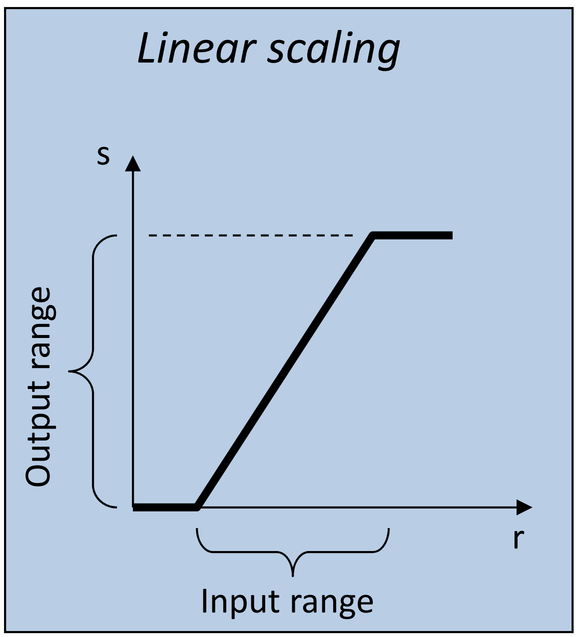
Image Filtering

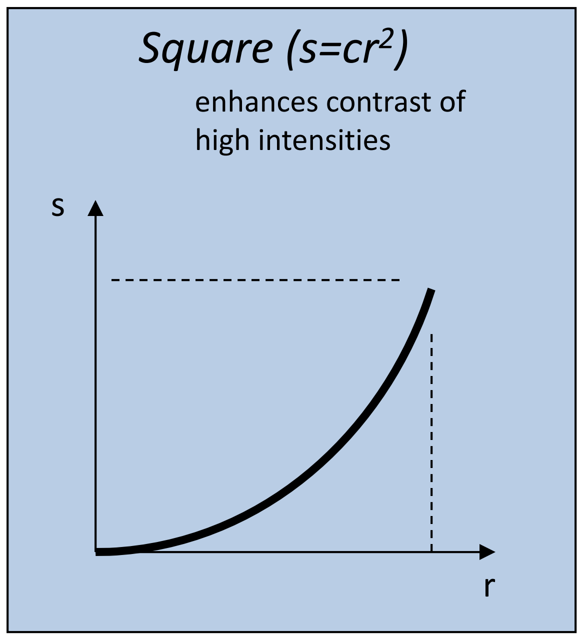
Image Filtering

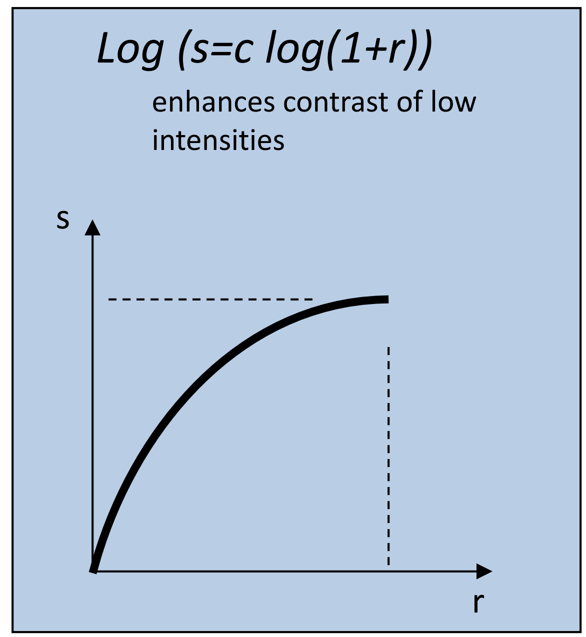
- We will briefly look at methods to
  - Reduce noise and enhance images
  - Detect features
  - (These topics are covered in more detail in EGGN 510)
- Topics
  - Gray level (point transforms)
  - Spatial (neighborhood) transforms
  - Binary image processing

Gray Level Transformations

- Point operations
  - *s = f(r)*
  - Map input pixel value *r* to output value *s*
- Examples







Gamma Correction

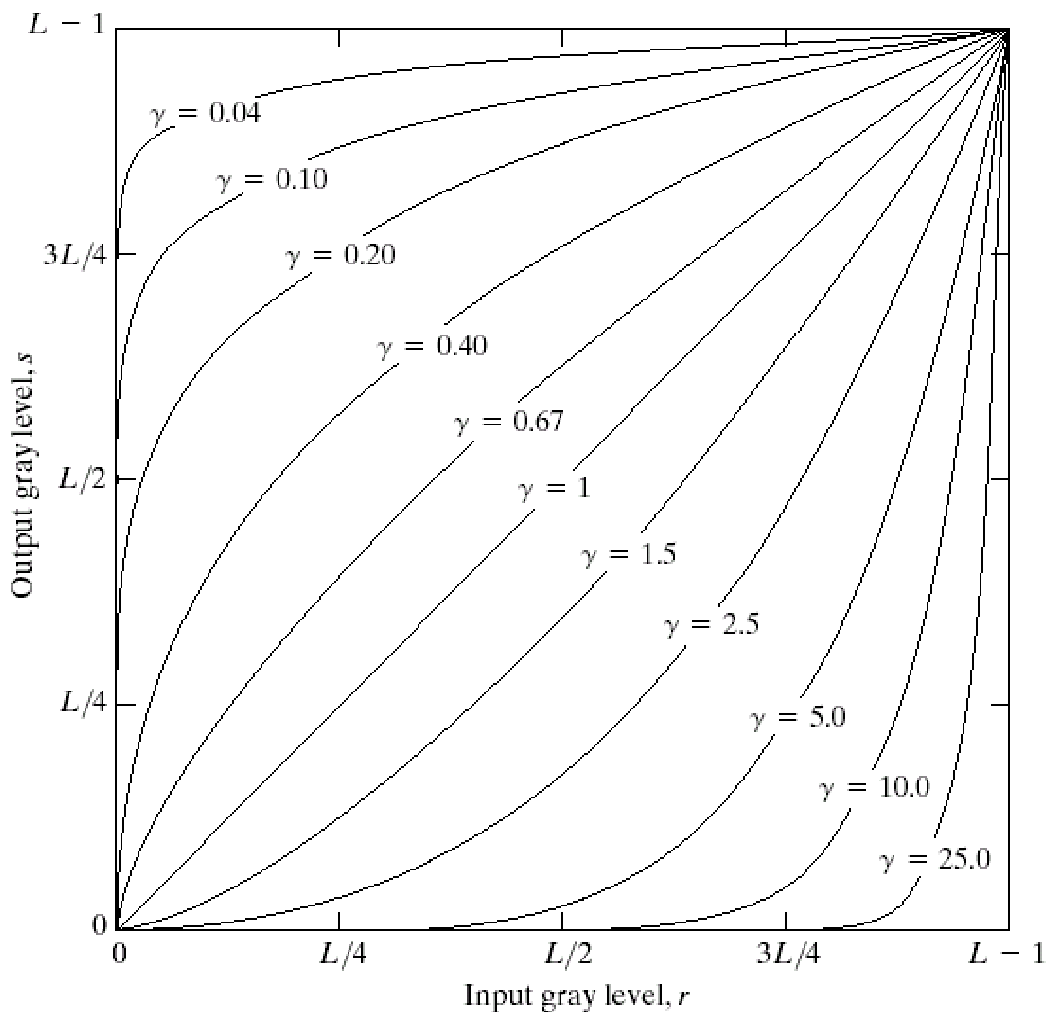


FIGURE 3.6 Plots of the equation $s = cr^{\gamma}$ for various values of γ ($c = 1$ in all cases).

a	b
c	d

FIGURE 3.9
(a) Aerial image. (b)–(d) Results of applying the transformation in Eq. (3.2-3) with $c = 1$ and $\gamma = 3.0, 4.0,$ and 5.0, respectively. (Original image for this example courtesy of NASA.)

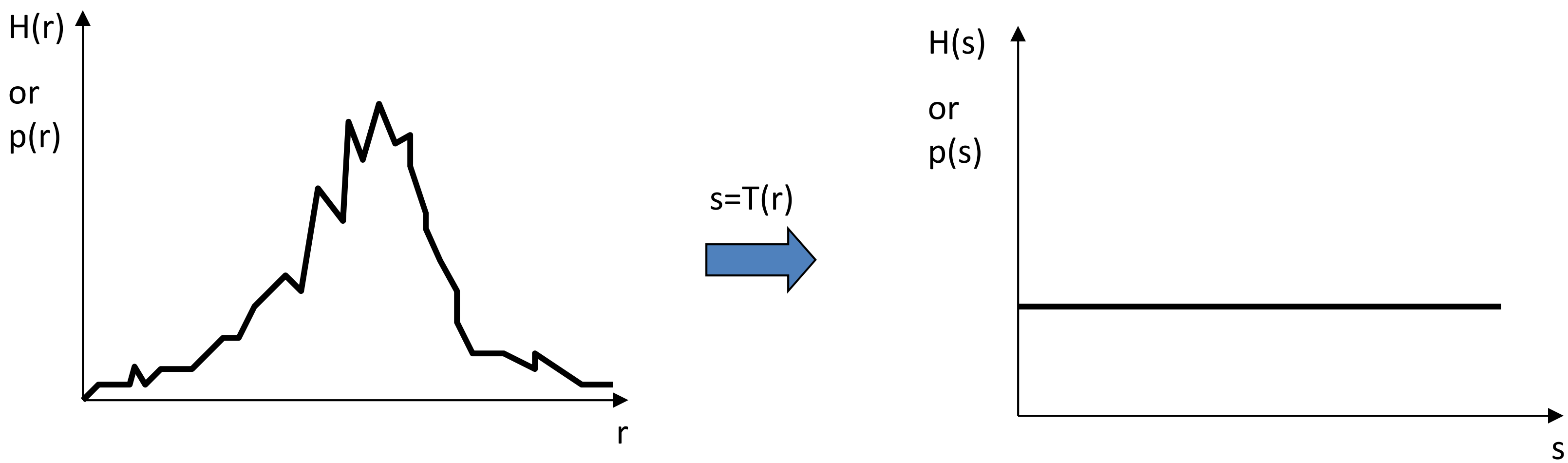
Demos

- Matlab
  - Load image pout.tif
    - Sample images are in C:\Program Files\MATLAB\R2010b\toolbox\image\imdemos
  - See histogram
    - `imhist`
  - Adjust limits (linear scaling)
    - `imtool`

- Photoshop
  - Can draw arbitrary transformation curve
  - Image->adjustments->curves …

Histogram Equalization

- Think of the histogram H(r) as the (scaled) probability distribution of the input image values
- Want histogram of the output image to be flat: *p(s)* = const

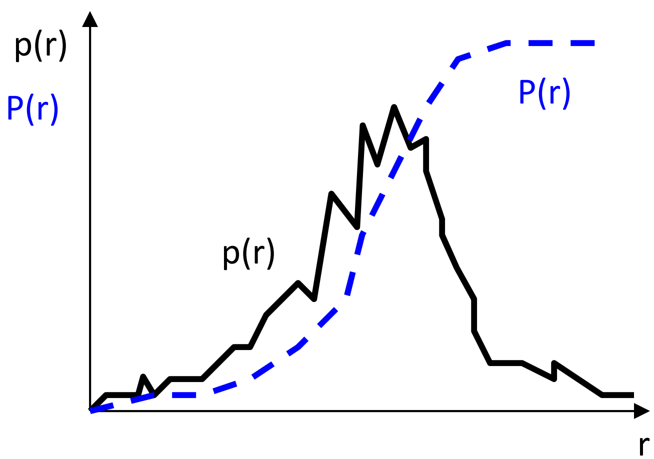


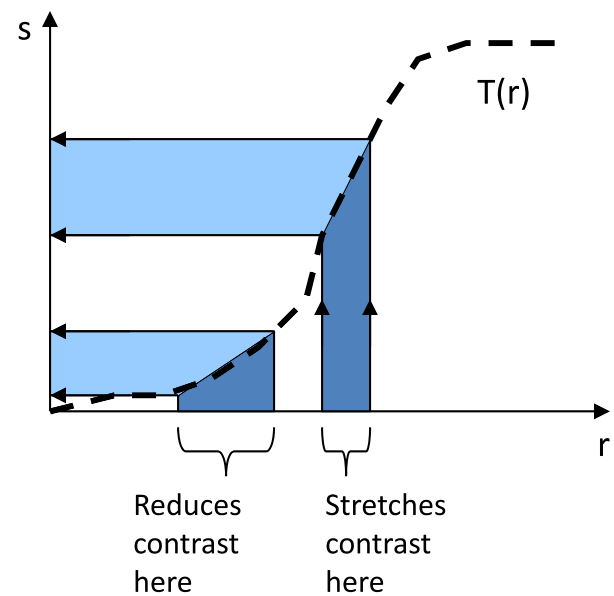
- This stretches contrast where the original image had many pixels with a certain range of gray levels and compresses contrast elsewhere

Mapping Function

- The desired mapping function is just the cumulative probability distribution function of the input image

$$P(r) = \int_0^r p_r(w)\,dw$$

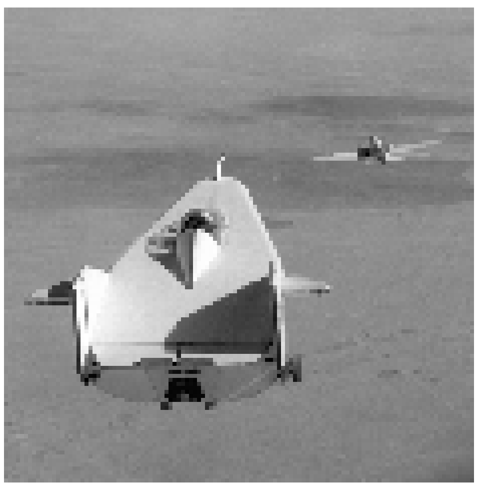




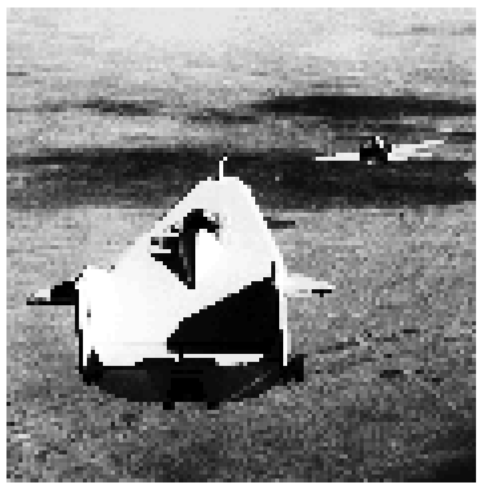
Examples

- Matlab histogram equalization
  - `histeq(I);`
  - To see transform function
    - `[I2,T]=histeq(I);`
    - `plot(T)`
- Adaptive histogram equalization
  - Apply to local neighborhoods
  - `adapthisteq(I);`

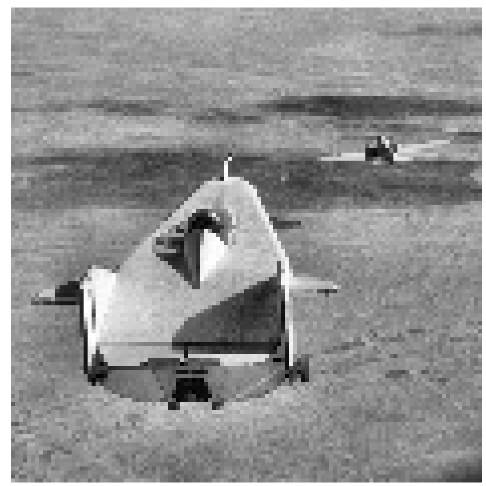
Try image "liftingbody.png"



Input image "liftingbody.png"



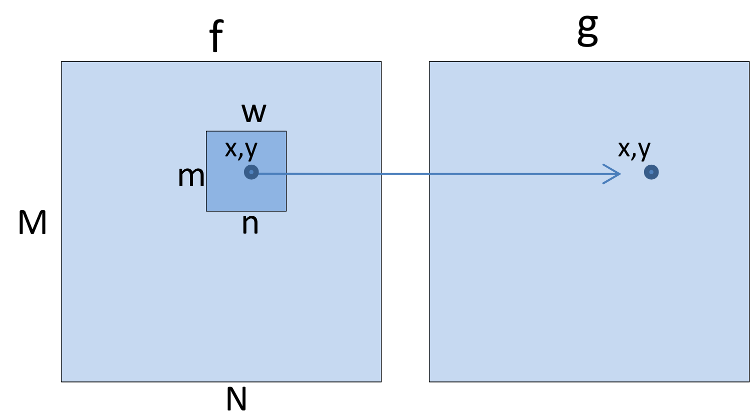
Histogram equalization



Adaptive histogram equalization

Spatial Filtering

- Filter or mask w, size m x n
- Apply to image f, size M x N
- Sum of products of mask coeffs with corresponding pixels under mask
- Slide mask over image, apply at each point
- Also called "cross-correlation"



$$g(x,y) = \sum_{s=-m/2}^{m/2} \sum_{t=-n/2}^{n/2} w(s,t)\; f(x+s, y+t)$$

$$= w(x,y) \otimes f(x,y)$$

Example

- Box or averaging filter
- Can use to smooth image (blur, remove noise)
- Manual calculation on corner image?

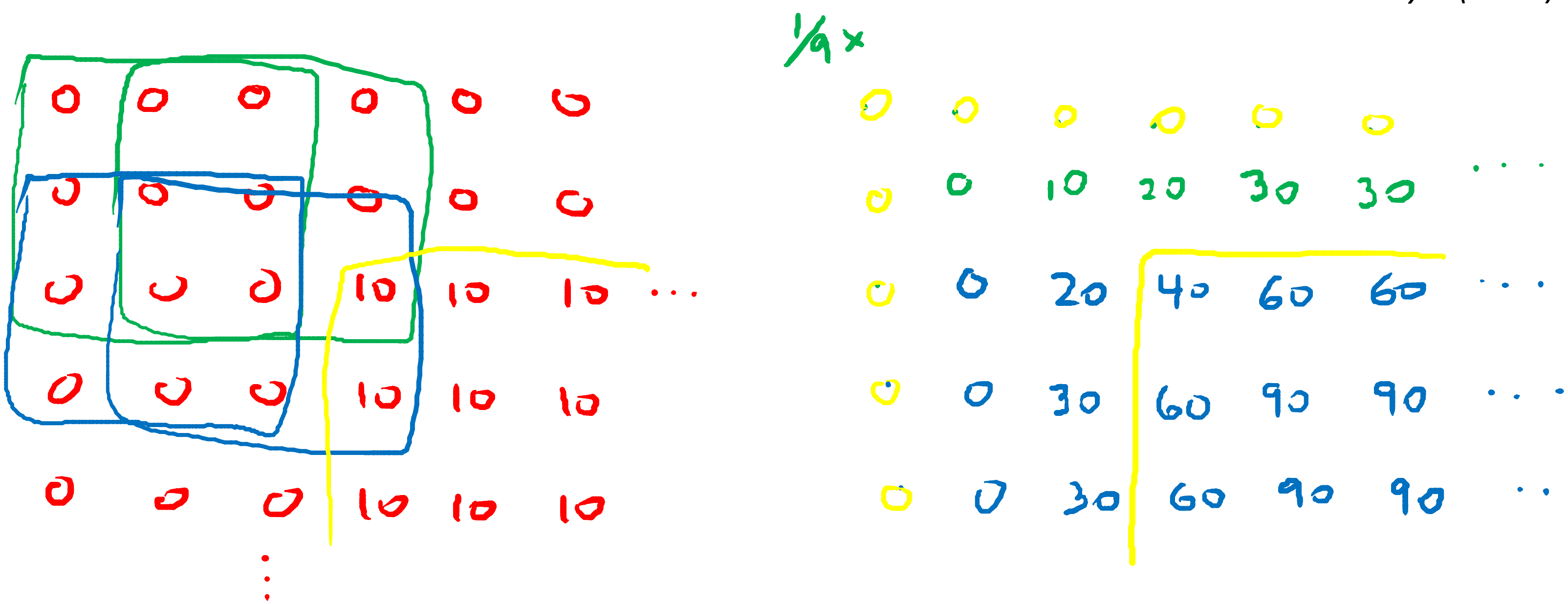
$\frac{1}{9} \times$

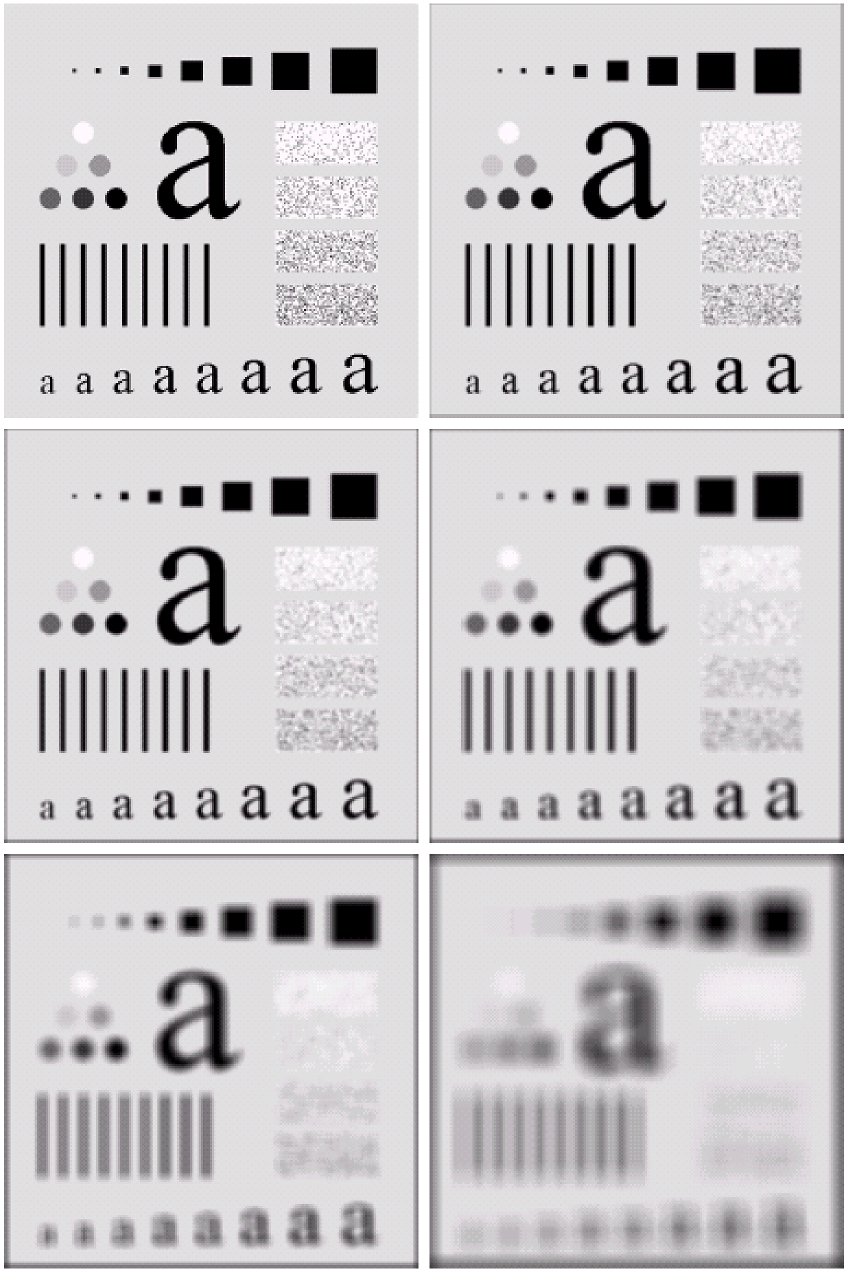
1	1	1
1	1	1
1	1	1

$\frac{1}{16} \times$

1	2	1
2	4	2
1	2	1

Remember to divide by 9 (or 16)





a b
c d
e f

FIGURE 3.35 (a) Original image, of size 500 × 500 pixels. (b)–(f) Results of smoothing with square averaging filter masks of sizes $n = 3, 5, 9, 15$, and 35, respectively. The black squares at the top are of sizes 3, 5, 9, 15, 25, 35, 45, and 55 pixels, respectively; their borders are 25 pixels apart. The letters at the bottom range in size from 10 to 24 points, in increments of 2 points; the large letter at the top is 60 points. The vertical bars are 5 pixels wide and 100 pixels high; their separation is 20 pixels. The diameter of the circles is 25 pixels, and their borders are 15 pixels apart; their gray levels range from 0% to 100% black in increments of 20%. The background of the image is 10% black. The noisy rectangles are of size 50 × 120 pixels.

Matlab Examples

- `fspecial` - useful to create filters
- `imfilter` - to apply to an image

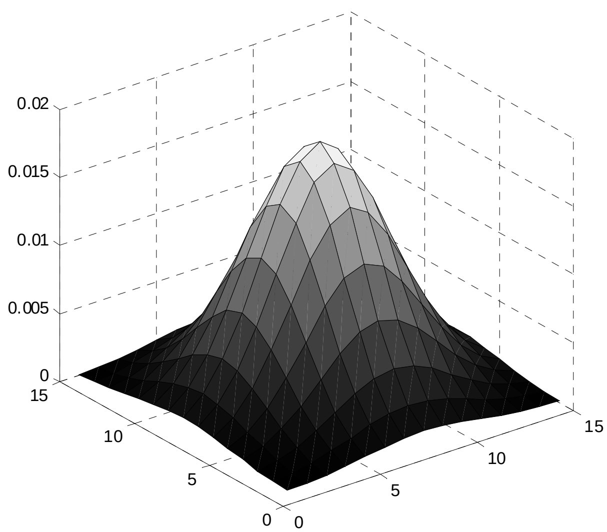
```
>> clear all
>> close all
>> I = zeros(200,200);
>> I(50:150, 50:150) = 1;
>> imshow(I,[])
>> w = [ 1 1 1; 1 1 1; 1 1 1]/9

w =

   0.1111   0.1111   0.1111
   0.1111   0.1111   0.1111
   0.1111   0.1111   0.1111

>> I2 = imfilter(I, w);
>> imtool(I2,[])
>> w = fspecial('average', [15 15])
```

```
>> I3 = imfilter(I, w);
>> imtool(I3,[])
>>
>>
>>
>> I = I + (rand(200,200)-0.5)*0.5;
>> imshow(I,[])
>> imshow(I,[])
>> w = fspecial('average', [5 5]);
>> impixelinfo
>> I4 = imfilter(I, w);
>> imshow(I4,[])
```

Gaussian Smoothing Filter

- Gaussian filter usually preferable to box filter
- Attenuates high frequencies better

$$h(x,y) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/(2\sigma^2)}$$



15x15 Gaussian, σ=3 (scaled to 1)

0.01	0.02	0.03	0.06	0.08	0.11	0.13	0.14	0.13	0.11	0.08	0.06	0.03	0.02	0.01
0.02	0.03	0.06	0.10	0.15	0.20	0.24	0.25	0.24	0.20	0.15	0.10	0.06	0.03	0.02
0.03	0.06	0.10	0.17	0.25	0.33	0.39	0.41	0.39	0.33	0.25	0.17	0.10	0.06	0.03
0.04	0.08	0.15	0.25	0.37	0.49	0.57	0.61	0.57	0.49	0.37	0.25	0.15	0.08	0.04
0.05	0.11	0.20	0.33	0.49	0.64	0.76	0.80	0.76	0.64	0.49	0.33	0.20	0.11	0.05
0.06	0.13	0.24	0.39	0.57	0.76	0.89	0.95	0.89	0.76	0.57	0.39	0.24	0.13	0.06
0.07	0.14	0.25	0.41	0.61	0.80	0.95	1.00	0.95	0.80	0.61	0.41	0.25	0.14	0.07
0.06	0.13	0.24	0.39	0.57	0.76	0.89	0.95	0.89	0.76	0.57	0.39	0.24	0.13	0.06
0.05	0.11	0.20	0.33	0.49	0.64	0.76	0.80	0.76	0.64	0.49	0.33	0.20	0.11	0.05
0.04	0.08	0.15	0.25	0.37	0.49	0.57	0.61	0.57	0.49	0.37	0.25	0.15	0.08	0.04
0.03	0.06	0.10	0.17	0.25	0.33	0.39	0.41	0.39	0.33	0.25	0.17	0.10	0.06	0.03
0.02	0.03	0.06	0.10	0.15	0.20	0.24	0.25	0.24	0.20	0.15	0.10	0.06	0.03	0.02
0.01	0.02	0.03	0.06	0.08	0.11	0.13	0.14	0.13	0.11	0.08	0.06	0.03	0.02	0.01

Matlab "surf" function

Convolution vs Correlation

- Cross-correlation of mask h(x,y) with image f(x,y)

$$g(x,y) = \sum_{s=-m/2}^{m/2} \sum_{t=-n/2}^{n/2} h(s,t)\; f(x+s, y+t) = h \otimes f$$

- Convolution of mask h(x,y) with image f(x,y)

$$g(x,y) = \sum_{s=-m/2}^{m/2} \sum_{t=-n/2}^{n/2} h(s,t)\; f(x-s, y-t) = h * f$$

- or

$$g(x,y) = \sum_{s=-m/2}^{m/2} \sum_{t=-n/2}^{n/2} f(x,y)\; h(x-s, y-t) = f * h$$

- Convolution same as correlation except that we first flip one function about the origin

Sharpening Spatial Filters

$$\frac{\partial f}{\partial x} = \lim_{\epsilon \to 0} \frac{f(x+\epsilon) - f(x)}{\epsilon}$$

- First derivative (can also do central difference)

$$\frac{\partial f}{\partial x} \approx f(x+1) - f(x)$$

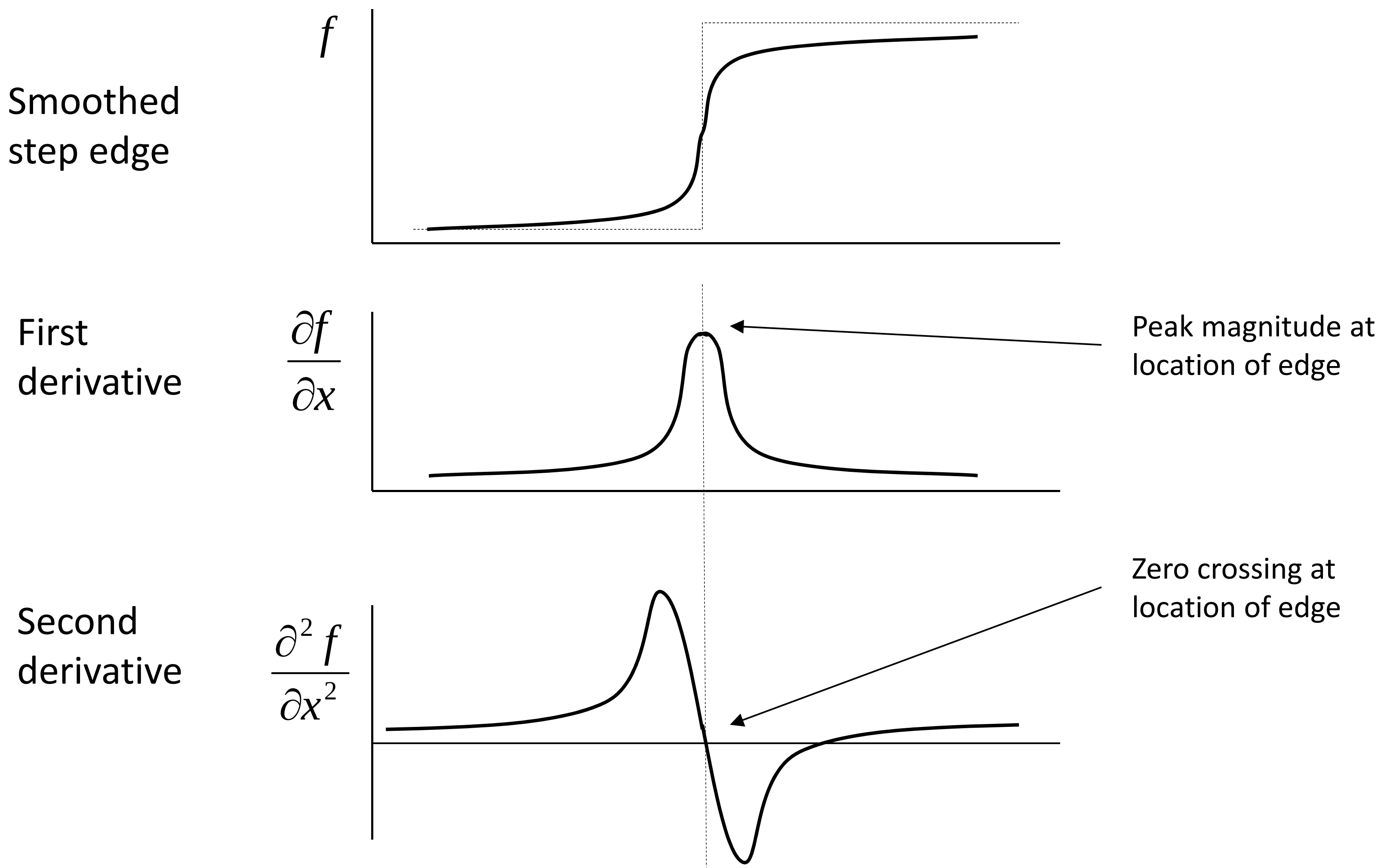
-1	+1

- Second derivative

$$\frac{\partial^2 f}{\partial x^2} \approx f(x+1) - 2f(x) + f(x-1)$$

+1	-2	+1

Edge Detection



Edge Operators for 2D Images

$\frac{\partial}{\partial x}$

-1	0	+1
-2	0	+2
-1	0	+1

$\frac{\partial}{\partial y}$

-1	-2	-1
0	0	0
+1	+2	+1

Sobel operators

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$$

0	1	0
1	-4	1
0	1	0

Laplacian operator

- Example:
  - Manual calculation of Sobel on corner
  - Compare correlation vs convolution

Matlab Examples

- Try image “moon.tif”
- Create Sobel masks
  - `hx = [ -1 0 1; -2 0 2; -1 0 1]; hy = hx’;`
- imfilter to do correlation
  - See difference if convert image to double first
  - `I=double(I);   % just changes type`
  - `I=im2double(I); % change type and scale to 0..1`
- Notes
  - `filter2` – same as imfilter but always converts to double
  - `conv2` – does convolution

Gradient

- Compute gradient components using first derivative operators

$$\nabla \mathbf{f} = \begin{bmatrix} \partial f / \partial x \\ \partial f / \partial y \end{bmatrix}$$

- Gradient magnitude shows location of edges in the image

$$\left|\nabla \mathbf{f}\right| = \left[\left(\frac{\partial f}{\partial x} \right)^2 + \left(\frac{\partial f}{\partial y} \right)^2 \right]^{1/2}$$

- Gradient angle shows direction of edge

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} \Big/ \frac{\partial f}{\partial x} \right)$$

Gradient in Matlab

- Compute gradient components using first derivative operators
  - `Dx = imfilter(I,hx)`

$$\nabla \mathbf{f} = \begin{bmatrix} \partial f / \partial x \\ \partial f / \partial y \end{bmatrix}$$

- Gradient magnitude peaks at locations of edges in the image
  - `(Dx.^2+Dy.^2) .^ 0.5`

$$|\nabla \mathbf{f}| = \left[\left(\frac{\partial f}{\partial x} \right)^2 + \left(\frac{\partial f}{\partial y} \right)^2 \right]^{1/2}$$

Note – the period in Matlab indicates a point-by-point operation instead of a matrix operation

- Gradient angle shows direction of edge
  - `atan2(Dy,Dx)`
  - `colormap jet`
  - `colorbar`

$$\theta = \tan^{-1} \left(\frac{\partial f}{\partial y} \Big/ \frac{\partial f}{\partial x} \right)$$