**Introduction to Image Processing**

**Lab 5**

## 1. Spatial Filtering

There are two main types of filtering applied to images:

* spatial domain filtering
* frequency domain filtering

For spatial domain filtering, we are performing filtering operations directly on the the pixels of an image.

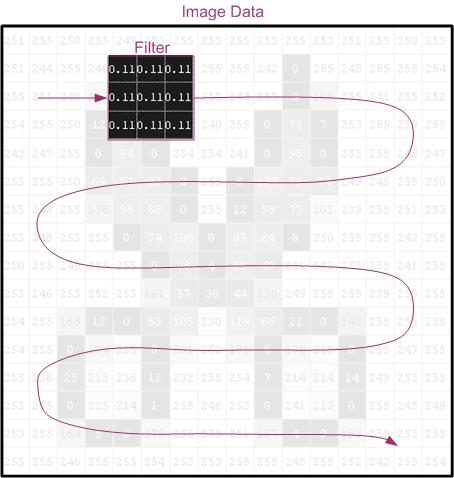
Spatial filtering is a technique that uses a pixel and its neighbors to select a new value for the pixel. The simplest type of spatial filtering is called linear filtering. It attaches a weight to the pixels in the neighborhood of the pixel of interest, and these weights are used to blend those pixels together to provide a new value for the pixel of interest. Linear filtering can be uses 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 Cat | Motion Blurred Cat |
| Sharpened Cat | Naive Sobel Edge Detected cat |

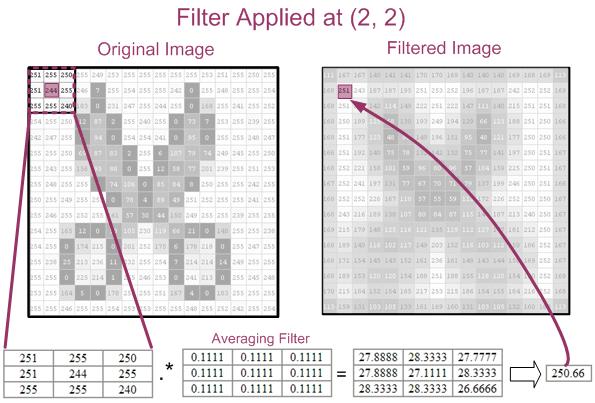
Sometimes a linear filter is not enough to solve a particular problem. In that case it is possible to use higher order math or full-blown MATLAB functions produce specialized results. Such non-linear filters are useful for smoothing only smooth areas, enhancing only strong edges or removing speckles from images.

**2.1 Basic Idea**

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.



The following diagram is meant to illustrate in further details how the filter is applied. The filter (an averaging filter) is applied to location 2,2.



Notice how the resulting value is placed at location 2,2 in the filtered image.

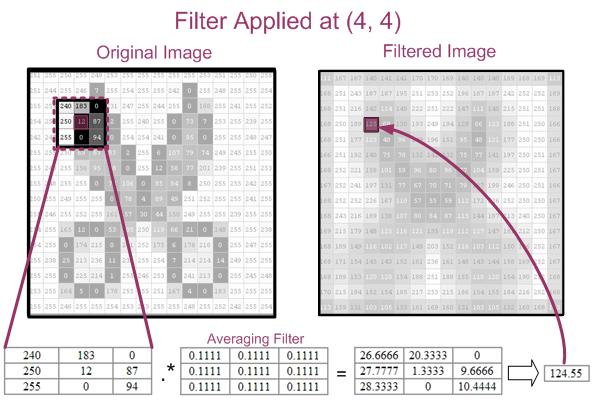
The breakdown of how the resulting value of 251 (rounded up from 250.66) was calculated mathematically is:

= 251\*0.1111 + 255\*0.1111 + 250\*0.1111 + 251\*0.1111 + 244\*0.1111 + 255\*0.1111 + 255\*0.1111 + 255\*0.1111 + 240\*0.1111

= 27.88888 + 28.33333 + 27.77777 + 27.88888 + 27.11111 + 28.33333 + 28.33333 + 28.33333 + 26.66666

= 250.66

 The following illustrates the averaging filter applied to location 4,4.



Once again, the mathematical breakdown of how 125 (rounded up from 124.55) was calculated is below:

= 240\*0.1111 + 183\*0.1111 + 0\*0.1111 + 250\*0.1111 + 12\*0.1111 + 87\*0.1111 + 255\*0.1111 + 0\*0.1111 + 94\*0.1111

= 124.55

 The following MATLAB function demonstrates how spatial filtering may be applied to an image:

function img = myfilter(f, w)

%MYFILTER Performs spatial correlation

% I=MYFILTER(f, w) produces an image that has undergone correlation.

% f is the original image

% w is the filter (assumed to be 3x3)

% The original image is padded with 0's

%Author: Nova Scheidt

% check that w is 3x3

[m,n]=size(w);

if m~=3 | n~=3

error('Filter must be 3x3')

end

%get size of f

[x,y]=size(f);

%create padded f (called g)

%first, fill with zeros

g=zeros(x+2,y+2);

%then, store f within g

for i=1:x

for j=1:y

g(i+1,j+1)=f(i,j);

end

end

%cycle through the array and apply the filter

for i=1:x

for j=1:y

img(i,j)=g(i,j)\*w(1,1)+g(i+1,j)\*w(2,1)+g(i+2,j)\*w(3,1) ... %first column

+ g(i,j+1)\*w(1,2)+g(i+1,j+1)\*w(2,2)+g(i+2,j+1)\*w(3,2)... %second column

+ g(i,j+2)\*w(1,3)+g(i+1,j+2)\*w(2,3)+g(i+2,j+2)\*w(3,3);

end

end

%Convert to uint--otherwise there are double values and the expected

%range is [0, 1] when the image is displayed

img=uint8(img);

To apply the filter to the **stock\_cut.jpg** image, the following calls were made:

w=[1/9 1/9 1/9

1/9 1/9 1/9

1/9 1/9 1/9]

stock\_cut=imread('stock\_cut.jpg');

results=myfilter(stock\_cut,w);

imtool(results)

**2.2 Filtering with imfilter**

Instead of using the M-File from above, you can use a function that comes as part of the Image Processing Toolkit. You can call it in the same way that myfilter was called above:

results=imfilter(stock\_cut, w);

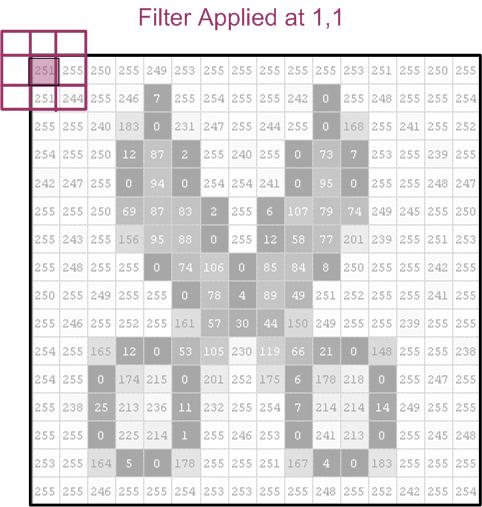
imfilter is more powerful than the simple myfilter. The following table, modified from page 94 of *Digital Image Processing, Using MATLAB*, by Rafael C. Gonzalez, Richard E. Woods, and Steven L. Eddins, summarizes the additional options available with imfilter.

|  |  |
| --- | --- |
| **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. |

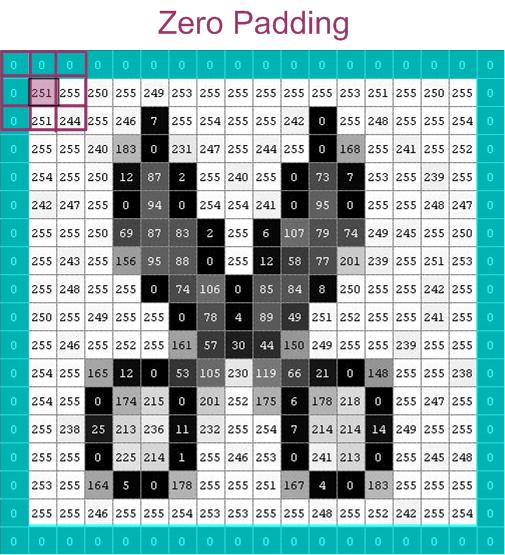
The following subsections discuss the imfilter options.

**2.2.1 Imfilter—Boundary Options**

The example above deliberately applied the filter at location 2,2. This is because there is an inherent problem when you are working with the corners and edges. The problem is that some of the "neighbors" are missing. Consider location 1,1:

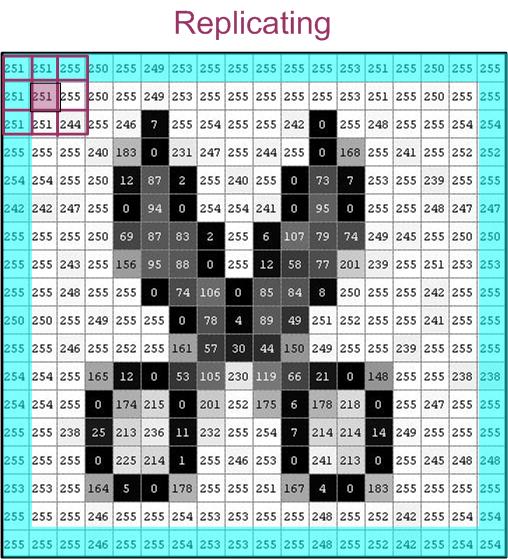


In this case, there are no upper neighbors or neighbors to the left. Two solutions, zero padding and replicating, are shown below. The pixels highlighted in blue have been added to the original image:



Zero padding is the default. You can also specify a value other than zero to use as a padding value.

Another solution is replicating the pixel values along the edges:



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.

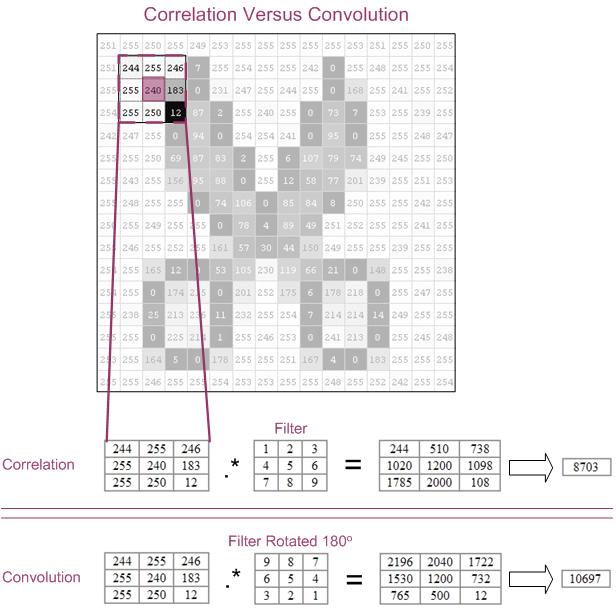
The following images show the results of the four different boundary options. The filter used below is a 5x5 averaging filter that was created with the following syntax:   
h=fspecial('average',5)

|  |  |
| --- | --- |
| http://www.cs.uregina.ca/Links/class-info/425/Lab3/Picts/cat/Boundary/cat_orig.jpg | |
| http://www.cs.uregina.ca/Links/class-info/425/Lab3/Picts/cat/Boundary/cat_zero.jpg  results1=imfilter(cat,h);  figure,imshow(results1)  title('Zero-Padded') | http://www.cs.uregina.ca/Links/class-info/425/Lab3/Picts/cat/Boundary/cat_replicate.jpg  results2=imfilter(cat,h,'replicate');  figure,imshow(results2)  title('Replicate') |
| http://www.cs.uregina.ca/Links/class-info/425/Lab3/Picts/cat/Boundary/cat_symmetric.jpg  results3=imfilter(cat,h,'symmetric');  figure,imshow(results3)  title('Symmetric') | http://www.cs.uregina.ca/Links/class-info/425/Lab3/Picts/cat/Boundary/cat_circular.jpg  results4=imfilter(cat,h,'circular');  figure,imshow(results4)  title('Circular') |

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.

**2.2.2 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 180o before performing multiplication. The following diagram is meant to demonstrate the two operations for position 3, 3 of the image:



This example is for demonstration purposes only. You will notice that the resulting values are not in the range of [0, 255]. To get better results, you can normalize the filter (in this case, divide by 45).

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');

**2.2.3 Imfilter—Size Options**

There are two size options 'full' and 'same'. The 'full' will be as large as the padded image, where as '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];

stock\_cut\_same=imfilter(stock\_cut,h); % 'same' is the default, but you can also

% include it as an argument

stock\_cut\_full=imfilter(stock\_cut,h,'full');

If you use imtool to view both of these images, you will note that the 'same' is 16x16, whereas 'full' is 18x18.

**2.3 Predefined Filters**

You can define the filters for spatial filtering manually or you can call a function that will create certain common filter matrices for you. 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 is meant to show 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) | http://www.cs.uregina.ca/Links/class-info/425/Lab3/Picts/cat/cat_orig.jpg |
| %motion blur  h=fspecial('motion', 20, 45);  cat\_motion=imfilter(cat,h);  figure, imshow(cat\_motion) | http://www.cs.uregina.ca/Links/class-info/425/Lab3/Picts/cat/cat_motion.jpg |
| %sharpening  %see section 7.6 (esp 7.6.2)  h=fspecial('unsharp');  cat\_sharp=imfilter(cat,h);  figure, imshow(cat\_sharp) | http://www.cs.uregina.ca/Links/class-info/425/Lab3/Picts/cat/cat_sharp.jpg |
| %horizontal edge-detection  %see section 7.2 and 7.3.1  h=fspecial('sobel');  cat\_sobel=imfilter(cat,h);  figure, imshow(cat\_sobel) | http://www.cs.uregina.ca/Links/class-info/425/Lab3/Picts/cat/cat_sobel.jpg |

## Exercise:

Download the **two\_cats.jpg** image and store it.  


1. Load the image data.
2. Use a spatial filter to find and display the horizontal edges of the image.
3. Use a spatial filter to find and display the vertical edges of the image   
   **hint:** read the MATLAB documentation on fspecial
4. Add the horizontal edge image to the vertical edge image to yield the following results:  
   
5. See if you can reproduce the following result, which is the edge magnitude map for for this image. The relevant instructions are in your textbook.  
   