




Face Recognition

with OpenCV and
LBPHFaceRecognizer

“Local Binary Pattern Histogram”



Tyler Vitanova
CSC570 Final Project



OpenCV 3.x

- I utilized OpenCV 3.0. One of the most up to date major versions.
- More Optimized, by adding OpenCL acceleration to most methods.
- Restructuring of Classes for better transparency of the API.

If you are starting in OpenCV, starting in 3 is recommended by many. However, since it is so recent, very little documentation exists for it. Most documentation is for 2.4, and with the changes made that can mean significant differences. Books are just now starting to come out for OpenCV 3.0 self teaching.

Built in Recognizers (cv::face namespace)

- **BasicFaceRecognizer**
 - `createEigenFaceRecognizer` (`int` num_components=0, `double` threshold=DBL_MAX)
 - `createFisherFaceRecognizer` (`int` num_components=0, `double` threshold=DBL_MAX)
- **LBPFaceRecognizer**
 - `createLBPHFaceRecognizer` (`int` radius=1, `int` neighbors=8, `int` grid_x=8, `int` grid_y=8, `double` threshold=DBL_MAX)

BasicFaceRecognizer

`createEigenFaceRecognizer` ¹

- **Num_components** - The number of components kept for this Principal Component Analysis.
 - The recommend number of components is 80.
- **Threshold** - The threshold applied in the prediction.

The actual details of how EigenFaces work is very difficult to read and understand. At a high level, EigenFaces use the PCA to create a covariance matrix of the training data. Test images are then projected onto the most significant eigenvectors. Using a small number of basis images(features) it can add these images together, each image being some ratio of the most significant eigenvectors, and create a fairly accurate representation of the original image for prediction.³



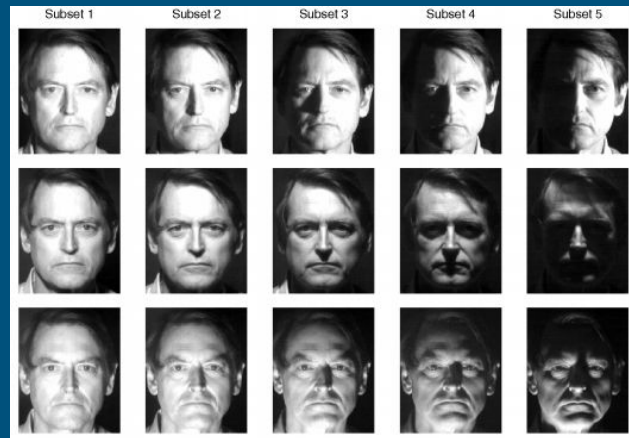
Eigenfaces with basis ratios ²

BasicFaceRecognizer

createFisherFaceRecognizer ¹

- **Num_components** - The number of components kept for this Linear Discriminant Analysis with the Fisherfaces criterion.
 - It is recommended to use all components (number of classes), by default this number is set to classes-1.
- **Threshold** - The threshold applied in the prediction. If the distance to the nearest neighbor is larger than the threshold, this method returns -1.

In general, FisherFaces performs the same techniques as EigenFaces to create a scatter matrix. The difference is comes with the LDA. EigenFaces PCA is good for idealized conditions, while FisherFaces LDA can perform well under suboptimal conditions likes poor/varying lighting on the subject.



FisherFace Test Set ⁴

LBPHFaceRecognizer

createLBPHFaceRecognizer ¹

- **Radius** - The radius used for building the circular Local Binary Pattern.
- **Neighbors** - The number of sample points to build a Circular Local Binary Pattern from. An appropriate value is to use 8 sample points. Keep in mind: the more sample points you include, the higher the computational cost.
- **Grid_x** - The number of cells in the horizontal direction, 8 is a common value used in publications. The more cells, the finer the grid, the higher the dimensionality of the resulting feature vector.
- **Grid_y** - The number of cells in the vertical direction, 8 is a common value used in publications. The more cells, the finer the grid, the higher the dimensionality of the resulting feature vector.
- **Threshold** - The threshold applied in the prediction. If the distance to the nearest neighbor is larger than the threshold, this method returns -1.

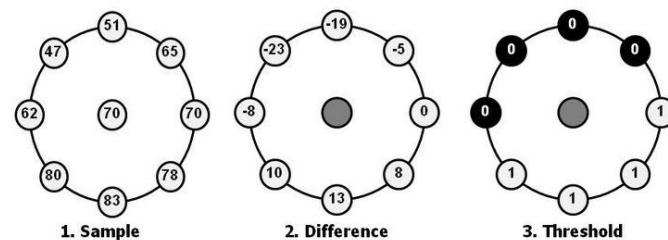
LBPHFaceRecognizer (cont.)

Local Binary Pattern uses KNN to define the value of each pixel based on its neighbors. Generally, K is recommended to be 8, for accuracy as well as speed, which creates a 3x3 matrix of neighbor pixels(see image). The value of each pixel is determined and then subtracted by the center pixel. If the difference is not negative the binary value is 1, otherwise it is 0. This byte value is then converted to an integer and assigned to the pixel as a label.

This requires images be in gray-scale for analysis.

The value of the LBP code of a pixel (x_c, y_c) is given by:

$$LBP_{P,R} = \sum_{p=0}^{P-1} s(g_p - g_c) 2^p \quad s(x) = \begin{cases} 1, & \text{if } x \geq 0; \\ 0, & \text{otherwise.} \end{cases}$$



$$1*1 + 1*2 + 1*4 + 1*8 + 0*16 + 0*32 + 0*64 + 0*128 = 15$$

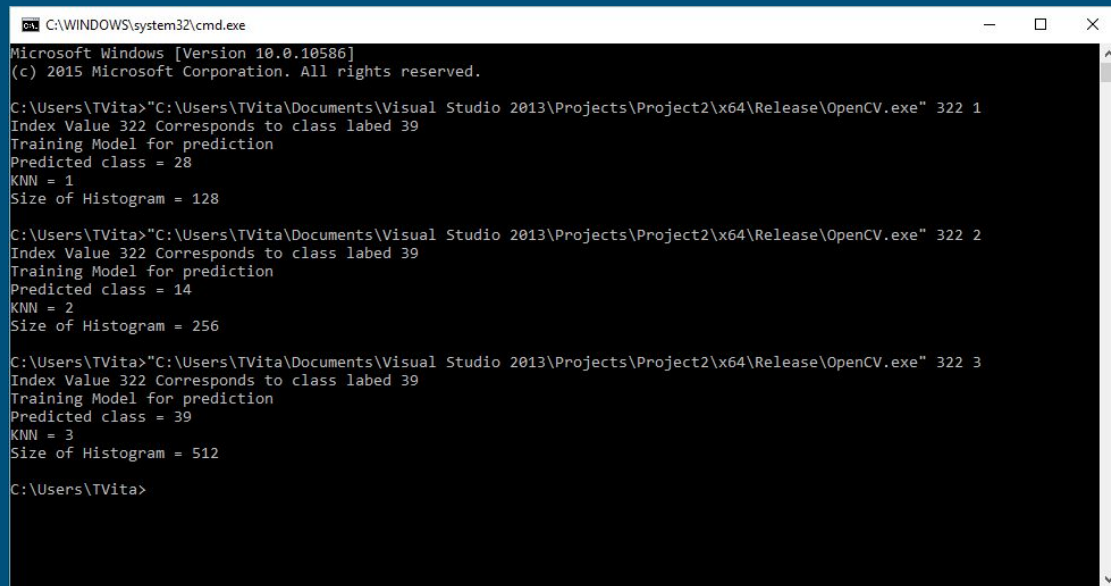
4. Multiply by powers of two and sum

LBP pixel value representation ⁵

Recognition Experiment

I tested both EigenFaces and LBPH even though I was drawn to LBPH initially. The biggest draw to LBPH(Local Binary Pattern Histogram) is that it supports updating the model as well as being incredibly fast in comparison to EigenFaces or FisherFaces. When running the program using EigenFaces with default parameters, training the model on 400 images took approximately 25 secs. With LBPH, using default parameters it takes less than 2 seconds.

The data set utilized in this experiment came from AT&T's open source face database. It contains 400 images; 10 images for each of the 40 subjects under idealized conditions. Because of this, LBP has near perfect accuracy, even with 1-NN. In my testing I found a single image belonging to label 39 was predicted as label 28 with 1-NN. As well as belonging to Label 14 with 2-NN. This misprediction was resolved only when reaching K = 3



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\TVita>"C:\Users\TVita\Documents\Visual Studio 2013\Projects\Project2\x64\Release\OpenCV.exe" 322 1
Index Value 322 Corresponds to class labeled 39
Training Model for prediction
Predicted class = 28
KNN = 1
Size of Histogram = 128

C:\Users\TVita>"C:\Users\TVita\Documents\Visual Studio 2013\Projects\Project2\x64\Release\OpenCV.exe" 322 2
Index Value 322 Corresponds to class labeled 39
Training Model for prediction
Predicted class = 14
KNN = 2
Size of Histogram = 256

C:\Users\TVita>"C:\Users\TVita\Documents\Visual Studio 2013\Projects\Project2\x64\Release\OpenCV.exe" 322 3
Index Value 322 Corresponds to class labeled 39
Training Model for prediction
Predicted class = 39
KNN = 3
Size of Histogram = 512

C:\Users\TVita>
```

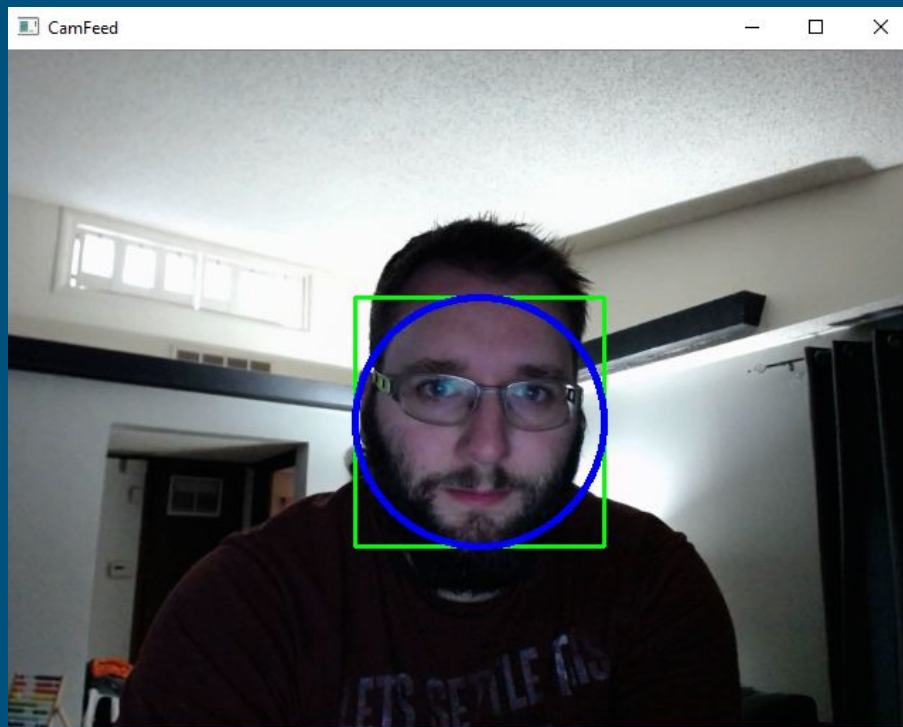

Stress Testing K-NN and LBPH

When 8-NN ran so quickly and since the K-NN values as little as 3 seemed to provide 100% accuracy on the AT&T face database, I wanted to stress test the algorithm and my machine. My question was, at what point would K-NN generate a histogram so large that my machine would cap its resources? I found that 18-NN generated a Histogram of size 16,777,216 and took roughly 45 seconds. When attempting K = 25, my CPU usage reached 25% (across 16 cores), and consumed all 96GB of memory. After 3 minutes I killed the execution.

```
C:\Users\TVita>"C:\Users\TVita\Documents\Visual Studio 2013\Projects\Project2\x64\Release\OpenCV.exe" 322 18
Index Value 322 Corresponds to class labeled 39
Training Model for prediction
Predicted class = 39
KNN = 18
Size of Histogram = 16777216
```

Future Improvements

Having built the core recognition model on an algorithm that supports model updating, future improvements would include real-time face detection and auto-training on new, unknown faces.



Citations

1. Anon, (2016). *Face recognition using Eigenfaces*. [online] Available at: <http://users.utcluj.ro/~tmarita/IOC/C7/C7.pdf> [Accessed 4 May 2016].
2. *Face recognition using Eigenfaces*. [online] Available at: <http://users.utcluj.ro/~tmarita/IOC/C7/C7.pdf> [Accessed 4 May 2016].
3. Dailey, M. (2016). *Matlab Tutorial*. [online] Cs.ait.ac.th. Available at: <http://www.cs.ait.ac.th/~mdailey/matlab/> [Accessed 6 May 2016].
4. Belhumeur, P., Hespanha, J. and Kriegman, D. (1997). *Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection*. [online] Available at: <http://www.cs.columbia.edu/~belhumeur/journal/facepami97.pdf> [Accessed 6 May 2016].
5. Pietik, M. (2010). Local Binary Patterns. *Scholarpedia*, 5(3), p.9775.

Bibliography

Docs.opencv.org. (2016). *OpenCV: cv::face::LBPHFaceRecognizer Class Reference*. [online] Available at: http://docs.opencv.org/master/df/d25/classcv_1_1face_1_1LBPHFaceRecognizer.html#gsc.tab=0 [Accessed 1 May 2016].

Chao, H. (2010). *Eigenfaces and Fisherfaces*. [online] Available at: <http://disp.ee.ntu.edu.tw/~pujols/Eigenfaces%20and%20Fisherfaces.pdf> [Accessed 5 May 2016].