Automated Attendance System in the Classroom Using Artificial Intelligence and Internet of Things Technology

Duy Dieu NGUYEN
Faculty of Information Science and Engineering
University of Information Technology – VNUHCM
Ho Chi Minh, Vietnam
dieund.14@ms.uit.edu.vn

Xuan Huy NGUYEN
Faculty of Information Science and Engineering
University of Information Technology – VNUHCM
Ho Chi Minh, Vietnam
huynx.15@ms.uit.edu.vn

The Tung THAN
Faculty of Computer Engineering
University of Information Technology – VNUHCM
Ho Chi Minh, Vietnam
tungtt@uit.edu.vn

Minh Son NGUYEN
Faculty of Computer Engineering
University of Information Technology – VNUHCM
Ho Chi Minh, Vietnam
sonnm@uit.edu.vn

2021 8th NAFOSTED Conference on Information and Computer Science (NICS) | 978-1-6654-1001-4/21/$31.00 ©2021 IEEE | DOI: 10.1109/NICS54270.2021.9700991

Abstract— **Computer vision is recently developing and applying in the utility apps serving people, facial recognition is one of its applications. Although the accuracy of the facial recognition is less than when compared to fingerprint recognition, iris recognition and Radio Frequency Identification (RFID) card recognition. But it is still widely used because the recognition process does not contact the device. With the advantage of the facial recognition method, we propose an automated attendance solution which uses embedded device integrated Artificial Intelligence technology (AI) and Internet of Things technology (IoT) in the smart classrooms. The highlight of the system is the ability to take attendance automatically and continuously throughout the learning period. When the students enter the class, the management department and the parents can know the student's participation status by viewing the report in the real-time system. The system consists of the main components: embedded device component with attached camera sensor, which is used for the process recognition and interacts with the Cloud server via IoT infrastructure; the Cloud server stores and provides data analysis devices for the administrator and parents. At the beginning of the roll call, the embedded device will receive an instruction to replace the old data with the new recorded data, which contains the characteristics and identifier code of the objects to be attended. The new data goes from the Cloud server in the respective classroom to the embedded device and then it compares with the data collected in the classroom. When the results are available, the embedded device interacts with the Cloud server to update the status of the students. The experimental results of the proposal system achieve accuracy per frame is 89%. The recognition speed of 82ms per face with a distance in the 4 - 15 meter range. The system which is an embedded system-based application solution has low operating costs and rapid deployment.**

Keywords— embedded system, facial recognition, real-time image processing on embedded system, automated attendance, smart-classroom.

I. INTRODUCTION

Currently, there are a lot of ways for taking attendance to capture a number of the students and determine the student's punctuality for school. The methods of manual attendance are facing some difficulties such as: taking attendance by signing the student list, which leads to cheating, omissions for latecomers, time-consuming for teachers to read student's names and to confirm the list; while taking attendance by voice recognition [6], iris recognition [7] or fingerprint recognition [3] which require student interaction. The facial recognition system implemented on computers or Cloud servers have been integrated into a lot of technology methods to solve problems such as: gate access control; timekeeping; dangerous object identification; intrusion stranger warning; searching for a person's image on social networks; or unlocking a smartphone, tablet via facial recognition… Most facial recognition algorithms need to use a large amount of system resources, making it difficult to integrate into the embedded device [5]. Thus, the integration of facial recognition with a large number of objects that need to be detected and classified on the embedded devices is a challenge. The embedded device is a compact computer system that is integrated both of hardware and software, it requires high stability and automation. However, the embedded devices are often limited in hardware resources (processing ability, storing ability, power consumption level, small memory,…) and software functions (due to hardware limitations, applications on the embedded device often have reduced features, few libraries for development, no operating system or operating system with many restrictions) than personal computers. Therefore, implementing the facial recognition algorithm for a school with a student population of 1000 to 3000 which is performed on the centralized Cloud server will spend a lot of infrastructure costs and processing-time. For this reason, this topic is oriented to integrate AI technology into embedded devices in the classes, which will improve the speed of information response from Cloud server to involved parties and reduce the performance cost of the infrastructure.

In this paper, we propose an automated attendance system in the classroom by facial recognition, which is integrated Artificial Intelligence (AI) and Internet of Things (IoT) technology into embedded devices and combined with Cloud server to complete a smart-classroom solution based on IoT Platform. The remainder of the paper is structured as follows: section 2 describes the automated attendance system architecture using AI-embedded device and Cloud-based, section 3 explains how the system works to recognite whether the students exist in classroom or not , the experimental result is presented in section 4 to demonstrate our automated attendance system in the classroom and finally the conclusion summarized in section 5

II. Design the Automated Attendance System

A. *Overview of Automated Attendance System on IoT Platform*

The attendance taking system in the classroom integrated AI and IoT technology into the embedded device has the architecture as shown in Fig. 1.

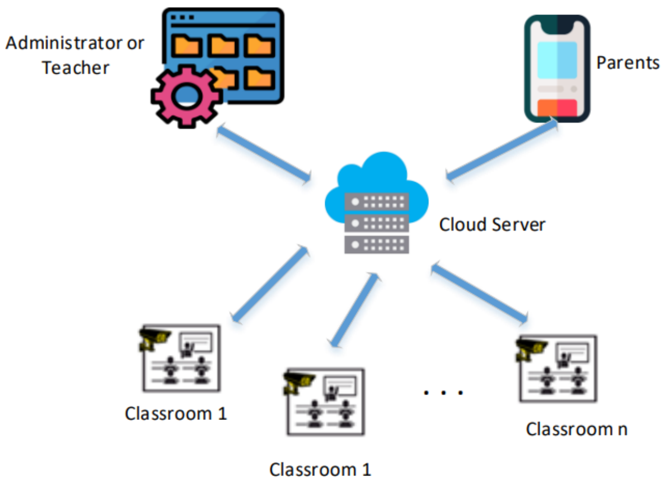


Fig. 1. System overview diagram.

- All identification and characteristic data used for recognition are stored on the Cloud server system. The embedded terminals are known as Node Clients.
- The administrator will build an identifying data for each room, each time and store it on the Cloud server.
- Before the roll call takes place, the Cloud server will delete the old data on the corresponding client node. At the same time, it will update with the new data which includes the list of students and the characteristic data used for identifying students.
- If the objects that are detected but are not in the list are sent down by the Cloud server, the system will record and send the characteristics back to the Cloud server. That helps the administrator to determine the new students.

B. *Object Recognition Process*

Firstly, from the images are captured by camera duo, the system determines the specific ID by identifying the features of the head posture and the faces in the frames, that will implement through different combinations of the algorithms:

- Using the get_frontal_face_detector() function of the dlib library for detecting the faces in the frames.
- Using shape predictor 68 face landmarks model for predicting the face position and rotating it to the frontal side.
- Using “dlib face recognition resnet model v1” for extracting the features, which contributes to the feature list of the faces in the current frame.

Then, the system synthesizes the characteristics of each face into a feature vector data and determines the ID of the objects by comparing it with the available data which is updated from the Cloud server.

We perform the face detection based on the method of Violas and John which is installed in OpenCV. Then, we continuously sample the correlation features to extract 68 features of eyes, nose, mouth, jaw area at different angles by integrating the dlib library. With the desire to increase the calculation speed and reduce the data size of the system, we use a face tracking technique in order to monitor the faces from current frame to next frame so that the system does not have to recognize the same object’s face many times. It’s useful for creating a face classification data record, i.e. a concise representation of the object’s face in a video sequence.

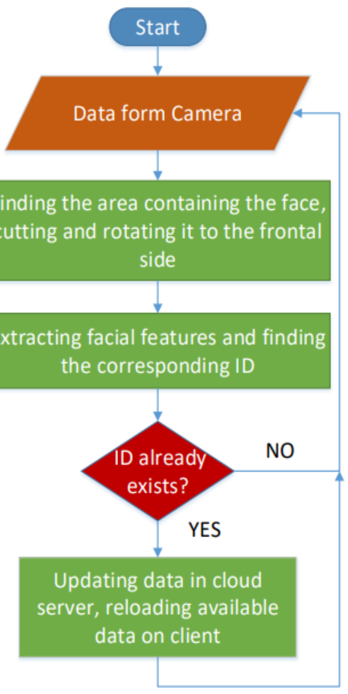

Start
Data form Camera
Finding the area containing the face, cutting and rotating it to the frontal side
Extracting facial features and finding the corresponding ID
ID already exists?
NO
YES
Updating data in cloud server, reloading available data on client

Fig. 2. Software system processing flowchart.

III. Face Recognition System

The camera captures numerous images which contain the faces of students in the classroom. The system determines the position of faces, extracts the features and classifies each object. For accuracy in feature classification of faces, we sample faces continuously by estimating head posture. Some different approaches use more techniques such as 3D data in depth image or temporary data in depth image video sequence, providing missing 3D data in 2D images. Video captures head movement continuously, so it provides useful data in order to estimate head posture. However, the iterative process of collecting temporary data causes the system to perform more computations and leads to an overload of system memory.

We perform the face detection based on the method of Violas and John which is installed in OpenCV. Then, we continuously sample the correlation features to extract 68 features of eyes, nose, mouth, jaw area at different angles by integrating the dlib library. With the desire to increase the calculation speed and reduce the data size of the system, we use a face tracking technique in order to monitor the faces from current frame to next frame so that the system does not have to recognize the same object’s face many times. It’s useful for creating a face classification data record, i.e. a concise representation of the object’s face in a video sequence.

A. *Facial Landmark Detection*

Facial landmark detection with dlib which is an algorithm finds out 68 key points according to coordinates (x, y) on the human face (Fig. 3).

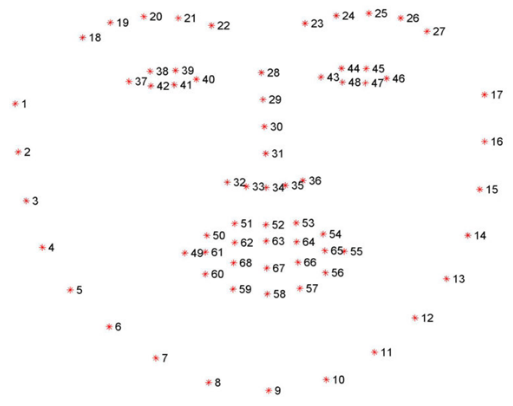


Fig. 3. Facial landmark finds out 68 key points on the face

To determine 68 key points on the human face, the facial landmark algorithm of dlib is trained with iBUG 300-W dataset. The more landmarks on the face can be identified, the more specificity, clarity and accuracy. However in this research, the algorithm is implemented on embedded devices which are limited in resources; therefore we stop at determining 68 key points.

B. *Convolutional Neural Network*

Face representation is the core of the recognition algorithm used in this system. The camera captures face images, then the system extracts features of each face, compares and classifies characteristics, with each of the characteristics saved as small-size data which are obtained by using Convolutional Neural Network (CNN). It is one of the algorithms in the Deep Learning model which brings together many algorithms to solve problems of processing multiple layers with complex structures. CNN model trains and tests each input image; passing it through convolution layers with Filters, Pooling layer and Fully Connected layer; applying Softmax function to classify an object with a probability value between 0 and 1 after that.

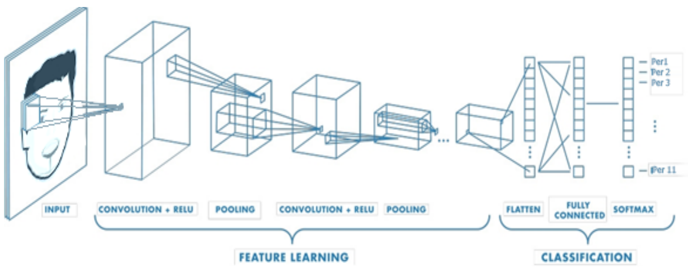


Fig. 4. Process of receiving and processing CNN network data

C. *Residual Network*

ResNet (R) is a CNN which is designed to work with hundreds or thousands of convolutional layers, it makes training possible and efficient to operate with hundreds or even thousands of layers of neural networks. With ResNet, many applications of computer vision including image classification are performance-enhanced. Some applications consist of object detection and facial recognition. Central idea of ResNet which uses Skip Connections to skip some of the layers in the neural network and feed the output of one layer as the input to the next layers. Such as block is called a residual block as shown in the following figure:

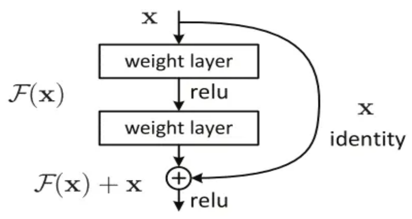

x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Fig. 5. Residual learning: a building block

Experiments show that this architecture is able to train neural networks with a depth of thousands of layers, so it promptly became the most popular architecture in the computer vision industry.

D. *K-Nearest Neighbors (KNN)*

KNN (K-Nearest Neighbors) is one of simplest supervised-learning algorithms widely used in classification. It does not learn anything from the dataset (KNN is classified as lazy learning). If it needs to predict labels of new data, it finds K neighbors and evaluates by the majority. For example, we have a dataset of labeled D points and dataset A which are unclassified, the process determines the label of A in KNN is:

- Measuring distance (Euclidian, Manhattan, Minkowski, Minkowski or weights)
- Choosing K points with the smallest distance from A
- Counting occurrences of each class.
- The class which has the most occurrences is the label of dataset A.

IV. Implements and Results

A. *Experimental Model in this Research*

The proposed automatic attendance system integrated the image processing into Jetson Nano Developer Kit which is connected to the cameras in order to collect images. The experiments in this research, the system recognizes the student's faces in the 2 - 15 meter range with high accuracy and short execution time. The information recorded by the system can be viewed or managed on mobile applications (iOS or Android) and web browsers on computers.

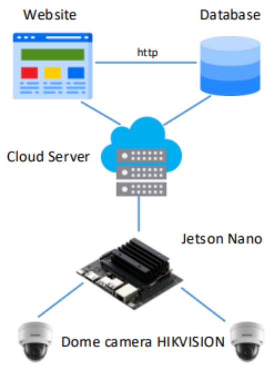


Fig. 6. Experimental system model

B. *Overview of the Relationship between Components*

The diagram in Fig. 7 shows three main components in an artificial intelligence system deployed on an embedded device. It includes cloud data center, AI processing block and embedded devices, end devices. This structure is implemented as an embedded system processing system model being developed in the present world [14].

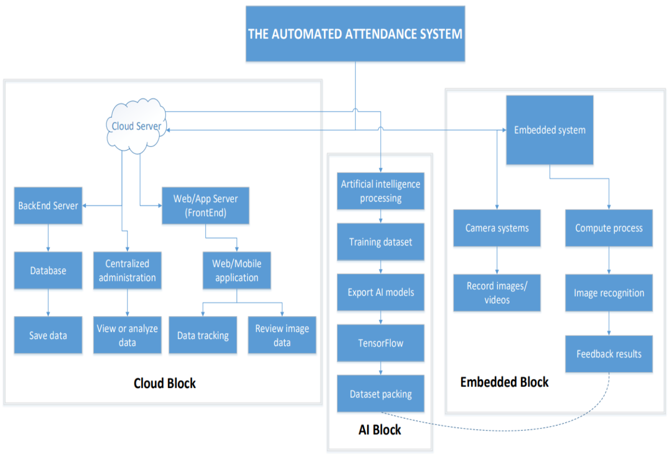


Fig. 7. Detailed model of the system

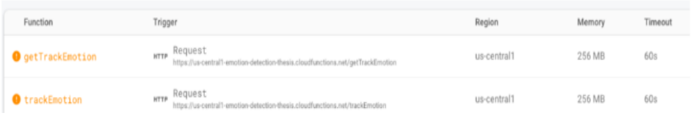
1) *Deploying cloud data center block:*

a) Functions: Central backend server performs the following tasks:

- Getting state data of the embedded system and send data from Cloud to the embedded system.
- Getting the data from the database and sending it to the web/app server.

b) Technology: Using C# programming language to create APIs that read and get data

c) Deployment



Function	Trigger	Region	Memory	Timeout
getTrackEmotion	HTTP Request https://us-central1-emotion-detection-thesis.cloudfunctions.net/getTrackEmotion	us-central1	256 MB	60s
trackEmotion	HTTP Request https://us-central1-emotion-detection-thesis.cloudfunctions.net/trackEmotion	us-central1	256 MB	60s

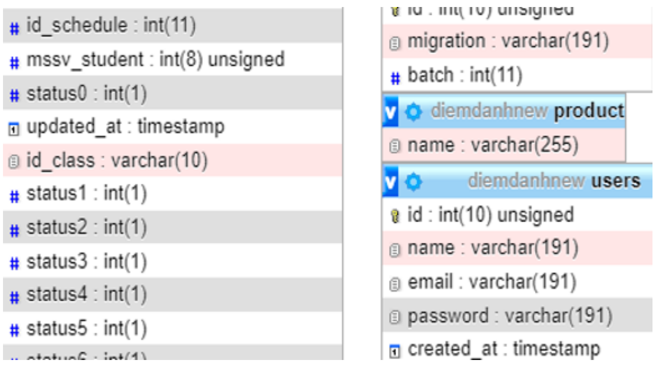
Fig. 8. Some functions are running in the central backend server

2) *Deploying database block:*

a) Functions: Database is stored following informations:

- General information about students.
- Admin account information.

b) Technology: Using NoSQL database is developed by MongoDB



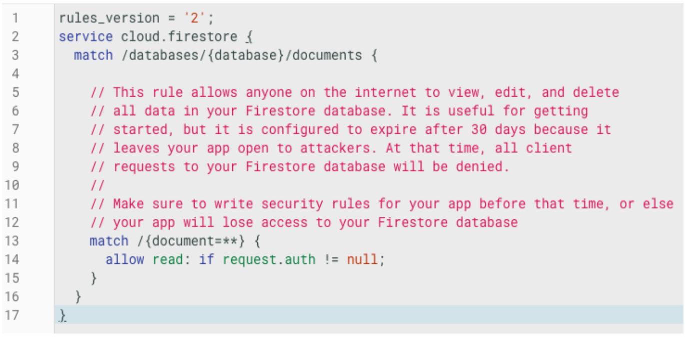
```
# id_schedule : int(11)
# mssv_student : int(8) unsigned
# status0 : int(1)
updated_at : timestamp
id_class : varchar(10)
# status1 : int(1)
# status2 : int(1)
# status3 : int(1)
# status4 : int(1)
# status5 : int(1)
status6 : int(1)
```

```
id : int(10) unsigned
migration : varchar(191)
# batch : int(11)
diemdanhnew product
name : varchar(255)
diemdanhnew users
id : int(10) unsigned
name : varchar(191)
email : varchar(191)
password : varchar(191)
created_at : timestamp
```

Fig. 9. Student data structure is stored in the database

c) Deployment: The database is the middle ground to connect web applications and devices. All web applications must go through the database to connect to the device. The device translates information which is stored in the device's internal memory and sends it to the database. That means even without a network connection, the collected information of the device is still not lost.

- In addition, with a view to enhancing database security, we have set rules so as to allow authorized users to access the database.



```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {

    // This rule allows anyone on the internet to view, edit, and delete
    // all data in your Firestore database. It is useful for getting
    // started, but it is configured to expire after 30 days because it
    // leaves your app open to attackers. At that time, all client
    // requests to your Firestore database will be denied.
    //
    // Make sure to write security rules for your app before that time, or else
    // your app will lose access to your Firestore database
    match /{document=**} {
      allow read: if request.auth != null;
    }
  }
}
```

Fig. 10. The rules for accessing the database

3) *Deploying Web server block:*

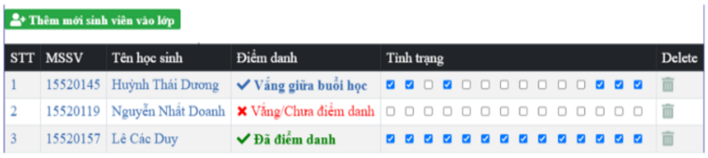
a) Functions: Web server is where the following information can be viewed:

- Information about attendance management.
- Information about the devices management.
- Information about the classroom management.
- Information about the students.

b) Technology: The system use Bootstrap, C# programing language and Websocket communication, so has the following advantages:

- Fast development time.
- Running cross-platform: Windows, Android, iOS
- Supporting hot reload to make programming easier.
- High performance, updated data in real-time.

c) Deployment



STT	MSSV	Tên học sinh	Điểm danh	Tình trạng	Delete
1	15520145	Huỳnh Thái Dương	✓ Vắng giữa buổi học	☑ ☑ ☐ ☑ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☑ ☑ ☑	🗑
2	15520119	Nguyễn Nhất Doanh	✗ Vắng/Chưa điểm danh	☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐	🗑
3	15520157	Lê Các Duy	✓ Đã điểm danh	☑ ☑ ☑ ☑ ☑ ☑ ☑ ☑ ☑ ☑ ☑ ☑ ☑ ☑	🗑

Fig. 11. Student attendance list displays WEB

4) *Deploying training AI model block:*

a) Functions: AI models are responsible for analyzing the kernel data from the camera device and presenting facial recognition results.

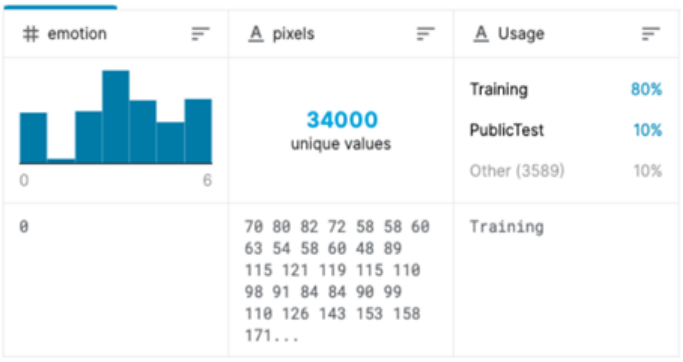
b) Technology: Using Google machine learning service with advantages:

- Easy to perform for users who do not have in-depth knowledge of machine learning.
- Assisting to export many types of the models with different aims.

- Using most optimal algorithms to train corresponding to each type of data.

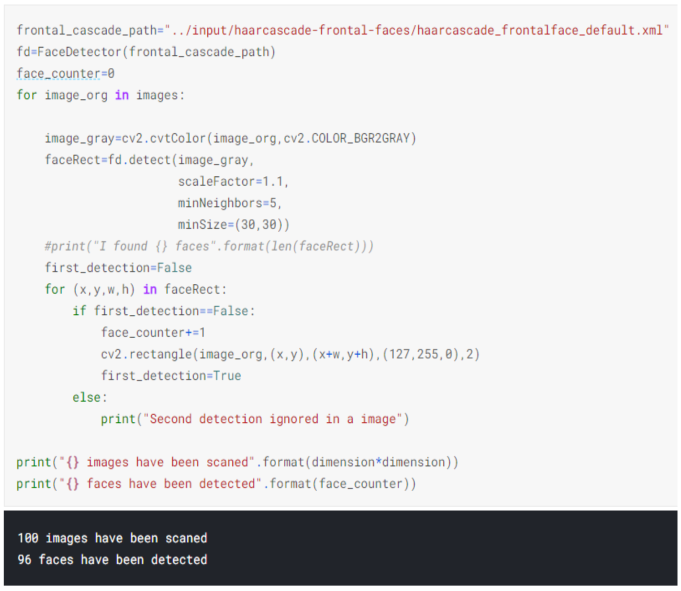
c) Deployment: We created a dataset of 34,000 images of 18 different students, which is about 1000 photos for each person.

- Image size: 48x48.
- Image color: black, white.
- Image characteristics: human faces sampled from camera.
- Image pixels: string of integers separated by spaces, consists of 48x48 consecutive integers corresponding to the number of pixels of an image, the magnitude of the integer depends on the density of each pixel.



# emotion	A pixels	A Usage
0 – 6	34000 unique values	Training 80% PublicTest 10% Other (3589) 10%
0	70 80 82 72 58 58 60 63 54 58 60 48 89 115 121 119 115 110 98 91 84 84 90 99 110 126 143 153 158 171...	Training

Fig. 12. Training results of facial recognition consists of 34000 records



```
frontal_cascade_path="../input/haarcascade-frontal-faces/haarcascade_frontalface_default.xml"
fd=FaceDetector(frontal_cascade_path)
face_counter=0
for image_org in images:

    image_gray=cv2.cvtColor(image_org,cv2.COLOR_BGR2GRAY)
    faceRect=fd.detect(image_gray,
                       scaleFactor=1.1,
                       minNeighbors=5,
                       minSize=(30,30))
    #print("I found {} faces".format(len(faceRect)))
    first_detection=False
    for (x,y,w,h) in faceRect:
        if first_detection==False:
            face_counter+=1
            cv2.rectangle(image_org,(x,y),(x+w,y+h),(127,255,0),2)
            first_detection=True
        else:
            print("Second detection ignored in a image")

print("{} images have been scaned".format(dimension*dimension))
print("{} faces have been detected".format(face_counter))
```

```
100 images have been scaned
96 faces have been detected
```

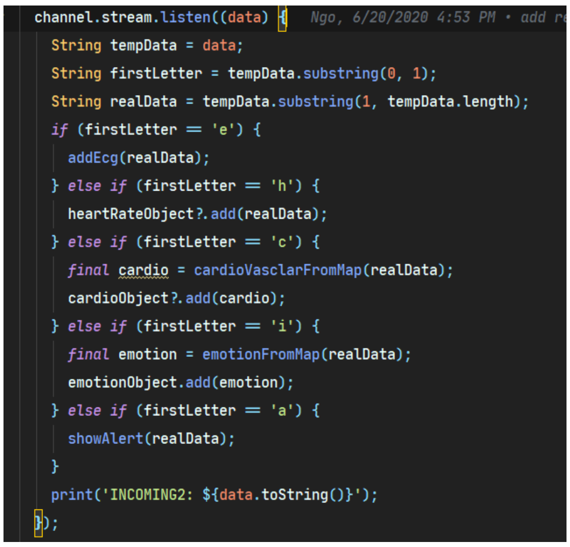
Fig. 13. Probability of the facial recognition with 100 images

5) Deploying embedded system:

a) Functions: The device has an ability to track faces of the students and send it to the BackEnd Server.

b) Technology: The devices used to create the embedded block include: Jetson Nano Developer Kit, Dome camera HIKVISION. Employing Websocket protocol to send images which are converted to base64 type to server.

c) Deployment: With a view to sending data continuously, we use TCP socket protocol to send data to Backend Server. We use C++ language on the embedded platform to program and upload source code to Jetson Nano 4GB.



```
channel.stream.listen((data) {   Ngo, 6/20/2020 4:53 PM • add re
  String tempData = data;
  String firstLetter = tempData.substring(0, 1);
  String realData = tempData.substring(1, tempData.length);
  if (firstLetter == 'e') {
    addEcg(realData);
  } else if (firstLetter == 'h') {
    heartRateObject?.add(realData);
  } else if (firstLetter == 'c') {
    final cardio = cardioVasclarFromMap(realData);
    cardioObject?.add(cardio);
  } else if (firstLetter == 'i') {
    final emotion = emotionFromMap(realData);
    emotionObject.add(emotion);
  } else if (firstLetter == 'a') {
    showAlert(realData);
  }
  print('INCOMING2: ${data.toString()}');
});
```

Fig. 14. Source code receiving data from Edge Backend Server

C. *Results of the program implementation*

The camera is mounted on the corner of the board with a 20 degree angle down towards the students, the face detection program will detect any face appearing in the frame that has been captured by the camera. With an aim of evaluating the overall performance of the proposed system exactly, we collected face images for attendance (as a library dataset) and videos for testing (as an exploratory dataset) in a classroom environment. The newly collected dataset includes 34 objects with 34000 images and a video sequence of each face. Static images for each object which are captured in the 3 - 10 meter range from the two system cameras were used as the library dataset, and 20fps for each video were used as the exploratory dataset.

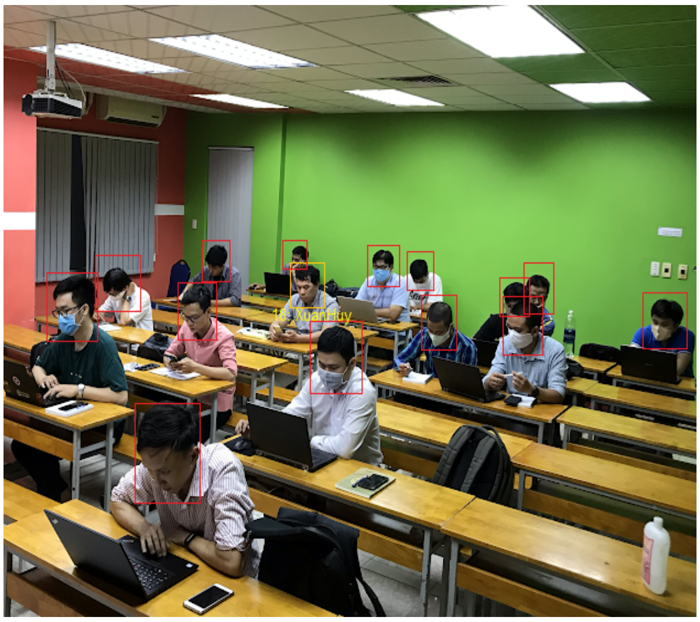


Fig. 15. Object detection and recognition

The average detection and recognition time (25 experiments with each distance) of the system, for each student appearing in the frame when the system has detected the object, the processing time of the objects is approximately.

With a detection time of 85.6ms and a recognition time of 25.1ms as shown in Table I.

TABLE I. EXECUTION TIME OF AUTOMATIC ATTENDANCE SYSTEM

Distance (meter)	Detection time (ms)	Recognition time (ms)
0.3	86	25
2	85	25
4	87	26
6	81	28
8	89	24
10	86	23

Through the experiment, the system begins to detect and recognize at a distance of 0.3 meters to 20 meters, the accuracy decreases with that distance, Fig. 16.

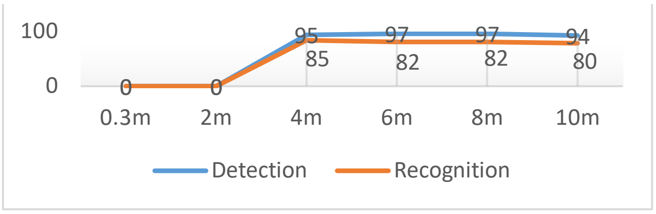


Fig. 16. Graph accuracy (%) of detection and recognition by distance.

With different camera focal lengths, the system will result in the accuracy of object recognition (Fig. 17).

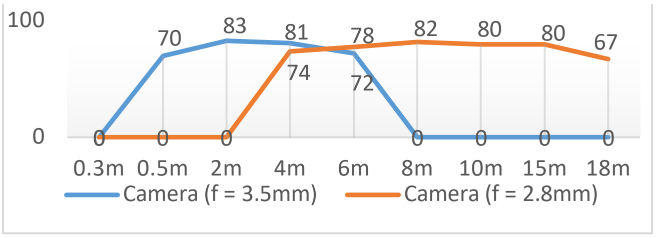


Fig. 17. Accuracy chart (%) of 2 types of lenses with different focal lengths.

Resources without face detection measured during 10 minutes in the frame (Fig. 18) are quite stable, CPU below 25% and GPU below 5%.

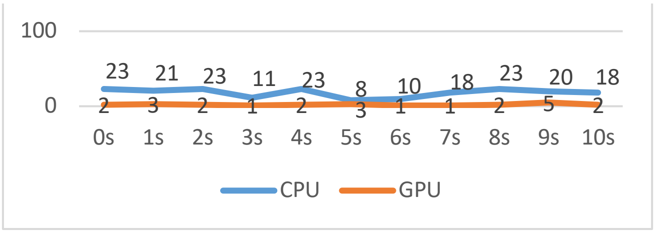


Fig. 18. Resources graph at no object

When detecting 15 objects appearing in the frame GPU and CPU increase significantly

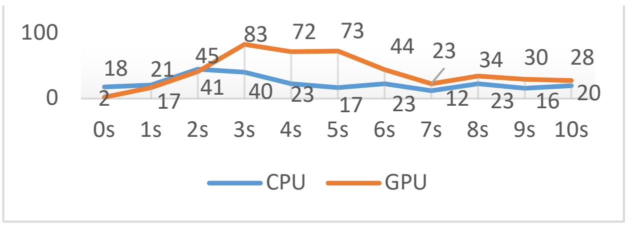


Fig. 19. Resources graph when there is an object in the frame

V. CONCLUSION

This paper presents the design and implementation of an independent, multi-modal access control system based on human face recognition at a distance. The proposed method can detect faces of different image sizes at a distance in a classroom making the system more user-friendly, giving teachers more time to teach and parents know exactly how their children go to school. The system achieves a correct recognition rate of 89.0% with a detection rate of 7 fps for 1 person in case the environment is well-lit and the face angle is from -20o to 20o in the opposite direction to the camera. When running the experiment on the pre-trained data set with 15 people appearing in the frame (containing both sampled and unsampled subjects), it reached 7 fps.

REFERENCES

[1] Maurizfa, Trio Adiono, "Smart Attendance Recording Device Based on Fingerprint Identification", 2021 International Symposium on Electronics and Smart Devices, June 2021.

[2] Seng Chun Hoo and Haidi Ibrahim, "Biometric-Based Attendance Tracking System for Education Sectors: A Literature Survey on Hardware Requirements", Journal of Sensor, Sep. 2019.

[3] Suraj Raj, Saikat Basu, "Attendance Automation Using Computer Vision and Biometrics-Based Authentication-A Review", Lecture Note on Data Engineering and Communications Technologies, Jun 2021.

[4] S, Kadry; K. Smaili, "A Design and Implementation of A Wireless Iris Recognition Attendance Management System", ISSN 1392 – 124X Information Technology and Control, Vol.36, No.3, 2007.

[5] Hicham El Mrabet, Abdelaziz Ait Moussa, "IoT-School Attendance System Using RFID Technology", International Journal of Interactive Mobile Technologies, Apr. 2020.

[6] Seon Ho Oh, Geon-Woo Kim & Kyung-Soo Lim, "Compact deep learned feature-based face recognition for Visual Internet of Things", Journal of SuperComputing, 2018.

[7] Andrew Boles, Paul Rad, "Voice biometrics: Deep learning-based voiceprint authentication system", 2017 12th System of Systems Engineering Conference, 2017.

[8] Alaa S. Al-Waisy, Rami Qahwaji, Stanley Ispon, Shumoos Al-Fahdawi & Tarek A. M. Nagem, "A multi-biometric iris recognition system based on a deep learning approach", Journal of Pattern Analysis and Applications, 2018.

[9] Kostiantyn S Khabarlak, Larysa S. Koriashikina, "Fast Facial Landmark Detection and Applications: A Survey", Computer Vision and Pattern Recognition, Apr 2021.

[10] Journal of Real-Time Image Processing (2021)

[11] Constantino Alvarez Casado & Miguel Bordallo Lopez, "Real-time face alignment: evaluation methods, training strategies and implementation optimization", Journal of Real-time Image Processing, 2021

[12] NVIDIA Developer, "Jetson Nano Developer Kit", [Online], 2019. Available: https://developer.nvidia.com/embedded/jetson-nano-developer-kit

[13] OpenCV,"OpenCV modules" [Online], 2020. Available: https://docs.opencv.org/

[14] Dlib,"Dlib C++ Library" [Online], 2020. Available: http://dlib.net/