PSDoodle: Searching for App Screens via Interactive Sketching

Soumik Mohian
Computer Science and Engineering Department
University of Texas at Arlington
Arlington, Texas, USA
soumik.mohian@mavs.uta.edu

Christoph Csallner
Computer Science and Engineering Department
University of Texas at Arlington
Arlington, Texas, USA
csallner@uta.edu

ABSTRACT
Keyword-based mobile screen search does not account for screen content and fails to operate as a universal tool for all levels of users. Visual searching (e.g., image, sketch) is structured and easy to adopt. Current visual search approaches count on a complete screen and are therefore slow and tedious. PSDoodle employs a deep neural network to recognize partial screen element drawings instantly on a digital drawing interface and shows results in real-time. PSDoodle is the first tool that utilizes partial sketches and searches for screens in an interactive iterative way. PSDoodle supports different drawing styles and retrieves search results that are relevant to the user’s sketch query. A short video demonstration is available online at: https://youtu.be/3cVLHFm5pY4

CCS CONCEPTS
• Software and its engineering → Software prototyping; Search-based software engineering. • Human-centered computing → Interaction techniques.

KEYWORDS
Sketch-based image retrieval, SBIR, user interface design, sketching, GUI, design examples, deep learning

1 INTRODUCTION
App screen examples help software engineers to accumulate requirements, understand current trends, and motivate to develop a compelling mobile app [4, 5]. An effective mobile search tool might have a positive impact to keep up with the extensive and increasing use of mobile apps in day-to-day life [8].

Professional designers search for example screens via keywords through various websites including Google, Dribbble1, and Behance2 [1]. Keyword-based search tools consider stylistic features (e.g., color, style) or image metadata (e.g, date, location, shapes) but fail to consider the contents of the screen [11]. Moreover, novice users often fail to formulate good keywords and thus fail to get the desired search results [5].

Visual (e.g., image or sketch) search methods are easy, quick to adopt [20], and commonly used during early software development phases [2, 10, 16, 19]. Dependency on a complete query screen makes the iterative nature of the development process tedious (e.g., as in SWIRE [6] or VINS [1]). Long preprocessing pipelines for a pen on paper approach such as SWIRE include taking a snap, denoising, and projection correction.

PSDoodle is designed for a novice user who cannot or does not want to develop a complete screen at an early phase of software development [14]. PSDoodle allows a user to draw one UI element at a time. PSDoodle provides an interactive drawing interface with support for mouse and touch screen devices. For user interaction, the drawing interface includes basic features such as redo, undo stroke, and remove the last icon.

With each stroke on the drawing interface, PSDoodle utilizes a deep neural network to classify the current UI element and instantly provides a top-3 classification with classification confidence scores. When a user finishes drawing the current UI element by pressing the “icon done” button or “d” on the keyboard, PSDoodle searches through 58k Rico [3] screens to fetch UI examples based on UI element type, position, and element shape as shown in Figure 1. PSDoodle fetches 80 screens and displays them at the bottom of the page within 2 seconds.

For evaluation, we enlisted ten software developers who had never used PSDoodle before and have no prior UI/UX design training. Participants used PSDoodle to draw a Rico screen until it appears in the top search results. PSDoodle’s top-10 screen retrieval accuracy was comparable to the state-of-the-art full-screen drawing techniques but reduced the drawing time to one half [14].

To summarize, PSDoodle is the first tool that provides an interactive iterative search-by-sketch screen experience. While the technical-track paper describes PSDoodle’s algorithm and evaluation [14], this paper adds details about PSDoodle’s implementation, deployment, support for different sketching styles, ability to surface several relevant search result screens, and user survey results. All PSDoodle source code, processing scripts, training data, and experimental results are open-source under permissive licenses [13, 15]. The tool can be freely used at: http://pixeltoapp.com/PSDoodle/

2 BACKGROUND
PSDoodle operates on Rico mobile app screens collected from 9.3k Android apps [3]. For each UI element in each screenshot, Rico includes the Android class name, textual information, x/y coordinates, and visibility. Liu et al. collected 73k Rico screen elements

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

MOBILESof’t22, May 17–24, 2022, Pittsburgh, PA, USA
© 2022 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9301-0/22/05.
https://doi.org/10.1145/3524613.3527807

and classified all screen components in Rico into 25 UI component kinds (e.g., checkbox, icon, image, text, text button), 197 text button ideas (e.g., no, login, ok), and 135 icon classes (e.g., add, menu, share, star) [12]. PSDoodle supports several frequent Android UI elements reported by Liu et al. PSDoodle utilizes all the hierarchy information and clustering information to find similarity with a user drawing and shows Rico screens as a search result.

PSDoodle incorporates datasets from Google’s Quick, Draw [9] (“QuickDraw”) and DoodleUINet [13] to train a deep neural network for recognizing a sketched UI element. QuickDraw offers some 50M doodles of 345 everyday categories, from “aircraft carrier” to “zigzag”. DoodleUINet gives 11k crowdworker-created doodles of 16 common Android UI element categories. QuickDraw and DoodleUINet represent each doodle as a stroke sequence. Each stroke consists of a series of straight lines drawn from the start to the end of a touch event (e.g., mouse button press and un-press) and each straight line is represented by the x/y coordinates of the endpoints. As an example of how such UI element doodles look like, the left screen in Figure 1 contains six UI element doodles—two sliders, a switch, a square, and two squiggles.

3 OVERVIEW AND DESIGN

Figure 2 shows PSDoodle’s overall architecture. PSDoodle’s website provides a canvas for drawing. The canvas collects the stroke information (x/y coordinates) from a user drawing and the website sends this information to Amazon AWS. PSDoodle uses a deep neural network trained using QuickDraw’s network architecture [14, 18]. From the stroke information, a Tensorflow implementation of PSDoodle’s neural network hosted on an AWS EC2 node predicts the current drawing’s icon category.

The web browser sends all the information rendered on canvas to the Amazon EC2 instance after a user completes drawing a UI element by pressing the “icon done” button. PSDoodle’s similarity metric uses element category, shape, position, and occurrence frequency to seek up the top-N matching screens in its dictionary of Rico screen hierarchies [14]. After performing the similarity calculation, PSDoodle sends the retrieved screen names to the website. The website fetches the necessary screens from an AWS S3 bucket and displays them to the user.

3.1 PSDoodle’s Website

PSDoodle’s website is hosted on an AWS EC2 general purpose instance (t2.large) with two virtual CPUs and 8 GB of RAM. PSDoodle provides a guided interactive tutorial (http://pixeltoapp.com/toolins/) for first-time users and a cheat sheet of supported UI elements in the top-left of the main page. To make these instructions self-explanatory, we recruited two UI designers via the freelancing

Figure 1: Example mobile app screen search user query (left) and PSDoodle’s top five search results out of 58k app screenshots. The result screens contain all sketched UI elements at about the location they appear in the query sketch.
website Upwork\(^3\), recorded their tool usage and incorporated their feedback. This feedback has yielded, e.g., uniform text, position, and size of PSDoodle’s buttons (Figure 3) plus a refactored cheat text (e.g., on when and how to draw a text button, Figure 4).

Figure 3: PSDoodle drawing UI, under which PSDoodle shows its current top-N Android search result screens (omitted).

PSDoodle’s drawing interface is a canvas element with support for both mouse and touch. PSDoodle provides basic drawing features for interactions. Users can undo or redo strokes and remove the last icon (Figure 3 top left). PSDoodle’s client-side JavaScript handles all the basic events like touch-start, touch-end, and a stack of strokes to handle redo/undo features. Each time the user adds a stroke to the current icon doodle, the PSDoodle website detects the touch-end event and sends an HTTP post request with the stroke coordinates to the AWS EC2. PSDoodle’s web application parses the response from AWS EC2 and shows its current top-N icon predictions (top right). A user can pick any of these three (and tap “Icon done” or ‘d’ on the keyboard) or continue editing the current icon doodle.

After the user adds (or removes) an icon, the website submits a search query containing all recognized UI elements currently on the canvas plus their on-canVAS locations. Based on PSDoodle’s similarity metric [14] the website retrieves the top-80 screens’ ids from AWS EC2. PSDoodle’s web application parses the response from AWS EC2 and shows its current top-3 icon predictions (top right). A user can pick any of these three (and tap “Icon done” or ‘d’ on the keyboard) or continue editing the current icon doodle.

Figure 4: Cheat sheet PSDoodle shows to users: 23 graphical primitives plus compound and nested UI elements.

PSDoodle uses a deep neural network similar to QuickDraw’s network architecture [18] to identify UI element class from strokes. Three 1-D convolutional neural networks (CNN) layers followed by three Bi-LSTM layers, and a fully-connected layer make up PSDoodle’s deep neural network. PSDoodle used DoodleUINet (600 doodles for every 16 classes) and a random 600-doodle sample of each of PSDoodle’s 7 QuickDraw classes to train the deep neural network and yielded a 94.5% test accuracy.

We adopted a faster TensorFlow implementation of the deep neural network in AWS EC2 that avoids regenerating the network graph for predicting class labels for each stroke. PSDoodle guides


3.2 Recognizing Individual UI Elements

Figure 4 gives an overview of PSDoodle’s visual query language, i.e., the 23 graphical primitives PSDoodle recognizes and how the user can combine them to create compound and nested UI elements. According to the number of element labels inferred by Liu et al. [12], DoodleUINet [13] covers several of the most popular UI elements in Rico. PSDoodle uses 7 QuickDraw classes that were a suitable match for UI sketching. PSDoodle provides placeholders for text (squiggle line), image (jailwindow), and for not directly supported icons (cloud). Placeholders help the user to avoid fine details of text and images. PSDoodle provides options to draw compound UI elements. For example, the Android text button is one “text” inside a “square” drawn separately. Sketching one UI element at a time permits users to nest UI elements within a container by drawing them separately.
a user to express their drawing intention by instantly showing the updated prediction with each stroke.

4 EXPLORING PSDOODLE’S USAGE

Following the most closely related work [6, 17], we evaluated screen search performance by measuring top-k (screen) retrieval accuracy [14]. We thus showed a participant a target screen to sketch and measured where in the result ranking the target screen appears. Top-k retrieval accuracy is the most common metric for sketch-based image retrieval tasks and correlates with user satisfaction [7]. We recruited 10 Computer Science students who had no prior UI/UX design experience. Each participant first spent on average some 9 minutes in PSDoodle’s interactive tutorial (http://pixeltoapp.com/toolIns/). We then gave Rico target screens to sketch.

Table 1 shows UI category drawings with different numbers of strokes in the test dataset, showing a variety of QuickDraw’s game participants’ and DoodleUINet crowdworkers’ drawing styles. PSDoodle can thus detect sketches with high confidence for different drawing styles. As an example, in all cases when PSDoodle classified a slider test sketch that consists of five strokes, it assigned a 100% confidence that the sketch is indeed a slider. But PSDoodle similarly had correctly classified (with high confidence) other slider test sketches, e.g., those consisting of four or three strokes. The likely reason for this behavior is that PSDoodle’s classifier has been trained on sketches from a variety of people. Similarly, adding more training samples from a wider variety of crowdworkers may make PSDoodle even more robust to different drawing styles.

Table 1: Average confidence PSDoodle has in a sketch of the given total stroke count belonging to sketch’s target category. For example, the network has 50% confidence on average for completed avatar sketches of a total of 7 strokes to be avatars.

<table>
<thead>
<tr>
<th>Cat.</th>
<th>Confidence by sketch’s total strokes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Camera</td>
<td>-</td>
</tr>
<tr>
<td>Cloud</td>
<td>-</td>
</tr>
<tr>
<td>Envel.</td>
<td>100</td>
</tr>
<tr>
<td>House</td>
<td>80</td>
</tr>
<tr>
<td>Jail-win</td>
<td>89</td>
</tr>
<tr>
<td>Square</td>
<td>98</td>
</tr>
<tr>
<td>Star</td>
<td>99</td>
</tr>
<tr>
<td>Avatar</td>
<td>-</td>
</tr>
<tr>
<td>Back</td>
<td>97</td>
</tr>
<tr>
<td>Cancel</td>
<td>-</td>
</tr>
<tr>
<td>Checkb.</td>
<td>100</td>
</tr>
<tr>
<td>Drop-d.</td>
<td>-</td>
</tr>
<tr>
<td>Forward</td>
<td>100</td>
</tr>
<tr>
<td>Left-arr.</td>
<td>80</td>
</tr>
<tr>
<td>Menu</td>
<td>-</td>
</tr>
<tr>
<td>Play</td>
<td>96</td>
</tr>
<tr>
<td>Plus</td>
<td>-</td>
</tr>
<tr>
<td>Search</td>
<td>100</td>
</tr>
<tr>
<td>Setting</td>
<td>100</td>
</tr>
<tr>
<td>Share</td>
<td>-</td>
</tr>
<tr>
<td>Slider</td>
<td>100</td>
</tr>
<tr>
<td>Squiggle</td>
<td>98</td>
</tr>
<tr>
<td>Switch</td>
<td>100</td>
</tr>
</tbody>
</table>

Supporting more UI element categories is mostly a matter of gathering additional training samples and retraining the deep neural network. Regarding the overall search results (Q2), users were generally positive. Following is one quote: “good, the results were similar to what I was looking for.” Another user said the following.

“I tested different shapes the overall result was good. [...] I was overall satisfied with what I tested.”

5 CONCLUSIONS

Current approaches to searching through existing repositories are either slow or fail to address the need of novice users. Interactive partial sketching, which is more structured than a keyword search and faster than complete-screen inquiries, is a viable option. PSDoodle is the first tool to offer interactive screen search with sketching and live search results.

ACKNOWLEDGMENTS

Christoph Csallner has a potential research conflict of interest due to a financial interest with Microsoft and The Trade Desk. A management plan has been created to preserve objectivity in research in accordance with UTA policy. This material is based upon work supported by the National Science Foundation (NSF) under Grant No. 1911017.
REFERENCES


