Problem 1:

Image 1:

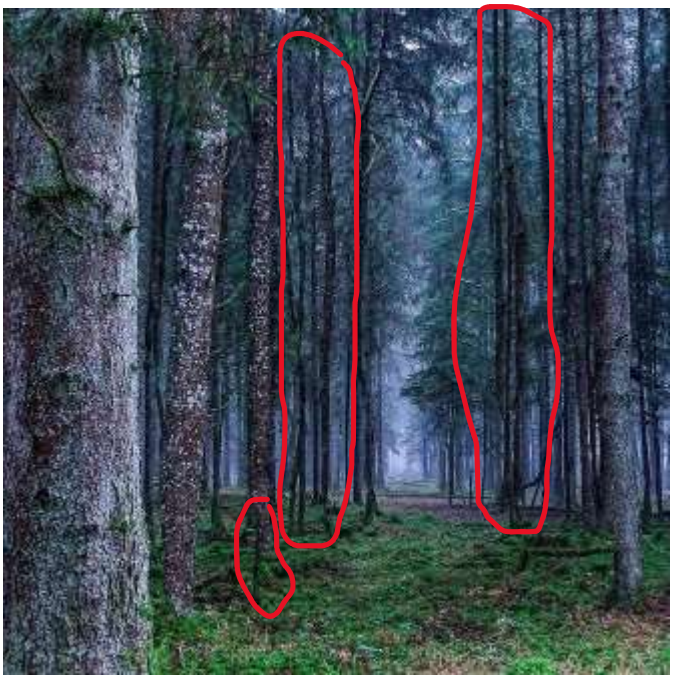
Before seam carving:

After seam carving:

Even though the center tree has increase in size slightly, there doesn't seem to be a massive loss in quality overall. The only real exceptions that I can see are the red circled clouds and tree that are squashed closer together.

Image 2 before seam carving:

Image 2 after carving:



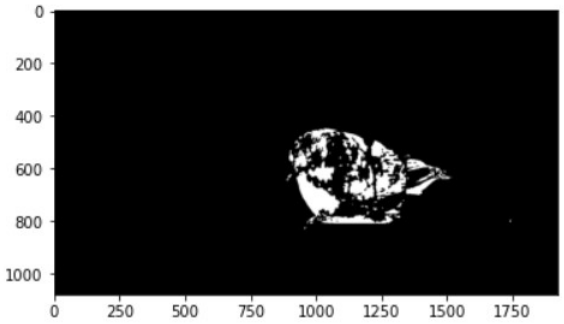
After applying seam carving, the darker trees are getting thinner because the dark trees have smaller gradients in both x and y directions due to continuity unlike the tree on the left hand side. Because of this, the seam carving ends up cutting them out rather than preserving them as part of the resizing.

Though this algorithm is basically classifying seams as non essential areas, it will cut out smooth areas that without being able to understand what objects or areas are essential to final image. So for the forest image, trimming of the darker trees means a loss of information or data in the final image. This algorithm works best on pictures that have a lot of continuity and smoothness and not so great for images that have higher gradient values and a lot of color changing.

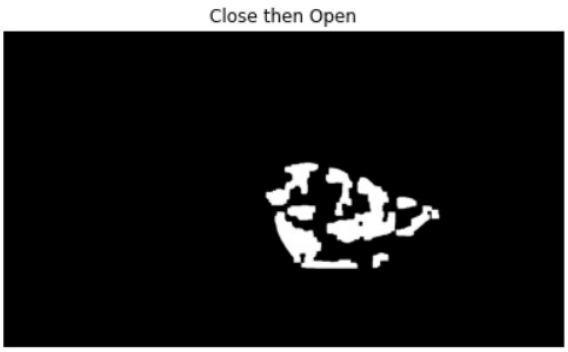
Problem 2:

a)

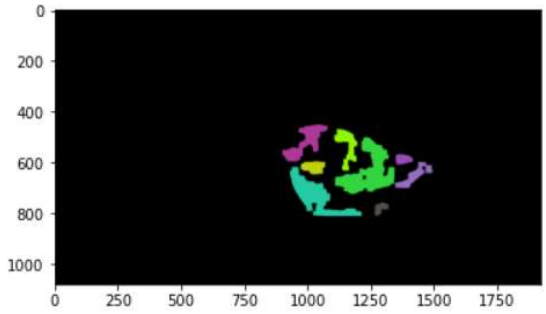
First thing I do, for the diff image, is zero out the region bounded by rows image.shape[0]*3//4 : image.shape[0], columns image.shape[1]//2 : image.shape[1] as this is an approximation of the region that is covered by the bird feeder in the foreground. By cancelling this out, we can make sure that the seed distortion doesn't throw off results when we start modifying the image further to close gaps and intricate features of birds. Next I zeroed the whole left hand side of the diff image since this bird detector only detects birds on the right side (bird feeder and roof occupy most of the right hand side with the exception of the trees). Next I median blur the image with kernel size 7 as a way to get rid of salt pepper noise. This proved to be quite helpful since very small changes in the tree movement and lighting are covered up in the negative bird detection image examples. I kept the thresholding step the same as the original juptyer notebook since it was working quite well at getting me the bird features:



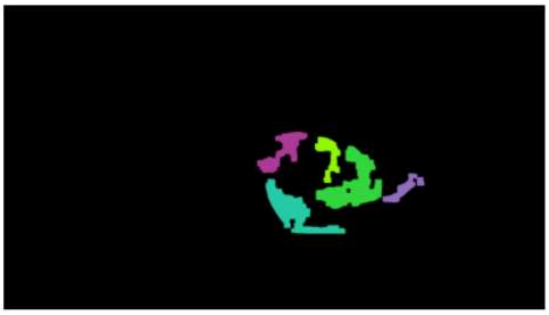
My decision criteria functions by decomposing the bird into strongly connected components and using the sums of distances between the components' centroids and the center of the image and location (x, y) image.shape[1]*3//4, image.shape[0]*3//4 as this is the lower right center. I chose this as the lower right center because the bird will appear on the right side and I wanted to choose the lower center since it accounts for if the bird comes too close to camera or is further away. If the sum is greater for the distances to the center, then no bird is detected. Otherwise a bird is detected. If no components were found, then the code determines no presence of birds. I used a (12,12) kernel performed an image close, then an open, followed by a (15,15) kernel open:



Running strongly connected components algorithm produces:



As a backup, if too many small components exist and are causing the sum of distances to center to be higher, then my code filters out components that are less than 4000 pixels in size and retries the distance measurement comparison. That is why initially I have such a large opening and closing kernel to get rid of small specs (noise from subtle tree movements or lighting changes) and amplify the birds features



b)

My algorithm is far from perfect and works on the assumption that bird detection will only succeed if the bird is sitting on the bird feeder that rests on the right half of the image. If the bird was flying in from the left side, the algorithm will fail since my preprocessing zeroes out the entire left half of the image. If a bird is leaning most of its body over the edge or hugs the ledge very closely will being too close to the camera, then the algorithm will likely fail since most of

the deconstructed components will have smaller distances to the center of the image. Another example of failure might occur if there is too much distortion with the portions of the trees that are closest to the ledge such as if an entire tree goes missing or sections are moving a lot due to wind. The opening and closing might amplify clusters of noise and give the false impression of a bird being present. As this algorithm doesn't take into account large changes in light, so potentially large enough changes in light will cause failure if the opening and closing don't filter out the smaller infractions.