

Deep-learning-based Machine Understanding of Sketches: Recognizing and Generating Sketches with Deep Neural Networks

Zifeng Huang



Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2020-13

<http://www2.eecs.berkeley.edu/Pubs/TechRpts/2020/EECS-2020-13.html>

January 10, 2020

Copyright © 2020, by the author(s).
All rights reserved.

Permission to make digital or hard copies of all or part 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. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

Acknowledgement

I would like to express my deepest gratitude towards my research advisor Professor John F. Canny, for his endless support of my work. I would like to thank my industry collaborators David Ha and Jeffrey Nichols at Google, for providing opportunities and resources that made this body of research possible. I would also like to thank Professor Björn Hartmann for providing valuable advice on this report.

Moreover, I would like to recognize my colleagues David Chan, Roshan Rao, Philippe Laban, Eldon Schoop, Daniel Seita and Jinkyu Kim, for the inspiring theoretical discussions and technical support. Finally, I am forever grateful to have my family and my closest friends Darren So, Kelvin Kwok, and Ricky Yeung, for their continuous mental support through various stages of graduate school.


**Deep-learning-based Machine Understanding of Sketches:
Recognizing and Generating Sketches with Deep Neural Networks**
by Forrest (Zifeng) Huang

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

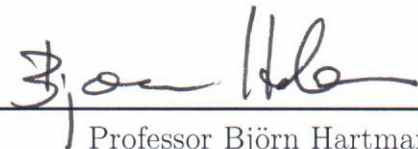
Approval for the Report and Comprehensive Examination:

Committee:



Professor John F. Canny
Research Advisor
12/18/2019

(Date)



Professor Björn Hartmann
Second Reader
1/10/2020

(Date)

Abstract

Sketching is an effective and natural method of visual communication among engineers, artists, and designers. This thesis explores several deep-learning-driven techniques for recognizing and generating sketches. We introduce two novel systems: 1) Swire, a system for querying large repositories of design examples with sketches; and 2) Sketchforme, a system that automatically composes sketched scenes from user-specified natural language descriptions. Through the development of these systems, we introduce multiple state-of-the-art techniques to perform novel sketch understanding and generation tasks supported by these systems. We also evaluate the performance of these systems using established metrics and user studies of interactive use-cases. Our evaluations show that these systems can effectively support interactive applications and open up new avenues of human-computer interaction in the domains of art, education, design, and beyond.

To Mum, Dad, and Anthony.

Contents

Contents	ii
List of Figures	iv
List of Tables	vi
1 Introduction	1
2 Related Work	3
2.1 Sketch-based Design and Image Manipulation Tools	3
2.2 Computational Assisted Sketch Authoring Tools and Tutorials	4
2.3 Sketch-based Image Retrieval and Datasets	4
2.4 Neural Sketch Generation Model and Datasets	5
3 Swire: Sketch-based User Interface Retrieval	6
3.1 Sketch Dataset	7
3.2 Deep Neural-network-based User Interface Retrieval	9
3.3 Experiments and Results	12
3.4 Applications	15
3.5 Limitations	17
4 Sketchforme: Composing Sketched Scenes from Text Descriptions for Interactive Applications	18
4.1 System Description	19
4.2 Model Training and Data Sources	22
4.3 Experiments and Results	24
4.4 Applications	30
4.5 Limitations	33
5 Conclusion and Future Work	34
5.1 Design and Engineering Applications in Other Domains	34
5.2 Conversational Sketch Suggestion and Tutorial System	34
5.3 Coloring and Animation	35

Bibliography

List of Figures

3.1	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.	7
3.2	Data Collection Procedure. We first send a UI screenshot (Left) and paper templates with ArUco markers to designers. Designers then sketch on the templates and send back photos or scans of the completed sketches (Middle). We then post-process the photos using Computer Vision techniques to obtain the final clean sketch dataset (Right).	9
3.3	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 attempt to encode matching pairs of screenshots and sketches with similar values.	10
3.4	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.	14
3.5	Query results for applications supported by Swire. Swire is able to retrieve interfaces only based on parts specified by users’ sketches while remaining agnostic to other parts of the UIs as shown in (i). Swire is also able to retrieve similar UIs in the dataset from queries of complete, high-fidelity UI screenshots in (ii). In (iii), Swire is able to query UIs with multiple sketches concurrently to retrieve user flows.	16
3.6	Failure Modes of UI Retrieval using Swire. Swire failed to understand a) custom and b) colorful UI elements.	17
4.1	Sketchforme synthesizes sketched scenes corresponding to users’ text descriptions to support interactive applications.	19
4.2	Overall system architecture of Sketchforme. Sketchforme consists of two steps in its sketch generation process.	20
4.3	Model architecture of (i) the Scene Composer and (ii) the Object Sketcher.	22

4.4	Heat-maps generated by super-positioning Sketchforme-generated/Visual Genome (ground-truth) data. Each horizontal pair of heat-maps corresponds to an object from a description.	25
4.5	Generated sketches of trees with various aspect ratios by the Object Sketcher in Sketchforme.	26
4.6	Complete scene sketches generated by Sketchforme.	26
4.7	Complete scene sketches generated by Sketchforme trained on the Abstract Scenes dataset that contains complex multi-object scenes.	27
4.8	Samples of sketches produced by humans and Sketchforme used in the AMT user study.	28
4.9	Results of the human perception user-study on Sketchforme. 64.6% of human-generated sketches and 36.5% of Sketchforme-generated sketches are perceived as human-generated in (i). In (ii), Sketchforme-generated sketches was considered more expressive than human-generated sketches for sketches of ‘a boat under a bridge.’ and ‘an airplane in front of a mountain.’	29
4.10	Applications enabled by Sketchforme. Sketchforme can augment (i) language-learning applications and significantly reduced the time taken for users to achieve similar learning outcomes. With the (ii) intelligent sketching assistant powered by Sketchforme, the user can create a partial sketch for Sketchforme to suggest multiple candidates for them to choose the adequate sketch they prefer from the description ‘a horse under a tree.’	32
4.11	Limitations of Sketchforme’s sketch generation process. In (i), the boats are significantly occluded by the bridges. In (ii), the elephants were represented with square bounding boxes which guided the system to sketch only the faces of the elephants.	33

List of Tables

3.1	Top-k Accuracy of Various Models on the Test Set. Swire significantly outperforms BoW-HOG filters for top-10 accuracy.	13
4.1	Overlap metric from Monte-Carlo simulations for each description between real data and Sketchforme-generated/heuristics-generated/random data.	25

Acknowledgments

I would like to express my deepest gratitude towards my research advisor Professor John F. Canny, for his endless support of my work. I would like to thank my industry collaborators David Ha and Jeffrey Nichols at Google, for providing the opportunities and resources that made research projects presented in this report possible. I would also like to thank Professor Björn Hartmann for providing valuable advice and comments on this body of research.

Moreover, I would like to recognize my colleagues David Chan, Roshan Rao, Philippe Laban, Eldon Schoop, Daniel Seita and Jinkyu Kim at Professor Canny's research group and the Berkeley Institute of Design, for the inspiring theoretical discussions and technical support which significantly catalyzed this body of research. Finally, I am forever grateful to have my family and my closest friends Darren So, Kelvin Kwok, and Ricky Yeung, for their continuous mental support through various stages of graduate school.

Chapter 1

Introduction

Sketching is a natural and effective way for people to communicate artistic and functional ideas. Sketches are widely used by artists, engineers, and educators as a thinking tool to materialize their vision. Sketching is also a popular form of artistic expression among amateur and professional artists. The abstract yet expressive nature of sketches enables humans to quickly communicate conceptual and high-level ideas visually. As such, sketchers can transform their intents into concrete illustrations and artifacts across multiple domains, and communicate these concepts tangibly while leaving out unnecessary details. These characteristics are most notably manifested in the use of sketches in design processes, where sketches are used by designers to iteratively discuss and critique high-level design concepts and ideas.

Because sketching is a naturalistic medium for communicating users' intent, computing systems that are able to recognize or generate sketches can support effective large-scale applications. This report introduces Swire and Sketchforme¹, two computing systems supported by modern deep-learning techniques that can recognize and generate sketches for interactive applications respectively.

Swire is a user interface (UI) retrieval system that ranks and retrieves UI screenshots from large-scale datasets. Swire allows designers to sketch their target UIs and receive relevant UI screenshots in the database for design inspiration or comparison against existing designs. We also contribute the first large-scale dataset of designer-drawn UI sketches that correspond to UI screenshots used to train Swire.

Built upon generative deep-learning models for sketches [13], Sketchforme composes sketched scenes based on natural language descriptions specified by users. Sketchforme uniquely factors the complex sketch generation task into layout composition and stroke rendering subtasks. Using these generated sketches, Sketchforme can potentially improve large-scale language learning applications by adding visual hints to foreign language phrases and

¹Swire and Sketchforme were previously published at two major conferences in the Human-Computer Interaction community. Swire was published in the proceedings of the 2019 CHI Conference on Human Factors in Computing Systems [18]. Sketchforme was published in the proceedings of the 32nd Annual Symposium on User Interface Software and Technology [17].

support sketching assistants that auto-complete sketched scenes based on users' instructions and preferences.

Chapter 2 reviews related work of computational sketch-based interactive applications in the domains of Computer Vision, Computer Graphics and Human-Computer Interaction. We then introduce and evaluate two sketch-based interactive systems mentioned above, Swire and Sketchforme, in Chapter 3 and Chapter 4 respectively. Towards the end of the report, we paint the landscape for future work and present several plausible research projects that build upon this body of research in Chapter 5. Some of these research directions are currently investigated by the author at the time of writing.

Chapter 2

Related Work

Sketching is a popular medium used across diverse fields and domains. A wide range of prior research work in Computer Vision, Computer Graphics, and Human-Computer Interaction communities have explored using sketches as input and output modalities of interactive applications, approaching this problem from both algorithmic and interaction perspectives. Such prior work demonstrates the effectiveness of sketch-based human-computer interactions and the efficacy of deep learning models in supporting sketch-based interactive tasks. This section attempts to survey a few categories of important and relevant work in literature.

2.1 Sketch-based Design and Image Manipulation Tools

Sketch-based interactions are commonly used in the early stages of the design process [32]. Thus, HCI researchers have explored sketch-based design applications to support interactions in these stages. A significant amount of prior work in this area targeted the domain of User interface/User Experience (UI/UX) design. The early work SILK [25] is the first system that allows designers to author interactive, low-fidelity UI prototypes by sketching. DENIM [30] allows web designers to prototype with sketches at multiple detail levels. More recently, researchers have integrated verbal text descriptions into crowd-sourced sketch-based UI prototyping tools [26].

Beyond the domain of UI/UX design, researchers have also developed sketch-based interactive systems supporting other design processes. DreamSketch [22] introduces a 3D sketch-based design interface that allows users to couple generative algorithms with sketch contexts for solving mechanical engineering design problems. Sketchsoup [1] automatically generates variations of users' input sketches that differ in perspective or structure, to help industrial designers more exhaustively explore the design space during the ideation process.

Perhaps more technically relevant to our work are image synthesis models developed by the Machine Learning community. These models can translate user-generated doodles into fine artwork [5] and realistic images [34]. These works utilize recent deep-learning-driven

techniques to manipulate and transform user-defined sketch-based semantics or graphics to closely-related domains.

2.2 Computational Assisted Sketch Authoring Tools and Tutorials

Prior works have augmented the sketching process with automatically-generated and crowd-sourced drawing guidance. ShadowDraw [27] and EZ-sketching [39] used edge images traced from natural images to suggest realistic sketch strokes to users. The Drawing Assistant [19] extracts geometric structure guides to help users construct accurate drawings. PortraitSketch [43] provides sketching assistance specifically for facial sketches by adjusting geometry and stroke parameters. Researchers also developed crowd-sourced web applications to provide real-time feedback for users to correct and improve sketched strokes [29].

In addition to assisted sketching tools, researchers also developed sketching tutorial systems to improve users' sketching proficiency. How2Sketch [14] automatically generates multi-step tutorials for sketching 3D objects. Sketch-sketch revolution [11] provides first-hand experiences created by sketch experts for novice sketchers.

2.3 Sketch-based Image Retrieval and 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 simplistic sketches with binary strokes depicting minimal user-defined features of the target natural images. 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 delineate 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 edge-maps of the original images that delineate boundaries between (foreground) objects and background scenes using edge-detection techniques. These approaches then match the edge-maps with the sketches created by users 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) to the more advanced Bag-of-words (BoW) Histogram of Oriented Gradients (HOG) filters.

With the recent increasing popularity of deep neural networks and crowdsourcing, researchers have developed large-scale pixel-based sketch datasets that correspond to natural image datasets to power neural-network-driven techniques for image retrieval tasks. The TU-Berlin [10] and Sketchy [36] sketch datasets consist of crowdsourced sketches drawn by crowdworkers after seeing 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 in the embedding space. 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 in the embedding space.

2.4 Neural Sketch Generation Model and Datasets

Beyond image recognition and retrieval, deep neural networks have also been used to generate sketches of various categories. The Sketch-RNN model is the first neural-network-based sketch generation model [13] that generate sketch strokes using an LSTM-based architecture. Sketch-RNN can either unconditionally generate stroke-based sketches based on object classes, or conditionally reconstruct sketches based on users' input sketches.

To facilitate the development of Sketch-RNN, researchers have crowdsourced the Quick, Draw! [21] dataset that contains sketches drawn by human users in 20 seconds according to various concept categories. The sketches are recorded at the stroke level with offsets from the previous points and stroke types (e.g., pen up, pen down), similar to a vector format.

Chapter 3

Swire: Sketch-based User Interface Retrieval

The ability to recognize and understand sketches can bring upon naturalistic and intuitive user experience in interactive systems. We believe such benefits can be potentially observed in computationally supported design applications because sketching is an effective visual medium for conveying abstract design ideas. For instance, UI/UX designers use sketches to expand novel ideas, visualize abstract concepts, and compare alternative designs [4]. Sketches also require minimal effort for designers to produce which allows them to rapidly generate inspiring and focused discussions central to high-level design ideas, without distractions from fine details. As such, designers sketch frequently in the design process, especially in the earlier stages of the process.

Another type of artifact that UI/UX designers take reference of frequently in the earlier stages of the design process is design example. Designers search, consult and curate design examples to gain inspiration, explore viable alternatives and form the basis for comparative evaluations [15, 3]. These examples embody rich information such as popular visual illustrations, common flow patterns and high-fidelity layout implementations [7] that can greatly augment various design tasks [24, 40, 33].

Retrieving relevant design examples from large-scale design 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. A successful UI design retrieval technique, for instance, 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. While keyword-based matching and pixel-based matching are either too low or high-level for this task, sketching might be a good medium for designers to use when searching UI corpuses, as it allows designers to specify coarse visual patterns while abstracting specific details from the UIs.

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, we introduce Swire, a sketch-based UI retrieval technique powered by neural networks in this chapter. To develop Swire, we collected the first large-scale sketch dataset consisting of 3802 sketches corresponding to 2201 UI examples from the Rico dataset [8] 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. We then introduce 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. The workflow of Swire is summarized in Figure 3.1.

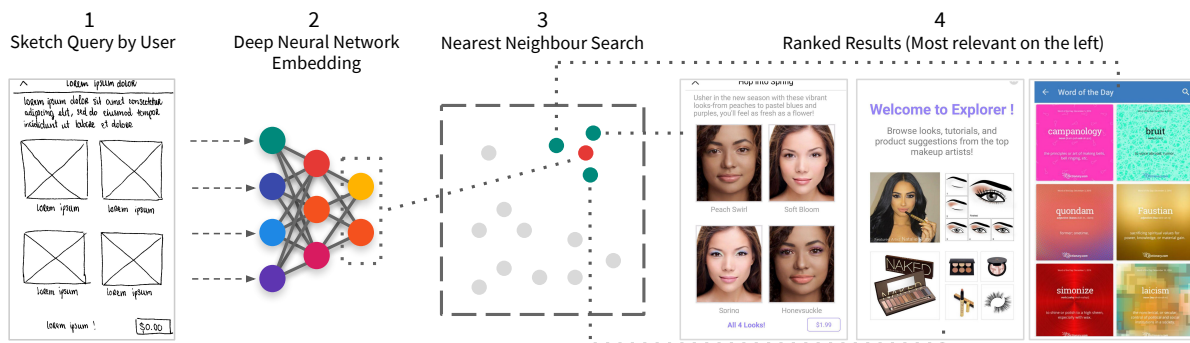


Figure 3.1: Overview of Swire. Swire encodes 1) a sketch query drawn by the user into 2) the sketch-screenshot embedding space using its deep neural network. Swire then performs a 3) nearest neighbor search in the embedding space and retrieves 4) design examples that have similar neural-network outputs as the user’s sketch query.

We also 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.

3.1 Sketch Dataset

Our approach towards 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 those with sketches and 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 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 relatively 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 screenshots of our curated UI 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 3.2. 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 scans, we supplied them with paper templates with frames

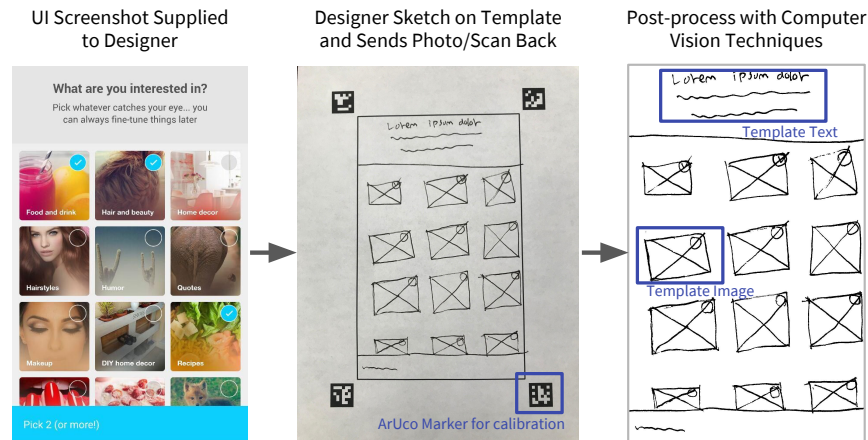


Figure 3.2: Data Collection Procedure. We first send a UI screenshot (Left) and paper templates with ArUco markers to designers. Designers then sketch on the templates and send back photos or scans of the completed sketches (Middle). We then post-process the photos using Computer Vision techniques to obtain the final clean sketch dataset (Right).

for them to sketch on as shown in Figure 3.2. These frames are annotated with four ArUco codes [31] 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.

3.2 Deep Neural-network-based User Interface Retrieval

The main component of Swire is a deep convolutional neural network. The operation 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 [36]. In the querying phase, we use Swire’s trained neural network to encode a user’s sketch query and retrieve UIs with the closest outputs to the user’s query’s output.

Many other best alternative solutions to sketch-based image retrieval mentioned in Chapter 2.3 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 [38], a shallow variant of the state-of-the-art network that won the ILSVRC2014 image recognition challenge [35]. 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 3.3.

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

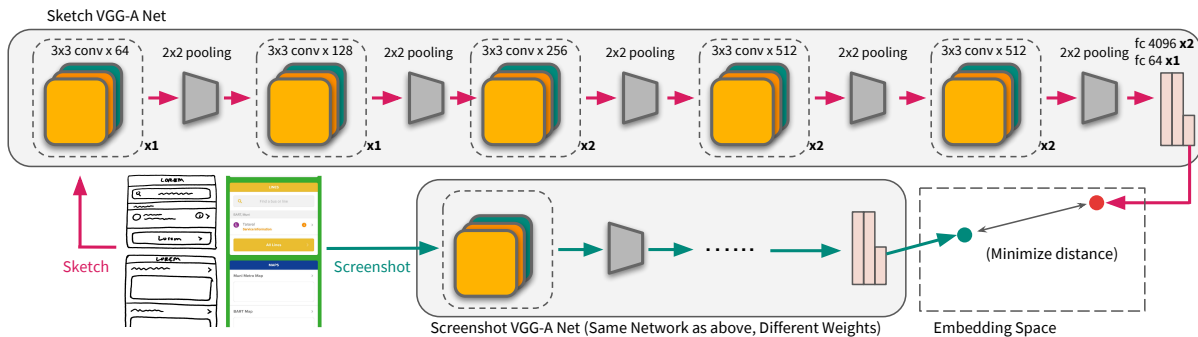


Figure 3.3: Network Architecture of Swire’s Neural Network. Swire’s neural network consists of two identical sub-networks similar to the VGG-A deep convolutional neural network. These networks have different weights and attempt to encode matching pairs of screenshots and sketches with similar values.

Triplet Loss

The model is trained with a Triplet Loss function [37, 44] 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

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.

3.3 Experiments and Results

Baseline

We implement a competitive non-neural baseline to evaluate the performance of our method. As described in Chapter 2.3, 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. We then extracted features from the edges using Bag-of-words (BoW) Histogram of Oriented Gradients (HOG) filters. BoW-HOG filters are 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 3.1. 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 3.1: Top-k Accuracy of Various Models on the Test Set. Swire significantly outperforms BoW-HOG filters for top-10 accuracy.

Qualitative Results

We visualize query results from the test set to qualitatively understand the performance of Swire in Figure 3.4. Swire is able to retrieve relevant menu-based interfaces despite the difference in the 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 Chapter 3.5.

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 3.4a). 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.

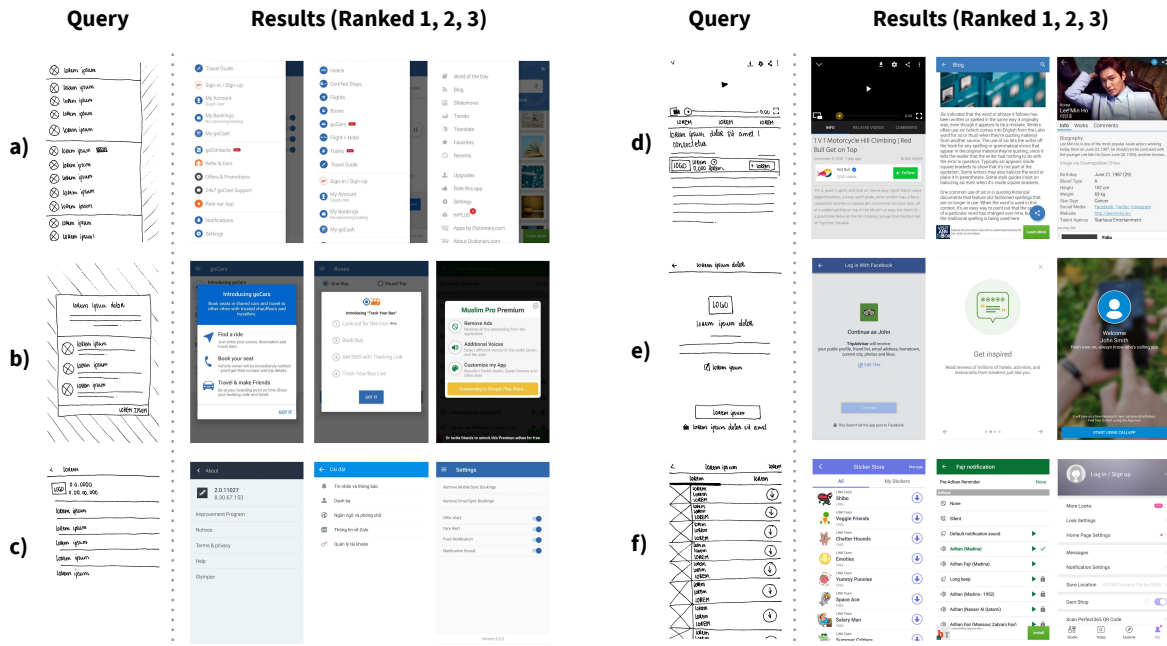


Figure 3.4: Query Results for Complete Sketches. Swire is able to retrieve common types of UIs such as sliding menus (a), settings (c), and login (e) layouts.

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.

3.4 Applications

In Chapter 3.3, 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 [32, 15], we explore the potential usage of Swire through several design applications in this chapter. 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 [4]. 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 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 3.5ii 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.

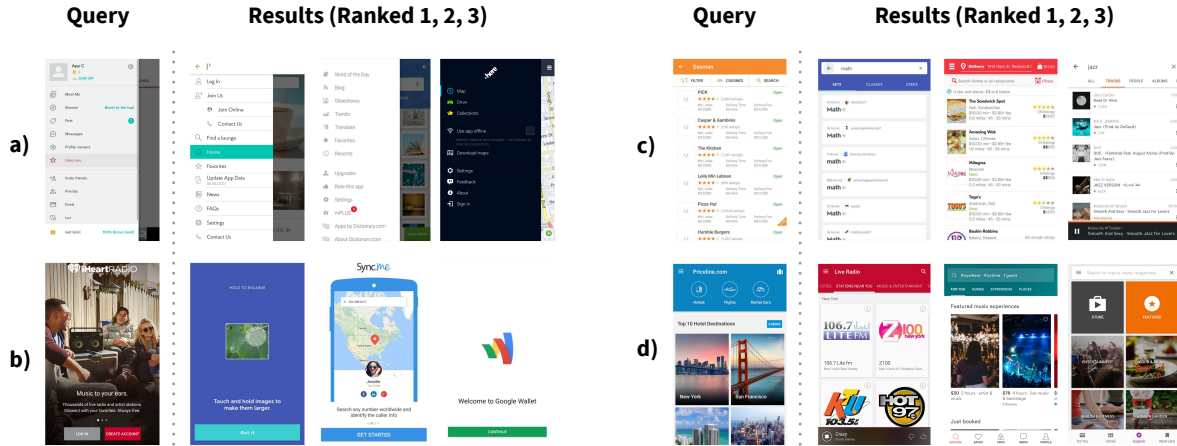
Evaluation with Alternative Designs

Designers often explore alternative design examples to support the implementation and comparative evaluation [15] of their own designs. Prior work in HCI research literature also recommends the use of parallel prototyping techniques to obtain better final products through extensive comparison [9]. Swire is able to support design comparisons because it enables querying for similar UIs with high-fidelity UI prototypes.

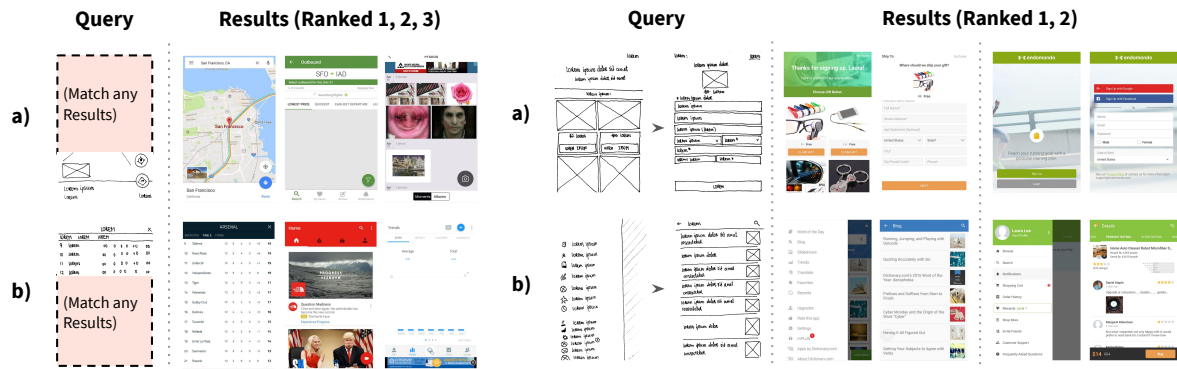
Swire is effective in retrieving similar UIs because the visual content of UI screenshots is reinforced with the semantic structure of sketches in the embedding space during training. Swire can thus be used as a semantically-aware similarity metric between interfaces.

Figure 3.5i 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.



(i) Alternative Design Query Results



(ii) Autocomplete Query Results

(iii) Flow Query Results

Figure 3.5: Query results for applications supported by Swire. Swire is able to retrieve interfaces only based on parts specified by users’ sketches while remaining agnostic to other parts of the UIs as shown in (i). Swire is also able to retrieve similar UIs in the dataset from queries of complete, high-fidelity UI screenshots in (ii). In (iii), Swire is able to query UIs with multiple sketches concurrently to retrieve user flows.

User Flow Examples

Beyond querying for single UIs, designers also use sketches to illustrate user experience at multiple scales [32], 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

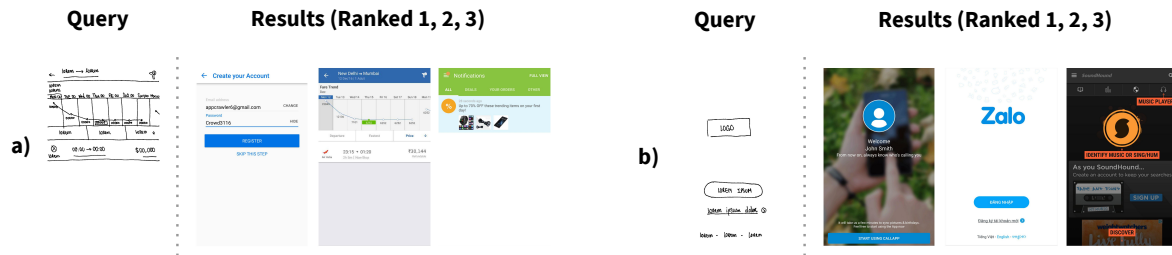


Figure 3.6: Failure Modes of UI Retrieval using Swire. Swire failed to understand a) custom and b) colorful UI elements.

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 3.5iii 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. Since Rico also contains transition details between each consequent UIs, these examples can demonstrate popular animation patterns [7] that provide inspiration to interaction and animation designers.

3.5 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 3.6 shows several modes of the failure cases we observed during the evaluation of Swire.

Rare, Custom UI Elements

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.

UI with Diverse Colors

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.

Chapter 4

Sketchforme: Composing Sketched Scenes from Text Descriptions for Interactive Applications

Beyond understanding sketches and using them as the input modality with Swire, computational systems that generate sketches and use them as outputs can lead to engaging user experiences. Having these systems produce diverse sets of sketches could further encourage the adoption of sketches for various applications, especially in applications when it is time-consuming or difficult for users to create sketches. These interactions could be useful for domains such as language learning and communication.

Recent advances in neural-network-based generative models drastically increased machines' ability to generate convincing graphical content, including sketches, from high-level concepts. The Sketch-RNN model [13] demonstrates that recurrent neural networks (RNNs) trained on crowd-sourced data can generate original sketches of various concept classes.

With the advancement in sketch-generation algorithms and the benefits of using sketches as outputs in interactive applications, this chapter introduces Sketchforme, the first system that is capable of synthesizing complex sketches for users while allowing them to maintain control over the sketches' content naturally using text descriptions. Sketchforme uses a novel, automated two-step neural method for generating sketched scenes from text descriptions. Sketchforme first uses its *Scene Composer*, a neural network that learned *high-level composition principles* from datasets of human-annotated natural images that contain text captions, bounding boxes of individual objects, and class information of the objects, to generate composition layouts of the scenes. Sketchforme then uses its *Object Sketcher*, a neural network that learned *low-level sketching mechanics* to generate sketches adhering to the objects' aspect ratios in the compositions. Finally, Sketchforme composes these generated objects of certain aspect ratios into meaningful sketched scenes.

We also build and evaluate several applications, including a sketch-based language learning system and an intelligent sketching assistant. These applications illustrate the potential value of Sketchforme in supporting novel sketch-based interactions (Chapter 4.4). In these

applications, Sketchforme creates new interactions and user experiences with the interplay between language and sketches. These features of Sketchforme are highlighted in Figure 4.1.

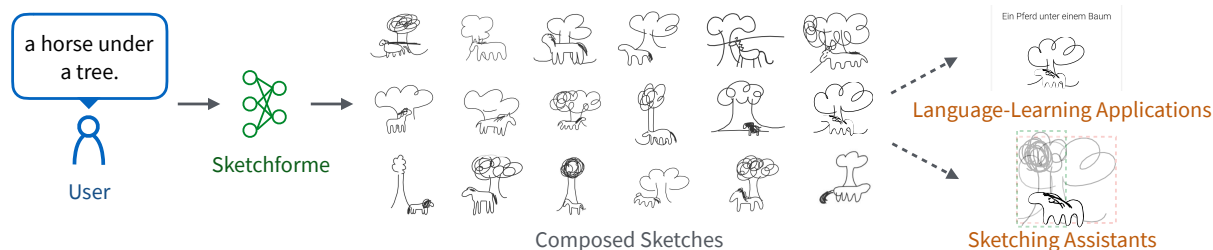


Figure 4.1: Sketchforme synthesizes sketched scenes corresponding to users’ text descriptions to support interactive applications.

4.1 System Description

To support applications that afford sketch and natural-language based interactions, we developed Sketchforme, the system that provides the core capability of synthesizing sketched scenes from natural language descriptions. Sketchforme implements a two-step approach to generate a complete scene from text descriptions as illustrated in Figure 4.2. In the first step, Sketchforme uses its *Scene Composer* to generate composition layouts represented by bounding boxes of individual objects. These bounding boxes dictate locations, sizes, and aspect ratios of objects in the scene. Sketchforme’s *Object Sketcher* then uses this information at the second step of the generation process to generate specific sketch strokes of these objects in their corresponding bounding boxes. These steps reflect a fundamental process suggested in many sketching tutorials, where the overall composition of the scene is drafted before filling in details that characterize each object [6].

By taking this two-step approach, Sketchforme is able to model high-level object relations critical to composing the scenes, enabling a multitude of applications that require such information. Moreover, this approach overcomes the difficulty for end-to-end sketch generation methods to capture global structures of sequential inputs [13]. End-to-end sketched scene generation also requires datasets of dedicated sketch-caption pairs that are difficult for crowd-workers to create [46].

Scene Composer: Generating Composition Layouts

To generate composition layouts of scenes, we first model composition layouts as a sequence of n objects (and start/end tokens), such that each object generated by the network is

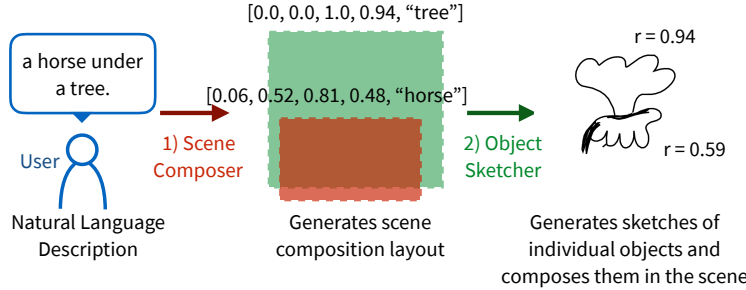


Figure 4.2: Overall system architecture of Sketchforme. Sketchforme consists of two steps in its sketch generation process.

represented with 8 values:

$$b_t = [x_t, y_t, w_t, h_t, l_t, \text{box}_t, \text{start}_t, \text{end}_t], t \in [1, n]$$

The first 5 values are fundamental data that describes bounding boxes of objects in the scene: x-position, y-position, width, height, and the class label. The last three values are boolean flags used as extra ‘tokens’ to mark the actual objects, the beginning of sequences and the end of sequences.

Using this sequential encoding of scenes, we designed a Transformer-based Mixture Density Network as our *Scene Composer* to generate realistic composition layouts. Transformer Networks [41] are state-of-the-art neural networks for sequence-to-sequence modeling tasks, such as machine translation and question answering. We use a Transformer Network to perform a novel task: generating a sequence of objects from a text description c , a sequence of words. As multiple scenes can correspond to the same text descriptions, we feed the outputs of the Transformer Network into Gaussian Mixture Models (GMMs) to model the variation of scenes, forming a Mixture Density Network [2].

The generation process of the composition layouts involves taking the previous bounding box b_{t-1} (or the start token) as an input and generating the current box b_t . At each time-step, the Transformer Network generates an output t_t conditioned on the text input c and previously generated boxes $b_{1..t-1}$ using self-attention and cross-attention mechanisms built into the architecture. This process is repeated for multiple bounding boxes until an end token is generated:

$$t_t = \text{Transformer}([b_{1..t-1}; c]) \quad (4.1)$$

t_t is then projected to the appropriate dimensionality to parameterize the GMMs with various projection layers W_{xy} and W_{wh} to model $P(x_t, y_t)$, the distribution of the bounding boxes’ positions, and $P(w_t, h_t)$, the distribution of the bounding boxes’ sizes. Sketchforme can then generate bounding boxes $[x_t, y_t, w_t, h_t]$ by sampling from these distributions in Equations 4.2

and 4.3. The GMMs use the projected values as mean and covariance parameters for mixtures of multivariate Gaussian distributions M . These values are passed through appropriate activation functions (Sigmoid, exp and tanh) to comply with the required range of the parameters.

$$P(x_t, y_t) = \sum_{i \in M} \Pi_{1,i} \mathcal{N}(x_t, y_t | \mu_1, \Sigma_1), [\Pi_1, \mu_1, \Sigma_1] = a(W_{xy}(t_t)) \quad (4.2)$$

$$P(w_t, h_t) = \sum_{i \in M} \Pi_{2,i} \mathcal{N}(w_t, h_t | \mu_2, \Sigma_2), [\Pi_2, \mu_2, \Sigma_2] = a(W_{wh}([x_t; y_t; t_t])) \quad (4.3)$$

While $P(x_t, y_t)$ is modeled only from the first projection layer W_{xy} , we consider $P(w_t, h_t)$ to be conditioned on the position of the boxes similar to [16]. To introduce this condition, we concatenate t_t and $[x_t, y_t]$ as inputs to the second projection layer as described in Equation 4.3. The probability of each generated bounding box being an actual object or a start/end token are generated using a softmax-activated third projection layer W_c from the Transformer output:

$$P(\text{box}_t, \text{start}_t, \text{end}_t) = \text{softmax}(W_c t_t) \quad (4.4)$$

In addition, Sketchforme separately uses an LSTM to generate class labels l_t because the class labels given certain descriptions are assumed to not vary across examples. The full architecture of the Scene Composer is shown in Figure 4.3i.

Object Sketcher: Generating Individual Sketches

After obtaining scene layouts from the Scene composer, we designed a modified version of Sketch-RNN model to generate individual objects in Sketchforme according to the layouts. We adopt the decoder-only Sketch-RNN that is capable of generating sketches of individual objects as sequences of individual strokes. Sketch-RNN’s sequential generation process involves generating the current stroke based on previously generated strokes, a method commonly used in sequence modeling tasks. Sketch-RNN also uses a GMM to model variation of sketch strokes.

While the decoder-only Sketch-RNN generates realistic sketches of individual objects in certain concept classes, the aspect ratios of the output sketches generated by the original Sketch-RNN cannot be constrained. Hence, sketches generated by the original Sketch-RNN may be unfit for assembling into scene sketches guided by the layouts generated by the Scene Composer. Further, naive direct resizing of the sketches can produce sketches of unsatisfactory quality for complex scenes.

We modified Sketch-RNN as the *Object Sketcher* that factors in the aspect ratios of objects when generating sketches. To incorporate this information in the Sketch-RNN model, we compute the aspect ratios of the training data and concatenate the aspect ratio $r = \frac{\Delta y}{\Delta x}$ of each sketch with the previous stroke as input to our modified Sketch-RNN in the sketch

generation process as shown in Figure 4.3ii. The new formulation and output of the modified Sketch-RNN for t -th stroke is:

$$[h_t; c_t] = LSTM([S_{t-1}; r; h_{t-1}; c_{t-1}]), y_t = Wh_t + b_t \quad (4.5)$$

Since each Sketch-RNN model only handles a single object class, we train multiple modified Sketch-RNN models based on multiple classes and use appropriate models based on class labels in the layouts generated by the Scene Composer for assembling the final sketched scene.

After generating the strokes, the Object Sketcher converts them into SVG paths and fills each object in white using the non-zero rule. The object corresponding to the first generated bounding box generated by the Scene Composer is then composed as the foreground of the sketched scene, and subsequently generated objects are placed in the background according to the order of generation.

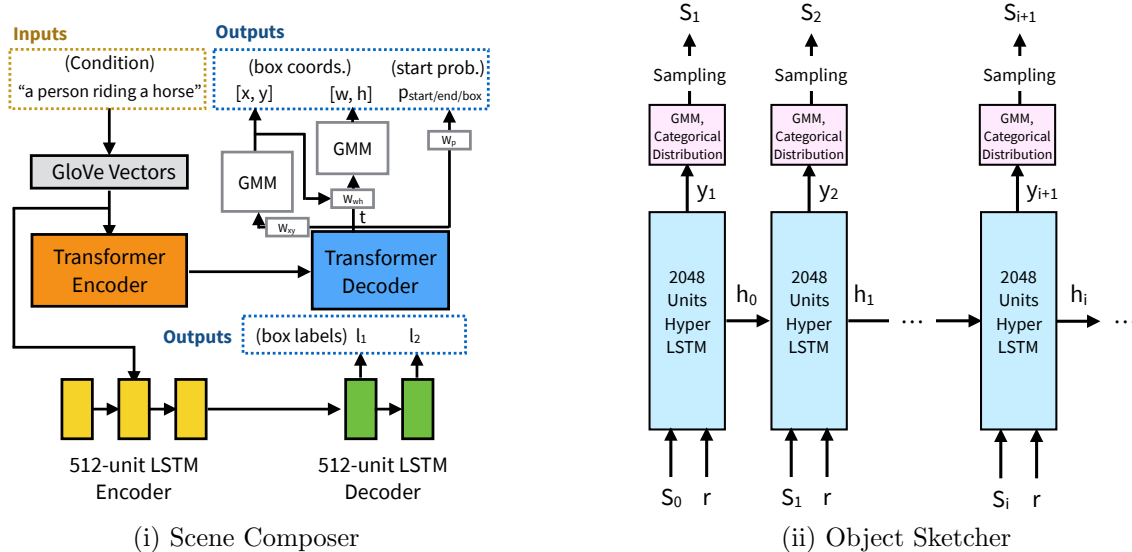


Figure 4.3: Model architecture of (i) the Scene Composer and (ii) the Object Sketcher.

4.2 Model Training and Data Sources

Sketchforme’s Scene Composer and Object Sketcher are trained on different datasets that encapsulate visual-scene-level knowledge and sketching knowledge separately. This relaxes the requirement for Sketchforme to be trained on natural language annotated datasets of sketched scenes that provide varied scenes corresponding to realistic scene-caption pairs.

We trained the Scene Composer using the Visual Genome dataset [23], which contains natural language region descriptions and object relations of natural images, to demonstrate

its flexibility in utilizing various types of scene-layout datasets. Object relations in the dataset each contains a ‘subject’ (e.g., ‘person’), a ‘predicate’ (e.g., ‘on’), and an ‘object’ (e.g., ‘car’) represented by class labels and bounding boxes of participating objects in the image. Natural language region descriptions are represented by bounding boxes of the regions and description texts that correspond to the regions. We reconcile these two types of information using region graphs in the dataset. With the paired data of natural language descriptions and relations, we train the Scene Composer to generate composition layouts. We selected relations that contain subsets of the 100 most commonly used object classes and 70 predicates in the dataset. This dataset of selected object classes and predicates contains 101,968 instances. We split this dataset in the scheme of: 70% training set, 10% validation set, and 20% test set.

The Object Sketcher is trained with the Quick, Draw! [21] dataset that contains 70,000 training sketches, 2,500 validation sketches and 2,500 test sketches for each of the 345 object categories in the dataset. As mentioned in Chapter 4.1, we preprocess the data by computing the aspect ratios of all sketches as inputs to the Object Sketcher in addition to the original stroke data.

Using these data sources, we train multiple neural networks of various configurations and loss functions in Sketchforme. The LSTM architectures in the Scene Composer for generating composition layouts are stacked with 2 hidden-layers of size 512. Similarly, the Transformer Network has the configuration $(d_{model}, N_{layers}) = (512, 6)$.

The Scene Composer is trained by minimizing the negative log-likelihoods of the position data L_{xy} and size data L_{wh} , and cross-entropy loss for categorical outputs L_p :

$$L_{xy} = - \sum_{i=1}^n \log(P(x_i, y_i)) \quad (4.6)$$

$$L_{wh} = - \sum_{i=1}^n \log(P(w_i, h_i)) \quad (4.7)$$

$$L_p = - \left(\sum_{t=1}^n P(\text{box}_t) \log(\text{box}_t) + P(\text{start}_t) \log(P(\text{start}_t)) + P(\text{end}_t) \log(P(\text{end}_t)) \right) \quad (4.8)$$

For generating the class labels, each l_t is represented as a 100-dimensional vector in our model, with each value $l_{i,t}$ corresponding to the output probability of the class. L_{class} is thus computed as:

$$L_{\text{class}} = - \sum_{t=1}^n \sum_{i=1}^{100} l_{i,t} \log(l_{i,t}) \quad (4.9)$$

We combine these losses with weight hyper-parameters to obtain a general training objective L_{SC} for the Scene Composer:

$$L_{SC} = \lambda_1 L_{xy} + \lambda_2 L_{wh} + \lambda_3 L_p + \lambda_4 L_{\text{class}} \quad (4.10)$$

We set $\lambda_1 = 1.0, \lambda_2 = 1.0, \lambda_3 = 1 \times 10^{-5}, \lambda_4 = 1 \times 10^{-3}$. We used the Adam Optimizer with an initial learning rate of 1×10^{-5} and $\beta_1 = 0.9, \beta_2 = 0.999$ to minimize the loss function. We used 5 mixtures in each of the GMMs. We chose these hyper-parameters based on empirical experiments.

The Object Sketcher uses an HyperLSTM cell [12] of size 2048 for the modified Sketch-RNN model. The loss function of the Sketch-RNN model is identical to the reconstruction loss L_R in the original Sketch-RNN model to maximize the log-likelihood of the generated probability distribution for each of the strokes S_t . The model is trained with an initial learning rate of 0.0001 and gradient clipping of 1.0.

4.3 Experiments and Results

Central to evaluating Sketchforme’s success is assessing its effectiveness in generating realistic and relevant sketches and layouts from text descriptions. We evaluated the data generated by Sketchforme at each step of the generation process qualitatively and quantitatively to demonstrate its effectiveness of generating sketched scenes. We further conducted two user studies on the overall utility of the generated sketches to explore their potential in supporting real-world applications.

Composition Layout Generation

The composition layouts generated by the Scene Composer in the first step of Sketchforme’s sketch generation process are represented as bounding boxes of individual objects in the scene. While the Scene Composer already directly maximizes the log-likelihood of the data, we can evaluate the performance of the model by visualizing and comparing heat-maps created by super-positioning instances of real data and generated data.

Because Sketchforme considers the text input when generating the composition layouts, we should only compare the generated bounding boxes with ground-truth bounding boxes from the dataset that are relevant to the text input. We obtain these ground-truth compositions by filtering the subjects, objects, and predicates based on the descriptions. For instance, the composition layouts generated from ‘a person riding a horse.’ are compared to all ground-truth layouts with ‘person’ subjects, predicates that are related to riding such as ‘on’, ‘on top of’ etc. and ‘horse’ objects.

Heat-maps in Figure 4.4 shows the distributions of Sketchforme-generated bounding boxes and ground-truth bounding boxes from the dataset. From these heat-maps, we can obtain a holistic view of the generation performance of the model by visually evaluating the similarity between the heat-maps. We observe similar distributions between the ground-truth layouts and the generated layouts based on all of the descriptions.

We can further approximate an overlap metric between the distributions using Monte-Carlo simulations to evaluate the model’s performance quantitatively. To estimate the degree of overlap between the generated data distribution and the dataset’s distribution, we gener-

ated 100 composition layouts for each description and randomly sampled 1000 data points within each bounding box in these layouts. We estimate the overlap between the distributions by counting the number of data points that lie within the intersections between any generated and ground-truth bounding boxes. We compare Sketchforme’s performance with both a heuristic-based bounding box generator and a naive random bounding box generator. The heuristic-based bounding box generator only generates the first bounding boxes above/below the second bounding boxes for descriptions with the above-related/below-related predicates. The random bounding-box generator samples random values that describe the bounding boxes from uniform distributions, which serves as a naive baseline. Table 4.1 shows the percentage of the 1000 data points that lie in the intersections. The metric value for overlap between real and Sketchforme-generated data is considerably higher than both the value for overlap between real and heuristic-generated data and the value for overlap between real and randomly generated data, which confirms our analysis from qualitative visual inspection of the heat-maps.

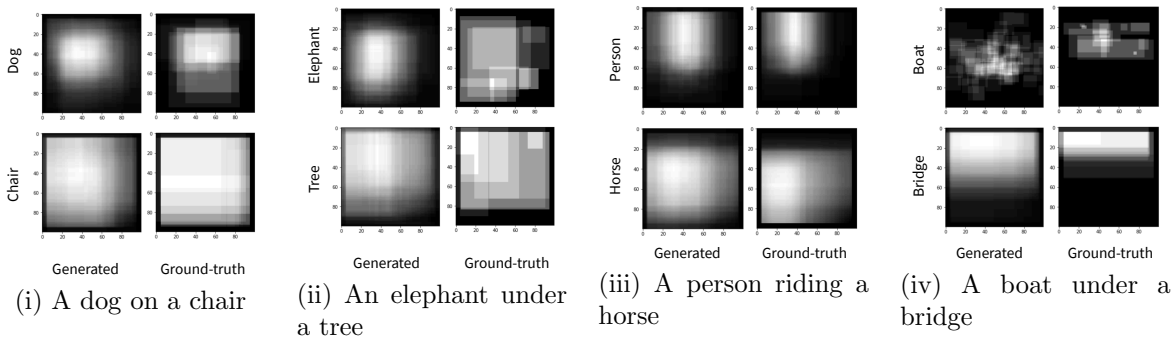


Figure 4.4: Heat-maps generated by super-positioning Sketchforme-generated/Visual Genome (ground-truth) data. Each horizontal pair of heat-maps corresponds to an object from a description.

Description	Sketchforme	Heuristics	Random
a dog on a chair	89.1%	64.4%	61.6%
an elephant under a tree	68.4%	40.3%	30.6%
a person riding a horse	94.0%	57.7%	51.5%
a boat under a bridge	31.8%	15.0%	6.85%

Table 4.1: Overlap metric from Monte-Carlo simulations for each description between real data and Sketchforme-generated/heuristics-generated/random data.

Generating Individual Object Sketches at Various Aspect Ratios

The main addition of Sketchforme to the original Sketch-RNN model is a new input that allows the Object Sketcher to generate sketches based on target aspect ratios ($r = \frac{\Delta y}{\Delta x}$) of completed sketches. We evaluate this approach by generating sketches of various aspect ratios. The Object Sketcher is able to adhere to input aspect ratios and generate individual object sketches coherent to the ratios. As shown in Figure 4.5, sketched trees generated with ratio $r = 1.0$ can be perceived as shorter than those generated with $r = 2.0$.

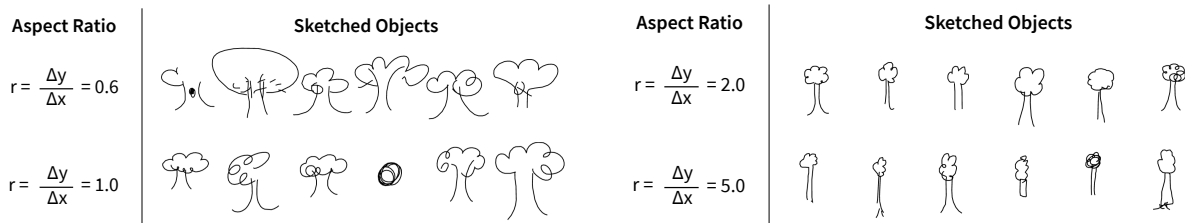


Figure 4.5: Generated sketches of trees with various aspect ratios by the Object Sketcher in Sketchforme.

Complete Scene Sketches

Combining the Scene Composer and the Object Sketcher, Sketchforme generates complete scene sketches directly from text descriptions. Several examples of the sketches are shown in Figure 4.1, Figure 4.6, and Figure 4.8. In these figures, sketches that correspond to ‘a boat under a bridge’ consist of small boats under bridges, whereas sketches that correspond to ‘an apple on a tree’ consist of small apples on large trees that follow the actual sizes and proportions of the objects. Moreover, Sketchforme is able to generalize to novel concepts of ‘a cat on top of a horse,’ such that the only relations involving a cat and a horse in the Visual Genome dataset which the model was trained on correspond to ‘a horse facing a cat.’ The sizes of cats and horses in these sketches are in proportion to their actual sizes, and the cat is adequately placed on the back of the horse, as shown in Figure 4.8.

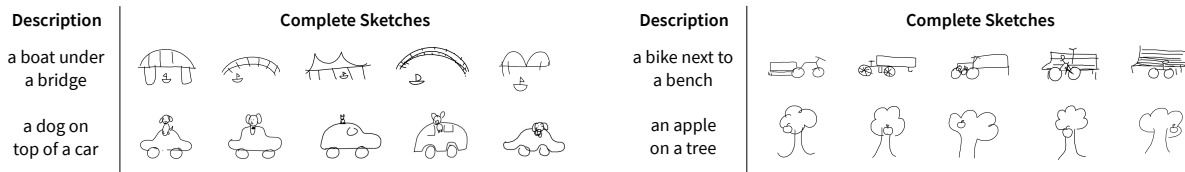


Figure 4.6: Complete scene sketches generated by Sketchforme.

We further trained Sketchforme on the Abstract Scenes dataset [45] to evaluate the approach’s generalizability on handling complex scenes. We included examples of sketches

generated by Sketchforme based on a) ‘Some squirrels near the pond under the trees on a sunny day’ (a multi-object multi-relation scene) and b) ‘Living room with some paintings on the wall’ (a scene consists of an abstract setting, i.e., living room) in Figure 4.7.

Sketchforme was able to generate these scenes adhering to the captions: for description a), Sketchforme was able to locate each object according to inter-object relations dictated by the description; for description b), Sketchforme was able to analyze the phrase ‘living room’ and generate couches and house plants in the scenes. When Sketchforme needs to handle a large number of objects, we found that it can be further improved by making a minor modification to the Scene Composer: feeding the generated classes into the Transformer network to synchronize the generated bounding boxes with the generated class labels of the objects. Moreover, we observed that the limitation of occlusions (further discussed in Chapter 4.5) is more apparent as the scenes become more crowded with objects, such as the multiple overlapping trees in scenes based on description a).

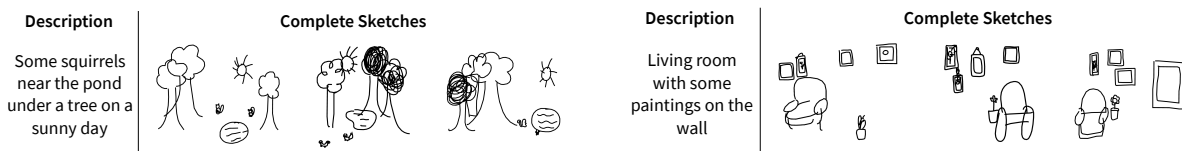


Figure 4.7: Complete scene sketches generated by Sketchforme trained on the Abstract Scenes dataset that contains complex multi-object scenes.

Human Perception User-Study

Sketchforme’s high-level goal is to augment users’ communication and learning processes by generating realistic, plausible, and coherent sketches for users to interact with. To complement the quantitative and qualitative evaluation of the sketches, we conducted a user study on Amazon Mechanical Turk (AMT) to gauge human subjects’ opinions on the sketches’ realism and ability to convey the descriptions used to generate them.

Study Procedure

We recruited 51 human subjects on AMT and asked them to each review 50 sketches generated by either humans or Sketchforme. These 50 sketches are generated from five descriptions. The human-generated sketches are obtained from another AMT task prior to this user study based on Quick, Draw! [21]. These human-generated sketches are shown in Figure 4.8. In this study, subjects are provided with complete sketched scenes and descriptions that the scenes are based on. Subjects are required to respond to the following questions:

1. Do you think this sketch was generated by a computer (AI) or a human?

- On a scale of 1-5 (1 represents that description conveyed very poorly, 5 represents that description conveyed very well), how well did you think the message is conveyed by the sketch?

The subjects are given 10 sketches as trial questions with answers to the first question at the beginning of the task. After completing the trial tasks, the subjects’ answers to the remaining 40 sketches are aggregated as the study results. This study protocol is similar to perception studies commonly used to evaluate synthetic visual content generation techniques in the deep learning community [20]. In addition, we collected comments from the users (if any) and their perceived overall difficulty of the task at the end of the task.

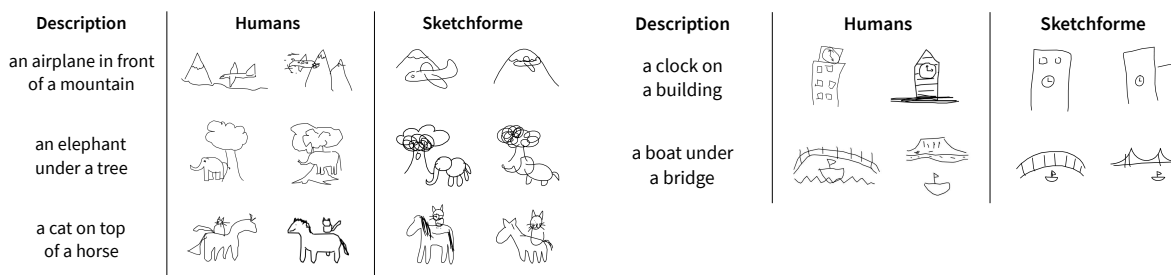


Figure 4.8: Samples of sketches produced by humans and Sketchforme used in the AMT user study.

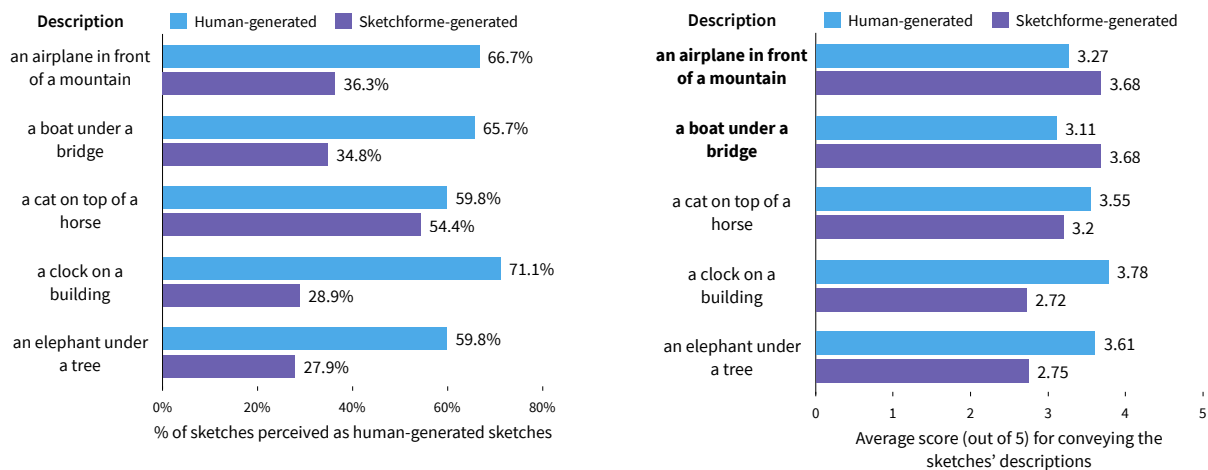
Results

The first question probes the realism of the sketches with a Turing-test-style question asking the subjects to determine whether the sketches are created by humans. As shown in Figure 4.9i, subjects on average considered 64.6% of the human-generated sketches as generated by humans, while they considered 36.5% of Sketchforme-generated sketches as generated by humans. Although the percentage of Sketchforme-generated sketches considered as generated by humans is significantly lower ($p < 1.05 \times 10^{-10}$, paired t-test) than that of human-generated sketches, individual participants commented in the study that it was difficult to distinguish between human-generated and Sketchforme-generated sketches. P2 mentioned that they “really couldn’t tell the difference in most images.” P6 commented that they “didn’t know if it was (a) human or a computer (that generated the sketches).” These results demonstrate the potential for Sketchforme in generating realistic sketched scenes.

We hypothesize one of the possible reasons for the lower percentage of Sketchforme-generated sketches to be considered as human-drawn is that the curves of the synthetic sketches are in general less jittery than human-drawn sketches. We suggest future work explore introducing stroke variation to generate more realistic sketches.

The results for the second question reflects the ability of the sketches to communicate the underlying descriptions that they are based on. The average score for human-generated

sketches is $\mu = 3.46$, whereas the average score for Sketchforme-generated sketches is $\mu = 3.21$ as shown in Figure 4.9ii. Although Sketchforme-generated sketches achieved lower scores overall, Sketchforme-generated sketches achieved statistically better average scores for sketches based on two of the descriptions: ‘a boat under a bridge’ and ‘an airplane in front of a mountain’ ($p < 0.0005$, paired t-test). This shows the competitive performance of Sketchforme-generated sketches in communicating the underlying descriptions for some scenes.



(i) Percentage of sketches considered by users as human-generated

(ii) Average score for conveying the sketches' descriptions

Figure 4.9: Results of the human perception user-study on Sketchforme. 64.6% of human-generated sketches and 36.5% of Sketchforme-generated sketches are perceived as human-generated in (i). In (ii), Sketchforme-generated sketches was considered more expressive than human-generated sketches for sketches of ‘a boat under a bridge.’ and ‘an airplane in front of a mountain.’

Sketch Interpretation User-Study

To further evaluate Sketchforme’s ability to deliver messages through sketches, we conducted an exploratory user-study to gauge users’ ability to translate Sketchforme-generated sketches back into natural language descriptions.

Study Procedure

We recruited 10 participants on AMT. Each participant was provided with 5 sketches generated by humans and 5 sketches generated by Sketchforme from the same descriptions, but

without the original text descriptions. The sketches are reused from the previous study, but the order and selection of the sketches from the set of sketches in the previous study are randomized for each participant. Participants are then asked to produce text descriptions that represent the meaning of each of the sketches. We ensured that the participants had not participated in any of our other studies.

Results

We aggregated the text descriptions produced by each of the participants and compared the subjects, objects, and predicates included in their text descriptions with the original descriptions used to generate the sketches. The percentage of user-generated descriptions that match the subjects/objects/predicates of the original descriptions are respectively 86%/72%/46% for human-generated sketches, and 86%/76%/38% for Sketchforme-generated sketches. While we did not observe a significant difference between the communication ability of both sets of sketches ($p > 0.269$, each paired t-test for subjects/objects/predicates), one novel insight brought by this study is that these sketches are particularly weak in conveying the predicates. Multiple descriptions provided by the users on both sets of sketches did not mention the predicates at all, such as 'an elephant and a tree,' or mentioned more general predicates, such as 'elephant walking near a tree,' for sketches based on 'an elephant under the tree.' Effectively conveying the descriptions' predicates through sketches can be an interesting issue for further research investigation.

4.4 Applications

In this chapter, we explore several applications that can benefit from Sketchforme's ability to generate compelling sketches from natural language descriptions.

Sketch-Assisted Language Learning

Sketches have been shown to improve memory [42]. As language learning is a memory-intensive task, Sketchforme could support language education applications based on sketches. These sketches can potentially create engaging and effective learning processes and avoid rote learning.

Language Learning Application

To explore the possibility of Sketchforme in supporting language learning, we built a basic language-learning application that aims to educate learners with a translation task from German to English. In this application, learners are presented with German phrases and asked to translate them to English in the form of multiple-choice questions similar to the process of learning term definitions from flash-cards. This application also implements the Leitner system [28] with three bins that repeats phrases which learners make the most

mistakes on most frequently. Under this system, the phrases are moved to different bins depending on the participants' familiarity with the translations.

We gathered 10 pairs of German-English sentences from a native German speaker and form 2 sets of 5 translations each. In addition, deceptive English sentences are added as other choices in the multiple-choice test to be selected by the learners in the application. We deployed this application on AMT to test the improvement of learning performance by presenting Sketchforme-generated sketches along with the phrases. The UI of the full application with a sketch presented to the users is shown in Figure 4.10i.

Study Procedure

The study consists of a training phase and a test phase for each participant. In the training phase, participants are presented with correct answers after answering each question. The participant can only advance to the next phase when they answer all questions correctly consecutively for all translations according to the Leitner system. In the test phase, participants are given one chance to provide their answer to all translations without seeing the correct answers. The participants are divided into two conditions, with the 'control' group only receiving phrases on their interface during training, and the 'treatment' group that receives both phrases and sketches generated by Sketchforme on their interface during the training phase. Both groups receive only the phrases on their interface during the test phase. Moreover, we use our two sets of translations for training and test phases alternatively, such that the participants will not get consecutive training and test phases for the same set of descriptions.

The performance of the participants during the study are monitored with multiple analytical metrics, including completion time of each phase, and scores in the test phase, etc. At the end of the study, we also provide surveys for them to rate the difficulty of the task and the usefulness of the sketches (if applicable) on five-point Likert scales, and ask them to provide any additional suggestions to the interface.

Results

We recruited 38 participants on AMT to participate in the study. While we did not see a significant difference ($p = 0.132$, unpaired t-test) in the correctness of answers in the test phase of the phrases between the 'control' and 'treatment' groups of participants, we discovered that the *time taken to complete the learning task* for the 'treatment' group (**246** seconds on average) was significantly less ($p = 0.011$, unpaired t-test) than the control group (**338** seconds on average). The 'treatment' group also generally found the sketches to be helpful for learning (rated 4.58 out of 5 in the post-study survey).

As Sketchforme is an automated system that is capable of generating sketches from free-form text descriptions, and with these promising results on sketch-assisted language learning, we envision Sketchforme to support and improve large-scale language learning applications in the future.

Intelligent Sketching Assistant

Since Sketchforme uses sequence models and a multi-step generation process to generate sketches, by design it can support interactive human-in-the-loop sketching systems. To demonstrate such capability, we built a prototype of an intelligent sketching assistant reflective of two potential use-cases:

Auto-completion of scenes

As Sketchforme’s Scene Composer consists of the Transformer Network, a sequence model that attends to previous objects in the scene to generate the upcoming object, we can complete unfinished user scenes instead of starting with a blank canvas by starting the generation with both the start token b_1 and an existing object in the scene created by the user b_2 . Figure 4.10ii shows examples of Sketchforme completing users’ sketch of a horse in step a) by adding potential sketched trees involved in the scene.

User-Steerable generation

Sketchforme’s Scene Composer is capable of generating multiple potential candidates sketched objects at each step of composing the final sketched scenes. As such, users can select their preferred scene layout from the candidates. Figure 4.10ii shows multiple candidates proposed by Sketchforme based on a text description in step b). Moreover, since Sketch-RNN is also capable of generating a variety of sketches, the users can select their preferred sketches of each individual object in the scene.

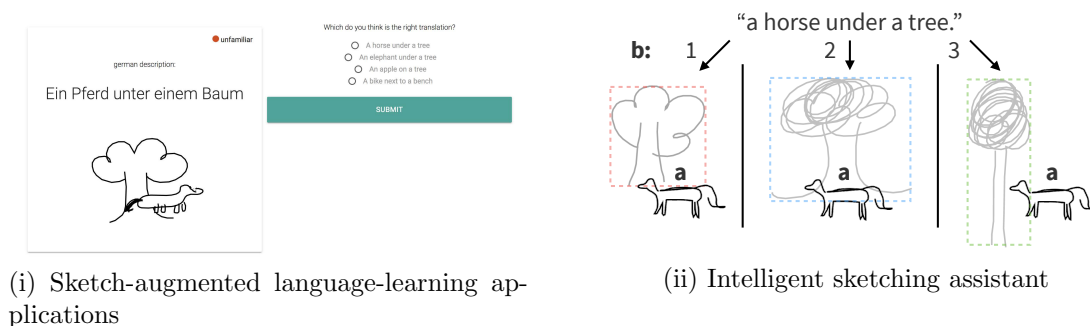


Figure 4.10: Applications enabled by Sketchforme. Sketchforme can augment (i) language-learning applications and significantly reduced the time taken for users to achieve similar learning outcomes. With the (ii) intelligent sketching assistant powered by Sketchforme, the user can create a partial sketch for Sketchforme to suggest multiple candidates for them to choose the adequate sketch they prefer from the description ‘a horse under a tree.’

4.5 Limitations

Occlusions and Layer Order

Sketchforme is trained to model scene compositions from a natural image dataset. In natural images, objects might occlude each other, hence affecting sizes and positions of the bounding boxes in the composition layouts. Figure 4.11i shows several boats that were inadequately placed in front of parts of the bridges that should have occluded the boats. To overcome this limitation, future systems can augment Sketchforme by including advanced vision models to determine the objects' layer order in the original natural image. The current Sketchforme system only considers a naive layer order determined by the generation sequence of the composition layout, which is 'subject' then 'object' from the dataset.

Moreover, novel methods should be developed to handle overlapping sketched objects generated from occluded compositions. For instance, the model that generates composition layouts could enforce constraints to avoid overlaps in the sketches or follow user-specified heuristics to handle overlaps.

Aspect Ratios might be Weak Signals for Object Poses

Sketchforme uses aspect ratios of bounding boxes as the primary signal to inform the shapes of sketches of individual objects. Although these shapes can be sufficient to determine the correct poses for objects in some classes, such as the 'tree' class, merely constraining the shapes might be weak signals for objects of other classes. These shapes can suggest incoherent perspectives or incomplete sketches, such as examples shown in Figure 4.11ii. In Figure 4.11ii, only faces of the elephants were sketched due to aspect ratios provided to the Object Sketcher, which is inappropriate for composing sketched scenes. To mitigate this limitation, future work could model the poses of objects in sketches and natural images more closely using other cues, such as complete masks of the objects.

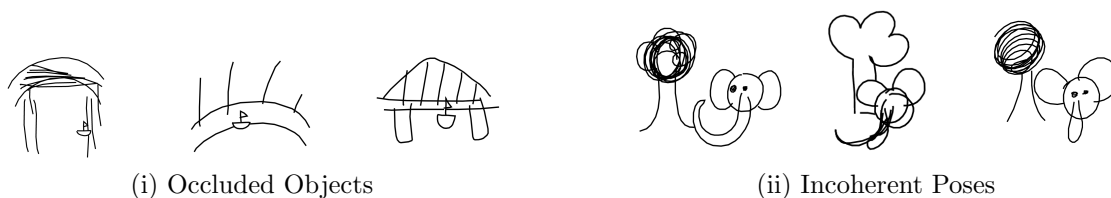


Figure 4.11: Limitations of Sketchforme's sketch generation process. In (i), the boats are significantly occluded by the bridges. In (ii), the elephants were represented with square bounding boxes which guided the system to sketch only the faces of the elephants.

Chapter 5

Conclusion and Future Work

In this report, we introduced two novel systems that recognize and generate sketches for interactive applications, and developed the associated deep-learning models that support these applications at the core of these systems. Swire enables designers to interact with large-scale UI datasets with sketches by adopting a sketch-based image retrieval technique. Sketchforme is capable of generating abstract scene sketches that involve multiple objects based on natural language descriptions.

We hope the findings from the investigation of these systems can inspire and accelerate future research to explore further, more sophisticated sketch-based human-computer interactions. In the remainder of this chapter, we discuss a few directions that we find promising extending this body of research.

5.1 Design and Engineering Applications in Other Domains

We showed computational systems can support sketching as an effective medium for retrieving relevant UI design inspiration with Swire. While the UI/UX design process embodies the ambiguous and creative properties of design thinking in general, design sketches in other domains, such as industrial design and mechanical engineering, might contain additional complexity of multiple perspectives, and/or larger numbers of basic components. Such complexity creates new challenges and opportunities for innovation in sketch-based design and engineering applications.

5.2 Conversational Sketch Suggestion and Tutorial System

To fully reap the benefits of the highly interactive media of sketching and natural language, and the high-level and low-level information of each element of the sketched scenes modeled

by Sketchforme, we believe future work should expand beyond generating a single sketched scene from a single text description, and explore conversational interfaces that guide users to progressively create sketches coherent to dialogues. Beyond generating sketches based on dialogues of sketching instructions, future work can also explore automatic storyboarding applications that generate sketch-illustrated stories with the support of relevant narration and description data.

Moreover, since both components involved in Sketchforme’s sketch generation process are capable of completing partial sketches created by users, Sketchforme can suggest possible strokes following incomplete user-drawn sketches for pedagogical purposes.

5.3 Coloring and Animation

The sketches generated by Sketchforme are binary sketch strokes without colors or animations. Future work should explore colored and/or animated sketches to enable richer user experiences. For instance, the natural image dataset that was used to train Sketchforme’s Scene Composer can be used to determine possible colors of the sketched objects. Video segmentation datasets can also be used to inform the motion of sketched objects in animation sequences.

Bibliography

- [1] Rahul Arora et al. “SketchSoup: Exploratory Ideation Using Design Sketches”. In: *Computer Graphics Forum* (2017). URL: <http://www-sop.inria.fr/revs/Basilic/2017/ADNBS17>.
- [2] Christopher M Bishop. *Mixture density networks*. Tech. rep. Citeseer, 1994.
- [3] Nathalie Bonnardel. “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. Loughborough, United Kingdom: ACM, 1999, pp. 158–165. ISBN: 1-58113-078-3. DOI: 10.1145/317561.317589. URL: <http://doi.acm.org/10.1145/317561.317589>.
- [4] Bill Buxton. *Sketching User Experiences: Getting the Design Right and the Right Design*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007. ISBN: 9780123740373.
- [5] Alex J. Champanand. “Semantic Style Transfer and Turning Two-Bit Doodles into Fine Artworks”. In: *CoRR* abs/1603.01768 (2016). arXiv: 1603.01768. URL: <http://arxiv.org/abs/1603.01768>.
- [6] Jamie Combs and Brenda Hoddinott. *Drawing for dummies*. Wiley, 2011.
- [7] Biplab Deka, Zifeng Huang, and Ranjitha Kumar. “ERICA: Interaction Mining Mobile Apps”. In: *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. UIST ’16. Tokyo, Japan: ACM, 2016, pp. 767–776. ISBN: 978-1-4503-4189-9. DOI: 10.1145/2984511.2984581. URL: <http://doi.acm.org/10.1145/2984511.2984581>.
- [8] Biplab Deka et al. “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. Quebec City, QC, Canada: ACM, 2017, pp. 845–854. ISBN: 978-1-4503-4981-9. DOI: 10.1145/3126594.3126651. URL: <http://doi.acm.org/10.1145/3126594.3126651>.
- [9] Steven P. Dow et al. “Parallel Prototyping Leads to Better Design Results, More Divergence, and Increased Self-efficacy”. In: *ACM Trans. Comput.-Hum. Interact.* 17.4 (Dec. 2010), 18:1–18:24. ISSN: 1073-0516. DOI: 10.1145/1879831.1879836. URL: <http://doi.acm.org/10.1145/1879831.1879836>.

- [10] Mathias Eitz, James Hays, and Marc Alexa. “How Do Humans Sketch Objects?” In: *ACM Trans. Graph. (Proc. SIGGRAPH)* 31.4 (2012), 44:1–44:10.
- [11] Jennifer Fernquist, Tovi Grossman, and George Fitzmaurice. “Sketch-sketch Revolution: An Engaging Tutorial System for Guided Sketching and Application Learning”. In: *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*. UIST ’11. Santa Barbara, California, USA: ACM, 2011, pp. 373–382. ISBN: 978-1-4503-0716-1. DOI: 10.1145/2047196.2047245. URL: <http://doi.acm.org/10.1145/2047196.2047245>.
- [12] David Ha, Andrew M. Dai, and Quoc V. Le. “HyperNetworks”. In: *CoRR* abs/1609.09106 (2016). arXiv: 1609.09106. URL: <http://arxiv.org/abs/1609.09106>.
- [13] David Ha and Douglas Eck. “A Neural Representation of Sketch Drawings”. In: *ICLR 2018*. 2018. 2018. URL: <https://openreview.net/pdf?id=Hy6GHpkCW>.
- [14] James W. Hennessey et al. “How2Sketch: Generating Easy-To-Follow Tutorials for Sketching 3D Objects”. In: *Symposium on Interactive 3D Graphics and Games* (2017).
- [15] Scarlett R. Herring et al. “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. Boston, MA, USA: ACM, 2009, pp. 87–96. ISBN: 978-1-60558-246-7. DOI: 10.1145/1518701.1518717. URL: <http://doi.acm.org/10.1145/1518701.1518717>.
- [16] Seunghoon Hong et al. “Inferring semantic layout for hierarchical text-to-image synthesis”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 7986–7994.
- [17] Forrest Huang and John F. Canny. “Sketchforme: Composing Sketched Scenes from Text Descriptions for Interactive Applications”. In: *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. UIST ’19. New Orleans, LA, USA: Association for Computing Machinery, 2019, pp. 209–220. ISBN: 9781450368162. DOI: 10.1145/3332165.3347878. URL: <https://doi.org/10.1145/3332165.3347878>.
- [18] Forrest Huang, John F. Canny, and Jeffrey Nichols. “Swire: Sketch-Based User Interface Retrieval”. In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. CHI ’19. Glasgow, Scotland Uk: Association for Computing Machinery, 2019. ISBN: 9781450359702. DOI: 10.1145/3290605.3300334. URL: <https://doi.org/10.1145/3290605.3300334>.
- [19] Emmanuel Iarussi, Adrien Bousseau, and Theophanis Tsandilas. “The Drawing Assistant: Automated Drawing Guidance and Feedback from Photographs”. In: *ACM Symposium on User Interface Software and Technology (UIST)*. St Andrews, United Kingdom: ACM, Oct. 2013. URL: <https://hal.inria.fr/hal-00840853>.

- [20] Phillip Isola et al. “Image-to-Image Translation with Conditional Adversarial Networks”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017).
- [21] Jonas Jongejan et al. *The Quick, Draw! - A.I. Experiment*. 2016. URL: <https://quickdraw.withgoogle.com/> (visited on 12/06/2017).
- [22] Rubaiat Habib Kazi et al. “DreamSketch: Early Stage 3D Design Explorations with Sketching and Generative Design”. In: *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*. UIST ’17. Québec City, QC, Canada: ACM, 2017, pp. 401–414. ISBN: 978-1-4503-4981-9. DOI: 10.1145/3126594.3126662. URL: <http://doi.acm.org/10.1145/3126594.3126662>.
- [23] Ranjay Krishna et al. “Visual Genome: Connecting Language and Vision Using Crowdsourced Dense Image Annotations”. In: *Int. J. Comput. Vision* 123.1 (May 2017), pp. 32–73. ISSN: 0920-5691. DOI: 10.1007/s11263-016-0981-7. URL: <https://doi.org/10.1007/s11263-016-0981-7>.
- [24] Ranjitha Kumar et al. “Bricolage: Example-based Retargeting for Web Design”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’11. Vancouver, BC, Canada: ACM, 2011, pp. 2197–2206. ISBN: 978-1-4503-0228-9. DOI: 10.1145/1978942.1979262. URL: <http://doi.acm.org/10.1145/1978942.1979262>.
- [25] James A. Landay. “SILK: Sketching Interfaces Like Crazy”. In: *Conference Companion on Human Factors in Computing Systems*. CHI ’96. Vancouver, British Columbia, Canada: ACM, 1996, pp. 398–399. ISBN: 0-89791-832-0. DOI: 10.1145/257089.257396. URL: <http://doi.acm.org/10.1145/257089.257396>.
- [26] Walter S. Lasecki et al. “Apparition: Crowdsourced User Interfaces That Come to Life As You Sketch Them”. In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. CHI ’15. Seoul, Republic of Korea: ACM, 2015, pp. 1925–1934. ISBN: 978-1-4503-3145-6. DOI: 10.1145/2702123.2702565. URL: <http://doi.acm.org/10.1145/2702123.2702565>.
- [27] Yong Jae Lee, C. Lawrence Zitnick, and Michael F. Cohen. “ShadowDraw: Real-time User Guidance for Freehand Drawing”. In: *ACM Trans. Graph.* 30.4 (July 2011), 27:1–27:10. ISSN: 0730-0301. DOI: 10.1145/2010324.1964922. URL: <http://doi.acm.org/10.1145/2010324.1964922>.
- [28] S. Leitner and R. Totter. *So lernt man lernen*. Angewandte Lernpsychologie ein Weg zum Erfolg. Herder, 1972. ISBN: 9783451162657. URL: <https://books.google.com/books?id=sD3AAQAACAAJ>.
- [29] Alex Limpaecher et al. “Real-time Drawing Assistance Through Crowdsourcing”. In: *ACM Trans. Graph.* 32.4 (July 2013), 54:1–54:8. ISSN: 0730-0301. DOI: 10.1145/2461912.2462016. URL: <http://doi.acm.org/10.1145/2461912.2462016>.

- [30] James Lin et al. “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. The Hague, The Netherlands: ACM, 2000, pp. 510–517. ISBN: 1-58113-216-6. DOI: 10.1145/332040.332486. URL: <http://doi.acm.org/10.1145/332040.332486>.
- [31] Rafael Munoz-Salinas. “ARUCO: a minimal library for Augmented Reality applications based on OpenCv”. In: *Universidad de Córdoba* (2012).
- [32] Mark W. Newman and James A. Landay. “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. New York City, New York, USA: ACM, 2000, pp. 263–274. ISBN: 1-58113-219-0. DOI: 10.1145/347642.347758. URL: <http://doi.acm.org/10.1145/347642.347758>.
- [33] T. A. Nguyen and C. Csallner. “Reverse Engineering Mobile Application User Interfaces with REMAUI (T)”. In: *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. Nov. 2015, pp. 248–259. DOI: 10.1109/ASE.2015.32.
- [34] Taesung Park et al. “Semantic Image Synthesis with Spatially-Adaptive Normalization”. In: *CoRR* abs/1903.07291 (2019). arXiv: 1903.07291. URL: <http://arxiv.org/abs/1903.07291>.
- [35] Olga Russakovsky et al. “ImageNet Large Scale Visual Recognition Challenge”. In: *CoRR* abs/1409.0575 (2014). arXiv: 1409.0575. URL: <http://arxiv.org/abs/1409.0575>.
- [36] Patsorn Sangkloy et al. “The Sketchy Database: Learning to Retrieve Badly Drawn Bunnies”. In: *ACM Trans. Graph.* 35.4 (July 2016), 119:1–119:12. ISSN: 0730-0301. DOI: 10.1145/2897824.2925954. URL: <http://doi.acm.org/10.1145/2897824.2925954>.
- [37] Florian Schroff, Dmitry Kalenichenko, and James Philbin. “Facenet: A unified embedding for face recognition and clustering”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 815–823.
- [38] K. Simonyan and A. Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: *International Conference on Learning Representations*. 2015.
- [39] Qingkun Su et al. “EZ-sketching: Three-level Optimization for Error-tolerant Image Tracing”. In: *ACM Trans. Graph.* 33.4 (July 2014), 54:1–54:9. ISSN: 0730-0301. DOI: 10.1145/2601097.2601202. URL: <http://doi.acm.org/10.1145/2601097.2601202>.
- [40] Amanda Swearngin et al. “Rewire: Interface Design Assistance from Examples”. In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. CHI '18. Montreal QC, Canada: ACM, 2018, 504:1–504:12. ISBN: 978-1-4503-5620-6. DOI: 10.1145/3173574.3174078. URL: <http://doi.acm.org/10.1145/3173574.3174078>.

- [41] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 5998–6008.
- [42] Jeffrey D. Wammes, Melissa E. Meade, and Myra A. Fernandes. “The drawing effect: Evidence for reliable and robust memory benefits in free recall”. In: *The Quarterly Journal of Experimental Psychology* 69.9 (2016). PMID: 26444654, pp. 1752–1776. DOI: 10.1080/17470218.2015.1094494. eprint: <https://doi.org/10.1080/17470218.2015.1094494>. URL: <https://doi.org/10.1080/17470218.2015.1094494>.
- [43] Jun Xie et al. “PortraitSketch: Face Sketching Assistance for Novices”. In: *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*. UIST ’14. Honolulu, Hawaii, USA: ACM, 2014, pp. 407–417. ISBN: 978-1-4503-3069-5. DOI: 10.1145/2642918.2647399. URL: <http://doi.acm.org/10.1145/2642918.2647399>.
- [44] Qian Yu et al. “Sketch Me That Shoe”. In: *Computer Vision and Pattern Recognition*. 2016.
- [45] C Lawrence Zitnick and Devi Parikh. “Bringing semantics into focus using visual abstraction”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2013, pp. 3009–3016.
- [46] Changqing Zou et al. “Sketchyscene: Richly-annotated scene sketches”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 421–436.