SketchingInterfaces: A Tool for Automatically Generating High-Fidelity User Interface Mockups from Hand-Drawn Sketches

Christoph Wimmer
TU Wien
Vienna, Austria
christoph.wimmer@inso.tuwien.ac.at

Alex Untertrifaller
TU Wien
Vienna, Austria
alex.untertrifaller@inso.tuwien.ac.at

Thomas Grechenig
TU Wien
Vienna, Austria
thomas.grechenig@inso.tuwien.ac.at

ABSTRACT

Sketches are an integral part of designing user interfaces and despite advances in design tools, many user interface designers still rely on pen and paper when creating these sketches because of their reliability, familiarity and ease of use. Rough sketches are well suited for quick ideation and experimentation, but are typically lacking in detail, resulting in limited readability and clarity when presented to others. In this paper we present the SketchingInterfaces tool, which uses machine learning to translate hand-drawn UI sketches into high-fidelity UI mockups in real-time. The goals of this tool are the following: (1) to speed up the design process, (2) to allow rapid iteration on near-final design artifacts, (3) to improve group collaboration and (4) to lower the threshold of required design skills. We present the results of a study evaluating the tool with six participants to demonstrate the viability of the approach.

CCS CONCEPTS

• **Human-centered computing → Systems and tools for interaction design**; **Empirical studies in interaction design**.

KEYWORDS

user interface design, design tools, sketching, prototyping, machine learning

ACM Reference Format:
Christoph Wimmer, Alex Untertrifaller, and Thomas Grechenig. 2020. SketchingInterfaces: A Tool for Automatically Generating High-Fidelity User Interface Mockups from Hand-Drawn Sketches. In *32ND AUSTRALIAN CONFERENCE ON HUMAN-COMPUTER INTERACTION (OzCHI '20), December 2–4, 2020, Sydney, NSW, Australia.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3441000.3441015

1 INTRODUCTION

Designing user interfaces is commonly understood to be an iterative process, wherein design artifacts of increasing fidelity are created and iteratively improved and refined. Designers begin by creating rough sketches to express their initial ideas and explore alternatives before moving on to increasingly more detailed representations of a final design. This exploration and experimentation with different ideas is considered a crucial part of user interface design in order to explore the possible solution space before choosing the most promising concepts to focus on [3]. Otherwise there's a risk to focus on the first possible solution that comes to mind, rather than finding the best possible solution to a given design problem.

Despite great technological advances in design tools, many designers still prefer to use hand-drawn sketches made with pen and paper for these early, exploratory phases of design ideation. Newman and Landay [13] found in a study of website design practice that "[a]lmost all of the designers did at least some sketching on paper, generally during the design exploration phase". A design industry survey from 2015 [19] with more than 4000 participants from 196 countries found that an overwhelming majority of 64 % of participants preferred pen and paper for their ideation and brainstorming phase.

The advantages of pen and paper for sketching are readily apparent: these tools are flexible, reliable, intimately familiar to everyone, easy to use, cheap and readily available. As such, sketching with pen and paper is particularly well suited for "the design exploration phase of a project, when designers wish to explore many design possibilities quickly without focusing on low-level details" [13].

However, sketches also have limitations and drawbacks. Rough, hand-drawn sketches typically lack detail and therefore fail to create an accurate impression of what the final user interface will actually look like. Sketches can be ambiguous, lacking in clarity and thus be open to misinterpretation. This can be particularly challenging when using sketches for communication and collaboration with other designers or stakeholders, who might gain a very different mental image of a final design when looking at sketches and filling in the blanks. In addition, the process of refinement when moving from rough sketches to detailed, high-fidelity representations of the final user interface design is still a rather laborious process.

To address these issues, we present the SketchingInterfaces tool. SketchingInterfaces is a proof-of-concept prototype which uses machine learning to recognize hand-drawn user interface sketches and convert them into detailed high-fidelity mockups in real-time. In doing so, the purposes of this tool are the following:

(1) to speed up the design process when moving from rough sketches to detailed high-fidelity mockups,
(2) to allow rapid iteration on near-final design artifacts,
(3) to improve group collaboration and,
(4) to lower the threshold of required design skills and technical skills, and by extension to foster inclusion by allowing

OzCHI '20, December 2–4, 2020, Sydney, NSW, Australia

ACM ISBN 978-1-4503-8975-4/20/12…$15.00
https://doi.org/10.1145/3441000.3441015

(a) (b) (c) (d)

Figure 1: Example of (a) sketch input and corresponding conversion results produced by (b) Sketch2Code [6], (c) uizard.io [18] and (d) SketchingInterfaces tool (samples from Sketch2Code and uizard.io taken on July 21, 2020)

> stakeholders of various skill levels to directly partake in UI design processes

by bridging the gap between physical artifacts such as hand-drawn sketches and detailed digital representations of the corresponding user interfaces. It must be stressed that the purpose of this tool is not to replace designers and their creative process by automating the creation of user interfaces, but rather to assist them in their work.

The contributions of this paper are twofold: first the architecture, design and functionality of the SketchingInterfaces tool, and second the results of an initial user study conducted to evaluate the utility and usability of the tool and to gain further insight into how such a tool might fit into a designer's workflow and process.

2 RELATED WORK

Buxton characterizes sketching as an archetypal design activity that is related and complementary to, but distinct from prototyping [2]. This distinction is borne of a difference in purpose and intent: whereas prototypes converge toward a clear goal or result through iterative improvement once a basic understanding of the expected outcome has been established and are more concentrated at the later stages of a design process, sketches are more often used in the early design stages for purposes of creative ideation and exploration of a design space at the outset. Sketches and prototypes are nevertheless related and their boundaries are fleeting: both are representations of a design concept and through iterative improvement and refinement, sketches can serve as a foundation and evolve into more detailed and refined prototypes. In moving from low-fidelity sketches to high-fidelity prototypes and mockups, designers have to expend additional time and effort [15] and a prior study found "an increase in workload specifically in frustration, temporal demand, effort, and decline in performance as the participants progressed from low to high fidelity" [17].

For this reason, supporting and improving the creation of sketches in user interface design processes has been a goal of researchers and tool creators for many years. An early example is SILK [7, 8], which "allows designers to quickly sketch an interface using an electronic pad and stylus" and to create storyboards from these sketches in order to express their interactive behaviour. Another example is DENIM [9], a tool for supporting designers in creating websites which "allows designers to quickly sketch out pages, create links among them, and interact with them in a run mode." A more recent example is EVE [17], which provides designers with a digital canvas to create low-fidelity sketches and subsequently transforms those sketches into more detailed medium- and high-fidelity prototypes with a UI element detector using TensorFlow Object Detection API. In contrast to the SketchingInterfaces tool, the aforementioned tools don't allow designers to use pen and paper for sketching, but rather provide a digital canvas for sketch creation.

With advances and more wide-spread adoption of machine learning, researchers and designers from both academia and industry have begun to explore its use as a tool for the automatic creation of user interfaces. A number of tools that translate hand-drawn user interface sketches into digital representations of those sketches exist: uizard.io [18] is a service (available in private beta as of July 2020) which allows designers to upload a photo of their user interface sketches for automatic, server-side conversion into a detailed UI mockup. Similarly, Sketch2Code [6] is a publicly available research prototype by Microsoft AI Lab which provides similar functionality, allowing designers to submit a photo of their user interface sketches for conversion into HTML code. However, there is a crucial difference regarding workflow between uizard.io and Sketch2Code compared to the SketchingInterfaces tool: whereas the former require designers to submit a picture as a static representation of their sketch for subsequent conversion, SketchingInterfaces converts live camera imagery, allowing designers to directly edit and refine their sketches (e.g. on a whiteboard), with those changes represented in real-time in the digital conversion of their sketches. A comparison of example conversion results produced by Sketch2Code, uizard.io and the SketchingInterfaces tool is demonstrated in Figure 1.

Similar concepts to those implemented in the SketchingInterfaces tool were demonstrated in prototypes by designers from Airbnb [20] and Xing [16]. Unfortunately no further information about these prototypes beyond the aforementioned weblog postings and their associated video demonstrations appear to be publicly available.

Beyond sketch recognition, researchers have also investigated the use of computer vision and machine learning for model building, automatic source code generation or semantic feature extraction from static UI screenshots and mockups. For example, REMAUI

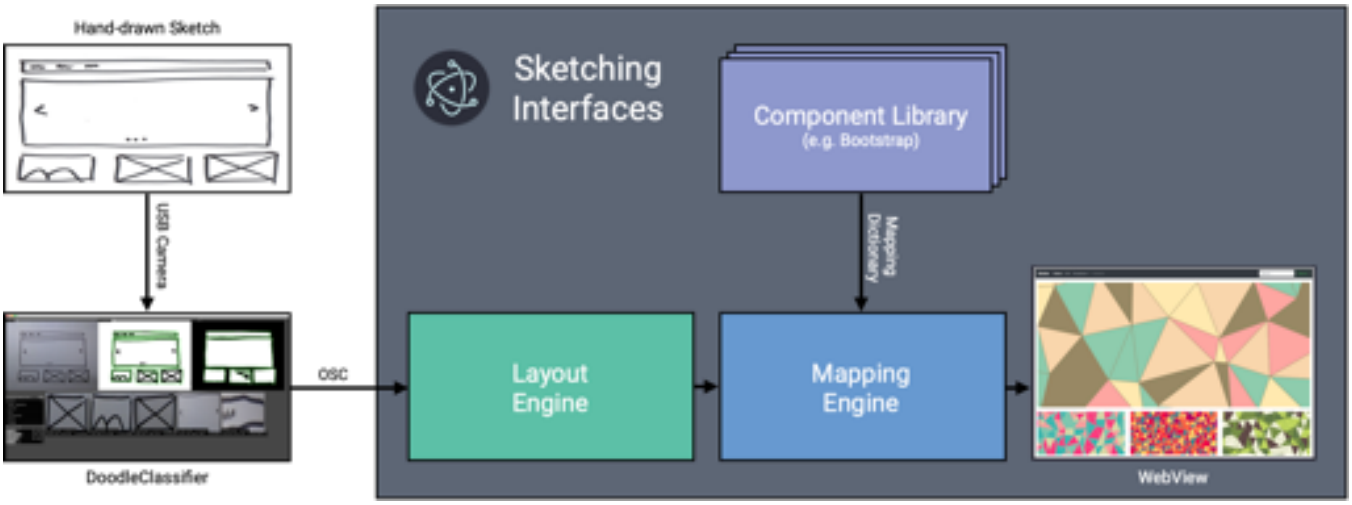


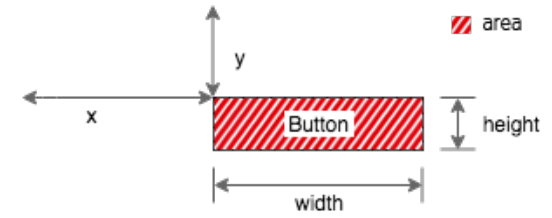
Figure 2: SketchingInterfaces architecture: Hand-drawn sketches serve as input and are analyzed by DoodleClassifier; DoodleClassifier communicates with the core SketchingInterfaces application using OSC protocol; the core SketchingInterfaces application consists of the following sub-components: layout engine, mapping engine, component libraries and an embedded WebView for result visualization

[14] infers working UI code from a screenshot or mockup of a mobile app using computer vision and optical character recognition, bridging the gap between detailed UI mockups and implementation. Pix2code [1], ReDraw [12] and a framework presented by Chen et al. [5] similarly generate UI code from a static input image such as a screenshot, but in contrast to REMAUI rely on convolutional neural networks for reverse engineering the input images into UI code. Chen et al. [4] presented a tool for generating UI skeletons based on an input image and Liu et al. [10] describe the use of a convolutional neural network to generate semantic annotations for mobile app UIs. Compared to the SketchingInterfaces tool, the aforementioned works rely on screenshots or detailed high-fidelity mockups as input rather than hand-drawn sketches. Furthermore and in contrast to these works, which largely focus on technical qualities and aspects, this paper focuses on use-oriented qualities and how such a tool can fit into a designer's workflow.

3 ARCHITECTURE, IMPLEMENTATION AND FUNCTIONALITY

Compared to similar, existing tools and projects [6, 16, 18, 20], SketchingInterfaces supports the following features and functionality: (1) real-time component detection, recognition and mockup visualization, (2) support for both desktop and mobile UIs, (3) multi-column layout support, (4) support for multiple front-end frameworks, (5) extensibility for additional front-end frameworks and component libraries, and (6) an easy-to-use application GUI. While other tools support some of these features, their combination in one unified tool is to the best of our knowledge unique.

The SketchingInterfaces tool consists of multiple components as shown in Figure 2. SketchingInterfaces uses DoodleClassifier [11] for real-time component detection and recognition from the live camera feed. DoodleClasifier is part of the Machine Learning for Artists (ml4a-ofx) toolkit, a collection of tools that enables creative coders and artists to experiment with machine learning in their work. DoodleClassifier uses the OSC protocol for real-time communication with the core SketchingInterfaces application. Each OSC message contains a list of components and each component contains a label descriptor of the detected component, its area size, its x- and y-coordinates, and its width and height (see Figure 3). A rather limited data set for training DoodleClassifier, consisting of six to ten training sketches for each supported UI component, was sufficient to achieve acceptable recognition accuracy for the prototype.



(a)

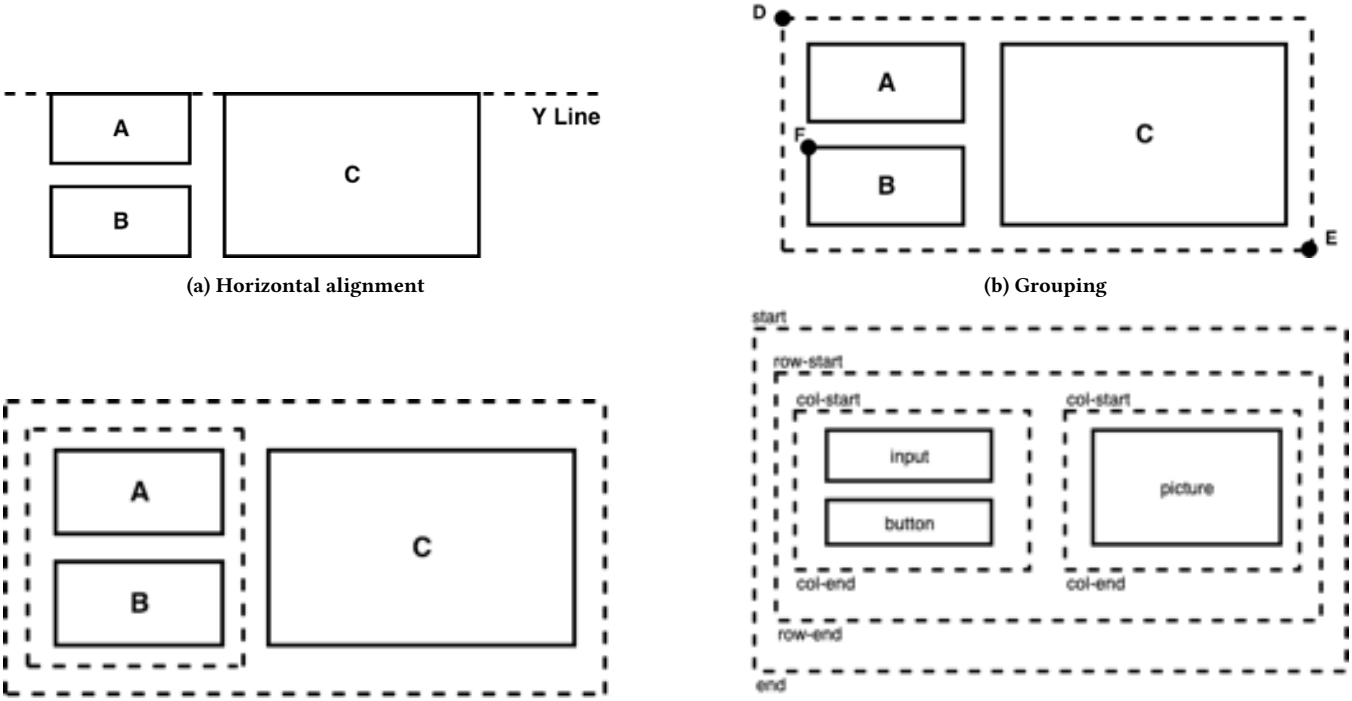
```
[ 'button', 5725, 301, 306, 120, 71 ]
```

Label descriptor | Area size | x | y coordinates | Width | Height

(b)

Figure 3: Component detection and example OSC message

The core SketchingInterfaces application is an Electron application consisting of the following parts: The LAYOUT ENGINE aligns components and places them in a precise layout grid using layouting heuristics based on the size and position of components detected by DoodleClassifier. In contrast to some existing tools



(a) Horizontal alignment

(b) Grouping

(c) Vertical alignment

(d) Generated layout grid

Figure 4: Layout processing performed by layout engine

which only support mobile app UIs with single-column layouts, the layout engine also supports desktop applications with more complex multi-column layouts. In a first step, the layout engine detects components that are horizontally aligned based on their originating y-coordinate, allowing for a certain tolerance threshold to account for imprecisions in hand-drawn sketches (see Figure 4a), and groups them in a container. It then iterates through all the other detected components to determine whether their originating x- and y-coordinates are inside the bounding box of an existing container and if they are, adds them to the corresponding container (see Figure 4b). In a final step, the components inside a container are vertically aligned in columns (see Figure 4c). The resulting layout is transmitted to the mapping engine for conversion into an HTML layout grid (see Figure 4d).

The MAPPING ENGINE translates recognized sketch components into their corresponding high-fidelity components. High-fidelity components can be provided by existing front-end frameworks (such as Bootstrap) or custom component libraries and style guides. Mappings from sketch components to high-fidelity components are defined in a dictionary in JSON format. The current version of SketchingInterfaces supports two front-end frameworks (Bootstrap and Material Design Lite), but support for additional frameworks and component libraries can be easily extended by creating new mapping dictionaries.

The resulting high-fidelity mockups are output in HTML format and displayed in an embedded WebView within the application. Users can switch between desktop and mobile viewports for mockup visualization and can toggle the display of grid guidelines to visualize the underlying layout grid.

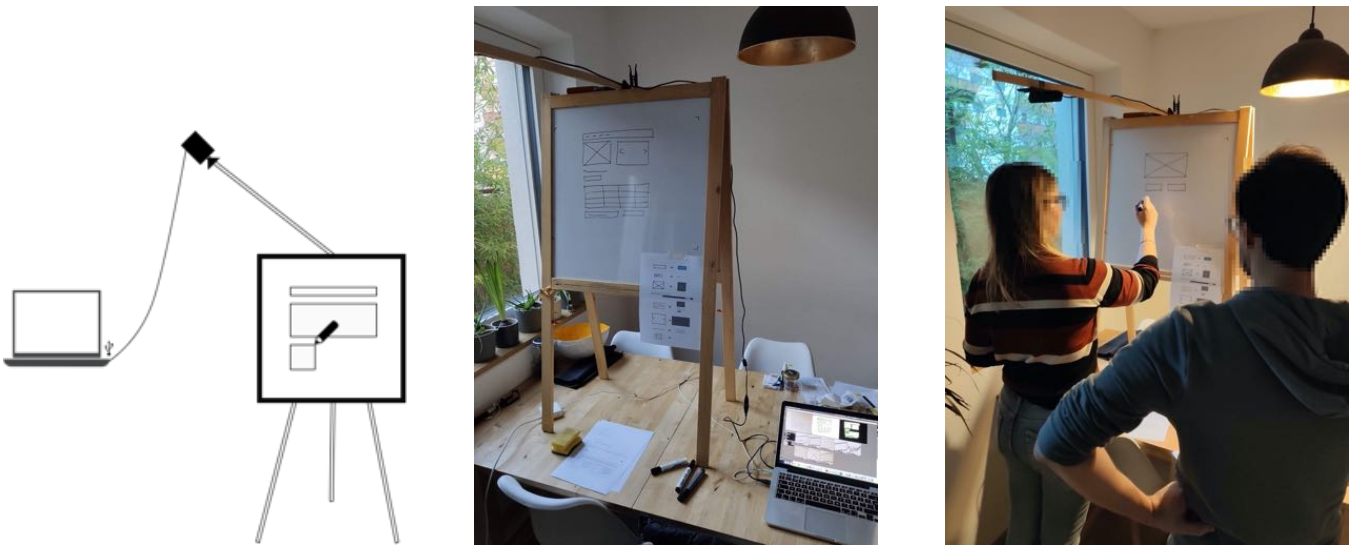
SketchingInterfaces currently supports the following UI components: button, input field, image, text, info card, carousel, navigation bar and table. Support for additional components can be added by extending the training data set of DoodleClassifier and by adding the corresponding mappings to the existing mapping dictionaries.

4 TOOL EVALUATION

To evaluate the SketchingInterfaces tool, the viability of its approach and how it influences the design process, we conducted a small-scale pilot study with six participants.

4.1 Study Design

Due to the exploratory and qualitative nature of the pilot study a non-probability purposive sampling technique was employed. Potential candidates for participation in the study were recruited among colleagues and acquaintances and those candidates were asked to refer additional potential candidates in their social circle. The six participants (2 female, 4 male; mean age 29.5) were selected to represent different skill sets and experiences with regards to user interface design and were classified in three groups: two participants were software developers, two participants were graphics designers and two participants had no professional experience in designing user interfaces. Participants with different skill sets and professional backgrounds were selected in order to identify differences in their expectations and requirements and to assess the



(a) (b) (c)

Figure 5: Study setup consisting of laptop, whiteboard and camera mounted on top of the whiteboard

usability of the tool for different stakeholders, particularly non-experts in user interface design. All participants voluntarily agreed to participate in this study and received no compensation for their participation.

Participants were provided with a short introduction of the tool, its capabilities and the supported UI components and were then instructed to complete a series of tasks. Upon completion of the tasks, participants were asked to answer a short questionnaire to gather their subjective impression of the tool and suggestions for improvement. All participants were instructed to complete a series of four tasks. Three of those tasks consisted of reproducing a user interface sketch that was provided to them (two desktop UIs, one mobile UI). These tasks were completed by each participant individually. The fourth and final task consisted of designing a simple mobile user interface for a time tracking app from scratch. Participants worked on this task in pairs in order to assess the tool in a collaborative multi-user setting and to observe collaboration strategies.

After completing the tasks, participants answered a short questionnaire consisting of four closed questions regarding the usability and utility of the tool (see Table 2 for questions and questionnaire results), to be answered on a ten-point semantic differential scale, and four open-ended questions regarding (1) missing functionality, (2) encountered errors, (3) whether the supported UI components were sufficient and suggestions for additional components and (4) general suggestions for improvement.

The study employed a laptop computer with the SketchingInterfaces tool installed and a 1080p Full HD webcam connected (see Figure 5). The camera was mounted on top of a whiteboard with a distance of 100 cm between camera and whiteboard. The position of the camera was chosen so that participants were not disturbed by the presence of the camera and could draw on the whiteboard without constraint. Participants used a black marker with 1mm stroke width to ensure reliable stroke recognition.

4.2 Evaluation Results

4.2.1 Task Completion and Errors. All six participants were able to quickly understand and use the tool as intended. All participants were able to largely fulfill the given tasks to their personal satisfaction, albeit with occasional errors in component recognition or placement. An example of task 2, consisting of the sketch for reproduction, sketches from three participants and the final output of the SketchingInterfaces tool, is shown in Figure 6. Four participants encountered errors in task 1 (3 recognition errors, 2 positioning errors), two participants encountered errors in tasks 2 and 3 (2 recognition errors in task 2, 3 recognition and 1 positioning error in task 3) and all participants were able to solve task 4 correctly without errors (see Table 1). In total, the participants sketched 129 components across all four tasks as part of the study. Of those, 121 components (93.8 %) were correctly recognized and 126 components (97.67 %) were correctly positioned within the screen layout.

4.2.2 Participant Comments and Observations. The results of the post-test questionnaire show that participants rated the tool as usable (M = 7.17, SD = 0.91) and useful for evaluating ideas and solutions (M = 8.17, SD = 0.69) on a ten-point scale (10 being best). Participants stated that the mockups generated from their sketches corresponded to their expectations (M = 7.83, SD = 1.07) and that they were able to solve the tasks to their satisfaction (M = 8.67, SD = 0.49) (see Table 2). Asked for missing functionality, one participant expressed his wish for more flexible handling of text blocks of variable length and two participants suggested more precise positioning with more fine-grained control regarding spacing between components. Regarding encountered errors, five participants observed the confusion between text input fields and navigation bars, the most common recognition error in the study. Asked whether

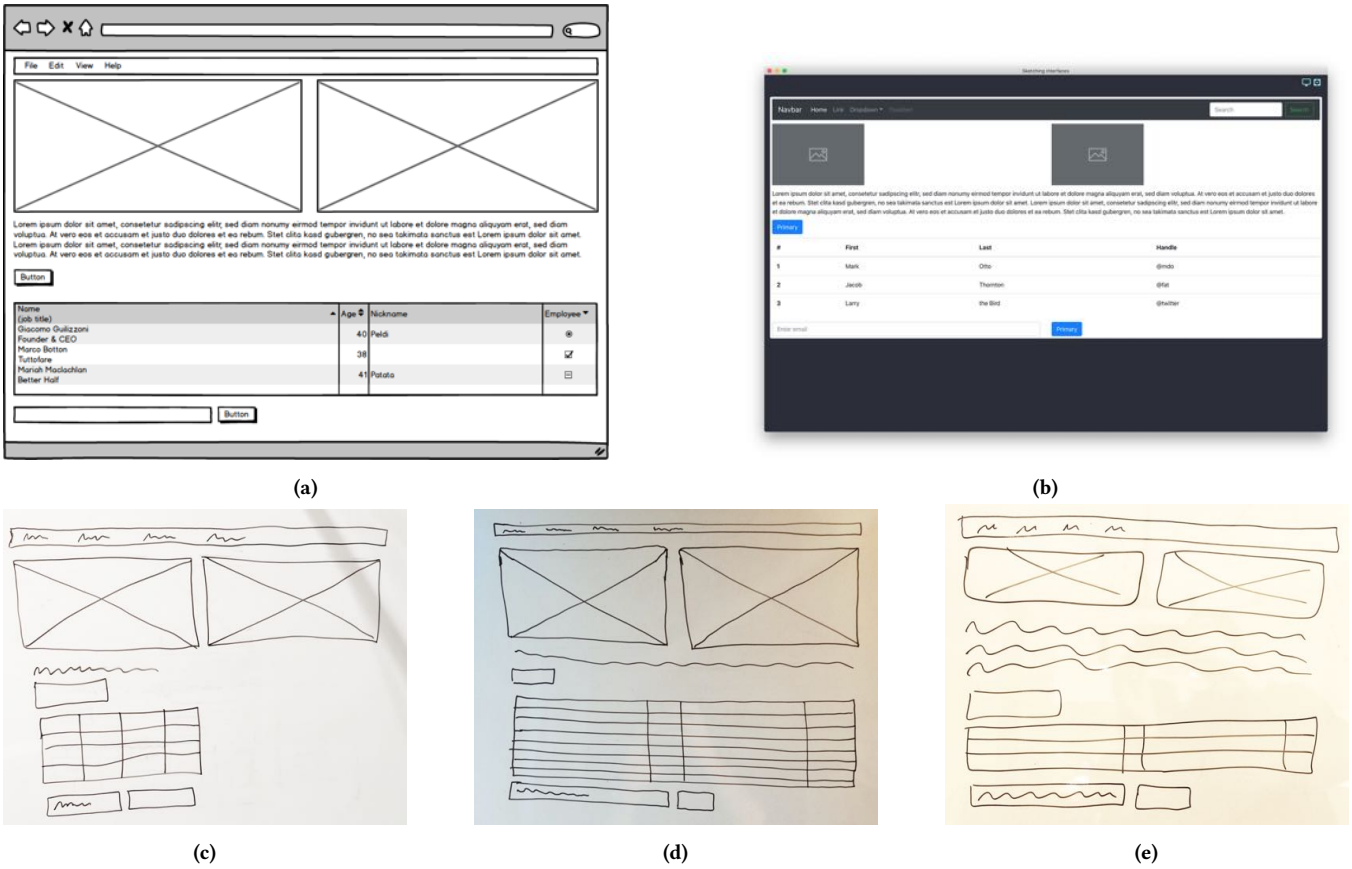


Figure 6: Example Task 2: (a) Sketch to be reproduced by participants (b) resulting output (c - e) sketches from three participants

Table 1: Task completion and number of encountered errors

	Task Completed		Number of	
	Correctly	With Errors	Recognition Errors	Positioning Errors
Task 1	2	4	3	2
Task 2	4	2	2	0
Task 3	4	2	3	1
Task 4	6	0	0	0

the supported user interface components were sufficient to solve the tasks and for suggestions for additional components, five participants found the available components to be sufficient. Suggestions for additional components included checkboxes, diagrams, password input fields and mobile optimized UI components. When questioned about suggestions for improvement, participants addressed the aforementioned encountered problem areas, as well as improvements in recognition reliability regarding environmental factors (e.g. lighting conditions) and support for customizable color schemes.

Observations from the study expectably revealed some degree of variation in drawing style between participants. For example, there were differences in how straight or wavy lines were drawn, however these differences had no noticeable impact on recognition accuracy. There were also differences in the sizing of components, especially for buttons and blocks of text, which did result in a degradation of recognition accuracy and required recalibration of detection parameters for smaller component sizes to alleviate these problems. Finally, there were also differences in the spacing between components. Placing components too close to each other caused errors in component detection as the tool conflated several individual components into one. Participants faced with these errors quickly adapted by increasing their component spacing to work around this problem. In general, participants quickly adapted to shortcomings of the tool: results improved and errors decreased with each completed task (compare Table 1).

Table 2: Questionnaire results

Question (10-point semantic differential scale)	Mean	SD
How usable is the tool? 1 = unusable | 10 = usable	7.17	0.91
How useful is the tool for evaluating ideas and problem solutions? 1 = useless | 10 = useful	8.17	0.69
Do the results generated from your sketch correspond to your expectations? 1 = deviate greatly | 10 = correspond well	7.83	1.07
Were you able to solve the tasks to your satisfaction? 1 = dissatisfied | 10 = satisfied	8.67	0.49

The fourth task, where participants created their own design solution to a given problem in pairs, was met with the greatest enthusiasm. Participants enjoyed letting their imagination and creativity run wild and all three groupings produced different solutions. All pairings exhibited a similar approach to solving the problem. They began with discussions about possible solutions before turning to sketching. The sketching itself was usually done by only one person, with the other person observing and providing feedback and comments. When the initial sketches were done, the teams discussed their solution and started making changes, e.g. by changing components or their positioning.

Regarding the different professional backgrounds and skill sets of the participants, some differences could be observed: participants with no design experience required more extensive guidance and instructions about the available UI components compared to software engineers and graphics designers. However, after this initial guidance, all participants were able to work on the tasks without problems, indicating the usability of the tool without design-specific experience or skills. In addition, the graphics designers had higher expectations regarding the quality of the design artifacts created by the tool compared to the other participants and suggested additional features for improvement, such as user-customizable fonts and color schemes.

5 DISCUSSION AND CONCLUSION

We consider the SketchingInterfaces tool in its current form to be a proof-of-concept research prototype, rather than a finished product. Despite its limitations, results from the initial study and evaluation are promising, indicating the tool's usefulness and viability of its approach. The tool is reliable and performant, supporting the real-time conversion of hand-drawn sketches into detailed, high-fidelity mockups. Given the limited size of the training data set, component recognition was surprisingly good with a correct component detection rate of 93.8 %. The layout engine also performed well, with a correct component positioning rate of 97.67 %. It must be stressed that a single, uniform set of training data was used for all participants and that the training data set was relatively small. The use of either individual, personalized training data for each participant or a larger, general training data set could possibly result in improved recognition accuracy, but was not investigated as part of this study.

Participants were largely able to solve the given tasks and were satisfied with the results. Participants quickly adjusted to shortcomings of the tool by adapting their drawing style. While ideally the tool and its component detection and recognition should be sufficiently robust to not require any adjustments on the part of the user, the results of the study suggest that users are able and willing to accommodate occasional errors, that they are able to productively work with the tool and that despite these adjustments they are still largely satisfied with the tool and its results. Of the four tasks presented to participants, the open-ended design task performed in pairs was particularly well received as it provided an opportunity to creatively engage with the task, compared to the more constrained nature of the other reproduction tasks. This indicates the tool's potential for quick, creative ideation and collaboration among peers.

5.1 Limitations and Future Work

Despite promising results from the initial study and evaluation, there remains ample room for extension and improvement: As mentioned, increasing the size of the training data set for component recognition or using personalized training data for each user could possibly result in improved recognition accuracy. Beyond changes to the training data, the use of more fine-grained or more advanced machine learning models and techniques for UI component recognition could also lead to improvements. Aside from recognition accuracy, additional features could improve the usefulness of the tool, e.g. extended support for additional front-end frameworks or settings for user-customizable fonts and color schemes (without the need to modify the front-end frameworks and component libraries directly). In addition to the currently supported live conversion and visualization of mockups, a snapshot feature allowing the capture and export of artifacts would be a worthwhile addition. As an alternative to using a hard-wired webcam, using a smartphone camera as an input source (e.g. by way of a companion app) would provide greater flexibility in tool setup and use. Furthermore, supporting the conversion of digital sketches (e.g. created with a stylus and graphics tablet) would also be feasible.

Finally, going beyond static mockups of user interfaces, the tool could be extended to support the creation of interactive prototypes using hyperlinks to navigate between individual mockups. Ideally these connections between mockups would be directly extracted from sketches and drawings, rather than requiring the manual creation of links in the application UI. For example, the tool could analyze a storyboard drawn on a whiteboard, consisting of individual screens and their interaction flows, and automatically create an

interactive high-fidelity prototype for testing and evaluation on a computer, tablet or smartphone.

Aside from technical improvements and feature additions, a more elaborate evaluation study in a realistic setting, e.g. embedding the tool in established design processes among practitioners from industry, could provide further insights into the usefulness of the tool.

REFERENCES

[1] Tony Beltramelli. 2018. pix2code: Generating code from a graphical user interface screenshot. In *Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems*. ACM, New York, NY, 1–6.

[2] Bill Buxton. 2006. What sketches (and prototypes) are and are not. In *CHI 2006 Workshop "'Sketching' Nurturing Creativity: Commonalities in Art, Design, Engineering and Research."*. ACM, New York, NY, 1–2. Retrieved July 31, 2020 from https://www.cs.cmu.edu/~bam/uicourse/Buxton-SketchesPrototypes.pdf.

[3] Bill Buxton. 2010. *Sketching User Experiences: Getting the Design Right and the Right Design*. Morgan Kaufmann, San Francisco, CA.

[4] Chunyang Chen, Ting Su, Guozhu Meng, Zhenchang Xing, and Yang Liu. 2018. From UI design image to GUI skeleton: a neural machine translator to bootstrap mobile GUI implementation. In *Proceedings of the 40th International Conference on Software Engineering*. ACM, New York, NY, 665–676.

[5] Sen Chen, Lingling Fan, Ting Su, Lei Ma, Yang Liu, and Lihua Xu. 2019. Automated cross-platform GUI code generation for mobile apps. In *2019 IEEE 1st International Workshop on Artificial Intelligence for Mobile (AI4Mobile)*. IEEE, New York, NY, 13–16.

[6] Microsoft AI Lab. 2018. Sketch2Code. Website. Retrieved July 31, 2020 from https://sketch2code.azurewebsites.net/.

[7] James A Landay. 1996. SILK: sketching interfaces like krazy. In *Conference companion on Human factors in computing systems*. ACM, New York, NY, 398–399.

[8] James A Landay and Brad A Myers. 2001. Sketching interfaces: Toward more human interface design. *Computer* 34, 3 (2001), 56–64.

[9] James Lin, Mark W Newman, Jason I Hong, and James A Landay. 2000. DENIM: finding a tighter fit between tools and practice for Web site design. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. ACM, New York, NY, 510–517.

[10] Thomas F Liu, Mark Craft, Jason Situ, Ersin Yumer, Radomir Mech, and Ranjitha Kumar. 2018. Learning design semantics for mobile apps. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*. ACM, New York, NY, 569–579.

[11] ml4a developers. 2018. DoodleClassifier. Website. Retrieved July 31, 2020 from https://ml4a.github.io/guides/DoodleClassifier/.

[12] Kevin Patrick Moran, Carlos Bernal-Cárdenas, Michael Curcio, Richard Bonett, and Denys Poshyvanyk. 2018. Machine Learning-Based Prototyping of Graphical User Interfaces for Mobile Apps. *IEEE Transactions on Software Engineering* 46, 2 (2018), 196–221.

[13] Mark W Newman and James A Landay. 2000. Sitemaps, storyboards, and specifications: a sketch of Web site design practice. In *Proceedings of the 3rd conference on Designing interactive systems: processes, practices, methods, and techniques*. ACM, New York, NY, 263–274.

[14] Tuan Anh Nguyen and Christoph Csallner. 2015. Reverse engineering mobile application user interfaces with remaui (t). In *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, New York, NY, 248–259.

[15] Jim Rudd, Ken Stern, and Scott Isensee. 1996. Low vs. high-fidelity prototyping debate. *interactions* 3, 1 (1996), 76–85.

[16] Markus Siering. 2017. Teaching a machine to convert wireframes into code. Website. Retrieved July 31, 2020 from https://tech.xing.com/teaching-a-machine-to-convert-wireframes-into-code-f9333e125e61.

[17] Sarah Suleri, Vinoth Pandian Sermuga Pandian, Svetlana Shishkovets, and Matthias Jarke. 2019. Eve: A sketch-based software prototyping workbench. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, 1–6.

[18] Uizard Technologies. 2019. Uizard. Website. Retrieved July 31, 2020 from https://uizard.io/.

[19] Khoi Vinh. 2015. The Tools Designers Are Using Today. Website. Retrieved July 31, 2020 from http://tools.subtraction.com/.

[20] Benjamin Wilkins. 2017. Sketching Interfaces: Generating code from low fidelity wireframes. Website. Retrieved July 31, 2020 from https://airbnb.design/sketching-interfaces/.