A MOBILE APPLICATION TO MARK ATTENDANCE USING A COMBINED BACKEND OF THE FIRESTORE DATABASE AND AMAZON AWS SERVICES

Andy Jiang[1], Yu Sun[2]

[1]Klein Oak High School, 22603 Northcrest Dr, Spring, TX 77389
[2]California State Polytechnic University, Pomona,
CA, 91768, Irvine, CA 92620

ABSTRACT

Since the beginning of the COVID-19 pandemic, education largely shifted away from the physical classroom and towards more digitally oriented platforms. This simplified classroom attendance problems greatly, as newly created programming scripts could easily track the students in a meeting room via their names. However, with the recent growing return to in person education, it has become apparent that the problem of attendance within the context of a non-virtual classroom environment has yet to be solved in an efficacious automated fashion. In larger classrooms, the severity of this problem becomes exacerbated even further, as teachers are forced to allocate valuable time for the purpose of marking attendance. The flourishing world of machine-learning based algorithms were the first solutions that we considered, and within the context of the premise, we concluded that facial recognition would likely be the most feasible and effective approach that we could use. This paper develops a mobile application to apply real time face recognition for the purpose of the above stated problem, using a combined backend of the Firestore database and Amazon AWS services. Applying our application to in person classrooms, the results show that our solutions are immensely effective in both saving time and reducing error.

KEYWORDS

Machine learning, Flutter, Facial Recognition

1. INTRODUCTION

In most educational environments, teachers or educators will be required to record the attendance statuses of each individual student, a task whose difficulty scales with the size of the class or group being taught [1]. While it may seem like a minor inconvenience, teachers may lose up to 18 hours of instructional value per school year performing this meaninglessly routine task (provided that they are using a minute a day to record attendance, in 6 separate classes for 180 days). Taking this into account, a method of relegating this menial task to an automated software would provide a marked boost in the efficiency over the course of a single school year [3]. Additionally, this particular solution is not only limited to the bounds of a classroom environment, although that was its original purpose. Marking an individual's arrival to their workplace could be easily automated with this technology, with the benefit of improved security and the aforementioned boost in efficiency. With these wide ranging use-cases in mind, it is

David C. Wyld et al. (Eds): CSML, NET, BDHI, SIPO, SOEA- 2023
pp. 87-98, 2023. CS & IT - CSCP 2023 DOI: 10.5121/csit.2023.130108

simple to visualize the potential benefits of such a solution, which could systematically perform such a duty in real-time, both with greater machine-learning driven efficiency and the removal of the potential for human error.

Within the sphere of existing solutions within the space, there are two main schools of thought: 1. the use of expensive and invasive hardware that tracks attendance via fingerprints, or more uncommonly, retinal scans, and 2. spreadsheets, that still require manual input. In the case of the first solution, the problem remains that teachers and educators lack the budget to acquire such technology. Simply put, this is far too specialized a tool, and relies on proprietary hardware that is infeasible for most individuals to obtain. In the second, the main question is unsolved entirely, as the individuals are yet still required to perform this task by hand.

In this paper, we utilize Amazon Rekognition service to power our machine learning pipeline. Allowing for an efficient and powerful backend for the recognition of faces, we use this software in tandem with Google's Firebase backend to store existing images, resulting in nearly instantaneous results [4]. We intend to create a fully automated system for the purposes of classroom attendance, which thus far has not had any major similar solutions. The structure of our app is based on the 'classes', or groups of individuals, and when the facial recognition begins, our app starts an active attendance process in which each face that shows on the feed is marked present. Subsequently, the process can be manually ended, or is automatically ended if all the students have been shown to be present.

We evaluated the results of our solution through extensive testing of our facial recognition algorithm, achieving flawless results with a sample-size of 20 separate faces. We show the usefulness of our approach through comprehensive evaluation of accuracy through various light conditions (250 lux, 500 lux, etc.). This shows the efficacy and flexibility of our algorithm in various environments, displaying flawless precision in normal light levels, and high accuracy in more fringe light conditions, in which the camera may strain to detect more minute facial features. Additionally, an experiment was conducted measuring the accuracy of our application in indoor conditions vs outdoor conditions.

The rest of the paper is organized as follows: Section 2 gives the details on the challenges thatwe met during the experiment and designing the sample; Section 3 focuses on the details of our solutions corresponding to the challenges that we mentioned in Section 2; Section 4 presents the relevant details about the experiment we did, following by presenting the related work in Section

5. Finally, Section 6 gives the conclusion remarks, as well as pointing out the future work of this project.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. Workflow Design

The most immediate and pressing concern in our design was how to optimize the efficiency of our workflow, in such a way that did not excessively strain the computing power of the local machine on which the software was running on. Developing a quick yet user-friendly interface was paramount for the viability of such an application, as without one, it would likely prove even less useful than existing solutions.

2.2. Mobile Image Parsing Into Usable Format

As we designed our application for mobile platforms, we needed to parse the image input file format into a usable file format which we could store, use, and send to our database backend. Additionally, we needed to perform facial recognition in real time, and thus we needed to process the images in a fast and efficient manner. The images returned by the device cameras were given in the YUV420 file format, which could not be parsed by our facial recognition system [5]. Therefore, we could not use it until further processing, which became a concern of efficiency as well.

2.3. Optimizing When to Call Face Recognition

Traditional facial recognition models are taxing to run without a great deal of processing power. Therefore, it was not feasible to design a process in which we could run a facial recognition machine learning algorithm on a live real-time camera stream, as then the process would be applied to each and every frame of the video captured by the camera. We used the Amazon AWS Rekognition system to provide an accurate and efficient facial recognition backend [9].

3. SOLUTION

AWS's Rekognition algorithm is a cloud service capable of recognizing and labeling faces based on a previous library of labeled image inputs, returning a certainty value based on how similar the two faces are, using details such as the shape of certain facial features, the structure of the face, etc [2]. The mobile app allows for the creation of various 'classes', groups made up of individuals with a unique ID, name, and profile picture. Upon the selection of one of these classes in the face recognition page, a live video will begin, and any human face within the view of the camera will be detected, and the program will take a snapshot of the face. The detected face will then be returned to the cloud Rekognition system, where it will be compared to each person in the class. Based on this, the given list of students are either marked 'present', or 'absent', and afterwards, the process can either be manually ended, or be terminated automatically. The results are formatted simply, as a list of students that did not show up to the facial recognition algorithm.

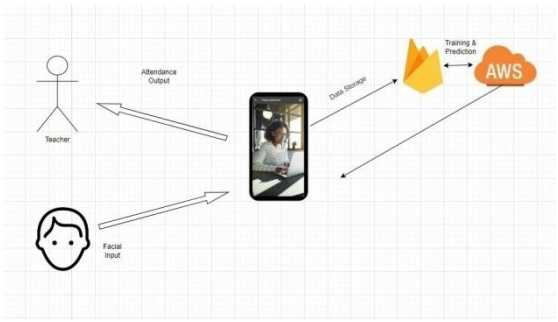

Figure 1. Overview of the solution

The core feature of this app is illustrated as follows:

3.1. Mobile App

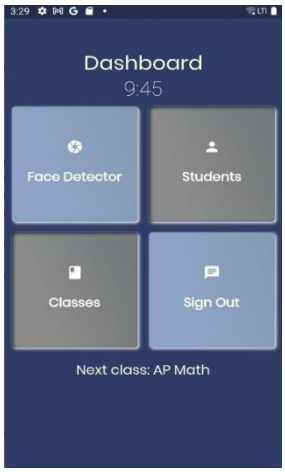

Figure 2. Dashboard page

The initial landing page shows a navigation menu in which the user can be transferred into one of three screens: a students page, a classes page, and a face detection page.

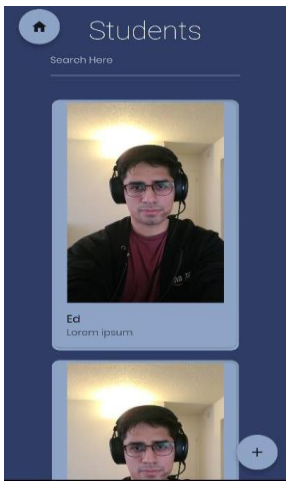

Figure 3. Student page

The students page displays the list of individuals stored within the database, with an option to add more.

Within the ‘Add Student’ page, there are two text fields in which you can enter the name and ID of a student, and a photo option. Upon submission, the information is pushed to the FireBase database, with the image being held within a generated URL. This data is stored within a ‘Students’ collection.

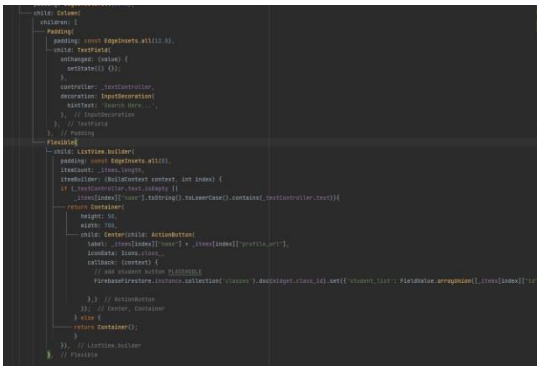

Figure 4. Screenshot of code 1

This code displays the Students in a ListView widget, pulling the information from the Firebase Firestore database and placing it in a Card widget for viewing purposes [15].

Figure 5. Class management page

Figure 6. Edit class page

The classes page similarly displays a list of classes, with the ‘Add’ page structured much the same way as the Students add page. Within the Classes database, each class is assigned a unique ID, and within each class is a list of student IDs, indicating the individuals comprising that group. Additionally, when a class is clicked on, you are taken to a screen in which you can edit the details of that specific class group.

Figure 7. Face detector

The face detection program is initialized by selecting a specific class, after which a live video camera feed will begin [10]. The UI consists simply of a display of the camera input stream. As soon as the face detection algorithm detects one or more individuals on the screen, then a Bounding Box will outline the detected face, then returning the use to a screen where their face will be returned to the AWS Rekognition server for processing. Based on the collection selected initially (the aforementioned class), the label returned will be compared to each individual within that group, and from there the ID that most accurately fits will be returned to the display for the user for confirmation. If confirmed to be correct, the app will mark that individual as present. If not, then the app continues the facial recognition process.

3.2. Face Detection

- The model we used
- How to integrate it in the app
- The special image format we used
- Get a code sample and explain the idea

```
Future<void> processImage(InputImage inputImage, CameraImage? image) async {
  if (isBusy) return;
  isBusy = true;
  final faces = await faceDetector.processImage(inputImage);
  print('Found ${faces.length} faces');
  print('Found test yusun');
  if (inputImage.inputImageData?.size != null &&
      inputImage.inputImageData?.imageRotation != null) {
    final painter = FaceDetectorPainter(
        faces,
        inputImage.inputImageData!.size,
        inputImage.inputImageData!.imageRotation);
    customPaint = CustomPaint(painter: painter);
  } else {
    customPaint = null;
  }
  isBusy = false;
  if (mounted) {
    setState(() {});
  }
```

Figure 8. Screenshot of code 2

For the face detection portion of the process, we used a classification model that utilized the

Tensor Flow framework. The model was stored locally on the app, allowing for greater efficiency and accuracy. The code sample above shows how we called the facial recognition function. When a face is detected, a bounding box is drawn around it, and the information is returned to the next screen, in which we return the face to the Amazon AWS Rekognition system [12].

3.3. Face Recognition

```
void loadAll() {
  _items = [];

  FirebaseFirestore.instance.collection("students").get().then((value) {
    value.docs.forEach((element) {
      print("ADDING ITEMS");
      print(element.data()["id"]);
      _items.add(element.data());
    });

    setState(() {});

    for (var item in _items) {
      if (item["id"] == widget.studentName) {
        print("this works!");
        profilePhoto.add(item["profile_url"]);
      } else {
        print("this does not!");
      }
    }
  }).catchError((e) {
    print("Failed to get the list");
    print(e);
    throw e;
  });
}
```

Figure 9. Screenshot of code 3

The backend of this application essentially consists of two working parts: the FireStore cloud database, which is used to store the necessary information (i.e. faces, names, IDs), and the Amazon AWS Rekognition system to power the face recognition aspect of the application [8]. Our app returns the detected face into the Rekognition system, which sends back the student ID of the detected face. The app then checks the Firebase database for the aforementioned student ID, and then displays the student information on a card, which the student may confirm or deny to be them.

4. EXPERIMENT

4.1. Experiment 1: The Accuracy of Face Detection

Run the face detection with different scenarios

1) day time vs night time
2) indoor vs outdoor
3) single face vs multiple face
4) still vs motion

The experiment conducted below measured the amount of ambient light in the surrounding environment.

Primarily, the main area of concern was that the accuracy of our algorithm was not accurate enough to be sufficiently useful for our purposes. To test the aforementioned accuracy of our algorithm, we put our app through a variety of different light conditions, ranging from 250 lux to 1500 lux, in order to see the accuracy.

4.2. Experiment 2: The Accuracy of Face Recognition

Run the face detection with different scenarios:

1. day time vs night time
2. indoor vs outdoor
3. single face vs multiple face
4. still vs motion
5. confusion with different sets of users (5, 10, 20, etc.)

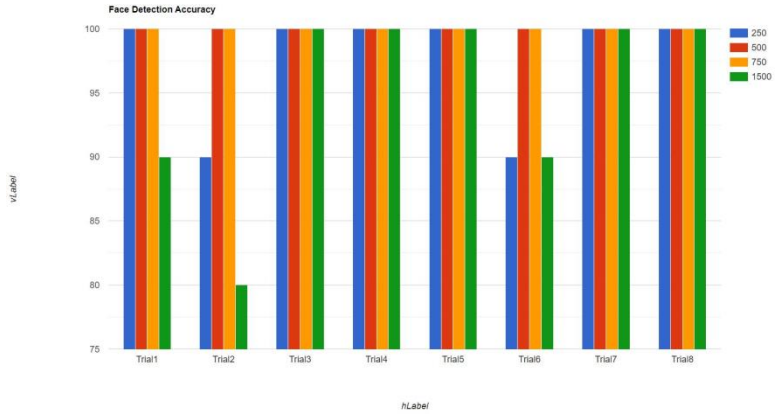

Figure 10. Face detection accuracy 1

The face detection algorithm scores perfect accuracy across all trials in 500 and 750 lux, but during 1500 lux, the camera occasionally suffers from overexposure with too much light.

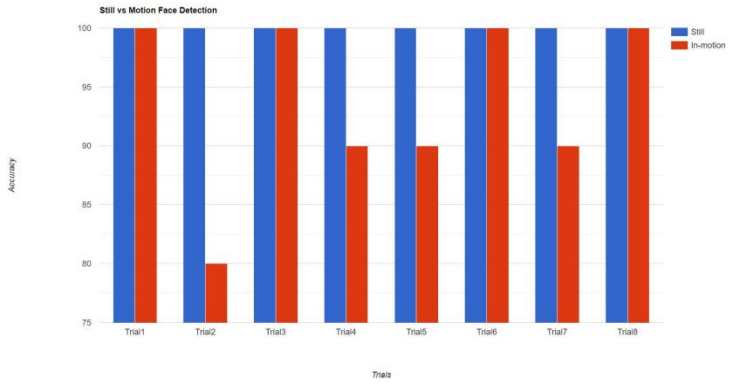

Figure 11. Still vs face motion detection

We conducted an experiment in which we had one individual step into the frame of the camera and pause, making direct eye contact with the lenses, then had that same individual walk through the frame of the camera at a slower walking pace. Our face detection algorithm achieved perfect results with the first still detections, but suffered somewhat from a decrease in accuracy when the individuals stayed in motion.

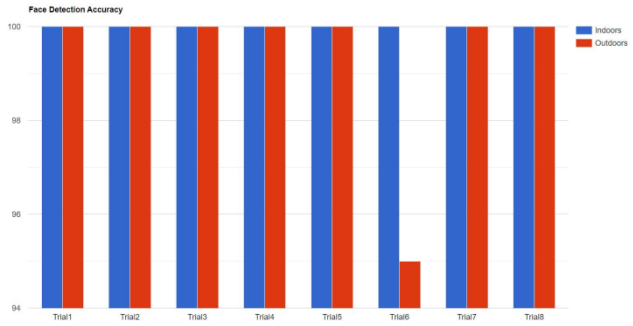

Figure 12. Face detection accuracy 2

Between indoor and outdoor environments, there was not a significant amount of variance of accuracy. Each trial was composed of twenty different tests, in which an individual would walk into the view of the camera, then exit after approximately one second.

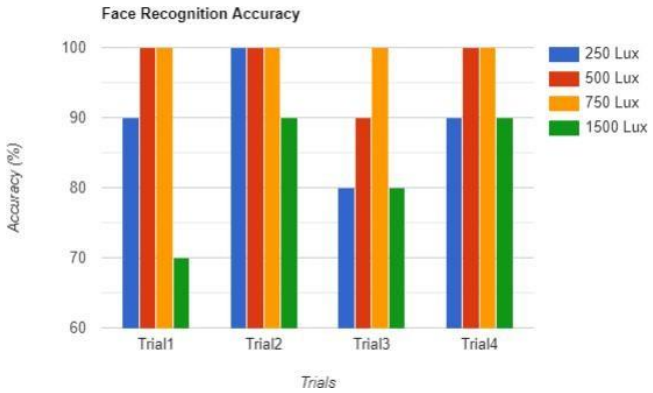

Figure 13. Face recognition accuracy

Within normal light conditions (500 to 750 lux), the application was capable of consistently providing accurate and precise results in an expedient manner, returning a marked total of ten correct test cases out of ten test cases. At higher light conditions, the camera began to lose image fidelity , and thus, the algorithm began to decrease in accuracy, with its lowest dipping down to 70% on the first trial [13]. However, the algorithm was still able to consistently return the correct label to match with the corresponding face. At the lowest light level of 250 lux, it appeared that the facial recognition algorithm began to lose sight of key features whilst some of the darkness obscured the face.

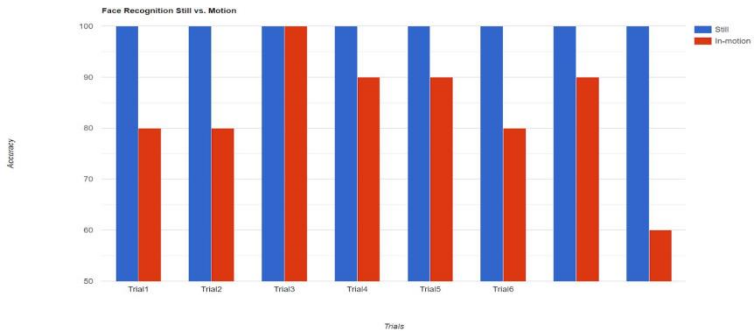

Figure 14. Face recognition still vs motion

While our face recognition feature was able to perform perfectly across all 8 trials, it showed a significant decrease in accuracy when the recipients were actively moving. Our algorithm could not consistently distinguish facial features when the sample snapshots provided were blurred, causing its drop in performance.

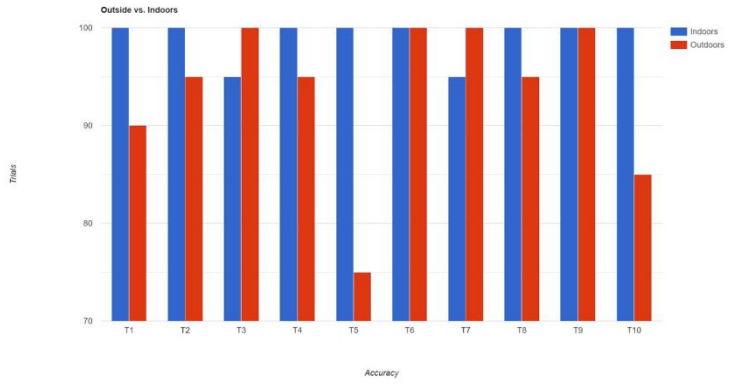

Figure 15. Outdoor vs indoor accuracy

We performed an additional experiment where the trials were performed indoors and outdoors. The results proved that the algorithm had acceptably high accuracy scores indoors, but struggled relatively heavily whilst outdoors. This may have to do with the time at which the experiments were performed, in which the light levels outdoors were significantly lower than that of the indoor trails. Still, the algorithm was able to perform relatively well.

This experiment proves that overall, taking into assumption that this application will most normally be used in normal light level conditions, this application is a sufficiently accurate solution for the purposes of classroom attendance. However, in certain low or high light conditions, the algorithm becomes significantly less accurate.

5. RELATED WORK

In this paper, the authors demonstrate the efficacy of this approach by combining two cutting edge feature extraction techniques (DWT and DCT), then subsequently applying a radial abscess function for the purposes of classifying faces within a classroom setting, which was met with a high success rate [6]. The methods outlined in their work were shown to be highly effective and efficient, saving a significant amount of time with around 80% accuracy.

Here, the authors use a Raspberry Pi-based solution that runs an LBPs facial recognition system on a Linux operating system, then storing the results of the attendance process onto a MySQL server [7]. This solution resulted in a high degree of success in a dataset of 11 faces, coming out to a 95% accuracy metric overall. Our work largely centered around a software-based approach, developing an app to allow for a more accessible solution.

In this paper, the authors tackle the theoretical aspects of this problem, exploring the implications of the implementation of such systems both managerially and logistically [3]. In essence, it provides an analysis of data collected from the implementation of a facial recognition system in classrooms within a university setting, and the advantages of automatic attendance systems replacing traditional manual attendance systems.

6. CONCLUSIONS

In this paper, we propose a novel approach to automating classroom attendance through a mobile application-based facial recognition algorithm, using a Firebase backend in tandem with AWS's Rekognition service. We conducted numerous experiments to assess the usefulness and efficiency of our solution in various conditions, testing both the face detection and the facial recognition portions of our program in different light conditions, environments, and movements. Overall, our experiments were able to verify the effectiveness of our application, resulting in significantly accurate scores that showed satisfactory efficacy [14].

As it stands, our current models of retrieving and processing data remain relatively inefficient. The facial recognition process manually searches through the entirety of the class, which may be optimized further. Additionally, portions of the UI can be made to be more accessible and easy to use. The facial recognition algorithm can be improved to accommodate further for low and high light conditions to improve accuracy and ease of use, which may be essential to improving the versatility of this app.

Implementing more efficient search functions for the purposes of quicker loading times may be done with reducing the number of search queries [11]. The UI can simply be made more streamlined in the future by optimizing the workflow. Transitioning from prebuilt facial recognition algorithms to customized facial classification machine learning algorithms mayprove vital in the pursuit of higher accuracy metrics within fringe light conditions.

REFERENCES

[1] Saparkhojayev, Nurbek, and Selim Guvercin. "Attendance Control System based on RFID-technology." International Journal of Computer Science Issues (IJCSI) 9.3 (2012): 227.

[2] Mishra, Abhishek. Machine Learning in the AWS Cloud: Add Intelligence to Applications with Amazon SageMaker and Amazon Rekognition. John Wiley & Sons, 2019.

[3] Tee, F. K., et al. "JomFacial Recognition Attendance Systems." International Conference on Emerging Technologies and Intelligent Systems. Springer, Cham, 2021.

[4] Moroney, Laurence. "The firebase realtime database." The Definitive Guide to Firebase. Apress, Berkeley, CA, 2017. 51-71.

[5] Ma, Changyue, et al. "A study of deep image compression for YUV420 color space." Applications of Digital Image Processing XLIV. Vol. 11842. SPIE, 2021.

[6] Lukas, Samuel, et al. "Student attendance system in classroom using face recognition technique." 2016 International Conference on Information and Communication Technology Convergence (ICTC). IEEE, 2016.

[7] Salim, Omar Abdul Rhman, Rashidah Funke Olanrewaju, and Wasiu Adebayo Balogun. "Class attendance management system using face recognition." 2018 7th International conference on computer and communication engineering (ICCCE). IEEE, 2018.

[8] Wingerath, Wolfram, Norbert Ritter, and Felix Gessert. "Real-Time Databases." Real-Time & Stream Data Management. Springer, Cham, 2019. 21-41.

[9] Jung, Soon-Gyo, et al. "Assessing the accuracy of four popular face recognition tools for inferring gender, age, and race." Twelfth international AAAI conference on web and social media. 2018.

[10] Singh, Anubhav, and Rimjhim Bhadani. Mobile Deep Learning with TensorFlow Lite, ML Kit and Flutter: Build scalable real-world projects to implement end-to-end neural networks on Android and iOS. Packt Publishing Ltd, 2020.

[11] Balke, Wolf-Tilo, Jason Xin Zheng, and Ulrich Güntzer. "Approaching the efficient frontier: cooperative database retrieval using high-dimensional skylines." International Conference on Database Systems for Advanced Applications. Springer, Berlin, Heidelberg, 2005.

[12] He, Yihui, et al. "Bounding box regression with uncertainty for accurate object detection." Proceedings of the ieee/cvf conference on computer vision and pattern recognition. 2019.

[13] Emami, Shervin, and Valentin Petrut Suciu. "Facial recognition using OpenCV." Journal of Mobile, Embedded and Distributed Systems 4.1 (2012): 38-43.

[14] Raghuwanshi, Anshun, and Preeti D. Swami. "An automated classroom attendance system using video based face recognition." 2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT). IEEE, 2017.

[15] Chock, Margaret, Alfonso F. Cardenas, and Allen Klinger. "Database structure and manipulation capabilities of a picture database management system (PICDMS)." IEEE Transactions on Pattern Analysis and Machine Intelligence 4 (1984): 484-492.