalty in mobile commerce. *Information & Management*. 2006; **43**(8): 950–963.
9. Li W, Harrold MJ, Görg C. Detecting user-visible failures in AJAX web applications by analyzing users' interaction behaviors. In *Proceedings of the 25th International Conference on Automated Software Engineering*. Antwerp, Belgium, 2010; 155–158. https://dl.acm.org/doi/10.1145/1858996.1859025
10. Responsinator. https://www.responsinator.com/
11. Responsive design checker. http://responsivedesignchecker.com
12. Viewport resizer. http://lab.maltewassermann.com/viewport-resizer/
13. Walsh TA, Kapfhammer GM, McMinn P. ReDeCheck: an automatic layout failure checking tool for responsively designed web pages. In *Proceedings of the International Conference on Software Testing and Analysis – Demonstration Papers*. Santa Barbara, CA, USA, 2017; 360–363. https://dl.acm.org/doi/10.1145/3092703.3098221
14. World Wide Web Consortium (W3C). HTML 5.2, 2017. https://www.w3.org/TR/html52/
15. Ryou Y, Ryu S. Automatic detection of visibility faults by layout changes in HTML5 web pages. In *Proceedings of the 11th International Conference on Software Testing, Validation and Verification*: Vasteras, Sweden, 2018; 182–192.
16. Althomali I, Kapfhammer GM, McMinn P. Automatic visual verification of layout failures in responsively designed web pages. In *International Conference on Software Testing, Verification and Validation*: Xi'an, China, 2019; 183–193.
17. Connolly R, Hoar R. *Fundamentals of Web Development*. Pearson: Cambridge, 2017.
18. (W3C) WWWC. XPath syntax. https://www.w3schools.com/xml/xpath_syntax.asp

19. Mahajan S, Halfond WGJ. Finding HTML presentation failures using image comparison techniques. In *Proceedings of the 29th International Conference on Automated Software Engineering*. Vasteras, Sweden, 2014; 91–96. https://dl.acm.org/doi/10.1145/2642937.2642966
20. CSS3 media queries: simple gotchas and easy fixes. https://www.crimsondesigns.com/blog/css3-media-queries-simple-gotchas-easy-fixes/
21. Walsh TA, McMinn P, Kapfhammer GM. Automatic detection of potential layout faults following changes to responsive web pages. In *Proceedings of the 30th International Conference on Automated Software Engineering*: Lincoln, NE, USA, 2015; 709–714.
22. Fighting layout bugs. https://code.google.com/archive/p/fighting-layout-bugs/
23. OpenCV: open-source computer vision library. https://opencv.org
24. Bradski G, Kaehler A. Learning OpenCV 3, 2016. O'Reilly.
25. Selenium: Web browser automation. http://www.seleniumhq.org/
26. Choudhary SR, Versee H, Orso A. WebDiff: automated identification of cross-browser issues in web applications. In *Proceedings of the 26th International Conference on Software Maintenance*: Timisoara, Romania, 2010; 1–10.
27. Mesbah A, Prasad MR. Automated cross-browser compatibility testing. In *Proceedings of the 33rd International Conference on Software Engineering*: Honolulu, Hawaii, 2011; 561–570.
28. Choudhary SR, Prasad MR, Orso A. CrossCheck: combining crawling and differencing to better detect cross-browser incompatibilities in web applications. In *Proceedings of the 5th International Conference on Software Testing, Verification and Validation*: Montreal, QC, Canada, 2012; 171–180.
29. Roy Choudhary S, Prasad MR, Orso A. X-PERT: accurate identification of cross-browser issues in web applications. In *Proceedings of the 35th International Conference on Software Engineering*: San Francisco, CA, USA, 2013; 702–711.
30. Mahajan S, Halfond WGJ. Detection and localization of HTML presentation failures using computer vision-based techniques. In *Proceedings of the 8th International Conference on Software Testing, Verification and Validation*: Graz, Austria, 2015; 1–10.
31. Mahajan S, Li B, Behnamghader P, Halfond WGJ. Using visual symptoms for debugging presentation failures in web applications. In *Proceedings of the 10th International Conference on Software Testing, Verification and Validation*: Chicago, IL, USA, 2016; 191–201.
32. Sampath S, Sprenkle S, Gibson E, Pollock L, Souter Greenwald A. Applying concept analysis to user-session-based testing of web applications. *Transactions on Software Engineering*. 2007; **33**(10): 643–658.
33. Sprenkle S, Gibson E, Sampath S, Pollock L. Automated replay and failure detection for web applications. In *Proceedings of the 20th International Conference on Automated Software Engineering*. Long Beach, CA, USA, 2005; 253–262. https://dl.acm.org/doi/abs/10.1145/1101908.1101947
34. Sprenkle SE, Pollock LL, Simko LM. Configuring effective navigation models and abstract test cases for web applications by analyzing user behaviour. *Software Testing, Verification and Reliability*. 2013; **23**(6): 439–464.
35. Walsh TA, Kapfhammer GM, McMinn P. Automatically identifying potential regressions in the layout of responsive web pages. *Software Testing, Verification and Reliability*. 2020; **30**(6): e1748.
36. Panchekha P, Torlak E. Automated reasoning for web page layout. In *Proceedings of the International Conference on Object-Oriented Programming, Systems, Languages, and Applications*. Amsterdam, Netherlands, 2016; 181–194. https://dl.acm.org/doi/10.1145/2983990.2984010
37. Panchekha P, Geller AT, Ernst MD, Tatlock Z, Kamil S. Verifying that web pages have accessible layout. In *Proceedings of the 39th International Conference on Programming Language Design and Implementation*. Philadelphia, PA, USA, 2018; 1–14. https://dl.acm.org/doi/10.1145/3192366.3192407
38. Hallé S, Bergeron N, Guerin F, Le Breton G. Testing web applications through layout constraints. In *Proceedings of the 8th International Conference on Software Testing, Verification and Validation*: Graz, Austria, 2015; 1–8.
39. Hallé S, Bergeron N, Guérin F, Le Breton G, Beroual O. Declarative layout constraints for testing web applications. *Journal of Logical and Algebraic Methods in Programming*. 2016; **85**: 737–758.
40. Burg B, Ko AJ, Ernst MD. Explaining visual changes in web interfaces. In *Proceedings of the 28th Annual Symposium on User Interface Software and Technology*. Charlotte, NC, USA, 2015; 259–268. https://dl.acm.org/doi/10.1145/2807442.2807473
41. Liang H-S, Kuo K-H, Lee P-W, Chan Y-C, Lin Y-C, Chen MY. SeeSS: seeing what I broke—visualizing change impact of cascading style sheets. In *Proceedings of the 26th Annual Symposium on User Interface Software and Technology*. St. Andrews, Scotland, United Kingdom, 2013; 353–356. https://dl.acm.org/doi/10.1145/2501988.2502006
42. Stocco A, Yandrapally R, Mesbah A. Visual web test repair. In *Proceedings of the 26th Joint Meeting of the European Software Engineering Conference and the Symposium on the Foundations of Software Engineering*. Lake Buena Vista, FL, USA, 2018; 503–514. https://dl.acm.org/doi/abs/10.1145/3236024.3236063
43. Mahajan S, Abolhassani N, McMinn P, Halfond WGJ. Automated repair of mobile friendly problems in web pages. In *Proceedings of the 40th International Conference on Software Engineering*: Gothenburg, Sweden, 2018; 140–150.
44. Mahajan S, Alameer A, McMinn P, Halfond WGJ. Automated repair of layout cross browser issues using search-based techniques. In *Proceedings of the International Conference on Software Testing and Analysis*. Santa Barbara, CA, USA, 2017; 249–260. https://dl.acm.org/doi/10.1145/3092703.3092726
45. Mahajan S, Alameer A, McMinn P, Halfond WGJ. XFix: Automated tool for repair of layout cross browser issues. In *Proceedings of the International Conference on Software Testing and Analysis*. Santa Barbara, CA, USA, 2017; 368–371. https://dl.acm.org/doi/10.1145/3092703.3098223

46. Mahajan S, Alameer A, McMinn P, Halfond WGJ. Automated repair of internationalization presentation failures in web pages using style similarity clustering and search-based techniques. *2018 IEEE 11th International Conference on Software Testing, Verification and Validation (ICST)*. Vasteras, 2018; 215–226. https://doi.org/10.1109/ICST.2018.00030
47. Mahajan S, Alameer A, McMinn P, Halfond WGJ. Effective automated repair of internationalization presentation failures in web applications using style similarity clustering and search-based techniques. *Software Testing, Verification and Reliability*. 2020: e1746.
48. Alameer A, Chiou PT, Halfond WGJ. Efficiently repairing internationalization presentation failures by solving layout constraints. In *Proceedings of the 12th International Conference on Software Testing, Validation and Verification*: Xi'an, China, 2019; 172–182.
49. Responsive design testing. http://mattkersley.com/responsive/
50. Firefox developer tools: responsive design mode. https://developer.mozilla.org/en-US/docs/Tools/Responsive_Design_Mode
51. ResponsivePX. http://responsivepx.com/
52. Memon AM, Soffa ML. Regression testing of GUIs. In *Proceedings of the 11th International Symposium on Foundations of Software Engineering*. Helsinki, Finland, 2003; 118–127. https://dl.acm.org/doi/10.1145/940071.940088
53. Brooks PA, Memon AM. Automated GUI testing guided by usage profiles. In *Proceedings of the 22nd International Conference on Automated Software Engineering*. Atlanta, Georgia, USA, 2007; 333–342. https://dl.acm.org/doi/10.1145/1321631.1321681
54. Hammoud FA, Masri W. GUICop: approach and toolset for specification-based GUI testing. *Software Testing, Verification and Reliability*. 2017; **27**(8): e1642.
55. Zaraket FA, Masri W, Adam M, Hammoud D, Hamzeh R, Farhat R, Khamissi E, Noujaim J. GUICop: specification-based GUI testing. In *Proceedings of the International Conference on Software Testing, Verification and Validation (ICST 2012)*: Montreal, QC, Canada, 2012; 747–751.
56. Chang T-H, Yeh T, Miller RC. GUI testing using computer vision. In *Proceedings of the International Conference on Human Factors in Computing Systems*. Atlanta, Georgia, USA, 2010; 1535–1544. https://dl.acm.org/doi/10.1145/1753326.1753555
57. Xie Q, Grechanik M, Fu C, Cumby C. Guide: a GUI differentiator. In *Proceedings of the International Conference on Software Maintenance*: Edmonton, AB, Canada, 2009; 395–396.
58. Morgado IC, Paiva AC. Mobile GUI testing. *Software Quality Journal*. 2018; **26**(4): 1553–1570.
59. Amalfitano D, Riccio V, Paiva ACR, Fasolino AR. Why does the orientation change mess up my Android application? From GUI failures to code faults. *Software Testing, Verification and Reliability*. 2018; **28**(1): e1654.
60. Moran K, Li B, Bernal-Cárdenas C, Jelf D, Poshyvanyk D. Automated reporting of GUI design violations for mobile apps. In *Proceedings of the 40th International Conference on Software Engineering*: Gothenburg, Sweden, 2018; 165–175.
61. Ardito L, Coppola R, Morisio M, Torchiano M. Espresso vs. EyeAutomate: an experiment for the comparison of two generations of Android GUI testing. In *Proceedings of the International Conference on Evaluation and Assessment on Software Engineering*. Copenhagen, Denmark, 2019; 13–22. https://dl.acm.org/doi/10.1145/3319008.3319022