Final Project Report

Project Option 2: Advanced face detection with Adaboost

Process and Discussion of Choices:
Deciding to go with the Adaboost project, our group understood that we had to design an algorithm that essentially built on itself and improved through training. Our group was able to create the adaboost detector by first loading in the training face and nonface images as a 3D matrix for storage efficiency. Our group decided to use 1000 face images and 100 nonface images. We believed that this would be a good amount for both faces since we're not using all of them but we're not using too little as we need a good number of images to properly train the algorithm. Non face images were cropped into the same size of face images for efficiency. Along with the faces themselves, the face and nonface integrals were stored in a 3D matrix as well. Once this training data was processed we generated 500 weak classifiers since we believed this would be an effective amount to train the images. We then processed the responses and labels then fed it into the Adaboost function to create our initial boosted classifier. We decided to display the best 50 results to get a good understanding of how the classifiers are acting. The table for this initial boosted classifier is in the results paragraph below. To create an even more effective detector, we incorporated bootstrapping. To incorporate bootstrapping we tested each image in the training faces and training nonfaces folder with our initial boosted classifier then reprocessed the images that were incorrect back into our training data. We did this process multiple times to create an even stronger detector. Each bootstrapping cycle we removed 50 faces and 10 nonfaces and put in 50 new incorrectly identified faces and 10 new incorrectly identified nonfaces. We chose these amounts because they're in the 5-10% range as we don't want to be replacing a lot of training images at once since it could negatively affect the detector. We didn't get to the cascading integration for the detector, but ideally if we did we would have created a cascade_classify function which recieved a window and compared a sequence of classifiers to it. In a cascade of classifiers, the first classifier is slow and not accurate, but as the sequence goes on the following classifier is slower and more accurate than the one before. So in an ideal implementation, the function would loop through each classifier in the cascade sequence and compare it with a threshold to either reject the face or continue to the next classifier. The image would not be identified as a face unless all the classifiers in the sequence did not reject it and the last one actually classifies it as a face.
We did not get to the skin detection portion of the project either but an ideal implementation would look similar to the skin_integral_example code. What would be required for this is to load the negative and positive skin detection histograms then feed it into the detect skin function with the image being detected. An appropriate threshold would be necessary that can apply to many images. Scaling then calculating the integral image would be necessary as well. The skin that is detected could ideally be applied both to the detector and training part. When applying it to the training part it would be used by assisting the algorithm in understanding that the faces it is training have skin and to look out for that so that it can better detect faces. An easier method would be to just apply the skin detection on the image that the face detector is using and only run the detector on the skin pixels that are identified.

Results and Analysis:
Below are the results of the initial and final call of the Adaboost function that creates the initial and final version of the boosted classifier, which is essentially made of many weak classifiers. The first column is

the numbered classifier and the second column is the error rate.

Initial Boosted Classifier

```
 1.0000    0.0345    0.0345  229.0000   -1.6652  187.6942

 1.0e+03 *

 0.0020    0.0000    0.0001    0.2290    0.0009   -1.5242

 3.0000    0.0482    0.1607   46.0000   -0.8266  142.1476

 4.0000    0.0200    0.1620    1.0000    0.8216  538.9800

 5.0000    0.0236    0.1399  449.0000   -0.9079   56.0580

 6.0000    0.0200    0.2045  182.0000   -0.6792  160.1950

 7.0000    0.0091    0.1827    1.0000    0.7491  538.9800

 8.0000    0.0064    0.1729   71.0000    0.7827   -8.8390

 9.0000    0.0091    0.2238  466.0000    0.6219  -39.9256

10.0000    0.0036    0.1884  284.0000   -0.7301   64.5356

11.0000         0    0.1219    1.0000    0.9871  538.9800

12.0000         0    0.1386  416.0000   -0.9133  143.7882

13.0000         0    0.2131   66.0000    0.6532 -336.8518

14.0000         0    0.1844    1.0000    0.7434  538.9800

15.0000         0    0.1687  175.0000    0.7973  -93.3508

16.0000         0    0.2178  139.0000   -0.6391   61.7150

17.0000         0    0.1923    1.0000    0.7177  538.9800

18.0000         0    0.1582  148.0000    0.8358 -132.7236

 1.0e+03 *

 0.0190         0    0.0002    0.3480    0.0006   -1.2432

20.0000         0    0.1944    1.0000    0.7110  538.9800

21.0000         0    0.1512  379.0000    0.8624  -35.4340

22.0000         0    0.2196  296.0000   -0.6340  602.8904

23.0000         0    0.1913    1.0000    0.7208  538.9800

24.0000         0    0.1619  366.0000    0.8222 -213.2842

25.0000         0    0.2367  241.0000   -0.5855   75.6584

26.0000         0    0.1954    1.0000    0.7077  538.9800

27.0000         0    0.1533  142.0000    0.8545  -10.7480

28.0000         0    0.2263  250.0000   -0.6148    1.9508

29.0000         0    0.1930    1.0000    0.7152  538.9800

30.0000         0    0.1696  400.0000    0.7944  -91.5800

31.0000         0    0.1940  177.0000    0.7121  -47.1768

32.0000         0    0.1868    1.0000    0.7356  538.9800

 1.0e+03 *

 0.0330         0    0.0001    0.1120    0.0009   -1.2560

 1.0e+03 *

 0.0340         0    0.0002    0.0780   -0.0008    1.0763
```

35.0000 0 0.1760 1.0000 0.7719 538.9800

1.0e+03 *

0.0360 0 0.0002 0.0570 -0.0008 2.3353

37.0000 0 0.2107 303.0000 0.6604 -32.2820

38.0000 0 0.1885 1.0000 0.7299 538.9800

39.0000 0 0.1681 435.0000 -0.7997 289.8428

40.0000 0 0.2089 22.0000 0.6657 -2.6850

41.0000 0 0.1930 1.0000 0.7154 538.9800

42.0000 0 0.1691 327.0000 0.7958 -480.4812

43.0000 0 0.2104 434.0000 -0.6614 21.4580

44.0000 0 0.1982 1.0000 0.6989 538.9800

1.0e+03 *

0.0450 0 0.0002 0.0780 -0.0009 1.0763

1.0e+03 *

0.0460 0 0.0002 0.0060 -0.0007 1.0601

47.0000 0 0.1847 1.0000 0.7423 538.9800

1.0e+03 *

0.0480 0 0.0002 0.4280 -0.0008 1.9933

49.0000 0 0.1953 284.0000 -0.7081 64.5356

50.0000 0 0.1991 1.0000 0.6958 538.9800

Boosted Classifier after Bootstrapping

1.0e+03 *

0.0010 0.0000 0.0000 0.0720 -0.0016 1.1635

2.0000 0.0364 0.1455 295.0000 -0.8851 764.6108

3.0000 0.0364 0.1950 288.0000 0.7089 -659.6920

1.0e+04 *

0.0004 0.0000 0.0000 0.0243 -0.0001 -1.8373

5.0000 0.0282 0.1771 416.0000 -0.7681 332.1328

6.0000 0.0245 0.2223 484.0000 0.6262 -167.4312

1.0e+03 *

0.0070 0.0000 0.0002 0.0620 0.0006 2.3887

8.0000 0.0082 0.1811 292.0000 -0.7544 513.0868

9.0000 0.0191 0.2383 250.0000 -0.5810 56.9186

1.0e+04 *

0.0010 0.0000 0.0000 0.0073 -0.0001 -1.4495

11.0000 0.0064 0.1960 466.0000 0.7058 -39.9256

12.0000 0.0027 0.2243 83.0000 0.6203 -233.3588

1.0e+04 *

0.0013 0.0000 0.0000 0.0050 -0.0001 -1.3170

14.0000 0 0.1688 148.0000 0.7971 -132.7236

15.0000 0 0.2192 63.0000 -0.6353 484.8932

1.0e+03 *

0.0160 0 0.0002 0.4780 0.0007 5.0018

17.0000 0 0.1818 106.0000 0.7520 -377.4256

18.0000 0 0.2573 296.0000 -0.5301 602.8904

1.0e+03 *

0.0190 0 0.0002 0.1130 -0.0006 -2.6930

1.0e+03 *

0.0200 0 0.0002 0.1120 0.0008 -1.2560

1.0e+03 *

0.0210 0 0.0003 0.0510 -0.0005 3.2290

1.0e+04 *

0.0022 0 0.0000 0.0050 -0.0001 -1.3170

23.0000 0 0.1792 204.0000 -0.7608 64.0424

24.0000 0 0.1836 481.0000 0.7461 -4.6318

1.0e+03 *

0.0250 0 0.0002 0.1510 0.0007 1.3254

26.0000 0 0.1940 99.0000 0.7120 -113.7508

27.0000 0 0.2572 40.0000 -0.5303 38.5350

1.0e+03 *

0.0280 0 0.0002 0.1860 -0.0006 -3.8008

29.0000 0 0.2167 117.0000 -0.6426 116.6620

30.0000 0 0.2275 168.0000 0.6112 -26.5286

1.0e+03 *

0.0310 0 0.0002 0.4740 0.0007 8.8142

32.0000 0 0.2066 148.0000 0.6727 -132.7236

1.0e+03 *

0.0330 0 0.0002 0.3480 0.0006 -1.2432

1.0e+03 *

0.0340 0 0.0002 0.1130 -0.0007 -2.6930

35.0000 0 0.1858 132.0000 -0.7386 13.7260

36.0000 0 0.2186 177.0000 0.6368 -47.1768

1.0e+03 *

0.0370 0 0.0002 0.0420 -0.0007 -1.3985

38.0000 0 0.1894 449.0000 -0.7270 228.8160

39.0000 0 0.2688 141.0000 -0.5004 380.0822

1.0e+03 *

0.0400 0 0.0002 0.1510 0.0007 1.3254

1.0e+03 *

```
  0.0410        0    0.0002    0.1660    0.0007   -1.2021

 42.0000        0    0.2098  385.0000   -0.6631   56.0704

1.0e+03 *

  0.0430        0    0.0002    0.2500   -0.0006   -1.1867

 44.0000        0    0.2158  342.0000   -0.6450    7.3370

 45.0000        0    0.2249   66.0000    0.6187 -336.8518

1.0e+05 *

  0.0005        0    0.0000    0.0007   -0.0000   -1.2262

 47.0000        0    0.2046  470.0000   -0.6788    5.0784

 48.0000        0    0.2603  181.0000   -0.5223   36.4832

1.0e+03 *

  0.0490        0    0.0002    0.3970   -0.0007   -1.1623

 50.0000        0    0.2168  447.0000   -0.6423   75.7046
```

The changes in the classifiers show an improvement that was brought by the bootstrapping. When the detector makes a mistake and that mistake is brought back for the algorithm to notice, the algorithm is essentially fixing it's mistake meaning it is bound to gradually improve in its detection and understanding. When the algorithm was tested on the actual test images, the results were produced below.

```
Elapsed time testing images is 1.241058 seconds.

Number of correct =
   766

Number of false negatives =
    40

Number of false positives =
     0

Accuracy =

    0.9504

Elapsed time detecting face  is 0.006496 seconds.
```

The cropped testing images and nonface images were processed and tested for accuracy. The accuracy was measured by dividing the correctly classified images with the total amount of test images that were tested. With the accuracy that was produced our group agrees that the skin detector completes its job but could be even more effective had we completed the skin detection and cascading portions of the project. The skin detection would undoubtedly decrease the number of false negatives since it would be able to recognize that a person is at least present since there is skin present and not be so quick to identify a face as not being there when it actually is. Cascading would help the efficiency as well, but more importantly would be able to quicken the process even more. The bootstrapping that we implemented helped the algorithm by introducing an aspect to the training that gives it a better grasp on classification which

helped it with detecting the faces.