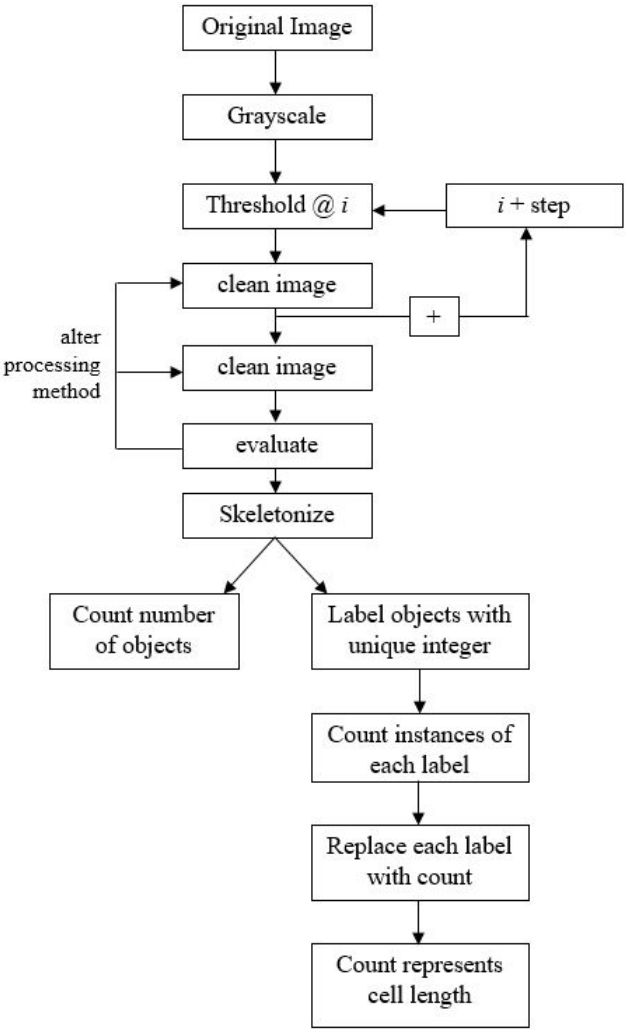
Nayiri Krzysztofowicz, Shimin Lei, Orian Churney
ECE 6782/4501: Digital Image Processing
26 September 2018

Methodology

The first task was to count the number of bacteria in the image provided, 'confocal.mat.' The image was loaded into matlab and converted to grayscale, then different threshold levels were tried to see whether the individual bacteria shapes could be captured. There seemed to be several levels of grey that represented the individual bacteria, depending on the depth. When a small threshold (~0.1) was used, the peripheral bacteria shapes would be captured, but the middle bacteria was seen as one large blob. When a larger threshold was used (~0.6), the peripheral bacteria was erased but the middle blob was separated into distinct bacteria. Using either threshold resulted in dramatically undercounting the bacteria present. As a result, we thresholded the image twice, at two distinct thresholds that would allow us to capture the individual bacteria at two levels of brightness. While some bacteria were double-counted, after overlaying the two binary images from the different thresholds, not many bacteria overlapped, so we assumed the double-counting could offset the bacteria that we were never able to isolate and count. After using the two thresholds, our bacteria count closely matched what we had counted by hand.

Next, the binary image was skeletonized to obtain the length of the bacteria. The sum of the white pixels of the skeleton image was found. This value would need to be paired with the magnification level of the bacteria image to compute the actual length of the bacteria, since the scale of the image was not provided. To count individual bacteria lengths, the bwlabel() MATLAB was applied to the skeleton image to assign each skeleton its unique integer label in the range of the number of cells. The resulting matrix was iterated over and the number of instances of each label were counted. These labels were replaced by their count number, so when the resulting matrix was displayed, the skeletons with the highest number of pixel counts were displayed brightest. In this way, we could visually see where the long or short bacteria were distributed throughout the image. The process



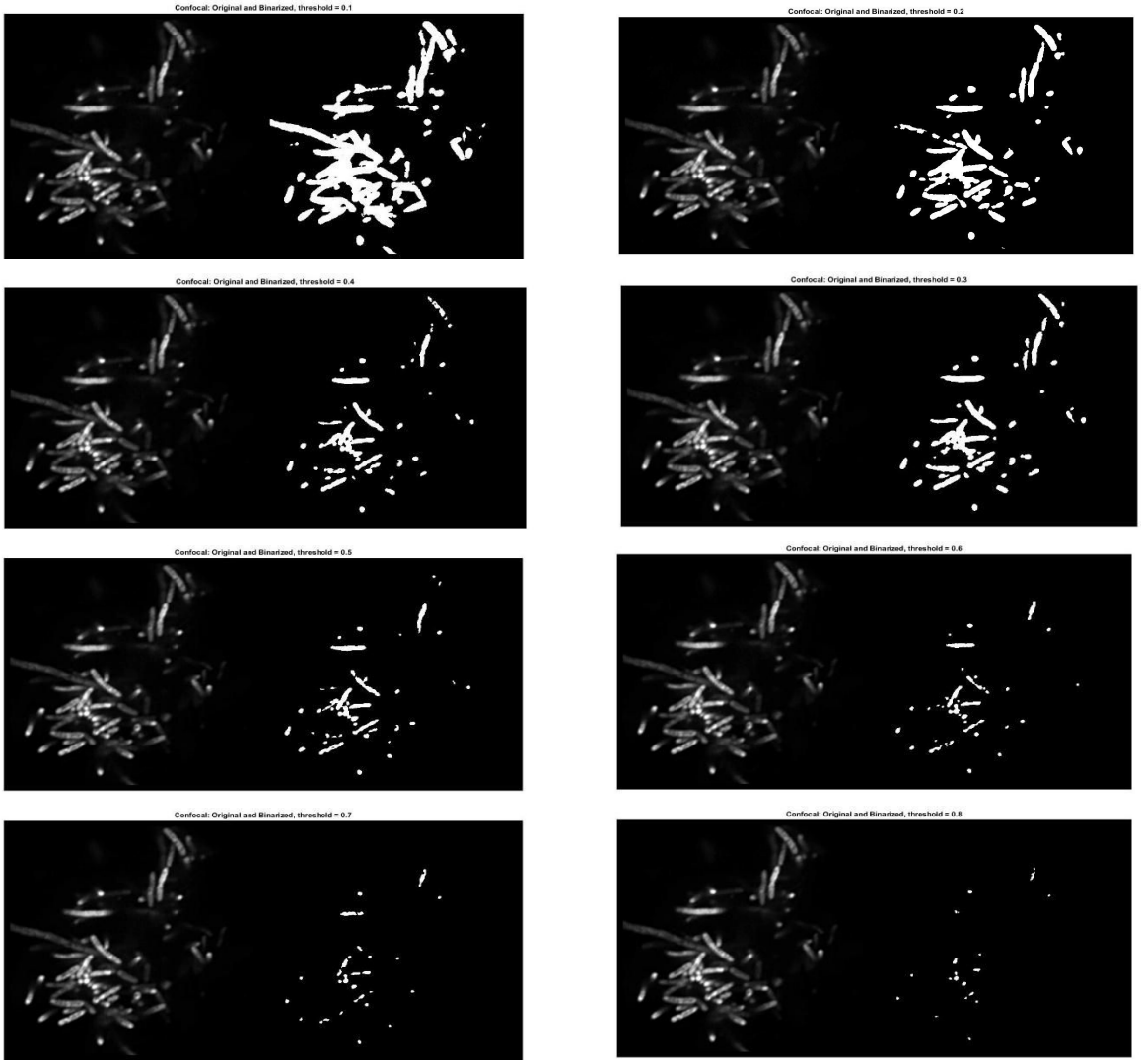
The above process was repeated for the second image, latticeLightSheet.jpg. The overall structure of the image processing method remained the same, but at every step of the process we had to visually verify that the expected results were being produced, and

modify any hard-coded values as we saw necessary. For example, for the lattice image a smaller lower bound on large object removal had to be set, as we found this produced a cleaner final binary image. Additionally, no further processing had to be done on the skeletonized image to remove branches, since there were virtually none to begin with.

Finally, for part four, the 3D.tif image was imported, and the stacks of the image were iterated through to obtain a 3D matrix. This was viewed in the MATLAB Volume Viewer app to get a better intuition of the data we were working with. The 3D slices looked similar the the latticeLightSheet.jpg image from the previous part. These 3D slices were iterated over and the same processing technique was used as before with just one image. This produced the binary image. To skeletonize the image, code online was used [1]. The bwlabeln function was used as before to label each separate skeleton, and the cell skeleton lengths counted. The final skeletonized image was displayed with the skeleton brightness varying with cell length.

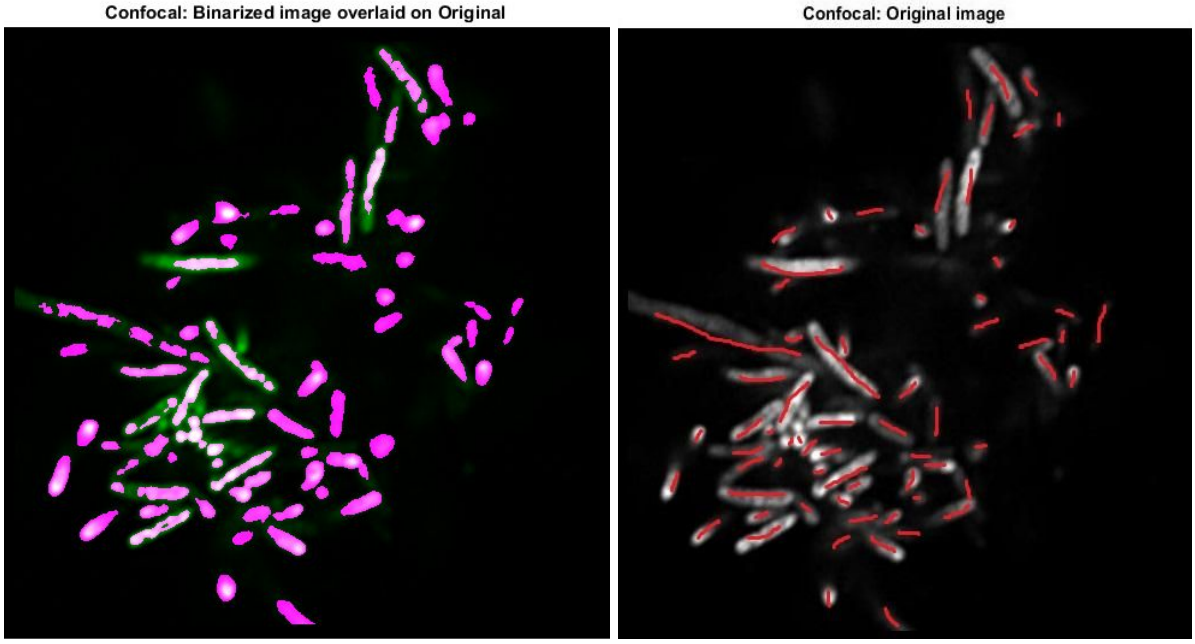
Results

Below are the bacteria thresholded from 0.1 to 0.8, in 0.1 increments. As the binary images show, each threshold value isolates different bacteria in the image.

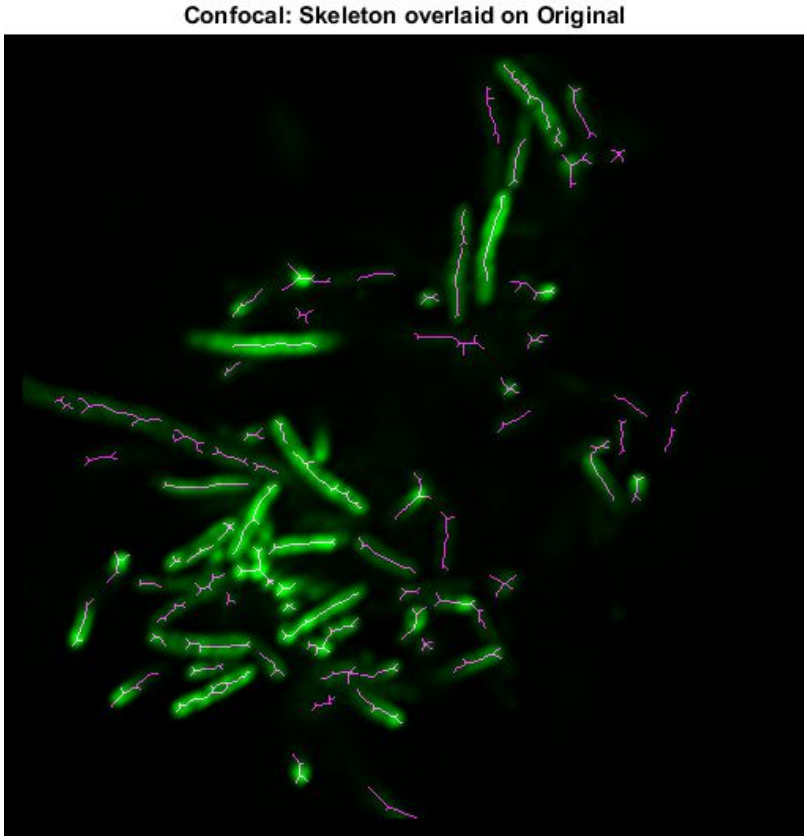


From these observations, the confocal image was thresholded from 0.1 to 1 in .001 increments, and the results minus small and large white objects were summed. In this way, as the medium-sized bacteria objects became visible in the image, they were added to the entire binary image. The results of this process are seen below, with the original and binary image displayed side-by-side, then overtop each other for comparison. The binary image (magenta) visually seemed to do an adequate job of capturing the count of the bacteria cells (green).

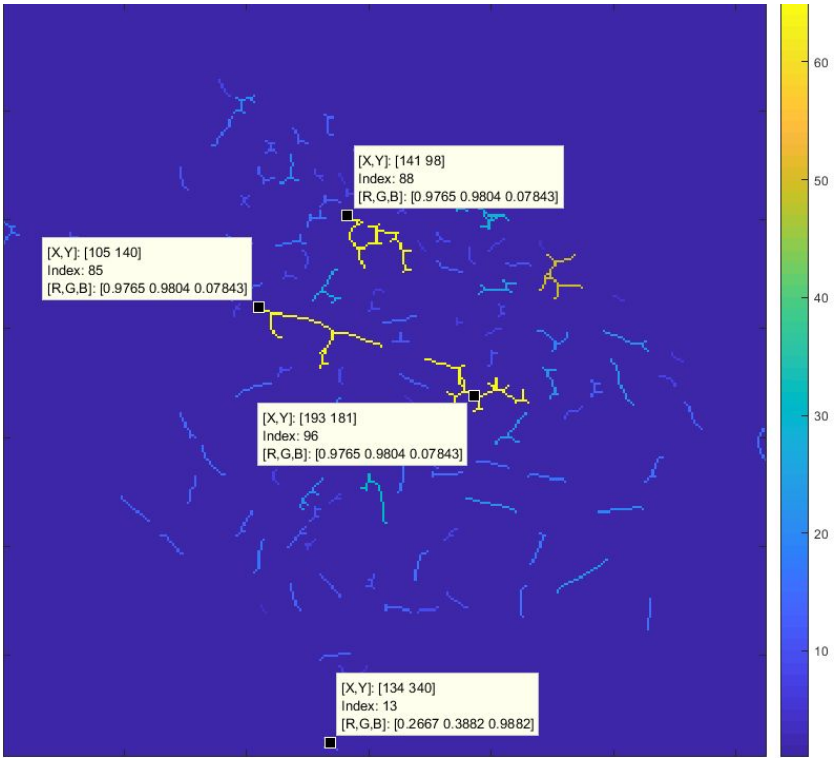
This process led to a bacteria count of 75. As a verification, the bacteria in the original image were counted, with the number totalling to around 70. The image below shows the bacteria we included in this count (red marks). The results were more accurate than we had hoped for, given how blurry and difficult to process the original image was.



To compute cell length, the binary image was skeletonized. Below is the original confocal image (green) and overlaid skeleton (magenta). The pixel count was found to be 2265, which would need to be translated to an actual distance using the microscope resolution for these photos to know the total cell length.



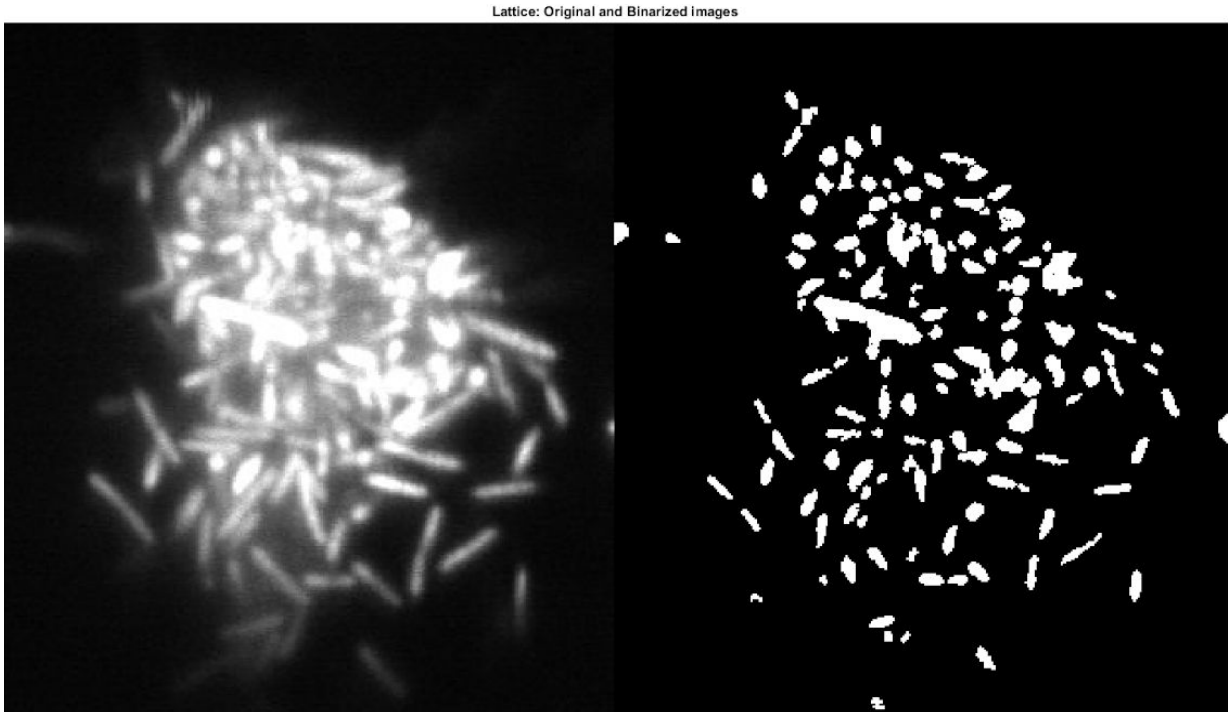
The skeleton length of each individual bacteria was found and displayed below, where blue cells are shorter length and yellow cells are longer. A few data cursors were also displayed for reference, showing that the longest bacteria were around 90 pixels long.



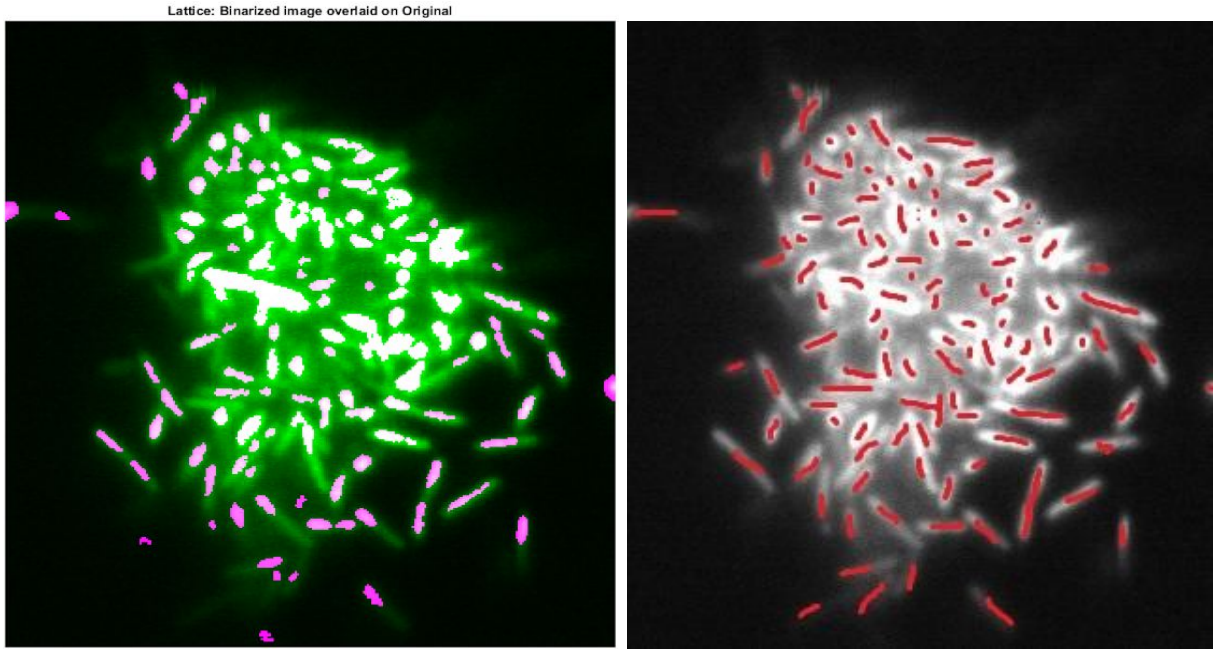
Finally, the lengths of these individual bacteria cells were counted, with longer cells displayed as bright yellow and short cells displayed in greens and blues (above). The shorter bacteria were evidently clustered in the center of the image, but our method did not do a good job of capturing the cell length of the long bacteria, since any thresholding we did either split the long bacteria or combined all the bacteria cells into one object.

Next, our image processing method was tested on the second image, 'latticeLightSheet.jpg.' The method worked somewhat smoothly for the second image, requiring only minor tweaks such as adjusting threshold values and sizes of objects to be removed. The same process of summing the lattice image thresholded at different values was followed, yielding the image below.

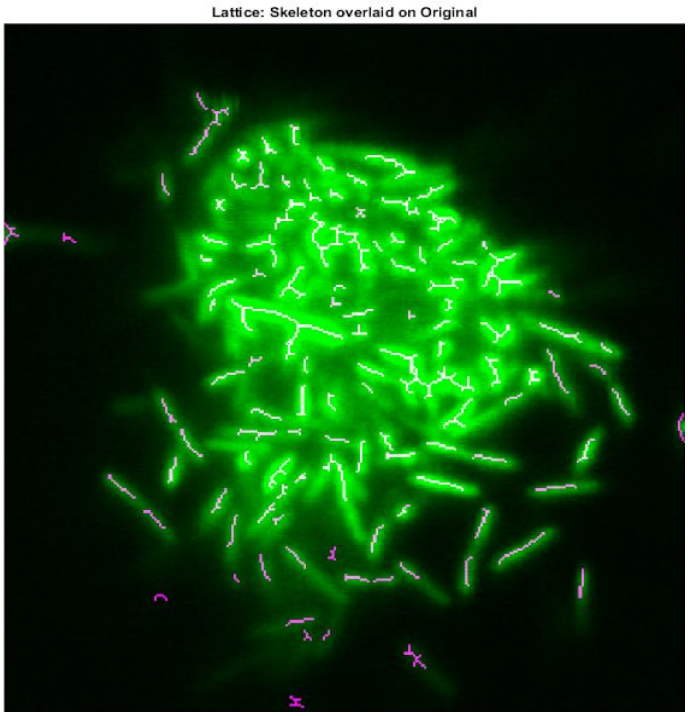
The method seemed to work even better for the new data, since it accounted for the different lighting in the image by only retaining small bright objects, which were the bacteria cells we wanted to isolate.



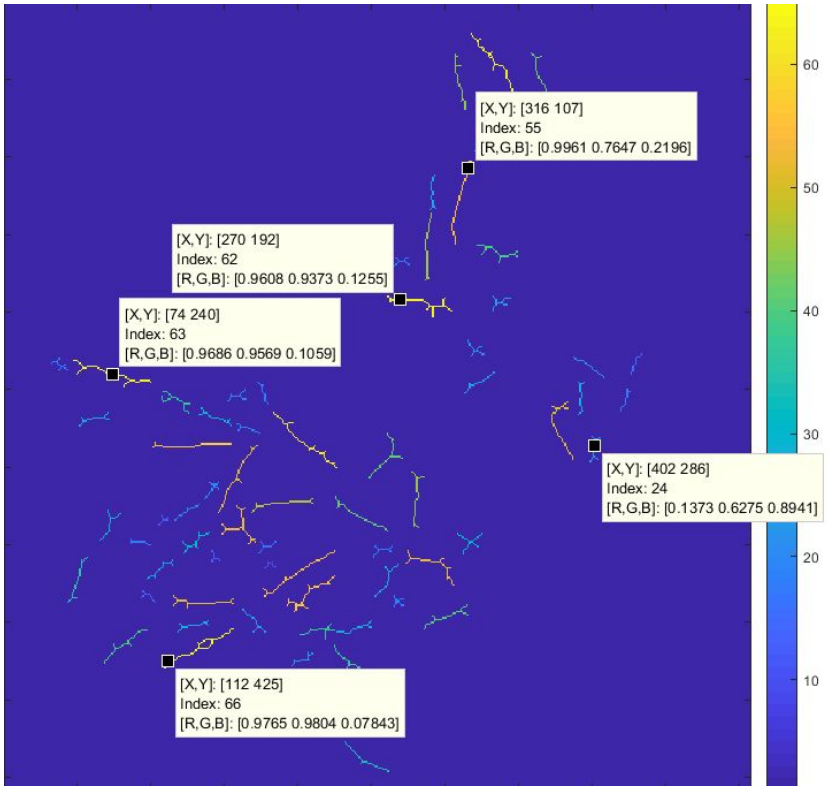
The binary image overlaid on the original is seen below. This process yielded 116 bacteria counted, whereas counting by hand we got around 130 bacteria. Again, we were surprised by how accurately our processing captured the bacteria count.



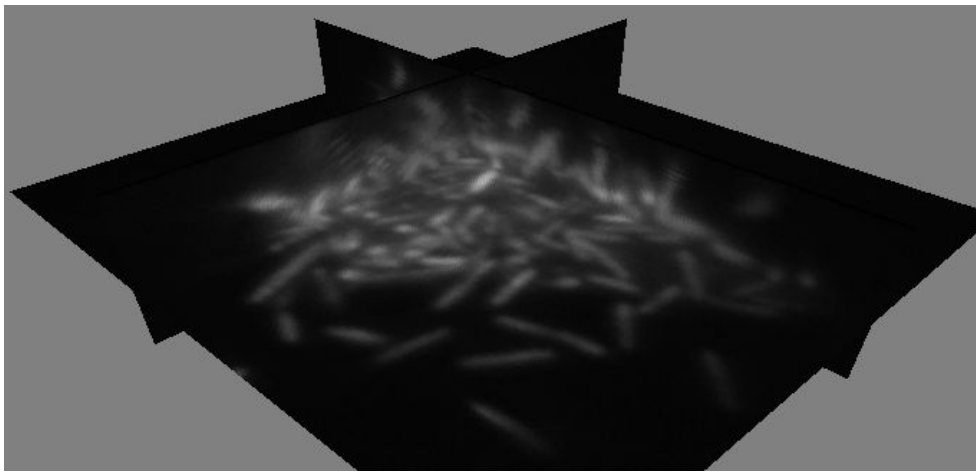
To compute cell length, the lattice image was skeletonized, shown below. The length of the skeleton was found to be 1750 pixels, which again would need the microscope resolution to scale to a real length unit.



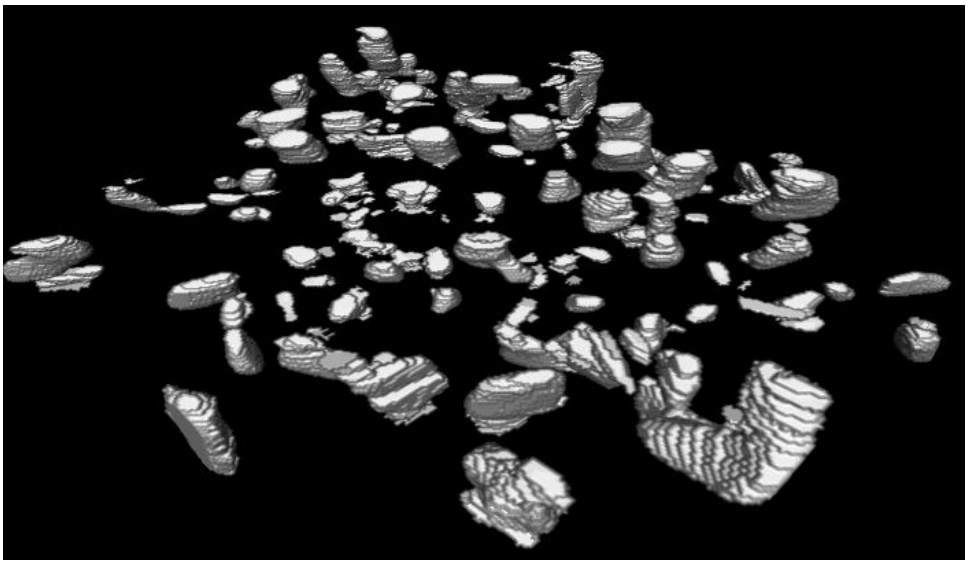
As before, the length of each skeleton was calculated and the skeleton image colored as shown below. A few sample skeleton length values are shown in the data cursor, with long skeletons being around 60 pixels, and shorter ones spanning around 20 pixels. Again, the shorter bacteria cells were generally clustered towards the center of the image.



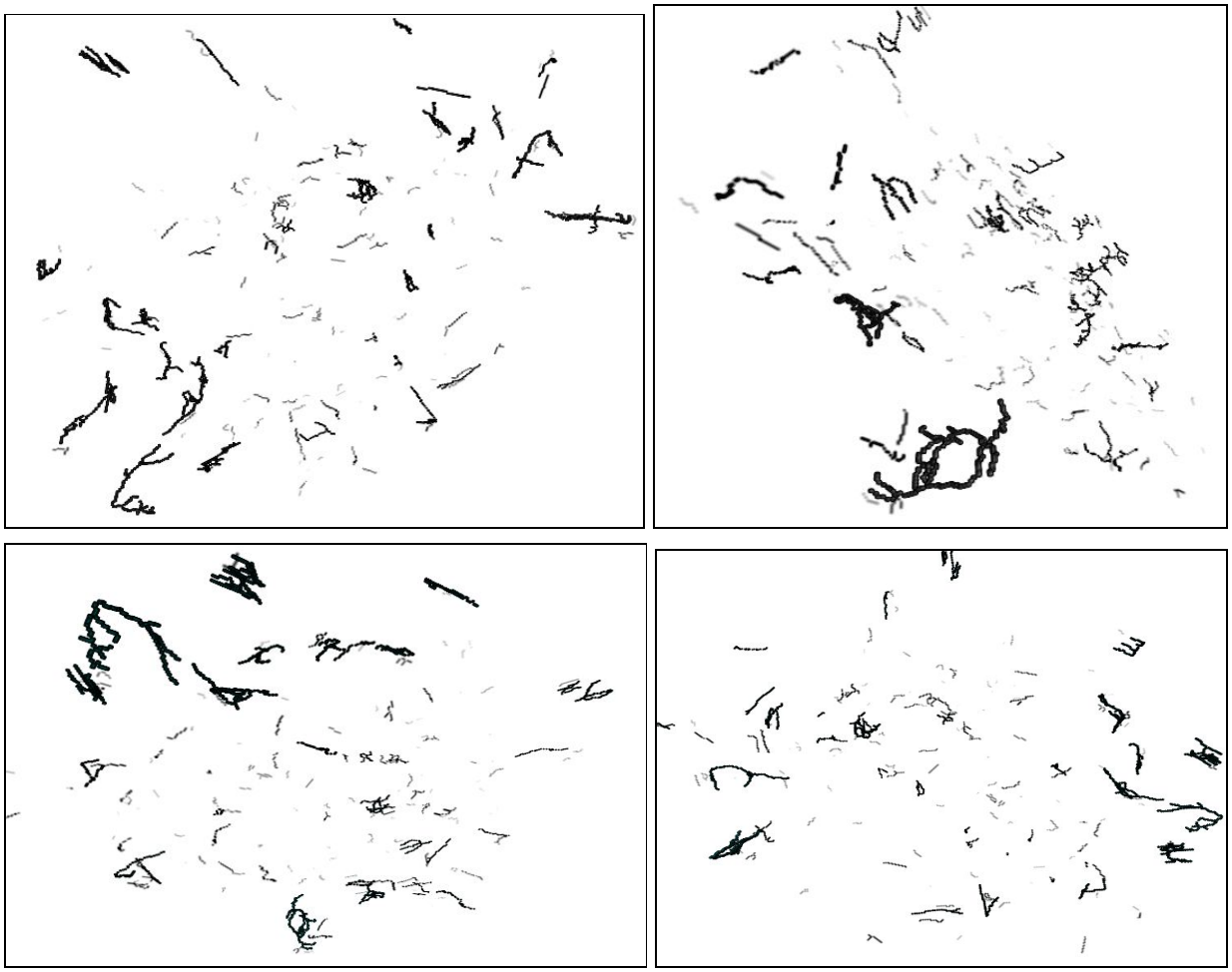
Finally, the 3D.tif image was loaded into MATLAB. Below is the volume-rendering of the 3D matrix.



Each "stack" of the 3D image was iterated through and processed as we had done before, to clean up the image. The binarized 3D lattice is seen below. Our thresholding evidently removed many of the bacteria compared to the original image, but no better process could be determined. The results of the thresholding are seen below.



A Skeleton3D function [1] was downloaded from online and the 3D matrix was skeletonized. MATLAB's bwlabeln() function was used to count the number of items in the skeleton and label each individual skeleton. This approach gave us a count of 118 cells. We suspect the processing on the image we did removed too many of the bacteria, but it was difficult to isolate the bacteria and remove the noise, so many of the bacteria were likely removed as well. The same process as before was once again used to count the number of pixels for each label, and turning those pixels into the counts. The final skeleton lengths are displayed below.



The darker greys indicate a longer cell length. As seen in the images, these long cells were distributed on the outside of the lattice, with the smaller bacteria located on the inside of the bacteria cluster.

MATLAB Code

For confocal.mat:

```
load('confocal.mat');
confocal = cluster1Processed5;
clear cluster1Processed5;

confocal=mat2gray(confocal);
figure, imshow(confocal), title('Confocal: Original image')

%% CELL COUNT
confocalBin = imbinarize(confocal,0.99);
for i=0.07:.001:1
      confocalBintemp = imbinarize(confocal,i);
      confocalBintemp = confocalBintemp & ~bwareaopen(confocalBintemp, 400);
      confocalBin = confocalBin | confocalBintemp;
end
confocalBin = imerode(confocalBin, strel('sphere',2));
confocalBin = bwareaopen(confocalBin, 20);
figure, imshowpair(confocal,confocalBin,'falsecolor'), title('Confocal:
Binarized image overlaid on Original');
figure, imshowpair(confocal,confocalBin,'montage'), title('Confocal: Original
and Binarized images');
[L,n]=bwlabel(confocalBin);    %n is total number
n

%% LENGTH
confocalBin = imbinarize(confocal,.1);
confocallength = bwmorph(confocalBin,'skel',Inf);
%figure, imshowpair(confocal,confocallength,'falsecolor'), title('skeleton')

%remove branches / noise and calculate length
%code from:
https://www.mathworks.com/matlabcentral/answers/88284-remove-the-spurious-edge-
of-skeleton

skel = confocallength;
B = bwmorph(skel, 'branchpoints');
E = bwmorph(skel, 'endpoints');
[y,x] = find(E);
B_loc = find(B);
Dmask = false(size(skel));
for k = 1:numel(x)
      D = bwdistgeodesic(skel,x(k),y(k));
      distanceToBranchPt = min(D(B_loc));
      Dmask(D < distanceToBranchPt-1) = true;
end
confocallength = skel - Dmask;
%figure, imshowpair(confocal,confocallength,'montage'), title('skeleton')
l = sum(confocallength(:)) % print red lymphocyte skeleton length
figure, imshowpair(confocal,confocallength,'falsecolor'), title('Confocal:
Skeleton overlaid on Original')
```

For latticeLightSheet.jpg:

```
lattice = imread('latticeLightSheet.jpg');
% figure, imshow(lattice), title('original')
lattice = mat2gray(lattice);

%% CELL COUNT
latticeBin = imbinarize(lattice,0.99);
for i=0.07:.01:1
      latticeBintemp = imbinarize(lattice,i);
      latticeBintemp = latticeBintemp & ~bwareaopen(latticeBintemp, 60);
      latticeBin = latticeBin | latticeBintemp;
end
latticeBin = bwareaopen(latticeBin, 10);
figure, imshowpair(lattice,latticeBin,'montage'), title('Lattice: Original and
Binarized images');
figure, imshowpair(lattice,latticeBin,'falsecolor'), title('Lattice: Binarized
image overlaid on Original');
[L,n]=bwlabel(latticeBin);     %n is total number
n           % display number of bacteria

%% LENGTH
latticeLength = bwmorph(latticeBin,'skel',Inf);
figure, imshowpair(lattice,latticeLength,'falsecolor'), title('Lattice:
Skeleton overlaid on Original')

% visually determined that additional processing of skeleton was unnecessary
l = sum(latticeLength(:)) % print red lymphocyte skeleton length
```

For 3D.tif:

```
%% LOAD ORIGINAL IMAGE
tiff_info_orig = imfinfo('3D.tif'); % return tiff structure, one element per
image
tiff_slice_orig = imread('3D.tif', 1);
tiff_stack_orig = tiff_slice_orig; % read in first image
%concatenate each successive tiff to tiff_stack
for ii = 2 : size(tiff_info_orig, 1)
      tiff_slice_orig = imread('3D.tif', ii);
      tiff_stack_orig = cat(3 , tiff_stack_orig, tiff_slice_orig);
end

%% THRESHOLD IMAGE
tiff_info = imfinfo('3D.tif'); % return tiff structure, one element per image
tiff_slice = imread('3D.tif', 1);
tiff_slice(tiff_slice<=15000) = 0;
tiff_slice(tiff_slice>15000) = 1;
tiff_stack = tiff_slice; % read in first image
%concatenate each successive tiff to tiff_stack
for ii = 2 : size(tiff_info, 1)
      tiff_slice = imread('3D.tif', ii);
```

```
      tiff_slice_mod = tiff_slice;
      tiff_slice_mod(tiff_slice_mod>0) = 0;
      for i=5000:500:25000
      temp = tiff_slice;
      temp(temp<=i) = 0;
      temp(temp>i) = 1;
      temp = medfilt2(temp, [9 9]);
      temp = temp & ~bwareaopen(temp, 200);
      temp = bwareaopen(temp,20);
      tiff_slice_mod = tiff_slice_mod | temp;
      end
      tiff_slice = tiff_slice_mod;
      tiff_stack = cat(3 , tiff_stack, tiff_slice);
end

%% SKELETONIZE AND COMPUTE LENGTHS
skel3D = Skeleton3D(tiff_stack);
[L3D,n3D] = bwlabeln(skel3D);
[x,y,z] = size(tiff_stack);
cellLengths = zeros(n3D,1);
for i=1:x
      for j=1:y
      for k=1:z
            if L3D(i,j,k) > 0
            cellLengths(L3D(i,j,k)) = cellLengths(L3D(i,j,k)) + 2;
            end
      end
      end
end
skel3DLengths = zeros(x,y,z);
for i=1:x
      for j=1:y
      for k=1:z
            if L3D(i,j,k) > 0
                  skel3DLengths(i,j,k) = cellLengths(L3D(i,j,k));
            end
      end
      end
end
 x = skel3DLengths(:,:,1);
 y = skel3DLengths(:,:,2);
 z = skel3DLengths(:,:,3);
 figure; plot3(x,y,z);
```

Contributions

Nayiri Krzysztofowicz, Shimin Lei and Orian Churney collaborated on the code and report. Method approach was generated together and coding split between group members.

References

[1] Skeleton 3D function. https://www.mathworks.com/matlabcentral/fileexchange/43400-skeleton3d

[2] Remove branches from skeleton.
https://www.mathworks.com/matlabcentral/answers/88284-remove-the-spurious-edge-of-skeleton