EE-3220 LABORATORY

Week 4

2-D Convolution and Image Processing

Goal – Investigate the ability to process and enhance images by treating the image as a function of two independent variables and convolving with a 2-dimensional FIR filter.

Materials - Laptop computer with MATLAB

Background – The general form of the function that describes an image is:

$$x(n,m),$$

where n and m represent the spatial position in a 2-dimensional plane and x represents the image intensity/color at that spatial location.

For a monochromatic, black-and-white image (grayscale), the value of the function is real and positive. Often a grayscale image is encoded with 8 bits, which provides 256 levels of image intensity, so $x(n,m)$ takes on integer values from 0 to 255.

In MATLAB an image is stored as a matrix where the index n denotes the row, or vertical, position of the image, and m denotes the column, or horizontal, position. The size of the matrix used to store the image is dependent on the number of picture elements (pixels) of the image.

Download 'Peppers.bmp' to your MATLAB directory. This classic image is used to test image processing algorithms. We can load and view images in MATLAB with the **imread()** and **imshow()** commands, respectively. Try the following:

```
x = imread('Peppers.bmp');
imshow(x)
```

The image is now stored in a matrix named **x.** The elements of the matrix represent grayscale values associated with each picture element, or pixel. It is standard practice to represent the pixel in the upper left corner as $x(1,1)$ and the pixel in the lower right corner as $x(N,M)$ for images that are N pixels tall, by M pixels wide.

1. *What are the dimensions of this image and how many pixels make up this image? The size() and prod() functions can help. Read the documentation for these functions.*

2. *Calculate how many bytes (recall that a byte is 8 bits) of storage are required for this image. Compare this with the number of bytes in the file, which you can find with* `getfield(dir('Peppers.bmp'),'bytes')`. *These numbers are **slightly** different; what could explain this? Search online for information about the BMP file format for ideas.*

3. *What are the grayscale values of x(20,20) and x(15,30)? What happens in MATLAB if you add together enough 8-bit grayscale values to exceed the maximum 8-bit unsigned value?*

Filtering is a common processing technique applied to images and 2-dimensional FIR filters are often used to enhance aspects of an image. Below is the general form of the two-dimension convolution sum,

$$y(n,m) = \sum_k \sum_l x(k,l)h(n-k,m-l)$$

$$= \sum_k \sum_l h(k,l)x(n-k,m-l)$$

or in compact notation,

$$y(n,m) = x(n,m) \circledast h(n,m)$$
$$= h(n,m) \circledast x(n,m),$$

where x is the input image, h is the impulse response function of the filter, and y is the filtered image.

When a 2-dimensional FIR filter is used, h consists solely of weighted 2-dimensional impulses shifted in location. For example $\delta(n-2, m+5)$ represents a 2-dimensional impulse function consisting of all 0s except for a single 1 located at $n = 2$ and $m = -5$.

Often the size of the FIR filter is limited to a 3-by-3 "kernel" represented as:

$$h(n,m) = \sum_{k=-1}^{1} \sum_{l=-1}^{1} b_{k,l}\delta(n-k,m-l)$$

The filtering kernel, h, can be represented as a matrix with coefficients equal to:

$b_{-1,-1}$	$b_{-1,0}$	$b_{-1,1}$
$b_{0,-1}$	$b_{0,0}$	$b_{0,1}$
$b_{1,-1}$	$b_{1,0}$	$b_{1,1}$

The kernel, h, is then convolved with the image using the convolution sum defined above and the resulting output pixel is computed as:

$$y(n,m) = b_{1,1}x(n-1,m-1) + b_{1,0}x(n-1,m) + b_{1-,1}x(n-1,m+1) + b_{0,1}x(n,m-1)$$
$$+ b_{0,0}x(n,m) + b_{0,-1}x(n,m+1) + b_{-1,1}x(n+1,m-1) + b_{-1,0}x(n+1,m)$$
$$+ b_{-1,-1}x(n+1,m+1)$$

Graphically, we can illustrate 2-dimensional convolution with a kernel as shown in the Figure 1. In this example the output pixel located at **n = 3**, **m=3** is calculated as:

$$y(3,3) = b_{1,1}x(2,2) + b_{1,0}x(2,3) + b_{1-,1}x(2,4) + b_{0,1}x(3,2) + b_{0,0}x(3,3) + b_{0,-1}x(3,4)$$
$$+ b_{-1,1}x(4,2) + b_{-1,0}x(4,3) + b_{-1,-1}x(4,4)$$

The remaining output pixels in the 3rd row are calculated by sliding the filter kernel horizontally across the input image and computing the convolution sum at each position. The entire image is processed in a similar manner by sliding the kernel one row down and repeating the process. Note the subscripts on the filter kernel are reversed. This is consistent with "flipping" the function when computing the convolution sum.

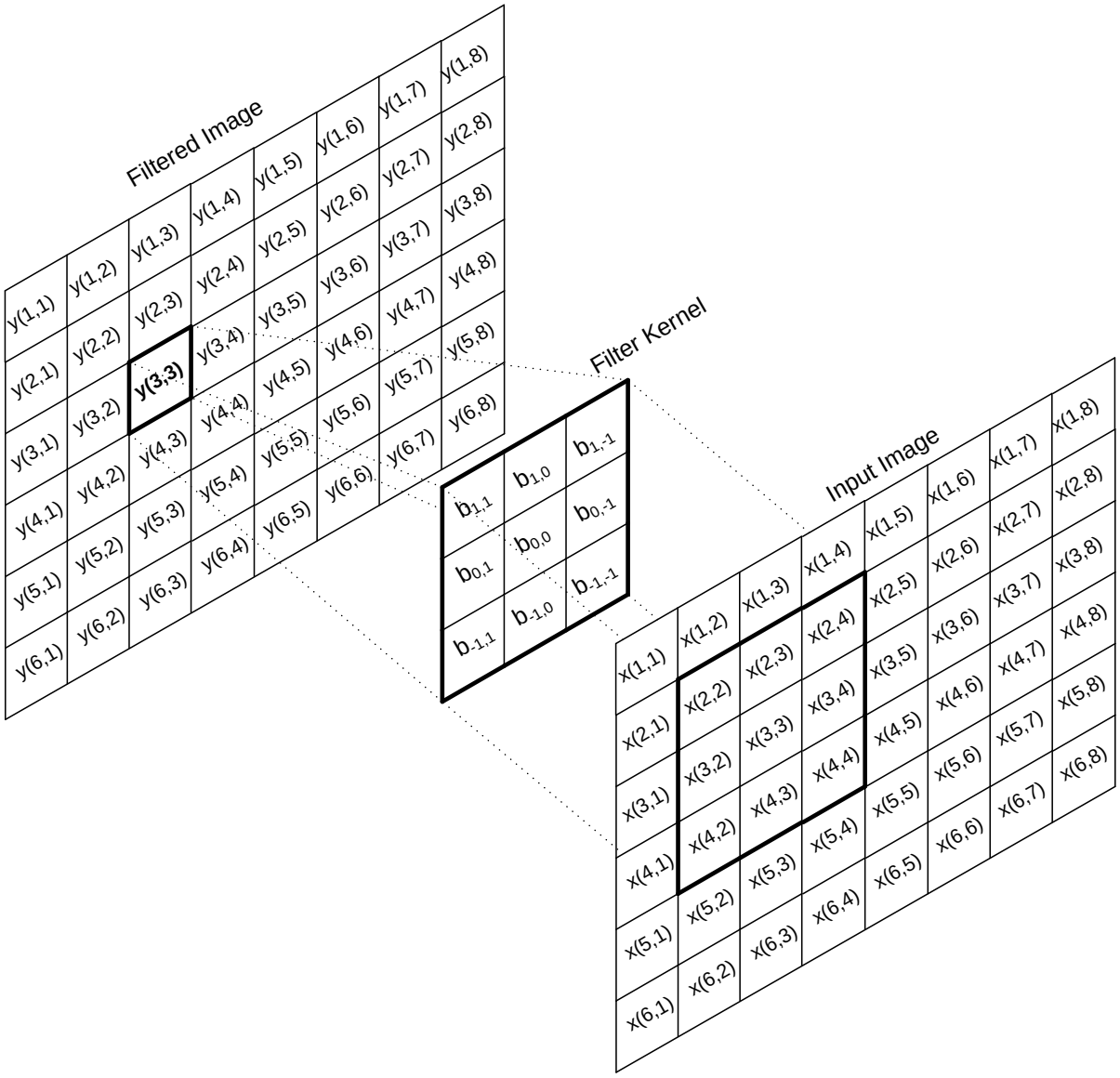

Figure 1. Graphical representation of two-dimensional convolution for image processing.

One example of a 3-by-3 image processing filter is the moving average kernel where each of the coefficients consists of a value of 1/9. The output pixel for this filter is an average of the input pixel plus the eight adjacent pixels. The kernel is given by $h =$

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

Create this FIR filter kernel in MATLAB with the following command

```
N = 3; h = ones(N)/(N*N);
```

The convolution between the filter kernel and the image can be implemented with nested "for" loops. However, it is much easier to use the function **imfilter(input_image, filter_kernel)** in MATLAB. This function handles the processing of the two-dimensional convolution sum. For example, type

```
y = imfilter(x,h);
```

The matrix **y** contains the pixels of the filtered image. Compare the unfiltered image, **x**, with the filtered image **y**.

```
subplot(2,1,1), imshow(x), title('Original Image')
subplot(2,1,2), imshow(y), title('Filtered Image')
```

4. *Provide the plots of both images and describe any visual differences you can detect between the two.*

Another popular filtering kernel is $h =$

-1	-1	-1
-1	9	-1
-1	-1	-1

This kernel results in an output image that emphasizes differences between the center pixel and the adjacent pixels. Note that the most negatively indexed element, $b_{-1,-1}$ in this case, has its indices shifted to 1,1 to satisfy MATLAB's indexing rules. Filter the original image with this kernel.

5. *Provide the plots of the original image and the filtered image for the above kernel. Describe any visual differences you can detect.*

Another filtering kernel to investigate is $h =$

-1	-1	-1
-1	8	-1
-1	-1	-1

6. *For the above kernel, what will the output pixel equal when the adjacent pixels all have the same intensity (grayscale) value, and what does this output value represent?*

One final set of kernels is $h\_1 =$

-1	2	-1
-1	2	-1
-1	2	-1

and $h\_2 =$

-1	-1	-1
2	2	2
-1	-1	-1

7. *Create two images, one filtered with h_1, and one filtered with h_2. Compare the results. Look carefully. You may find it helpful to plot them next to one another as subplots. What does each filter do? What does each accentuate?*

Submittal

The submittal should follow all instructions on the provided grading checklist and cover sheet.

The standard summary section is not required this week.

Additional Problems

Submit answers to the following additional problems with your lab. Be sure to include your MATLAB code and any graphs.

Consider T_2 and T_5 from Problem 2.11.

1. Prove that T_2 is linear. "Prove" means show that the statement is true in general for any inputs.
2. Prove that T_5 is not time-invariant.

Let x1 = [3 7 5] and x2 = [4 2 -3], where both start at n=0. Perform the following tests by writing MATLAB code and comment on whether your results are consistent with the above.

3. Test the systems for linearity using x1+x2.
4. Test the systems for time-invariance using x1 and a delay of 4 samples.