Table of Contents

Read Data

```
clear;
camera = imread('lab2/images_lab2/cameraman.png');
wagon = imread('lab2/images_lab2/wagon.png');
```

Q1 - Different Kernels

We will apply three different kernels in the spatial domain with one sharpening, one smoothing and apply them in different sizes.

Firstly we begin with showing the original image that we will use.

```
figure;
imshow(camera);
```

Now, let's introduce a mean filter of size 3×3 and apply the convolution using **imfilter**. The mean filtered image is shown below.

```
h1 = fspecial('average', 3);
meancamera3 = imfilter(camera, h1);

figure;
imshow(meancamera3);
```

And using mean filter of sie 7×7 we get

```
h2 = fspecial('average', 7);
meancamera7 = imfilter(camera, h2);

figure;
imshow(meancamera7);
```

And lastly, a 31×31 mean filter.

```
h3 = fspecial('average', 31);
meancamera31 = imfilter(camera, h3);

figure;
imshow(meancamera31);
```

Lets introduce gaussian filters and perform the same calculations as above. Since the gaussian filter is based on the gaussian distribution We also have an additional parameter in addition to the size of the kernel, namely σ. We have assumed this to be $\sigma = 3$ for this exercise. We use the function imgaussfilt due to the documentation recommending to use that one instead of imfilter.

```
gausscamera3 = imgaussfilt(camera, 3, FilterSize=3);

figure;
imshow(gausscamera3);
```

And using a gaussian filter size of 7×7 we get

```
gausscamera7 = imgaussfilt(camera, 3, FilterSize=7);

figure;
imshow(gausscamera7);
```

And lastly using a gaussian filter size of 31×31 we get

```
gausscamera31 = imgaussfilt(camera, 3, FilterSize=31);

figure;
imshow(gausscamera31);
```

For a sharpening (high pass) filter, we will use a unsharp masking of the mean filter for different sizes. Below we do this for a 3×3 unsharping mean filter.

```
size = 3;
h4 = fspecial('average', size);
h4 = h4 * -1;
h4(size - floor(size/2), size - floor(size/2)) = h4(size -
 floor(size/2), size - floor(size/2)) + 1;

msharpcamera3 = imfilter(camera, h4);

figure;
imshow(msharpcamera3);
```

And once again for the mask of size 7×7

```
size = 7;
h5 = fspecial('average', size);
h5 = h5 * -1;
h5(size - floor(size/2), size - floor(size/2)) = h5(size -
 floor(size/2), size - floor(size/2)) + 1;

msharpcamera7 = imfilter(camera, h5);

figure;
imshow(msharpcamera7);
```

And lastly, one last time for the mask of size $31 \times 31£

```
size = 31;
h6 = fspecial('average', size);
h6 = h6 * -1;
h6(size - floor(size/2), size - floor(size/2)) = h6(size -
 floor(size/2), size - floor(size/2)) + 1;

msharpcamera31 = imfilter(camera, h6);

figure;
imshow(msharpcamera31);
```

Q2 - Low Pass Filters

In fspecial we have the filters **average**, **disk** and **gaussian**. All of which are instances of low-pass filters. They calculate new pixel values by incorporating neighbouring pixel values and "average" the rapid changes. The **average** filter does this by calculating the sample mean of the defined neighbourhood, disk uses thhe same notion but uses a circular notion of distance from the center and the gaussian uses a gaussian distribution where pixels far away affect the new pixel value less.

Q3 - Construct new filters from simple arithmetic.

A low pass filter can be denoted $lp(x, y)$. We know from the textbook that a high-pass image $hp(x, y) = \delta(x, y) - lp(x, y)$ i.e subtracting a low pass from the original image.

```
imshow(camera - gausscamera31);
```

A bandreject filter is a original image minus a low pass and a high pass filter. But since we have defined the highpass filter in terms of a low pass, we can also define bandreject from only low pass filters.

```
imshow(camera - (gausscamera31 + (camera - meancamera31)));
```

Q4 - The Sobel Filter

Lets apply the filter on the cameraman image. Since this is a directional kernel, some arithmetic must be done on the two images.

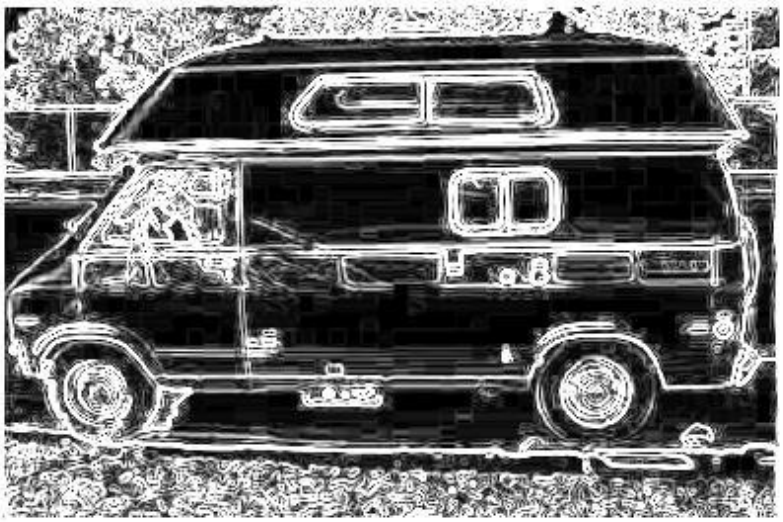
```
h7 = fspecial('sobel');
camerasobelx = imfilter(double(camera), h7);
camerasobely = imfilter(double(camera), h7');

figure;
imshow(uint8(sqrt(camerasobelx.^2 + camerasobely.^2)));
```

We also do the same of the wagon image

```
wagonsobelx = imfilter(double(wagon), h7);
wagonsobely = imfilter(double(wagon), h7');

figure;
imshow(uint8(sqrt(wagonsobelx.^2 + wagonsobely.^2)));
```



Published with MATLAB® R2021a