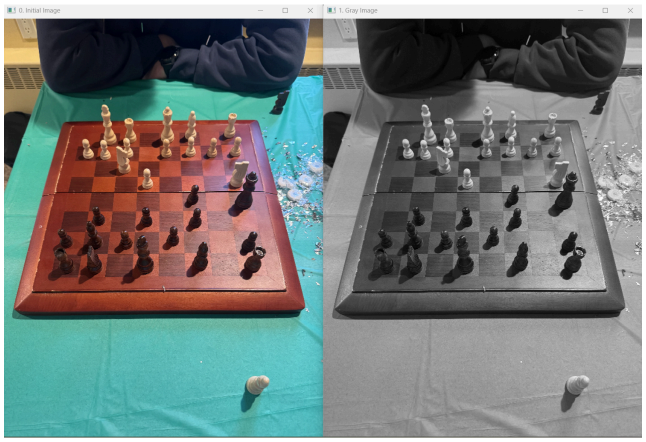
<u>Hough Lines</u>

Below is a formalized and structured description of how my application utilizes OpenCV and Hough Lines as of February 25th, 2026.

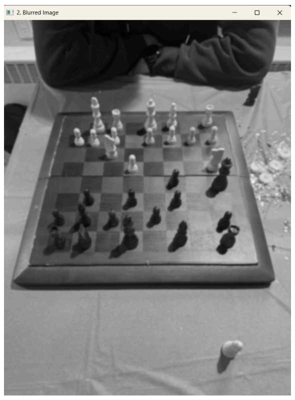


Initialization:

We first begin by loading in the initial image of the board.

Graying of Image:

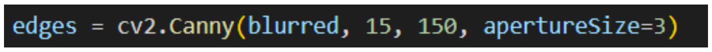
Next we gray the image. We do this because to test edges, we detect contrast between colors. If we leave the image in RGB, we would be comparing pixels in 3 channels, by converting the image to grayscale we can compare pixels on 1 channel.



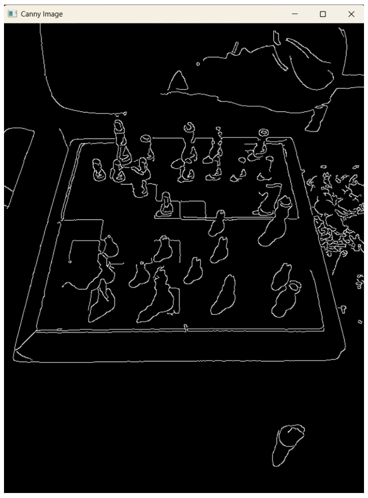
Blurring of Image:

We blur the image in order to reduce the amount of ‘noise’ in the image. By blurring the image we smooth out the image and allow for more continuous lines. This greatly decreases the number of false positives that occur and provides clearer edges for detection.

Canny Image:

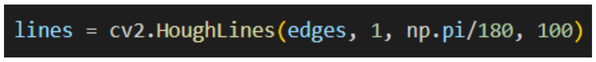


```
edges = cv2.Canny(blurred, 15, 150, apertureSize=3)
```



Canny converts the images to all the found edges, and connects them. We end up with the image right after running the above line of code on our blurred image. The 15 represents the *lower threshold* while the *150* represents the upper threshold. This means that any pixel with a gradient of 150 is considered a strong edge and is immediately accepted. Any with a gradient less than 15 is discarded immediately. All pixels between the upper and lower threshold are kept, if they are connected to a strong edge. apertureSize affects how gradients are computed. You can choose between values 3, 5, and 7. A higher aperture size means more detail, which can often lead to more false positives, which is why I have kept mine to 3 at this stage.

Hough Lines:



```
lines = cv2.HoughLines(edges, 1, np.pi/180, 100)
```

The list *lines* store all our generated hough lines. This is the part of the process I am not fully sure what the *1*, or *np.pi/180* values do, nor have I played around with what these values. The last parameter *100* is how long an edge needs to be for it to be considered a line. After playing around with different values, I determined that 100 gives us the least amount of false positives while still picking up the edges that I need.