An Efficient Hand Gesture Recognition System Based on Deep CNN

Hung-Yuan Chung
Department of Electrical Engineering
National Central University
Taoyuan, Taiwan

Yao-Liang Chung*
Department of Communications, Navigation and Control Engineering
National Taiwan Ocean University
Keelung, Taiwan

Wei-Feng Tsai
Department of Electrical Engineering
National Central University
Taoyuan, Taiwan

Abstract— **The goal of this paper is to use a webcam to instantly track the region of interest (ROI), namely, the hand region, in the image range and identify hand gestures for home appliance control (in order to create smart homes) or human-computer interaction fields. Firstly, we use skin color detection and morphology to remove unnecessary background information from the image, and then use background subtraction to detect the ROI. Next, to avoid background influences on objects or noise affecting the ROI, we use the kernelized correlation filters (KCF) algorithm to track the detected ROI. The image size of the ROI is then resized to 100x120 and then entered into the deep convolutional neural network (CNN), in order to identify multiple hand gestures. Two deep CNN architectures are developed in this study that are modified from AlexNet and VGGNet, respectively. Then, the above process of tracking and recognition is repeated to achieve an instant effect, and the system's execution continues until the hand leaves the camera range. Finally, the training data set can reach a recognition rate of 99.90%, and the test data set has a recognition rate of 95.61%, which represents the feasibility of the practical application.**

Keywords— Hand detection; hand tracking; KCF; deep CNN; hand gesture recognition

I. INTRODUCTION

The process of hand gesture recognition generally has two parts: detection and recognition, wherein the recognition is also divided into dynamic hand gestures and static hand gestures. The static hand gesture means a fixed hand gesture, and the dynamic hand gesture refers to continuous motion recognition such as waving and grabbing.

First, for detection, the background and the hand are generally segmented using the skin segmentation method, and then the noise processing is performed and the background subtraction method is used to obtain the desired region of interest (ROI), namely, the hand region. In recent years, because of the Kinect [1] depth camera introduced by Microsoft, many depth information-based methods have emerged, such as Keskin et al. [2] and Memo et al. [3], which use the random forest [4], a machine learning method, to train the model to capture the hand's skeletal structure. However, because the price of such a camera is relatively expensive compared to a typical web camera and tends to be affected by the light source of the location, there are still many limitations in terms of application.

As for the identification aspect, its essence is classification. Classification is performed by setting different hand gestures into different categories, using manually set decision criteria (traditionally) or trained classification models (in recent years). The traditional method is to perform the recognition by using the convex hull [5, 6] of the hand after performing skin segmentation. Recognition is determined by the number of polygon edges generated by a hand gesture, and only the numbers from 1 to 5 can be recognized. For example [7], controlling the robot arm with this method can make very little variations, and it is susceptible to complex background interference. Moreover, the hand must be completely facing the lens and this cannot be done on some complicated hand gestures. In recent years, many research teams have adopted machine-learning methods to train models for classification, such as support vector machine [8], hidden markov model [9], convolutional neural network (CNN) [10], recurrent neural network [11] and so on. Among them, CNN is more popular in the field of recognition, and has better results than other methods, mainly because it can get the required feature values from the input picture, and can learn the difference between different samples well by using a large number of samples in its training. However, in the past, its development has been limited due to the speed of hardware computing. In recent years, due to the advancement of semiconductor manufacturing, the computing speed of graphics processing units is getting faster, and the bottleneck of hardware processing speed has been addressed, allowing the CNN network to develop rapidly to become the deep CNN network. The most iconic one is AlexNet [12], an object recognition network, which won the championship at the 2012 ImageNet [13].

However, we found that in the typical hand gesture recognition process (detection plus recognition), if there is an object or noise similar to the skin color in the background, interference can easily occur, resulting in the detection of the wrong ROI, so the background selection has to be restricted to a specific block to avoid disturbance. For example, the background of [14] is limited to a small desktop area (using a webcam facing down to the desktop). Therefore, the study aims to loosen this restriction by focusing on the recognition of static hand gestures, and the ROI of the moving object can be tracked during the system execution and the recognition can be performed immediately. By achieving this goal, the recognition process will no longer be limited to a narrow region. Thus, this study adds a tracking mechanism between the two steps of detection and recognition to avoid problems

* denotes the corresponding author (e-mail: ylchung@email.ntou.edu.tw)

caused by objects in other backgrounds with similar skin color and track hands that may be moving at the same time.

Fig. 1 is a schematic diagram of the proposed overall hand gesture recognition concept, which effectively combines three key components, namely, hand detection, hand tracking, and hand recognition. For the hand detection section, we first perform skin segmentation on input image to remove the extra background information, and then process the noise to reduce the small damage in some images. Finally, we use the background subtraction method [15, 16] to get the ROI of the hand position. In terms of the hand tracking section, we adopt the kernelized correlation filters (KCF) [17] algorithm as the basis for calculation, which has been a widely used algorithm for image tracking in recent years. The core concept is to extract the ROI features of the target position of the first frame and train a model. Then, after the next frame comes in, the trained model will do the calculation to get a new predicted position. As for the hand recognition aspect, we use the deep CNN network to extract and recognize the hand features of the ROI. This study uses two deep CNN architectures for comparison that are modified from AlexNet [12] and VGGNet [18], respectively. After the model training is completed, the recognition rate of the test set can reach more than 95%.

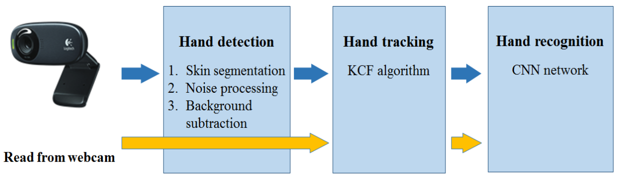

Fig. 1. The schematic diagram of the overall hand gesture recognition concept. The webcam initializes the tracking algorithm after detecting the ROI of the first frame entering the lens, and the ROI block is resized to enter into the deep CNN network for recognition, as shown by the blue arrow-guided path. After that, the tracking algorithm continues to track new incoming frames (i.e., skip hand detection) and recognize them, as shown by the orange arrow-guided path.

The contributions of this study can be summarized as follows.

- We proposed a new hand recognition system that effectively combines the three key components, which are hand detection, hand tracking, and hand recognition.
- With respect to recognition, we designed two deep CNN networks (which were modified from the two classic networks, AlexNet [12] and VGGNet [18]). They are able to achieve sufficient recognition accuracy while reducing the computation load (because we effectively reduce the size of the network) to achieve instant tracking recognition. In particular, the modified version of VGGNet can achieve a recognition rate of 99.9% for the training set and 95.61% for the test set, demonstrating the feasibility of the practical application.
- The proposed hand gesture recognition system has considerable potential to be used in related fields such as controlling home appliances (in order to create smart homes) or human-computer interaction.

The following sections can be organized as follows. Section II describes the hand detection method. Section III describes the hand tracking method. Section IV designs two network architectures for deep CNN-based hand gesture recognition. Section V demonstrates experimental results. Finally, Section VI concludes and points out the potential direction for future research.

II. Hand Detection Method

The hand detection process is shown in Fig. 2. To capture the position of the hand, the first step is to use the skin segmentation method and segment the unwanted background information by using YCbCr [19] (as shown in Fig. 3(b)). Due to its high separation of brightness and chroma, and its simple formula conversion, the execution speed is improved, making it suitable for use in a real-time system. The second step is to process the noise to remove some small noises (as shown in Fig. 3(c)). This includes erosion, expansion processing, and smoothing of morphological image processing. The third step is to use the background subtraction method to obtain the ROI (as shown in Fig. 3(d)). The tracking algorithm is then used to continuously track the ROI (introduced in Section III).

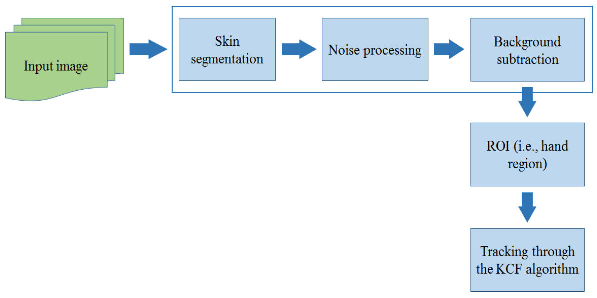

Fig. 2. Hand detection process.

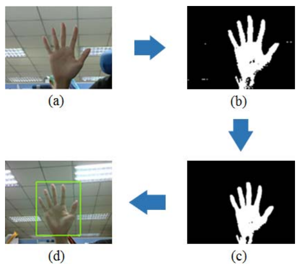

Fig. 3. (a) indicates the input image, (b) indicates that skin segmentation has been performed, (c) indicates the image obtained after the morphological noise reduction method is used, and the ROI (inside the green frame) received after the background subtraction method is used as shown in (d).

III. Hand Tracking

After obtaining the ROI, we use the KCF algorithm [17] to track the detected ROI in order to avoid the noise of a skin color similar to the background or the interference of moving objects with similar skin color. The target to be tracked (i.e., ROI) is used as the positive sample, and the circulant matrix displacement is used to generate multiple samples of the same size as negative samples. They are together used for training the model. After that, the new frame comes in, a correction calculation is performed on the trained model to find the (possibly moving) position of the ROI and achieve the tracking effect.

The KCF algorithm is divided into two stages: training and detection (tracking), as shown in Fig. 4. The first stage is the training stage (as shown in Fig. 4(a)): when the first frame of the ROI detected by the background subtraction method comes in, the ROI is used as the tracking target (positive sample) for training. First, it is used to generate multiple training samples (negative samples); each sample (positive sample plus negative sample) is then used as input for training, after which a Gaussian probability density function (PDF) model can be obtained. The sample obtains higher PDF when it is closer to the tracking target and the PDF is lower when the sample is farther from the tracking target. The second stage is the tracking phase (as shown in Fig. 4(b)): when the new frame comes in, we capture image by the position of the ROI of the previous frame, and the displacement is generated to produce different samples. The new frame and the samples are entered the trained model of Fig. 4(a) and a correction calculation is performed. Next, the position of the maximum value is designated as the updated ROI. After obtaining the new target position, the tracking target image will be taken again and the step shown in Fig. 4(a) will be repeated to train and update the model; the system then wait for the next frame to come and continue to track (repeating the above process). The system stops executing when the hand leaves the camera range

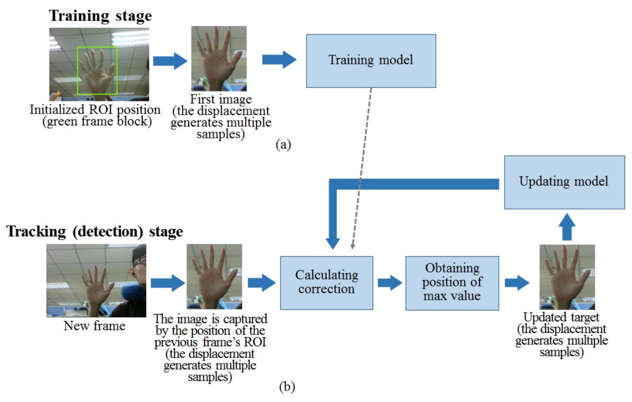
Training stage
Initialized ROI position (green frame block)
First image (the displacement generates multiple samples)
Training model
(a)
Updating model
Tracking (detection) stage
New frame
The image is captured by the position of the previous frame's ROI (the displacement generates multiple samples)
Calculating correction
Obtaining position of max value
Updated target (the displacement generates multiple samples)
(b)

Fig. 4. KCF tracking flow chart, where (a) is the training stage process and (b) is the tracking stage process.

IV. Deep CNN Architecture for Hand Gesture Recognition

We then adjust the size of the ROI to 100x120 and enter it into the deep CNN for hand gesture recognition. This study designed two deep CNN architectures. Architecture 1 is modified based on AlexNet [12], and Architecture 2 is modified based on VGGNet [18]. Both modifications mainly allow for network size reduction and sufficient recognition accuracy to be achieved in an efficient manner.

A. Architecture 1 (version modified from AlexNet)

The modified AlexNet architecture is shown in Fig. 5. Fig. 6 shows its detailed internal parameters.

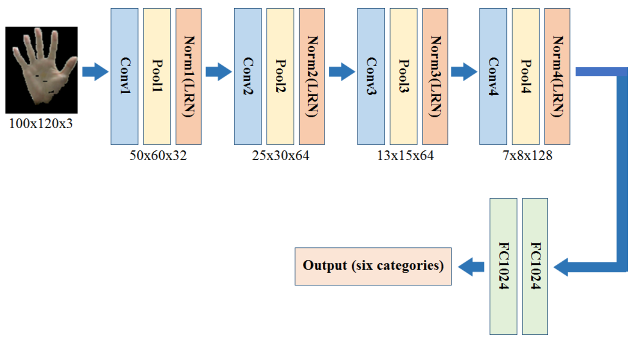

Fig. 5. Architecture 1 (version modified from AlexNet).

Input 100x120x3
Conv1(5x5x32)stride1/ReLU
Maxpool1(2x2)stride2
Norm1(LRN)
Conv2(3x3x64)stride1/ReLU
Maxpool2(2x2)stride2
Norm2(LRN)
Conv3(3x3x64)stride1/ReLU
Maxpool3(2x2)stride2
Norm3(LRN)
Conv4(3x3x128)stride1/ReLU
Maxpool4(2x2)stride2
Norm4(LRN)
Dropout
Full1(1024)
Full2(1024)
Output (6 categories)

Fig. 6. Detailed parameters of Architecture 1 (version modified from AlexNet).

1. Convolutional Layer

This architecture uses four convolutional layers. The number and sizes of convolution kernels for each layer are not the same. Deeper layers use more kernels, allowing the system to capture more features. Specifically, the four layers sequentially use 32-64-64-128 convolution kernels and their sizes are 5x5-3x3-3x3-3x3, respectively. In order to make the output feature map size stay the same, the zero-padding method is adopted, that is, the addition of zeros around the original image, so as to maintain the size of the original image and reduce the impact of the image edge. In addition, each layer of the convolutional layer is immediately followed by a rectified linear unit (ReLU) activation function.

2. Pooling Layer

Down sampling is conducted after the feature map is obtained by the convolution layer to make the size of the sampled image becomes 1/4 of the original one and; that is, the edge length of the image will all become 1/2. To be specific, this study uses max-pooling for sampling, by using 2x2 kernel to take the maximum value of the internal elements of the image, and the stride is 2. Four times of max-pooling are used in total, so the final input to the fully connected layer has a picture size of 7x8. In addition, a layer of local response normalization (LRN) is added on each pooling layer.

3. Fully Connected Layer

We use two fully connected layers. The input parameters are set as 1024 neurons, and finally output has six categories (six common hand gestures), as shown in Fig. 7. To reduce the problem of system over-fitting, we add the dropout method [20, 21] before inputting the fully connected layer. That is, during the training process, the value in the node becomes zero with a certain probability according to demand; the node is then unconditionally discarded and shall not be updated for the training; finally, when the network test is performed, all parameters are aggregated. This method is very helpful in training networks with less training data.

4. Training Method

Before the training, the weight parameters in each layer of the network are randomly set to initialize the network. After the training image is input, the initialized network is used to perform estimations, and the output result is obtained. After the result goes through the softmax function, the error between the predicted result and the value of the real label is calculated using the Cross Entropy loss function. Here, we use one-hot encoded labels to label the output values (set the correct category value to 1, all others to 0), as shown in Fig. 7. For example, if the input sample is in the first category, the actual value of the sample output of the label will be [1,0,0,0,0,0].

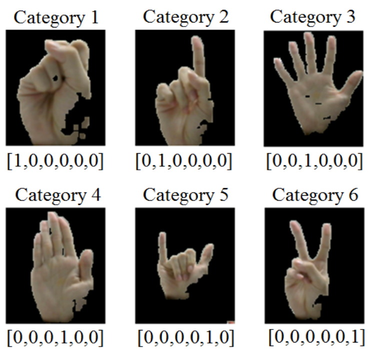

Fig. 7. Labels of six common hand gestures and corresponding hand gestures.

After obtaining the calculated loss function value, we use the adaptive moment estimation (Adam) [22] algorithm as the method for performing the weight value update, which can train the model relatively efficiently.

B. Architecture 2 (version modified from VGGNet)

The modified VGGNet architecture is shown in Fig. 8. Fig. 9 shows its detailed internal parameters.

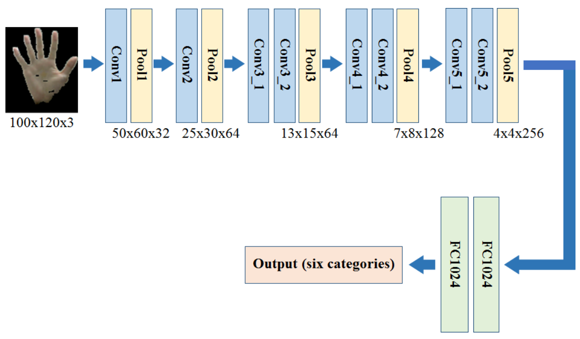

Fig. 8. Architecture 2 (version modified from VGGNet).

Input 100x120x3
Conv1(5x5x32)stride1/ReLU
Maxpool1(2x2)stride2
Conv2(3x3x64)stride1/ReLU
Maxpool2(2x2)stride2
Conv3_1(3x3x64)stride1/ReLU
Conv3_2(3x3x64)stride1/ReLU
Maxpool3(2x2)stride2
Conv4_1(3x3x128)stride1/ReLU
Conv4_2(3x3x128)stride1/ReLU
Maxpool4(2x2)stride2
Conv5_1(3x3x256)stride1/ReLU
Conv5_2(3x3x256)stride1/ReLU
Maxpool5(2x2)stride2
Full1(1024)
Dropout
Full2(1024)
Output (6 categories)

Fig. 9. Detailed parameters of Architecture 2 (version modified from VGGNet).

1. Convolutional Layer

Architecture 2 uses eight convolutional layers in total. Similar to Architecture 1, deeper layers use more kernels, allowing the system to capture more different features. Specifically, the eight layers sequentially use 32-64-64-64-128-128-256-256 convolution kernels and their sizes are 5x5-3x3-3x3-3x3-3x3-3x3-3x3-3x3, respectively. In order to ensure that the output feature map size stays the same, the design is similar to Sec. IV.A.1, in the sense that it adds zero-padding and ReLU layers.

2. Pooling Layer

The max-pooling mechanism is used in the same way as it is used in the case of Sec. IV.A.2, and max-pooling is used five times here, so the image size of the input to the fully connected layer is 4x4. However, there is no LRN layer after each pooling layer.

3. Fully Connected Layer

As with Sec. IV.A.3, a total of two fully connected layers are used here and the input parameters are set as 1024 neurons, and finally output has six categories. However, unlike Sec. IV.A.3, the dropout mechanism here is not placed before the two fully connected layers, but is instead placed on the output of each fully connected layer.

4. Training Method

The training method of Architecture 2 is the same as that of Architecture 1, as described in Sec. IV.A.4.

V. Experimental Results

This section will compare and analyze the results of the two proposed deep CNN architectures. Since the internal parameters and depths of the two architectures are somewhat different, the recognition rate will vary.

In order to fix the size of input images (such that the number of neurons in the fully connected layer can be fixed), we select the detected ROI by a ratio of 4 to 5 (width to height), resize it to 100x120, and then enter it into the CNN for instant tracking and recognition.

A. Training Data

If unnecessary information in the training data can be reduced in advance (preprocessing) when training the network, then the network can be trained faster, which will enable it to achieve better results with a lower level of complexity. Therefore, we use the skin color detection and fuzzifier method to process the original data, and remove most of the background to obtain the training data (as shown in Fig. 7) to train the model.

800 images were collected for each hand gesture (so a total of 4800 training images were used for the training model because there were six hand gestures in total), each of which had different backgrounds and angles (as shown in Fig. 10). In addition, after the skin segmentation process, there may still be some background information that cannot be removed. Thus, this study also used these different backgrounds as training images, so that the model can learn the required features more completely and correctly, and ignore unnecessary information. Finally, we used 300 test images to verify the model.

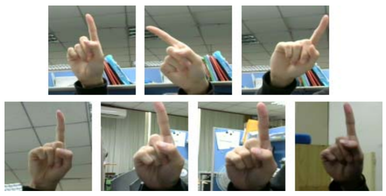
Fig. 10. Examples of original data before preprocessing, including original data from different backgrounds and angles.

B. Network Recognition Results

For Architecture 1, its network parameters and their training results are listed in Table 1 and Table 2, respectively. It is noted that after each image is pre-processed, we normalize the original pixel value 0~255 to -0.5~0.5.

Table 1. Network parameter settings of Architecture 1

Learning rate	10^{-5}
Dropout probability	0.2
Optimizer	Adam optimizer
Batch size	64
Total number of training	975 epochs
Training set	800 x 6 = 4800 images
Test set	300 images
Image preprocessing	Pixel value/255 − 0.5

Table 2. Training results of Architecture 1

Training set recognition rate	99.68%
Test set recognition rate	84.99%

For Architecture 2, its network parameter settings and training results are listed in Table 3 and Table 4, respectively. Referring to [12], in the preprocessing of each image, we subtract the three channels (i.e., RGB) of the image by 103.939, 116.779, and 123.68, respectively, which are the mean values of the RGB channels of each pixel for all the training images in ImageNet database.

Table 3. Network parameter settings for Architecture 2

Learning rate	10^{-3}
Dropout probability	0.5
Optimizer	Adam optimizer
Batch size	64
Total number of training	43 epochs
Training set	800 x 6 = 4800 images
Test set	300 images
Image preprocessing	103.939, 116.779, and 123.68 are subtracted from the three channels (i.e., RGB) of the image, respectively

Table 4. Training results of Architecture 2

Training set recognition rate	99.90%
Test set recognition rate	95.61%

The comparison between Table 2 and Table 4 highlights that the multiple convolutions and deeper networks of Architecture 2 can raise the model's recognition accuracy rate, and allow for the recognition rate of the test set to reach 95.61%. In addition, Table 5 clearly shows that Architecture 2 does not have a large number of parameters in the input of fully connected layer. Instead, its deeper network can get better features without being a computational burden. In conclusion, the proposed hand gesture recognition system should be sufficient to achieve the effect of instant tracking and recognizing hand gestures.

Table 5. Comparison of parameter sizes between the two Architectures

	Architecture 1	Architecture 2
Storage space taken up by network parameters	163975KB	75618KB
Parameter quantity of the last convolutional layer	7168	4096

VI. Conclusions and Future Research

This study successfully combines the traditional image processing method with the tracking method and the deep CNN that has been popular in recent years in hand gesture recognition research, achieving good recognition results given a reasonable computational load. The proposed overall hand gesture recognition system effectively combines three key components, namely, hand detection, hand tracking, and hand recognition. For hand detection, we use skin segmentation, noise processing, and background subtraction to detect the ROI of the first frame entering the webcam. For hand tracking, we use this ROI as the initial position of the tracking, and train the initial model by using the KCF algorithm. After the next frame comes in, the correlation value is calculated to find the position of the maximum value and update the model to achieve a tracking effect. As for hand gesture recognition, we use the two proposed deep CNNs (i.e., modified version of AlexNet and VGGNet, respectively) to extract and recognize the hand features of the ROI. The experimental data show that the training set can achieve a recognition rate of 99.9%, and the test set has a recognition rate of 95.61%. Based on the above observations and reasons, it is believed that the proposed hand gesture recognition system is quite feasible in practical applications, especially in controlling appliances (in order to create smart homes) in the house or human-computer interaction.

In the future, we hope to be able to change the method of capturing ROI from the detection and tracking of images to the use of depth detection networks. However, this will require a considerable large labeling database of the hand region, as well as levels of speed and recognition accuracy that are able to cope with the implementation of instant applications. Another research direction is to examine the use depth detection networks to detect the skeletal structure of the hand and find its position after first inputting the original image, so as to achieve more accurately recognize hand movements. However, the cost and time of implementing training data labeling for the hand's skeletal structure will also pose challenges. We believe these are the areas worth studying, as well as the trends for the future.

ACKNOWLEDGEMENT

The authors wish to thank the financial support of the Ministry of Science and Technology (MOST), R.O.C., under Contract MOST 106-2221-E-019-011-MY2.

REFERENCES

[1] Microsoft: Kinect for Windows. Available online. https://developer.microsoft.com/zh-tw/windows/kinect

[2] C. Keskin, F. Kıraç, Y. E. Kara, and L. Akarun, "Real time hand pose estimation using depth sensors," In *Proc. The 13th IEEE International Conference on Computer Vision Workshops*, Barcelona, Spain, November 6–13, 2011.

[3] A. Memo, L. Minto and P. Zanuttigh, "Exploiting silhouette descriptors and synthetic data for hand gesture recognition," In *Proc. Smart Tools and Apps in Computer Graphics*, Verona, October 15–16, 2015.

[4] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, October 2001.

[5] S. Suzuki and K. Abe, "Topological structural analysis of digitized binary images by border following," *Computer Vision, Graphics, and Image Processing*, vol. 30, no. 1, pp. 32–46, April 1985.

[6] J. Sklansky, "Finding the convex hull of a simple polygon," *Pattern Recognition Letters*, vol. 1, no. 2, pp. 79-83, December 1982.

[7] S. Ganapathyraju, "Hand gesture recognition using convexity hull defects to control an industrial robot," in *Proc. The 3rd International Conference on Instrumentation Control and Automation*, Ungasan, Indonesia, August 28–30, 2013.

[8] T.-N. Nguyen, D.-H. Vo, H.-H. Huynh, and J. Meunier, "Geometry-based static hand gesture recognition using support vector machine," In Proc. *The 13th International Conference on Control Automation Robotics & Vision*, Singapore, December 10–12, 2014.

[9] T. Starner and A. Pentland, "Real-time american sign language recognition from video using hidden markov models," In *Proc. International Symposium on Computer Vision*, FL, USA, November 21–23, 1995.

[10] B. Zhang, C. Quan, and F. Ren, "Study on CNN in the recognition of emotion in audio and images," in Proc. The IEEE/ACIS 15th International Conference on Computer and Information Science, Okayama, Japan, June 26–29, 2016.

[11] T. Koizumi, M. Mori, S. Taniguchi, and M. Maruya, "Recurrent neural networks for phoneme recognition," In *Proc. The Fourth International Conference on Spoken Language Processing*, PA, USA, October 3–6, 1996.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," In *Proc. The 25th International Conference on Neural Information Processing Systems*, Nevada, USA, December 3–6, 2012.

[13] ImageNet Large Scale Visual Recognition Challenge (ILSVRC). Available online. http://www.image-net.org/challenges/LSVRC/", 2010–2017.

[14] M. Han, J. Chen, L. Li, and Y. Chang, "Visual hand gesture recognition with convolution neural network," in *Proc. The 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, Shanghai, China, May 30–June 1, 2016.

[15] P. KaewTraKulPong and R. Bowden, "An improved adaptive background mixture model for realtime tracking with shadow detection," In *Proc. The 2nd European Workshop on Advanced Video Based Surveillance Systems*, 2001.

[16] Z. Zivkovic and F. van der Heijdenb, "Efficient adaptive density estimation per image pixel for the task of background subtraction," *Pattern Recognition Letters*, vol. 27, no. 7, pp. 773–780, May 2006.

[17] J. F. Henriques, R. Caseiro, P. Martins and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 3, pp. 583–596, 2015.

[18] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," In *Proc. The 3rd International Conference on Learning Representations*, San Diego, CA, USA, May 7–9, 2015.

[19] Y. Zhu, C. Huang, and J. Chen, "Face detection method based on multi-feature fusion in YCbCr color space," in *Proc. The 5th International Congress on Image and Signal Processing*, Chongqing, China, October 16–18, 2012.

[20] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[21] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv:1207.0580*, 2012.

[22] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *Proc. The 3rd International Conference on Learning Representations*, San Diego, USA, May 7–9, 2015.