**ASSIGNMENT 1 REPORT**

**Suchismitha Vedala**

1. Resample

Given an image and the factor by which the image has to be scaled along the x and y directions and the interpolation method

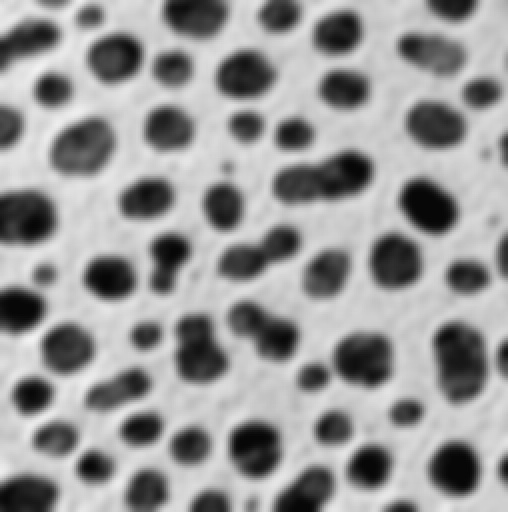
1. Nearest Neighbor:

In the nearest neighbor interpolation we first determine the size of the new image which will have the size of the original image scaled by the factor given. We then create an array mage of the new size and fill it with zeros.

For every pixel of the new image , we assign the pixel value of the old image closest to the current image. This is done by following: we divide every pixel coordinates of the old image with fx and fy along the width and height directions and find the intensity value of the old image with these new coordinates and assign this value to the original pixel coordinates of the new image.

This process is repeated until all pixel values of new image are assigned.

Below is an image of performing nearest\_neighbour interpolation on cells.png image with fx=2 and fy=2



1. Bilinear

In the Bilinear interpolation we first determine the size of the new image which will have the size of the original image scaled by the factor given. We then create an array mage of the new size and fill it with zeros similarly like in a.

However for bilinear interpolation, the intensity value at a pixel is computed by computing the linear interpolation twice around the surrounding pixels.

For every given pixel coordinate i,j in the new image, we first compute the surrounding coordinates.

We divide each pixel value with fx and fy respectively.

The four points are calculated as follows:

1. We use floor to calculate lower bound points since floor returns the lower value after division.(x1,y1)
2. We use ceil to calculate upper bound points since ceil returns the upper bound after division(x2,y2)

Cases to consider:

If (x1==x2)or (y2==y1)

This case gives division by zero, here we make y2=y1+1 or x2=x1+1

If(x2==width of image) or (y2==height of image)

This case gives integer out of bounds. So we make

X2=w-1

And x1=x2-1

Similarly y2=h-1

Y1=y2-1

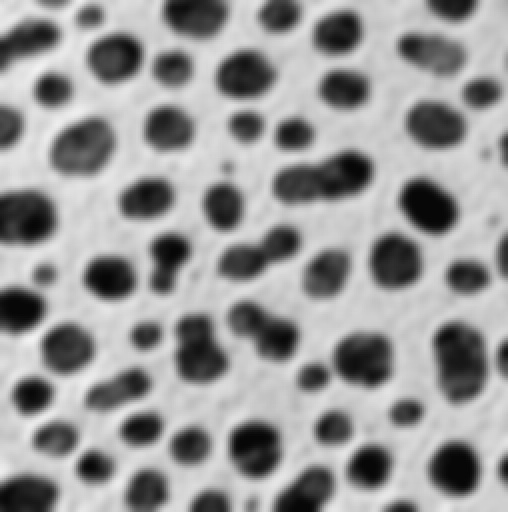
We find 4 points(x1,y1),(x2,y1),(x1,y2),(x2,y2)

Then we compute the linear interpolation between (x1,y1),(x2,y1) and (x1,y2),(x2,y2)

Finally we interpolate the intensity value between these two values and assign the value to the pixel coordinates in the new image.

This process is repeated until all pixel values of new image are assigned.

Below is an image of performing bilinear interpolation on cells.png image with fx=2 and fy=2



Linear interpolation:

This method is called by the bilinear interpolation. For given (x1,y1),(x2,y1) with intensity I1 and I2, the intensity I at point x,y is given by

I=(((x2-x)/(x2-x1))\*I1)+(((x-x1)/ x2-x1))\*)\*I2)

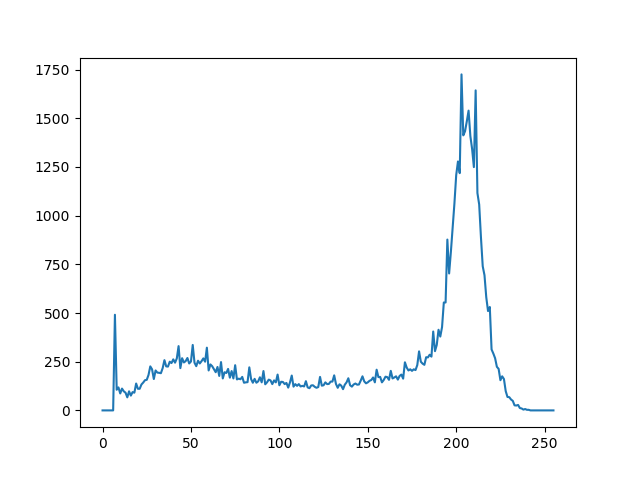
2.Resize

Given an image , we binaries the image, i.e make it a array of 0s and 255s.

1. Compute Histogram:

Read the image as gray scale image. For each pixel value, we increment the hist array with the counts of the intensity values of the image. This is then plotted using matplot.

Below is the histogram plotted



Optimal Threshold:

Optimal threshold is calculated as follows:

1.Find the initial mean from the histogram.

2. Find the average of the expected values from 0 to initial mean.

3. Find the average of the expected values from initial mean to end.

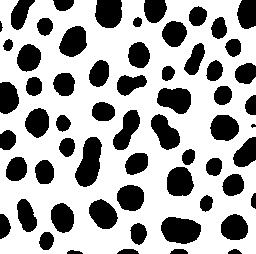
4. optimal threshold is the mean of the two.

Repeat steps 2-4 until the threshold becomes constant. We return that value as our optimal threshold.

Binary image:

For every pixel in the image, if the intensity at that pixel is greater than or equal to optimal we assign the value 255 else 0. This gives us binary image.

Binary image of cells.png



1. Blob coloring:

Since the given output image has a reverse visualization of binary(white and black), we first inverse the given binary image ,i.e making 0’s 255 and 255’s 0. In our image, we need to find all those regions which are different, i.e the white spots. Therefore we check for only those pixels whose value is not 0. We perform blob coloring algrorithm and assign region numbers to each pixels. All pixels with similar number fall into one region. We return the region, list of all the pixels with that region number.

1. Cell Counting:

The area of the regions is the number of pixels present in the region, i.e 10 pixels in a region: area=10

The centroid of regions is found by taking all the x and y coordinates in lists,

X\_centroid=sum(x\_list)/length(x\_list)

Y\_centroid=sum(y\_list)/length(y\_list)

Region Mark:

We now mark the centroid points on the image using opencv and also mark the area, region number along it.

Below is the image of marked regions.

