APPM 3310 Project:

Identifying Dog Breeds with Facial Recognition

By:

Gretchen Conley[[1]](#footnote-0)

Sophia Machen[[2]](#footnote-1)

Jackson Curry[[3]](#footnote-2)

Austin Hegemann[[4]](#footnote-3)

Abstract

The topic of facial recognition has begun to be used in a wide variety of applications. For our project we wanted to tackle something both challenging yet familiar, so we chose to implement facial recognition on dogs. Our goal was to create an interface through which a user can input a cropped face of a dog of their choosing from which our program would use facial recognition to return the correct breed for the dog that the user entered.

Our program was created using an approach similar to that of the work done by Muller, Magaia, and Herbst’s paper on facial recognition. First, we created a database of dogs comprised of 5 different images of a particular dog for each of the 7 breeds we chose. From this we found facial averages of eigenfaces using SVD factorization. We then compared our input image against our database to find the smallest difference between its vector projection and the vector projections of the other dogs in our database. Finally we found the breed of the dog in our database which the new, test dog was closest to. Through multiple trials and after several different versions of our program we found that our final version of the program was able to correctly identify the dog breed approximately 83% of the time which was really exciting.

Attribution

Gretchen Conley was the lead MATLAB programmer and helped in writing the final report. Sophia Machen helped formulate ideas for how our ideas could be implemented as code, and contributed to writing as well as editing the final report. Austin Hegemann offered MATLAB support, helped to create the training set library, found excellent papers to reference, and helped in the writing and editing of the final report. Jackson Curry also helped to create the training set library, contributed to writing and editing the final report, and offered key insights as to how the algorithm could be improved. All group members were involved in research, formulation, and analysis of our final product which was a facial recognition program to identify dog breeds.

Introduction

The task of facial recognition can be performed in a variety of ways, however one mathematical model can be applied to categorize image data into different preset classes. Input image data, while highly variable, can in fact be broken down into patterns such as the location and relative distance of various features, e.g. the eyes, nose, and mouth. Mathematically, these features can be efficiently represented in terms of eigenfaces (the mathematical representation of their principal components). The extraction of these components from original input data can be done using Principal Component Analysis (PCA) coupled with Singular Value Decomposition (SVD). PCA can be used to transform images into their corresponding eigenfaces, which represent only the most important features of an image, upon which comparisons can be made. SVD compares eigenfaces to an orthonormal basis with their projections onto the basis being the primary tool for comparison. These projections allow one to compare images to a previously declared set, and the similarities between projections can be used to calculate similarity.

The purpose of this research is to use facial recognition, traditionally applied to human faces, and determine if it can recognize different breeds of dogs. While the faces of dogs and humans differ greatly, the same model should be applicable. Individual dog breeds have very specific facial features, just as different humans have different features. The locations of the nose, mouth, eyes, and ears are characteristic of different dog breeds, and thus should be able to be compared using PCA, just as is done with humans. The same model, as outlined in “Singular Value Decomposition, Eigenfaces, and 3D Reconstructions” by Muller, Magaia, and Herbst, will be utilized. However, instead of using human faces, generic pictures of various dog breeds will be used to establish a database, and input images will be compared to this database. The database will be composed of images of Golden Retrievers, Corgis, Pugs, Schnauzers, King Charles Spaniels, Bull Terriers, and Greyhounds. These dogs were chosen due to their highly individualistic features and characteristics. Each database photo is comprised of a portrait view of the dog, with the face in full focus, a blurred background, with the entire face is present in the picture, as seen in the figures below.



|  |  |  |  |
| --- | --- | --- | --- |
| Figure 1: Corgi Reference | Figure 2: Golden Retriever Reference | Figure 3: Greyhound Reference | Figure 4: Pug Reference |

We aim to prove that the methods of facial recognition can be used on a wide range of applications, and that the same formulas can be applied.

Mathematical Formulation

The algorithm used is three-fold, first establishing a database of eigenfaces, then transforming an input image into eigenfaces, and finally comparing the input image to the database through projections. In order to transform database images into eigenfaces, each image is converted to grayscale and standardized to the same size (100x100). In Matlab, each image is represented as a 2D nxn matrix of 8-bit intensity values, in this case we standardize our ‘n’ to 100 so that each image matrix is the same size, and thus is easy to average and manipulate. This matrix is then converted into a positive definite symmetric matrix using the Matlab command posdef(matrix). Then, each database image is converted to a column vector by concatenating the columns of the image matrix resulting in a size of 10,000x1. The average matrix of the database is calculated in function (1), where M is the total number of images in the database. Then, this average matrix is then subtracted by each original face as is represented in function (2):

(Function 1)

(Function 2)

Once this has been done, a “training set” must be created. Each is a new column of a matrix, “A,” which constitutes the training set, and is size mxn, with m being the number of pixels in the compressed images, and n is the number of images in the database. Finally, the singular value decomposition is computed using the Matlab command [U,S,V] = svd(A), such that

(Function 3)

Let the first ‘n’ columns of U form an orthonormal basis for the training set. Each individual column that makes up this orthonormal basis can equivalently be called an “eigenface”.

The eigenfaces are important, and cool, to find because form them, you can create any image in the training set using a linear combination of these eigenfaces since they are an orthonormal basis which spans the entire training set. Next, we project each vector from the training set onto this orthonormal basis. In the report we will sometimes refer to these vectors which we define as the projection onto the set of eigenvectors of the difference of the current dog image and the average dog image as the “eigenface representation” for a particular dog. Note that this term of “eigenface representation” is the same term that Muller, Magaia, and Herbst defined on page 531 of their paper.

After the database has been established, an input image can be compared. And similar to what we did for our database images, this image must be converted to grayscale, converted into a positive definite symmetric matrix, and concatenated into a 10,000x1 vector. In order to compare the input image to the images contained in the training set, the input image (after it had been manipulated to its eigenface components) is projected onto the orthonormal basis of the training set of eigenfaces. All of the training set vectors are also projected onto the basis. Finally, the differences between the projection of the input image and the training set images are computed, and the training set vector projection most similar to input image vector projection, the one with the smallest difference, is selected. Indexing is used to trace the most similar training set vector back to the database dog image. This is the dog breed that the input image most closely resembles. Our program also outputs the input image next to the image it was found to most closely resemble in addition to the statement of which dog breed it most closely corresponds to.

Examples & Numerical Results

First, our method developed “eigenfaces” of the dogs. The eigenfaces formed the orthonormal basis that we projected our training set and input vectors to.

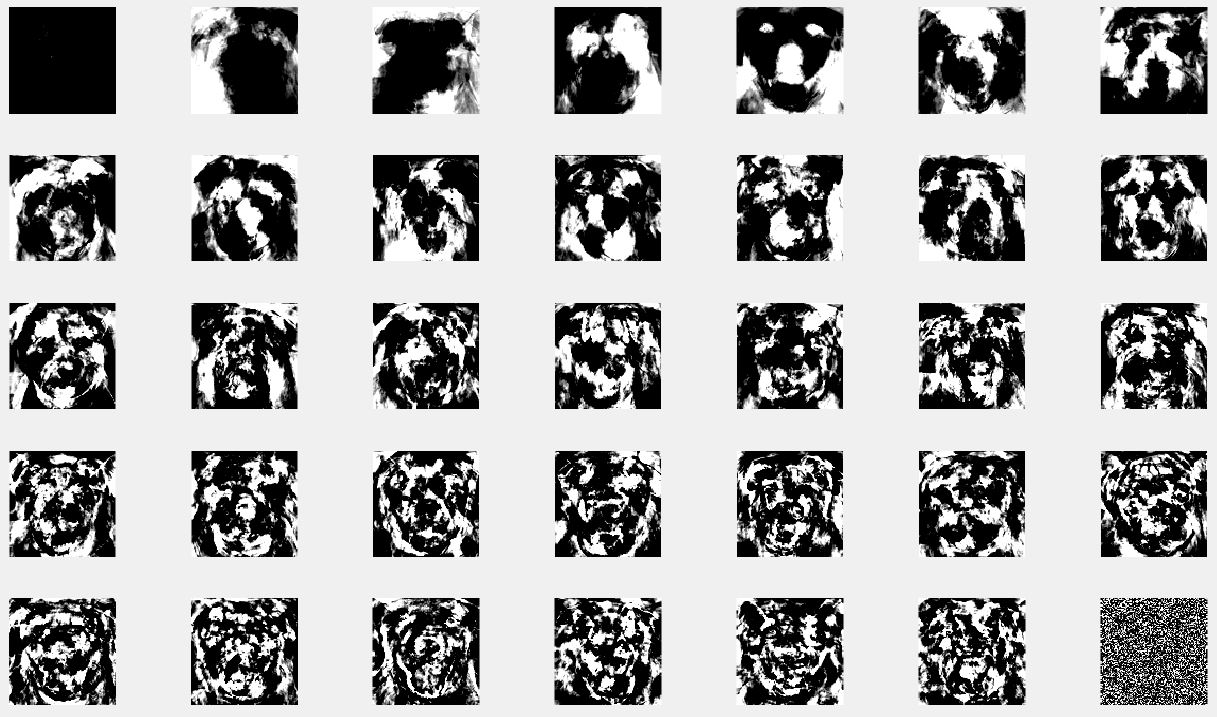
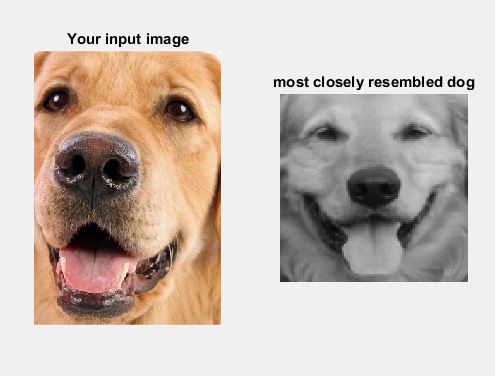


Figure 5: These are the 35 eigenfaces we found for our database

To test this method, images were gathered from the internet of various dogs from the breeds in our database. These images were input, compressed to vectors, and projected into the orthonormal basis of the training set.



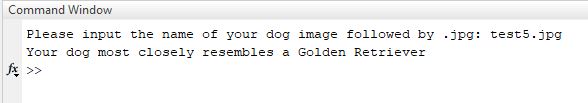


Figure 6/7: Example of MATLAB output with a user input image

The difference between the input image projection and the database projections were calculated. The database breed corresponding to the lowest error was output as the identified breed.

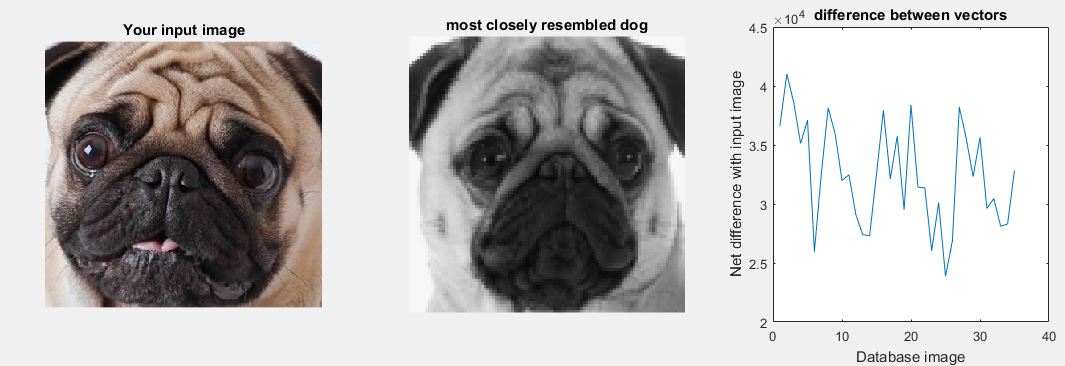


Figure 8: Example output of a test “Pug” image correctly identified by our method

The “most closely resembled dog” is the image of the dog from the database whose vector projection most closely matched that of the input image. As seen in the chart above, it is not unusual for another dog in the database to have a strong correlation with the input image.



Figure 9: Input “Spaniel” image incorrectly identified as “Golden Retriever”

With some input images, this can cause an incorrect dog breed to be identified if a second image has a stronger correlation. As shown in the chart above right, the spaniel images (images 31-35) have an image that is correlated with the input image. However, the golden retriever images (images 11-15) have one outlier that is more strongly correlated with the input image. This results in an incorrect output identification of “golden retriever” for an input image of a spaniel.

A possible error could be due to the input image of a puppy face which contains different features than adult dogs, which our database is solely comprised of. This could be countered by creating a larger database of dog images (say, 10,000 dog images instead of 35) containing pictures of all ages of dogs as well with various backgrounds to account for this error.



Figure 10: Example of Method applied to dog breed outside of database

Applying our algorithm to a breed outside of our database showed correlation with many different database images (note the similar low error in the chart above right for many database images). This is to be expected since we don’t have a database containing all breeds of dogs, so parts of the input dog could correlate with different features of database dogs. For example, the input dog could have similar eyes to one breed, similar nose to a second breed, and similar ears to a third breed. This would cause it to show correlation with several breeds of dogs and several database images. In future applications of our method we would add more breeds of dogs to our database to reduce this error, as the input dog would likely most correlate all traits most strongly with its own breed.



Figure 11: Example of Method applied to image from the database

On the other hand, applying our algorithm to a database image returned error of zero with the image in the database. This is expected because the vector for the input image projected into the orthonormal basis will be the same as the corresponding database projected vector. Thus, the difference between the vectors is zero. The method will always correctly identify the database image, and return the greyscale version.

In summary, the absolute difference between the projected input image and projected database images is calculated. The smallest difference represents the most highly correlated vectors - vectors created by the most similar combination of basis vectors. The database dog vector that has the highest correlation is identified as the dog breed of the dog in the input image.

Discussion, Future Work & Conclusions

During the testing phase we learned a lot more about the intricacies of how our program made a final decision for which breed it thought the new input dog was. To recap, what our program did was it compared this new dog’s eigenface representation against the eigenface representation of every single dog in our training set, and then chose the breed of whichever dog had the least difference. So in our specific case, we compared the new dog’s eigenface representation against the eigenface representation of each of the 35 dogs in the training set, and then we returned the breed of the “chosen” dog from our training set as our best guess. Thus, as we add more dogs into our training set our program theoretically should work better and better because there would be a higher chance that a new input dog would look very similar to a dog of the same breed in our library resulting in the correct breed being chosen more often.

Now, while this approach worked well in general, we found that there has been one issue which stemmed from whenever we had an irregular image in our training set. The problem was that one irregular image in the training set had the power to make an entire breed of dogs not work correctly in our program. One specific example of this that we came across during our development of the project was a pug in our training set whose cheeks were particularly saggy making its face look similar in shape to the typical Schnauzer dog which has the long fur all around the mouth. Because of this one strange looking pug that we had in our training set, for almost any new Schnauzer picture that we tested our program would identify it as a pug. Then when we tested these same Schnauzer pictures again after removing that one weird pug photo from the training set, we found that our program was capable of identifying all the Schnauzers correctly. Other possible problems in our training set could arise from other factors such as the angle of the photo, the background lighting, and how the picture was cropped. Due to the relatively small size of our training set we were able to identify these strange photos in our training set and replace them.

Later our group had a discussion about what would happen if we expanded our training sets to a size where this method of identifying and replacing abnormal photos would not be feasible. This led us to debate whether we would have to come up with an alternative algorithm to avoid this issue, or if making the database really large would make our current algorithm work without any necessary maintenance from us. Our alternative algorithm that we came up with was that for each breed we would find the average column vector for the entire breed and then find the eigenface representation of this single vector instead of finding the eigenface representation for each separate dog. The first advantage to this algorithm that we saw was that no matter how large your training set becomes you will only have to compare the new dog you are testing to however many options there are for its breed. For example, in our current program we would only have to check the similarity between the new dog’s eigenface representation and the 7 averaged breed representations instead of having to compare it to all 35 representations. However, if our database was expanded to 1,000 images with 100 breeds then it would only have to compare to 100 representations instead of all 1,000 thus saving time. Another advantage to this algorithm is that one strange image in our training set would no longer have the power to ruin the results for an entire breed. This is because in this scenario with many images in our training set, one strange photo would only be able to affect the average of the breed ever so slightly. We did end up testing this new algorithm using our current library, but it didn’t perform nearly as well as our original algorithm. While we believe that this difference in performance was mainly due to the relatively small size of our photo library, we aren’t completely sure and this would definitely be a good topic for us to continue researching in the future.

Overall our group was really happy with how our project turned out. We were proud of the fact that we proved facial recognition of dogs is possible as well as the fact that our program had approximately an 83% success rate. It also amazed all of us to see how topics we had learned from this course, such as SVD decomposition and creating orthonormal bases, could be combined and applied to solve the real world problem of facial recognition which we imagined to be extremely complex before starting this project.

References

*Singular Value Decomposition, Eigenfaces, and 3D Reconstructions*, by Muller, Magaia, and Herbst, SIAM Review (2004).

Turk, M and Pentland, A. *Face recognition using eigenfaces*. In Proc. of Computer Vision and Pattern Recognition, pages 586–591. IEEE, June 1991b. Retrieved from http://www.cs.wisc.edu/~dyer/cs540/handouts/mturk-CVPR91.pdf

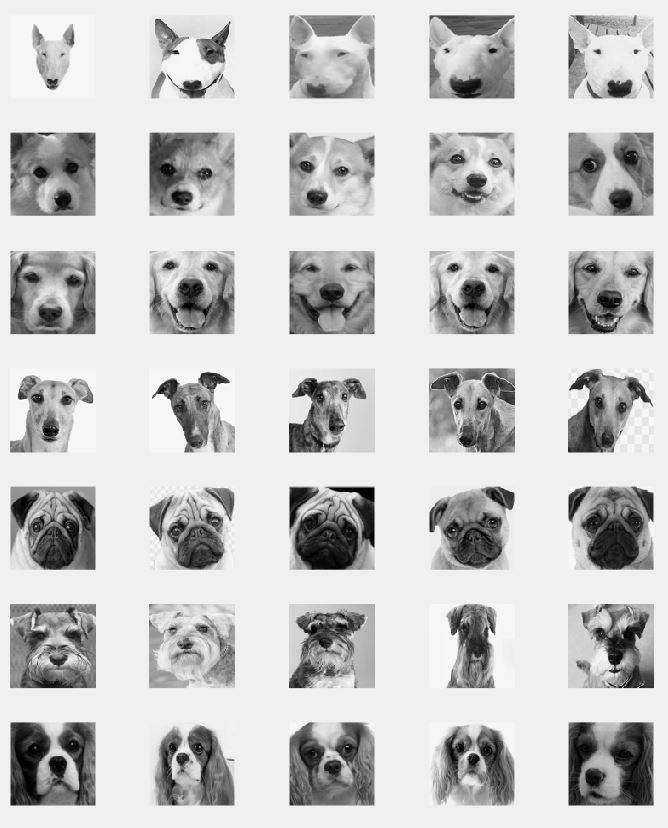
Pissarenko, D. (2002). *Eigenface-based facial recognition*. Retrieved from http://openbio.sourceforge.net/resources/eigenfaces/eigenfaces.pdf

Appendix

### **Notation**

|  |  |
| --- | --- |
|  | Database Set |
|  | Face image i in the Database Set |
|  | New (input) image |
|  | Average face |
|  | Number of eigenfaces |
|  | Column of Matrix A |
| A | Matrix (the training set) |
| U | Unitary Matrix in the SVD of Matrix A |
| S | Diagonal Matrix in the SVD of Matrix A with non-negative real numbers on the diagonal |
| V | Conjugate Transpose of the unitary matrix in the SVD of Matrix A |
| B | The columns of B are the projections of each vector from the training set onto the orthonormal basis U, the columns of B are eigenfaces. |

**Dog Database Images**

****

**Matlab Code**

***Main Code Script***

%Dog eigenfaces

%Clear workspace of miscellaneous data

clc

clear all

%Create a database & load in images

% Breed A - Bulldog - Removed due to inconsistent data set

% Breed B - Bull Terrier

% Breed C - Corgi

% Breed D - Golden

% Breed E - Greyhound

% Breed F - Pug

% Breed G - Schnauzer

% Breed H - Spaniel

%forming a single vector/ unfolding the image into a single vector (stack

%rows as columns into one vector)

%Bull Terrier Database

b1=imread('Bull Terrier.jpg');

b2=imread('bull terrier 2.jpg');

b3=imread('bull terrier 3.jpg');

b4=imread('bull terrier 4.jpg');

b5=imread('bull terrier 5.jpg');

pb1=posdef(b1);

pb2=posdef(b2);

pb3=posdef(b3);

pb4=posdef(b4);

pb5=posdef(b5);

Bavgmat= (1/5)\*(pb1+pb2+pb3+pb4+pb5);

%Corgi Database

c1=imread('Corgi.jpg');

c2=imread('corgi2.jpg');

c3=imread('corgi3.jpg');

c4=imread('corgi4.jpg');

c5=imread('corgi5.jpg');

pc1=posdef(c1);

pc2=posdef(c2);

pc3=posdef(c3);

pc4=posdef(c4);

pc5=posdef(c5);

Cavgmat= (1/5)\*(pc1+pc2+pc3+pc4+pc5);

%Golden Database

d1=imread('golden1.jpg');

d2=imread('golden2.jpg');

d3=imread('golden3.jpg');

d4=imread('golden4.jpg');

d5=imread('golden5.jpg');

pd1=posdef(d1);

pd2=posdef(d2);

pd3=posdef(d3);

pd4=posdef(d4);

pd5=posdef(d5);

Davgmat= (1/5)\*(pd1+pd2+pd3+pd4+pd5);

%Greyhound database

e1=imread('greyhound.jpg');

e2=imread('greyhound2.jpg');

e3=imread('greyhound3.jpg');

e4=imread('greyhound4.jpg');

e5=imread('greyhound5.jpg');

pe1=posdef(e1);

pe2=posdef(e2);

pe3=posdef(e3);

pe4=posdef(e4);

pe5=posdef(e5);

Eavgmat= (1/5)\*(pe1+pe2+pe3+pe4+pe5);

%Pug Database

f1=imread('Pug.jpg');

f2=imread('pug1.jpg');

f3=imread('pug2.jpg');

f4=imread('pug3.jpg');

f5=imread('pug4.jpg');

pf1=posdef(f1);

pf2=posdef(f2);

pf3=posdef(f3);

pf4=posdef(f4);

pf5=posdef(f5);

Favgmat= (1/5)\*(pf1+pf2+pf3+pf4+pf5);

%Schnauzer Database

g1=imread('Schnauzer.jpg');

g2=imread('schnauzer2.jpg');

g3=imread('schnauzer3.jpg');

g4=imread('schnauzer4.jpg');

g5=imread('scnauzer5.jpg');

pg1=posdef(g1);

pg2=posdef(g2);

pg3=posdef(g3);

pg4=posdef(g4);

pg5=posdef(g5);

Gavgmat= (1/5)\*(pg1+pg2+pg3+pg4+pg5);

%Spaniel Database

h1=imread('Spaniel.jpg');

h2=imread('Spaniel1.jpg');

h3=imread('spaniel2.jpg');

h4=imread('spaniel3.jpg');

h5=imread('spaniel5.jpg');

ph1=posdef(h1);

ph2=posdef(h2);

ph3=posdef(h3);

ph4=posdef(h4);

ph5=posdef(h5);

Havgmat= (1/5)\*(ph1+ph2+ph3+ph4+ph5);

%Average face

avgmat= (1/7)\*(Bavgmat+Cavgmat+Davgmat+Eavgmat+Favgmat+Gavgmat+Havgmat);

%Create the training set - the difference between each dog face in

%the database with the average of all dog faces in the database

dogfaces=[pb1 pb2 pb3 pb4 pb5 pc1 pc2 pc3 pc4 pc5 pd1 pd2 pd3 pd4 pd5 pe1...

pe2 pe3 pe4 pe5 pf1 pf2 pf3 pf4 pf5 pg1 pg2 pg3 pg4 pg5 ph1 ph2 ph3 ph4 ph5];

%tset is a matrix of our training set vectors

tset=zeros(10000,35);

for i=1:35

tset(:,i)=dogfaces(:,i)-avgmat ;

end

% find the SVD of the training set using the built in svd function

[U,D,V]=svd(tset);

%Take the first 35 columns of U as the orthonormal basis for the eigenspace

Uv=U(:,1:35);

%displays eigenfaces as an image

for j=1:35

subplot(5,7,j);

imshow(vec2mat(128\*(U(:,j))));

end

%Ask the user to input their dog type that they wish to identify

file=input('Please input the name of your dog image followed by .jpg: ','s');

inmat=imread(file);

%Turn the input image into a vector of the same dimension as those in our

%training set

facevec=posdef(inmat);

unknownvec=facevec-avgmat;

%Project the vectors from our training set onto the orthonormal basis

eigenface=[];

for l=1:35

eigenface(:,l)=Uv'\*tset(:,l);

end

%Find the difference between the input image vector projected into the

%orthonormal basis and the new vector projected into the orthonormal basis

err=[];

for k=1:35

err(:,k)=abs(Uv'\*unknownvec-eigenface(:,k));

end

%Sum the elements in the difference vector to find the "net difference"

%between the input image and each training set image

sums=[];

for m=1:35

sums(m)=sum(err(:,m));

end

%Then, find and locate the minimum difference to identify the dog breed

dognames={'Bull Terrier','Bull Terrier','Bull Terrier','Bull Terrier','Bull Terrier', 'Corgi', 'Corgi', 'Corgi', 'Corgi', 'Corgi', 'Golden Retriever','Golden Retriever', 'Golden Retriever', 'Golden Retriever', 'Golden Retriever', 'Greyhound','Greyhound','Greyhound','Greyhound','Greyhound','Pug','Pug','Pug','Pug','Pug','Schnauzer','Schnauzer','Schnauzer','Schnauzer','Schnauzer','King Charles Spaniel','King Charles Spaniel','King Charles Spaniel','King Charles Spaniel','King Charles Spaniel'};

dogtypes=sums;

diff=min(sums);

dogidx=find(dogtypes==diff);

%Display result

fprintf('Your dog most closely resembles a %s \n', dognames{dogidx} )

figure

subplot (1,3,1)

imshow(inmat)

title ('Your input image');

subplot (1,3,2)

imshow (vec2mat(dogfaces(:,dogidx))/265)

title ('most closely resembled dog')

subplot (1,3,3)

plot(sums)

title ('difference between vectors')

xlabel('Database image')

ylabel('Net difference with input image')

***Referenced Function: Posdef***

function [ newvec ] = posdef( inmat )

%Image Compression & formatting function

%Use Luminance\_L function to convert to greyscale

outimg=luminance\_L(inmat);

%Convert to data type double for function operations

outimg=double(outimg);

%resize the image so that all are the same size

outimg=imresize(outimg, [100 100]);

%Reshape into a single vector

newvec=reshape(outimg,[],1);

end

***Referenced Function: Luminance\_L***

function [ outimg ] = luminance\_L( inimg )

%This function creates a grayscale version of an image with given adjusted

%values to the red, green, and blue layers

[h,i,j]=size(inimg);

M=zeros(h,i,1);

N=zeros(h,i,1);

P=zeros(h,i,1);

for k=1:j

if k==1

M=uint8(inimg(:,:,k)\*.299);

elseif k==2

N=uint8(inimg(:,:,k)\*.587);

elseif k==3

P=uint8(inimg(:,:,k)\*.114);

end

end

%Add the layers together to form a 2D matrix for output

outimg=M+N+P;

end

***Referenced Function: vec2mat***

function [ outmat ] = vec2mat( invec )

% Changes a eigenface vector back into a matrix

outmat=reshape(invec',100,100);

end

1. 106535358 - 003 [↑](#footnote-ref-0)
2. 106549584 - 003 [↑](#footnote-ref-1)
3. 107043585 - 002 [↑](#footnote-ref-2)
4. 105972639 - 003 [↑](#footnote-ref-3)