MediaPipe Hands: On-device Real-time Hand Tracking

Fan Zhang Valentin Bazarevsky Andrey Vakunov
Andrei Tkachenka George Sung Chuo-Ling Chang Matthias Grundmann
Google Research
1600 Amphitheatre Pkwy, Mountain View, CA 94043, USA
{zhafang, valik, vakunov, atkach, gsung, chuoling, grundman}@google.com

Abstract

We present a real-time on-device hand tracking solution that predicts a hand skeleton of a human from a single RGB camera for AR/VR applications. Our pipeline consists of two models: 1) a palm detector, that is providing a bounding box of a hand to, 2) a hand landmark model, that is predicting the hand skeleton. It is implemented via MediaPipe[12], a framework for building cross-platform ML solutions. The proposed model and pipeline architecture demonstrate real-time inference speed on mobile GPUs with high prediction quality. MediaPipe Hands is open sourced at https://mediapipe.dev.

1. Introduction

Hand tracking is a vital component to provide a natural way for interaction and communication in AR/VR, and has been an active research topic in the industry [2] [15]. Vision-based hand pose estimation has been studied for many years. A large portion of previous work requires specialized hardware, *e.g*. depth sensors [13][16][17][3][4]. Other solutions are not lightweight enough to run real-time on commodity mobile devices[5] and thus are limited to platforms equipped with powerful processors. In this paper, we propose a novel solution that does not require any additional hardware and performs in real-time on mobile devices. Our main contributions are:

- An efficient two-stage hand tracking pipeline that can track multiple hands in real-time on mobile devices.
- A hand pose estimation model that is capable of predicting 2.5D hand pose with only RGB input.
- And open source hand tracking pipeline as a ready-to-go solution on a variety of platforms, including Android, iOS, Web (Tensorflow.js[7]) and desktop PCs.

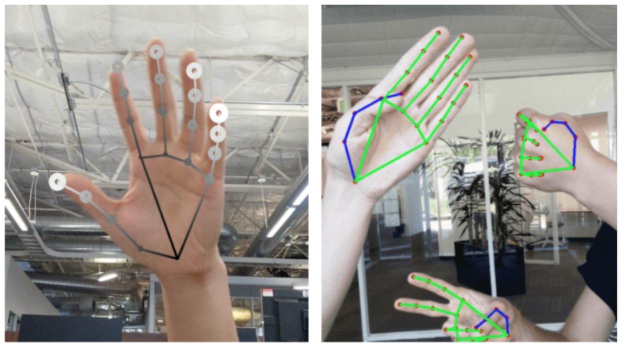

Figure 1: Rendered hand tracking result. (Left): Hand landmarks with relative depth presented in different shades. The lighter and larger the circle, the closer the landmark is towards the camera. (Right): Real-time multi-hand tracking on Pixel 3.

2. Architecture

Our hand tracking solution utilizes an ML pipeline consisting of two models working together:

- A palm detector that operates on a full input image and locates palms via an oriented hand bounding box.
- A hand landmark model that operates on the cropped hand bounding box provided by the palm detector and returns high-fidelity 2.5D landmarks.

Providing the accurately cropped palm image to the hand landmark model drastically reduces the need for data augmentation (*e.g*. rotations, translation and scale) and allows the network to dedicate most of its capacity towards landmark localization accuracy. In a real-time tracking scenario, we derive a bounding box from the landmark prediction of the previous frame as input for the current frame, thus avoiding applying the detector on every frame. Instead, the detector is only applied on the first frame or when the hand prediction indicates that the hand is lost.

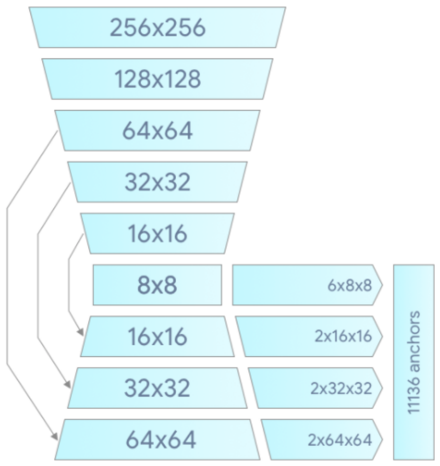

Figure 2: Palm detector model architecture.

2.1. BlazePalm Detector

To detect initial hand locations, we employ a single-shot detector model optimized for mobile real-time application similar to BlazeFace[1], which is also available in MediaPipe[12]. Detecting hands is a decidedly complex task: our model has to work across a variety of hand sizes with a large scale span (∼20x) and be able to detect occluded and self-occluded hands. Whereas faces have high contrast patterns, *e.g.*, around the eye and mouth region, the lack of such features in hands makes it comparatively difficult to detect them reliably from their visual features alone.

Our solution addresses the above challenges using different strategies.

First, we train a palm detector instead of a hand detector, since estimating bounding boxes of rigid objects like palms and fists is significantly simpler than detecting hands with articulated fingers. In addition, as palms are smaller objects, the non-maximum suppression algorithm works well even for the two-hand self-occlusion cases, like handshakes. Moreover, palms can be modelled using only square bounding boxes [11], ignoring other aspect ratios, and therefore reducing the number of anchors by a factor of 3∼5.

Second, we use an encoder-decoder feature extractor similar to FPN[9] for a larger scene-context awareness even for small objects.

Lastly, we minimize the focal loss[10] during training to support a large amount of anchors resulting from the high scale variance. High-level palm detector architecture is shown in Figure 2. We present an ablation study of our design elements in Table 1.

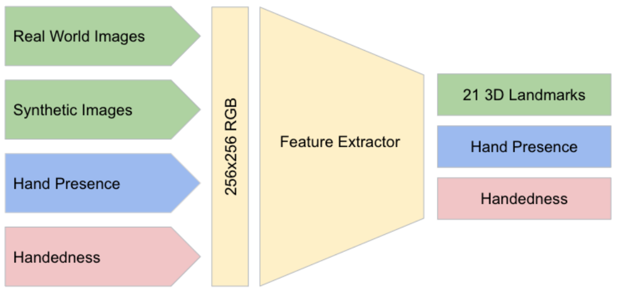

Figure 3: Architecture of our hand landmark model. The model has three outputs sharing a feature extractor. Each head is trained by correspondent datasets marked in the same color. See Section 2.2 for more detail.

2.2. Hand Landmark Model

After running palm detection over the whole image, our subsequent hand landmark model performs precise landmark localization of 21 2.5D coordinates inside the detected hand regions via regression. The model learns a consistent internal hand pose representation and is robust even to partially visible hands and self-occlusions. The model has three outputs (see Figure 3):

1. 21 hand landmarks consisting of x, y, and relative depth.
2. A hand flag indicating the probability of hand presence in the input image.
3. A binary classification of handedness, *e.g.* left or right hand.

We use the same topology as [14] for the 21 landmarks. The 2D coordinates are learned from both real-world images as well as synthetic datasets as discussed below, with the relative depth w.r.t. the wrist point being learned only from synthetic images. To recover from tracking failure, we developed another output of the model similar to [8] for producing the probability of the event that a reasonably aligned hand is indeed present in the provided crop. If the score is lower than a threshold then the detector is triggered to reset tracking. Handedness is another important attribute for effective interaction using hands in AR/VR. This is especially useful for some applications where each hand is associated with a unique functionality. Thus we developed a binary classification head to predict whether the input hand is the left or right hand. Our setup targets real-time mobile GPU inference, but we have also designed lighter and heavier versions of the model to address CPU inference on the mobile devices lacking proper GPU support and higher accuracy requirements of accuracy to run on desktop, respectively.

3. Dataset and Annotation

To obtain ground truth data, we created the following datasets addressing different aspects of the problem:

- In-the-wild dataset: This dataset contains 6K images of large variety, *e.g.* geographical diversity, various lighting conditions and hand appearance. The limitation of this dataset is that it doesn't contain complex articulation of hands.
- In-house collected gesture dataset: This dataset contains 10K images that cover various angles of all physically possible hand gestures. The limitation of this dataset is that it's collected from only 30 people with limited variation in background. The in-the-wild and in-house dataset are great complements to each other to improve robustness.
- Synthetic dataset: To even better cover the possible hand poses and provide additional supervision for depth, we render a high-quality synthetic hand model over various backgrounds and map it to the corresponding 3D coordinates. We use a commercial 3D hand model that is rigged with 24 bones and includes 36 blendshapes, which control fingers and palm thickness. The model also provides 5 textures with different skin tones. We created video sequences of transformation between hand poses and sampled 100K images from the videos. We rendered each pose with a random high-dynamic-range lighting environment and three different cameras. See Figure 4 for examples.

For the palm detector, we only use in-the-wild dataset, which is sufficient for localizing hands and offers the highest variety in appearance. However, all datasets are used for training the hand landmark model. We annotate the real-world images with 21 landmarks and use projected ground-truth 3D joints for synthetic images. For hand presence, we select a subset of real-world images as positive examples and sample on the region excluding annotated hand regions as negative examples. For handedness, we annotate a subset of real-world images with handedness to provide such data.

4. Results

For the hand landmark model, our experiments show that the combination of real-world and synthetic datasets provides the best results. See Table 2 for details. We evaluate only on real-world images. Beyond the quality improvement, training with a large synthetic dataset leads to less jitter visually across frames. This observation leads us to believe that our real-world dataset can be enlarged for better generalization.

Our target is to achieve real-time performance on mobile devices. We experimented with different model sizes and found that the "Full" model (see Table 3) provides a good trade-off between quality and speed. Increasing model capacity further introduces only minor improvements in quality but decreases significantly in speed (see Table 3 for details). We use the TensorFlow Lite GPU backend for on-device inference[6].

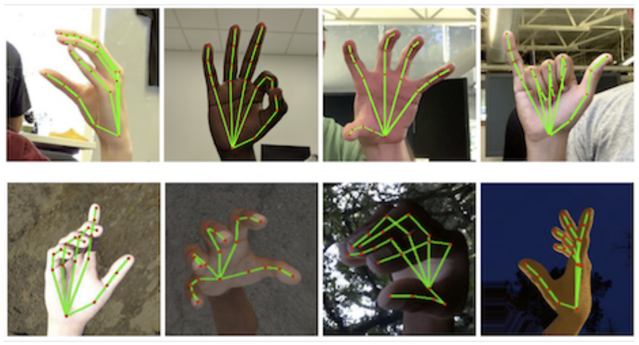
Figure 4: Examples of our datasets. (Top): Annotated real-world images. (Bottom): Rendered synthetic hand images with ground truth annotation. See Section 3 for details.

Model Variation	Average Precision
No decoder + cross entropy loss	86.22%
Decoder + cross entropy loss	94.07%
Decoder + focal loss	95.7%

Table 1: Ablation study of palm detector design elements of palm detector.

Dataset	MSE normalized by palm size
Only real-world	16.1%
Only synthetic	25.7%
Combined	13.4%

Table 2: Results of our model trained from different datasets.

Model	Params (M)	MSE	Time(ms) Pixel 3	Time(ms) Samsung S20	Time(ms) iPhone11
Light	1	11.83	6.6	5.6	1.1
Full	1.98	10.05	16.1	11.1	5.3
Heavy	4.02	9.817	36.9	25.8	7.5

Table 3: Hand landmark model performance characteristics.

5. Implementation in MediaPipe

With MediaPipe[12], our hand tracking pipeline can be built as a directed graph of modular components, called Calculators. Mediapipe comes with an extensible set of Calculators to solve tasks like model inference, media processing, and data transformations across a wide variety of devices and platforms. Individual Calculators like cropping, rendering and neural network computations are further optimized to utilize GPU acceleration. For example, we employ TFLite GPU inference on most modern phones.

Our MediaPipe graph for hand tracking is shown in Figure 5. The graph consists of two subgraphs — one for hand detection and another for landmarks computation. One key optimization MediaPipe provides is that the palm detector only runs as needed (fairly infrequently), saving significant computation. We achieve this by deriving the hand location in the current video frames from the computed hand landmarks in the previous frame, eliminating the need to apply the palm detector on every frame. For robustness, the hand tracker model also outputs an additional scalar capturing the confidence that a hand is present and reasonably aligned in the input crop. Only when the confidence falls below a certain threshold is the hand detection model reapplied to the next frame.

6. Application examples

Our hand tracking solution can readily be used in many applications such as gesture recognition and AR effects. On top of the predicted hand skeleton, we employ a simple algorithm to compute gestures, see Figure 6. First, the state of each finger, *e.g.* bent or straight, is determined via the accumulated angles of joints. Then, we map the set of finger states to a set of predefined gestures. This straightforward, yet effective technique allows us to estimate basic static gestures with reasonable quality. Beyond static gesture recognition, it is also possible to use a sequence of landmarks to predict dynamic gestures. Another application is to apply AR effects on top of the skeleton. Hand based AR effects currently enjoy high popularity. In Figure 7, we show an example AR rendering of the hand skeleton in neon light style.

7. Conclusion

In this paper, we proposed MediaPipe Hands, an end-to-end hand tracking solution that achieves real-time performance on multiple platforms. Our pipeline predicts 2.5D landmarks without any specialized hardware and thus, can be easily deployed to commodity devices. We open sourced the pipeline to encourage researchers and engineers to build gesture control and creative AR/VR applications with our pipeline.

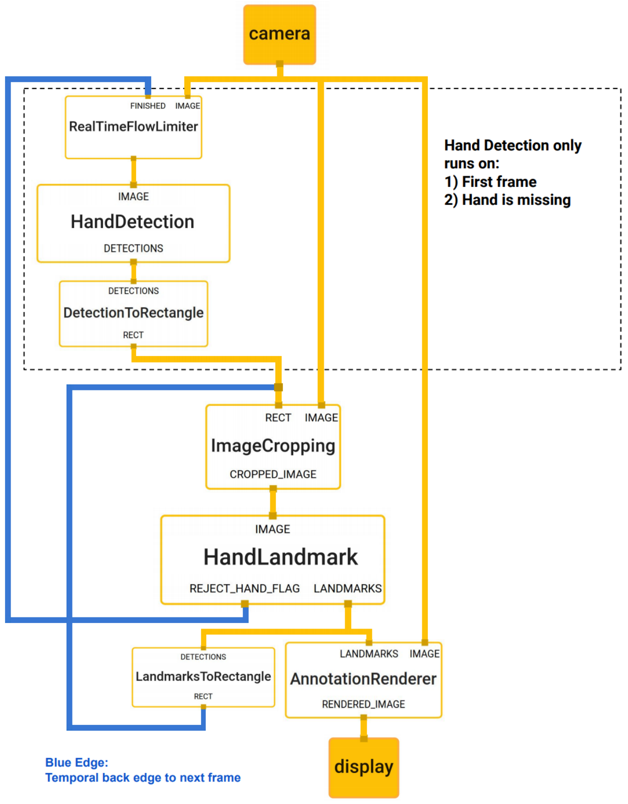

Figure 5: The hand landmark model's output controls when the hand detection model is triggered. This behavior is achieved by MediaPipe's powerful synchronization building blocks, resulting in high performance and optimal throughput of the ML pipeline.

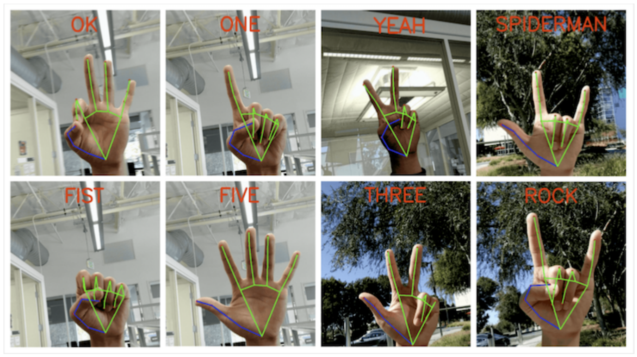

Figure 6: Screenshots of real-time gesture recognition. Semantics of gestures are rendered at top of the images.

References

[1] Valentin Bazarevsky, Yury Kartynnik, Andrey Vakunov, Karthik Raveendran, and Matthias Grundmann. Blazeface: Sub-millisecond neural face detection on mobile gpus.

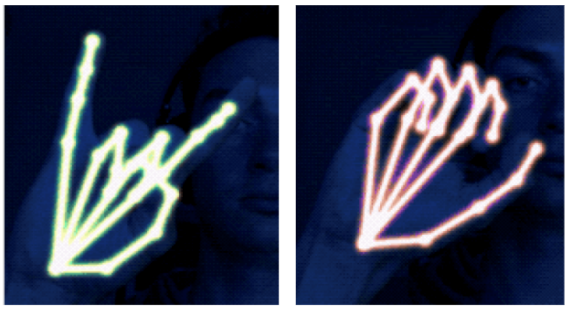

Figure 7: Example of real-time AR effects based on our predicted hand skeleton.

CoRR, abs/1907.05047, 2019. 4322

[2] Facebook. Oculus Quest Hand Tracking. https://www.oculus.com/blog/oculus-connect-6-introducing-hand-tracking-on-oculus-quest-facebook-horizon-and-more/. 4321

[3] Liuhao Ge, Hui Liang, Junsong Yuan, and Daniel Thalmann. Robust 3d hand pose estimation in single depth images: from single-view cnn to multi-view cnns. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3593–3601, 2016. 4321

[4] Liuhao Ge, Hui Liang, Junsong Yuan, and Daniel Thalmann. Robust 3d hand pose estimation from single depth images using multi-view cnns. *IEEE Transactions on Image Processing*, 27(9):4422–4436, 2018. 4321

[5] Liuhao Ge, Zhou Ren, Yuncheng Li, Zehao Xue, Yingying Wang, Jianfei Cai, and Junsong Yuan. 3d hand shape and pose estimation from a single rgb image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 10833–10842, 2019. 4321

[6] Google. Tensorflow lite on GPU. https://www.tensorflow.org/lite/performance/gpu_advanced. 4323

[7] Google. Tensorflow.js Handpose. https://blog.tensorflow.org/2020/03/face-and-hand-tracking-in-browser-with-mediapipe-and-tensorflowjs.html. 4321

[8] Yury Kartynnik, Artsiom Ablavatski, Ivan Grishchenko, and Matthias Grundmann. Real-time facial surface geometry from monocular video on mobile gpus. *CoRR*, abs/1907.06724, 2019. 4322

[9] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. *CoRR*, abs/1612.03144, 2016. 4322

[10] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *CoRR*, abs/1708.02002, 2017. 4322

[11] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: single shot multibox detector. *CoRR*, abs/1512.02325, 2015. 4322

[12] Camillo Lugaresi, Jiuqiang Tang, Hadon Nash, Chris McClanahan, Esha Uboweja, Michael Hays, Fan Zhang, Chuo-Ling Chang, Ming Guang Yong, Juhyun Lee, Wan-Teh Chang, Wei Hua, Manfred Georg, and Matthias Grundmann. Mediapipe: A framework for building perception pipelines. volume abs/1906.08172, 2019. 4321, 4322, 4324

[13] Iason Oikonomidis, Nikolaos Kyriazis, and Antonis A Argyros. Efficient model-based 3d tracking of hand articulations using kinect. 4321

[14] Tomas Simon, Hanbyul Joo, Iain A. Matthews, and Yaser Sheikh. Hand keypoint detection in single images using multiview bootstrapping. *CoRR*, abs/1704.07809, 2017. 4322

[15] Snapchat. Lens Studio by Snap Inc. https://lensstudio.snapchat.com/templates/object/hand/. 4321

[16] Andrea Tagliasacchi, Matthias Schröder, Anastasia Tkach, Sofien Bouaziz, Mario Botsch, and Mark Pauly. Robust articulated-icp for real-time hand tracking. In *Computer Graphics Forum*, volume 34, pages 101–114. Wiley Online Library, 2015. 4321

[17] Chengde Wan, Thomas Probst, Luc Van Gool, and Angela Yao. Self-supervised 3d hand pose estimation through training by fitting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10853–10862, 2019. 4321