Semester I, 2019 – 2020 September 18, 2019

CS517: Digital Image Analysis

Lab 3: Spatial Filtering

Aim:

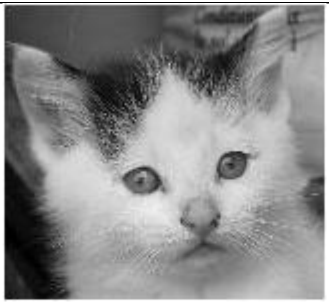
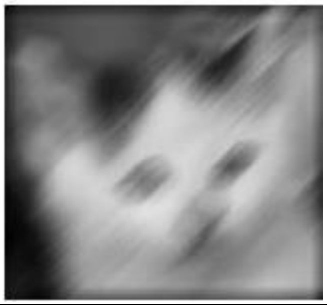
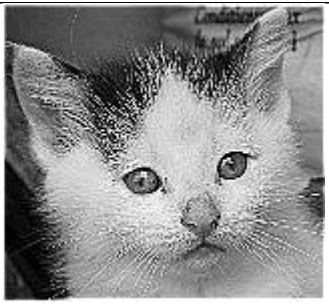
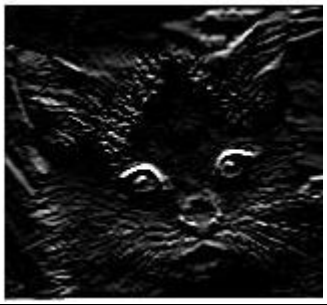
- To help understand the use of filters for noise reduction and image enhancement

Let's get started!

- Create a directory structure to hold your work for this course and all the subsequent labs:
    - Suggestion: `CS517/Lab3`

Spatial Filtering

- Spatial filtering is a technique that you can use to smooth, blur, sharpen, or find the edges of an image. The following four images are meant to demonstrate what spatial filtering can do. The original image is shown in the upper left-hand corner.

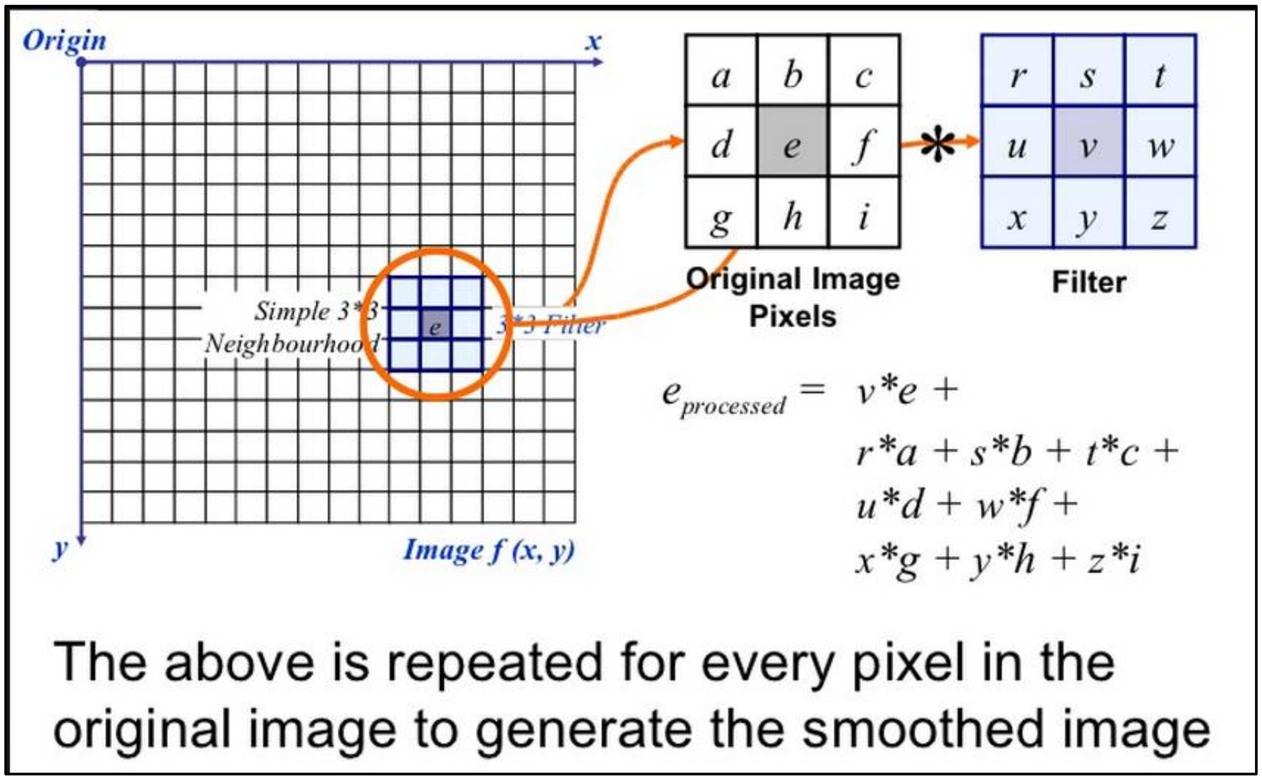
Original	Blurred
	
Shaprened	Edge Detection
	

Basic Idea

- There are two main types of filtering applied to images:
    1. Spatial domain filtering
    2. Frequency domain filtering

- In the next lab, we will talk about frequency domain filtering, which makes use of the Fourier Transform. For spatial domain filtering, we are performing filtering operations directly on the pixels of an image.

- Spatial Filtering is sometimes also known as neighborhood processing. Neighborhood processing is an appropriate name because you define a center point and perform an operation (or apply a filter) to only those pixels in predetermined neighborhood of that center point. The result of the operation is one value, which becomes the value at the center point's location in the modified image. Each point in the image is processed with its neighbors. The general idea is shown below as a "sliding filter" that moves throughout the image to calculate the value at the center location. *See lecture slides for more details!*


Origin
x
y
Simple 3*3 Neighbourhood
e
3*3 Filter
Image f (x, y)
a b c
d e f
g h i
*
r s t
u v w
x y z
Original Image Pixels
Filter
$e_{processed} = v*e + r*a + s*b + t*c + u*d + w*f + x*g + y*h + z*i$
The above is repeated for every pixel in the original image to generate the smoothed image

Using `imfilter`

You can use a function that comes as part of the Image Processing Toolkit, `imfilter`, to perform spatial filtering.

```
results = imfilter(f, w);
```

The following table from *Digital Image Processing, Using MATLAB*, by Rafael C. Gonzalez, Richard E. Woods, and Steven L. Eddins, summarizes the additional options available with `imfilter`.

Options	Description
Filtering mode	
'corr'	Filtering is done using correlation. This is the default.
'conv'	Filtering is done using convolution.
Boundary Options	
P	The boundaries of the input image are extended by padding with a value, P (written without quotes). This is the default, with value 0.
'replicate'	The size of the image is extended by replicating the values in its outer border.
'symmetric'	The size of the image is extended by mirror-reflecting it across its border.
'circular'	The size of the image is extended by treating the image as one period a 2-D periodic function.
Size Options	
'full'	The output is of the same size as the extended (padded) image.
'same'	The output is of the same size as the output. This is achieved by limiting the excursions of the center of the filter mask to points contained in the original image. This is the default.

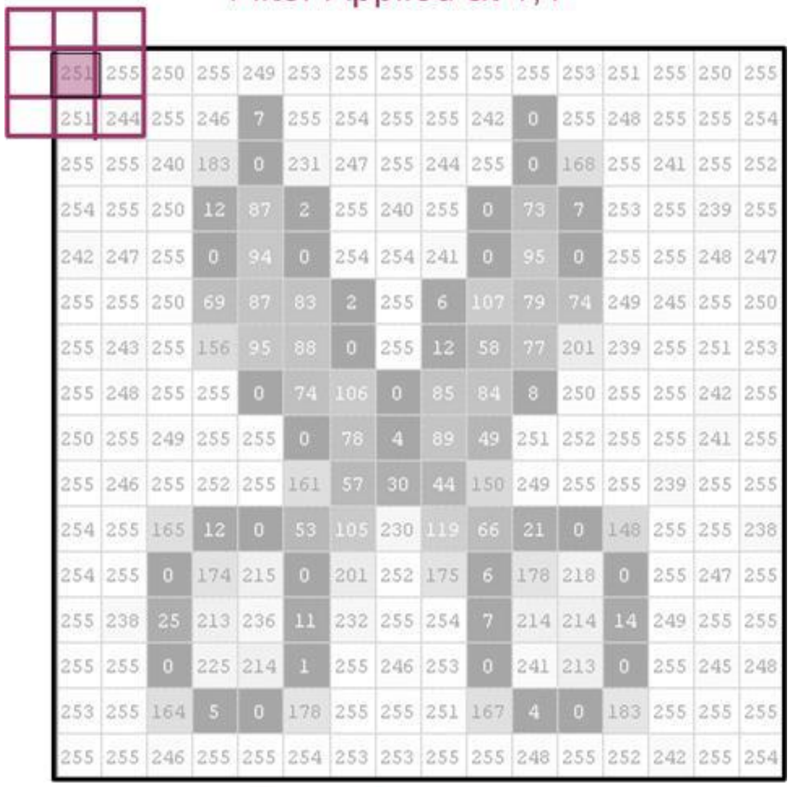
The following subsections discuss the **`imfilter`** options.

`imfilter`--Boundary Option

1. There is an inherent problem when you are working with the corners and edges in Spatial Filtering. The problem is that some of the "neighbors" are missing. Consider location (1,1): See image below!

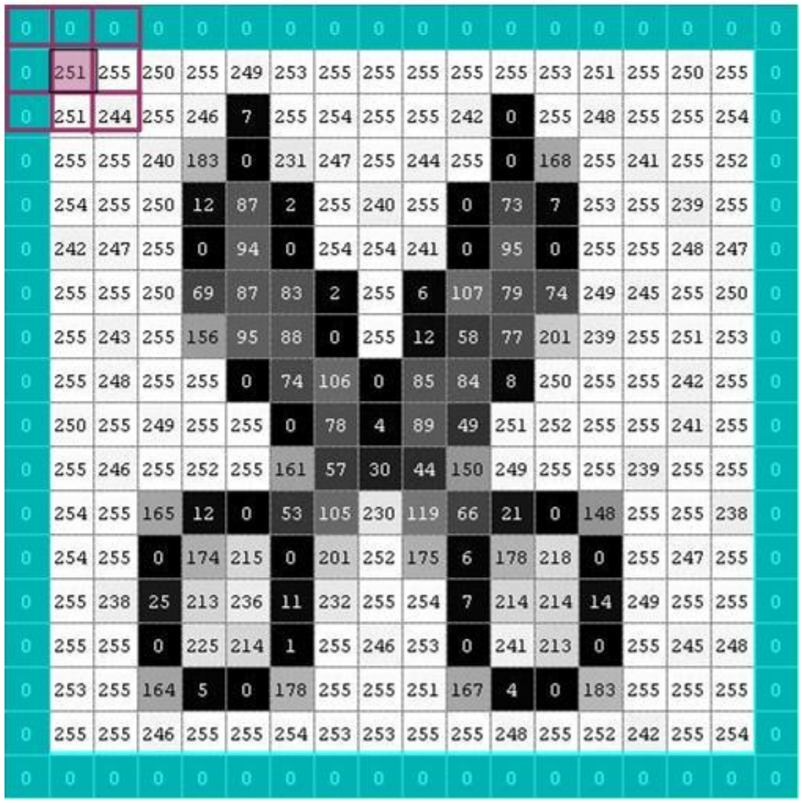
2. In this case, there are no upper neighbors or neighbors to the left. Two solutions, zero padding and replicating, are shown below. The pixels highlighted in blue have been added to the original image:

Filter Applied at 1,1



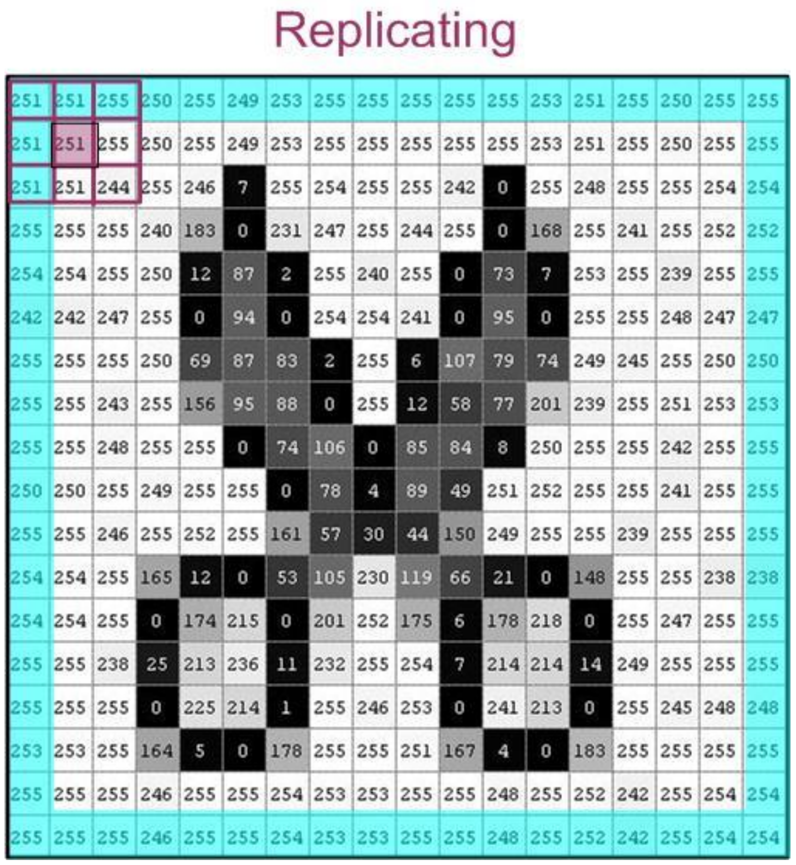
251	255	250	255	249	253	255	255	255	255	255	253	251	255	250	255
251	244	255	246	7	255	254	255	255	242	0	255	248	255	255	254
255	255	240	183	0	231	247	255	244	255	0	168	255	241	255	252
254	255	250	12	87	2	255	240	255	0	73	7	253	255	239	255
242	247	255	0	94	0	254	254	241	0	95	0	255	255	248	247
255	255	250	69	87	83	2	255	6	107	79	74	249	245	255	250
255	243	255	156	95	88	0	255	12	58	77	201	239	255	251	253
255	248	255	255	0	74	106	0	85	84	8	250	255	255	242	255
250	255	249	255	255	0	78	4	89	49	251	252	255	255	241	255
255	246	255	252	255	161	57	30	44	150	249	255	255	239	255	255
254	255	165	12	0	53	105	230	119	66	21	0	148	255	255	238
254	255	0	174	215	0	201	252	175	6	178	218	0	255	247	255
255	238	25	213	236	11	232	255	254	7	214	214	14	249	255	255
255	255	0	225	214	1	255	246	253	0	241	213	0	255	245	248
253	255	164	5	0	178	255	255	251	167	4	0	183	255	255	255
255	255	246	255	255	254	253	253	255	255	248	255	252	242	255	254

- Zero Padding



0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	251	255	250	255	249	253	255	255	255	255	255	253	251	255	250	255	0
0	251	244	255	246	7	255	254	255	255	242	0	255	248	255	255	254	0
0	255	255	240	183	0	231	247	255	244	255	0	168	255	241	255	252	0
0	254	255	250	12	87	2	255	240	255	0	73	7	253	255	239	255	0
0	242	247	255	0	94	0	254	254	241	0	95	0	255	255	248	247	0
0	255	255	250	69	87	83	2	255	6	107	79	74	249	245	255	250	0
0	255	243	255	156	95	88	0	255	12	58	77	201	239	255	251	253	0
0	255	248	255	255	0	74	106	0	85	84	8	250	255	255	242	255	0
0	250	255	249	255	255	0	78	4	89	49	251	252	255	255	241	255	0
0	255	246	255	252	255	161	57	30	44	150	249	255	255	239	255	255	0
0	254	255	165	12	0	53	105	230	119	66	21	0	148	255	255	238	0
0	254	255	0	174	215	0	201	252	175	6	178	218	0	255	247	255	0
0	255	238	25	213	236	11	232	255	254	7	214	214	14	249	255	255	0
0	255	255	0	225	214	1	255	246	253	0	241	213	0	255	245	248	0
0	253	255	164	5	0	178	255	255	251	167	4	0	183	255	255	255	0
0	255	255	246	255	255	254	253	253	255	255	248	255	252	242	255	254	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

3. Zero padding is the default. You can also specify a value other than zero to use as a padding value.
4. Another solution is replicating the pixel values along the edges:



Replicating

251	251	255	250	255	249	253	255	255	255	255	255	253	251	255	250	255	255
251	251	255	250	255	249	253	255	255	255	255	255	253	251	255	250	255	255
251	251	244	255	246	7	255	254	255	255	242	0	255	248	255	255	254	254
255	255	255	240	183	0	231	247	255	244	255	0	168	255	241	255	252	252
254	254	255	250	12	87	2	255	240	255	0	73	7	253	255	239	255	255
242	242	247	255	0	94	0	254	254	241	0	95	0	255	255	248	247	247
255	255	255	250	69	87	83	2	255	6	107	79	74	249	245	255	250	250
255	255	243	255	156	95	88	0	255	12	58	77	201	239	255	251	253	253
255	255	248	255	255	0	74	106	0	85	84	8	250	255	255	242	255	255
250	250	255	249	255	255	0	78	4	89	49	251	252	255	255	241	255	255
255	255	246	255	252	255	161	57	30	44	150	249	255	255	239	255	255	255
254	254	255	165	12	0	53	105	230	119	66	21	0	148	255	255	238	238
254	254	255	0	174	215	0	201	252	175	6	178	218	0	255	247	255	255
255	255	238	25	213	236	11	232	255	254	7	214	214	14	249	255	255	255
255	255	255	0	225	214	1	255	246	253	0	241	213	0	255	245	248	248
253	253	255	164	5	0	178	255	255	251	167	4	0	183	255	255	255	255
255	255	255	246	255	255	254	253	253	255	255	248	255	252	242	255	254	254
255	255	255	246	255	255	254	253	253	255	255	248	255	252	242	255	254	254

5. As a note, if your filter were larger than 3x3, then the "border padding" would have to be extended. For a filter of size 3x3, 'replicate' and 'symmetric' yield the same results.

6. The following images show the results of the four different boundary options. The filter used below is a 3x3 averaging filter that was created with the following syntax:

```
h =[1/9 1/9 1/9;
    1/9 1/9 1/9;
    1/9 1/9 1/9]
```

`results1=imfilter(cat,h);`
`figure,imshow(results1)`
`title('Zero-Padded')`
	
	`results2=imfilter(cat,h,'replicate');`
	`figure,imshow(results2)`
	`title('Replicate')`

`results3=imfilter(cat,h,'symmetric');`
`figure,imshow(results3)`
`title('Symmetric')`
	
	`results4=imfilter(cat,h,'circular');`
	`figure,imshow(results4)`
	`title('Circular')`

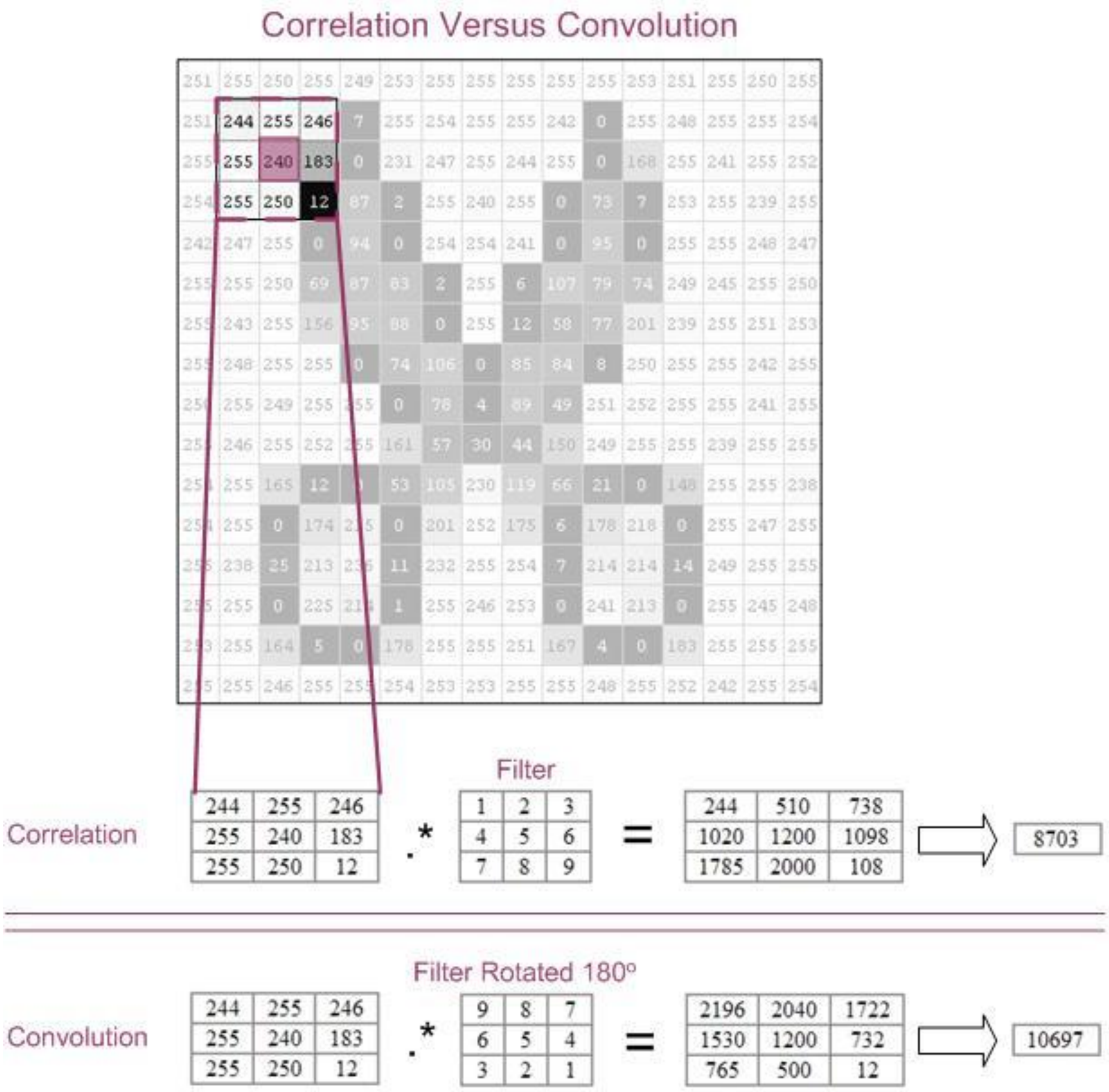
7. The disadvantage of zero padding is that it leaves dark artifacts around the edges of the filtered image (with white background). You can see this as a dark border along the bottom and right-hand edge in the zero-padded image above.

`imfilter` - Filtering Mode (Correlation versus Convolution)

With **`imfilter`** , you can choose one of two filtering modes: *correlation* or *convolution*. The difference between the two is that convolution rotates the filter by 180^o before performing multiplication. The following diagram is meant to demonstrate the two operations for position (3, 3) of the image:

The following MATLAB code demonstrates correlation and convolution:

```
h = [1 2 3; 4 5 6; 7 8 9];

h = h/45;

result_corr = imfilter(cat,h);   %correlation is the default, you can

                                 %also send 'corr' as an argument

result_conv = imfilter(cat,h,'conv');
```



imfilter - Size Options

There are two size options 'full' and 'same'. The 'full' will be as large as the padded image, whereas 'same' will be the same size as the input image.

To create a 'full' and 'same' image, you can use the following MATLAB syntax:

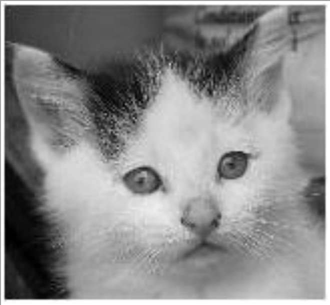
```
h =[0.1111 0.1111 0.1111; 0.1111 0.1111 0.1111; 0.1111 0.1111 0.1111];

cat_same = imfilter(cat,h);  %'same' is the default, but you can also

                             % include it as an argument

cat_full = imfilter(cat,h,'full');
```

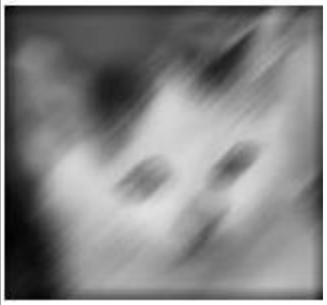
Creating Special Filters

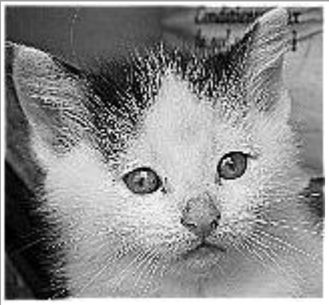
You can define the filters for spatial filtering manually or you can call a function that will create the filter matrices. The function, called `fspecial`, requires an argument that specifies the kind of filter you would like. A full description of `fspecial` is available in MATLAB help--type: `doc fspecial`

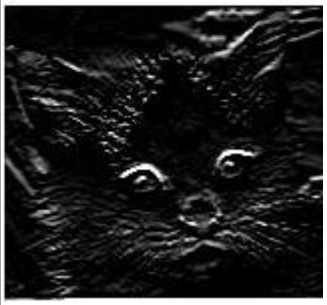
The following table shows you three filters, created by `fspecial`, and the results on an image of a cat:

MATLAB Code	Resulting Image

`%original picture`
`cat=imread('cat.jpg');`
`figure, imshow(cat)`
	

`%motion blur`
`h=fspecial('motion', 20, 45);`
`cat_motion=imfilter(cat,h);`
`figure, imshow(cat_motion)`
	

`%sharpening`
`h=fspecial('unsharp');`
`cat_sharp=imfilter(cat,h);`
`figure, imshow(cat_sharp)`
	

`%horizontal edge-detection`
`h=fspecial('sobel');`
`cat_sobel=imfilter(cat,h);`
`figure, imshow(cat_sobel)`
	

Exercise1: Large Object Detection

Experiment with different sizes of `'average'` filter using `fspecial` on standard.jpeg image
Further apply thresholding on the resulting image using `im2bw` function. Try to achieve the following:

1. Find the size of the filter with which all the four patches with pepper noise are eliminated when thresholding is applied on smoothed image
2. Find the size of the filter with which all the vertical lines are eliminated when thresholding is applied on smoothed image

Show the original image, smoothed image and thresholded image labeled with the filter kernel size for each of the scenarios above. Your code should be in `myObjDetection.m`

Exercise2: Unsharp Masking

Write your own m-function for unsharp (`myUnsharp.m`) masking of a given image to produce a new output image (see lecture slides). Your function should apply the following steps:

1. Apply smoothing to produce a blurred version of the original image,
2. Subtract the blurred image from the original image to produce an edge image,
3. Add the edge image to the original image to produce a sharpened image.

Apply your unsharp filter to the image clown.tif and show the original image and the sharpened image.

Exercise3: Edge Filters

Try out and compare the following edge filters (`myEdgeFilters.m`) on the image clown.tif. The filters are described in the lecture slides.

1. Sobel edge detector: size 3×3, vertical Gx and horizontal Gy. Compute the approximate magnitude |G| of the filter responses: |G| = |Gx| + |Gy|
2. Laplacian for edge detection: size 3 × 3.

Show the original image and all result images (labeled with the used filter kernel and filter kernel size). Compare the results and give a qualitative comment.

Submitting your work:

- All source files and class files as one tar-gzipped archive.
  - When unzipped, it should create a directory with your ID. Example: **P2008CS1001–L1** (NO OTHER FORMAT IS ACCEPTABLE!!! Case sensitive!!!)
  - ***Negative marks if the TA has to manually change this to run his/her scripts!!***
- Source / class files should include the following: (Case-Sensitive file names!!)
  - `myObjDetection.m`
  - `myUnsharp.m`
  - `myEdgeFilters.m`
- ***Negative marks for any problems/errors in running your programs***
- Submit/Upload it to Moodle