Gaussian Low and High Pass Filters

Andrew Lo
Department of Electrical Engineering
Ryerson University, Undergraduate
Toronto, Ontario

***Abstract*— The purpose of this paper was to investigate and implement Gaussian low and high pass filters for digital image processing. A Gaussian filter is a filter using the Gaussian function, where the filter can be applied in the spatial domain or the frequency domain. Gaussian filters are used commonly through the graphic and filming industries. Gaussian low pass filter is used to blur the image and the Gaussian high pass filter is used to sharpen the image.**

Keywords: Gaussian Low Pass Filter, Gaussian High Pass Filter, Discrete Fourier Transform(DFT), Inverse Discrete Fourier Transform(IDFT), Edge Detection, Bokeh Effect, Central Limit Theorem(CLT)

I. Introduction

Digital image processing has come a long way in recent years. Modern technologies can process images and video data with desired effects at a much higher rate. This process involves a filter convolution with an image or video data either in the spatial or frequency domain. One of the most used filters in signal processing is the Gaussian filters. Gaussian filters follow the central limit theorem, which reflects on many real-life applications. Thus, the processed images are much more pleasant to human eyes.

II. Theory

A. Gaussian Low Pass Filter

Gaussian low pass filter is a filter with Gaussian function as the coefficients on the mask. A Gaussian low pass filter is used to blur or smooth the image. The effect of the blurring depends on the variance, the mean, and the block size of the filter. In the spatial domain, the filter size should be odd and usually spans from [-3σ.3σ] [1]. The image is first padded with zeros or border pixel coefficients, then the Gaussian filter is convoluted through every single pixel on the image. Lastly, the padded border is removed for the output image. The 2-D Gaussian filter in the spatial domain is:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{\left(-\frac{(x-\mu_x)^2+(y-\mu_y)^2}{2\sigma^2}\right)}$$

It is important to note that the Gaussian function's forward and inverse Fourier transforms are real [2]. This means the filter function is identical in spatial and frequency domain. To apply the Gaussian filter in the frequency domain, first, we need to pad the MxN image with 2M and 2N of zeros. Then modulate padded image with $(-1)^{x+y}$ to center its transform and Compute DFT. Then the filter is applied to the whole frequency image, such a process is done by computing the multiplication. The reason behind multiplication is because the Fourier Transform of convolution is multiplication. Then IDFT is performed and only real value is taken. Lastly, the image is modulated again with $(-1)^{x+y}$ and the MxN image is extracted on the top left of the image [2]. This process is to avoid the discontinuities that occur when computing the DFT of the image. The modulation helps to center the frequency components so it is easier to apply the mask. In the frequency domain, the Gaussian filter is given as:

$$H(u,v) = e^{-\frac{D^2(u,v)}{D_0^2}}$$

Where D0 is a replacement of variance σ, and it represents the cutoff frequency for the filter. The blurring effect of the Gaussian filter applied in the spatial domain is the same as applied in the frequency domain, however, doing it in the frequency domain has a higher computational cost but easier to design the filter.

B. Gaussian High Pass Filter

Gaussian high pass filter utilizes an inverse Gaussian function as its mask and sharpens the image. In the spatial domain, the Gaussian high pass filter is harder to design. The 1-D formula is given as:

$$f(x,\mu,\lambda) = \left(\frac{\lambda}{2\pi x^3}\right) e^{-\frac{\lambda(x-\mu)^2}{2\mu^2 x}}$$

Where μ is the mean and λ is a shape parameter. In the frequency domain, the filter is much easier to design. Using the same zero padding and modulation procedures, the filter is applied to the whole frequency map. The filter in the frequency domain is given as:

$$H(u,v) = 1 - e^{-\frac{D^2(u,v)}{D_0^2}}$$

The processed image is the sharpen component of the image not the whole image sharpen. One extra step is usually taken where the sharpen component is added back to the original image to enhance the image.

III. Methodology

A. Gaussian Low Pass Filter

1-D Gaussian with different means and variances can be shown using the **gaussplot.m**. This plot gives us a general preliminary understanding of the Gaussian functions.

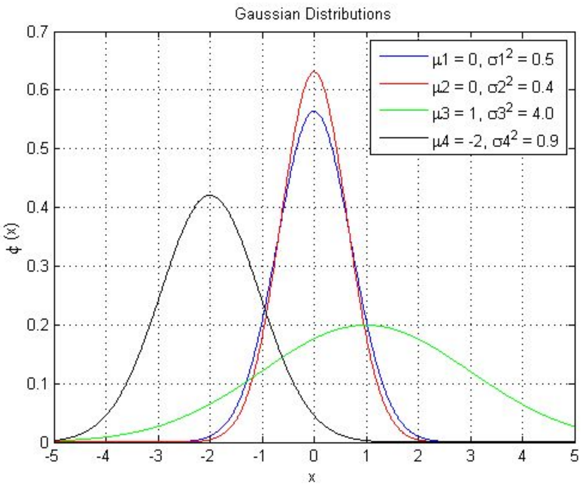

Figure 1: Gaussian Functions with Different Means and Variances

In 2-D, the Gaussian function looks identical, such filter can be realized using the **gauss2d.m**. The center pixel has the highest intensity and the corner edges have the lowest intensities.

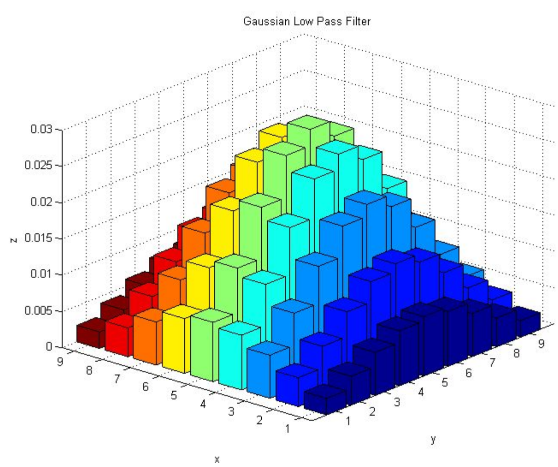

Figure 2: Gaussian Functions in 2-D

The function to blur the image in the spatial domain is called ***Gaussian.m***. This algorithm takes into consideration the mask sizes and type of padding (zero/duplication). The algorithm then computes the variance for the mask and calls two special functions – ***fspecial ()*** and ***imfilter ()***. ***fspecial ()*** is an embedded Matlab function used to design different filters for digital image processing. The user feeds in the desired type of filter, mask size and the variance of the filter, and it will output the corresponding coefficient mask ready to be convoluted. ***Imfilter ()*** is another embedded Matlab function that filters the multidimensional arrays, this function allows the user to convolute the original image with the desired mask. Finally, the algorithm ***Gaussian.m***_will output the original image along with the filtered image.

The function that computes Gaussian low pass filtering in the frequency domain is called ***fgaussian.m***. This function also deals with the Gaussian high pass filtering which will be explained in part II. The Gaussian low pass filter has 3 inputs: original image, filter type, and D0 (cutoff frequency). The first step of the algorithm is to caste the original image to double, this step is important because modulation requires double to work (negatives). Then 2Nx2M zero pad the image and the original image is copied on to the 2Nx2M picture by using for loops. The modulation is done after a Fast Fourier Transform (radix 2) by calling the ***fftshift ().*** The algorithm computes the distance calculation is Euclidean distance (other calculation methods can be used as well) along with for loops to calculate 2-D distances in the image. Then the algorithm calls ***flipdim ()*** which flip array along specified dimension for mirror distance metric to all four quadrants. Finally, the Gaussian filter can be defined since the distance matrix is discovered. The Gaussian filter is multiplied with the FFT image to obtain the new filtered image in the frequency domain. The data then passes through an inverse fast Fourier transform-2 and crop back to original size NxM by using for loop. The algorithm will output the cropped image in which its high-frequency components will disappear and only the low-frequency component will remain. The image will look blur as intended by the function.

B. Gaussian High Pass Filter

In 2-D a Gaussian high pass filter looks like the following.

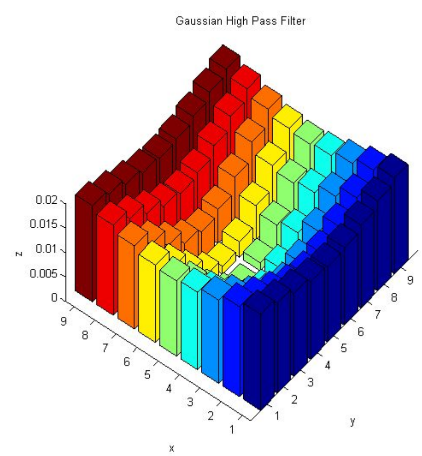

Figure 3: Inverse Gaussian Functions in 2-D

A 2-D Gaussian high pass filter has the lowest intensity at the center and the highest intensities at the corners. It is expected from an inverse Gaussian filter since we are subtraction 1 with the Gaussian function.

For the Gaussian high pass filter, the function is called ***invgaussian.m.*** The algorithm reads in the input image and the desired mask size. The filter computes the variance and creates the Gaussian low pass filter using ***fspecial ()***. Then using the max intensity of the filter and subtract with the whole filter to create a Gaussian high pass filter. ***Imfilter ()*** is used to convolute the filter with the image. This is an alternative way of creating a Gaussian high pass filter than the one defined in Theory, however, it creates many problems.

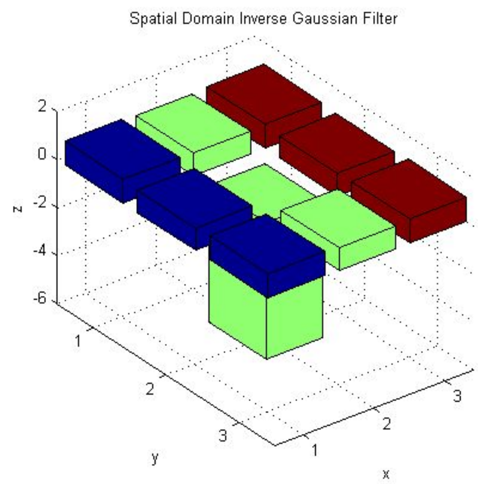

Figure 4: Inverse Gaussian Functions in invgaussian.m

As shown in figure 4, the alternative way is missing the λ shape parameter. The plot shows the 8-neighbor pixels are almost the same and the center pixel is extremely low. In the experiment, only a few preset mask size will work, otherwise, the image will output completely white.

In the frequency domain, the Gaussian high pass filter is implemented in ***fgaussian.m,*** and the steps for the function is similar. The minor difference is when the filter is multiplied with the mask, the mask is first subtracted (mask=1-mask). This is also one of the advantages of the Gaussian filters in the frequency domain where the mask can be used interchangeably.

C. Difference Gaussian Filter

This is an experiment that designs a filter using the difference of two Gaussian filters. The function is called ***diffgaussian.m***, and it reads the input image, masks size, variance 1, and variance 2. Variance 1 and 2 are defined by the user for research purposes. A difference Gaussian behaves similar to Laplacian filter or Edge Detector. The 1-D Gaussian difference function looks like the following.

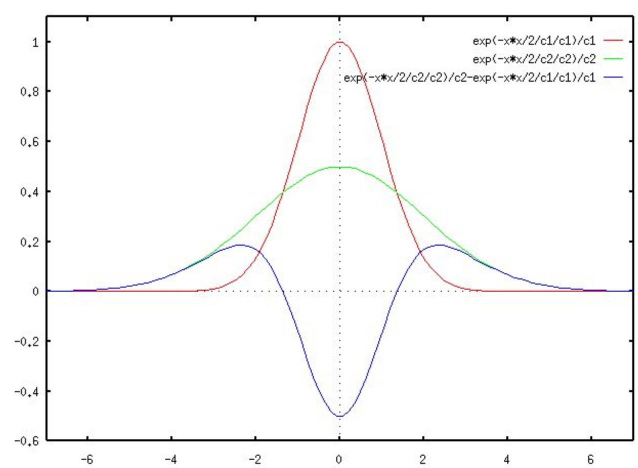

Figure 5: Difference of Gaussian [3]

In 2-D, the difference Gaussian filter looks similar to the 1-D with difference Gaussian filter shown in figure 4.

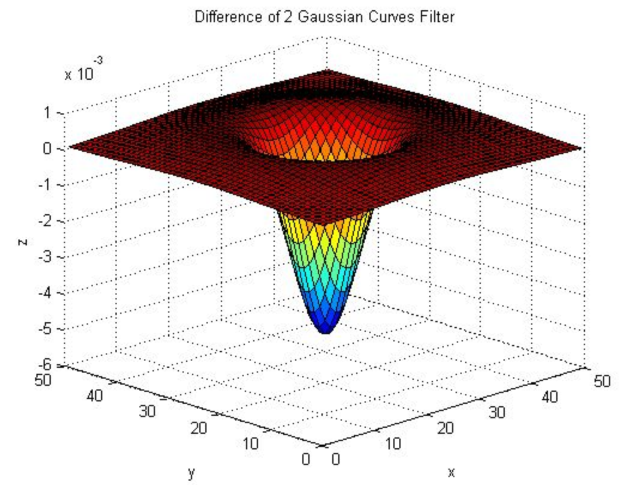

Figure 6: Difference Gaussian Filter in 2-D

IV. Results

A. Gaussian Low Pass Filter

Figure 7: Gaussian Low Pass Filter Applied in Spatial Domain

In the spatial domain, the result of ***Gaussian.m*** is shown above. The mask size of the filtered image is 50, and we can see that a lot of sharp details are gone and if the mask size increases, the blurring will also increase.

For frequency domain, we will be analyzing a popular controversial image being floating around on the internet. The Albert Einstein mixed with Marilyn Monroe shown below.

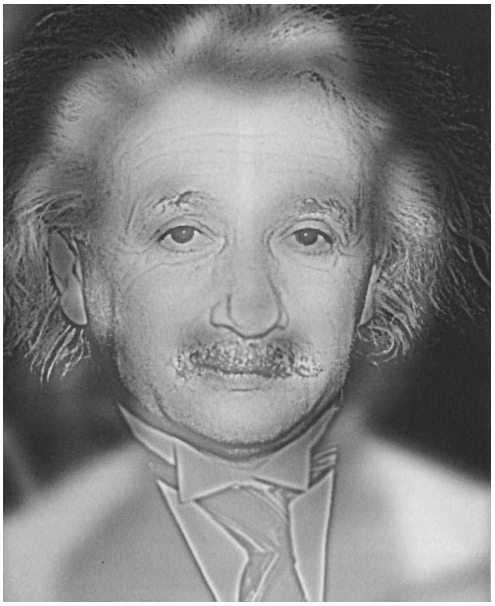

Figure 8: Albert Einstein or Marilyn Monroe?

The above image is fed through the ***fgaussian.m*** and the resulting Gaussian low passed image is

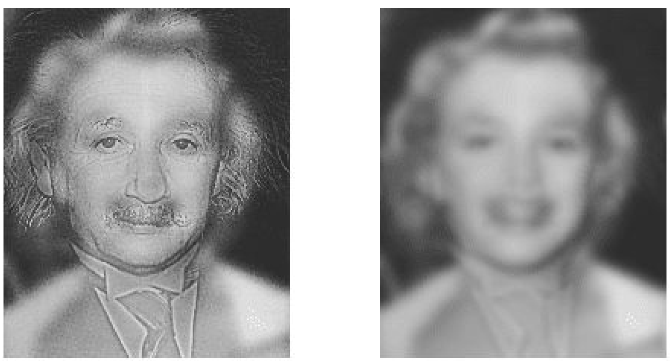

Figure 9: Imaged Processed by Gaussian Low Pass Filter in Frequency Domain

Shown in figure 9, the blurred image shows a picture of Marilyn Monroe.

B. Gaussian High Pass Filter

In the spatial domain, the Gaussian high pass ***invsgaussian.m****,* shows only the high frequency-component.

Figure 10: Gaussian High Pass Filter Applied in Spatial Domain

As shown in the above figure, the sharp changes remained in the picture, and the subtle changes are gone and shown as a white background.

In the frequency domain, the function ***fgaussian.m*** using the inverse type of Gaussian, the user can retrieve a different picture shown in figure 9.

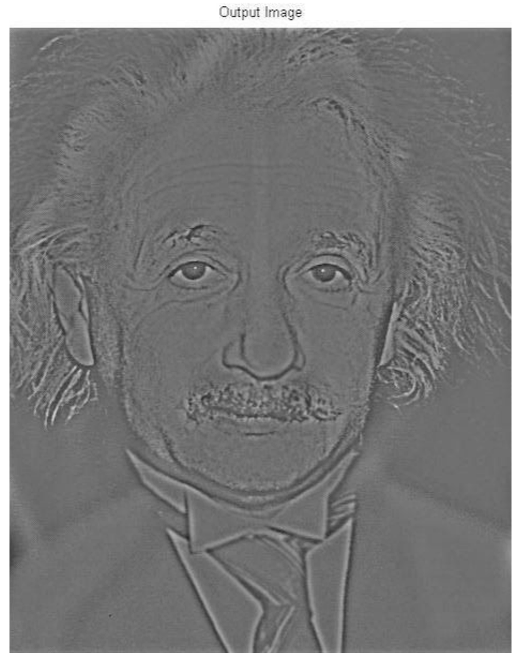

Figure 11: Image Processed by Gaussian High Pass Filter in Frequency Domain

In figure 11, the picture shows only the sharpen component, and it is clear to say that the picture resembles Albert Einstein rather than Marilyn Monroe.

C. Difference Gaussian Filter

Applying ***diffgaussian.m***, the image shows the edge detection characteristics in the spatial domain.

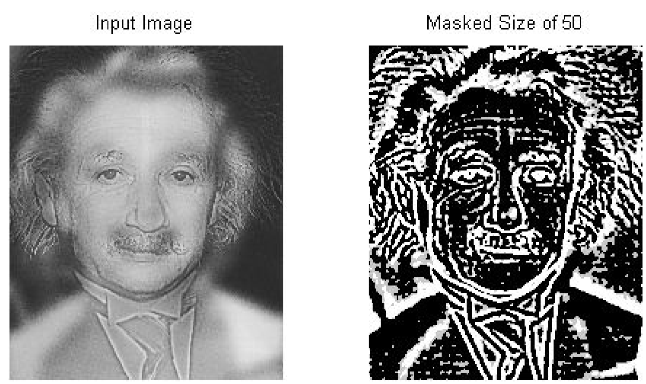

Figure 12: Imaged Processed by Difference Gaussian in Spatial Domain

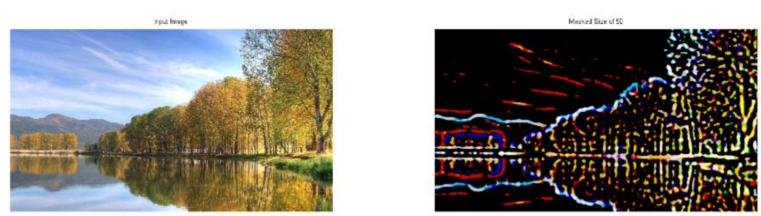

Figure 13: Imaged Processed by Difference Gaussian in Spatial Domain

V. Discussion

The Albert Einstein and Marilyn Monroe picture is a mixed picture. The scientist smoothed a picture of Marilyn Monroe and added Albert Einstein's details on to the picture. The eyes and the angle of the face are matching, thus making the picture very believable. If we look at the frequency map of this picture, it is easy to see the two different frequency components being mixed.

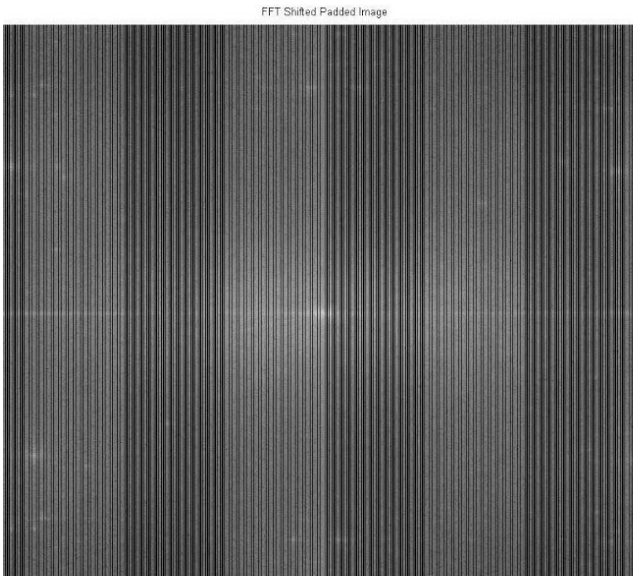

Figure 14: Frequency Map of the Image

Observing the two different dark and white stripes on the frequency map, the reason behind these stripes is because they are mixed from two different images. This is why image mixing is sometimes easy to detect by looking at the Fourier transform of the image.

For the Gaussian low pass filter, it is easy to implement in both spatial and frequency domains. The effects of the low pass filters are similar in both domains. It provides effective blurring but in the more eye-pleasing ways than the average filter.

One of the other uses for the Gaussian low pass filter is that it is good at filtering out Gaussian noise. Gaussian noise is a noise that is common in many real-life signal processing applications. Gaussian low pass filter is excellent for filtering such noise because of CLT.

The high pass filter extracts the high-frequency components and filters out the low-frequency components. From figures 9 and 11, we can see by applying a low pass filter or high pass filter, we can see observe two different people's faces. This allows us to see things in different perspectives of an image which is something we cannot in the past decades.

Many applications now rely on Gaussian filter for smoothing image, edge detection (not sensitive to noise), Laplacian of Gaussian filtering (smooth then sharp, a similar method to unsharp and high boost filtering), video games, camera, and space images. These applications all require fast computing processes and are now affordable to be done by anyone at any level.

VI. Conclusion

Gaussian filtering is a more eye-pleasing filter technique that takes into consideration of probability and statistics of the image data. In both spatial and frequency domains, the filter achieves its goal either smoothing using a Gaussian low pass filter or sharpening using a Gaussian low pass filter. Despite the Gaussian high pass filter being a bit harder to implement in the spatial domain, it is considered to be achievable as shown in the experiment.

References

[1] Kyan M. (2013). ELE882_W2013_L06_FilteringInTheFrequencyDomain.pdf . https://courses.ryerson.ca/bbcswebdav/pid-2212614-dt-content-rid-3370600_2/courses/ele882_w13_01/ELE882_W2013_L06_FilteringInTheFrequencyDomain.pdf Retrieved 1 April 2013.

[2] Rafael C., Woods, Richard E. (2008). *Digital Image Processing*. Pearson Education, Inc. pp.270-286. ISBN 978-0-13168-728-8.

[3] Drakos N., Moore R.(1999). http://fourier.eng.hmc.edu/e161/lectures/gradient/node10.html Retrieved 1 April 2013.

Appendix

Gaussianplot.m

```
function gaussplot(mu,va,x1,x2,color,lc)
sd=sqrt(va); %Standard deviation
x=[x1:0.01:x2]; %Create points for x-axis
y =
1./(sd.*sqrt(2*pi)).*exp(-0.5*((x-mu)/sd).^
2); %Gaussian Equation
plot (x,y,color,'DisplayName',['\mu'
num2str(lc) ' = ' num2str(mu) ', \sigma'
num2str(lc) '^2 = ' num2str(va)]);
legend('-DynamicLegend');
xlabel('x');
ylabel('\phi (x)');
title('Gaussian Distributions');
hold on;
end
```

Gauss2d.m

```
function [output] = gauss2d(N,std,type)
%%    Example
%  gauss=gauss2d(9,3)
%%    Algorithm
  [x y]=meshgrid(round(-N/2):round(N/2),
round(-N/2):round(N/2));
  f=1-exp(-x.^2/(2*std^2)-y.^2/(2*std^2));
  bar3(f);
  grid;
  output=f;
 end
```

Gaussian.m

```
function [output]=gaussian(input,s,padding)
%% Example
%[output]=gaussian('MarilynAlbert.jpg',1000
```

```
,'duplication');
%[output]=gaussian('Nature.jpg',50,'duplica
tion');
%[output]=gaussian('house.jpg',50,'duplicat
ion');
%% Algorithm
image=imread(input);
[row col]=size(input);
variance=(s-1)/6;
switch padding
    case 'zero'

Mask=fspecial('gaussian',s,variance);
        output=imfilter(image,Mask);
        subplot(1,1,1);
        imshow(image);
        title('Input Image');
        subplot(1,1,2);
        imshow(output);
        title(sprintf('Masked Size of
%i',s));
    case 'duplication'

Mask=fspecial('gaussian',s,variance);

output=imfilter(image,Mask,'replicate');
        subplot(1,2,1);
        imshow(image);
        title('Input Image');
        subplot(1,2,2);
        imshow(output);
        title(sprintf('Masked Size of
%i',s));
    otherwise
        warning('Error! Enter zero or
duplication only.');
end
end
```

Invgaussian.m

```
function [output]=invgaussian(input,s)
%% Example
%[output]=invgaussian('Nature.jpg',20);
%% Algorithm
image=imread(input);
[row col]=size(input);
variance=(s-1)/6;
Mask=fspecial('gaussian',s,variance);
inverseMask= max(max(Mask))-Mask;
output=imfilter(image,inverseMask,'replicat
e');
subplot(1,2,1);
imshow(image);
title('Input Image');
subplot(1,2,2);
imshow(output);
title(sprintf('Masked Size of %i',s));
end
```

Diffgaussian.m

```
function
[output]=diffgaussian(input,s,v1,v2)
%% Example
%[output]=diffgaussian('MarilynAlbert.jpg',
50,12,5);
%[output]=diffgaussian('Nature.jpg',50,12,1
1);
%[output]=diffgaussian('Carl_Friedrich_Gaus
s.jpg',50,12,11);
%[output]=diffgaussian('jennifer.jpg',50,12
,11);

%% Algorithm
image=imread(input);
[row col]=size(input);
Mask1=fspecial('gaussian',s,v1);
Mask2=fspecial('gaussian',s,v2);
Mask=Mask1-Mask2;
bar3(Mask);
figure
output=200*imfilter(image,Mask,'replicate')
;
subplot(1,2,1);
imshow(image);
title('Input Image');
subplot(1,2,2);
imshow(output);
title(sprintf('Masked Size of %i',s));
end
```

Fgaussian.m

```
function [output] =
fgaussian(image,filtertype,D0)
%% Example
%
fgaussian('MarilynAlbert.jpg','gaussian');
%
fgaussian('MarilynAlbert.jpg','invgaussian'
);
%% Algorithm
input=double(imread(image));
[M N]=size(input);
P=2*M; %Used for padding
Q=2*N; %Used for padding
imagepad=zeros(P,Q);
for i=1:M
    for j=1:N
        imagepad(i,j)=input(i,j);
    end
end
```

```
imageFFT=fft2(imagepad);  %Fast Fouriour Transform
imageFFT=fftshift(imageFFT); %Fast fourier shift

distance=zeros(M,N); %Compute distance calculation
for i=1:M
        for j=1:N
            distance(i,j)=sqrt((i-1)^2+(j-1)^2);
        end
end

%Mirror distance metric to all 4 quadrants
a=flipdim(distance,1);
b=flipdim(a,2);
c=flipdim(b,1);
b=horzcat(b,a);
distance=horzcat(c,distance);
distance=vertcat(b,distance); %Euclidean Distance

GaussianFilter=exp(-(distance.^2)./(2.*(D0.^2)));

if strcmp(filtertype, 'gaussian')==1
    filtered=imageFFT.*GaussianFilter; %Filter image F(u,v)H(u,v)
    filtered_padded=ifft2(ifftshift(filtered)); %Inverse shift and transform
    output=zeros(M,N);%crop filtered image from padding
    for i=1:M
         for j=1:N
             output(i,j)=filtered_padded(i,j);
         end
    end
    figure;
    imshow(output,[]);
    title('Output Image')
    output=uint8(output);
end

if strcmp(filtertype, 'invgaussian')==1
    filtered=imageFFT.*(1-GaussianFilter); %Filter image F(u,v)H(u,v)
    filtered_padded=ifft2(ifftshift(filtered)); %Inverse shift and transform
    output=zeros(M,N);%crop filtered image from padding
    for i=1:M
         for j=1:N
             output(i,j)=filtered_padded(i,j);
         end
    end
    figure;
    imshow(output,[]);
    title('Output Image')
    output=uint8(output);
end
end
```