DeepFinger: A Cascade Convolutional Neuron Network Approach to Finger Key Point Detection in Egocentric Vision with Mobile Camera

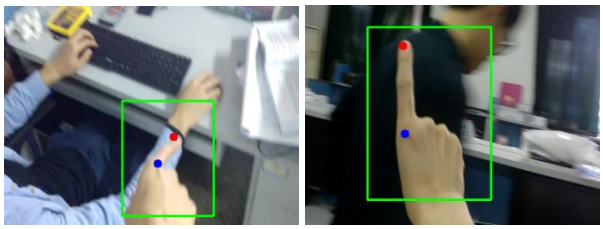
Yichao Huang, Xiaorui Liu, Lianwen Jin, Xin Zhang*
School of Electronic and Information Engineering
South China University of Technology, Guangzhou 510640, China
*Email: eexinzhang@scut.edu.cn

***Abstract*—In this paper, we introduce a new approach to finger key point detection. For RGB images captured from an egocentric vision with a mobile camera, fingertip point detection remains a challenging problem due to various factors, like background complexity, illumination variety, hand shape diversity, and image blur cause by camera movements. To address these issues, we propose a bi-level cascade structure of a convolutional neuron network (CNN). The first-level CNN generates a bounding box of hand region by filtering a large proportion of complicated background information. Using the bounding box area as input, the second-level CNN including an extra branch returns accurate fingertip location with a multi-channel dataset. Our approach is the first attempt of finger key point detection from an egocentric vision with a mobile camera. The proposed method achieves satisfying and significant better results compared to previous fingertip detection methods based on handcraft features.**

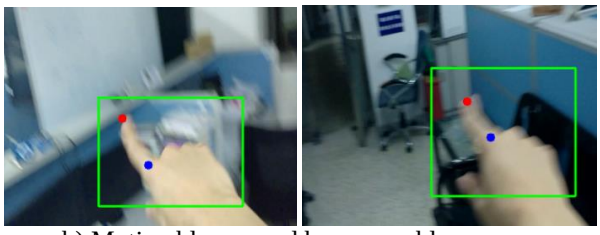
Keywords—cascade convolutional neuron network (CNN); finger key point detection; multi-channel dataset;

I. INTRODUCTION

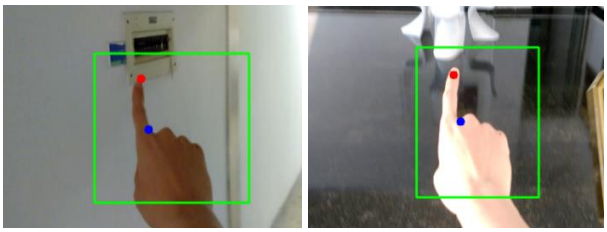
In past decades, human–computer interaction has been primarily implemented through direct contact with peripheral devices, such as keyboards, mouse, and touch screens [1, 2]. However, with the existence of smart eyewear, such as Google Glass [32], Microsoft Hololens [33], conventional patterns of information interaction have demonstrated an inconvenient and poor user experience compared to the touchless mode [3]. Smart glasses require computer vision algorithms that enable the device to understand hand signals. With accurate hand detection, tracking and recognition, computers can produce appropriate responses. Research on the hand modeling has been underway for decades and many methods have been proposed; nevertheless, each method has some limitations. For example, skin-color-based models are sensitive to illumination [4, 6, 10], background models require a fixed camera [12, 13], and RGB-D models require a depth camera and indoor environment [15, 16, 17]. The recent, rapid development of deep learning technology implies a new means of conceptualizing hand modeling. Deep learning has proved to be very effective at computer vision problems [20, 21, 22]; accordingly, numerous works on gesture recognition and classification based on deep learning have been published [18, 19]. Nonetheless, few researchers have explored how to detect fingertips in purely color images, which is a regression problem for deep learning. Therefore, our task is to detect finger key points, including the fingertip and forefinger joint, with RGB images taken by an egocentric vision mobile camera. This approach, however, incurs the challenges of background complexity, illumination variety, hand shape diversity, and motion blur caused by camera movements, as illustrated in Figure 1.



a) Background variety (e.g., another hand, facial skin color).



b) Motion blur caused by a movable camera.



c) Illumination diversity, such as dark and light.

Figure 1. Result of proposed method applied to challenging images.

The objective of this study is to address the above issues and propose a novel approach to finger key point detection in raw RGB real-world images based on a bi-level cascade convolutional neuron network (CNN). The second-level CNN is well designed by expanding the auxiliary branch from the middle of the CNN to combine low- and high-dimensional features. Our dataset was comprised of pre-processed images

with additional channels to emphasize edge features in the learning of our model.

II. Related Work

Before discussing our method in detail, we will review several works in fingertip detection and other key point detection methods that have inspired our work.

A. Fingertip Detection

Fingertip detection has long been a nontrivial task in computer vision. Raheja introduced a method of detecting fingertips using skin filters and image cropping [4]. In addition, Kang developed a technology of detecting fingertips by skin color segmentation as well as contour extraction followed by fingertip analysis [5]. Nevertheless, methods based on a skin color model share the dilemma that the clean background color and stable light are required to enable correct segmentation on account of the sensitivity of the skin color filter. Using RGB images to solve this problem is difficult; however, using RGB-D images is another option [14, 15, 16, 17]. Zhang, for example, proposed an algorithm for detecting fingertips with the help of Microsoft Kinect [14]. Its accuracy is satisfactory, enabling the user to write in the air.

B. Deep Learning in Key Point Detection

To improve state-of-the-art accuracy on key point detection, researchers have begun to apply CNN. Sun and Wang proposed a new approach for estimating facial key points [23]. They introduced a tri-level CNN, which was carefully designed for accurate facial key point detection. Their work improved the accuracy by approximately 50% compared to other methods. Posture estimation using CNN, introduced by Toshev and Szegedy [24], is also impressive for its body joint detection in real-world images. This method includes a cascade DNN and achieves state-of-the-art results in human pose estimation. Gesture recognition by deep learning has been presented as well. Oberweger, for example, proposed a deep learning method for gesture recognition for RGB-D images and developed a 3D model [25].

Our proposed method includes a cascade CNN with a well-designed structure and auxiliary domain knowledge of edge feature. Several challenges exist in the development of this approach. Firstly, to address background variety and the blur problem caused by a movable camera, we designed a cascade CNN. The first-level CNN includes the bounding box region with the hand for filtering the background information. Then, the point detection work is assigned to the second-level CNN. For accurate point regression, we extend an extra branch in our second-level CNN, which provides both shallow features and deep features for the model to learn. In several previous works based on skin color [4, 5, 10], the models could not perform detection well because the skin-color feature dependency caused errors during illumination changes or when skin-color-like objects exist. To reduce the weight of color features learned by our model, we apply auxiliary domain knowledge to the dataset with extra channels that enhance edge features.

III. Cascade Convolutional Network

For our task of precisely detecting fingers with fingertip points and finger joint points, the very first challenge is background variety. Within a 640x480 image, numerous objects exist. Consider Figure 1.a, for instance, which includes papers, a keyboard, a screen, and another person's hand and face. Compared to the background information, our target foreground occupies only a small part, approximately 16% to 30% in the frame. Therefore, we need a cascade structure as described in Figure 2. The first level of the network roughly locates the bounding box of the hand area in the frame. With the first-level network, the model can filter approximately 70% of the background information in the frame. This is not helpful for precise fingertip detection. After decreasing the size, the second-level network continues to strive for an exact result of the finger key point coordinates. In the bounding box, the hand region occupies more than 50% of the pixels, which enables the model to learn features of the hand rather than the irrelevant background. The two networks are separately trained with a specific dataset.

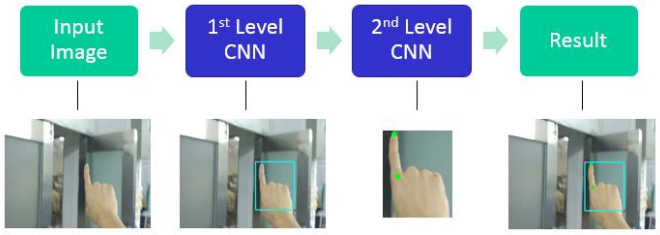


Figure 2. Detection framework.

The outputs of the system are points, which contain x and y coordinates. Therefore, we use the Euclidean loss function.

$$F_{loss}(x_{pi}, y_{pi}) = \frac{1}{n} * \sum_{i=1}^{n} [(x_{pi} - x_{ti})^2 + (y_{pi} - y_{ti})^2] \qquad (1)$$

where x_{pi} denotes the *i*th coordinate of prediction and x_{ti} denotes the ground truth of the *i*th point. All coordinates are normalized by the width and height (e.g. 640 and 480 for first-level CNN), thereby making it possible to keep points in the correct position no matter what resizing action the model takes. For the first-level network, we make *n* equal to 2, which means we obtain two points, the top-left point and the bottom-right point as output. We then use the two outputs to make the bounding box, which becomes the input region of the second-level network. In the second-level network, we make *n* equal to 2, which represents the fingertip and the forefinger joint.

A. Dataset Preprocessing

The dataset is preprocessed before entering the cascade system. By pre-processing, we can emphasize some confirmed prior knowledge, which helps to promote the accuracy of the cascade system.

According to previous research [8, 9], different color spaces produce different performances on skin-color objects, such as faces and hands. For instance, the hue channel in the HSV space shows a significant clustering characteristic with skin color. Through experiments, we found that the HSV color

space is the best fit for learning skin features compared to RGB and YCrCb color spaces.

Previous research has used color filters to segment hand regions [4, 5, 10], yet color features perform poorly when the background is complex, especially when the illumination varies. Our model is intended to focus more on other features, such as edges, shapes, and contours, rather than on color alone. Thus, we add extra edge channels, which are implemented by Laplacian calculation [26] and simple linear enhancement.

$$\Delta f_{i,j} = \frac{\partial^2 f_{i,j}}{\partial x^2} + \frac{\partial^2 f_{i,j}}{\partial y^2} \quad (2)$$

$$\mathrm{d} = \frac{255}{\max(\Delta \mathrm{f})} * \Delta f_{i,j} \quad (3)$$

where $f_{i,j}$ denotes the value of the pixel in the *i*th row and *j*th column, and *d* denotes the destination pixel value.

We consider a four-channel data and a six-channel data; namely, HSVE and HSVEEE. In HSVE, the extra edge channel E is generated from the V channel, whereas in HSVEEE, the extra three edge channels E are separately generated from H, S, and V channels.

B. *Level One: Bounding Box Location*

With the preprocessed training dataset and validation dataset, we train a CNN model for the bounding box location. The bounding box is fixed by two points in the frame: the top left and bottom right. By producing four outputs in order, we form the box and obtain the region of interest containing the hand.

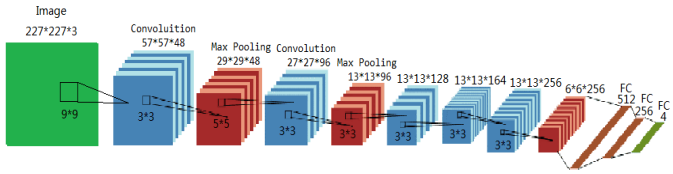


Figure 3. First-level CNN for the bounding box detection.

The network structure, which is shown in Figure 3, is similar to Alex Net [22]; however, in this case, several variables are changed, such as smaller kernel size and output numbers, and the full connect is slimmer. With the revised variables, we achieve better feature representation specifically for our task.

C. *Level Two: Finger Key Point Location*

From the first-level network, an approximately 300x300 region on average is selected, in which the hand is the main object. This region is then sent to the second-level network. The second-level network handles production of an accurate location, which requires a careful design.

By calculating the average of the bounding box width and height, we resize the whole bounding box from varying sizes to 99x99, while labels remain the same because they are normalized. From that point, the bounding box images go through convolutional and pooling layers. The CNN model contains six convolutional layers; for every two convolutional layers, a pooling layer follows. An extra branch of full connections is established from the second pooling layer; then, a concatenated layer follows to combine the two fully connected layers. Figure 4 presents the structure of second-level CNN with multi-channel bounding box input.

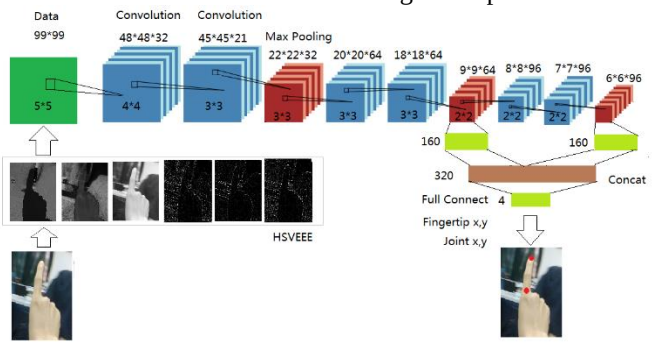


Figure 4. Second-level CNN for finger key point detection.

A deeper CNN may benefit for extracting higher abstract features. However, for a regression problem that requires preciseness, both shallow and deep features are significant. Therefore, we include a branch path from the shallower layer rather than a straightforward structure. In this case, the model can learn from both shallow and deep features. The model learns weights for such features and determines the best suited assemble.

IV. Experiments and Discussions

A. *Dataset and Environment*

We conducted experiments on a dataset with 30,000 training frames and 3,556 validation frames. It should be noted that our training dataset was captured inside our lab, whereas the validation dataset was captured using different scenes, including the interior laboratory, second floor corridor of our building, first floor resting area, and notice board. We believe that the varying scenes in the validation dataset help confirm the generalization ability of our model. The labels we collected for supervised learning included four key points in a frame that describe the hand and finger: the top left and bottom right points of the bounding box, as well as the fingertip and forefinger joint points inside the bounding box. Each point contained two coordinates; therefore, eight normalized labels were collected. The Caffe [27] deep learning framework was applied for its well-organized architecture and outstanding computational ability.

B. *Color Space Selection and Edge Enhancement*

Previous research has shown that the RGB color space is not a suitable space for skin color description. Comparatively, HSV and YCrCb have gained better reputations [9, 10, 14]. We tried different color spaces with the same network. The experimental results shows that, HSV performed slightly better than the other two color spaces with an approximate 15% enhancement.

After we confirmed the color space, we used additional edge channels as input. This improved the performance of the network compared to the conventional three-channel HSV input. The experimental results indicated that a multi-channel input, especially when the edge channels are balanced with

color channels, denoted as HSVEEE, performs better in comparison.

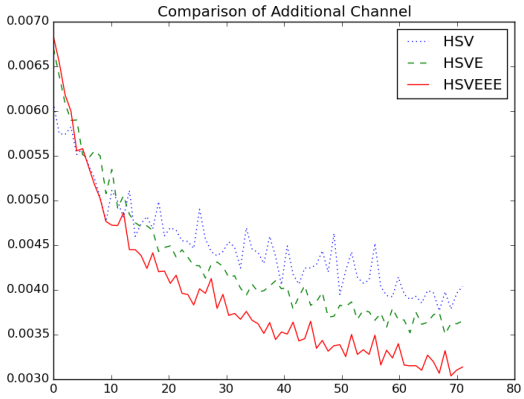
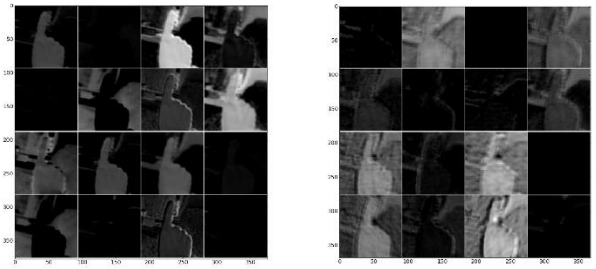


Figure 5. Comparison of HSV, HSVE and HSVEEE with the same network in fine-tuning stage.

Figure 5 shows that considering an auxiliary channel reduces loss with the same network. By producing a feature map, we found that the auxiliary channel helps emphasize the significance of edge features. Figure 6 shows the feature map of the first convolutional layer of our second-level CNN using the HSV and HSVEEE inputs. Without the auxiliary channel, the model primarily learned skin color features, as shown in Figure 6.a, which can lead to errors, such as mixing parts of the background with the hand area. We intend for our model to balance different features; therefore, we added the auxiliary channels of edges. From Figure 6.b we found edge feature enjoy larger proportion than Figure 6.a.



a) Without the auxiliary channel; and b) with the auxiliary channels.

Figure 6. Feature maps learnt by CNN.

C. *First-level Network Performance*

The ground truth of the bounding box on the top-left and bottom-right points has relatively significant noise due to subjective reasons (i.e., it is difficult to discern exactly where the top-left and bottom-right points are); nevertheless, the first-level CNN effectively locates it. This shows the ability to search the area of the hand, no matter what the gesture is. It learns the hand features, including the skin color filter, shape filter, and other features. According to the feature map that we extract, human skin color has the largest weight among all the features because it is special compared to other objects. The kernel and shallow feature maps in Figure 7 show that our first-level network basically learns the skin color and edge, which are relatively reliable in the 640x480 image.

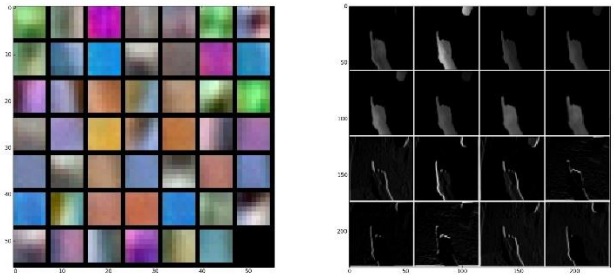


Figure 7. CNN kernels and some feature maps.

Below we define an overlapping rate F_o. It is a formulation similar to the F-measure in information retrieval [31]. This formulation can be regarded as the criteria of accuracy during detection or tracking of the hand region [29, 30].

$$F_o = \frac{(S_p \cap S_t)}{(S_p \cup S_t)} = \frac{2 * TP}{2 * TP + FN + FP} \quad (4)$$

where S_p and S_t represent the predicted bounding box area and ground truth bounding box area, respectively. The numerator represents the overlapping area; the denominator represents the combination area.

When $F_o(S_p) > 0.7$, we consider that the model has produced a correct bounding box. Based on this definition, the performance of different methods are given in Figure 8.

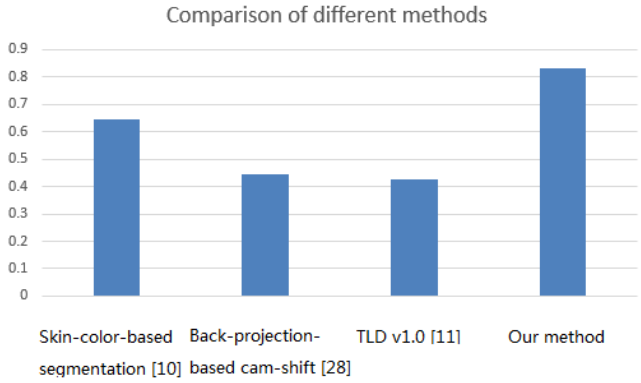
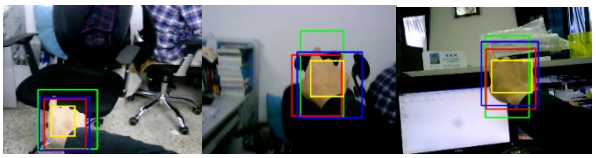


Figure 8. Comparison of different methods.

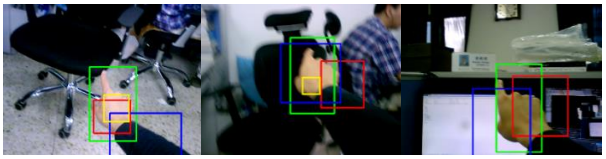
We tested three other methods on the same dataset, and each method has some limitations. As for skin-color-based segmentation model [10] and camshaft algorithm [28], the detector is unstable due to sensitivity of color change or illumination change. Tracking-learning-detection framework (TLD) [11], however, cannot tackle hand pose diversity since it applied sliding window technique in tracking. Moreover, TLD framework is not suitable for connected domain detection due to its feature extraction algorithm.

Representative samples are shown in Figure 9. The results indicate that using CNN for detection produces higher accuracy because the higher dimension features provide greater robustness in the given task, especially when

confronting background variety, illumination diversity, and motion blur.



a) Effective detection of each method.



b) Ineffective detection of color filter method [10] (yellow) and camshaft method [28] or TLD [11] (blue), whereas the proposed method provides effective detection (green).

Figure 9. Representative frames using different detectors.

Compared to hand-crafting feature methods, such as the segmentation algorithm based on the skin color Gaussian model [10], the cam-shift algorithm [28], and the tracking-learning-detection (TLD) framework [11], the first-level CNN of our method performs better for the given task.

D. Second-level Network Performance

In the second-level network, we added an extra branch from the shallower pooling layer to the end of the CNN to combine the low- and high-dimensional features. In some cases, the low-dimensional features can help locate finger points. Comparatively, an architecture with a branch obtains more precision than one without a branch while performing better than a multi-branch structure.

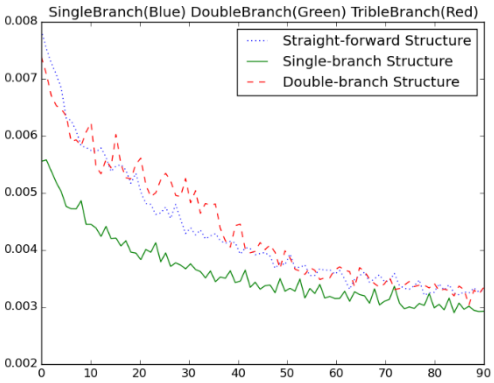
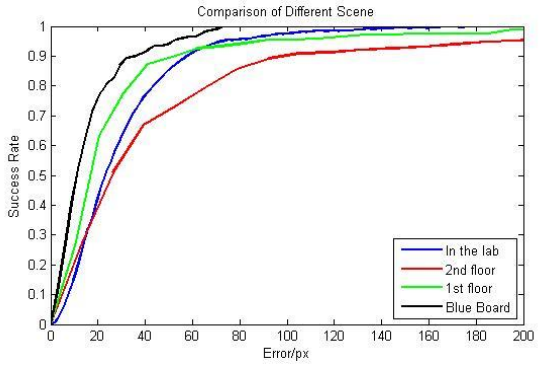


Figure 10. Comparison of the structures in fine-tuning stage.

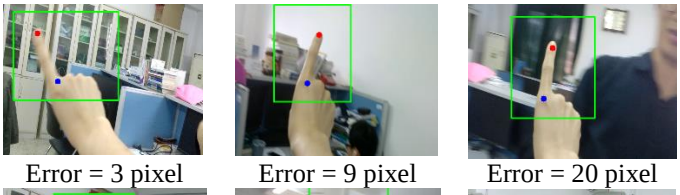
As shown in Figure 10, the network with one extra branch surpassed the other structure in terms of both accuracy and convergence speed.

E. Cascade Network Performance

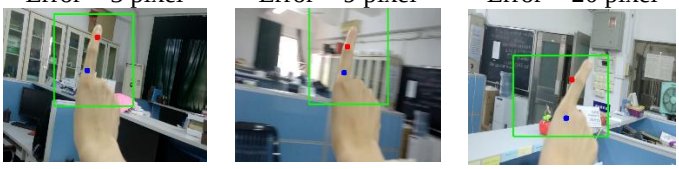
We built the cascade system by combining the first and second networks. We evaluated the system using a validation dataset, which was captured from different scenes, and it produced a satisfactory result. Figure 11.b depicts the instance of different values of pixel errors, while Figure 11.a shows the general result, described by the curve, of the validation sets.



a) Success rate curve under the pixel error threshold.



Error = 3 pixel | Error = 9 pixel | Error = 20 pixel



Error = 40 pixel | Error = 60 pixel | Error = 80 pixel

b) Different pixel error instances

Figure 11. Performance comparison of different scenes

As shown in Figure 11.a, we calculated the errors in different scene and drew the success rate curve under the pixel errors. Even when using all the training images captured in the lab, the model retains good performance in the other scenes, such as the rest area or in front of the blue notice board. We present in Figure 12 a brief period of frames from the validation set and the corresponding detection results.

Figure 12. Illustration of detection results by our method for a period video frame

V. Conclusion

We proposed an effective and efficient approach to finger key point detection from an egocentric perspective. A bi-level cascade convolutional neural network framework was designed with the use of dataset preprocessing for auxiliary edge channels. In the proposed mode, the first-level network provides a rough bounding box that contains the hand region, while the well-designed second-level network includes an extra branch that precisely locates the fingertip and finger joint. We evaluated our method using a diverse validation dataset, and both levels performed well. By employing the cascade system, our method achieved a relatively small error between the prediction and ground truth in several different scenes.

Further research is required on this topic to improve fingertip detection accuracy. In terms of the data source, a highly accurate benchmark of finger key points should be established. With a more precise dataset, the CNN model can be modified to provide better performance. For video sequences, time continuity should be considered and the model should therefore combine detection and tracking.

References

1. A. Dix, Human-computer interaction. Springer US, 2009.
2. J. Preece, et al. Human-computer interaction. Addison-Wesley Longman Ltd., 1994.
3. Z. Lv, L. Feng, S. Feng, and H. Li, "Extending Touch-less Interaction on Vision Based Wearable Device," IEEE Virtual Reality Conference, March 2015.
4. J. L. Raheja, D. Karen, and C. Ankit, "Fingertip detection: a fast method with natural hand." International Journal of Embedded Systems and Computer Engineering, 2011.
5. S. K. Kang, M. Y. Nam, P. K. Rhee, "Color Based Hand and Finger Detection Technology for User Interaction," International Conference on Convergence and Hybrid Information Technology, pp. 229-236, Aug 2008.
6. M. J. Alam, M. Chowdhury, "Detection of Fingertips based on the Combination of Color Information and Circle Detection," IEEE International Conference on Industrial and Information Systems (ICIIS), pp. 572-576, Dec 2013.
7. L. W. Howe, F. Wong, A. Chekima, "Comparison of Hand Segmentation Methodologies for Hand Gesture Recognition," International Symposium on Information Technology (ITSim), vol. 2, pp. 1-7, Aug 2008.
8. X. Zhu, J. Yang, and A. Waibel. "Segmenting hands of arbitrary color," IEEE International Conference on Automatic Face and Gesture Recognition, pp. 446-453, Mar 2000.
9. V. Spruyt, A. Ledda, and S. Geerts. "Real-time multi-colour space hand segmentation," IEEE International conference on Image Processing (ICIP), pp. 3117-3120, Sept 2010.
10. A. Y. Dawod, J. Abdullah, and M. J. Alam, "Adaptive skin color model for hand segmentation," International Conference on Computer Applications and Industrial Electronics (ICCAIE), pp. 486-489, 2010
11. Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-Learning-Detection," IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), vol. 32, no. 2, pp. 1409-1422, 2012.
12. Z. Zivkovic, "Improved Adaptive Gaussian Mixture Model for Background Subtraction," International Conference on Pattern Recognition (ICPR), vol.2, pp. 28 – 31, Aug 2004.
13. O. Barnich, M. Van Droogenbroeck, "Vibe: A universal background subtraction algorithm for video sequence," IEEE Transactions on Image Processing, vol. 20, issue. 6, pp. 1709 – 1724, Dec 2010.
14. X. Zhang, Z. Ye, L. Jin, Z. Feng, "A New Writing Experience: Finger Writing in the Air Using a Kinect Sensor", IEEE on Multimedia, vol. 20, pp.85-93, Nov 2013.
15. G. Rogez, J. S. Supancic III, M. Khademi, J. M. M. Montiel, D. Ramanan, "3D Hand Pose Detection in Egocentric RGB-D Images," arXiv:1412.0065 [cs.CV], Nov 2014
16. J. L. Raheja, A. Chaudhary, K Singal, "Tracking of Fingertips and Centres of Palm using KINECT," IEEE International Conference on Computational Intelligence, Modelling and Simulation, pp. 248-252, Sep, 2011.
17. Liu, Li, and Ling Shao. "Learning discriminative representations from RGB-D video data." International joint conference on Artificial Intelligence. AAAI Press, pp. 1493-1500, 2013.
18. J. Nagi, et al, "Max-pooling convolutional neural networks for vision-based hand gesture recognition," IEEE International Conference on Signal and Image Processing Applications (ICSIPA), pp. 342-347, Nov 2011.
19. H. Lin, M. Hsu, and W. Chen. "Human hand gesture recognition using a convolution neural network," IEEE International Conference on Automation Science and Engineering (CASE), pp. 1038-1043, Aug 2014.
20. Y. Bengio, "Learning deep architectures for AI." Foundations and trends® in Machine Learning, 2009
21. Deng, Jia, et al. "Imagenet: A large-scale hierarchical image database." IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 248-255, June 2009.
22. A. Krizhevsky, S. Ilya, and G. E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems, pp. 1097-1105, 2012.
23. Y. Sun, X. Wang, and X. Tang, "Deep convolutional network cascade for facial point detection," IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3476-3483, June 2013.
24. A. Toshev, and C. Szegedy, "Deeppose: Human pose estimation via deep neural networks." IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1653-1660, June 2014.
25. M. Oberweger, P. Wohlhart, and V. Lepetit. "Hands Deep in Deep Learning for Hand Pose Estimation." In Proceedings of 20th Computer Vision Winter Workshop (CVWW), pp. 21-30, 2015.
26. P. J. Burt, and E. H. Adelson. "The Laplacian pyramid as a compact image code." IEEE Transactions on Communications, vol. 32, pp. 532-540, 1983.
27. Y. Jia, et al, "Caffe: Convolutional architecture for fast feature embedding." ACM International Conference on Multimedia, 2014.
28. C. Chen, M. Zhang, K. Qiu, Z. Pan, "Real-Time Robust Hand Tracking Based on Camshift and Motion Velocity," International Conference on Digital Home (ICDH), pp. 20-24, Nov 2014.
29. S. Hong, T. You, S. Kwak, B. Han, "Online Tracking by Learning Discriminative Saliency Map with Convolutional Neural Network," arXiv: 1502.06796 [cs.CV], 2015.
30. N. Wang, j. Shi, D. Yeung, j. Jia, "Understanding Diagnosing Visual Tracking System," arXiv: 1504.06055v1 [cs.CV], 2015.
31. L. Čehovin, A. Leonardis, M. Kristan, "Visual object tracking performance measures revisited," arXiv: 1502.05803 [cs.CV], 2015.
32. "Google Glass," [Online]. Available: https://www.google.com/glass/start/"
33. "Microsoft Hololens," [Online]. Available: http://www.microsoft.com/microsoft-hololens/en-us