Attendance System using Machine Learning-based Face Detection for Meeting Room Application

Rahmat Muttaqin[1]
Telematika, Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Bandung, Indonesia

Nopendri[2]
School of Industrial and System Engineering
Universitas Telkom, Bandung, Indonesia

Syifaul Fuada[3]
Universitas Pendidikan Indonesia
Bandung, Indonesia

Eueung Mulyana[4]
School of Electrical Engineering and Informatics
Institut Teknologi Bandung, Bandung, Indonesia

***Abstract*—In a modern meeting room, a smart system to make attendance quickly is mandatory. Most of the existing systems perform manual attendance, such as registration and fingerprint. Despite the fingerprint method can reject the Unknown person and give the grant access to the Known person, it will take time to register first a person one-by-one. Moreover, it is possible to create long queues for fingerprint checking before entering the meeting room. Machine learning, along with the Internet of Things (IoT) technology is the best solution; it offers many advantages when applied in the meeting rooms. Generally, the method used is to create a presence by detecting faces. In this paper, we present a facial recognition authentication based on machine learning technology for connection to the meeting rooms. Furthermore, specific website to display the detection result and data storage design testing is developed. The method uses 1) the Dlib library for deep learning purposes, 2) OpenCV for video camera processing, and 3) Face Recognition for Dlib processing. The proposed system allows placing the multiple cameras in a meeting room as needed. However, in this work, we only used one camera as the main system. Tests conducted include identification of one Known person, identification of one Unknown person, identification of two people, and three people. The parameter to be focused is the required time in detecting the number of faces recorded by the camera. The results reveal that the face can be recognized or not recognized, then it will be displayed on the website.**

***Keywords*—Detection; face; IoT; meeting room; attendance**

I. Introduction

In this industry 4.0 era as nowadays, the development of machine learning technology is growing rapidly. Research in this area and its use-case in various fields have been demonstrated widely [1]. By using machine learning, the devices can perform separate types of learning, such as image [2-3], voice, shortest path, video pattern recognition, fault in power transmission systems [4], patterns in communication systems [5], etc. Besides, there are also developments in the Internet of Things (IoT) technology that utilize internet connections for data transfer without human intervention. This technology is usually applied to communication on microcontroller devices to send sensor data, gateways in the form of single board computers, and clouds [6]. Machine learning can be incorporated with IoT to build an intelligent system, one of them is a real-time attendance system. This technology can be a solution to streamline existing systems where most of the attendance systems still utilize the fingerprint method (including RFID-based machines). Indeed, it consumes a lot of time, especially at the personal registration and fingerprint checking (check-log) stages [7-11]. With machine learning, a person's or the people's presence in a meeting room can be detected quickly and accurately. This learning machine requires a lot of data to perform excellent detection.

Another technique like the Histogram Oriented Gradients (HOG) actually can be used as a method for face recognition application [12-15]. Facial recognition is a method for character recognition on faces that are successfully detected. To get better results compared to the method of skin color-based face detection; it can be performed by using a combination of Haar Cascade & Binary Pattern [16], or Haar-like & Adaboost features [17]. The detected objects are then classified into three types orientation based on the object's motion: horizontal, vertical, and rotational. But Machine learning method is able to provide various advantages over traditional methods when trained using large amounts of data. The more data, the better the results. For facial recognition application, we can employ two commonly used open-source libraries, i.e., Dlib and OpenCV [18]. Dlib is created using C++ language, which includes tools used for machine learning [19]. In addition, Dlib offers linear algebra where the central core uses the Basic Linear Algebra Subprogram (BLAS) [19].

Face detection is one area of research that has great potential for further development. By far, there have been many studies on Machine Learning-based face detection, the most recent research is reported by *S. Khan*, et al. [20]. Their system has successfully detected 2 to 12 people within the classroom and got 100% accurate detection results. Even so, we still want to explore this topic to get the most optimal performance facial recognition, where the system can work in real-time, which is then applied to meeting rooms. The faster the face detection, the more efficient the attendance process. Since [20] focused on the detection accuracy and the time required for detection was not explicitly presented; therefore, for this work, we will more emphasize on detection speed.

In this work, the system limitations are determined as follows: A node contains a single PC and a single camera. In the system implementation, the server and node have resided in the same PC. We limit the dataset in the form of an image under 1600 x 1500 pixels. Similar to [20], we only analyze the detection between the Unknown and Known person(s). The Unknown data used for every detected person is determined one Unknown without specific ID, it means there is the same ID for all Unknown people. The detection method employs only HOG and convolutional neural network (CNN).

Four sections compose this paper. First is Introduction that elaborates the related work and research motivation, 2) Methods discusses the intelligent attendance system for meeting room including the division of subsystems to be developed, 3) Results and Analysis, and the last section is 4) Conclusion.

II. Methods

This system was developed under the Linux OS Ubuntu Desktop 18.04 LTS. The smart meeting room consisted of three subsystems, namely, Back-End, Front-End, and Node as illustrated in Fig. 1.

Each subsystem has a specific purpose. In this system, data is first obtained from the camera to the node. Local stream uses Real Time Streaming Protocol (RTSP). The node will process the data to be recognized by all the datasets existed on the node. After processing, the data is sent to the server (Back-End) to be saved to the database. Then Front-End will request data at specific intervals to the Back-End for real-time data up to 10 seconds before. The data will be displayed under the Back-End. To support IoT, the HyperText Transfer Protocol (HTTP) was utilized as a communication protocol between Client to Front-End.

A. *Front End Design*

In general, Front-End functions are for: 1) Requesting data to the Back-End by using the RESTful API; 2) Displaying data and images obtained from Back-End; and 3) giving the grant access to user/admin to the system. This section used the Vue.js framework based on the Javascript programming language. On Linux OS, the platforms that must be available are NPM and Node.js, which can be installed using the Ubuntu package manager. Some commands were carried out first until finally, it produced a folder called `dist`, which contained an `index.html` file. The file can be installed on the webserver to be accessed.

The main modules used are 1) Axios, used for RESTful API requests in the form of an HTTP POST or GET methods, 2) Vue-Router, used for routing or addressing from the display, 3) Vuetify, used for material design, and 4) Vuex as state management. The of Front-End's work-flow diagram can be seen in Fig. 2. The flowchart initially shows a list of available rooms. When a room is clicked or selected, the person/face data in the database will be displayed. A more detailed explanation will be explained in the next paragraph.

As in Fig. 2, first step is the Dashboard requests a list of rooms by doing an HTTP GET request to the Back-End part to the following link http://[IP:port]/attendance/room/. IP and port are matched with server address where Back-End is installed. The response received by the Front-End is in the form of data in JavaScript Object Notation (JSON) format, i.e., {“listdata”:[{“id”:1,”name”:”Kelas A”},{“id”:2, “name”:”Ruang Rapat”}]}. Later, the data is parsed to take the ID and name parts. For example, to retrieve the ID data: [“listdata”][0][“id”]. If the room data has been obtained, the list will be displayed on the dashboard using v-for and v-bind: href tags. Later, the dashboard display is appeared; when the room name is clicked, the system will request the Back-End server, which is a list of people within the room. The data is requested using HTTP POST to the following address: http://[ip]:[port]/attendance/currentposition/.

A JSON formatted parameter that is {“location“: [room_id]} must exist and be set in the body of HTTP requests. Id_ruangan is obtained on previous requests. For example, the data obtained is in the form of codes as below, where the data is a list of JSON: {“listdata:[{“person_id”:1,”person__name”:”syifaul”,”timestamp”:1568689950.1496825,”profile_image_link”:“/storage/image/profile/0000001_c.jpg”},{“person_id”:2,”person__name”:”tq”,”timestamp”:1568699955.1496825,”profile_image_link”:“/storage/image/profile/0000002_c.jpg”}]”}. If we want to take the link of profile image data, it can be achieved by the following way: data[“listdata”][0][“profile_image_link”]. This data is used to display images of people present which are appeared on the dashboard.

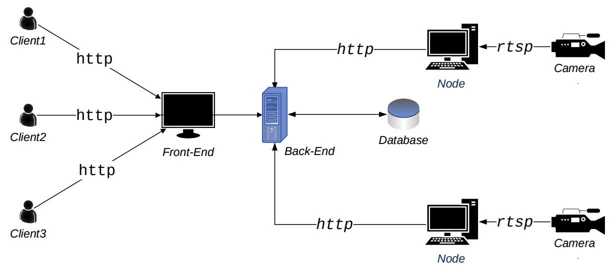

Fig. 1. Smart Meeting Room System Architecture.

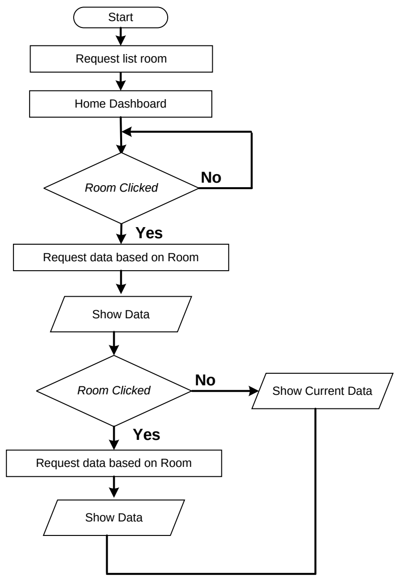

Fig. 2. Flowchart of Front-End.

B. *Back End Design*

In general, the functions of Back-End are: 1) Providing an API to be accessed by Front-End in requesting data, 2) Providing an API used by nodes to send and store captured data, 3) Saving data to the database as seen in Table I, and 4) Registering the user names and rooms.

TABLE I. DATA BASE STRUCTURE

Name of Table	Function	Components
Person	Contains users that are recognized by the node.	ID: Integer Name: String (Length limit 40) Profile_image: Image File
Location	place / location of the camera is	ID: Integer Name: String (40)
Camera	Camera ID	ID: Integer Location: Foreignkey (Location)
Position	A collection of users detected, both registered and not	ID: Integer Person: Foreignkey (Person) Camera: Foreignkey (Camera) Location: Foreignkey (Location) Timestamp: Float Profile_image_link: String

In this work, the Back-End employs the Django v2.2 Framework, so the programming language is Python3.6. However, on Ubuntu 18.04 version, Python3.6 is already installed. Therefore, we do not need install Python again. To register, we need access to Django Admin which can be traced using a web browser to the following address: http://[ip]:[port]/admin. The first step is a request to enter the superuser's username and password that has been created. If it has been filled-in and submitted, the dashboard of Back-end admin will appear. On the dashboard, the superuser can register person, location and camera. However, registering a camera requires a room parameter. Herein, before registering it, we need to register the room first.

In the Back-End, there are several APIs that function as a communication tool with other parts, namely Login, Refresh, Position, and Current Position. Each function is described as follows:

- *Login*, which is used to get tokens in the form of JWT format, which are then used to access several available APIs. The method used is HTTP Post, the body is JSON with the following format: {"username" "admin","password" "bandung"}. Username is registered in the create superuser menu or in the Django admin window. In the response section, there are two keywords, namely refresh and access. Access is used in subsequent API requests, whereas refresh is for requesting a new token or re-requesting the token if the access has expired. The token duration before being refreshed is about 5 minutes. Meanwhile, the use of a refreshed token can be up to 1 day before the validity period expires.
- *Position*, which is used to transmit a person's data detected by the camera. A person is a list of user IDs in the dataset. Location is the location's ID where the user was detected, the camera is the ID of the camera used, and the timestamp is the time when the data was sent to the server.
- *The Unknown*, which is used to send data to the server if the camera detects an Unknown person. An Unknown person is a person who is not registered in the existing dataset. Each Unknown ID has a face photo that is sent to the server. The face photo data is converted to base64. When received by the server, the photo data is decoded into a JPG file format. The image is then saved in the storage media folder, which is useful as a link for the detected Unknown ID profile image.
- *Current Position*, which is used to get a list of users who are currently in the meeting room, even some time ago. The data taken for the past time can be adjusted according to the need, usually the time is set 2 to 300 seconds ago.

C. *Node Design*

In general, the functions of the node are: 1) Face coding, which is converting the image data sets in the provided data set into a 128-d vector, 2) Face capturing, which detects the location of face captured by the camera, 3) Face recognition, which is finding the owner face, and 4) Sending data to the server. Two libraries that are needed for this purpose: Python library and NVDIA CUDA.

Phyton library consists of Dlib and OpenCV: the libraries that provide tools for machine learning and image processing, respectively. Later, Face_recognition, a library that simplifies the use of Dlib. Meanwhile, NVIDIA CUDA is used as a driver to utilize the NVIDIA graphics card core in machine learning processing.

Face Encoding is the process of classifying faces into 128-d vectors (128 real numbers) that represent faces Known to the computer, as exhibited in Fig. 3. The classification is done using previously trained network (pre-trained network) using approximately three million images in the Dlib library, so we only need to use the pre-trained network. In the trainer, there is a configuration that must be set in the `config.ini` file, as follows:

```
File faceconfig.ini
[Trainer]
dataset_folder= dataset
encodings_output=
trainer_output/djangodata.pickle
detection_method=cnn
```

These parameters are filled depending on the needs. The detection method can use CNN or HOG. Dataset_folder is a folder containing the datasets in the form of images collection. The image folder can be different depending on the image identity (in the form of the name or the personal identity concerned in the image or photo). For example, in folder A contains photos of A, in folder B contains B, in folder C contains photos of C, and so on. These folders are all stored in a folder called dataset. The dataset folder is placed in the root folder so that the settings are written as follows dataset_folder = dataset. While Encodings_output is the output file encoded in this process.

In Fig. 3, initialization is processed at the beginning, such as determining the location of the dataset, encoding the output location, the face detection method, etc. For the face detection method, CNN is used by using the Dlib library, which is set using an NVIDIA CUDA graphics card. Thus, the processing is more accurate and faster.

For each detected image, a name for the image folder will be taken and named according to the detected person's name. Then the image color changes from RGB OpenCV to RGB Dlib format. Afterward, the face detection was carried out using CNN. The detected faces are marked by making a rectangle around the face which is then converted into 128 real numbers. For each encoding of that one image, the existing encoding list is updated with matching names. The resulting encoding and name are made into a JSON model with the given format: {"encodings": knownEncodings, "names": knownNames}. KnownEncodings is a list of detected encoding (128-d vector), whereas knownNames is a detected name. Each encoding and name detected at one time occupies the same list element. The JSON result is written into a file with a pickle extension. When the format is written, all of them are converted into Hex numbers. This pickle extension file will be used to recognize faces detected by the camera.

In the Face recognition process, it requires a file with a pickle extension generated from the Face Encoding process. In this process, the `faceconfig.ini` file must be set up first, as follows:

```
Faceconfig.ini
[Detector]
camera_id= 3
location_id = 2
; time periodic for sending message to server
in seconds
periodic = 2
; time periodic for capturing/save to disk
unknown picture and send unknown data to server
in seconds
periodic_unknown = 3
; input file.pickle
encodings_input                              =
trainer_output/djangodata.pickle
; hog or cnn
detection_method=cnn
; 1 to show display or 0 to hide display
display = 1
; 1 to save video to disk, 0 not save video
save_video_to_disk = 1
;input_type can be a file,stream_local or
stream_ip
;input_type= file
input_type= stream_ip
;if input_type = file, set the video
destination file
input_video = lunch_scene.mp4
;if input_type = stream_ip, set ip
camera_address
camera_address                               =
rtsp://admin:QXJMCF@192.168.0.100
; unknown id
unknown_id = 99999999
[Server]
;username and password server
username = admin
password = bandung
server_address = 127.0.0.1
server_port = 8000
secure = 0
```

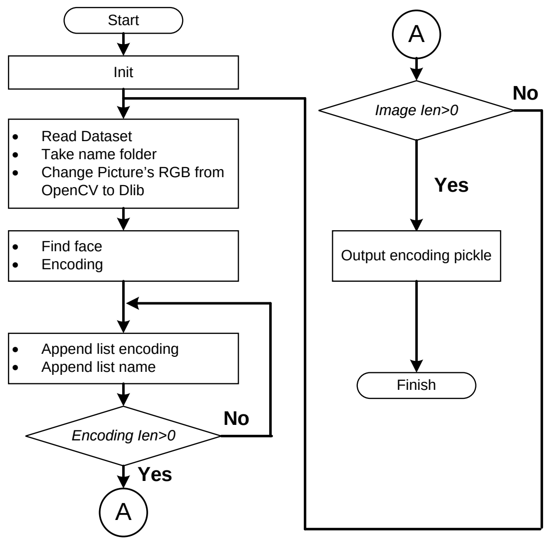

Fig. 3. Flowchart of Face Encoding.

In the file above, the camera_id and location_id values are integers, in which the values were determined by the admin during the Back-End setup. Periodic denotes the length time required for transmission of detected data. Periodic_unknown denotes the period for sending Unknown data if the detected face is not found in the encoding file with the pickle extension. Encodings_input represents a file from the face encoding process which is used as a reference for face detection. Detection_method is used as a method for detecting faces: if CNN is selected, it can be written cnn, or hog if HOG is selected. Display is used to display video from a camera where the value 1 = yes, and 0 = no. Save_video_to_disk to save the video from the camera to a storage place, where value 1 = yes, and 0 = no. Input_type is used for input data settings. If it comes from an input file, write File, then stream_local for local stream and stream_ip for RTSP inputs. Input_video contains the address of the video input if input_type is filled with files. Camera_address represents the camera address when using stream_ip. Later, Unknown_id is the Unknown identification set on the Back-End by the admin. Username and password are created by the admin on the Back-End, where the username is used as authentication in sending files to the server. Server_address and Server_port are Back-End addresses. Secure is set to 0 if using HTTP, and 1 if HTTPS.

In Fig. 4 initialization is done at the beginning, such as specifying the location of the timeout dataset, etc.

After that in each loop, it reads the streaming video data per frame, changes the color from BGR to RGB. Accordingly, it can be read by Dlib, and reduces the reading frame size to speed up the process. The next step is face detection using CNN by utilizing an NVIDIA graphics card. Every detected face is made a kind of square or box-shaped divider to mark the face area. The marked area is encoded into a 128-d vector. The results of the encoding if there are two faces, there will be two different results. The encoding results are stored in a list. Each data in the list is compared with the encoding data that is already owned. Compare_face with the input pickle extension file can be used to perform the comparison function.

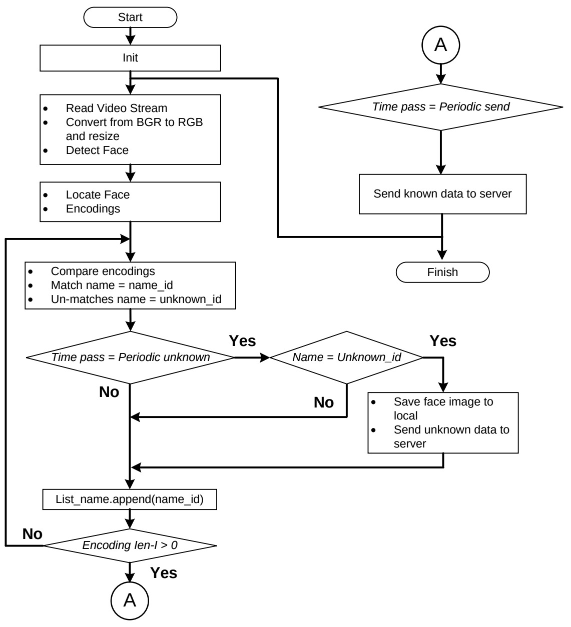
Start
Init
Read Video Stream
Convert from BGR to RGB and resize
Detect Face
Locate Face
Encodings
Compare encodings
Match name = name_id
Un-matches name = unknown_id
Time pass = Periodic unknown
Yes
Name = Unknown_id
Yes
No
No
Save face image to local
Send unknown data to server
List_name.append(name_id)
No
Encoding len-I > 0
Yes
A
A
Time pass = Periodic send
Send known data to server
Finish

Fig. 4. Flowchart of Face Recognition.

In the compare_face function, the tolerance parameter is set to 0.5. Every data that matches, its name or ID is stored in a name index list. In that list, a name is taken based on how often that name appears in the name index list. Only one name that comes out is put into a list again, that is the list of names containing anyone detected in a frame. If the encoding result data does not match the data in the pickle extension file, then the data is classified as Unknown. Image files (faces) retrieved from unknown data are then stored on the local drive, part of the captured data is sent to the server for display. The APIs used to send Unknown data are:

Method: POST
URL: http://ip:port/attendance/unknown/

```
Body:{"person":[99999999],"location":2,"camera":
3,"timestamp":                        1556783276,’
profile_image_unknown_data’:<base64encoding>}
Response: status= HTTP_201_CREATED
```

The APIs, as mentioned earlier, send Unknown data every certain period if Unknown data is detected. In this case, the Unknown data cannot be distinguished from one another. All Unknown images will be stored in the local media/Unknown folder. The admin can sort these images into additional datasets and send some of them to the server using the API above. The Unknown image sent to the server will first be converted to base64 format. The data is wrapped in a JSON, as can be seen in the previous API. If the data sent can be recognized, then the data is sent to the server within a certain period using the API. In the API, there is no image sending because the image which will be displayed taken from the profile.

Method: POST
URL: http://ip:port/attendance/position/

```
Body:{"person":[1,5,6],"location":2,"camera":3,"t
imestamp": 1556783276"}
Response: status= HTTP_201_CREATED
```

III. Results and Discussion

The implementation of a real-time attendance system uses a camera with a resolution of 1080p, a laptop, a router, and connecting cables, as shown in Fig. 5.

Testing is carried out using a two-person dataset following with ID as the name of each dataset folder, where the ID used is 1 and 6. In folder 1 is inserted a user image with ID 1 and folder 6 with user ID 6. Initially, face encoding is carried out. Hence, the pickle format data is generated. Next, the first thing to do is to set the Back-End, Front-End, and Node sections, respectively. Then, Node will run face recognition.

A. Single Person Test (Known vs Unknown)

In this section, tests are performed on faces that have been registered with the system. The face has ID = 6. The face that is detected and recognized by the system will be displayed on the website, as in Fig. 6(a), where the Known’s face captured by the video can be displayed smoothly on the website. The left-side image is the video display captured by the camera and processed using a particular script, whereas the right-side image is the display on the website. The data displayed is the most recent data up to 10 seconds ago. The data retrieval period can be changed as needed. The website display is updated (refreshed) every second.

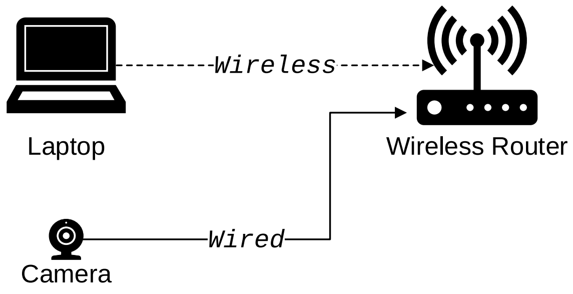

Fig. 5. System Implementation.

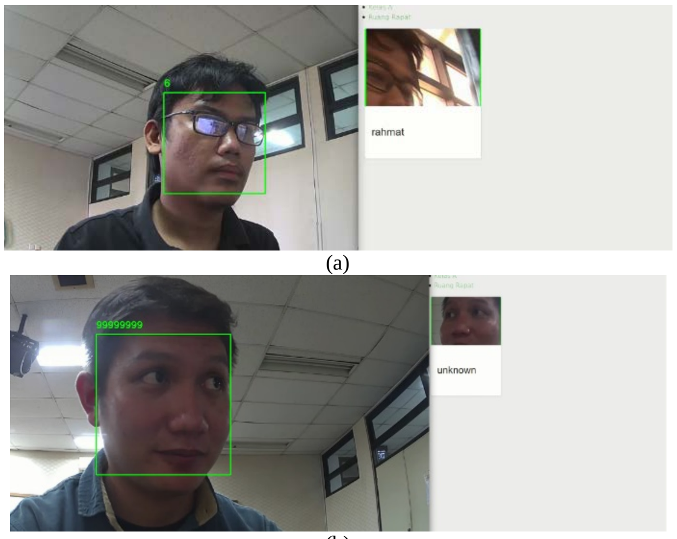

Fig. 6. Testing for: (a) One Known Person; (b) One Unknown Person.

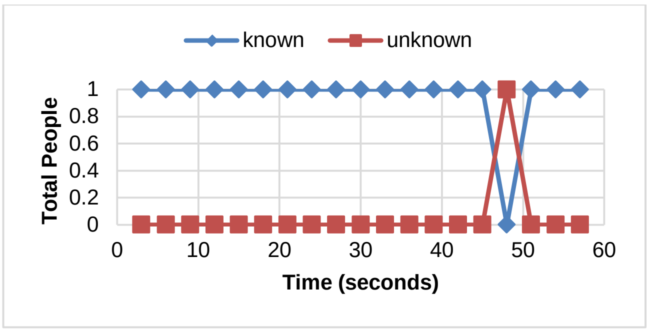

Fig. 7. Testing for One Known Person with a Data Interval of 3 Seconds.

Fig. 7 depicts a comparison chart between registered (Known) and unregistered (Unknown) person. Data collection was done every 3 seconds with 60-second intervals. The graphics obtained are quite stable. There should be two Known straight lines on line 1 and Unknown on line 0. It showed that at 48 seconds, an error has occurred on the detected face. The Unknown test is performed on a user with an Unknown identity as visualized in Fig. 6(b). The user's face does not exist in the data set and is not even registered to the existing system. Any faces that are Unknown, will be set their ID as 99999999.

In Fig. 8, we obtain a very stable graph. There is no wrong data at all. This situation indicates that data retrieval with a more extended period can achieve more stable data than shorter periods.

Fig. 9 and Fig. 10 exhibit the Unknown graph with sample periods at intervals of 3 seconds and 5 seconds, respectively. The detection of the Unknown person tends to be unstable; this is because the Unknown faces sometimes tend to be detected as a registered user. Since the dataset has no Unknown images, the system will tend to group Unknown faces into one of the existing datasets. This tendency is caused by the tolerance factor in the library used. The tolerance value is set to 0.5; if set to be smaller, the face will be more challenging to identify. When the two images are compared, then a graph with data capture at a larger interval shows better results.

B. Two People Test

This test involves two people, namely Known and Unknown in the same camera as shown in Fig. 11. The obtained known to the unknown test chart with data interval of 3 seconds and 5 seconds are delineated in Fig. 12 and Fig. 13, respectively. A graph that should be obtained is a straight line at the number = 1. In this situation, data retrieval at intervals of 3 seconds is better than 5 seconds.

Every detected unknown face will be stored in an "Unknown" folder at a node, as shown in Fig. 14. The obtained data are set at certain intervals depending on the determined configuration. The images that have been stored in that folder are then sorted by the admin and labeled as a new dataset.

C. Three People Test

This test employs three people, composed of two known faces and one unknown face, as visualized in Fig. 15. The result is depicted in Fig. 16. The expectation result is known as a straight line at number 2 and an unknown line at number 1. However, the results obtained are very oscillating. At that time, in 60 seconds, eight data were correctly known and five were utterly unknown. The results obtained in this study are still not stable in detecting the presence of Known and Unknown faces. This condition is probably caused by an algorithm that has not been able to handle large-scale users.

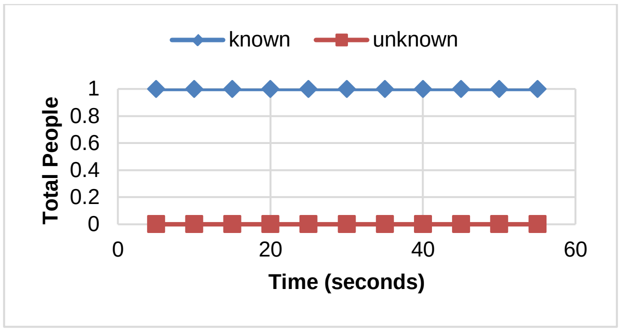

Fig. 8. Testing for One Known Person with a Data Interval of 5 Seconds.

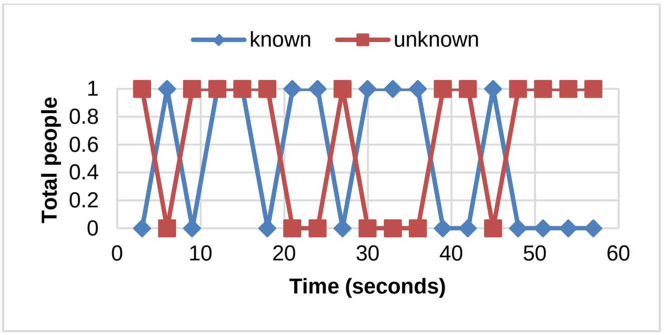

Fig. 9. Testing for One Unknown Person with a Data Interval of 3 Seconds.

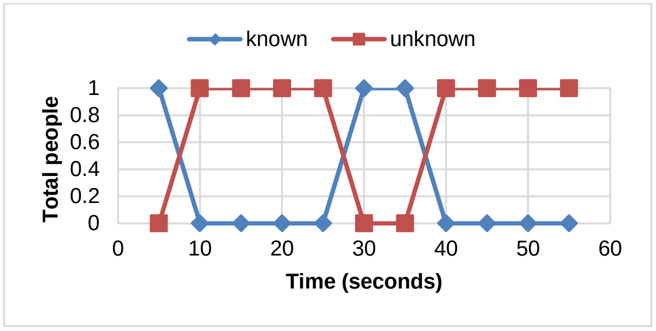

Fig. 10. Testing for One Unknown Person with a Data Interval of 5 Seconds.

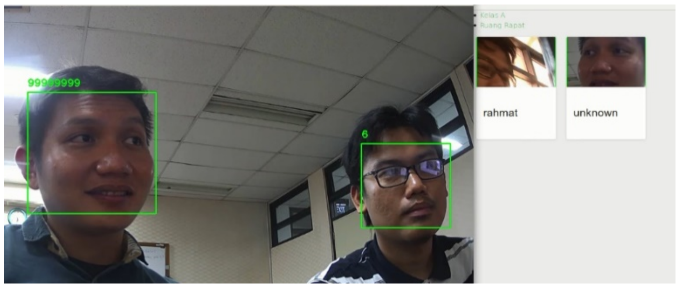

Fig. 11. Testing for Two People: Known Person vs. Unknown Person.

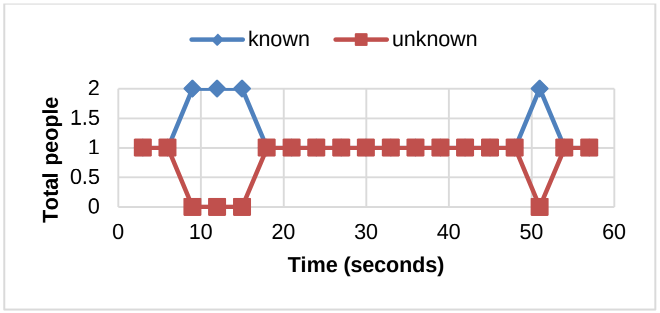

Fig. 12. Comparison Chart of Testing Two People, One Known and One Unknown, by Data Interval of 3 Seconds.

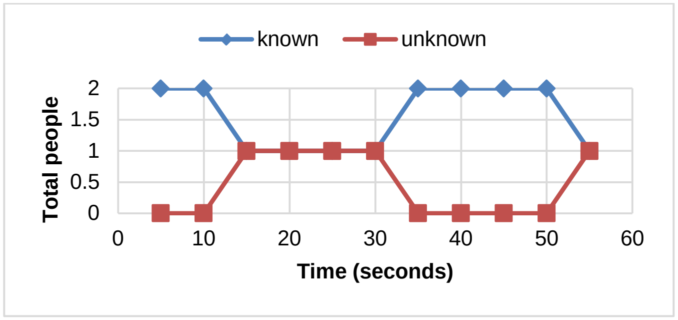

Fig. 13. Comparison Chart of Testing Two People, Known and Unknown, by Data Interval of 5 Seconds.

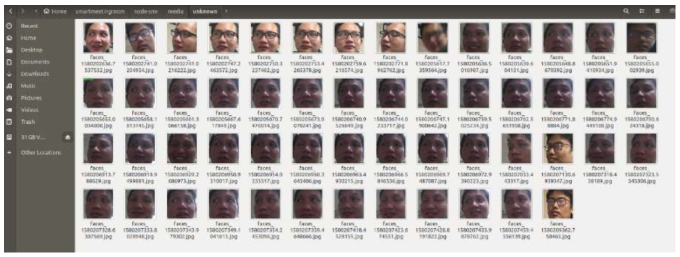

Fig. 14. Folder of Unknown Database in the Node.

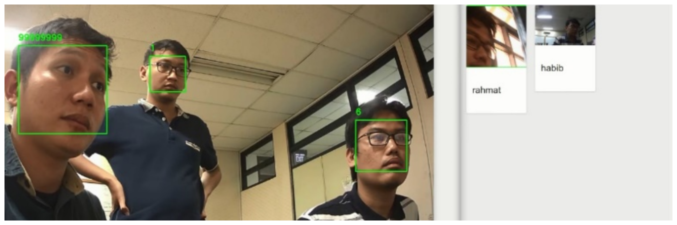

Fig. 15. Testing Three People.

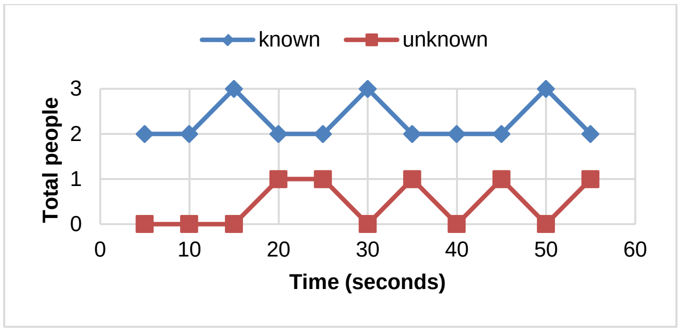

Fig. 16. Graph Testing Three People with an Interval of 3 Seconds.

IV. Conclusion and Future Works

In this work, we have developed a system that can be applied for smart meeting rooms and get results well functionally as expected. The system can detect faces, send data to the server, display the data, and save unknown faces in a folder on the node. If the detection is carried out on a registered user (known), then the system is quite stable. However, if the detection is done on an unregistered user (unknown) user is less stable. Instability result can occur when detecting the faces of three or more people in one camera. Also, there is a minimum distance to be able to detect faces accurately; which is about 2 meters.

Based on the summarized results, it is worth to improve the stability aspect in detecting more than three people for further research. Special algorithms will be applied in the system.

References

[1] J.M. Zhang, et al., "Machine Learning Testing: Survey, Landscaped and Horizons," IEEE Transaction on Software Engineering, 2020.

[2] A. Elmahmudi and H. Ugail, "Deep face recognition using Imperfect Facial Data," Future Generation Computer Systems, Vol. 99, pp. 213-225, October 2019.

[3] S. Sharma, et al., "Face Recognition System using Machine Learning Algorithm," Proc. of the 5th Int. Conf. on Communication and Electronics Systems, 2020.

[4] S. Fuada, H.A. Shiddieqy, and T. Adiono, "A High-Accuracy of Transmission Line Faults (TLFs) Classification based on Convolutional Neural Network," Unpublished.

[5] N. Kato, et al., "Ten Challenges in Advancing Machine Learning Technologies toward 6G," IEEE Wireless Communications, Vol. 27(3), pp. 96-103, June 2020.

[6] T. Adiono, "Challenges and opportunities in designing Internet of Things," Proc. of the 1st Int. Conf. on Information Technology, Computer, and Electrical Engineering, Nov 2014.

[7] S. Fuada, et al., "Workshop Internet-of-Things untuk Guru dan Siswa Sekolah Menengah di Purwakarta, Jawa Barat, Guna Menunjang Kompetensi Era Industri 4.0," Unpublished.

[8] E. Setyowati, et al., "Mesin Absensi RFID berbasis Internet-of-Things (IoT) untuk Meningkatkan Pengetahuan Siswa di Purwakarta terhadap Teknologi," J. Pengabdian Kepada Masyarakat (DIKEMAS), Vol. 3(3), pp. 67-74, 2019.

[9] K.P. Aji, U. Darussalam, N.D. Nathasia, "Perancangan Sistem Presensi untuk Pegawai dengan RFID berbasis IoT Menggunakan NodeMCU ESP8266," J. of Information Technology and Computer Science, Vol. 5(1), pp. 25-32, 2020.

[10] Gagandeep, et al., "Biometric Fingerprint Attendance System: An Internet of Things Application," Innovations in Computer Science and Engineering, pp. 523-530, 2018.

[11] M.D. Rahmatya and M.F. Wicaksono, "Design of Student Attendance Information System with Fingerprints," IOP Conf. Series: Materials Sci. and Eng., Vol. 662(2), 2019.

[12] T. Adiono, K.S. Prakoso, C.D. Putratama, B. Yuwono, and S. Fuada, "HOG-AdaBoost Implementation for Human Detection Employing FPGA ALTERA DE2-115," Int. J. of Advanced Computer Science and Applications, Vol. 9(10), pp. 353-358, 2018.

[13] D. Monzo, et al., "A Comparative Study of Facial Landmark Localization Methods for Face Recognition Using HOG descriptors," Prof. of the 20th Int. Conf. on Pattern Recognition, pp. 1330-1333, 2010.

[14] T. Adiono, K.S. Prakoso, C.D. Putratama, B. Yuwono, and S. Fuada, "Practical Implementation of Real-time Human Detection with HOG-AdaBoost in FPGA," Prof. of IEEE Region 10 Conf. (TENCON, pp. October 2018. DOI: 10.1109/TENCON.2018.8650453.

[15] S. Fuada, et al., "A High Frame-rate Cell-based HOG Human Detector Architecture and Its FPGA Implementation," Unpublished.

[16] D. Qu, et al., “An Automatic System for Smile Recognition Based on CNN and Face Detection,” Proc. of the IEEE Int. Conf. on Robotics and Biomimetics (ROBIO), pp. 243-247, 2018.

[17] B.T. Chinimili, et al., “Face recognition-based attendance system using Haar Cascade and Local Binnary Pattern Histogram Algorithm,” Proc. of the 4th Int. Conf. on Trends in Electronics and Informatics, June 2020.

[18] Suwarno and Kevin, “Analysis of Face Recognition Algorithm: Dlib and OpenCV,” J. of Informatics and Telecommunication Engineering (JITE), Vol. 4(1), pp. 173-184, July 2020.

[19] S. Sharma, et al., “FAREC — CNN based efficient face recognition technique using Dlib,” Prof. of the Int. Conf. on Advanced Communication Control and Computing Technologies (ICACCCT), pp. 192-195, 2016.

[20] S. Khan, et al., “Real-Time Automatic Attendance System for Face Recognition using Face API and OpenCV,” Wireless Personal Communication, Vol. 113, pp. 469-480, 2020.