**Project Assignment 2 Report (Group 1)**

Patrick Ropp (propp), Easwaran Ramamurthy (eramamur), Siddharth Gurdasani (sgurdasa), Aniket Rao (**arao????**)

1. **Implementation of Gaussian Filters**

**PART 1:**

The aim of question 1.1 was to implement a Gaussian filter in MATLAB with varying standard deviation sigma as user input. The code for this question is given below and also submitted in q1part1.m.

% Implementation of a Gaussian Filter.

%Reading Image.

I = imread('image1.tiff');

%Asking user for input of sigma.

sigma = input('Please Enter the value of sigma:');

%Assuming size of gaussain matrix as 15 by 15.

hsize = [15 15];

%Creating a two-dimensional Gaussian filter h.

h = fspecial('gaussian', hsize , sigma);

%Convolving the 2D image with the gaussian filter.

I2 = conv2(double(I),double(h), ‘same');

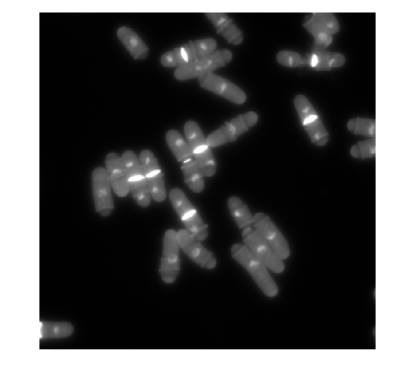
I2 = double(I2);

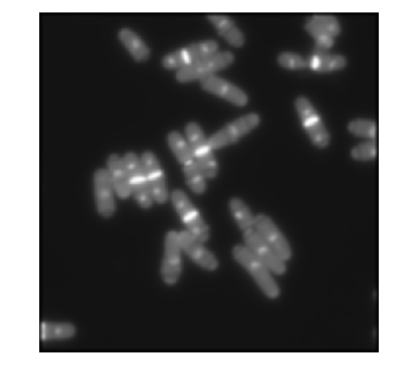
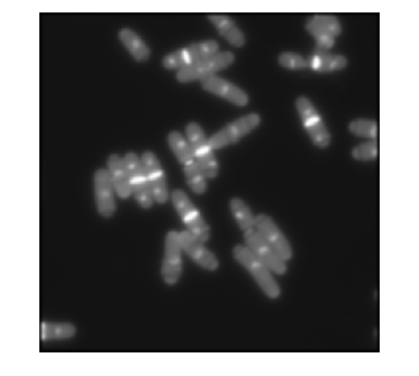
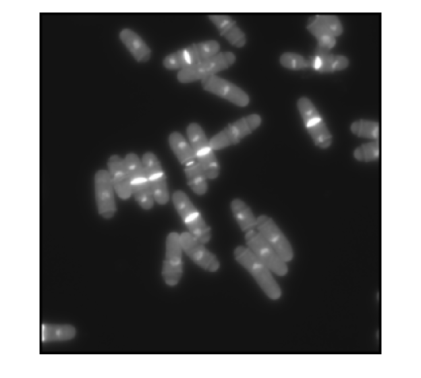
% Displaying the original image, then after getting sigma input, displaying

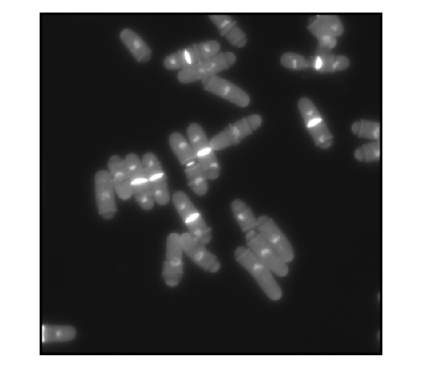
% filtered image.

figure, imshow(I,[])

figure, imshow(I2,[])

The original image is shown below:

The images below are the ones with sigma = 1,2,5,7 respectively.



As sigma increases, the images get blurrier. A Gaussian filter is essentially a low pass filter. Thus, it removes noise but also removes detail. Large sigma means a wider Gaussian filter that in turn means greater smoothening. Thus, as sigma goes from 1 to 7, a greater blur is seen as greater amount of detail is removed.

**PART 2:**

The aim of question 1.2 was to calculate image derivates. The original image used was the same as question 1.1. The code is given below and also submitted in file q1part2.m.

% Implementation of image derivatives.

%Reading Image.

I = imread('image1.tiff');

%Asking user for input of sigma.

sigma = input('Please Enter the value of sigma:');

%Assuming size of gaussain matrix as 15 by 15.

hsize = [15 15];

%Creating a two-dimensional Gaussian filter h.

h = fspecial('gaussian', hsize , sigma);

%hx is gradient along x-axis direction and hy is gradient along y-axis

%direction

[hx,hy] = gradient(h);

%Convolving the 2D image with the gaussian filter in the x direction.

I2 = conv2(double(I),double(hx),’same’);

I2 = double(I2);

%Convolving the 2D image with the gaussian filter in the y direction.

I3 = conv2(double(I),double(hy));

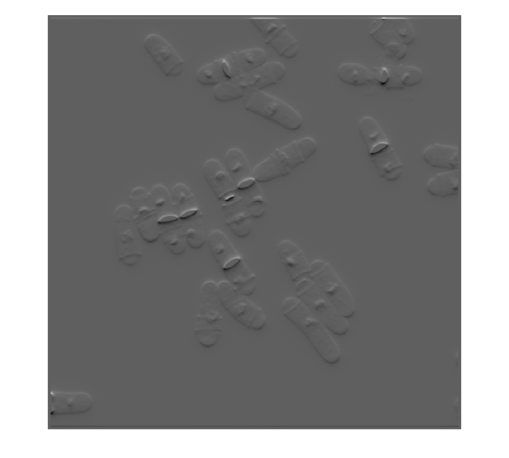
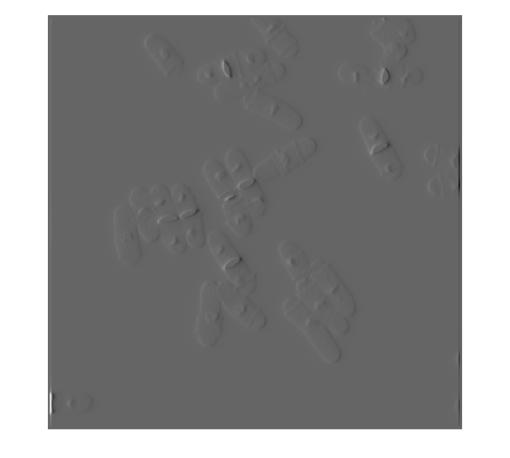
I3 = double(I3);

%Displaying original image, followed by derivative images.

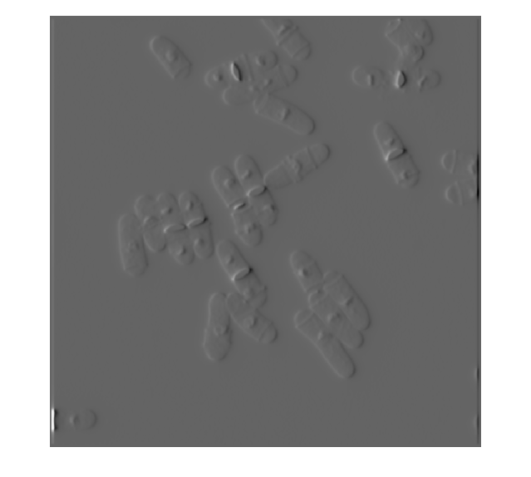
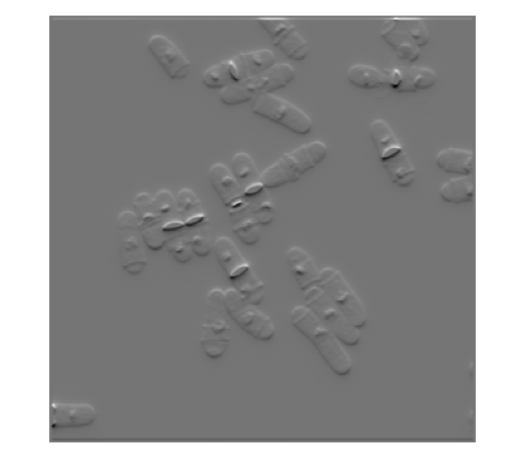
figure, imshow(I,[])

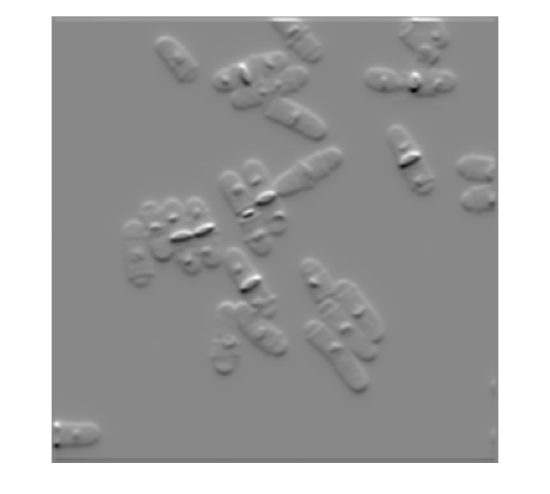
figure, imshow(I2,[])

figure, imshow(I3,[]);

For sigma =1 , the derivative in the horizontal direction and derivative in the vertical direction are shown below respectively.

Similarly for sigma = 2,



And for sigma = 5 ,

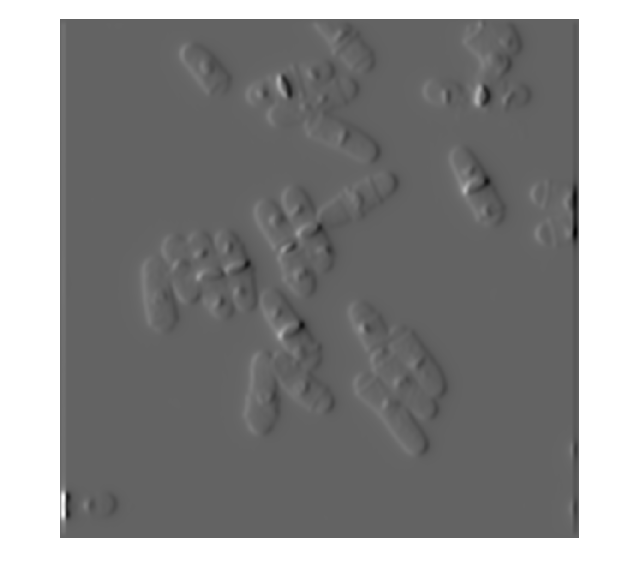


Image gradients measure the change in intensity levels of an image. For the images on the left, as derivatives are taken along horizontal direction, any change in intensity level along the horizontal direction can be seen prominently. The vertical bright bands are more prominent on the images on the left because these represent a sudden change in intensity along the x-direction. Conversely, for the images on the right, the derivatives are taken along the vertical direction and the horizontal bright bands are more prominent as the filter looks for sudden changes in intensity in the y-direction.

**PART 3:**

For part 3, we implemented the non-recursive anisotropic Gaussian Convolution filter given in Geusebroek et al., 2003, IEEE Trans. Image Processing as follows. The code is also submitted in file q1part3AnisotropicFilter.m.

function [ gt ] = q1part3AnisotropicFilter( imageFile, sigmaU, sigmaV, theta )

%function to apply an anisotropic filter to the given image and return the

%filtered image

% imageFile - image file to be filtered

% sigmaU - lateral direction sigma (horizontal)

% sigmaV - longitudinal direction sigma (vertical)

% theta - orientation angle in degrees

% gt - image matrix that is filtered along the xt axis corresponding to

% given theta

% reading image file

I = imread(imageFile);

% Calculating sigmaX, tanPsi and other parameters of the gaussians

% Variables are named according to the paper Geusebroek et al., 2003, IEEE

% Transactions on Image Processing

% converting theta to radians

theta = degtorad(theta);

sigmaX = abs(sigmaU\*sigmaV/sqrt(sigmaV^2\*cos(theta)^2+sigmaU^2\*sin(theta)^2));

tanPsi = ((sigmaV^2)\*(cos(theta)^2)+(sigmaU^2)\*(sin(theta)^2))/((sigmaU^2-sigmaV^2)\*cos(theta)\*sin(theta));

psi = atan(tanPsi);

sigmaPsi = abs((1/sin(psi))\*(sqrt(sigmaV^2\*cos(theta)^2+sigmaU^2\*sin(theta)^2)));

% assuming filter size in x direction is 15 pixels

N = 15;

% linear space for the gaussian

x = linspace(-floor(N/2),floor(N/2), N+1);

% x axis 1D gaussian filter

wX = exp(x.^2/(-2\*sigmaX^2))/(sqrt(2\*pi)\*sigmaX);

% initializing the output image of x-axis filtering

gx = zeros(size(I));

%height of image

h = size(I,1);

%filtering using x-axis filter

%this is a simple convolution of each row with the Gaussian filter and so

%we only loop through the rows and convolute them individually.

for y2=1:h,

gx(y2,:) = conv(double(I(y2,:)'),double(wX), 'same')';

end

%initializing variables for t-axis filtering

%filter size in t direction

M = 15;

%width of image

w = size(I,2);

%linear space along t-axis

t = linspace(0,floor(M/2), floor(M/2)+1);

%gaussian filter along t-axis

wT = exp(t.^2/(-2\*sigmaPsi^2))/(sqrt(2\*pi)\*sigmaPsi);

% initializing output image matrix to zeros

gt = zeros(size(I));

% interpolation coefficient set to 0.5 to give equal importance to pixels

% on both sides of a given pixel

a=0.5;

%mean

mu = abs(tanPsi);

%individually applying the convolution to each pixel

for y2=1:h,

for x2=1:w,

%adding effect of the pixel itself

gt(y2,x2) = wT(1)\*gx(y2,x2);

%adding up terms in the summation

for j=2:floor(M/2)+1,

innerSum = 0;

%checking boundaries and adding accordingly (equivalent to zero

%padding)

if and(floor(x2-j/mu)>0,y2-j>0)

innerSum = innerSum+a\*gx(y2-j,floor(x2-j/mu));

end

if and(floor(x2+j/mu)<=w,y2+j<=h)

innerSum = innerSum + a\*gx(y2+j,floor(x2+j/mu));

end

if and(floor(x2-j/mu)-1>0,y2-j>0)

innerSum = innerSum + (1-a)\*gx(y2-j,floor(x2-j/mu)-1);

end

if and(floor(x2+j/mu)+1<=w,y2+j<=h)

innerSum = innerSum + (1-a)\*gx(y2+j,floor(x2+j/mu)+1);

end

%accumulating the sum

gt(y2,x2) = gt(y2,x2)+wT(j)\*(innerSum);

end

end

end

end

The figures obtained at different orientations at in longitudinal direction and in lateral direction are given below. These are in increasing order of angle from left to right with angles = 30, 60, 90, 120 and 150 degrees respectively.

anisotropic_30degrees.tiffanisotropic_60degrees.tiffanisotropic_90degrees.tiff

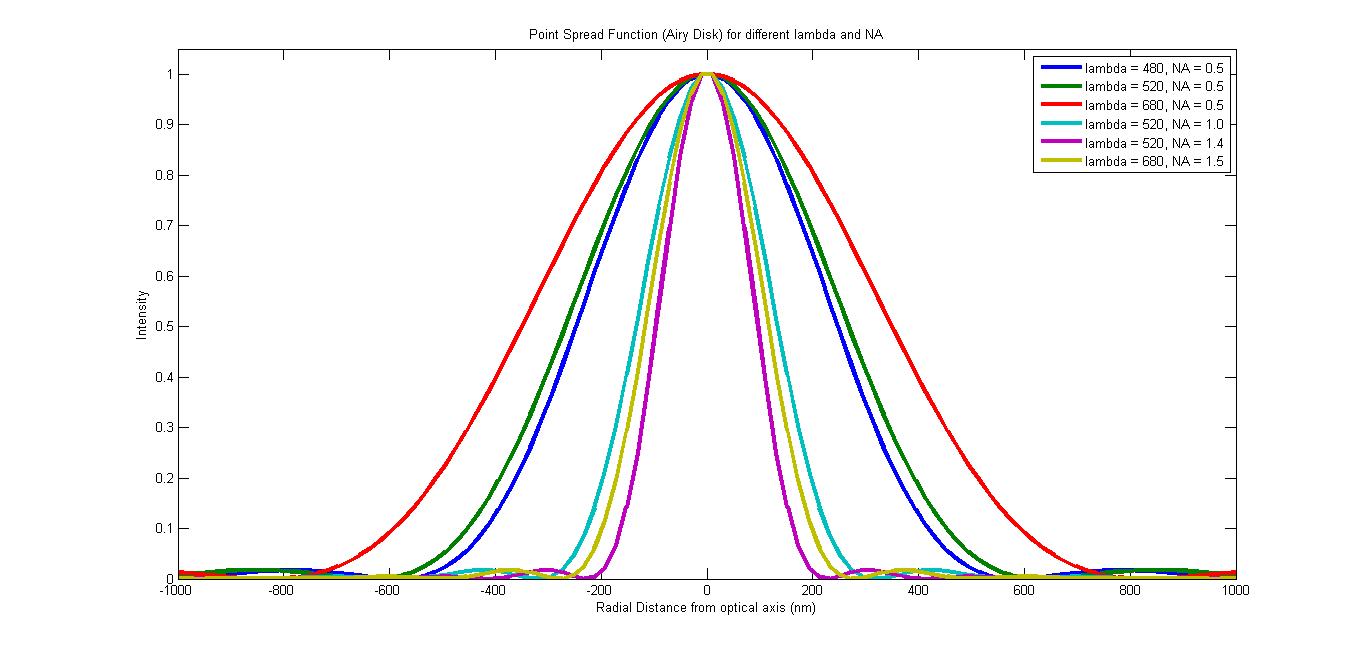
anisotropic_120degrees.tiffanisotropic_150degrees.tiff

On further analysis, we can see that the filters at 30 degrees and 150 degrees result in the same image output. Also, the filters at 60 degrees and 120 degrees result in the same image output. The image filtered at 90 degrees is equal to the image filtered in part 1 of this question 1. Further, we can see a slight tilt in the bright spots when comparing the 30 degree filtered image with the 60 degree filtered image and also when comparing these two with the 90 degree filtered image.

1. **(put report for Q2 here)**
2. **Fitting Gaussian to an Airy Disk**

**PART 1:**

The plot showing the airy disk for all combinations of and NA is shown below:



Formula used for calculation of airy disk function at a given distance from optical axis:

where is a Bessel function of first kind with order .

Table showing variation of airy disk radius corresponding to the wavelength and NA given:

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No.** | **(nm)** | **NA** | **Approximate Airy Disk radius (nm)** |
| 1 | 480 | 0.5 | 586 |
| 2 | 520 | 0.5 | 635 |
| 3 | 680 | 0.5 | 830 |
| 4 | 520 | 1.0 | 317 |
| 5 | 520 | 1.4 | 227 |
| 6 | 680 | 1.5 | 277 |

**NOTE:** The approximate airy disk radius was found out by trying to find the lowest at which the airy disk function was close to . This was done in MATLAB and the approximate airy disk radii given are our best guesses from the data we had.

We can see that for a fixed NA, the radius of the airy disk is directly proportional to the wavelength. This can be seen from observations 1-3 in the table where the NA is fixed at 0.5 and is increasing. As we increase , we see that the radius increases proportionally.

We can also see that for a fixed , the radius of the airy disk is inversely proportional to the numerical aperture NA. This can be seen from observations 2,4 and 5 where is fixed at 520 nm and NA is increasing. We see in this case that the airy disk radius is inversely proportional to NA and decreases with increasing NA.

From the plot below we can see that is fully correlated with the airy disk radius.

**Summary:** The airy disk radius increases through observations 1-3 as the wavelength increases at fixed NA. The airy disk radius decreases through observations 2 and 4-5 as the NA increases at fixed . In observation 6, although the NA increases compared to observation 5, the huge increase in wavelength from 520 nm to 680 nm makes the radius slightly larger than for observation 5.

**NOTE:** Code submitted in file: q3\_plotAiryDisk.m. This code generates the individual point spread function plots for a particular given and NA. This contains a function that also takes as input whether to fit a Gaussian or not as the third argument (required for part 2).

Figure showing all 6 airy disks in one graph submitted in file: q3part1plot.fig and q3part1plot.jpg. This figure is also shown above.

**SIDE NOTE:** Code to generate the combined airy disk plots (shown above) for the 6 observations is submitted in files: q3part1\_makeFigure.m and q3part1\_getAiryDiskFunction.m

**PART 2:**

Individual plots showing fitted Gaussians for all 6 observations are shown below. The and NA used are mentioned in the title of the figures.

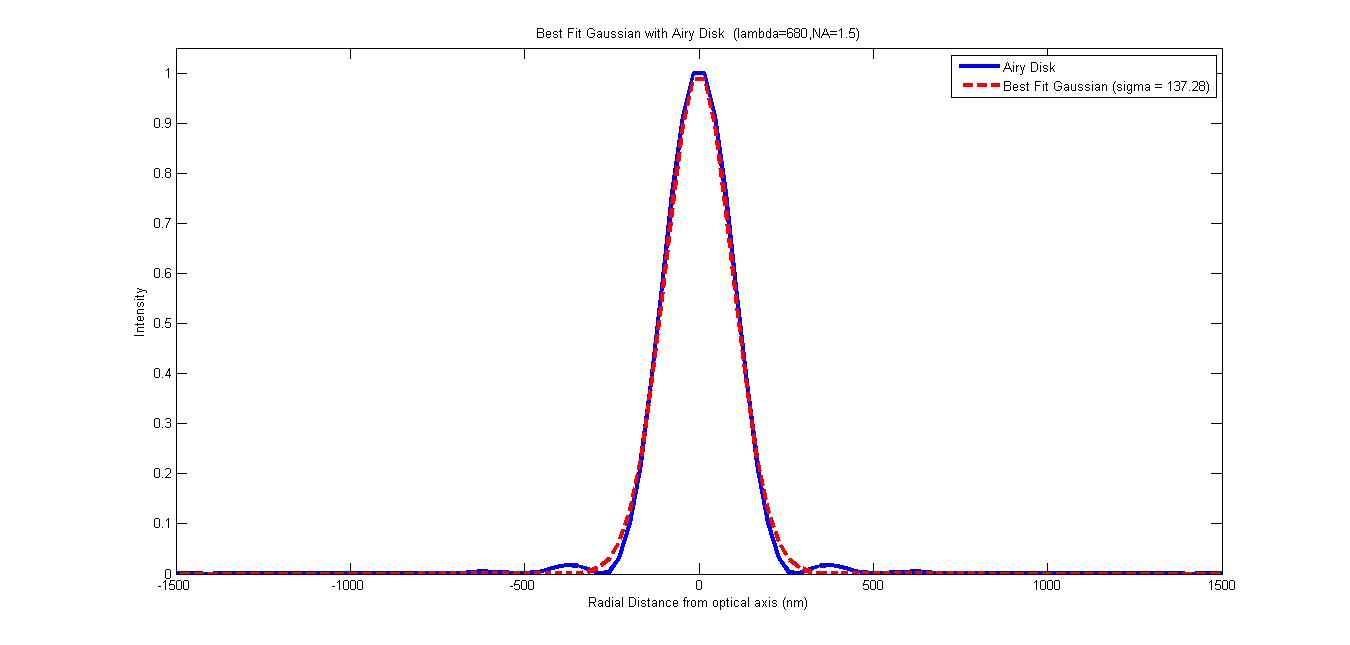
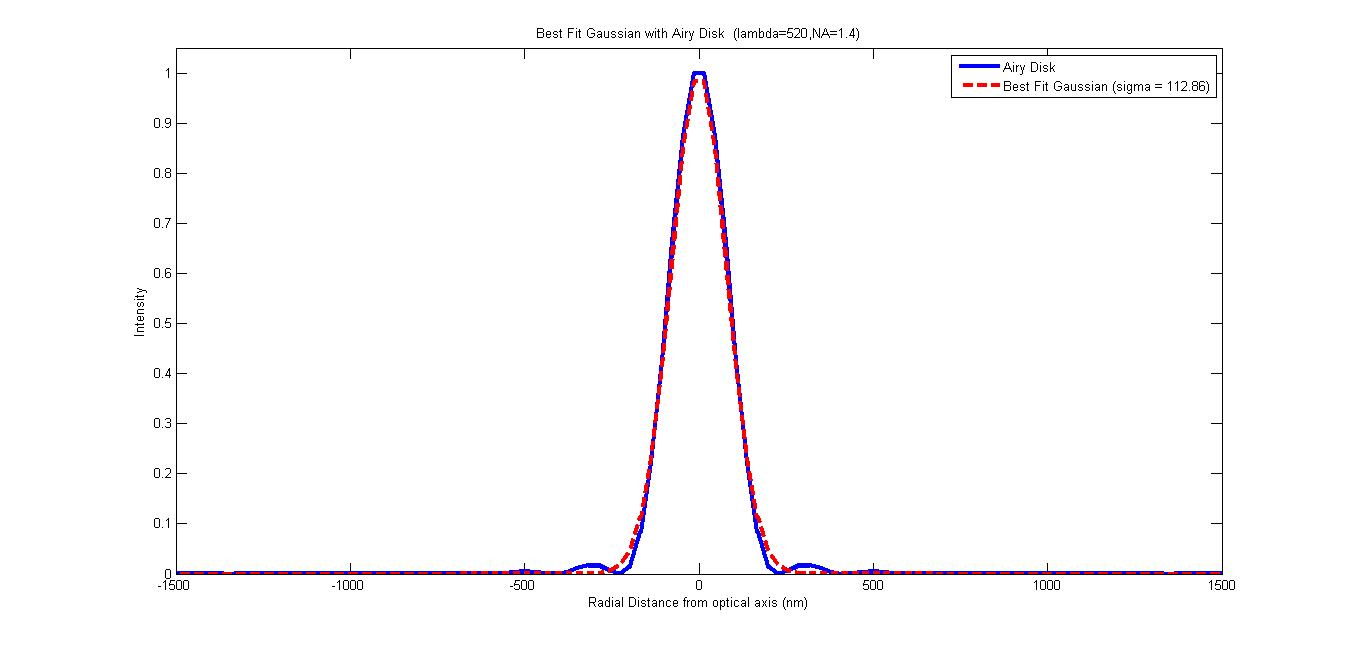
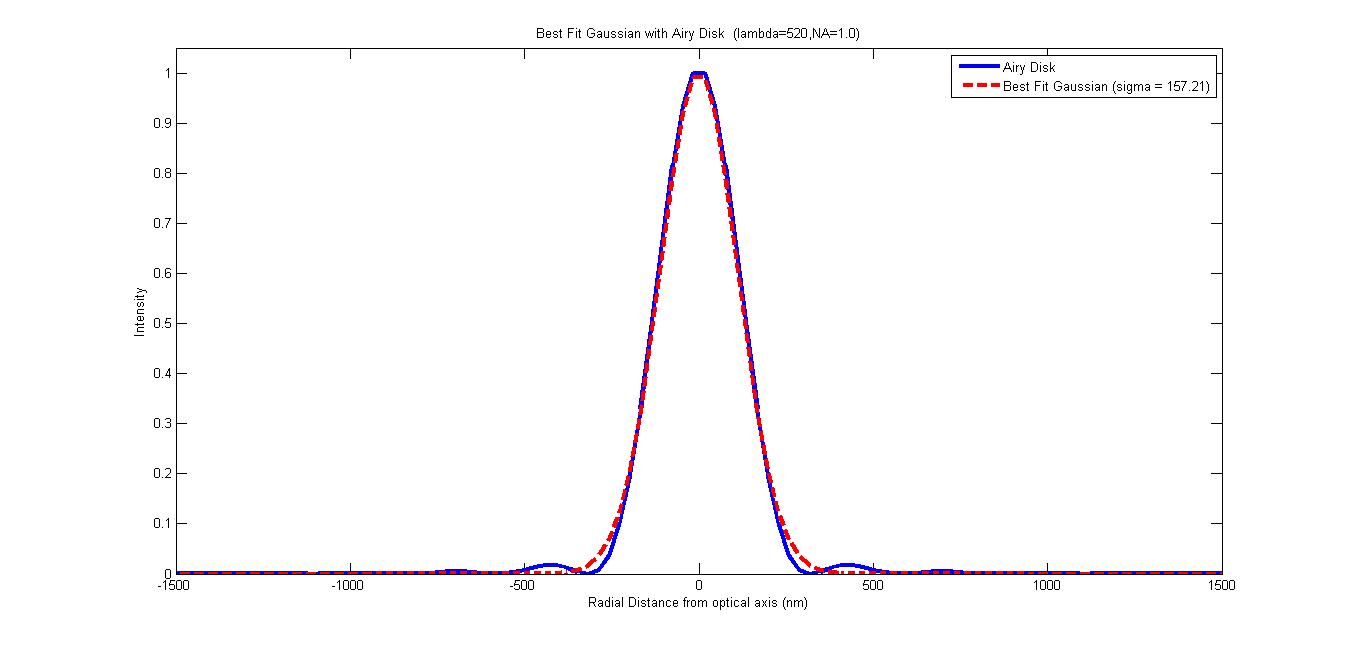
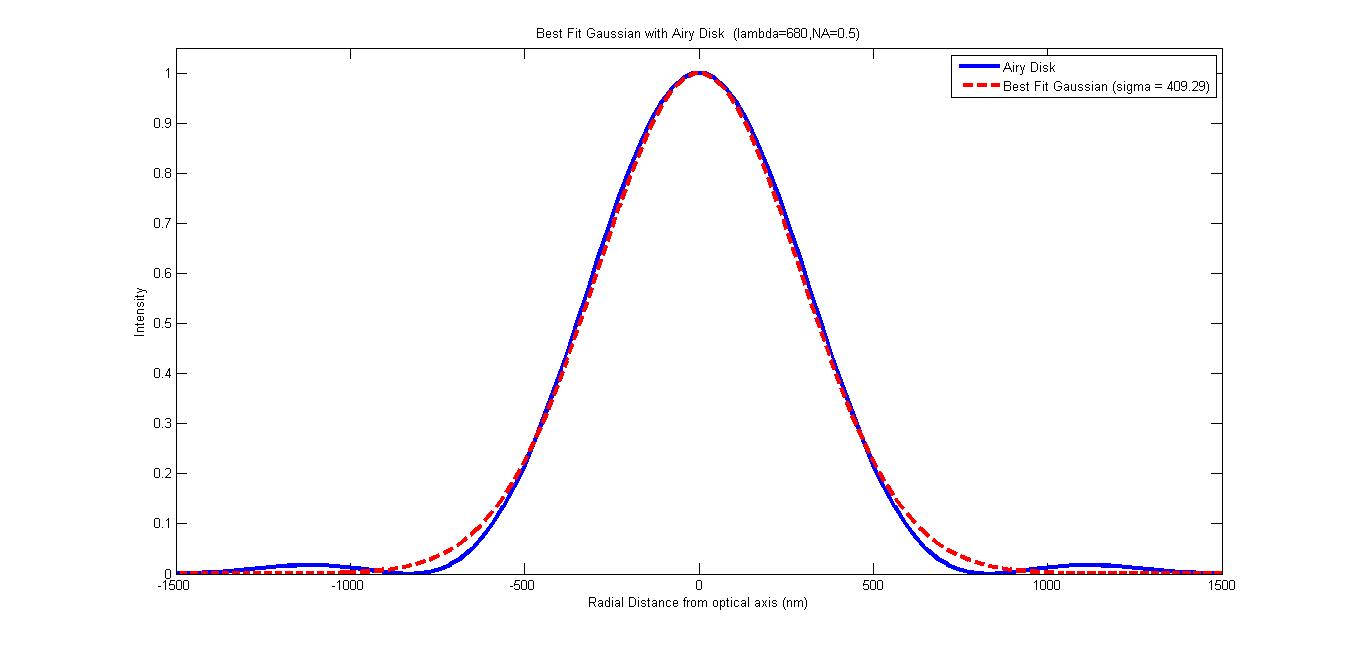
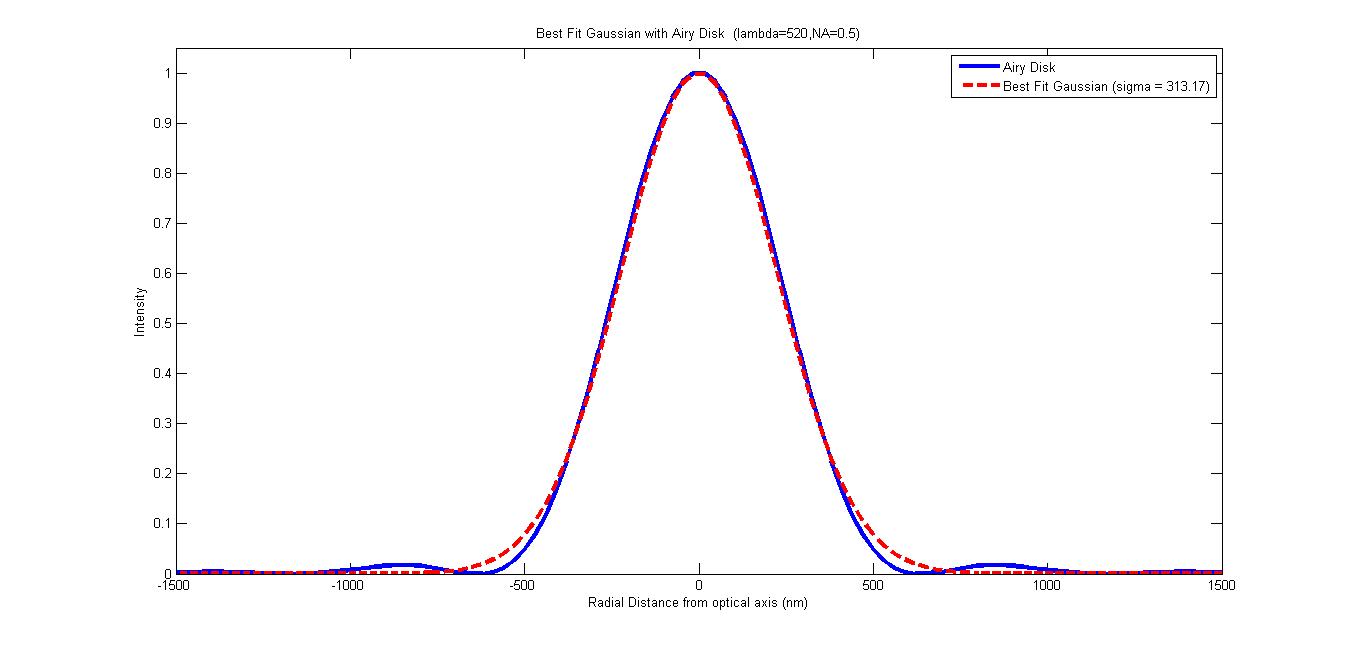
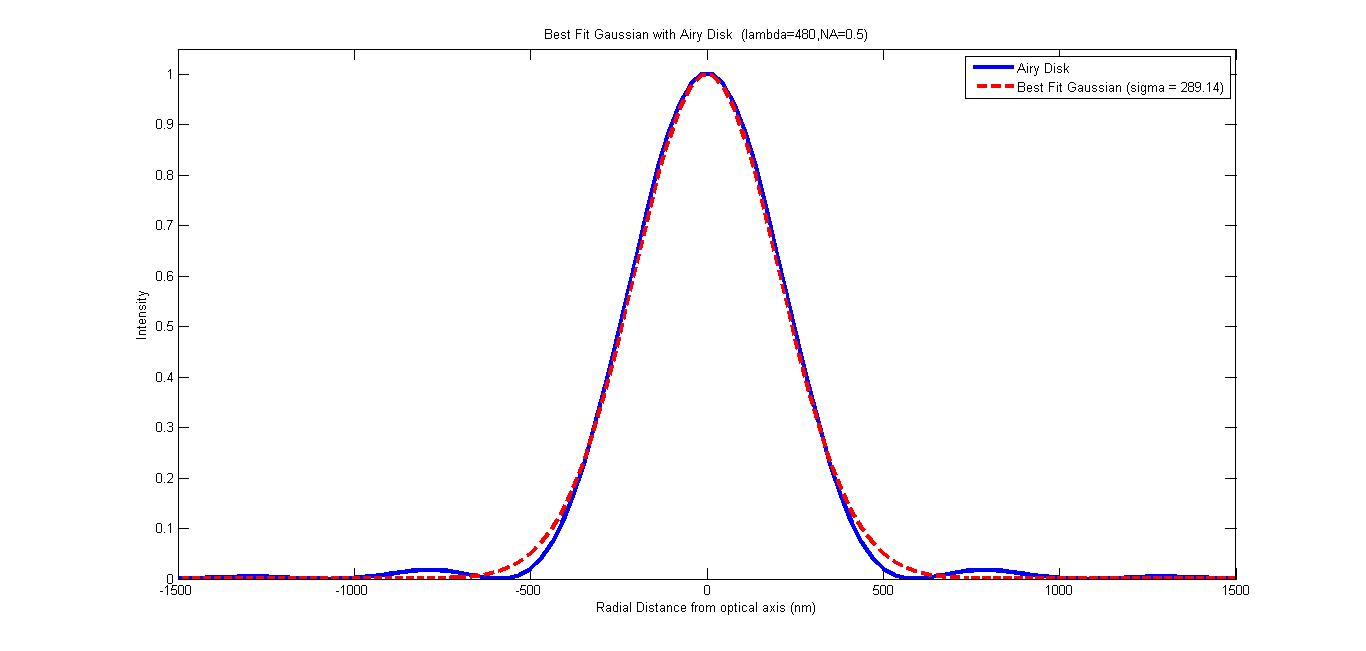


Table summarizing the gaussian standard deviation values found after fitting the Gaussians:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.No.** | **(nm)** | **NA** | **Approximate Airy Disk radius (nm)** | **Gaussian (nm)** |
| 1 | 480 | 0.5 | 586 | 289.14 |
| 2 | 520 | 0.5 | 635 | 313.17 |
| 3 | 680 | 0.5 | 830 | 409.29 |
| 4 | 520 | 1.0 | 317 | 157.21 |
| 5 | 520 | 1.4 | 227 | 112.86 |
| 6 | 680 | 1.5 | 277 | 137.28 |

The airy disk radius and the gaussian are completely correlated as can be seen from the plot below:

We observe that the airy disk radius is approximately twice the standard deviation of the fitted Gaussian i.e . The fact that we can fit a Gaussian to all these airy disks makes us come to the conclusion that the airy disk can be suitably approximated using a Gaussian kernel with where is the airy disk radius.

**NOTE:** It was assumed that the magnitude of the Gaussian was since the airy disk was normalized before plotting. The Gaussian was fitted using the MATLAB function that minimizes the least square difference between the gaussian kernel and the point spread function (airy disk).