Prof. Raghuveer Parthasarathy
University of Oregon; Fall 2024

Physics 410/510 Image Analysis: Homework 8

Solutions

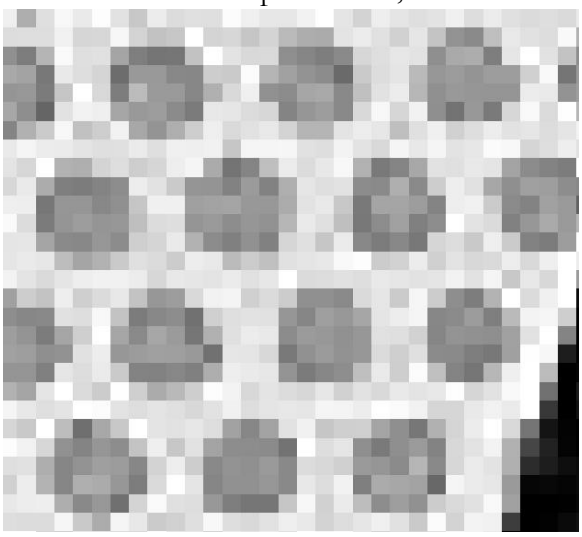
1 Spot removal. Download the file "Lichtenstein_imageDuplicator_1963.png" – a painting called "Image Duplicator" by Roy Lichtenstein (1963). (....)

(a) ....

(b) **[510 students]** Now consider the color image. See if you can figure out a way to remove *only* the red dots. ...t.

Solution

Zooming in, we can see that the dots are 6 or 7 pixels wide, and are darker than the background:



Therefore a **closure** of radius 4 pixels should erase the dots, since dilation will replace all these pixels with the brighter background and erosion won't revive the dots.

Python:

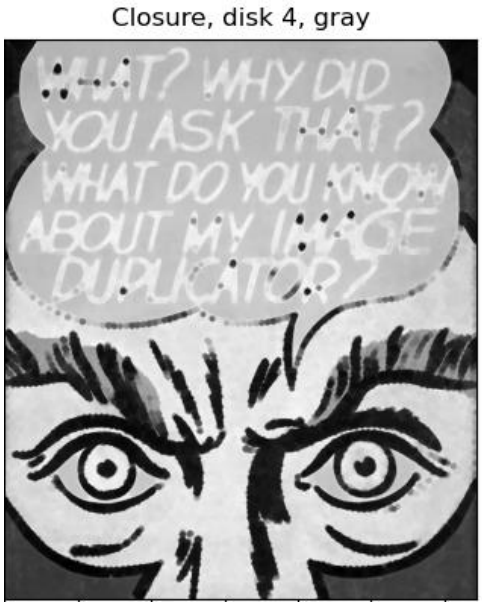
from skimage.morphology import (erosion, dilation, closing, opening, disk)

```
ste = disk(4)
A_gray_close = closing(A_gray, ste)
```

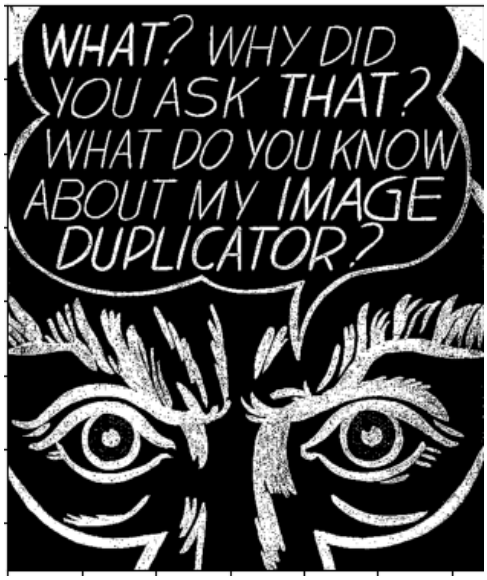
Or in one line (not counting the import):

```
A_gray_close = closing(A_gray, disk(4))
```

This looks pretty good:



We're destroying the letters, unfortunately. I notice that the letters are very dark. What if we make a binary mask by simple thresholding, keeping only very dark pixels (<20, for example):



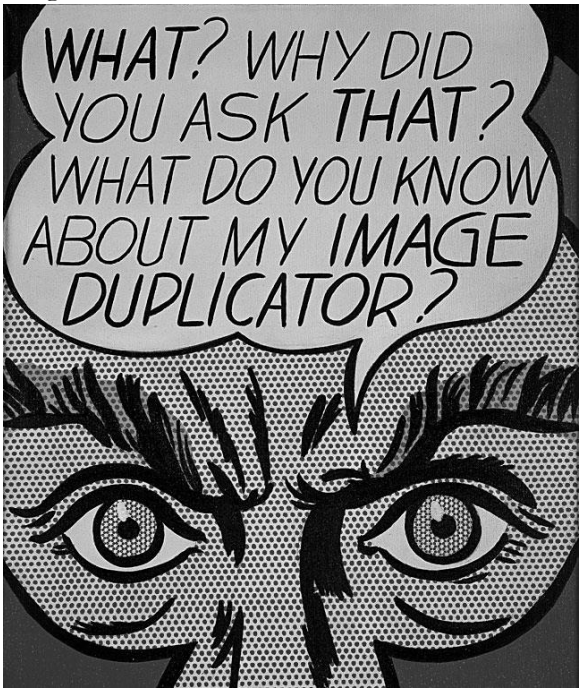
This is good, but a bit speckled. Let's close it! I use a disk of radius 2:

Finally, let's make an image that combines these. I want the original image in pixels where the threshold mask (`im_dark_close`) is 1 and the closed image (`im_gray_close`) in pixels where the threshold mask is zero:

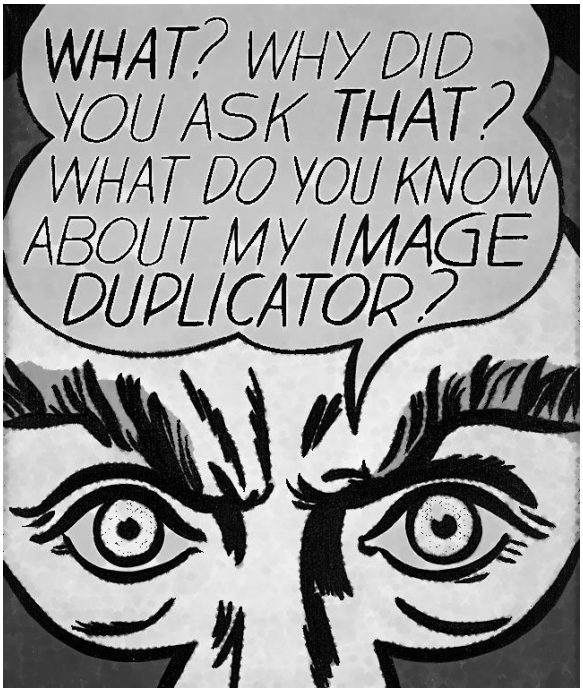
```
im_final = im_gray_close*(~im_dark_close) + im_gray*im_dark_close
```

Voila!

Original



Final



(b)

Solution

Now let's consider the color channels separately:

Let's again close the image (in all channels), but keep the closed image only at pixels that are mostly red, and high intensity. This should close the red dots while keeping the original image pixels in other regions. Specifically, our criteria will be that:

(1) the red channel is at least 50% more than average of green and blue channels

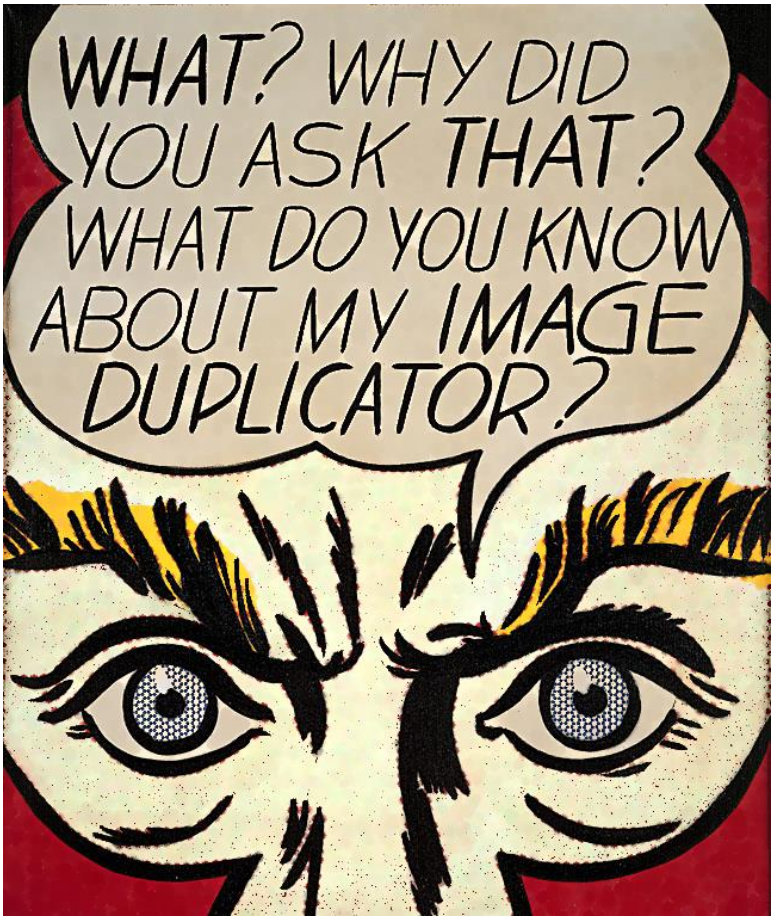
(2) the red intensity is at least 100

The closure requires closing each channel separately, unless we create a 3D structuring element. (That's fine; there are many ways to do this.)

Python

```
...
isred = np.logical_and(im[:,:,0]>th_red ,
            (im[:,:,0] > (0.75*(im[:,:,1]+im[:,:,2])to force to be "3D"
# 3 layers of this; need to force to be "3D"
isred = np.tile(isred[...,None], (1, 1, 3))


im_color_final = im_close*isred + im*~(isred) # original image in other
pixels
```

The result:



2 Segmentation by thresholding (4 pts.) ... let's revisit thresholding. ....

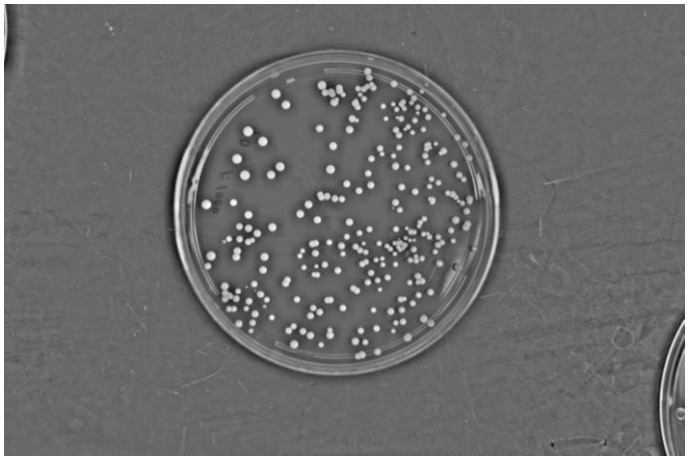
(a) (3 pts.) Apply whatever filtering seems appropriate to the yeast colony image. ... Then, threshold to make a binary image that highlights the colonies ....

Solution

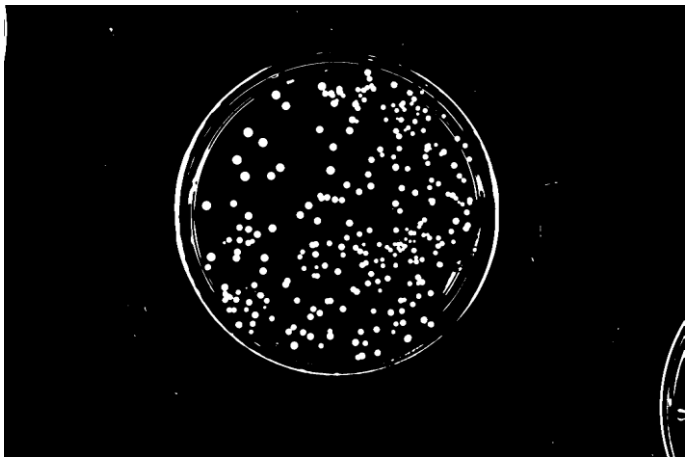
High-pass filtering removes "slowly" varying changes in intensity, and can be implemented as 1 minus a low-pass filter. We certainly want to retain the colonies, so we pick a Gaussian filter size considerably larger than the colonies; I chose 50 pixels.

Low-pass filtering smooths features smaller than the feature size. We again want to retain the colonies, but smooth over smaller noise, so we pick a Gaussian filter size considerably smaller than the colonies; I chose 4 pixels.

The result of the filtering:



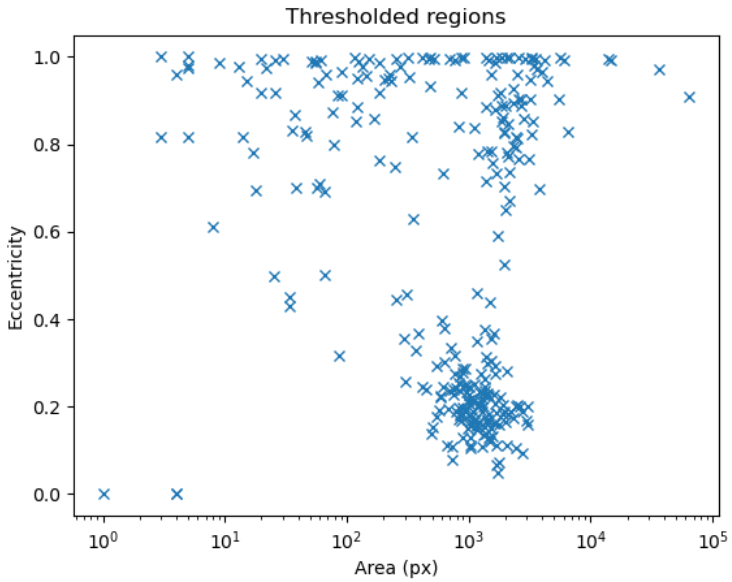
Then thresholding – Otsu's threshold looks fine:



(b) … make a scatter plot of Eccentricity vs. Area; use a logarithmic axis for area.

Solution

This is straightforward using the code provided.



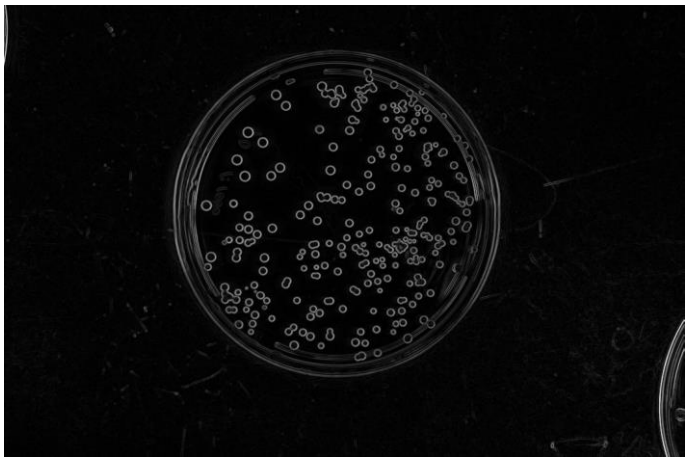
Note that there's a big "blob" of regions of similar area and low eccentricity – these are the colonies!

3 Gradient magnitude (3 pts.) Convolve the filtered (not thresholded) yeast colony image with 5x5 Sobel filters (shown below) to get the gradients in x and y. Then calculate, display, and submit the gradient magnitude (i.e. sqrt(G_x^2 + G_y^2)) image.

Solution

We convolve with Gx and Gy. In Python:

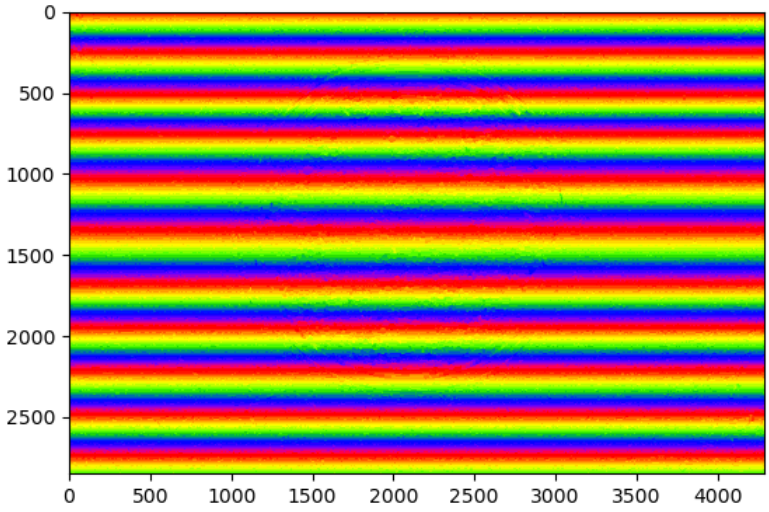
```
im_Gx = ndi.convolve(im_lp, Sobelx, mode='nearest');
im_Gy = ndi.convolve(im_lp, Sobely, mode='nearest');
im_gradmag = np.sqrt(im_Gx**2 + im_Gy**2)
```



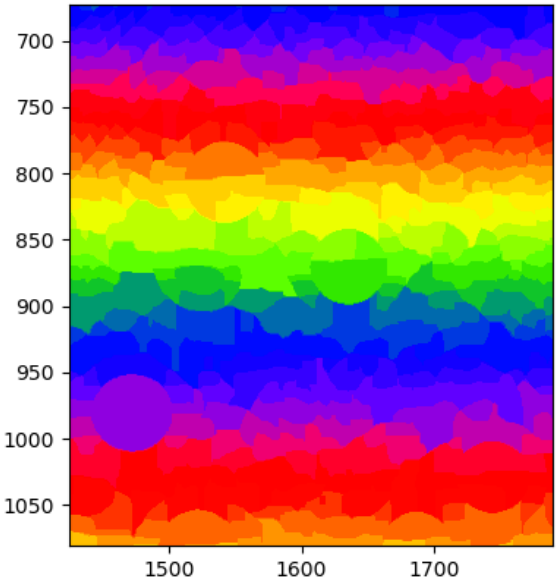
4 Watershed segmentation without markers. (1 pt.) Apply watershed segmentation to the gradient magnitude image ("`im_gradmag`….

Solution

Here it is:



Zooming in makes it more obvious how terrible the segmentation is:



+1 for something like this.

5 Watershed segmentation with markers (7 pts.)

(a) (3 pts.) Make an array that's 1 at the local maxima of the filtered (not thresholded) image of yeast colonies … and 0 elsewhere.

Solution

Even with the low-pass filtering, there are lots of local maxima. There are even a lot that are above my intensity threshold!

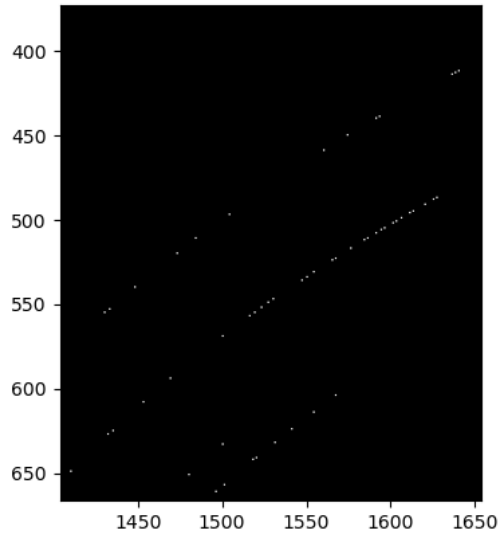
Here's some Python code using "manual" dilation

```
from skimage.morphology import (disk, dilation)
disk_radius = 3
ste = disk(disk_radius)
im_dil = dilation(im_lp, ste)
im_max_image = im_lp == im_dil
```

Here's some Python code using "peak_local_max"

```
from skimage.feature import peak_local_max

im_max_indices = peak_local_max(im_lp) # indices

im_max = np.zeros(im.shape)
im_max[im_max_indices[:,0],im_max_indices[:,1]]=1
```

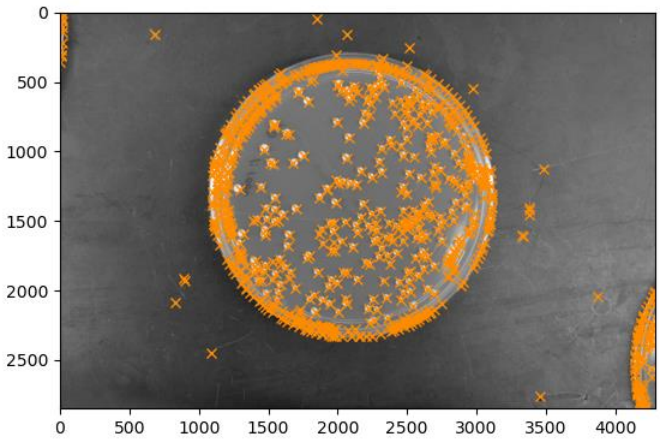
In either case the dots are too small to see, but we can zoom into the image:



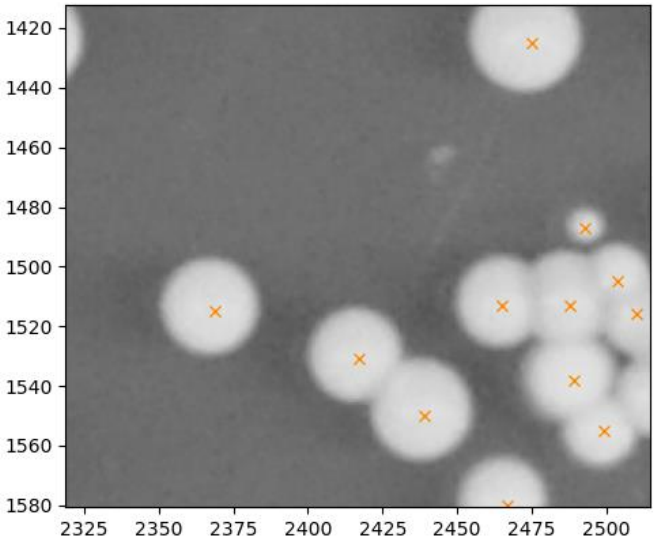

```
plt.figure()
plt.imshow(im_max, 'gray');

# Only keep above-threshold maxima
im_max = im_max*im_bw;
[max_rows, max_cols] = np.where(im_max==1)
plt.figure()
plt.imshow(im,'gray');
```

```
plt.plot(max_cols, max_rows, marker='x', color='yellow', linestyle =
'none')
```

Here they are:



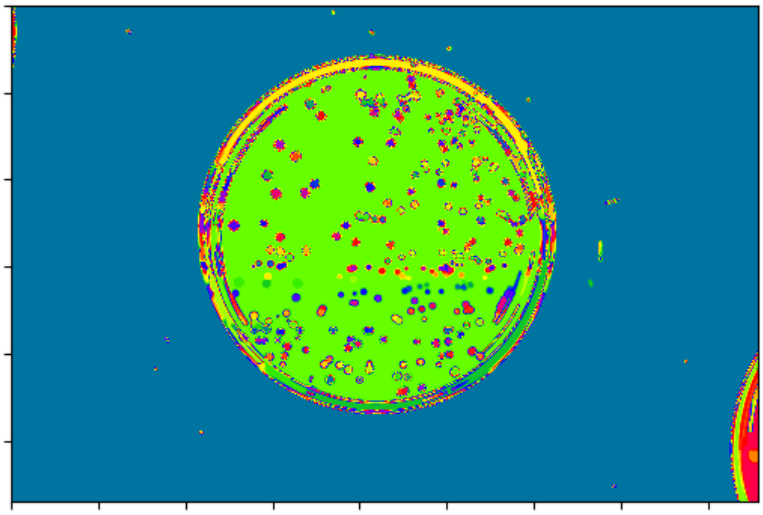
Zoomed in:



(b) Apply watershed segmentation to the gradient magnitude array, but using the above-threshold local maxima as markers. ...

Solution

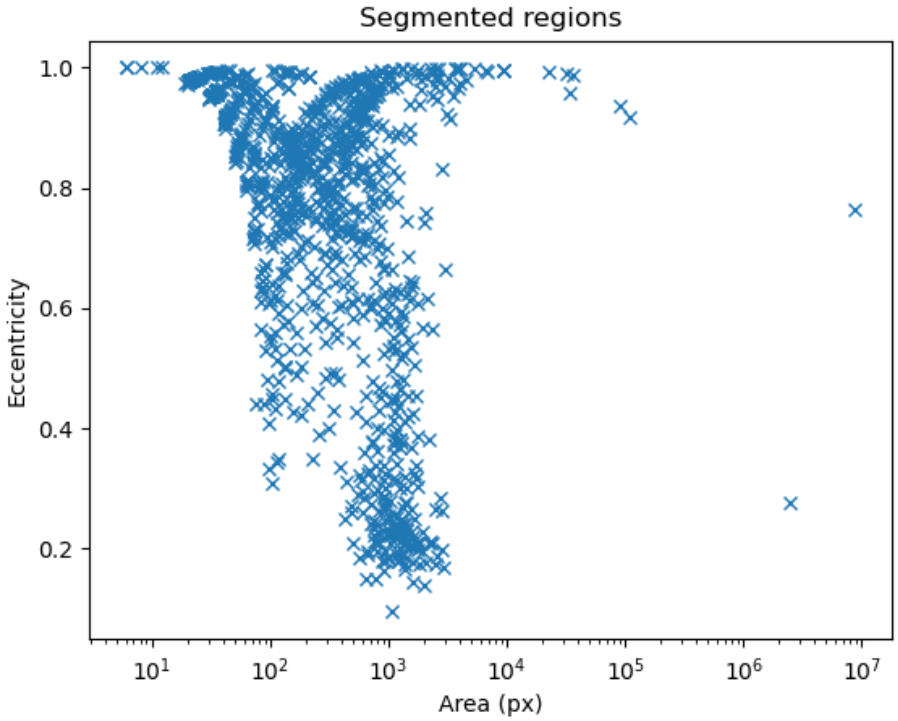
Just using the above-threshold maxima from above:



Pretty good!
There are still too many regions, mainly along the dish edges.

(c) …and make a plot of eccentricity vs. Area…

Solution



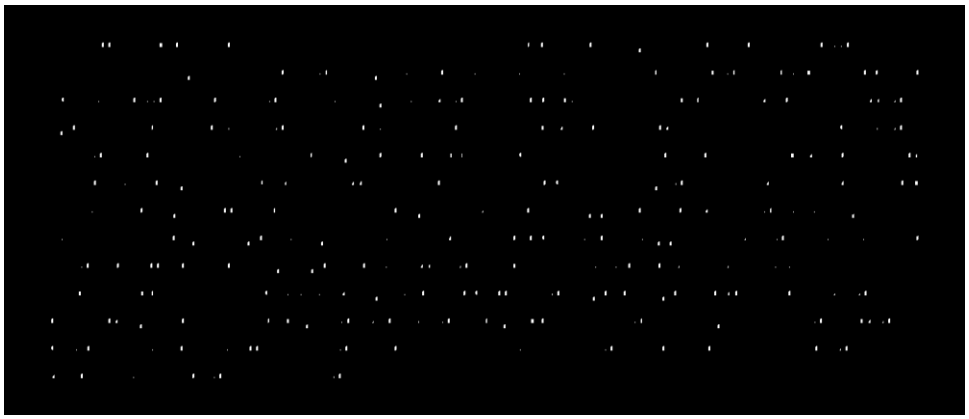
As noted above, there are still too many regions. We could, if we wanted, select those that have an area of about 1000 and eccentricity around 0.2, which seem to be the colonies. But let's wait to think about feature values...

6 [Extra Credit] Morphological Reconstruction.. (3 pts.) ....

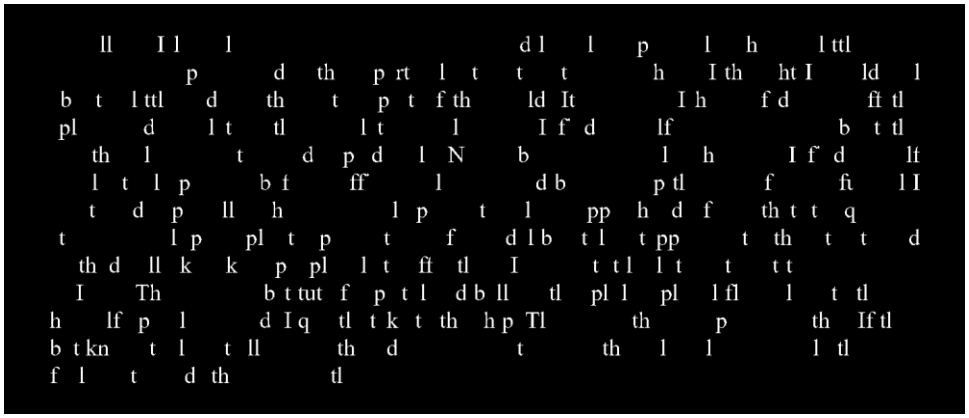
Solution

I used a structuring element that's a 19 pixel long vertical line. This works, for keeping vertical elements!

Eroded:



Reconstructed:



We see that letters containing vertical elements are reconstructed!