Facial Recognition With Facial Accessories

Antonio Castro, Aaron Jouvenat
University of Nebraska-Lincoln

Objective

Our goal was to create a facial recognition system that is more reliable for those who may have something covering their face. This is a useful tool because people often have articles of clothing covering their faces. Some examples where this is common is up north during the winter where people dawn scarves and hats for winter or the middle east where the sun calls for protective clothing for their skin. Overall the system aims not to necessarily reinvent the wheel when doing facial recognition as much as combine and augment already existing technologies in order to increase the hit rate of facial recognition.

Due to the limited time of the class our goal as to create a system that mainly focused on front on views of the person. This makes things more simple as we don't have to worry about facial features being obscured by angles.

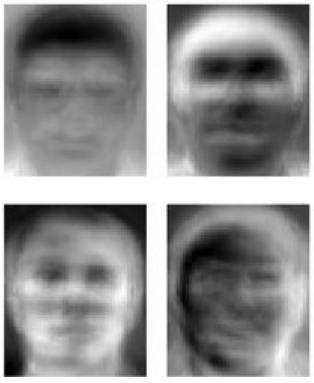
Overview

What we implemented is a mix of the eigenface system and the gait energy image. The eigenface matching system works as it sounds. We calculate the eigenvectors of the correlation matrix and looking for the largest eigenvector. This method was chosen because it is a well known process that has been proven to be successful in facial recognition in the past. Gait energy image which is usually used to identify people by their walking cycle proved effective in identifying people by just their faces. We found this method was viable because each image was slightly offset which allowed us to take each image and build a gait energy image with these as its base. Both methods have a high chance rate of matching on images that contain faces without clothes obstructing them. Our goal is to combine the results of the two and threshold which technique to use at certain situations in an attempt to increase our hit percentage.

Data Set

The data set we used was from the university of Cambridge. The face set consists of 40 sets of ten images. The images consist of forward facing people from the shoulders up. The images have the people set in slightly different positions each time which is what makes us able to use the gait energy image method. The main issue we ran into initially is that we could not find an image set that consisted of people with clothes obscuring their face. We went through a

series of other options. Below in Figure 1 you can see an example of some early testing images we used. To create these we simply used microsoft paint to paint black objects onto people. Although rudimentary this method was very helpful in deciding how much of the face obscured still returned a fairly reasonable result. This is because we could easily cover different parts of the face and record whether the results were correct or not. Figure 2 Shows some examples of tests that were done to see what parts of the face were most crucial. To do this we simply has a growing “curtain” or black spread across each person's face. Now understandably these images are less than professional though they are great for testing. So for our final version we have photoshopped different articles of clothing onto our test images in an attempt to more accurately represent what these images would look like with clothing on. This can be seen in Figure 3.



Eigenfaces

Eigenfaces is essentially the name given to eigenvectors that are used for Face recognition. The eigenvectors are derived from the covariance matrix of the probability distribution of face images.The secret of getting the eigenfaces can be generated by performing Principal component analysis (PCA) on a large set of images depicting different human faces. Eigenfaces are considered a standardized set of face ingredients derived from the analysis of many different faces. For example one’s face may be composed of the average face plus 10% from eigenface 1, 30% from eigenface 2 and -3% from eigenface 3. The eigenfaces that are created appear as light and dark areas that are arranged in specific patterns. This pattern can be seen in Figure 4 below and represents how different features of a face are singled out to be evaluated and scored. The way we evaluate the clusters of faces is but PCA. This is a Statistical procedure that used orthogonal transformation to convert a set of observations of possibly

correlated variables into t set of values of linearly uncorrelated variables called principal components. By finding the largest eigenvector from our covariance matrix we are able to assure that this vector has the highest variance which means that it accounts for as much data as possible.

We will now go through the process of recognition using eigenfaces. For initial testing We started by picking an index. This was done initially when testing so that we could choose a random face from the 400 face database and try to find a match. For current testing however I selected the faces I wanted to try to find a match for. It is very important that we normalize the images and resample them so we can line up all of the faces eyes, lips, etc. We then represent every image as a vector.

Figure 5 shows our first step. We create a vector of 1's we will use this late to return our mean vector to its original size. Line 2 shows us getting the mean of all of the training images vectors. This gives us a mean face. After multiplying this mean vector times our vector of 1's we subtract this matrix from the set of all of the training images. This makes each eigenface more distinguishable.

```
O=uint8(ones(1,size(v,2)));
m=uint8(mean(v,2));
vzm=v-uint8(single(m)*single(O));
```

Figure 6 is where we compute the eigenvectors of our correlation matrix. The first thing we need to do is to get our correlation matrix. We do this by multiplying our newly found matrix with the mean removed by itself transposed. Once we get the correlation matrix we need to get the eigenvalue of it. This is very easily done by using one of matlab's functions named eig. This function takes in a matrix and returns its eigenvector. We then multiply the eigenvector with our mean removed matrix. We then take the 10 largest eigenvalues because this will give us the most variance.

```
L=single(vzm)'*single(vzm);
[V,D]=eig(L);
V=single(vzm)*V;
V=V(:,end:-1:end-(N-1));
```

Figure 7 is where we calculate the signature for each image. This signature for each image represents what make them unique from the other eigenfaces and is very important for distinguishing the eigenface. We store all of these signatures in cv where each row is a signature for one image. The system right now is running 20 signatures on each image which is represented by N.

```
cv=zeros(size(v,2),N);
for i=1:size(v,2)
    cv(i,:)=single(vzm(:,i))'*V;
end
```

Fig. 7

Figure 8 is where the classifying occurs. We take our image which we represent as IM. We immediately subtract the mean vector from our test image. This once again increases the distinctness of our images vector and allows classification to be more easily completed. We then multiply the outcome of of the subtraction with the ten largest eigenvectors which is represented as s. Inside of our for loop we create a vector of all of our signature points our newly found vector s. We take the min of the value of this vector because this would mean that the s value was the largest. This would mean that the eigenvector was the largest and that it has the most variance.

```
subplot(122);
p=IM-m;                                % Subtract the mean
s=single(p)'*V;
z=[];
for i=1:size(v,2)
    z=[z,norm(cv(i,:)-s,2)];  % norm of signatures - (ei
    if(rem(i,20)==0),imshow(reshape(v(:,i),112,92)),end;
    drawnow;
end

[a,i]=min(z);
```

Fig. 8

are not constructed from a video sequence, but may consider many facial poses from that angle. Then feature extraction is performed to measure things like distance ratios between biological features of the face i.e. eyes, mouth, ears

Eigenface Results

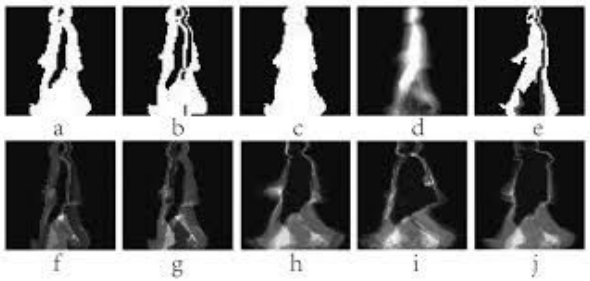
Results with the eigenface system was very successful on normal facial recognition. Without clothing obstructing the faces view we go a very solid 93% match rate. Light obstruction such as as a hat or scarf only covering the chin still allowed it to shine at a 85% match rate. Heavy clothing however shaved down the hits to just under 60%. This is obviously due to the reduced signatures we could extract due to obstruction.

GEI techniques applied to facial recognition

The motivation behind this was to see if the techniques applied of the construction of a Gait Energy Image (GEI) and the comparison methods could be applied to facial recognition. This was an experiment to see if it was possible to exploit an already existing computer vision method, but for another purpose for which it was not originally intended. A GEI is used for gait analysis, when the subject is wearing a variety of clothes. The underlying locomotion does not change, unless the clothing inhibits the subject, however it does make other gait analysis techniques less accurate since clothing may significantly alter the subject's silhouette, in turn affecting feature extraction.

In facial recognition, the data sets may involve a single view per training set, and

etc. then a test image is classified against the training data. Below is a GEI construction as it is applied to gait.



Above: Gei construction for gait analysis

Image construction

First we perform noise removal using the medfilt2() function provided by matlab to remove any small artifacts that may potentially disrupt image construction. This filter uses a 7x7 kernel, and helps remove salt and pepper noise that may have been introduced in the image due to flaws in digital photography. Next, we convert the image to the binary format, meaning the image is converted to pure black or white pixels based on some thresholding value. Next all the images in the training set are added together, adding each corresponding pixel position to each other, then averaged by dividing by the number of the training images. To account for the clothing variation, a gaussian blur filter is then applied to help further reduce image imperfections.
For the test image, the same process is performed but only on a single test image instead of multiple images. Furthermore, the training image was constructed from the first 9 of the subject images, while the tenth served as the test image. Meaning

High level algorithm overview

The steps to construct a training image, process and calculate the differences are as follows:

For each subject (1-6)

For each training image (1-9)
Convert to grayscale
Remove noise
Convert to binary format
Add to sum image
End

Average sum image pixels
Apply gaussian blur
Write sum image to disk
Calculate baseline difference and save to training matrix

For each test image (1-6)
Convert to grayscale
Remove noise
Convert to binary format
Add to sum image
Average sum image pixels
Apply gaussian blur
Calculate differences and write to test matrix
End
End

Run knn classifier
Display results

The algorithm can be simplified by saving the training image and/or baseline percent differences so they don't have to

the training image did not include data from the test image.
be run every single trial. Thus two loops, the outer loop, and the training construction loop, reducing the complexity from O(n^(2m)) + O(KNN) to O(n^2) + O(KNN). Where n = number of subjects, m = number of test images per subject, and O(KNN) is the complexity of the knn classification process.

Above: Here we show the processing of a subject's training image, it's binary representation, and finally how it's summed, added then average to the final training image.
Initially a training matrix was constructed of all zeros, meaning the ideal test match would have a 0% difference between the training data; however, this proved to be flawed as no correct matches came as a result. First we tested unobstructed faces

Testing and Comparison
To compare the training image and test image, two methods were used. A direct difference and a percentage difference. Then, a knn classification was performed to find the kth nearest neighbor. In this case, we only returned the first closest neighbor. To run the matlab knn classifier, a training data matrix, test data matrix and the group identifiers needed to be provided. The knn classifier works by comparing the training data row to each test data row and returning the associated training id that is the closest based on euclidian distance.

Percentage difference formula

$$\frac{|V_1 - V_2|}{\frac{(V_1+V_2)}{2}} \times 100 =$$

Here V1 and V2 are the sum training image and test image respectively.

Direct difference formula

$$D(G_g, G_p) = \frac{\sum_{x,y} |G_g(x,y) - G_p(x,y)|}{\sqrt{\sum_{x,y} G_g(x,y) \sum_{x,y} G_p(x,y)}}$$

Here Gg and Gy are the sum training image and test image respectively.

The x and y represent each pixel coordinate.

and had a 0% match among all 6 of the subjects.

Next we tried building a baseline for the training data by randomly selecting an ith image from the training data set to perform the differences. The idea was that we could get the ideal difference of a known match, then the test image would be closer to those baseline values. Now the result was 33.33% correct matches.

Next we looked at changing values within the used matlab functions and using different functions. PLaying with the sigma value of the gaussian blur did not change the results, although anything over .4 reduced the accuracy to 0%. Next we tried changing the kernel size for the noise reduction, but that had no effect. The last thing we tried was to use a different binary conversion. Upon reverting to the matlab function im2bw() from imbinarize() result in 100% match among all 6 subjects. We believe this is due to the flipping of the background/foreground color swap that better highlighted the subject's silhouette.

Now it was time to use test images of faces with hats and scarves. Since we did not have an actual data set with these items being worn naturally, we used Adobe Photoshop to place images we found online with a matching view over the subjects faces. A total of 3 subjects would have items, 2 hats and 1 scarf (subjects 1,3, and 6 respectively). When the algorithm was run, ⅝ were positively matched.

Above: A sum image built using imbinarize()

Above: Clothing accessories used for the test images.

Data

Output for normal face recognition and obstructed is located in FacialRec\project\report. Matches are highlighted with "<-". Additionally, the program may be run in matlab by running

facialrec_gei_technique.m located in FacialRec\project\code.

Thresholding To Combine Methods

Thresholding To Combine Methods

As of now we have addressed the methods we have used seperately. Our goal of the project however was to use these two methods in conjunction with each other. To do this we used thresholding to decide on which method to use. Since we end up with two different types of values it is important to be able to relate them to each other. For instance the result of Eigenfaces gives us values in the range 0-200000000. This number represents our match assurance of our test face to our training face. 0 in this case is 100% assurance where 5000000 is 80 % and quickly drops off to 40% as it gets to 15000000. We can relate these percents to our sum face where our resulting face comparisons return a percent match. Using these percentages we can safely relate both methods to each other and figure out which method we should use in each situation.

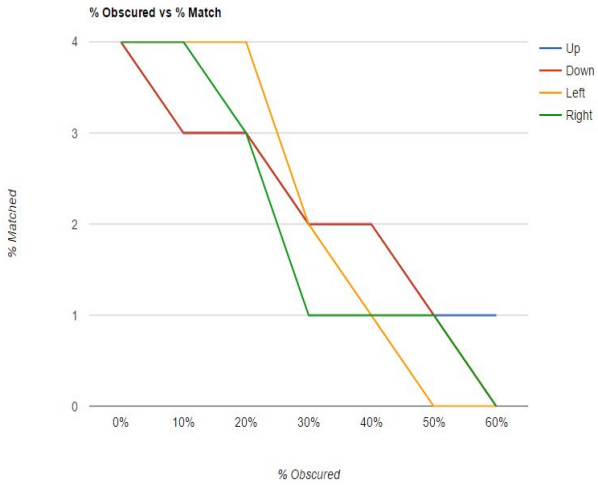
Results

It is difficult to get a good idea of how successful our system is just by the clothes the person is wearing. For instance we could say that our system is very good at detecting people correctly when they have hats on, however what does this mean? Hats vary from obscuring the hairline to obscuring down the the subjects eyes. So while we did initially do testing on people with clothes and did get results we found it difficult to convert our findings into anything meaningful. We initially did testing and found that people with light clothing we resulted in about 85% hit rate. Lighter clothing meant that the person was wearing a scarf or hat but not both. Furthermore the scarf did not cover above the mouth. Heavier clothing which included test images that put a mix of hats scarves and glasses resulted in a much lower 50% hit rate.

In an attempt to make our results more meaningful, in conjunction with our photoshopped test images we also ran our system on test images with varying levels of obscuration. The results of using Eigenfaces in conjunction with sum image processing can be seen below in the line graph. Breaking the graph down we see that we have four different lines. These lines represent the obscuring direction we tested the images in. For instance below are a series of images that show the same face progressively being covered upwards. This is represented by the “up” line on our our graph. This process was done in order to stress test the system in order to see where our system generally stops being reliable. This process was also initially expected to tell us what parts

of the face are more important. For instance if “up” returned greater hit results than “down” we could confidently say that objects that obscure the bottom half of one's face do not have much impact on our facial recognition. Unfortunately however the data we received revealed mixed results where the upper area of the head had more defining features than the bottom and vise versa depending on the face. The sample size we used in testing was small however so maybe on a macro scale wer could get answers to these questions however with the time and resources we had, efforts were better placed elsewhere. Regardless of this shortcoming we did get a general idea of the level of accuracy we have throughout the obscuring process. Working backwards we see that anything above 60% obscuration in any which direction results in almost 0% accuracy. 60% obscuration however is hardly seen in average day headwear so the system was not expected to succeed at such extremes. Between 20% and 30% is the upper limit of our system for any kind of reliable match for faces. The last picture on the right represents a 30% obstruction. It is quite obvious that this level of obstruction would be the equivalent to a high scarf.

.



Future Plans

There is a ton of tweaking that can be done in terms of which method to be selected. The thresholds we have to decide on the method of matching to use may not necessarily be the best as they are currently. As it is now we have the selection of methods to be decided by pure percentage of match. This however might not be the best course of action seeing as the different methods excel in different areas. For instance through testing we are aware that our mean image method works better for sets with faces that are offset just a bit while eigenfaces works better over a range of angles of the face. In this instance if we were to be given the situation where we were given as set of faces slightly offset we could give priority to the mean image method. Another way we could improve our system is to add more facial

recognition methods to be chosen. There are methods like Linear Discriminant Analysis or Elastic Bunch Graph Matching that could allow us to raise our hit rate even higher.

Concluding remarks

Implementing a pre-existing facial recognition system with the eigenface method was very enlightening when we compared results of the normal face test images, versus the occluded images. The same is true for implementing a method similar to a gait energy image, but instead applying the sum of images to the human face from sequence of images not from a movement cycle. Initially, this one was met with some resistance by giving extremely poor results, so we had to come up with clever ways to improve the algorithm so that it could become more and more accurate. In the end we believe both are valid methods for not only using on normal facial recognition, but they can be applied to facial recognition where the test subject's face is occluded by an environmental object or clothing.

Member contributions

Task	Antonio Castro	Aaron Jouvenat
Data set locating and research	3 hours	3 hours
Eigenface implementation	10 hours	0 hours
GEI Technique implementation	0 hours	10 hours
Data gathering and analysis	5 hours	5 hours
Report	5 hours	5 hours

References

Individual Recognition Using Gait Energy Image - Ju Han, Bir Bhanu

FEATURE SELECTION ON GAIT ENERGY IMAGE FOR HUMAN IDENTIFICATION - Khalid Bashir, Tao Xiang, Shaogang Gong

Average Gait Differential Image Based Human Recognition - Jinyan Chen, and Jiansheng Liu

https://www.scribd.com/document/76465241/Eigenfaces-Face-Recognition-MATLAB

https://jeremykun.com/2011/07/27/eigenfaces/