Person Recognition From Face Images

Julian Velez

Southern Connecticut State University

CSC 481-01

1. Introduction

Facial recognition software systems are becoming more and more relevant. These systems are increasingly being used in a cacophony of applications including, but not limited to, biometric authentication, homeland security, and individually focused marketing and advertising. As such there is a litany of techniques which can be employed in the development of such systems. In this experiment, we will apply a feature focused approach using the K-Nearest Neighbors algorithm for matching identified faces.

2. Problem Description

We have been given a face data-set, from which we will identify persons from face images using a K-Nearest Neighbors classifier on seven specific features. In this case, because we are matching faces on a one to one basis, this particular experiment will require a 1-NN classifier. We will first need to extract the features from the data. Next, we will need to compare the features and deploy the classifiers. Finally, we will measure the performance of the classifier in terms of identification accuracy, precision, and recall rate.

3. Data Sets and Experiment Design

For this experiment, we are using a collection of pictures of 10 people, five male and five female. For each person, we have two images, collected in two different sessions and situations. Each image contains 22 markup points, with each point being represented by (x, y) coordinates in a separate data file. We will use these 22 points in the execution of our experiment. They will be used in the computation of the computation of the seven identified defining features, which we will use for facial comparison.

4. Defining Features and Feature Extraction

This experiment will make use of seven identified features:

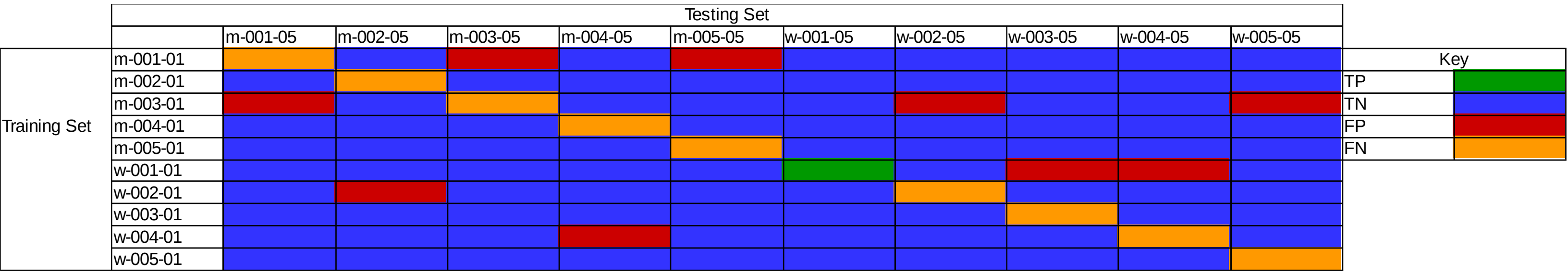
1. Eye length ratio: length of eye over distance between points 8 and 13
2. Eye distance ratio: distance between center of two eyes over distance between points 8 and 13
3. Nose ratio: distance between points 15 and 16 over distance between 20 and 21
4. Lip size ratio: distance between points 2 and 3 over distance between 17 and 18
5. Lip length ratio: distance between points 2 and 3 over distance between 20 and 21
6. Eye-brow length ratio: distance between points 4 and 5 (or distance between points 6 and 7) over distance between 8 and 13
7. Aggressive ratio: distance between points 10 and 19 over distance 20 and 21

We will extract these features from the data files by calculating the necessary Euclidean distances and further processing those results.

5. Result and Analysis

5.1 Confusion Matrix and Statistics

The results of our experiment produced the following confusion matrix:



		Testing Set									
		m-001-05	m-002-05	m-003-05	m-004-05	m-005-05	w-001-05	w-002-05	w-003-05	w-004-05	w-005-05
Training Set	m-001-01	🟧	🟦	🟥	🟦	🟥	🟦	🟦	🟦	🟦	🟦
	m-002-01	🟦	🟧	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦
	m-003-01	🟥	🟦	🟧	🟦	🟦	🟦	🟥	🟦	🟦	🟥
	m-004-01	🟦	🟦	🟦	🟧	🟦	🟦	🟦	🟦	🟦	🟦
	m-005-01	🟦	🟦	🟦	🟦	🟧	🟦	🟦	🟦	🟦	🟦
	w-001-01	🟦	🟦	🟦	🟦	🟦	🟩	🟦	🟥	🟥	🟦
	w-002-01	🟦	🟥	🟦	🟦	🟦	🟦	🟧	🟦	🟦	🟦
	w-003-01	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟧	🟦	🟦
	w-004-01	🟦	🟦	🟦	🟥	🟦	🟦	🟦	🟦	🟧	🟦
	w-005-01	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟧

Key	
TP	🟩
TN	🟦
FP	🟥
FN	🟧

where TP is a true positive result, TN is a true negative result, FP is a false positive result, and FN is a false negative result.

From our confusion matrix, we were able to calculate the following statistics for our experiment as well as each classifier:

- Precision: TP / (TP + FP)
- Recall Rate: TP / (TP + FN)
- Accuracy: (TP + TN) / total

For the experiment as a whole, we found:

Precision = 0.1

Recall Rate = 0.1

Accuracy = 0.82

With regard to our classifiers, they all but one had the same statistic values. Their precision and recall rates both equaled 0 and they had accuracies of 0.80. Our exception to this was w-001 which had a precision, recall rate, and accuracy of 1.

5.2 Analysis

Looking at the performance of our experiment, we can see that our classifiers did not perform very well, as a group. All except for one failed to find a correct match. I would suspect that the only reason that our accuracy rates are as high as they are is due to the nature of the classification technique, being 1-NN and therefore giving a decently high level of true negative results.

I hypothesize that our features may not be sufficient for the purposes of this experiment. It may be that adding more features to our test will improve the precision values of our outcome. Another possibility may be that a K-NN test is not the best match for this particular problem. Other techniques may provide better results.

6. Conclusion

In summary, we used the K-Nearest Neighbors (specifically 1-NN) technique for matching faces. We used seven features, which we extracted from the image data files. Only one face was able to be matched correctly. Further experimentation, using more or different features and techniques may be necessary for improving the results of our current experiment.

7. References

Class Notes

8. Appendix

8.1 Extracted Feature Values

m-001-01:

Eye Length Ratio: 0.1933

Eye Distance Ratio: 0.4238

Nose Ratio: 0.1660

Lip Size Ratio: 4.4743

Lip Length Ratio: 0.4362

Eye-Brow Length Ratio: 0.4267

Aggressive Ratio: 0.9445

m-001-05:

Eye Length Ratio: 0.1833

Eye Distance Ratio: 0.4453

Nose Ratio: 0.1629

Lip Size Ratio: 3.3823

Lip Length Ratio: 0.3989

Eye-Brow Length Ratio: 0.4220

Aggressive Ratio: 0.9259

m-002-01:

Eye Length Ratio: 0.2087

Eye Distance Ratio: 0.4803

Nose Ratio: 0.1354

Lip Size Ratio: 3.2876

Lip Length Ratio: 0.4129

Eye-Brow Length Ratio: 0.3744

Aggressive Ratio: 0.9818

m-002-05:

Eye Length Ratio: 0.1821

Eye Distance Ratio: 0.4903

Nose Ratio: 0.1380

Lip Size Ratio: 3.1397

Lip Length Ratio: 0.3808

Eye-Brow Length Ratio: 0.3580

Aggressive Ratio: 0.9036

m-003-01:

Eye Length Ratio: 0.2390

Eye Distance Ratio: 0.4689

Nose Ratio: 0.1478

Lip Size Ratio: 3.4981

Lip Length Ratio: 0.4837

Eye-Brow Length Ratio: 0.3507

Aggressive Ratio: 0.9621

m-003-05:

Eye Length Ratio: 0.1768

Eye Distance Ratio: 0.4846

Nose Ratio: 0.1724

Lip Size Ratio: 4.8612

Lip Length Ratio: 0.4331

Eye-Brow Length Ratio: 0.3757

Aggressive Ratio: 0.9367

m-004-01:

Eye Length Ratio: 0.2369

Eye Distance Ratio: 0.4660

Nose Ratio: 0.1315

Lip Size Ratio: 6.9409

Lip Length Ratio: 0.4385

Eye-Brow Length Ratio: 0.3693

Aggressive Ratio: 0.9681

m-004-05:

Eye Length Ratio: 0.2057

Eye Distance Ratio: 0.4668

Nose Ratio: 0.1514

Lip Size Ratio: 3.8732

Lip Length Ratio: 0.4429

Eye-Brow Length Ratio: 0.3617

Aggressive Ratio: 1.0012

m-005-01:

Eye Length Ratio: 0.2455

Eye Distance Ratio: 0.4810

Nose Ratio: 0.1213

Lip Size Ratio: 10.4283

Lip Length Ratio: 0.4368

Eye-Brow Length Ratio: 0.3365

Aggressive Ratio: 0.9059

m-005-05:

Eye Length Ratio: 0.1904

Eye Distance Ratio: 0.5095

Nose Ratio: 0.1174

Lip Size Ratio: 4.3470

Lip Length Ratio: 0.4007

Eye-Brow Length Ratio: 0.3593

Aggressive Ratio: 0.8603

w-001-01:

Eye Length Ratio: 0.2662

Eye Distance Ratio: 0.4850

Nose Ratio: 0.1536

Lip Size Ratio: 3.1096

Lip Length Ratio: 0.4569

Eye-Brow Length Ratio: 0.3629

Aggressive Ratio: 1.0557

w-001-05:

Eye Length Ratio: 0.2440

Eye Distance Ratio: 0.4904

Nose Ratio: 0.1593

Lip Size Ratio: 2.8390

Lip Length Ratio: 0.4251

Eye-Brow Length Ratio: 0.3861

Aggressive Ratio: 1.0435

w-002-01:

Eye Length Ratio: 0.2070

Eye Distance Ratio: 0.4715

Nose Ratio: 0.1457

Lip Size Ratio: 3.2671

Lip Length Ratio: 0.4024

Eye-Brow Length Ratio: 0.3351

Aggressive Ratio: 0.9169

w-002-05:

Eye Length Ratio: 0.2145

Eye Distance Ratio: 0.4500

Nose Ratio: 0.1546

Lip Size Ratio: 3.3245

Lip Length Ratio: 0.4347

Eye-Brow Length Ratio: 0.3308

Aggressive Ratio: 0.9608

w-003-01:

Eye Length Ratio: 0.2218

Eye Distance Ratio: 0.5046

Nose Ratio: 0.1456

Lip Size Ratio: 3.5064

Lip Length Ratio: 0.4367

Eye-Brow Length Ratio: 0.3494

Aggressive Ratio: 0.9154

w-003-05:

Eye Length Ratio: 0.1862

Eye Distance Ratio: 0.5201

Nose Ratio: 0.1668

Lip Size Ratio: 2.6463

Lip Length Ratio: 0.3742

Eye-Brow Length Ratio: 0.3852

Aggressive Ratio: 0.9054

w-004-01:

Eye Length Ratio: 0.2127

Eye Distance Ratio: 0.4834

Nose Ratio: 0.1349

Lip Size Ratio: 4.1490

Lip Length Ratio: 0.4287

Eye-Brow Length Ratio: 0.3730

Aggressive Ratio: 0.9337

w-004-05:

Eye Length Ratio: 0.1747

Eye Distance Ratio: 0.5023

Nose Ratio: 0.1396

Lip Size Ratio: 2.9980

Lip Length Ratio: 0.4192

Eye-Brow Length Ratio: 0.3660

Aggressive Ratio: 0.9468

w-005-01:

Eye Length Ratio: 0.2330

Eye Distance Ratio: 0.4616

Nose Ratio: 0.1363

Lip Size Ratio: 4.1649

Lip Length Ratio: 0.4508

Eye-Brow Length Ratio: 0.3582

Aggressive Ratio: 0.9948

w-005-05:

Eye Length Ratio: 0.2056

Eye Distance Ratio: 0.4704

Nose Ratio: 0.1455

Lip Size Ratio: 3.2174

Lip Length Ratio: 0.4553

Eye-Brow Length Ratio: 0.3760

Aggressive Ratio: 1.0333

8.2 Source Code

```
# Julian Velez

# 2/15/2017

# an ai program for recognizing faces by comparing seven key features:


import numpy as np

import math


# reads data into a 3d list (face_list[i][j][k])

# i -> the i+1 image

# j -> the j point in the image (starting from 0)

# k -> 0 is x and 1 is y

def init_data():

    face_list=[]


    for i in range(1,6):

        face_list.append(np.genfromtxt("m-00"+str(i)+"/m-00"+str(i)+"-01.pts",

                                        skip_header=3, skip_footer=1))

        face_list.append(np.genfromtxt("m-00"+str(i)+"/m-00"+str(i)+"-05.pts",

                                        skip_header=3, skip_footer=1))

    for i in range(1,6):

        face_list.append(np.genfromtxt("w-00"+str(i)+"/w-00"+str(i)+"-01.pts",

                                        skip_header=3, skip_footer=1))
        face_list.append(np.genfromtxt("w-00"+str(i)+"/w-00"+str(i)+"-05.pts",
                                        skip_header=3, skip_footer=1))

    return face_list

# compute the distance between two points
def find_dist(x1, y1, x2, y2):
    dist = math.sqrt((x2-x1)**2+(y2-y1)**2)

    return dist

# compute the eye length ratio of each image
# length of eye (maximum of two) over distance between points 8 and 13
def eye_length_ratio(face_list):
    list = face_list
    eye_length_ratios = []

    for i in range(0, len(list)):
        # find length of eye 1
        eye_1_x1 = list[i][9][0]
        eye_1_y1 = list[i][9][1]
        eye_1_x2 = list[i][10][0]
        eye_1_y2 = list[i][10][1]
        eye_1 = find_dist(eye_1_x1, eye_1_y1, eye_1_x2, eye_1_y2)

        # find length of eye 2
        eye_2_x1 = list[i][11][0]
        eye_2_y1 = list[i][11][1]
        eye_2_x2 = list[i][12][0]
        eye_2_y2 = list[i][12][1]
        eye_2 = find_dist(eye_2_x1, eye_2_y1, eye_2_x2, eye_2_y2)

        # find distance between points 8 and 13
        eight_x = list[i][8][0]
        eight_y = list[i][8][1]
        thirteen_x = list[i][13][0]
        thirteen_y = list[i][13][1]
        dist = find_dist(eight_x, eight_y, thirteen_x, thirteen_y)

        if eye_1 > eye_2:
            ratio = eye_1/dist
        else:
            ratio = eye_2/dist

        eye_length_ratios.append(ratio)

    return eye_length_ratios

# compute the eye distance ratio of each image
# distance between center of two eyes over distance between points 8 and 13
def eye_dist_ratio(face_list):
    list = face_list
    eye_dist_ratios = []

    for i in range(0, len(list)):
        left_eye_x = list[i][0][0]
        left_eye_y = list[i][0][1]
        right_eye_x = list[i][1][0]
        right_eye_y = list[i][1][1]
        eye_center_dist = find_dist(left_eye_x, left_eye_y, right_eye_x, right_eye_y)

        eight_x = list[i][8][0]
        eight_y = list[i][8][1]
        thirteen_x = list[i][13][0]
        thirteen_y = list[i][13][1]
        dist = find_dist(eight_x, eight_y, thirteen_x, thirteen_y)

        ratio = eye_center_dist/dist

        eye_dist_ratios.append(ratio)

    return eye_dist_ratios

# compute the nose ratio of each image
# distance between points 15 and 16 over distance between 20 and 21
def nose_ratio(face_list):
    list = face_list
    nose_ratios = []

    for i in range(0, len(list)):
        fifteen_x = list[i][15][0]
        fifteen_y = list[i][15][1]
        sixteen_x = list[i][16][0]
        sixteen_y = list[i][16][1]
        nostril_dist = find_dist(fifteen_x, fifteen_y, sixteen_x, sixteen_y)


        twenty_x = list[i][20][0]

        twenty_y = list[i][20][1]

        twentyone_x = list[i][21][0]

        twentyone_y = list[i][21][1]

        dist = find_dist(twenty_x, twenty_y, twentyone_x, twentyone_y)


        ratio = nostril_dist/dist


        nose_ratios.append(ratio)


    return nose_ratios


# compute the lip size ratio of each image

# distance between points 2 and 3 over distance between 17 and 18

def lip_size_ratio(face_list):

    list = face_list

    lip_size_ratios = []


    for i in range(0, len(list)):

        two_x = list[i][2][0]

        two_y = list[i][2][1]

        three_x = list[i][3][0]

        three_y = list[i][3][1]

        length = find_dist(two_x, two_y, three_x, three_y)


        seventeen_x = list[i][17][0]
        seventeen_y = list[i][17][1]
        eighteen_x = list[i][18][0]
        eighteen_y = list[i][18][1]
        size = find_dist(seventeen_x, seventeen_y, eighteen_x, eighteen_y)

        ratio = length/size

        lip_size_ratios.append(ratio)

    return lip_size_ratios

# compute the lip length ratio of each image
# distance between points 2 and 3 over distance between 20 and 21
def lip_length_ratio(face_list):
    list = face_list
    lip_length_ratios = []

    for i in range(0, len(list)):
        two_x = list[i][2][0]
        two_y = list[i][2][1]
        three_x = list[i][3][0]
        three_y = list[i][3][1]
        length = find_dist(two_x, two_y, three_x, three_y)

        twenty_x = list[i][20][0]
        twenty_y = list[i][20][1]
        twentyone_x = list[i][21][0]
        twentyone_y = list[i][21][1]
        dist = find_dist(twenty_x, twenty_y, twentyone_x, twentyone_y)

        ratio = length/dist

        lip_length_ratios.append(ratio)

    return lip_length_ratios

# compute the eye-brow length ratio of each image
# distance between points 4 and 5 (or distance between points 6 and 7 whichever is
larger)
# over distance between 8 and 13
def brow_length_ratio(face_list):
    list = face_list
    brow_length_ratios = []

    for i in range(0, len(list)):
        four_x = list[i][4][0]
        four_y = list[i][4][1]
        five_x = list[i][5][0]
        five_y = list[i][5][1]
        left_brow = find_dist(four_x, four_y, five_x, five_y)

        six_x = list[i][6][0]
        six_y = list[i][6][1]
        seven_x = list[i][7][0]
        seven_y = list[i][7][1]
        right_brow = find_dist(six_x, six_y, seven_x, seven_y)
        eight_x = list[i][8][0]
        eight_y = list[i][8][1]
        thirteen_x = list[i][13][0]
        thirteen_y = list[i][13][1]
        dist = find_dist(eight_x, eight_y, thirteen_x, thirteen_y)

        if left_brow > right_brow:
            ratio = left_brow/dist
        else:
            ratio = right_brow/dist

        brow_length_ratios.append(ratio)

    return brow_length_ratios

# compute the aggressive ratio of each image
# distance between points 10 and 19 over distance between 20 and 21
def aggressive_ratio(face_list):
    list = face_list
    aggressive_ratios = []

    for i in range(0, len(list)):
        ten_x = list[i][10][0]
        ten_y = list[i][10][1]
        nineteen_x = list[i][19][0]
        nineteen_y = list[i][19][1]
        dist_1 = find_dist(ten_x, ten_y, nineteen_x, nineteen_y)
        twenty_x = list[i][20][0]

        twenty_y = list[i][20][1]

        twentyone_x = list[i][21][0]

        twentyone_y = list[i][21][1]

        dist_2 = find_dist(twenty_x, twenty_y, twentyone_x, twentyone_y)


        ratio = dist_1/dist_2


        aggressive_ratios.append(ratio)


    return aggressive_ratios


# print the feature values for each image

def print_features(ratio_1, ratio_2, ratio_3, ratio_4, ratio_5, ratio_6, ratio_7):

    i = 0

    while i < 10:

        if i % 2 == 0:

            print("m-00" + str((i+2)/2) + "-01 feature values:")

        else:

            print("m-00" + str((i+1)/2) + "-05 feature values:")

        print("Eye Length Ratio: " + str(ratio_1[i]))

        print("Eye Distance Ratio: " + str(ratio_2[i]))

        print("Nose Ratio: " + str(ratio_3[i]))

        print("Lip Size Ratio: " + str(ratio_4[i]))

        print("Lip Length Ratio: " + str(ratio_5[i]))

        print("Eye-Brow Length Ratio: " + str(ratio_6[i]))

        print("Aggressive Ratio: " + str(ratio_7[i]) + "\n")
        i = i + 1
    while i < 20:
        if i % 2 == 0:
            print("w-00" + str((i/2)-4) + "-01 feature values:")
        else:
            print("w-00" + str(((i-1)/2)-4) + "-05 feature values:")
        print("Eye Length Ratio: " + str(ratio_1[i]))
        print("Eye Distance Ratio: " + str(ratio_2[i]))
        print("Nose Ratio: " + str(ratio_3[i]))
        print("Lip Size Ratio: " + str(ratio_4[i]))
        print("Lip Length Ratio: " + str(ratio_5[i]))
        print("Eye-Brow Length Ratio: " + str(ratio_6[i]))
        print("Aggressive Ratio: " + str(ratio_7[i]) + "\n")
        i = i + 1

# calculate distances between each image
# use feature values to find the closest match
def find_match(ratio_1, ratio_2, ratio_3, ratio_4, ratio_5, ratio_6, ratio_7):
    min_dist = float('inf')
    min_index = 0

    for i in range(1, len(ratio_1),2):
        for j in range(0, len(ratio_1),2):
            if i == j:
                continue
            else:
                dif_1 = ratio_1[i]-ratio_1[j]
                dif_2 = ratio_2[i]-ratio_2[j]
                dif_3 = ratio_3[i]-ratio_3[j]
                dif_4 = ratio_4[i]-ratio_4[j]
                dif_5 = ratio_5[i]-ratio_5[j]
                dif_6 = ratio_6[i]-ratio_6[j]
                dif_7 = ratio_7[i]-ratio_7[j]

                dist =
math.sqrt(dif_1**2+dif_2**2+dif_3**2+dif_4**2+dif_5**2+dif_6**2+dif_7**2)

                if (dist < min_dist):
                    min_dist = dist
                    min_index = j
        if (i < 11):
            if (min_index < 10):
                print("Image m-00" + str((i+1)/2) + "-05 is closest to image m-00" +
                        str((min_index+1)) + "-01")
            else:
                print("Image m-00" + str((i+1)/2) + "-05 is closest to image w-00" +
                        str(((min_index+1)/2)-4) + "-01")
        else:
            if (min_index < 10):
                print("Image w-00" + str(((i+1)/2)-5) + "-05 is closest to image m-
00" +
                        str((min_index+1)) + "-01")
            else:
                print("Image w-00" + str(((i+1)/2)-5) + "-05 is closest to image w-
00" +
                        str(((min_index+1)/2)-4) + "-01")
        min_dist = float('inf')


# the main function

def main():

    # initialize data

    face_list = init_data()


    # calculate the seven features for each image

    eye_length_ratios = eye_length_ratio(face_list)

    eye_dist_ratios = eye_dist_ratio(face_list)

    nose_ratios = nose_ratio(face_list)

    lip_size_ratios = lip_size_ratio(face_list)

    lip_length_ratios = lip_length_ratio(face_list)

    brow_length_ratios = brow_length_ratio(face_list)

    aggressive_ratios = aggressive_ratio(face_list)


    # print the features for each image

    print_features(eye_length_ratios, eye_dist_ratios, nose_ratios, lip_size_ratios, 
lip_length_ratios,

                   brow_length_ratios, aggressive_ratios)


    # find the matches for each image using the above features

    find_match(eye_length_ratios, eye_dist_ratios, nose_ratios, lip_size_ratios, 
lip_length_ratios,

                   brow_length_ratios, aggressive_ratios)

    print("\n")


main()
```