Image Convolution Kernels

Laboratory Manual for LBYEC4A Final Project

I. 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

II. THEORETICAL BACKGROUND

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.

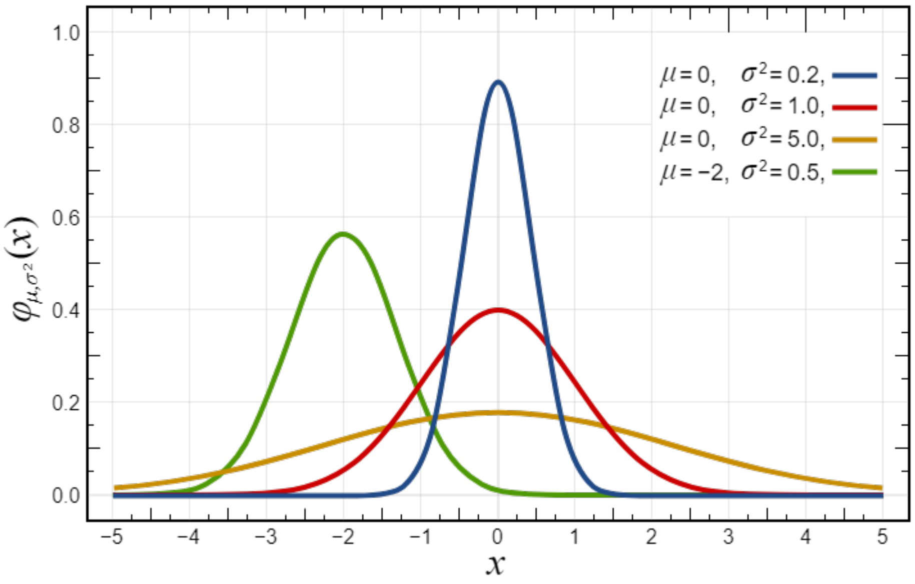

Figure 2.1. Gaussian Bell Curve [6].

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.

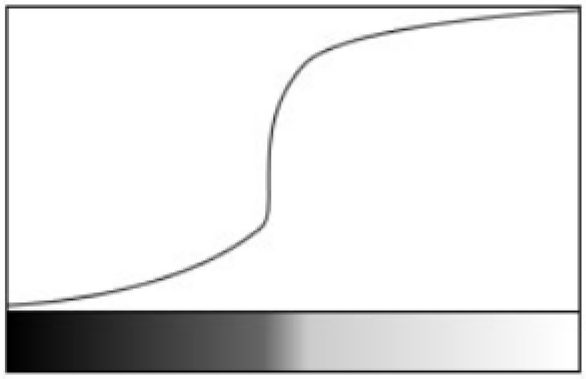

Figure 2.2. Pixel Intensity with Regard to Grayscale Color from Black to White [8].

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.

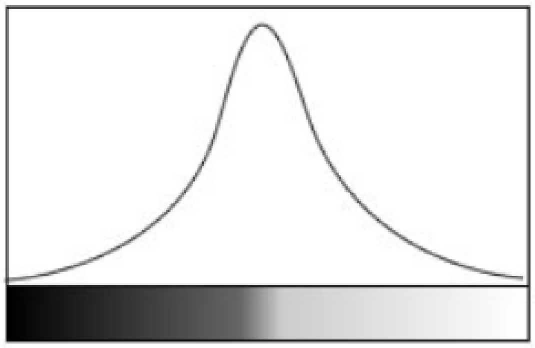

Figure 2.3. First Derivative of Pixel Intensity with Regard to Grayscale Color from Black to White [8].

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.

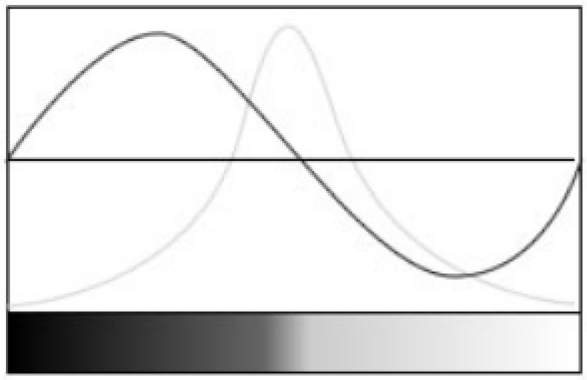

Figure 2.4. Second Derivative of Pixel Intensity with regard to Grayscale Color from Black to White [8].

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).

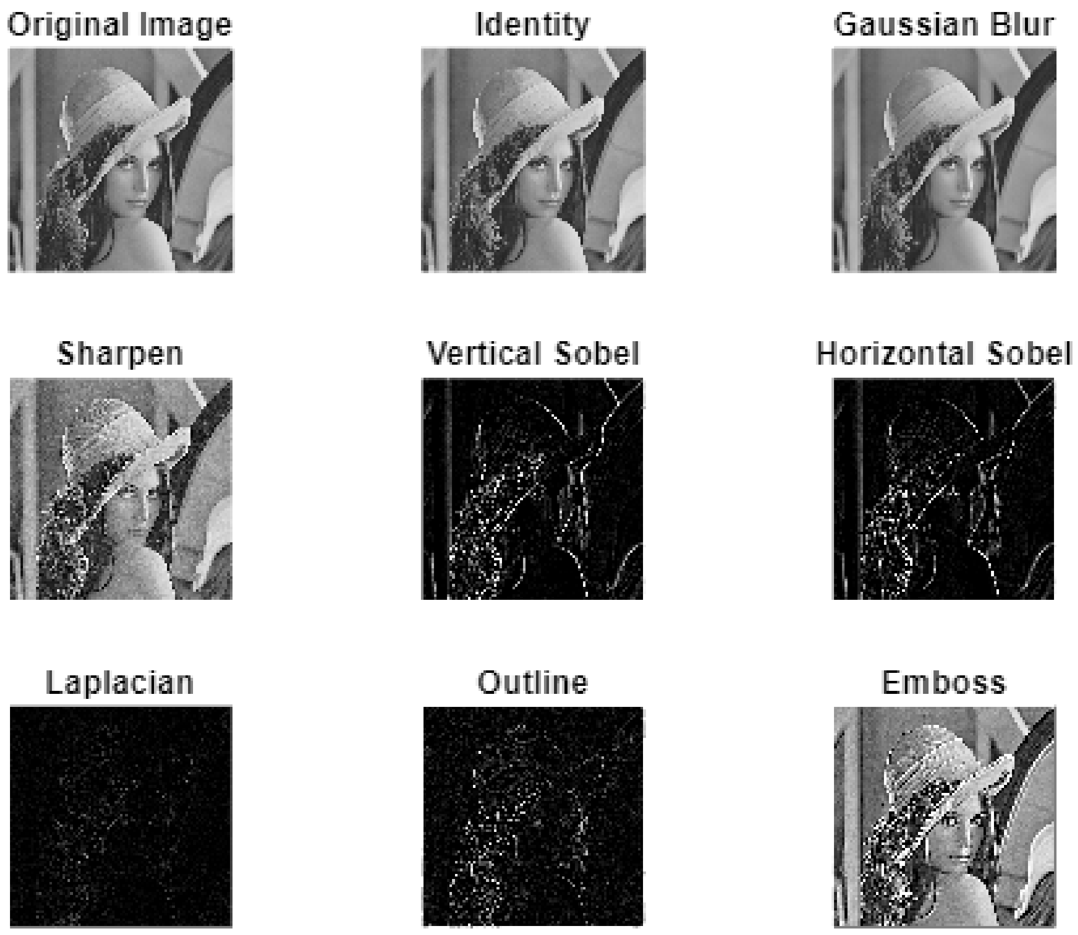

Figure 2.5. Test Images for the Different Features Extracted by the Listed Convolution Kernels.

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
```

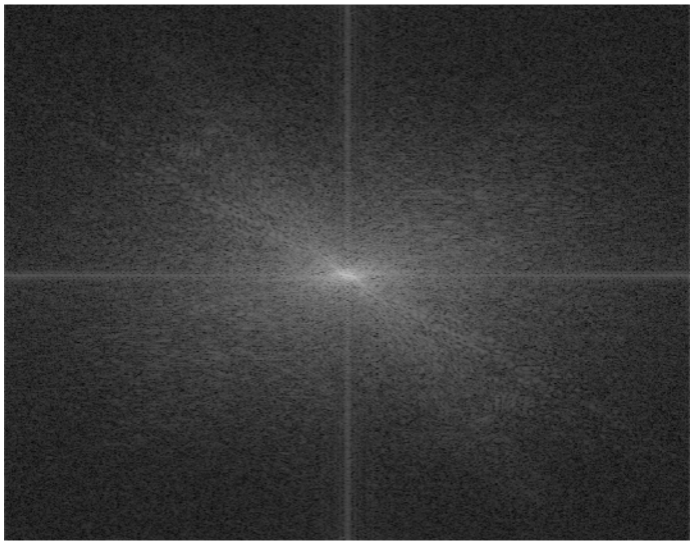

Figure 2.6. 2D Magnitude Spectrum Representation of lena.tiff.

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)]);
```

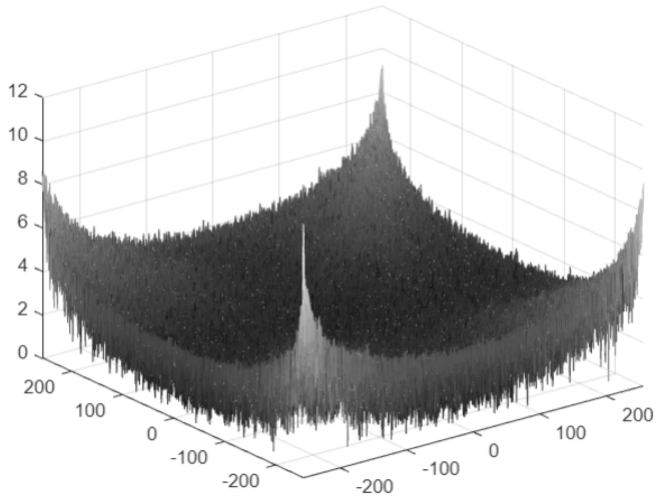

Figure 2.7. 3D Magnitude Spectrum Representation of lena.tiff.

III. MATLAB FUNCTIONS

This section is allotted to give a brief description of the functions to be used in the conduct of the exercise. This list does not include user-defined functions.

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].

IV. EXERCISE PROPER

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.

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].