Computer Vision I

Edge Detection

Roi Santos Mateos

Escola Técnica Superior de Enxeñaría, USC

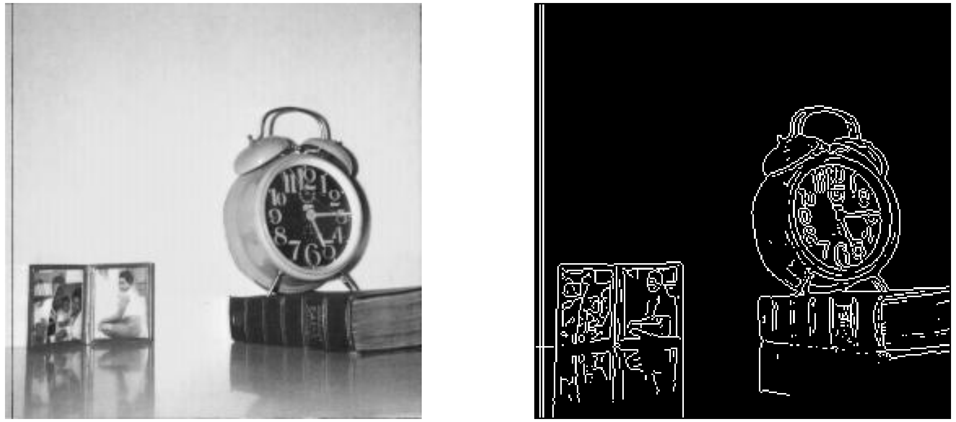
Academic year 2025/26

MASTER IN ARTIFICIAL INTELLIGENCE

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA
UNIVERSIDADE DA CORUÑA
Universidade de Vigo

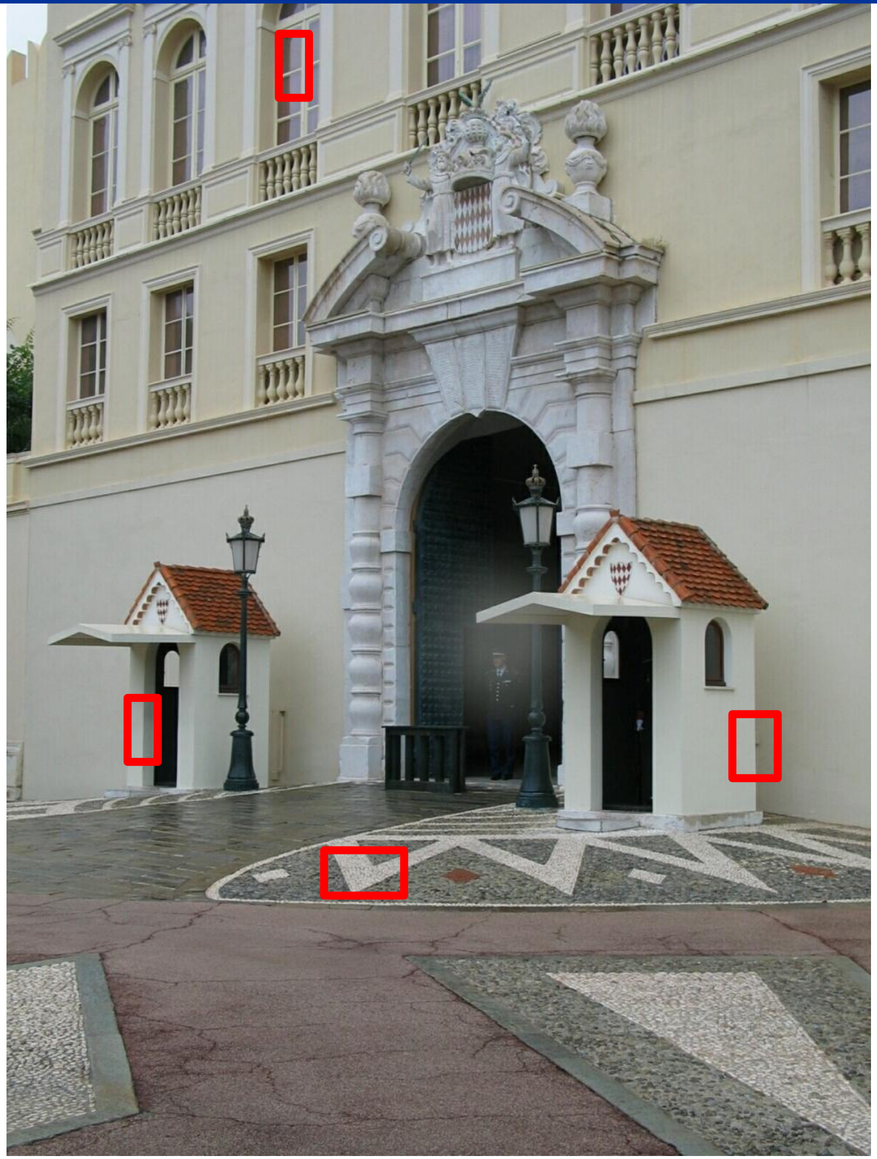
Edge Detection

- **Goal:** Identify abrupt changes (discontinuities) in image intensity
  - Most semantic and shape information from an image can be encoded in its edges
    - Information for object detection
    - Recovering of geometry and viewpoint
    - ...



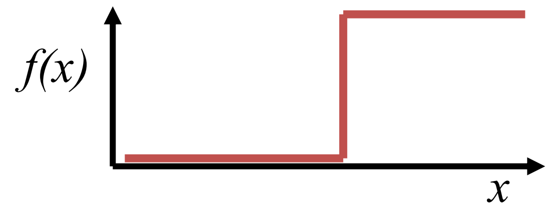
Edge Detection

- Origin of edges:
  - Surface normal discontinuity
    - left sentry box
  - Surface color discontinuity
    - floor
  - Illumination discontinuity
    - window
  - Depth discontinuity
    - right sentry box

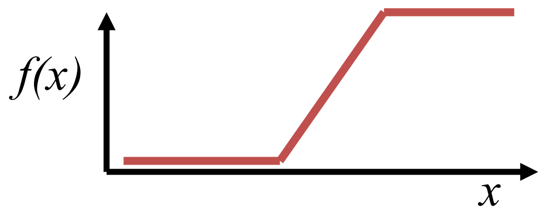


Edge Detection

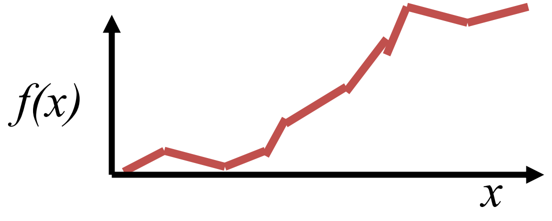
- Ideal edge



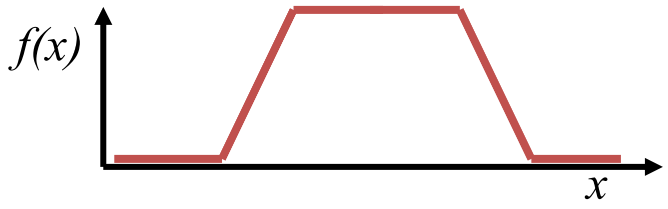
- Usually, real edges have the shape of a ramp due to sensor processing during capture



- Real "noisy" edge



- (Wide) Line or blob



Edge Detection

- Features of 1D edges:
  - Derivative magnitude $|df/dx|$:

$$\frac{df}{dx} = \lim_{\Delta x \to 0} \frac{f(x) - f(x - \Delta x)}{\Delta x} = f'(x) = f_x$$

  - Derivative direction (left, right), sign of df/dx:

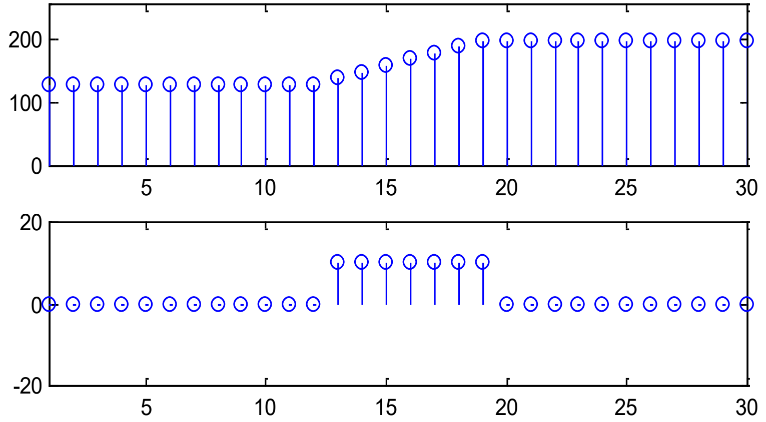
$$\frac{f_x}{|f_x|}$$

- Discrete derivative (finite differences approximation)

$$\frac{df}{dx} = \frac{f(x) - f(x-1)}{1} = f'(x)$$

$$\frac{df}{dx} = f(x) - f(x-1) = f'(x)$$

$f'(x)=B1*f(x)$, $B1=[-1\ 1]$



$f(x)$

$B1*f$

Derivative computation expressed as a convolution

Derivative Kernels

- 1D Derivative kernels in 2D (image coordinates system)

x-derivative

-1	1

y-derivative

-1
1

- Roberts filters (Central derivative):

-1	0	1

-1
0
1

Derivative Kernels

- Prewitt filters:

3x3 mask, x-derivative

-1	0	1
-1	0	1
-1	0	1

3x3 mask, y-derivative

-1	-1	-1
0	0	0
1	1	1

- Sobel filters:
  - Approximation of a Gaussian derivative.

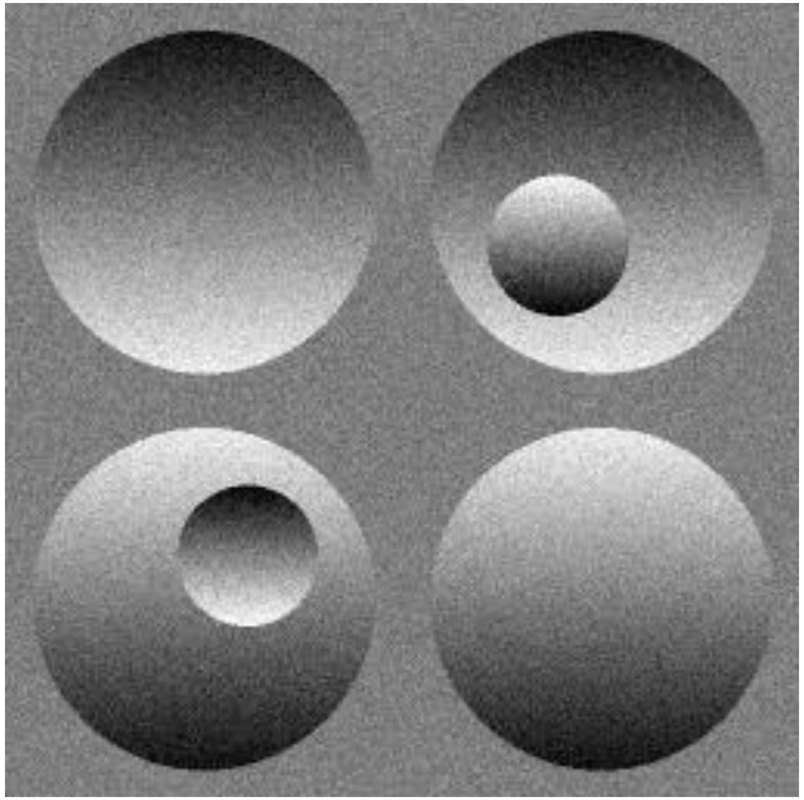
3x3 mask, x-derivative

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

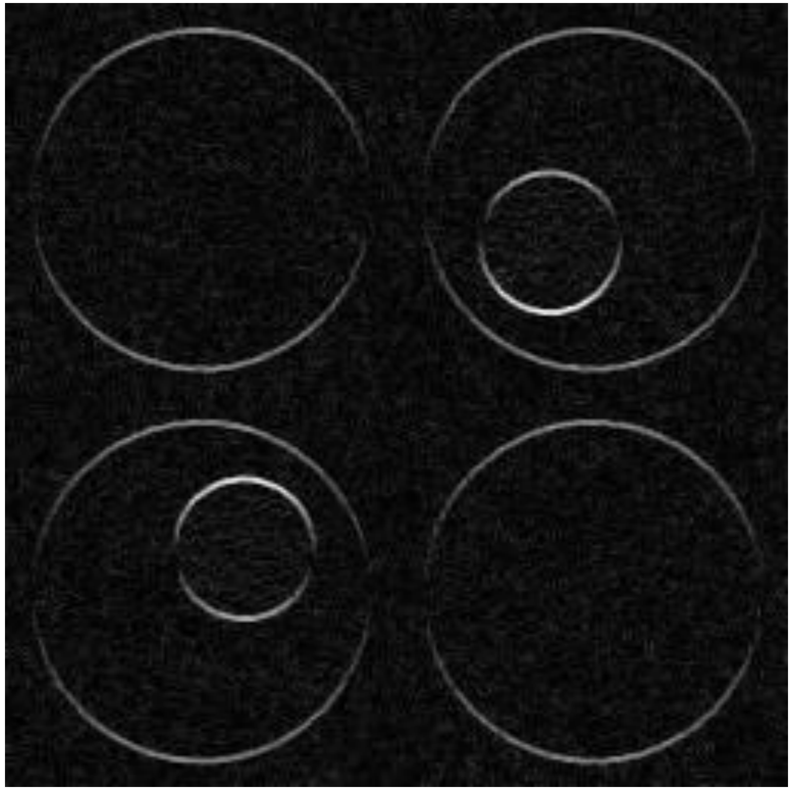
3x3 mask, y-derivative

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

Derivative Kernels



Input image



Convolution with Sobel-y (3x3)

2D Gradient

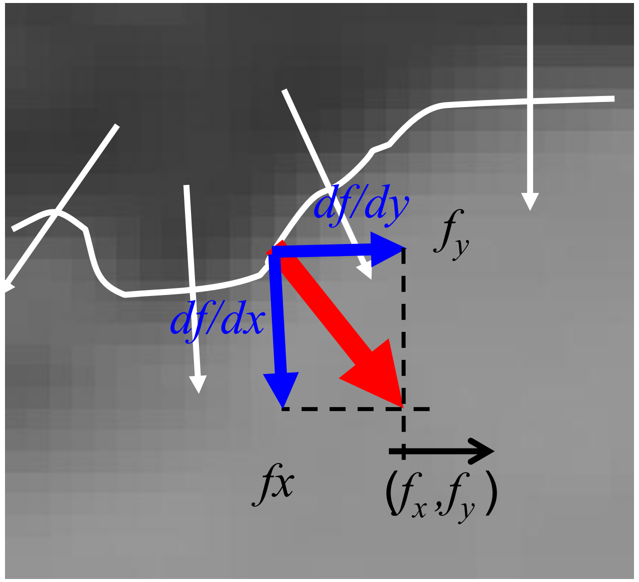
- 2D gradient is a vector with direction of largest slope.

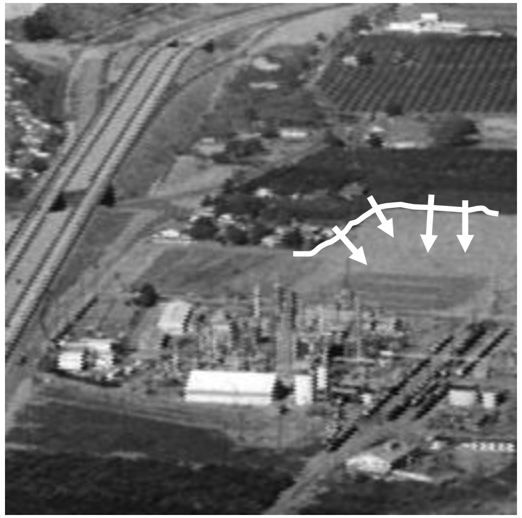
$$\nabla f(x,y) = \begin{bmatrix} \dfrac{\partial f(x,y)}{\partial x} \\ \dfrac{\partial f(x,y)}{\partial y} \end{bmatrix} = \begin{bmatrix} f_x \\ f_y \end{bmatrix}$$

Magnitude and Orientation:

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

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



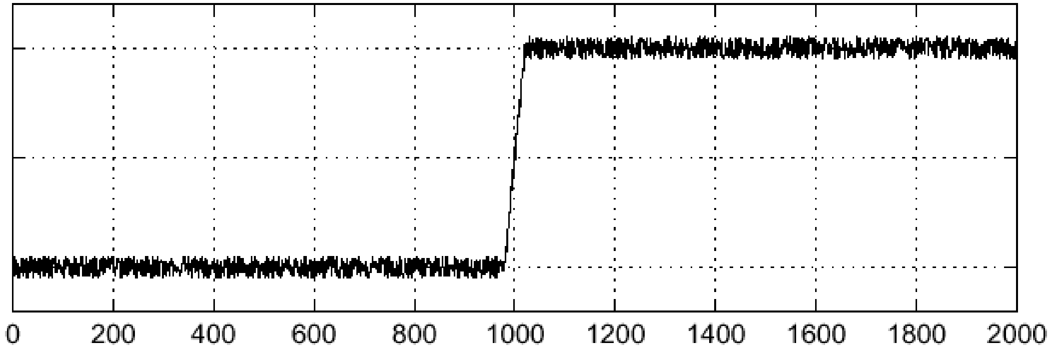


Be aware of the coordinate system libraries are using. Be consistent!

Noise Effect

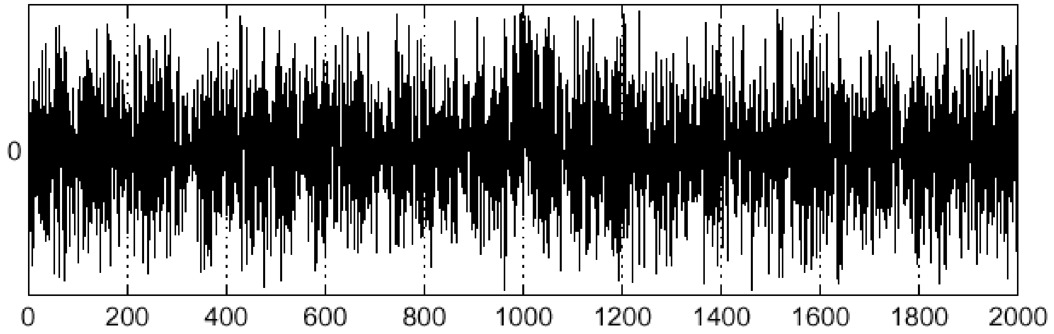
- Finite difference filters respond strongly to noise
  - Image noise results in pixels that look very different from their neighbors
    - That leads to spurious edges
    - The larger the noise, the larger gradient magnitude

$f(x)$



  - Where is the edge?

$\frac{d}{dx}f(x)$



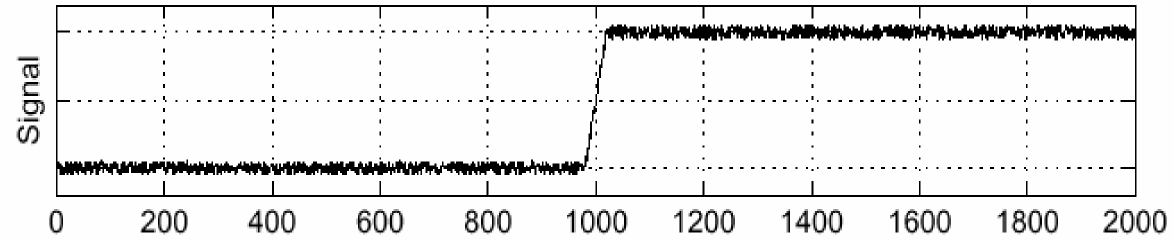
Noise Effect

- Smoothing reduce noise:
  - An image can be seen as a function with values that are both slowly varying and corrupted by random noise.
    - Can be useful to replace each pixel value(s) by some kind of local average of surrounding pixels.
    - Averaging can reduce the level of noise without (much) biasing the value obtained.
  - Gaussian is a true low-pass filter, so will not cause high frequency artifacts.

Noise Effect

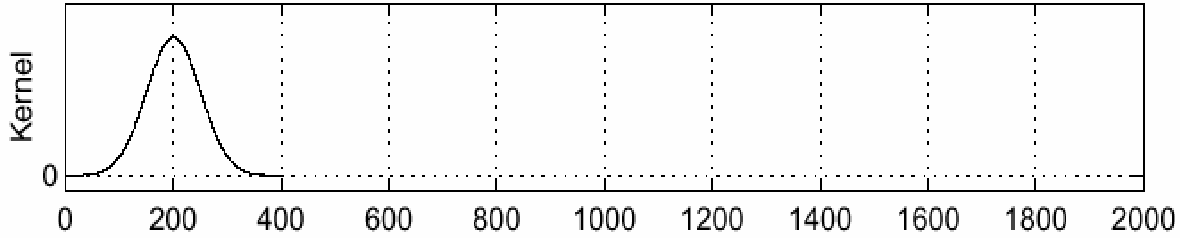
- Noisy signal:

f



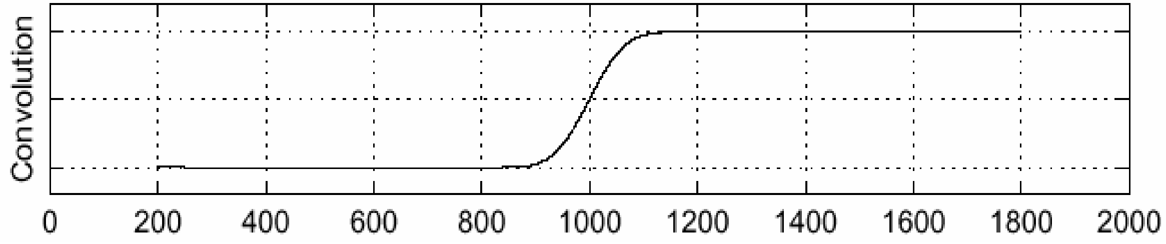
- Gaussian kernel:

g



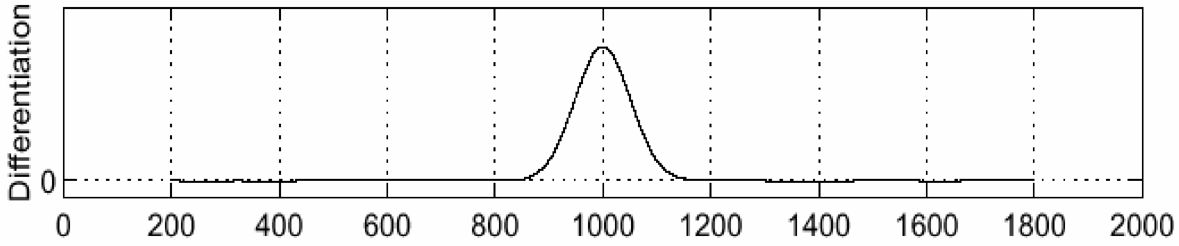
- Signal smoothing:

$f * g$



- Edges in ridges (or rifts):

$\frac{d}{dx}(f * g)$



Noise Effect

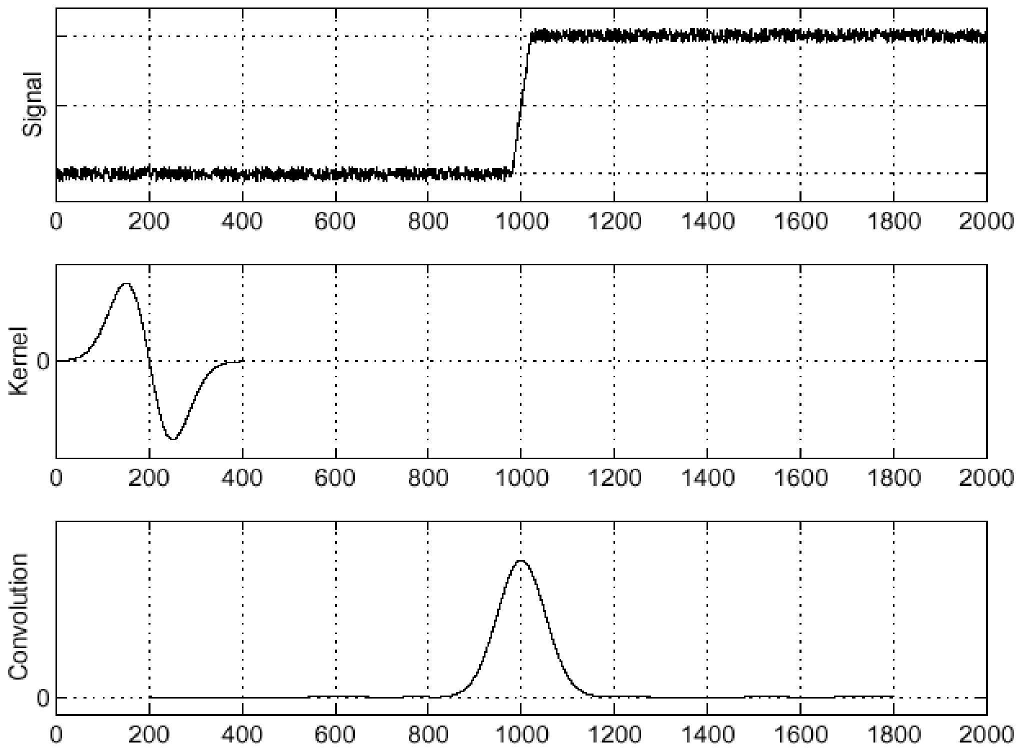
- Alternatively:

$$\frac{d}{dx}(f * g) = f * \frac{d}{dx} g$$

f

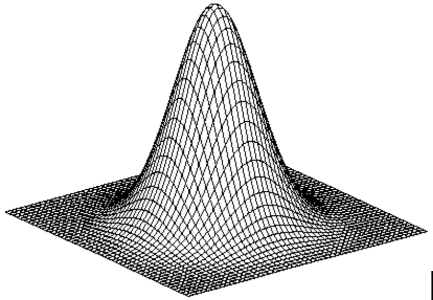
$\frac{d}{dx} g$

$f * \frac{d}{dx} g$



Derivative of 2D Gaussian

- 2D Gaussian:
- Convolution kernels:
- 2D Gaussian Derivatives



*

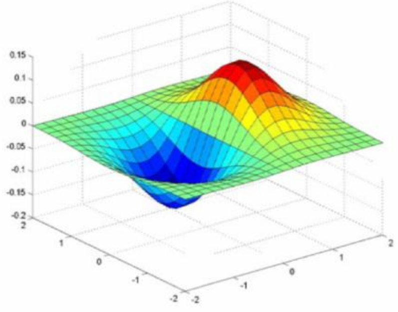
-1	0	1

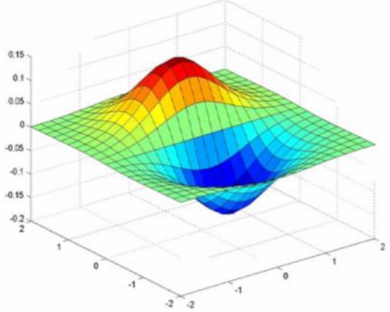
=

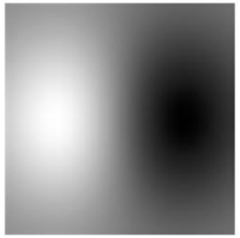
*

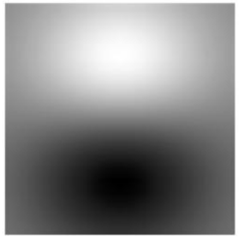
-1
0
1

=



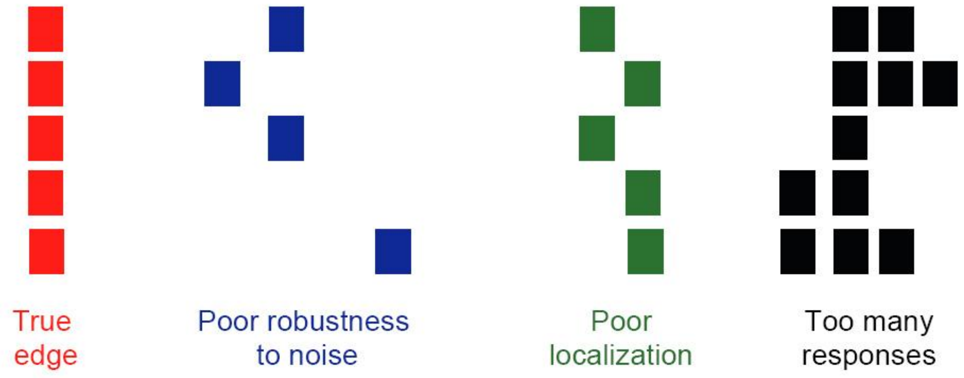






Edge Detector

- Criteria for a good edge detector:
  - **Good detection:** minimize the probability of false positives (spurious edges caused by noise), and false negatives (missing real edges).
  - **Good localization:** detect edges as close as possible to true edges.
  - **Single response:** return only one point for each true edge point.



Canny Edge Detector

- Canny detector most widely used edge detector
- Optimized to detect step edges
- Steps:
  1. Suppresses Noise (gaussian smoothing)
  2. Computes gradient magnitude and direction
  3. Applies Non-Maximum Suppression to get single response
  4. Uses hysteresis and connectivity analysis to complete edges

```
# Full Canny
edges = cv2.Canny(img, threshold1=50, threshold2=120, L2gradient=True)  # L2 uses sqrt

# Manual gradient (Sobel ~ DoG)
gx = cv2.Sobel(img, cv2.CV_32F, 1, 0, ksize=3)
gy = cv2.Sobel(img, cv2.CV_32F, 0, 1, ksize=3)
mag = cv2.magnitude(gx, gy); ang = cv2.phase(gx, gy, angleInDegrees=True)
```

Canny Edge Detector

- Suppress Noise: Gaussian smoothing

$$S = g * I$$

Input image

Smooth image

Gaussian filter

- Compute gradient magnitude and direction.

$$\nabla S = \nabla(g * I) = (\nabla g) * I)$$

It is more efficient to combine smoothing and gradient computation in a single operation

$$\nabla S = \begin{bmatrix} S_x \\ S_y \end{bmatrix} = \begin{bmatrix} \frac{\partial g}{\partial x} \\ \frac{\partial g}{\partial y} \end{bmatrix} * I = \begin{bmatrix} g_x \\ g_y \end{bmatrix} * I = \begin{bmatrix} g_x * I \\ g_y * I \end{bmatrix}$$

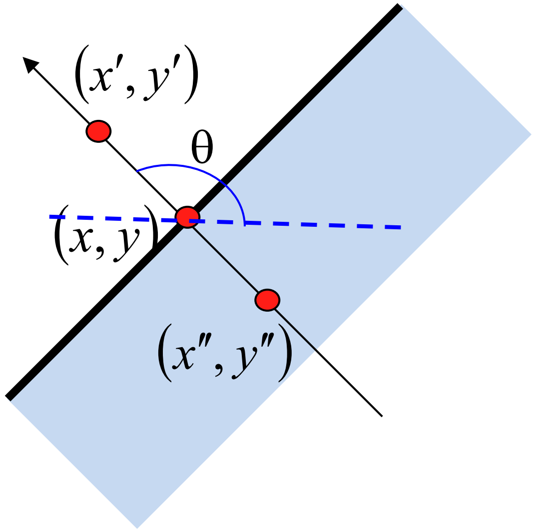
$$Magnitude = |\nabla S| = \sqrt{S_x^2 + S_y^2}$$

$$Direction = \theta = \tan^{-1} \frac{S_y}{S_x}$$

First derivative of the Gaussian optimizes trade-off between detection *(minimum noise effect)* and localization

Canny Edge Detector

- Non-Maximum Suppression to get single response
  - Edge occurs where gradient reaches a maximum
  - Suppress non-maxima gradient even if it passes threshold

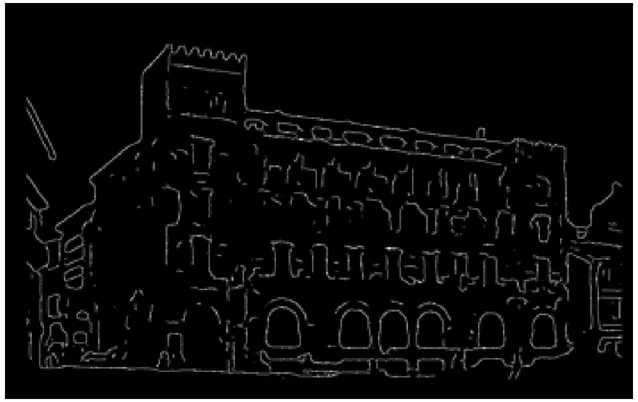


$$M(x,y)=\begin{cases} |\nabla S|(x,y) & \begin{array}{l} if\ |\nabla S(x,y) > Th| \\ \&\ |\nabla S|(x,y) > |\nabla S|(x',y') \\ \&\ |\nabla S|(x,y) > |\nabla S|(x'',y'') \end{array} \\ 0 & \text{otherwise} \end{cases}$$

(x',y') and (x'',y'') are the neighbors of (x,y) in $|\nabla S|$ along the direction normal to an edge (θ)

Canny Edge Detector

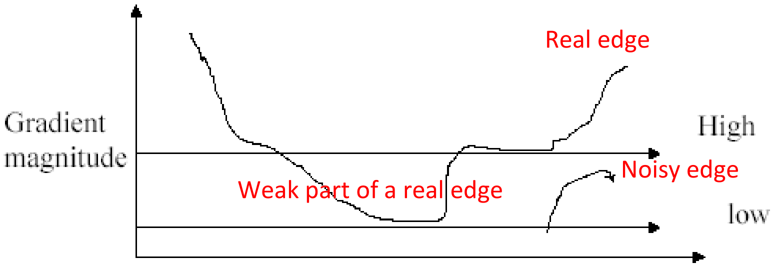
- Edge occurs where gradient reaches a local maximum
  - Suppress potential noisy edges (gradient below a threshold)
  - Suppress non-maxima gradient even if it passes threshold

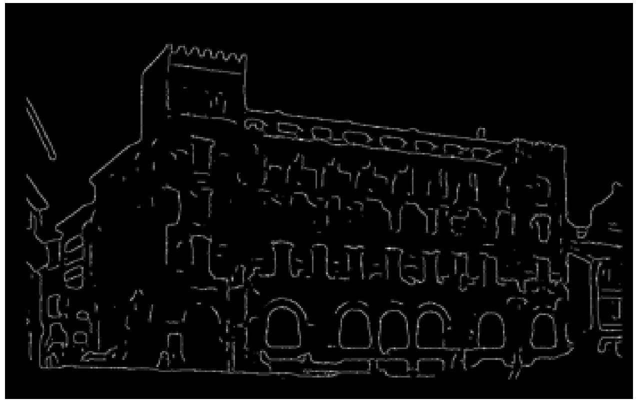


Edge map after non-maxima suppression

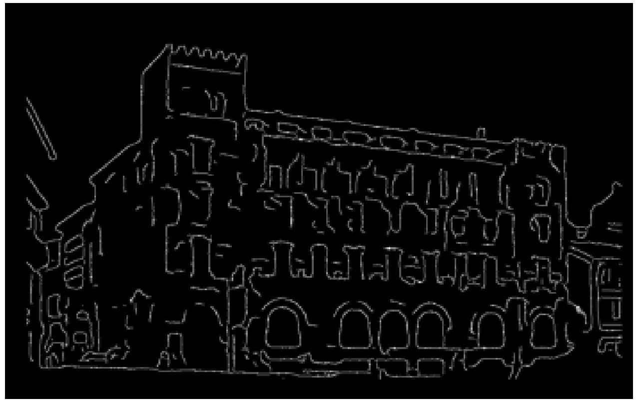
Canny Edge Detector

- Use hysteresis and connectivity analysis to complete edges
  - Thresholding gradient magnitude to discard noisy edges has its cons:
    - Some real edge parts could be erroneously removed





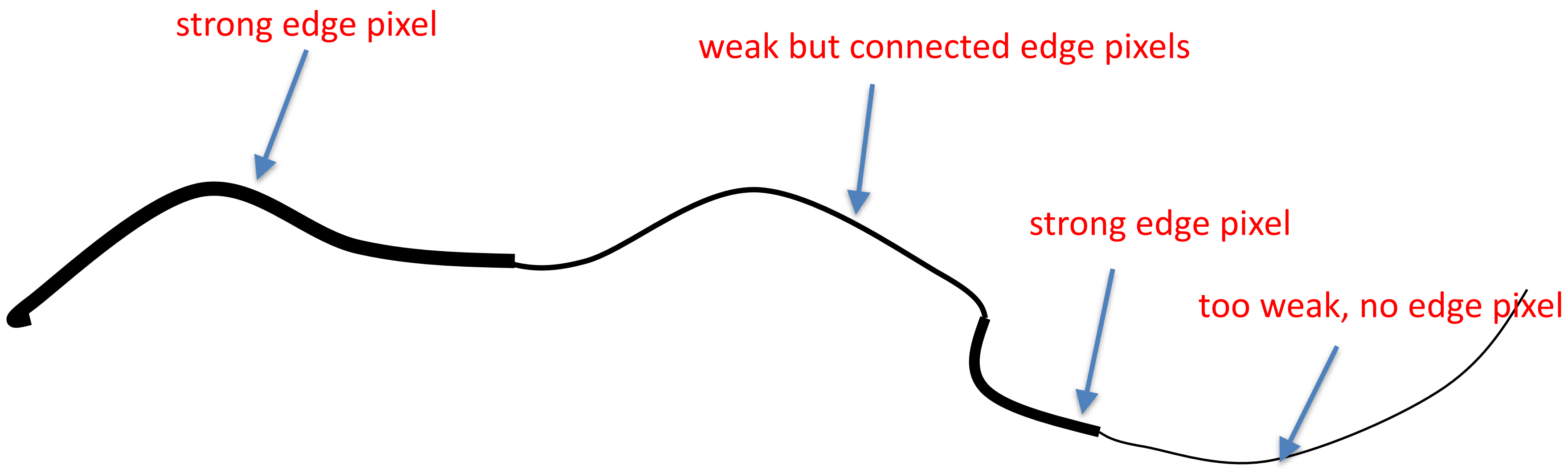
Edge map after non-maxima suppression



Edge map after hysteresis and connectivity analysis

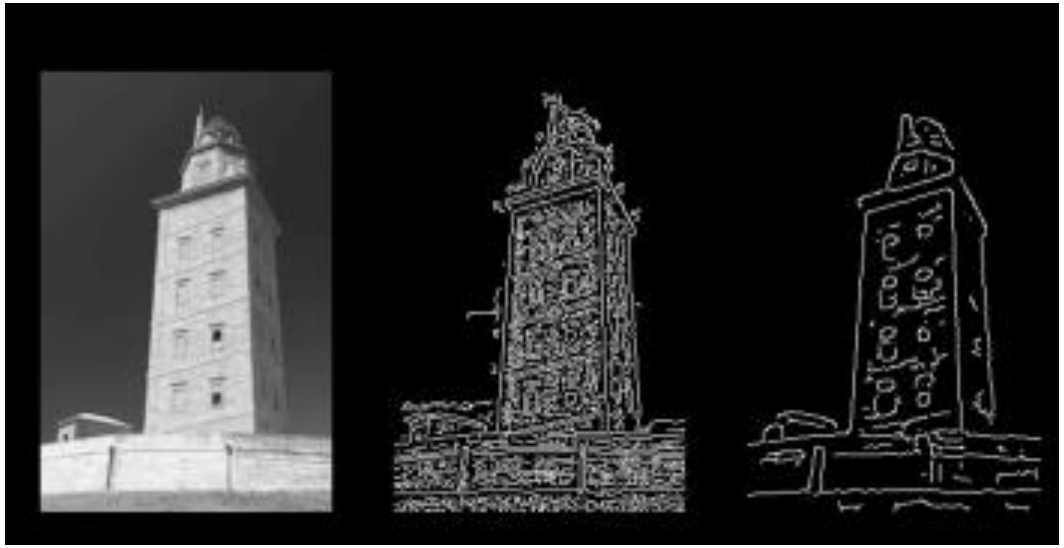
Canny Edge Detector

- Thresholding and linking (hysteresis):
  - Define two thresholds: Low and High
  - Use the High threshold to start edge curves and the Low to continue them
  - Points withe less than Low gradient are not edges



Canny Edge Detector

- The choice of σ depends on desired behavior
  - large σ detects large scale edges
  - small σ detects fine details



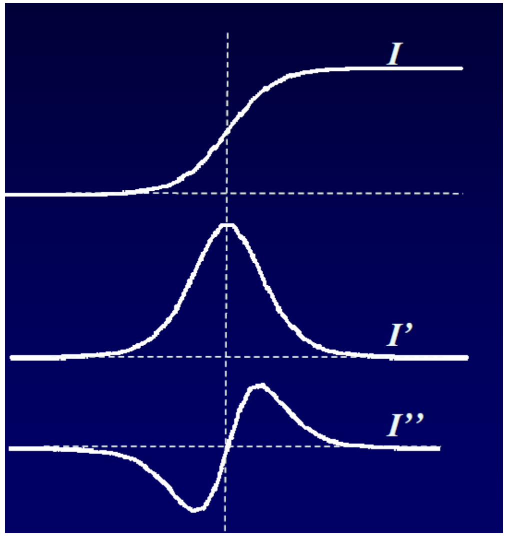
source small sigma large sigma

2nd Derivative Edge Detectors

- 1D Second derivative:
  - Finite differences approximation:

$$\frac{\partial^2 I}{\partial x^2} = \frac{I(x_{i+1}, y) + I\big((x_{i-1}, y)\big) - 2I((x_i, y))}{\Delta x^2}$$

$I''(x)=B2*I(x),$ $B2=[1\ -2\ 1]$



  - Kernel $B2$:

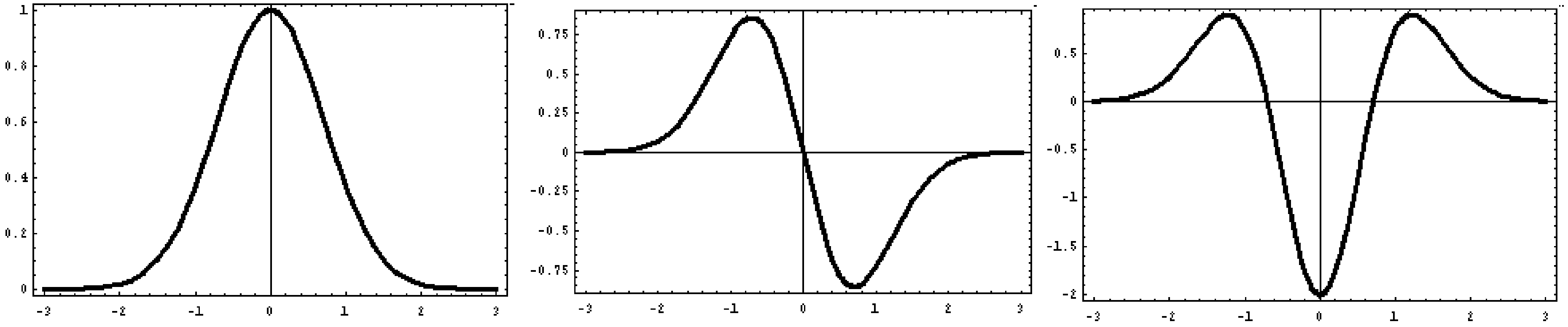
1	-2	1

  - Extreme points of the 1st derivative are zeros of the 2nd derivative
  - Edges points at zero crossings of the 2nd derivative

2nd Derivative Edge Detectors

Noise mitigation

- To mitigate effect of noise:
  - Convolution with a Gaussian kernel + 2nd derivative
  - Equivalently: Convolution with the 2nd derivative of a Gaussian kernel



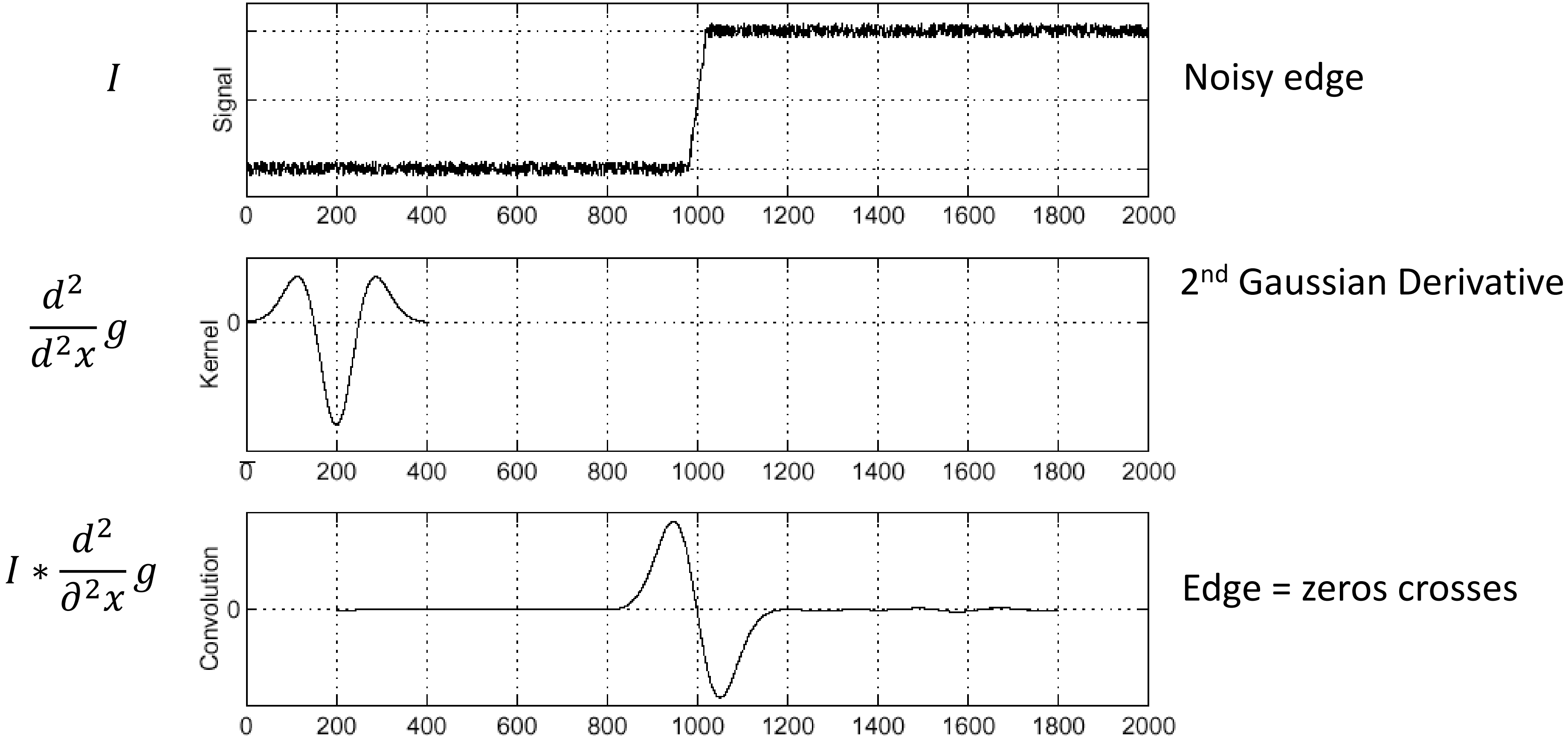
$g(x) \sim e^{-x^2/\sigma^2}$ | $dg(x)/dx$ | $d^2g(x)/dx^2$

Gaussian | 1st derivative of Gaussian | 2nd derivative of Gaussian

2nd Derivative Filters

Noise mitigation: Gaussian + second derivative

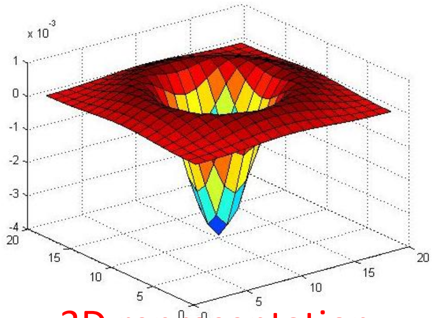
- 2nd Derivative Filters



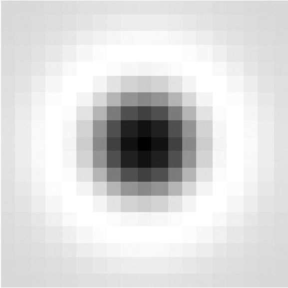
- 2D 2nd derivative (Laplacian):

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

- Smothed 2D 2nd derivative: Laplacian of Gaussian



3D representation



2D representation

$$LoG(I) = I * \frac{\partial^2 g}{\partial^2 x} + I * \frac{\partial^2 g}{\partial^2 y}$$

- ▷ It is not directional
  - Contrast between central pixels and sorroundings
  - Does not bring information about edge orientation
  - This makes the finding of zero crosses more difficult

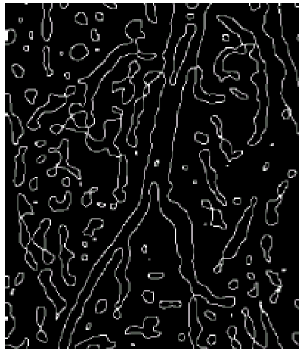
Marr-Hildreth Edge Detector

- Marr and Hildreth proposed a Gaussian Filter combined with the Laplacian (LoG) for edge detection.
  - Gaussian smoothing to remove high frequency noise.
  - Laplacian edge enhancement.
  - Detection criteria:
    - zero crossing in the 2nd derivative
      - Threshold at zero of LoG(I) and contouring of binary image.
    - Only edge pixels with first derivatives above a threshold are kept.

I

LoG(I)

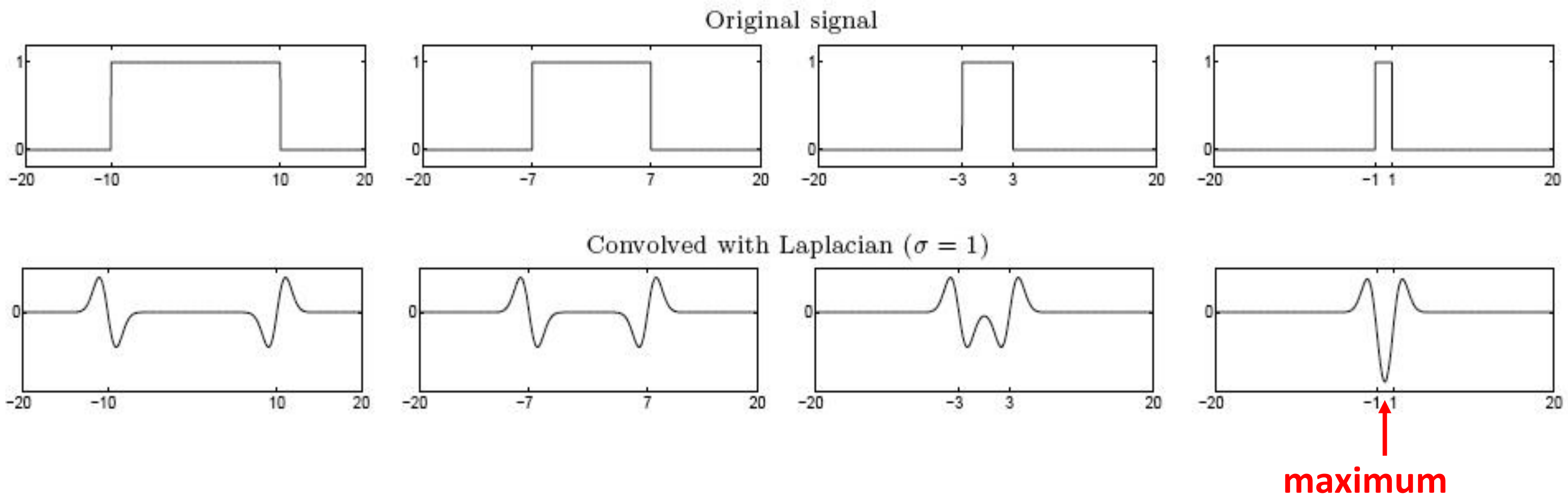
Thresholded LoG(I)



Edge map

2nd Derivative Blob Detector

- Response of an edge to the 2nd gaussian derivative filter: a wave
- Response to two close edges: superposition of two waves
- **Blob**: given a specific filter scale, the magnitude of the convoluted signal (superposed waves) reaches the maximum at a blob center of the same scale.



Hough Transform

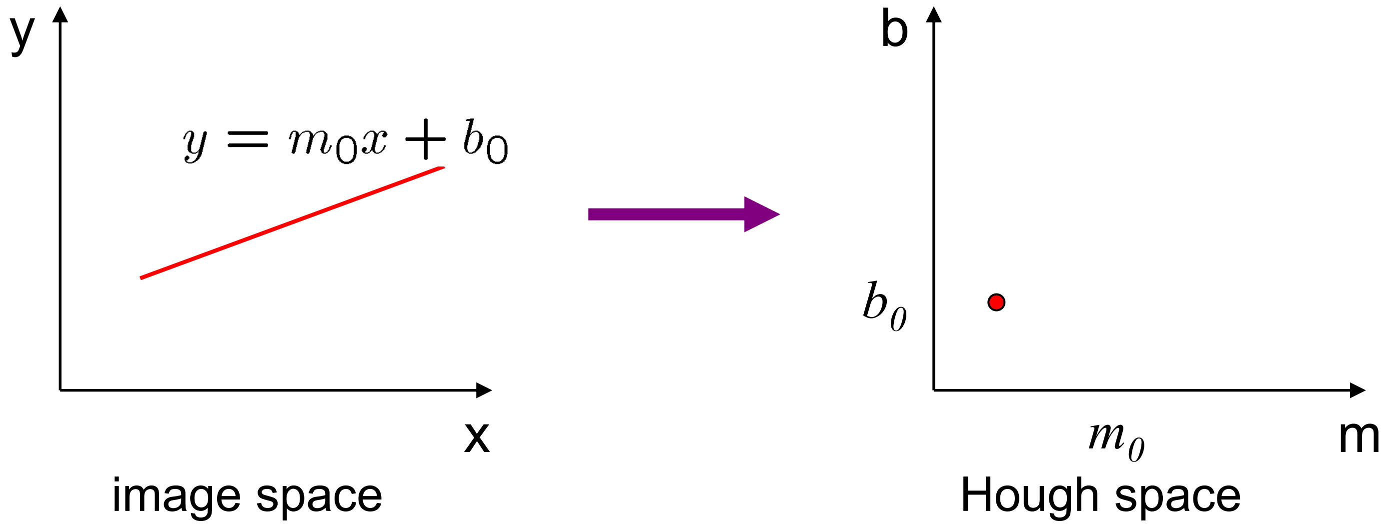
- So far, we haven’t found edge segments, only edge points
  - Usually, pixels that partially describe objects boundaries.

- How can we find and describe more complex features?
  - The Hough transform is a common approach to find parameterised lines, circles and other structures ONLY if their parametric equation is known.
  - Hough transform gives robust detection under noise and partial occlusion

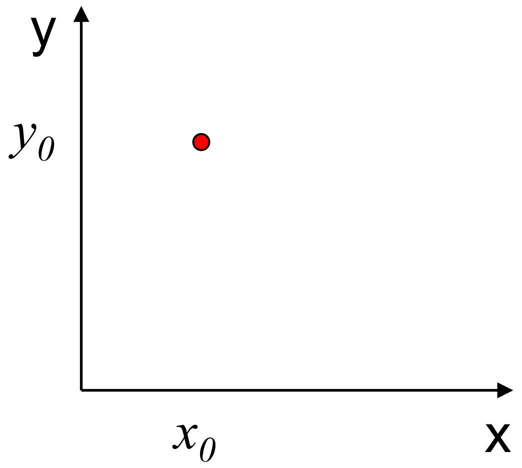
Hough Transform

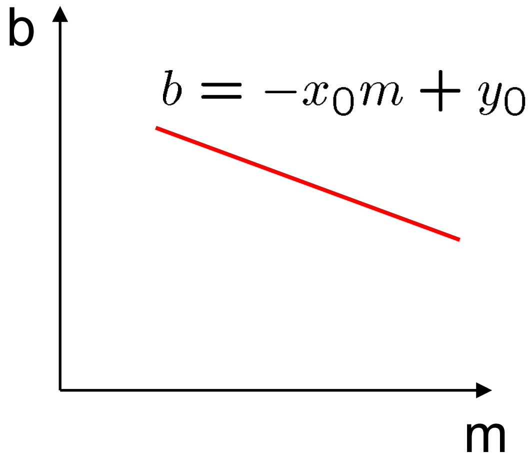
- Assume that we have performed some edge detection processing.
- We will see how Hough (H) Transform (T) can be used to find the location of contours with simple parameterized shapes.
  - lines
  - circles
  - …

- Consider the equation of a 2D line: $y = mx + b$
  - (x,y): pixel coordinates in 2D image space
  - (m,b): line parameters in 2D space Hough space
- Connection between image (x,y) and Hough (m,b) spaces
  - A line in image space corresponds to a point in Hough space
    - All points on a line in (x,y) space map to a point in (m,b) space:



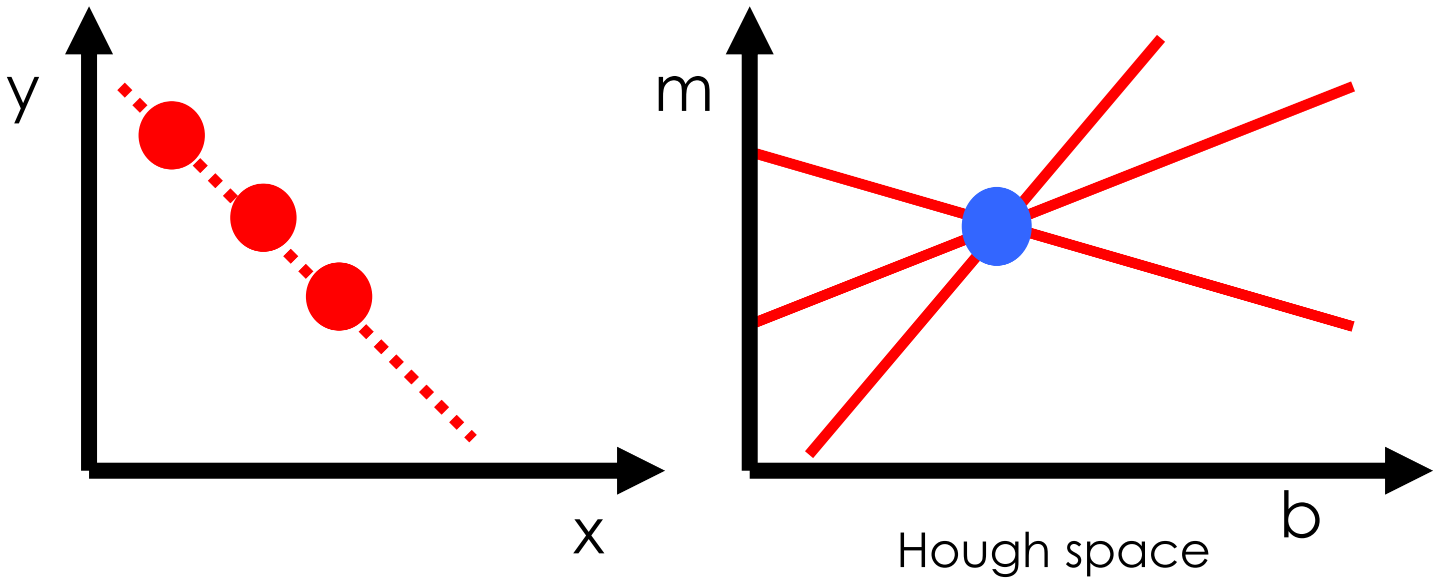
- Connection between image ***(x,y)*** and Hough ***(m,b)*** spaces
  - Where does a point (x_0, y_0) in the image space map to?
    - To the solutions of $b = -x_0 m + y_0$
    - To all the points in a line in Hough space



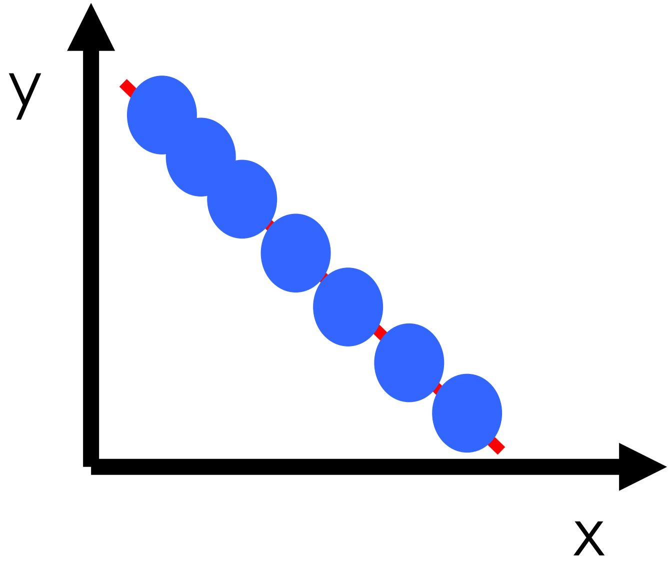


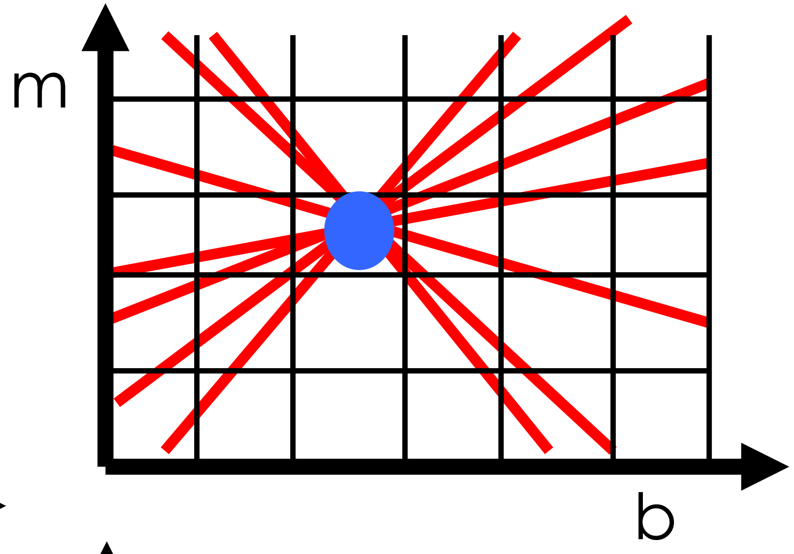
- Two points (x_1, y_1) and$(x_2\ y_2)$ define a line in the (x, y) plane, and give rise to two different lines in (m,b) space. In (m,b) space these lines intersect in a point (m', b').
- N points on the same line in (x,y) plane give place to N intersections of lines en (m,b) space

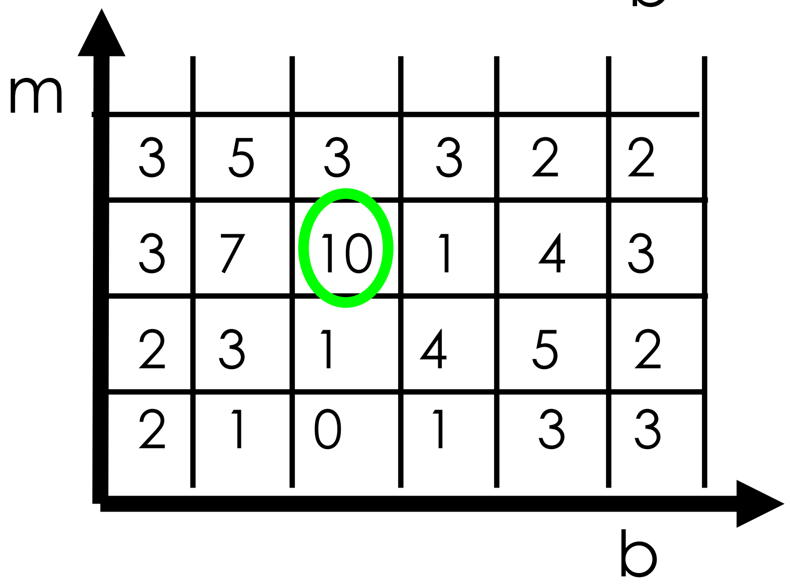
Each point in (x,y) gives a line in (m,b) plane



- Hough Transform







Hough Transform

Line Detection

- Usually, HT use a different parameterization: $d = x\cos\theta + y\sin\theta$
  - d is the perpendicular distance from the line to the origin
  - θ is the angle this perpendicular makes with the x axis

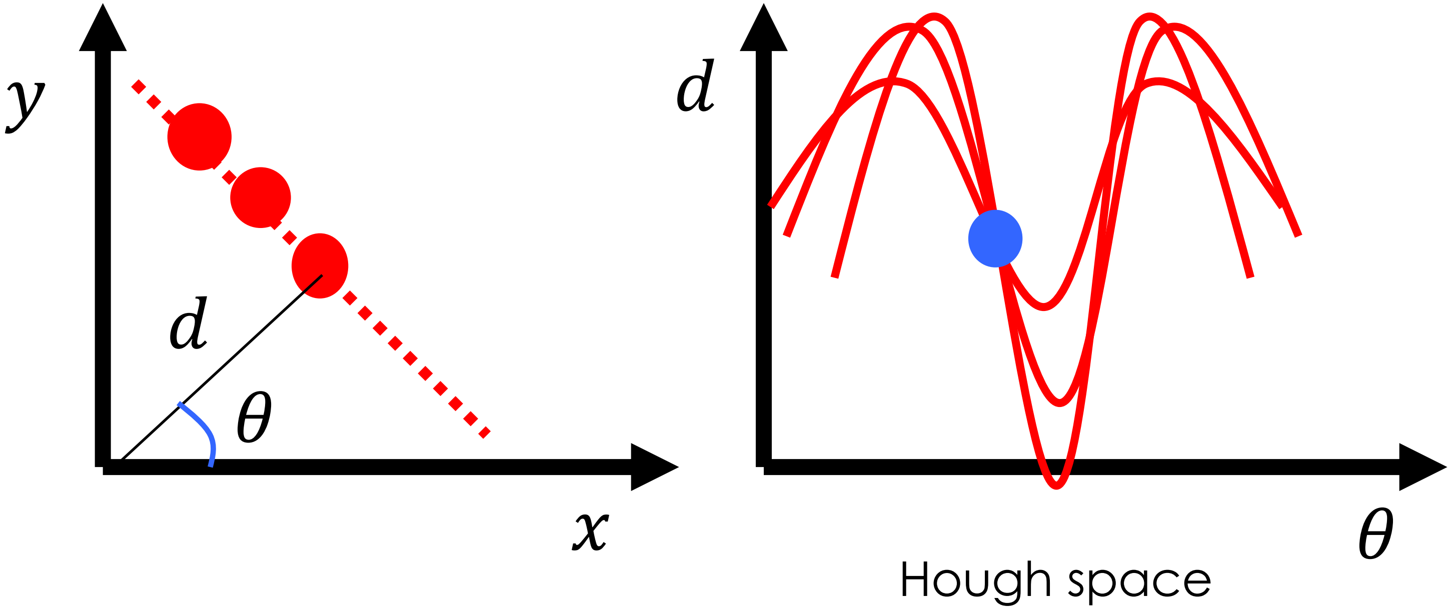
- Basic Hough transform algorithm
  1. Initialize $\mathrm{H}[d, \theta]=0$
  2. for each edge point $\mathrm{I}[x,y]$ in the image

     Estimate θ based on its gradient *(discarding other orientations)*

     Compute $d = x\cos\theta + y\sin\theta$

     $\mathrm{H}[d, \theta] \mathrel{+}= 1$ *(usually give more votes to stronger gradients)*
  3. Find the value(s) of (d, θ) where $\mathrm{H}[d, \theta]$ is maximum
  4. The detected line in the image is given by $d = x\cos\theta + y\sin\theta$

- Issue: parameter space is unbounded…

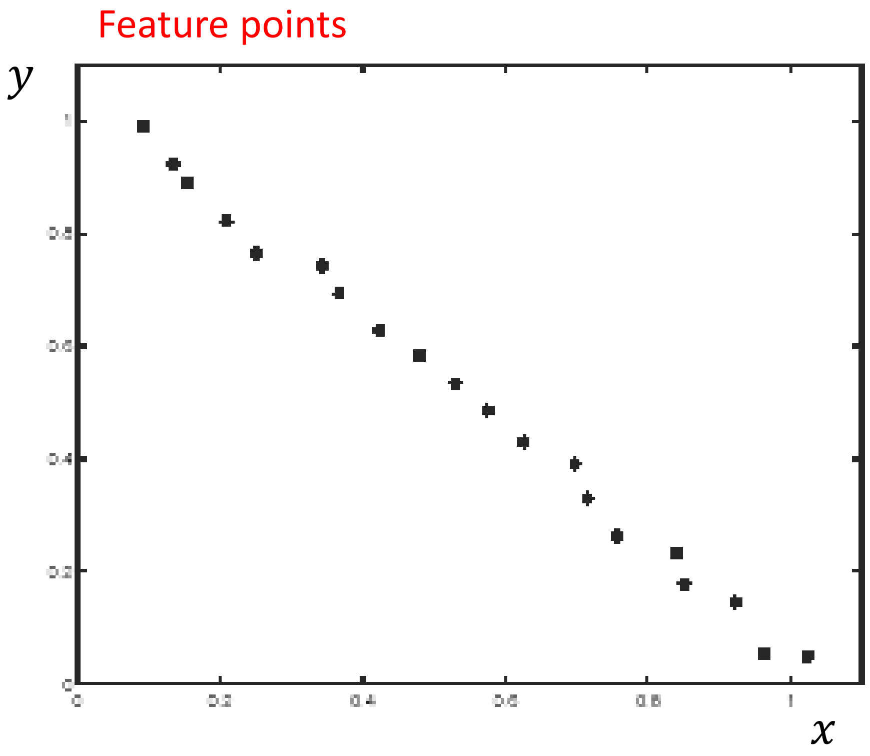


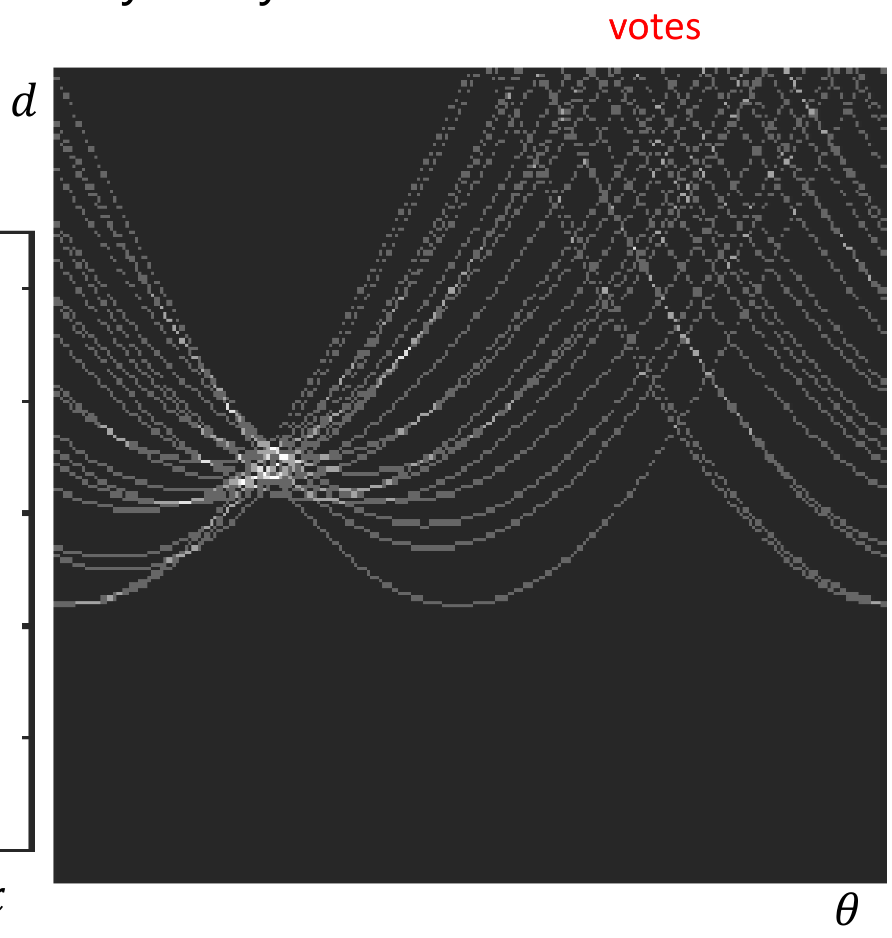
$$d = x\cos\theta + y\sin\theta$$

Hough Transform

Line Detection

- Issue: parameter space is unbounded and data is usually noisy

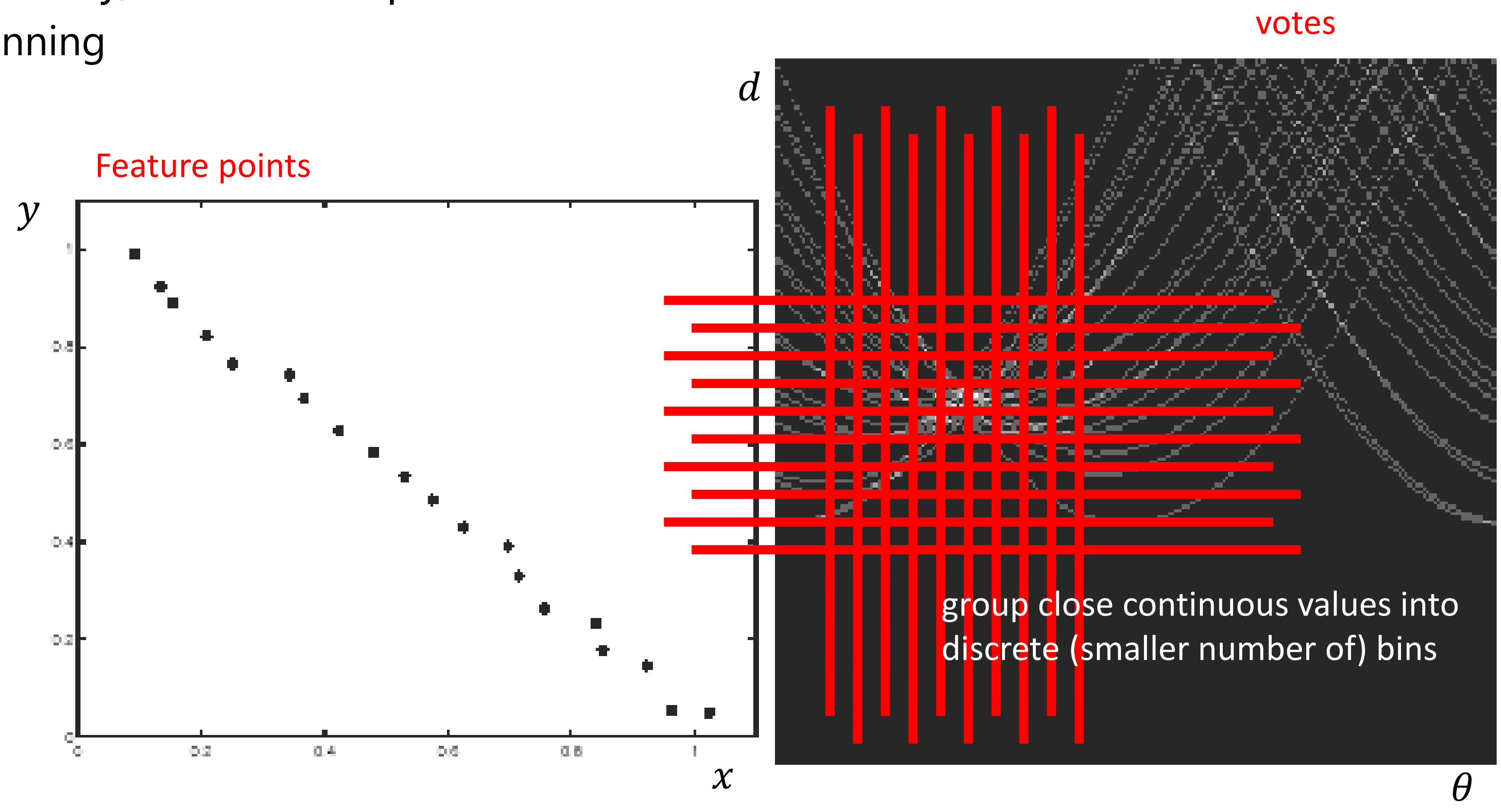




Hough Transform

Line Detection

- Data is usually noisy, there is not perfect intersection
  - Solution: binning



- Top (20) voted lines