Activity 3: Linear Filters

Primary Goal: Learn how to create and apply linear filters.
Secondary Goal: Identify different varieties of linear filters and understand their applications.

A linear filter is an operation that can be expressed as a convolution on an image f:

$$g = f * w$$

The weights matrix w is called the filter or kernel.

Linear filters have 3 main applications:

1. Blurring / smoothing an image
2. Deblurring / sharpening an image
3. Detecting edges or other image features

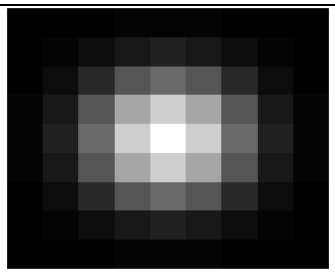
1.) Blurring / Smoothing an Image

We can create a $N \times N$ Gaussian filter with a specified mean μ and standard deviation σ with the Matlab `fspecial` command:

```
w = fspecial ('gaussian', [N,N], σ);
```

We can then apply a filter w to an image A with Neumann boundary conditions using the command:

```
B = imfilter (A, w);
```



Gaussian Filter

By default, `imfilter` uses Dirichlet (zero) boundary conditions. You can obtain slightly better results at the image boundary by adding the parameter `'replicate'` to the call to `imfilter` to use Neumann (reflective) boundary conditions.

Load the built-in "peppers" image.

```
A = imread('peppers.png');
```

a.) Apply 9x9 Gaussian filters to the peppers image with standard deviations $\sigma = 0.8$ and $\sigma = 2.5$. What happens as σ gets larger?

`The image becomes more blurry`

b.) Create a motion blur filter with movement distance 10 pixels and angle 120° by typing:

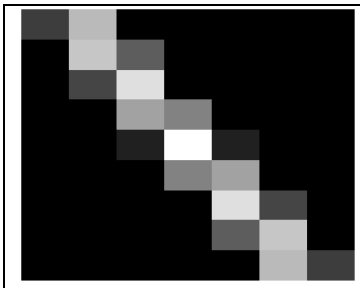
```
w = fspecial ('motion', 10, 120);
```

Apply this filter to the original peppers image. How does this blur differ from the Gaussian filter blur?

`It blurs the image at specified angel.`

c.) Increase the movement length from 10 to 30 pixels. What happens to the image?

`The amount of blur increases`

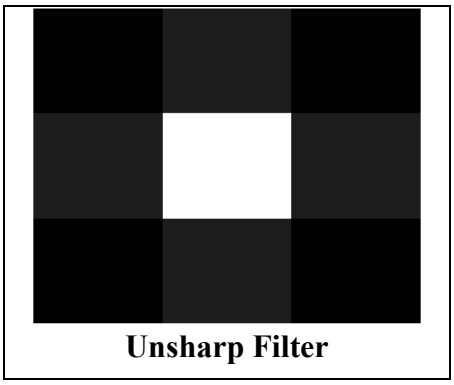


Motion Blur Filter

2.) Deblurring / Sharpening an Image

We can create an Unsharp Filter with sharpening parameter α where $0 \leq \alpha \leq 1$ by:

```
w = fspecial ('unsharp', α);
```



Unsharp Filter

Load the built-in "moon" image.

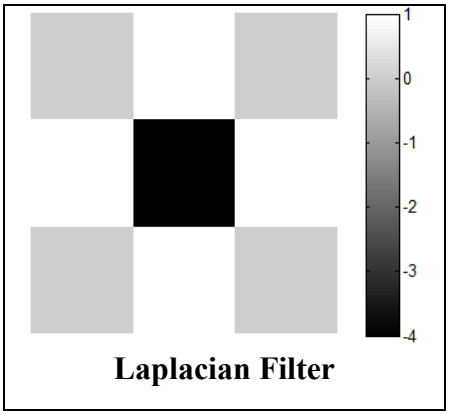
```
A = imread('moon.tif');
```

a.) Apply an unsharp filter to the moon image with parameter $\alpha = 0.9$. Compare this to the original image. What did the unsharp filter do to the image?

It deblurs the image and sharpens the edges

An alternative way to enhance an image is to subtract a Laplacian filter from the original image. We can generate a Laplacian filter with the command:

```
w = fspecial ('laplacian', 0);
```



Laplacian Filter

Since this filter has negative weights, the outputs may be positive or negative values. We have to be careful to not apply this to an 8-bit image since the resulting image will not necessarily be in the range [0,255]. So we have to first convert our image to a floating point format before we apply the filter.

```
A = double(A);
```

b.) Apply a Laplacian filter to the double-formatted moon image. Since the result has positive and negative values, be sure to use `imagesc` to display the image rather than `imshow`. What does the resulting image detect?

It detects edges in the image

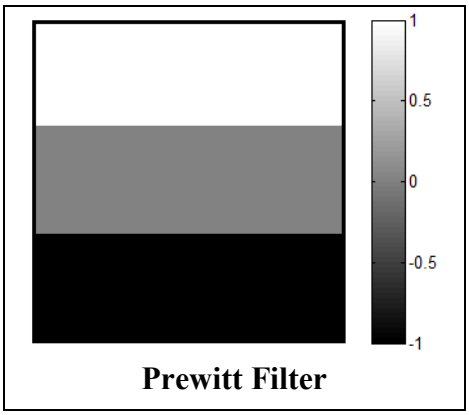
c.) Now subtract the Laplacian filtered image from the original image. What happens to the image? How does this compare to unsharp filtering?

It sharpens the image and removes blur

3.) Detecting Edges

We saw in the previous problem that the Laplacian filter is useful for detecting edges. The Prewitt filter is another popular edge detector. We can create the filter with:

```
w = fspecial ('prewitt');
```



Prewitt Filter

Load the built-in "Barbara" image and convert to double format.

```
load woman;
A = double(X);
```

a.) Apply the Prewitt filter to the Barbara image.

```
H = imfilter (A, w);
```

Be sure to use `imagesc` to display the image. What type of edges does the Prewitt filter detect? `It detects horizontal edges`

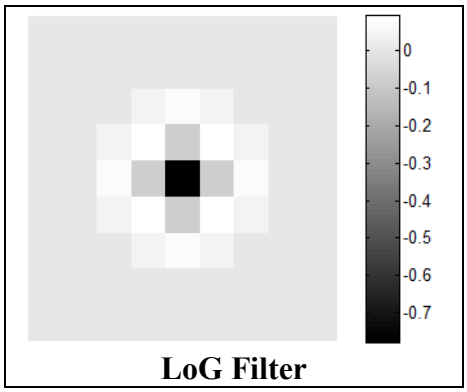
b.) We can rotate the Prewitt filter by taking the transpose. Try applying the transpose of w by typing:

```
V = imfilter (A, w');
```

What type of edges does this filter detect?

`The transpose of the Prewitt filter, detects the verical edges`

Another popular edge detector is the Laplacian of Gaussian (LoG) filter. This filter combines the detection properties of the Laplacian filter with the smoothing properties of a Gaussian filter. This means the LoG edge detector will be less sensitive to noise and texture.



LoG Filter

We can create a 9x9 LoG filter with standard deviation $\alpha = 0.8$ by:

```
w = fspecial ('log', [9,9], 0.8);
```

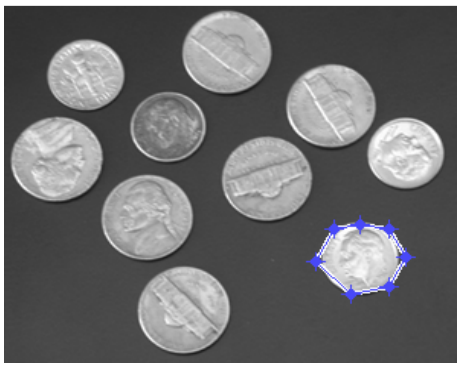
c.) Apply this LoG filter to the original image A. What does this filter detect?

`It detects the edges of the image.`

4.) Region-of-Interest Filtering

Sometimes we want to apply a linear filter to just part of the image. Load and display the built-in "coins" image.

```
A = imread('coins.png');
imshow(A)
```



Next we are going to manually segment the dime in the lower right corner of the image. Type the command:

```
D = roipoly;
```

This will allow you to click on the image, forming a "region-of-interest" polygon with vertices you clicked on. Make a full loop so that the last vertex you select was the first vertex you created. When you are finished drawing your polygon, right-click on the image and click on create mask.

You have just created a binary image mask D with 1's inside the region you selected.

```
imshow(D)
```

The Matlab command `roifilt2` will apply a linear filter to an image just like `imfilter`, but only in the region where the specified mask equals 1. For example, to blur the dime we could apply a 7x7 mean (average) filter.

```
w = fspecial('average', [7,7]);
B = roifilt2(w, A, D);
imshow(B)
```

Mean Filter

Display images A and B side-by-side to compare the results.

```
subplot(121);  imshow(A);  subplot(122);  imshow(B)
```

Now try sharpening the dime in the lower left corner by applying an Unsharp Filter to the dime in image A.