& Security

Tunisian Republic
Ministry of Higher Education and Scientific Research
Tunis Business School

Security Project Report

Submitted to

Tunis Business School

by

Jihen Boughattas
Yosser Fhal
Islem Smai
Doua Teyeb
Yassmine Yazidi

Facial Recognition

Defended on *??/05/2025* in front of:

Mrs. Manel AbdelKader Professor

Abstract

These days, we rely on technology more than ever, even for simple tasks. Facial detection and recognition have become a part of our daily lives. Whether it's organizing photos by faces in our phone gallery, unlocking a device with a quick look, or using face images as a form of biometric ID for official purposes, this technology is all around us.

This project introduces the basic concepts behind Face Detection and Face Recognition using OpenCV, a computer vision library developed by Intel. It also includes a step-by-step guide to building a face recognition system using Python with OpenCV, and shows how it works on both Windows and macOS.

The main goal of the project is to recognize faces that the system has been trained to identify, using a webcam to capture real-time video. When a known face appears, the program shows the person's name on the screen.

In the future, this kind of system could be expanded for practical uses like automatic attendance in schools or workplaces, helping to save time and reduce manual work.

Contents

Dedication **I**

Abstract **I**

1 Introduction **1**

1.1 Overview of Facial Authentication Systems 1

1.2 Why Facial Authentication Matters? 1

1.3 Face Detection vs. Face Recognition 1

1.3.1 Face Detection 2

1.3.2 Face Recognition 2

1.4 Project Scope 2

2 Technical Concepts **3**

2.1 Algorithm For Face Detection 3

2.1.1 Filtering 3

2.1.2 Resizing 3

2.1.3 Color Mode Conversion & Skin Detection 4

2.1.4 Morphological Operations 4

2.1.5 Elimination of Non-Face Regions 5

3 Mathematical Concepts & Security **6**

3.1 General Concept 6

3.2 Example Case: How a Facial Recognition System Works 6

3.3 Security & Accuracy Concerns 7

3.3.1 False Positives and False Negatives 7

3.3.2 Spoofing Attacks & Solutions 7

4 Authentication Flow Diagram **9**

4.1 Diagram 9

4.2 Steps 10

4.3 Components vs. Roles 10

5 The Existing Solutions **12**

5.1 OpenCV with Haar Cascades: 12
5.1.1 How It Works: 12
5.1.2 Use Cases: 12
5.1.3 Limitations: 12
5.2 Dlib with HOG (Histogram of Oriented Gradients): 13
5.2.1 How It Works: 13
5.2.2 Use Cases: 13
5.2.3 Limitations: 13
5.3 MTCNN (Multi-task Cascaded Convolutional Networks) 13
5.3.1 How It Works: 13
5.3.2 Use Cases: 14
5.3.3 Limitations: 14
5.4 FaceNet 14
5.4.1 How It Works: 14
5.4.2 Use Cases: 14
5.4.3 Limitations: 14
5.5 VGGFace / VGGFace2 15
5.5.1 How It Works: 15
5.5.2 Use Cases: 15
5.5.3 Limitations: 15
5.6 OpenCV Deep Neural Network (DNN) Module 15
5.6.1 How It Works: 15
5.6.2 Use Cases: 15
5.6.3 Limitations: 16

Chapter 1

Introduction

1.1 Overview of Facial Authentication Systems

Facial authentication systems are designed to identify or verify a person's identity by analyzing their facial features. This process uses advanced image processing and pattern recognition techniques to convert facial data into digital information, which is then compared against stored data for verification.

In recent years, facial authentication has become increasingly common due to its ease of use and non-intrusive nature. It is now widely used in smartphones, laptops, online exams, secure logins, and identity verification systems such as KYC (Know Your Customer) processes in banking.

1.2 Why Facial Authentication Matters?

Facial authentication offers a secure and convenient way of verifying identity without the need for passwords or physical tokens. With the rise of remote work, online education, and digital services, the need for efficient and reliable authentication methods has grown significantly.

This technology helps in:

- Reducing fraud and unauthorized access.
- Automating verification processes.
- Enhancing user experience through faster and contactless authentication.

As security concerns increase, biometric systems like facial recognition are being adopted across various industries to add an extra layer of safety and accountability.

1.3 Face Detection vs. Face Recognition

It is important to understand the difference between face detection and face recognition:

1.3.1 Face Detection

Face detection is the process of locating a face within an image or video frame. The system determines where a face is by identifying facial structures such as eyes, nose, and mouth. It does not identify who the person is; it only confirms that a face exists in the frame.

1.3.2 Face Recognition

Face recognition goes a step further by identifying or verifying the person whose face has been detected. This is done by comparing the detected face with previously stored face data. The system analyzes specific facial features, such as the distance between the eyes, the shape of the cheekbones, and the contour of the jaw, and creates a unique facial signature, often referred to as a *faceprint*.

1.4 Project Scope

This project focuses on developing a web-based facial authentication system using OpenCV, a widely used open-source library for computer vision tasks. The application uses a webcam to capture live video, detects the user's face, and compares it with stored images to determine identity.

Although this project is a small-scale implementation, it can be further extended to serve more advanced use cases like:

- Online exam proctoring.
- Meeting or conference login verification.
- Law enforcement and surveillance systems.
- Device and application access control.

Chapter 2

Technical Concepts

2.1 Algorithm For Face Detection

The face detection algorithm begins by taking a color image as input and converting it to grayscale to simplify the data. The image is then resized to reduce processing time. Next, the program detects the edges in the image using the Sobel operator, which highlights areas where brightness changes suddenly—these are often the outlines of objects like faces.

A threshold value, based on the average gray level of the image, is used to create a binary (black-and-white) version where edges appear in white. To make the detected outlines clearer, a dilation step is applied to thicken the borders. The algorithm then fills in any holes inside these shapes to form complete face-like regions. To clean up the image, small unwanted areas are removed using erosion and dilation, helping retain only the important parts like the face. Finally, the image is resized back to its original dimensions, now ready with the face region identified.

2.1.1 Filtering

The first step in the face detection process is filtering, which is used to clean up the image by removing unnecessary noise and small details that do not contribute to the face detection task. This step helps smooth the image, making it easier to focus on important features such as edges and contours, thus improving the efficiency of the next steps of the algorithm.

2.1.2 Resizing

Once the image is filtered, it is resized to reduce its dimensions. This step is done to speed up the face detection process. By reducing the image size, the algorithm can analyze the image faster without losing significant information needed to detect faces. After the analysis, the image will be resized back to its original dimensions for further processing.

2.1.3 Color Mode Conversion & Skin Detection

The image is converted from RGB to grayscale for simplification. Additionally, skin detection is performed using the HSV (Hue, Saturation, and Value) color model, also referred to as HSB when considering brightness.

- **Hue (H)**: represents the color's angle on a color wheel, ranging from 0 to 360 degrees. In the context of skin detection, the angle typically falls within a specific range of 0 to 50 degrees to focus on skin tones.
- **Saturation (S)**: indicates the intensity of the color, with higher values representing more vivid colors. For skin detection, the saturation must fall between 0.18 and 0.68 to identify skin regions effectively.
- **Value (V)**: refers to the brightness of the color, and for skin tones, it must fall within the range of 0.35 to 1. This ensures that the detected areas are neither too dark nor too light, keeping the focus on natural skin colors.

These ranges allow accurate skin detection before grayscale conversion to assist in finding potential face areas in the next steps.

2.1.4 Morphological Operations

Morphological operations help refine the binary image by enhancing or reducing regions of interest.

Dilatation

The dilatation operation is applied to thickening the detected edges, making them more prominent. This is particularly useful for connecting edges that might have been broken, ensuring the program can recognize continuous shapes like a face outline.

Filling

After dilatation, the filling operation is used to fill in any gaps or holes within the detected face regions. This helps in forming a solid shape, turning incomplete outlines into full, recognizable face regions.

Erosion

To remove small unwanted objects or noise, the erosion operation is performed. Erosion shrinks the white shapes, effectively eliminating tiny non-face objects and cleaning up the image so only the relevant regions remain.

2.1.5 Elimination of Non-Face Regions

After morphological operations, non-face regions are removed using two techniques:

Pixels Techniques

The algorithm uses pixel-based techniques to analyze the number of white pixels in each detected region. If the region has too few pixels or is not of the right size to be a face, it is discarded.

Ratio Techniques

In addition to pixel count, ratio techniques are applied to check the shape and size of the detected regions. A typical face has a specific height-to-width ratio, so any regions that do not match this expected face shape are removed. This ensures that only regions that resemble faces are kept for further analysis.

Chapter 3

Mathematical Concepts & Security

3.1 General Concept

Modern facial recognition uses:

- **Convolutional Neural Networks (CNNs)** to extract face features.
- **Embedding vectors** as compact representations of faces.
- **Euclidean distance** to measure similarity between faces.

A CNN extracts features (like eye distance, jawline shape) and converts them into a numerical vector. Comparing two such vectors using Euclidean distance determines face similarity:

- Small distance $\Rightarrow$ likely same person
- Large distance $\Rightarrow$ likely different persons

The Euclidean distance between two embedding vectors $X = [x_1, x_2, \ldots, x_n]$ and $Y = [y_1, y_2, \ldots, y_n]$ is given by:

$$d = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2}$$

3.2 Example Case: How a Facial Recognition System Works

Step 1: Registering a Face

- Capture face of employee (e.g., Yassmine)
- CNN processes it to generate embedding vector:

$$Yassmine'sembedding = [0.12, -0.34, 0.08, \ldots, 0.91]$$

- Save vector as her digital signature

Step 2: Verifying a Face

- A new image is captured
- CNN generates new embedding vector
- System compares new vector with saved one

Step 3: Distance Calculation

- Saved embedding: $[0.12, -0.34, 0.08, \dots, 0.91]$
- New embedding: $[0.10, -0.30, 0.09, \dots, 0.90]$
- Compute Euclidean distance: $d = 0.55$
- If $d < 0.6 \Rightarrow$ Match $\Rightarrow$ Unlock door
- If $d > 0.6 \Rightarrow$ No match $\Rightarrow$ Access denied

3.3 Security & Accuracy Concerns

Facial recognition systems are susceptible to specific challenges:

3.3.1 False Positives and False Negatives

- **False Positive**: Unauthorized person recognized as authorized.
- **False Negative**: Authorized person not recognized due to lighting or expression changes.

How to prevent such concerns?

- Carefully tune Euclidean distance threshold
- Balance sensitivity to reduce both error types

3.3.2 Spoofing Attacks & Solutions

Spoofing Attacks

One serious security issue is spoofing, where someone tries to fool the system using a photo, video, or even a 3D mask of another person.

- Example: holding up a printed photo of Yassmine's face to the camera might trick a simple system into unlocking the door.

Basic CNNs that only extract features from static images cannot tell whether the face is real or live.

How to prevent such concerns?

Liveness detection techniques:

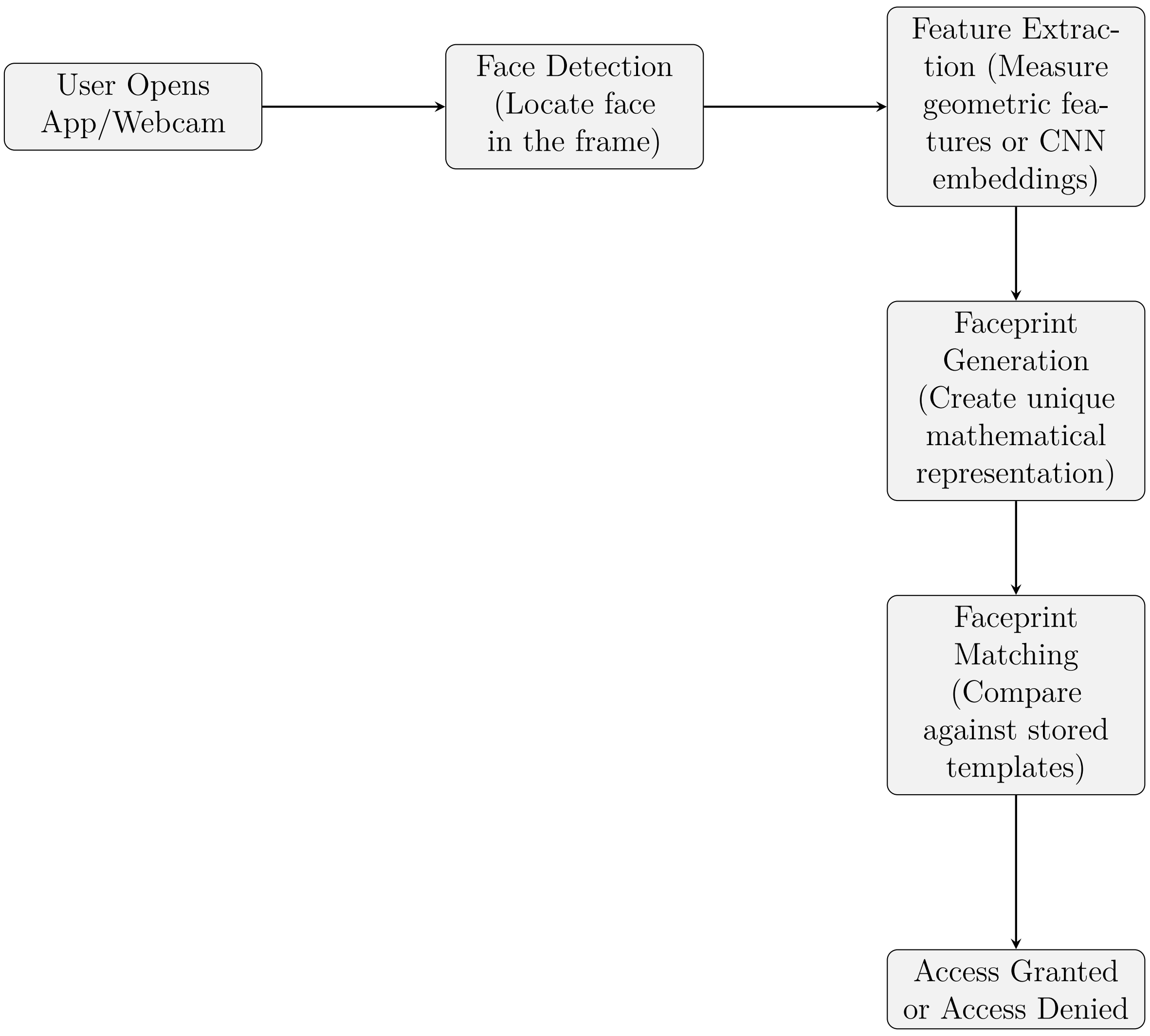
- Ask user to blink, smile, or turn head
- Use infrared or depth sensors
- Analyze micro-movements or thermal patterns

A second CNN model can be trained to detect signs of life, ensuring the detected face is genuine.

Chapter 4

Authentication Flow Diagram

4.1 Diagram


User Opens App/Webcam
Face Detection (Locate face in the frame)
Feature Extraction (Measure geometric features or CNN embeddings)
Faceprint Generation (Create unique mathematical representation)
Faceprint Matching (Compare against stored templates)
Access Granted or Access Denied

4.2 Steps

1. **User opens App/Webcam:**
   - The user activates the camera via an app or web interface.
2. **Face detection:**
   - Algorithms like Haar cascades, MTCNN, or YOLO isolate the face from the background.
   - Output: A bounding box around the face (e.g., using OpenCV or MediaPipe).
3. **Feature extraction:**
   - Geometric approach: Measure distances (e.g., eye spacing, jawline shape).
   - Deep learning approach: Use CNNs (e.g., FaceNet, ResNet) to generate embeddings.
   - Normalization: Adjust for lighting, pose, or expression.
4. **Faceprint generation:**
   - Convert features into a numerical vector (e.g., [0.34, -1.2, 4.5, ...]).
   - Storage: Faceprints are hashed/encrypted for security.
5. **Faceprint matching:**
   - Compare the generated faceprint to enrolled templates in a database.
   - Similarity metrics:
     - Euclidean distance (for geometric features).
     - Cosine similarity (for deep learning embeddings).
   - Threshold: Match is valid if similarity exceeds a pre-defined score (e.g., $\geq$95%).
6. **Access decision:**
   - Granted: User is authenticated (e.g., unlocks a device).
   - Denied: Authentication fails (e.g., alerts user or retries).

4.3 Components vs. Roles

Component	**Role/Function**
1. Camera/Webcam	Captures real-time images or video of the user's face.
2. Face Detection Module	Identifies and isolates the face within the captured image/video (e.g., bounding boxes).
3. Preprocessing/Normalization Module	Adjusts for lighting, pose, or resolution issues to standardize the face im

Component	**Role/Function**
4. Feature Extractor	• Converts the detected face into a numerical vector (faceprint) • Geometric: Measures distances (eyes, nose, jawline) • Deep Learning: Uses CNNs (e.g., FaceNet) to generate embedding
5. Faceprint Database	Securely stores enrolled faceprints (encrypted/hashed templates).
6. Matching Engine	• Compares the new faceprint with stored templates using similarity metrics • Euclidean distance (geometric features) • Cosine similarity (deep learning embeddings)
7. Decision Module	Grants/denies access based on a similarity threshold (e.g., $\geq$95% match)
8. Security Components	• Liveness Detection: Ensures the face is real (e.g., blinking, 3D dep • Encryption: Protects faceprints during storage/transmission
9. User Interface (UI)	Provides feedback (e.g., "Access Granted/Denied") and handles retries.

Chapter 5

The Existing Solutions

Before we dive into how facial authentication systems work, it's worth seeing what pieces of the puzzle are already on the table. Over the years, developers and researchers have come up with different tools and techniques to solve parts of the problem;some fit perfectly for simple tasks, while others are more complex but offer better accuracy. Each solution brings something unique to the overall picture.

The following technologies have become benchmarks in the field of face detection and recognition.

5.1 OpenCV with Haar Cascades:

(Open Source Computer Vision Library) is one of the most popular libraries used in computer vision. Haar Cascades is a classical machine learning-based approach, It is a part of OpenCV and one of the first widely adopted solutions primarily used for face detection.

5.1.1 How It Works:

Haar Cascades uses simple features like edges and lines to detect faces. It scans the image at different scales and positions, using a series of classifiers trained on many images to find facial features like the eyes, nose, and mouth.

5.1.2 Use Cases:

- Real-time face detection in lightweight applications
- Entry-level academic or hobbyist projects
- Simple surveillance systems

5.1.3 Limitations:

- Poor accuracy with rotated, tilted, or partially occluded faces

- Not robust against lighting changes
- Mostly useful for detection, not recognition

5.2 Dlib with HOG (Histogram of Oriented Gradients):

Dlib stands out as a robust library that excels in both face detection and recognition tasks. Its HOG-based face detector is renowned for its accuracy and efficiency, making it a popular choice in various applications.

5.2.1 How It Works:

HOG extracts edge direction features from images, combined with a linear SVM for classification. Dlib also provides facial landmark detection and a pre-trained recognition model.

5.2.2 Use Cases:

- Face alignment and recognition tasks
- Medium-scale authentication systems
- Applications that require decent accuracy without GPU

5.2.3 Limitations:

- Not as fast as Haar Cascades
- Struggles with extreme face poses
- Recognition accuracy is lower than deep learning models

5.3 MTCNN (Multi-task Cascaded Convolutional Networks)

A deep learning-based face detector that performs face detection and facial landmark localization simultaneously.

5.3.1 How It Works:

MTCNN uses three stages of CNNs to detect faces at various scales and refine bounding boxes. It also extracts five key facial landmarks (eyes, nose, mouth corners).

5.3.2 Use Cases:

- Applications requiring accurate face detection and alignment
- Useful as a pre-processing step before recognition
- Works well with mobile or web-based facial apps

5.3.3 Limitations:

- Requires more computational power
- Can be slower on devices without GPU
- May produce false positives in crowded scenes

5.4 FaceNet

One of the most advanced models for face recognition, FaceNet converts faces into embedding vectors for accurate identity comparison.

5.4.1 How It Works:

Uses a deep CNN trained with triplet loss to map faces to a 128-D space. The model ensures that the distance between same-person vectors is small, and between different persons is large.

5.4.2 Use Cases:

- High-accuracy facial authentication systems
- Large-scale face recognition (e.g., social networks, smart cities)
- Works well with face clustering and verification tasks

5.4.3 Limitations:

- Requires a good GPU for training/inference
- Model size and setup may be too heavy for small applications
- Needs careful threshold tuning for optimal results

5.5 VGGFace / VGGFace2

VGGFace is a deep learning model developed by the Visual Geometry Group at Oxford, trained on a large facial dataset.

5.5.1 How It Works:

Like FaceNet, it generates embedding vectors using deep CNNs. VGGFace2 is the improved version, with better diversity in age, ethnicity, and pose.

5.5.2 Use Cases:

- Academic research and enterprise-level solutions
- Systems that require robustness to face variations
- Face verification, clustering, and identification

5.5.3 Limitations:

- Large model, not lightweight
- Needs GPU and sufficient memory
- Training from scratch is resource-intensive

5.6 OpenCV Deep Neural Network (DNN) Module

OpenCV's DNN module allows the use of pre-trained deep learning models for tasks like face detection. It supports models such as ResNet, SSD, and MobileNet for more accurate and robust face detection than Haar Cascades.

5.6.1 How It Works:

The DNN module loads a deep learning model (often from TensorFlow, Caffe, or ONNX formats) and runs inference to detect faces. Popular setups include using a pre-trained SSD with ResNet-10 architecture, which outputs bounding boxes for faces.

5.6.2 Use Cases:

- Real-time face detection with better accuracy than classical methods
- Projects where speed and lightweight deployment matter (e.g., mobile apps)
- Situations needing a balance between speed and modern deep learning accuracy

5.6.3 Limitations:

- Still not as accurate as state-of-the-art face recognition models (e.g., FaceNet)
- Requires a bit more setup and knowledge of model files
- Not ideal for recognizing or verifying identities (only for detection)

Summary Comparison Table

Solution	**Type**	**Accuracy**	**Speed**	**Use case**
Haar Cascades	Classical	Low	High	Simple apps, basic detecti
OpenCV DNN	Deep learning	Medium-High	Medium	General face detection
Dlib + HOG	Hybrid (Classic + ML)	Medium	High	Landmark detection, mid-
MTCNN	Deep learning	High	Medium	Face detection + alignmen
FaceNet / VGGFace	Deep learning	Very High	Low-Medium	High-accuracy recognition

Overall, each face detection tool has its own job. Some, like Haar Cascades, are quick and easy to use but don't work great with tricky angles or poor lighting. Others, like DNNs or FaceNet, take more time to set up but are much smarter and more accurate. It really depends on what you're trying to build. For basic projects, simple tools might be enough. But if you want better results, especially in real-life situations, deep learning models are the way to go. It's all about picking the right tool for the task.