Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

CSU44053- COMPUTER VISION

Draughts Assignment- Report

Submitted by:

Parul Kumari

21356575

Part 1: For the first part, we were asked to classify the image as part of a white piece, a black piece, and part of a white square on the board. Given the constraints of lightning change and the appearances of the pieces and the board.

Analysis and techniques used: Used perspective transformation to align the image in a plane point of view because it was not parallel to the image plane. Also, I used perspective transformation as the given image could not be corrected with affine transformation.

Next, greyscale is used to convert the RGB image to a grey image. Below is the image after applying greyscaling.

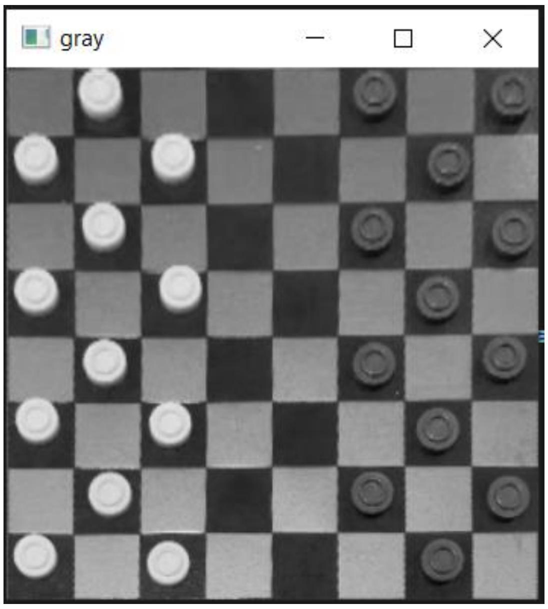


Image after geyscaling

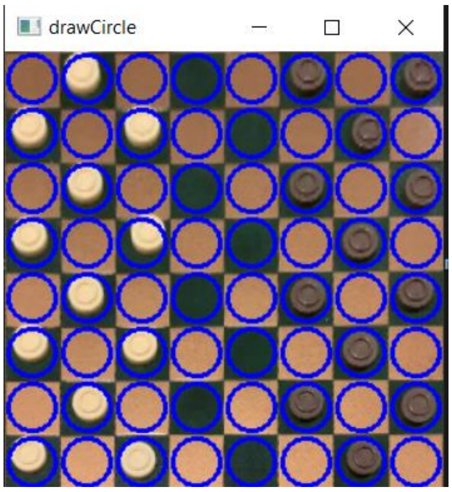
Next, to find circles I performed various steps mentioned below:

a. Hough Transform: Transformation from image space to probability of the existence of features like lines, circles, and other generalized shapes. To locate circles, initialized the accumulator to 0 and located every possible circle by incrementing cells for every edge point.
b. Thresholding: After subtracting the static background from the original frame, we get our object. And we do binary thresholding to get results. In particular, we applied Otsu thresholding.
c. Erosion: Removing pixels. Applied to make regions smaller and remove noise.

d. Dilation: Adds pixels around borders. Makes the regions bigger and fills small holes.

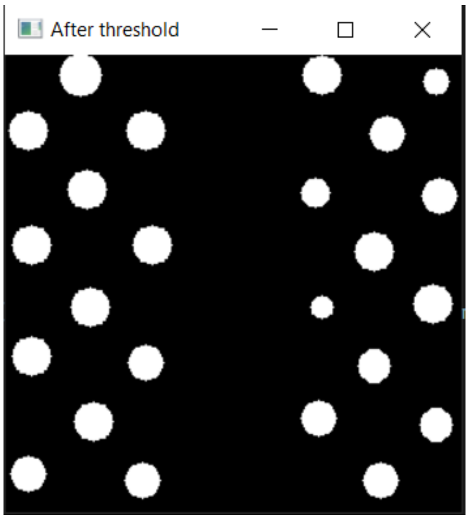
I could say I applied opening which is also applying dilation after erosion.

Below are the detected circles after doing the mentioned steps:

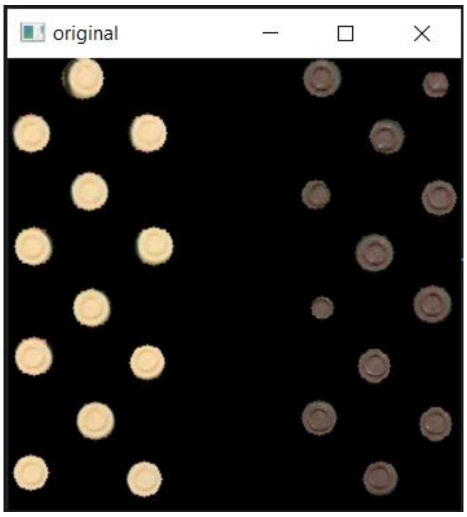


Detected circles

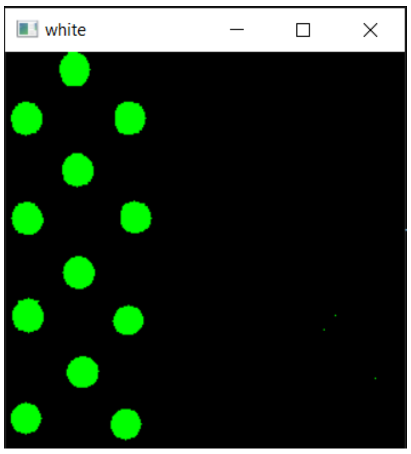
Now, the binary image is multiplied with the original image and the final image with all things in different colors.



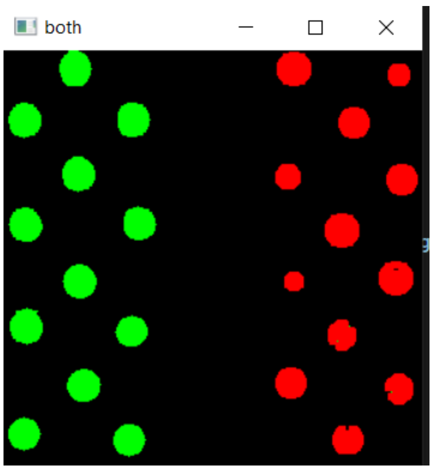
Binary image after threshold



Multiplied with the Original image



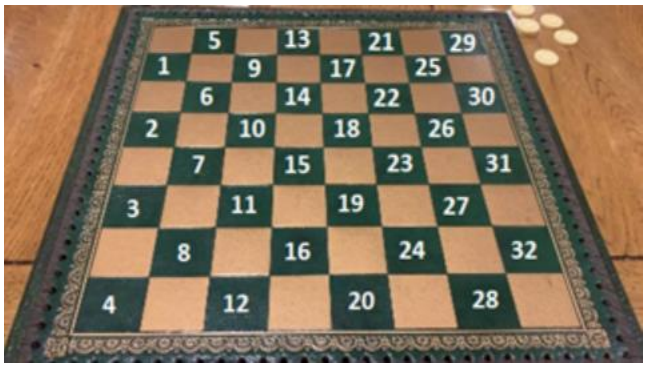
White pieces



White and black pieces

Part 2: We were given the locations of four corners of the board ((114.0, 17.0), (53.0, 245.0), (355.0, 20.0), (433.0, 241.0)) and asked to determine if there is any piece in each square and of which colour the piece is. Then, record the locations of those pieces and compare them to the provided ground truth.

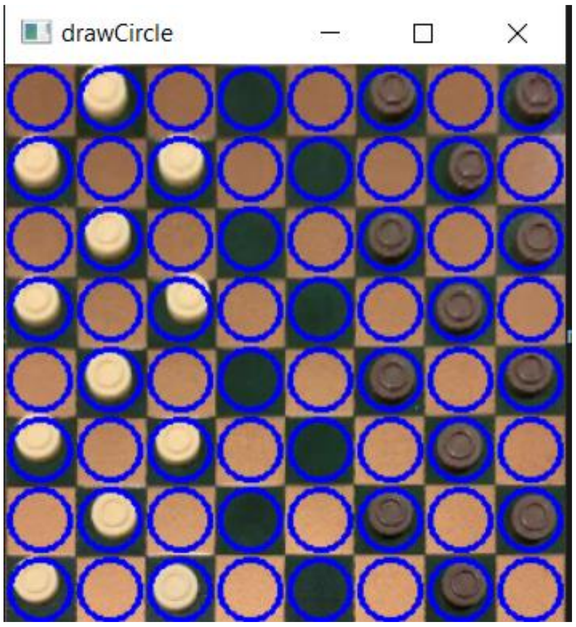
Analysis and technique used: piece locations were specified in the portable draughts notation format where the numeric notations are:

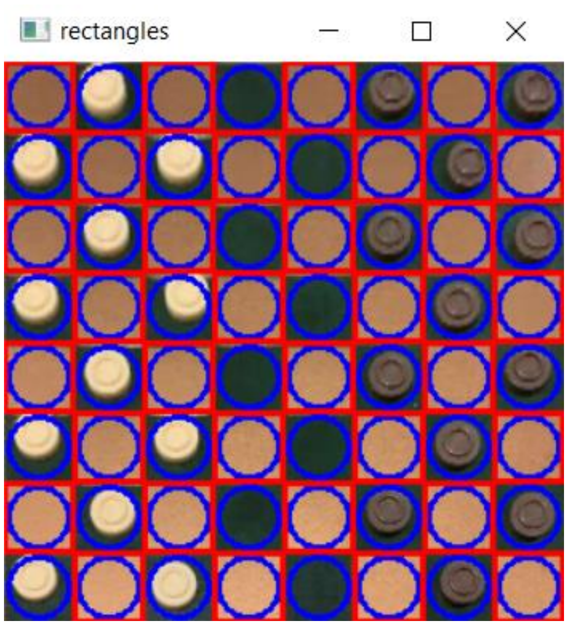


Now, all that data is assigned to the draughts board structure, and classes for each object are calculated and assigned to one colored image.

Pieces are updated in the boxes; 1-64 boxes are used. I used only black ones first during the processing. No black or white piece could be placed on a white box; therefore, white boxes are not used.

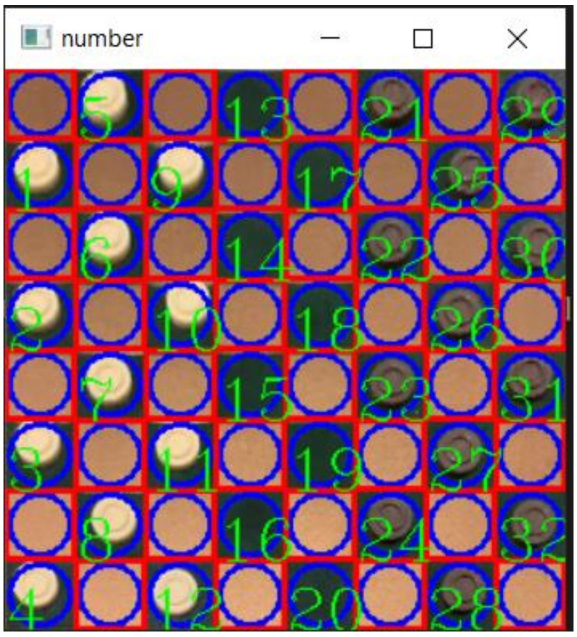
Now, both, white and black pieces, are assigned with the threshold accordingly. After this, we got the class for each piece and the box structure is updated.





Next, I loaded the ground truth and compared all the images with the ground truth and found all the moves are correct.

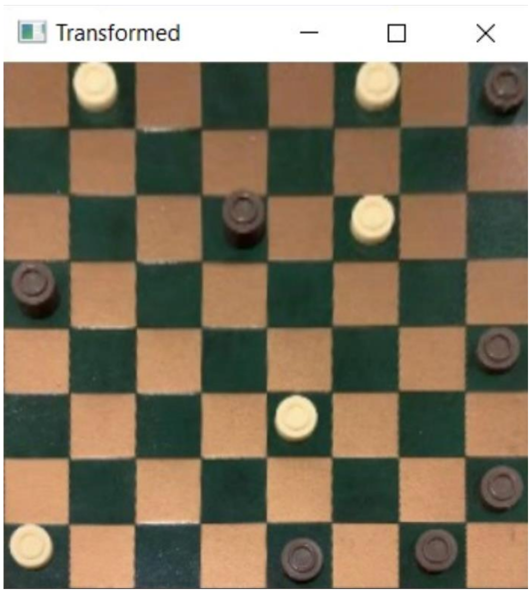
Below is the image with output piece locations and comparison:



Part 3: We were asked to find and record the moves made against using the Portable Draughts notation and analyse the performance of the system in terms of the frames found and correct recorded moves.

Analysis and technique used: the given video of the draughts game is loaded frame by frame and then images are transformed.

Below is what the images from the video would be like after the transformation:



Using the codes of part 1 and part 2, I checked 2 cases:

a. for normal moves when circle remains the same
b. for kill move when the circles are reduced

The concept is the frames are recorded after every hand movement and the pieces and boxes are observed. For every change in the pieces and boxes, a movement is recorded.

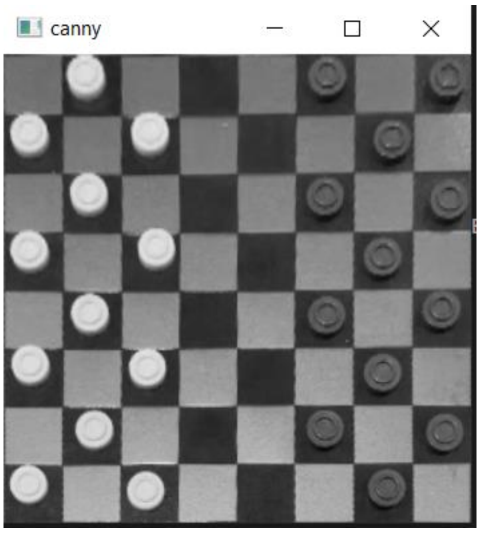
I observed that the stable count during the hand movements is 8 which means no moves are recorded after that. Now the result is compared with the given ground truth and found to be correct.

Part 4: In this part, we were asked to process any static image of the board and determine the locations of the four corners. We were given the approaches that we could perform to do the task.

Analysis and technique used:

1. **Hough lines**: a filter that is used to detect lines, circles, or any generalized shape.
   I applied the canny filter in OpenCV with thresholds of 200 and 250, which is a technique for edge detection, a combination of 1st and 2nd derivative detectors. It first removes noise, adds a smoothening filter, and then computes the image gradient to highlight regions with high derivatives. The algorithm then applies a double threshold that is inbuilt to find weak and strong points. The result is binary.
   Next, with a value of 150, I applied binary transform and converted that into colour for adding red lines.
   Finally, Houghlines applied for $\pi/180$ to 80.

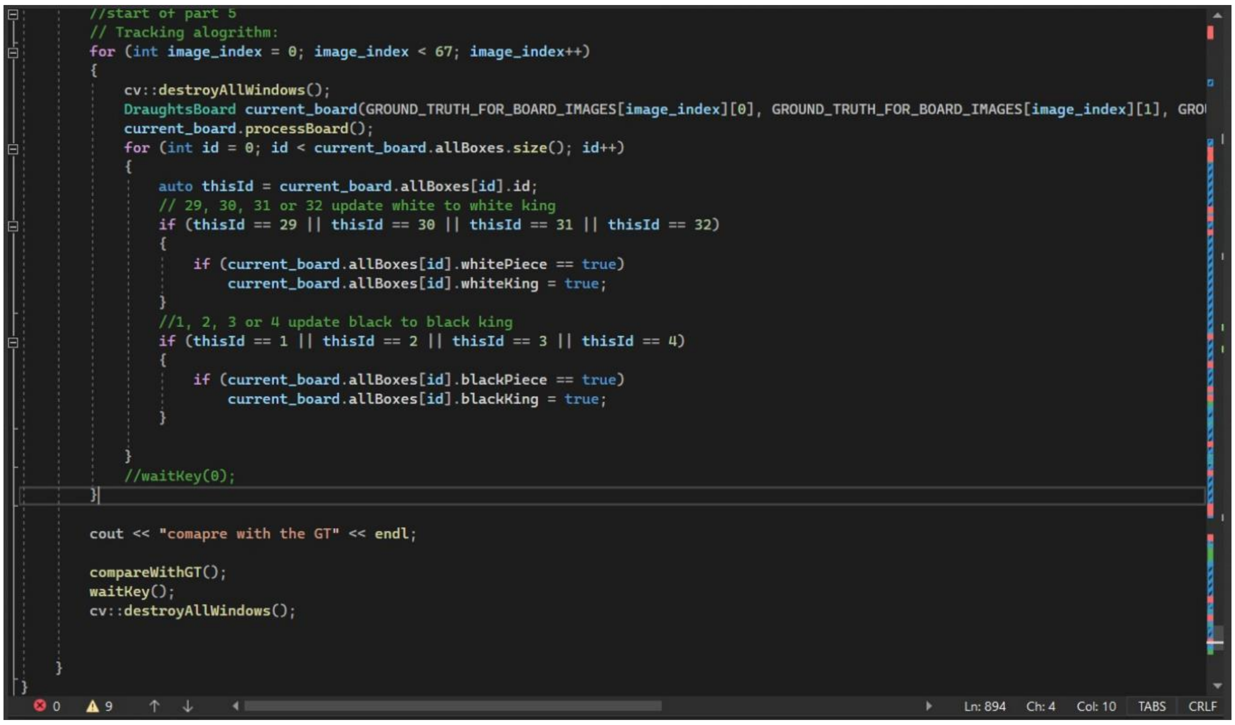
   After the canny filter, we got the image below:

2. **Contour Lines:**
   Contour means a chain of points. Contour segmentation is used for edge data representation, border detection, and line segment extraction. Contours don't correspond to individual objects in the world. We were asked to experiment with finding the border locations of the draught board.
   First, I converted the image to grayscale and then applied thresholding. Then, I used the function find contours in OpenCV, and contours were drawn so then contour approximation could be done.

3. **Chessboard detection:** First I changed it to greyscale and then used the OpenCV function findchessboardcorners(). It detected the internal corners of the board

Part 5: It was asked to develop a code to distinguish between normal pieces and kings.



```
        //start of part 5
        // Tracking alogrithm:
        for (int image_index = 0; image_index < 67; image_index++)
        {
            cv::destroyAllWindows();
            DraughtsBoard current_board(GROUND_TRUTH_FOR_BOARD_IMAGES[image_index][0], GROUND_TRUTH_FOR_BOARD_IMAGES[image_index][1], GRO
            current_board.processBoard();
            for (int id = 0; id < current_board.allBoxes.size(); id++)
            {
                auto thisId = current_board.allBoxes[id].id;
                // 29, 30, 31 or 32 update white to white king
                if (thisId == 29 || thisId == 30 || thisId == 31 || thisId == 32)
                {
                    if (current_board.allBoxes[id].whitePiece == true)
                        current_board.allBoxes[id].whiteKing = true;
                }
                //1, 2, 3 or 4 update black to black king
                if (thisId == 1 || thisId == 2 || thisId == 3 || thisId == 4)
                {
                    if (current_board.allBoxes[id].blackPiece == true)
                        current_board.allBoxes[id].blackKing = true;
                }

            }
            //waitKey(0);
        }

        cout << "comapre with the GT" << endl;

        compareWithGT();
        waitKey();
        cv::destroyAllWindows();

    }
}
```

I concluded from previous techniques that pixel count from back projection might not be the right approach because the detection of 2 pieces could not be done by background segmentation.

Followed approach: A piece gets converted from normal to a king piece when it reaches the other end of the chessboard. For example, a white piece when reaches box number 30 or 31 and when a black piece reaches box number 1 or 2, it converts into a king.

Now, I applied the function, and the images are compared with the ground truth.

Resources:

- **Slides and videos from blackboard Computer Vision**