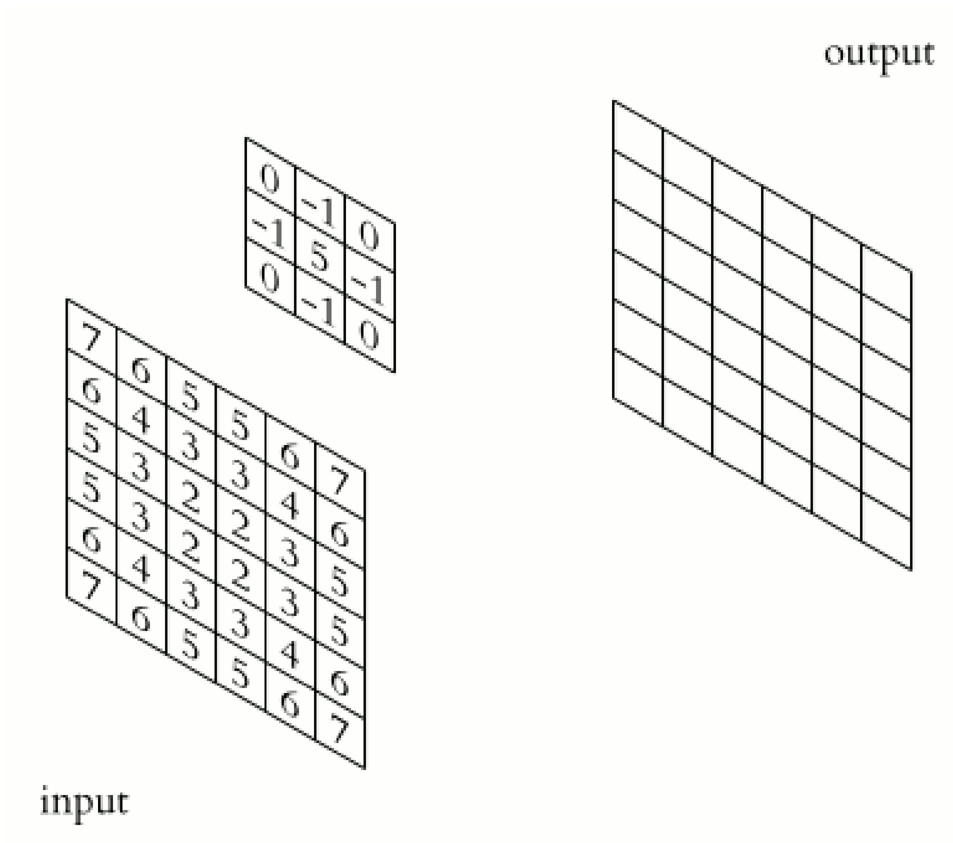
Image Filtering, Convolution

- Image filtered by convolving with a filter kernel
- Convolution denoted by “$*$”

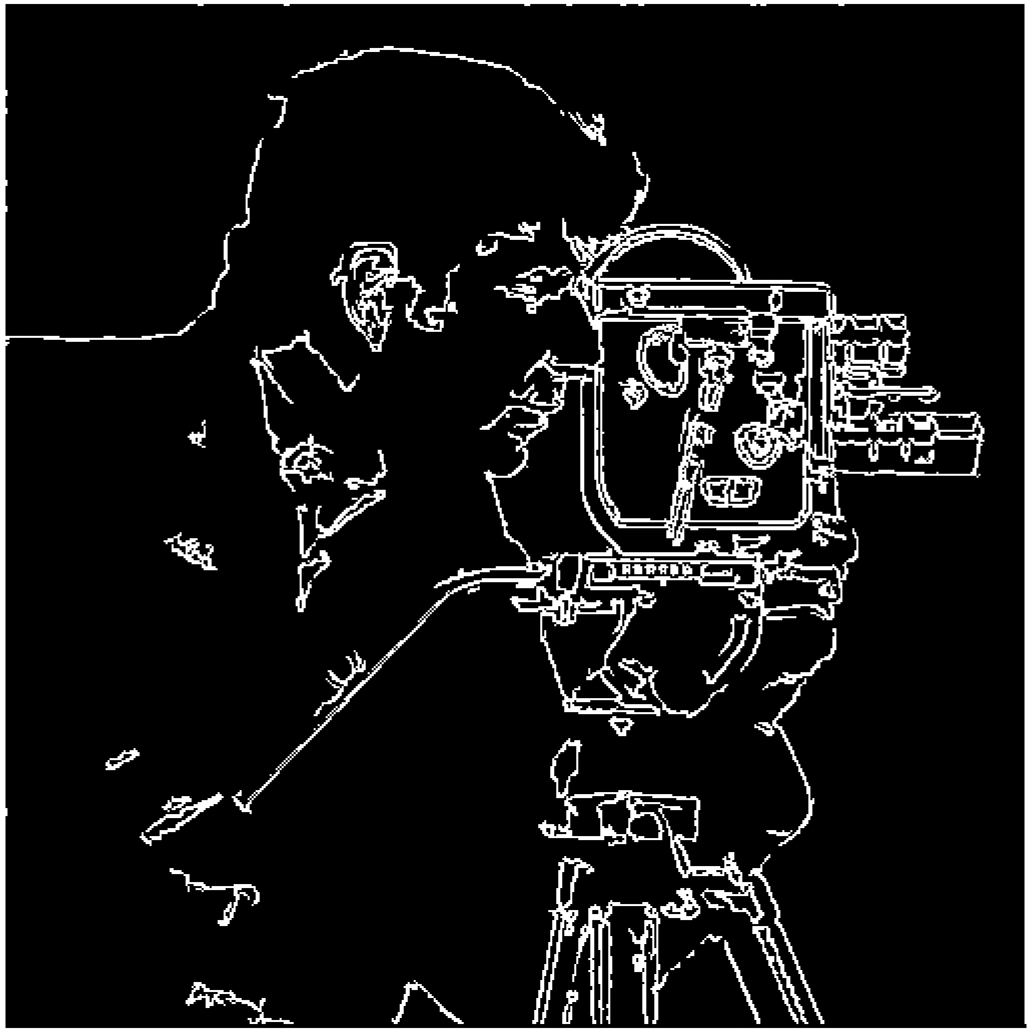
$$I_{output} = k * I_{input}$$



http://en.wikipedia.org/wiki/Kernel_(image_processing)

Original	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge-Detect	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Blur*	$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	

Exercise 2 – Edge Detection

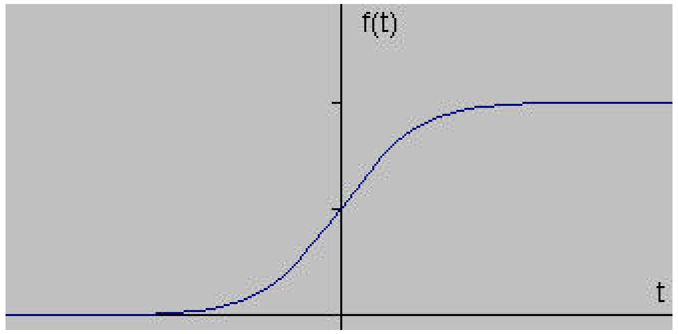


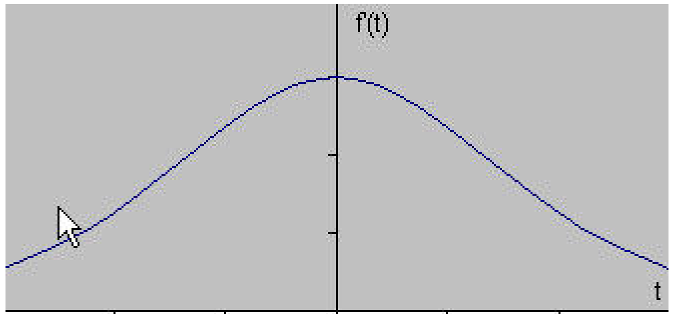
http://vision.cs.arizona.edu/nvs/research/image_analysis/edge.html

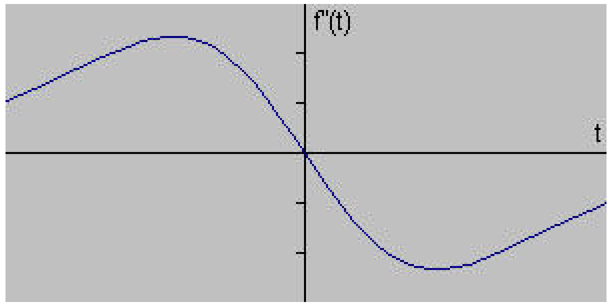
Edges

Edges in images are areas with strong intensity contrasts

- Change is measured by derivative in 1D
- Biggest change, derivative has maximum magnitude
- Or 2nd derivative is zero







http://www.pages.drexel.edu/~weg22/edge.html

Gradient Method

Gradient Vector
$$\mathbf{g}(x,y) = \begin{bmatrix} g_x(x,y) \\ g_y(x,y) \end{bmatrix} = \begin{bmatrix} (k_x * f)(x,y) \\ (k_y * f)(x,y) \end{bmatrix}$$

Gradient Magnitude
$$|\mathbf{g}| = \left(g_x^2 + g_y^2\right)^{1/2}$$

Direction
$$\theta = \tan^{-1}\left(\frac{g_y}{g_x}\right)$$

Use *atan2* in Matlab!

Sobel kernel

Approximate of the 2D derivative of an image

$$k_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad k_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Prewitt kernel

Approximate of the 2D derivative of an image

$$k_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad k_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Canny Edge Detection

Combine noise reduction and edge enhancement.

1. Apply derivative of Gaussian filter
2. Non-maximum suppression
   - Thin multi-pixel wide “ridges” down to single pixel width
3. Hysteresis
   - Accept all edges over low threshold that are connected to edge over high threshold

Derivative of Gaussian kernel

- Need smoothing to reduce noise prior to taking derivative

  fspecial('gaussian', 5, 1.4)

0.0121	0.0261	0.0337	0.0261	0.0121
0.0261	0.0561	0.0724	0.0561	0.0261
0.0337	0.0724	0.0935	0.0724	0.0337
0.0261	0.0561	0.0724	0.0561	0.0261
0.0121	0.0261	0.0337	0.0261	0.0121

- We can use derivative of Gaussian filters
  - because differentiation is convolution, and convolution is associative:

$$D * (G * I) = (D * G) * I$$

Non-maximum suppression

- The edge direction angle is rounded to one of four angles representing vertical, horizontal and the two diagonals.
- Select the single maximum point across the width of an edge.
    - Maximum: The gradient magnitudes of the two neighbors in edge normal direction are smaller.

courtesy of G. Loy

Hysteresis

- Idea: Maintain two thresholds T_{high} and T_{low}
  - Use T_{high} to find strong edges to start edge chain
  - Use T_{low} to find weak edges which continue edge chain
- Typical ratio of thresholds is roughly

$$T_{high} / T_{low} = 2$$

Hysteresis

Strong edges only
> T_{high}

Weak edges
> T_{low}

gap is gone

Strong + connected weak edges

courtesy of G. Loy

Matlab

conv2

Two-dimensional convolution

fspecial('gaussian',HSIZE,SIGMA)

Creates a two-dimensional gaussian filter

edge

Built-in matlab function for finding edges

bwselect(BW1,C,R,N)

Returns a binary image containing the objects that overlap the pixel (R,C)

Test Image