Media Elektrik Journal

e-ISSN: 1907-1728 | p-ISSN: 2721-9100
Vol. 21, No. 2, 2024

Open Access

IMPLEMENTATION OF AN IDENTIFICATION SYSTEM WITH FACIAL IMAGE PROCESSING (EIGENFACE) USING MATLAB APPLICATION

Nur Dua Fathansyah [1]*, Reni Rahmadewi [2], Damar Adzani Susanto [3], Wisnu Kuncoro Jati [4]

[1]Electrical Engineering, Singaperbangsa State University Karawang, Indonesia
[2]Electrical Engineering, Singaperbangsa State University Karawang, Indonesia
[3]Electrical Engineering, Singaperbangsa State University Karawang, Indonesia
[4]Electrical Engineering, Singaperbangsa State University Karawang, Indonesia
* Corresponding author: 2010631160082@student.unsika.ac.id

ARTICLE INFO

Received: 27 Apr 2023
Accepted: 9 Jun 2023

ABSTRACT

Facial recognition has emerged as a prominent personal identification system, especially in the industrial sector where traditional attendance card machines have been prevalent. However, manual systems pose several drawbacks, including susceptibility to fraud, lack of flexibility, and resource wastage in card production. To address these issues, this research proposes a shift from card-based attendance systems to a facial recognition-based system using MATLAB application. The study aims to design an identification system based on the eigenface method to process facial images. It begins with a thorough literature review to gather relevant references. Subsequently, a specialized MATLAB application was developed for face identification, and its accuracy was tested. The research utilizes trained data consisting of 45 photos and tested data consisting of 15 photos to evaluate the system's accuracy. The test results reveal a 100% accuracy rate in system identification. Notably, the accuracy varies when identifying faces with different background images, indicating the robustness of the MATLAB application. Overall, the findings suggest that MATLAB can effectively implement an image processing attendance system using the eigenface method.

Keywords: Eigenface method, MATLAB application, Attendance system, Image processing

1. Introduction

Presence is a very important thing for an agency. Presence is used to monitor employee performance as proof of attendance. Attendance activities are currently carried out by each employee by making direct contact with the attendance device, so it takes a long time. The presence system is related to the security system in terms of monitoring people going in and out of a room. You need to be careful about strangers entering the room. A lack of security levels can increase the problem of theft, making it difficult for the police to carry out facial recognition investigations [1].

Facial recognition is a personal identification system that uses a person's facial characteristics. Facial recognition itself is a branch of biometrics, which is a scientific field that uses the physical characteristics of a person to determine or reveal their identity. With a recognition system like this, the system is used to recognize a person's identity automatically using computer technology and aims to increase system security so that the self-recognition system's ability to recognize targets correctly. Facial recognition has been developed for many security applications, attendance applications and population data collection applications. Using a face as an identifier has many benefits, especially its practicality because it requires an image for identification [2]. Attendance implementations such as machines with timecards are sometimes still used in the case of employee absences. Apart from that, fingerprint absence is also sometimes still used. Both have problems in terms of scanning, security, and frequent data error rates.

The solution to this problem is to carry out this research which aims to produce a better and more reliable attendance system. The attendance system for facial recognition created is a facial identification system that uses Matlab. Apart from creating a facial recognition system, this research also aims to test how accurate the facial recognition method (eigenface) is with the Matlab application so that later problems with identifying facial conditions we can find out how far this application works.

2. Materials and Methods

a. Research methods.

The research began with a literature study about facial recognition-based attendance systems. Then, the author developed a facial detection and recognition application based on references to test the accuracy of the attendance system in individual identification. After application testing is complete, the facial recognition-based system concludes the results. The research sequence can be seen in Figure 1.

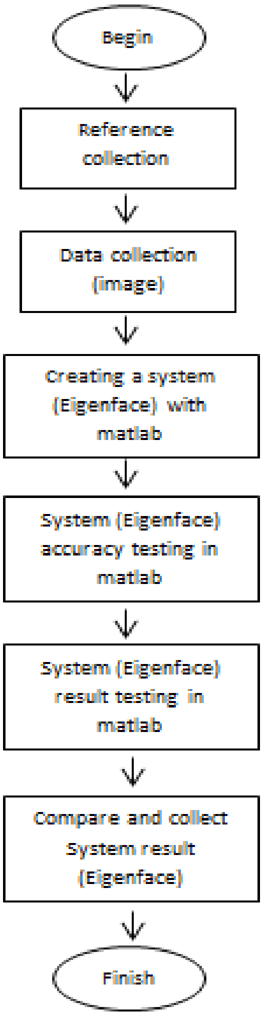

Begin
Reference collection
Data collection (image)
Creating a system (Eigenface) with matlab
System (Eigenface) accuracy testing in matlab
System (Eigenface) result testing in matlab
Compare and collect System result (Eigenface)
Finish

Figure 1. Research Flowchart

b. Research materials.

The implementation of the facial detection and recognition system was created to test the accuracy identified by the Matlab application, in general there are three main parts, namely the input, process and output. System input uses a laptop computer or PC, system processes use the Matlab application, system output uses the Matlab application, but the results are in the form of images. The system block diagram can be seen in Figure 2.

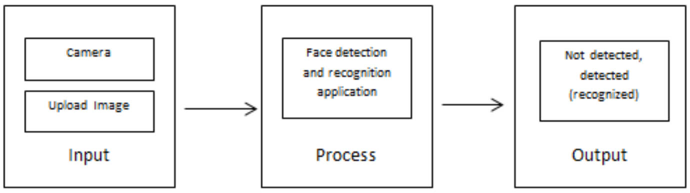

Camera
Upload Image
Input
Face detection and recognition application
Process
Not detected, detected (recognized)
Output

Figure 2. System Block Diagram

c. System

It can be seen in the block diagram that the input system comes from the camera. The image from the camera is then uploaded, then processed by the application and the output results in the form of a statement from the system such as not detected or detected. As in the face detection system, the absence page contains a face image capture frame which will automatically take an image, if a face is detected in the frame, then the image will be taken along with the year, month, date, hour-minute-second at that time. then the data is processed by the system to be saved to the data base.

The training feature becomes a frame to retrieve the identity of the facial image, the identity of the object and the facial image are stored in the database. Without processing the training data, the object cannot be detected. The system (application) output is an object condition in the form of a face not found, a face not registered, and a face not detected because it failed. It is hoped that when looking at the facial image data that has been trained, the system will display facial identification based on the data being tested.

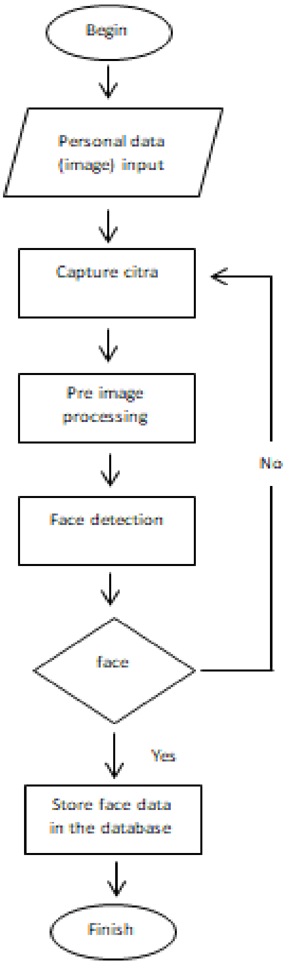


Figure 3. System training data flowchart

Application testing is carried out in the form of feature detection and facial recognition testing. This testing is carried out using the Matlab application through several image processing processes to be able to recognize images in the training data in accordance with the design objectives. The image data requirements for the system trained by the Matlab application are in the points below.

a. Face images with different backgrounds.
b. Draw a face with different pixels.
c. Registered facial images.
d. Identical facial images
e. Facial image identified.

Test data is one of the system output provisions, along with the requirements for system testing by the Matlab application in the points below.

a. The face faces the camera at a distance of ± 45 cm, with a maximum tilt of 5 degrees left and right.
b. Lighting and quality during storage and facial recognition are sufficient, not too little, or too much.
c. The results recorded are test results (detected or not) and detection accuracy.
d. The training images used were 45 images for each person. The use of 45 training images is based on the ability of training face recognition and uses at least 1 test image [1].
e. The test images were 15 images with raw information from different image pixels, then represented using the principal component analysis (PCA) method [3].

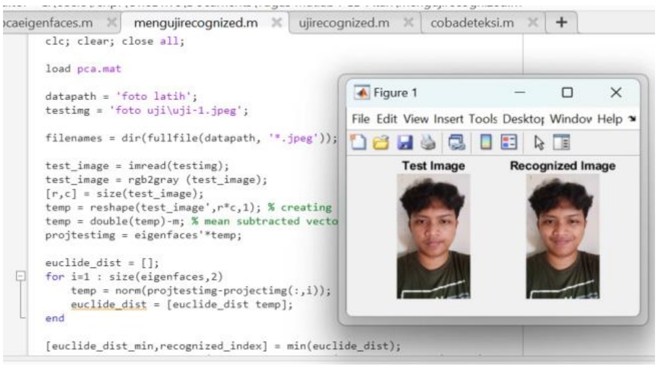
3. Results

a. Figures and Tables

Table 1 is a description of the results that prove how far the Matlab application works in identifying faces using the individual recognition method (PCA Method). The table states that green (successful test results) and red (unsuccessful test results). Even though (figure.1) to (figure.15), the results prove a 100% accuracy rate on training data and test data.

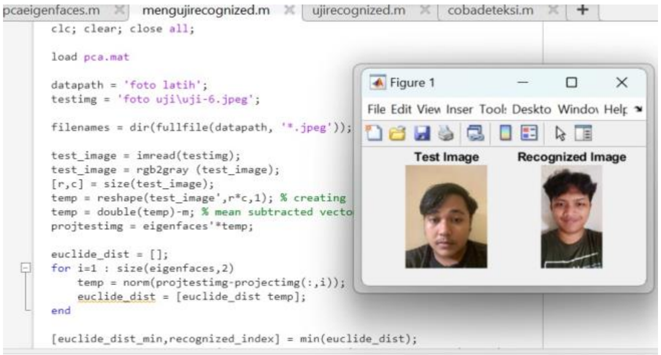
Table 1. Number of Test Results

Test Image	recognize	Does not recognize
1st	🟩	
2nd		🟧
3rd		🟧
4th	🟩	
5th		🟧
6th		🟧
7th	🟩	
8th		🟧
9th		🟧
10th	🟩	
11th		🟧
12th		🟧
13th	🟩	
14th		🟧
15th		🟧



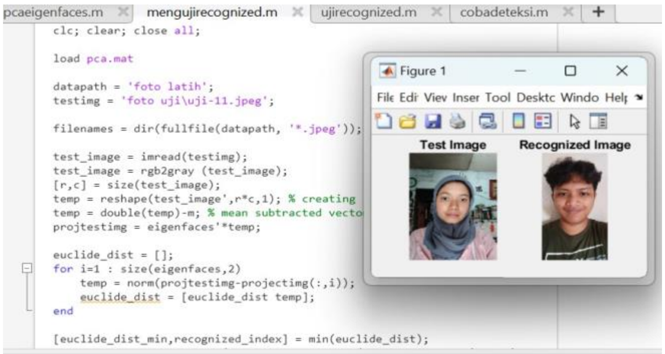
```
clc; clear; close all;

load pca.mat

datapath = 'foto latih';
testimg = 'foto uji\uji-1.jpeg';

filenames = dir(fullfile(datapath, '*.jpeg'));

test_image = imread(testimg);
test_image = rgb2gray (test_image);
[r,c] = size(test_image);
temp = reshape(test_image',r*c,1); % creating
temp = double(temp)-m; % mean subtracted vecto
projtestimg = eigenfaces'*temp;

euclide_dist = [];
for i=1 : size(eigenfaces,2)
    temp = norm(projtestimg-projectimg(:,i));
    euclide_dist = [euclide_dist temp];
end

[euclide_dist_min,recognized_index] = min(euclide_dist);
```

Figure 1. 1st test



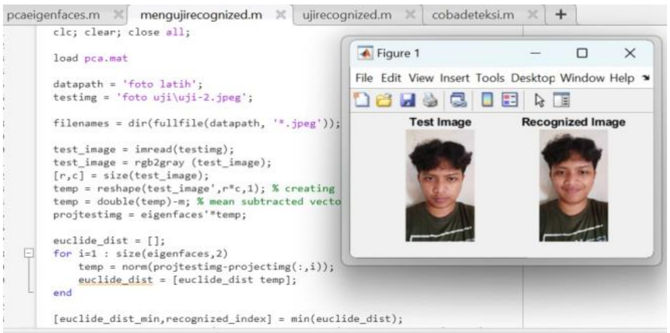
```
clc; clear; close all;

load pca.mat

datapath = 'foto latih';
testimg = 'foto uji\uji-6.jpeg';

filenames = dir(fullfile(datapath, '*.jpeg'));

test_image = imread(testimg);
test_image = rgb2gray (test_image);
[r,c] = size(test_image);
temp = reshape(test_image',r*c,1); % creating
temp = double(temp)-m; % mean subtracted vecto
projtestimg = eigenfaces'*temp;

euclide_dist = [];
for i=1 : size(eigenfaces,2)
    temp = norm(projtestimg-projectimg(:,i));
    euclide_dist = [euclide_dist temp];
end

[euclide_dist_min,recognized_index] = min(euclide_dist);
```

Figure 2. 2nd test



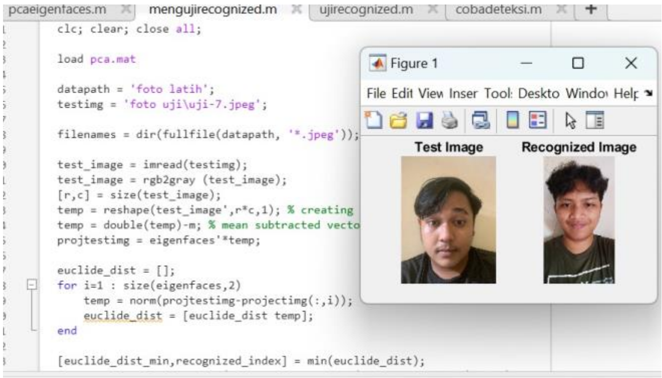
```
clc; clear; close all;

load pca.mat

datapath = 'foto latih';
testimg = 'foto uji\uji-11.jpeg';

filenames = dir(fullfile(datapath, '*.jpeg'));

test_image = imread(testimg);
test_image = rgb2gray (test_image);
[r,c] = size(test_image);
temp = reshape(test_image',r*c,1); % creating
temp = double(temp)-m; % mean subtracted vecto
projtestimg = eigenfaces'*temp;

euclide_dist = [];
for i=1 : size(eigenfaces,2)
    temp = norm(projtestimg-projectimg(:,i));
    euclide_dist = [euclide_dist temp];
end

[euclide_dist_min,recognized_index] = min(euclide_dist);
```

Figure 3. 3rd test



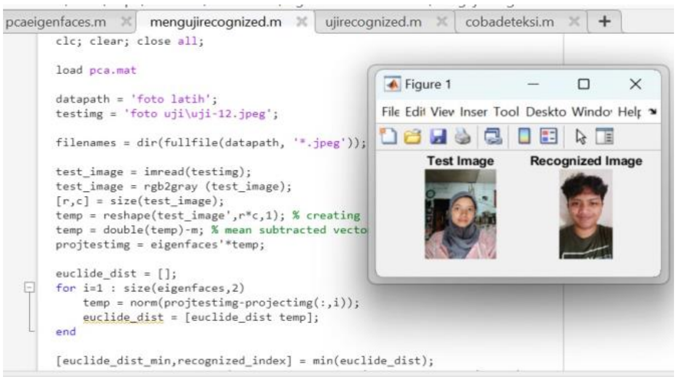
```
clc; clear; close all;

load pca.mat

datapath = 'foto latih';
testimg = 'foto uji\uji-2.jpeg';

filenames = dir(fullfile(datapath, '*.jpeg'));

test_image = imread(testimg);
test_image = rgb2gray (test_image);
[r,c] = size(test_image);
temp = reshape(test_image',r*c,1); % creating
temp = double(temp)-m; % mean subtracted vecto
projtestimg = eigenfaces'*temp;

euclide_dist = [];
for i=1 : size(eigenfaces,2)
    temp = norm(projtestimg-projectimg(:,i));
    euclide_dist = [euclide_dist temp];
end

[euclide_dist_min,recognized_index] = min(euclide_dist);
```

Figure 4. 4th test



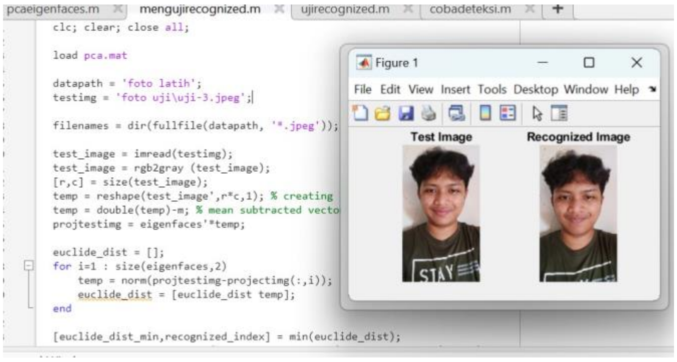
```
clc; clear; close all;

load pca.mat

datapath = 'foto latih';
testimg = 'foto uji\uji-7.jpeg';

filenames = dir(fullfile(datapath, '*.jpeg'));

test_image = imread(testimg);
test_image = rgb2gray (test_image);
[r,c] = size(test_image);
temp = reshape(test_image',r*c,1); % creating
temp = double(temp)-m; % mean subtracted vecto
projtestimg = eigenfaces'*temp;

euclide_dist = [];
for i=1 : size(eigenfaces,2)
    temp = norm(projtestimg-projectimg(:,i));
    euclide_dist = [euclide_dist temp];
end

[euclide_dist_min,recognized_index] = min(euclide_dist);
```

Figure 5. 5th test



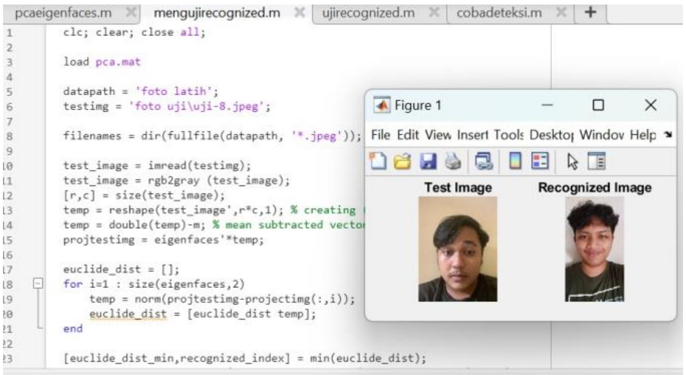
```
clc; clear; close all;

load pca.mat

datapath = 'foto latih';
testimg = 'foto uji\uji-12.jpeg';

filenames = dir(fullfile(datapath, '*.jpeg'));

test_image = imread(testimg);
test_image = rgb2gray (test_image);
[r,c] = size(test_image);
temp = reshape(test_image',r*c,1); % creating
temp = double(temp)-m; % mean subtracted vecto
projtestimg = eigenfaces'*temp;

euclide_dist = [];
for i=1 : size(eigenfaces,2)
    temp = norm(projtestimg-projectimg(:,i));
    euclide_dist = [euclide_dist temp];
end

[euclide_dist_min,recognized_index] = min(euclide_dist);
```

Figure 6. 6th test



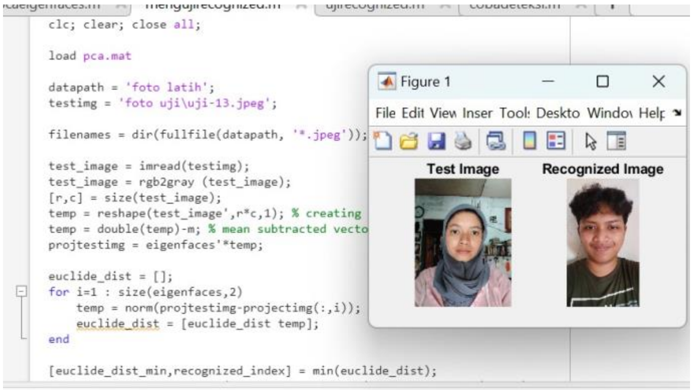
```
clc; clear; close all;

load pca.mat

datapath = 'foto latih';
testimg = 'foto uji\uji-3.jpeg';

filenames = dir(fullfile(datapath, '*.jpeg'));

test_image = imread(testimg);
test_image = rgb2gray (test_image);
[r,c] = size(test_image);
temp = reshape(test_image',r*c,1); % creating
temp = double(temp)-m; % mean subtracted vecto
projtestimg = eigenfaces'*temp;

euclide_dist = [];
for i=1 : size(eigenfaces,2)
    temp = norm(projtestimg-projectimg(:,i));
    euclide_dist = [euclide_dist temp];
end

[euclide_dist_min,recognized_index] = min(euclide_dist);
```

Figure 7. 7th test



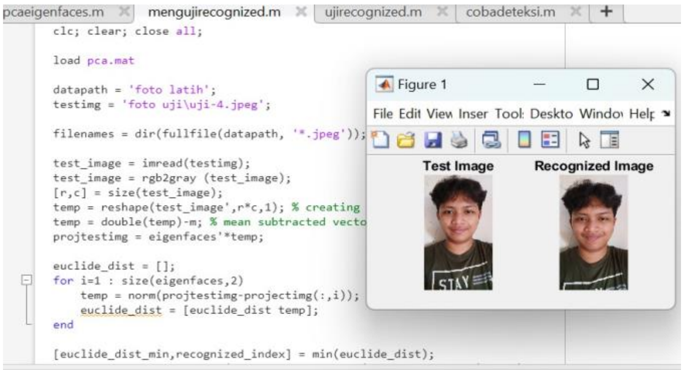
```
clc; clear; close all;

load pca.mat

datapath = 'foto latih';
testimg = 'foto uji\uji-8.jpeg';

filenames = dir(fullfile(datapath, '*.jpeg'));

test_image = imread(testimg);
test_image = rgb2gray (test_image);
[r,c] = size(test_image);
temp = reshape(test_image',r*c,1); % creating
temp = double(temp)-m; % mean subtracted vecto
projtestimg = eigenfaces'*temp;

euclide_dist = [];
for i=1 : size(eigenfaces,2)
    temp = norm(projtestimg-projectimg(:,i));
    euclide_dist = [euclide_dist temp];
end

[euclide_dist_min,recognized_index] = min(euclide_dist);
```

Figure 8. 8th test



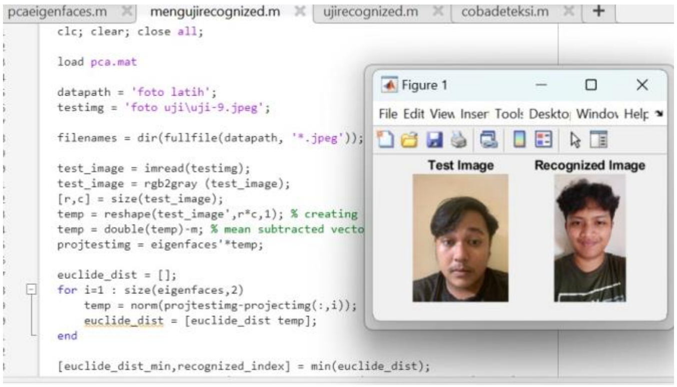
```
clc; clear; close all;

load pca.mat

datapath = 'foto latih';
testimg = 'foto uji\uji-13.jpeg';

filenames = dir(fullfile(datapath, '*.jpeg'));

test_image = imread(testimg);
test_image = rgb2gray (test_image);
[r,c] = size(test_image);
temp = reshape(test_image',r*c,1); % creating
temp = double(temp)-m; % mean subtracted vecto
projtestimg = eigenfaces'*temp;

euclide_dist = [];
for i=1 : size(eigenfaces,2)
    temp = norm(projtestimg-projectimg(:,i));
    euclide_dist = [euclide_dist temp];
end

[euclide_dist_min,recognized_index] = min(euclide_dist);
```

Figure 9. 9th test



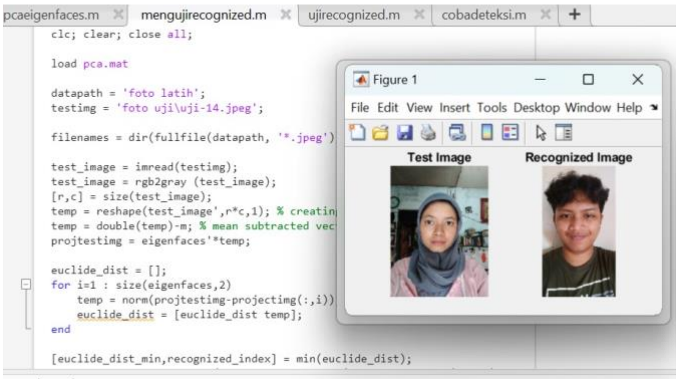
```
clc; clear; close all;

load pca.mat

datapath = 'foto latih';
testimg = 'foto uji\uji-4.jpeg';

filenames = dir(fullfile(datapath, '*.jpeg'));

test_image = imread(testimg);
test_image = rgb2gray (test_image);
[r,c] = size(test_image);
temp = reshape(test_image',r*c,1); % creating
temp = double(temp)-m; % mean subtracted vecto
projtestimg = eigenfaces'*temp;

euclide_dist = [];
for i=1 : size(eigenfaces,2)
    temp = norm(projtestimg-projectimg(:,i));
    euclide_dist = [euclide_dist temp];
end

[euclide_dist_min,recognized_index] = min(euclide_dist);
```

Figure 10. 10th test



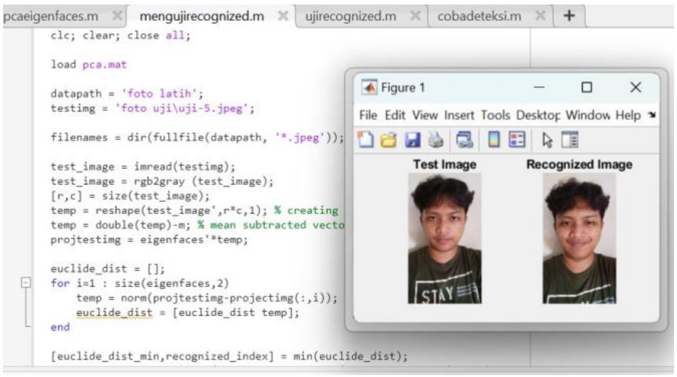
```
clc; clear; close all;

load pca.mat

datapath = 'foto latih';
testimg = 'foto uji\uji-9.jpeg';

filenames = dir(fullfile(datapath, '*.jpeg'));

test_image = imread(testimg);
test_image = rgb2gray (test_image);
[r,c] = size(test_image);
temp = reshape(test_image',r*c,1); % creating
temp = double(temp)-m; % mean subtracted vecto
projtestimg = eigenfaces'*temp;

euclide_dist = [];
for i=1 : size(eigenfaces,2)
    temp = norm(projtestimg-projectimg(:,i));
    euclide_dist = [euclide_dist temp];
end

[euclide_dist_min,recognized_index] = min(euclide_dist);
```

Figure 11. 11th test



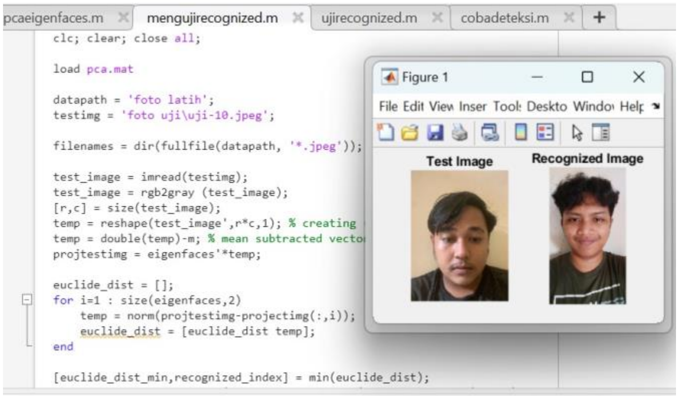
```
clc; clear; close all;

load pca.mat

datapath = 'foto latih';
testimg = 'foto uji\uji-14.jpeg';

filenames = dir(fullfile(datapath, '*.jpeg')

test_image = imread(testimg);
test_image = rgb2gray (test_image);
[r,c] = size(test_image);
temp = reshape(test_image',r*c,1); % creatin
temp = double(temp)-m; % mean subtracted vec
projtestimg = eigenfaces'*temp;

euclide_dist = [];
for i=1 : size(eigenfaces,2)
    temp = norm(projtestimg-projectimg(:,i))
    euclide_dist = [euclide_dist temp];
end

[euclide_dist_min,recognized_index] = min(euclide_dist);
```

Figure 12. 12th test



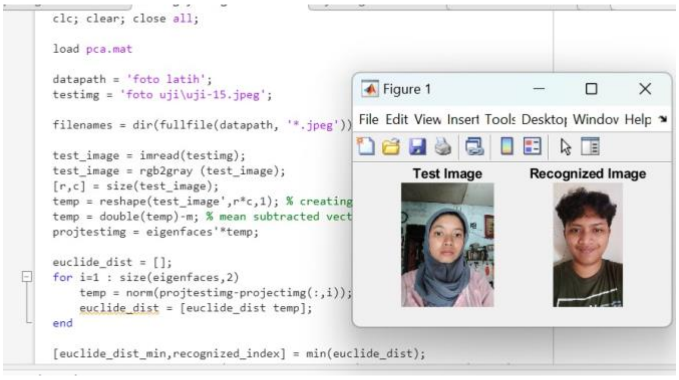
```
clc; clear; close all;

load pca.mat

datapath = 'foto latih';
testimg = 'foto uji\uji-5.jpeg';

filenames = dir(fullfile(datapath, '*.jpeg'));

test_image = imread(testimg);
test_image = rgb2gray (test_image);
[r,c] = size(test_image);
temp = reshape(test_image',r*c,1); % creating
temp = double(temp)-m; % mean subtracted vecto
projtestimg = eigenfaces'*temp;

euclide_dist = [];
for i=1 : size(eigenfaces,2)
    temp = norm(projtestimg-projectimg(:,i));
    euclide_dist = [euclide_dist temp];
end

[euclide_dist_min,recognized_index] = min(euclide_dist);
```

Figure 13. 13th test



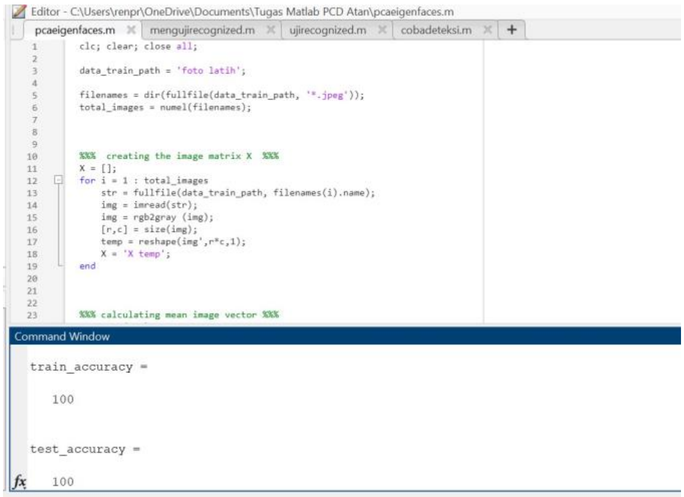
```
clc; clear; close all;

load pca.mat

datapath = 'foto latih';
testimg = 'foto uji\uji-10.jpeg';

filenames = dir(fullfile(datapath, '*.jpeg'));

test_image = imread(testimg);
test_image = rgb2gray (test_image);
[r,c] = size(test_image);
temp = reshape(test_image',r*c,1); % creating
temp = double(temp)-m; % mean subtracted vecto
projtestimg = eigenfaces'*temp;

euclide_dist = [];
for i=1 : size(eigenfaces,2)
    temp = norm(projtestimg-projectimg(:,i));
    euclide_dist = [euclide_dist temp];
end

[euclide_dist_min,recognized_index] = min(euclide_dist);
```

Figure 14. 14th test



```
clc; clear; close all;

load pca.mat

datapath = 'foto latih';
testimg = 'foto uji\uji-15.jpeg';

filenames = dir(fullfile(datapath, '*.jpeg'))

test_image = imread(testimg);
test_image = rgb2gray (test_image);
[r,c] = size(test_image);
temp = reshape(test_image',r*c,1); % creating
temp = double(temp)-m; % mean subtracted vect
projtestimg = eigenfaces'*temp;

euclide_dist = [];
for i=1 : size(eigenfaces,2)
    temp = norm(projtestimg-projectimg(:,i));
    euclide_dist = [euclide_dist temp];
end

[euclide_dist_min,recognized_index] = min(euclide_dist);
```

Figure 15. 15th test


Editor - C:\Users\renpr\OneDrive\Documents\Tugas Matlab PCD Atan\pcaeigenfaces.m
pcaeigenfaces.m
mengujirecognized.m
ujirecognized.m
cobadeteksi.m
Command Window

```
clc; clear; close all;

data_train_path = 'foto latih';

filenames = dir(fullfile(data_train_path, '*.jpeg'));
total_images = numel(filenames);



%%%  creating the image matrix X  %%%
X = [];
for i = 1 : total_images
    str = fullfile(data_train_path, filenames(i).name);
    img = imread(str);
    img = rgb2gray (img);
    [r,c] = size(img);
    temp = reshape(img',r*c,1);
    X = 'X temp';
end



%%% calculating mean image vector %%%
```

```
train_accuracy =

   100


test_accuracy =

   100
```

Figure 16. accuracy of training data and test data

b. Mathematical Formulation

Equation (1) is a color image consisting of three color layers such as an RGB (Red Green Blue) image. RGB images can be converted into gray scale images using the formula [3] [4].

Gray scale $= k.R + k.G + k.B$ (1)

Equation (2) is*training image*becomes 1 single matrix. For example, if the image has size H × W pixels and the number is N, then the resulting flatvector has dimensions of a matrix with the form N × 1 or a linear matrix [3].

Flatfector mean $= 1\ x\ [H\ x\ W]$ (2)

Equation (3) (4) (5) (6) (7) (8) is the PCA method process. This method is used to reduce data dimensions by creating new variables which are linear combinations of the original variables. The main goal of PCA is to reduce data complexity by retaining as much information as possible. This process involves transforming the data into a new coordinate system where each new variable is uncorrelated with each other and includes the maximum variation of the original data. Next, the Eigenface calculation process is carried out with certain steps [3] [5] [6].

1. Prepare data by creating a set S consisting of all training images, (Γ1, Γ2, …, ΓM)

   $S = \{\Gamma 1, \Gamma 2, \ldots, \Gamma M\}$ (3)

2. Take the middle value or mean (Ψ)

   $\Psi = \frac{1}{M}\sum_{n=1}^{M} \Gamma n$ (4)

3. Find the difference (Φ) between the training image value (Γi) and the middle value (Ψ)

   $\Phi_i = \Phi_n - \Psi$ (5)

4. Calculating the value of the covariance matrix (C)

   $$\Psi = \frac{1}{M}\frac{1}{M}\sum_{n=1}^{M} \Phi_n\, \Phi_T = AA^T$$

   $L = A^T A, L = \Phi_M^T \Phi_n$ (6)

5. Calculating the eigenvalue (λ) and eigenvector (v) of the covariance matrix (C)

   $C\ x\ v_i = \lambda_i \times v_i$ (7)

6. After the eigenvector (v) is obtained, then the eigenface (μ)

   $$\mu_i = \sum_{K=1}^{M} v_{lk} \,.\, \Phi_k$$

   $l = 1, \ldots, \mathrm{M}$ (8)

4. Discussion

This research is based on the development of technology that makes difficult mathematical calculations easier. In the contents of this document, it is explained that the Matlab application is used in calculating the eigenfaces method. In several developments, this research follows previous research on eigenfaces with a widely known discussion of the use of vision cameras [4]. However, this research uses a different implementation in image processing.

Based on images of facial conditions during absence or presence, this research talks a lot about image processing of faces detected by vision cameras. The data used in this research works to test the accuracy of the eigenfaces method. This research was carried out using a dataset consisting of two types, namely training data and test data. The training data consists of 45 photos which are used to train the model or algorithm used in the research, while the test data consists of 15 photos which are used to test the extent to which the model or algorithm can recognize patterns or characteristics that have been learned by the training data.

During the research process, various stages were carried out including facial image recognition based on the condition of the photo when it was absent from pixel, background and quality tests which were different for everyone. The results of this system show that the accuracy of the training data and test data reaches 100%. This means that the model or algorithm used can perfectly recognize patterns or characteristics from both types of data. However, interesting results emerged when testing 3 different people, each with 15 photos. Even though the accuracy rate reaches 100%, this test data is considered failed. Success in recognizing patterns or characteristics of individuals is not followed by success in classifying them uniquely (different background and facial conditions) such as absence or presence.

An interesting aspect in this research is the relationship between pixel size, backround, and different image qualities in the introduction of test data to training data. There is a possibility that different pixel sizes can cause inaccuracies in recognizing individuals because the condition of everyone's absence or presence is different in all these aspects. So, the accuracy level of the Matlab identification system reaches 100%, there are still complexities and challenges in dealing with individual variations in similar recognition characteristics. especially in terms of pixels, background, and image quality.

5. Conclusions

In this research, the application of the Matlab application to process facial images using the eigenfaces method resulted in a value of 100% accuracy in the recorded images. There are 2 data available, namely training data and test data, both of which of course have different amounts. In the training data there are 45 images and in the test data there are 15 images.

Eigenfaces were carried out in the Matlab application, with the aim of this research to show an easy way to calculate (eigenfaces) facial images that occur on devices that have recognized camera vision technology. This research succeeded in achieving accuracy in pattern recognition from training data and test data. However, testing of different individuals failed to uniquely classify them.

This is caused by the difficulty of distinguishing individuals with similar characteristics, especially related to image pixel size, image background, and image quality. Therefore, this research highlights the complexity in dealing with individual variations and suggests further research to improve individual recognition capabilities from training data and test data conditionally on different pixels, background, and image quality with other application processes.

References

[1] Pratama and Angga Wahyudi Kurniawan, "Facial Image Recognition using YCbCr Color Space Feature Extraction and Principal Component Analysis (PCA) Method for Automatic Student Attendance," Journal of Technology Development Information and Computer Science, vol. V, no. 3, pp. 1158–1164, 2021.

[2] S. Sriyati, A. Setyanto, and E. E. Luthfi, "Literature Review: Pengenalan Wajah Menggunakan Algoritma Convolutional Neural Network," Jurnal Teknologi Informasi dan Komunikasi (TIKomSiN), vol. 8, no. 2, 2020.

[3] Nurdiansyah and Rudi, "FACIAL RECOGNITION SYSTEM USING EIGENFACE," vol. 3, no. 4, 2023.

[4] Theresia and Darujati and Cahyo, "IMAGE PROCESSING FOR FACE RECOGNITION USING OPENCV," Syntax Admiration Journal, vol. 2, no. 3, 2021.

[5] A. Fau and F. T. Waruwu, "DETECT THE SIMILARITY OF DIGITAL IMAGES USING THE EIGENFACE METHOD," JURTEKSI (Jurnal Teknologi dan Sistem Informasi), vol. 7, no. 2, pp. 133–142, 2021.

[6] Susi Tamba, "Designing an Employee Attendance Application with Face Detection Using the Eigenface Method," Journal of Informatics, Electrical and Electronics Engineering, vol. 2, no. 1, 2022.