Swire: Sketch-based User Interface Retrieval

Forrest Huang*
University of California, Berkeley
Berkeley, California, U.S.A.
forrest_huang@berkeley.edu

John F. Canny
University of California, Berkeley
Berkeley, California, U.S.A.
canny@berkeley.edu

Jeffrey Nichols
Google LLC
Mountain View, California, U.S.A.
jwnichols@google.com

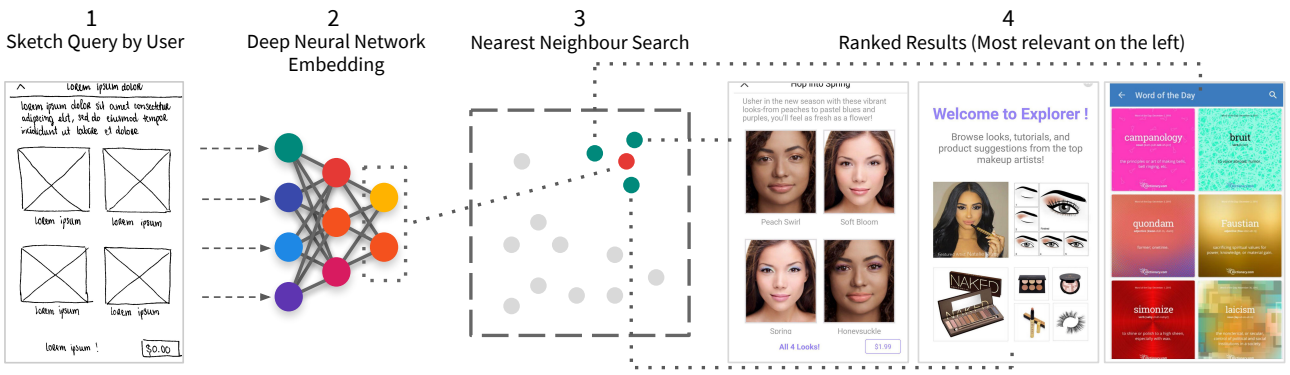

Figure 1: Overview of Swire. Swire encodes 1) a sketch query drawn by the user into 2) the sketch-screenshot embedding space using its deep neural-network. Swire then performs a 3) nearest neighbor search in the embedding space and retrieves 4) design examples that have similar neural-network outputs as the user's sketch query.

ABSTRACT

Sketches and real-world user interface examples are frequently used in multiple stages of the user interface design process. Unfortunately, finding relevant user interface examples, especially in large-scale datasets, is a highly challenging task because user interfaces have aesthetic and functional properties that are only indirectly reflected by their corresponding pixel data and meta-data. This paper introduces Swire, a sketch-based neural-network-driven technique for retrieving user interfaces. We collect the first large-scale user interface sketch dataset from the development of Swire that researchers can use to develop new sketch-based data-driven design interfaces and applications. Swire achieves high performance for querying user interfaces: for a known validation task it retrieves the most relevant example as within the top-10 results for over 60% of queries. With this technique, for the first time designers can accurately retrieve relevant user interface examples with free-form sketches natural to their design workflows. We demonstrate several novel applications driven by Swire that could greatly augment the user interface design process.

CCS CONCEPTS

• **Human-centered computing** → **Human computer interaction (HCI)**; • **Computing methodologies** → *Computer vision tasks*; • **Information systems** → Information retrieval.

KEYWORDS

data-driven design; user interface design; sketching; design examples; information retrieval; computer vision; deep learning

ACM Reference Format:
Forrest Huang, John F. Canny, and Jeffrey Nichols. 2019. Swire: Sketch-based User Interface Retrieval. In *CHI Conference on Human Factors in Computing Systems Proceedings (CHI 2019), May 4–9, 2019, Glasgow, Scotland UK*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3290605.3300334

*Work done as an intern and student researcher at Google LLC.

CHI 2019, May 4–9, 2019, Glasgow, Scotland UK

ACM ISBN 978-1-4503-5970-2/19/05.
https://doi.org/10.1145/3290605.3300334

1 INTRODUCTION

Design examples are commonly used in multiple stages of the design process. Designers search, consult and curate design examples to gain inspiration, explore viable alternatives

and form the basis for comparative evaluations [2, 9]. The recent introduction of data-driven design and the emergence of large-scale user interface (UI) datasets provide designers with large corpuses of real examples that reflect multiple facets of successful designs [11]. These examples embody rich information such as popular visual illustrations, common flow patterns and high-fidelity layout implementations [6] that can greatly augment various design tasks [12, 18, 25].

Retrieving relevant design examples from large-scale UI datasets, however, can be a daunting task. While designers can easily search for general categories of examples (e.g., UIs from all Dating Apps), conducting fined-grained searches based on visual layouts and content is much more difficult. Researchers have developed image-based UI retrieval system [27], but this technique alone is often unsuitable for retrieving UIs because of their inherent semantic complexity. A successful UI retrieval technique needs to 1) allow users to easily express their query criteria in a way that can cover both visual layout and content; and 2) match these criteria with design information obfuscated by raw pixels and code in the design examples.

Sketching is an effective visual medium for conveying design ideas. Designers use sketches to expand novel ideas, visualize abstract concepts, and rapidly compare alternatives [3]. Moreover, sketches are commonplace in design processes and typically require minimal effort for designers to produce. This suggests that sketches might be a good method by which designers could specify query criteria when searching UI corpuses, which motivates our investigation of sketch-based UI retrieval techniques in this paper.

Using sketches as the querying modality also lends itself to the recent success of machine learning techniques in recognizing visual patterns. Since both sketches and UI screenshots contain complex visual features, we can develop deep-neural-network-based models to effectively learn correspondences between sketches and UI screenshots for retrieval.

Driven by the utility of using sketching as a medium for UI retrieval and the effectiveness of machine learning vision models, this paper introduces Swire, a sketch-based UI retrieval technique powered by neural networks. This paper offers two major contributions. First, we collected the first large-scale sketch dataset consisting of 3802 sketches corresponding to 2201 UI examples from the Rico dataset [5] drawn by experienced UI designers recruited on an online freelance work platform. This dataset allows us to develop techniques capable of learning UI sketch patterns and supports future work in this area. Second, this paper introduces a versatile neural-network-based UI retrieval technique adopted from a common machine learning method used for sketch-based image retrieval. This technique enables sketches to be used by designers as a novel interaction modality to interact with large-scale UI datasets. We present a quantitative evaluation of the accuracy of the model and derive qualitative insights from sample queries, expert evaluation, and embedding values that reflect the concepts learned by the network. Furthermore, we demonstrate Swire's capability to support multiple novel sketch-based data-driven design applications that tightly integrate into the design process. With Swire, we hope to enable design applications that help designers effortlessly gain inspirations, evaluate designs and communicate novel ideas.

2 RELATED WORK

Mobile User Interface Repositories

Design examples are commonly curated and collected by designers to explore the design space and inspire new design ideas. The value brought by these examples has led to the creation of several large-scale mobile UI repositories, such as UXArchive [1], ERICA [6] and Rico [5]. Swire is developed by collecting a dataset of sketches from designers based on designs found in Rico.

The Rico dataset consists of 72,219 unique UI examples from 9,722 Android apps. Each example consists of a screenshot and the corresponding view hierarchy data. Since Rico is crowdsourced from real crowdworkers' interactions with the app, Rico also captures the user interaction information (e.g., clicks that lead to the current interface) during the data collection process.

The Rico dataset has been shown to be useful in a number of machine-learning driven applications. The most notable application is a deep auto-encoder model that retrieves similar UIs based on the position of text and non-text elements [5]. While this model demonstrates Rico's applicability on deep learning techniques, Swire explores retrieval with sketching, a modality that encapsulates richer requirements for a machine-learned model.

Sketch-based User Interface Authoring and Manipulation

Sketch-based interactions are commonly used in the early stages of the design process [17]. Thus, HCI researchers have explored sketch-based design applications to support these stages. SILK [13] is the first system that allows designers to author interactive, low-fidelity UI prototypes by sketching. DENIM [14] allows web designers to prototype with sketches at multiple detail levels.

Inspired by the effectiveness of sketching for authoring and manipulating UI designs, Swire uses sketching as a interaction modality that allows designers to naturally query for relevant examples from large-scale UI datasets.

Sketch-based Image Retrieval and Sketch Datasets

Sketch-based Image Retrieval is a frequently studied problem in the Computer Vision community. The standard sketch-based image retrieval task involves users creating a simplistic sketch with binary strokes depicting minimal user-defined features of the target natural image. For instance, when a user desires to retrieve an image of a bird in a certain pose, the user would only sketch the outline of the target body of the bird and lines that delineates the bird's wing.

Since users often focus on the target objects within the images when attempting to retrieve these images, typical approaches in prior work are to first obtain an edge-map of the original image that delineates the boundary between the (foreground) object and the background scene using an edge-detection technique, such as the Canny Edge Detector [4]. These approaches then match the edge-map with the user-created sketch using image similarity techniques. Researchers have developed a variety of image similarity metrics to improve retrieval performance, from the basic Peak Signal-to-Noise Ratio (PSNR) [26] to the more advanced Bag-of-words (BoW) Histogram of Oriented Gradients (HOG) filters [10].

With the recent increasing popularity of neural networks and crowdsourcing, researchers have developed large-scale sketch datasets that correspond to natural image datasets to power neural-network-driven techniques for image-retrieval tasks. The TU-Berlin [8] and Sketchy [22] sketch datasets consist of crowdsourced sketches that are collected from crowdworkers by presenting them the original corresponding natural images. Using these corresponding sketch-image pairs, neural networks are trained to directly encode matching sketches and images to similar low-dimensional outputs. When retrieving images with a sketch query, the natural images are ranked by the distance (e.g., Euclidean Distance) between their neural-network outputs and the sketch query's outputs.

Swire is greatly influenced by neural-network-based image retrieval methods described above. In addition, we explore the use of BoW-HOG filters as a baseline method to demonstrate and contrast the high effectiveness of neural-network-based Swire for retrieving UIs using sketches.

3 USER INTERFACE SKETCH DATASET

Our approach to recognizing and deriving patterns from sketches requires a dataset of actual sketches stylistically and semantically similar to designers' sketches of UIs. To our knowledge, no large-scale public datasets of UI sketches are currently available, especially coupled with corresponding screenshots of real-world UIs. Hence, we collected sketches created by designers based on screenshots of original UIs in the Rico dataset. We hope to support the development of future sketch-based data-driven design applications by releasing the dataset. The dataset is available at https://github.com/huang4fstudio/swire.

Designer Recruitment and Compensation

We recruited 4 designers through the freelancing platform Upwork. All designers reported having at least occasional UI/UX design experience and substantial sketching experience. In addition, all designers reported receiving formal training in UI design and degrees in design-related fields. They were compensated 20 USD per hour and worked for 60-73 hours.

Dataset Statistics

We collected 3802 sketches of 2201 UI examples from 167 popular apps in the Rico dataset. Each sketch was created with pen and paper in 4.1 minutes on average. Many UI examples were sketched by multiple designers. 71.0% of the examples were sketched by 2 designers, 28.1% of the examples were sketched by 1 designer and the remaining examples (< 1%) were sketched by 3 designers in our dataset. Our 4 designers sketched 505/1017/1272/1008 UIs respectively based on their availability. We allocated batches of examples to different combinations of designers to ensure the generality of the dataset.

We did not have the resources to generate sketches for every UI in the Rico dataset, so we curated a diverse subset of well-designed UI examples that cover 23 app categories in the Google Play Store and were of average to high design quality. We omitted poorly designed UIs from the dataset because of the relative small size of the dataset for neural network training. Noise introduced into training by poor designs had the potential to negatively impact the training time and quality of our model.

Data Collection and Postprocessing Procedure

We supplied the screenshots of our curated examples to the recruited designers and asked them to create sketches corresponding to the screenshots with pen and paper. They were prompted to reconstruct a low-fidelity sketch from the screenshot as if they were the designers of the interfaces. We instructed them to replace all actual image content in the screenshot with a sketched placeholder (a square with a cross or a mountain) and replace dynamic text in the screenshot with template texts as shown in Figure 2. We added these instructions to obtain sketches with a more unified representation focused on the design layout of various UIs. These instructions also make it easier for the neural network to learn the concepts of images and text within the constraints of our small dataset.

In order to efficiently collect and calibrate sketches created by multiple designers in various formats of photos and

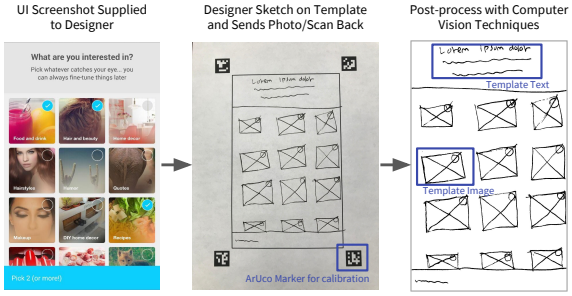

Figure 2: Data Collection Procedure. We first send a UI screenshot (Left) and paper templates with ArUco markers to designers. Designers then sketch on the template and sends back a photo or a scan of the completed sketch (Middle). We then post-process the photo using Computer Vision techniques to obtain the final clean sketch dataset (Right).

scans, we supplied them with paper templates with frames for them to sketch on as shown in Figure 2. These frames are annotated with four ArUco codes [16] at the corners to allow perspective correction. All photos and scans of the sketches are corrected with affine transformation and thresholded to obtain binary sketches as final examples in the dataset.

4 DEEP NEURAL-NETWORK-BASED USER INTERFACE RETRIEVAL

The main component of Swire is a deep convolutional neural network. The development of Swire consists of a training phase and a querying phase. During the training phase, we train Swire's deep neural network to generate similar low-dimensional outputs (64-dimensions) for matching pairs of screenshots and sketches, and dissimilar outputs for non-matching pairs of screenshots and sketches. This training scheme is shown to be useful for sketch-based image retrieval [22]. In the querying phase, we use Swire's trained neural network to encode a user's sketch query and retrieve UIs with the closest output to the user's sketch's output.

Many other best alternative solutions to sketch-based image retrieval mentioned in Section 2 use fixed image features of the original image extracted with edge detection methods. These methods may work for certain types of UI designs that exhibit strong edges, such as a grid-based photo viewer, but this approach can be inadequate when the sketches of the UIs do not directly correspond to the edges. For example, list-based UIs without clear dividers will have edge-maps which correspond less to their sketches compared to their grid-based counterparts with clear dividers.

Swire's adoption of cross-modal embedding training has the advantage that it creates a unified embedding space for both sketches and UIs with learned concepts based on their correspondences. This means Swire can be used to search a dataset of UIs using either sketches or actual screenshots as the querying modality.

Network Architecture

Since the system is required to match correspondence between images, we used two convolutional sub-networks to handle the two inputs of sketch-screenshot pairs.

These two sub-networks are similar to VGG-A [24], a shallow variant of the state-of-the-art network that won the ILSVRC2014 image recognition challenge [21]. Our network consists of 11 layers, with five convolutional blocks and three fully-connected layers. Each convolutional block contains two (one for the first two blocks) convolutional layers with 3x3 kernels and one max-pooling layer. The convolutional layers in the five blocks have 64, 128, 256, 512, and 512 filters respectively. The first two fully-connected layers have 4096 hidden units. The last layer has 64 hidden units and outputs the 64-dimension embedding used for querying. The activation functions of all layers except the last layer are ReLU. The network architecture is described in detail in Figure 3.

The final 64-dimensional output embeddings of the sub-networks are trained to produce adequate embeddings represented as codes in the last layer. The model is trained with a pairwise sampling scheme described in the following subsection.

Triplet Loss

The model is trained with a Triplet Loss function [23, 28] that involves the neural-network outputs of three inputs: an 'anchor' sketch s, a 'positive' matching screenshot i and a 'negative' mismatched screenshot i'. This forms two pairs of input during training. The positive pair $p(s,i)^+$ consists of a sketch-screenshot pair that correspond to each other. The negative pair $p(s,i')^-$ consists of a sketch-screenshot pair that does not correspond. The negative pair is obtained with the same sketch from the positive pair and a random screenshot sampled from the mini-batch.

During training, each pair $p(s,i)$ is passed through two sub-networks such that the sketch sample s is passed through the sketch sub-network and outputs an embedding $f_s(s)$, and we similarly obtain the neural-network output of the screenshot $f_i(i)$. We compute the l^2 distance D between the neural network outputs. For the positive pair,

$$D(p(s,i)^+) = ||f_s(s) - f_i(i)||_2$$

Similarly, for the distance of the negative pair,

$$D(p(s,i')^-) = ||f_s(s) - f_i(i')||_2$$

With these distances, we formulate a triplet loss function,

$$L = D(p(s,i)^+) + \max(0, m - D(p(s,i')^-))$$

m = margin between positive and negative pairs

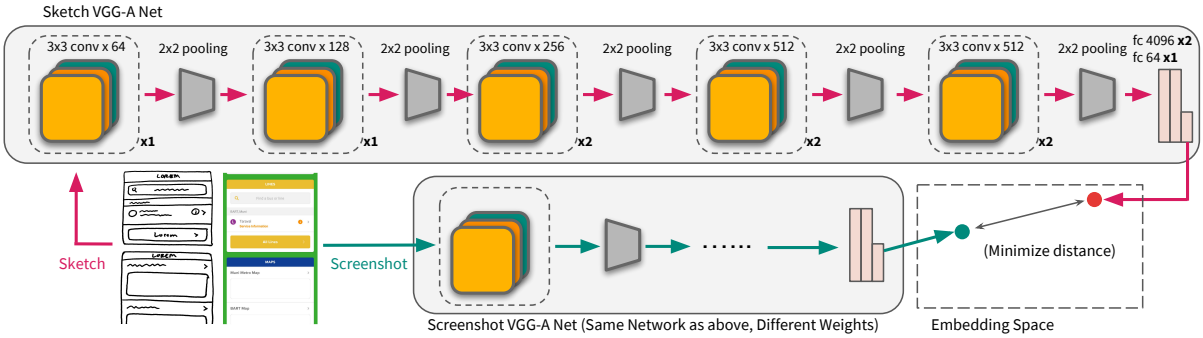

Figure 3: Network Architecture of Swire's Neural Network. Swire's neural network consists of two identical sub-networks similar to the VGG-A deep convolutional neural network. These networks have different weights and attempts to encode matching pairs of screenshots and sketches with similar values.

We maintain a margin m between the positive and negative pairs to prevent the network from learning trivial solutions (zero embeddings for all samples).

Data and Training Procedure

Since we collected data from four separate designers, we split the data and used data collected from three designers for training and from one designer for testing. This is to ensure that the model generalizes across sketches produced by different designers. In addition, we do not repeat interfaces from the same apps between the training and test sets. This creates 1722 matching sketch-screenshot pairs for training and 276 pairs for testing.

During training, the sketches and screenshots are resized to 224×224 pixels, and the pixel values are normalized between $(-1, 1)$ centered at 0. The network is trained using a Stochastic Gradient Descent Optimizer with a mini-batch size of 32. The learning rate is 1×10^{-2}. The margin is 0.2 in all models. All hyper-parameters listed above were determined by empirical experiments on the training set.

Querying

When the user makes a query with a drawn sketch, the model computes an output by passing the sketch through the sketch sub-network. This output is then compared with all neural-network outputs of the screenshots of UI examples in the dataset using a nearest neighbor search. The UI results are ranked by the distance between their outputs and the user's sketch's output.

5 RESULTS

Baseline

We implement a competitive non-neural baseline to evaluate the performance of our method. As described in Section 2, typical methods of sketch-based image retrieval involve two steps: 1) extract an edge-map from the original image to be queried, 2) match the edge-map using a specific similarity metric. Using this framework, we first extracted the edges of the screenshots using the Canny Edge detector [4]. We then extracted features from the edges using Bag-of-words (BoW) Histogram of Oriented Gradients (HOG) filters. BoW-HOG filters is an advanced method of computing similarity between images. It captures edge features in an image by computing the magnitude of gradients across the entire image with respect to multiple orientations. This method summarizes image features with fixed-length vectors that describe the occurrences and characteristics of edges in images. This method is highly effective for sketch-based image retrieval as it focuses on the characteristics of edges while being insensitive to local translations and rotations.

After obtaining these fixed-length vectors, we compare them using Euclidean Distance as a simple metric to obtain similarity values between images, and subsequently use these values to query for closest matching images (design screenshots in our case) to the sketch queries.

Quantitative Results

We use a test set that consists of 276 UI examples to compare Top-1 and Top-10 performances of BoW-HOG filters and Swire. The results are summarized in Table 1. We observe that Swire significantly outperform BoW-HOG filters for Top-10 performance at 60.9%. For Top-1 accuracy, Swire achieves an accuracy of 15.9% which only slightly outperformed the strong baseline of BoW-HOG filters at 15.6%. This shows Swire to be particularly effective for retrieving complex examples from the dataset compared to the BoW-HOG filters. We believe deep-learning-based Swire is advantageous compared to BoW-HOG filters that rely on matching edge-maps because UI sketches have semantic complexities that are not captured by edge-maps of screenshots.

Technique	Top-1	Top-10
(Chance)	0.362%	3.62%
BoW-HOG filters	15.6%	38.8%
Swire	**15.9%**	**60.9%**

Table 1: Top-k Accuracy of Various Models on the Test Set. Swire significantly outperforms BoW-HOG filters.

Qualitative Results

We visualize query results from the test set to qualitatively understand the performance of Swire in Figure 4. Swire is able to retrieve relevant menu-based interfaces despite the difference in visual appearance of the menu items (Example a). Swire is also able to retrieve pop-up windows implemented in various ways despite the drastic difference in the dimensions of the pop-up windows (Example b). We observe similar efficacy in retrieving settings (Example c), list-based (Example f), and login layouts (Example e). Nevertheless, we observe that Swire sometimes ignores smaller details of the interfaces described by sketched elements. This limitation will be further discussed in Section 7.

Expert Evaluation

To better evaluate Swire's performance from professional users' perspectives, we recruited 5 designers on Upwork with substantial experience in mobile UI/UX design to evaluate selected results from the test set. There was no overlap between these designers and those recruited for creating the dataset. We provided them with 9 sets of query sketches and the corresponding Top-10 retrieved results for each query from the test set. The 9 sets consist of 3 'best' results (the corresponding screenshot of the sketch query is retrieved as the Top-1 result), 3 'mediocre' results (the corresponding screenshot of the sketch query is retrieved within the Top-10 results, but not Top-1), and 3 'poor' results (the corresponding screenshot of the sketch query is not retrieved within the Top-10 results). We asked the designers to provide comments on each set of results regarding the relevance between the sketches and the screenshots, and to comment on the potential integration of this tool into their design workflows.

Most designers agreed that all retrieved results in the 'best' result sets are relevant to the query, and they would be satisfied with the results. They were especially satisfied with a result set of sliding menus (also shown in Figure 4a). They were able to identify the results as 'variations on the theme of navigation drawers' (D3) or 'slide out modal pattern.' (D2) Moreover, the designers also expressed satisfaction towards some sets of 'mediocre' results. Most were satisfied with a set of results that 'show variations of the top tabbed navigation' (D5) which is a common design pattern.

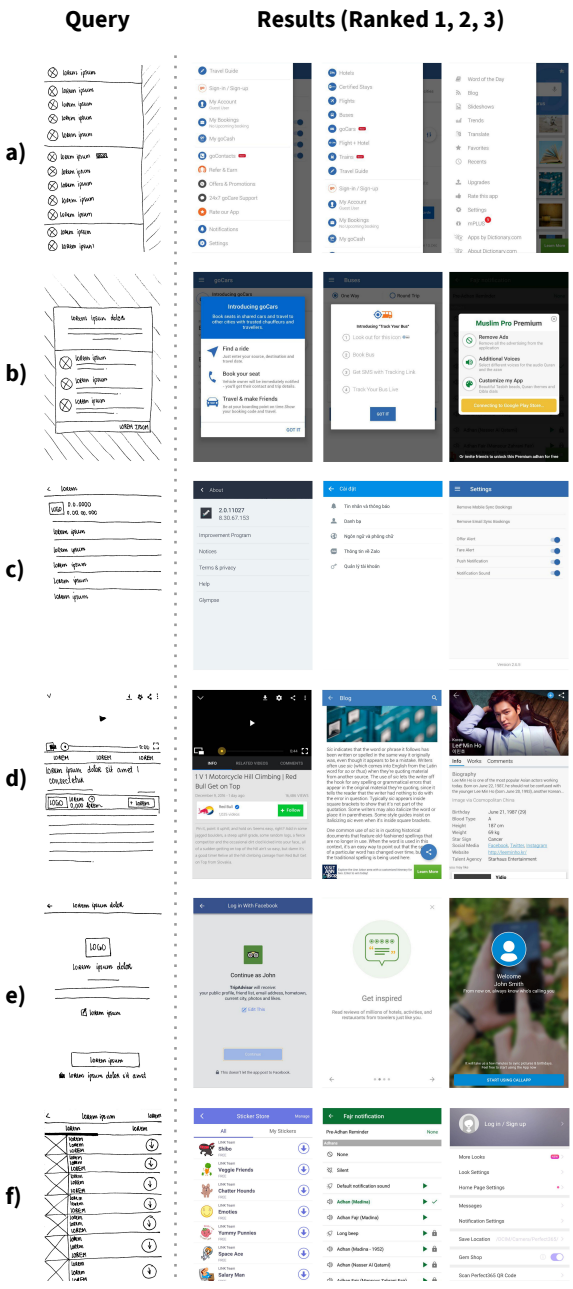

Figure 4: Query Results for Complete Sketches. Swire is able to retrieve common types of UIs such as sliding menus (a), settings (c), and login (e) layouts.

On the other hand, some designers considered the 'poor' results unsatisfactory. For example, designers were less satisfied with the model's performance on a sign-up sketch, commenting that the model only gathered screens with similar element layouts while ignoring the true nature of the

input fields and buttons in the query (D3). However, D4 considered 'rows of design elements' common in the results relevant to the sketch, and D1 considered two similar sign-up screens retrieved by the model as strong results even they did not match up perfectly with the sketch.

In general, we observed that designers were more satisfied with the results when the model was able to retrieve results that are semantically similar at a high-level instead of those with matching low-level element layouts. Notably, D1 commented that we 'probably already considered the common UI sketch patterns and train' our 'system to match it up with image results,' which reflects the effectiveness of Swire in detecting common UI patterns in some instances provided that it was not specifically trained to recognize these patterns. All designers also considered Swire to be potentially useful in their workflows for researching, ideating and implementing novel designs.

Embedding Understanding

Since we obtained fixed-length embedding values by encoding all UIs in the dataset with our trained networks, we project each data point in the training set in this embedding space to a 2-D figure using the dimensionality reduction method t-SNE [15] (see Figure 5). t-SNE is an optimization-based method that is capable of constructing projections that faithfully present distances between data points in the original dimensionality. We find clear clusters of sliding menus, Facebook login screens and onboarding screens, which suggests that Swire is categorizing UIs at least in part by their overall structure.

6 APPLICATIONS

In Section 5, we evaluated and validated Swire's effectiveness for generally finding design examples through sketch-based queries. Since both sketches and UI design examples are commonly used in early stages of the user interaction design process as reported by a variety of prior studies [9, 17], we explore the potential usage of Swire through several design applications in this section. Prototypes of these applications implemented with the Jupyter Notebook are available at https://github.com/huang4fstudio/swire.

Auto-completing Partial Designs

Sketches are often used for rapid exploration of potential design solutions [3]. Designers use partial sketches to express core ideas, while leaving out parts of the interface in sketches for considering viable design alternatives. We trained an alternative model Swire-segments on partial sketches of UIs, which allows us to 'auto-complete' the remaining UI by retrieving a variety of examples that are only required to match

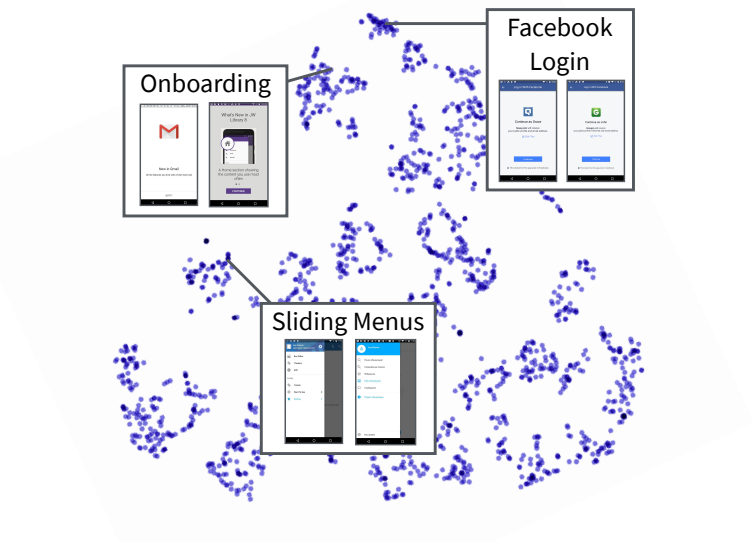

Figure 5: t-SNE Plot of the Embedding Space. Swire is able embed semantically similar UIs into multiple clusters. For instance, tutorial screens, onboarding screens and sliding menu each form their own clusters in the embedding space.

parts of the UI sketched by the user. This model allows designers to quickly gain design inspirations that are relevant to the key UI elements desired by them.

In the training and querying phases of Swire-segments, UI examples are split into small parts. Designers can thus specify one-or-more parts of the UI to be matched by the model with the examples in the dataset. We compute an embedding for each part of the interface and match only the embeddings of the parts specified by the users for retrieval. Example a in Figure 6 demonstrates that Swire-segments is able to retrieve multiple designs that all contain the Floating Action Button (FAB, a popular Android design paradigm) but with diverse layouts. Swire-segments is also able to retrieve interfaces with only tab-based top bars in common (see Example b). These examples show that Swire-segments is able to remain agnostic to the unspecified part of the sketch queries.

Evaluation with Alternative Designs

Designers often explore alternative design examples to support the implementation and comparative evaluation [9] of their own designs. HCI research literature also recommends the use of parallel prototyping techniques to obtain better final products through extensive comparison [7]. Swire is able to support design comparisons because it enables querying for similar UIs with high-fidelity UI prototypes.

Swire is effective in retrieving similar UIs because the visual content of UI screenshots are reinforced with the semantic structure of sketches in the embedding space during

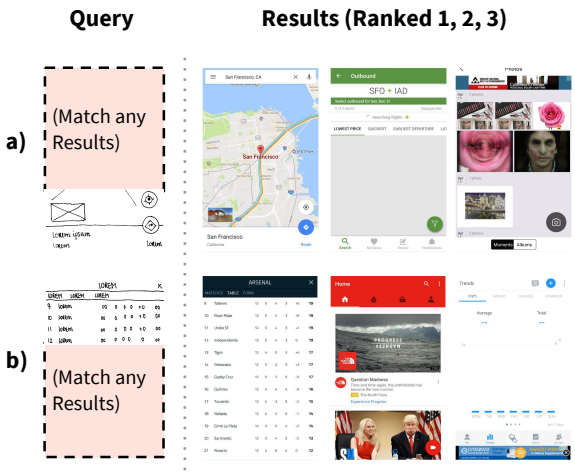

Figure 6: Autocomplete Query Results. Swire is able to retrieve interfaces only based on parts specified by users' sketches while remaining agnostic to other parts of the UIs.

training. Swire can thus be used as a semantically-aware similarity metric between interfaces.

Figure 7 shows that Swire retrieves similar menus (Example a), login screens (Example b), list-based UIs (Example c), and grid-based UIs (Example d) when querying with high-fidelity screenshots. Most notably, Swire is able to retrieve multiple types of list-based UIs despite differences among the individual items within the lists in Example c. This enables effective comparison between similar designs with slight variations.

User Flow Examples

Beyond querying for single UIs, designers also use sketches to illustrate user experience at multiple scales [17], such as conveying transitions and animations between multiple interfaces. Since the Rico dataset also includes user interaction data, we use this data to enable flow querying with Swire. Designers can use this application to interact with interaction design examples that can accelerate the design of effective user flows.

To query flow examples in the dataset, since Swire creates a single embedding for each UI, we can match an arbitrary number of interfaces in arbitrary order by concatenating the embedding values during the ranking process of querying. Figure 8 shows the results of querying for two sketches that occur consequently in a user interaction. Swire is able to retrieve registration (Example a) and 'closing menu' (Example b) flows that are commonly implemented by designers.

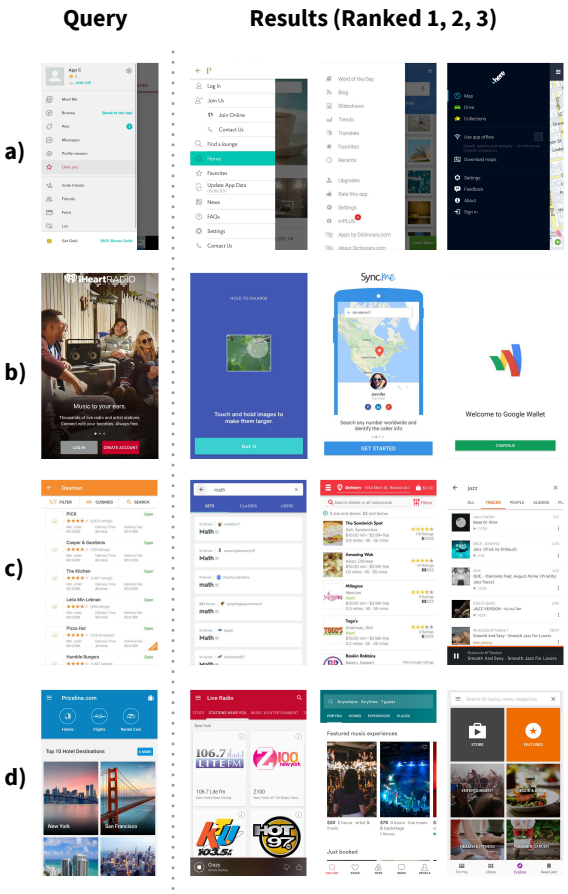

Figure 7: Alternative Design Query Results. Swire is able to retrieve similar UIs in the dataset from queries of complete, high-fidelity UI screenshots.

Since Rico also contain transition details between each consequent UIs, these examples can demonstrate popular animation patterns [6] that provide inspiration to interaction and animation designers.

7 DISCUSSION

Limitations

Despite Swire's success in retrieving relevant UI examples, we observed its inability to obtain a fine-grained semantic understanding of certain sketches. Figure 9 shows several modes of the failure cases we observed during the evaluation of Swire.

The first mode occurs when Swire handles rare, custom UI elements as exhibited by Example a. Swire failed to understand the sophisticated weather chart and retrieved another interface with similar layouts as the most relevant result with the query.

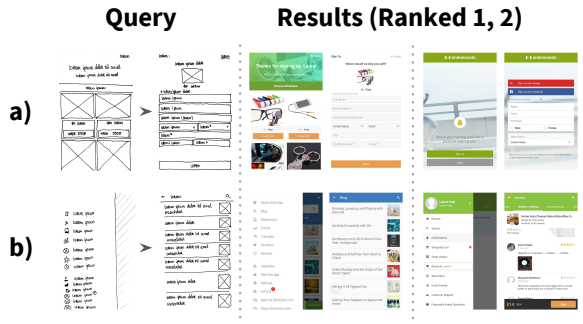

Figure 8: Flow Query Results. Swire is able to query UIs with multiple sketches concurrently to retrieve user flows.

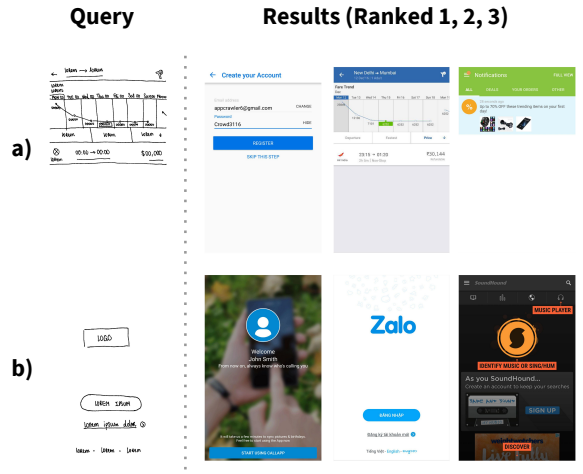

Figure 9: Failure Modes of UI Retrieval using Swire. Swire failed to understand a) custom and b) colorful UI elements.

The second mode is Swire's failure in understanding UIs with diverse colors, such as those with image backgrounds. In Example b, Swire confused a login screen with a background image, although the most relevant UI was still ranked in the second place.

Future Work

There are a number of improvements to the current Swire model that could be made in future work.

As Swire focuses primarily on the high-level layout information of the sketches, we believe that it could be improved through incorporating an understanding and control of individual elements in the sketches. One viable solution is to train an element-level sketch recognition model to recognize specific types of elements sketched by the users in certain regions, such as using a Region Proposal Network introduced in Faster-RCNN [19].

The model currently does not explicitly consider stylistic features and context information in its embedding space. In future work, contextual information can potentially be included in the model by processing content in the interfaces using topic modeling. Stylistic understanding of interfaces can be approached using feature engineering by considering visual features in the interface using style heuristics [20] and users' feedback.

The neural network in Swire currently only takes screenshots and sketches as inputs. While visual content provides some structural and semantic information about the UIs, we believe Swire can be improved by also including structured UI Hierarchy trees consisting of each element's properties as an additional input to the network. The inclusion of UI Hierarchies would add rich structural and semantic information that could potentially improve Swire's understanding of UIs.

A natural extension to this query model work would be to explore generative models that produce high-fidelity UI mock-ups from sketch-based inputs. While multiple automated methods have been developed in the past, they have failed to gain traction due to their unpredictability and the low ceiling of their generated interfaces. Recent advances in deep-learning methods for program synthesis contribute new promising results in this area, and could suggest path forward to sketch-based UI generation. We believe that the dataset contributed by this paper can support the development of such approaches to UI generation.

Finally, while this paper demonstrates Swire's capability and potential in supporting design applications, these applications are currently rough prototypes that are not yet suitable for everyday use by designers. We plan to further develop these applications and explore how they integrate into the design and software engineering processes. Studies of their usage will inform design and implementation choices, such as the visual representation of UI examples in the application and the underlying datasets to be queried.

8 CONCLUSION

This paper presents Swire, a sketch-based UI retrieval technique that enables designers to interact with large-scale UI datasets using sketches. During the development of Swire, we collected a dataset of sketches corresponding to UIs that is able to support researchers in developing further sketch-based data-driven applications. Trained on this dataset, Swire's flexible deep-learning model achieves high performance in retrieving UIs and supports multiple practical design applications. Through the development of Swire, we hope to provide designers with relevant materials and computational resources to focus on creative and innovative tasks in the design process.

ACKNOWLEDGMENTS

The authors would like to thank all reviewers for their insightful and constructive comments. The authors would also like to thank all designers on Upwork that created the sketches in the dataset and reviewed the results.

REFERENCES

[1] Arthur Bodolec, Nathan Barraille, and Chris Polk. [n. d.]. The iPhone app archive. http://uxarchive.com/

[2] Nathalie Bonnardel. 1999. Creativity in Design Activities: The Role of Analogies in a Constrained Cognitive Environment. In *Proceedings of the 3rd Conference on Creativity & Cognition (C&C '99)*. ACM, New York, NY, USA, 158–165. https://doi.org/10.1145/317561.317589

[3] Bill Buxton. 2007. *Sketching User Experiences: Getting the Design Right and the Right Design*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[4] John Canny. 1986. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence* 6 (1986), 679–698.

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

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

[7] Steven P. Dow, Alana Glassco, Jonathan Kass, Melissa Schwarz, Daniel L. Schwartz, and Scott R. Klemmer. 2010. Parallel Prototyping Leads to Better Design Results, More Divergence, and Increased Self-efficacy. *ACM Trans. Comput.-Hum. Interact.* 17, 4, Article 18 (Dec. 2010), 24 pages. https://doi.org/10.1145/1879831.1879836

[8] Mathias Eitz, James Hays, and Marc Alexa. 2012. How Do Humans Sketch Objects? *ACM Trans. Graph. (Proc. SIGGRAPH)* 31, 4 (2012), 44:1–44:10.

[9] Scarlett R. Herring, Chia-Chen Chang, Jesse Krantzler, and Brian P. Bailey. 2009. Getting Inspired!: Understanding How and Why Examples Are Used in Creative Design Practice. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '09)*. ACM, New York, NY, USA, 87–96. https://doi.org/10.1145/1518701.1518717

[10] Rui Hu and John Collomosse. 2013. A Performance Evaluation of Gradient Field HOG Descriptor for Sketch Based Image Retrieval. *Comput. Vis. Image Underst.* 117, 7 (July 2013), 790–806. https://doi.org/10.1016/j.cviu.2013.02.005

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

[12] Ranjitha Kumar, Jerry O. Talton, Salman Ahmad, and Scott R. Klemmer. 2011. Bricolage: Example-based Retargeting for Web Design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. ACM, New York, NY, USA, 2197–2206. https://doi.org/10.1145/1978942.1979262

[13] James A. Landay. 1996. SILK: Sketching Interfaces Like Krazy. In *Conference Companion on Human Factors in Computing Systems (CHI '96)*. ACM, New York, NY, USA, 398–399. https://doi.org/10.1145/257089.257396

[14] 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 (CHI '00)*. ACM, New York, NY, USA, 510–517. https://doi.org/10.1145/332040.332486

[15] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, Nov (2008), 2579–2605.

[16] Rafael Munoz-Salinas. 2012. ARUCO: a minimal library for Augmented Reality applications based on OpenCv. *Universidad de Córdoba* (2012).

[17] 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 (DIS '00)*. ACM, New York, NY, USA, 263–274. https://doi.org/10.1145/347642.347758

[18] T. A. Nguyen and C. Csallner. 2015. Reverse Engineering Mobile Application User Interfaces with REMAUI (T). In *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. 248–259. https://doi.org/10.1109/ASE.2015.32

[19] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*. 91–99.

[20] Daniel Ritchie, Ankita Arvind Kejriwal, and Scott R. Klemmer. 2011. D.Tour: Style-based Exploration of Design Example Galleries. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology (UIST '11)*. ACM, New York, NY, USA, 165–174. https://doi.org/10.1145/2047196.2047216

[21] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Fei-Fei Li. 2014. ImageNet Large Scale Visual Recognition Challenge. *CoRR* abs/1409.0575 (2014). arXiv:1409.0575 http://arxiv.org/abs/1409.0575

[22] Patsorn Sangkloy, Nathan Burnell, Cusuh Ham, and James Hays. 2016. The Sketchy Database: Learning to Retrieve Badly Drawn Bunnies. *ACM Trans. Graph.* 35, 4, Article 119 (July 2016), 12 pages. https://doi.org/10.1145/2897824.2925954

[23] Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 815–823.

[24] K. Simonyan and A. Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Representations*.

[25] Amanda Swearngin, Mira Dontcheva, Wilmot Li, Joel Brandt, Morgan Dixon, and Andrew J. Ko. 2018. Rewire: Interface Design Assistance from Examples. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. ACM, New York, NY, USA, Article 504, 12 pages. https://doi.org/10.1145/3173574.3174078

[26] S. Winkler and P. Mohandas. 2008. The Evolution of Video Quality Measurement: From PSNR to Hybrid Metrics. *IEEE Transactions on Broadcasting* 54, 3 (Sept 2008), 660–668. https://doi.org/10.1109/TBC.2008.2000733

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

[28] Qian Yu, Feng Liu, Yi-Zhe Song, Tao Xiang, Timothy Hospedales, and Chen Change Loy. 2016. Sketch Me That Shoe. In *Computer Vision and Pattern Recognition*.