Problem 1 (Spot Removal).

Solution. I fought with part (b) a bunch and finally called it. I was able to make a pretty good mask out of it by tweaking some parameters but I know the approach after this is going to involve a convoluted function to replace those areas masked with the color that it's nearest neighbor.

```
# # was running into dpi issues only on this pc ugh (work pc)
plt.figure(dpi=300)


img_gray = io.imread('images/Lichtenstein_imageDuplicator_1963_gray.png')
img_color = io.imread('images/Lichtenstein_imageDuplicator_1963.png')

# plt.imshow(img_gray, cmap='gray')
# plt.imshow(img_color)

selem = disk(3)

img_closed = closing(img_gray, selem)
img_dilated = dilation(img_gray, selem)
img_eroded = erosion(img_gray, selem)
img_opened = opening(img_gray, selem)

plt.imshow(img_opened, cmap='gray')

# closing and dilated are similar while erosion and opening are similar
# all of them remove dots, the first two mess with letters and the second two
# darken the face quite a bit (washing it all out)

red_channel = img_color[:, :, 0]
green_channel = img_color[:, :, 1]
blue_channel = img_color[:, :, 2]

red_mask = (red_channel > 150) & (green_channel < 100) & (blue_channel < 100)
large_mask = dilation(red_mask, disk(1))

# plt.imshow(red_mask)

img_removed = img_color.copy()
img_removed[large_mask] = [0, 0, 0]

plt.imshow(img_removed)
```

Figure 1: Code written for Problem 1

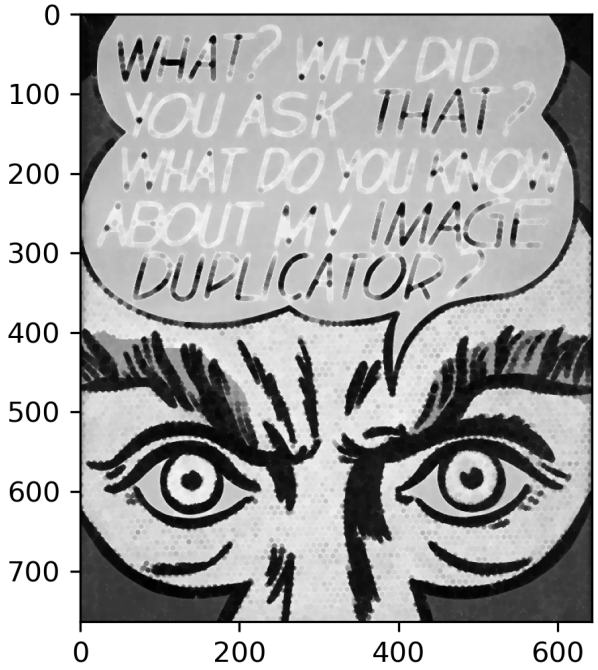


Figure 2: Closing

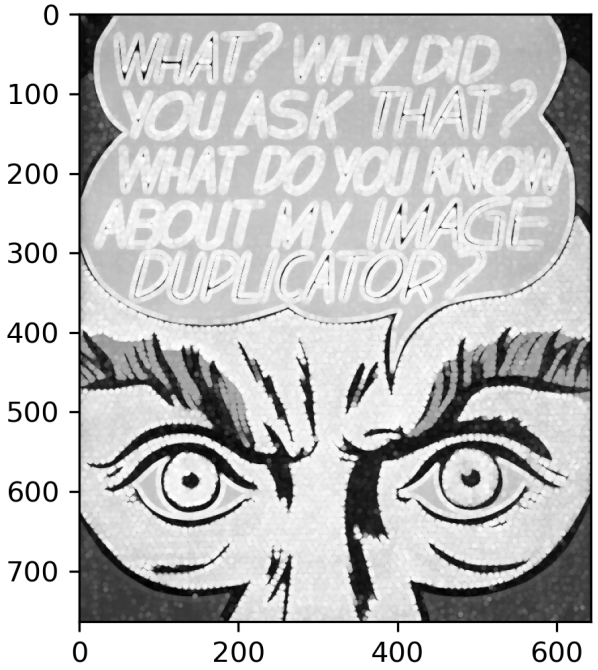


Figure 3: Dilation

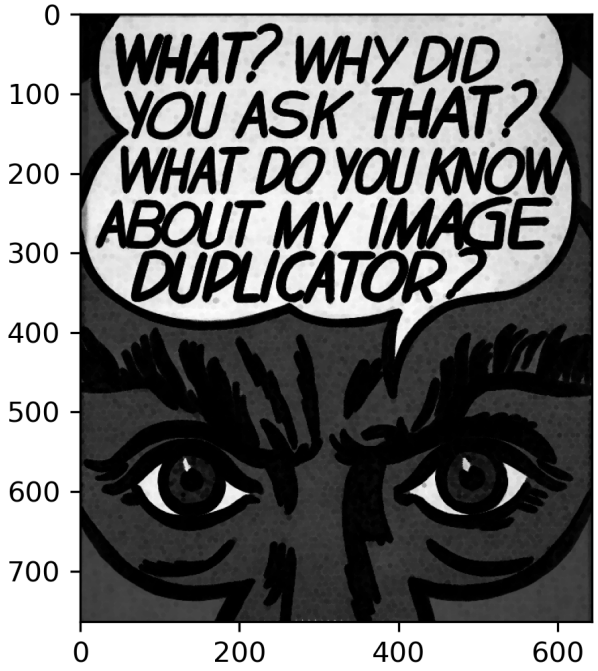


Figure 4: Erosion

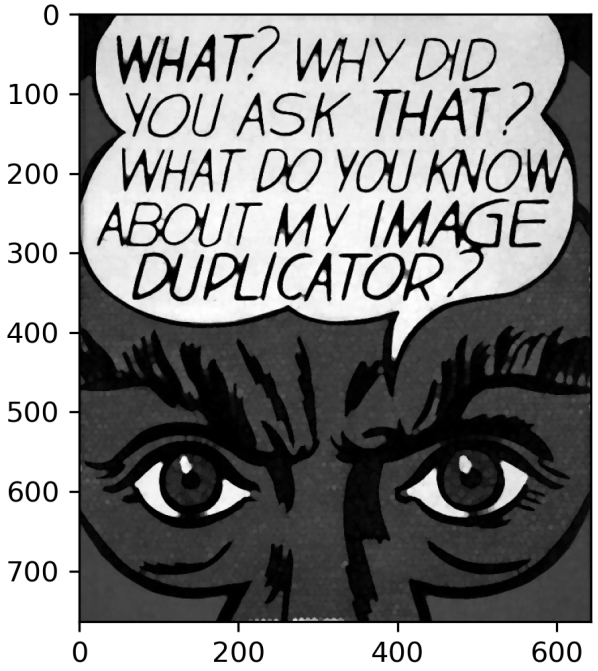


Figure 5: Opening

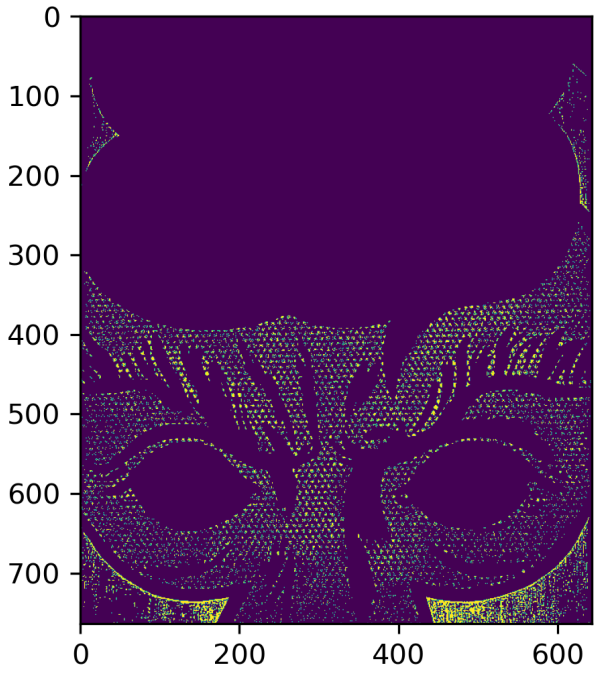


Figure 6: Red mask.

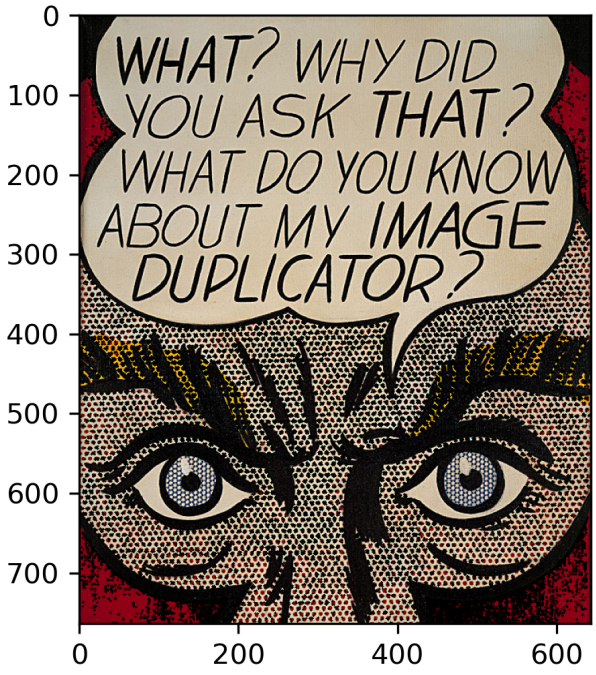


Figure 7: Subtracting the mask from the image.

So the mask looks like it would work well, if I could find a way to subtract just the mask and replace it with nearest neighbors. We would lose some of the color in the solid red parts but this would handle the red dots really well I think!

Problem 2 (Segmentation by thresholding).

Solution.

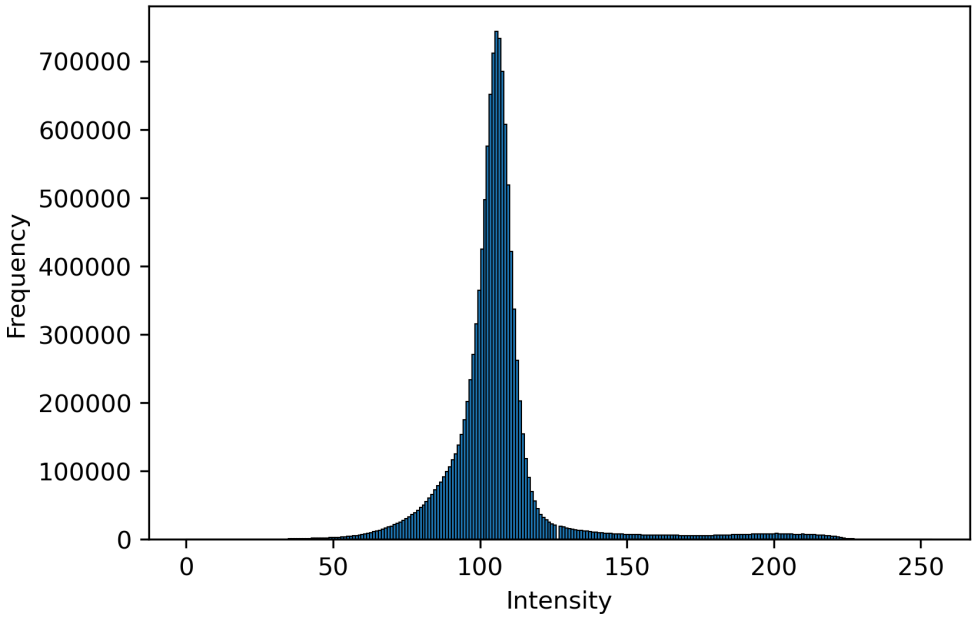


Figure 8: After applying a high pass filter, a histogram of intensity values. I chose 150 to be the cut off for values before converting it to a binary image.

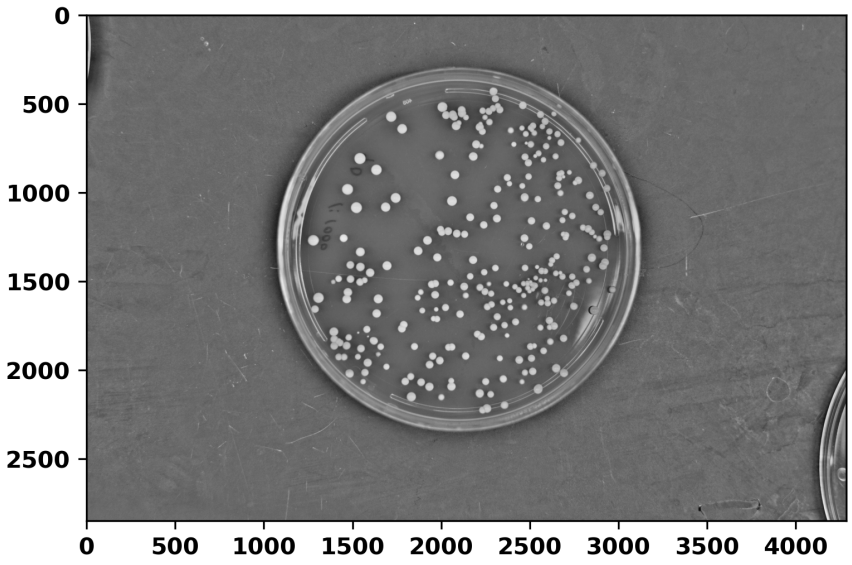


Figure 9: The image after applying a highpass filter.

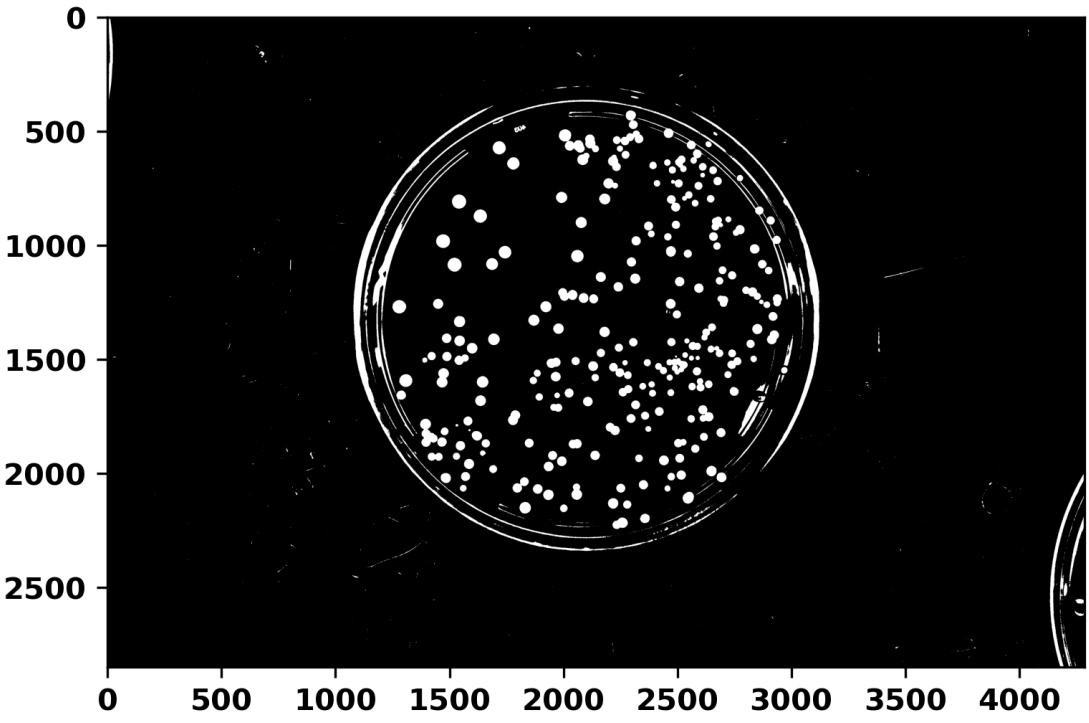
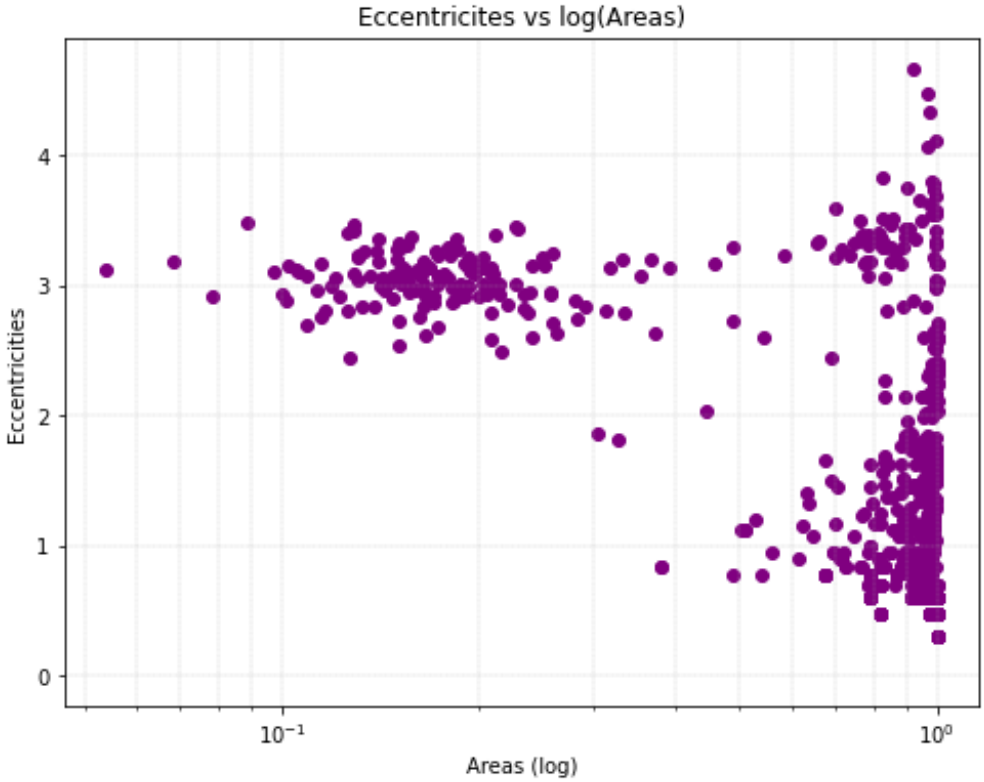


Figure 10: The image after thresholding



Problem 3 (Gradient magnitude).

Solution.

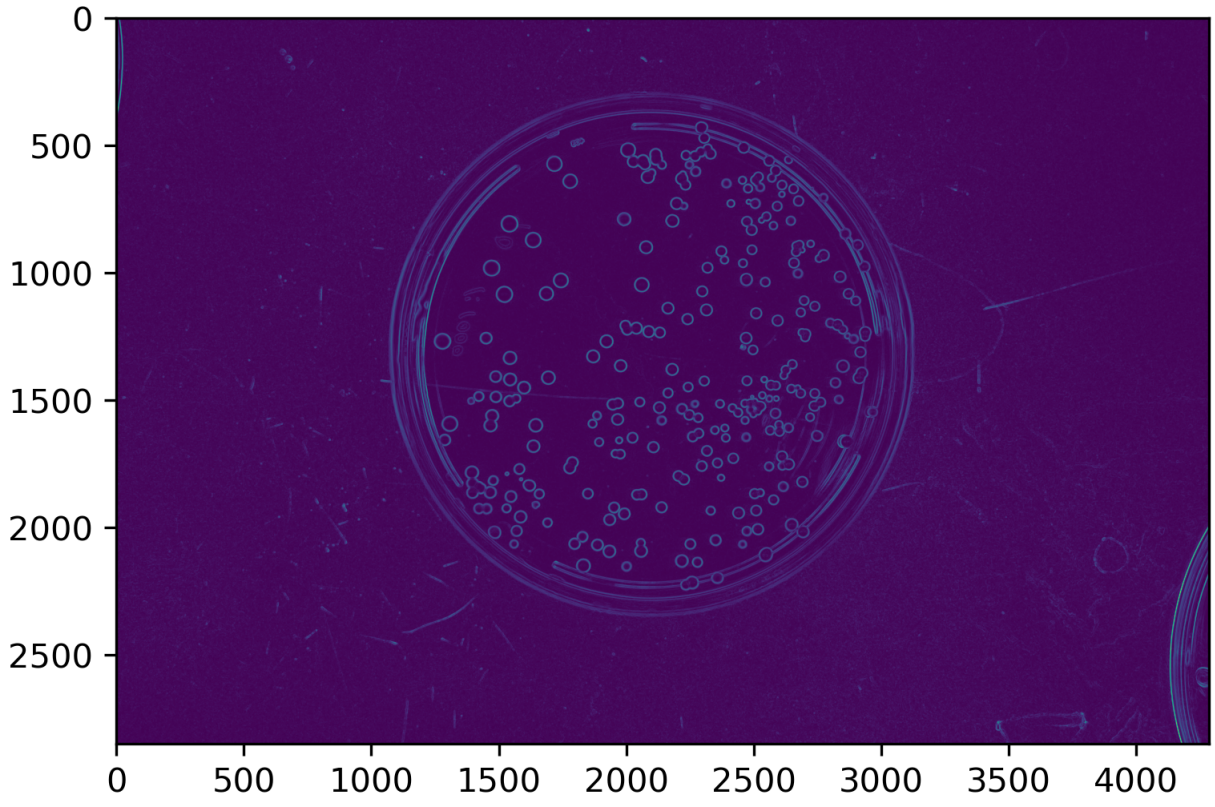


Figure 11: I'm not quite sure what caused the blue tint but I actually liked it!

Problem 4 (Watershed segmentation without markers).

Solution.

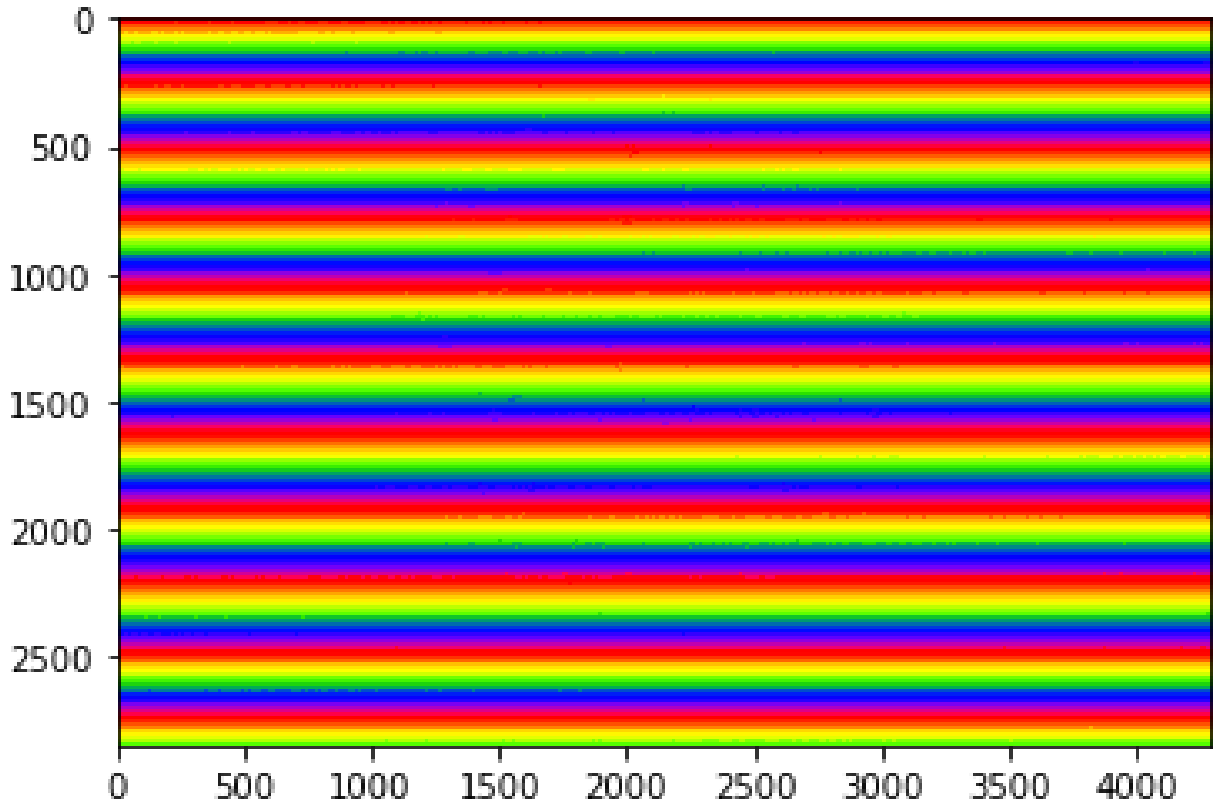


Figure 12: This indeed looks terrible. You can see in some of the lines that it isn't a straight line across the horizontal, it's indeed got noise associated with.

Problem 5 (Watershed segmentation with markers).

Solution.

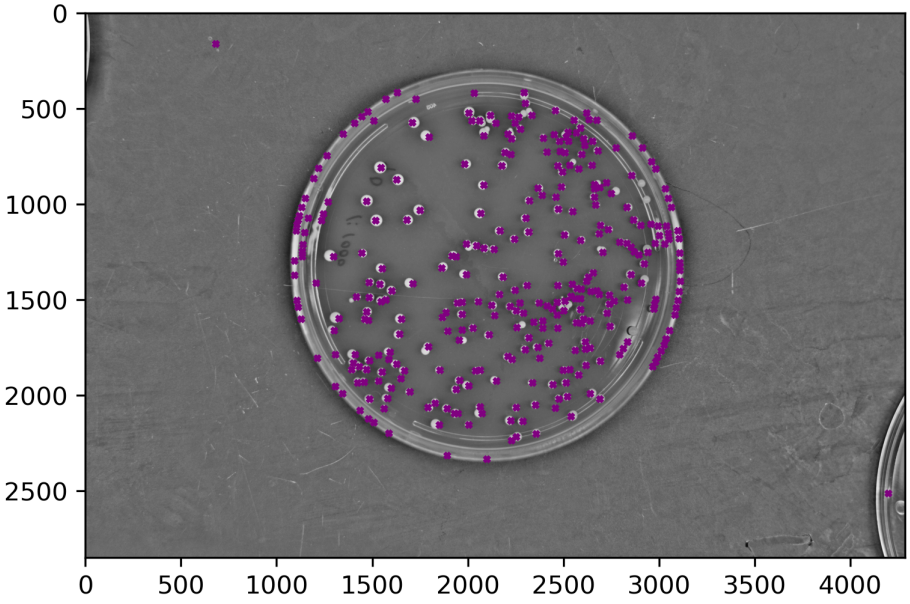


Figure 13: Markers seem to land on the colonies well but I'd have to tweak my previous work to keep it off of the edge of the dish.

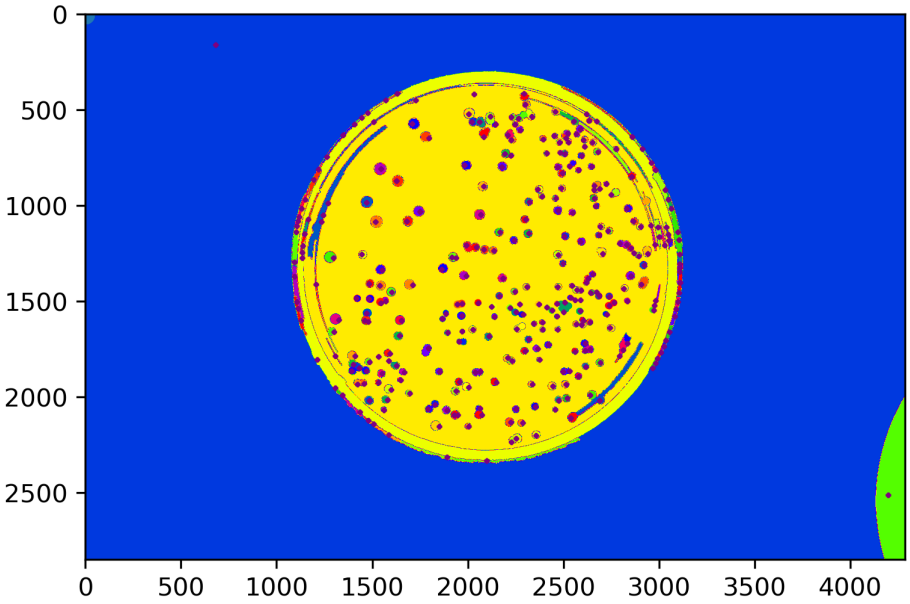


Figure 14: This does a reasonable job of distinguishing between the colonies that are touching but some are not so good (zoomed in next image).

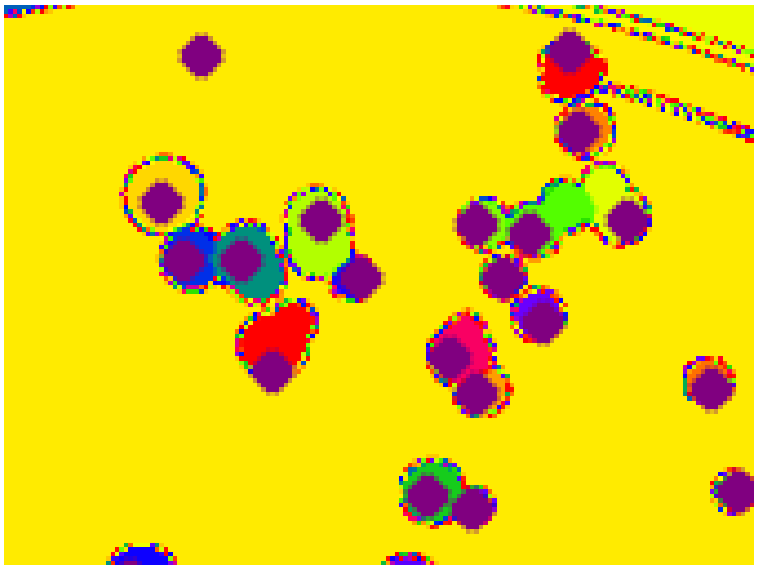


Figure 15: Here we can see some discrepancies between colonies that seem to be touching.

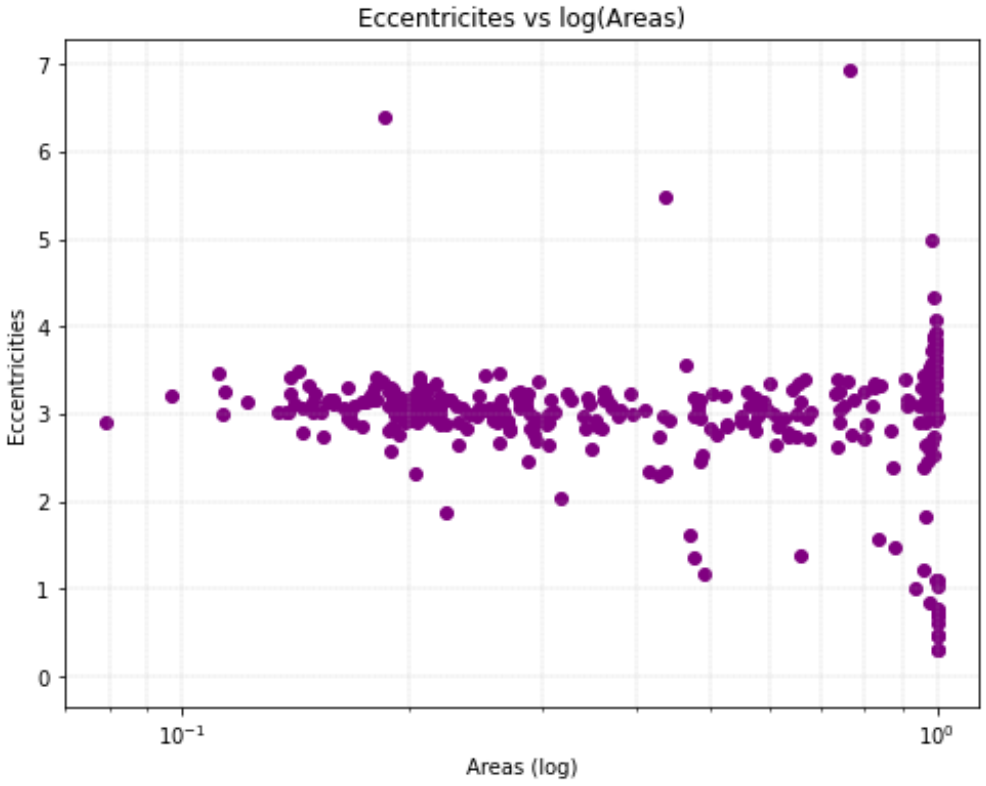


Figure 16: It looks like we have some grouping on the left and far right. I think one could cut out the outliers on the y-axis by maybe making some stipulations for what should be considered a colony or not. Perhaps a mean of some sort?