# A Preliminary Investigation into a Deep Learning Implementation for Hand Tracking on Mobile Devices

Monica Gruosso*, Nicola Capece†, Ugo Erra* and Francesco Angiolillo*
*Department of Mathematics, Computer Science, and Economics,
University of Basilicata, Potenza, Italy 85100
Email: monica.gruosso@unibas.it, ugo.erra@unibas.it, angiolillo.francesco@gmail.com
†School of Engineering,
University of Basilicata, Potenza, Italy 85100
Email: nicola.capece@unibas.it

***Abstract*—Hand tracking is an essential component of computer graphics and human-computer interaction applications. The use of RGB camera without specific hardware and sensors (e.g., depth cameras) allows developing solutions for a plethora of devices and platforms. Although various methods were proposed, hand tracking from a single RGB camera is still a challenging research area due to occlusions, complex backgrounds, and various hand poses and gestures. We present a mobile application for 2D hand tracking from RGB images captured by the smartphone camera. The images are processed by a deep neural network, modified specifically to tackle this task and run on mobile devices, looking for a compromise between performance and computational time. Network output is used to show a 2D skeleton on the user's hand. We tested our system on several scenarios, showing an interactive hand tracking level and achieving promising results in the case of variable brightness and backgrounds and small occlusions.**

***Index Terms*—Deep Learning, Human-Computer Interaction, Image Processing, Hand Tracking**

## I. Introduction

The ability to perceive the position or movement of hands in space is an important activity to improve the user experience in a multiplicity of domains and technological platforms. For example, it can be the basis for understanding sign language [1] or recognizing gestures [2]. Hand tracking can be useful in human-computer interaction [3]–[5], VR/AR [6], [7], and robotics applications [8], [9]. The hand tracking problem consists of estimating the hand's position in the image or the 3D space. It can be performed by directly estimating its joints or using a model that produces probability density maps (heatmaps) for each joint [10]. Traditional approaches are based on markers, gloves, depth sensors, additional and specific hardware setup [11]–[13]. In recent years, several approaches based on deep learning have been developed, considering the impressive results obtained in particular for classification, detection, and regression tasks related to images [14]–[20]. Many of them propose markerless and image-based methods, but still require multiple [21], [22] and depth [23], [24] cameras. Since RGB cameras are widely available and inexpensive, the most recent works are based on a single monocular RGB input image without the need for supplementary information [25], [26]. Despite scientific and technological progress in this field, accurate hand tracking from RGB camera inputs is still challenging. Input size reduction makes the task more difficult. In addition, an incorrect estimate can be caused by hand occlusion, different or noisy backgrounds, and variable lighting conditions.

This paper focuses on 2D hand tracking in image-space from RGB inputs acquired by a smartphone camera. We designed a mobile app in which images are processed by a neural network to obtain 2D joint heatmaps. The output is post-processed to derive a 2D skeleton and interactively display it on the user's hand (Fig. 1). All computation, network loading, and inference are performed on devices locally. The network used is a deep convolutional model based on RegNet [26], which was developed for real-time hand tracking from monocular RGB images and achieved impressive results on desktop hardware. Since it was not created as a specific lightweight model for mobile or embedded devices, we sought a compromise between accurate predictions and computational time, modifying the baseline model at inference time. Therefore, we tested the proposed neural network on a desktop device, obtaining good performance at the inference phase and reducing computational times by $0.04$ seconds on average for each frame compared to the RegNet baseline model. Hands were captured in different positions, with different backgrounds, variable brightness, and small self occlusions or caused by objects. Finally, we tested our app and calculated the inference times by comparing different devices. Promising results were obtained on a high-end smartphone that achieved around $18$ fps, showing interactive hand tracking.

The remainder of this paper is structured as follows: Section II provides an overview of several interesting approaches related to RGB images; Section III describes the neural network model; Section IV describes the design and implementation of our app; Section V illustrates the result obtained and the experiments conducted; Section VI presents final considerations and future directions.

DOI 10.1109/AIVR50618.2020.00079

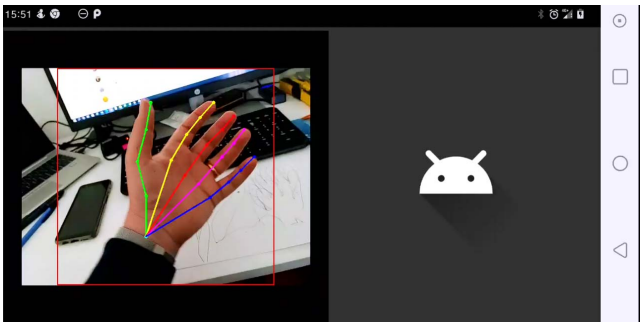

Fig. 1. An intuitive graphical user interface was designed. In particular, users can see a red square bounding box indicating the center crop area. Frames are continuously captured and processed on the fly by the neural network running locally on mobile devices. The output is shown through a 2D hand skeleton that interactively overlaps the hand.

## II. Related Work

Hand tracking from RGB images is a very active research field, especially due to the numerous and important applications. In addition, there are open and still unresolved challenges due to similarities among fingers, significant self-occlusions and object occlusions, different background colors, and complexity of hand pose, shape, and gesture [10].

In this section, we discuss some interesting methods regarding RGB images that require additional data to improve accuracy or a multi-camera setup. Finally, we explore some approaches based on a single monocular RGB camera.

### A. Methods based on RGB images and additional data

One of the most interesting traditional methods was developed by Sridhar *et al.* [27], which proposed a hybrid approach for interactive hand motion tracking. They combined RGB images captured from multiple cameras, depth sensor data, and a user-specific hand model to estimate the 3D hand pose. The advantages of the fusion of RGB and depth data were further investigated by Kazakos *et al.* [28]. Indeed, they built a double-stream deep convolutional neural network that analyzed depth and RGB images in parallel and fused their features at an intermediate layer. In the last years, weakly supervised deep learning approaches were developed. In particular, Cai *et al.* [29] presented a method that takes advantage of the depth map information acquired by an RGB-D camera during training, while only RGB inputs were used during the test phase. Similarly, Dibra *et al.* [30] and Ge *et al.* [31] proposed to fine-tune the networks on real monocular data by leveraging the depth map in the training step to enhance estimation accuracy.

### B. Methods based on a multi-camera setup

Another method developed by Sridhar *et al.* [32] employed 5 RGB cameras and hand shape representation in 3D based on a sum of anisotropic Gaussians. Simon *et al.* [21] presented a hand keypoint detector trained using the Panoptic Studio [33] that contains more than 500 cameras in a spherical space. They trained a weak detector on a small synthetic hands dataset to localize a subset of keypoints using some views. Secondly, a triangulation phase was performed using the position and parameters of all cameras to filter incorrect detections and obtain accurate 3D estimations.

### C. Methods based on a single monocular RGB camera

Several deep learning approaches based on a single RGB image were proposed. Zimmermann *et al.* [25] developed a three-step pipeline for estimating 3D hand joints using a single RGB image. Firstly, hand segmentation was performed using a convolutional neural network called HandSegNet. Then a detection-based network, PoseNet, produced 2D heatmaps. Finally, a third network named PosePrior predicted relative and normalized 3D coordinates, which were used to obtain the 3D pose estimation anchored on the palm center. Panteleris *et al.* [34] also proposed a method consisting of three phases: hand detection using YOLOv2 neural network [35]; 2D keypoints localization following the approach illustrated in [21]; and the absolute 3D pose estimation from 2D keypoints using inverse kinematics. Contrary to the previous works, others were based on two phases. In particular, Wang *et al.* [36] designed a cascaded convolutional network to estimate hand mask and 2D pose, while Iqbal *et al.* [37] presented a method for 3D hand pose reconstruction from 2.5D heatmaps estimated from a monocular image using a convolutional network. Finally, Mueller *et al.* [26] proposed a deep convolutional model, called RegNet, able to estimate 2D and 3D hand joints from a single RGB image followed by a refinement module, providing a robust hand tracking system and outperforming several methods, such as [25].

## III. Neural Network Architecture

Our neural network is based on RegNet [26], which is a real-time hand joints regressor. RegNet was trained to predict 2D joint heatmaps and 3D joint positions from a square RGB-only image. The authors demonstrated that the network benefits from jointly regressing both 2D and 3D joints at training time. The first part of RegNet is composed of 10 residual blocks of the ResNet50 model [38] followed by two branches: one consisting of convolutional and deconvolutional layers and the other consisting of fully connected layers. These two branches provide 2D and 3D hand joints predictions, respectively. The second part of RegNet is a refinement module built to better couple 2D and 3D predictions. In particular, a custom layer called projection layer was designed to orthogonally project the intermediate 3D predictions obtained from the first part of the network.

A large amount of data is usually needed to train this type of deep network. Since accurate annotation of the hand joints of thousands of images can be impractical, especially in the case of 3D joints, synthetic data is often used. Although the ground truth for synthetic hands is easy to obtain, they lack realism and can lead to poor performance on real images. Indeed, Mueller *et al.* [26] created and released a dataset consisting of 440k images that include both synthetic and "real" (GANerated) images, which were generated by GeoConGAN,

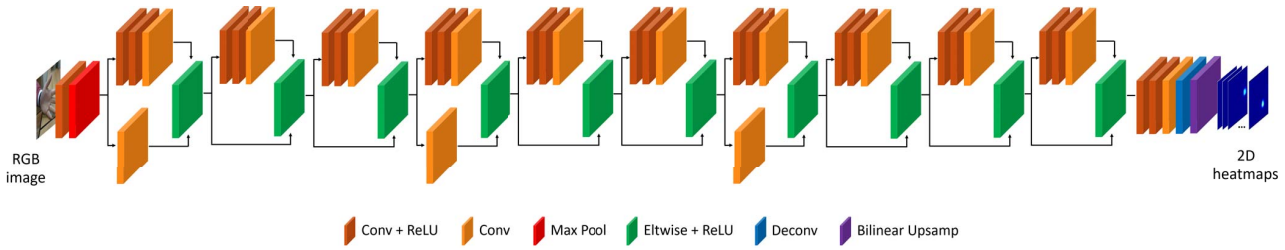


Fig. 2. Network architecture at inference time. The input is a square RGB-only image and the output is a tensor consisting of 2D joint heatmaps. All layers are drawn with blocks of the same size to simplify their graphical representation.

a geometrically consistent image-to-image translation network based on CycleGAN [39]. As usual for generative adversarial networks (GANs) [40], [41], two antagonistic networks are trained in the form of a minimax game: a generator learns to generate new synthetic data, and a discriminator learns to determine if a sample is real or not. CycleGAN is a particular type of GAN that can learn the relationship between image domains to translate an image from a source domain to a target domain in a bidirectional way, without paired input-output examples. In the case of GeoConGAN, two conditional generators and two discriminators were trained on unpaired real and synthetic images to simultaneously learn the mapping between synthetic and real images and vice versa.

We focused on RegNet considering the impressive results achieved, although it was not designed specifically for mobile devices. In particular, we were interested in *(i)* the global hand positions in the 2D image-space, and *(ii)* inference on mobile, which has fewer resources than a computer. Therefore, we decided to consider only the first part of RegNet at inference time, i.e., without the refinement module and with the only branch related to 2D predictions, and find a compromise between performance and computational costs. The model used is shown in Fig. 2. Inputs are RGB images captured by the smartphone camera and pre-processed before being passed into the neural network. Firstly, they are center cropped; secondly, images are resized to $128 \times 128$ and normalized in the range $[-1, 1]$. We experimentally verified that the resizing of images according to this criterion still allows obtaining the features necessary to tackle the hand tracking problem, partially reducing the inference time and achieving acceptable performance in terms of accuracy of the 2D predictions. Moreover, computational time was further reduced, considering the lower network depth compared to the original architecture. More information is provided in Section V-A.

The network output is a tensor with size $32 \times 32 \times 22$ consisting of 2D heatmaps in which the first 21 channels contain information about the wrist and 20 finger joints. The last tensor channel is for background, as common practice in approaches employing heatmaps [42], [43]. Then, at the inference phase, the position $(\hat{x}_k, \hat{y}_k)$ of the $k$-th joint in the image-space is recovered from the maximum values of the corresponding predicted heatmap $\hat{H}_k$ resized to the input size:

$$(\hat{x}_k, \hat{y}_k) = \arg\max_{(x,y)} \hat{H}_k(x, y), \quad k \in \{1, ..., K = 21\} \qquad (1)$$

Fig. 3 shows an input RGB image, the 22 heatmaps obtained, and an overlap between the hand and the recovered positions of the 2D joints.

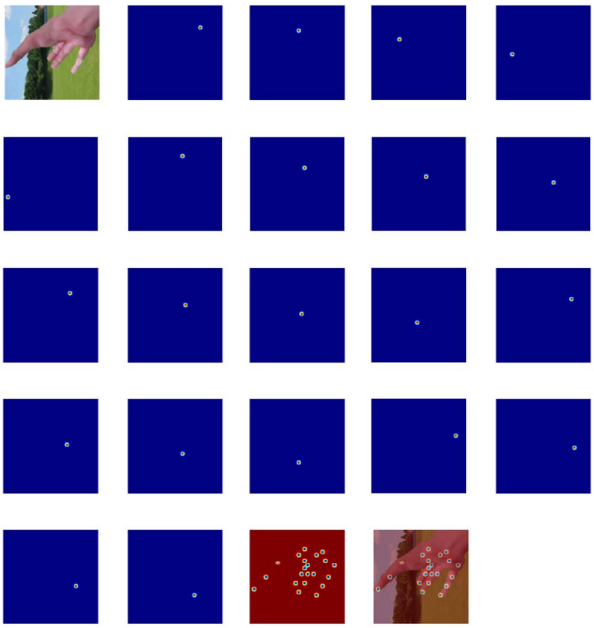

Fig. 3. The image in the top left corner is the RGB input, followed by the output resized heatmaps. The maximum values of each of the first 21 heatmaps contain information about a joint, as shown by the colored dots. The the second-last plot shows the background heatmap. The lower right corner depicts an overlap between the input and the recovered 2D joint predictions.

## IV. System Design and Implementation

We designed a mobile app for hand tracking from RGB images captured by a smartphone camera and processed by a deep convolutional neural network. It was developed in Java, C++ native code, and OpenCV [44], building an Android Studio project requiring the Android Native Development Kit (NDK) and the Java Native Interface (JNI) framework. Our application is composed of two parts. The first one is related to the creation and management of the graphical interface, and responsible for communicating with the application logic. The graphical user interface is simple and intuitive. Users can see

a red square bounding box located at the center of the image, indicating the crop area. Frames are continuously acquired and the output obtained is visualized interactively through a 2D skeleton that overlaps the hand (Fig. 1). The second part is responsible for pre-processing the acquired images, loading the proposed network architecture and the pre-trained weights corresponding to the layers considered, and run the model on mobile devices locally.

## V. Results

We conducted several tests to verify the effectiveness of the neural network and the performance of our mobile app. Firstly, we performed network inference on a computer to qualitative evaluate the output obtained and the computation time on the CPU and GPU. Moreover, we compared the proposed neural network with the baseline model. Subsequently, we tested our hand tracking method on smartphone devices.

### *A. Test on desktop hardware and comparison with the baseline model*

In the case of desktop inference, images were loaded and processed using OpenCV, the network was queried using Caffe and Python, and 2D hand skeletons were depicted with Python. Some results are shown in Fig. 4. Hands in different positions and with several background scenes can be seen. Sufficiently accurate results were also obtained in the case of partially occluded hands, although some errors in the prediction of some joints can be found, as shown in the last panels of Fig. 4. In addition, we compared the performance of our network architecture with the RegNet baseline model [26] using the same desktop configuration: a notebook with i5-8th generation CPU and 8GB RAM. Our approach achieved about 12 fps analyzing a video stream, spending an average of 0.08 seconds to obtain the prediction for each frame. The baseline model took about 0.04 seconds longer for each frame, achieving 8 fps. This corroborates that our approach achieves good performance and provides remarkable savings in computation time. In contrast, there was a noticeable reduction in response times on the GPU. We tested our model using Google Colab with a Tesla P100 GPU, making predictions in about 3 milliseconds per frame and reaching more than 300 fps.

### *B. Test on mobile devices*

In the case of our mobile application, the network inference was performed in the native code using OpenCV in Android, as described in Section IV. Fig. 5 shows some results obtained from RGB video frames captured by the smartphone camera and processed by the neural network model directly on mobile devices. The obtained heatmaps were used to calculate the 2D positions of the joints and draw the hand skeletons. Different backgrounds, variable brightness, and partial hand occlusions were considered. It is remarkable that the prediction obtained is plausible and sufficiently accurate even in the case of an object occluding a considerable part of the hand, despite the challenging scenario (last panel of Fig. 5). A video demo[1] of our work shows other results obtained by running our application.

Fig. 4. Results obtained by querying the neural network, which is the first part of RegNet, using a desktop device. The first six panels show the 2D skeletons in the case of open hands with different backgrounds, while the last six show the predictions in the case of partially occluded hands.

Since most mobile or embedded devices have limited computational performance and GPUs are not always available, performing AI-based image analysis approaches is not always feasible. Therefore, we focused on CPU performance and tested our app on three types of medium and high range smartphones, comparing response time. The calculated time is the overall pre-processing and network inference times. Table I illustrates the average frame rate and the mean inference time per image obtained on the CPU of tested mobile computing system-on-a-chips (SoCs). All platforms have an ARM-based octa-core CPU. The highest frame rate was achieved on the LG G8s smartphone with a Snapdragon 855 SoC, which processed each image and ran the model in about 0.05 seconds allowing for interactive hand tracking. Worse performances were achieved on the Samsung A7 smartphone with a Exynos 7885 SoC and the Huawei Honor 9 Lite with a HiSilicon Kirin 659 SoC. We are confident that lower response time, hence

[1]Some video results obtained with our hand tracking app are available at the following link: http://graphics.unibas.it/www/HandTrackingMobile/index.md.html

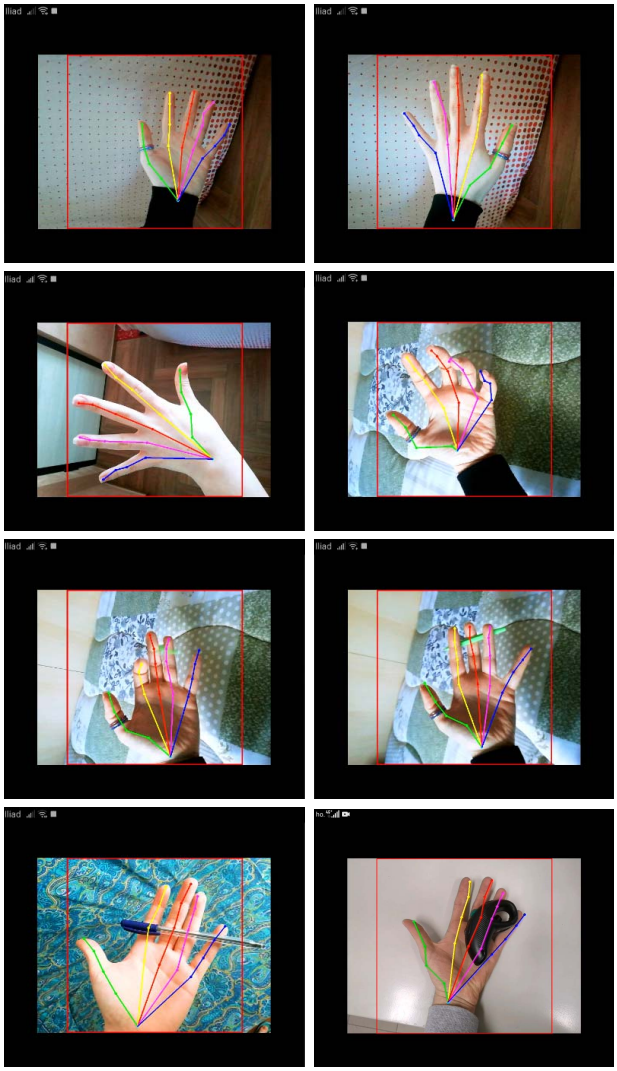

Fig. 5. 2D hand tracking on a mobile device. The RGB images were captured by the smartphone camera and processed by the proposed network architecture. Several scenarios were evaluated: open and partially closed hands are shown in the first five panels, while several occlusions are considered in the last three panels. Note that a bigger object occluding much of the hand is shown in the last panel. Although it is a challenging scenario, the prediction is plausible and sufficiently accurate.

higher fps, can be obtained with the latest and next-generation processors, such as Snapdragon 855+ and Snapdragon 865.

## VI. Conclusion

We developed an Android app for hand tracking from RGB images acquired by the smartphone camera. Input images were passed to a neural network based on RegNet [26], a model for regressing hand joints positions that achieved remarkable results, even though it was not specifically conceived to run on devices with limited resources, such as mobile and embedded platforms. Since we were interested in 2D positions and the execution on mobile devices, we focused on the first part of RegNet, removing the branch related to intermediate 3D joint

TABLE I
Performance achieved on the CPU of medium and high-end smartphones. The second column shows the average frame rate, whereas the third shows the mean time required to pre-process each image and obtain the network prediction.

| SoC | Frame rate (fps) | Inference time (s) |
|---|---|---|
| Samsung Exynos 7885 | 3 | 0.3 |
| Huawei HiSilicon Kirin 659 | 12 | 0.08 |
| Qualcomm Snapdragon 855 | 18 | 0.05 |

predictions, the projection layer, and the refinement module during the inference phase (Fig. 2), reducing computational times. In particular, we compared the proposed architecture with the RegNet baseline model using the same desktop hardware configuration. Our network obtained 0.04 seconds less for each frame prediction on average, confirming that our network architecture allows obtaining remarkable savings in computation time. Moreover, we tested our model on a Tesla P100 GPU, reaching more than 300 fps. Subsequently, inference on mobile devices was analyzed, and performance on their CPUs was considered. The lower response time was achieved on a high-end device with a Qualcomm Snapdragon 855 SoC that allowed getting hand joints predictions in about 0.05 seconds for each frame (18 fps). Furthermore, we performed qualitative evaluations using photos in which the hands were captured on different backgrounds and in various positions. Acceptable results were obtained even in the case of partial occlusions considering objects of different sizes, as can be seen in the last panels of Fig. 4 and 5.

Although accurate results were achieved in most scenarios, there are some limitations. In particular, they can be found in the case of multiple hands in the scene, completely closed and almost totally occluded hands. In addition, the bounding box was centered on the app camera preview. Therefore, it would be interesting to add a hand detection step before the hand tracking step. Another possible future development could concern the robustness of the system towards users of other ethnic groups. In that case, a more diverse training dataset containing different skin colors should be collected and used. Finally, in the future, we could try to speed up the network inference process, for example using the Android Neural Networks API (NNAPI), which is available for devices with Android 8.1 or higher. It provides acceleration for some higher-level machine learning frameworks, such as TensorFlow Lite, on Android devices with GPU or NPU.

## References


[1] F. Yin, X. Chai, and X. Chen, "Iterative reference driven metric learning for signer independent isolated sign language recognition," in *European Conference on Computer Vision*. Springer, 2016, pp. 434–450.

[2] H.-S. Yeo, B.-G. Lee, and H. Lim, "Hand tracking and gesture recognition system for human-computer interaction using low-cost hardware," *Multimedia Tools and Applications*, vol. 74, no. 8, pp. 2687–2715, 2015.

[3] A. Markussen, M. R. Jakobsen, and K. Hornbæk, "Vulture: a mid-air word-gesture keyboard," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2014, pp. 1073–1082.

[4] G. Caggianese, L. Gallo, and P. Neroni, "An investigation of leap motion based 3d manipulation techniques for use in egocentric viewpoint," in *International Conference on Augmented Reality, Virtual Reality and Computer Graphics*. Springer, 2016, pp. 318–330.

[5] G. Caggianese, N. Capece, U. Erra, L. Gallo, and M. Rinaldi, "Freehand steering locomotion techniques for immersive virtual environments: A comparative evaluation," *International Journal of Human-Computer Interaction*, 2020.

[6] T. Lee and T. Hollerer, "Multithreaded hybrid feature tracking for markerless augmented reality," *IEEE transactions on visualization and computer graphics*, vol. 15, no. 3, pp. 355–368, 2009.

[7] Y. Jang, S.-T. Noh, H. J. Chang, T.-K. Kim, and W. Woo, "3d finger cape: Clicking action and position estimation under self-occlusions in egocentric viewpoint," *IEEE Transactions on Visualization and Computer Graphics*, vol. 21, no. 4, pp. 501–510, 2015.

[8] M. H. K. Ali, M. A. Azman, Z. H. Ismail *et al.*, "Real-time hand gestures system for mobile robots control," *Procedia Engineering*, vol. 41, pp. 798–804, 2012.

[9] G. Baulig, T. Gulde, and C. Curio, "Adapting egocentric visual hand pose estimation towards a robot-controlled exoskeleton," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 0–0.

[10] B. Doosti, "Hand pose estimation: A survey," *arXiv preprint arXiv:1903.01013*, 2019.

[11] R. Y. Wang and J. Popović, "Real-time hand-tracking with a color glove," *ACM transactions on graphics (TOG)*, vol. 28, no. 3, pp. 1–8, 2009.

[12] I. Oikonomidis, N. Kyriazis, and A. A. Argyros, "Efficient model-based 3d tracking of hand articulations using kinect." in *BmVC*, vol. 1, no. 2, 2011, p. 3.

[13] C. Qian, X. Sun, Y. Wei, X. Tang, and J. Sun, "Realtime and robust hand tracking from depth," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1106–1113.

[14] A. Kendall, M. Grimes, and R. Cipolla, "Posenet: A convolutional network for real-time 6-dof camera relocalization," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2938–2946.

[15] F. Schroff, D. Kalenichenko, and J. 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.

[16] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[17] N. Capece, U. Erra, and A. V. Ciliberto, "Implementation of a coin recognition system for mobile devices with deep learning," in *2016 12th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*. IEEE, 2016, pp. 186–192.

[18] M. Gruosso, N. Capece, U. Erra, and N. Lopardo, "A deep learning approach for the motion picture content rating," in *2019 10th IEEE International Conference on Cognitive Infocommunications (CogInfoCom)*. IEEE, 2019, pp. 137–142.

[19] N. Capece, F. Banterle, P. Cignoni, F. Ganovelli, R. Scopigno, and U. Erra, "Deepflash: Turning a flash selfie into a studio portrait," *Signal Processing: Image Communication*, vol. 77, pp. 28–39, 2019.

[20] M. Gruosso, N. Capece, and U. Erra, "Human segmentation in surveillance video with deep learning," *Multimedia Tools and Applications*, 2020.

[21] T. Simon, H. Joo, I. Matthews, and Y. Sheikh, "Hand keypoint detection in single images using multiview bootstrapping," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017, pp. 1145–1153.

[22] L. Ge, H. Liang, J. Yuan, and D. Thalmann, "Robust 3d hand pose estimation from single depth images using multi-view cnns," *IEEE Transactions on Image Processing*, vol. 27, no. 9, pp. 4422–4436, 2018.

[23] H. Liang, J. Yuan, and D. Thalmann, "Parsing the hand in depth images," *IEEE Transactions on Multimedia*, vol. 16, no. 5, pp. 1241–1253, 2014.

[24] C. Wan, T. Probst, L. Van Gool, and A. Yao, "Crossing nets: Combining gans and vaes with a shared latent space for hand pose estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 680–689.

[25] C. Zimmermann and T. Brox, "Learning to estimate 3d hand pose from single rgb images," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 4903–4911.

[26] F. Mueller, F. Bernard, O. Sotnychenko, D. Mehta, S. Sridhar, D. Casas, and C. Theobalt, "Ganerated hands for real-time 3d hand tracking from monocular rgb," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 49–59.

[27] S. Sridhar, A. Oulasvirta, and C. Theobalt, "Interactive markerless articulated hand motion tracking using rgb and depth data," in *Proceedings of the IEEE international conference on computer vision*, 2013, pp. 2456–2463.

[28] E. Kazakos, C. Nikou, and I. A. Kakadiaris, "On the fusion of rgb and depth information for hand pose estimation," in *2018 25th IEEE International Conference on Image Processing (ICIP)*. IEEE, 2018, pp. 868–872.

[29] Y. Cai, L. Ge, J. Cai, and J. Yuan, "Weakly-supervised 3d hand pose estimation from monocular rgb images," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 666–682.

[30] E. Dibra, S. Melchior, A. Balkis, T. Wolf, C. Oztireli, and M. Gross, "Monocular rgb hand pose inference from unsupervised refinable nets," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 1075–1085.

[31] L. Ge, Z. Ren, Y. Li, Z. Xue, Y. Wang, J. Cai, and J. Yuan, "3d hand shape and pose estimation from a single rgb image," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2019, pp. 10 833–10 842.

[32] S. Sridhar, H. Rhodin, H.-P. Seidel, A. Oulasvirta, and C. Theobalt, "Real-time hand tracking using a sum of anisotropic gaussians model," in *2014 2nd International Conference on 3D Vision*, vol. 1. IEEE, 2014, pp. 319–326.

[33] H. Joo, T. Simon, X. Li, H. Liu, L. Tan, L. Gui, S. Banerjee, T. Godisart, B. Nabbe, I. Matthews *et al.*, "Panoptic studio: A massively multiview system for social interaction capture," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 1, pp. 190–204, 2017.

[34] P. Panteleris, I. Oikonomidis, and A. Argyros, "Using a single rgb frame for real time 3d hand pose estimation in the wild," in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2018, pp. 436–445.

[35] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.

[36] Y. Wang, C. Peng, and Y. Liu, "Mask-pose cascaded cnn for 2d hand pose estimation from single color image," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 11, pp. 3258–3268, 2018.

[37] U. Iqbal, P. Molchanov, T. Breuel Juergen Gall, and J. Kautz, "Hand pose estimation via latent 2.5 d heatmap regression," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 118–134.

[38] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[39] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017.

[40] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[41] I. Goodfellow, "Nips 2016 tutorial: Generative adversarial networks," *arXiv preprint arXiv:1701.00160*, 2016.

[42] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh, "Convolutional pose machines," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2016, pp. 4724–4732.

[43] T. Zhang, H. Lin, Z. Ju, and C. Yang, "Hand gesture recognition in complex background based on convolutional pose machine and fuzzy gaussian mixture models," *International Journal of Fuzzy Systems*, pp. 1–12, 2020.

[44] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.