Laboratory of Image Processing

Image Filtering

Pier Luigi Mazzeo

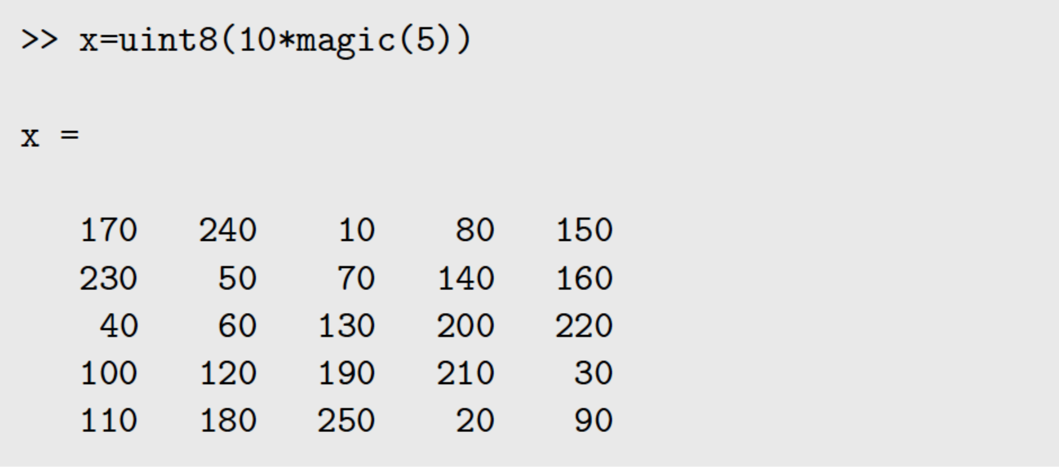
pierluigi.mazzeo@cnr.it

Institute of Applied Sciences and Intelligent Systems

ScienceApp

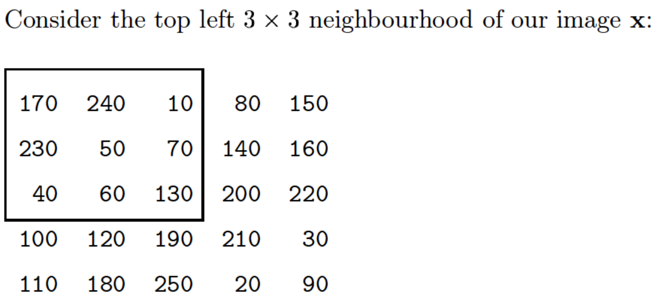
National Research Council of Italy

Filtering



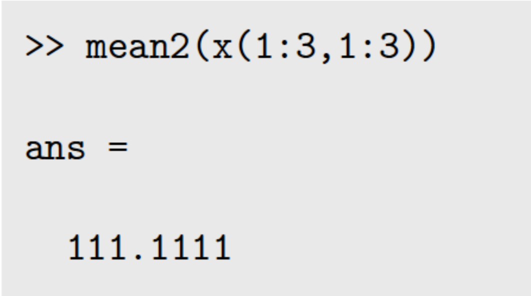
```
>> x=uint8(10*magic(5))

x =

   170   240    10    80   150
   230    50    70   140   160
    40    60   130   200   220
   100   120   190   210    30
   110   180   250    20    90
```

Neighbourhood processing

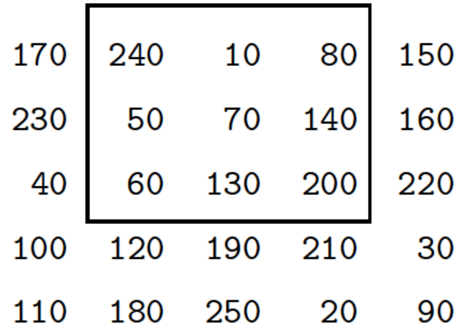


Consider the top left 3 × 3 neighbourhood of our image **x**:

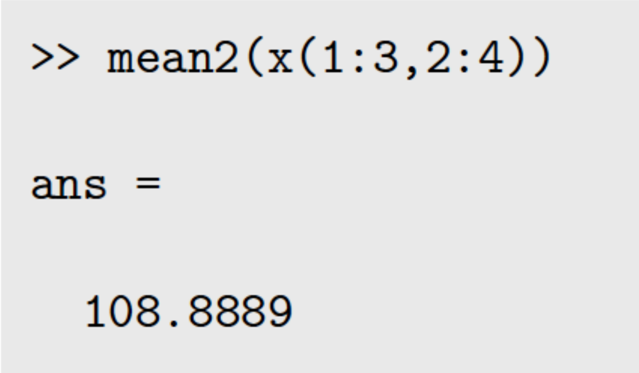
170	240	10	80	150
230	50	70	140	160
40	60	130	200	220
100	120	190	210	30
110	180	250	20	90



```
>> mean2(x(1:3,1:3))

ans =

  111.1111
```

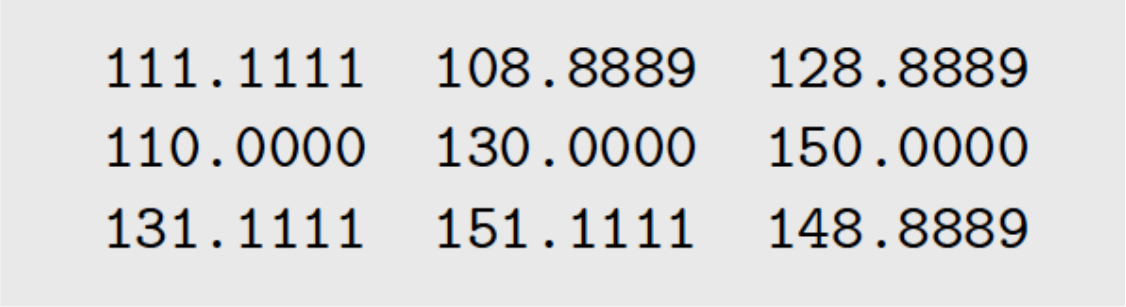


170	240	10	80	150
230	50	70	140	160
40	60	130	200	220
100	120	190	210	30
110	180	250	20	90



```
>> mean2(x(1:3,2:4))

ans =

  108.8889
```

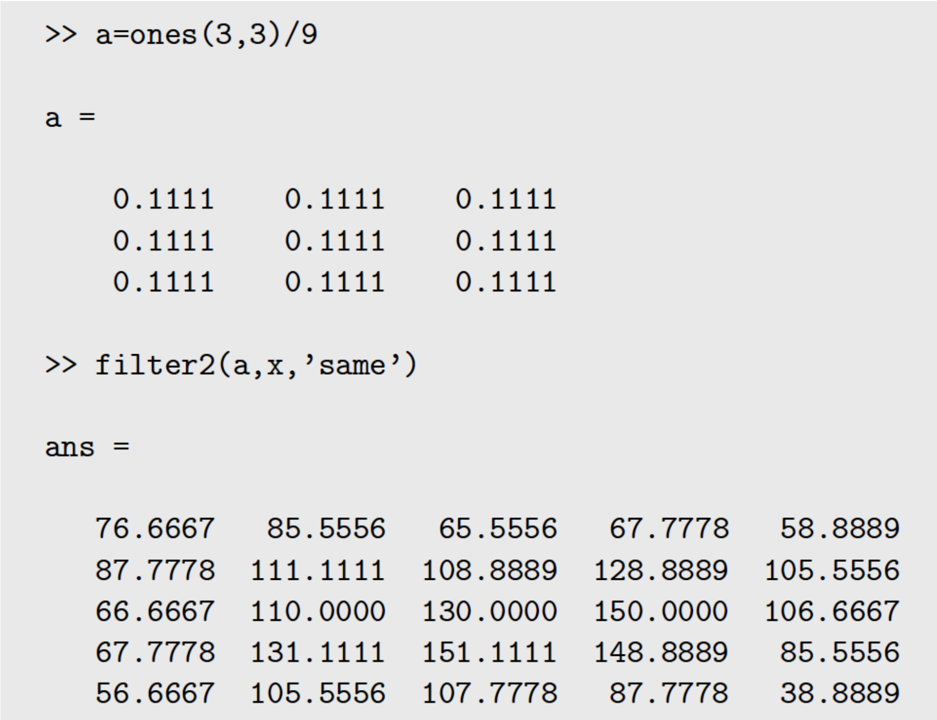
Mean



```
111.1111   108.8889   128.8889
110.0000   130.0000   150.0000
131.1111   151.1111   148.8889
```

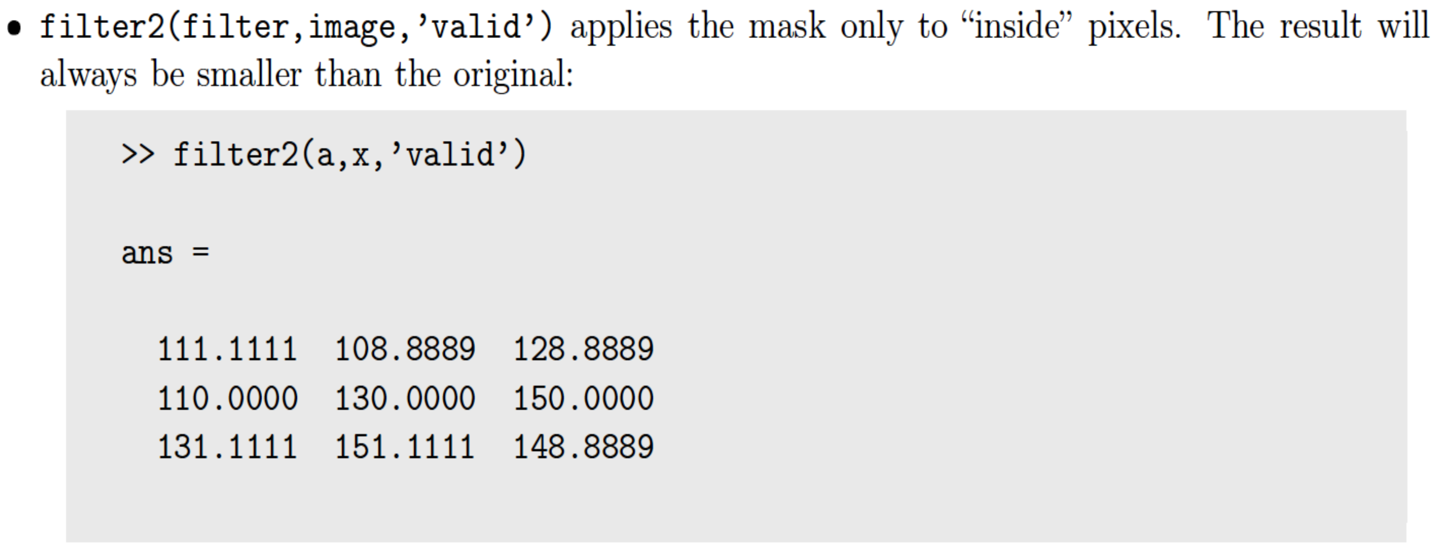
Filtering in MatLab

$$\begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix} = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$



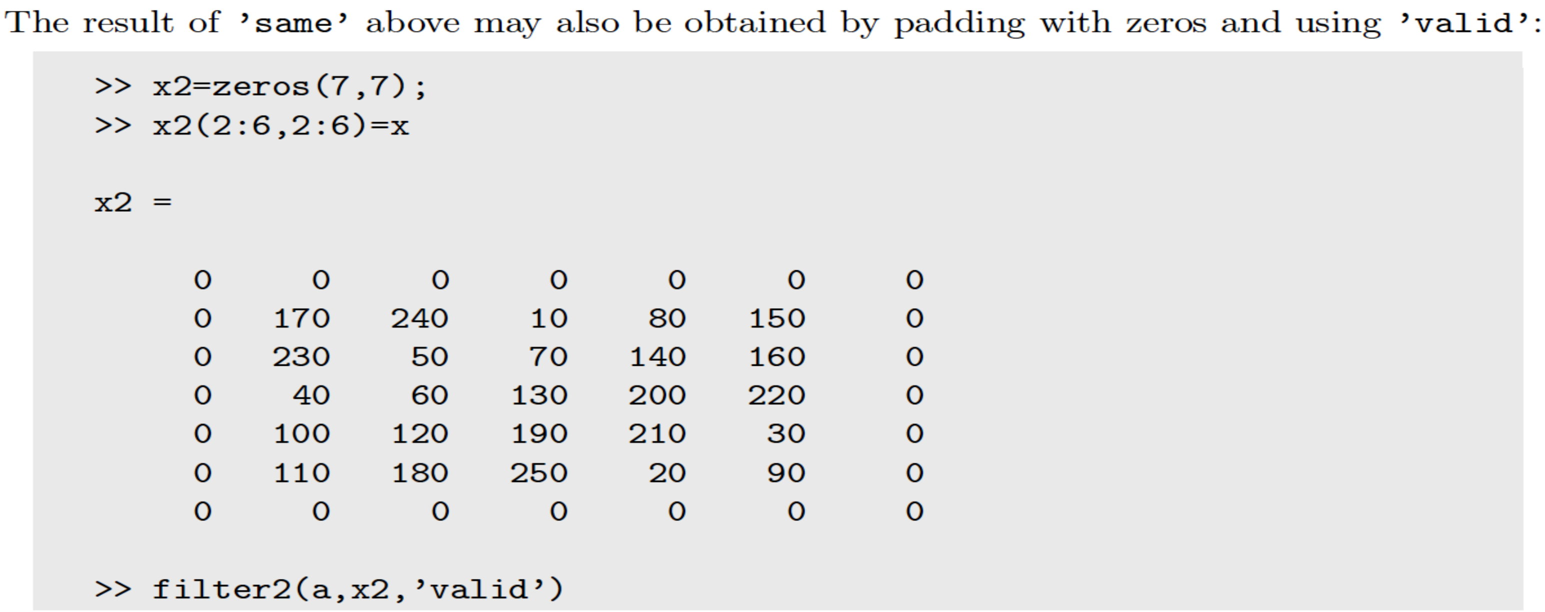
```
>> a=ones(3,3)/9

a =

    0.1111    0.1111    0.1111
    0.1111    0.1111    0.1111
    0.1111    0.1111    0.1111

>> filter2(a,x,'same')

ans =

   76.6667   85.5556   65.5556   67.7778   58.8889
   87.7778  111.1111  108.8889  128.8889  105.5556
   66.6667  110.0000  130.0000  150.0000  106.6667
   67.7778  131.1111  151.1111  148.8889   85.5556
   56.6667  105.5556  107.7778   87.7778   38.8889
```

filter2



- `filter2(filter,image,'valid')` applies the mask only to "inside" pixels. The result will always be smaller than the original:

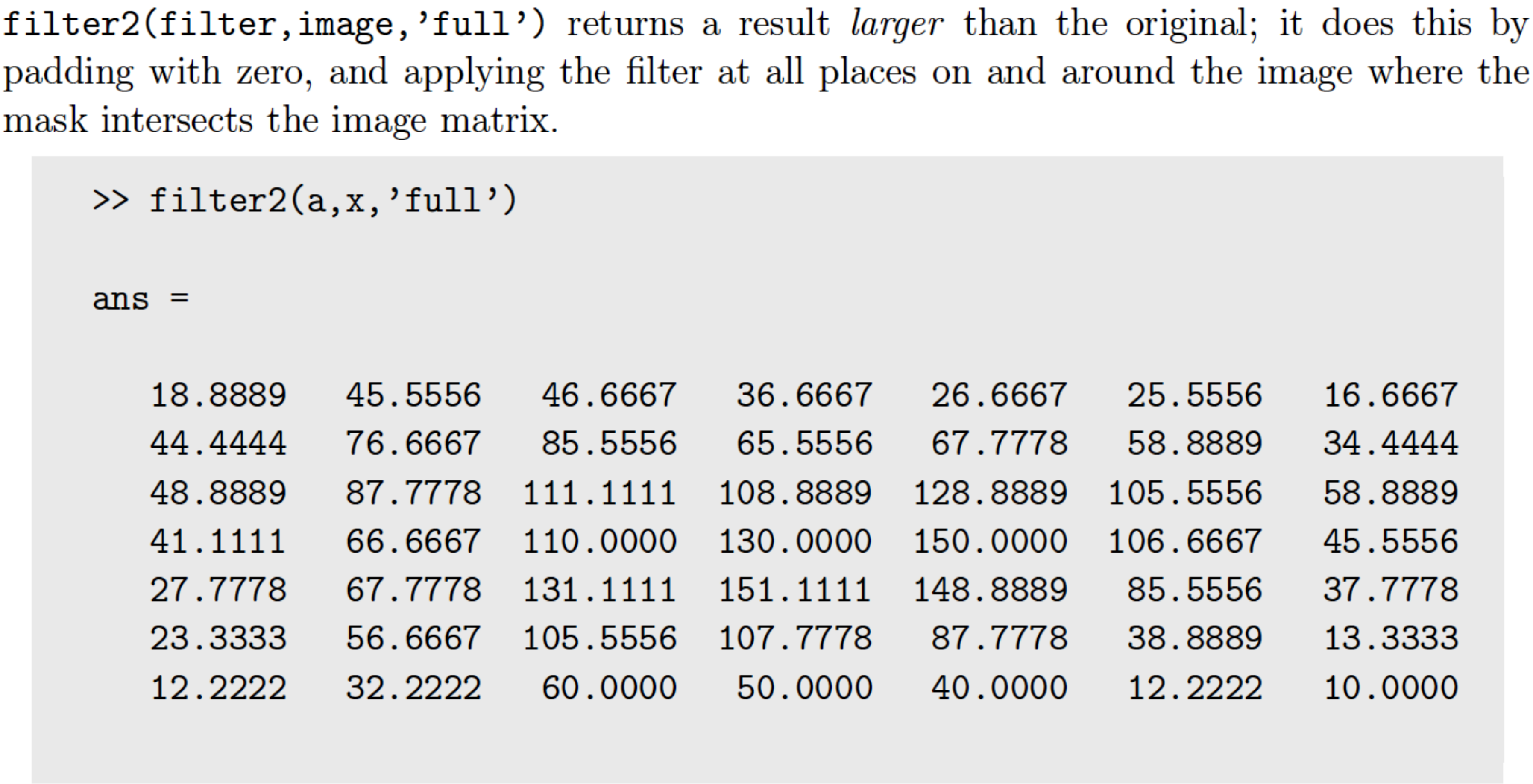
```
>> filter2(a,x,'valid')

ans =

  111.1111  108.8889  128.8889
  110.0000  130.0000  150.0000
  131.1111  151.1111  148.8889
```



The result of `'same'` above may also be obtained by padding with zeros and using `'valid'`:

```
>> x2=zeros(7,7);
>> x2(2:6,2:6)=x

x2 =

     0     0     0     0     0     0     0
     0   170   240    10    80   150     0
     0   230    50    70   140   160     0
     0    40    60   130   200   220     0
     0   100   120   190   210    30     0
     0   110   180   250    20    90     0
     0     0     0     0     0     0     0

>> filter2(a,x2,'valid')
```

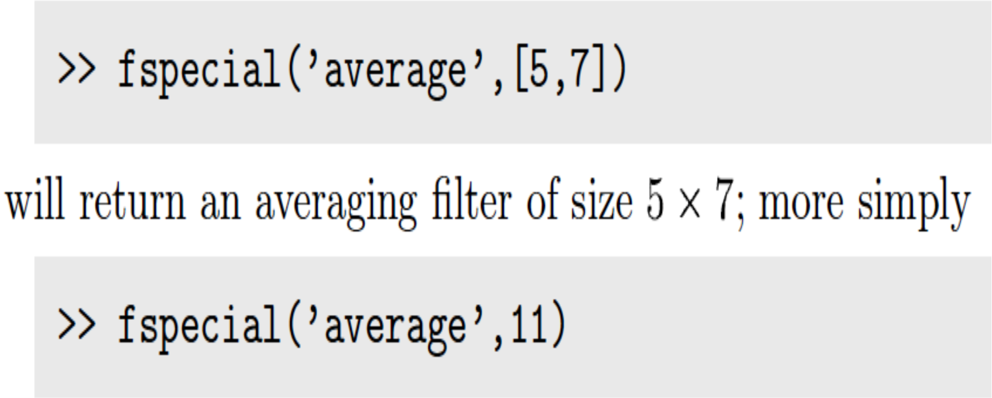
filter2 (cont.)



`filter2(filter,image,'full')` returns a result *larger* than the original; it does this by padding with zero, and applying the filter at all places on and around the image where the mask intersects the image matrix.

```
>> filter2(a,x,'full')

ans =

   18.8889   45.5556   46.6667   36.6667   26.6667   25.5556   16.6667
   44.4444   76.6667   85.5556   65.5556   67.7778   58.8889   34.4444
   48.8889   87.7778  111.1111  108.8889  128.8889  105.5556   58.8889
   41.1111   66.6667  110.0000  130.0000  150.0000  106.6667   45.5556
   27.7778   67.7778  131.1111  151.1111  148.8889   85.5556   37.7778
   23.3333   56.6667  105.5556  107.7778   87.7778   38.8889   13.3333
   12.2222   32.2222   60.0000   50.0000   40.0000   12.2222   10.0000
```

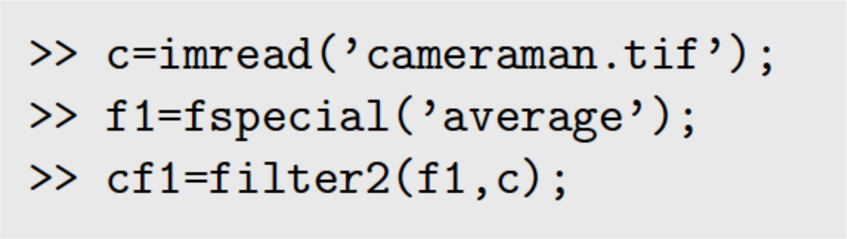
fspecial (low pass filters)



```
>> fspecial('average',[5,7])
```

will return an averaging filter of size 5 × 7; more simply

```
>> fspecial('average',11)
```



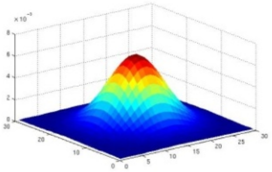
```
>> c=imread('cameraman.tif');
>> f1=fspecial('average');
>> cf1=filter2(f1,c);
```

Produce an Average filtering image using 9x9 filter and a 25x25 filter

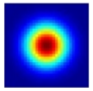
fspecial (low pass filters)

```
>> hsize = 10;
>> sigma = 5;
>> h = fspecial('gaussian' hsize, sigma);
```

```
>> mesh(h);
```



```
>> imagesc(h);
```

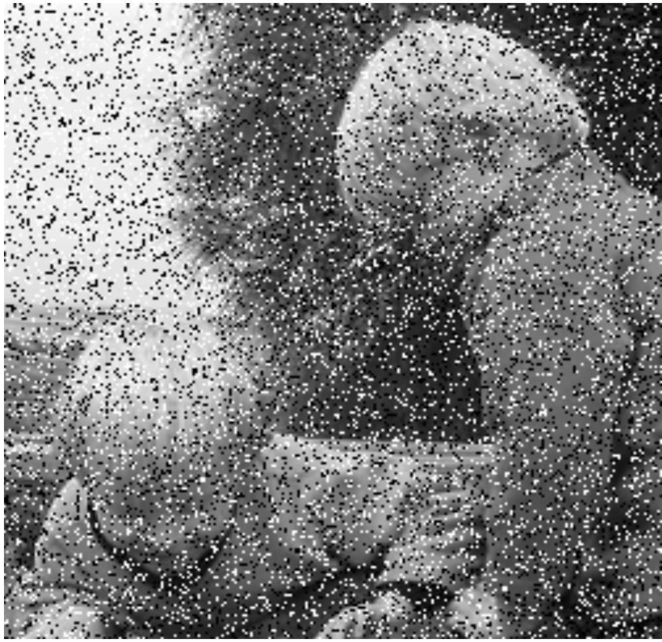


```
>> outim = imfilter(c, h);
>> imshow(outim);
```

fspecial (low pass filters)

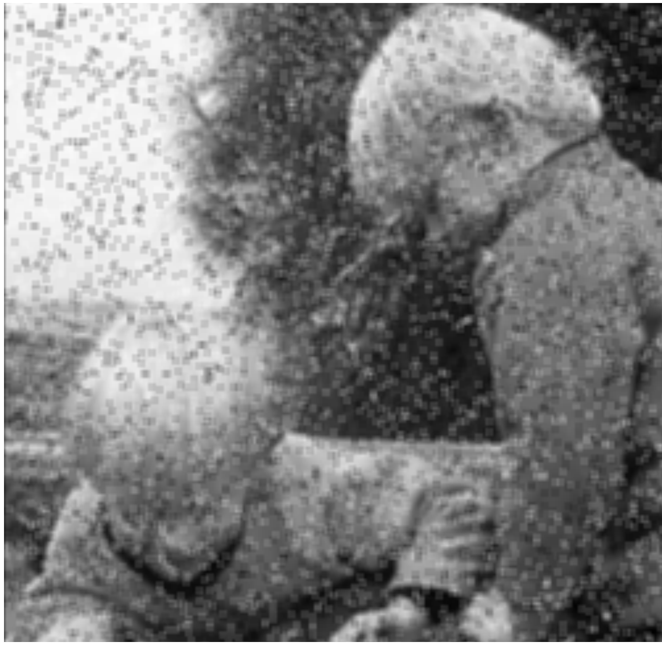
```
for sigma=1:3:10
   h = fspecial('gaussian‘, fsize,
   sigma);
   out = imfilter(c, h);
   imshow(out);
   pause;
end
```

Salt and pepper noise



```
tw=imread('twins.tif');
t=rgb2gray(tw);
t_sp=imnoise(t, 'salt & pepper‘, 0.1);
```

Low-pass filtering



```
a3=fspecial('average');
t_sp_a3=filter2(a3,t_sp);
```

Ex#1. Try 7x7 averaging filter and
3x3 median filter (`medfilt2`)

Gaussian noise

```
t_ga=imnoise(t, 'gaussian');
% default: mean=0, var=0.01
```

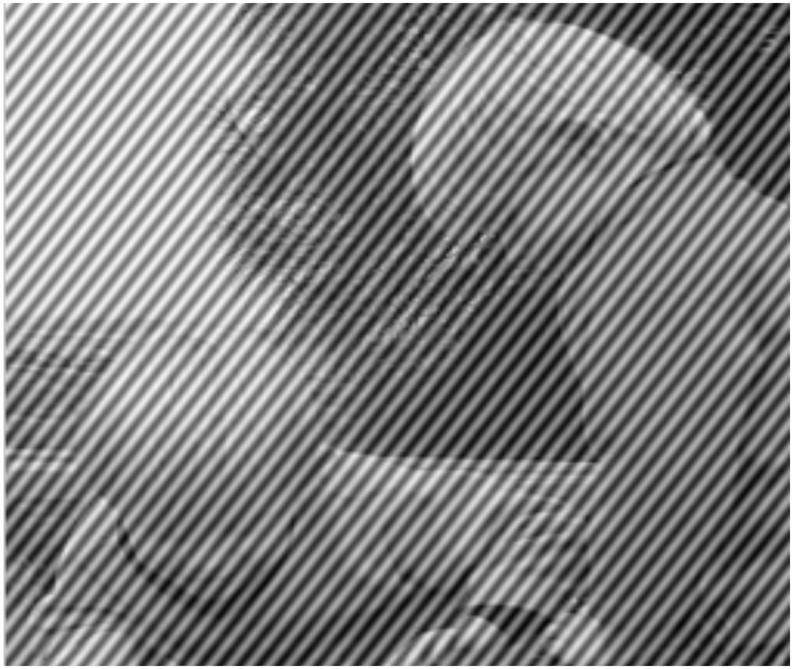
Ex#2: image averaging
For the twin image, generate
10 Gaussian noisy images. Then
take a the average of them.

Ex#3: average filter
Try 3x3 and 5x5 average filter
to the `t_ga` noisy image

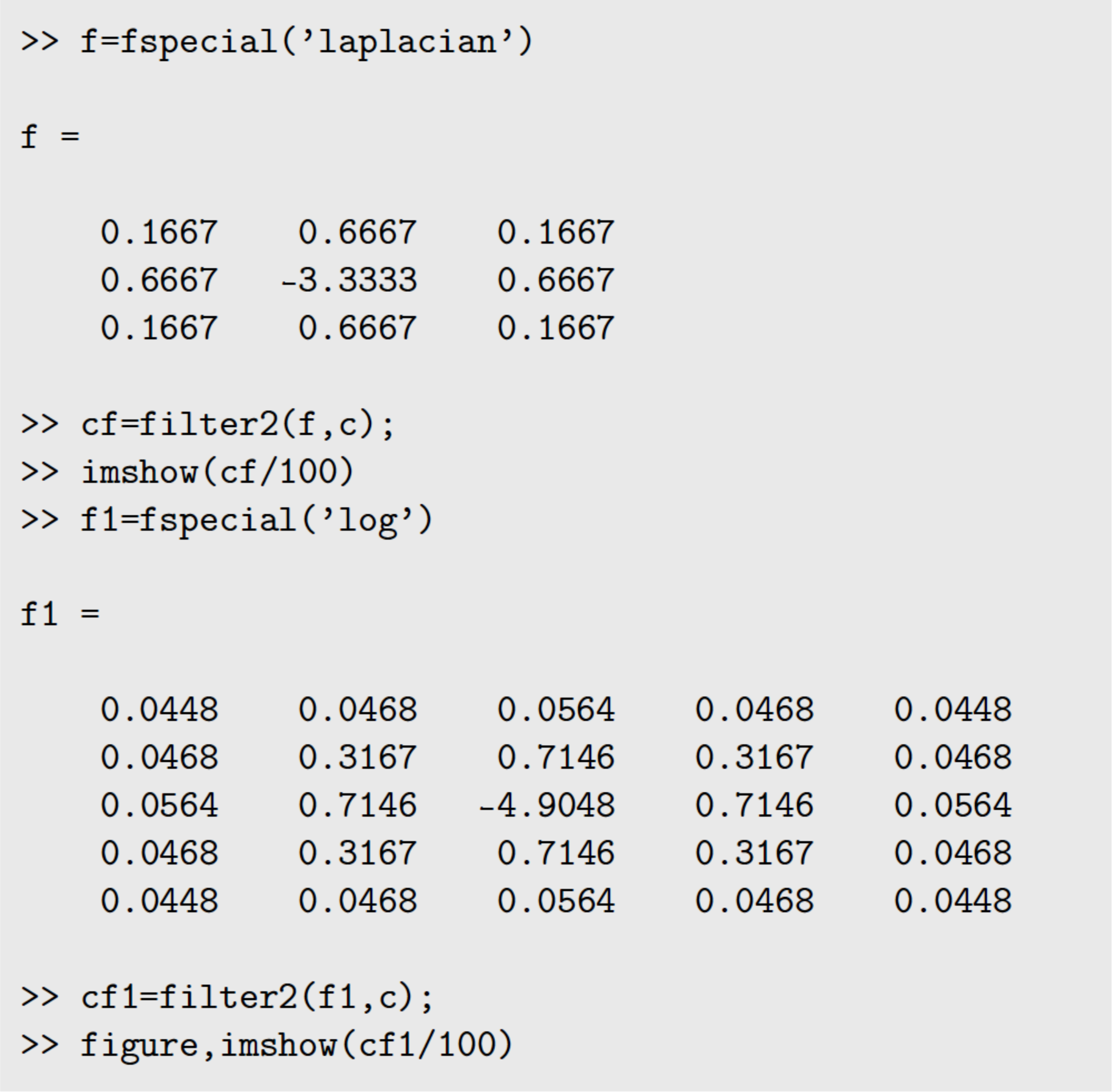
Periodic noise

```
[x,y]=meshgrid(1:size(t,1), 1:size(t,2));
p=sin(x+y/1.5)+1;
t_pn=(double(t)/128+p)/4;
```

Ex#4. Show the DFT of `t_pn`

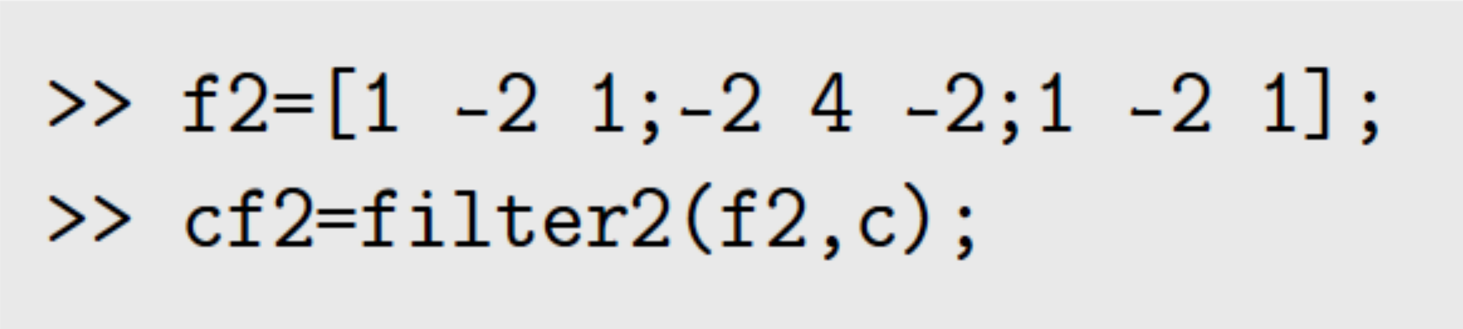


fspecial (high pass filters)

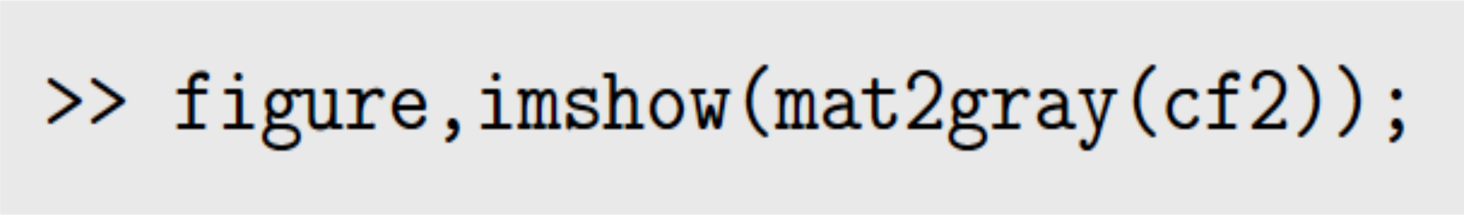


```
>> f=fspecial('laplacian')

f =


    0.1667    0.6667    0.1667
    0.6667   -3.3333    0.6667
    0.1667    0.6667    0.1667

>> cf=filter2(f,c);
>> imshow(cf/100)
>> f1=fspecial('log')

f1 =


    0.0448    0.0468    0.0564    0.0468    0.0448
    0.0468    0.3167    0.7146    0.3167    0.0468
    0.0564    0.7146   -4.9048    0.7146    0.0564
    0.0468    0.3167    0.7146    0.3167    0.0468
    0.0448    0.0468    0.0564    0.0468    0.0448

>> cf1=filter2(f1,c);
>> figure,imshow(cf1/100)
```

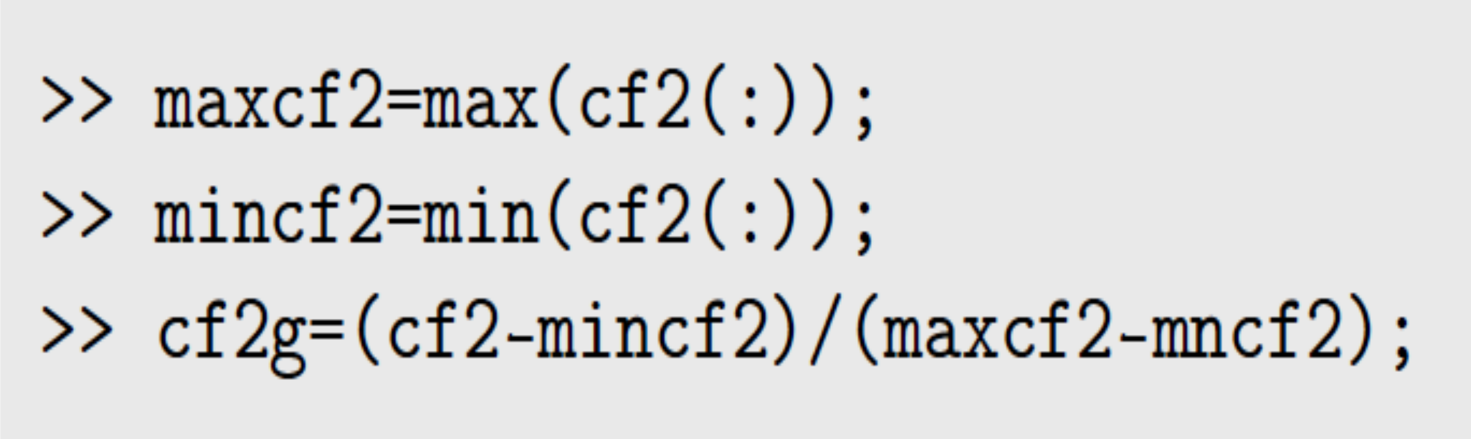
fspecial (high pass filters)



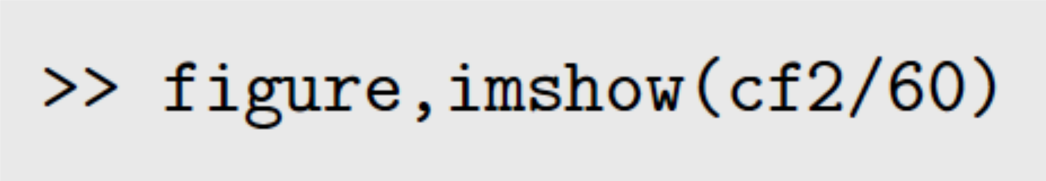
```
>> f2=[1 -2 1;-2 4 -2;1 -2 1];
>> cf2=filter2(f2,c);
```



```
>> figure,imshow(mat2gray(cf2));
```

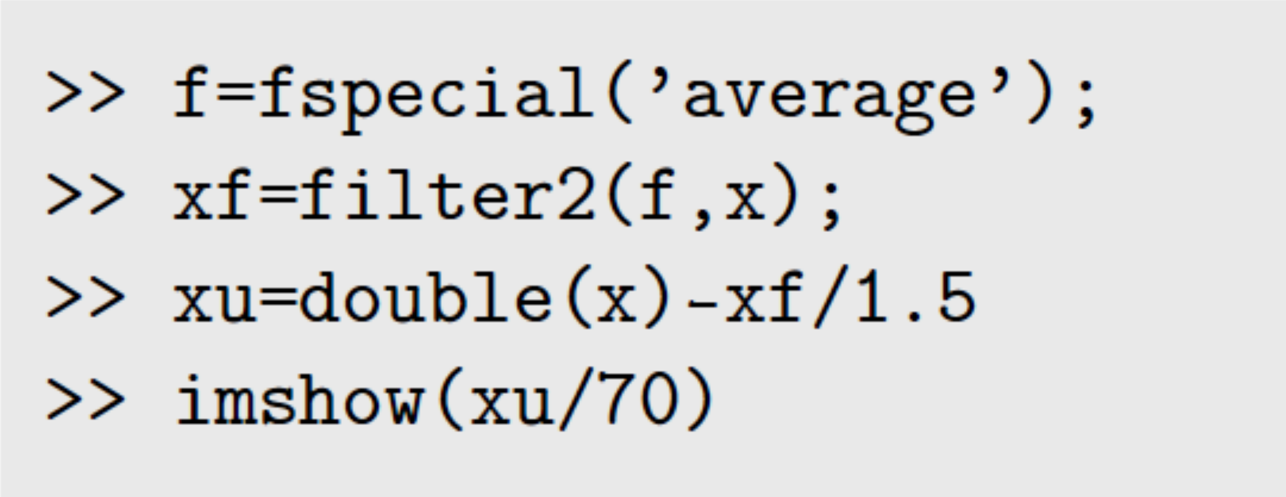


```
>> maxcf2=max(cf2(:));
>> mincf2=min(cf2(:));
>> cf2g=(cf2-mincf2)/(maxcf2-mncf2);
```

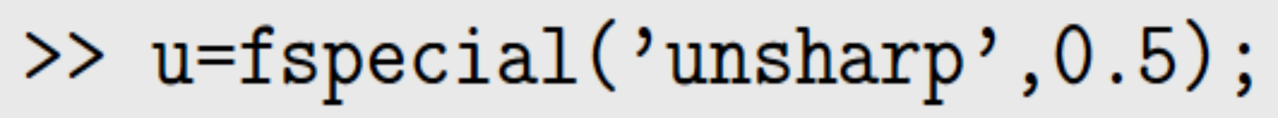


```
>> figure,imshow(cf2/60)
```

Unsharp masking



```
>> f=fspecial(’average’);
>> xf=filter2(f,x);
>> xu=double(x)-xf/1.5
>> imshow(xu/70)
```



```
>> u=fspecial(’unsharp’,0.5);
```

```
>> My = fspecial(‘sobel’);
>> outim = imfilter(double(c), My);
>> imagesc(outim);
>> colormap gray;
```

Find the image ‘pelicans.tif’ from internet and use u filter

Canny edge detector

MATLAB: **`edge(c, ‘canny’);`**

```
>>help edge
```

```
I = imread('circuit.tif');
BW1 = edge(I,'prewitt');
BW2 = edge(I,'canny');
figure, imshow(BW1)
figure, imshow(BW2)
```