Hindawi
Journal of Robotics
Volume 2021, Article ID 3986497, 13 pages
https://doi.org/10.1155/2021/3986497

Research Article

Hand Gesture Recognition Algorithm Using SVM and HOG Model for Control of Robotic System

Phat Nguyen Huu and Tan Phung Ngoc

School of Electronics and Telecommunications, Hanoi University of Science and Technology, Hanoi, Vietnam

Correspondence should be addressed to Phat Nguyen Huu; phat.nguyenhuu@hust.edu.vn

Received 7 May 2021; Accepted 3 June 2021; Published 17 June 2021

Academic Editor: L. Fortuna

In this study, we propose the gesture recognition algorithm using support vector machines (SVM) and histogram of oriented gradient (HOG). Besides, we also use the CNN model to classify gestures. We approach and select techniques of applying problem controlling for the robotic system. The goal of the algorithm is to detect gestures with real-time processing speed, minimize interference, and reduce the ability to capture unintentional gestures. Static gesture controls are used in this study including on, off, increasing, and decreasing. Besides, it uses motion gestures including turning on the status switch and increasing and decreasing the volume. Results show that the algorithm is up to 99% accuracy with a 70-millisecond execution time per frame that is suitable for industrial applications.

1. Introduction

Today, science and technology develop very quickly making new technologies and ideas easy to apply for the industry to increase productivity and work efficiency. As a result, industrial robots become faster, smarter, and cheaper. More and more companies are beginning to integrate the technology in conjunction with their workforce. This does not mean that robots are replacing humans while it is true that some of the more undesirable jobs are being filled by machines. This trend has several more positive outcomes for the manufacturing industry.

The actions of the robot are directed by a combination of programming software and controls. Typically, industrial robots are preprogrammed to perform repetitive tasks. However, there are still jobs that require human interaction. Human-robot interaction is aimed at controlling robots that perform jobs that humans cannot work directly. Today, the common control systems are mainly screen and keyboard interaction and it is directly on the robot or remote control. However, it will not be convenient and not user-friendly in some cases.

Currently, a new research direction towards the usability of industrial robot control is gesture control. The robot will observe human gestures through sensors mounting on the body or through an image from the camera to perform corresponding actions that have been set up. The basic advantage of the approach is flexibility and speed for the operator that raises safety requirements for users of heavy robots. Image processing today is no longer complicated achieving high-speed equivalent to real-time or even faster since control methods by image analysis are handy for the user and high efficiency.

In this study, we, therefore, propose a gesture recognition algorithm using support vector machines (SVM) and histogram of oriented gradient (HOG) based on the previous work [1]. Besides, we also use the CNN model to classify gestures. The goal of the algorithm is to detect gestures with real-time processing speed, minimize interference, and reduce the ability to capture unintentional gestures. The static gesture controls include on, off, up, and down in this study. Besides, the dynamic gestures in this study include the following:

(i) Toggle state switch is hand from spread state upwards, into grip state

(ii) Up order is hand from outstretched state up to left

(iii) Down order is hand from outstretched state up to right

The rest of the study is presented as follows. In Section 2, we will present related work. In Sections 3 and 4, we present and evaluate the effectiveness of the proposed model, respectively. Finally, we give a conclusion in Section 5.

2. Related Work

The problem of visual hand recognition and tracking is quite challenging. Many approaches used position markers or colored bands to make the problem of hand recognition easier. However, they cannot be considered as a natural interface for the robot control due to their inconvenience. The motion recognition problem can be solved by combining basic image processing problems, namely, object detection, recognition, and tracking. There are many image processing algorithms that have been developed in target detection and recognition. They are divided into two main groups, namely, advanced machine learning (ML) and deep learning (DL) techniques [2–14].

ML techniques are general terms commonly used with basic feature extraction methods from original data and then combining, for example, SVM, decision tree, and nearest-neighbor, to train identity models. There are several extraction techniques for typical object detection as follows:

(i) Viola–Jones target detection technique [2]: it is the first technique in real-time target detection based on Haar feature extraction. This technique is commonly used in face detection.

(ii) Scale-invariant feature transform (SIFT) [3]: the special feature of SIFT is scale-invariant since it gives stable results with different aspect ratios of the image. Besides, the algorithm is rotation-invariant that ensures the result with different rotations of the object.

(iii) HOG [4]: it is calculated on a dense grid of cells and normalized the contrast between blocks to improve accuracy. It is mainly used to describe the shape and appearance of an object of the image.

Advanced DL techniques often use multilayered convolutional neural networks training on labeled datasets. Several techniques are commonly applied in object detection and recognition that include the following:

(i) Region proposals (R-CNN, Fast R-CNN, Faster R-CNN, and cascade R-CNN) [5]: the method proposes areas capable of containing the object and performs identification to save computational capacity.

(ii) Single shot multiBox detector (SSD) [6] such as YOLO and ReneDet: the main idea of SSD comes from using bounding boxes by preinitializing boxes at each location on the image. The SSD will compute and evaluate information at each location to see if there is an object or not. If there is an object on that site, it will determine which one it is. Based on the results of close proximity, SSD will compute an amalgamation box covering the object.

Since the detection and recognition algorithms require a large amount of computation and the accuracy cannot reach 100%, the object tracking techniques in gesture recognition are also widely applied to ensure the continuous real-time recording of subject location and avoid interference in multisubject environments. There are many targeting algorithms for image processing such as BOOSTING [7], MIL, KCF [8], TLD, MEDIANFLOW [9], GOTURN [10], MOSSE [11], and CSRT [12, 13]. Depending on each problem towards the accuracy or processing speed, we can select the right algorithm.

Besides, the problem requires processing image recognition in real time. Using CPU, GPU, and FPGA has their own advantages depending on the specific application for image processing algorithms. Image processing algorithms usually consume a lot of computing resources. In many cases, the continuously growing performance of CPUs is sufficient to handle such tasks within a specified time. However, GPU and FPGA processors are widely used to replace CPUs for image processing applications. Besides, CNN (cellular neural network) technology is an analog parallel computing paradigm defined in space and found by the locality of connections between processing elements (cells or neurons). It has been introduced as a special high-speed parallel neural structure for image processing and recognition [14].

3. Proposed Algorithm

The goal of the algorithm is to detect gestures with real-time processing speed, minimize interference, and reduce the ability to capture unintentional gestures. Their datasets are depicted in Figure 1.

The proposed algorithm will perform as shown in Figure 2. In the model, the image is processed in real time. We will then conduct hand-holding area detection based on the previous model. The system then extracts the region of interest (ROI) of the frame. The object tracking module will get the coordinates of ROI and lock the track object for the next frames. Next, the identity module is activated to evaluate ROI whether the gesture has started or not. As a result, it will decide to continue to perform ROI of the next frames to find the ending and drawing gesture.

Ifthe algorithm did not detect any gestures, we will reinitiate the process. If an operation has been repeated for too long, we will start the program again.

3.1. Overview of Techniques

3.1.1. Detecting Object. The technique has two requirements. The first requirement is to detect the image that contains the object or not. The second requirement is to find out the

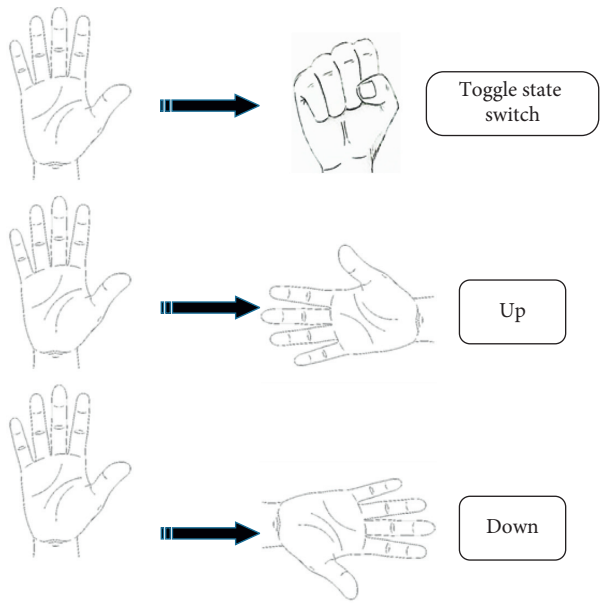


Figure 1: Gesture dataset for controlling.

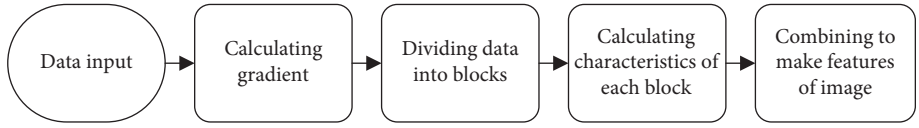


Figure 2: Proposed gesture recognition model.

position of the subject of the image. As introduced in the previous section, there are many algorithms that perform the task. In this study, the requirement is the accuracy of the results as well as being fast enough to operate for real-time applications. In the system, the action is the object detection operation. Therefore, it is necessary to select techniques with relatively fast calculation speed. Therefore, our implementation idea is to use the multiscale and sliding window technique to separate the image into ROI. We then crop the image from those ROI regions and extract their HOG feature. The SVM technique is used to classify whether an image contains an object or not. We then conclude that areas are likely to contain objects. Finally, we use the non-maximum suppression technique to find the most suitable ROI.

3.1.2. Tracking Object. When the algorithm has detected the frame containing the object as well as the ROI area, the next thing is to lock and track the target when it is moving or may be partially deformed in the next frames.

The use of the tracking object will be necessary since the user of gesture will take place in a few seconds. If we continue to use classification and detection techniques to conclude, it will be difficult to achieve the desired processing speed or it may lead to false conclusions. For example, the action initiates ending the gesture that comes from two completely different objects.

During the intermediate stage between starting and ending of gesture, the subject may be partially deformed. It is difficult to use detection techniques. The requirement for object tracking technique in this problem is fast adhesion with acceptable accuracy. In this study, we select kernelized correlation filter (KCF) algorithm to track the object. The algorithm has good speed and consistent accuracy. It will not restore tracking target when losing its target that will reduce noise for the gesture control system.

3.1.3. Object Classification Technique. Object classification techniques are particularly applied for image processing. In artificial intelligence applications, the classifier requirements are to be able to distinguish the gestures that begin with each other. The required accuracy is high and this will ensure the desired level of control accuracy. Processing speed is not required too high since information containing object is known and categorization does not occur continuously in every frame.

When the object (gesture is detected) contains the target, the next task is to recognize them. Once gesture recognition begins, we continue to track the target and end gesture. To perform the task, we choose the convolutional neural network (CNN) model. CNN is used in many problems such as image recognition, video analysis, MRI images, and natural language processing.

3.2. Object Detection and Partitioning Techniques. As analyzed in Section 3.1, we choose HOG characteristic extraction technique, combined with the SVM classification

algorithm for the proposed model in Figure 2. HOG characteristic is proposed by Lee and Chung [4]. The typical HOG idea comes from the object of form and state. It can be characterized by the intensity and direction distribution of pixel value and is represented as a vector calling a gradient vector. Gradient is a vector of elements that represent how fast a pixel of value changes. The gradient vector value brings a lot of useful information. It represents the change in the luminance value of pixels. The gradient vector value changes when the pixel is in the corner and edge areas of the object. Therefore, the HOG feature is effective in choosing the representation of posture.

The essence of the HOG method is to use information about the distribution of intensity gradients or edge directions to describe local objects in an image. The HOG operators are implemented by dividing an image into subregions calling cells. We will compute a histogram of directions of gradients for points for each cell. To combine the histograms together, we get a representation of the original image. To enhance recognition performance, local histograms can be normalized for contrast by calculating an intensity threshold in an area larger than the cell calling blocks. We will use the threshold value to normalize all cells in the block. The result after the normalization step will be a feature vector that is more invariant to changes in lighting conditions. The following are the steps to extract the features of HOG:

(1) Step 1: calculating the gradient vector for each pixel. For a grayscale image, the pixel values are from 0 to 255. If a pixel with neighboring values is left, right, above, and below, the pixel of the gradient vector is represented by different pair.

Let I_x and I_y be the different values of two pairs of left and right, and up and down pixels. The gradient vectors are calculated using the following formula:

$$G = \sqrt{I_x^2 + I_y^2}, \qquad \theta = \arctan\frac{I_x}{I_y}. \tag{1}$$

(2) Step 2: creating blocks. We divide the output image of the previous step into equal blocks. Each block is divided into 4 cells where each cell has an equal number of pixels. The blocks are stacked on top of each other as shown in Figure 3. The number of blocks is calculated using the following formula:

$$n_{\text{block}} = \left(\frac{w_i - w_b \times w_c}{w_c} + 1\right) \times \left(\frac{h_i - h_b \times h_c}{h_c} + 1\right), \tag{2}$$

where w_i, h_i, w_b, h_b, w_c, and h_c are the width and height of image, block, and square, respectively.

(3) Step 3: calculating characteristic vector. We compute the characteristic vector for each cell in block. We then divide the directional space into p bin (the number of typical vector dimensions of a cell). The angle of inclination of coordinate pixel (x, y) is discrete into p bin. We unsigned HOG discretization ($p = 9$) according to the following expression:

$$B(x, y) = \text{round}\left(\frac{p \times \alpha(x, y)}{\pi}\right) \text{mod } p. \tag{3}$$

Unsigning HOG ($p = 18$), we have

$$B(x, y) = \text{round}\left(\frac{p \times \alpha(x, y)}{2\pi}\right) \text{mod } p, \tag{4}$$

where the bin value is determined by the total variable intensity of pixels.

A block consists of 4 cells. Joining four cells, we get the feature vector of a block. The characteristic vector dimension of the block is $4 \times p$ bin with $p = 9$ (unsigning HOG) or 18 (signing HOG).

(4) Step 4: calculating the characteristic vector:

We normalize the feature vector of blocks by dividing by their magnitude. Combining feature vectors of each block to make up the image, we have the HOG feature. The number of characteristic vector dimensions of the image is calculated by

$$\text{size}_{\text{feature image}} = n_{\text{block/image}} * \text{size}_{\text{feature/block}}, \tag{5}$$

where $n_{\text{block/image}}$ is the block and $\text{size}_{\text{feature/block}}$ is the number of characteristic vector dimensions per block.

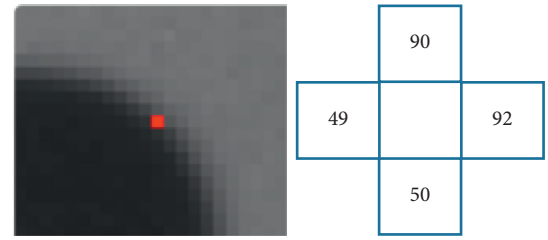


Figure 3: Calculating blocks of gradient vector.

SVM is a machine learning algorithm belonging to the supervised learning group. It is used in classification or regression problems. It is a binary classification algorithm. The SVM takes the input and classifies them into two different classes. SVM training algorithm builds a model to classify them into those two categories.

The idea of SVM is to find a hyperlane to separate data points. This hyperplane will divide the space into different domains and each domain will contain a type of data. For example, we have a dataset of blue and red points placed on the same plane. We can find a line to separate the set of red and blue points as shown in Figure 4 [15].

However, we need more than one straight line to divide it for complex datasets. We use an algorithm to map them to more dimensional space (n dimensions) and find the hyperplane. The example in Figure 5 converts data from two-dimensional space to three-dimensional space [15].

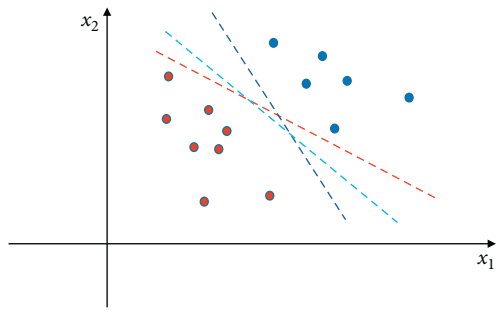


Figure 4: Example of classifying dataset.

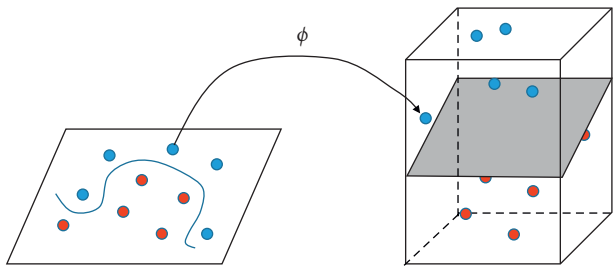


Figure 5: Mapping data from two-dimensional space to three-dimensional space.

There are many superplanes to divide a dataset. However, we need to adhere to the following principles for best optimization:

(i) Firstly, we must definitely be able to divide the dataset.

(ii) Secondly, the rule is that the distance from the nearest point of a certain layer to the superplane must be as large as possible. This distance is known as margin.

Margin is the distance between the superplane to the two nearest data points corresponding to two subclasses. SVM tries to optimize the algorithm by maximizing the margin value. Therefore, we have to find the best superplanar to divide the two data layers.

The problem is to find two boundaries of two data layers that the distance between these two lines is the greatest. The green layer of boundary will pass through one or several green points. The border of the red layer will pass through one or several red points. The blue and red points lying on the two borders are called the vector supports since they are responsible for the superflat finding as shown in Figure 6 [15].

Superplane is represented by the function $W.X_i = b$ and X are vectors; $\langle .W.X_i \rangle$ is the product of two vector scalar. We have to classify the positive (blue) labeled class dataset as 1 and the negative (red) labeled class data as -1.

Superplane separates the two data layers H_0 to satisfy $W.X_i + b = 0$. The hyperplane creates two half-spaces of data as follows:

The space of negative layer data X_i satisfies $W.X_i \leq -1$ and the space of positive layer data X_i satisfies $W.X_i \geq 1$.

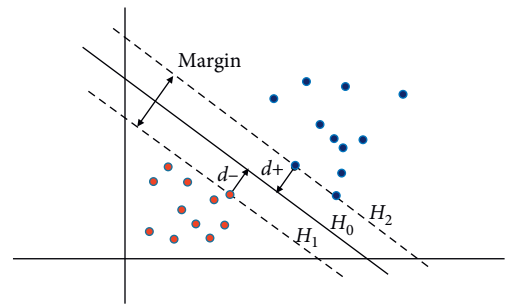


Figure 6: Example of supersets for two-dimensional space.

We next choose two support hyper lane H_1 passing through points of negative layer and H_2 passing points of the positive layer that are parallel to H_0 where the distance from H_1 to H_0 is $d-$, distance from H_2 to H_0 is $d+$, and $m = d_- + d_+$ is the margin level.

The optimal superplanar is the separating hyperplane with the largest margin. The theory of machine learning has shown that a superplane minimizes the limit of error. To calculate the margin m, we have the following:

The distance from a point X_k to the superflat H_0 is $|W.X_k + b|/W$, where W is the length of vector W calculated as

$$W = W.W = \sqrt{w_1^2 + w_2^2 + w_3^2 + \cdots w_n^2}. \tag{6}$$

The distance from a point X_i on H_1 to H_0 is

$$d_- = \frac{|W.X_i + b|}{\|W\|} = \frac{1}{\|W\|}. \tag{7}$$

The distance from a point X_j on H_2 to H_0 is

$$d_- = \frac{|W.X_j + b|}{\|W\|} = \frac{1}{\|W\|}. \tag{8}$$

Therefore, we can calculate the margin as

$$m = d_- + d_+ = \frac{2}{\|W\|}. \tag{9}$$

Therefore, the model training of the SVM technique corresponds to the minimization problem $2/\|W\|$ in condition

$$\begin{aligned} W.X_i + b &\leq -1, \text{if } y_i = -1, \\ W.X_i + b &\geq 1, \quad \text{if } y_i = 1. \end{aligned} \tag{10}$$

This is the condition of hard-margin problem of SVM. The determination of H_0 hyper lanes is assumed under ideal conditions: the dataset can be linearly separated and find two marginal hyper lanes H_1 and H_2 without data points between them. Therefore, if these points do not satisfy the condition, the problem will not find a solution.

3.3. Pyramid Technique. As analyzed above, it is basically possible to classify images containing objects using HOG and SVM. However, the subject is only a small part of the

actual image. If you only categorize the whole image, the results will be inaccurate. Therefore, it is necessary to have an algorithm to determine the position and size of the object with high accuracy.

Multiscale is an image representing many ratios. Using image pyramids allows us to find objects of the image at different scales. We have the original size in terms of width and height based on the pyramid. The image is resized (subsampled) and optionally smoothed (usually through Gaussian blurring) at each subsequent layer. They are subsampled gradually until several stop criteria are met or when the minimum size is reached and subsampling is no longer required.

The second important component is the sliding window. A sliding window is a fixed-sized rectangle that slides from left to right and top to bottom in an image. We will extract the ROI, start the classifier, and get predictive results at each step.

Combining with the image pyramid, the sliding window allows localizing objects in different positions and various proportions of the input image. The results after processing can be multiple in one image. There are multiple outcomes for an object that are at different scale levels at one location or neighboring locations.

The result shows that our classifier is returning the growing probability of the object. However, there is only one object and we need to collapse and delete the excess results. To solve the problem, we apply the nonmaximum suppression (NMS) method that will reduce the overlapping regions.

The idea of the approach is as follows:

(i) We have a set of ROI regions that are called R with corresponding S confidence points and an overlap threshold N. At the same time, initialize an empty list D.

(ii) Select the ROI area with the highest confidence point and remove from R, and add D.

(iii) Compare the newly added ROI region to D in R through the intersection over union index (IoU). If the threshold value is greater than the originally initialized overlap threshold N, then remove those ROI regions from R.

(iv) Continue to select ROI region with the highest confidence point currently available in R and add to D.

(v) Compare the IoU value of the area just added to D with the rest of regions of R; if it is greater than the overlap threshold; then remove from R.

(vi) Continue performing until there are no more elements of R.

(vii) The ending results are the elements of set D.

3.4. Real-Time Object Tracking Technique. Due to real-time applications, modern object trackers try to reconcile as many samples as possible and keep the computation low. Kernelized correlation filter (KCF) is a variant of correlation filter. In the filter, the correlation between two samples is taken. When these samples match, the correlation value is the highest. A correlation can be found between the version of root (ROI area contains the tracked target) and ROI region at the same location in the next frame. This indicates the direction the tracked subject has moved in.

In the standard correlation filter, the following object model is not updated. If the object image changes significantly, the tracker performance decreases. In the KCF tracker, the model of the object being monitored is updated directly and continuously using the linear ridge regression model.

The process of adhering to targets using the basic KCF method consists of the following steps:

(i) Determination of grip area: it can be the initial user-defined area or an area detected by the system from the previous frame

(ii) Description of features: define the characteristics of the image area

(iii) Regression training: the detected ROI features will be added to form a dataset including past and present features as a basis for rapid training

(iv) The results after regression training are a new model, and the model is the basis for the next target detection step

The characteristics of the KCF method are relatively high accuracy, medium speed, and especially inability to recover when losing targets in a short time. Therefore, we choose the method for the paper.

3.5. Proposed Algorithm. The proposed algorithm uses a dataset from live-stream images of the camera for the purpose of testing and solving fundamental problems. Details of the proposed algorithm are as shown in Figure 7.

3.5.1. Preprocessing. Preprocessing the input image before recognition is the step that improves image quality to eliminate noise and increase the ability to recognize the correct type of gesture. During the preprocessing step, we use adaptive histogram equalization techniques and an averaging filter to eliminate noise that enhances the quality of the input image. Besides, we also standardize the image size before processing.

In the preprocessing step, we use the following techniques:

(i) Image resizing: the image is resized to a new size in order to synchronize processing steps, reduce image size, and save the number of calculations.

(ii) Contrast limited adaptive histogram equalization (CLAHE):it is a method to help balance histogram charts with limiting contrast levels. The image is divided into small blocks called cells (8×8). Each of these blocks is then charted as normal. Therefore, the histogram will be limited to one small area. If any cell exceeds a specified contrast level, those

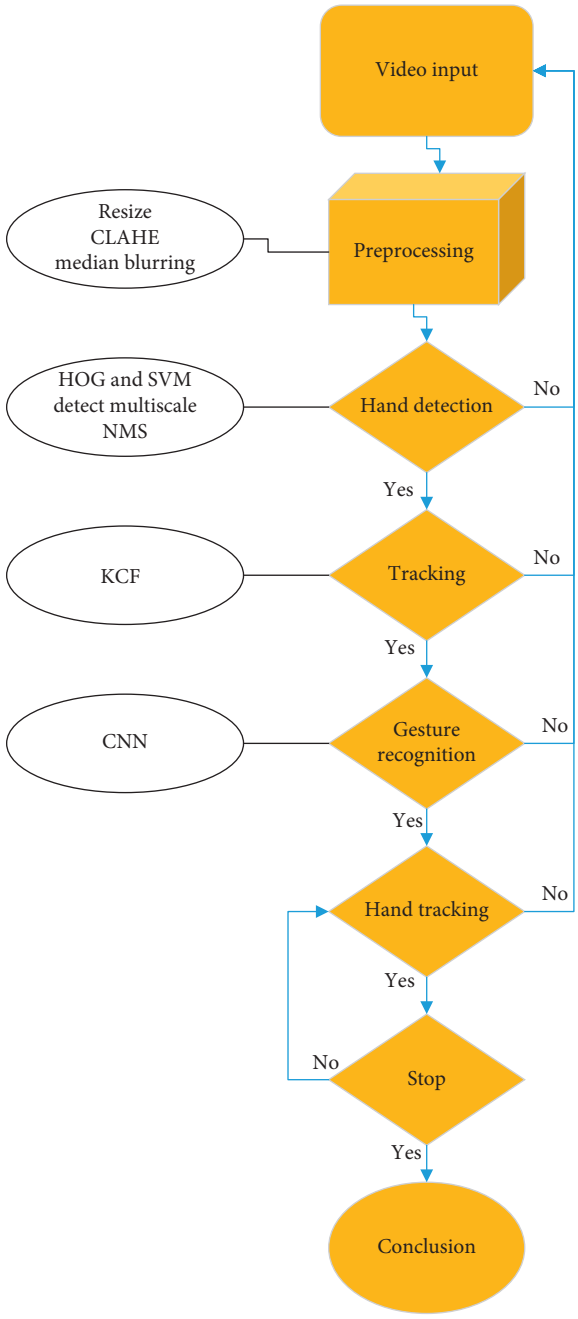


Figure 7: Proposed gesture recognition algorithm.

pixels will be clipped and distributed uniformly to the other cells before applying histogram equalization. To eliminate intercell bias, the linear interpolation balance method is used as shown in Figure 8.

(iii) Median filter: we will take the average of all pixels in kernel area and center element to replace with an average value. This is highly effective against the noise of images. In the filters, the center element with the new value is calculated and effectively reduced noise. Its multiplication area size must always be an odd positive integer. We use a multiplier area of 5×5 pixels. The result is shown in Figure 9.

3.5.2. Building Training Sample. The construction of the training sample will be based on our actual images to bring the most realistic training results for smart indoor application. The training sample is taken from the user of actual images.

(1) Develop a Detection Training Dataset. The detection training sample is images of a human hand at the beginning of motion. Our training sampling method is performed using a python application as the following idea:

(i) Continuously receiving frames from the webcam of the laptop and displaying them on the screen.

(ii) Drawing on the frames as follows: ROI areas are fixed as 190×190 pixels and started from the top left corner of the first frame. We then move from the next frames using the sliding window method with a horizontal of 6 pixels and a vertical of 80 pixels.

(iii) Executing the command to save a frame within a period of about 50 milliseconds, the file name is stored in a text file with coordinates and size of ROI area corresponding to $(x, y, x + 190, y + 190)$.

We run the application and move our hands within ROI on each frame. We get a folder containing image templates and a text file containing hand position information in the respective image. We perform one more time to build the training dataset as shown in Figure 10.

For example, the content of the corresponding ROI zone information file is as follows: "0: (6, 0, 196, 190), 1: (12, 0, 202, 190), 2: (18, 0, 208, 190), 3: (24, 0, 214, 190), 4: (30, 0, 220, 190), 5: (36, 0, 226, 190), 6: (42, 0, 232, 190), 7: (48, 0, 238, 190), 8: (54, 0, 244, 190), 9: (60, 0, 250, 190), 10: (66, 0, 256, 190)."

We then perform a random selection of images and check the location of the file. We then evaluate whether the sample quality is consistent with the ROI or not. We can perform more times in the different backgrounds and the lighting to add the dataset.

(2) Developing Identity Training Dataset. For identifying the training sample, we perform with a similar idea to the detection training pattern. In this case, the output is the ROI region image that cuts out from the frames as shown in Figure 11.

We increase the number of runs and change the execution environment for more diverse background sample results. We perform with each starting and ending position. The results will be four datasets corresponding to 1 starting pose and 3 ending positions. We used the program to build 4 datasets corresponding to the following poses: spread your hand up, left, right, and hold your hand. Each dataset contains 1000 featured images.

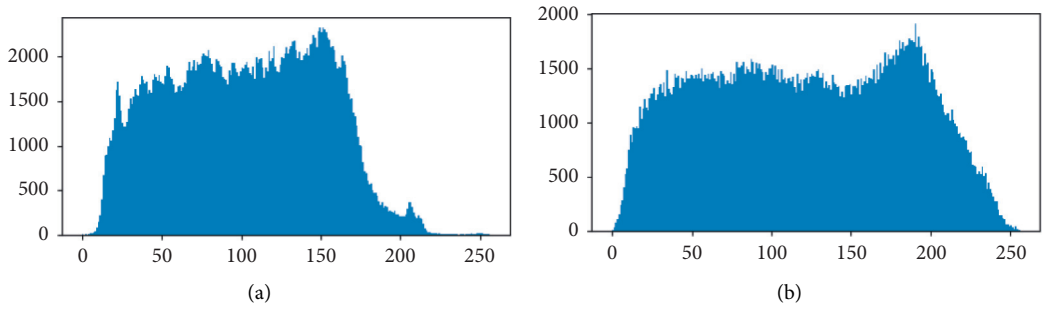


Figure 8: Result of balancing adaptive histogram: (a) original and (b) output image.

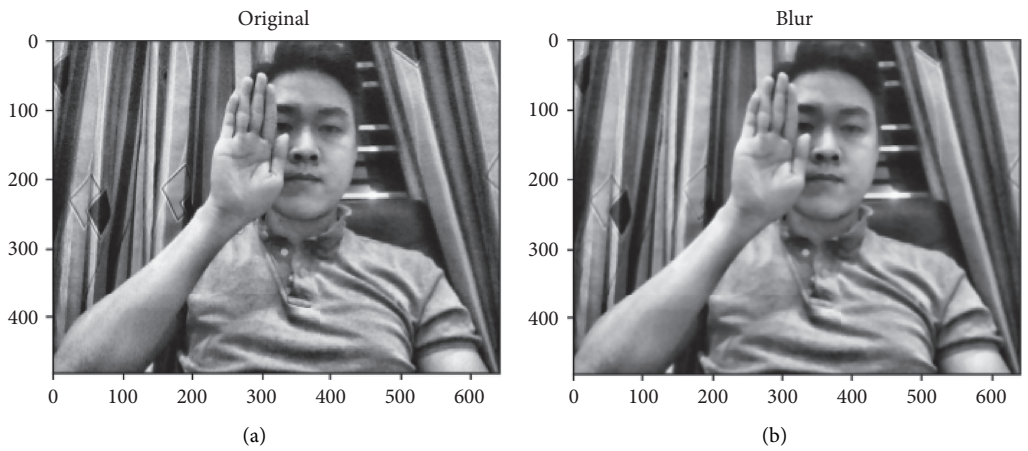


Figure 9: Images before and after filtering median.

Figure 10: Dataset contains hand patterns in different positions.

3.5.3. Training Model to Detect Beginning Posture. To train the detection model, we use HOG and SVM techniques supporting in the dlib library with input data as follows: the sample image is a list element consisting of numpy matrices representing the image. The list of ROI regions contains objects of form "dlib.rectangle" with the condition $C > 0$. As

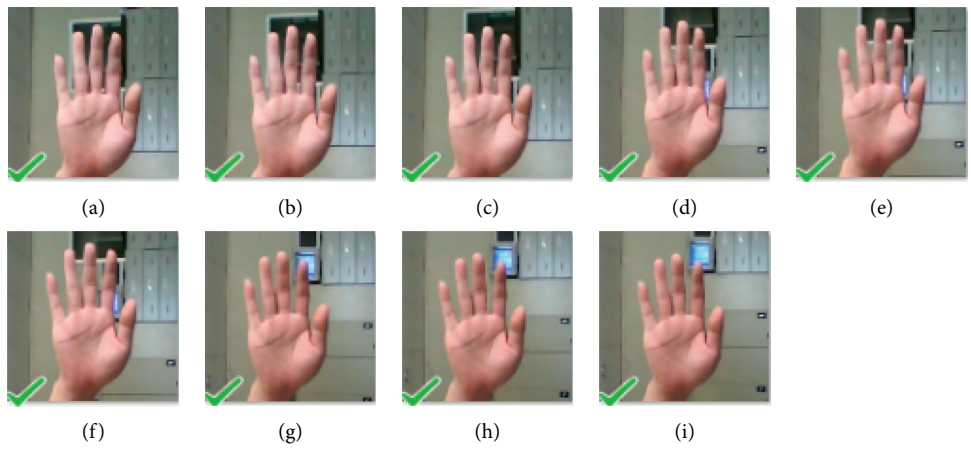


Figure 11: Several hand up datasets for training model. (a) 1239.jpg. (b) 1237.jpg. (c) 1235.jpg. (d) 1219.jpg. (e) 1217.jpg. (f) 1215.jpg. (g) 1199.jpg. (h) 1197.jpg. (i) 1195.jpg.

mentioned above, a small C value is a larger allowable deviation that can lead to underfitting. If C is too large, the tolerance is small that can cause overfitting. Therefore, it is necessary to choose the appropriate C parameter as shown in Figure 12.

We note the sample set list length and ROI area. In our self-build script, we test and choose parameter $C = 2$. Testing the training results with module "dlib.test" (simple_object_detector) produces the following: *training metrics: precision is 1, recall is 0.995825, and average precision is 0.995825.*

Results will be stored as a "∗.svm" file for detection.

3.5.4. Training Gesture Recognition Model. Firstly, we will build a dataset from a previously created set of 4 folders for 4 poses. Data are read from folders one by one and labeled. The results of the dataset are listed in the list with the number of elements equal to the number of sample images. Each element has a structure consisting of a matrix of image descriptions and a label of the sample.

The training uses "tensorflow.keras" to build the model. Through the training process with many types of structures, the model with the number and size of layers is selected as follows:

dense_layers = [0, 1, 2]
layer_sizes = [32, 64, 128]
conv_layers = [1, 2, 3]

We have selected the optimal model with three layers of Conv2D with sizes 32, 64, and 128, respectively. ReLU activation function is three-layer MaxPooling2D and one Flatten layer, respectively. The output is a dense layer of size 128 and a layer of dense of size 4. An input image will have four outputs. Table 1 is a description of a neural network. In Table 1, total parameters are 240,772; trainable parameters are 240,772; and nontrainable parameters are 0.

The results after training with parameter epochs = 300 are shown in Table 2.

In Table 2, the training results show that the model is completed when the val_loss index is very small.

4. Simulation and Result

4.1. Setup. We perform simulation on a computer with Core i5 4310 CPU configuring at 2 GHz without GPU. We evaluated accuracy and execution time for three scenarios including hand zone recognition, static gesture recognition, and motion gesture recognition. In our study, the resolution of the input video is able to change depending on applications. The ROI regions of detecting objects are resized into 64×64 or 128×128.

4.2. Result

4.2.1. Hand Detection Results. We performed reevaluation of target detection results by HOG and SVM using images with many different backgrounds. The results are shown in Table 3 and Figure 13. Figure 13 shows a number of cases where the hand is determined to be faulty due to the background change. We found that image brightness is an important factor to improve the accuracy of the algorithm.

4.2.2. Static Posture Identification Results. According to the CNN model, the output of the classifier will be a 4-element sequence. Each element represents a classifier label and has a value between 0 and 1. When the representative value of the label is close to 1, the result of the classifier is similar to the label. We choose a limit of 0.85. The label will be selected when the corresponding value is greater than 0.85. If there is no label with a corresponding value greater than 0.85, the result will be counted as unrecognizable. If there is a label with the corresponding value greater than 85% but not the correct label identified before the inspection, the result is also counted as false identification.

The results are shown in Table 4 and Figure 14. In Figure 14(a), the actual state is the first one (upward state). However, the results show the third state directing to right

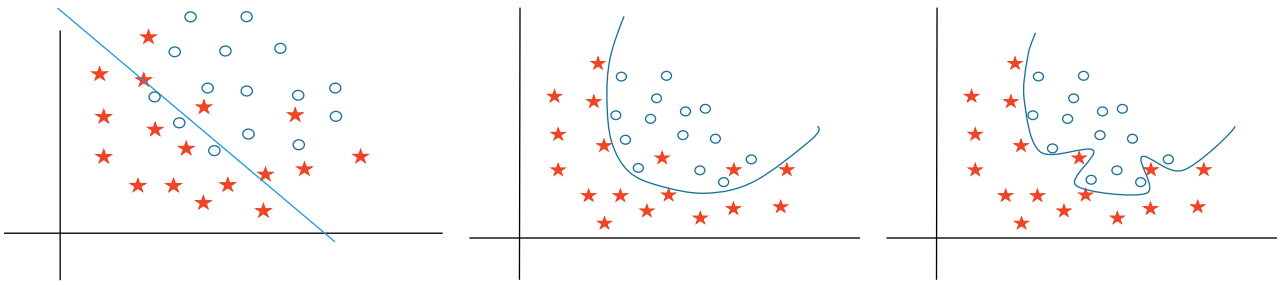


Figure 12: Result of the training process with C parameter based on [16] (a) underfitting, (b) fitting, and (c) overfitting.

Table 1: Sequential parameter model.

Layer	Output shape	Number of parameters
conv2d_1	(None, 126, 126, 32)	320
Activation_1	(None, 126, 126, 32)	0
max_pooling2d_1	(None, 42, 42, 32)	0
conv2d_2	(None, 40, 40, 64)	18496
Activation_2	(None, 40, 40, 64)	0
max_pooling2d_2	(None, 13, 13, 64)	0
conv2d_3	(None, 11, 11, 128)	73856
Activation_3	(None, 11, 11, 128)	0
max_pooling2d_3	(None, 3, 3, 128)	0
Flatten_1	(None, 1152)	0
Dense_1	(None, 128)	147584
Activation_4	(None, 128)	0
Dense_2 (dense)	(None, 4)	516
Activation_5	(None, 4)	0

Table 2: The result of the training model.

Epoch	Time for step (seconds)	Loss	Accuracy	Val_loss	Val_accuracy
1	21	1.2469	0.5143	0.7816	0.7944
2	14	0.0568	0.9881	0.0281	0.9944
...	...	...	...	...	...
298	15	$2.7532e-08$	1.0000	$1.0010e-06$	1.0000
299	15	$2.7248e-08$	1.0000	$9.9832e-07$	1.0000
300	15 ms/step	$2.6680e-08$	1.0000	$9.9170e-07$	1.0000

Table 3: Results of object detection.

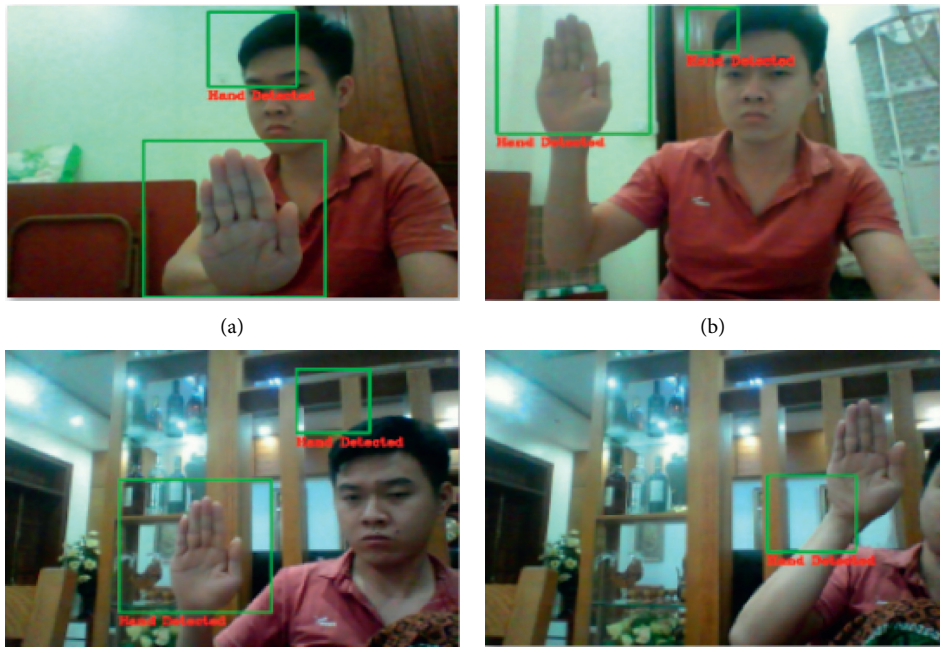
Posture	Number of tests	Number of false identifications	Identification time (milliseconds/images)	Error rate (%)
Spreading arm up	1000	90	63.47	9

that has the highest reliability. In Figure 14(b), the actual state is the fourth state (toggle state). However, the results show the first state (upward) that has the highest reliability.

4.2.3. Dynamic Posture Identification Results. We perform by real-time webcam. As a result, image processing speed has achieved real-time with selecting configuration computer. The results are shown in Table 5.

Due to the limited number of gestures, we have not fully evaluated the effectiveness of the proposed method. However, we can judge by the accuracy of hand position detection steps as well as the step of recognizing the beginning and ending posture. The results in Table 5 indicate that the algorithm improves with accuracy over 86%.

We perform to compare our proposal with other methods. The results are shown in Table 6. In Table 6, we can see that the detection with datasets in different environments still gives special error results when the background changes fast. However, the result is acceptable since the detection can take place continuously at a high speed (0.06 seconds). Therefore, the CNN model has very high accuracy (over



(a) (b)

(c) (d)

Figure 13: Results of detecting hand error for several cases: (a) face, (b) hair, (c) background, and (d) wrist.

Table 4: Result of static posture recognition for 1000 images.

Posture	False identification rate (‰)	Identification time for image (milliseconds)	Accuracy
Holding hands	4	69.41	0.99
Spread posture up	5	72.15	0.99
Spread left posture	11	66.52	0.98
Opening right hand	7	67.32	0.99

(a) (b)

Figure 14: Result of false identification for (a) up and (b) toggle gestures.

Table 5: Result of dynamic posture recognition for 30 videos.

Posture	Number of false identifications	Accuracy
Switch state (on/off)	2	0.93
Increasing	4	0.87
Decreasing	3	0.90

96%) for all postures. This result is suitable for real-time applications.

The proposed method outperforms other methods such as hand gesture recognition and detection using boosted classifiers and active learning [19] with approximately 70% accuracy. Another method also has a relatively high accuracy [20]. However, the method is based on face recognition along with motion detection and based on movement history. However, the method has the disadvantage of being difficult to apply for an environment with a lot of noise relating to colors and gestures.

In the proposed method, we aim to take advantage of the rapid detection advantages of HOG and SVM in coordination with identification using the CNN model. The advantage of the model is highly accurate but requiring relatively strong GPU for real-time applications. Therefore, we have an average speed processing system that still produces acceptable results (90%).

Table 6: Results of comparison with other hand gesture detection systems.

Method	Training data (frame)	Platform	Accuracy (%)	Detection frame (seconds)	Hardware
Combination edge detectio [17]	3154	CPU	82	From 10 to 15	CPU (i5, 2.3 GHz, 16 GB RAM)
HOG characters and SVM [18]	1000	CPU	91	N/A	N/A
Boosted classifiers and active learning [19]	300	CPU	70	0.089	Pentium 4, 3.2 GHz 1 GB RAM
Our proposal	1000	CPU	90	0.15	CPU (i5, 2 GHz, 4 Gb RAM)

4.2.4. Discussion. The authors [21–27] performed for the CNN processor with low hardware configuration for image processing. In [21], real-time requirements for video processing applications are fully satisfied that allows early segmentation to be used and efficient preprocessing technique to perform sophisticated routines for configuration. The results show the feasibility of real-time image processing to support the gesture recognition process of robot control applications. In [26], the authors proposed a novel algorithm for the local binary pattern (LBP) feature extraction using CNN. Using the dynamic parallelism of CNN, the feature can be performed effectively in terms of power consumption and speed.

When using hardware as described in Section 4.1, we find that the computer simulation uses 10 to 15% of CPU for detecting mode and 60 to 70% CPU in classifying mode. RAM is used (including emulation software) less than 510 Mb. It can be seen that the hardware is just enough for processing image. Therefore, we will approach to optimize hardware for image processing in the next step. Besides, the authors [26] show the feasibility of using CNN for basic image processing. In [26, 27], the authors have shown an idea using CNN for deep learning applications. To optimize the system, the requirements for the size of the input image are 128×128 when the minimum processing speed and RAM is 30 frames per second and 512 Mb, respectively. In our study, we use the training algorithm with only 1000 images, a personal computer with low configuration, and an execution time of fewer than 0.15 milliseconds per frame. It is completely applicable to hardware configurations using common CNN chips.

5. Conclusion

In this study, we have built a gesture recognition algorithm based on HOG incorporating SVM that is able to apply to robotic systems. The result shows that the accuracy of the proposed algorithm is improved up to 99%. However, the gesture dataset is not large enough since the effectiveness of the proposed method is not high. Therefore, we are able to improve the accuracy of detection and recognition steps at the beginning and ending gesture.

In the future, we will perform the next steps to increase the frame rate per second, to improve accuracy by increasing the resolution of the input image or using methods in our previous paper [28, 29], and to combine neural networks with other networks to increase the efficiency of calculations and performance with any object.

Data Availability

The authors confirm that the data of this study are built by themselves. Other data that are not theirs are fully referenced in this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was carried out in the framework of the project funded by the Ministry of Education and Training (MOET), Vietnam, under Grant no. B2020-BKA-06. The authors would like to thank the MOET for their financial support.

References

[1] P. N. Huu, T. P. Ngoc, and H. T. Manh, "Proposing gesture recognition algorithm using hog and svm for smart-home applications," *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, Springer International Publishing, vol. 379, pp. 315–323, , New York, NY, USA, 2021.

[2] P. Viola and M. Jones, "Robust real-time face detection," in *Proceedings of the Eighth IEEE International Conference on Computer Vision*, July 2001.

[3] D. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the Seventh IEEE International Conference On Computer Vision*, September 1999.

[4] H.-J. Lee and J.-H. Chung, "Hand gesture recognition using orientation histogram," in *Proceedings of the IEEE. IEEE Region 10 Conference. TENCON 99. "Multimedia Technology for Asia-Pacific Information Infrastructure" (Cat. No.99CH37030)*, September 1999.

[5] R. Girshick, "Fast R-CNN," in *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[6] C. Ning, H. Zhou, Y. Song, and J. Tang, "Inception single shot multibox detector for object detection," in *2017 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*, July 2017.

[7] L. Breiman, "Bias, variance, and arcing classifiers," University of California, Los Angeles, CA, USA, 460, 1996.

[8] L. Dalei, L. Ruitao, and Y. Xiaogang, "Object tracking based on kernel correlation filter and multi-feature fusion," in *Proceedings of the 2019 Chinese Automation Congress (CAC)*, November 2019.

[9] T. Datta, S. Han, M.-J. Kim, V. Maik, and J. Paik, Keypoint-based object tracking using modified median flow," in

Proceedings of the 2016 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia), October 2016.

[10] C. Wang, H. K. Galoogahi, C.-H. Lin, and S. Lucey, "Deep-lk for efficient adaptive object tracking," in *Prceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018.

[11] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui, "Visual object tracking using adaptive correlation filters," in *Proceedings of the 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, June 2010.

[12] F. Feng, X.-J. Wu, and T. Xu, "Object tracking with kernel correlation filters based on mean shift," in *Proceedings of the 2017 International Smart Cities Conference (ISC2)*, September 2017.

[13] J. A. T. Olivero, C. M. B. Anillo, J. P. G. Barrios, E. M. Morales, E. J. Gachancipá, and C. A. Z. d. l. Torre, "Comparing state-of-the-art methods of detection and tracking people on security cameras video," in *Proceedings of the 2019 XXII Symposium On Image, Signal Processing And Artificial Vision (STSIVA)*, April 2019.

[14] P. Arena, M. Bucolo, S. Fazzino, and M. Frasca, "The CNN paradigm: shapes and complexity," *International Journal of Bifurcation and Chaos*, vol. 15, no. 7, pp. 2063–2090, 2005.

[15] S. Kandukuri, A. Klausen, H. V. Khang, and K. Robbersmyr, "Fault diagnostics of wind turbine electric pitch systems using sensor fusion approach," *Journal of Physics: Conference Series*, vol. 1037, no. 3, p. 32036, 2018.

[16] A. Tharwat, "Parameter investigation of support vector machine classifier with kernel functions," *Knowledge and Information Systems*, vol. 61, no. 3, p. 12, 2019.

[17] M. Kounavis, "Fingertip detection without the use of depth data, color information, or large training data sets," in *Proceedings of the 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, October 2017.

[18] K.-P. Feng and F. Yuan, "Static hand gesture recognition based on hog characters and support vector machines," in *Proceedings of the 2013 2nd International Symposium on Instrumentation and Measurement, Sensor Network and Automation (IMSNA)*, December 2013.

[19] H. Francke, J. R.-d. Solar, and R. Verschae, "Real-time hand gesture detection and recognition using boosted classifiers and active learning," in *Advances in Image and Video Technology*, D. Mery and L. Rueda, Eds., Springer, Berlin, Germany, pp. 533–547, 2007.

[20] C.-C. Hsieh, D.-H. Liou, and D. Lee, "A real time hand gesture recognition system using motion history image," in *Proceedings of the 2010 2nd International Conference on Signal Processing Systems*, July 2010.

[21] P. Arena, A. Basile, M. Bucolo, and L. Fortuna, "An object oriented segmentation on analog CNN chip," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 50, no. 7, pp. 837–846, 2003.

[22] P. Kaluzny and S. Kuklinski, "Properties of cellular neural networks in selected image processing applications," in *Proceedings of the IEEE International Workshop On Cellular Neural Networks And Their Applications*, December 1990.

[23] C.-C. Lee, J. P. d. Gyvez, Color image processing in a cellular neural-network environment," *IEEE Transactions on Neural Networks*, vol. 7, no. 5, pp. 1086–1098, 1996.

[24] K. R. Crounse and L. O. Chua, "Methods for image processing and pattern formation in cellular neural networks: a tutorial," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 42, no. 10, pp. 583–601, 1995.

[25] S.-A. Chen, J.-F. Chung, S.-F. Liang, and C.-T. Lin, "Cellular neural network (CNN) circuit design for modeling of early-stage human visual system," in *Proceedings of the IEEE International Workshop On Biomedical Circuits And Systems, 2004*, December 2004.

[26] O. Lahdenoja, M. Laiho, and A. Paasio, "Local binary pattern feature vector extraction with CNN," in *Porceedings of the 2005 9th International Workshop On Cellular Neural Networks And Their Applications*, May 2005.

[27] A. Horváth, M. Hillmer, Q. Lou, X. S. Hu, and M. Niemier, "Cellular neural network friendly convolutional neural networks—CNNs with CNNs," in *Proceedings of the Design, Automation Test In Europe Conference Exhibition (DATE), 2017*, March 2017.

[28] N. H. Phat, T. Q. Vinh, and T. Miyoshi, "Video compression schemes using edge feature on wireless video sensor networks," *Journal of Electrical and Computer Engineering*, vol. 2012, Article ID 421307, 20 pages, 2012.

[29] P. N. Huu, V. Tran-Quang, and T. Miyoshi, "Image compression algorithm considering energy balance on wireless sensor networks," in *Proceedings of the 8th IEEE International Conference on Industrial Informatics (INDIN 2010)*, July 2010.