PERSONNEL IDENTIFICATION FOR MILITARY FACILITIES ACCESS

<u>BIOMETRIC SYSTEMS COURSE – CYBERSECURITY MASTER'S DEGREE</u>

VINCENZO FRANCESCO ZERBO – MATR. 1937077
FEDERICO LAGRASTA – MATR. 1907104

OBJECTIVE

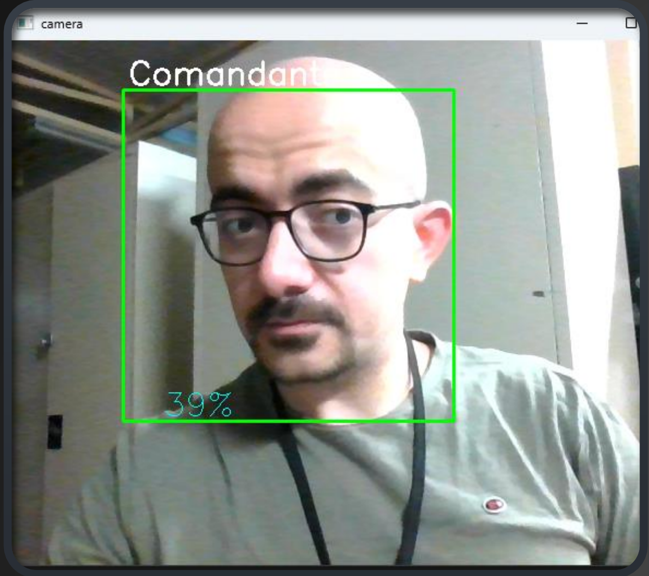
- Recognize Authorized/Unauthorized personnel through webcam face recognition
- Using OpenCV
    - For Face Detection and Data Gathering
    - To Train the recognizer with Data
    - To Recognize Visitors
- **Requirements**:
    - Dataset: http://vis-www.cs.umass.edu/lfw/
    - Pillow
    - CMAKE
    - Haar Cascade classifier
    - Numpy

FACE RECOGNITION

It is a technology that involves identifying and verifying the identity of individuals based on their facial features. It analyzes and compares unique patterns and characteristics of a person's face to determine their identity.

The process typically involves capturing an image or a video frame containing a person's face and extracting facial features, converted into a mathematical representation known as a face template or faceprint.

To recognize a face, the obtained face template is compared to a database or a set of known face templates to find a potential match. This comparison can be performed using various algorithms, including **statistical methods**, **machine learning techniques**, or deep **neural networks**, which are trained to recognize patterns and similarities between faces.



OPENCV

OpenCV (Open Source Computer Vision Library) is an open-source computer vision and machine learning software library.

It provides a comprehensive set of functions and algorithms that enable developers to perform various computer vision tasks such as image and video processing, object detection and tracking, facial recognition, and more.

OpenCV provides a vast collection of functions and modules that cover various aspects of computer vision. Some of the key features and capabilities of OpenCV include:

- **Image and video I/O**: OpenCV allows reading, writing, and processing of images and videos in various formats.
- **Image and video processing**: It provides functions for image filtering, transformation, enhancement, and geometric operations.
- **Machine learning**: OpenCV integrates machine learning algorithms for tasks like classification, clustering, and regression.

PHASE 1 – FACE DETECTION

Before anything, we must capture a face to recognize it when we'll compare with a new face during next phases

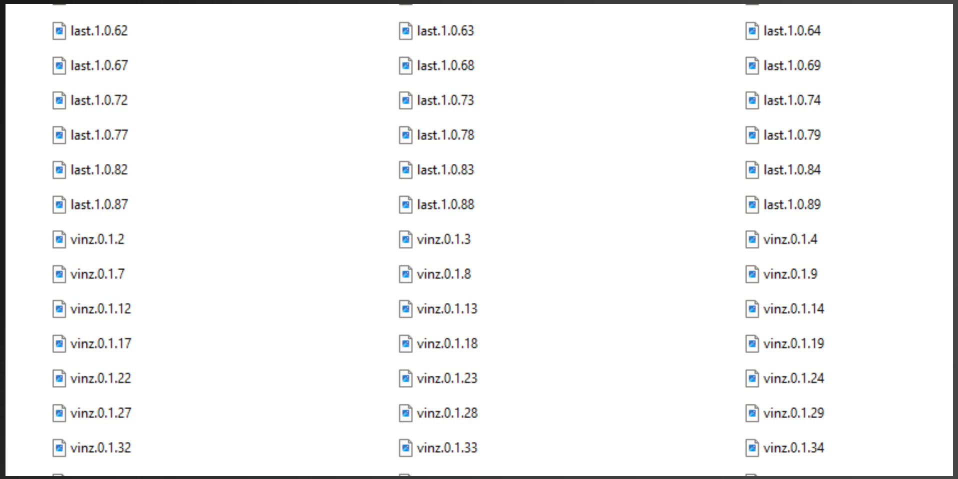
In this case, we'll ask for:

- Last Name
- Code (0 or 1) to distinguish Authorized/unauthorized people.
- Identification code (unique)

60 frames-example will be used to detect and recognize each visitor.

Haar Cascade classifier will elaborate the data, and OpenCV contains a lot of pre-trained classifiers for face, eyes, smile, etc.

DATASET CREATION



```
last.1.0.62     last.1.0.63     last.1.0.64
last.1.0.67     last.1.0.68     last.1.0.69
last.1.0.72     last.1.0.73     last.1.0.74
last.1.0.77     last.1.0.78     last.1.0.79
last.1.0.82     last.1.0.83     last.1.0.84
last.1.0.87     last.1.0.88     last.1.0.89
vinz.0.1.2      vinz.0.1.3      vinz.0.1.4
vinz.0.1.7      vinz.0.1.8      vinz.0.1.9
vinz.0.1.12     vinz.0.1.13     vinz.0.1.14
vinz.0.1.17     vinz.0.1.18     vinz.0.1.19
vinz.0.1.22     vinz.0.1.23     vinz.0.1.24
vinz.0.1.27     vinz.0.1.28     vinz.0.1.29
vinz.0.1.32     vinz.0.1.33     vinz.0.1.34
```

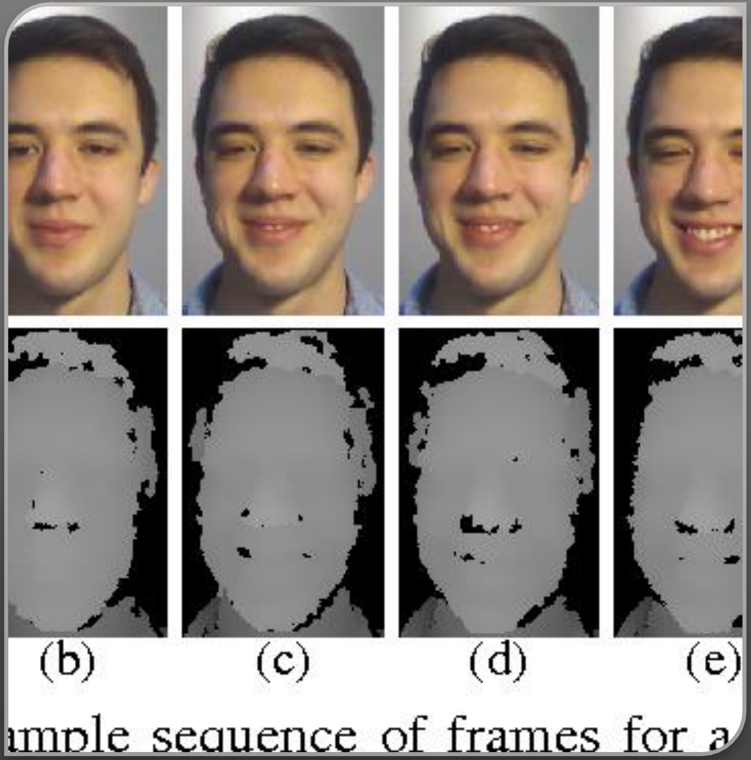
- Using data from the previous step, we create a dataset where the photos of each user (portion used for face detecting) are stored.
- For final usage, we need to remember that within the name we’ll find «1» prefix for Authorized personnel, «0» prefix for Unauthorized personnel.

DATASET CREATION

Code:

```
surname = input('\n [!] Input the last name of the user > ‘)

authorization = int(input('\n [!] Input 1 for authorized, 0 for unauthorized. Any other value will
be considered as unauthorized > ‘))


# Make sure the identifier is unique per person

identifier = int(input('\n [!] Input an incremental identifier (MAKE SURE IT\'S UNIQUE!) > '))

…

cv2.imwrite("dataset/" + str(surname) + '.' + str(authorization) + '.' + str(identifier) + '.' +
str(count) + ".jpg", gray[y:y+h,x:x+w])
```

PHASE 2 - TRAINING OPENCV RECOGNIZER



So, through a specific function of OpenCV, we use all data from our dataset to train the Recognizer in order to use it for visitors recognition of a military barrack.

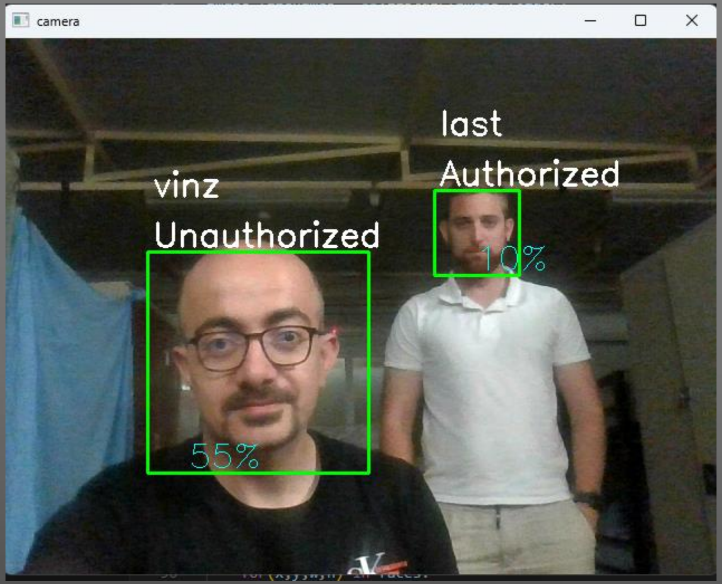
The **train()** function is then called on the recognizer object, passing the faces and labels as arguments to train the recognizer using the provided data.

```
faces,ids = getImagesAndLabels(path)
```

```
recognizer.train(faces, np.array(ids))
```

All the results will be stored on a dedicated «trainer/» directory

PHASE 3 - RECOGNITION



In this phase, each face will be captured on our camera and if this person had his face trained before, the recognizer will make a "prediction" showing “Authorized” or “Unauthorized” and how confident the recognizer is with this match.

04_TEST_DATASET & 05_TESTING

The 04_test_dataset and 05_testing files have the purpose of verifying the accuracy of the Recognizer through the use of 2 **test datasets** (test_dataset & test_dataset2 folder2).

The datasets contain a series of faces to which the values «Authorised» and «Unauthorised» have been randomly assigned.

Once the **05_testing** script starts, the following are then calculated:

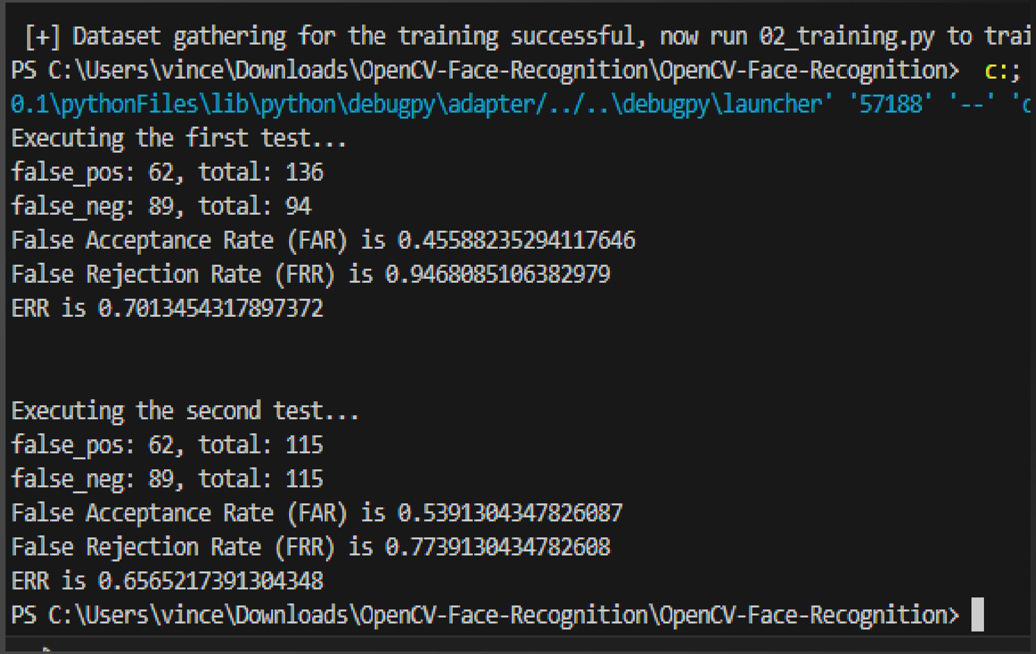
- False Acceptance Rate (FAR);
- False Rejection Rate (FRR);
- Equal Error Rate (EER).

04_TEST_DATASET & 05_TESTING

The datasets include both the same people from the training set and different other unknown faces from an open-source dataset.

We can see an inversely proportional behavior between FAR and FRR.

As expected, the FAR and FRR are pretty high, and this is because the model used needs more fine tuning, both from the perspective of how the data is gathered and how it’s processed by the face recognition library.



```
[+] Dataset gathering for the training successful, now run 02_training.py to trai
PS C:\Users\vince\Downloads\OpenCV-Face-Recognition\OpenCV-Face-Recognition>  c:;
0.1\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '57188' '--' 'c
Executing the first test...
false_pos: 62, total: 136
false_neg: 89, total: 94
False Acceptance Rate (FAR) is 0.45588235294117646
False Rejection Rate (FRR) is 0.9468085106382979
ERR is 0.7013454317897372


Executing the second test...
false_pos: 62, total: 115
false_neg: 89, total: 115
False Acceptance Rate (FAR) is 0.5391304347826087
False Rejection Rate (FRR) is 0.7739130434782608
ERR is 0.6565217391304348
PS C:\Users\vince\Downloads\OpenCV-Face-Recognition\OpenCV-Face-Recognition>
```

CONCLUSION

We built a proof of concept of a system that could enable the Italian Army to use authomated access control, using the open-source tool **OpenCV** Recognizer and the **Python** programming language.

Possible future works:

- Use a wider faces dataset with which to train the machine learning algorithm;
- Increase the accuracy of Gathering & Training mechanisms;
- Integrate our system with the Administration Software of the Italian Army;