De La Salle University – Manila

Gokongwei College of Engineering

Department of Electronics and Computer Engineering

Signals, Spectra and Signal Processing Laboratory

LBYEC4A – EK2

Final Project

Creation of Laboratory Manual on Image Convolution Kernels

by

Amadora, John Ernesto Jr.

Montaño, Rafael Dominic

Tablada, Alyssa Joie

Submitted to:

Engr. Stephen Ruiz

I. Abstract

Image processing techniques use filters to enhance an image through modifying its contrast, brightness, resolution, and noise level. Image processing is done to transform the image and obtain useful information for various applications like medicine. Although introductory information was given for the course of Signals, Spectra, and Signal Processing Laboratory (LBYEC4A), the authors want to expand the knowledge of the students through creating a laboratory manual and a corresponding solution set to serve as the professor's guide in checking the exercise proper. The authors were able to accomplish the following objectives: (1) To identify the convolution kernel based on the given description; (2) To describe and observe the difference of each convolution kernel upon application to an image; (3) To compare the 2D and 3D magnitude spectra plot of the convolution kernels and filtered images; and (4) To observe the effects of increasing the dimensions of the matrix for the Gaussian blur kernel.

Keywords – image processing, convolution kernel, 2D and 3D magnitude spectrum, Gaussian blur

II. Introduction

An image kernel is a small matrix used to apply effects on images to smoothen, sharpen, intensify, or enhance some parts of the image that the user would like to highlight [1]. Your choice of kernel affects the output image. Different sizes can be used for a kernels, however, 3x3 is commonly used. Also, one of its restrictions is that the convolution matrix is only comprised of integers.

In this experiment, the following objectives should be accomplished:

- To identify the convolution kernel based on the given description
- To describe and observe the difference of each convolution kernel upon application to an image
- To compare the 2D and 3D magnitude spectra plot of the convolution kernels and filtered images
- To observe the effects of increasing the dimensions of the matrix for the Gaussian blur kernel

The authors only aim to achieve these objectives by creating the laboratory manual and its corresponding solution set. However, the testing of the effectiveness of the laboratory manual and solution set, and feedback gathering are not part of this project.

III. Theoretical Consideration

Convolution kernels are matrices that are used to extract or show specific and desired features from an image [2]. These features can range from increasing contrast, sharpening, smoothening, blurring, enhancing, and much more. The desired feature to be extracted from the input image can be communicated to the kernel by modifying its matrix elements. The numerical elements on the kernel can also be referred to as "weights" or "bias". One can increase or decrease the "bias" of a matrix in order to serve the desired feature needed.

As the name suggests, convolution kernels are applied to an input image through the operation of convolution. Convolution is the mathematical process of combining two signals to form a third signal – one that represents the synthesization of both signals [3]. In the context of matrices, the convolution makes use of purely matrix operations, with matrix multiplication and addition being the most frequent operations. The matrix of weights or kernels are multiplied with the input to extract any features desired by the user.

Convolution kernels can be represented as one dimensional (1D), two dimensional (2D), or three dimensional (3D). A one-dimensional convolutional kernel in essence is a vector. The applications of these are mostly in time series data analysis. Two dimensional convolutional kernels are conventionally represented as square matrices — having an equal number of rows and columns. This is to create a balanced network of parameters to distribute the weights and biases of the matrices equally. The applications of two-dimensional kernels are used in image processing and in deep learning models. Three dimensional convolutional kernels are much more complex representations of matrices. It is used in computer vision to extract spatial features from an input signal in three dimensions [4].

In this laboratory experiment, there are eight convolution kernel patterns to be used and implemented. These are identity, Gaussian blur, sharpen, Sobel Operator for Vertical Edge Detection, Laplace Operator, Sobel Operator for Horizontal Edge Detection, Outline Kernel, and Emboss Kernel. Each of which shall be discussed in this section.

The identity convolution kernel is simply an identity matrix — containing an element of 1 in the middle surrounded by values of 0. It is also termed as the "do-nothing" convolutional kernel as it replicates the input image, hence the name "identity" [5].

The Gaussian blur convolution kernel is designed to blur or smoothen the image in a manner that follows the Gaussian bell curve. The pattern of the matrix adds more weight to the center of the matrix as compared to the edges. The rate at which the weight diminishes from the center follows the Gaussian function [6]:

$$G(x) = \frac{1}{\sqrt{2\pi}\delta} e^{\frac{-x^2}{2\delta^2}}$$

Furthermore, Gaussian blurring essentially makes use of a low pass filter, attenuating all higher image frequencies and allowing all lower image frequencies to pass through.

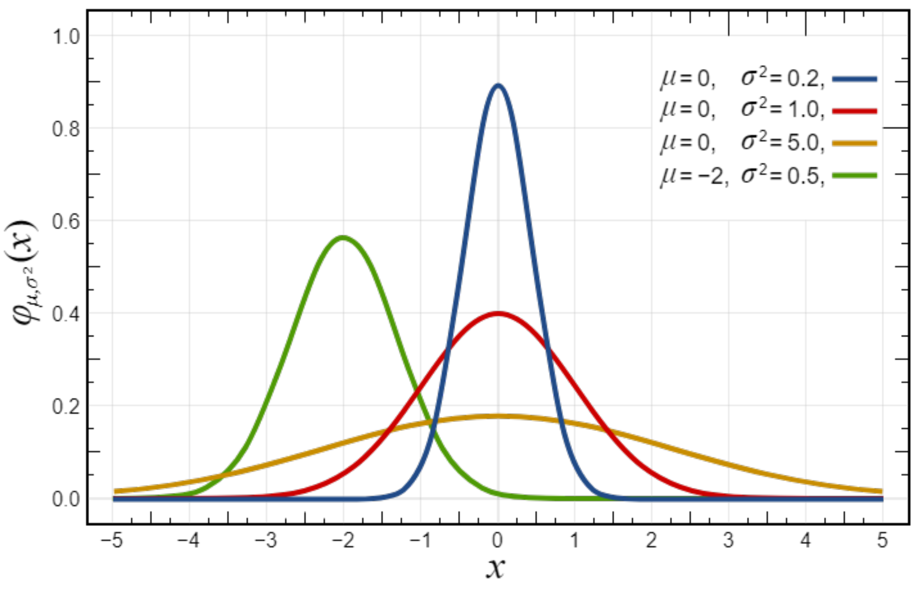


Figure 2.1. Gaussian Bell Curve [6].

The sharpen convolutional kernel is designed to add contrast to the input image, giving it the feature of being more sharpened. It provides the opposite effect to the Gaussian blurring, which decreases the contrast. Thus, the sharpening convolutional kernel works in essence to a high pass filter — which attenuates all lower image frequencies and permits all higher image frequencies to pass through [7].

For edge detection convolutional kernels, the laboratory activity presents three types of kernels, which are the Sobel Operator for Vertical and Horizontal Edge Detection, Laplacian Operator, and Outline Kernels. Edge detection convolutional kernels are matrices that prominently emphasize and extract the edges of objects in images. The theory behind edge detection lies in the intensity of pixels.

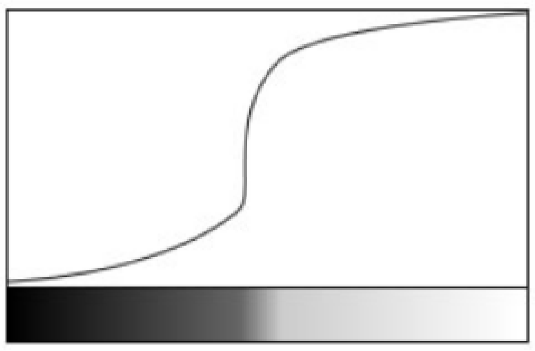


Figure 2.2. Pixel Intensity with Regard to Grayscale Color from Black to White [8].

Figure 2.2 presents a graph that describes the behavior of pixel intensity in relation to the grayscale colors gradient from black to white. It can be observed that the pixel intensity increases abruptly at the transition point between black and white. The pixel intensity shoots up at the "edge" of the black and white gradient transition. It can be further evaluated by taking the first derivative of the graph.

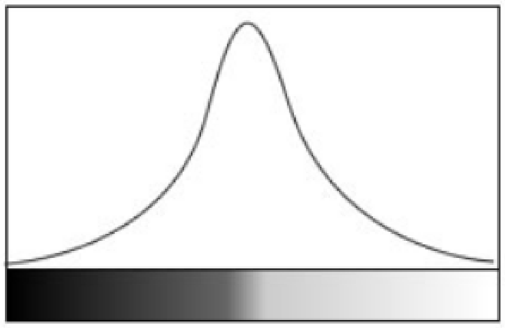


Figure 2.3. First Derivative of Pixel Intensity with Regard to Grayscale Color from Black to White [8].

Figure 2.3. shows the first derivative of the graph in Figure 2.3. Notice how the peak of the graph is aligned to the edge of the black to white gradient transition. What an edge detection convolutional kernel does is perform a mathematical calculation of derivation to extract only the pixels with a significantly high intensity value.

The Sobel Operator for Vertical and Horizontal Edge Detection works with this gradient based method [8]. For vertical edge detection, the derivation is performed along the y-axis while horizontal edge detection performs the derivation along the x-axis. To attain the strength of an edge, the following formula is used:

$$\sqrt{G_x^2 + G_y^2}$$

Where Gx = Horizontal Edge Detection
Gy = Vertical Edge Detection

The orientation of the edge is given as: arctan(Gy/Gx). Following the Sobel Operators is the Laplacian Operator. Unlike the Sobel Operator that requires the use of two convolutional kernels – horizontal and vertical, the Laplacian edge detector only requires one convolutional kernel. It was able to simplify the process through its use of a second derivative operation to further examine the relationship between pixel intensity and the gradient.

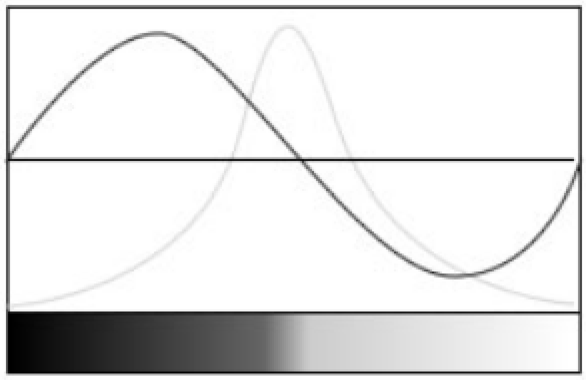


Figure 2.4. Second Derivative of Pixel Intensity with regard to Grayscale Color from Black to White [8].

Figure 2.4 presents the second derivative graph of pixel intensity regarding the grayscale gradient. It can be observed that a zero-crossing phenomenon occurs at the moment of transition between the black and white colors. This position on the gradient represents the edge of an object and thus could be extracted by designing a convolutional kernel to extract all pixels that respond with a zero-crossing intensity. A drawback to this convolution kernel is its sensitivity to noise due to the second derivative method used that increases the probability of noise appearing on the zero-crossing axis.

The last edge detection convolution kernel is the Outline kernel which simply shows the outline in or edges of objects in images using a specially designed matrix whose edge elements are identical to impose an emphasis on the edges of objects.

The final convolution kernel in this laboratory activity is the Emboss kernel. The Emboss kernel provides the perception of depth in an image. It produces a three-dimensional mold of a 2D image resembling a paper emboss where certain pixels and objects stand out into the foreground (high relief) while other parts are attenuated into the background (low relief).

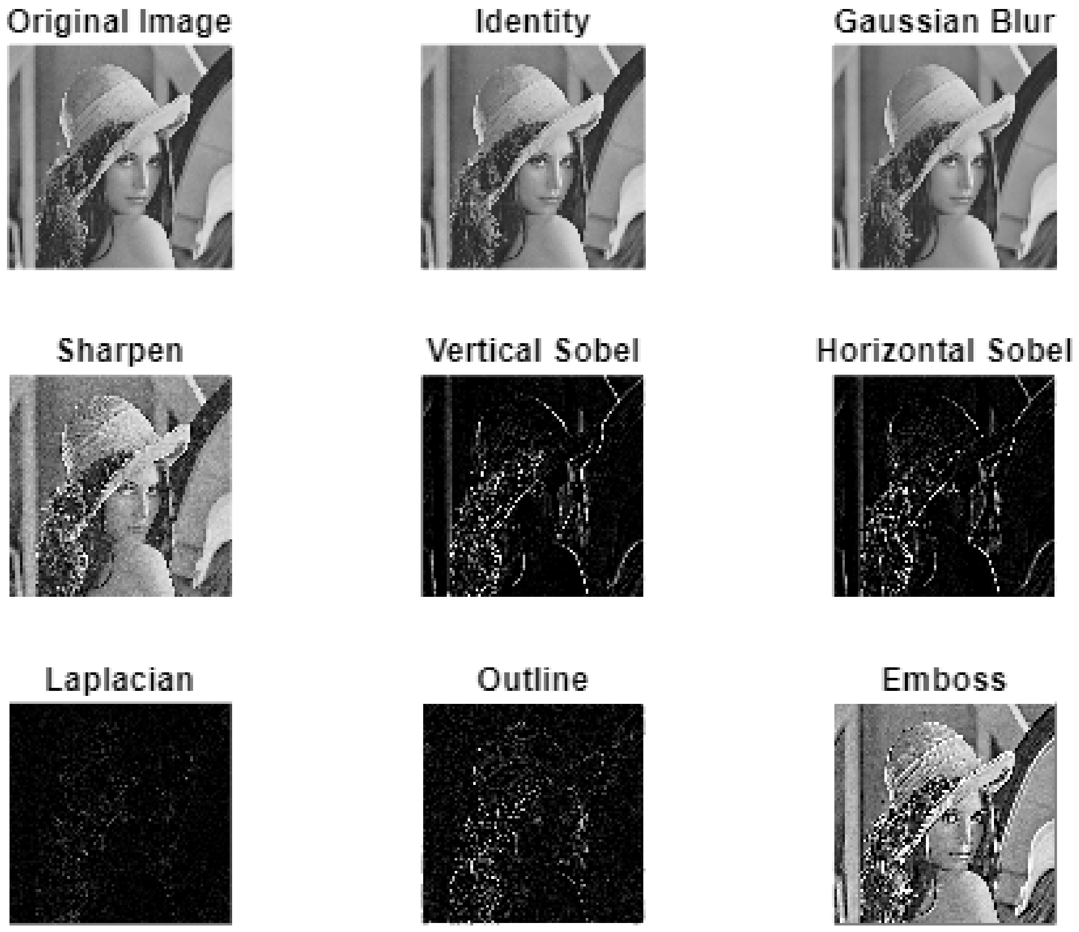


Figure 2.5. Test Images for the Different Features Extracted by the Listed Convolution Kernels.

As it can be recalled in previous experiments, the 2D magnitude spectrum can be obtained using the following lines of code:

```
F = fftshift(fft2(double(mat2gray(img)), 512, 512));
Ilog = log(1+abs(F));
colormap(gray); imagesc(Ilog); axis off
```

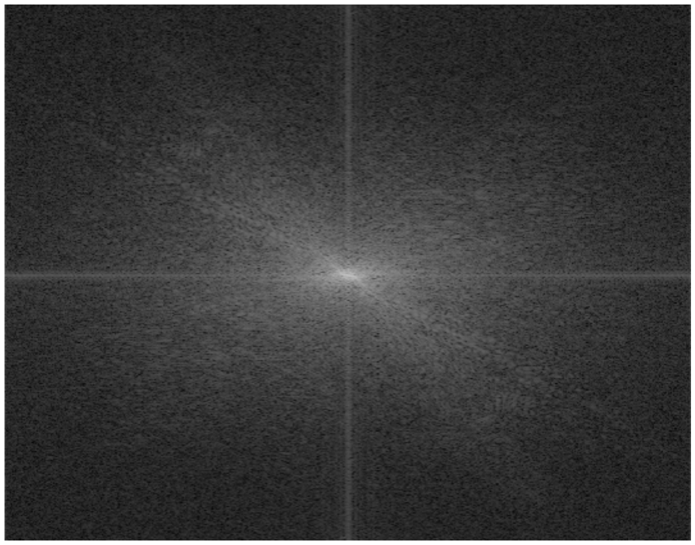


Figure 2.6. 2D Magnitude Spectrum Representation of lena.tiff.

The 3D magnitude spectrum can be obtained using the following lines of code:

```
H=fftshift(fft2(double(mat2gray(img)), 512, 512));
Hlog=log(1+abs(H));
u=-256:255;
v=-256:255;
[u,v]=meshgrid(u,v);
mesh(u,v,[Hlog(257:512,257:512),Hlog(257:512,1:256);
Hlog(1:256,257:512) Hlog(1:256,1:256)]);
```

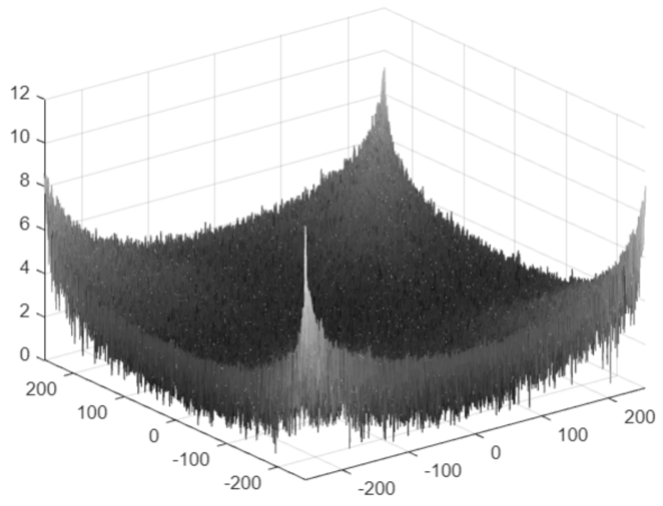


Figure 2.7. 3D Magnitude Spectrum Representation of lena.tiff.

IV. Methodology

Aligned with the course LBYEC4A, this project utilizes MATLAB. The project is divided into two main parts: the creation of the laboratory manual and the solution set. Since the goal of the laboratory manual is to serve as an extension of the lesson on convolution kernels, the authors first examined the points that were already tackled in the previous experiments. The authors then determined the gaps that should be addressed to further elevate the discussion of the topic.

After further research, the authors decided to further expand the discussion in terms of: (1) introduction of eight different 3x3 convolution kernels; (2) comparison of the 2D and 3D magnitude spectra plots of these convolution kernels; and (3) observation of the effects of a 3x3 and a 5x5 Gaussian Blur kernel. This is then applied in the construction of the laboratory manual.

As for the construction of the solution set, the authors first created prompts or questions that the students will answer in relation to the lecture. The authors then created their solution based on the prompts given and utilized this for the solution set or guide for the professors' reference.

The prompts in the exercise are as follows:

1. Encode the following convolution matrices in MATLAB

$H_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$H_2 = \left(\frac{1}{16}\right) * \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	$H_3 = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$
$H_4 = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$	$H_5 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	$H_6 = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$
$H_7 = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	$H_8 = \begin{bmatrix} -2 & -1 & 0 \\ -1 & 1 & 1 \\ 0 & 1 & 2 \end{bmatrix}$	

2. Using the grayscale version of the image *dog.png*, apply all the convolution kernels in #1. Show the original image and all the filtered images in one figure (use subplot). Ensure that the resolution of the image is 512x512. From the given descriptions in the theoretical background, determine the name of each convolution kernel. Put this as the title of the subplot.

3. Plot the 2D and 3D Magnitude Spectrum of all the filters in #2.

4. Plot the 2D and 3D Magnitude Spectrum of all the filtered images in #2.

5. Using dog.png, filter the image using a 3x3 and 5x5 Gaussian Blur. Compare the results obtained when using approximate kernels and when using fspecial(). Show the filtered images in one figure and plot the 3D magnitude spectrum in another figure. Use subplot for both instances.

Along with this, the following MATLAB functions are utilized.

conv2(A, B)

The function returns the convolution of matrices A and B [9].

fftshift(X, dim)

The function rearranges the fourier transform x by shifting the zero-frequency component to the center of the array and operates along the dimension *dim* [10].

fft2(X)

The function returns the two-dimensional Fourier transform of X [11].

fspecial(“gaussian”, hsize, sigma)

The function returns a rotationally symmetric Gaussian lowpass filter of size *hsize* with standard deviation *sigma* [12].

imfilter(A, H)

The function filters matrix A with the filter H [13].

imread(x, y)

The function reads the image x with the file format y [14].

imshow(image)

The function displays the grayscale image of the *image* [15].

mat2gray(x)

The function converts image x to a grayscale image [16].

meshgrid(x, y)

The function creates a mesh grid or 2-D grid coordinates based on matrices x and y [17].

rgb2gray(x)

The function converts the RGB image x to grayscale [18].

uint8(x)

The function converts the values in x to type uint8 [19].

V. Result

1. Encode the following convolution matrices in MATLAB

$H_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$H_2 = \left(\frac{1}{16}\right) * \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	$H_3 = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$
$H_4 = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$	$H_5 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	$H_6 = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$
$H_7 = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	$H_8 = \begin{bmatrix} -2 & -1 & 0 \\ -1 & 1 & 1 \\ 0 & 1 & 2 \end{bmatrix}$	

```
H1 = [0 0 0;...
        0 1 0;...
        0 0 0];
H2 = (1/16)*...
        [1 2 1;...
        2 4 2;...
        1 2 1];
H3 = [0 -1 0;...
        -1 5 -1;...
        0 -1 0];
H4 = [1 0 -1;...
           2 0 -2;...
           1 0 -1];
H5 = [0 1 0;...
        1 -4 1;...
        0 1 0];
H6 = [1 0 -1;...
           2 0 -2;...
           1 0 -1];
H7 = [-1 -1 -1;...
        -1 8 -1;...
        -1 -1 -1];
H8 = [-2 -1 0;...
        -1 1 1;...
        0 1 2];
```

2. Using the grayscale version of the image dog.png, apply all the convolution kernels in #1. Show the original image and all the filtered images in one figure (use subplot). Ensure that the resolution of the image is 512x512. From the given descriptions in the theoretical background, determine the name of each convolution kernel. Put this as the title of the subplot.

```
%Acquire image and place on your current directory
%read the image using imread() and store it into a variable
I = imread('dog', 'png');
%Convert image to greyscale
Igs = rgb2gray(I);


%Filter image with all convolution kernels.
%Use function imfilter()
%Assure 512x512 image using uint8()
Idenfilconv = uint8(conv2(Igs, Iden, 'same'));
Idenfil = uint8(imfilter(Igs, Iden));
Igbfil = uint8(imfilter(Igs, Gblur));
Ishfil = uint8(imfilter(Igs, Sharp));
Ivsfil = uint8(imfilter(Igs, VertSob));
Ihsfil = uint8(imfilter(Igs, HorSob));
Ilapfil = uint8(imfilter(Igs, Lap));
Ioutlfil = uint8(imfilter(Igs, Outl));
Iembofil = uint8(imfilter(Igs, Embo));


%Show all the filtered images in one figure
figure(1);
subplot(331); imshow(Igs); title('Original Image')
subplot(332); imshow(Idenfil); title('Identity')
subplot(333); imshow(Igbfil); title('Gaussian Blur')
subplot(334); imshow(Ishfil); title('Sharpen')
subplot(335); imshow(Ivsfil); title('Vertical Sobel')
subplot(336); imshow(Ihsfil); title('Horizontal Sobel')
subplot(337); imshow(Ilapfil); title('Laplacian')
subplot(338); imshow(Ioutlfil); title('Outline')
subplot(339); imshow(Iembofil); title('Emboss')
```

- H1 = Identity
- H2 = Gaussian Blur
- H3 = Sharpen
- H4 = Sobel Operator for Vertical Edge Detection
- H5 = Laplacian Operator
- H6 = Sobel Operator for Horizontal Edge Detection
- H7 = Outline Kernel
- H8 = Emboss Kernel

Expected Output:

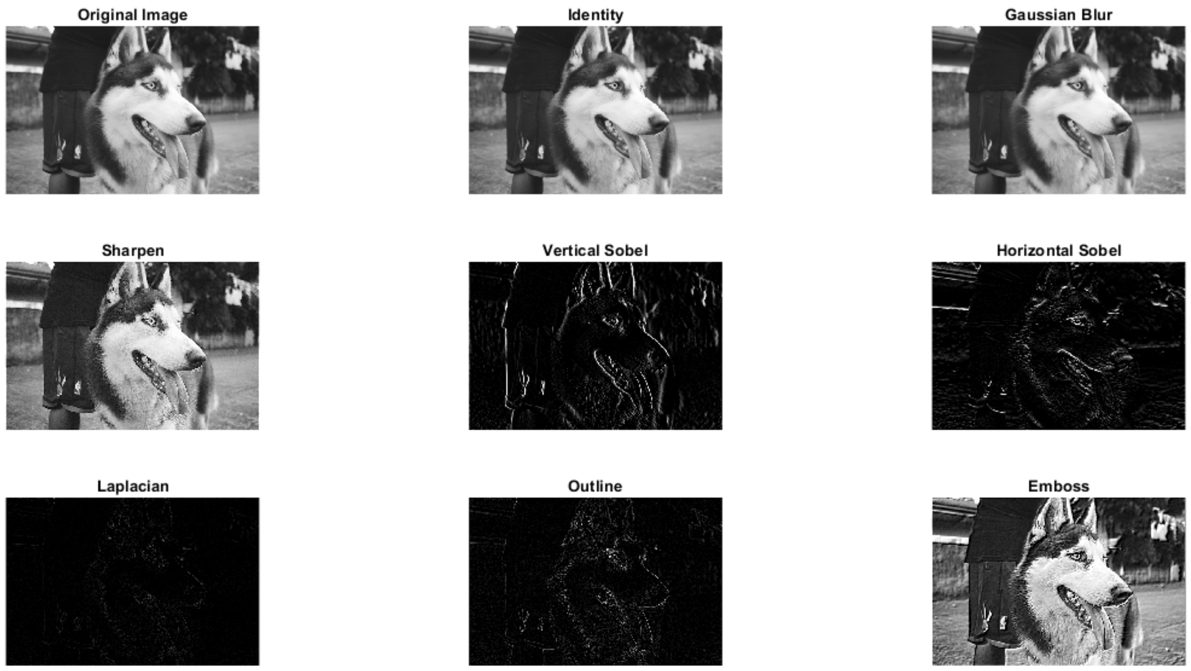


Figure 5.1. Subplot of Original Image with Filtered Images.

3. Plot the 2D and 3D Magnitude Spectrum of all the filters in #2.

```
%Plot the 2D Magnitude Spectrum of All Filters
%Initialize a function to calculate and plot the 2D magnitude spectrum
figure(2);
subplot(331); imgmagspec2D(Iden); title('Identity')
subplot(332); imgmagspec2D(Gblur); title('Gaussian Blur')
subplot(333); imgmagspec2D(Sharp); title('Sharpen')
subplot(334); imgmagspec2D(VertSob); title('Vertical Sobel')
subplot(335); imgmagspec2D(HorSob); title('Horizontal Sobel')
subplot(336); imgmagspec2D(Lap); title('Laplacian')
subplot(337);imgmagspec2D(Outl); title('Outline')
subplot(338); imgmagspec2D(Embo); title('Emboss')

%Plot the 3D Magnitude Spectrum of All Filters
%Initialize a function to calculate and plot the 3D magnitude spectrum
figure(3);
subplot(331); imgmagspec3D(Iden); title('Identity')
subplot(332); imgmagspec3D(Gblur); title('Gaussian Blur')
subplot(333); imgmagspec3D(Sharp); title('Sharpen')
subplot(334); imgmagspec3D(VertSob); title('Vertical Sobel')
subplot(335); imgmagspec3D(HorSob); title('Horizontal Sobel')
subplot(336); imgmagspec3D(Lap); title('Laplacian')
subplot(337);imgmagspec3D(Outl); title('Outline')
subplot(338); imgmagspec3D(Embo); title('Emboss')
```

Expected Output:

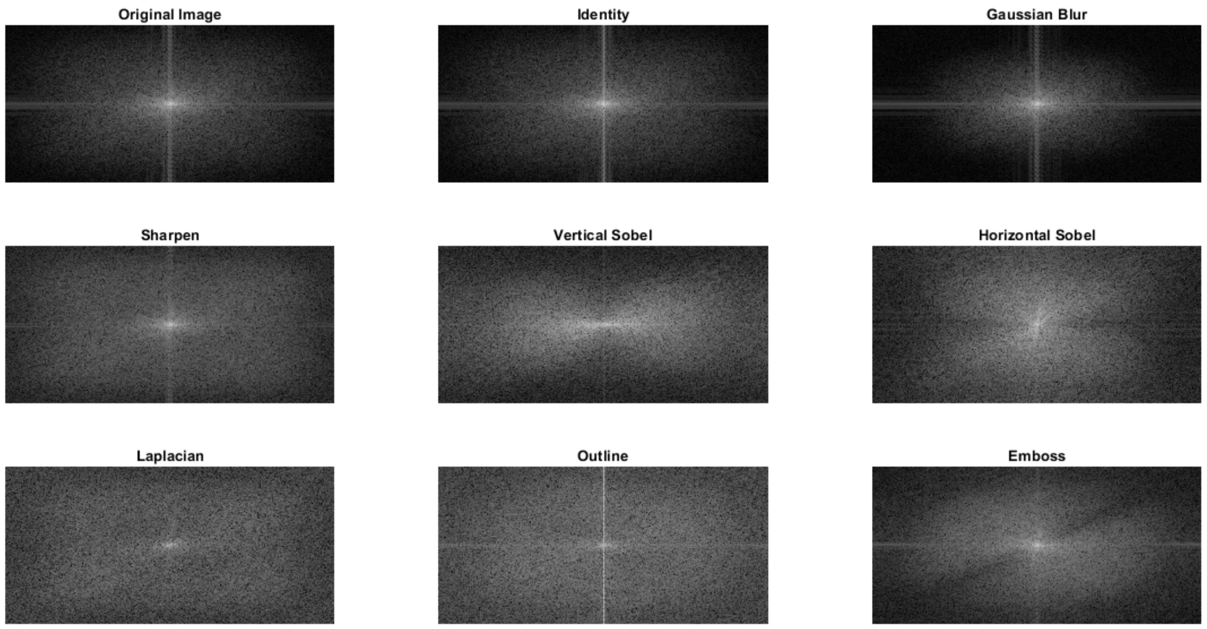


Figure 5.2. 2D Magnitude Spectrum of All Filters.

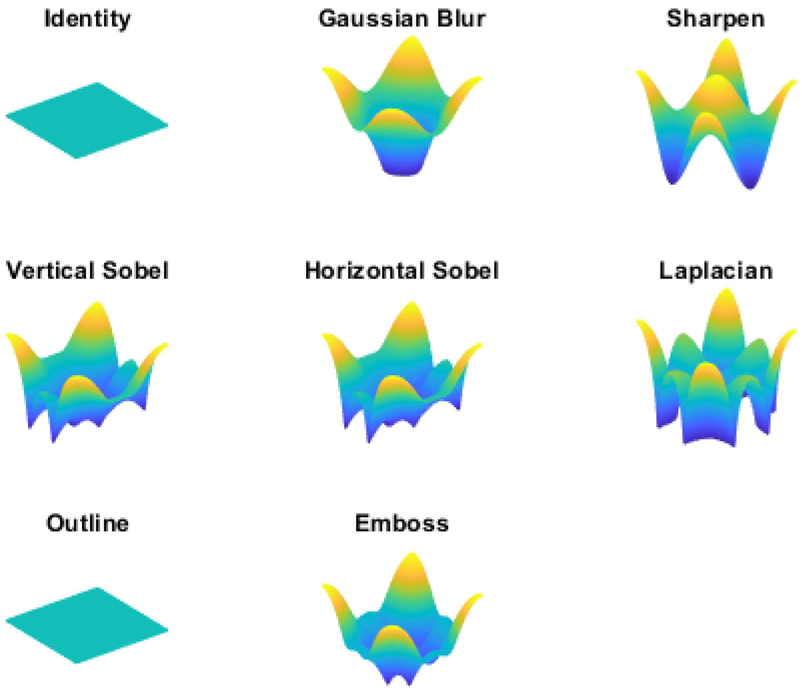


Figure 5.3. 3D Magnitude Spectrum of All Filters.

4. Plot the 2D and 3D Magnitude Spectrum of all the filtered images in #2.

```
%These are the functions used for the section. Note that this is placed at the
end of the file as prescribed by MATLAB's convention in user-defined functions.

function imgmagspec2D(img) %image must be a convolution kernel or a grayscale
image
F = fftshift(fft2(double(mat2gray(img)), 512, 512));
Ilog = log(1+abs(F));
colormap(gray); imagesc(Ilog); axis off
end

function imgmagspec3D(img)
H=fftshift(fft2(double(mat2gray(img)), 512, 512));
Hlog=log(1+abs(H));
u=-256:255;
v=-256:255;
[u,v]=meshgrid(u,v);
mesh(u,v,[Hlog(257:512,257:512),Hlog(257:512,1:256);
Hlog(1:256,257:512) Hlog(1:256,1:256)]); axis off
end


%Plot the 2D Magnitude Spectrum of Filtered Images
figure(4);
subplot(331); imgmagspec2D(Igs); title('Original Image')
subplot(332); imgmagspec2D(Idenfil); title('Identity')
subplot(333); imgmagspec2D(Igbfil); title('Gaussian Blur')
subplot(334); imgmagspec2D(Ishfil); title('Sharpen')
subplot(335); imgmagspec2D(Ivsfil); title('Vertical Sobel')
subplot(336); imgmagspec2D(Ihsfil); title('Horizontal Sobel')
subplot(337); imgmagspec2D(Ilapfil); title('Laplacian')
subplot(338); imgmagspec2D(Ioutlfil); title('Outline')
subplot(339); imgmagspec2D(Iembofil); title('Emboss')

%Plot the 3D Magnitude Spectrum of Filtered Images
figure(5);
subplot(331); imgmagspec3D(Igs); title('Original Image')
subplot(332); imgmagspec3D(Idenfil); title('Identity')
subplot(333); imgmagspec3D(Igbfil); title('Gaussian Blur')
subplot(334); imgmagspec3D(Ishfil); title('Sharpen')
subplot(335); imgmagspec3D(Ivsfil); title('Vertical Sobel')
subplot(336); imgmagspec3D(Ihsfil); title('Horizontal Sobel')
subplot(337); imgmagspec3D(Ilapfil); title('Laplacian')
subplot(338); imgmagspec3D(Ioutlfil); title('Outline')
subplot(339); imgmagspec3D(Iembofil); title('Emboss')
```

Expected Output:

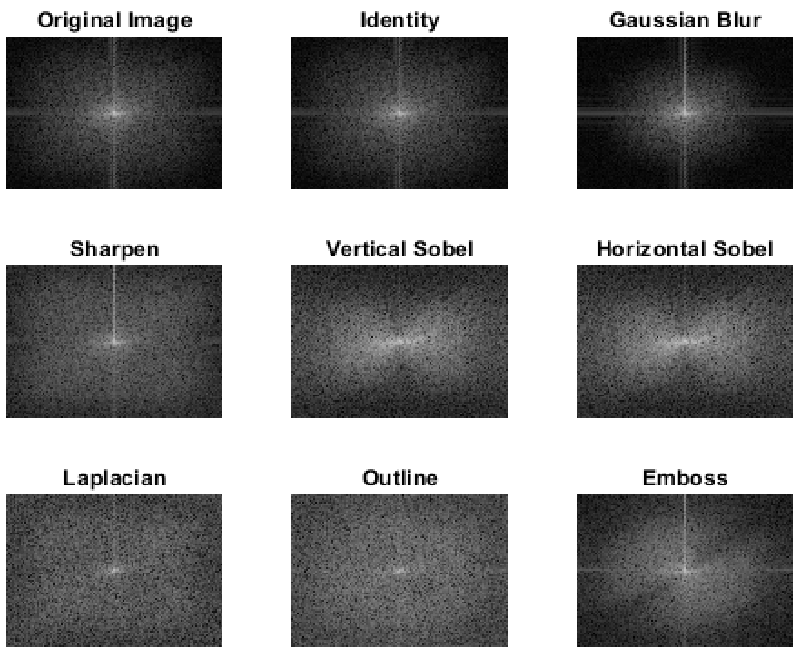


Figure 5.4. 2D Magnitude Spectrum of All Filtered Images.

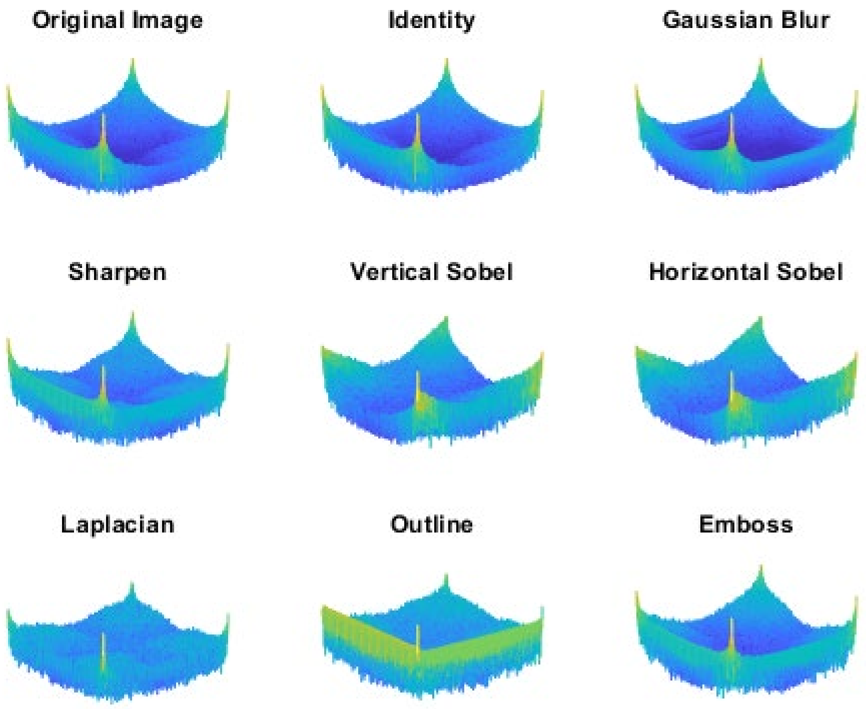


Figure 5.5. 3D Magnitude Spectrum of All Filtered Images.

5. Using dog.png, filter the image using a 3x3 and 5x5 Gaussian Blur. Compare the results obtained when using approximate kernels and when using fspecial(). Show the filtered images in one figure and plot the 3D magnitude spectrum in another figure. Use subplot for both instances.

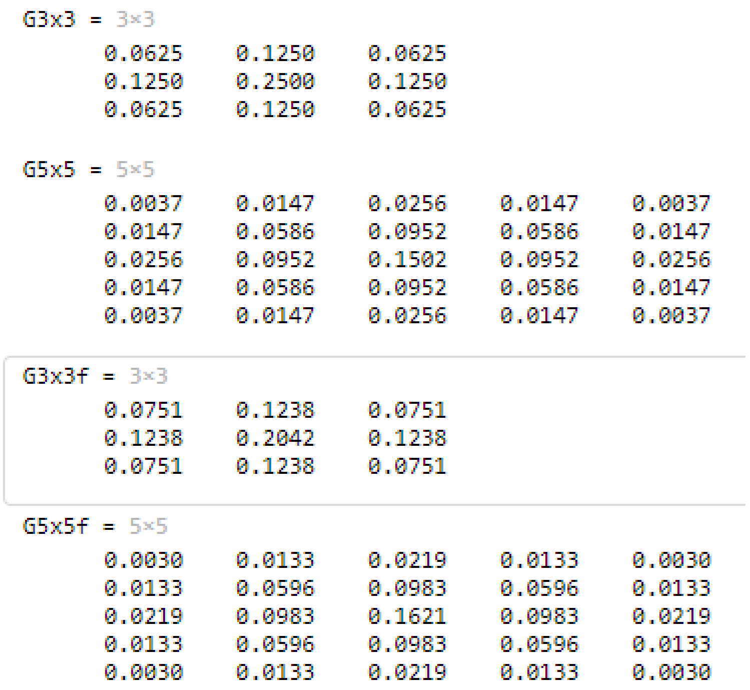
```
%Initialize Gaussian Kernels. Use the approximate kernels and
%kernels obtained from fspecial
G3x3 = (1/16)*...
        [1 2 1;...
        2 4 2;...
        1 2 1]


G5x5 = (1/273)*[1 4 7 4 1;...
                4 16 26 16 4;...
                7 26 41 26 7;...
                4 16 26 16 4;...
                1 4 7 4 1]
G3x3f = fspecial("gaussian", [3 3], 1)
G5x5f = fspecial("gaussian", [5 5], 1)
%Filter image using approximate 3x3 and 5x5 kernels,
%and 3x3 and 5x5 gaussian filter functions
Ig3x3app = uint8(imfilter(Igs, G3x3));
Ig5x5app = uint8(imfilter(Igs, G5x5));
Ig3x3f = uint8(imfilter(Igs, G3x3f));
Ig5x5f = uint8(imfilter(Igs, G5x5f));


%Show filtered images in one figure
figure(6);
subplot(221); imshow(Ig3x3app); title('Approximate 3x3 Gaussian Kernel')
subplot(222); imshow(Ig5x5app); title('Approximate 5x5 Gaussian Kernel')
subplot(223); imshow(Ig3x3f); title('Exact 3x3 Gaussian Kernel')
subplot(224); imshow(Ig5x5f); title('Exact 5x5 Gaussian Kernel')


%Plot the 3D magnitude spectrum of the images in one figure
figure(7);
subplot(221); imgmagspec3D(G3x3);  title('Approximate 3x3 Gaussian Kernel')
subplot(222); imgmagspec3D(G5x5); title('Approximate 5x5 Gaussian Kernel')
subplot(223); imgmagspec3D(G3x3f); title('Exact 3x3 Gaussian Kernel')
subplot(224); imgmagspec3D(G5x5f); title('Exact 5x5 Gaussian Kernel')
```

Expected Output:



```
G3x3 = 3×3
      0.0625    0.1250    0.0625
      0.1250    0.2500    0.1250
      0.0625    0.1250    0.0625

G5x5 = 5×5
      0.0037    0.0147    0.0256    0.0147    0.0037
      0.0147    0.0586    0.0952    0.0586    0.0147
      0.0256    0.0952    0.1502    0.0952    0.0256
      0.0147    0.0586    0.0952    0.0586    0.0147
      0.0037    0.0147    0.0256    0.0147    0.0037

G3x3f = 3×3
      0.0751    0.1238    0.0751
      0.1238    0.2042    0.1238
      0.0751    0.1238    0.0751

G5x5f = 5×5
      0.0030    0.0133    0.0219    0.0133    0.0030
      0.0133    0.0596    0.0983    0.0596    0.0133
      0.0219    0.0983    0.1621    0.0983    0.0219
      0.0133    0.0596    0.0983    0.0596    0.0133
      0.0030    0.0133    0.0219    0.0133    0.0030
```

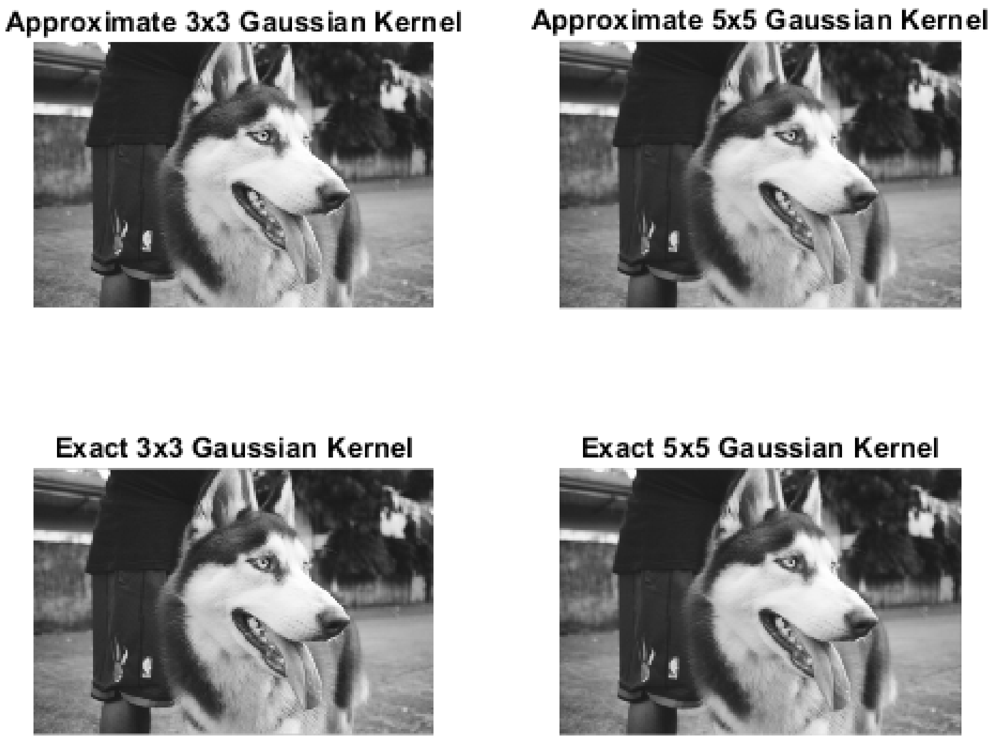

Approximate 3x3 Gaussian Kernel
Approximate 5x5 Gaussian Kernel
Exact 3x3 Gaussian Kernel
Exact 5x5 Gaussian Kernel

Figure 5.6. Output Images of Approximate and Exact 3x3 and 5x5 Gaussian Blur Kernels.

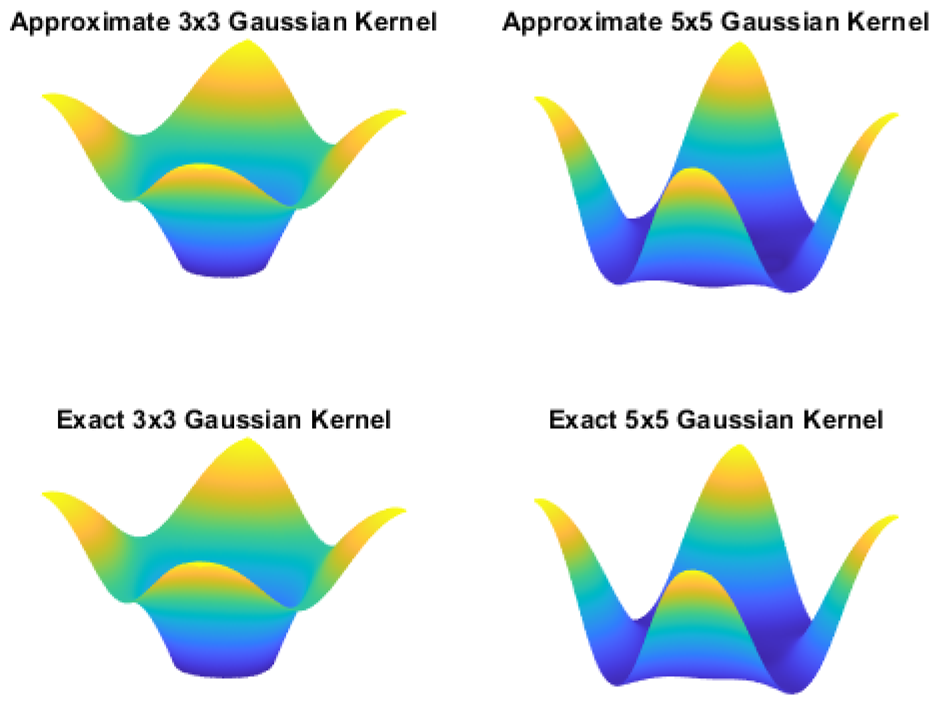


Figure 5.7. 3D Magnitude Spectrum of Approximate and Exact 3x3 and 5x5 Gaussian Blur Kernels.

VI. Discussion

The construction of the laboratory manual will not be discussed given that the sections of introduction and theoretical considerations were already able to give sufficient background to the topic. Instead, the authors will discuss the solution manual to describe the thought process behind the code and to explain the logic behind the expected outputs.

The exercise proper is divided into three main parts: (1) exploration of different convolution matrices and identification which is covered by Tasks 1 and 2; (2) comparison of plots of the 2D and 3D magnitude spectra of all the filters and filtered images which is covered by Tasks 3 and 4; and (3) comparison of results of approximate kernels and actual kernels for 3x3 and 5x5 Gaussian Blur which is covered by Task 5.

For the first part, eight convolution matrices were given. Task 1 tells the user to encode these matrices in MATLAB. The image *dog.png* is read using the function *imread()* and stored in matrix I. Afterwards, the values of the matrix I is converted to its grayscale equivalent using the *rgb2gray()* function and stored in Igs. The image undergoes filtering with the corresponding convolution kernel. In a subplot, the corresponding outputs were shown using *imshow()*.

Upon comparison with the lecture proper and the descriptions given in the laboratory manual, the students would be able to determine the names of the corresponding kernels. Delving deeper into each matrix (see Figure 5.1), first, the identity matrix appears the same as the original image, thus the name identity. Next, the Gaussian Blur returns a less sharp image as some details appear to be less sharp compared to the original image. The third image shows a sharpened image since its edges and textures are more focused and sharper. The vertical and horizontal Sobel operators are easily distinguishable since the direction of the edges are more prominent in vertical and horizontal directions as implied by the respective names of the masks. Outline is also easy to describe as it shows the edges of the photo. Emboss is also a standout as it is the only image that makes the existing image more 3D as highlights and shadows are added to the image. This only leaves the student to put Laplacian for the sixth image since it is the only image that is quite difficult to describe since it is an edge detector used to compute second derivatives of an image.

For the second part, the code for the magnitude spectra in both 2D and 3D are already obtainable through the laboratory manual. Putting this as a user-defined function, the 2D magnitude spectra are shown in the 2D magnitude spectra of the filters (see Figure 5.2) and the filtered images (see Figure 5.4) are the same. However, there are differences in the result of the 3D magnitude spectra of the filters and filtered images as seen in Figures 5.3 and 5.5.

For the last part, the approximate kernels are defined directly as given in the theory while the exact kernels were given through the function *fspecial()*. Applying the same process as the previous numbers, the two matrices are then convolved using *imfilter()* and converted to unsigned integers of 8 bits using the function *uint8()*. The resulting images are then shown using *imshow()* while the 3D magnitude spectrum of each image is shown through the user-defined function *imgmagspec3D()*. Although there are disparities between the values of the exact and the approximate kernels, the resulting images in Figure 5.6 are almost the same which goes to show that the approximation achieves similar, if not the same, results. This is also proven by the results in Figure 5.7 which shows the same plot for the approximate and exact values for both 3x3 and 5x5 Gaussian kernels.

VII. Conclusion

The authors were able to create a laboratory manual that allows students to further explore different convolution kernels, compare 2D and 3D magnitude spectra plot of the convolution kernels and filtered images, and observe the effects of increasing the dimensions of the matrix for the Gaussian blur kernel. Aside from this, the authors were able to create an accompanying solution set for the exercise proper.

Due to the scope and limitations set, the authors are unable to assess the effectiveness of the laboratory manual in teaching the topics to the students and the effectiveness of the exercise proper to assess the students' knowledge in the subject matter. The authors suggest testing the laboratory manual and exercise proper with a selected number of respondents to check if it meets the required educational standards, especially in terms of the clarity of the instructions given, before integrating this lesson as part of the course proper. The authors also suggest in getting insights from professors and students alike in order to improve the existing outputs.

VIII. Authors Contribution

Member	**Contribution**
John Ernesto Amadora Jr.	Abstract and Results Solution Set Presentation
Rafael Dominic Montaño	Introduction and Theoretical Consideration Laboratory Manual Presentation
Alyssa Joie F. Tablada	Methodology, Discussion, and Conclusion Laboratory Manual Poster Presentation

IX. References

[1] V. Powell, "Image Kernels Explained Visually," [Online]. Available: https://setosa.io/ev/image-kernels/. [Accessed 28 March 2023].

[2] P. Ganesh, "Types of Convolution Kernels: Simplified | by Prakhar Ganesh | Towards Data Science," [Online]. Available: https://towardsdatascience.com/types-of-convolution-kernels-simplified-f040cb307c37. [Accessed 02 April 2023].

[3] "Chapter 6 - Convolution | The Scientist and Engineer's Guide to Digital Signal Processing," [Online]. Available: https://www.analog.com/media/en/technical-documentation/dsp-book/dsp_book_ch6.pdf. [Accessed 02 April 2023].

[4] "convolutional neural networks - When should I use 3D convolutions | StackExchange," [Online]. Available: https://ai.stackexchange.com/questions/13692/when-should-i-use-3d-convolutions#:~:text=3D%20convolutions%20should%20be%20used,images%20and%20medical%20image%20segmentation. [Accessed 02 April 2023].

[5] "image - What is the "do-nothing" convolution kernel - Stack Overflow," [Online]. Available: https://stackoverflow.com/questions/5824704/what-is-the-do-nothing-convolution-kernel. [Accessed 02 April 2023].

[6] "Blurring Images - Image Processing with Python," [Online]. Available: https://datacarpentry.org/image-processing/06-blurring/. [Accessed 02 April 2023].

[7] The blog at the bottom of the sea, "Image Sharpening Convolution Kernels," [Online]. Available: https://blog.demofox.org/2022/02/26/image-sharpening-convolution-kernels/. [Accessed 02 April 2023].

[8] AI Shack, "The Sobel and Laplacian Edge Detectors," [Online]. Available: https://aishack.in/tutorials/sobel-laplacian-edge-detectors/. [Accessed 02 April 2023].

[9] MathWorks, "2-D Convolution - MATLAB conv2," [Online]. Available: https://www.mathworks.com/help/matlab/ref/conv2.html. [Accessed 28 March 2023].

[10] MathWorks, "Shift zero-frequency component to center of spectrum - MATLAB fftshift," [Online]. Available: https://www.mathworks.com/help/matlab/ref/fftshift.html. [Accessed 28 March 2023].

[11] MathWorks, "2-D fast Fourier transform - MATLAB fft2," [Online]. Available: https://www.mathworks.com/help/matlab/ref/fft2.html. [Accessed 28 March 2023].

[12] MathWorks, "Create predefined 2-D filter - MATLAB fspecial," [Online]. Available: https://www.mathworks.com/help/images/ref/fspecial.html. [Accessed 28 March 2023].

[13] MathWorks, "N-D filtering of multidimensional images - MATLAB imfilter," [Online]. Available: https://www.mathworks.com/help/images/ref/imfilter.html. [Accessed 28 March 2023].

[14] MathWorks, "Read image from graphics file - MATLAB imread," [Online]. Available: https://www.mathworks.com/help/matlab/ref/imread.html. [Accessed 28 March 2023].

[15] MathWorks, "Display image - MATLAB imshow," [Online]. Available: https://www.mathworks.com/help/images/ref/imshow.html. [Accessed 28 March 2023].

[16] MathWorks, "Convert matrix to grayscale image - MATLAB mat2gray," [Online]. Available: https://www.mathworks.com/help/images/ref/mat2gray.html. [Accessed 08 March 2023].

[17] MathWorks, "2-D and 3-D grids - MATLAB meshgrid," [Online]. Available: https://www.mathworks.com/help/matlab/ref/meshgrid.html. [Accessed 28 March 2023].

[18] MathWorks, "Convert RGB image or colormap to grayscale - MATLAB rgb2gray," [Online]. Available: https://www.mathworks.com/help/matlab/ref/rgb2gray.html. [Accessed 28 March 2023].

[19] MathWorks, "8-bit unsigned integer arrays - MATLAB uint8," [Online]. Available: https://www.mathworks.com/help/matlab/ref/uint8.html. [Accessed 28 March 2023].