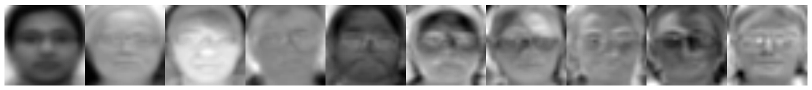
Jesse Weisberg
CSE 559a Computer Vision
Furukawa

Project 3: Eigenfaces

Testing recognition with cropped class images

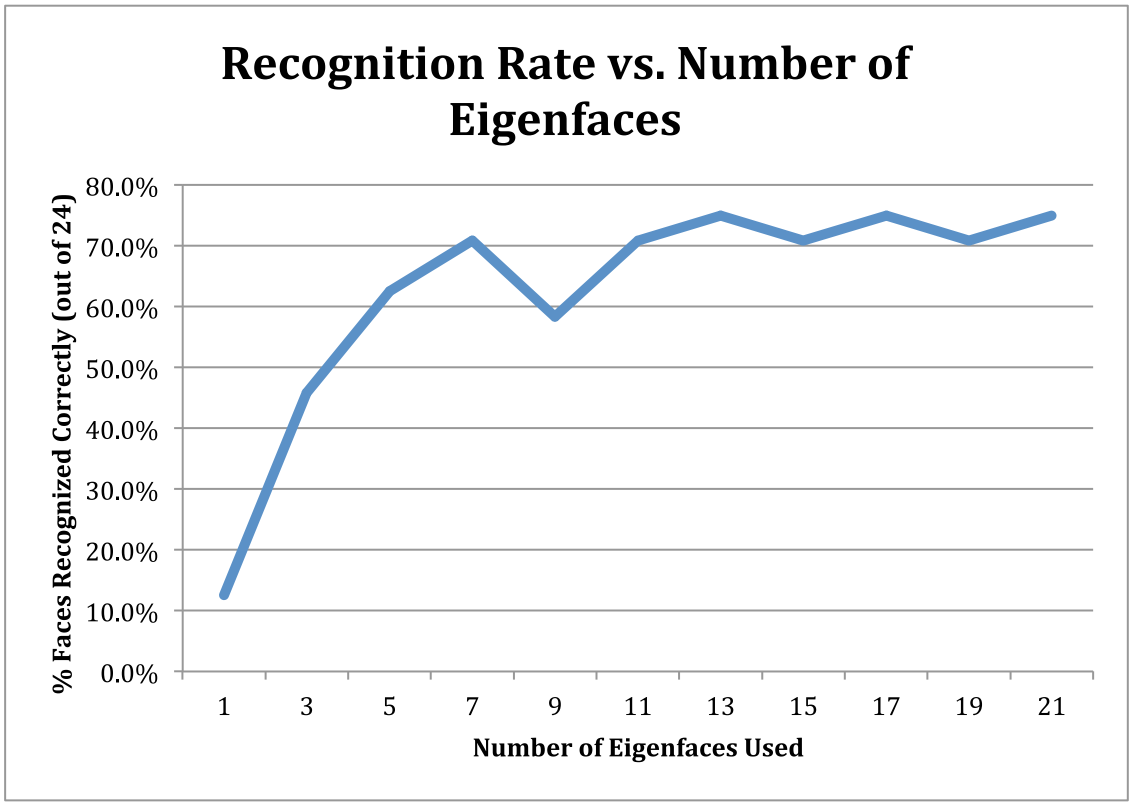
Average Face:

Eigenfaces:



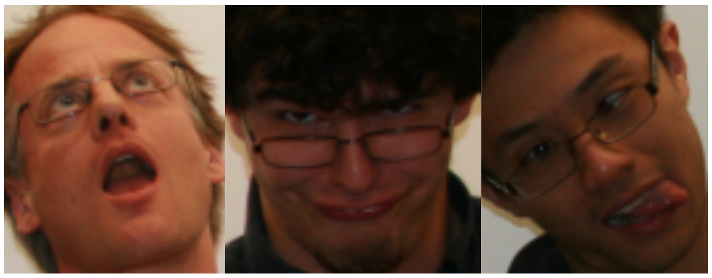
Projection/Reconstruction:

Recognition Analysis:



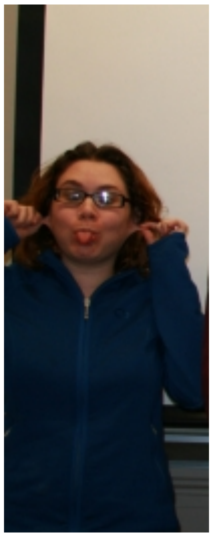
From the graph, it seems like the performance of the recognition algorithm plateaus at around 13 eigenfaces. Thus, it seems like it really only takes about 13 eigenfaces to really span the set of faces, and any more added on are practically extraneous. The tradeoffs are faster computation for less eigenfaces used, but higher recognition rate for more eigenfaces (up until the plateau region, or 13 eigenfaces). For the greatest results with the least time for computation, using around 13 eigenfaces seems like the best option from these results.

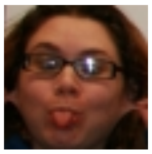
Some faces that gave the recognition algorithm trouble were “interesting” 2, 4, and 15. This is probably due to the extreme variance in rotation, light, or contorting in these faces compared to their normal counterparts. 2, 4, and 15 are shown below in that order, and these qualities can clearly be seen in these photos. 2 and 15 are each ranked in the top 3 candidates, so they are close to being correct, but 4 is not very close (ranking 7th in most cases). This is probably due to the extreme shading compared to the original, movement of glasses, rotation of face, and contortion of face.



Cropping and finding faces

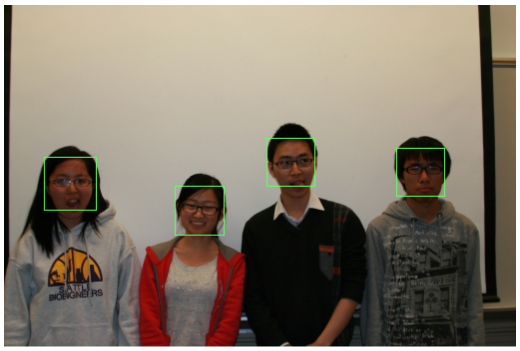
Result for test_single.tga:

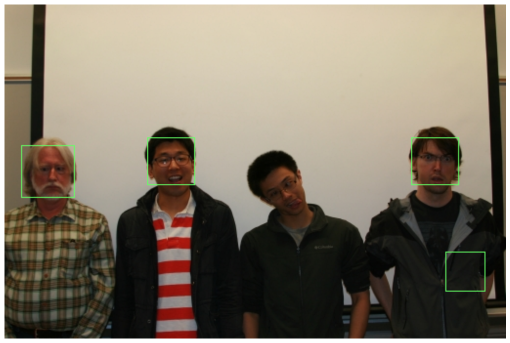


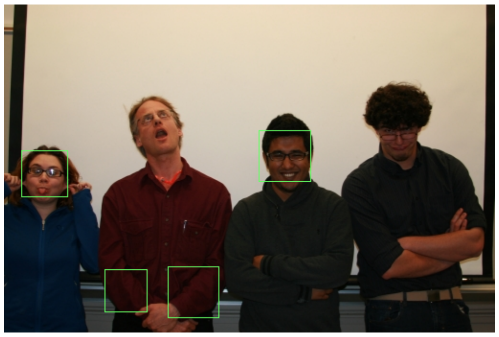


Thus the findface function works as expected in this example.

Results for marking group07, group10, and group 02







The parameters used for these photos were the ones given (for cropping, .35 .55 .05, and for marking, .3 .6 .05). These seemed close to the most efficient parameters. I tried decreasing the step size to .01 to see if that could potentially make the algorithm more accurate, but it did not.

When the faces hold relatively neutral expressions and face the camera directly, the findface function is much more accurate (though the latter seems to have a more adverse effect on results).

I observe that while we detect some faces correctly, many false positives occur in this implementation. This might be due to using the suggested formula for error:

mse * (distance of face from average) / (variance of face)

I believe this may cause areas with significant variance to be incorrectly detected. However, I also observe false positives in low texture regions. I think that the remaining errors are due to the small training set, variations in pose and illumination, and scale parameters that do not precisely coincide with the faces in the group picture.

Verify Face

To answer this question, I wrote a script, “verifyScript.py,” that iterated through the verifying process of each interesting face with each neutral face in the userbase. Then, I calculated the false positive (fp) and false negative (fn) rates for a number of thresholds (for details see source). At an mse threshold of 50000, fp = 4.51%, fn = 1.04%. At an mse threshold of 48000, fp = 3.82%, fn= 1.39%. So, around this value (48000-50000) gave me optimal results, but it just becomes a question of whether you prefer false positives or false negatives, because as you decrease the threshold you reach more false negatives, but as you increase the threshold you get more false positives.