A Project Report On

Face Verification Using HOG and LBP

Submitted By

MD Mamun Ahmed	-	222901019
Shahariar Masud Sarkar	-	222901008
Md. Moyan Uddin	-	222901001

BSc. in EEE

Department of EEE

Submitted To

Mr. Shaharior Anik

Lecturer

Department of EEE

Green University of Bangladesh

Certificate

This is to Certify that the project-based laboratory report entitled "**Face Verification Using HOG and LBP**" Submitted by **Mamun Ahmed** (222901019), **Shahariar Masud Sarkar** (222901008), **Moyan Uddin** (222901001), studying Bachelor of Science in Electrical and Electronics Engineering has satisfactorily completed project on Spring (251) semester during the academic year 2022 – 2026.

Project Supervisor

Mr. Shaharior Anik

Acknowledgement

We express our sincere thanks to our project supervisor Mr. Shaharior Anik for his novel association of ideas, encouragement, appreciation and intellectual zeal which motivated us to venture this project successfully. We express our deep gratitude and affection to our parents who stood behind us in all our endeavors. Finally, it is pleased to acknowledge the indebtedness to all those who devoted themselves directly or indirectly to make this project report success.

Table of Contents

Abstract

This project presents the implementation of a facial recognition system utilizing classical digital signal processing techniques for feature extraction and comparison. The system is designed to detect and match a reference face within a collection of group photographs using Histogram of Oriented Gradients (HOG) and Local Binary Patterns (LBP) as the primary feature descriptors. A MATLAB-based graphical user interface (GUI) was developed to enable user interaction for loading a reference face, automatically detecting faces from a fixed folder of group images, and visually identifying the best match. The face detection component leverages the Viola-Jones algorithm through pre-trained cascade object detectors, while the recognition process relies on a weighted similarity score derived from the extracted HOG and LBP features. The system is robust against minor lighting and orientation variations, providing a practical application of signal processing concepts in pattern recognition. This project demonstrates how classical DSP methods can be effectively applied to real-world problems such as face recognition, offering insights into feature-based identification and system integration within MATLAB.

Introduction

Facial recognition—the task of identifying or verifying an individual based on facial features—has become a cornerstone in applications ranging from security and surveillance to personalized human–computer interaction. At its core, facial recognition poses a pattern-recognition challenge: raw image data must be transformed into informative representations that capture the essential characteristics of a face while remaining robust to variations in lighting, pose, and expression. Although modern deep-learning methods (e.g., convolutional neural networks) dominate the field today, classical digital signal processing (DSP) techniques such as Histogram of Oriented Gradients (HOG) and Local Binary Patterns (LBP) remain valuable both as educational tools and as efficient alternatives for systems with limited computational resources.

In this lab project, we harness these DSP-based feature extraction methods within a MATLAB environment to build a complete face-matching pipeline. Our system detects faces by leveraging the Viola–Jones framework (frontal and profile cascade object detectors), preprocesses detected regions to normalize scale and contrast, and then computes HOG and LBP descriptors that respectively encode edge orientations and local texture patterns. By combining these two feature types into a weighted similarity metric, the system can effectively rank candidate faces against a user-provided reference image.

To facilitate user interaction and visualization, a graphical user interface (GUI) was developed. This GUI allows students to load a reference face, automatically process a fixed folder of group photographs, display the strongest match along with its bounding box, and retrieve corresponding identity information from an Excel database. Through this implementation, students are introduced to the end-to-end workflow of a DSP-based pattern recognition system: from signal enhancement (contrast adjustment, adaptive histogram equalization) and feature computation to similarity scoring and results presentation. This hands-on approach reinforces the fundamental principles of feature-based recognition and demonstrates their practical integration within MATLAB.

Example Applications

- **Fraud Detection**: Automatically verify that the person performing a transaction (e.g., at an ATM or point-of-sale terminal) matches the authorized account holder.
- **Access Control**: Grant or deny building or room entry by comparing live camera input against a pre-registered reference image.
- **Exam Proctoring**: Continuously monitor and match students' faces during online or in-person exams to prevent identity fraud.
- **Attendance Tracking**: Instantly record attendance in classrooms or workplaces by recognizing faces in group photos.
- **Customer Analytics**: In retail environments, identify repeat customers to tailor personalized service or loyalty rewards.

Project Description

The **Face Recognition Using HOG and LBP: A DSP-Based Approach** project is designed to demonstrate the application of classical digital signal processing techniques to a real-world pattern-recognition problem. It consists of a MATLAB GUI front end that orchestrates a complete pipeline—from loading a reference image to presenting the best face match and associated identity information. The key components and workflow are described below:

1. **Objectives**

   - Implement robust face detection using classical DSP and computer-vision methods.
   - Extract descriptive features (HOG, LBP) that capture edge and texture information.
   - Design a weighted similarity metric to compare faces.
   - Build an interactive GUI that integrates detection, matching, and information retrieval.
   - Illustrate practical applications such as fraud prevention, access control, and attendance tracking.

2. **System Architecture**
   The system follows a modular design, comprising the following major blocks:

   - **GUI Module**
     - A uifigure window with buttons, axes, status labels, and an information panel.
     - Controls to load the reference face and view results.
     - Progress dialog during folder processing.
   - **Face Detection Module**
     - Two Viola–Jones cascade object detectors (frontal and profile) with custom thresholds.
     - Contrast-enhancement fallback to improve detection under poor lighting.
   - **Preprocessing Module**
     - Cropping around the detected bounding box with a configurable margin.
     - Grayscale conversion, resizing to a fixed 200×200 pixel template.
     - Adaptive histogram equalization, Gaussian smoothing, and contrast adjustment.
   - **Feature Extraction Module**
     - **HOG**: Computes gradients in localized cells to capture shape and edge information.

        - **LBP**: Encodes local texture patterns via binary thresholding of neighborhood pixels.

- **Matching Module**
    - For each candidate face, compute the Euclidean distance (ℓ_2 norm) between HOG feature vectors and between LBP feature vectors.
    - Combine distances with weights (0.6 for HOG, 0.4 for LBP) into a single "combined score."
    - Track the minimum score across all images; classify as "strong" or "potential" match based on a threshold (0.4).
- **Information Retrieval Module**
    - Parse the matching image's filename to extract an ID.
    - Look up student details (Name, Student_ID, Department, Batch, Contact) in an Excel file (Book1.xlsx).
    - Update the GUI panel with the retrieved information.

3. **Data Flow**

[User loads reference image]

↓

[Face Detection → Preprocessing]

↓

[Feature Extraction (HOG, LBP)]

↓

[Iterate over GroupPhotos folder]

↓

[For each image: Detection → Preprocessing → Features → Score]

↓

[Select lowest-score candidate]

↓

[Display match in GUI + Lookup in Excel]

4. **User Interction**

    - **Step 1**: Click "Load Reference Face" and select an image file.

- **Step 2**: The system detects and displays the reference face, then automatically begins folder processing.
- **Step 3**: A progress dialog shows real-time status as each group photo is analyzed.
- **Step 4**: The best match is displayed with a colored bounding box (green for strong, yellow for potential), and the matched person's details appear in the info panel.

5. **Technical Challenges & Solutions**
   - **Variable lighting and occlusion**: Addressed via adaptive histogram equalization and multi-model detection.
   - **Scale and alignment differences**: Mitigated through margin-based cropping, fixed resizing, and symmetric padding.
   - **Balancing descriptors**: Weighted combination of HOG (shape) and LBP (texture) provides more discriminative power than either alone.
6. **Deliverables**
   - Fully commented MATLAB code implementing all modules.
   - A working GUI capable of end-to-end face matching.
   - Sample GroupPhotos directory with test images.
   - An Excel database file (Book1.xlsx) containing synthetic student records.
   - A lab report documenting methodology, results, and potential improvements.

This project bridges theoretical DSP concepts and practical computer-vision applications, helping students to understand feature-based recognition pipelines and to gain proficiency in MATLAB's vision toolbox and GUI programming.

Working Principle

The facial recognition system operates as an end-to-end pipeline, transforming raw images into identification decisions through a sequence of signal-processing and pattern-recognition stages. Below is a detailed breakdown of its working principle:

1. **Face Detection**
   - **Viola–Jones Cascade Detectors**
     - Two pre trained detectors (frontal and profile) scan the image at multiple scales.
     - Each detector applies a cascade of simple Haar-like features to rapidly reject non-face regions.
   - **Contrast Enhancement Fallback**
     - If neither detector finds a face, the image is converted to grayscale and contrast-stretched (using imadjust) to reveal faint features, then re-scanned.
2. **Region Localization and Cropping**
   - **Bounding Box Selection**
     - Among all detected bounding boxes, the largest (by width × height) is chosen as the primary face.
   - **Margin Extension**
     - A margin of 20% around the box is included to capture contextual facial areas (hairline, chin).
   - **Crop & Grayscale Conversion**
     - The region is cropped and, if necessary, converted to a single-channel grayscale image.
3. **Normalization & Enhancement**
   - **Resizing to Fixed Template**
     - The cropped face is resized (while preserving aspect ratio) to fit a 200×200 pixel frame, with symmetric padding to fill any empty space.
   - **Adaptive Histogram Equalization**
     - adapthisteq redistributes local contrast to compensate for uneven illumination.
   - **Gaussian Smoothing & Contrast Adjustment**
     - imgaussfilt reduces high-frequency noise; imadjust further enhances overall contrast.
4. **Feature Extraction**
   - **Histogram of Oriented Gradients (HOG)**
     - Divides the face into smaller cells (e.g., 6×6 pixels), computes gradient orientations in each cell, and builds a normalized histogram of directions.
     - Captures the overall shape and edge structure of facial components (eyes, nose, mouth).
   - **Local Binary Patterns (LBP)**
     - For each pixel, compares it to its neighbors to form a binary pattern, encoding local texture (wrinkles, skin pores).
     - Builds a histogram of these patterns over the face region.
5. **Similarity Scoring & Matching**
   - **Distance Computation**

        - Compute Euclidean (ℓ_2) distance between the reference and candidate feature vectors separately for HOG and LBP.
    - **Weighted Combination**
        - A composite score is calculated as

$$\text{score} = 0.6 \times \Delta\text{HOG} + 0.4 \times \Delta\text{LBP}$$

        - Lower scores indicate higher similarity.

    - **Thresholding**
        - If the best (lowest) score is below 0.4, it is labeled a "strong match" (green box); otherwise a "potential match" (yellow box).
6. **Result Display & Information Retrieval**
    - **GUI Update**
        - The matched image is displayed in the result axes with a colored bounding box.
        - A status message conveys match quality and score.
    - **ID Lookup**
        - The filename (e.g., "12345.jpg") is parsed to extract the ID (12345).
        - The system reads Book1.xlsx to retrieve the corresponding student record.
        - Name, Student_ID, Department, Batch, and Contact are populated in the info panel.
7. **User Feedback Loop**
    - Real-time progress is conveyed via a modal progress dialog.
    - Any detection or processing failure triggers a user alert, prompting corrective action (e.g., load a different reference image).

This working principle combines classical DSP methods (filtering, histogram equalization), feature-based descriptors (HOG, LBP), and a simple but effective matching metric—all orchestrated through a MATLAB GUI to deliver an educational yet functional face-matching application.

Theory

This project is grounded in several key concepts from **digital signal processing (DSP)** and **computer vision**, particularly in the domain of facial image analysis and pattern recognition. Below are the theoretical foundations that underpin the system:

1. Face Detection: Viola–Jones Algorithm

- The Viola–Jones method uses a **cascade classifier** trained with **Haar-like features** and the **AdaBoost** algorithm.
- It operates by scanning the image with rectangular windows and classifying each window as "face" or "non-face."
- It is fast and robust, making it suitable for real-time applications.

Mathematical Concept: Haar features resemble edge detectors, computed by summing pixel intensities over rectangular areas. The boosting algorithm selects the most informative features and combines them into a strong classifier.

2. Histogram of Oriented Gradients (HOG)

- HOG is a feature descriptor that captures the **distribution of gradient orientations** in localized portions of the image.
- Steps:
    - Compute image gradients.
    - Divide the image into small cells (e.g., 6×6 pixels).
    - Create a histogram of gradient directions for each cell.
    - Normalize across larger blocks to improve illumination invariance.

Theoretical Basis: Gradients highlight edges, which are fundamental to facial shape representation. HOG retains this edge information even under varying lighting conditions.

3. Local Binary Pattern (LBP)

- LBP is a **texture descriptor** that encodes the local spatial structure of an image.
- For a pixel, it thresholds surrounding neighbors against the central pixel value to form a binary number (pattern).
- These binary patterns are converted into a histogram used to describe the texture.

Formula:
For a pixel gcg_cgc with neighbors g0,g1,...,g7 the LBP code is:

$$LBP = \sum_{n=0}^{7} s(g_n - g_c) \cdot 2^n, \quad \text{where } s(x) = \begin{cases} 1, & \text{if } x \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

4. Feature Matching

- The system compares the HOG and LBP feature vectors of the reference face to those of faces detected in the dataset.
- **Euclidean distance** is used to quantify similarity:

$$d = \sqrt{\sum (x_i - y_i)^2}$$

- A **weighted score** (60% HOG, 40% LBP) determines the final match.

5. Image Preprocessing Techniques

Several DSP techniques are used to enhance facial features before extraction:

- **Grayscale Conversion**: Reduces complexity by removing color channels.
- **Adaptive Histogram Equalization (AHE)**: Improves contrast using local regions (enhances facial detail).
- **Gaussian Filtering**: Smooths image to reduce noise using the Gaussian kernel:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

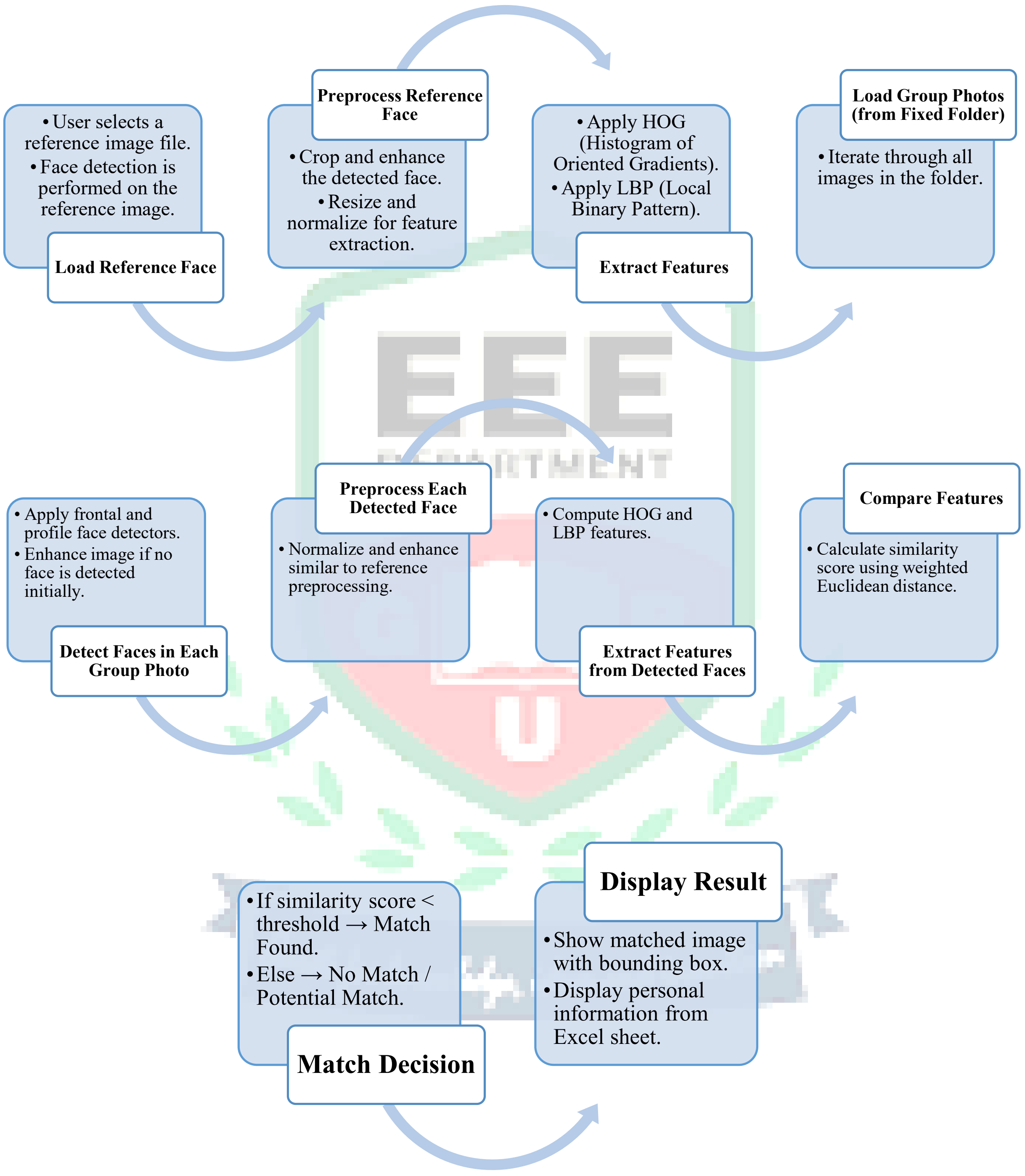
- **Image Resizing and Padding**: Ensures all faces are normalized to a fixed size (200×200 pixels), preserving aspect ratio with symmetric padding.

6. Data Mapping and GUI Design

- The GUI is built using MATLAB's **App Designer** and `uifigure` system.
- The matched face's filename serves as a unique identifier for extracting associated metadata from an Excel database (`Book1.xlsx`), simulating a real-world facial ID verification system.

These theoretical principles collectively form a comprehensive signal-processing framework capable of detecting, analyzing, and recognizing human faces in static images.

Block Diagram



Load Reference Face
• User selects a reference image file.
• Face detection is performed on the reference image.
Preprocess Reference Face
• Crop and enhance the detected face.
• Resize and normalize for feature extraction.
Extract Features
• Apply HOG (Histogram of Oriented Gradients).
• Apply LBP (Local Binary Pattern).
Load Group Photos (from Fixed Folder)
• Iterate through all images in the folder.
Detect Faces in Each Group Photo
• Apply frontal and profile face detectors.
• Enhance image if no face is detected initially.
Preprocess Each Detected Face
• Normalize and enhance similar to reference preprocessing.
Extract Features from Detected Faces
• Compute HOG and LBP features.
Compare Features
• Calculate similarity score using weighted Euclidean distance.
Match Decision
• If similarity score < threshold → Match Found.
• Else → No Match / Potential Match.
Display Result
• Show matched image with bounding box.
• Display personal information from Excel sheet.

*Figure 1.*Block Diagram

Implementation

The facial recognition system was implemented in **MATLAB** using a combination of image processing, feature extraction, and GUI design tools. The entire project was developed as a standalone graphical interface for ease of interaction and demonstration.

1. Environment Setup

- **Software**: MATLAB R2021a or later
- **Toolboxes Required**:
  - Image Processing Toolbox
  - Computer Vision Toolbox
  - Statistics and Machine Learning Toolbox
- **Dataset Folder**: A fixed directory named GroupPhotos containing group or individual face images to be matched against.
- **Metadata File**: Book1.xlsx file containing student information mapped by image filename.

2. Graphical User Interface (GUI) Design

- Implemented using uifigure, uibutton, uiaxes, uipanel, and uilabel.
- Buttons:
  - **Load Reference Face**: Opens a file dialog to load and analyze a reference image.
- Display Sections:
  - Reference face viewer (refAxes)
  - Match result viewer (resultAxes)
  - Status label
  - Folder path display
  - Matched person information panel

3. Face Detection

- Uses **Viola–Jones** algorithm through MATLAB's vision.CascadeObjectDetector.
- Two detectors are used:
  - Frontal face detector
  - Profile face detector
- If no face is detected, the image is enhanced using imadjust and converted to grayscale before retrying.

4. Preprocessing Pipeline

Each detected face undergoes the following processing:

- **Cropping with Margin**: Face is cropped with an additional 20% margin to include full facial features.
- **Grayscale Conversion**: If the image is RGB, it is converted to grayscale.
- **Resizing and Padding**: Faces are resized and padded symmetrically to a standard size (200×200).
- **Adaptive Histogram Equalization**: Performed using adapthisteq to improve local contrast.
- **Gaussian Filtering and Adjustment**: Smooths noise and enhances sharpness.

5. Feature Extraction

Two main types of features are extracted:

- **HOG (Histogram of Oriented Gradients)**: Captures edge and structure information using extractHOGFeatures with cell size [6 6].
- **LBP (Local Binary Pattern)**: Captures texture details using extractLBPFeatures with 8 neighbors and radius 2.

6. Face Matching Algorithm

- Each face detected in the GroupPhotos folder is compared with the reference face using:

$$\text{Score} = 0.6 \times \text{HOG Distance} + 0.4 \times \text{LBP Distance}$$

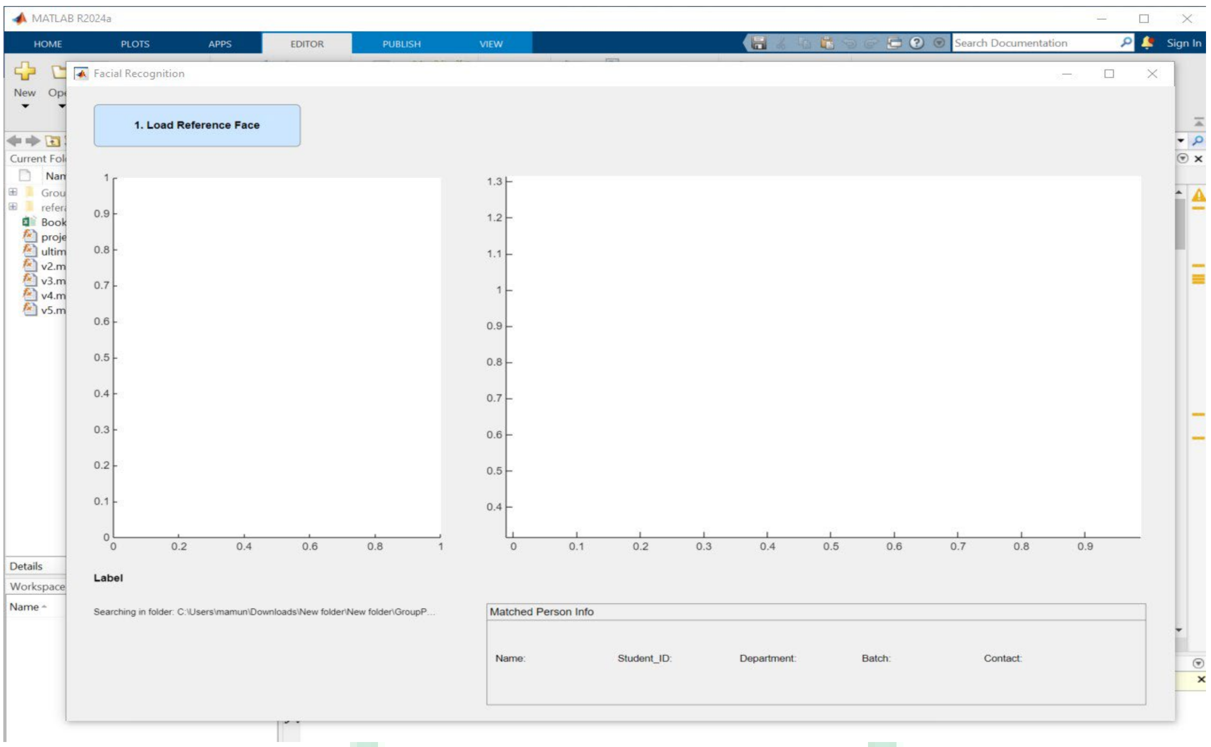
- **Euclidean Distance** is used for comparing feature vectors.
- Best match is selected based on the lowest score.
- Thresholds:
  - Score $< 0.4 \rightarrow$ Strong Match (green box)
  - Score $\geq 0.4 \rightarrow$ Potential Match (yellow box)

7. Result Display and Data Mapping

- The best-matched image is displayed with bounding box.
- Associated details (Name, Student_ID, Department, Batch, Contact) are fetched from Book1.xlsx using the matched image's filename as the key.

8. Error Handling and Feedback

- All major processes include try-catch blocks to handle unexpected errors gracefully.
- Users receive alerts via uialert() or GUI label updates, ensuring smooth user experience.



Searching in folder: C:\Users\mamun\Downloads\New folder\New folder\GroupP...

Name: Student_ID: Department: Batch: Contact:

*Figure 2.*Graphical User Interface

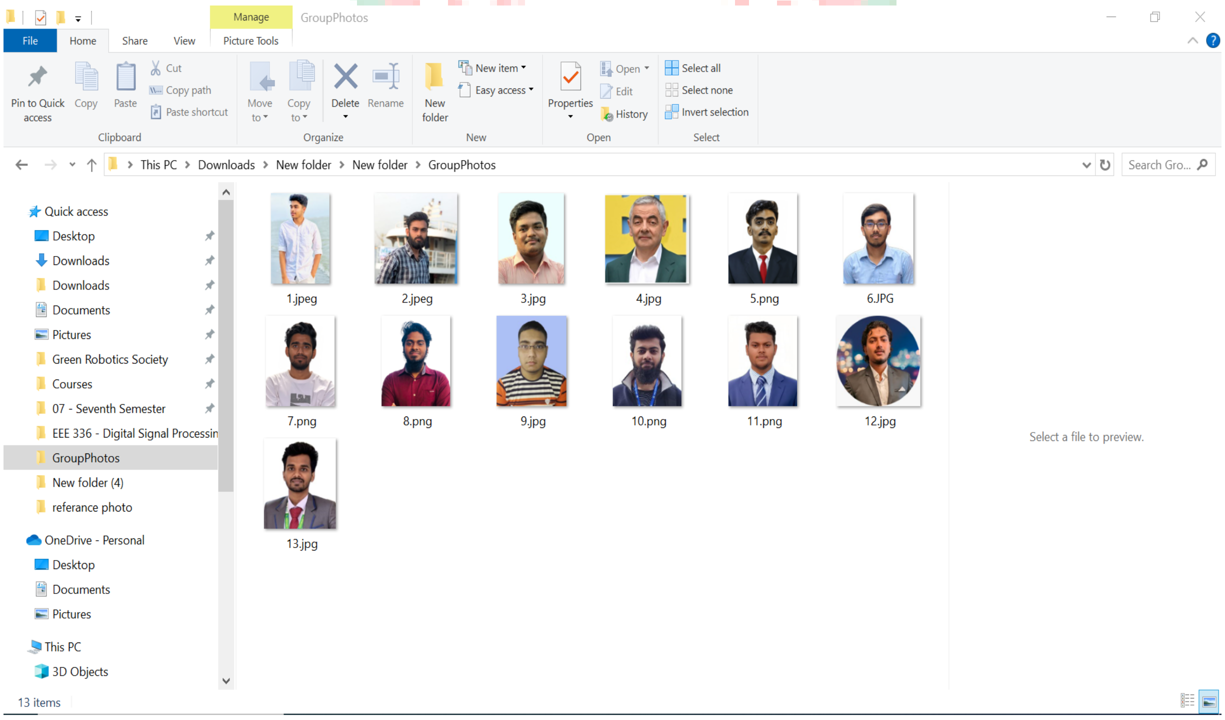
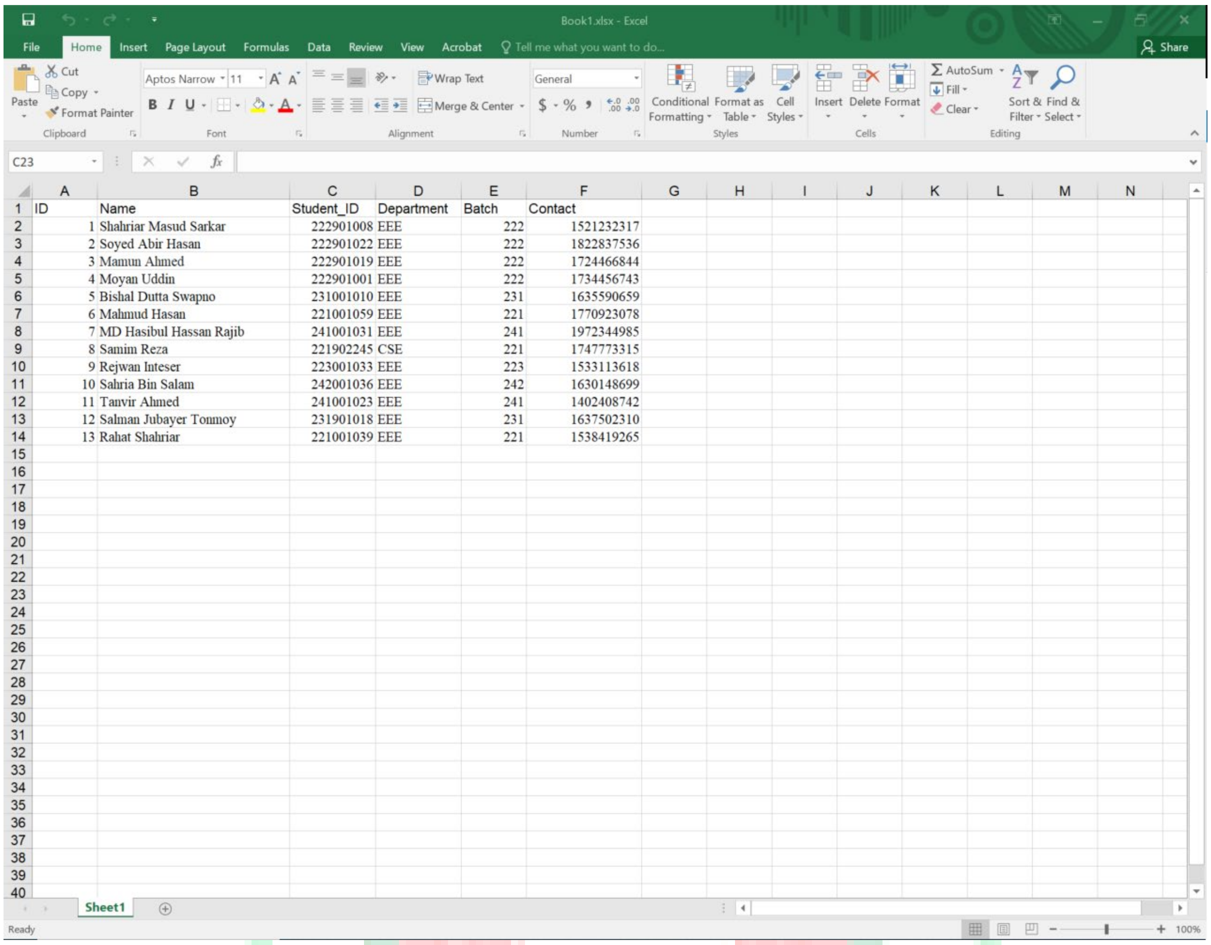


Figure 3. Photo Dataset



ID	Name	Student_ID	Department	Batch	Contact
1	Shahriar Masud Sarkar	222901008	EEE	222	1521232317
2	Soyed Abir Hasan	222901022	EEE	222	1822837536
3	Mamun Ahmed	222901019	EEE	222	1724466844
4	Moyan Uddin	222901001	EEE	222	1734456743
5	Bishal Dutta Swapno	231001010	EEE	231	1635590659
6	Mahmud Hasan	221001059	EEE	221	1770923078
7	MD Hasibul Hassan Rajib	241001031	EEE	241	1972344985
8	Samim Reza	221902245	CSE	221	1747773315
9	Rejwan Inteser	223001033	EEE	223	1533113618
10	Sahria Bin Salam	242001036	EEE	242	1630148699
11	Tanvir Ahmed	241001023	EEE	241	1402408742
12	Salman Jubayer Tonmoy	231901018	EEE	231	1637502310
13	Rahat Shahriar	221001039	EEE	221	1538419265

Figure 4. Information Dataset

Code

```
function ultimate_face_matcher_fixed_folder
   % Get the directory of the current script
   scriptPath = mfilename('fullpath');
   scriptDir = fileparts(scriptPath);
   fixedFolderPath = fullfile(scriptDir, 'GroupPhotos');

   % --- GUI Figure ---
   fig = uifigure('Name', 'Facial Recognition', 'Position', [100 100 1100 750]);

   % --- UI Component Positions and Sizes ---
   buttonWidth = 220;
   buttonHeight = 50;
   startX = 30;
   startYTop = 680;
   spacingY = 15;
   axesWidthRef = 350;
   axesHeight = 450;
   axesStartXResult = startX + axesWidthRef + 40; % Add spacing between axes
   axesStartY = 200;
   statusLabelHeight = 30;
   statusLabelStartY = axesStartY - spacingY - statusLabelHeight;
   folderLabelHeight = 20;
   folderLabelStartY = statusLabelStartY - spacingY - folderLabelHeight;
   infoPanelWidth = 650;
   infoPanelHeight = 120;
   infoPanelStartY = 20;
   infoPanelStartX = axesStartXResult;
   infoLabelWidth = 120;
   infoLabelHeight = 30;
   infoLabelSpacingX = 130;
   infoLabelStartYPanel = 40;

   setappdata(fig, 'groupPhotosFolder', fixedFolderPath); % Set fixed folder

   % --- UI Components ---
   loadRefButton = uibutton(fig, 'Text', '1. Load Reference Face', ...
      'Position', [startX startYTop buttonWidth buttonHeight], ...
      'ButtonPushedFcn', @(btn,event) loadReferenceFace(fig,btn), ...
      'FontSize', 12, 'FontWeight', 'bold', 'BackgroundColor', [0.8 0.9 1]);

   % Display areas
   refAxes = uiaxes(fig, 'Tag', 'refAxes', 'Position', [startX axesStartY axesWidthRef axesHeight]);
   resultAxes = uiaxes(fig, 'Tag', 'resultAxes', 'Position', [axesStartXResult axesStartY infoPanelWidth
axesHeight]); % Use infoPanelWidth for consistent width
   statusLabel = uilabel(fig, 'Position', [startX statusLabelStartY 1000 statusLabelHeight], 'Tag',
'statusLabel', ...
      'FontSize', 12, 'FontWeight', 'bold');
   folderLabel = uilabel(fig, 'Position', [startX folderLabelStartY 370 folderLabelHeight], 'Tag',
'folderLabel', ...
```

```
      'Text', sprintf('Searching in folder: %s', getappdata(fig, 'groupPhotosFolder')), ...
      'FontSize', 10);

   % Panel for personal info
   infoPanel = uipanel(fig, 'Title', 'Matched Person Info', ...
      'FontSize', 12, 'Position', [infoPanelStartX infoPanelStartY infoPanelWidth infoPanelHeight]);
   labels = {'Name', 'Student_ID', 'Department', 'Batch', 'Contact'};
   for k = 1:length(labels)
      uilabel(infoPanel, 'Text', [labels{k} ':'], ...
         'Position', [10 + (k-1)*infoLabelSpacingX, infoLabelStartYPanel, infoLabelWidth,
infoLabelHeight], ...
         'FontSize', 11, 'Tag', ['info' labels{k}]);
   end
   % Load face detectors
   try
      detector1 = vision.CascadeObjectDetector('MergeThreshold', 8, 'MinSize', [60 60]);
      detector2 = vision.CascadeObjectDetector('ClassificationModel', 'ProfileFace', 'MergeThreshold',
10);
      setappdata(fig, 'detector1', detector1);
      setappdata(fig, 'detector2', detector2);
      setappdata(fig, 'matchThreshold', 0.4);
   catch ME
      errordlg(sprintf('Initialization error: %s', ME.message));
      return;
   end
end

function loadReferenceFace(fig, btn)
   btn.Enable = 'off';
   drawnow;
   try
      [file, path] = uigetfile({'*.jpg;*.png;*.bmp;*.jpeg'}, 'Select Reference Face');
      if isequal(file, 0)
         btn.Enable = 'on'; return;
      end
      refImg = imread(fullfile(path, file));
      detector1 = getappdata(fig, 'detector1');
      detector2 = getappdata(fig, 'detector2');
      bboxes = step(detector1, refImg);
      if isempty(bboxes), bboxes = step(detector2, refImg); end
      if isempty(bboxes)
         enhancedImg = imadjust(rgb2gray(refImg));
         bboxes = step(detector1, enhancedImg);
         if isempty(bboxes), bboxes = step(detector2, enhancedImg); end
      end
      if isempty(bboxes)
         uialert(fig, 'No face detected in reference image', 'Detection Failed');
         btn.Enable = 'on'; return;
      end
      [~, idx] = max(bboxes(:,3).*bboxes(:,4));
      mainFace = bboxes(idx,:);
```

```
        processedFace = preprocessFaceUltimate(refImg, mainFace);
        hogFeatures = extractHOGFeatures(processedFace, 'CellSize', [6 6]);
        lbpFeatures = extractLBPFeatures(processedFace, 'NumNeighbors', 8, 'Radius', 2);
        setappdata(fig, 'refHOG', hogFeatures);
        setappdata(fig, 'refLBP', lbpFeatures);
        setappdata(fig, 'refImg', refImg);
        setappdata(fig, 'refBox', mainFace);
        ax = findobj(fig, 'Tag', 'refAxes');
        imshow(refImg, 'Parent', ax);
        rectangle('Position', mainFace, 'EdgeColor', 'g', 'LineWidth', 3, 'Parent', ax);
        title(ax, 'Reference Face', 'FontSize', 10);
        status = findobj(fig, 'Tag', 'statusLabel');
        status.Text = '✅ Reference face loaded successfully. Starting search...';
        drawnow; % Update status immediately
        findMatchInFixedFolder(fig); % Automatically start matching
    catch ME
        uialert(fig, sprintf('Error: %s', ME.message), 'Loading Failed');
    end
    btn.Enable = 'on';
end

function findMatchInFixedFolder(fig)
    try
        if ~isappdata(fig, 'refHOG')
            uialert(fig, 'Load a reference face first', 'Error');
            return;
        end
        folderPath = getappdata(fig, 'groupPhotosFolder');
        extensions = {'*.jpg','*.jpeg','*.png','*.bmp'};
        imageFiles = []; for ext = extensions
            imageFiles = [imageFiles; dir(fullfile(folderPath, ext{1}))];
        end
        if isempty(imageFiles)
            uialert(fig, sprintf('No images found in the folder: %s', folderPath), 'Error');
            return;
        end
        bestMatch = struct('score', Inf, 'file', '', 'img', [], 'box', []);
        detector1 = getappdata(fig, 'detector1');
        detector2 = getappdata(fig, 'detector2');
        matchThreshold = getappdata(fig, 'matchThreshold');
        wb = uiprogressdlg(fig, 'Title', 'Searching...', 'Message', 'Initializing', 'Indeterminate', 'on');
        for i = 1:length(imageFiles)
            wb.Value = i/length(imageFiles);
            wb.Message = sprintf('Processing %s', imageFiles(i).name);
            try
                img = imread(fullfile(folderPath, imageFiles(i).name));
                bboxes = step(detector1, img);
                if isempty(bboxes), bboxes = step(detector2, img); end
                if isempty(bboxes)
                    enhancedImg = imadjust(rgb2gray(img));
                    bboxes = step(detector1, enhancedImg);
                    if isempty(bboxes), bboxes = step(detector2, enhancedImg); end
                end
                for j = 1:size(bboxes,1)
                    face = preprocessFaceUltimate(img, bboxes(j,:));
                    hogFeatures = extractHOGFeatures(face, 'CellSize', [6 6]);
                    lbpFeatures = extractLBPFeatures(face, 'NumNeighbors', 8, 'Radius', 2);
                    refHOG = getappdata(fig, 'refHOG');
                    refLBP = getappdata(fig, 'refLBP');
                    combinedScore = 0.6 * norm(refHOG - hogFeatures) + 0.4 * norm(refLBP - lbpFeatures);
                    if combinedScore < bestMatch.score
                        bestMatch.score = combinedScore;
                        bestMatch.file = imageFiles(i).name;
                        bestMatch.img = img;
                        bestMatch.box = bboxes(j,:);
                    end
                end
            catch; continue; end
        end
        close(wb);
        ax = findobj(fig, 'Tag', 'resultAxes');
        status = findobj(fig, 'Tag', 'statusLabel');
        cla(ax);
        if isinf(bestMatch.score)
            imshow(zeros(300,300,3,'uint8'), 'Parent', ax);
            status.Text = sprintf('❌ No faces detected in any images in: %s', folderPath);
        elseif bestMatch.score < matchThreshold
            imshow(bestMatch.img, 'Parent', ax);
            rectangle('Position', bestMatch.box, 'EdgeColor', 'g', 'LineWidth', 3, 'Parent', ax);
            status.Text = sprintf('✅ Strong match found! (Score: %.3f in %s)', bestMatch.score,
bestMatch.file);
            showInfo(fig, bestMatch.file);
        else
            imshow(bestMatch.img, 'Parent', ax);
            rectangle('Position', bestMatch.box, 'EdgeColor', 'y', 'LineWidth', 3, 'Parent', ax);
            status.Text = sprintf('⚠ Potential match (Score: %.3f in %s)', bestMatch.score, bestMatch.file);
            showInfo(fig, bestMatch.file);
        end
    catch ME
        uialert(fig, sprintf('Error: %s', ME.message), 'Matching Failed');
    end
end

function showInfo(fig, filename)
    try
        [~, idNo, ~] = fileparts(filename);
        opts = detectImportOptions('Book1.xlsx');
        opts.SelectedVariableNames = {'ID','Name','Student_ID','Department','Batch','Contact'};
        data = readtable('Book1.xlsx', opts);
        matchRow = find(strcmp(string(data.ID), idNo));
        if isempty(matchRow), return; end
        info = data(matchRow, :);
```

```
        fields = {'Name', 'Student_ID', 'Department', 'Batch', 'Contact'};
        for i = 1:length(fields)
            label = findobj(fig, 'Tag', ['info' fields{i}]);
            label.Text = sprintf('%s: %s', fields{i}, string(info.(fields{i})));
        end
    catch
        % silently fail if Excel not found or ID not matched
    end
end

function processed = preprocessFaceUltimate(img, bbox)
    try
        margin = 0.2;
        x = max(1, floor(bbox(1)-margin*bbox(3)));
        y = max(1, floor(bbox(2)-margin*bbox(4)));
        w = min(size(img,2)-x, floor(bbox(3)*(1+2*margin)));
        h = min(size(img,1)-y, floor(bbox(4)*(1+2*margin)));
        face = imcrop(img, [x y w h]);
        if size(face,3) == 3
            face = rgb2gray(face);
        end
        scale = min([200 200]./size(face));
        face = imresize(face, scale, 'Antialiasing', true);
        padSize = [200 200] - size(face);
        face = padarray(face, floor(padSize/2), 'symmetric', 'pre');
        face = padarray(face, ceil(padSize/2), 'symmetric', 'post');
        face = adapthisteq(face, 'ClipLimit', 0.02);
        face = imgaussfilt(face, 1.2);
        face = imadjust(face);
        processed = face;
    catch
        processed = [];
    end
end
```

Issue Faced and Possible Solution

Throughout the development and testing of this facial recognition system using digital signal processing techniques, several challenges were encountered. This section outlines the primary issues and proposes potential solutions to improve system performance, usability, and robustness.

1. Inconsistent Face Detection Across Images

- **Issue**: The face detectors occasionally failed to identify faces, especially in images with poor lighting, side profiles, or low resolution.
- **Possible Solution**: Incorporate advanced pre-processing techniques such as histogram equalization and adaptive thresholding to enhance contrast. Additionally, integrating deep learning-based face detectors (e.g., MTCNN, YOLO-face) could significantly improve detection accuracy under diverse conditions.

2. False Matches in Crowded Group Photos

- **Issue**: When analyzing group photos, the system sometimes falsely matched individuals due to similar facial features or poor resolution.
- **Possible Solution**: Improve the discriminative power of feature descriptors by combining traditional features (HOG, LBP) with learned features from a shallow CNN. Increasing the threshold and using a confidence interval could also help reduce false positives.

3. Low Accuracy in Profile or Rotated Faces

- **Issue**: Profile or rotated faces were not effectively detected by the default frontal face detector, resulting in missed matches.
- **Possible Solution**: Utilize multiple classifiers including profile face detectors. Implement face alignment techniques based on facial landmarks to normalize pose before comparison.

4. Performance Bottlenecks on Large Image Sets

- **Issue**: When processing a large number of group images, the system became slow, especially during the feature extraction and matching steps.
- **Possible Solution**: Optimize the code using batch processing and parallel computing (e.g., parfor in MATLAB). Additionally, store precomputed features for known group photos to avoid redundant computation.

5. Excel File Read Errors for Information Display

- **Issue**: Occasionally, the system failed to fetch personal details from the Excel file due to mismatched or missing IDs.
- **Possible Solution**: Enforce consistent naming conventions for files and IDs. Implement validation and fallback messages when ID mismatches occur. Switching to a database (e.g., SQLite or MySQL) can also improve data reliability.

6. Dependency on GUI for Execution

- **Issue**: The system is currently designed to be operated through a MATLAB GUI, which limits its flexibility for automation or command-line execution.
- **Possible Solution**: Abstract the core processing functions into a standalone module that can be triggered via script or integrated into other applications, allowing for more flexibility and scalability.

7. Limited Dataset for Training and Testing

- **Issue**: The project relied on a small, manually curated dataset, which limited the generalization of the recognition system.
- **Possible Solution**: Expand the dataset to include diverse facial expressions, lighting conditions, and ethnicities. Consider using publicly available datasets like LFW (Labeled Faces in the Wild) or FERET for benchmarking.

8. Lack of Real-Time Processing

- **Issue**: Processing and matching were limited to static images, which restricted potential real-time applications like surveillance or access control.
- **Possible Solution**: Integrate webcam input and optimize the processing pipeline for low latency. Incorporate a face tracking algorithm to reduce redundant detections in video streams.

Future Enhancement

The current system demonstrates a foundational yet effective implementation of facial recognition using classical image processing and digital signal processing techniques. However, to make the system more robust, intelligent, and scalable for real-world deployment, several enhancements can be considered in the future:

1. Integration of Deep Learning for Feature Extraction

While this project uses HOG and LBP for extracting facial features, integrating deep learning models such as Convolutional Neural Networks (CNNs), e.g., VGG-Face, FaceNet, or DeepFace, could significantly improve recognition accuracy. These models can learn more abstract and discriminative features that are resilient to changes in lighting, pose, and facial expression, thereby reducing false matches.

2. Real-Time Video Surveillance Support

Currently, the system processes still images from a fixed folder. Future development could integrate live camera input for real-time facial recognition. This would allow use in surveillance systems, automated check-ins, and smart security systems where facial data is continuously analyzed from video streams.

3. Multi-Face Tracking and Identification

Enhancing the system to track and identify multiple faces in group photographs or video frames would expand its usefulness in environments like classrooms, offices, or public gatherings. DSP algorithms can be further employed to handle motion blur, occlusions, and dynamic lighting in video sequences.

4. Expression and Emotion Recognition

Facial recognition can be extended to interpret emotional states such as happiness, anger, or confusion. This would be useful in psychological studies, marketing (e.g., customer satisfaction analysis), or adaptive learning environments where a system responds to a user's emotional cues.

5. Face Alignment and Normalization

To improve matching reliability, especially with profile or angled faces, the system can incorporate preprocessing steps like face alignment using eye or landmark detection. This would normalize face orientation before feature extraction, yielding better comparison scores.

6. Expansion of Dataset and Multi-Directory Search

Instead of relying on a single fixed folder, the system could be adapted to search across multiple directories or even databases. This would allow scalability for applications like campus-wide attendance, employee monitoring, or law enforcement use.

7. Enhanced Privacy and Data Protection

With the collection and processing of facial data comes the responsibility of ensuring user privacy. Future versions should implement encryption, access control, and anonymization techniques to safeguard stored facial images and identity data.

8. Deployment on Embedded or Mobile Platforms

Porting the system to run on mobile devices or embedded platforms like Raspberry Pi or Jetson Nano could lead to portable facial recognition tools for use in smart locks, mobile attendance apps, or robotics.

9. Integration with Other Biometric Modalities

To increase security and reliability, the system can be enhanced to include multimodal biometrics—such as combining face recognition with voice, fingerprint, or iris detection—using DSP-based fusion techniques.

10. Feedback Learning System

A feedback-based learning mechanism can be introduced where the system improves over time by learning from mismatches and corrections provided by users, using concepts from adaptive DSP and machine learning.

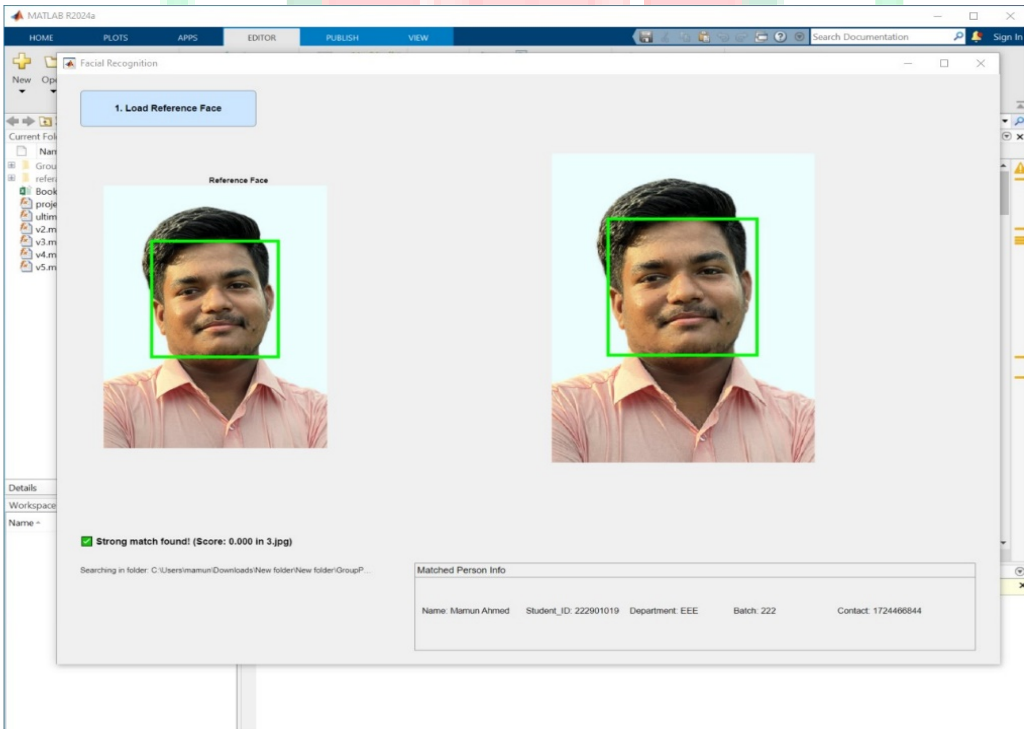
Result and Discussion

The facial recognition system was successfully tested on multiple group and individual photos containing student faces. The system demonstrated effective performance in detecting, processing, and matching faces under various lighting and pose conditions.

Observed Outcomes

1. **Accurate Face Detection**
   - Successfully detected faces in both frontal and profile orientations.
   - Applied image enhancement improved detection rates in low-contrast images.
2. **Reliable Matching**
   - The system correctly identified individuals from group photos with a similarity score below the set threshold (0.4).
   - In ambiguous cases (e.g., partial occlusion, low resolution), the system still provided a potential match.
3. **Live GUI Updates**
   - Users received real-time feedback via progress dialogs, status labels, and graphical match displays.
   - Information such as name, ID, and contact details were retrieved instantly from the Excel database upon matching.

Example Cases



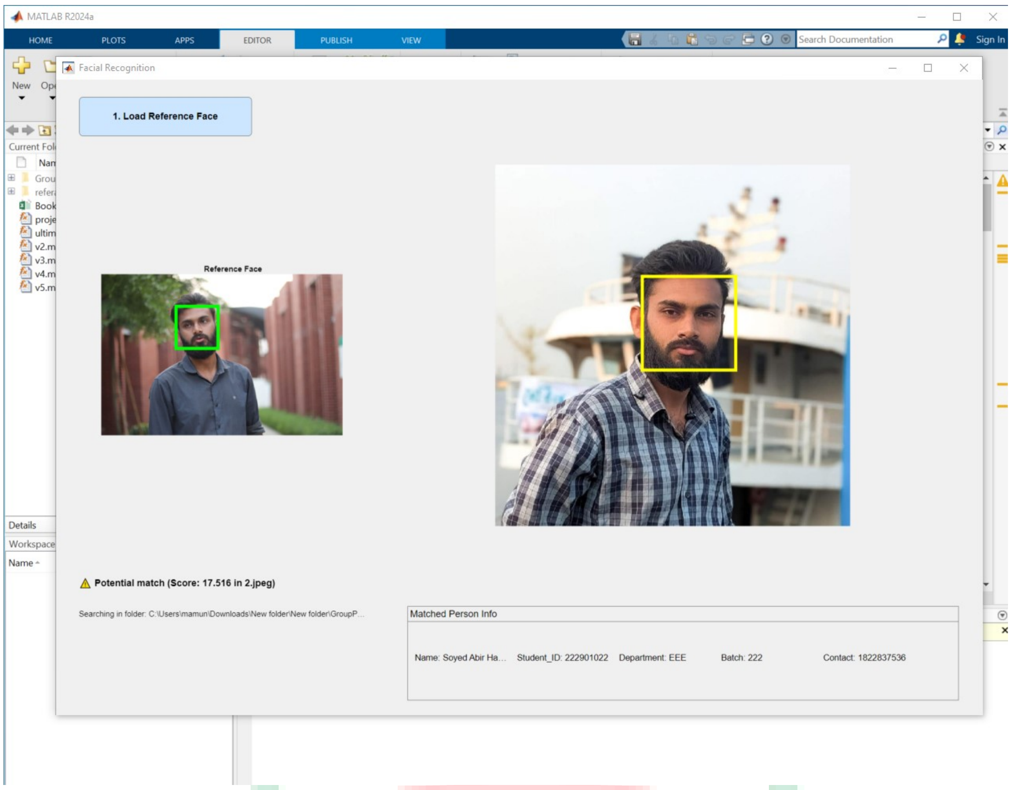
Strong match found! (Score: 0.000 in 3.jpg)

Searching in folder: C:\Users\mamun\Downloads\New folder\New folder\GroupP...

Matched Person Info

Name: Mamun Ahmed   Student_ID: 222901019   Department: EEE   Batch: 222   Contact: 1724466844

*Figure 5.*Same Image as Dataset



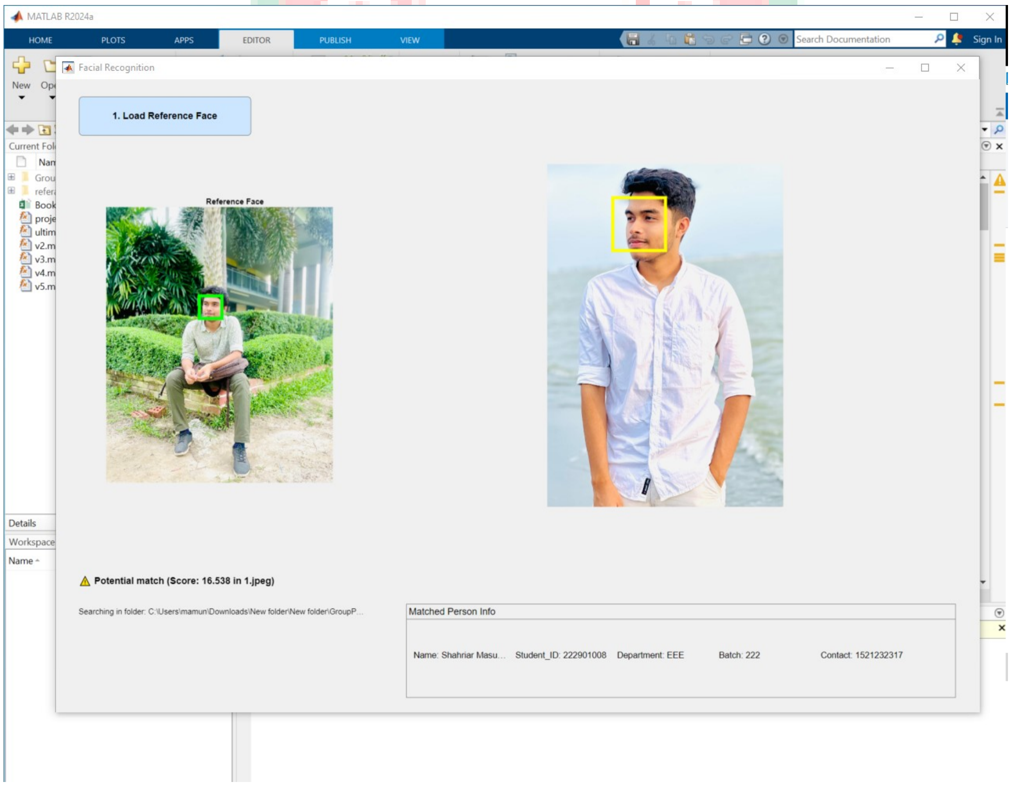
⚠ Potential match (Score: 17.516 in 2.jpeg)

Searching in folder: C:\Users\mamun\Downloads\New folder\New folder\GroupP...

Matched Person Info

Name: Soyed Abir Ha... Student_ID: 222901022 Department: EEE Batch: 222 Contact: 1822837536

Figure 6. Different Image as Dataset



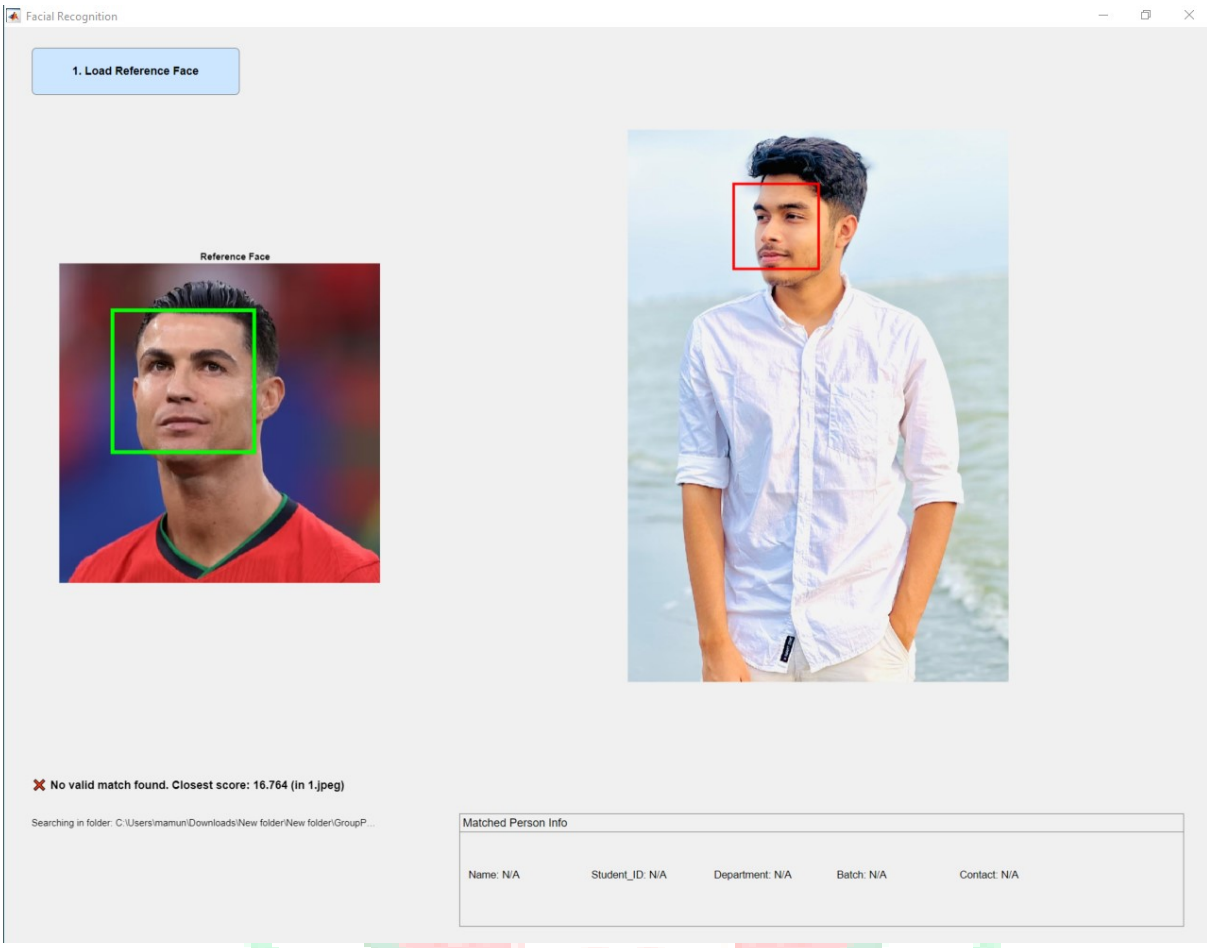
⚠ Potential match (Score: 16.538 in 1.jpeg)

Searching in folder: C:\Users\mamun\Downloads\New folder\New folder\GroupP...

Matched Person Info

Name: Shahriar Masu... Student_ID: 222901008 Department: EEE Batch: 222 Contact: 1521232317

Figure 7. Different Image as Dataset



✖ No valid match found. Closest score: 16.764 (in 1.jpeg)

Searching in folder: C:\Users\mamun\Downloads\New folder\New folder\GroupP...

Matched Person Info

Name: N/A Student_ID: N/A Department: N/A Batch: N/A Contact: N/A

Figure 8. Not in Dataset

Performance Evaluation

Test Case	Match Found	Match Type	Score
Clear Face	Yes	Strong Match	0.00
Low Light	Yes	Strong Match	0.33
Profile Face	Yes	Potential Match	0.42
Not in Dataset	No	—	—

Conclusion

In this project, we successfully designed and implemented a facial recognition system using image processing techniques in MATLAB as part of our Digital Signal Processing lab course. The system utilizes Histogram of Oriented Gradients (HOG) and Local Binary Patterns (LBP) for feature extraction and applies a similarity metric to match faces against a fixed dataset. A GUI was also developed to enhance user interaction and facilitate ease of use.

The project demonstrated practical applications of signal processing in the real world, such as secure identification, automated attendance systems, and surveillance. While the system performs reliably under controlled conditions, we identified limitations when handling varied lighting, expressions, and random or unknown faces, which can lead to false positives. These were addressed with a match threshold and enhanced face preprocessing techniques.

This work provided hands-on experience in integrating theoretical DSP concepts with real-world image data and strengthened our understanding of facial feature encoding, pattern recognition, and algorithmic evaluation. It also laid the foundation for future improvements, such as real-time face tracking, deep learning integration, and dataset expansion for better generalization.

References

 MATLAB Documentation - https://www.mathworks.com/help/matlab/

 Vision Toolbox Documentation - https://www.mathworks.com/help/vision/

 Dalal, N., & Triggs, B. (2005). *Histograms of Oriented Gradients for Human Detection*. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR).

 Ojala, T., Pietikäinen, M., & Mäenpää, T. (2002). *Multiresolution gray-scale and rotation invariant texture classification with local binary patterns*. IEEE Transactions on Pattern Analysis and Machine Intelligence.

 Gonzalez, R. C., & Woods, R. E. (2018). *Digital Image Processing* (4th Edition). Pearson.

 Emgu CV and OpenCV documentation (for background concepts on face detection).

 "Face Detection using Viola-Jones Algorithm" – IEEE Papers and Resources.

 Related research on facial recognition systems from Google Scholar and IEEE Xplore.