ABSTRACT

The security and accuracy of voting systems are crucial for ensuring free and fair elections. Traditional voting methods, both paper-based and electronic, are often prone to issues such as voter fraud, impersonation, and tampering. This project proposes a biometric-based voting machine leveraging image processing techniques to address these challenges. The system uses biometric traits, such as fingerprints or facial recognition, for voter authentication. Image processing methods, including preprocessing, feature extraction, and pattern matching, are employed to ensure accurate and reliable identification.

The system is implemented using MATLAB, incorporating a user-friendly interface for voters to cast their votes securely. Once authenticated, the voter can select their preferred candidate, and the vote is securely stored in an encrypted database, ensuring data integrity. The proposed system enhances voting efficiency, reduces the risk of fraud, and ensures a transparent election process. With the growing importance of security and trust in democratic systems, this project demonstrates how advanced biometric and image processing technologies can revolutionize voting systems.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 Overview of the Project

Voting is a fundamental process in any democratic system, ensuring that citizens can express their preferences in governance. Over the years, voting methods have evolved from paper-based systems to electronic voting machines, and now, towards advanced biometric systems. However, traditional voting systems often face challenges such as voter fraud, impersonation, ballot tampering, and logistical inefficiencies. These issues compromise the integrity of elections and erode public trust in the electoral process.

To address these challenges, this project proposes a voting machine based on image processing techniques. The system integrates biometric authentication, leveraging the unique physical traits of voters, such as fingerprints or facial features, for secure and accurate voter verification. Image processing plays a pivotal role in this system, as it enables the capture, analysis, and matching of biometric data to ensure reliable voter identification.

This project focuses on implementing a prototype of the biometric voting system using MATLAB. The proposed system ensures transparency, security, and efficiency in the voting process, making it a robust alternative to traditional systems.

1.2 Objectives

The main objectives of this project are:

1. To design a secure and efficient voting system using biometric authentication.
2. To employ image processing techniques for real-time voter verification.
3. To eliminate common voting malpractices such as multiple voting and impersonation.
4. To develop a user-friendly interface that simplifies the voting process for voters and administrators.
5. To ensure the confidentiality and integrity of votes through secure data storage and encryption.

These objectives aim to create a secure and transparent system that upholds the principles of democracy and electoral integrity.

1.3 Importance of Biometrics in Voting

Biometrics refers to the use of unique physical or behavioral characteristics for identification and authentication. In the context of voting, biometrics significantly enhances security by ensuring that only registered voters can participate, and each voter can cast only one vote. Biometric systems

are difficult to forge or bypass, making them highly effective in preventing voter fraud.

Image processing techniques further augment the reliability of biometric systems by enabling the accurate capture, enhancement, and matching of biometric data. Common biometric modalities used in voting systems include:

1. Fingerprint Recognition: Analyses ridge patterns unique to each individual.
2. Facial Recognition: Maps facial features such as the distance between eyes, nose, and mouth.
3. Iris Scanning: Examines the intricate patterns in the iris of the eye.

Countries like India, Ghana, and Estonia have implemented biometric-based voting systems to address electoral challenges. These systems have shown significant improvements in voter authentication accuracy, reduced fraud, and increased public trust.

This introduction provides a detailed background on the project, highlights its objectives, and emphasizes the importance of biometrics in enhancing the security and efficiency of voting systems.

1.4 Key Features

1.Biometric Authentication

- The system uses unique biometric traits such as fingerprints or facial features for voter identification.
- Prevents impersonation and ensures that only registered voters can participate in the election.

2. Image Processing Techniques

- Advanced image processing methods, including preprocessing, feature extraction, and pattern matching, are employed for accurate voter verification.
- Enhances the reliability and accuracy of the biometric data analysis.

3. User-Friendly Interface

- A simple and intuitive graphical user interface (GUI) allows voters to interact with the system seamlessly.
- Guides voters through the process of authentication and vote casting with minimal technical knowledge required.

4. Real-Time Verification

- Processes biometric data in real-time, enabling quick and efficient voter authentication.
- Reduces wait times during elections and improves overall voting experience.

5. Secure Data Storage

- All biometric data and votes are stored securely using encryption techniques.

- Ensures the confidentiality and integrity of the votes, preventing tampering or unauthorized access.

6. Fraud Prevention
   - Eliminates common electoral malpractices such as multiple voting, fake registrations, and proxy voting.
   - Ensures that each voter can cast only one vote.

7. Scalability
   - The system can be scaled to accommodate large voter populations, making it suitable for local, national, or global elections.

8. Efficient Vote Counting
   - Automates the vote counting process, reducing human error and delivering faster results.

9.Audit Trail
   - Maintains a secure log of voter authentication and vote casting for transparency and accountability.
   - Helps in post-election audits to verify the integrity of the election process.

10. Cost-Effective
    - Reduces the need for extensive manpower and physical resources associated with traditional voting systems.
    - Offers a long-term solution with minimal maintenance costs.

These key features make the proposed voting machine a secure, efficient, and reliable solution for modern electoral challenges.

CHAPTER 2

Literature Survey

The literature survey highlights existing research and systems in the domain of voting technologies, focusing on image processing and biometric authentication for secure and efficient voting.

2.1 Existing Voting Systems

1. Paper-Based Voting
   - Paper ballots have been the traditional method of voting.
   - Limitations: Prone to tampering, manual errors in counting, and logistical challenges during transport and storage.

2.Electronic Voting Machines (EVMs)
   - Widely adopted for faster vote counting and reduced logistical efforts.
   - Limitations: Vulnerable to hacking, lack of voter authentication mechanisms, and inability to prevent impersonation.

3. Online Voting
   - Promises convenience, especially for remote voters.
   - Limitations: Concerns about cybersecurity, voter authentication, and vote integrity.

4. Biometric-Based Voting
   - A more secure alternative that uses unique voter traits for authentication.
   - Benefits: Reduces fraud, ensures one-person-one-vote, and integrates seamlessly with digital systems.

2.2 Role of Image Processing in Security

Image processing is integral to biometric-based voting systems, ensuring accurate and efficient voter authentication. It involves the following steps:

1. Image Acquisition
   - Capturing biometric data such as fingerprints or facial images using scanners or cameras.

2. Preprocessing
   - Enhancing the quality of biometric images through noise removal, normalization, and segmentation.

3. Feature Extraction
   - Extracting unique traits from the images, such as fingerprint ridges or facial landmarks.

4. Pattern Matching
   - Comparing extracted features with stored templates to verify the voter's identity.

These techniques ensure that the system is robust against variations in image quality, lighting conditions, and user errors.

2.3 Biometric Techniques and Applications

Biometric systems rely on the uniqueness and permanence of physical traits for identification. The key techniques used in voting systems include:

1. Fingerprint Recognition
   - Captures and analyzes ridge patterns unique to an individual.
   - Widely used due to its high accuracy and availability of inexpensive scanners.
2. Facial Recognition
   - Uses facial landmarks such as the eyes, nose, and mouth for identification.
   - Advantageous in scenarios where physical contact is not feasible.
3. Iris Scanning
   - Maps the intricate patterns in the iris, offering extremely high accuracy.
   - Requires specialized hardware, making it less cost-effective for large-scale deployment.
4. Hybrid Systems
   - Combines multiple biometric modalities (e.g., fingerprint and facial recognition) for enhanced security.

Applications of biometric systems extend beyond voting to areas like banking, border control, and access management, showcasing their reliability and adaptability.

2.4 Research Studies on Biometric Voting Systems

1. Biometric Authentication in Voting
   - A study highlighted the efficiency of fingerprint recognition systems in preventing multiple voting and impersonation.
   - Demonstrated a False Acceptance Rate (FAR) of less than 2%.
2. Integration of Image Processing and Machine Learning
   - Research on facial recognition for voting systems used image processing and machine learning algorithms to improve accuracy and speed.
3. Global Implementations
   - India: The Aadhaar-based authentication system integrates fingerprints for voter identification.
   - Estonia: Combines online voting with secure biometric authentication.
4. Challenges Identified
   - Issues with poor-quality images, aging effects on biometrics, and spoofing attempts were noted.
   - Proposed solutions include advanced preprocessing techniques and multi-modal biometrics.

2.5 Gaps Identified in Existing Systems

1. Lack of robust security measures in traditional and electronic voting systems.

2. Limited implementation of image processing techniques in voter authentication.

3. Insufficient scalability of biometric systems for large-scale elections.

4. High costs of advanced biometric hardware.

By addressing these gaps, this project aims to create a cost-effective, scalable, and secure biometric voting system using image processing techniques.

This literature review provides a foundation for understanding the limitations of existing systems and the potential of biometric-based solutions.

2.6 Motivation

The motivation for developing a voting machine using image processing stems from the increasing need for secure, accurate, and efficient voting systems in modern democracies. Elections are the cornerstone of any democratic process, and the integrity of voting systems directly impacts public trust and governance. However, traditional and even many electronic voting systems continue to face significant challenges such as voter fraud, impersonation, and electoral manipulation. These issues undermine the credibility of election results, potentially leading to political instability and loss of public confidence in the democratic process.

This project is motivated by the following key factors:

1. Need for Enhanced Security

In many countries, the vulnerability of traditional voting methods to fraud, such as ballot stuffing, multiple voting, and identity theft, remains a pressing issue. Image processing and biometric systems provide a more secure way to authenticate voters, ensuring that each individual can cast only one vote, thus eliminating the risk of impersonation or fraudulent voting.

2. Rising Threats to Election Integrity

With the rise of digital technologies, electronic voting systems have been adopted by various nations. However, these systems are not immune to security breaches, including hacking and data tampering. Biometric authentication offers a way to mitigate these threats, making it significantly harder for unauthorized individuals to manipulate the election process. The integration of image processing enhances the robustness of these biometric systems by ensuring accurate and reliable voter identification.

3. Increasing Demand for Efficiency

Traditional voting methods are often slow and cumbersome, with manual counting and the potential for human error. This can lead to delays in the announcement of results, which can create

confusion and discontent among voters. By incorporating image processing into biometric systems, voting machines can ensure faster and more accurate verification of voters, leading to quicker results and a more efficient election process.

4.Technological Advancements

With advances in image processing algorithms, machine learning, and artificial intelligence, it has become increasingly feasible to implement robust, high-accuracy biometric authentication systems for voting. The ability to analyze fingerprints or facial features with high precision allows for the development of cost-effective and scalable biometric systems. As a result, such technology can be applied to large-scale elections, enhancing both security and efficiency.

5. Global Trends in Biometric Voting

Several countries, such as India with its Aadhaar system, and Estonia with its e-voting system, have begun integrating biometric authentication into their electoral processes. These examples demonstrate the growing acceptance of biometric systems for voting and highlight the potential for widespread adoption. This project aims to contribute to this global trend by developing a voting machine that combines image processing with biometric authentication to ensure a fair and transparent voting process.

6. Transparency and Trust

A secure, efficient, and transparent voting system fosters trust among voters, which is essential for a functioning democracy. Biometric-based voting machines, powered by image processing, can increase transparency by providing accurate voter verification and an audit trail that ensures accountability at every step of the voting process. This transparency helps build voter confidence, reducing the likelihood of election-related disputes.

By addressing these critical issues, this project seeks to create a reliable, secure, and efficient voting machine that leverages image processing techniques for biometric authentication. Ultimately, the goal is to contribute to the development of a next-generation voting system that upholds the principles of democracy, security, and trust.

This section outlines the driving forces behind the project, emphasizing the importance of improving security, efficiency, and transparency in the electoral process.

CHAPTER 3

INTRODUCTION TO IMAGE PROCESSING

3.1 IMAGE

An image is defined as a two-dimensional function F(x,y), where x and y are spatial coordinates, and the amplitude of F at any pair of coordinates (x,y) is called the intensity of that image at that point. When x,y, and amplitude values of F are finite, we call it a digital image.

In other words, an image can be defined by a two-dimensional array specifically arranged in rows and columns.

Digital Image is composed of a finite number of elements, each of which elements have a particular value at a particular location. These elements are referred to as picture elements, image elements, and pixels. A Pixel is most widely used to denote the elements of a Digital Image.

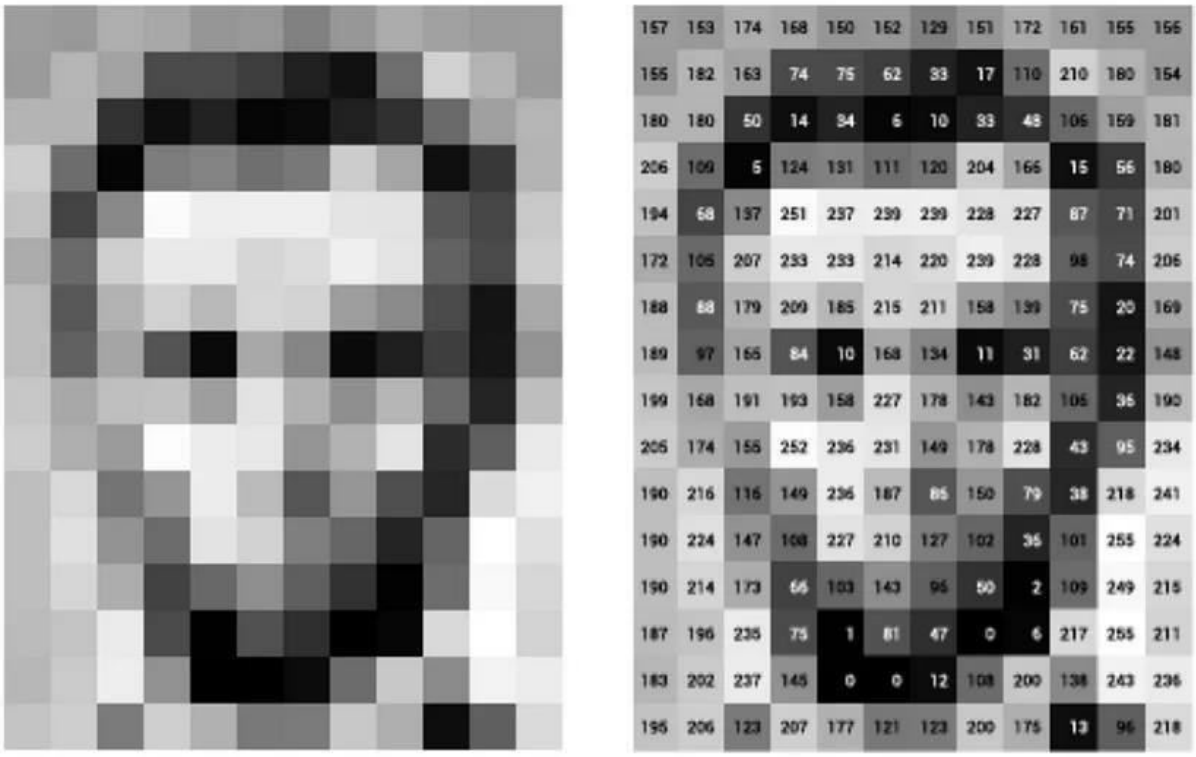

Fig3.1: Images and corresponding pixels

Image Processing

Image processing is a method to convert an image into digital form and perform some operations on it, in order to get an enhanced image or to extract some useful information from it. It is a type of signal dispensation in which input is image, like video frame or photograph and output may be image or characteristics associated with that image. Usually, Image Processing system includes treating images as two- dimensional signals while applying already set signal processing methods to them. It is among rapidly growing technologies today, with its applications in various aspects of a business. Image Processing forms core research area within engineering and computer science disciplines too.

Image processing basically includes the following three steps:

i) Importing the image with optical scanner or by digital photography.

ii) Analyzing and manipulating the image which includes data compression and image enhancement and spotting patterns that are not human eyes like satellite photographs.

iii) Output is the last stage in which results can be altered image or report that is based on image analysis.

Image processing actions can be grouped into three sub-areas:

Image compression, which reduces the memory requirements by removing the redundancy present in the image, that is, the image information which is not perceptible to the human eye.

Image pre-processing which consists of improving the visual quality of the image by reducing noise, pixel calibration and standardization, enhancing the edge detection, and making the image analysis step more reliable based on objective and well-established criteria. The term image pre-processing, in general, is referred to all manipulations on an image, each of which produces a new imag

3.2 Purpose of Image processing

The purpose of image processing is divided into 5 groups.

✓ Visualization – Observe the objects that are not visible.

✓ Image sharpening and restoration – To create a better image.

✓ Image retrieval – Seek for the image of interest.

✓ Measurement of pattern – Measures various objects in an image.

✓ Image Recognition – Distinguish the objects in an

3.3 BLOCK DIAGRAM OF IMAGE PROCESSING

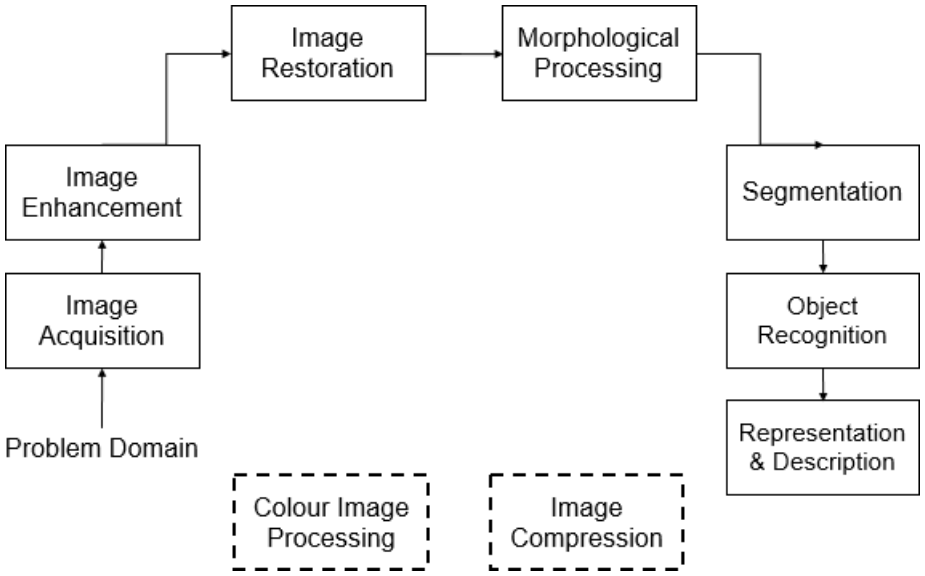

Fig 3.2: Flow Chart of Digital Image Processing

Detailed explain about flowchart:

- **Problem Domain to Image Acquisition:** This block represents the initial phase where a problem is identified and an image is captured as data to address or analyze the problem.
- **Image Acquisition to Image Enhancement:** Indicates the process of improving the quality of the acquired image, making it clearer and more defined for analysis.

- **Image Enhancement to Image Restoration:** Describes refining the enhanced image further, correcting flaws and restoring its quality.
- **Colour Image Processing:** This dashed block signifies processing specifically tailored for colour images, optimizing their quality and clarity.
- **Image Compression:** Indicates reducing the file size of images while maintaining their quality, facilitating easier storage and sharing.
- **Morphological Processing to Segmentation:** Represents transforming an image into segments making it easier for analysis or object recognition.
- **Segmentation to Object Recognition:** Denotes identifying distinct objects within segmented parts of an image for further analysis or action.

Object Recognition to Representation & Description: Illustrates interpreting recognized objects and describing them in understandable terms

3.4 Approach

Image processing is a method to convert an image into digital form and perform some operations on it, in order to get an enhanced image or to extract some useful information from it. It is a type of signal dispensation in which input is image, like video frame or photograph and output may be image or characteristics associated with that image. Usually, Image Processing system includes treating images as two- dimensional signals while applying already set signal processing methods set.

Image processing or pre-processing encompasses a broad range of operations, which may be treated as an end in themselves, or are intended to simplify or enhance subsequent analysis. Pre-processing improves the image data by removing unintended distortions or enhancing some image features that are important for further processing and creating a more suitable image than the original for specific application.

Benefits of Image Processing

Visualization helps in identification of the objects that are not visible.

Image processing is faster and cost effective, Noise free. Image sharpening and restoration – To create a better image.

CHAPTER 4
FLOW CHART

4.1 Flow Chart

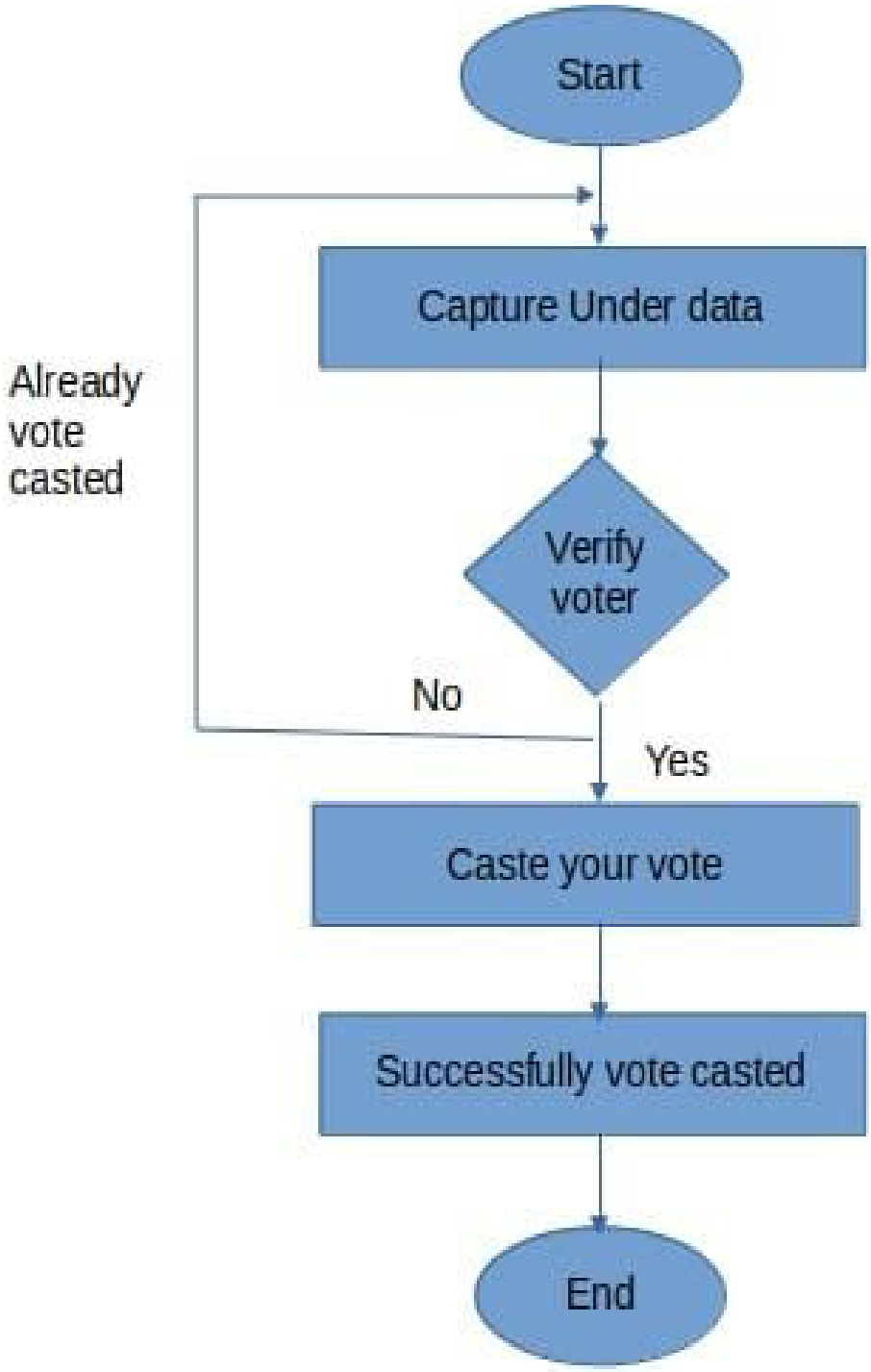

Fig :4.1 flow chart

Flow chat description:

This flowchart illustrates the process of verifying and casting a vote in an election system. Here's the step-by-step explanation:

1. Start :

- The process begins at the starting point.

2. Capture User Data :

- The system collects the necessary information from the user, such as identification details (e.g., voter ID or personal data).

3. Verify Voter:

- The system checks the validity of the user's information against the voter database to confirm if the person is an authorized voter.

- This step ensures that only eligible voters proceed to the next stage.

4. Decision Point (Already Voted):

- If the system detects that the voter has already cast their vote, it loops back, preventing duplicate voting.

- If the voter hasn't voted yet, the process moves forward.

5. Cast Your Vote:

- The voter is allowed to cast their vote once verification is successful.

6. Vote Successfully Cast:

- The system confirms that the vote has been successfully recorded.

7. End:

- The process concludes, ensuring the vote is securely and accurately registered.

This flowchart represents a typical voting system emphasizing voter authentication and prevention of multiple votes.

CHAPTER 5
SOFTWARE IMPLEMENTATION

5.1 MATLAB:

MATLAB (Matrix Laboratory) is a high-level programming language and numerical computing environment widely used in academia and industry. It offers a wide range of tools for data analysis, visualization, and algorithm development. Some of its key features include:

Built-in mathematical functions: MATLAB provides a comprehensive set of mathematical functions, including linear algebra, optimization, and statistics.

Interactive environment: The MATLAB environment allows users to interact with their data and algorithms in an intuitive way, making it easy to test and debug code.

Visualization tools: MATLAB provides powerful tools for data visualization, including 2D and 3D plotting, and animations.

Toolboxes and add-ons: MATLAB offers a variety of toolboxes and add-ons for specific domains such as signal processing, control systems, and image processing.

Integrations: MATLAB can integrate with other programming languages, such as C, C++, and Java, allowing users to combine the best of both worlds. In conclusion, MATLAB is a versatile tool for numerical computing and data analysis that can be applied to a wide range of applications. Whether you are working in academia or industry MATLAB offers a comprehensive and user-friendly environment for developing and testing your algorithms.

Uses of MATLAB:

- Performing numerical linear algebra
- Numerical computation of Matrices • Data analysis and visualization
- Plotting graphs for larger data sets
- Developing algorithms
- Creating interfaces for the user that is the GUI- Graphical User Interface and other applications that is the API – Application Programming Interface.

Commands used in MATLAB:

Here are some common commands used in MATLAB:

1. disp - display a message or the value of a variable in the Command Window
2. clear - clear workspace variables and command history
3. clc - clear the Command Window
4. close all - close all figures
5. plot - plot data in a 2D line plot
6. xlabel and ylabel - add labels to the x-axis and y-axis, respectively
7. title - add a title to the plot
8. grid - display gridlines on the plot
9. legend - add a legend to the plot
10. subplot - divide the figure into multiple subplots
11. mean - compute the mean of a vector or matrix
12. median - compute the median of a vector or matrix
13. std - compute the standard deviation of a vector or matrix
14. var - compute the variance of a vector or matrix
15. sort - sort the elements of a vector or matrix in ascending or descending order
16. size - compute the size of a matrix
17. length - compute the length of a vector
18. sum - compute the sum of the elements of a vector or matrix
19. prod - compute the product of the elements of a vector or matrix
20. find - find the indices of non-zero elements in a vector or matrix.

5.2 ALGORITHMS

Voter Verification and Voting Process

1. Initialization and Face Image Loading:
   - Load the list of registered face images stored in a specified folder.
   - If the folder is not found, prompt the user to select the folder containing the images.

2. Voter Verification:
   - Allow the user to load a test image for verification (e.g., using a webcam or selecting a file).
   - Convert the test image to grayscale and resize it to a fixed size (e.g., 100x100).
   - Use Principal Component Analysis (PCA) to compare the test image with registered face images.
   - If a match is found, verify the voter and proceed to the voting pad.
   - If no match is found, notify the user that the voter was not recognized.

3. Voting Pad:
   - If the voter is verified, show the voting interface.
   - Display a list of candidates along with their images.
   - Allow the user to select a candidate from the displayed options.
   - Enable the user to cast their vote by selecting the candidate and pressing a "Cast Vote" button.
   - If the user has already voted, display a message saying "You have already voted" and disable further voting.

4. Vote Casting and Status Update:
   - After a vote is cast, record the vote status in a file (voted_status.txt).
   - If the user attempts to vote again, the system will check the voted_status.txt file to ensure they cannot vote multiple times.
   - If the user has voted, display a message: "You have already voted.
5. Reset Voting Status:
   - Provide an option for an administrator or user to reset the voting status (i.e., clear the voted_status.txt file).
   - The reset function allows the system to mark the voter as not having voted, allowing them to vote again in a future session.

6. Handling User Interface and Interactions:
   - Create a graphical user interface (GUI) that includes buttons for loading images, refreshing the voting status, displaying the results, and casting votes.
   - The interface handles user clicks, updates status, and shows appropriate messages based on the voting conditions (e.g., "Please select a candidate" if no candidate is selected).

Workflow

1. Load Registered Faces:
   - Check if the folder path for registered faces exists or prompt the user to select it.

-

2. Voter Image Selection:
   - Ask the user to load a test image for verification.

3. Face Recognition:
   - Use PCA to compare the test image with registered faces and determine if there is a match.

4. Voting Interface:
   - If the voter is verified, display the voting interface with candidate images and selection options.
5. Cast Vote:
   - If a candidate is selected, store the vote status and mark the voter as having voted.

6. Voting Status Management :
   - Create a graphical user interface (GUI) that includes buttons for loading images, refreshing the voting status, displaying the results, and casting votes.
   - Optionally, reset the voting status by deleting the status file.

This algorithm encapsulates the full process of voter verification, voting, and status management using face recognition and an interactive voting pad interface.

CHAPTER 6

CODE IMPLEMENTATION

6.1 MATLAB Code:

```
% Main Script for Voter Verification and Voting
% Load registered face images
faceImages = load_registered_faces();
if isempty(faceImages)
    disp('No face images loaded. Exiting...');
    return;
end
% Allow user to select a test image
[testFile, testPath] = uigetfile({'.jpg;.png;*.bmp', 'Image Files'}, 'Select a test image');
if isequal(testFile, 0)
    disp('No test image selected. Exiting...');
    return;
end
% Read the test image
testImage = imread(fullfile(testPath, testFile));
% Recognize the test image
recognizedIndex = recognize_face(testImage, faceImages);
% Check if the voter is recognized
if recognizedIndex > 0
    disp(['Voter verified: Face index ', num2str(recognizedIndex)]);
    figure;
    imshow(faceImages(:,:,recognizedIndex));
    title(['Recognized Face ', num2str(recognizedIndex)]);
    voting_pad();
else
    disp('Voter not recognized. Exiting...');
    return;  % Exit if the voter is not recognized
end
function recognizedIndex = recognize_face(testImage, faceImages)
  % Convert test image to grayscale and resize
    grayTestImage = rgb2gray(testImage);
    resizedTestImage = imresize(grayTestImage, [100, 100]);
    testVector = resizedTestImage(:);  % Flatten to a column vector
    % Flatten registered images
    [height, width, numImages] = size(faceImages);
    faceVectors = reshape(faceImages, height * width, numImages);  % Convert each image to
a column vector
    % Apply PCA on the face image data (centered data)
    [coeff, ~, ~] = pca(double(faceVectors'));  % PCA on the data
    % Project test image onto PCA space
```

```
    testProjection = coeff' * double(testVector);
  % Compute distances
    distances = sum((testProjection - coeff' * faceVectors).^2, 1);
    disp('Distances to registered faces:');
    disp(distances);
  % Set a threshold for recognition
    distanceThreshold = 150;  % Adjust this value based on your testing
   % Find the closest match
    [minDistance, recognizedIndex] = min(distances);
  % Check against the threshold
    if minDistance > distanceThreshold
        recognizedIndex = 0;  % No close match
    else
        disp(['Recognized Index: ', num2str(recognizedIndex), ' with Distance: ',
num2str(minDistance)]);
    end
end
function faceImages = load_registered_faces()
    % Check if the folder path for registered faces exists
    if exist('faces_folder_path.txt', 'file')
        % Read folder path from file
        fid = fopen('faces_folder_path.txt', 'r');
        facesFolder = fgetl(fid);
        fclose(fid);
    else
        % Ask user to select the folder if not found
        facesFolder = uigetdir('', 'Select the folder containing registered faces');
        if facesFolder ~= 0
            % Save the selected folder path for future use
            fid = fopen('faces_folder_path.txt', 'w');
            fprintf(fid, '%s', facesFolder);
            fclose(fid);
        else
            facesFolder = '';  % Return empty if no folder is selected
            disp('No folder selected. Exiting...');
            return;
        end
    end
   % Initialize a 3D array to store face images (height x width x numImages)
    faceImages = []
    % Get a list of all image files in the folder (only .jpg files in this case)
    imageFiles = dir(fullfile(facesFolder, '.jpg'));  % Change to '.png' or '*.bmp' if needed
   if isempty(imageFiles)
        disp('No image files found in the folder.');
        return;
    end
    % Get the size of the first image to initialize faceImages matrix
    firstImage = imread(fullfile(facesFolder, imageFiles(1).name));
    grayFirstImage = rgb2gray(firstImage);
    resizedFirstImage = imresize(grayFirstImage, [100, 100]);
```

```
 % Resize for consistency
   [height, width] = size(resizedFirstImage);
 % Get the dimensions of the resized image
 % Initialize the faceImages matrix to store all face images
   faceImages = zeros(height, width, length(imageFiles));  % 3D matrix to store each face as
a slice
  % Process each registered face image
   for i = 1:length(imageFiles)
      % Read the image
      img = imread(fullfile(facesFolder, imageFiles(i).name));
       % Convert the image to grayscale (necessary for face recognition)
      grayImage = rgb2gray(img)
      % Resize the image to a standard size (e.g., 100x100 pixels)
      grayImage = imresize(grayImage, [100, 100]);  % Resize for consistency
       % Store the preprocessed image in the 3D matrix (each image is a slice)
      faceImages(:,:,i) = grayImage;
   end

   % Display success message
   disp(['Loaded ', num2str(length(imageFiles)), ' registered faces.']);
end
function voter_interface()
   % Check if the faces folder path exists, otherwise ask the user to select it
   facesFolder = getFacesFolder();
   if isempty(facesFolder)
      disp('No faces folder selected. Exiting...');
      return;
   end

   % Create a figure for the voter interface
   hFig = figure('Name', 'Voter Verification', 'Position', [100, 100, 400, 300]);

   % Create a button to load the test image
   uicontrol('Style', 'pushbutton', 'String', 'Load Test Image', ...
           'Position', [100, 220, 200, 30], ...
           'Callback', @loadImageCallback);

   % Create a button to refresh voting status
   uicontrol('Style', 'pushbutton', 'String', 'Refresh Voting Status', ...
           'Position', [100, 180, 200, 30], ...
           'Callback', @resetVotingStatusCallback);

   % Text field to display the result
   resultText = uicontrol('Style', 'text', 'Position', [50, 150, 300, 30], ...
                       'FontSize', 12, 'String', '');

   % Callback function for loading and recognizing the image
   function loadImageCallback(~, ~)
      % Check if the voter has already voted
      if hasVoted()
```

```
            resultText.String = 'You have already voted!';
            return;  % Exit if the voter has already voted
        end

        % Load registered face images from the selected folder
        faceImages = load_registered_faces(facesFolder);

        if isempty(faceImages)
            resultText.String = 'No registered faces found.';
            return;
        end

        % Allow user to select a test image
        [testFile, testPath] = uigetfile({'.jpg;.png;*.bmp', 'Image Files'}, 'Select a test image');
        if isequal(testFile, 0)
            resultText.String = 'No test image selected.';
            return;
        end

        % Read the test image
        testImage = imread(fullfile(testPath, testFile));

        % Recognize the test image
        recognizedIndex = recognize_face(testImage, faceImages);

        % Update the result text based on recognition
        if recognizedIndex > 0
            resultText.String = ['Voter verified: Face index ', num2str(recognizedIndex)];
            voting_pad();  % Proceed to voting pad
            % Mark as voted
            writeStatusToFile('voted_status.txt', true);
        else
            resultText.String = 'Voter not recognized.';
        end
    end

    % Callback function to reset the voting status
    function resetVotingStatusCallback(~, ~)
        resetVotingStatus();  % Reset the voting status when clicked
        resultText.String = 'Voting status has been reset. Voters can vote again.';
    end
end

% Function to get the faces folder path from the file (or ask for input if not present)
function facesFolder = getFacesFolder()
    if exist('faces_folder_path.txt', 'file')
        % Read folder path from file
        fid = fopen('faces_folder_path.txt', 'r');
        facesFolder = fgetl(fid);
        fclose(fid);
```

```
    else
        % Ask user to select the folder if not found
        facesFolder = uigetdir('', 'Select the folder containing registered faces');
        if facesFolder ~= 0
            % Save the selected folder path for future use
            fid = fopen('faces_folder_path.txt', 'w');
            fprintf(fid, '%s', facesFolder);
            fclose(fid);
        else
            facesFolder = '';  % Return empty if no folder is selected
        end
    end
end
% Function to load registered faces from the saved folder path
function faceImages = load_registered_faces(facesFolder)
    % Initialize a 3D array to store face images (height x width x numImages)
    faceImages = [];
        % Get a list of all image files in the folder (only .jpg files in this case)
    imageFiles = dir(fullfile(facesFolder, '.jpg'));  % Change to '.png' or '*.bmp' if needed
     if isempty(imageFiles)
        disp('No image files found in the folder.');
        return;
    end
    % Get the size of the first image to initialize faceImages matrix
    firstImage = imread(fullfile(facesFolder, imageFiles(1).name));
    grayFirstImage = rgb2gray(firstImage);
    resizedFirstImage = imresize(grayFirstImage, [100, 100]);  % Resize for consistency
    [height, width] = size(resizedFirstImage);  % Get the dimensions of the resized image
     % Initialize the faceImages matrix to store all face images
    faceImages = zeros(height, width, length(imageFiles));  % 3D matrix to store each face as
a slice
    % Process each registered face image
    for i = 1:length(imageFiles)
        % Read the image
        img = imread(fullfile(facesFolder, imageFiles(i).name));
         % Convert the image to grayscale (necessary for face recognition)
        grayImage = rgb2gray(img);
        % Resize the image to a standard size (e.g., 100x100 pixels)
        grayImage = imresize(grayImage, [100, 100]);  % Resize for consistency
         % Store the preprocessed image in the 3D matrix (each image is a slice)
        faceImages(:,:,i) = grayImage;
    end
end
% Function to check if the voter has voted
function status = hasVoted()
    statusFile = 'voted_status.txt';
    if exist(statusFile, 'file')
        status = readStatusFromFile(statusFile);
    else
        status = false;  % If file doesn't exist, voter has not voted
```

```
    end
end

% Function to reset the "voted" status
function resetVotingStatus()
    statusFile = 'voted_status.txt';
    if exist(statusFile, 'file')
        delete(statusFile);  % Delete the file to reset status
    end
end

% Function to write the voting status (true/false) to the file
function writeStatusToFile(fileName, status)
    fileID = fopen(fileName, 'w');
    fprintf(fileID, '%d', status);
    fclose(fileID);
end

% Function to read the status from the file
function status = readStatusFromFile(fileName)
    fileID = fopen(fileName, 'r');
    status = fscanf(fileID, '%d');
    fclose(fileID);
end
function voting_pad()
    % Check if the voter has already voted
    if hasVoted()
        disp('You have already voted.');
        return;
    end

    % Create a figure for the voting pad
    hFig = figure('Name', 'Voting Pad', 'Position', [100, 100, 500, 400]);

    % Define candidate names and their image file paths
    candidates = {'Candidate A', 'Candidate B', 'Candidate C'};
    candidateImages = {'C:\Users\Administrator\Desktop\s.jpg',
'C:\Users\Administrator\Desktop\c.jpg', 'C:\Users\Administrator\Desktop\a.jpg'};

    % Create a group for candidate selection
    uicontrol('Style', 'text', 'String', 'Select a Candidate:', ...
            'Position', [150, 330, 200, 20], 'FontSize', 12);

    % Variable to store the selected candidate index
    selectedCandidateIndex = 0;

    % Display candidates with their images and radio buttons
    for i = 1:length(candidates)
        % Display the candidate image
        img = imread(candidateImages{i});
```

```
        subplot(3, 2, (i - 1) * 2 + 1);  % Position for the image
        imshow(img);
        title(candidates{i});

        % Create a radio button for each candidate
        uicontrol('Style', 'radiobutton', 'String', '', ...
                'Position', [10, 240 - (70 * (i - 1)), 30, 30], ...
                'FontSize', 12, 'Callback', @(src, event) selectCandidate(i));
    end

    % Create a button to cast the vote
    uicontrol('Style', 'pushbutton', 'String', 'Cast Vote', ...
            'Position', [200, 20, 100, 30], ...
            'Callback', @castVoteCallback);

    % Result display text
    resultText = uicontrol('Style', 'text', 'Position', [50, 5, 400, 30], ...
                        'FontSize', 12, 'String', '');

    % Callback function to handle candidate selection
    function selectCandidate(index)
        selectedCandidateIndex = index;  % Store the selected candidate index
    end

    % Callback function to cast the vote
    function castVoteCallback(~, ~)
        if selectedCandidateIndex == 0
            resultText.String = 'Please select a candidate!';
        else
            votedCandidate = candidates{selectedCandidateIndex};
            resultText.String = ['You voted for: ', votedCandidate];
            % Store the vote status (voted) in a file
            markAsVoted();
        end
    end
end

% Function to check if the voter has already voted
function status = hasVoted()
    % Check if a "voted" status file exists
    statusFile = 'voted_status.txt';
    if exist(statusFile, 'file')
        % If the file exists, read the status
        status = readStatusFromFile(statusFile);
    else
        status = false;  % If file doesn't exist, voter has not voted
    end
end

% Function to mark the voter as having voted
```

```
function markAsVoted()
    % Write the "voted" status to a file
    statusFile = 'voted_status.txt';
    writeStatusToFile(statusFile, true);  % Set status to true (voted)
end

% Function to reset the voting status (for refresh button)
function resetVotingStatus()
    % Reset the "voted" status to false (or delete status file)
    statusFile = 'voted_status.txt';
    writeStatusToFile(statusFile, false);  % Set status to false (not voted)
end

% Function to write the status (true/false) to a file
function writeStatusToFile(fileName, status)
    fileID = fopen(fileName, 'w');
    fprintf(fileID, '%d', status);
    fclose(fileID);
end

% Function to read the status from the file
function status = readStatusFromFile(fileName)
    fileID = fopen(fileName, 'r');
    status = fscanf(fileID, '%d');
    fclose(fileID);
end
function status = hasVoted()
    % Check if a "voted" status file exists
    statusFile = 'voted_status.txt';
    if exist(statusFile, 'file')
        % If the file exists, read the status
        status = readStatusFromFile(statusFile);
    else
        status = false;  % If file doesn't exist, voter has not voted
    end
end

function resetVotingStatus()
    % Reset the "voted" status to false (or delete status file)
    statusFile = 'voted_status.txt';
    writeStatusToFile(statusFile, false)  ; % Set status to false (not voted)
end

function writeStatusToFile(fileName, status)
    % Write the status (true/false) to a file
    fileID = fopen(fileName, 'w');
    fprintf(fileID, '%d', status);
    fclose(fileID);
end
```

```
function status = readStatusFromFile(fileName)
    % Read the status from the file
    fileID = fopen(fileName, 'r');
    status = fscanf(fileID, '%d');
    fclose(fileID);
end
function refresh_interface()
    % Create a separate figure for the refresh interface
    hFig = figure('Name', 'Refresh Voting Status', 'Position', [100, 100, 400, 200]);

    % Create a button to refresh the voting status
    uicontrol('Style', 'pushbutton', 'String', 'Reset Voting Status', ...
            'Position', [100, 80, 200, 40], 'Callback', @resetVotingStatusCallback);

    % Display status text
    statusText = uicontrol('Style', 'text', 'Position', [50, 40, 300, 30], ...
                        'FontSize', 12, 'String', 'Click the button to reset voting status.');

    % Callback function to reset the voting status
    function resetVotingStatusCallback(~, ~)
        resetVotingStatus();  % Reset the "voted" status
        statusText.String = 'Voting status has been reset. Voter can vote again.';
    end
end

% Function to reset the "voted" status
function resetVotingStatus()
    % Reset the "voted" status by writing 'false' to the status file
    statusFile = 'voted_status.txt';
    writeStatusToFile(statusFile, false);  % Set status to false (not voted)
end

function writeStatusToFile(fileName, status)
    % Write the status (true/false) to a file
    fileID = fopen(fileName, 'w');
    fprintf(fileID, '%d', status);
    fclose(fileID);
end
```

CHAPTER 7

RESULTS

7.1 Results

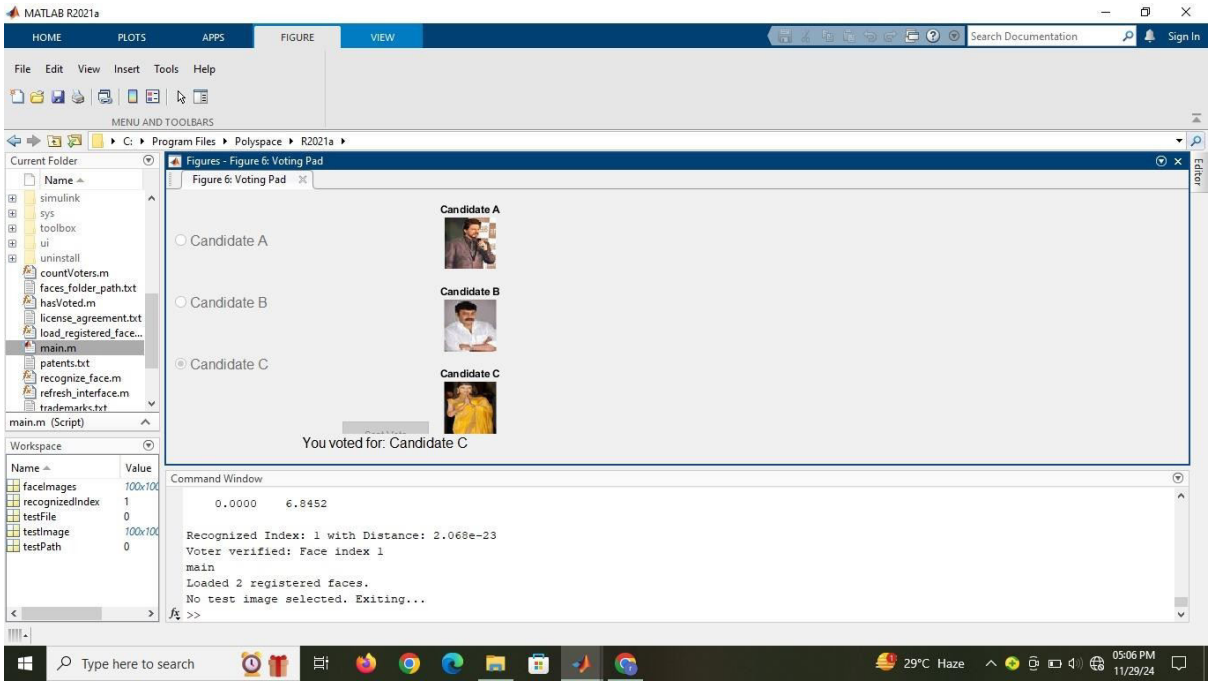

Fig 7.1

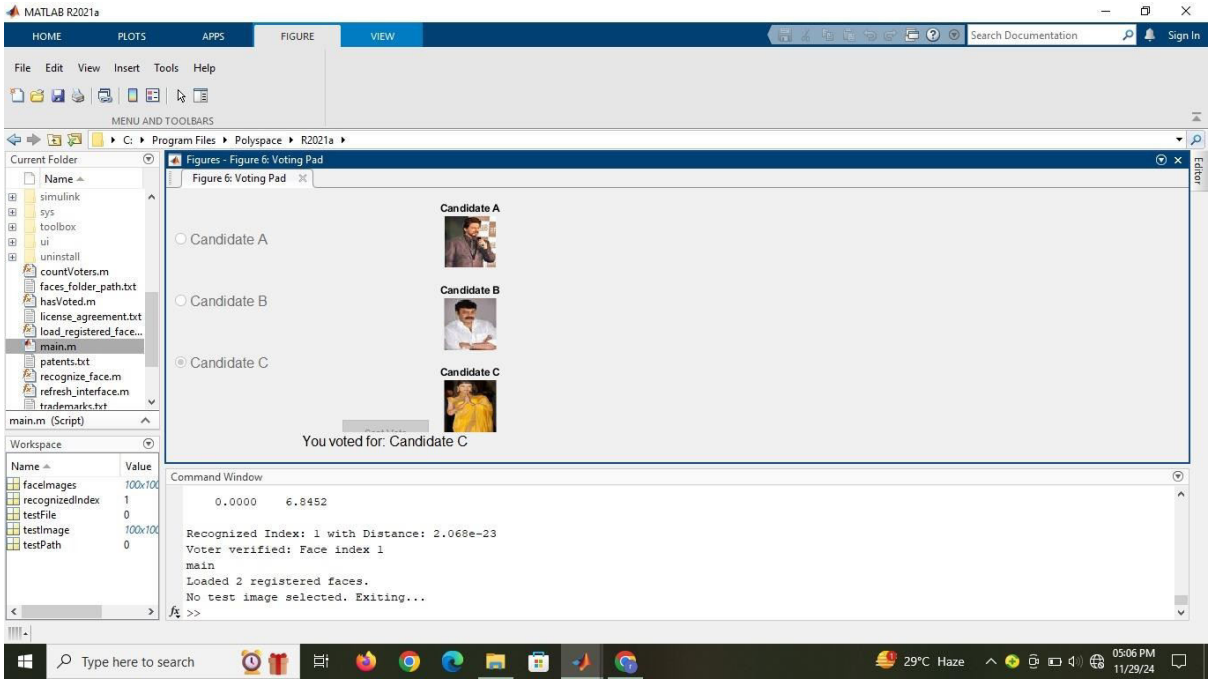

Fig 7.2

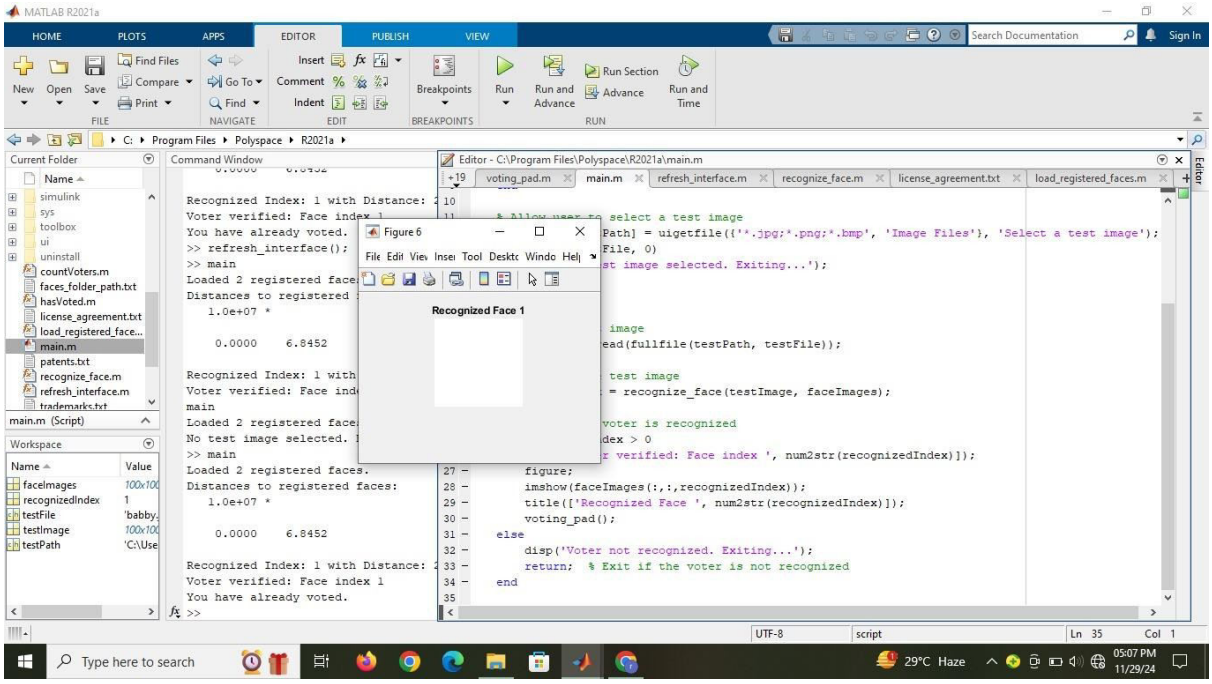

Fig 7.3

CHAPTER 8

CONCLUSION AND FUTURE

SCOPE

8.1 CONCLUSION:

This facial recognition-based voting system is a secure and efficient method for verifying voter identity and ensuring the integrity of the voting process. The key features of the system are as follows:

1. Facial Recognition for Voter Authentication: The system uses facial recognition to verify the identity of the voter before allowing them to cast their vote. This ensures that only registered voters can vote, reducing the risk of identity fraud.

2. Prevention of Multiple Voting: By tracking the voting status of each user (using a text file to record whether the voter has voted), the system prevents voters from casting multiple votes, thus maintaining the fairness of the election process.

3. User-Friendly Interface: The system provides an intuitive graphical user interface (GUI) for the voter to interact with. It allows for easy selection of candidates and ensures a smooth voting experience.

4. Modular Design: The code is modular, with distinct functions for face recognition, managing voter status, and interacting with the GUI. This makes it easier to modify or extend the system for additional features, such as adding more candidates or integrating with a database for larger-scale applications.

5. Scalability and Security: While the current implementation is suitable for small-scale usage with a limited number of registered voters, the system can be extended to accommodate larger voter databases. Enhanced security measures, like encryption of voter data, could further strengthen the system against potential security breaches.

In summary, this system demonstrates the potential of integrating biometric authentication with digital voting processes, offering a more secure and reliable method of voting. It can be further developed to support larger-scale elections, improve real-time face recognition accuracy, and integrate with modern databases for voter registration and results management.

8.2 FUTURE SCOPE:

The facial recognition-based voting system can be further enhanced and scaled up for broader adoption. Here are some potential areas for future development:

1. Scalability and Database Integration:

Larger Voter Databases: Integrating the system with a centralized database for storing voter information (e.g., names, photographs, and voting records) would allow the system to handle a larger number of voters. This would be essential for implementing the system in national or state-level elections.

Distributed System: The system could be expanded to work in a distributed architecture, allowing for real-time voting data synchronization across multiple locations.

2.Election Integrity and Monitoring:

Real-Time Monitoring: Introducing features for real-time election monitoring, such as tracking the total number of votes cast and ensuring no multiple votes are cast by the same person, could increase the transparency of the election process.

Audit Trails: Maintaining a secure and immutable audit trail of each vote cast will increase transparency and allow for post-election audits.