Hindawi
Security and Communication Networks
Volume 2023, Article ID 9816853, 1 page
https://doi.org/10.1155/2023/9816853

Retraction

Retracted: A Visual Human-Computer Interaction System Based on Hybrid Visual Model

Security and Communication Networks

Received 11 July 2023; Accepted 11 July 2023; Published 12 July 2023

This article has been retracted by Hindawi following an investigation undertaken by the publisher [1]. This investigation has uncovered evidence of one or more of the following indicators of systematic manipulation of the publication process:

(1) Discrepancies in scope
(2) Discrepancies in the description of the research reported
(3) Discrepancies between the availability of data and the research described
(4) Inappropriate citations
(5) Incoherent, meaningless and/or irrelevant content included in the article
(6) Peer-review manipulation

The presence of these indicators undermines our confidence in the integrity of the article's content and we cannot, therefore, vouch for its reliability. Please note that this notice is intended solely to alert readers that the content of this article is unreliable. We have not investigated whether authors were aware of or involved in the systematic manipulation of the publication process.

Wiley and Hindawi regrets that the usual quality checks did not identify these issues before publication and have since put additional measures in place to safeguard research integrity.

We wish to credit our own Research Integrity and Research Publishing teams and anonymous and named external researchers and research integrity experts for contributing to this investigation.

The corresponding author, as the representative of all authors, has been given the opportunity to register their agreement or disagreement to this retraction. We have kept a record of any response received.

References

[1] Z. Ke, Z. Chen, H. Wang, and L. Yin, "A Visual Human-Computer Interaction System Based on Hybrid Visual Model," *Security and Communication Networks*, vol. 2022, Article ID 9562104, 13 pages, 2022.

Hindawi
Security and Communication Networks
Volume 2022, Article ID 9562104, 13 pages
https://doi.org/10.1155/2022/9562104

Research Article

A Visual Human-Computer Interaction System Based on Hybrid Visual Model

Ziyi Ke,[1] Ziqiang Chen,[2] Huanlei Wang,[3] and Liang Yin [1]

[1]*Beijing University of Chemical Technology, Beijing 100029, China*
[2]*FIT of Macau University of Science and Technology Macau SAR, Macau 999078, China*
[3]*Guangdong Medical University, Dongguan 523000, Guangdong, China*

Correspondence should be addressed to Liang Yin; yinliang@mail.buct.edu.cn

Received 13 April 2022; Revised 21 May 2022; Accepted 1 June 2022; Published 30 June 2022

Academic Editor: Mohammad Ayoub Khan

The traditional human-computer interaction is mainly through the mouse, keyboard, remote control, and other peripheral equipment electromagnetic signal transmission. This paper aims to build a visual human-computer interaction system through a series of deep learning and machine vision models, so that people can achieve complete human-computer interaction only through the camera and screen. The established visual human-computer interaction system mainly includes the function modes of three basic peripherals in human-computer interaction: keyboard, mouse (X-Y position indicator), and remote control. The convex hull method was used to switch between these three modes. After issuing the mode command, Gaussian mixture was used to quickly identify the moving human body to narrow the scope of our image processing. Subsequently, finger detection in human body was realized based on the Faster-RCNN-ResNET50-FPN model structure, and realized the function of moving mouse and keyboard through the relationship between different fingers. At the same time, the recognition of human body posture was done by using MediaPipe BlazePose, and the action classification models were established through the Angle between body movements so as to realize the control function of remote control. In order to ensure the real-time performance of the interactive system, according to the characteristics of different data processing processes, CPU and GPU computing power resources are used to cross-process images to ensure the real-time performance. The recognition accuracy of the human-computer interaction system is above 0.9 for the key feature points of human body, and above 0.87 for the recognition accuracy of four kinds of command actions. It is hoped that vision-based human-computer interaction will become a widely used interaction mode in the future.

1. Introduction

With the continuous development of computer technology and artificial intelligence, many new forms of human-computer interaction (HCI) [1] have appeared. The tracking and recognition of human movements based on computer vision is becoming a new generation of human-computer interaction [2]. In practical application scenarios, the realization of human motion recognition and tracking needs real-time work, which puts forward higher requirements for algorithm processing speed, accuracy, and hardware conditions. Heterogeneous computing based on CPU and GPU has become one of the mainstream modes in the field of high-performance computing. Meanwhile, most PC are equipped with GPU, and Nvidia offers a mature set of accelerated computing languages, libraries, and tools (CUDA C/C ++) for accelerated computing [3], which greatly improves the processing speed of complex high-precision models. This makes it possible to preprocess images in real-time operation.

Over the past decade, many researchers have made great efforts to help computers better understand human movement language. In 2014, RCNN was published in CVPR2014 by Girshick and Donahue [4], which applied CNN method to target detection task for the first time. In the same year, Girshick improved this method and proposed the Fast R-CNN [5]. In 2015, Ren and He proposed the Faster R-CNN [6] and proposed the RPN method to generate candidate regions of objects. In this method, the traditional image processing algorithm is no longer needed to generate

candidate regions. In 2017, He and Kioxari proposed the Mask R-CNN [7]. In 2015, four scholars from Microsoft Research proposed convolutional neural network [8]. Ghiasi and Lin proposed FPN algorithm in their paper published by CVPR in 2017 [9], using feature pyramid for target detection. This method is helpful for preserving image features in the process of image processing. Actions input from a regular camera can already be tracked in real time using Google tools (e. g., Google Mediapipe): Zhang and Bazarevsky provide a real-time device hand tracking pipeline for AR/VR applications that can predict hand bones from a single RGB camera [10]. Moreover, the model has been further improved, and the lightweight model can be widely used in various intelligent terminal devices. Caputo et al. optimized the processing models of static gesture language and dynamic gesture language [11]. Ahmadzadeh Esmaeilabadi established a graphics preprocessing method with small computational load [12]. Based on a variety of visual models, there are many visual interaction schemes in engineering technology, but most of them have realized limited functions. For mature HCI system also need to establish a set of complete interaction system. Besides, the switching of different interactive functions also needs to be improved.

In this paper, based on the Faster-RCNN-ResNET50-FPN, MediaPipe BlazePose and other various Machine learning models, we mainly explore how to build a real-time visual interactive system, which can replace the physical peripheral interactive devices such as mouse, keyboard, and remote control to control the computer. The visual interaction system can be used as a supplement to the existing physical interaction, sensor interaction, and customized interaction. It is a cost-effective human-computer interaction scheme and provides a new choice for human-computer interaction.

2. Preliminaries

2.1. System Definition and Segmentation. The visual human-computer interaction scheme proposed by Saada and Mohamed gives us a lot of inspiration [13]. The input data is a sequence of images captured by the camera. Images used as training sets were obtained through cameras in advance, and key points needed to be detected and tracked were defined according to the needs of constructing visual interaction system. Meanwhile, algorithm model was built to track and locate these key points for the following visual interaction system. Visual interaction system is mainly divided into three modes: (1) mouse control, (2) remote control switch, and (3) virtual keyboard input. The switching between the modes is carried out after the corresponding actions are recognized by the machine, that is, the three modes are switched through gesture recognition. First of all, the interactive system will first conduct gesture recognition, and identify different gestures to decide which mode to use. Here, the three patterns are realized by the recognition of the left hand and right hand thumb and index finger, the recognition of the limb, and the recognition of the finger of one hand. After the image is acquired by the camera, the GPU is used to accelerate the image preprocessing, and then the motion is recognized and tracked by the trained model. Finally, corresponding instructions are given to the computer. The entire process is shown in Figure 1:

2.2. Gesture Recognition and Mode Selection. In order to implement the three different forms of interaction, each of them requires a different detection point. By splitting the three models into three parts and enabling only one algorithm at a time, the interactive system can speed up its processing performance by avoiding the need to categorize all checkpoints each time. Different gestures can be used at the start of the process, and three different modes can be launched for different gestures. Since this step is expected to be carried out on the host and only a simple classification of gestures is required, the convex hull method with low accuracy is used to detect the outline of gestures so as to initiate different interaction modes [14]. Convex hull is relative to a set of points, also known as the convex hull of a set of points. For a set of points, if there is a convex polygon that completely contains all the points in the set, then the convex polygon is called the convex hull of the set. The shape of the hand is a simple polygon outline, so here the convex hull can be used to identify the outline of the gesture. Considering that the system only needs to recognize several gestures, it is reasonably making use of the characteristics of different gestures on the tightness of the convex hull to distinguish these gesture contours. In this paper, the tightness of the gesture contour to its convex hull as the gesture convexity is represented by δ, and its value is the area ratio of the gesture contour to the convex hull.

$$\delta = \frac{\text{contour Area}}{\text{hull Area}}, \tag{1}$$

hullArea is the area of convex hull, contourArea is the area of gesture contour, then it can identify three different gesture commands by the ratio of their areas. The following Figure 2 shows a simple definition of four start gestures that will launch different detection programs after recognition of the gesture.

This detection box will always remain in the top right corner of the screen, and only those within this area will be identified using convex hull method. To switch mode, it is only needed to move hands to the corresponding box on the screen and make the corresponding switching gesture. At the same time, convex hull method can also get good performance when running on CPU.

2.3. Image Processing. Background subtraction can be used to quickly lock images of moving objects before input to trained models. Gaussian mixture method is a common method for image preprocessing, which is very effective for modeling background. Gaussian mixture method is used to subtract background from video stream to separate foreground and background. The background is constantly updated from the frame sequence, and a mixed K-Gaussian distribution is used to classify pixels as foreground or background. Here, the intensity of continuous variation is classified as foreground intensity, and the intensity of long periods of absence is classified as background intensity.

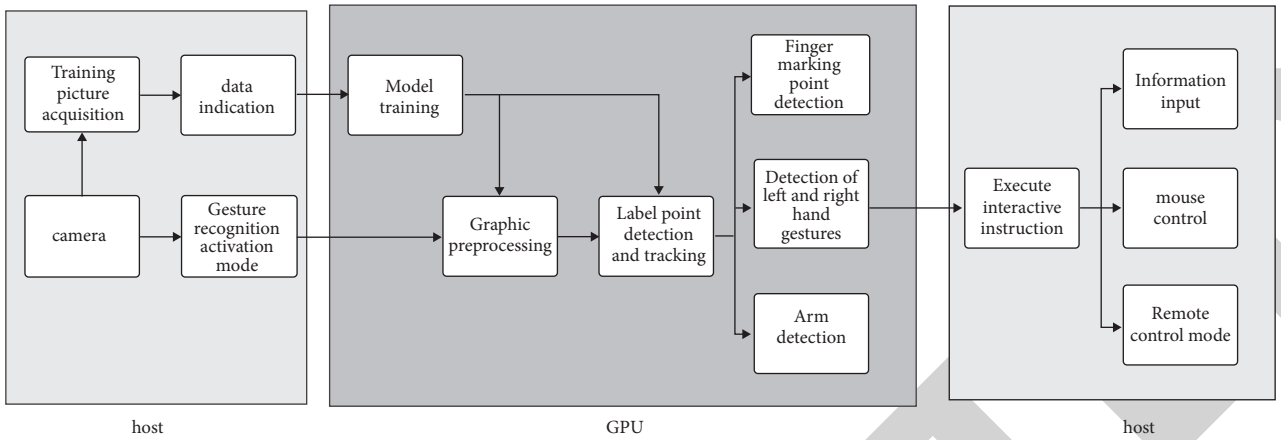


Figure 1: Overall structure.

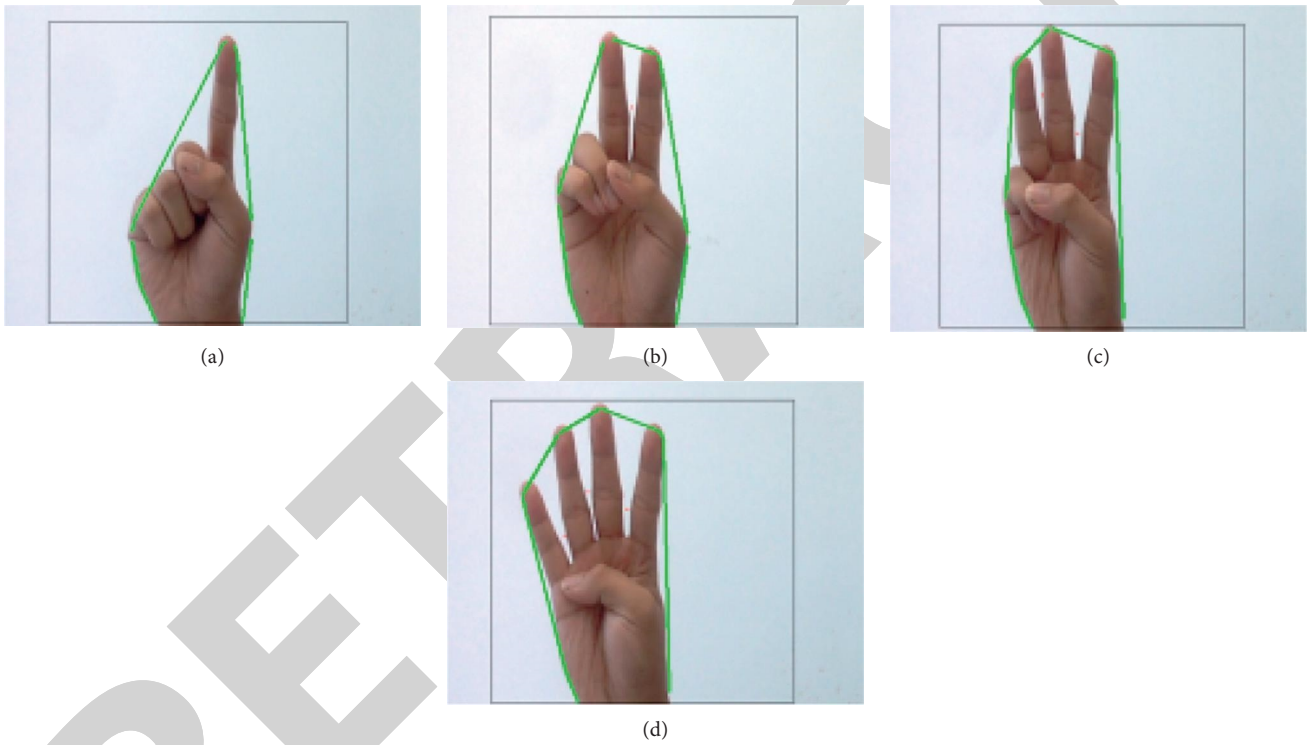


Figure 2: Convex hull method for four kinds of priming recognition gestures. (a) Mouse. (b) Control. (c) Keyboard. (d) Exit.

Then, the boundary between foreground and background is detected, and also the smallest and largest coordinate points.

X and Y in the boundary are found. Mixed Gaussian (MoG) [15] uses K Gaussian components, each with a weight $w_{i,t}$, an intensity mean $u_{i,t}$, and a standard deviation $\sigma_{i,t}$

$$\sum_{i=1}^{k} w_{i,t}\eta\left(u; u_{i,t}, \sigma_{i,t}\right). \tag{2}$$

MoG uses multiple Gaussian components to simulate the background of a pixel. Each Gaussian component is represented by three parameters: mean value (BG gray intensity), weight, and deviation. By comparing the newly input pixel value with the original pixel value, the changing area can be quickly determined to locate the detected object, which can reduce the range of subsequent target detection and tracking and improve the accuracy of the model, as shown in Figure 3.

Different from target detection, Gaussian mixture method is sensitive to moving targets, but insensitive to stationary objects. In an interactive system, the range of movement of human beings in the interaction process is limited, so it is only necessary to determine the position of human beings detected when they move, which can greatly reduce the need for hardware. This kind of image processing does have a disadvantage; relying on the CPU alone to process the image will reduce the input FPS while it can still meet the simple positioning requirements of the interactive

(a) (b) (c)

Figure 3: Primary background elimination, after the video stream is processed, the portrait is extracted first to reduce the workload of the following model. (a) Input. (b) Mixture of Gaussians (MoG). (c) Background subtraction.

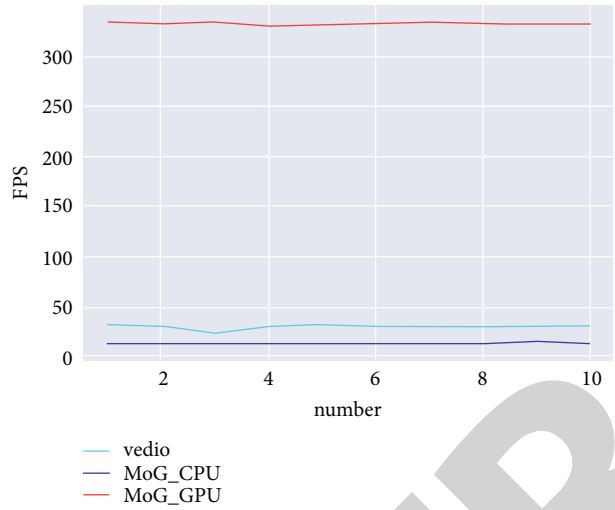


Figure 4: The difference of real-time performance of video stream under the support of the same algorithm and different hardware devices.

system. As shown in Figure 4, if GPU acceleration is used in this process, image processing efficiency can be greatly improved. The following figure shows the statistics of ordinary CPU video stream, FPS after MoG processing under CPU and GPU acceleration.

3. Interaction Detection

3.1. Gesture Control Mouse. In the visual interaction system, we hope that the mouse on the screen can follow the movement of the finger of the right hand, and at the same time can adjust the continuous changes of volume, size, and zoom by the distance between the index finger and thumb of the left hand. Not surprisingly, it all has to do with how to detect the thumb and index finger positions on both hands in the video. Here, the Faster-RCNN-ResNet50-FPN algorithm is used to detect the locations of these key points. The training data set mainly consists of the pictures obtained by the camera, and the four finger tips are marked for the training of the model.

The main part is mainly composed of Transform, RESNET-50, FPN, RPN, and ROI, as we could see in Figure 5. Due to the different sizes of background images obtained during image pretreatment, it is easier to use neural network for processing

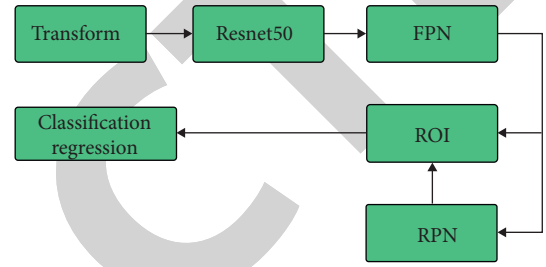


Figure 5: Overall structure.

Table 1: Architecture of ResNet50.

Layer name	Output size	50-layer
conv1	112 × 112 56 × 56	7 × 7,64, stride 2 3 × 3 max pool, stride 2
conv2_x	56 × 56	$\begin{bmatrix} 1\times1, & 64 \\ 3\times3, & 64 \\ 1\times1, & 256 \end{bmatrix}\times3$
conv3_x	28 × 28	$\begin{bmatrix} 1\times1, & 128 \\ 3\times3, & 128 \\ 1\times1, & 512 \end{bmatrix}\times4$
conv4_x	14 × 14	$\begin{bmatrix} 1\times1, & 256 \\ 3\times3, & 256 \\ 1\times1, & 1024 \end{bmatrix}\times6$
conv5_x	7 × 7	$\begin{bmatrix} 1\times1, & 512 \\ 3\times3, & 512 \\ 1\times1, & 2048 \end{bmatrix}\times3$

after image pretreatment. The first step is to transform the image and coordinate through a partial Transform. Then, it was sent to resNET-50 for feature extraction. Next, it is fed to FPN, which will build a feature pyramid and provide input feature map to RPN. RPN will generate the region proposals. Finally, object region, labels of each region, and scores of each region are detected through ROI.

The input from Transform is a list of images, and the output is the transformed image tensor. There are three main steps. First, the image is normalized. The resulting tensor values are now distributed near 0, which is easier for neural network processing. The input image is scaled based on length or width, resulting in a scaled image. Interpolate in PyTorch is used to interpolate an image tensor batch_-images. The ResNet50-FPN is used of which its source code is provided in torch vision, which is shown in Table 1.

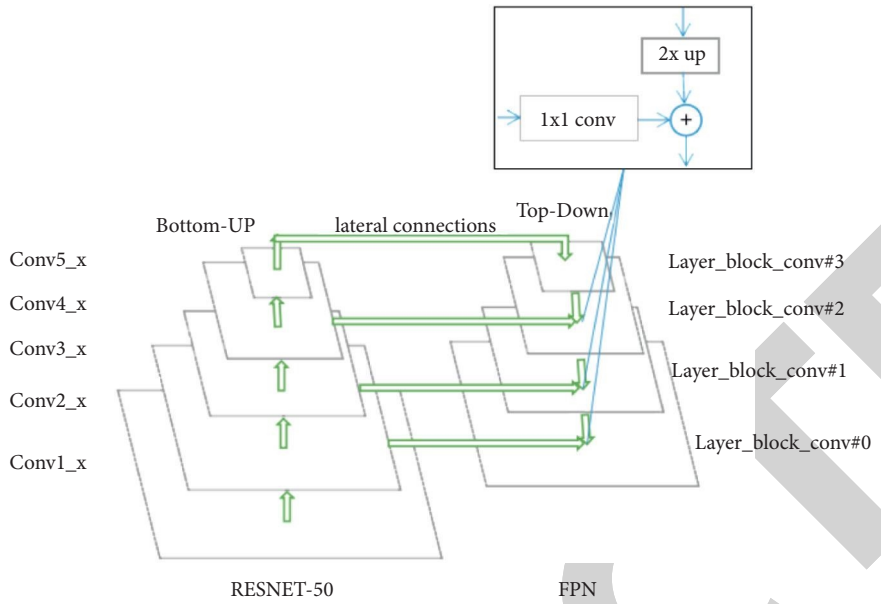


Figure 6: The Resne-50 is used in combination with FPN action diagrams.

If only Faster RCNN is used for target detection, ROI will apply only to the last layer. This has no problem in the detection of large targets, but if it is used in the detection of small targets like fingers, when the convolutional pooling reaches the last layer, the semantic information is actually gone. Therefore, in order to solve the problem of multi-scale detection, the feature pyramid network (FPN) is introduced [16]. FPN is designed to utilize the pyramid form of CNN level features and generate feature pyramids with strong semantic information at all scales [17]. Therefore, top-down structure and horizontal connection are designed in FPN to integrate shallow layer with high resolution and deep layer with rich semantic information. The model is divided into three parts: the bottom-up pathway on the left and the top-down pathway on the right. ResNet50 is exported from each layer on the left to the pathway on the right, which is called lateral connections. Feedforward calculation of CNN is the bottom-up path. For ResNets, it is mainly the feature activation output of the last residual structure at each stage. The top-down approach is to sample the upper-level feature map with stronger semantics, and then connect the features with lateral connections to the features of the previous layer, so that the upper-level features are strengthened. The connection details are displayed in the rectangular box. The process of connection is to double upsampling (nearest upsampling method) for the high-level features, and then convolve it with the corresponding previous layer by 1*1, and then combine the features of the two by adding between pixels. The process is repeated until the finest feature map is produced. In general, FPN can be expressed by the following formula:

$$\mathrm{FPN} = \mathrm{Top} - \mathrm{downpathway} + \mathrm{laterconnection}. \tag{3}$$

Finally, the model uses a 3*3 convolution kernel to process the fused feature images to generate the final required feature images. These outputs will be fed into the RPN network for background foreground dichotomy and bound-box regression, as shown in Figure 6.

RPN (Region Proposal Network) is mainly used to generate regional Proposal, a multi-scale Anchor based on the introduction of the Network model, with SoftMax to sort anchors (background or foreground), And Bounding Box Regression is used to make Regression prediction for anchor to obtain the precise position of the Proposal, which is used for subsequent target identification and detection.

RPN determines whether an Anchor belongs to the foreground or background by calculating the IoU of Anchor and tag. IoU refers to the proportion of the common parts of two boxes to all parts (the overlap ratio). When IoU is greater than a certain value, the truth value of this Anchor is foreground; when IoU is lower than a certain value, the truth value of this Anchor is background. In terms of the true value of the offset, assuming that the central coordinates of Anchor are x_a and y_a, the width and height are w_a and h_a, respectively, the central coordinates of label are x and y, and the width and height are w and h, respectively, the corresponding offset truth calculation formula is obtained

$$\begin{aligned} t_x &= \frac{(x - x_a)}{\mathrm{wa}}, \\ t_y &= \frac{(y - y_a)}{\mathrm{ha}}, \\ t_\omega &= \log\left(\frac{w}{\mathrm{wa}}\right), \\ t_h &= \log\left(\frac{h}{\mathrm{ha}}\right). \end{aligned} \tag{4}$$

The position offset t_h and t_y are normalized by width and height, while the width and height offset t_h and t_w are logarithmic, which further limits the range of offset and

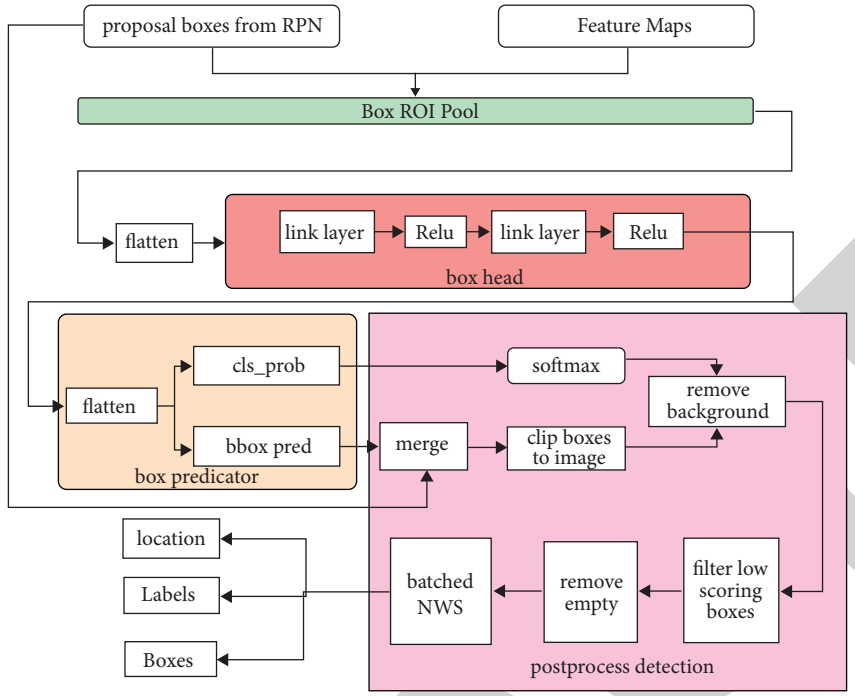


Figure 7: The flow chart of ROI.

facilitates prediction. Given the above truth values, RPN obtains the predicted values of category and offset, respectively, through the convolutional network in order to calculate the loss. To be specific, RPN needs to predict the probability that each Anchor belongs to foreground and background, as well as the offset of real objects relative to Anchor. In addition, after the predicted offset is obtained, the predicted offset can be applied to the corresponding Anchor to obtain the actual position of the prediction box. If there is no Anchor, it is necessary to directly predict the coordinate of each frame for object detection. Since the coordinate of the frame varies greatly, it is difficult for the network to converge and make accurate prediction. Anchor is equivalent to providing a prior ladder, so that the model can predict the offset of Anchor and better approach the real object. The loss function of RPN includes classification and regression:

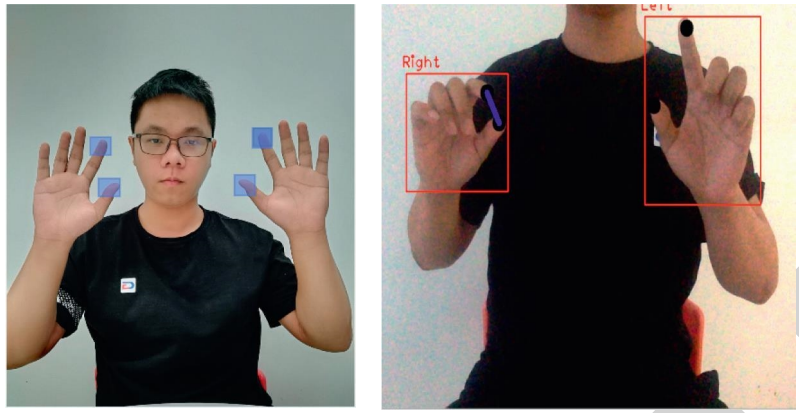
$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{\text{cls}}}\sum_i L_{\text{cls}}(p_i,p_i^*) + \lambda\frac{1}{N_{\text{reg}}}\sum_i p_i^* L_{\text{reg}}(t_i,t_i^*), \quad \sum_i L_{\text{cls}}(p_i,p_i^*), \tag{5}$$

represents the loss of classification for 256 filtered Anchors, with P_i being the category truth value for each Anchor and P^* being the predicted category for each Anchor. Since the role of RPN is to select the Proposal, it does not require the segmentation of which kind of prospect, so it is dichotomous at this stage, and cross entropy loss is used. $\sum_i p_i^* L_{\text{reg}}(t_i,t_i^*)$ represents a return to loss. Regression loss uses the smooth_{L1} function, and the specific formula is

$$\mathbb{L}_{\text{reg}}(t_i,t^*) = \sum i \in x\cdot y\cdot w\cdot h\,\text{smooth}_{L_1}(t_i - t_1^*),$$
$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2, & \text{if } |x|<1, \\ |x| - 0.5, & \text{otherwise.} \end{cases} \tag{6}$$

As can be seen from the second formula of the formula, smooth_{L1} function combines the first-order loss function with the second-order loss function. The reason is that when there is a large gap between the predicted offset and the truth value, the derivative of the second-order function is too large and the model is easy to diverge rather than converge. Therefore, the first-order loss function with a smaller derivative is adopted when it is greater than 1. Finally, the convolutional layer processes the feature maps of each layer and splices them, and the sequence after splicing is arranged. The ROI steps are mainly as shown in Figure 7: ROI will input Proposal Boxes selected from RPN and multi-layer feature map output from FPN into ROI Pool. Box ROI Pool determines which layer feature map to select for ROI Pool operation according to the area of Proposal Box. It is then fed into the Box Head, which consists of two fully connected layers. Next, the results obtained from Box Head processing are further classified and the Box offset is used for numerical regression. Softmax was applied to the classification results, and the final classification was carried out. Then, the position offset regression results of Box were combined with Proposal boxes to obtain the adjusted Detection boxes. Finally, maximum value suppression (NMS) was applied to filter out valid results.

Box ROI Pool can process multiple images at the same time and detect the objects in each image, respectively. Therefore, first of all, convert to ROi format is required to merge the feature images of each layer of each image



Fingertip detection (a) virtual mouse (b)

FIGURE 8: After recognizing and tracking the finger position, the mathematical relationship between the coordinate systems of four objects is used to simulate the mouse.

together for unified ROI Align processing. Setup_scales configure mapper objects for the first four output feature maps (the smallest pool layer is not used). Mapper is a new concept introduced in FPN. It mainly calculates the area of the Proposal Box and calculates which feature layer to perform ROI Align according to the area. It can be expressed as the following formula:

$$k = k_0 + \log_2\left(\frac{\sqrt{w_h}}{224}\right). \tag{7}$$

The image below shows the effect of the model. At the same time, the coordinates of the obtained boxes are calculated to calculate their midpoint. With the midpoint as the center of the circle, a small circle is drawn, which is proportional to the coordinate movement of the cursor in the computer.

The physical mouse provides four types of physical input to the host: right click, left click, scroll wheel, and move cursor.

(1) Move cursor—the detection point of the detected finger could be matched with the coordinate point of the mouse on the screen, and the mouse on the screen will follow the index finger by Autopy, which is a simple cross-platform GUI automation kit that can be used to control the keyboard and mouse. In this process, the detected interfinger coordinates in the image are converted into corresponding mouse coordinates in a certain proportion, and then the transformed coordinates are sent to Autopy to move the mouse on the screen.

(2) Mouse right-click—from the detected coordinate points, the distance between the two coordinate points can be calculated using the distance formula (8). When the index finger touches the thumb, the distance becomes very small. When the distance is less than a certain value, the right mouse click command will be sent to the host.

$$\text{distance} = \sqrt{(y_2 - y_1)^2 + (x_2 - x_1)^2}. \tag{8}$$

(3) Mouse left-click—the same principle, if the index finger and middle finger contact, the computer will be transmitted to click the left button instruction.

(4) Scroll wheel—the scroll wheel is essentially a continuous change, which can be equaled by the distance between the left index finger and thumb. The change in the distance between the two fingers is proportional to the change in the mouse wheel. The visual effect is shown in Figure 8.

3.2. Gestures to Keyboard. Based on the above model, this step becomes quite simple. We modelled the virtual keyboard form used by Hsuan et al. in the usability study of multiple vibratory tactile feedback stimuli [18] and combined pynput and visual methods to build a virtual keyboard on the screen (thirty characters and a space bar are displayed on the screen in a loop). When the two conditions are met, the system inputs the corresponding characters into the device. The first is that the index finger and thumb touch each other, and the second is that the contact point should be inside a printed rectangular letter box on the screen. In this way, visual interaction systems can use their fingers to select letters to type when facing the screen. As shown in Figure 9, it is the virtual keyboard generated on the screen.

3.3. Gestures to Remote Control. In this paper, MediaPipe BlazePose lite was used to fulfill the achievement of the real-time body pose tracking and through action recognition, action information is transformed into our interactive command information. MediaPipe BlazePose [19]. This approach provides human pose tracking by employing machine learning (ML) to infer 33, 2D landmarks of a body from a single frame. BlazePose pinpoints more of the key

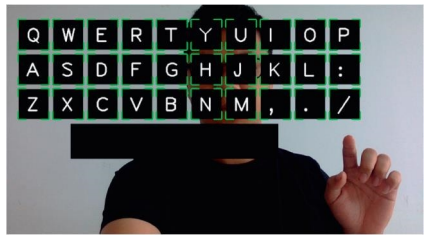


Figure 9: The virtual keyboard. Creating a virtual keyboard on the screen, character input can be realized by recognizing the distance between keyboard characters and fingertip.

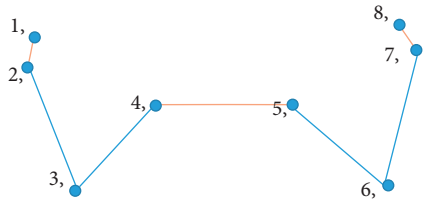


Figure 10: Limb detection point. According to the needs of body movements, eight self-defined human body key points need to be detected and tracked. (a) Left hand heart. (b) Left wrist. (c) Left arm joint. (d) Left shoulder joint. (e) Right shoulder joint. (f) Right arm joint. (g) Right wrist. (h) Right hand heart.

points, and its sibling model, BlazePose lite, can achieve over 40FPS performance on CPU. For body language recognition, there are not many detection points available. Since most human body language is expressed by a series of movements of hands and arms, we use eight detection points for this visual interaction system, as shown in Figure 10.

Considering the differences between the left-handed and right-handed people in different populations, the research only focus on the right-handed people for the next four instructions. In the remote control mode, the visual interaction system needs to give instructions to the computer by detecting the gesture of the detected object. Trying to identify four body movements, they are arm up, down, left, and right four reverse smooth. Based on the previous method of detecting key points of limbs, the coordinate positions of key points of palm, wrist, arm, and shoulder can be obtained. At the same time, pose detection model and time series model can be used to detect the movement recognition of arm waving in different directions [20], but this is not conducive to the construction of real-time interaction. In study, an interesting phenomenon was found that when the arm swung in different directions, the angles of the two joints changed significantly from the normal posture. So, you can do multiple categories and identify these four movements by collecting data from these angles. Through the coordinate points returned by the network model, every three close coordinate points can determine an Angle (angle is shown in Figure 11). For instance, the network sends back three coordinate points (x_1,y_1) (x_2,y_2) (x_3,y_3). So, this Angle can be calculated by this formula:

$$\theta = \arccos \cdot \left[\frac{(x_1 - x_2)(x_3 - x_2) + (y_1 - y_2)(y_3 - y_2)}{\sqrt{(y_1 - y_2)^2 + (x_1 - x_2)^2}\sqrt{(y_3 - y_2)^2 + (x_3 - x_2)^2}} \right]. \tag{9}$$

The model used for training here mainly repeats these four actions in the camera, thus collecting 4892 pairs of Angle data for the training of the model. The collected data is drawn into a scatter graph, and the Angle between the two limbs is called the horizontal axis and the vertical axis of the image, respectively. As can be seen from the scatter plot, the points corresponding to these actions are obvious, as shown in Figure 12 and Table 2, which means that the data set can be easily distinguished by multiple classification models.

There are a variety of algorithm models for classification. Data classification could be done by using various models, so as to find out the model with the best performance for our action classification. Then, some classical classification algorithms will be used to make the comparison.

(1) k-Nearest Neighbors (kNN) (different distances)

(2) Decision Tree (Entropy/Gini)

(3) Random Forest (number of trees, Gini, Entropy, (no) bootstrapping)

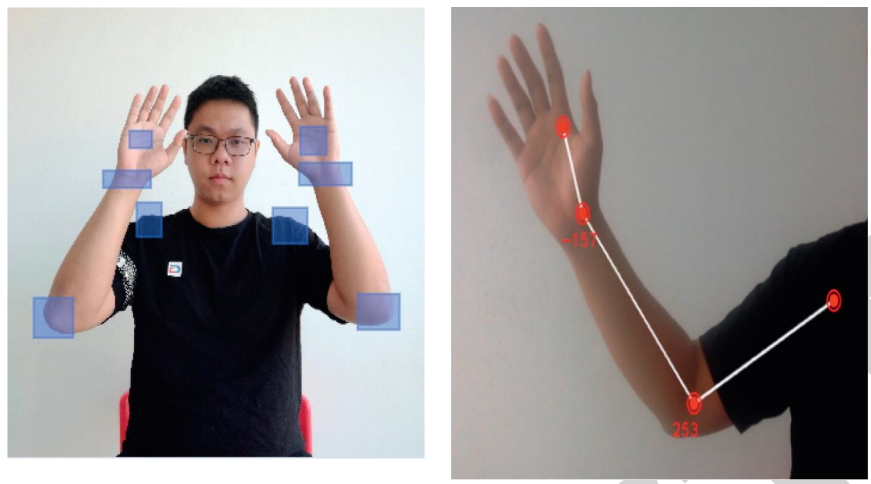
(4) Gaussian kernel SVM (RBF)

(5) Deep neural network (Sigmoid activation/ReLu activation)

Most of these algorithms can be found in Python's Sklearn library and are classic classification algorithms. For K-nearest Neighbors (kNN), the model behaves differently if different distance methods are used. In the experiment, we, respectively, used Euclidean K-NN, Manhattan K-NN, and Minkowski K-NN KNNS with different distances to classify the data points. At the same time, both Gini impurity criterion and entropy criterion will be used to measure the quality of the split. The Gini impurity measures the frequency at which any element of the data set will be mislabelled when it is randomly labeled, whereas entropy is a measure of information that indicates the disorder of the features with the target. If using the character P for probability, they can be written in the following two formulas

$$\begin{aligned} \text{GiniIndex} &= 1 - \sum_j p_j^2, \\ \text{Entropy} &= -\sum_j p_j \cdot \log_2 \cdot p_j. \end{aligned} \tag{10}$$

The third model is random forest. We mainly try to find the model with the highest accuracy by using different numbers of trees, the methods (Gini and Entropy) for splitting and (no) bootstrapping. We cyclically tested the accuracy of the model from 1 to 1000 trees, and structurally the accuracy did not change much after the number of trees



(a) (b)

Figure 11: The key points are detected and tracked, and two angle relations are constructed through the coordinate relations of the key points, which are used for gesture recognition. (a) Detection of key points of limbs. (b) Angle model between joints.

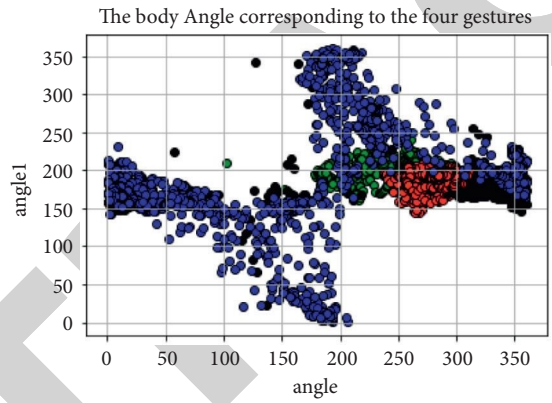


Figure 12: The body angle corresponding to four gestures. By repeatedly collecting the data of two angles of four gestures, it can judge the types of gestures by the characteristics of the two angles.

Table 2: The angles calculated by the detected limb nodes.

	East_angle	East_angle1	West_angle	West_angle1	Up_angle	Up_angle1	Down_angle	Down_angle1
1	233.9112101	160.7384397	160.7384397	161.9888326	270.539793	194.3570452	181.8763717	335.3924992
2	236.2409878	208.7810938	154.2881748	207.3077962	271.0865778	194.7662916	184.3999738	332.1019876
3	236.0207256	208.9682472	160.7877393	203.1866738	270.9976778	194.3265265	190.5816355	305.7505843
4	237.8808253	206.2495742	157.5754429	215.0381846	271.0748416	194.6171873	190.2804779	310.7521513
5	237.7074799	207.1708902	154.0873884	173.8235978	271.7457149	192.471614	180.6669049	347.2925441
6	237.3341667	211.6287803	159.1671883	167.6419484	270.6189166	191.7456334	175.832698	8.508703316
7	237.8064669	210.6465166	160.5405952	165.734688	270.2126508	191.7095657	171.3416346	16.75895842
8	238.909603	211.3171845	160.0909223	166.8243379	270.0260721	190.7648494	169.7960263	20.63558056
9	238.7817063	209.7652164	159.0786308	158.8888677	268.4592591	190.7307791	173.015128	9.625248156
10	238.1619349	209.926914	159.0057672	162.6954323	268.2550024	191.0791822	183.5702534	2.922074496
11	239.1276187	211.9197267	160.573348	164.5675398	268.8947866	190.5398551	183.9392989	21.01658488
12	240.8562331	215.9951808	174.9425694	160.4717909	267.7364216	189.767331	180.0274624	27.25325567
13	239.5769707	214.8217346	168.2938777	168.6934751	267.4030335	188.8413938	173.70782	41.3129428
14	236.7658171	213.5535815	169.8761187	167.9412875	267.5801093	188.5749343	147.0888028	70.18818811
15	235.2610628	213.4668425	183.4526659	162.0393303	268.3741338	188.1267289	110.3386268	107.7058291
...	...	...	...	...	...	...	...	...
1221	255.2871037	211.9451889	27.74305317	164.8374219	283.2114361	194.7066666	203.2408752	230.8560136
1222	256.4778067	209.71394	29.52417006	165.2423816	283.287503	194.0362435	203.6928564	226.910986
1223	256.7575089	209.8267042	27.91685417	165.337185	281.8335229	196.3093648	204.9666287	245.5905584

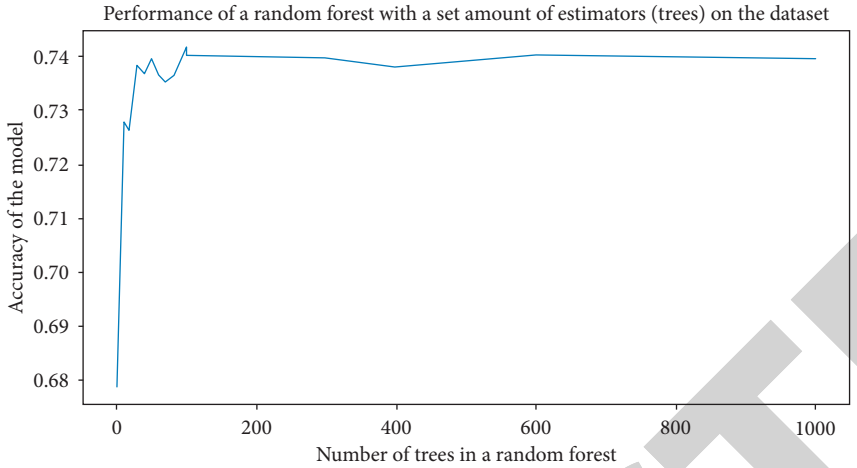


Figure 13: The influence of the number of trees in the random forest on the accuracy of the model is counted so as to select the random forest model with the highest accuracy.

Table 3: The accuracy of different classification algorithms for Angle data processing.

Algorithm model	Euclidean k-NN	Manhattan k-NN	Minkowski k-NN	Entropy decision tree
Accuracy	0.84473	0.84555	0.8478	0.82692
Algorithm model	Gini decision tree	Entropy random forest	Gini random forest	Bootstrapping random forest
Accuracy	0.82523	0.73876	0.85271	0.73808
Algorithm model	No bootstrapping random forest	Gaussian kernel SVM (RBF)	Sigmoid NN	ReLu NN
Accuracy	0.72656	0.8754	0.8584	0.85207

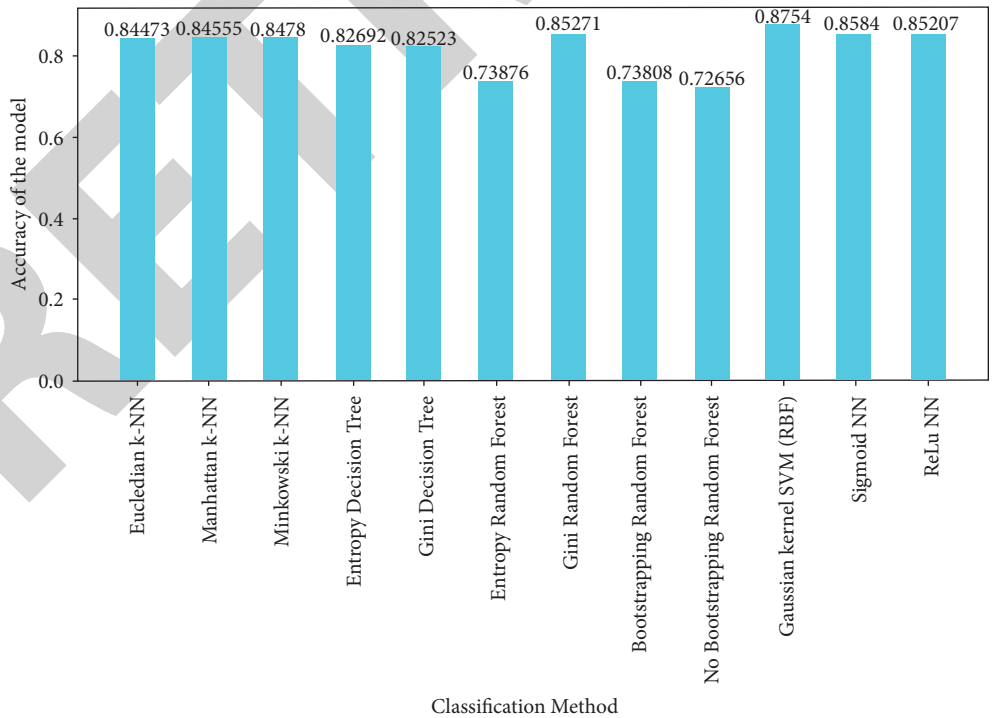


Figure 14: Multiple classification models are used for multi-classification processing of the two angles collected as well as comparing their accuracy.

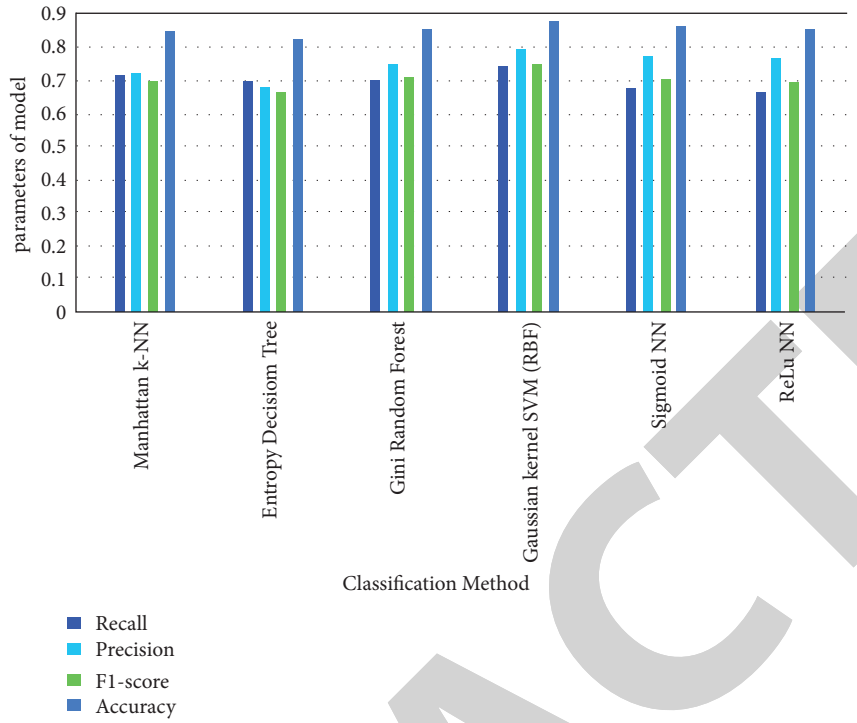


Figure 15: Performance of different classifiers model. Further performance comparison is conducted on the models after the first screening to select the models that can effectively classify data.

Table 4: Algorithms with relatively high accuracy are extracted for further evaluation, including accuracy, F1-score, and recall. SVM (RBF) has the best performance among various algorithms.

Calssification method	Manhattan k-NN	Entropy decisio tree	Gini random forest	Gaussian kernel SVM (RBF)	Sigmoid NN	ReLu NN
Recall	0.711495	0.694532	0.695101	0.740983	0.671318	0.660655
Precision	0.716116	0.677113	0.745203	0.793791	0.766872	0.764888
F1-score	0.694213	0.664086	0.704173	0.744384	0.700354	0.691554
Accuracy	0.844735	0.824281	0.852715	0.875404	0.858406	0.852071

Table 5: The evaluation results of the index finger and thumb detection model.

label	Accuracy	Recall	F1-score
r_thumb	0.774	0.71	0.73
l_index_finger	0.896	0.79	0.77
l_thumb	0.910	0.89	0.83
r_index_finger	0.896	0.93	0.84

Table 6: The accuracy of eight nodes on a limb.

label	Left hand heart	Left wrist	Left arm joint	Left shoulder joint
Accuracy	0.9986	0.983	0.9254	0.9688
Label	Right hand heart	Right wrist	Right arm joint	Right shoulder joint
Accuracy	0.973	0.9963	0.9945	0.9945

reached 600, so the number of trees at 600 will be set in the model, as shown in Figure 13.

For the Gaussian kernel, we are using the RBF kernel. The RBF kernel on two samples x and x' represented as feature vectors in some input space, is defined as:

$$K(x, x') = \exp\left(-\frac{\|x - x\prime\|^2}{2\sigma^2}\right). \quad (11)$$

Finally, we tried two neural networks with different cores (Sigmoid activation/ReLu activation) and going to use all of these models to train data sets and compare them to find the ones that are more accurate. We summarize the result data in Table 3 into Figure 14

The model with high probability of correctness is selected to calculate its other parameters such as recall, precision, and F1-score , shown in Figure 15 and Table 4. Gaussian kernel SVM (RBF) is more appropriate to help with the problem.

4. Results

Table 5 mainly shows the performance of detection such as accuracy, F1-score, and recall. The right thumb, index finger,

and left index finger were all about 90 percent accurate, but the right thumb was far behind. This error may be due to manually annotating the data set. The index finger of the right hand and the thumb of the left hand are slightly lower than the other two. At the same time, for f1-score performance of different thresholds, when the threshold size is 0.5, F1-score is the largest.

According to the data in Table 6, the model performs well in the recognition of human bone points, and all bone points have an accuracy rate of more than 0.9.

The first Angle is the special angle between the detection point at the wrist and two adjacent points, and the second Angle1 is the Angle formed by the detection point at the arm and the adjacent Angle. Table 2 includes two angles as the arm slides in all four directions. It can be seen from the results that when different actions are made, the changes of the two angles are in different intervals, which enables the computer to judge the actions by classifying the angles.

It shows common machine learning models in machine learning. The first considering indicator is the accuracy of motion classification. We select models with better accuracy for further selection.

It can be seen from Table 4 that SVM (RBF) is more suitable for solving problems, and its four indicators are better than the other models.

5. Conclusions

The detection accuracy of each finger of gesture is above 0.9, which can accurately detect finger movements and give control instructions to the computer. The recognition of human bone points can reach more than 0.97, which can better lay the position of human limbs. By calculating the included Angle between the key points and using a variety of classification algorithms to classify the actions, the classification accuracy of the four instructions reaches 0.875. This paper also constructed a visual human-computer interaction system with basic functions, and achieved good performance in human bone point recognition and finger recognition. In the future, it is hoped to further refine and distill these models so that HCI systems can complete high-performance reasoning and efficient HCI with only CPU.

Data Availability

No data were used to support this study.

Disclosure

The authors received no financial support for the research, authorship, and/or publication of this article.

Conflicts of Interest

The authors declare that they have no conflicts of interest or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] D. Sapiens Pargmana and E. Eriksson, "The Future of Computing and Wisdom: Insights from Human–Computer Interaction," *Futures*, vol. 113, Article ID 102434, 2019.

[2] M. Al-Ma'aitah, A. Alwadain, and A. Saad, "Application Dependable Interaction Module for Computer Vision-Based Human-Computer Interactions," *Computers & Electrical Engineering*, vol. 97, Article ID 107553, 2021.

[3] C Tomas and P. L. B. Chaffe, "SW2D-GPU: A Two-Dimensional Shallow Water Model Accelerated by GPGPU," *Environmental Modelling & Software*, vol. 145, Article ID 105205, 2021.

[4] R. Girshick and J. Donahue, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," in *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[5] R. Girshick, "Fast R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 1440–1448, Cambridge, MA, USA, June 2015.

[6] S. Ren and K. He, "Faster R.-C. N. N.: Towards Real-Time Object Detection with Region Proposal Networks," *EEE Transactions on Pattern Analysis and Machine Intelligence (IEEE)-*, vol. 39, no. 6, pp. 1137–1149, 2017.

[7] K. He and G Kioxari, "Mask R-CNN," *Deep Neural Network*, 2017.

[8] S. I. A. KumarGoswami, "Conceptual understanding of convolutional neural network-a deep learning approach," *Procedia Computer Science*, vol. 132, pp. 679–688, 2018.

[9] G. Ghiasi and T.-Y Lin, "NAS-FPN: Learning Scalable Feature Pyramid Architecture for Object Detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7036–7045, Long Beach, CA, USA, June 2019.

[10] F. Zhang and V. Bazarevsky, "MediaPipe Hands: On-Device Real-Time Hand Tracking," *Computer Vision and Pattern Recognition*, vol. 7, Article ID 10214, 2006.

[11] A. Caputoa, A Andrea Giachetti, and S. Soso, "SHREC 2021: Skeleton-Based Hand Gesture Recognition in the Wild," *Computers & Graphics*, vol. 99, pp. 201–211, 2021.

[12] M. E Ahmadzadeh Esmaeilabadi, "3D Virtual Reality Navigation by Human's Head Movements Using a Generic Webcam," *Procedia Technology*, vol. 3, pp. 121–131, 2012.

[13] A. Saada and A. A. Mohamed, "An Integrated Human Computer Interaction Scheme for Object Detection Using Deep Learning," *Computers & Electrical Engineering*, vol. 96, Article ID 107475, 2021.

[14] Y. Xu, D.-W. Park, and P. GouChol, "Hand gesture recognition based on convex defect," *Detection"International Journal of Applied Engineering Research ISSN*, vol. 12, no. 18, pp. 7075–7079, 2017.

[15] H. Tabkhi, "Algorithm and architecture co-design of Mixture of Gaussian (MoG) background subtraction for embedded vision," in *Proceedings of the 2013 Asilomar Conference on Signals, Systems and Computers (2013)*, pp. 1815–1820, IEEE, Pacific Grove, CA, USA, November 2013.

[16] L. Z. QiangWang, "An Interconnected Feature Pyramid Networks for Object Detection," *Journal of Visual Communication and Image Representation*, vol. 79, Article ID 103260, 2021.

[17] Y. Liu and S. Wang, "A Quantitative Detection Algorithm Based on Improved Faster R-CNN for marine Benthos," *Ecological Informatics*, vol. 60, Article ID 101228, 2021.
[18] C. Hsuan, K. Tzu-Chieh, and H. Shana Smith, "Usability Study of Multiple Vibrotactile Feedback Stimuli in an Entire Virtual Keyboard Input," *Applied Ergonomics*, vol. 90, Article ID 103270, 2020.
[19] V. Bazarevsky and I. Grishchenko, "BlazePose: On-Device Real-Time Body Pose Tracking," *ArXiv*, vol. 2006, Article ID 10204, 2020.
[20] K. Y. SuibinHuang, Y. Kun, and X. Hua, "A New Head Pose Tracking Method Based on Stereo Visual SLAM," *Visual Communication and Image Representation*, vol. 82, Article ID 103402, 2021.