Cell Identification using Morphological Operations

Warren Pinto

Computer Science

Ryerson University

April 10, 2014

**Abstract**

*Cell count and identification of the cell wall, nucleus, and nucleus’ wall are computed in a given image using morphological and the canny edge detector. MATLAB code is tested on an image and is illustrated with sample images.*

**1 Introduction**

Counting and identifying cells in a picture can be a very long task if done by hand. Hence, it would be easier to automate counting and identifying cells when given an input image. Morphological Operations can be used to identify the number of cells that are in a given picture. Morphological operations can also be used to identify the boundaries or the “cell wall” of a cell. Some useful morphological operations that can be used are *thicken, shrink, close, bridge and erode.*

The *thicken* morphological operation can be used to thicken or increase the dimensions of an object, thereby giving it a bigger representation of the object. Conversely, the *shrink* morphological operation can be used to shrink an object to a desirable dimension. *Close* morphological operation allows us to fill “holes” that an object may have, allowing for proper shrink procedures to be made. The *bridge* operation is useful such that any small or one pixel gaps between two objects are connected. However to get the desirable results from morphological operations, it is suggested to convert an image to binary using *thresholding* functions. For example, a thresholding function may look like:

G(x,y) = {255 f(x,y) > T,

{0 f(x,y) < T.

,where T is the threshold point, f(x,y) is the input image at pixel x,y and, G(x,y) is the output function at pixel x,y. Using these image processing techniques, we will compute the cell count and cell identification of a given image.

**2 Image preprocessing**

The given image must first be converted to its grayscale image using MATLAB’s *rgb2gray* function as this would make it easier to compute the binary image. To compute the binary image, the thresholding function is used. The threshold function that can be used is as follows:

G(x,y) = {255 f(x,y) < 100,

{0 f(x,y) >= 100.

The thresholding function gives us the new binary image that we are going to work. We can select the right threshold level T from looking at the image histogram. The intensity at which the most of the pixels lie is usually the background of the cell or its cell membrane. The intensity at which we can find the cell nucleus is where the pixels are grouped together in the darker intensities of the histogram. From **Figure 1**, we notice that the most of the cell nucleus’ pixels are located below 100 intensity.

**3 Performing Morphological Operations**

From the image preprocessing step, we are given the binary image, which we will use to perform the morphological operations, as shown in **Figure 2**. The first morphological operation that is used is the *close*. When the binary image was computed in the previous image preprocessing step, holes and gaps were created. Thus to solve this problem, the close operation was used to fill in the gaps.

Once the gaps have been filled, we use the *thicken* operation on the resulting close operation. The thicken operation is used (10x) such that it gives us a good round object to work within the next step. This thicken operation will also be useful later on in the cell identification phase. Once the thicken operation is complete, we can then shrink the blobs into a single pixel.

**4** **Cell Identification phase**

Because each of the blobs now shrunk to a pixel, we can easily compute the number of cells (white pixels in 4th image of **Figure 2**) there are in the image. The also allows us to mark where in the image each cell nucleus is located and its corresponding identification number (**Figure 3**).

The thicken operation that was used in the previous phase can also be used to identify the cell walls of the cells that have a nucleus. Because the thicken operation was used 10x, this allowed us to create a “boundary” effect with the other thickened nucleus’. The thickened blocks will represent the membrane of the cells.

Now that a boundary effect was created, we can use the canny edge detector to detect edges, which in our case, the edges represent the cell walls (**Figure 4**). To get the boundary of the cell nucleus, we use the *erode* morphological function. The first closed binary image is eroded twice and subtracted from the original binary close image. This gives us the boundary of the cell nucleus. The resulting image is applied to the original image to give us a better view of the effects of the program on the actual image (**Figure 5**).

**5 Discussion**

Although this program can count and identify the cells, there are some cases where the program might fail. Since this program assumes that a cell will have a clear visible nucleus present, it can easily pinpoint where the cell is on the image. However in reality, there are several different types of cells in the world and can thus decrease the accuracy of the program as this program may only work for a specific subset of cell images.

Another flaw that we witness is the placement of the microscope pointer in the image. This can damage the results of the cell count of the image as it has darker intensity pixel values and may be considered as a cell for the program. Thus, a clean image is needed to properly count and classify the cells.

The application of counting points on can image can also be used of other purposes as well. For example, the program’s pinpointing and counting can be used to pinpoint the stars of a sky (**Figures 6 - 8**).

**6 Conclusion**

Morphological Operations are simple and powerful tools that can help us identify various properties of an image or object, and can effectively be used to automate identification procedures of a given image. Morphological operations can work with a variety of images, such as the NASA sky image used above.

**9 Figures**

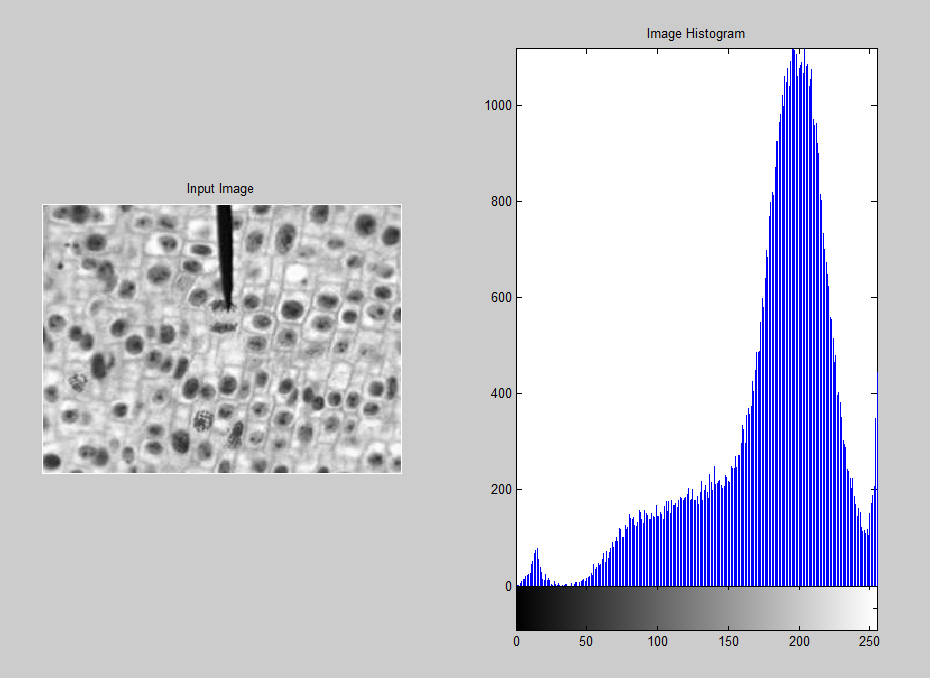


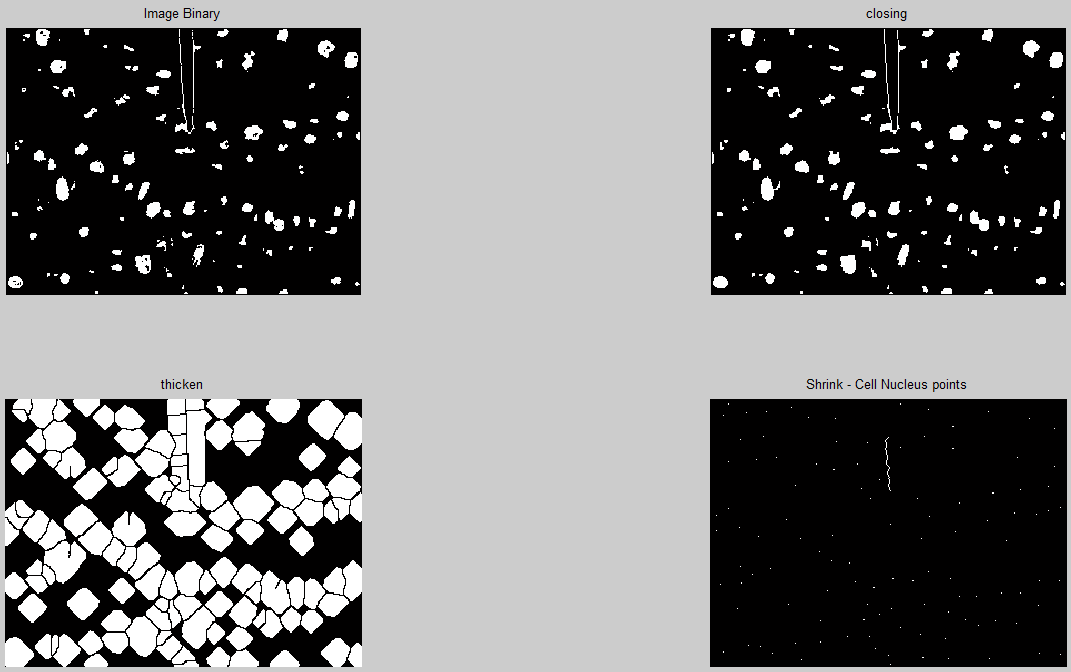
Figure – The input image and its Histogram. 

Figure – Morphological Operations

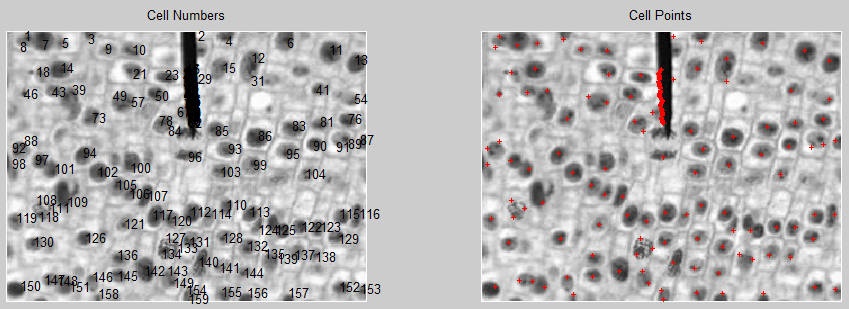


Figure – Cell count and position on the image

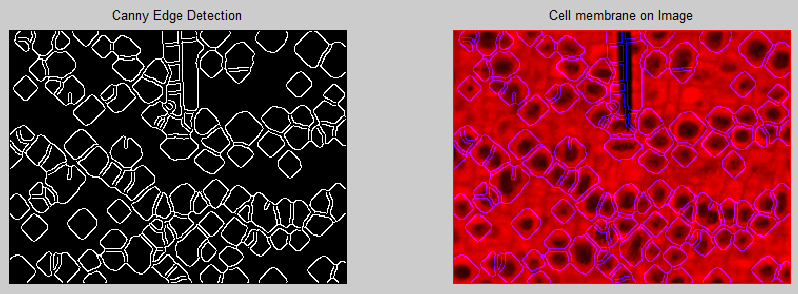


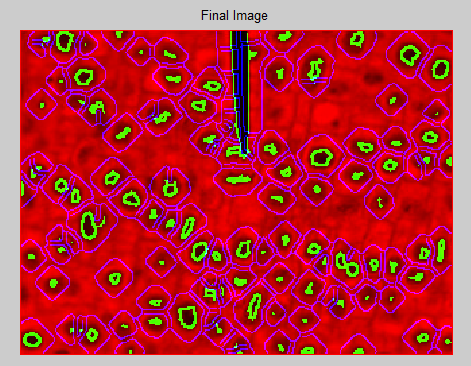
Figure  – Identification of cell membrane using Canny Edge Detection

Figure – The final resulting image, Cell Membrane and Cell Nucleus

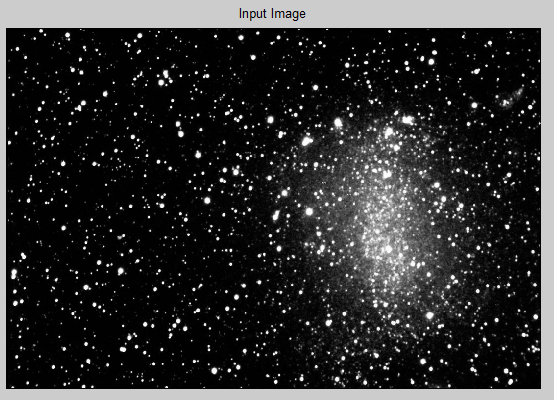


Figure – NASA input image

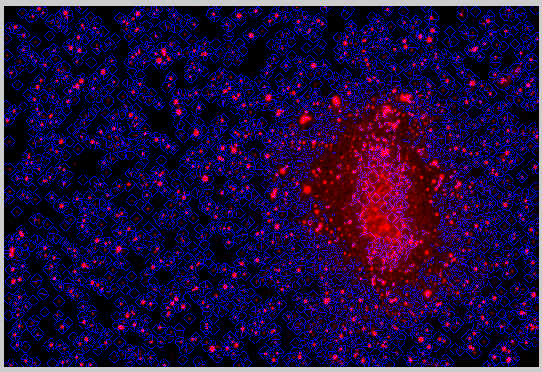


Figure – Stars and its boundaries

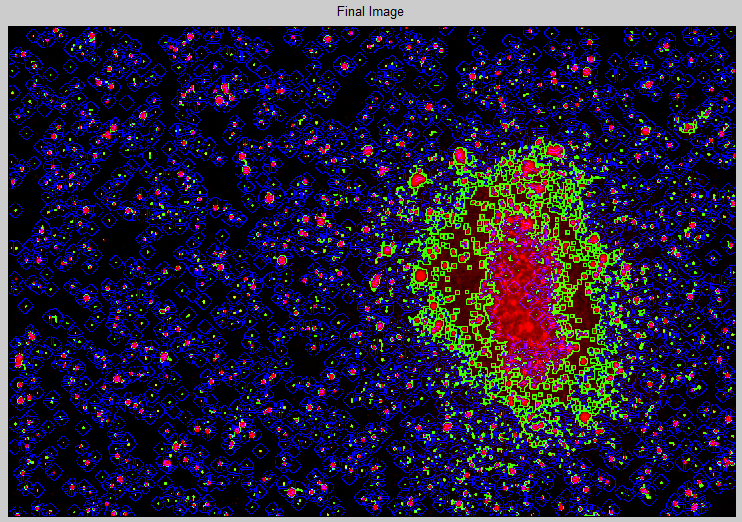


Figure – Final result of the NASA input image.

**8 Code**

clear all;

close all;

%read in cell image

im = imread('images\cells.jpg');

im = rgb2gray(im);

subplot(1,2,1); imshow(im);

subplot(1,2,2); imhist(im);

[row col] = size(im);

im\_binary = zeros(row, col);

%apply thresholding to isolate cell nucleus

for i=1:row

for j=1:col

if(im(i, j) > 50 && im(i, j) < 100)

im\_binary(i, j) = 255;

elseif(im(i, j) <= 50) %apply thresholding to remove microscope pointer

im\_binary(i, j) = 0;

else

im\_binary(i, j) = 0;

end

end

end

figure; subplot(2,2,1); imshow(im\_binary); title('Image Binary');

%perform morphological operations:

%CLOSE - fill gaps in the binary image:

im\_binary = bwmorph(im\_binary, 'close');

subplot(2,2,2); imshow(im\_binary); title('closing');

%THICKEN - thicken the blobs of the resulting binary image. Will

%be used later to indentify cell walls:

im\_binary\_thinken = bwmorph(im\_binary, 'thick', 10);

subplot(2,2,3); imshow(im\_binary\_thinken); title('thicken');

%SHRINK - shrink each blob to a pixel length:

im\_binary\_1px = bwmorph(im\_binary\_thinken, 'shrink',20);

subplot(2,2,4); imshow(im\_binary\_1px); title('Shrink - Cell Nucleus points');

figure; subplot(1,2,1); imshow(im); hold on; title('Cell Numbers');

cellCount = 0;

for i=1:row

for j=1:col

if(im\_binary\_1px(i, j) > 0)

cellCount = cellCount + 1;

text(j, i, sprintf('%d', cellCount), 'Units', 'data')

end

end

end

subplot(1,2,2); imshow(im); hold on; title('Cell Points');

cellCount = 0;

for i=1:row

for j=1:col

if(im\_binary\_1px(i, j) > 0)

cellCount = cellCount + 1;

plot(j,i, 'r+', 'MarkerSize', 3);

end

end

end

%part2 thincken cell walls or blobs

[row col] = size(im);

im\_binary2 = zeros(row, col);

im\_binary2rgb = zeros(row, col,3);

%use canny edge detector to find edges of the blobs.

im\_binary2 = edge(im\_binary\_thinken, 'canny');

figure; subplot(1,2,1); imshow(im\_binary2); title('Canny Edge Detection');

for i=1:row

for j=1:col

if(im\_binary2(i, j) > 0)

im(i,j,3)=255;

end

end

end

subplot(1,2,2); imshow(im); title('Cell membrane on Image');

%part3 cell nucleus identification

im\_binary3 = bwmorph(im\_binary, 'erode',2);

im\_binary3 = im\_binary - im\_binary3;

im\_binary3rgb = zeros(row, col,3);

for i=1:row

for j=1:col

if(im\_binary3(i, j) > 0)

im(i,j,2)=255;

end

end

end

figure; imshow(im); title('Final Image');