##### ***Facial Recognition Integrated with MLL***

*M Swati, N Roshan Reddy*

Table of Contents

[Abstract 3](#_Toc9248744)

[Facial Recognition 3](#_Toc9248745)

[How does it work? 3](#_Toc9248746)

[Face Detection vs. Facial Recognition 4](#_Toc9248747)

[Face detection 4](#_Toc9248748)

[Facial recognition 4](#_Toc9248749)

[Facial recognition Implementation Overview 4](#_Toc9248750)

[Registration and Facial recognition Workflows 5](#_Toc9248751)

[Multimedia Life Line (MLL) Solution Overview 7](#_Toc9248752)

[Integration with MLL 8](#_Toc9248753)

[Software Requirements 8](#_Toc9248754)

[Performance Considerations 8](#_Toc9248755)

[Other Recommendations for Improvements 9](#_Toc9248756)

[Location References 9](#_Toc9248757)

[Code Overview 9](#_Toc9248758)

# Abstract

*In Today's word, there does not exist a way to automatically detect un-authorized, suspicious people at particular location, such as a Campus, and send notification to the Authorities to address the matter before any potential offense or criminal activity has been committed. We can leverage existing Facial Recognition APIs to provide automatic detection of suspicious people at a particular location by comparing the faces identified with existing database, in  conjunction with the Multi Media Life Line (MLL) application, and send automatic alerts.*



# Facial Recognition

Facial recognition is a category of [biometric](https://searchsecurity.techtarget.com/definition/biometrics) software that maps an individual's facial features mathematically and stores the data as a faceprint. The software uses deep learning algorithms to compare a [live capture](https://searchsecurity.techtarget.com/definition/live-capture) or digital image to the stored faceprint in order to verify an individual's identity.

# How does it work?

The software identifies nodal points on a human face. In this context, nodal points are endpoints used to measure variables of a person’s face, such as the length or width of the nose, the depth of the eye sockets and the shape of the cheekbones. The system works by capturing data for nodal points on a digital image of an individual’s face and storing the resulting data as a faceprint. The faceprint is then used as a basis for comparison with data captured from faces in an image or video.

# **Face Detection vs. Facial Recognition**

Face detection utilizes machine learning algorithms to discern a “human forms” in a photo or video by focusing upon distinct facial features, such as eyes, etc., and distinguishes them from other objects. No attempt is made to obtain a positive identification of the person(s) within the visible field. The digital image is *not* matched against a database of known individuals. It is anonymous by definition. Thus, it does not know the “who” that is in the image

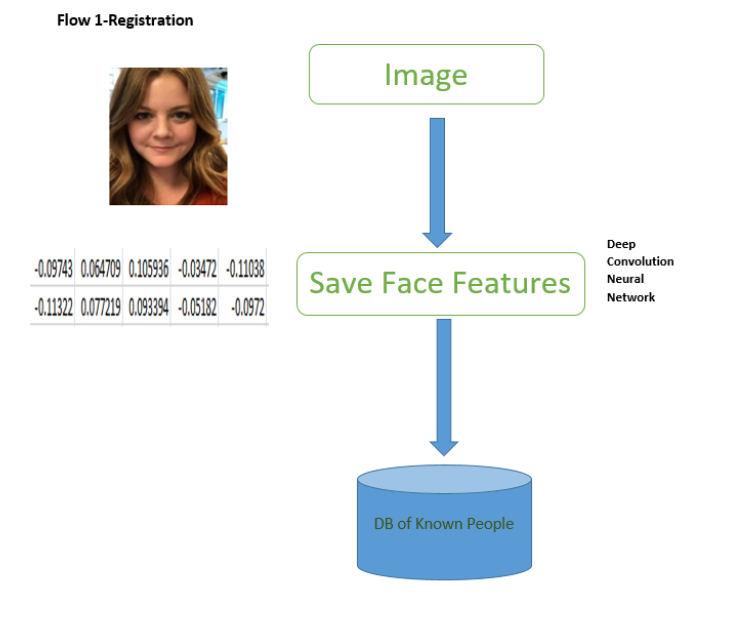
Facial recognitionis the next stepthat follows Face Detection. That is face detection is a pre-requisite for facial recognition. This methodology concentrates on making a positive identification of the detected individual against a database that archives personal information. Confidence factor is a key metric to avoid improper identification. Facial recognition is typically done with via Neural Networks (NNs), especially Convolutional Neural Networks (CNNs).

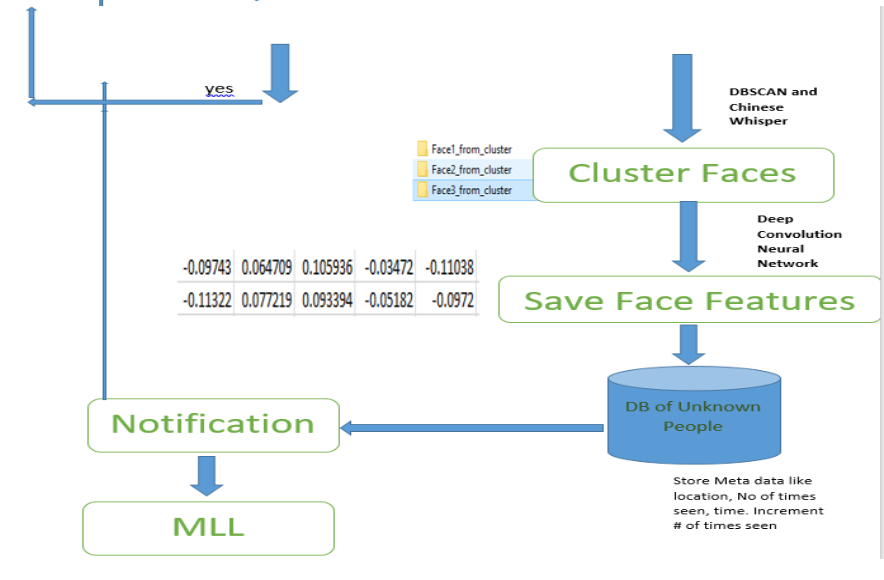
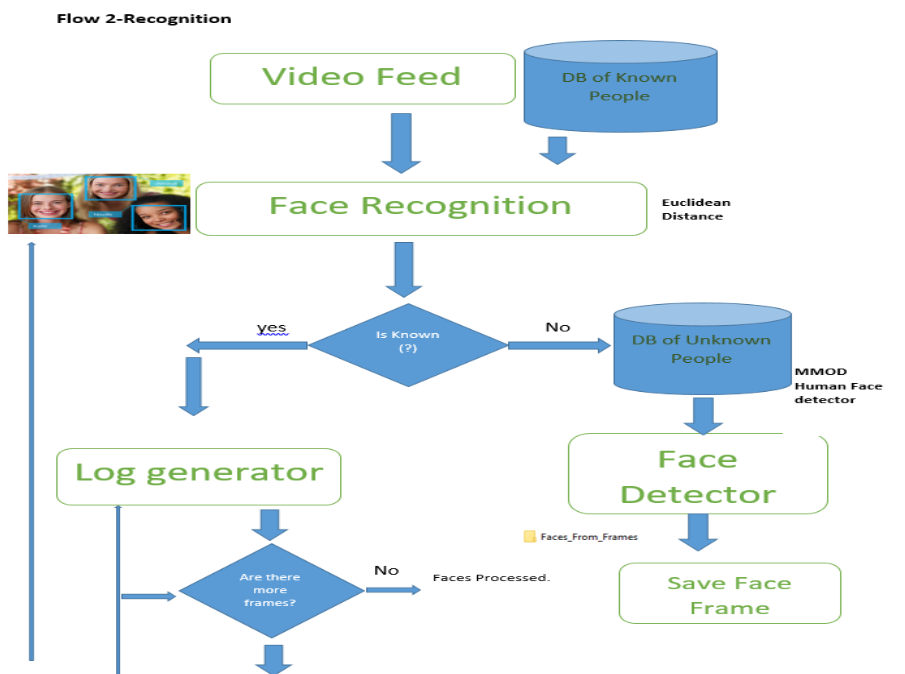
# Facial recognition Implementation Overview

The video feed is given as an input to the **Face Detector,** which crops and saves the faces detected in the frames by using Maximum-Margin Object Detector (MMOD) with CNN based feature via [Dlib](http://dlib.net/), an open source Machine learning framework (CNN is the standard supervised deep learning algorithm for image classification). The face cluster uses the unsupervised learning algorithms DBSCAN and Chinese Whisper which are run on these saved faces from the frame. As a result of clustering identical looking faces are collected and put into one folder. Thus, at the end of cluster faces we end up with several folders each having identical looking faces. These folders are provided as input to **Save Face Features** which uses the Deep CNN to return 128-dimension face encoding. The **Facial Recognizer** makes use of these saved features in the CSV(Comma Separated Values) files to classify/identify the known and unknown faces against a  **database** that is registered ahead of time with known (“authorized identifies”). The Dlib library states that it can provide up to 99.38% accuracy. The known and unknown identities are logged. A notification is sent to the **MLL application** in case of an unknown person. The second diagram “**flow 2 – Recognition**” represents this flow.

The first diagram below “**flow 1 – Registration**” represents the flow when valid/known identities are registered in the database. The registration should happen prior to performing facial recognition software. Maintaining this database is currently outside the scope of this project.

# Registration and Facial recognition Workflows





# Multimedia Life Line (MLL) Solution Overview

The Multimedia Life Line (MLL) application is aimed at helping secure areas, such as a College Campus or any enterprise. It is designed to aid law enforcement and/or first responders to quickly react and address any reported incident or hazardous situation. It is able to address a large number of reports and correlate rapidly to allow the security or law enforcement to take appropriate action in real time. By automating time consuming tasks, it will allow them to address the matter quickly.

It consists of a wide range of features:

* + Data visualization
  + Historic view of past events
  + Data management with easy to use UI
  + Extensible and Scalable
  + Utilizes modern tooling
  + Two way communication with Police and someone nearby the incident
  + Supports mobile devices, desktops, and beacons
  + Interacts with Twitter and email to notify of authenticated incidents.

It comprises of 5 major blocks:

* + The Command Center
  + The Mobile app
  + IoT Device Integration
  + Camera Integration
  + Social Media Integration

For more information on MLL, consult [Multimedia Life Line Design Document](https://unisyscorp-my.sharepoint.com:443/:b:/g/personal/michael_didomenico_unisys_com/ETRPSP3aKRFDmugG2SHF7dUBO0ics50UTE86084wA9fCSw?e=4%3austOaT%3aorigemail&at=9).

# Integration with MLL

1. Run the facial recognition algorithm to recognize the known and unknown faces.
2. As unknown faces (i.e. unauthorized people) are detected, the shell script “Face\_test.sh” will run automatically for each unknown face to load the message “UNKNOWN FACE FOUND!” to the Alerts database of MLL.
3. Then MLL App can run accordingly to address situation

# Software Requirements

The following Software must be loaded onto to your systems:

* MONGODB 3.6.10
* Node JS 10.15
* The Latest Dlib framework(dlib 19.16.0 is used here)
* Anaconda 5.3.1
* Opencv-python 3.2.0.6
* Python 3.6.2
* CMake 3.14
* Sckit-Learn 0.19.0

# Performance Considerations

We are utilizing the CNN API from Dlib framework. It is optimized for running on a system or cloud framework with GPU Support as recommend by the dlib libraries. If you are running this application on a system that does not have the GPU, it may run slower than expected. We have not done performance characterization. This should be considered for a future activity.

# Other Recommendations for Improvements

In our research project, we have utilized the Dlib framework since it is easy to use and could build prototype quickly. However, the leading framework at the moment is generally considered to be **OpenFace**. This should be looked at.

Maintaining the Known people and Unknown people databases need to be managed. For example, purging of people that no longer belong to either database.

# Location References

SVN Link to the repository and documentation for the facial detection/recognition software, along with MLL:

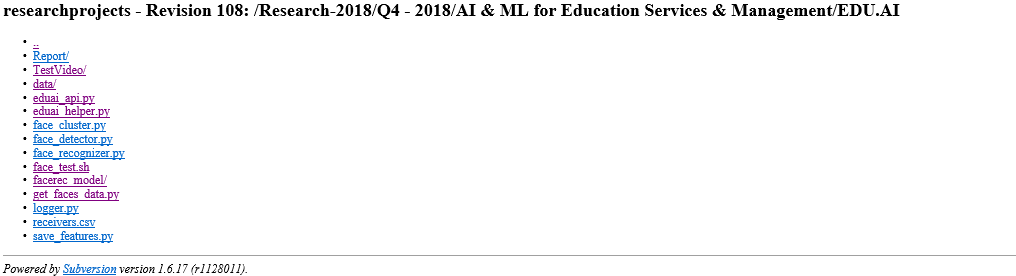
<https://ustr-svn-1.na.uis.unisys.com/researchprojects/Research-2018/Q4%20-%202018/AI%20&%20ML%20for%20Education%20Services%20&%20Management/>

# Code Overview

As mentioned earlier, we are utilizing Dlib, “a toolkit for making real world machine learning and data analysis applications in C++. While the library is originally written in C++, it has good, easy to use Python bindings.”

Below is a guide to the source code

The Facial Recognition and Detection software contains the following Python modules and test data.



**Report** –This contains the generated report of comparison of various face cluster algorithms available through unsupervised learning

**Test Video**-Input video to the face detection process

**Data**-The data folder contains the following subfolders generated by running the application.

**\*)Faces\_From\_Video** – Contains all the cropped images of faces after running the face\_detector program on the input video. This will act as input path to the face\_cluster program.

**\*)Cluster\_output** – Contains the subfolder of faces organized in cluster at the end of the face\_cluster program.

**\*)Csvs\_from\_video** – Contains the CSV files generated for each individual after running the save\_features program

**Eduai\_api**- Contains the functions definition for dlib’s get\_frontal\_face\_detector, face\_recognition\_model\_v1, shape\_predictor, euclidean\_distance and cosine\_similarity calculations.

**Eduai\_helper**- Contains the helper class for VideoHandler.

**Face\_detector**-Takes the input test video file and reads the frames in the video. It crops and saves the faces in the frames in the folder Faces\_From\_Video.

**Face\_cluster** – The cropped faces in the folder Faces\_From\_Video is run through the cluster algorithm dlib.chinese whispers clustering and DBSCAN and the cluster output is saved in the output folder Cluster\_output.

**Save\_features**- Uses the 128d\_features() to get the face descriptors .This is written into the CSV files using the function write\_into\_csv. The output CSV files are stored in the folder Csvs\_from\_video. Finally we use compute\_the\_mean() method to compute the mean value of each face features, and then save to features\_all.csv.

**Face\_recgonizer**- Based on the Euclidean distance range, the face recognizer identifies and maps the person in the played video with their respective names. If the distances.min() > 0.5, then the person is tagged as unknown and warning mail is sent .

**Face\_test.sh-** This shell script file has the message to be automatically inserted in MLL alerts database when the unknown face is recognized.

**Logger-**Logs the information about the incoming faces. If new unknown face is found, the logger sends a small jpg image of the unknown face along with waning message via outlook application .