HCI Vision for Automated Analysis and Mining of Web User Interfaces

Maxim Bakaev[1(✉)], Sebastian Heil[2], Vladimir Khvorostov[1], and Martin Gaedke[2]

[1] Novosibirsk State Technical University, Novosibirsk, Russia
{bakaev, xvorostov}@corp.nstu.ru
[2] Technische Universität Chemnitz, Chemnitz, Germany
{sebastian.heil, martin.gaedke}@informatik.tu-chemnitz.de

Abstract. Most techniques for webpage structure and design mining are based on code analysis and are detached from a human user's perception of the web user interface (WUI). Our paper is dedicated to approaches that instead focus on analysis of webpage's visual representation – the way it is rendered in different browsers and environments and delivered to the end user. Specifically, we describe the software tool that we built, which takes a WUI screenshot and produces structured and machine-readable representation (JSON) of interface elements as made out by a human user. The implementation is based on OpenCV (image recognition functions), dlib (trained detector for the elements' classification), and Tesseract (label and content text recognition). To demonstrate feasibility of the approach, we describe application of our analyzer tool to auto-calculate certain measures for a WUI and to predict users' subjective impressions. Particularly, we assess UI visual complexity, which is known to significantly influence both cognitive and affective aspects of interaction. The results suggest the analyzer's output is mostly characteristic of the users' visual perception and can be useful for auto-assessing and comparing WUIs.

Keywords: Web design mining · HCI vision · Image recognition
Human factors · Visual complexity

1 Introduction

Web Engineering sees a growing demand for means to auto-assess quality of web user interfaces (WUIs), particularly their usability and subjective attractiveness for target users [1, 2]. In webpage structure mining, code analysis techniques aimed on HCI-related UI assessment are mostly able to gauge user- and task-independent aspects. Examples include validating HTML/CSS, checking correspondence to accessibility guidelines, finding "bad usability smells" [1], etc. As nowadays web pages are increasingly produced by web engineering frameworks or content management systems,

M. Bakaev and S. Heil—Both authors contributed equally to the work.

T. Mikkonen et al. (Eds.): ICWE 2018, LNCS 10845, pp. 136–144, 2018.
https://doi.org/10.1007/978-3-319-91662-0_10

their code becomes reasonably accessibility-compliant and "smell-free", but that alone does not ensure good usability. Extending the approach to some interaction dimensions is problematic: e.g., to investigate web page layout and spatial properties of WUI elements, a code analysis tool basically has to incorporate a web browser's rendering engine.

So, UI analysis and mining see increasing application of computer vision techniques, which are already widely used in medicine, robotics, production quality control, etc. A basic computer vision task is image recognition, which is identification of visual objects and their classification into known subsets (object types). The identification is carried out through image segmentation or discovery of shapes, based on detection of edges, surfaces or textures, etc. Visual analysis of UIs based on computer vision techniques, which we call *human-computer interaction (HCI) vision*, has certain particular features compared to the other application domains. Those that make the analysis easier include: absence of noise, glare, or difference in lighting; perfect angle of view; no need to consider movement; mostly complete, not partially covered objects. Arguably, the main challenge is visual variability within WUI elements types due to different design styles: even whitespace is rarely white in webpages.

Still, the important advantage of HCI vision is the guarantee that the analyzer deals with the same interface the user witnesses, while HTML/CSS code is unpredictable in this (e.g. it can be rendered differently in different browsers and environments). For UI visual analysis, template-matching based recognition, e.g., with the popular and free OpenCV library, is probably the most straightforward approach, but it's relatively slow. An influential novation in the field was the VIPS (VIsion-based Page Segmentation) algorithm for detection of webpage layout structure based on its visual representation [3]. It inspired the potent Bento/Bricolage solution for semantic page segmentation and design mapping, in which image analysis is primary and supplemented by DOM [4]. Indeed, most recent approaches combine the visual analysis with code mining [5], which in particular makes them capable of increasing performance of web page segmentation and extracting textual content without the "costly" recognition stage [6]. The analysis results are generally reverse-engineered interface semantics represented as DOM-like tree or spatial graph [7] or some interface metrics, related to content (e.g., shares of text, images and whitespace on web page), colors or visual complexity [8].

Visual complexity is one of the most investigated metrics in webpage analysis, since this measure affects user's cognitive load, subjective perceptions of aesthetics and usability etc. [8]. However, its automated assessment in WUIs has until recently been limited, although largely seen as desirable [9]. The three major factors behind the visual-spatial complexity of UI are arguably the number of objects, their diversity, and the regularity of their spatial allocation [10]. Accurate evaluation of the latter is most problematic (as it's fundamentally the non-computable descriptive complexity) and is often approximated with a compression or entropy-related measure.

In our paper we describe implementation of a WUI visual analyzer and evaluate its viability in predicting users' perception. Though the process that we proposed and implemented can be suitable for many UI mining tasks, in the current work we applied it to the analysis of visual complexity. In Sect. 2 we describe the architecture of our analyzer tool and provide some technical details. In Sect. 3 we evaluate the data produced by the analyzer and explore whether they can be characteristic of users'

perception of WUIs. In the final section we interpret the results of the data statistical analysis and provide conclusions on the analyzer's capacities and current limitations.

2 The WUI Visual Analyzer

As shown in Fig. 1, our UI analysis architecture consists of two parts: the Analyzer Frontend and the Visual Analysis backend. The Analyzer Frontend is a Web Application[1] that communicates with the Visual Analysis Backend, implemented as Web Service, via an HTTP Interface. Our visual page analysis algorithm is based on [7]: it takes a screenshot of a user interface as input and tries to identify the UI elements of which the interface is formed. Based on the "atomic" UI elements, higher-level structures can be identified through analysis of the visual hierarchy of the interface, using closeness, alignment, containment etc.

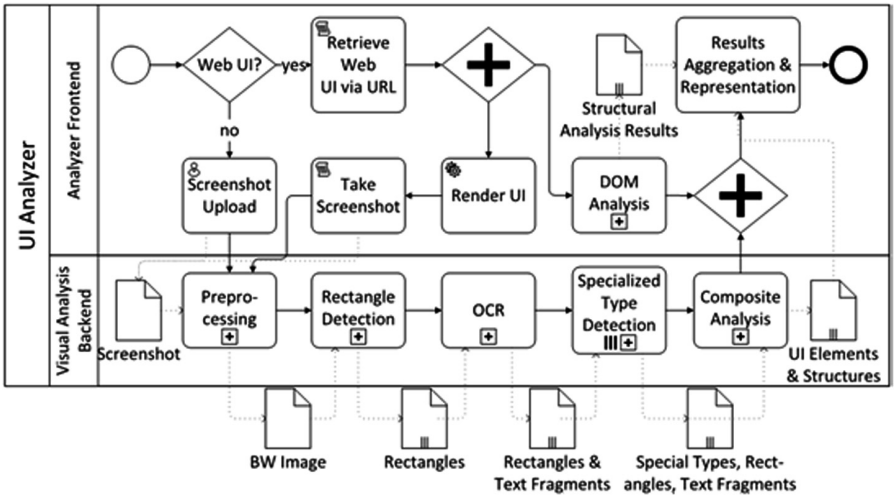

Fig. 1. Overview of the analysis process.

Preprocessing. To improve the edge detection results, preprocessing of inputs into black-and-white inputs is required. Those images are represented as binary matrices. To achieve this input, the screenshot is converted to grayscale. The binary image is then produced using a threshold function. Due to the relatively low amount of color in user interfaces compared to general images, separate binary images from different color channels are not providing significant improvements to subsequent processing. Upscaling before conversion to black-and-white has shown to improve detection rate significantly. We are using the respective *OpenCV*[2] functions to perform this step.

[1] available online: http://webmining.wuikb.tech/ratio/screenshot.php.

[2] https://opencv.org/.

Detection of Rectangular Areas. In order to identify UI elements like frames, buttons or textboxes, this step detects rectangular areas of interest. This is done using OpenCV's edge detection for horizontal and vertical lines on the binary matrix. The resulting list of vectors is then checked for rectangles by looking for convex shapes with 4 corners above a minimum size, resulting in 4-tuples $r = (x, y, width, height)$.

Optical Character Recognition (OCR). To identify text in the UI, we use a combination of OpenCV's close-edge detection and *Tesseract*[3]. First, areas with close edges are identified as candidates. These areas are then upscaled and converted to black edges on white background. If Tesseract yields a result on the snippets, they are considered as text. The bounding rectangles are annotated with the textual content and added to the previous list of rectangles.

Detection of Special UI Element Types. This step uses specialized detectors for different types of UI elements (e.g. *radiobuttons*, *checkboxes*, *dropdown* menus). The detectors are trained on one particular type using supervised learning with the Felzenszwalb HOG feature extractor implemented in *dlib*[4]. Radiobuttons and checkboxes in checked and unchecked state have to be separately trained. Training datasets have to represent different styles of UI elements, e.g. Windows and MacOS styles. For native operating system style UI elements, the datasets can be very small whereas for WUIs with various CSS styles, larger datasets are required. The detected UI elements are added to the previous list as bounding rectangles with annotated type.

Analysis of Composite Structures. In this step, composite structures are identified based on the un-typed rectangles and text fragments detected in previous steps using decision tree rules. Text fragments are classified as *label* according to proximity to other objects, *word* if isolated, *line* if min. five horizontally neighboring words with equal vertical alignment exist, and *text* of min. 2 lines in vertical proximity. Labeled/unlabeled buttons, dropdowns and textfields are identified combining the text and object locations. Areas are detected as rectangles containing other UI elements (Fig. 2 shows visualized results of this step). The identified UI elements and structures are represented as JSON in the HTTP API:

```
{"elements": [
    { "height": 35, "positionX": 10, "positionY": 50,
      "text": "Name", "type": "label", "width": 40 },
    { "height": 35, "positionX": 70, "positionY": 50,
      "text": "Enter name", "width": 250,
      "type": "labeled dropdown or textfield" },
    { "height": 20, "positionX": 130, "positionY": 100,
      "text": "OK", "type": "button", "width": 50 },
  ],
 "page": { "height": 1024, "width": 1280}}
```

[3] https://github.com/tesseract-ocr/tesseract.

[4] http://dlib.net/.

Results Aggregation and Representation. The analyzer frontend aggregates the visual analysis results with the results from DOM analysis and calculates several metrics for the UI. The first group included the 3 metrics related to visual complexity:

1. The number of all identified UI elements: *AElements*. Note that this number results from visual analysis and is agnostic to programming styles and HTML elements invisible to users, in contrast to elements in DOM analysis.
2. The number of different elements types identified by the analyzer: *AVocab*.
3. Compression rate *ACompress*, the current representation of UI's spatial regularity, is currently calculated as the area S_0 (in pixels) of the webpage divided by the file size F (in bytes) of the image compressed using the JPEG-100[5] algorithm.

Fig. 2. A web interface screenshot with UI elements recognition results highlighted.

The three metrics of the second group characterized relative (i.e. divided by S_0) shares of the areas in the UI covered by the different types of UI elements:

4. Textual content, i.e. area under all elements recognized as textline: *AText*.
5. Graphic and mixed content, i.e. area under all elements of the other types: *AOther*.
6. Whitespace, i.e. all the remaining pixels: *AWhite*.

3 Evaluation

The goal of the experimental evaluation was to test whether data provided by the analyzer can be useful in characterizing users' perceptions of web UI, particularly the complexity. Thus, the main hypotheses were: (1) the analyzer metrics are correlated

[5] cf. ISO/IEC 10918-1:1994.

with the respective subjective evaluations by users; (2) the analyzer metrics can predict subjective complexity of web UIs. Since asking the users to express the subjective metrics in absolute quantities would be unattainable, we chose to rely on ordinal values. We also employed an entropy measure presumably reflecting regularity of UI.

3.1 Experimental Survey

Experimental Design. The material in the experiment was 21 operating websites of 11 German universities and 10 Russian universities (in all cases, English versions were used). Based on the experiment's goals, we devised 7 evaluation scales (see in Table 1), for each of which 7-point Likert scale was used (1 being "completely disagree", 7 – "completely agree"). The independent variables were the 3 complexity metrics produced by the analyzer (AElements, AVocab, and ACompress), the 3 "areal" metrics (AText, AOther, and AWhite), plus the entropy measure *MEntropy*. In the current work, the latter was obtained by applying the standard Matlab's *entropy(I)* function (returns a scalar value E reflecting the entropy of grayscale image I) to the websites screenshots.

Table 1. The experimental scales and descriptive statistics

Scale (statement)	Var. name	Range	Mean (SD)
This webpage has many elements	*SElements*	2.94–5.81	4.59 (0.94)
The elements in the webpage are very diverse	*SVocab*	2.78–5.21	4.18 (0.77)
The elements in the webpage are well-ordered	*SOrder*	2.84–5.18	4.31 (0.50)
The webpage has a lot of text	*SText*	2.65–6.21	3.92 (0.99)
The webpage has a lot of graphics	*SImg*	2.27–5.81	4.00 (1.17)
The webpage has a lot of whitespace	*SWhite*	2.46–6.13	3.51 (0.88)
The webpage appears very complex	*SComplex*	2.57–5.83	3.60 (0.75)

The Apparatus and Procedure. The survey to collect data was implemented in LimeSurvey software, and the participants used web browser to interact with it. Some of them worked in university computer rooms, while the others used their own computer equipment with varying screen resolutions, to better represent the real context of use. Each subject was asked to evaluate the screenshots of the 21 websites' homepages (presented one by one in random order) per the 7 scales. On average, it took each participant 30.3 min to complete the survey, and the data collection session lasted 19 days overall. We used screenshots, not the actual websites, to ensure uniformity of the experimental material between the participants and with the analyzer.

The Participants. In total, 63 participants (30 male, 33 female) provided their evaluations of the websites. The convenience sampling method was applied, with most of the participants being students or universities staff members. The self-denoted age ranged from 19 to 72, mean 27.6, SD = 8.07. The self-denoted nationalities were Russian (65.1%), German (17.5%), Argentinian (4.8%), and others (including

Bulgarian, Vietnamese, Korean, etc.). Submissions by another 13 participants were discarded as being incomplete (none of them had at least 50% of websites evaluated).

3.2 The Data Analysis Results

Descriptive Statistics. In total, 9261 evaluations were collected in experiment, of which 95.2% were considered valid; one website (#14) was removed from the analysis due to technical problem with the screenshot. Table 1 presents averaged data that we will further use in the analysis. We are aware about certain controversy in calculating mean for ordinal data, but we are going to only use the averaged values as the relative metrics for the websites, i.e. continue treating them as ordinal data. The Shapiro-Wilk tests suggested that for SWhite ($p = 0.01$), SElements ($p = 0.05$) and SOrder ($p = 0.05$) the normality hypotheses had to be rejected.

The Analysis of Correlations. To measure associations between the respective values, we used Kendall's tau-b, as non-parametric statistic (without assumption of normality) for ordinal scales. In the analyzer performance analysis, we found significant correlations between AElements and SElements ($\tau = 0.526$, $p = 0.001$), AElements and SVocab ($\tau = 0.501$, $p = 0.002$), AElements and SImg ($\tau = 0.347$, $p = 0.032$), AElements and AVocab ($\tau = 0.447$, $p = 0.011$), ACompress and SWhite ($\tau = 0.347$, $p = 0.032$).

In the areal metrics, AText was significantly correlated with SVocab ($\tau = –0.322$, $p = 0.048$), SText ($\tau = 0.495$, $p = 0.002$), and SImg ($\tau = –0.484$, $p = 0.003$). AWhite was correlated with SElements ($\tau = 0.421$, $p = 0.009$), SVocab ($\tau = 0.491$, $p = 0.003$), SText ($\tau = –0.379$, $p = 0.019$), SImg ($\tau = 0.600$, $p < 0.001$), and SWhite ($\tau = –0.337$, $p = 0.038$).

In the complexity analysis, significant correlations with SComplex were found for SElements ($\tau = 0.582$, $p < 0.001$) and SVocab ($\tau = 0.440$, $p = 0.007$), while SOrder ($\tau = –0.299$, $p = 0.068$) and AElements ($\tau = 0.307$, $p = 0.06$) were correlated with SComplex at $\alpha = 0.07$. MEntropy was correlated with SVocab ($\tau = 0.364$, $p = 0.025$), SOrder ($\tau = 0.372$, $p = 0.023$), SText ($\tau = –0.474$, $p = 0.004$), SImg ($\tau = 0.505$, $p = 0.002$), and SWhite ($\tau = –0.453$, $p = 0.005$), but, unexpectedly, not with SComplex ($\tau = –0.032$, $p = 0.845$).

The Regression Analysis. To test whether the assessed UI complexity (SComplex) could be predicted by the analyzer, we used ordinal regression with AElements and ACompress factors (to match the ordinal regression's assumptions, AVocab was excluded as highly correlated with AElements). We also included the MEntropy factor, as correlated with the assessments in the survey, but with none of the analyzer's metrics. The resulting ordinal regression model was highly significant ($\chi^2(3) = 13.83$, $p = 0.003$, Nagelkerke pseudo $R^2 = 0.501$), and the proportional odds assumption held ($p = 0.187$). Of the three factors, only AElements ($b = 0.048$, $W(1) = 9.25$, $p = 0.002$) and ACompress ($b = –2.178$, $W(1) = 5.06$, $p = 0.024$) were significant, but not MEntropy ($W(1) = 1.391$, $p = 0.238$).

4 Conclusions

The results of the correlations analysis suggest that AElements was characteristic of several subjective metrics, including SElements ($\tau = 0.526$). On the contrary, AVocab didn't have significant associations with the users' perceptions. AElements was also correlated with SComplex ($\tau = 0.307$), the overall subjective complexity of web UI. In the areal analyzer's metrics, AText was characteristic of SText ($\tau = 0.495$) and several other subjective metrics. AWhite was correlated with most of the subjective metrics, but these associations were contrary to what would be expected: e.g. there was negative correlation with SWhite ($\tau = -0.337$). Actually, ACompress was better in characterizing the UI subjective ampleness ($\tau = 0.347$). For AOther no significant correlations were found. These findings suggest that the analyzer is worthy at counting the number of UI elements and the area under textual content, but so far is poor at determining the types of UI elements and the area under graphics (AOther and hence AWhite).

The analyzer's metrics were also significant in the regression model predicting web UI subjective complexity ($p = 0.003$, pseudo-$R^2 = 0.501$). MEntropy, calculated with Matlab's image entropy function, wasn't helpful in predicting complexity, which suggests that the frequency analysis it implements does not reflect UI subjective regularity well. The model is ordinal regression, and automating relative ranking of UIs (e.g. different versions of the same webpage design) is indeed desirable in Web Engineering. Overall, our visual analyzer's output is mostly characteristic of the users' visual perception and can be useful for auto-assessing and comparing WUIs.

Our further work plans will be aimed at improving the analyzer's capacity in recognizing the types of UI elements, supplementing it with webpage code analysis (DOM). We also plan to study the regularity dimension of UI complexity: either elaborating on MEntropy, since it was correlated with most subjective metrics, or employing machine learning approaches to train predictive ANN models.

Acknowledgement. The reported study was funded by RFBR according to the research project No. 16-37-60060 mol_a_dk.

References

1. Grigera, J., Garrido, A., Rivero, J.M., Rossi, G.: Automatic detection of usability smells in web applications. Int. J. Hum. Comput. Stud. **97**, 129–148 (2017)
2. Bakaev, M., Mamysheva, T., Gaedke, M.: Current trends in automating usability evaluation of websites: can you manage what you can't measure? In: Proceedings of IEEE 11th International Forum on Strategic Technology (IFOST), pp. 510–514 (2016)
3. Cai, D., Yu, S., Wen, J.-R., Ma, W.-Y.: Extracting content structure for web pages based on visual representation. In: Zhou, X., Orlowska, Maria E., Zhang, Y. (eds.) APWeb 2003. LNCS, vol. 2642, pp. 406–417. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36901-5_42
4. Kumar, R., Talton, J.O., Ahmad, S., Klemmer, S.R.: Bricolage: example-based retargeting for web design. In: Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems, pp. 2197–2206 (2011)

5. Sanoja, A., Gançarski, S.: Block-o-matic: a web page segmentation framework. In: Proceedings of IEEE International Conference on Multimedia Computing and Systems (ICMCS), pp. 595–600 (2014)
6. Pu, J., Liu, J., Wang, J.: A vision-based approach for deep web form extraction. In: Park, James J.(Jong Hyuk), Chen, S.-C., Raymond Choo, K.-K. (eds.) MUE/FutureTech -2017. LNEE, vol. 448, pp. 696–702. Springer, Singapore (2017). https://doi.org/10.1007/978-981-10-5041-1_111
7. Kong, J., et al.: Web interface interpretation using graph grammars. IEEE Trans. Syst. Man Cybern. Part C (Appl. Rev.) **42**(4), 590–602 (2012)
8. Reinecke, K., et al.: Predicting users' first impressions of website aesthetics with a quantification of perceived visual complexity and colorfulness. In: Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems, pp. 2049–2058 (2013)
9. Wu, O., Hu, W., Shi, L.: Measuring the visual complexities of web pages. ACM Trans. Web (TWEB) **7**(1), 1 (2013)
10. Bakaev, M., Razumnikova, O.: Opredeleine slozhnosti zadach dlya zritelno-prostranstvennoi pamyati i propustkoi spospobnosti cheloveka-operatora. Upravlenie bol'shimi sistemami = Large-Scale Systems Control **70**, 25–57 (2017). (in Russian)