Access Control Using Deep Learning

Geng LiangYu   
MTech (IS) Year 2 (PT)  
NUS (ISS) Singapore  
e0384909@u.nus.edu

Tan Chin Gee  
MTech (IS) Year 2 (PT)  
NUS (ISS) Singapore  
e0384927@u.nus.edu Ong Boon Ping  
MTech (IS) Year 2 (PT)  
NUS (ISS) Singaporee0384803@u.nus.edu

Han Dongchou Francis  
MTech (IS) Year 2 (PT)  
NUS (ISS) Singapore  
0385045@u.nus.edu

Sponsor: Han Dongchou Francis Senior Director Oracle Solution Centers (JAPAC)

### Project Supervisor: Dr Zhu Fangming Senior Lecturer & Consultant, Artificial Intelligence Practice NUS ISS

Cao Liang  
MTech (IS) Year 2 (PT)  
NUS (ISS) Singapore  
e0384184@u.nus.edu

*Abstract*—An Access Control solution is proposed to protect an important asset for the main Singapore office of Oracle Corporation. The asset concerned is a Smart City Demo Asset that is displayed at the reception area of the office. The solution deploys Deep Learning technology and it differentiates authorized personnel from other people when they are in close proximity to the asset. The reason for such a security system is that the demo asset concerned is a setup with parts that are (1) delicate, (2) have a certain cost, and (3) integral to the functioning ability of the whole demo.

# Introduction

A Smart City Demo Asset that is currently being built and targeting for completion by July 2020 is to be placed at the reception area of the Oracle office. The project plan is given in Appendix A. This area is frequented by more than a thousand people a day and the demo asset needs to be protected against damages and theft from unauthorized access. However, Oracle would like to retain the esthetic nature of the office, and so no physical barrier can be deployed in this area (Fig 1). As a result, we are proposing the use of Deep Learning to recognize authorized personnel who can access the asset. An alert mechanism will be set up to inform stakeholders of unauthorized access when it happens, in real time. An intruder detection will also be set up to activate this system as soon as someone walks near the restricted area.



Fig 1: Reception Area of Oracle Office

# High Level System Flow

The overall system comprised 3 modules: Intruder Detection module, Face Recognition module, and Backend Service module. All these modules were running as separate python process, and backend service module communicated with another 2 modules through HTTP restful API services.

## Intruder Detection Module System Flow

When the intruder detection module detected the suspicious intruder status, it informed the backend service module by invoking the backend service restful HTTP API (Fig 2).



Fig 2: Intruder Detection Module System Flow

## Face Recognition Module System Flow

The Face Recognition module invoked the backend service API, and queried the backend service for any suspicious intruder status. Once there was any reported suspicious intruder status, the face recognition module started to detect and identify the person’s face. If the face was not recognized, the face recognition module sent out SMS / Email alert with the backend service (Fig 3).



Fig 3: Face Recognition Module System Flow

## Backend Service Module System Flow

Here is a figure of the Backend Service module (Fig 4).



Fig 4: Backend Service Module System Flow

## D. System Integration Architecture

The 3 modules were integrated into a system as illustrated below (Fig 5).



Fig 5: System Integration Architecture

## E. Security Access System Flow

The entire system of the Smart City Demo Asset that is currently targeted for deployment has a security access system flow as illustrated below.

1. System Flow Description
   1. System Setup
      1. CAM1 – Camera 1 for Intrusion Detection pointing at an immediate proximity area
      2. CAM2 – Camera 2 for Face Recognition pointing forward
   2. User Setup
      1. People authorized to work on the Smart City Demo asset
      2. Train 3 people, along with some negative samples
   3. Intrusion Detection
      1. Idle State
         1. CAM1 – idle
         2. CAM2 - idle
      2. Presence Detected
         1. CAM1 – when someone gets very close to the asset, trigger CAM2 Face Recognition
      3. Start Face Recognition
         1. CAM2 – triggered and run Python program
         2. Check if the detected person is one of the 3 authorized people, or unauthorized
         3. Capture the face of the unauthorized person
         4. Output the result
         5. Name of the authorized person OR Face of the unauthorized person
         6. Alert Mechanism
      4. Alert
         1. Send SMS to asset owner (94389636)

State the following message

* Detection Alert
* Unauthorized person: <date>, <time>
  + - 1. Send email ([francis.han@oracle.com](mailto:francis.han@oracle.com))

State the following message

* Detection Alert
* Unauthorized person: <date>, <time>
* Attach picture of unauthorized person
  + - 1. Append message into log file

1. System Flow
   1. Intrusion Detection Flow

Purpose: Check whether there is any suspicious object for intrusion and report detection

Proposed API:

# Parameters:

# frame: Open CV captured frame image or video file frame image

# Return: List of detection result and mask image; Detection result is True if intrusion is detected, and False if not detected

Function: def perform\_intrusion\_detection(frame)

* 1. Face Recognition Flow

Purpose: Perform face recognition and retrieve list of recognized and unknown face name and images

Proposed API:

# Parameters:

# frame: Open CV captured frame image or video file frame image

# Return: List of face name, face location (x, y, w, h) and face image; Unrecognized face name is “unknown”. E.g., [[“unknown”, [10,10,200,200], unknown\_face\_image], [“people1\_name”, [15,15,200,200], people1\_face\_image], [“people2\_name”, [25,25,200,200], people2\_face\_image]]

Function: def perform\_face\_recognition(frame)

* 1. Alert Message Flow

Call AlertClient API:

1. Send SMS message

send\_sms(sms\_message, target\_phone=94389636)

1. Send email

send\_email\_with\_images(email\_content, subject\_message, alert\_type=’Notice’, image\_name\_prefix=’image\_’, image\_list=[])

# System Design

## Detecting a presence

When a person appears within the proximity of the demo asset, the security system will be turned on. This is made possible with a camera and an accompanying algorithm. The algorithm runs a background segmentation code and is constantly on a lookout for any change in the image. When a change happens, it triggers the face recognition algorithm to start operating.

## Recognizing the Face

The input is a video stream captured by one camera with a 4K resolution. The OpenCV library is employed to capture the video stream and to process the image frames. The captured images are transformed into Numpy arrays for processing. When recognition is done, the result is then shared with some peripheral systems.

In order to recognize the face from the image captured, a classifier is required. To enable fast classification, encoding of the face detection is also required. In this paper, face recognition was using pre-trained model to encode the face and then recognize the face using classifier (Fig 6).

Fig 6: Face detection and recognition process

Classifier requires training before it can be used to predict the face as belonging to which person of interest. Hence, sampling and training must be done (Fig 7).



Fig 7: Face detection and recognition model training

# Face Sampling and Augmentation

In this context, face sampling consists of collection of images that belong to person of interest and negative targets.

Images were taken from 3 persons of interest in various angles, light exposure and expression. Doing this will improve the accuracy and precision of the model.

Images were also taken or collected from non-targeted persons; this was aimed at reducing false positive cases.

However, only limited number of samples can be collected. Eighty-two images were collected for targeted person and 37 non-targeted images were collected. Augmentation was used to increase the number of samples.

In this context, the images were flipped horizontally, rotated or translated. To further enhance the model accuracy, gaussian blur and Laplacian of Gaussian filter were used on existing images and generated blurred or sharpened images.

As a result, 102,750 images were generated and used for training process.

# Solution

The system can monitor multiple faces on the image frame. Apart from the OpenCV2, the Haar Wavelet face cascade method (standard function used in OpenCV2) was used to capture the human face pixels. Also, Multi-Task Cascaded Convolutional Neural Network (MTCNN) was used to refine the face area captured using Haar Wavelet. Image capturing was disallowed if MTCNN cannot capture eye/nose/mouth location.

## Constraints

Lighting can be different between the testing ground and training ground. Bright lighting (which is mainly yellow lighting that has significant R and G component) can reduce the contrast between the face and surrounding environment. Face area which was not exposed to the light may not be recognized.

Though targeted face is known, the application ground is an open space that allows free movement. Hence, accuracy varies between individual visitor.

For the targeted face, not all facial expression/hair style can be captured. Spectacle effect can also be an extra factor to the accuracy of recognition.

## Face Detection - Haar Face Cascade

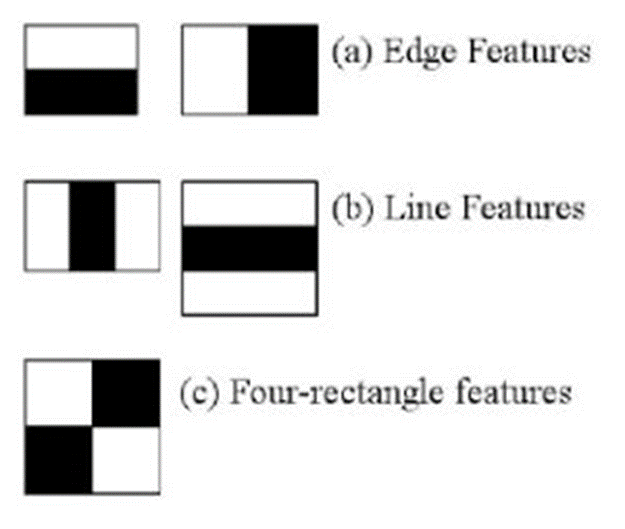
In order to allow fast face detection, OpenCV Haar Face Cascade was used.

Haar Cascade [1] has 4 stages:

1. Haar Feature Selection (Fig 8)
2. Creating Integral Images
3. Adaboost Training
4. Cascading Classifiers

For each feature, we tried to find out what is the best threshold that can predict whether the trained images are human faces or not.

However, not all features obtained through integral images were useful. The model was applied on the features that were considered to be a weak classifier as each of them only held information at a certain spatial region of the whole image.

Fig 8: Haar Feature Selection

Adaboost classifier is actually an ensemble classifier of the weak classifier that was applied on each Haar feature. Boosting allows stronger feature classifier to be ensembled as a weighted averaged strong classifier.

Finally, the Adaboost classifiers were cascaded in stages. As a result, the image area that fitted the face features the most were highlighted.

## Face Landmark Detection and Alignment - MTCNN

MTCNN [2] was used for face landmark detection and face boundary detection.

MTCNN has 3 stages of neural networks.

At 1st stage, the input image was scaled down multiple times to build an **image pyramid.** Each scaled version of image pyramid was passed through its convolutional neural network (CNN) that is known as Proposal Network (P-Net).

P-Net uses 12x12 kernel with strides 2 to process input image. P-Net will produce the bounding box for each 12x12 kernel. The resultant bounding boxes that have higher probability (generated by P-Net as face classification result) will be parsed to the next stage.

At the 2nd stage, all the candidate bounded images from stage 1 were fed into CNN that is known as Refine Network or R-Net. 2nd stage uses 24x24 kernel on the inputs.

At each 24x24 kernel output, CNN processes another set of bounding boxes and probabilities. So, we were able to reject some of the candidate images from stage 1. This was how the 2nd stage refined 1st stage result.

*a**b* 

At 3rd stage, 48x48 kernel was applied on 2nd stage candidate bounded images.

While producing new bounding coordinates and probability of fitting the face feature, it also computes the face landmark coordinate.

Five face landmark points (left/right eye, left/right mouth, nose tip) were computed. The 5-points were obtained though minimization of Euclidean loss.

Best bounding coordinate and 5 face landmark coordinates were selected. Hence, 3rd stage is known as O-Network or output network (Fig 9).

OpenCV Haar Face Cascade is based on Haar features. Hence, it is subjected to error where features of other objects fit the threshold given.

MTCNN can identify the face landmark (eye/mouth/nose) of an image. When a Haar Cascade image is processed by MTCNN, we can reject the image captured from Haar Cascade from further processing if face landmark coordinates are not returned from MTCNN.

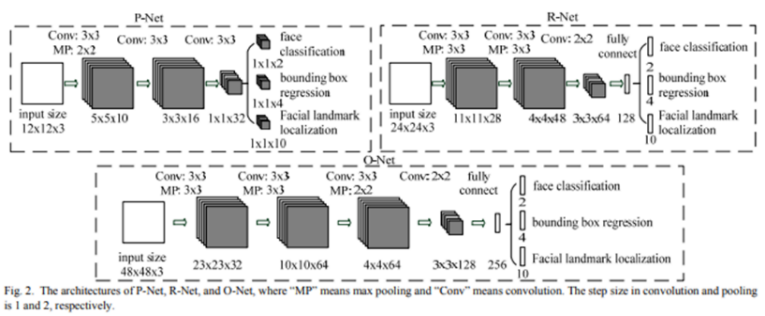


Fig 9: Architectures of P-Net, R-Net, O-Net

In this project, the face image pixels within the MTCNN boundary were captured (MTCNN outputs boundary coordinate instead of image).

These identified face image pixels were resized into (160, 160). The pixels which are too far away from left eye/right eye will be cropped. Forehead and pixels under mouth will also be cropped. This will reduce possible noise due to varying hairstyles and background.

Based on left and right eye position, alignment was made so that left and right eye is within 10 degree to the horizontal axis.

## Face Encoding – DLIB based face\_recognition

Face encoding was done through face\_recognition python library developed by Ageitgey [3]. This library is mainly based on open source Dlib. It returns result in 128-bit code which is suitable for face encoding.

The face\_recognition process has several steps to refine and encode the face image:

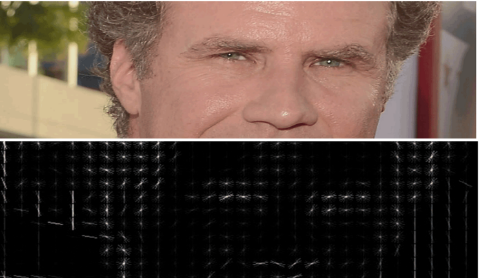
* HoG transformation (further refine bounding area from MTCNN in this project) [4]
* dlib face landmark detection and face projection
* neural network trained using triplet loss for error computation

Histogram of Gradient is an 2D array that consists of gradient at each pixel. Gradient was to be computed from intensity difference across its neighbouring pixels.

Face Cascade and MTCNN should have filtered out images which were not showing sufficient face features at this point. HoG will refine the bound of image by taking out more background pixels.

HoG was working since nose, mouth, eyes should have unique gradient features and relative distance across nose, mouth, eyes, hair and background, that will define the face area (Fig 10).

HoG is less sensitive to light intensity (Gradient usually considers surround pixels rather than light intensity at one pixel) and can define the features without the influence of local light intensity. (MTCNN, face\_cascade should have taken out image with extreme light intensities, so less intensity will not A close up of a person

Description automatically generatedhave side effect). DLIB provides a fast computation of HoG (Fig 11).

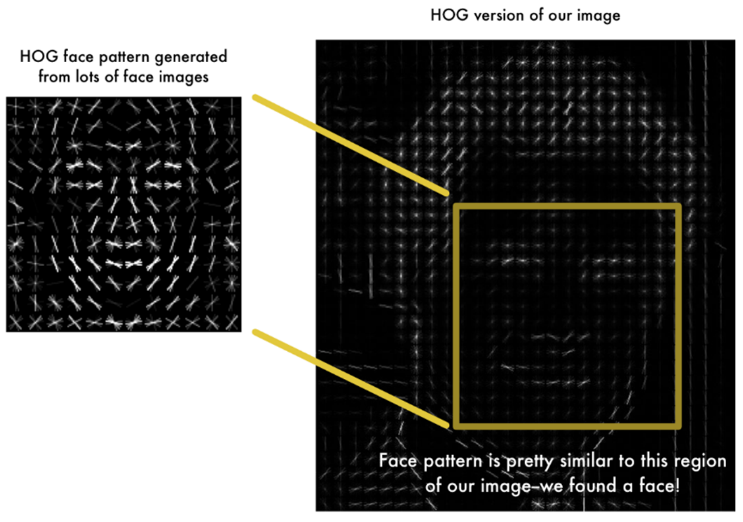
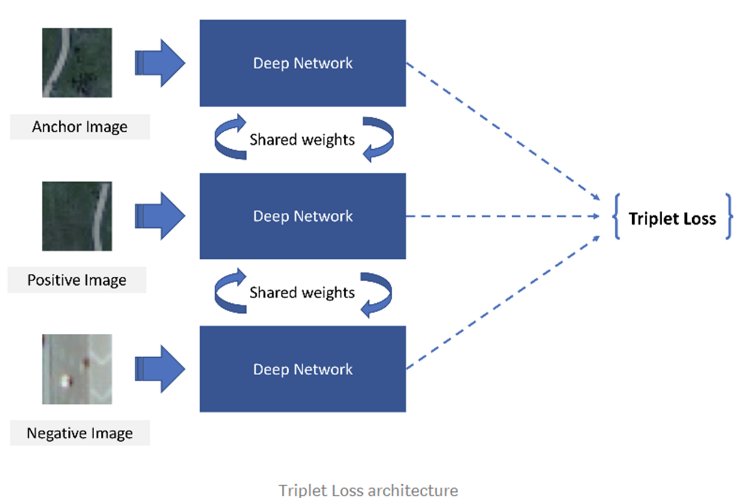
Fig 10: HOG Representation

Fig 11: HOG Based Face Recognition

A picture containing strainer, rain

Description automatically generated With HoG refined image, Dlib is able to output 68 landmark points (Fig 12).

Fig 12: 68 Landmark Points

Dlib has a neural network pre-trained with 68-point [**iBUG 300-W dataset**](https://ibug.doc.ic.ac.uk/resources/facial-point-annotations/) **[5]**

With Dlib’s landmark output, face\_recognition library can make face projection (Fig 13).

Slanting face can be re-projected with all the 68 points at the largest alignment possible. Nose was centered while the image was re-projected based on direction on other landmark points as well. (Compared to MTCNN with 5 points, total re-projection is possible).

Fig 13: Face Re-projection

With Dlib face re-projection, Ageitgey was able to train a neural network with a much smaller sample size. It defines the input to the network to have the right face alignment. Ageitgey’s network is computing triplet loss for error correction (Fig 14).

Fig 14: Triplet Loss Network Architecture

Triplet loss computation [6] [7] means three images are input into the network at the same time. Though there are three images, all the images are going through shared layers (this means all three sub-networks are exactly the same layers with the same weight).

The 3 images are training face images of a known person, another picture of the same known person and a picture of a totally different person.

With this combination, the weight between similar faces are closer while the weight between different faces are further apart. This applies to all 128-bits that represent 128 measurements made on the face features.

A close up of text on a white background

Description automatically generatedWith pre-trained weights from the face\_recognition python library, up to 98% accuracy is possible (Fig 15).

Fig 15: Triplet Loss Explained

# Classifier

At the end of the face recognition process, an ensemble of machine learning algorithms was used to further augment the accuracy and consistency. This ensemble included the KNN, SVM, Logistic Regression and MLP. A voting mechanism was then employed for the ensemble.

## KNN

#### KNN is suitable for face recognition. In this context, Minkowski distance when defining the nearest neighbor. The accuracy is enhanced with weight reduced with the distance increased.

#### As a result, using 9:1 train-and-test split on 102,750 samples, the sklearn KNN model accuracy can reach 99.3-99.8% in the dataset with number of neighbor set to 5.

## SVM

SVM is also commonly used for face recognition.

Transformation of the 128-byte code enables support vector plane to be found.

With good image preprocessing, Dlib based face recognition library usually encodes the image accurately. This allows SVM training by using 128-byte code.

Using 9:1 train-and-test split on 102,750 samples, SVM usually gives accuracy around 98.9-99.3%.

## Logistic Regression

##### Logistic regression is suitable for binary classification using probability function.

##### After image was encoded into 128-byte code through Dlib based face recognition library, logistic regression model is built for each targeted face.

##### Logistic regression is not a linear regression model and so it allow a better binary class prediction using 128-byte code.

Using 9:1 train-and-test split on 102,750 samples, Logistic Regression usually gives accuracy around 96.9-99.3%.

## Multi-layer Perceptron

##### Multi-layer Perceptron is a multi-layer neural network formed by perceptron. Each perceptron will have a bias, and multiple inputs with trainable weight.

##### A hundred and twenty hidden layers were used in this context using lbfgs solver. lbfgs solver is optimizing a log loss function.

##### Hence, it was able to provide different classification result compared to other networks.

Using 9:1 train-and-test split on 102,750 samples, Logistic Regression usually gives accuracy around 97.4-99.5%.

## Voting

##### Voting ensemble model was used on SVM/KNN/MLP/Logistic regression. This was because all the 4 models are having binary outputs.

##### Since there are 4 targeted faces, we will perform voting for each targeted face independently. If an image is recognized to be 2 or more targeted faces, it will not considered to be recognized by the model. This was to prevent close ambiguity.

By assigning each model to have equal vote, hard voting was made. The training result on 102,750 images showed 99.3%-99.9% accuracy.

# Result

User acceptance test is given in Appendix B. Training results show that voting ensemble gave higher accuracy than individual model on person of interest 2 and person of interest 3. For person of interest 1, KNN gave higher accuracy (Table 1).

In terms of false positive rate, voting network gave less false positive on person of interest 3. Voting ensemble model also gave close result to the best model (Table 2).

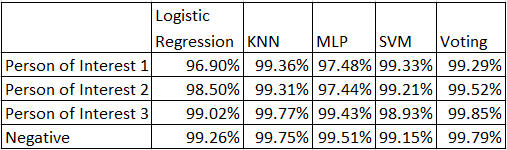


Table 1: Accuracy of classifier models

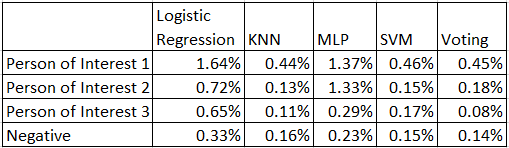


Table 2: False positive rate of classifier models

During application, voting model and weight were loaded. The face image was captured and gone through all the preprocessing through face cascade, MTCNN and DLIB based face\_recognition encoding, before making prediction using voting model.

The model gave good prediction when eyes, mouth, nose were captured in the image. The model prediction is using 0.1s per frame.

# Alert Mechanism

The alert mechanism utilizes the messaging services of Twilio, a cloud communication company and Gmail. The backend service sent SMS messages using the API call provided by Twilio. Gmail was tapped as the platform for sending email alert.

Two scenarios were tested with the email and SMS alert triggered by the face recognition module on detecting an unknown face. There was a 100 percent success rate for sending the alert with a low latency of less than a few seconds in both scenarios. The alert was not sent as intended under the scenario when the face recognition module detected a known face.

# Project Limitations

## Masked Faces

Masked faces suddenly became a common sight because of the Covid-19 pandemic. This was an unexpected development that caught the development of the project by surprise. By this point, the development of the system to recognize unmasked faces was well underway, leaving the team with no choice but to continue development with this limitation not addressed.

## Camera Blind Spot

The system has a second camera mounted on the wall that recognizes the faces of people who had trespassed beyond the designated boundary placed in front of the Smart City demo. However, the wall-mounted camera was found to have blind spots that failed to capture images both to its side and underneath it. The second camera cannot be fixed at a much higher vantage point because of its operational requirement of having to capture faces at eye-level height, which is unlike the placement of the first camera. This limitation occurs because of the rigidity in fixing a camera’s view.

# Future Improvements

## Recognizing Masked Faces

The system would need to be equipped with the capability to recognize masked faces in the times of Covid-19 pandemic. Removing one’s mask is required for the face recognition module to work, but this would be contrary to the sustained efforts of controlling the spread of the virus. The next phase of improving the system will concentrate on the recognition of masked faces.

## Roving Camera

A fixed camera is not as flexible in its deployment as one that can be free roaming. An autonomous intelligent robotic system equipped with a camera can address this limitation. A wheeled robot can be programmed to move its camera to point in the direction of the intruder.

## Face Recognition Using Multiple Frames of Images

With a camera-equipped roving robot, the system now has the possibility of moving around the intruder to capture different angles of the face. Chaining several frames of images using the Long-Short Term Memory (LSTM) framework might lead to higher face recognition accuracy.

## D. Recognizing Activity

The system developed using the LSTM framework is likely to widen its scope of operation, by recognizing human activity. The current system can only recognize faces. But if the system can also recognize human activity such as theft, then the system can provide more detailed information to aid the decision-making of the office staff.

# Conclusion

The project set out to develop a system that can intelligently detect intruders who came too close to a valuable asset. The system achieved the goal that was set for it, by being able to detect intruder in the restricted zone and identifying whether the intruder was a known or unknown person. The office staff is immediately informed of the trespassing event by the system accordingly.

The system utilized the technique of background subtraction together with a camera mounted above the Smart City Demo to implement the functionality of intruder detection. The Face Recognition module next takes over when triggered by the signal generated by the detection of intruders. The technique used in the Face Recognition module included Haar Face Cascade for fast face detection, MTCNN for face alignment and landmarks detection, and dlib for face encoding.

Multiple models such as KNN, SVM, Logistic Regression, and MLP were compare with the performance of ensemble voting in accurately recognizing known and unknown faces. Results suggested that the voting ensemble model gave the best performance in face recognition. The system next completes its access control task by sending email and SMS to office staff for their further actions.

Our sponsor Oracle see the potential of this project to be replicated for other critical assets such as IT operation room, data centers, and HR office file cabinet. The project also showcased Oracle solution components driving this system, and thus highlighting Oracle cloud solution to potential cliental.

##### Appendix A

##### Project Plan

A screenshot of a cell phone

Description automatically generated

##### Appendix B

##### User Acceptance Test

|  |  |  |
| --- | --- | --- |
| **Scenario** | **Requirement** | **Result** |
|  |  |  |
| **Camera Placement** |  |  |
| Camera A directed at restricted area and Camera B looking straight | Precheck on camera angles | Ok |
|  |  |  |
| **Intruder Detection (I)** |  |  |
| People are near but not in the restricted zone | Camera A does not detect presence of people | Passed |
| **Intruder Detection (II)** |  |  |
| At least one person is in the restricted zone | Camera A detects presence of people | Passed |
| **Face Recognition (I)** |  |  |
| One unknown person in the restricted zone | Unknown (Unauthorized) person is detected (with bounding box and text 'Unknown') | Passed |
| Email is sent with attachment of picture of person | Passed |
| SMS is sent with message | Passed |
| **Face Recognition (II)** |  |  |
| One known person in the restricted zone | Known (Authorized) person is detected (with bounding box and name of person) | Passed |
| No alert is sent | Passed |
| **Face Recognition (III)** |  |  |
| Multiple persons in the restricted zone (both unknown and known) | Detection as per above conditions apply | Passed |
| Email is sent with attachment of picture of person | Passed |
| SMS is sent with message | Passed |
| **Face Recognition (IV)** |  |  |
| Detected person moves away | Face recognition process stops and revert to Intruder Detection (I) | Passed |

##### Acknowledgment

This project would not have been possible without the generous support and help of many individuals and organizations. The authors would like to express our sincere thanks to all of them.

We would like to express our gratitude towards Oracle Corporation for their sponsorship of the project. We thank Dr Zhu Fang Ming for his guidance and mentorship throughout the project. Lastly, we would like to acknowledge all the supports that we have receive from the Institute of Systems Science.

##### References

1. P. Viola and M. Jones. “Rapid object detection using a boosted cascade of simple features,” in *Proceedings, IEEE Conference on Computer vision and Pattern Recognition*, 2001, pp. 511-518.
2. Zhang, K., Zhang, Z., Li, Z., and Qiao, Y. “Joint face detection and alignment using multitask cascaded convolutional networks.” *IEEE Signal Processing Letters*, 23(10):1499–1503, 2016.
3. GitHub. Ageitgey/face\_recognition. Available at: <https://github.com/ageitgey/face_recognition>. Accessed 8 Sep 2019.
4. N. Dalal and B. Triggs. “Histograms of oriented gradients for human detection,” in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2005, pp. 886-893.
5. Intelligent Behaviour Understanding Group. iBUG 300-W dataset. Available at: <https://ibug.doc.ic.ac.uk/resources/facial-point-annotations/>. Accessed 8 Sep 2019.
6. E. Hoffer and N. Ailon. “Deep metric learning using triplet network,” in *International Workshop on Similarity-Based Pattern Recognition*, 2015, pp. 84-92.
7. F. Schroff, D. Kalenichenko and J. Philbin. “FaceNet: A unified embedding for face recognition and clustering,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 815-823.