18326577 - Róisín Ní Bhriain

Draughts Assignment

Part One - Classifying Pixels

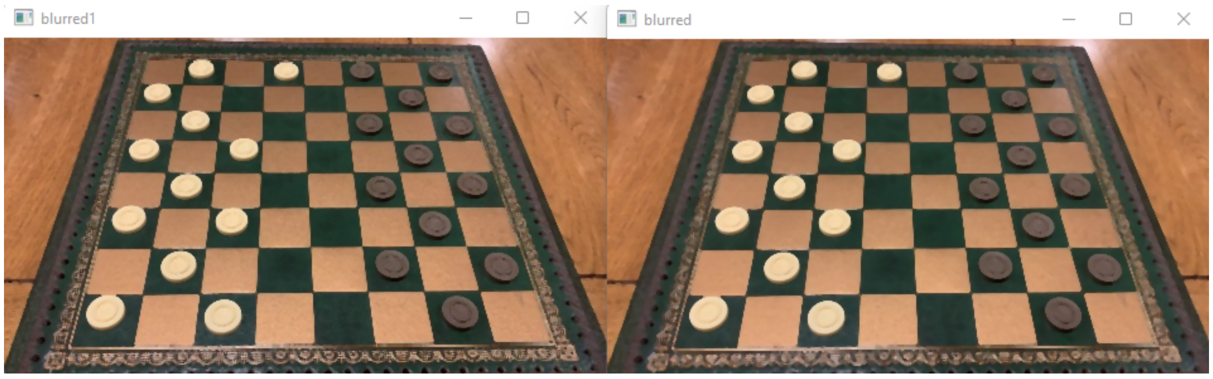


Figure 1: The blurred images from left to right after median blurring of 3 then a second median blurring of 3.

I originally tried to do gaussian blurring but this did not work as well as median blurring. I found that doing a single round of median blurring did not blur the noise enough and using too high of a median value was blurring the image too much and warping some of the squares.

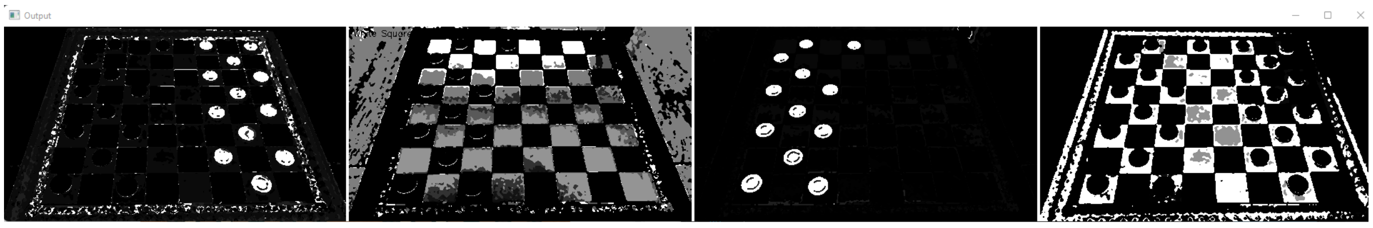


Figure 2: Backprojection Results in order (left to right): Black pieces, white squares, white pieces, black squares.

The backprojection with the blurring worked well. The results of the backprojection are a bit varied for the different samples. The white pieces are very clear and there doesn't seem to be too many extra pixels that are classified as white pieces. The white squares are not as good presumably due to the lighting.

This can be rectified with adaptive thresholding. I used OTSU thresholding on the two kinds of pieces. The back projection worked quite well with picking up the two types of squares. I did an otsu threshold on the pieces as this is an adaptive threshold and it would hold up in different lighting conditions. I also did a close and a dilate on the pieces as this closed a lot of the holes in the binary image.

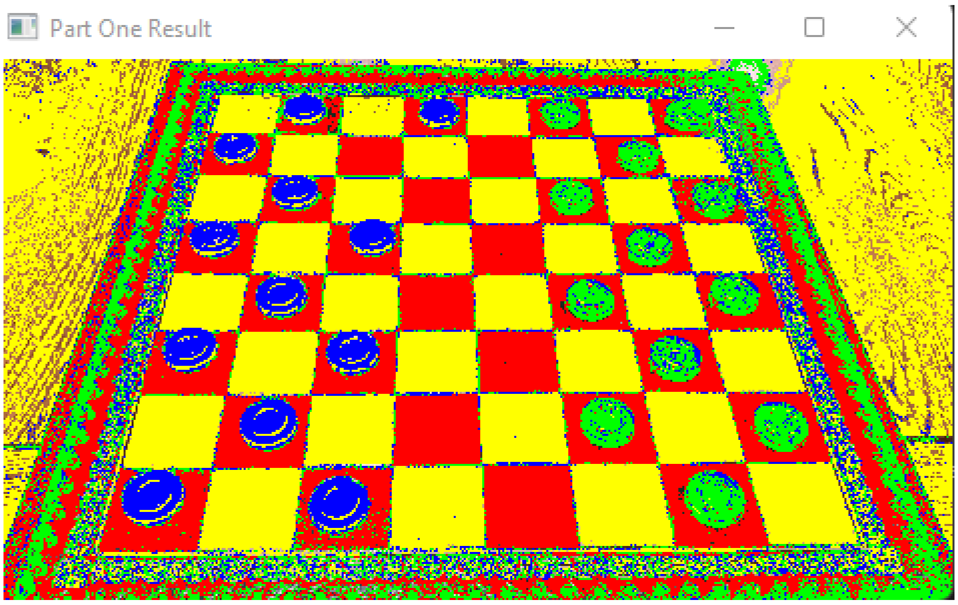


Figure 3: The first image with just backprojection.

Figure 3 shows the image with back projection but without the blurring. As you can see there's a good bit of noise. But it is clear where the squares are and where the draughts pieces are.

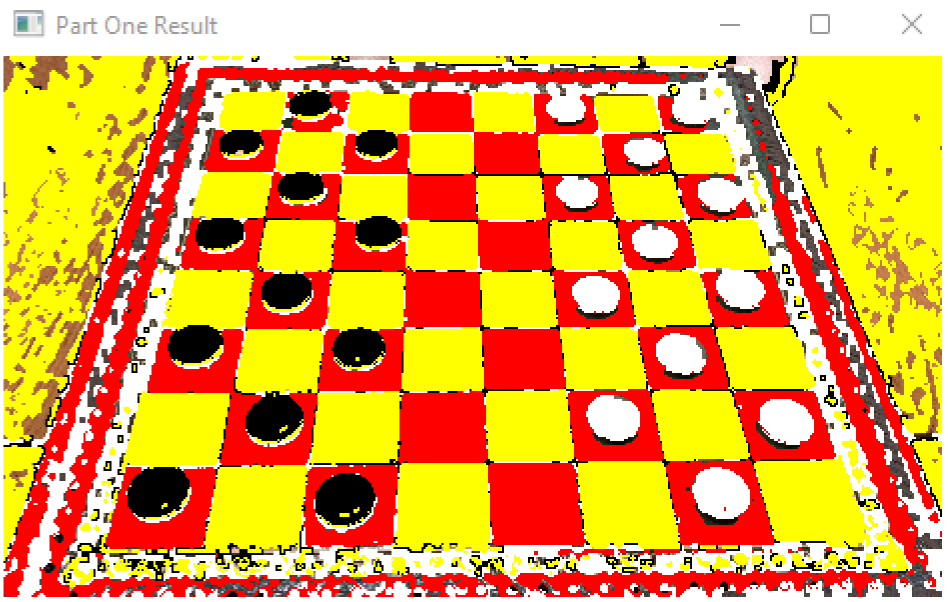


Figure 4: The image with median smoothing, backprojection and some adaptive thresholding with a close.

The black pieces are coloured white, the white pieces are coloured black, the black squares are coloured red and the white squares are coloured yellow. When you compare figure 4 and figure 3 you can see that there's a big improvement. There's a lot less noise and the squares are more accurate. There are still a few gaps in the pieces which I attempted to fix using a closing operation on the adaptive thresholded images but this didn't work perfectly 100% of the time.

Robustness
I found that the technique worked well on some of the static images but not every single one. It worked well on the first few images but some of the later images were not identified as well. I think the backprojection is not a very robust method as the sample images will need to change based on the lighting.

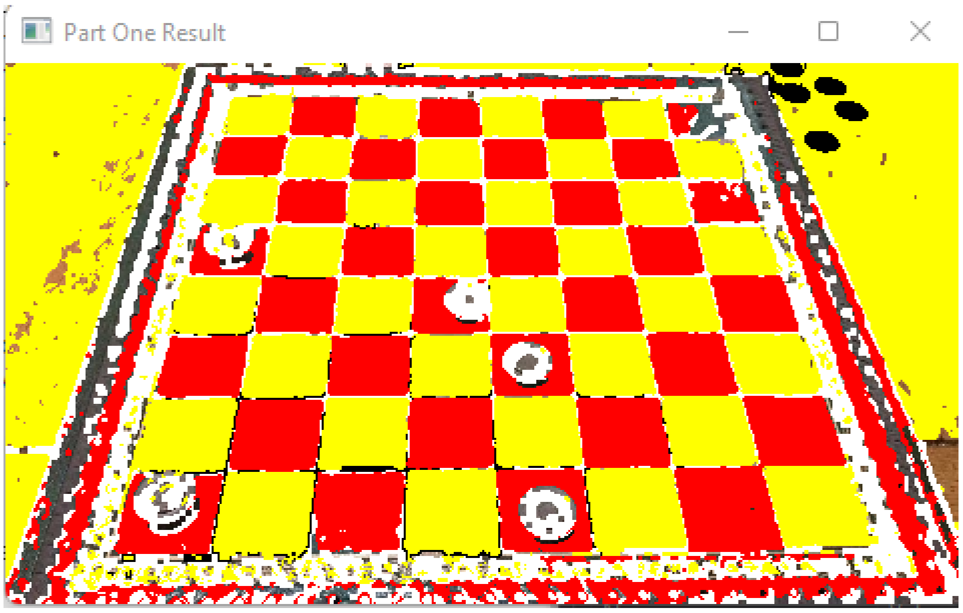


Figure 5: DraughtsGame1Move68.JPG after part one pixel colouring.

As you can see here the black pieces aren't picked up as well as they were in the original. I think this may be due to lighting changes. To improve this Adaptive thresholding is the way to go as this copes well with lighting changes. There are also a few shadows in the image particularly at the foreground of the image. Using KMeans may also help before doing the back projection, I attempted this but the accuracy was not as good as I wanted it to be so I decided to do without.

Part Two - Accuracy

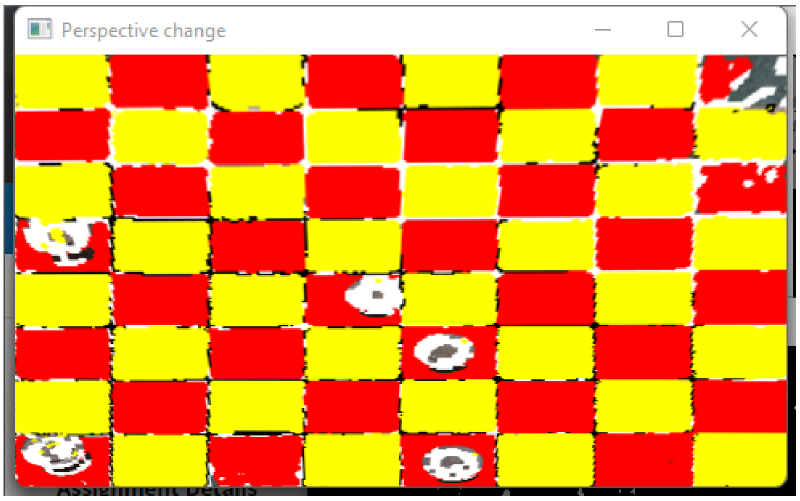
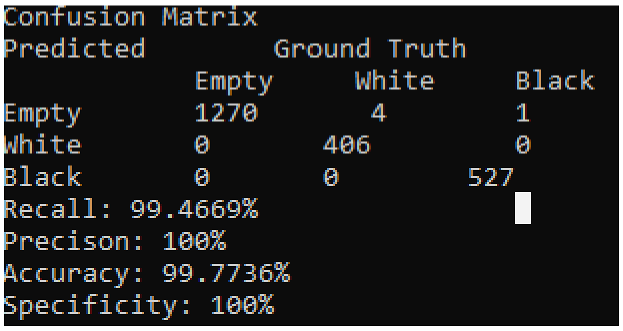


Figure 6: Draughts board after using a perspective change with the four points given.

My first step in part two was to do a perspective change on the board using the four corners that were given in the assignment. I did this perspective change on the image that was returned from part one. I then sectioned off each section using the rect function by dividing the board into rows/8 and columns/8.

For each square I checked if it was a black square. Then I checked whether there were more black piece pixels or white piece pixels. I would return either black or white or an empty square over a certain threshold. Then I would compare the predicted vs the ground truth. I returned that it would be a piece of its colour no if it was a King or just a Man on a square.



```
Confusion Matrix
Predicted        Ground Truth
            Empty      White      Black
Empty       1270         4          1
White       0         406          0
Black       0         0         527
Recall: 99.4669%
Precison: 100%
Accuracy: 99.7736%
Specificity: 100%
```

Figure 7: Confusion matrix from part two.

Recall: ((406 + 527) / (406 + 527 + 4 + 1)) * 100 = 99.4669%

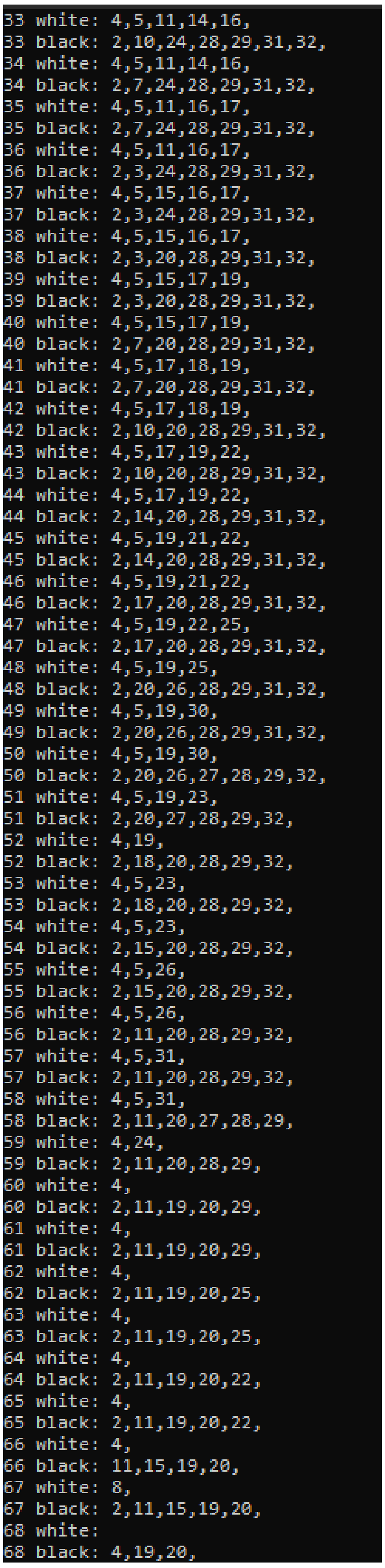
Precision: ((406 + 527) / (406 + 527)) * 100 = 100%

Accuracy: ((1270 + 406 + 527) / (406 + 527 + 1270 + 4 + 1)) * 100 = 99.7736%

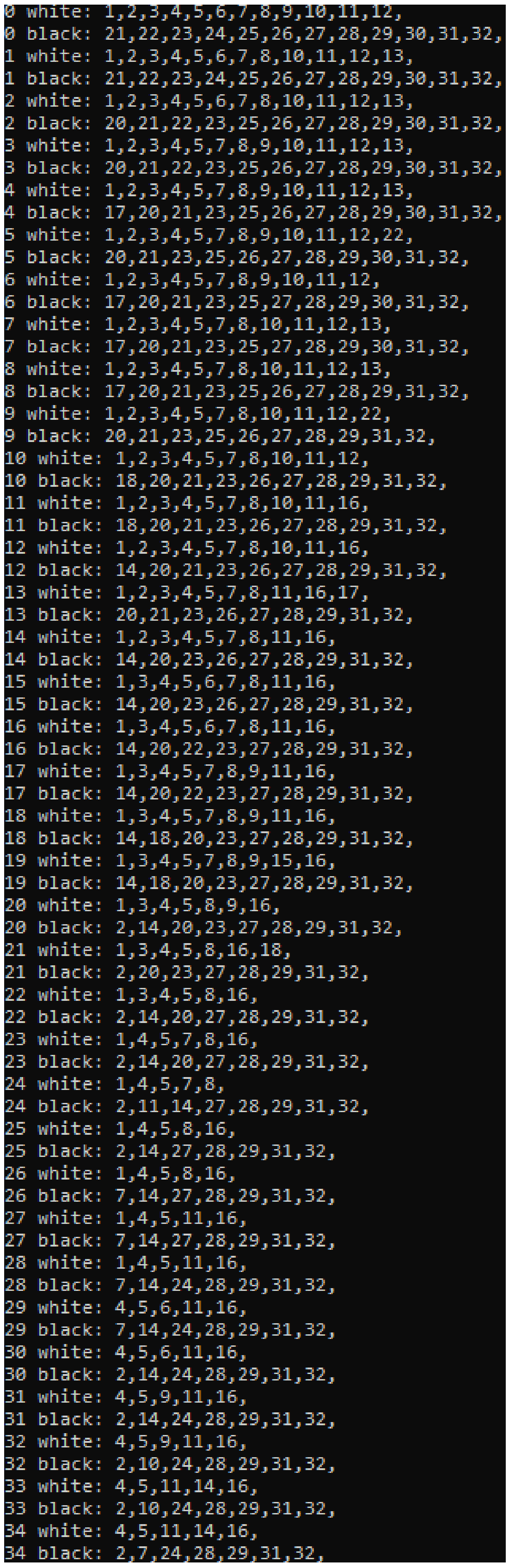
Specificity: (1270) / (1270 + 0 + 0) = 100%

These calculations were done based on the confusion matrix shown above in figure 7. The method I used is very close to perfect. The problem with this is that it relies very heavily on the pixels being identified in the first part. If the lighting changed or the pieces were smaller there is a chance that they would not be identified correctly. However, the good news is that the method I used does distinguish quite clearly between the white and black pieces. There are no pieces being mixed up, in fact the only error comes from predicting empty pieces where there should be either white or black pieces.

The method I used is not as robust as I'd want it to be as it relies heavily on the analysis from part one. The other problem that there is that it is quite specific to the number of pixels in my system. If the board was different and the squares were different or the pieces were smaller then the number of pixels per square would be different and the hard coded values that I have used as thresholds would have to change.



```
33 white: 4,5,11,14,16,
33 black: 2,10,24,28,29,31,32,
34 white: 4,5,11,14,16,
34 black: 2,7,24,28,29,31,32,
35 white: 4,5,11,16,17,
35 black: 2,7,24,28,29,31,32,
36 white: 4,5,11,16,17,
36 black: 2,3,24,28,29,31,32,
37 white: 4,5,15,16,17,
37 black: 2,3,24,28,29,31,32,
38 white: 4,5,15,16,17,
38 black: 2,3,20,28,29,31,32,
39 white: 4,5,15,17,19,
39 black: 2,3,20,28,29,31,32,
40 white: 4,5,15,17,19,
40 black: 2,7,20,28,29,31,32,
41 white: 4,5,17,18,19,
41 black: 2,7,20,28,29,31,32,
42 white: 4,5,17,18,19,
42 black: 2,10,20,28,29,31,32,
43 white: 4,5,17,19,22,
43 black: 2,10,20,28,29,31,32,
44 white: 4,5,17,19,22,
44 black: 2,14,20,28,29,31,32,
45 white: 4,5,19,21,22,
45 black: 2,14,20,28,29,31,32,
46 white: 4,5,19,21,22,
46 black: 2,17,20,28,29,31,32,
47 white: 4,5,19,22,25,
47 black: 2,17,20,28,29,31,32,
48 white: 4,5,19,25,
48 black: 2,20,26,28,29,31,32,
49 white: 4,5,19,30,
49 black: 2,20,26,28,29,31,32,
50 white: 4,5,19,30,
50 black: 2,20,26,27,28,29,32,
51 white: 4,5,19,23,
51 black: 2,20,27,28,29,32,
52 white: 4,19,
52 black: 2,18,20,28,29,32,
53 white: 4,5,23,
53 black: 2,18,20,28,29,32,
54 white: 4,5,23,
54 black: 2,15,20,28,29,32,
55 white: 4,5,26,
55 black: 2,15,20,28,29,32,
56 white: 4,5,26,
56 black: 2,11,20,28,29,32,
57 white: 4,5,31,
57 black: 2,11,20,28,29,32,
58 white: 4,5,31,
58 black: 2,11,20,27,28,29,
59 white: 4,24,
59 black: 2,11,20,28,29,
60 white: 4,
60 black: 2,11,19,20,29,
61 white: 4,
61 black: 2,11,19,20,29,
62 white: 4,
62 black: 2,11,19,20,25,
63 white: 4,
63 black: 2,11,19,20,25,
64 white: 4,
64 black: 2,11,19,20,22,
65 white: 4,
65 black: 2,11,19,20,22,
66 white: 4,
66 black: 11,15,19,20,
67 white: 8,
67 black: 2,11,15,19,20,
68 white:
68 black: 4,19,20,
```



```
0 white: 1,2,3,4,5,6,7,8,9,10,11,12,
0 black: 21,22,23,24,25,26,27,28,29,30,31,32,
1 white: 1,2,3,4,5,6,7,8,10,11,12,13,
1 black: 21,22,23,24,25,26,27,28,29,30,31,32,
2 white: 1,2,3,4,5,6,7,8,10,11,12,13,
2 black: 20,21,22,23,25,26,27,28,29,30,31,32,
3 white: 1,2,3,4,5,7,8,9,10,11,12,13,
3 black: 20,21,22,23,25,26,27,28,29,30,31,32,
4 white: 1,2,3,4,5,7,8,9,10,11,12,13,
4 black: 17,20,21,23,25,26,27,28,29,30,31,32,
5 white: 1,2,3,4,5,7,8,9,10,11,12,22,
5 black: 20,21,23,25,26,27,28,29,30,31,32,
6 white: 1,2,3,4,5,7,8,9,10,11,12,
6 black: 17,20,21,23,25,27,28,29,30,31,32,
7 white: 1,2,3,4,5,7,8,10,11,12,13,
7 black: 17,20,21,23,25,27,28,29,30,31,32,
8 white: 1,2,3,4,5,7,8,10,11,12,13,
8 black: 17,20,21,23,25,26,27,28,29,31,32,
9 white: 1,2,3,4,5,7,8,10,11,12,22,
9 black: 20,21,23,25,26,27,28,29,31,32,
10 white: 1,2,3,4,5,7,8,10,11,12,
10 black: 18,20,21,23,26,27,28,29,31,32,
11 white: 1,2,3,4,5,7,8,10,11,16,
11 black: 18,20,21,23,26,27,28,29,31,32,
12 white: 1,2,3,4,5,7,8,10,11,16,
12 black: 14,20,21,23,26,27,28,29,31,32,
13 white: 1,2,3,4,5,7,8,11,16,17,
13 black: 20,21,23,26,27,28,29,31,32,
14 white: 1,2,3,4,5,7,8,11,16,
14 black: 14,20,23,26,27,28,29,31,32,
15 white: 1,3,4,5,6,7,8,11,16,
15 black: 14,20,23,26,27,28,29,31,32,
16 white: 1,3,4,5,6,7,8,11,16,
16 black: 14,20,22,23,27,28,29,31,32,
17 white: 1,3,4,5,7,8,9,11,16,
17 black: 14,20,22,23,27,28,29,31,32,
18 white: 1,3,4,5,7,8,9,11,16,
18 black: 14,18,20,23,27,28,29,31,32,
19 white: 1,3,4,5,7,8,9,15,16,
19 black: 14,18,20,23,27,28,29,31,32,
20 white: 1,3,4,5,8,9,16,
20 black: 2,14,20,23,27,28,29,31,32,
21 white: 1,3,4,5,8,16,18,
21 black: 2,20,23,27,28,29,31,32,
22 white: 1,3,4,5,8,16,
22 black: 2,14,20,27,28,29,31,32,
23 white: 1,4,5,7,8,16,
23 black: 2,14,20,27,28,29,31,32,
24 white: 1,4,5,7,8,
24 black: 2,11,14,27,28,29,31,32,
25 white: 1,4,5,8,16,
25 black: 2,14,27,28,29,31,32,
26 white: 1,4,5,8,16,
26 black: 7,14,27,28,29,31,32,
27 white: 1,4,5,11,16,
27 black: 7,14,27,28,29,31,32,
28 white: 1,4,5,11,16,
28 black: 7,14,24,28,29,31,32,
29 white: 4,5,6,11,16,
29 black: 7,14,24,28,29,31,32,
30 white: 4,5,6,11,16,
30 black: 2,14,24,28,29,31,32,
31 white: 4,5,9,11,16,
31 black: 2,14,24,28,29,31,32,
32 white: 4,5,9,11,16,
32 black: 2,10,24,28,29,31,32,
33 white: 4,5,11,14,16,
33 black: 2,10,24,28,29,31,32,
34 white: 4,5,11,14,16,
34 black: 2,7,24,28,29,31,32,
```

Figure 8: The locations of all the pieces in part two.

This figure shows where the black pieces and the white pieces are in each static image.

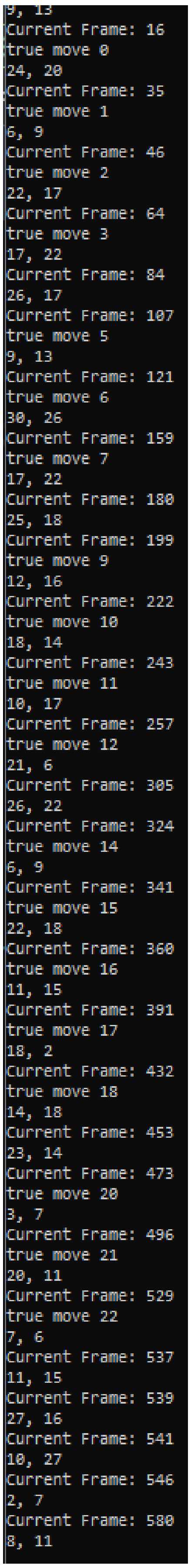
Part Three - Video Frames

I used a gaussian mixture model. I used it to check if there was movement in any of the frames from the last frame to the current frame. I then used absdiff on the threshold of the moving parts between the two frames to find the percentage of movement between the frames. If there is no movement between the two frames then I counted the number of stills until there were five still frames.

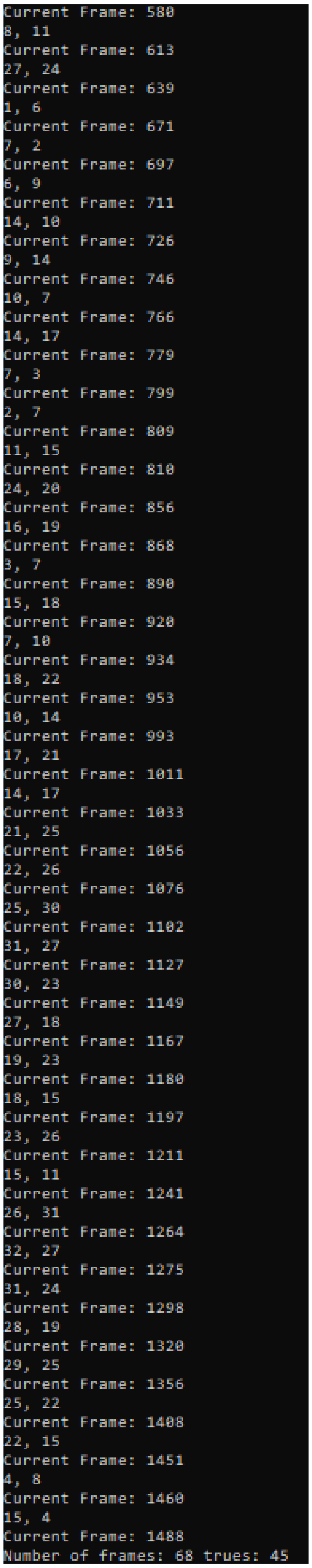
I also used my part two and part one to record moves made and to check if a full move has been made. If there was a difference between the last move and the current move then there has been a move made. It would locate where the new empty space was and where the piece was now located after being moved.

The problem I had with this section was comparing what I had with the ground truth. From a quick glance it seemed that a lot of the moves were being recorded correctly. When I tried to compare against the ground truth it didn't seem to record a lot of them correctly. The first few were correct and then my system skipped a move and after that my system was one step behind so none of them recorded correctly. However, on a closer look it seems that recording the moves using my part two and part one did work a good bit of the time even if it wasn't perfect.

I also didn't account for when a piece had been removed from the board due to another player capturing it. This could have been recorded wrong in my analysis as there would be two empty squares rather than one. This is something I would consider for making it more accurate.



```
9, 13
Current Frame: 16
true move 0
24, 20
Current Frame: 35
true move 1
6, 9
Current Frame: 46
true move 2
22, 17
Current Frame: 64
true move 3
17, 22
Current Frame: 84
26, 17
Current Frame: 107
true move 5
9, 13
Current Frame: 121
true move 6
30, 26
Current Frame: 159
true move 7
17, 22
Current Frame: 180
25, 18
Current Frame: 199
true move 9
12, 16
Current Frame: 222
true move 10
18, 14
Current Frame: 243
true move 11
10, 17
Current Frame: 257
true move 12
21, 6
Current Frame: 305
26, 22
Current Frame: 324
true move 14
6, 9
Current Frame: 341
true move 15
22, 18
Current Frame: 360
true move 16
11, 15
Current Frame: 391
true move 17
18, 2
Current Frame: 432
true move 18
14, 18
Current Frame: 453
23, 14
Current Frame: 473
true move 20
3, 7
Current Frame: 496
true move 21
20, 11
Current Frame: 529
true move 22
7, 6
Current Frame: 537
11, 15
Current Frame: 539
27, 16
Current Frame: 541
10, 27
Current Frame: 546
2, 7
Current Frame: 580
8, 11
```



```
Current Frame: 580
8, 11
Current Frame: 613
27, 24
Current Frame: 639
1, 6
Current Frame: 671
7, 2
Current Frame: 697
6, 9
Current Frame: 711
14, 10
Current Frame: 726
9, 14
Current Frame: 746
10, 7
Current Frame: 766
14, 17
Current Frame: 779
7, 3
Current Frame: 799
2, 7
Current Frame: 809
11, 15
Current Frame: 810
24, 20
Current Frame: 856
16, 19
Current Frame: 868
3, 7
Current Frame: 890
15, 18
Current Frame: 920
7, 10
Current Frame: 934
18, 22
Current Frame: 953
10, 14
Current Frame: 993
17, 21
Current Frame: 1011
14, 17
Current Frame: 1033
21, 25
Current Frame: 1056
22, 26
Current Frame: 1076
25, 30
Current Frame: 1102
31, 27
Current Frame: 1127
30, 23
Current Frame: 1149
27, 18
Current Frame: 1167
19, 23
Current Frame: 1180
18, 15
Current Frame: 1197
23, 26
Current Frame: 1211
15, 11
Current Frame: 1241
26, 31
Current Frame: 1264
32, 27
Current Frame: 1275
31, 24
Current Frame: 1298
28, 19
Current Frame: 1320
29, 25
Current Frame: 1356
25, 22
Current Frame: 1408
22, 15
Current Frame: 1451
4, 8
Current Frame: 1460
15, 4
Current Frame: 1488
Number of frames: 68 trues: 45
```

Figure 9: All the frames found and the moves in them.

Part Four - Four Corners

<u>Hough Lines</u>

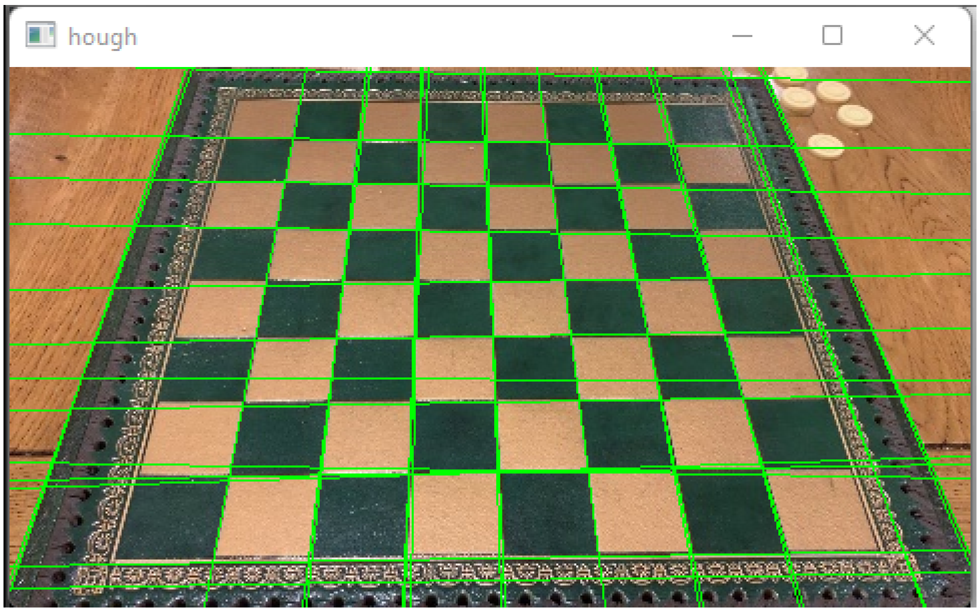


Figure 10: Hough lines on the empty board.

After smoothing twice using median blurring it reduces the noise that was giving lots of lines at the bottom of the image. As you can see this method isn't perfect. The two inner edges inside the frame are not found on the left and the top. It does seem to find the rest of the lines of the board however. It required some tuning of the parameters to get this right and it isn't 100% perfect. I suspect this isn't an optimal technique if the lighting is different or there is a good bit of noise.

<u>Contour following and straight line segment extraction</u>

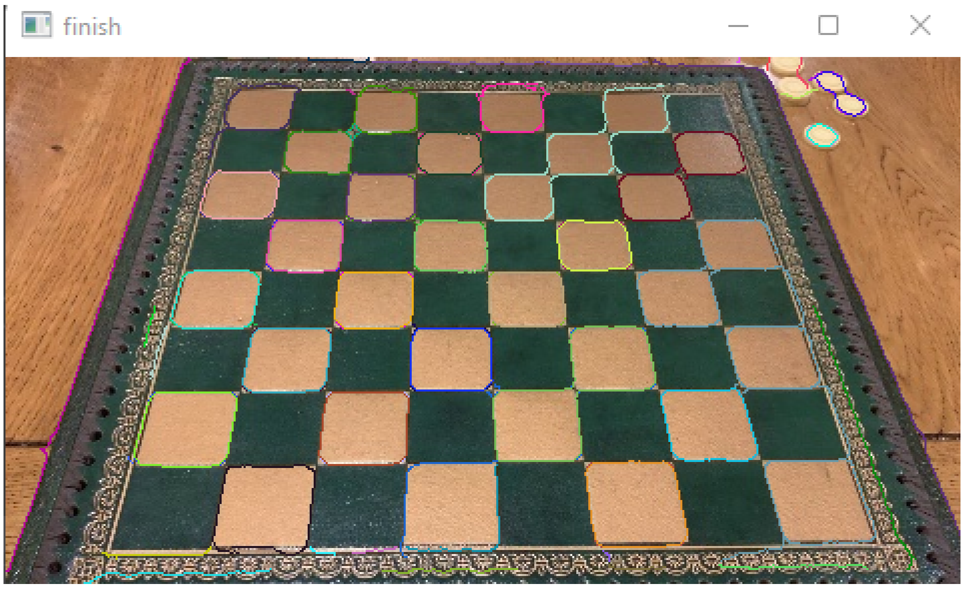


Figure 11: Contour following with no chain approx.

This does not work very well as you can see from figure 11. It does work in some locations of the board but it doesn't find every line which is an issue as it doesn't seem to find the actual edges of the board.

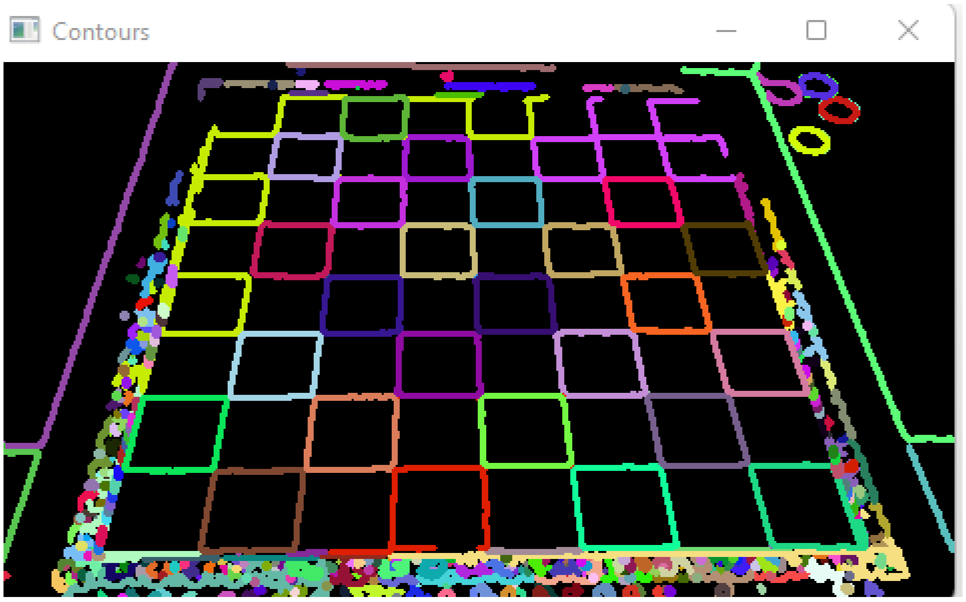


Figure 12: Contour following with a simple chain approx.

Using a simple chain approx works a lot better as you can see in figure 12 but again it is not perfect. The top two corners are not detected at all.

<u>Find Chessboard Corners</u>

This attempts to find a chessboard pattern and if the corners are in a certain order it returns true. It finds the 7x7 internal corners in this case. This requires a white space around the board to make the detection more robust in different environments. As this board has a frame around the outside I think in different lighting this wouldn't work so well.

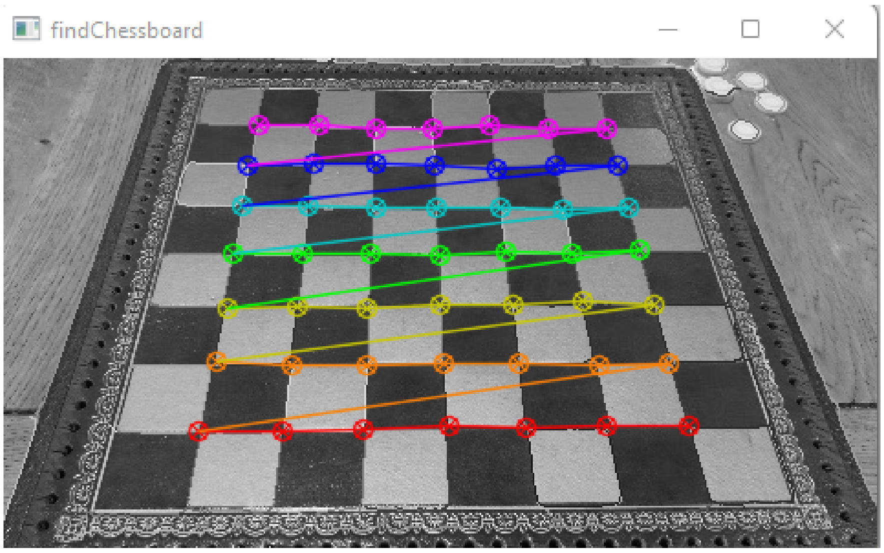


Figure 13: After using draw chessboard corners.

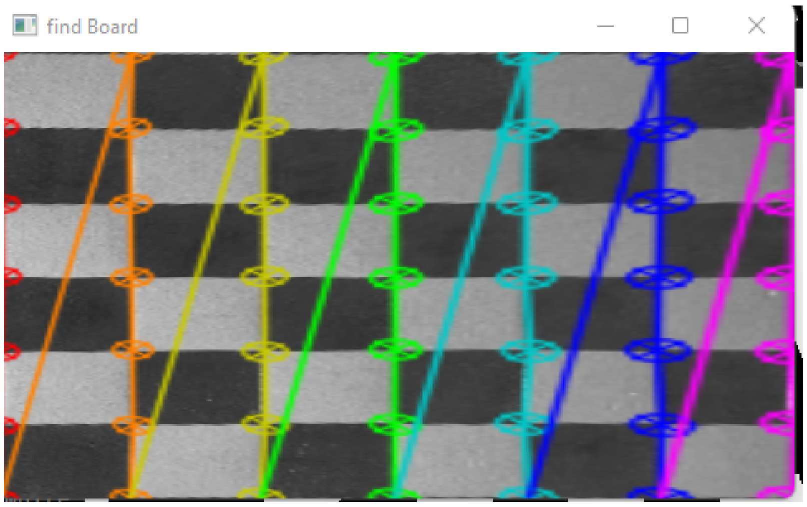
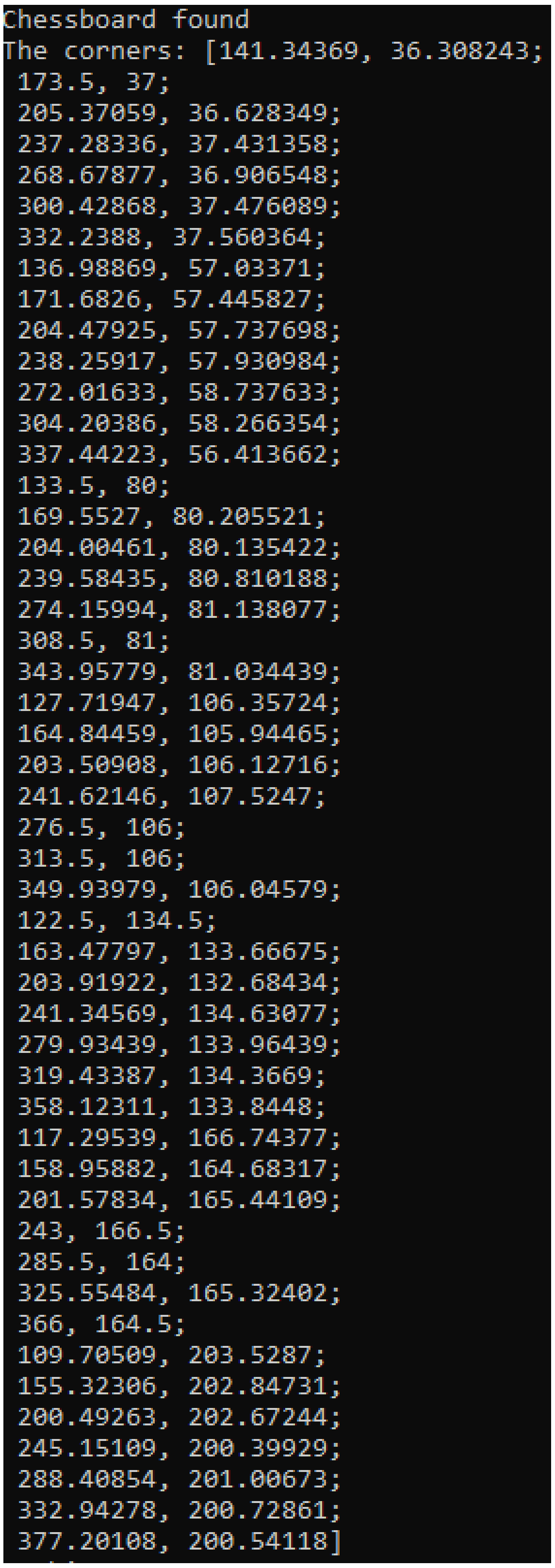


Figure 14: After doing perspective change using the corners from the function.

As you can see this technique does not provide the corners for outer corners but instead provides the inner corners.

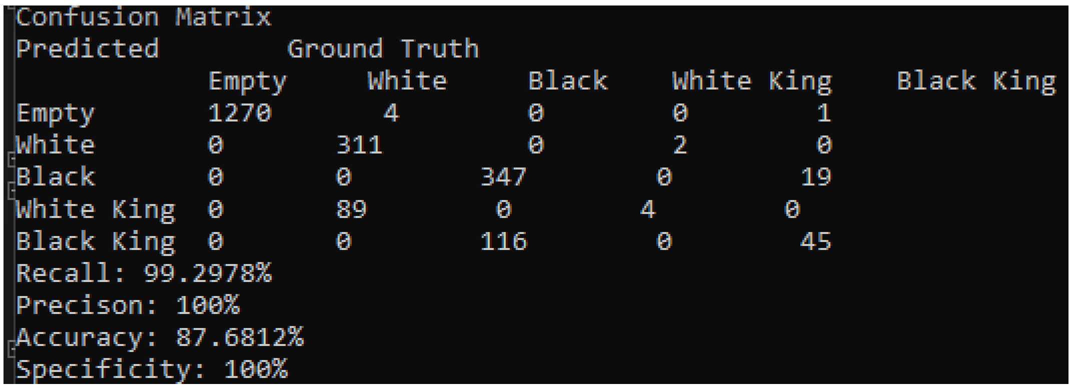
<u>Final comment</u>
After analysing all of the techniques I think that the most appropriate in this case is the Hough lines detection. The find chessboard corners is also a good technique but I think that because it requires a white background it may not work well in a natural setting.



```
Chessboard found
The corners: [141.34369, 36.308243;
 173.5, 37;
 205.37059, 36.628349;
 237.28336, 37.431358;
 268.67877, 36.906548;
 300.42868, 37.476089;
 332.2388, 37.560364;
 136.98869, 57.03371;
 171.6826, 57.445827;
 204.47925, 57.737698;
 238.25917, 57.930984;
 272.01633, 58.737633;
 304.20386, 58.266354;
 337.44223, 56.413662;
 133.5, 80;
 169.5527, 80.205521;
 204.00461, 80.135422;
 239.58435, 80.810188;
 274.15994, 81.138077;
 308.5, 81;
 343.95779, 81.034439;
 127.71947, 106.35724;
 164.84459, 105.94465;
 203.50908, 106.12716;
 241.62146, 107.5247;
 276.5, 106;
 313.5, 106;
 349.93979, 106.04579;
 122.5, 134.5;
 163.47797, 133.66675;
 203.91922, 132.68434;
 241.34569, 134.63077;
 279.93439, 133.96439;
 319.43387, 134.3669;
 358.12311, 133.8448;
 117.29539, 166.74377;
 158.95882, 164.68317;
 201.57834, 165.44109;
 243, 166.5;
 285.5, 164;
 325.55484, 165.32402;
 366, 164.5;
 109.70509, 203.5287;
 155.32306, 202.84731;
 200.49263, 202.67244;
 245.15109, 200.39929;
 288.40854, 201.00673;
 332.94278, 200.72861;
 377.20108, 200.54118]
```

Figure 15: Corners found by the findChessboardCorners.

Part Five - Kings



```
Confusion Matrix
Predicted       Ground Truth
            Empty     White     Black    White King    Black King
Empty       1270        4         0         0            1
White       0        311          0         2            0
Black       0        0         347        0         19
White King  0        89         0        4         0
Black King  0        0         116        0         45
Recall: 99.2978%
Precison: 100%
Accuracy: 87.6812%
Specificity: 100%
```

Figure 16: Extended confusion matrix for Part 5.

Here the system I used was very similar to my system for part two. I checked to see what colour piece was in the square by counting pixels. I then checked for if there was a king in the square by tracking the location of the piece. If it had made it to the edge then it was a king. This doesn't work perfectly and I would use a different system to keep track of the kings and their moves.

Another method to do this would be to check the circularity of the pieces. If the piece was a man on a square then the circularity would be 100% but if it was a King then the piece may not have 100% circularity. Of course this doesn't account for if the second piece has been placed perfectly on top of the other one. But this doesn't happen often. I also considered increasing the threshold in the analysis of the pieces but I think this would depend on the lighting and the samples used in the back projection.