RESEARCH-ARTICLE

ARSketch: Sketch-Based User Interface for Augmented Reality Glasses

ZHAOHUI ZHANG, Rokid, Inc., Building 22, Creative Business Center, No. XI Xixi, No. 808, Wen'er West Road, Xihu District, https://www.rokid.com/en, Hangzhou

HAICHAO ZHU, Chinese University of Hong Kong, Hong Kong, Hong Kong

QIAN ZHANG, University of California, Los Angeles, Los Angeles, CA, United States

Published: 12 October 2020

Citation in BibTeX format

MM '20: The 28th ACM International Conference on Multimedia
October 12 - 16, 2020
WA, Seattle, USA

Conference Sponsors:
SIGMM

ARSketch: Sketch-Based User Interface for Augmented Reality Glasses

Zhaohui Zhang*
Rokid Corporation Ltd.
zhaohui.zhang@rokid.com

Haichao Zhu*
The Chinese University of Hong Kong
hczhu@cse.cuhk.edu.hk

Qian Zhang
University of California, Los Angeles
zhangqian@cs.ucla.edu

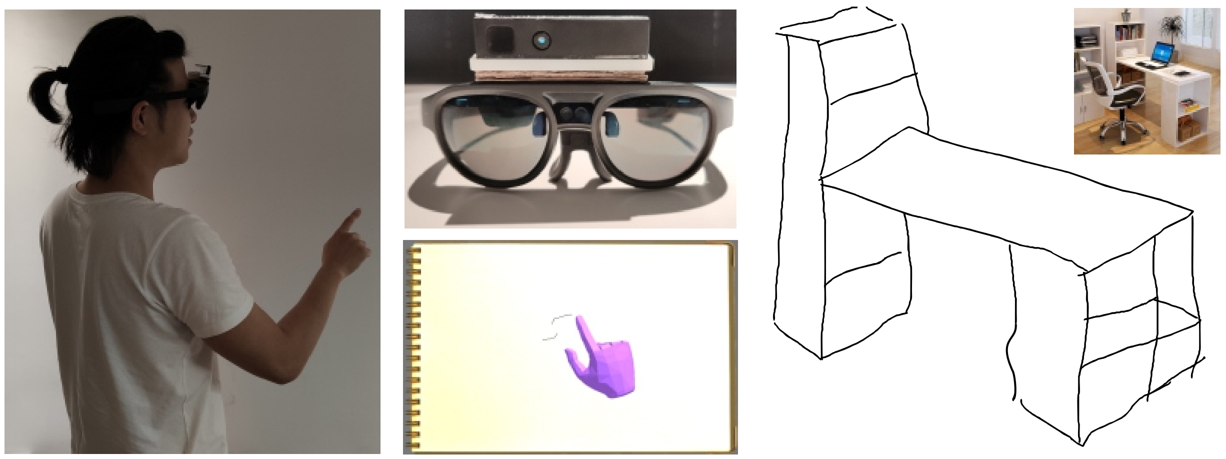


Figure 1: Overview of ARSketch: (Left) A user is wearing our prototype to create a sketch. (Middle Top) This prototype uses an optical-see through head-mounted AR glass (Rokid Glass) extended by a depth camera (pmd flexx). (Middle Bottom) The sketch interface shown on the glass display. The user sees a virtual sketch pad and a virtual hand mesh in front. (Right) A table sketch created by the user.

ABSTRACT

Hand gesture interaction is a key component in Augmented Reality (AR) / Mixed Reality (MR). Users usually interact with AR/MR devices, e.g., Microsoft HoloLens, etc., via hand gestures to express their intentions and the devices will recognize the gestures and respond accordingly to users. However, the use of such technique so far is limited to only a few less-expressive hand gestures, which, unfortunately, are insufficient or inadequate to input complex information.

To tackle this problem, we introduce a sketch-based neural network-driven user interface for AR/MR glasses, called ARSketch, which enables drawing sketches freely in air to interact with the devices. ARSketch combines: (1) hand pose estimation that estimates the egocentric hand poses in an energy-efficient way, (2) sketch generation that generates sketches using key point positions of hand poses, and (3) sketch-photo retrieval that takes sketches as inputs to retrieve relevant photos. The evaluation results on our collected sketch dataset demonstrate the efficacy of ARSketch for user interaction.

*Both authors contributed equally to this research.

MM '20, October 12–16, 2020, Seattle, WA, USA.
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-7988-5/20/10…$15.00
https://doi.org/10.1145/3394171.3413633

CCS CONCEPTS

• **Human-centered computing → Mixed / augmented reality**.

KEYWORDS

user interface; augmented reality; hand pose estimation/tracking; sketch; computer vision; deep learning; image retrieval

ACM Reference Format:
Zhaohui Zhang, Haichao Zhu, and Qian Zhang. 2020. ARSketch: Sketch-Based User Interface for Augmented Reality Glasses. In *Proceedings of the 28th ACM International Conference on Multimedia (MM '20), October 12–16, 2020, Seattle, WA, USA.*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3394171.3413633

1 INTRODUCTION

Hand gesture recognition is a common task, as well as a core challenge, for head-mounted Augmented Reality (AR) / Mixed Reality (MR) devices, e.g., Microsoft HoloLens, Magical Leap One. It is one of the most important and frequently used user interaction technique. Using gesture-based interaction, a user can tell the device hist/her intentions via hand gestures without touching keyboards or screens, and the device can get visual input by recognizing these gestures. For example, a user can use two fingers to indicate his/her intention for a view magnification with desired zoom by moving his/her thumb and index fingers apart.

Although hand gesture interaction provides substantial benefits and is naturally available to a broad user base, current gestures that can be recognized are associated with limited quantity and expressiveness. First, only a few hand gestures are recognizable for most of the existing AR devices. Basically, hand gesture recognition is a classification problem and the gestures to be recognized should be predefined. If a user interact with AR/MR devices via any gestures that are not from the predefined ones, the device will make no response because it cannot capture his/her intention. For example, the state-of-the-art methods [19, 34, 38] support 25, 83, and 249 classes of gestures respectively. Second, the existing recognizable gestures are not expressive enough for complex information. For example, a user is moving to a new apartment and he/she wants to preview the furniture arrangement with his/her AR device. He/She stands in the living room and wants to tell the device about the dining table and chairs with his/her favorite style. Unfortunately, all existing gestures fails to convey his/her intention to the device and this makes him/her upset.

Our observation is that a user can interact with a computer by creating a *sketch* to express his/her intention. Sketch, as a method of human computer interaction, has been introduced in SketchPad [29] and successfully applied in a broad range of applications, such as 3D modeling [26], image retrieval [14] and shape retrieval [8]. Inspired by this sketch-based interaction between human and computer, we propose to use sketch for complex input on head-mounted AR devices. However, sketching on AR devices is non-trivial and out of reach for methods of sketching on papers. Normally, when a user is drawing on a paper/pad, a force provides a feedback whether the user is drawing or not. On the contrary, when a user is drawing in air, there is no explicit signal to tell the device whether the current hand movement is a part of the sketch strokes or not. In summary, the challenges are: (1) extracted sketches from tracking hand poses in air are full of noises and jitters, making it difficult to extract resulting strokes, and (2) users are reluctant to draw details of objects, such like textures, due to the limited degrees of Field of View (FoV) and the resolutions.

In this paper, we propose a sketch-based neural network-driven user interface on AR glasses, called ARSketch. With ARSketch, a user can interact with his/her AR glasses by creating a simplistic sketch using binary strokes which depict his/her desired input to the system. This sketch is generated by the captured hand movement from the camera of AR device. Then the AR device can return information based on the sketch input as shown in Figure 1. ARSketch is comprised of three components: glass side *hand pose estimation*, *sketch generation*, and server side retrieval-based *sketch auto-completion*. First, ARSketch utilizes an energy-efficient neural network-based method to estimate the egocentric hand poses. Second, using this knowledge of hand poses, it generates sketches based on key point positions and utilizes L_0 smoothing to reduce the noises and jitters. Third, it takes sketches from the glass side as input to another neural network followed by a nearest neighbour search to automatically complete what the user is drawing on server. Although ARSketch is built for one AR glass, our idea is applicable to other AR/MR devices. With the help of the sketch-based interaction, users now can draw a table to let the AR device display the arrangement.

In summary, this work makes the following contributions:

- We design a novel sketch-based user interface for AR glasses.
- The proposed hand pose estimation method for egocentric images achieves state-of-the-art performance in terms of both accuracy and computational efficiency.
- The proposed efficient key points-based sketch generation is the first work, to our knowledge, on noise-tolerant sketch generation.
- We significantly improve the user experience by automatically completing input sketches.

2 RELATED WORK

2.1 Sketch Recognition and Classification

Since the pioneering work SketchPad [29] in the early 60s, a broad range of techniques have been proposed to improve this sketch-based human computer interaction technique, such as sketch recognition, retrieval and classification in the field of computer vision or computer graphics.

Eitz et al. [7] collect a human sketch dataset, namely TU-Berlin sketch benchmark, to investigate how humans sketch objects. They also propose a computational recognition method by using local features, bag-of-feature representation and support vector machines to classify this dataset. Although they demonstrate an interactive sketch recognition system, the performance gap between their methods and humans is not negligible. Scheneider et al. [27] extend Eitz's work by using Fisher vectors to encode local features for sketch recognition task. They achieve a classification rate that is close to human performance on TU-Berlin sketch benchmark.

Most recent approaches investigate deep learning techniques for sketch classification. For example, SketchNet [37] is proposed to extract sketch feature representation for sketch classification task with web images. This network is built on top of AlexNet [16] and it takes a triplet composition of sketch, positive as well as negative real images as the input. Their results show that the neural network representation outperforms traditional features. In [4], John Collomosse et al. extend the triplet input to a triplet loss and propose a neural network based on [30]. This network can capture structural and style similarity over photos and sketches for sketch-based image retrieval (SBIR). Hashing is introduced in [17] to speed up SBIR by end-to-end learning a novel binary coding. This method achieves state-of-the-art performance in terms of retrieval accuracy and time/storage complexity. [23] presents a cross-category generalization method for SBIR to tackle the cross-category retrieval problem.

Sketches drawing in air are distinct from sketches drawing on a pad or paper, existing methods fail to achieve high performance for the noisy AR sketches. Differs from the above methods that are designed specifically for traditional sketches, ARSketch generates sketches based on key point positions instead of recording hand trajectory and utilizes L_0 smoothing to reduce noises and jitters.

2.2 3D Hand Pose Estimation and Gesture Recognition

Hand pose estimation, as well as hand gesture recognition, has been explored widely using various kinds of techniques in the field of computer vision. Related neural network-based hand pose

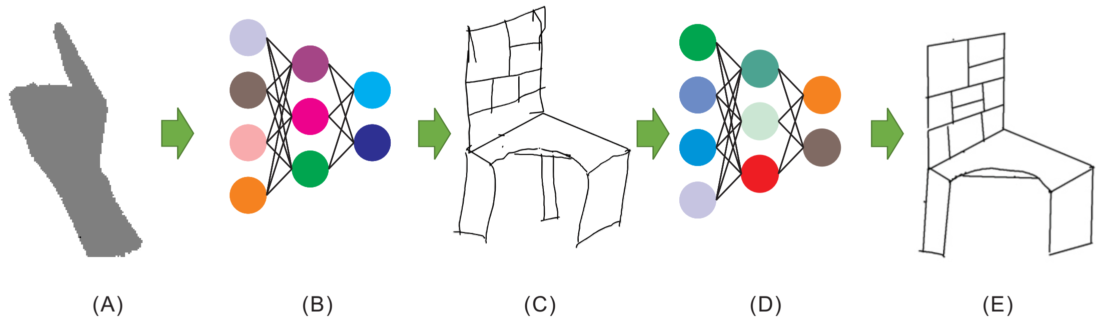


Figure 2: ARSketch Overview. (A) Input depth image. (B) Hand pose estimation network. (C) Generated sketch using hand positions. (D) Sketch network. (E) Retrieved results.

estimation and gesture recognition approaches using depth images are reviewed as follows.

The goal of hand pose estimation is to estimate the 3D location of hand joints from one or more frames recorded from a depth camera. The neural network based methods can be roughly classified into two categories: 2D CNNs and 3D CNNs. The 2D CNNs estimate hand pose directly from depth images, representative methods include a cascaded multistage method [3, 39], a structure-aware regression approach [32], and a hierarchical tree-like structured CNN [18]. These methods take depth maps as 2D images for the input of neural networks. Therefore, these neural networks are unable to fully capture the 3D information. The 3D CNNs convert 2D depth images into 3D data structures, such as 3D voxel grids [20] or D-TSDF volumes [10], and they produce state-of-the-art results. These methods are effective, however, it is still challenging to deploy an energy efficient system on a resource limited platforms. ARSketch is inspired by [13], which uses depthwise separable convolutions for computation/energy efficiency.

Hand gesture recognition aims to understand the semantic meaning of hand gestures, like zooming in gesture using two fingers. Related methods can be roughly classified by how to model the spatial-temporal information. [15, 28] extract features of each input frame individually. Then video classifiers are trained by converting frame features to video temporal descriptors. C3D [33] models spatial-temporal information using 3D neural networks. It takes a sequence of frames as input and extracts features directly. Several approaches make use of recurrent neural networks (RNN) or long short term memory (LSTM) models. For example, [38] models temporal information over multiple consecutive frames. As the focus of ARSketch is hand pose tracking for sketch, as opposite to hand gesture recognition, ARSketch uses [38] directly for hand gesture recognition to control the sketch process.

2.3 Air-Writing Recognition

Air-writing recognition refers to recognizing linguistic characters or words that are written in a free space by hand or finger movements. In [1] and [2], Mingyu et al. propose a novel Hidden Markov Model (HMM)-based method to recognize characters generated by hand gesture tracking with a 6 DOF motion tracking system. The HMM models fully exploit the combination of pure optical, pure inertial, and complete 6-DOF features. [5] proposes to capture the spatial-temporal information of hand movement with a Myo-armband to creat documents. Its recognition module is comprised of one CNN and two gated Recurrent Unit (gRU) networks. The outputs from these three networks are fused to get the final prediction of the characters written in air. Prasun et al. [25] propose to track the trajectory of a marker tip, and use a neural network to recognize digits and different language characters from the trajectories. This method is fully independent of any depth or motion sensor. [21] proposes to recognize air-writing in video. This system detects hands using Faster R-CNN followed by hand segmentation and distance-weighted curvature entropy for tip detection and tracking. In [6], a 3D drawing system was proposed on Microsoft HoloLens.

ARSketch focuses on sketch analysis specifically for AR glasses. This is unique compared to all the above works that target digit or character recognition tasks for non-AR devices.

3 ARSKETCH SYSTEM OVERVIEW

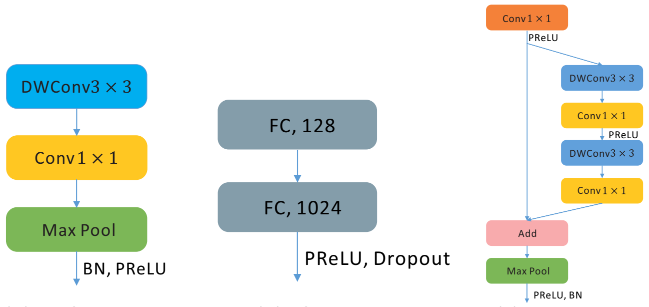
ARSketch, as shown in Figure 2, is a novel sketch-based user interface on AR glasses that combines hand pose estimation and sketch auto-completion. It addresses the complex input issue for AR glasses by sketching in air. The input of ARSketch is a sequence of depth images captured by a Time-of-Flight (ToF) camera. Figure 2 details the three components of ARSketch that work in concert: (A) hand pose estimation (Section 4.1) that provides a neural network based pipeline to estimate hand poses in an energy-efficient way, (B) sketch generation (Section 4.2) that generate sketches using key point positions of hand poses, and (C) retrieval-based auto completion (Section 4.3) that takes sketches as input to another neural network followed by a nearest neighbor search to auto-complete sketches by retrieving relevant ones.

We develop our system using a Rokid Glass [24] equipped with a pmd flexx camera as shown in Fig 1. *The Rokid Glass* is an optical see-through head-mounted AR glass released in 2018 by Rokid Corporation Ltd. This glass has 150 grams equipped with a Qualcomm APQ835 CPU with 4 GB RAM. The display is monocular and its field-of-view (FoV), resolution and frame rate are 30°, 1280×960 and 60, respectively. It uses Android system and supports USB-C connectivity. *The pmd flexx camera*

4 ARSKETCH SOFTWARE

4.1 Hand Pose Estimation

Hand pose estimation is to estimate the 3D location of hand joints from one or more frames recorded from a depth camera. Existing methods are mainly designed for gestures with non-egocentric



(A) DWConv BLK (B) FC BLK (c) Res BLK

Figure 3: C-Net and R-Net Architectures.

Table 1: C-Net and R-Net Configurations

Net	Operator	Output
C-Net	3× DWConv BLK	12 × 12 × 48
	FC 128 , ReLU	128
	FC 3	3
	Softmax	3
R-Net	2× DWConv BLK	48 × 48 × 12
	2× DWRes BLK	12 × 12 × 48
	2× FC BLK	1, 024
	FC 63	63

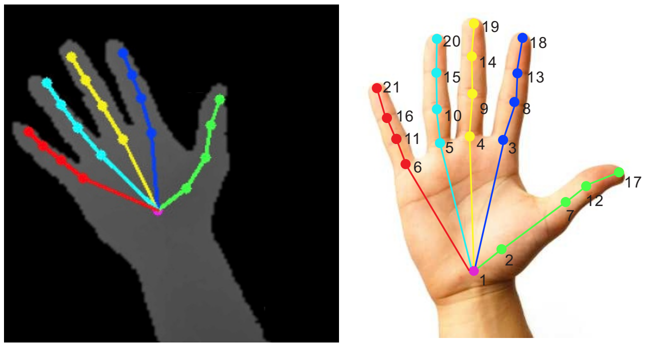
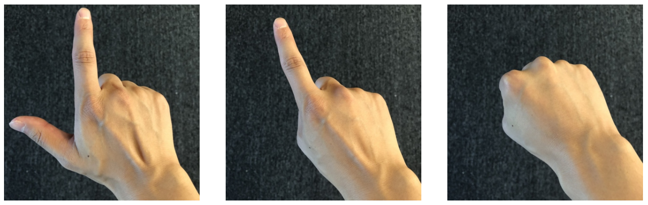


Figure 4: The 21 hand joints model in ARSketch. The numbers indicate the output order of R-Net



(A) Pointing (B) Moving (C) Stopping

Figure 5: Basic Gestures in ARSketch

view, which is distinct from gestures captured by a egocentric cameras mounted on a AR/MR device. For this, we propose a new 3D hand pose estimation method for egocentric depth images. The full pipeline includes *foreground extraction*, *hand region classification* and *hand pose regression*.

Foreground Extraction To estimate hand poses, we need to get the region of hands. In egocentric view, hands usually are the closest objects to the depth camera and are very likely in the foreground. Thus, the first step is to extract the foreground region. Given a depth image, we segment it into regions based the flood fill algorithm, and use the region with smallest average depth values as foreground. Small regions exceed a threshold will be filtered out in this procedure. We then extract a fixed-size patch from this foreground region based on [3, 22]. The center of the patch, denoted as c, is determined by calculating the center of mass of this region. The extracted patch is then resized into 96 × 96, and its corresponding depth value for each pixel is first truncated into $[c - u/2, c + u/2]$ and then normalized into $[-1, 1]$, where u is empirically set as 300 mm. Besides, depth values that are outside the patch are truncated as well to improve the system robustness.

Hand Region Classification Considering that the nearest region is not always hand, after we get the fixed-size cube, we need to check whether it holds true or not. We form this problem into a three-category classification problem: left hand, right hand and no hand. For this, we design a three-category classification neural network, denoted as C-Net. The summary of this network is given in Table 1 and Figure 3. This network consists of three depthwise convolutional (DWConv) blocks and two fully-connected (FC) layers. Here we resort to the depthwise separable convolutions [13] to improve computation/energy efficiency on a mobile device, The number of output channels in these three DWConv blocks are 12, 24, 48, respectively. The first FC layer is followed by a ReLU activation, while the second FC layer connects to a softmax layer with 3-dimension output, which represents the classification probability of foreground for each class. The strides of all max pooling layers are set to 2.

Instead of training C-Net with Mean Squared Error, we train it by using a Cross Entropy Loss because the Cross Entropy Loss gives more emphasis on correct labels than the Mean Squared Error. The loss function is defined as:

$$L_E = -\sum_{l=1}^{M} y_{i,l} \log(p_{i,l}) \tag{1}$$

where M is 3 as the number of classes, $y_{i,l}$ indicates whether the class label l is correct for the input i or not, and $p_{i,l}$ is the predicted class label.

Hand Pose Regression In this step, we estimate the poses of hand joints, if the fixed-size cube is classified as a hand region in the previous step. As shown in Figure 4, we resort to the 21 hand joints model because it describes all the motion information of a hand.

We use a neural network-based method to estimate the 3D positions of hand joints. The proposed network, denoted as R-Net, is detailed in Table 1 and Figure 3. It consists of two DWConv blocks, two depthwise residual (DWRes) blocks and two FC blocks. Similar to C-Net, we also utilize the depthwise separable convolutions for acceleration and energy efficiency on mobile devices. In addition, the third and forth blocks also use the residual connections [12] to improve accuracy. The number of output channels of the first four blocks are 12, 12, 24, 48, respectively. And there is no max pooling operation in the first block. For the two FC blocks, each of them consists of two FC layers, a PReLU and a dropout layer. The second fully-connected block finally connects to a FC layer with 63-dimension output, which represents the 3D positions of 21 hand joints. The strides of all max pooling layer are set to 2. R-Net is designed for right hand pose regression, and the left hand regions will be first flipped along the vertical axes and then taken as the right one.

We train R-Net with a Wing Loss [9], because the Wing Loss is robust for both small and large pose deviations. Given an estimated pose p_i and its corresponding ground truth q_i, the Wing Loss is defined as:

$$L_W = \begin{cases} w \ln(1 + \|x_i\|/\epsilon) & \text{if } \|x_i\| < w \\ \|x_i\| - C & \text{otherwise} \end{cases} \tag{2}$$

where $x_i = p_i - q_i$, w controls the width of non-linear part to be within $[-w, w]$, ϵ limits the curvature of the nonlinear part, and $C = w - w\ln(1+x_i/\epsilon)$ links the linear and non-linear parts together. The parameter w and ϵ are empirically set as 14 and 2 respectively.

Training Procedure Both C-Net and R-Net are first pretrained using the data of Hands 2017 Challenge [36]. This task provides 2, 965 frames of fully annotated hands that are interacting with different objects. All these frames are captured in egocentric setting. We then fine tune these networks using dataset collected from our targeting hardware prototype to improve performance. The collected dataset includes 30, 000 full annotated frames in total, and then we splits them into 25, 000 and 5, 000 for training and testing, respectively. In addition, the training data is randomly translated in the range of $[-30, 30]$ millimeters and randomly rotated in the range of $[-\pi/18, \pi/18]$ for data augmentation. For C-Net, we set the batch size to 1, 024 and the learning rate to 0.0015, and we use Adam as the optimizer. These parameters for R-Net are 256, 0.0008, and Adam, respectively.

4.2 Sketch Generation

When the system starts, a virtual sketch pad that works as the sketch interface is shown on the glass (see Figure 1). The projection of the virtual view is set to orthographic because of vergence-accommodation conflict. The hand gesture tracking system of ARSketch tracks the hand position and projects it to the sketch pad on the fly.

Unfortunately, the recorded hand trajectories cannot be used as sketches directly. Noises in the depth maps and the slight hand shakiness make it difficult to to draw straight lines and to extract strokes consequently. To address this issue, we propose to record key point positions instead of recording every hand position. We define three basic and static gestures of the right hand as shown in Figure 5 to control when and where to record the key points. These three gestures are called *pointing*, *moving* and *stopping* respectively. ARSketch uses our internal accelerated version of [38] to recognize these gestures on every frame. Based on these basic and static gestures, we define three modes to draw sketches:

- *Freedrawing* allows a user to draw continuously. The user need to perform the *pointing* gesture all the time to indicate ARSketch to record all the hand positions as a hand moves in air.
- *Pointline* allows a user to draw a line by specifying key points.

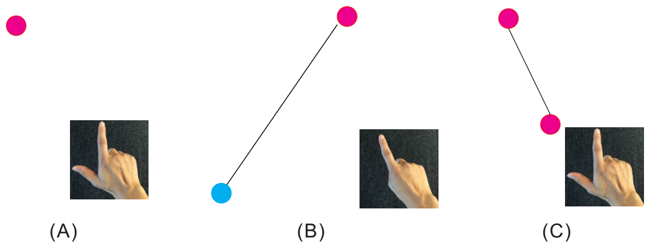


Figure 6: Drawing a line using *Pointline* mode.

- *Auto-Freedrawing* allows ARSketch to complete users' drawing with *Freedrawing* mode on-the-fly, as discussed in Section 4.3.

Figure 6 shows how to draw a line using the *Pointline* mode. The user first puts a key point (the red dot) using the *pointing* gesture (A). Then the user uses the *moving* gesture to move his/her hand to a desired position in (B). This point is marked as "blue" because it is a temporal position. Next the user puts a key point (the second red dot) using the *pointing* gesture again to indicate ARSketch to record the current position (C). Finally the user terminates drawing by performing the *stopping* gesture.

To reduce noises and jitters, we apply L_0 smoothing to each dimension of the curves separately to reduce noises and jitters. The final sketches are obtained as:

$$\min_h \sum_i (h_i - g_i)^2 \quad \text{s.t. } c(h) = k \tag{3}$$

where g and h are input and output signals, $c(h) = k$ indicates that k non-zero gradients exist in the result, and $c(h)$ is the counting operator of non-zero gradients:

$$c(h) = \#\{i| \ |h_i - h_{i-1}| \neq 0\} \tag{4}$$

4.3 Sketch Auto-Completion

Given a user's input sketch (partial sketch or whole sketch), ARSketch automatically complete this sketch in its *Auto-Freedrawing* mode by matching the correspondences between it with sketches in database. For this, we collect a sketch dataset with our prototype, extract features for the collected sketch dataset via a neural network, and retrieve the most relevant image based on a k-nearest neighbors (k-NN) search with L_2 distance.

Sketch Data Collection In the data collection phase, users are required to wear our prototype and draw sketches for images that are shown on the glass display. We recover the hand poses from depth images and then show the sketches on the AR display. We collect sketch data for ten categories of objects, including table, banana, apple, door, cup, chair, shoe, lamp, bike and car. Each category contains 120 photos. We show the photo one by one to the participants and ask them to sketch the main object. Finally, we collected 2,425 sketch images to make sure each photo should be sketched at least twice. Each sketch is created in 8 minutes on average. Some examples of our dataset are given in Figure 8.

Neural Network Based Feature Extraction To match the correspondence between sketches, we design a neural network based on EfficientNet-B0 [31] to extract their features in advance. Table 2 gives the detailed architecture. The input of this network is an image with size 224 × 224, while the output is a 128-dimensional

Table 2: The summary of sketch network architecture

Block	Operator	Input Size	#Channels	#Layers
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	14×14	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	128	1

vector. The 128-dimensional outputs are further trained to produce an adequate embedding by using a Triplet Loss in the last layer. Beside the input and output, the rest of the architectures are the same as that of the original EfficientNet-B0.

Triplet Loss. We train the neural network with a Triplet Loss function that involves the neural network outputs of three inputs: an 'anchor' sketch s, a 'positive' matched sketch image p^+ and a 'negative' mismatched sketch image p^-. This gives us both positive pair and negative pair of inputs for training. The positive pair $\mathbf{p}\left(s, p^+\right)$ contains a sketch pair corresponded to each other. The negative pair $\mathbf{p}\left(s, p^-\right)$ is a sketch pair that are not corresponded.

During training, each triplet $\mathbf{p}(s, p^+, p^-)$ is passed through the sketch network to get the embeddings, $f(s)$, $f(p^+)$ and $f(p^-)$. We then compute the L_2 distance, denoted as D, between these outputs. For the positive pair, D is calculated as

$$D\left(\mathbf{p}\left(s, p^+\right)\right) = ||f(s) - f(p^+)||^2 \tag{5}$$

Similarly, for the distance of the negative pair, we have

$$D\left(\mathbf{p}\left(s, p^-\right)\right) = ||f(s) - f(p^-)||^2 \tag{6}$$

Based on these distances, a triplet loss function is formulated as,

$$L_T = D(\mathbf{p}(s, p^+)) + \max(0, m - D(\mathbf{p}(s, p^-))) \tag{7}$$

where m is the margin between positive and negative pairs. This margin m is maintained to prevent the network from learning zero embeddings for all samples.

Training Procedure. We first pretrain the neural network on QMUL Shoe [35]. This dataset contains 2,000 photos and 6,648 SVG vector format sketch images. Considering that these sketches are not drawn with ARSketch, we add random noises and jitters for data augmentation. Next, we fine tune our network with our collect sketch dataset. We split it into 2,000 for training and 425 for testing. The training data are randomly rotated in the range of $[-\pi/18, \pi/18]$, and randomly scaled in the range of $[-1.2, 1.2]$ for data augmentation. During the training phase, all sketch images are resized to 224×224 pixels, and the pixel values are normalized to be within $[-1, 1]$. The network is trained using an Adamax Optimizer. The learning rate, mini-batch size, and the margin are set to 0.0006, 132, and 0.4, respectively. Step-wise learning rate scheduler is used. **Retrieval-based Auto-Completion** When a user inputs a sketch with *Auto-Freedrawing* mode, ARSketch compares the difference between this sketch and all sketches stored in the dataset based on K-NN search. The retrieval results are ranked by using L_2 distance, and ARSketch will return the most relevant sketch to the user.

Table 3: Accuracy Comparison for Hand Pose Estimation

Method	3D error (mm)
REN [11] w/o fine tune	32.5
REN [11] w/ fine tune	12.18
Pose-REN [3] w/o fine tune	29.6
Pose-REN [3] w/ fine tune	9.4
ARSketch	7.42

Table 4: Performance Comparison for Hand Pose Estimation

Method	MFLOPS	Time (ms)
REN [11]	301	85
Pose-REN [3]	354	100
ARSketch w/ MobileNet	14.75	5
ARSketch w/o MobileNet	77.18	30

5 EXPERIMENTS AND RESULTS

5.1 Hand Pose Estimation

We measure the 3D average joint errors (in mm) of different methods on testing dataset for accuracy comparison. Table 3 shows the results, in which both results for competitors with/without fine tuning on our collected training data are listed. Without fine tuning, REN and Pose-REN perform poor on our testing dataset. This is mainly caused by the differences between statistic distributions of their training datasets and our collected training dataset. Our dataset is tailored-made for our AR device, and it contains more severe self-occlusion cases than their training data. After fine tuning on our training dataset, the performance of our competitors is greatly improved. Nevertheless, ARSketch still outperforms the competitors because ARSketch is specifically designed for our use case and it's less likely to overfit than the competitors.

Figure 7 presents qualitative comparison between ARSketch and Pose-REN (with fine tuning) on our testing dataset. Pose-REN performs well on data without severe self-occlusions (Column 1 and Column 7), while the quality degrades notably for cases with self-occlusions (Column 2 and Column 5). However, ARSketch is able to achieve similar results with ground truth for all cases. The ego-centric C-Net of ARSketch filters out inputs frames without hands and helps R-Net focusing on pose estimation, both of which lead to significant quality/performance improvements.

Table 4 summarizes the computation performance of different methods in terms of MFLOPS and latency on our hardware device with an embedded Qualcomm APQ835 CPU and 4 GB RAM. REN and Pose-REN have 13× and 15× larger FLOPs than ARSketch respectively. The large amount of computations make them difficult to run on mobile devices in real time. On the contrary, ARSketch exploits the MobileNet architecture, and C-Net filters out frames without hands. Both of them help ARSketch to reduce computation and achieve real time performance.

5.2 Sketch Retrieval

Quantitative Results We compare Top-1 and Top-10 accuracy performance of ARSketch with ResNet-50 [12]. We choose ResNet-50 as our competitors because ResNet-50 is a successfully neural network which is used in many computer vision applications. In our

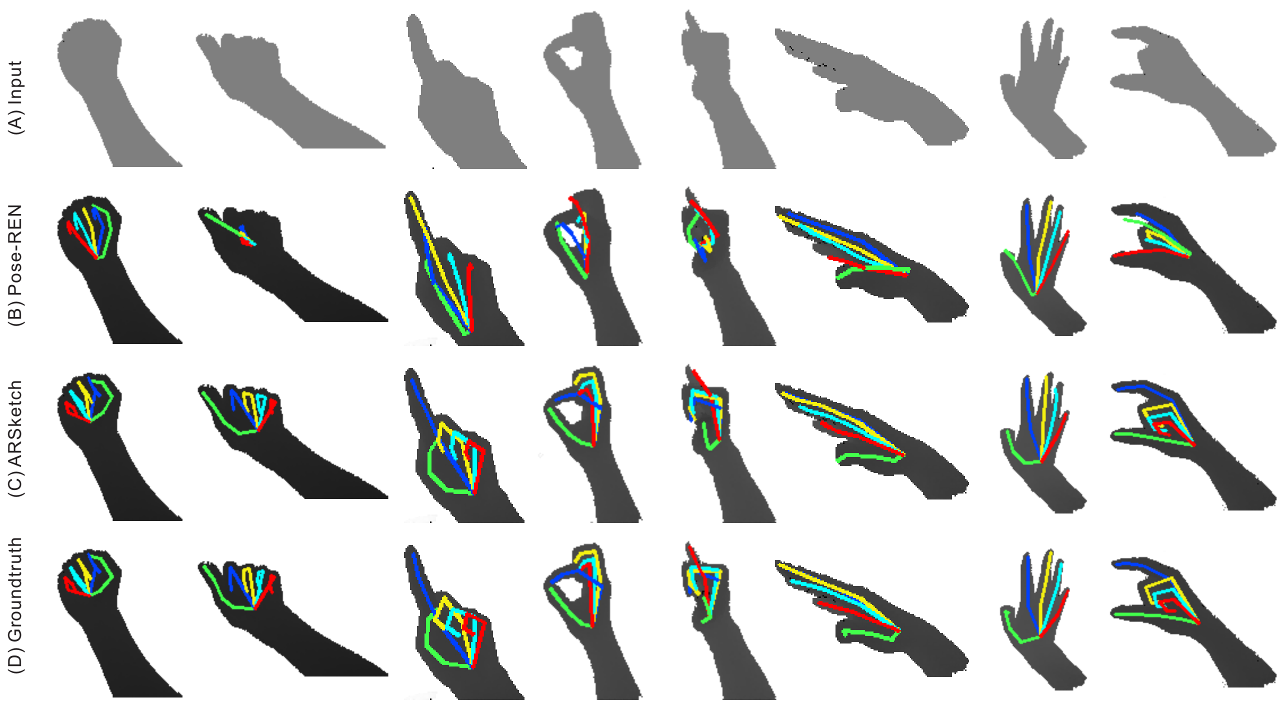


Figure 7: Qualitative Comparison Between ARSketch and Pose-REN for Hand Pose Estimation

experiment, ResNet-50 is trained with our training sketch dataset and then tested on our testing sketch dataset.

The output of ResNet-50 is also set to a 128 dimensions vector. As shown in Table 5, ARSketch can capture the semantic meaning of complex sketches effectively, and outperforms ResNet-50 in terms of Top-1 and Top-10 performance at 53.4% and 82.1% respectively.

Qualitative Results We visualize query results from the test set to qualitatively understand the performance of ARSketch in Figure 8. ARSketch is able to retrieval similar sketches based on input images, validating that ARSketch can be effectively used for sketch auto-completion.

6 USER STUDY

We evaluate user performance and experience with three drawing modes introduced in Section 4.2. We recruited 30 participants (9 female, 21 male) with a mean age of 27 (SD = 4.2). 26 of them have no experience with AR glasses. Their professions include graduate students, engineers and product designers. Before the start of the experiment, we introduced the AR glass and how to use ARSketch and participants were required to complete a training task. Then, participants were required to complete two main drawing tasks in the experiment.

6.1 Training Task

The goal of this task is to make participants be familiar with ARSketch. Participants are required to complete drawing lines using *Pointline* and *Freedrawing*. There are eight sub-tasks. In each sub-task, a line is presented on the glass display, and the participants are required to draw the line two times. The first time is using *Pointline* and the second time is using *Freedrawing*. Participants repeat these tasks as many times as possible, until they are able to finish all these eight tasks in 20 seconds.

Table 5: Top-k Accuracy for Sketch Retrieval.

Method	Top-1	Top-10
ResNet-50	44.5 %	74.9 %
ARSketch	53.4 %	82.1 %

6.2 Drawing Basic Shapes

In this task, participants are required to complete drawing basic shapes including triangles, rectangles, pentagons, hexagons, ellipses, and hearts. Selective drawings are given in Figure 9. Users can either use *Pointline* mode or *Freedrawing* mode. Compared to lines created by using an electronic pad, which are not straight enough, the drawn polygons using ARSketch are almost perfect straight. This is because we record key points instead of all positions. For ellipse and heart shape that do not contain straight lines, sketches obtained by ARSketch are not as good as that created by using an electronic pad. This is because the user needs to keep pointing gesture all the time to inform the device to record all hand poses. Slight hand shakiness and limitations of sensors will introduce noises and jitters to the curve.

6.3 Drawing Templates

In this task, participants are required to perform drawing follow templates which are complex sketch images. Then participants are request to draw each template either using the *Freedrawing* mode or *Auto-Freedrawing* mode randomly. There will be twenty templates for each participant.

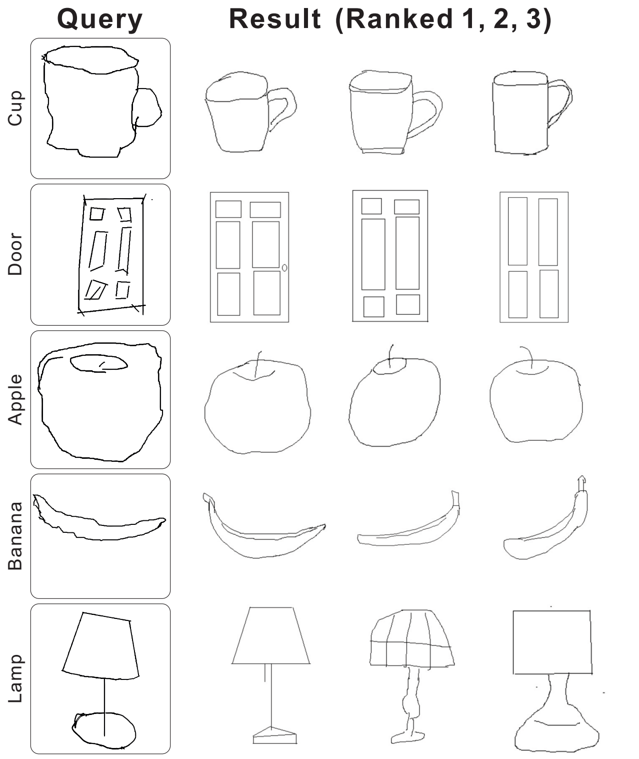


Figure 8: Query Results for Sketches. ARSKETCH **is able to retrieve similar sketches.**

Table 6: Drawing basic shapes task: average drawing time and average drawing accuracy

Mode	Times (s)	Scores
Pointline	15.7 ± 5.3	7.7 ± 2.3
Freedrawing	10.2 ± 3.9	7.5 ± 1.9

6.4 Results

We use two metrics to evaluate the two main drawing tasks.
Drawing Time Task drawing time is measured based on the elapsed time between the first and last drawing point.
Drawing Accuracy We recruited four helpers with design experience from Upwork to evaluate the drawings. For the drawing basic shapes task, each drawing is rated in the scale of 0 to 10 as how good the drawing is. For the drawing templates task, each drawing is rated in the scale of 0 to 10 as how similar it is to its corresponding template.

The average time and drawing accuracy of drawing basic shapes task are given in Table 6. The *Pointline* mode and *Freedrawing* mode give similar accuracy scores, but *Pointline* mode takes 30% more time on average than *Freedrawing* mode, because this mode requires to draw each line one by one.

For drawing template task, the average time and drawing accuracy are given in Table 7. *Auto-Freedrawing* can achieve up to 2× speedup compared with *Freedrawing* in drawing templates, demonstrating that *Auto-Freedrawing* can lower the users barrier to interact with AR devices.

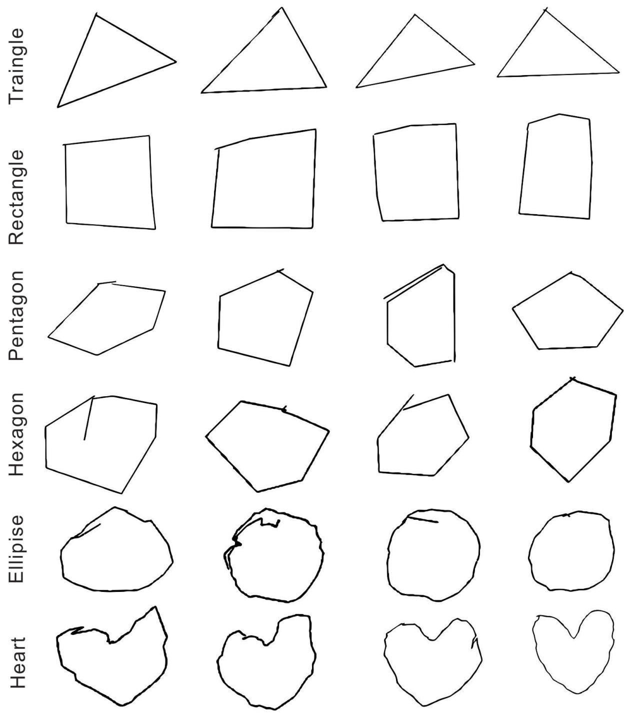


Figure 9: Sketches of basic shapes generated by ARSKETCH**. We show the drawings of four participants.**

Table 7: Drawing templates task : average drawing time and average drawing accuracy

Mode	Times (m)	Scores
Freedrawing	5.8 ± 2.1	6.8 ± 1.2
Auto-Freedrawing	2.2 ± 0.8	8.1 ± 0.3

7 CONCLUSION

Hand gestures are naturally and frequently used in interacting with AR/MR devices. Traditional hand gesture-based interaction is limited to only a few less-expressive gestures. ARSKETCH is a novel sketch-based neural network-driven user interface for AR/MR glasses, which is also applicable to other AR devices. ARSKETCH has three components that work in concert: *hand pose estimation*, *sketch generation*, and *sketch auto-completion.* The evaluation results on our collected sketch dataset demonstrate that: 1. The hand pose estimation of ARSKETCH for egocentric images achieves state-of-the-art performance in terms of both accuracy and computational efficiency, and 2. the interface provide efficient user experience for sketch generation.

ACKNOWLEDGEMENT

We would like to Yiming Chen (DiDi Labs) and anonymous reviewers for their valuable feedback. We thank Xinghong Hu (ByteDance) and Menghan Xia (The Chinese University of Hong Kong) for proofreading the manuscript. This work is supported by Rokid Corporation Limited.

REFERENCES

[1] Mingyu Chen, Ghassan AlRegib, and Biing-Hwang Juang. 2015. Air-writing recognition-Part I: Modeling and recognition of characters, words, and connecting motions. *IEEE Transactions on Human-Machine Systems* 46, 3 (2015), 403–413.

[2] Mingyu Chen, Ghassan AlRegib, and Biing-Hwang Juang. 2015. Air-writing recognition-Part II: Detection and recognition of writing activity in continuous stream of motion data. *IEEE Transactions on Human-Machine Systems* 46, 3 (2015), 436–444.

[3] Xinghao Chen, Guijin Wang, Hengkai Guo, and Cairong Zhang. 2019. Pose guided structured region ensemble network for cascaded hand pose estimation. *Neurocomputing* (2019).

[4] John Collomosse, Tu Bui, Michael J Wilber, Chen Fang, and Hailin Jin. 2017. Sketching with style: Visual search with sketches and aesthetic context. In *Proceedings of the IEEE International Conference on Computer Vision.* 2660–2668.

[5] Ayushman Dash, Amit Sahu, Rajveer Shringi, John Gamboa, Muhammad Zeshan Afzal, Muhammad Imran Malik, Andreas Dengel, and Sheraz Ahmed. 2017. Airscript-creating documents in air. In *Proceedings of the IEEE International Conference on Document Analysis and Recognition*, Vol. 1. IEEE, 908–913.

[6] John J Dudley, Hendrik Schuff, and Per Ola Kristensson. 2018. Bare-handed 3D drawing in augmented reality. In *Proceedings of the 2018 Designing Interactive Systems Conference.* 241–252.

[7] Mathias Eitz, James Hays, and Marc Alexa. 2012. How do humans sketch objects? *ACM Transactions on Graphics* 31, 4 (2012), 44–1.

[8] Mathias Eitz, Ronald Richter, Tamy Boubekeur, Kristian Hildebrand, and Marc Alexa. 2012. Sketch-based shape retrieval. *ACM Transactions on Graphics* 31, 4 (2012), 31.

[9] Zhen-Hua Feng, Josef Kittler, Muhammad Awais, Patrik Huber, and Xiao-Jun Wu. 2018. Wing loss for robust facial landmark localisation with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 2235–2245.

[10] Liuhao Ge, Hui Liang, Junsong Yuan, and Daniel Thalmann. 2017. 3d convolutional neural networks for efficient and robust hand pose estimation from single depth images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 1991–2000.

[11] Hengkai Guo, Guijin Wang, Xinghao Chen, Cairong Zhang, Fei Qiao, and Huazhong Yang. 2017. Region ensemble network: Improving convolutional network for hand pose estimation. In *Proceedings of the IEEE International Conference on Image Processing.* IEEE, 4512–4516.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 770–778.

[13] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861* (2017).

[14] Forrest Huang, John F Canny, and Jeffrey Nichols. 2019. Swire: Sketch-based User Interface Retrieval. In *Proceedings of the CHI Conference on Human Factors in Computing Systems.* ACM, 104.

[15] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. 2014. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition.* 1725–1732.

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems.* 1097–1105.

[17] Li Liu, Fumin Shen, Yuming Shen, Xianglong Liu, and Ling Shao. 2017. Deep sketch hashing: Fast free-hand sketch-based image retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 2862–2871.

[18] Meysam Madadi, Sergio Escalera, Xavier Baró, and Jordi Gonzalez. 2017. End-to-end global to local cnn learning for hand pose recovery in depth data. *arXiv preprint arXiv:1705.09606* (2017).

[19] Pavlo Molchanov, Xiaodong Yang, Shalini Gupta, Kihwan Kim, Stephen Tyree, and Jan Kautz. 2016. Online detection and classification of dynamic hand gestures with recurrent 3d convolutional neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 4207–4215.

[20] Gyeongsik Moon, Ju Yong Chang, and Kyoung Mu Lee. 2018. V2v-posenet: Voxel-to-voxel prediction network for accurate 3d hand and human pose estimation from a single depth map. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 5079–5088.

[21] Sohom Mukherjee, Sk Arif Ahmed, Debi Prosad Dogra, Samarjit Kar, and Partha Pratim Roy. 2019. Fingertip Detection and Tracking for Recognition of Air-Writing in Videos. *Expert Systems with Applications* (2019).

[22] Markus Oberweger, Paul Wohlhart, and Vincent Lepetit. 2015. Training a feedback loop for hand pose estimation. In *Proceedings of the IEEE International Conference on Computer Vision.* 3316–3324.

[23] Kaiyue Pang, Ke Li, Yongxin Yang, Honggang Zhang, Timothy M Hospedales, Tao Xiang, and Yi-Zhe Song. 2019. Generalising Fine-Grained Sketch-Based Image Retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 677–686.

[24] Rokid. 2018. *Rokid Glass.* https://glass.rokid.com/en

[25] Prasun Roy, Subhankar Ghosh, and Umapada Pal. 2018. A CNN Based Framework for Unistroke Numeral Recognition in Air-Writing. In *Proceedings of the International Conference on Frontiers in Handwriting Recognition.* IEEE, 404–409.

[26] Ryan Schmidt, Brian Wyvill, Mario Costa Sousa, and Joaquim A Jorge. 2006. Shapeshop: Sketch-based solid modeling with blobtrees. In *ACM SIGGRAPH 2006 Courses.* ACM, 14.

[27] Rosália G Schneider and Tinne Tuytelaars. 2014. Sketch classification and classification-driven analysis using fisher vectors. *ACM Transactions on Graphics* 33, 6 (2014), 174.

[28] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[29] Ivan E Sutherland. 1964. Sketchpad a man-machine graphical communication system. *Simulation* 2, 5 (1964), R–3.

[30] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer Vision and Pattern Recognition.* 1–9.

[31] Mingxing Tan and Quoc V Le. 2019. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *arXiv preprint arXiv:1905.11946* (2019).

[32] Jonathan Taylor, Lucas Bordeaux, Thomas Cashman, Bob Corish, Cem Keskin, Toby Sharp, Eduardo Soto, David Sweeney, Julien Valentin, Benjamin Luff, et al. 2016. Efficient and precise interactive hand tracking through joint, continuous optimization of pose and correspondences. *ACM Transactions on Graphics* 35, 4 (2016), 143.

[33] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. 2015. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision.* 4489–4497.

[34] Jun Wan, Yibing Zhao, Shuai Zhou, Isabelle Guyon, Sergio Escalera, and Stan Z Li. 2016. Chalearn looking at people rgb-d isolated and continuous datasets for gesture recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops.* 56–64.

[35] Qian Yu, Feng Liu, Yi-Zhe Song, Tao Xiang, Timothy M Hospedales, and Chen-Change Loy. 2016. Sketch me that shoe. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 799–807.

[36] Shanxin Yuan, Qi Ye, Bjorn Stenger, Siddhant Jain, and Tae-Kyun Kim. 2017. Bighand2. 2m benchmark: Hand pose dataset and state of the art analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 4866–4874.

[37] Hua Zhang, Si Liu, Changqing Zhang, Wenqi Ren, Rui Wang, and Xiaochun Cao. 2016. Sketchnet: Sketch classification with web images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 1105–1113.

[38] Yifan Zhang, Congqi Cao, Jian Cheng, and Hanqing Lu. 2018. Egogesture: a new dataset and benchmark for egocentric hand gesture recognition. *IEEE Transactions on Multimedia* 20, 5 (2018), 1038–1050.

[39] Zhaohui Zhang, Shipeng Xie, Mingxiu Chen, and Haichao Zhu. 2020. HandAugment: A Simple Data Augmentation Method for Depth-Based 3D Hand Pose Estimation. *arXiv* (2020), arXiv–2001.