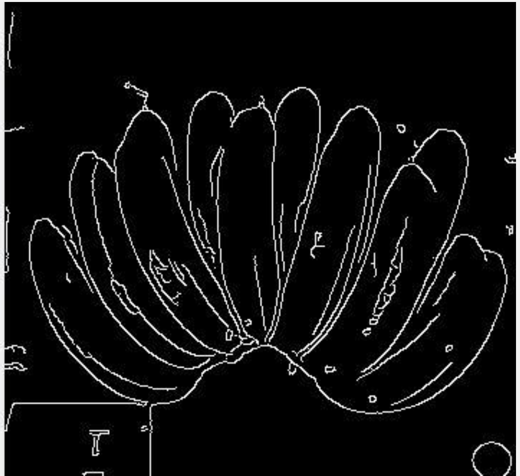
Department of Computer Science University of Bristol

COMS30030 - Image Processing and Computer Vision



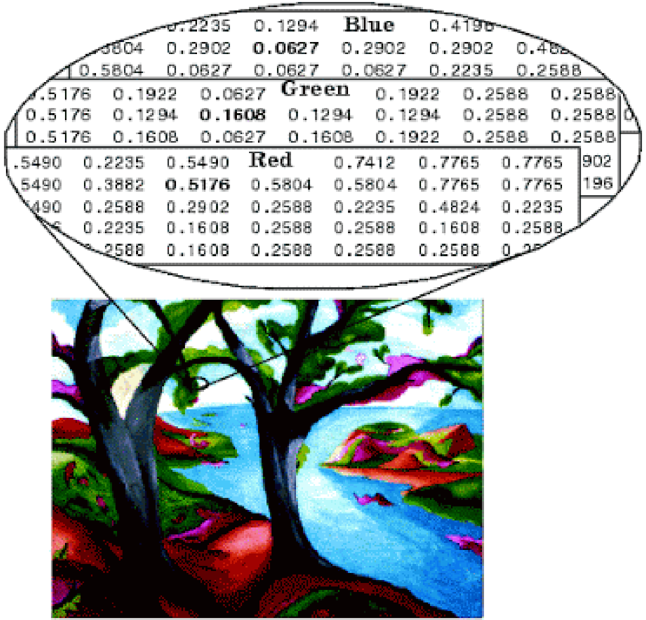
Lecture 04

Edge & Shape Detection

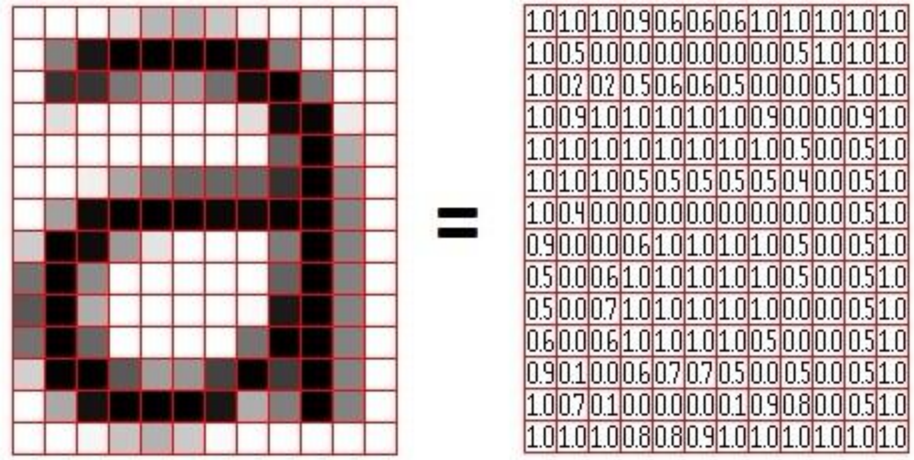
Majid Mirmehdi | majid@cs.bris.ac.uk

Beyond the Matrix

- Images are matrices of numbers
- Shapes are more informative than individual pixel values



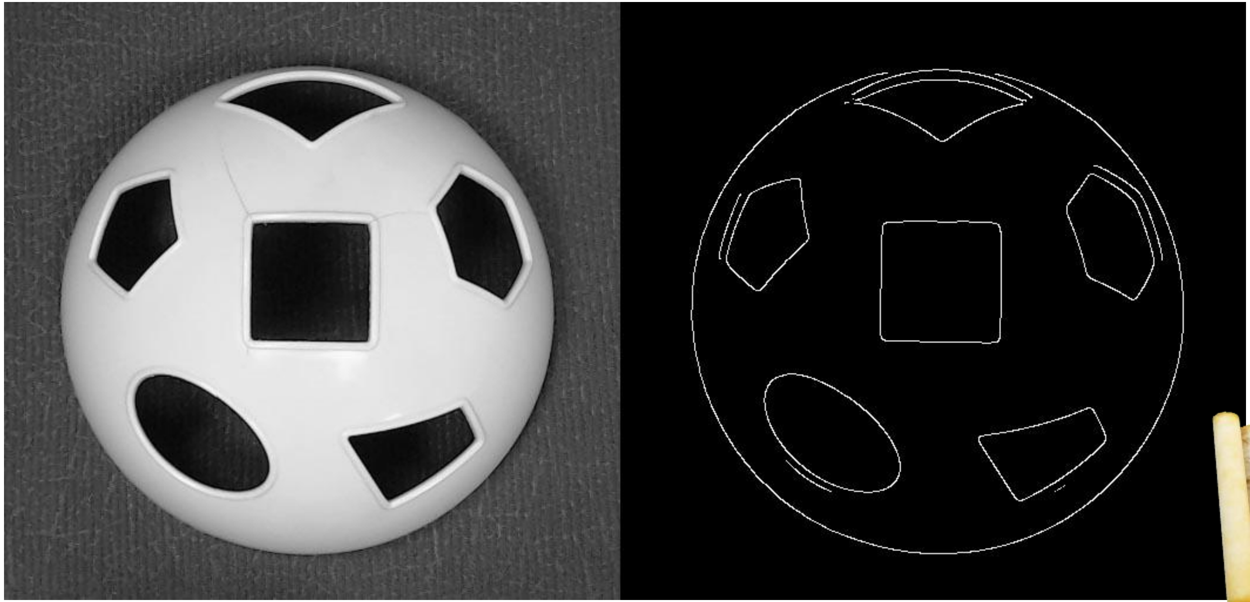
Source: http://www.ece.northwestern.edu/local-apps/matlabhelp/toolbox/images/intro8.html



Source: https://www.datasciencecentral.com/profiles/blogs/image-classification-with-hsv-color-model-processing

What are edges?

- Edges highlight the contour of shapes
- They can be used to identify objects

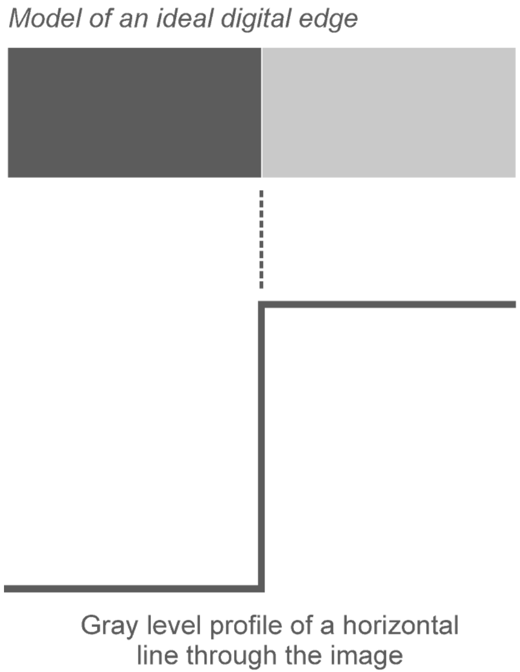


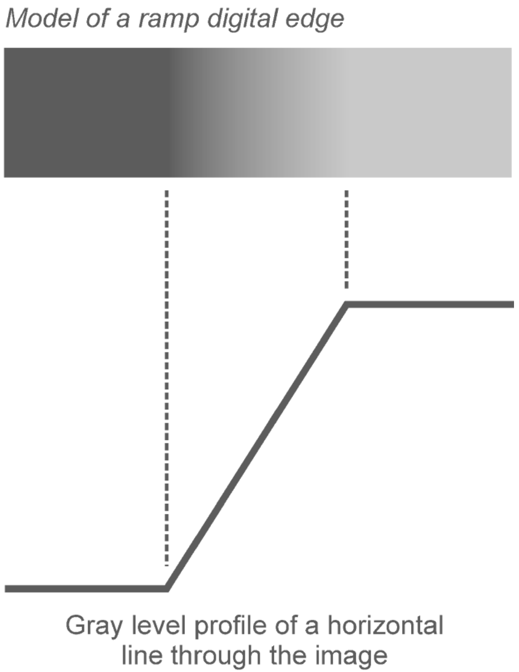
Source: http://www.walrusvision.com/wordpress/introduction-to-edge-detection/

Edge map

What are edges?

- Edges are those places in an image that correspond to object boundaries.
- Edges are pixels where image brightness changes (relatively) abruptly.





Why detect Edges?

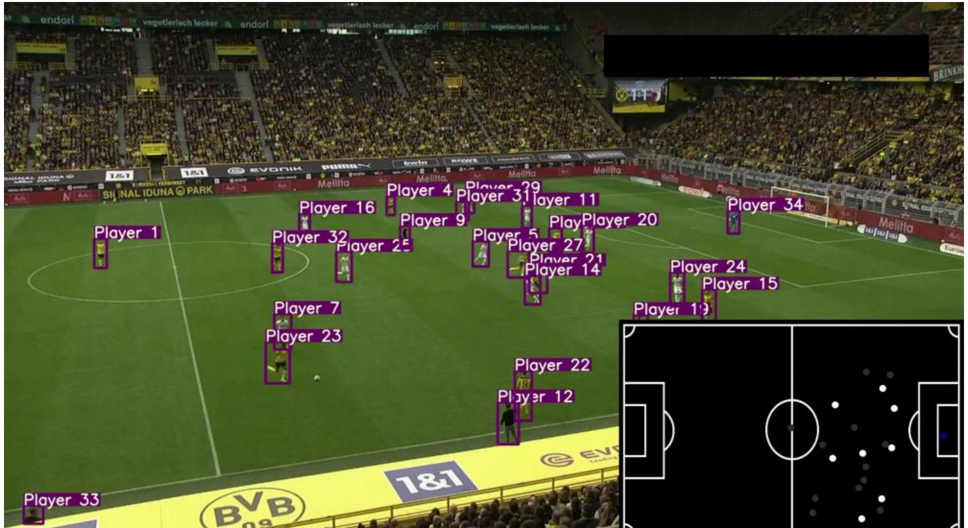
- **Edges**: Sharp changes of image brightness
- **Sources**: Object boundaries, patterns, shadows, etc.
- For segmentation: finding object boundaries

Why detect Edges?

- **Edges**: Sharp changes of image brightness
- **Sources**: Object boundaries, patterns, shadows, etc.
- For segmentation: finding object boundaries
- For recognition: extracting patterns

Why detect Edges?

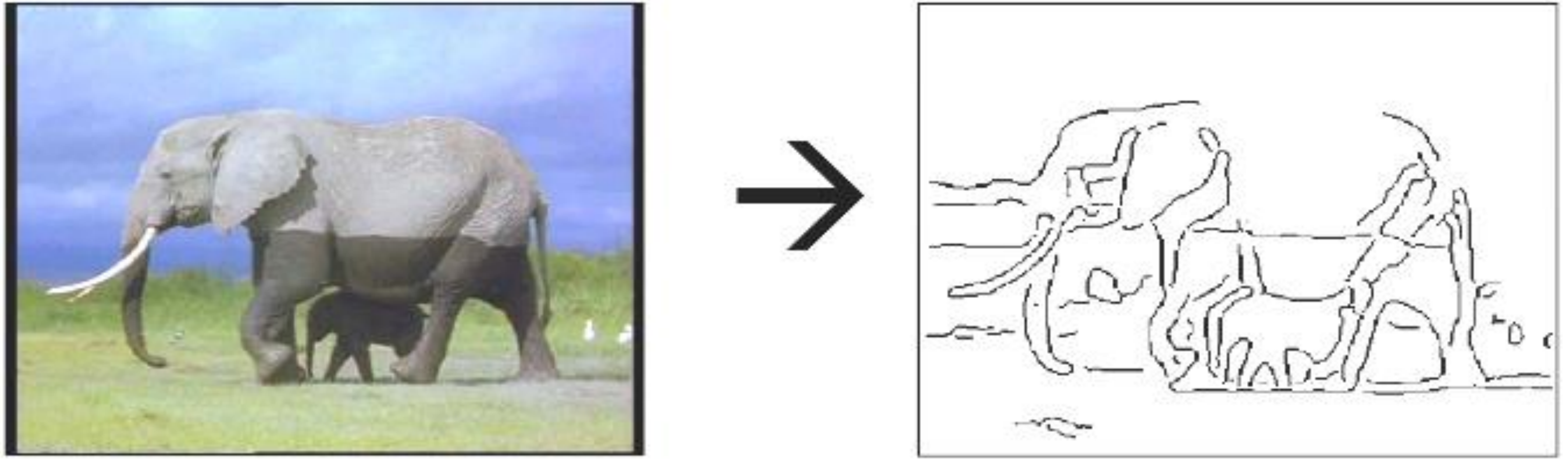
- **Edges**: Sharp changes of image brightness
- **Sources**: Object boundaries, patterns, shadows, etc.
- For segmentation: finding object boundaries
- For recognition: extracting patterns
- For motion analysis: reliable tracking regions



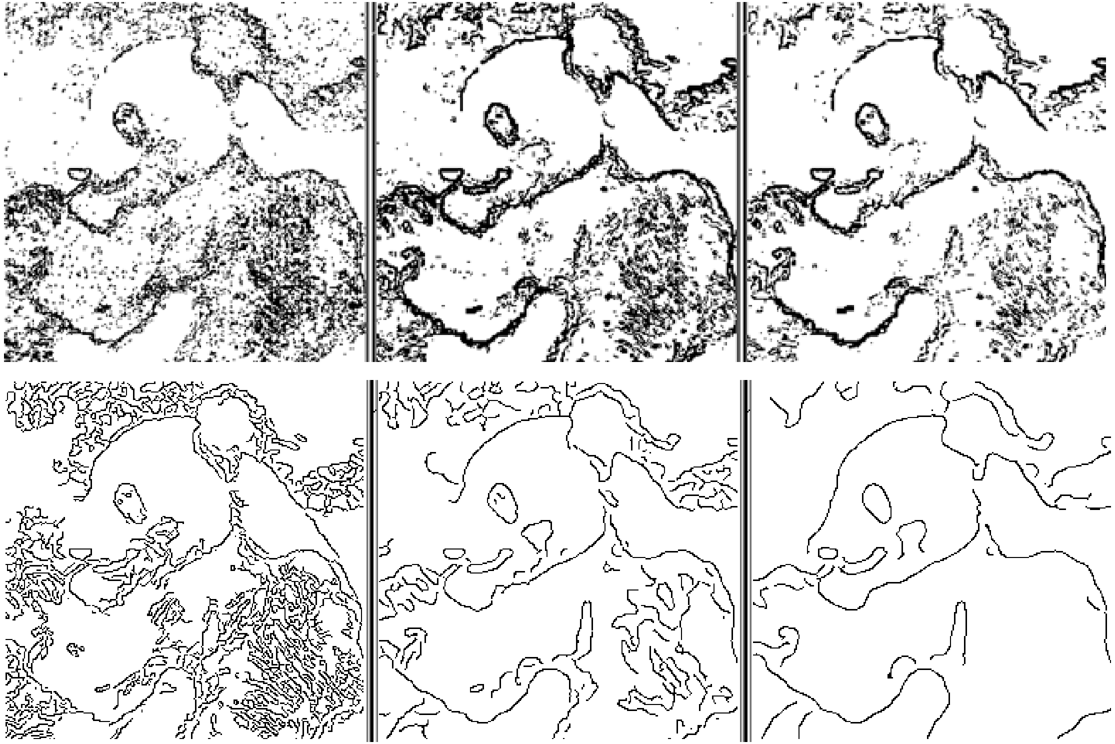
https://medium.com/latinxinai/into-football-analytics-how-to-track-football-players-using-yolov8-2d64ed31199c

Meaningful edges vs. nuisance edges

- **Edges**: Sharp changes of image brightness



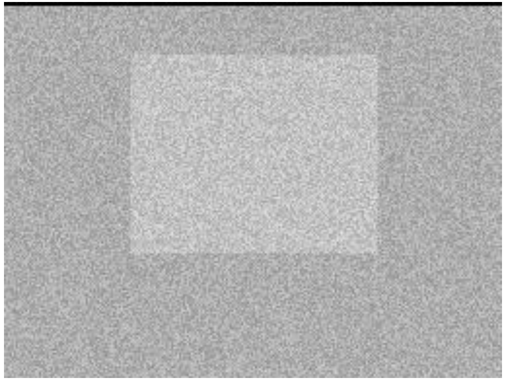
Difficult edges

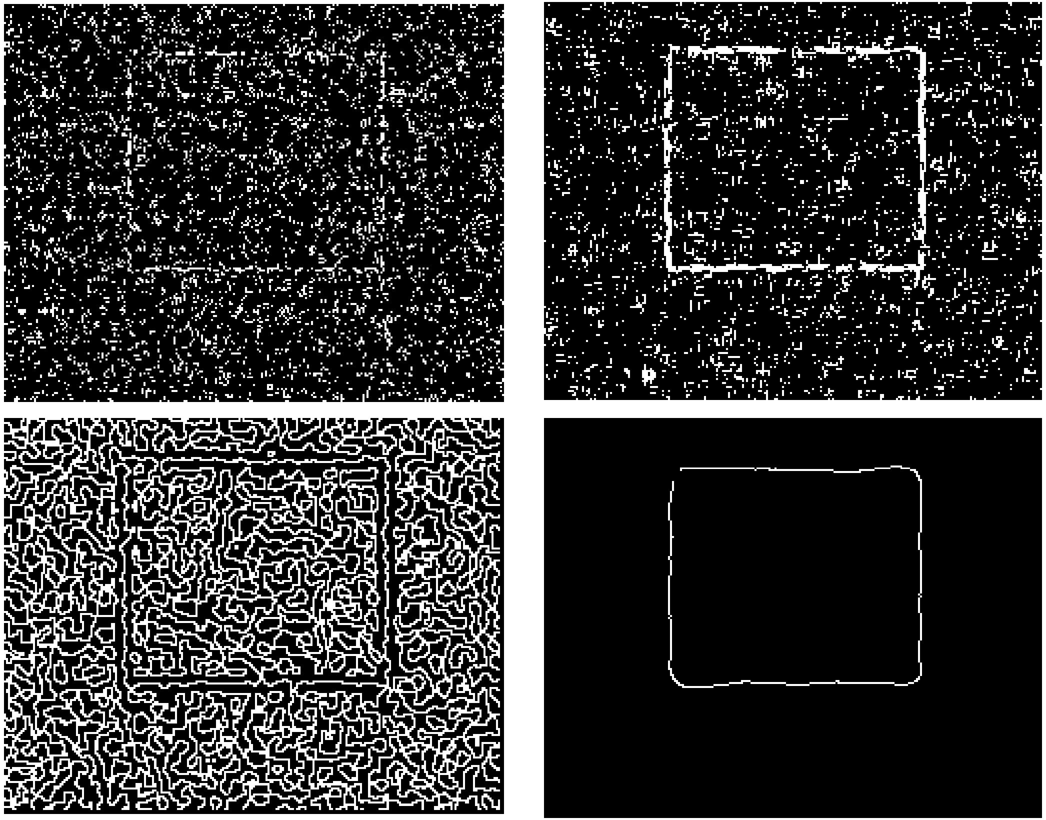


http://fourier.eng.hmc.edu/e161/lectures/canny/node1.html

Very difficult edges

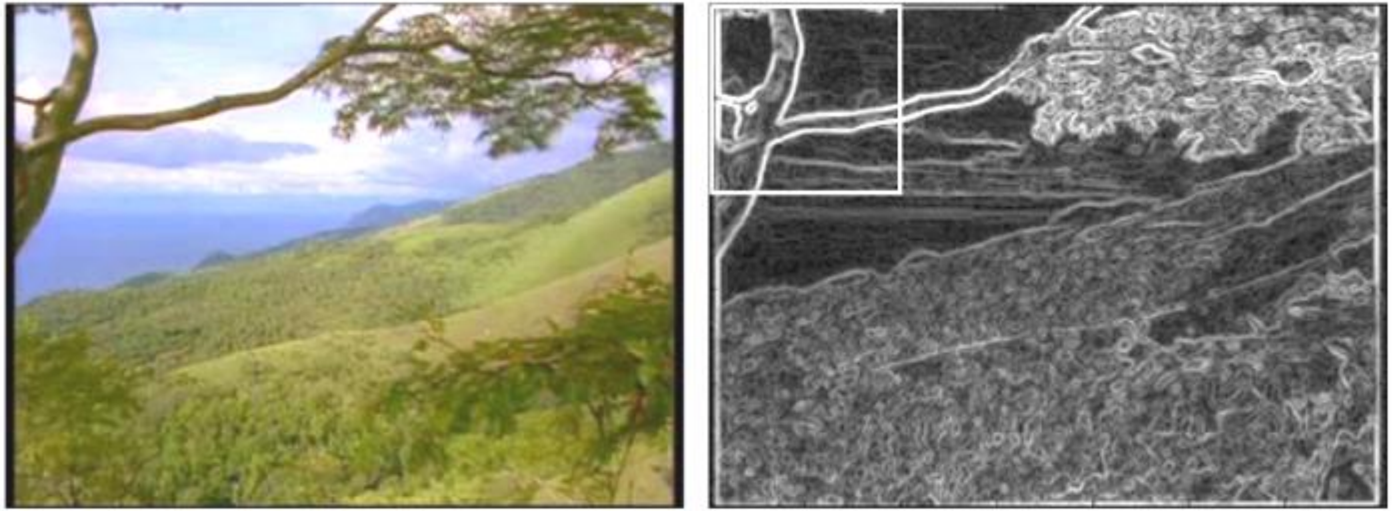
Different techniques work for different cases.





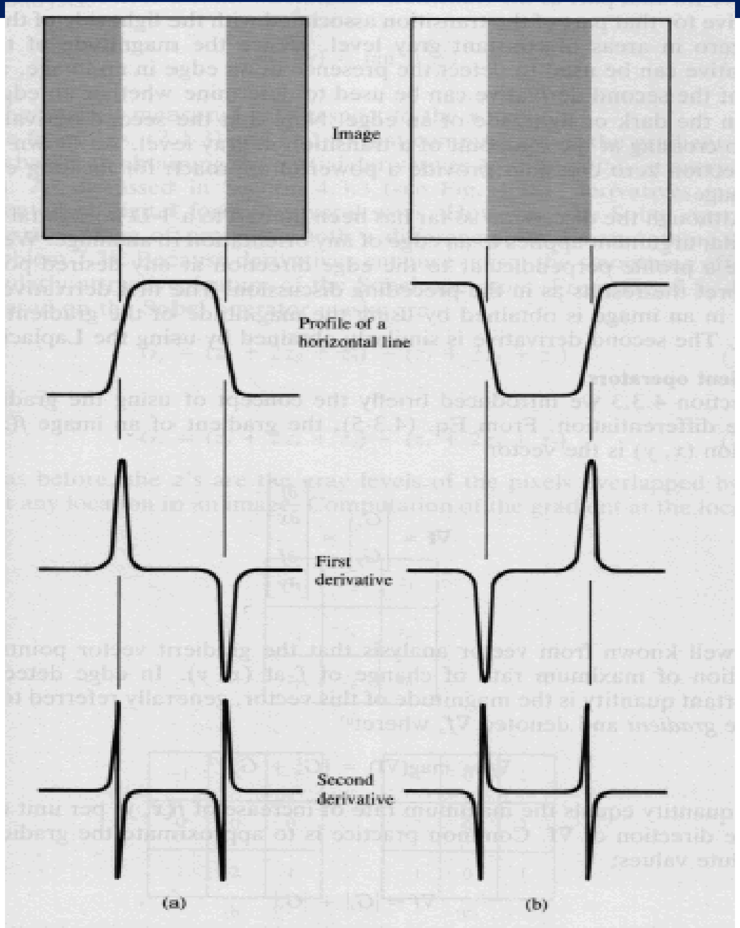
Edge Detection Strategy

- **Recognition Strategy:** determine a `**measure of change**' in a pixel's neighbourhood
- First derivatives in 2D space → image gradient



Edge Detection Strategy

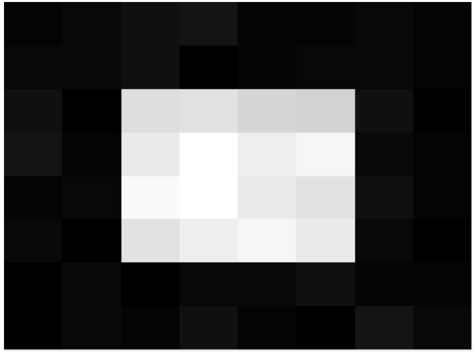
- Magnitude of the 1st derivative can be used to detect the presence of an edge
- Zero-crossing of the 2nd derivative at the midpoint of a transition in gray level, which provides a powerful approach for locating the edge
- The sign of the 2nd derivative usually used to determine whether an edge pixel lies on the dark or light side of an edge

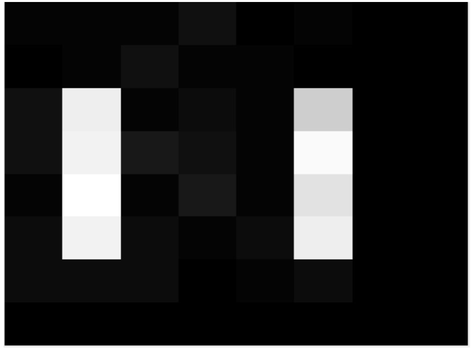


Edge Detection Strategy

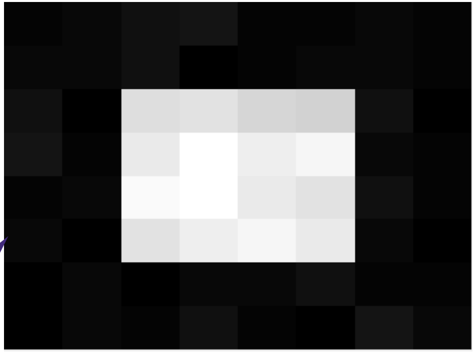
Horizontal differencing detects vertical edges

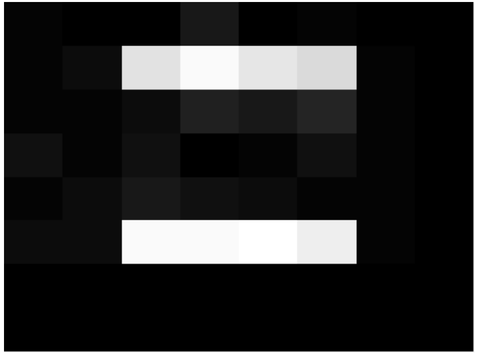
Difference





Difference



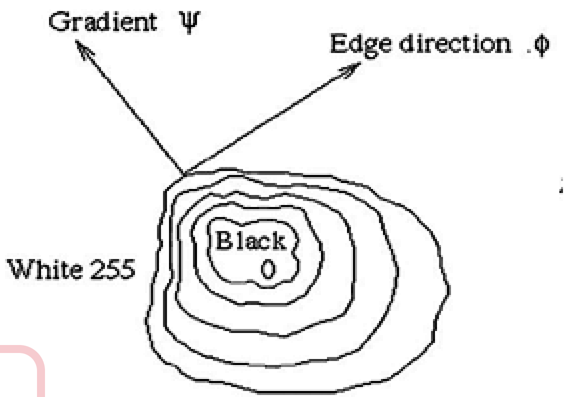


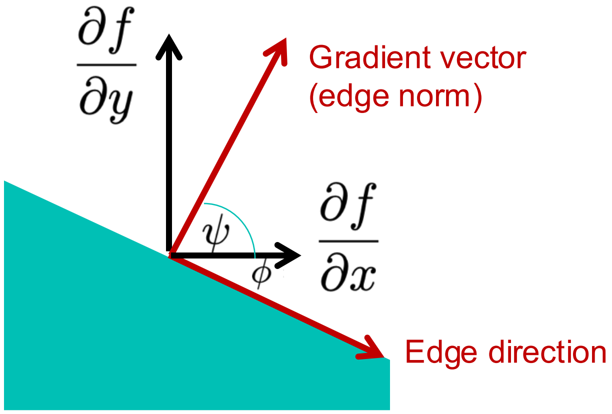
Vertical differencing detects Horizontal edges

The Image Gradient

A **vector** variable

- Direction ψ of the maximum growth of the function
- Magnitude $|\nabla f(x,y)|$ of the growth
- Perpendicular to the edge direction ϕ





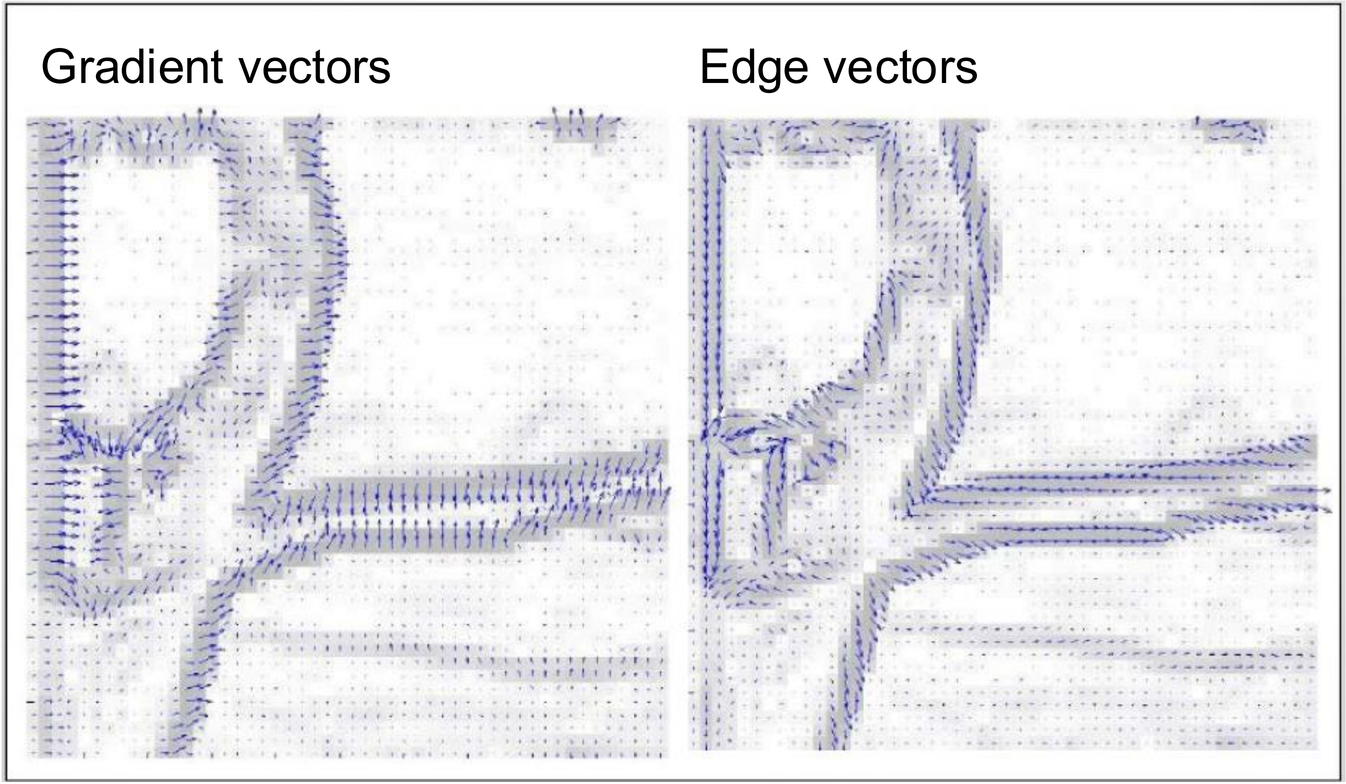
$$\nabla f(x,y) = \frac{\partial f}{\partial x}\hat{x} + \frac{\partial f}{\partial y}\hat{y}$$

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

$$\psi = \arctan\left(\frac{\partial f/\partial y}{\partial f/\partial x}\right)$$

$$\phi = \psi - \frac{\pi}{2}$$

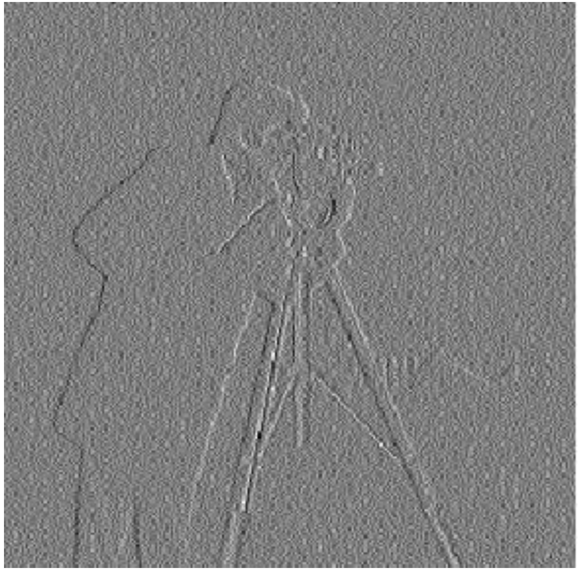
Example: Gradient & Edge Vectors



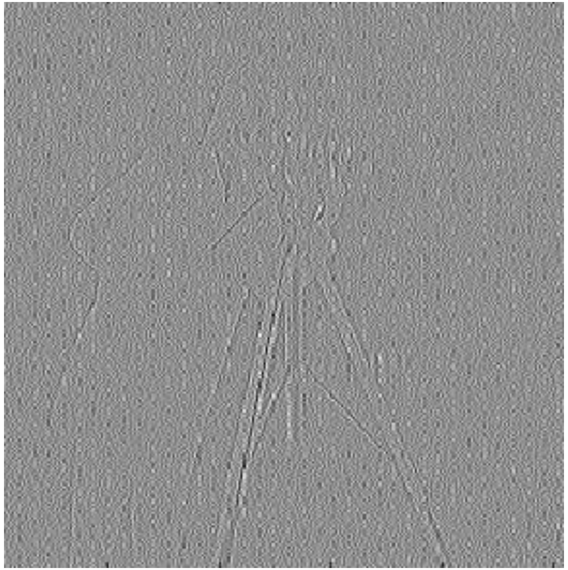
Simple derivatives too noisy

- Straight differencing of pixels is sensitive to noise

Noisy cameraman



First derivative

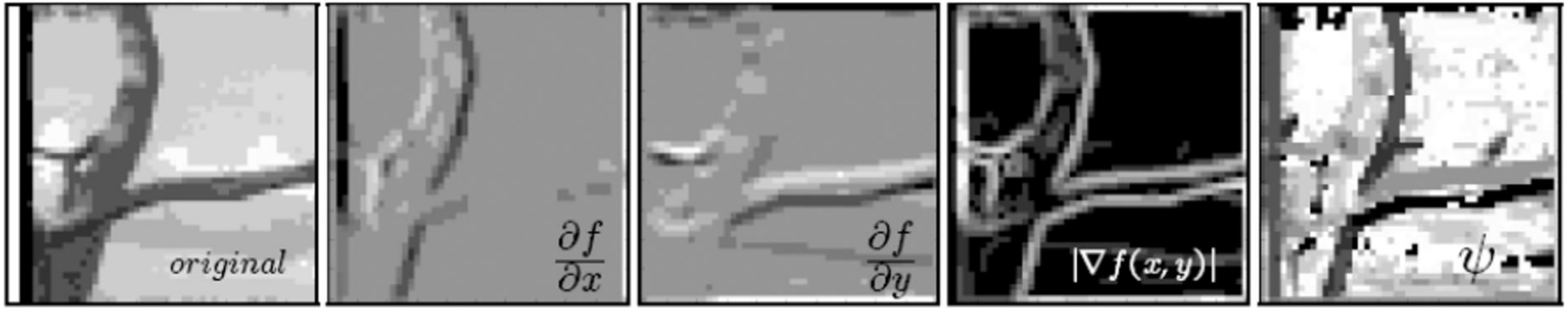


Second derivative

Gradient Extraction via Filtering

$$h_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad h_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

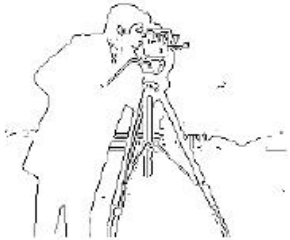
$$grad(f) = |\nabla f(x,y)| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} \qquad \psi = \arctan\left(\frac{\partial f/\partial y}{\partial f/\partial x}\right)$$

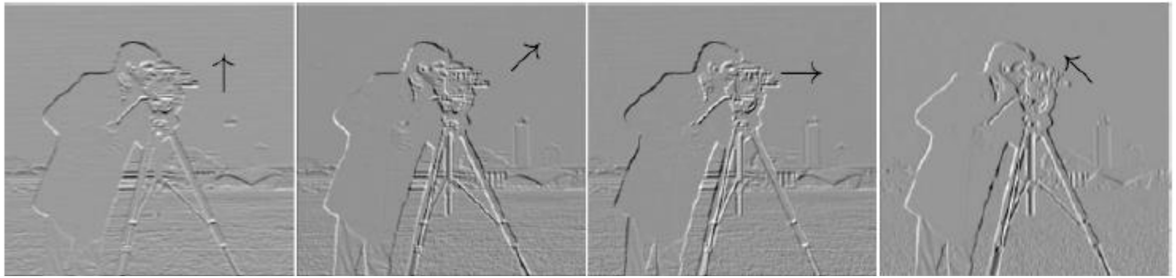


Prewitt Operator

- Central difference $\frac{\partial f}{\partial x} \approx \frac{f(x+1) - f(x-1)}{2}$
- Mask $[-1\ 0\ 1]$ is very sensitive to noise
- For 3x3 mask, ∇f can be estimated **in 8 directions**

$$h_{hor} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}, h_{dia} = \begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix}, \ldots$$





Sobel Operator

- As Prewitt, Sobel relies on central differences
- Greater weight to the central pixels

$$h_{hor} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, h_{ver} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

- Can be approx. as derivative of a Gaussian
- First Gaussian smoothing, then derivation

$$\frac{\partial}{\partial x}(I * G) = I * \frac{\partial G}{\partial x}$$

Note, the filters are named by the way they seek out gradients, e.g. horizontal and vertical gradients to detect vertical and horizontal edges, respectively.

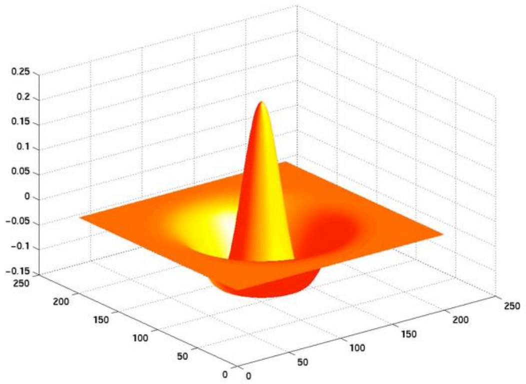
Sobel Operator

(a) original image

(b) Sobel edge magnitude

(c) thresholded magnitude

Laplacian of a Gaussian Edge Detector

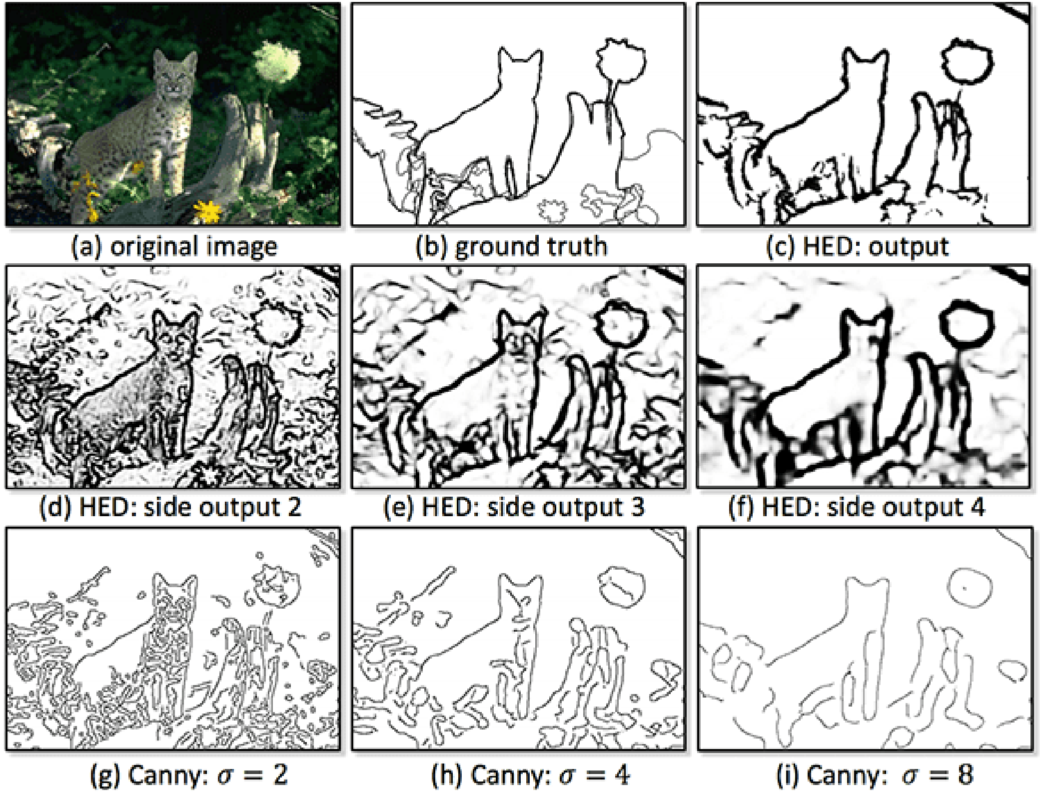


0	0	−1	0	0
0	−1	−2	−1	0
−1	−2	16	−2	−1
0	−1	−2	−1	0
0	0	−1	0	0

There are many edge detection methods…

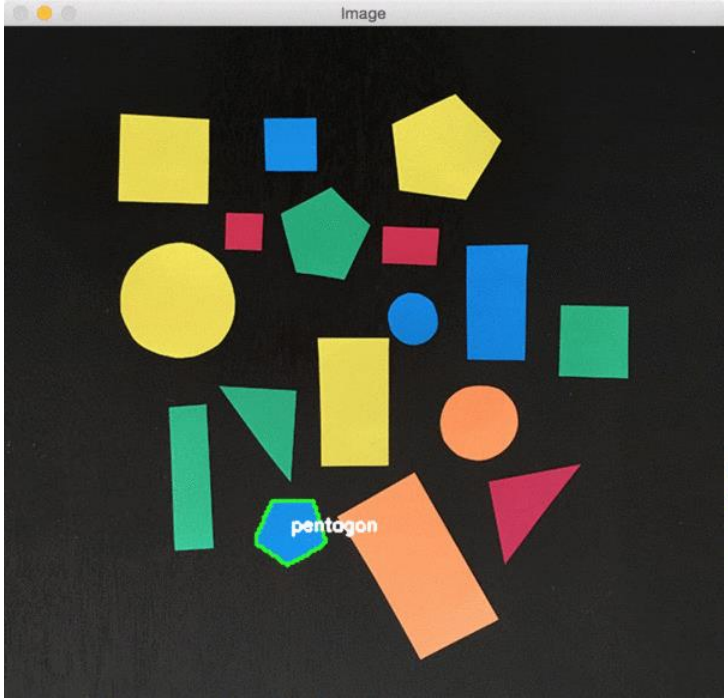
Most, if not all, edge detection methods operate with one or more parameters/thresholds.

Is there ever a perfect edge map…

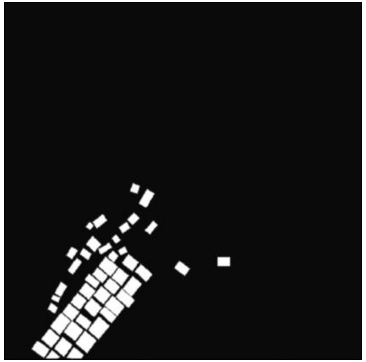


https://pyimagesearch.com/2019/03/04/holistically-nested-edge-detection-with-opencv-and-deep-learning/

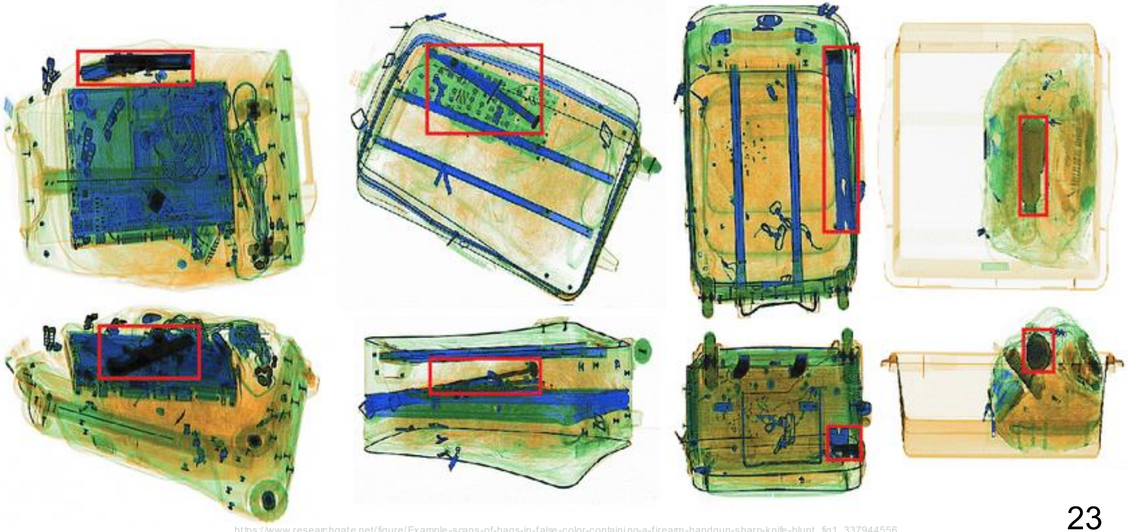
Shape Detection



https://pyimagesearch.com/2016/02/08/opencv-shape-detection/



https://www.hindawi.com/journals/cin/2022/4831223/



https://www.researchgate.net/figure/Example-scans-of-bags-in-false-color-containing-a-firearm-handgun-sharp-knife-blunt_fig1_337944556

Line Detection via the Hough Transform

In image space, a line is points (x, y) with gradient m, intercept c

$$y = mx + c$$

In parameter space, a line is points (m, c), with gradient $-x$, intercept y

$$c = -xm + y$$

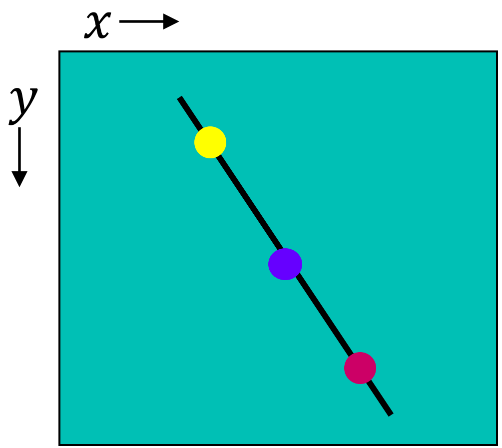
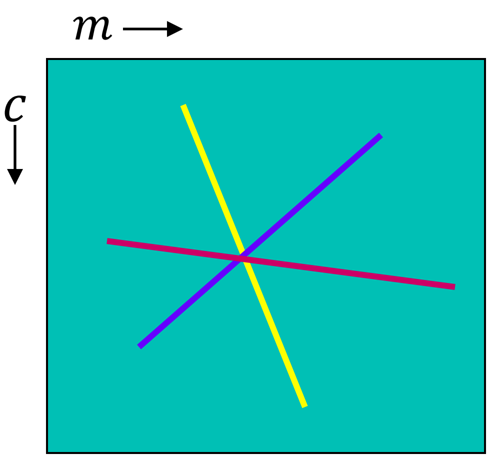


Image space

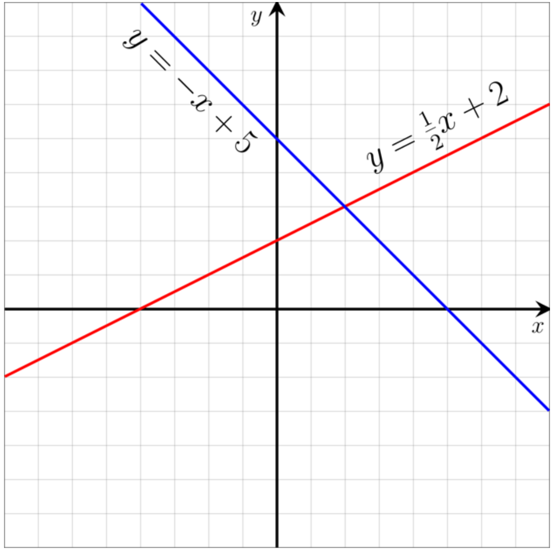


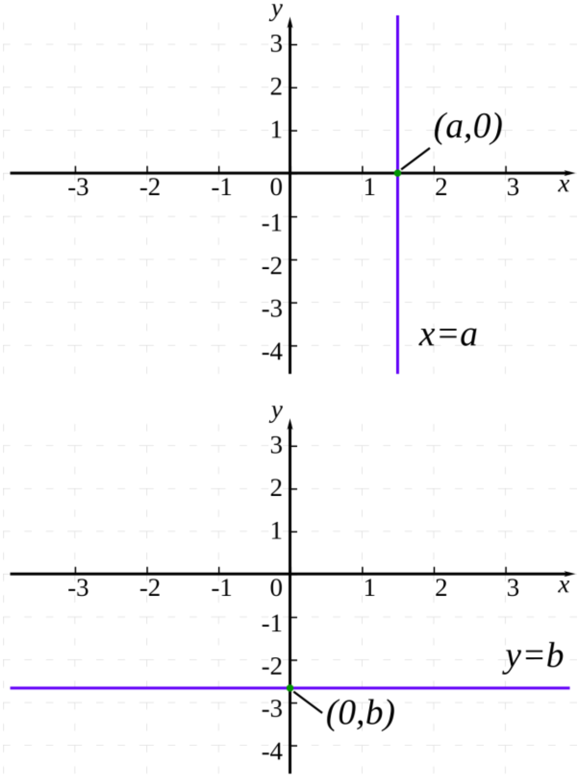
parameter space

Accumulate votes! The coordinates of the peak are the parameters m,c of the line!

Line Representation

$$y = mx + c$$



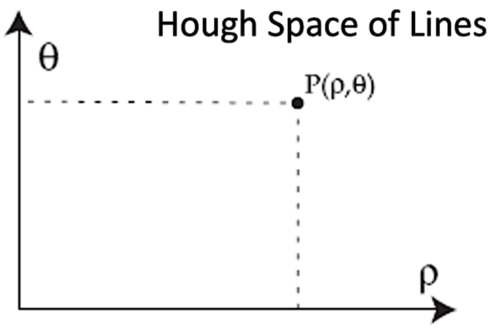


No y intercept!
slope is infinite!

Line Representation

A straight line in 2D space can also be described in its polar form:

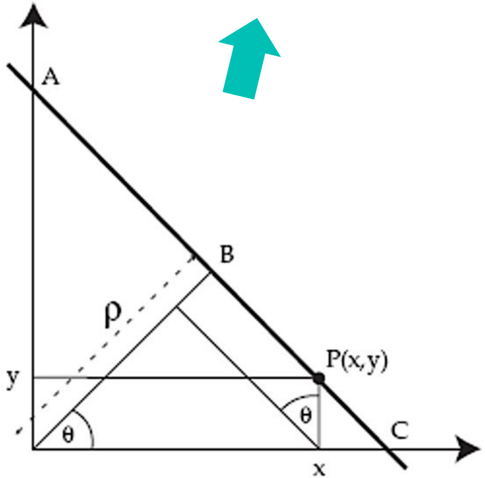
$$f(x, y, \rho, \theta) = x\cos\theta + y\sin\theta - \rho = 0$$



It can be represented in the 2D parameter space by point *(x,y)*

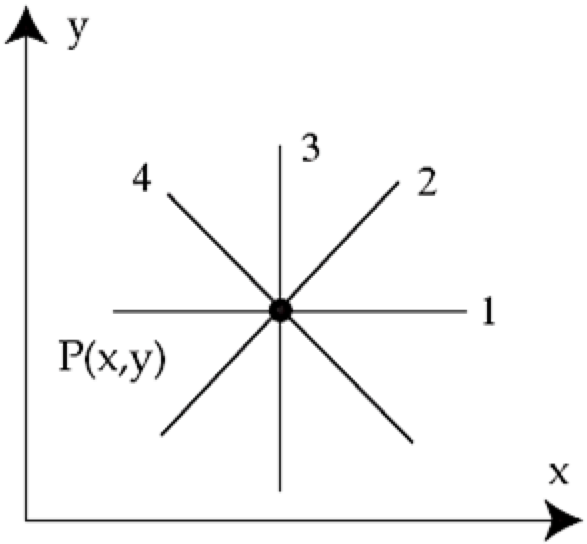
ρ = distance between the straight line and the origin,

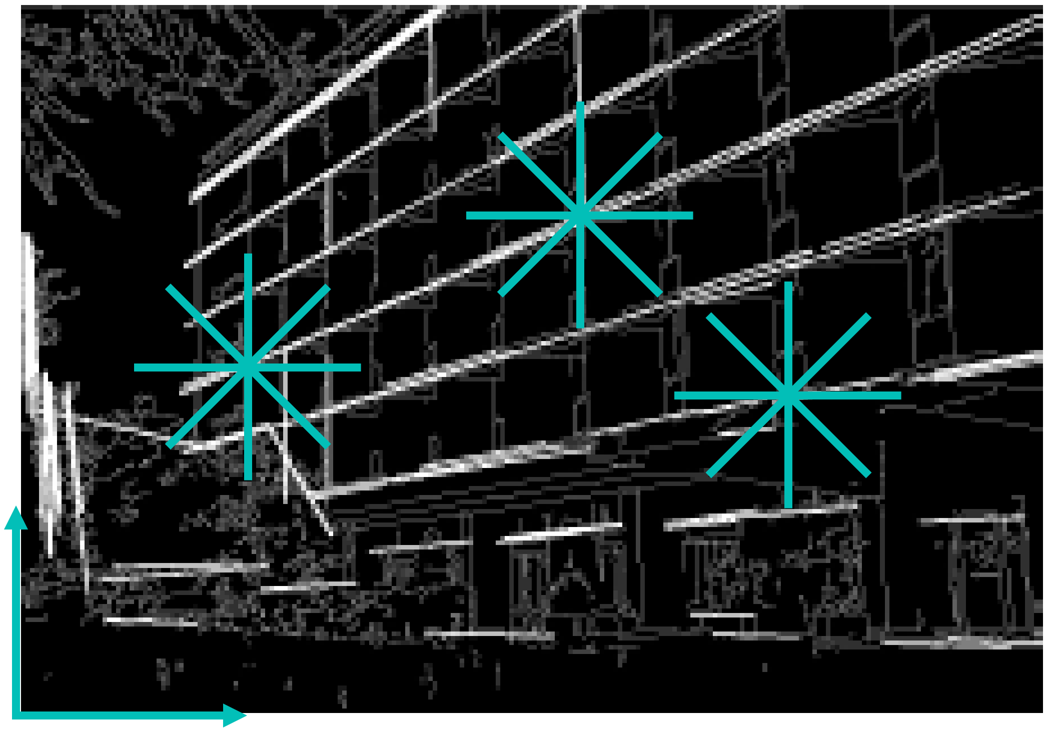
θ = angle between the distance vector and the positive *x*-direction.



Line Representation

$$f(x, y, \rho, \theta) = x\cos\theta + y\sin\theta - \rho = 0$$

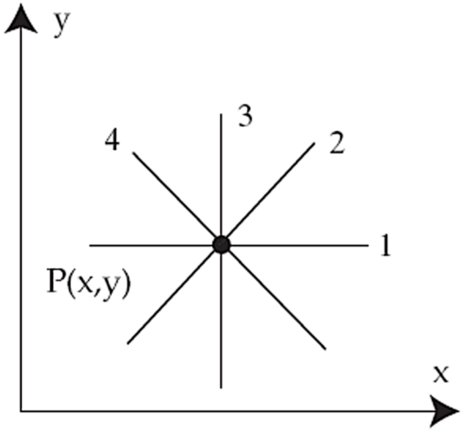


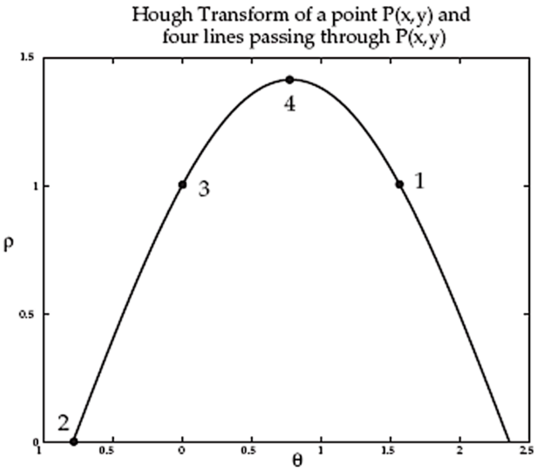


The Hough Space

A point (x, y) in the image space is transformed into a sinusoidal curve in the parameter space. A point (ρ, θ) on this sinusoidal curve represents a straight line passing through the point (x, y) in the image space.

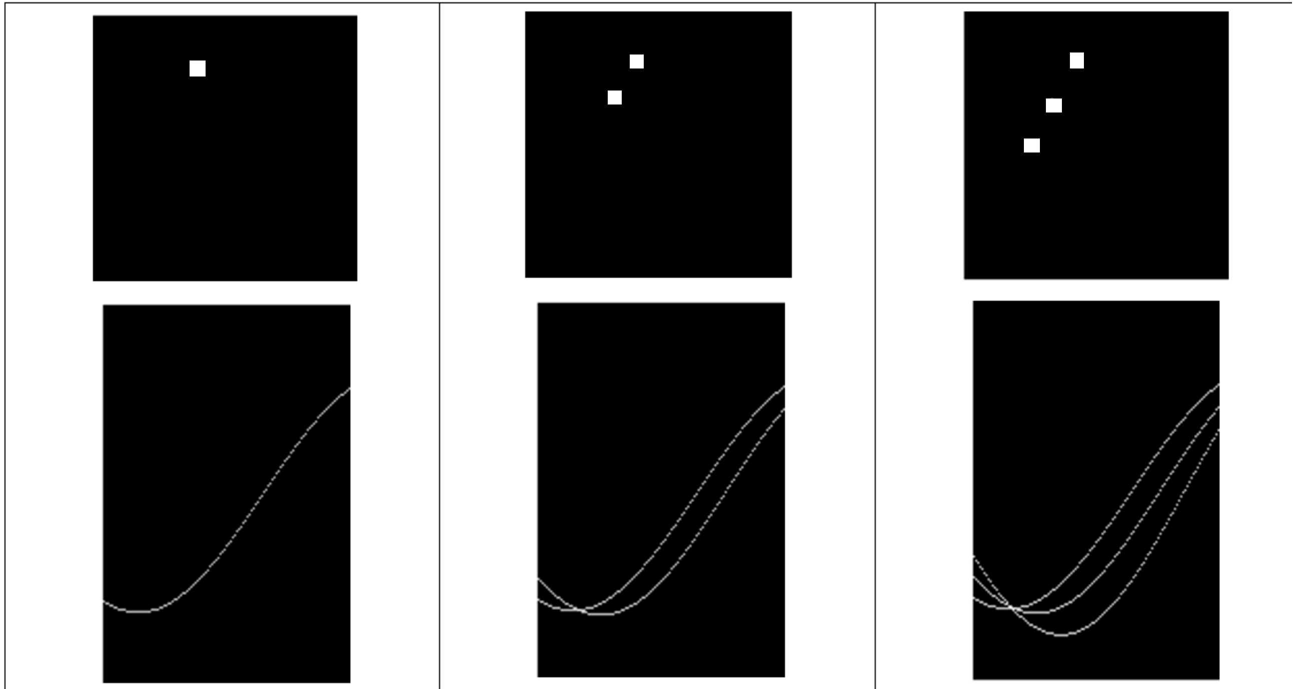
	point 1	point 2	point 3	point 4
θ	$\pi/2 = 1.571$	$-\pi/4 = -0.785$	0	$\pi/4 = 0.785$
ρ	1	0	1	1.4142





Line Representation

$$f(x, y, \rho, \theta) = x\cos\theta + y\sin\theta - \rho = 0$$



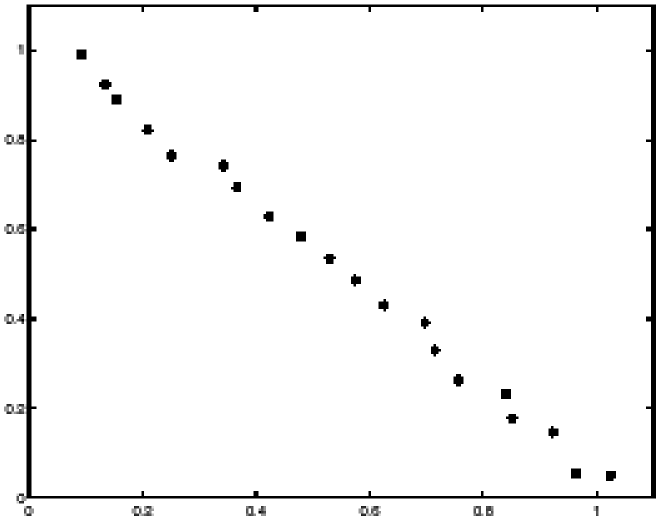
(a) for one point

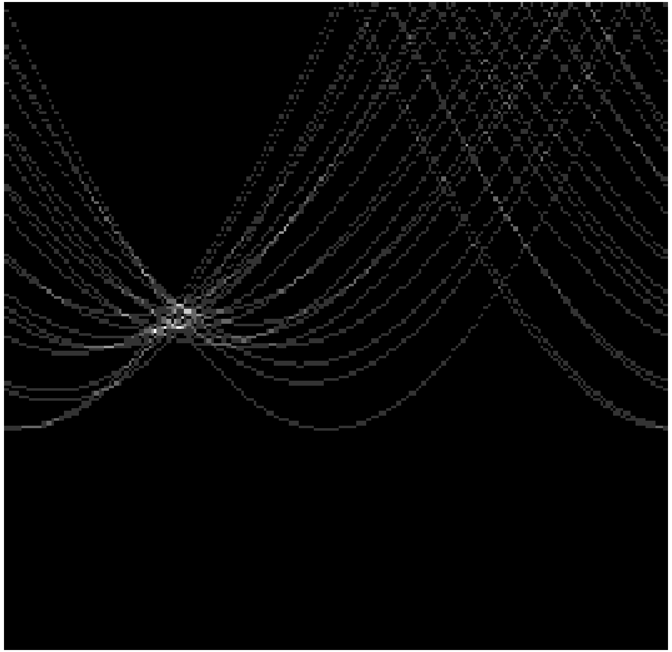
(b) for two points

(c) for three points

Line Representation

$$f(x, y, \rho, \theta) = x \cos \theta + y \sin \theta - \rho = 0$$

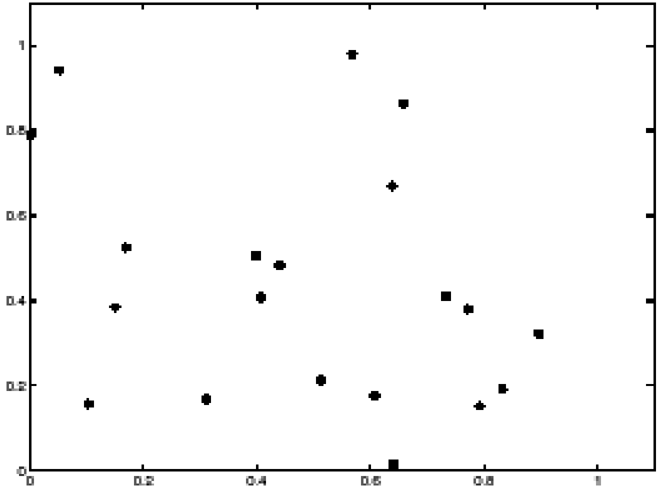


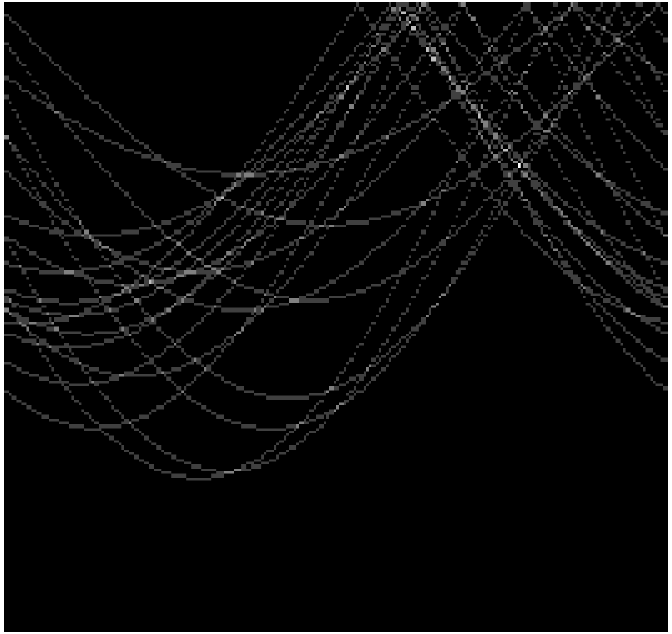


Peak gets fuzzy and hard to locate

Line Representation

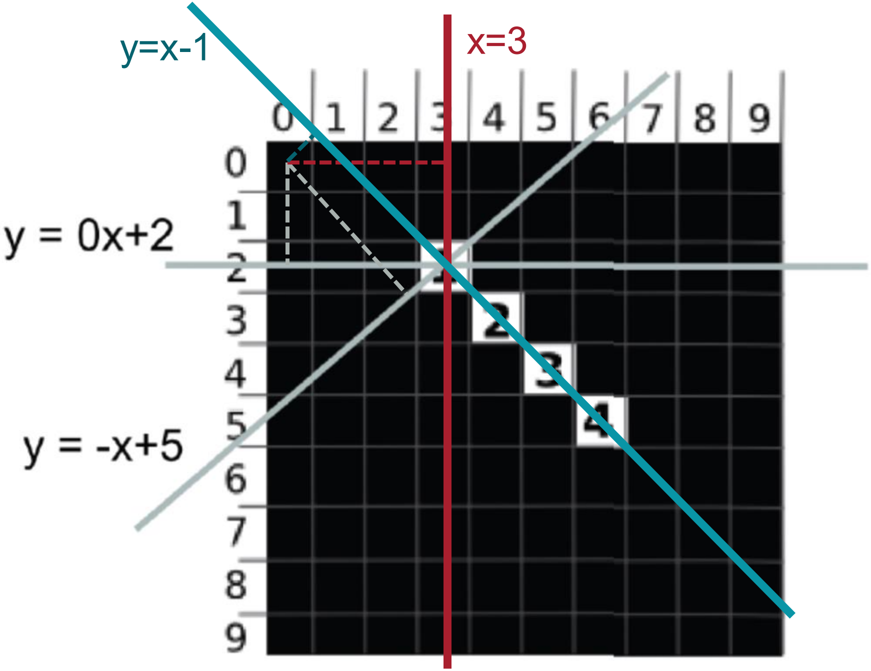
$$f(x, y, \rho, \theta) = x\cos\theta + y\sin\theta - \rho = 0$$

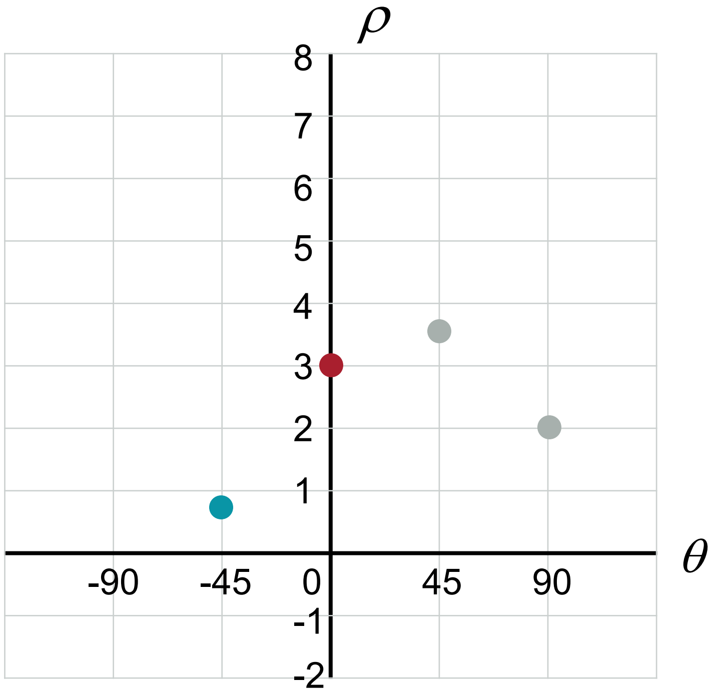




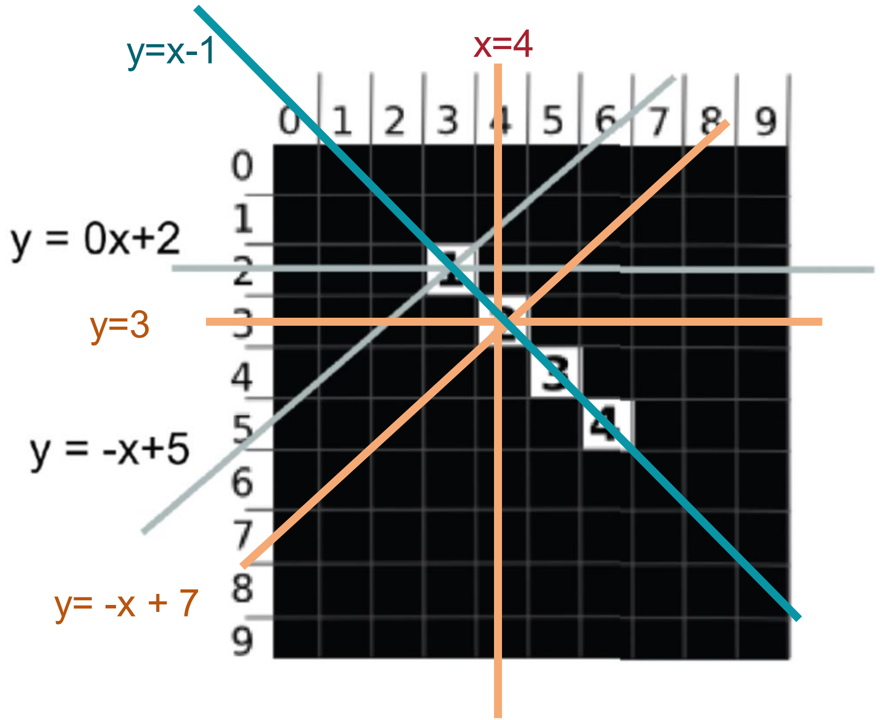
Noise can lead to spurious peaks in the accumulator array

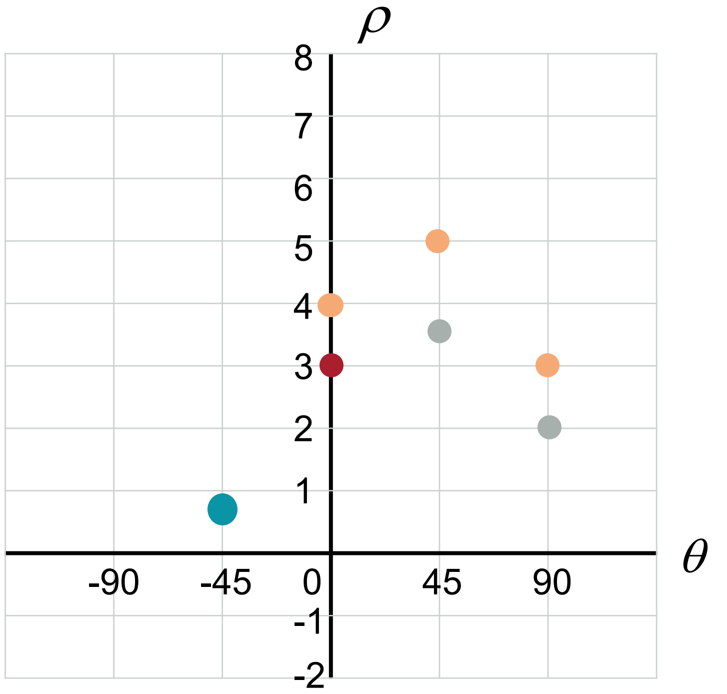
Building the Hough Space



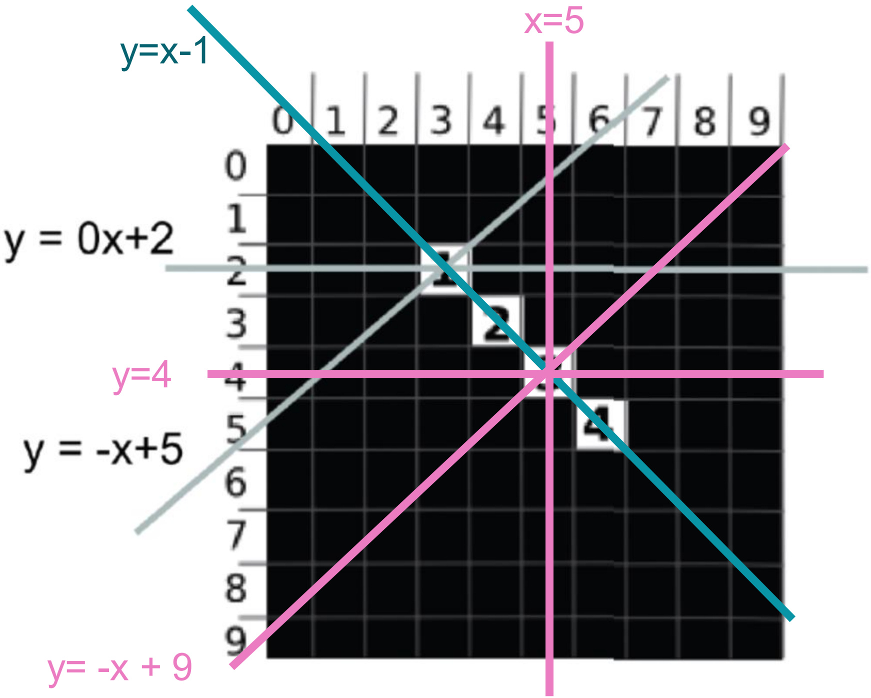


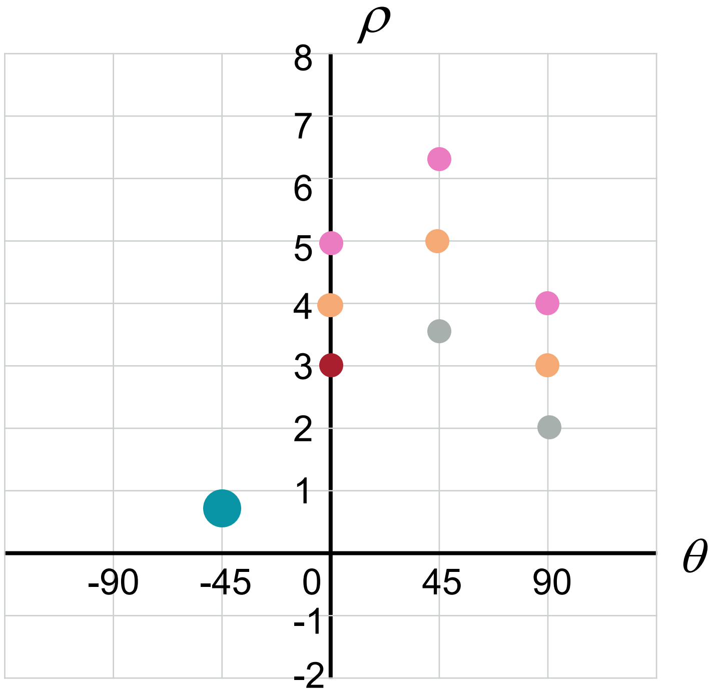
Building the Hough Space



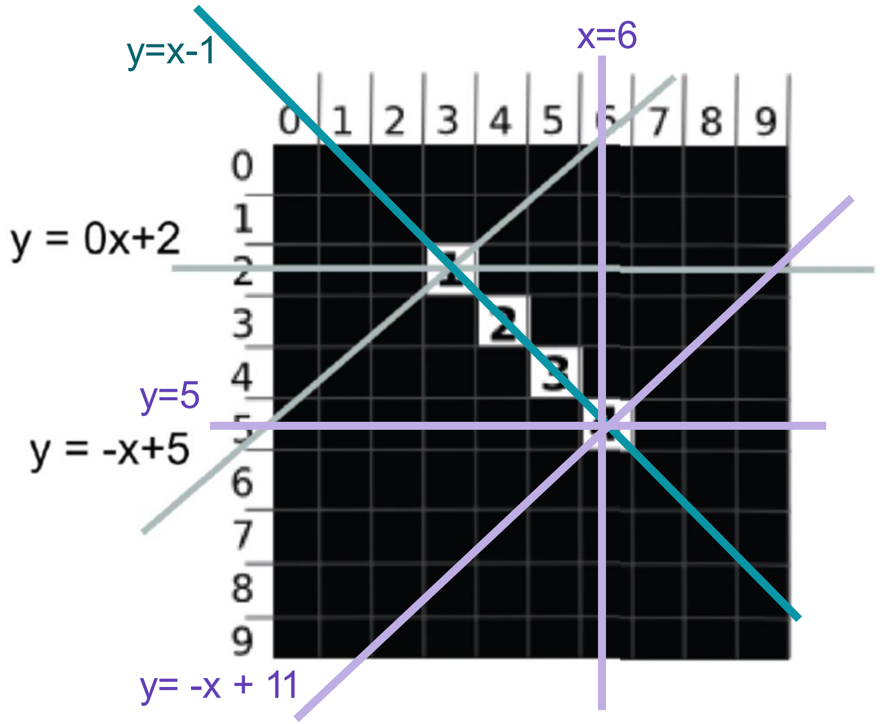


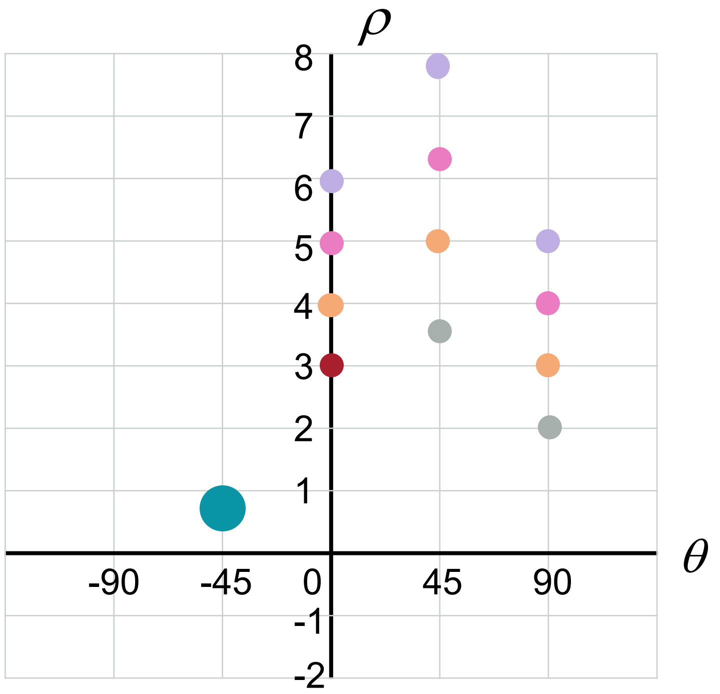
Building the Hough Space



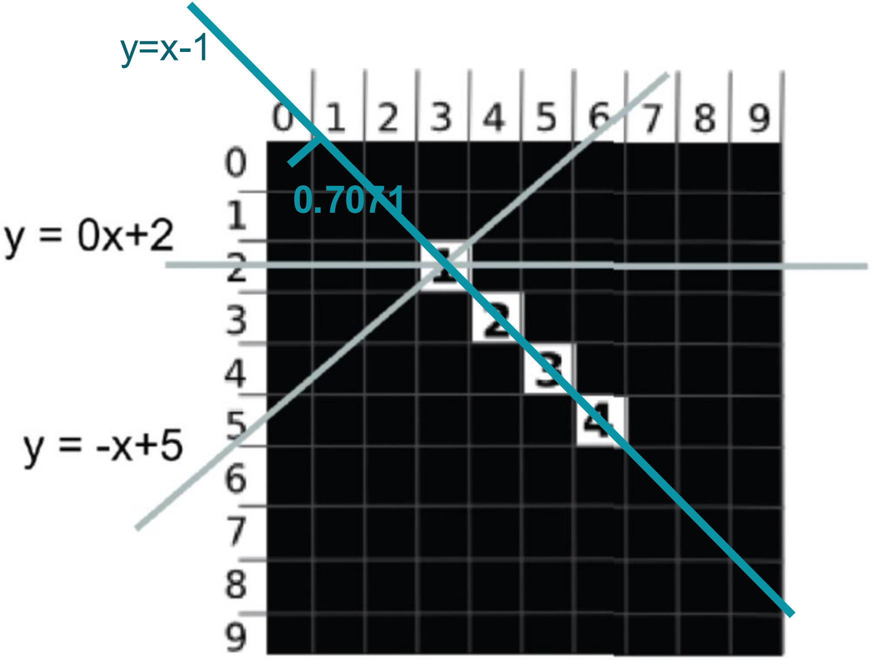


Building the Hough Space

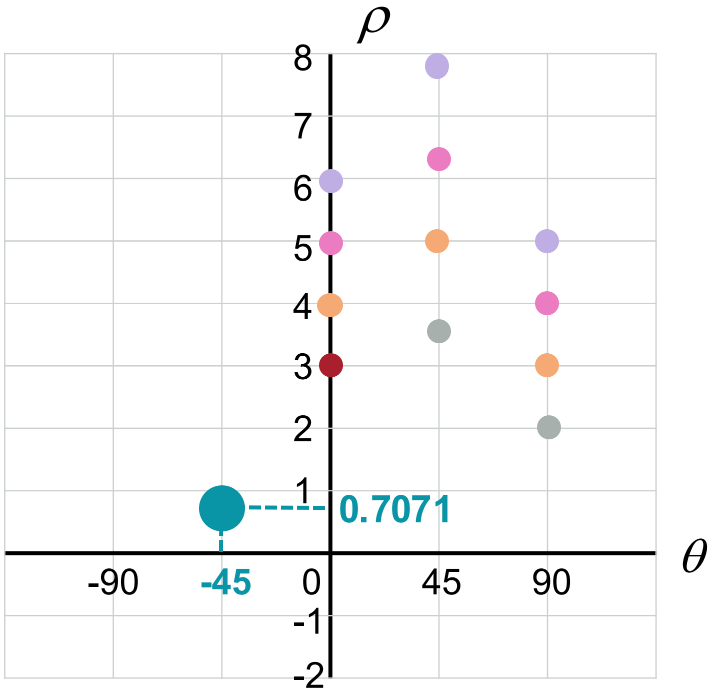




Building the Hough Space



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



0.7071 = x * 0.7071 + y * 0.7071

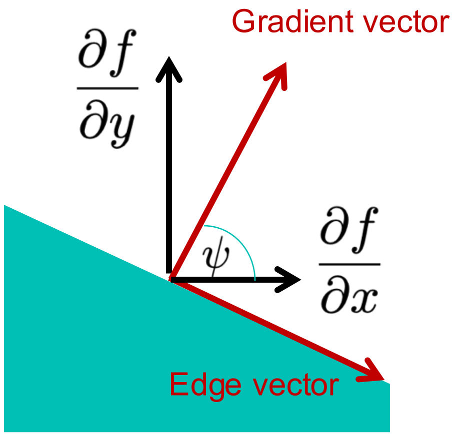
Line Detection Algorithm

1. Make available an $n = 2$ dimensional array $H(\rho, \theta)$ for the parameter space;
2. Find the gradient image: $G(x, y) = |G(x, y)| \angle G(x, y)$;
3. For any pixel satisfying $|G(x, y)| > T_s$, increment all elements on the curve $\rho = x\ cos\ \theta + y\ sin\ \theta$ in the parameter space represented by the H array:

$$\forall \theta \quad | \quad \rho = x\ cos\ \theta + y\ sin\ \theta$$
$$H(\rho, \theta) = H(\rho, \theta) + 1;$$

4. In the parameter space, any element $H(\rho, \theta) > T_h$ represents a straight line detected in the image.

Line Detection using Gradient Information



1. Make $n = 2$ dimensional array $H(\rho, \theta)$
2. Find the gradient image: $G(x, y) = |G(x, y)| \angle G(x, y)$;
3. For any pixel satisfying $|G(x, y)| > T_s$,

$$\forall \theta \quad | \quad \angle G(x, y) - \Delta\theta \leq \theta \leq \angle G(x, y) + \Delta\theta$$

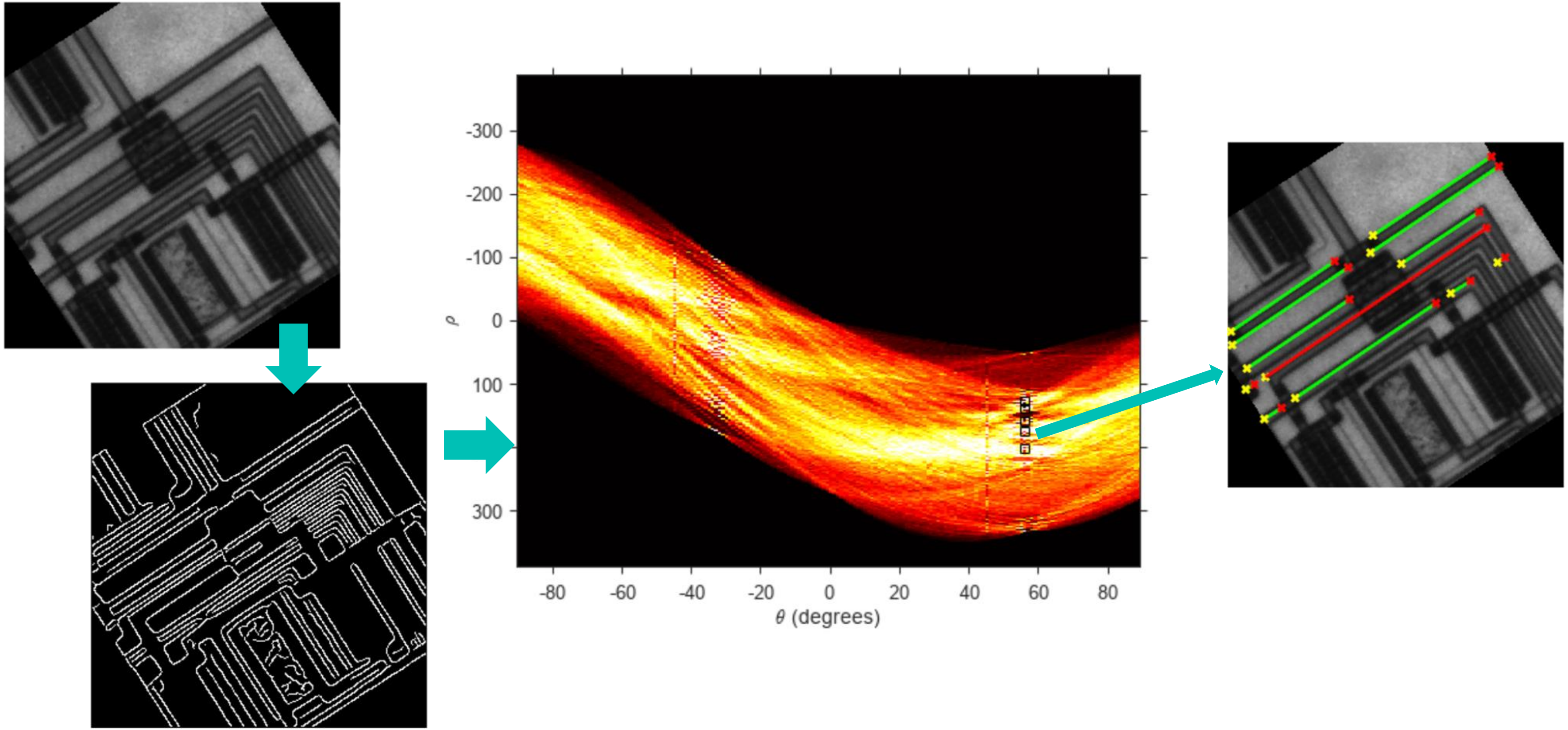
$$\rho = x \ cos \ \theta + y \ sin \ \theta$$

$$H(\rho, \theta) = H(\rho, \theta) + 1;$$

where $\Delta\theta$ defines a small range in θ to allow some room for error in $\angle G$.

. Any element $H(\rho, \theta) > T_h$ represents a straight line

Line Detection Example

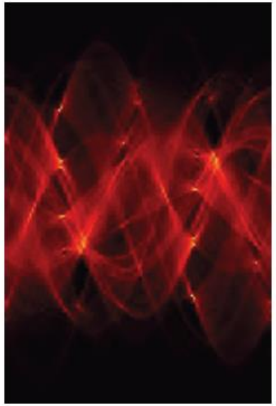


https://uk.mathworks.com/help/images/hough-transform.html

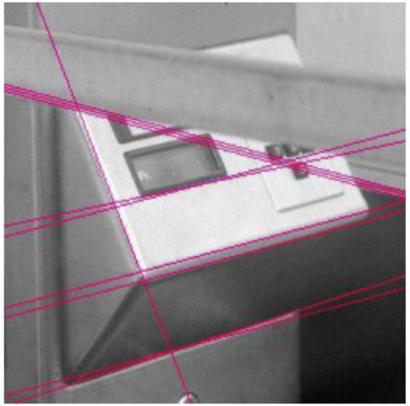
Line Detection Example

Original

Edges



Parameter Space



Hough Lines

Circle Detection Algorithm

1. For any pixel satisfying $|G(x, y)| > T_s$, increment all elements satisfying the two simultaneous equations

$$\forall r, \quad \begin{cases} x_0 = x \pm r \; cos \; \angle G \\ y_0 = y \pm r \; sin \; \angle G \end{cases}$$

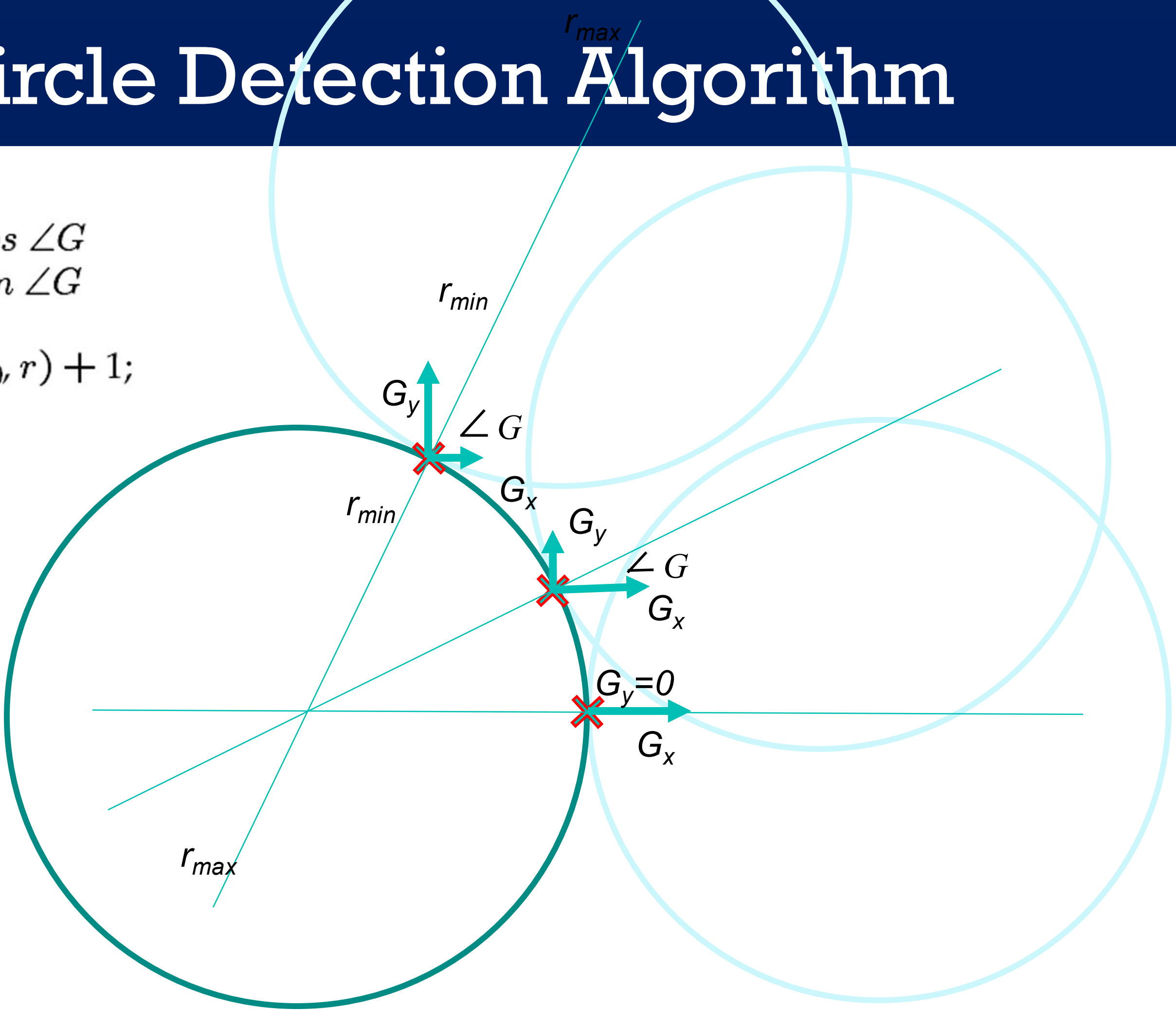
$$H(x_0, y_0, r) = H(x_0, y_0, r) + 1;$$

2. In the parameter space, any element $H(x_0, y_0, r) > T_h$ represents a circle with radius r located at (x_0, y_0) in the image.

Circle Detection Algorithm

$$\forall r, \quad \begin{cases} x_0 = x \pm r \; cos \; \angle G \\ y_0 = y \pm r \; sin \; \angle G \end{cases}$$

$$H(x_0, y_0, r) = H(x_0, y_0, r) + 1;$$



Circle Detection Algorithm

(1)

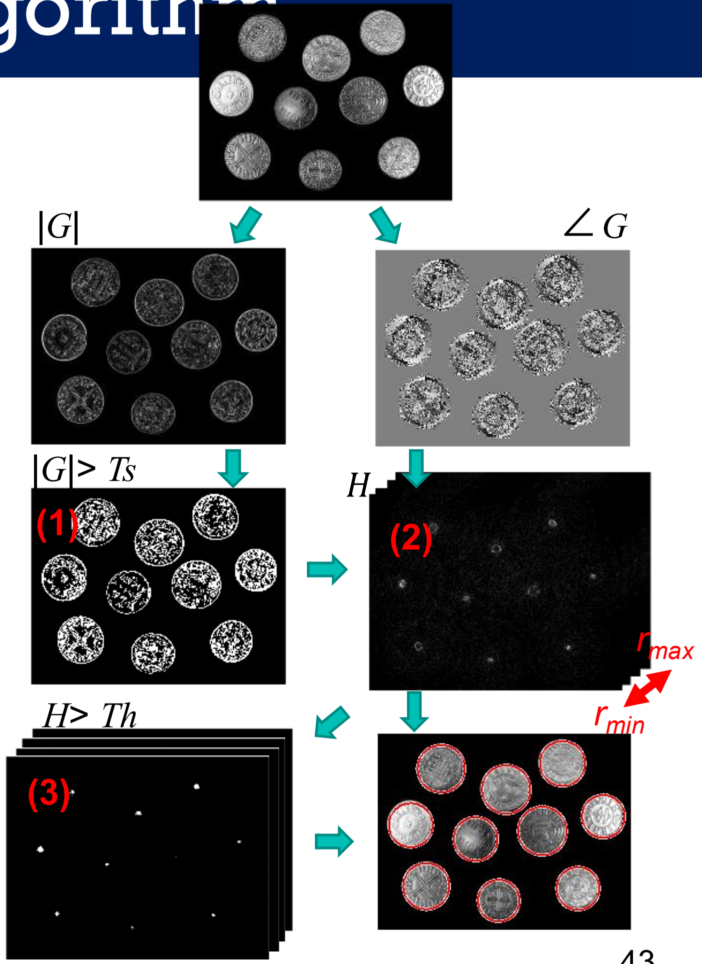
1. For any pixel satisfying $\boxed{|G(x, y)| > T_s}$, increment all elements satisfying the two simultaneous equations

(2)

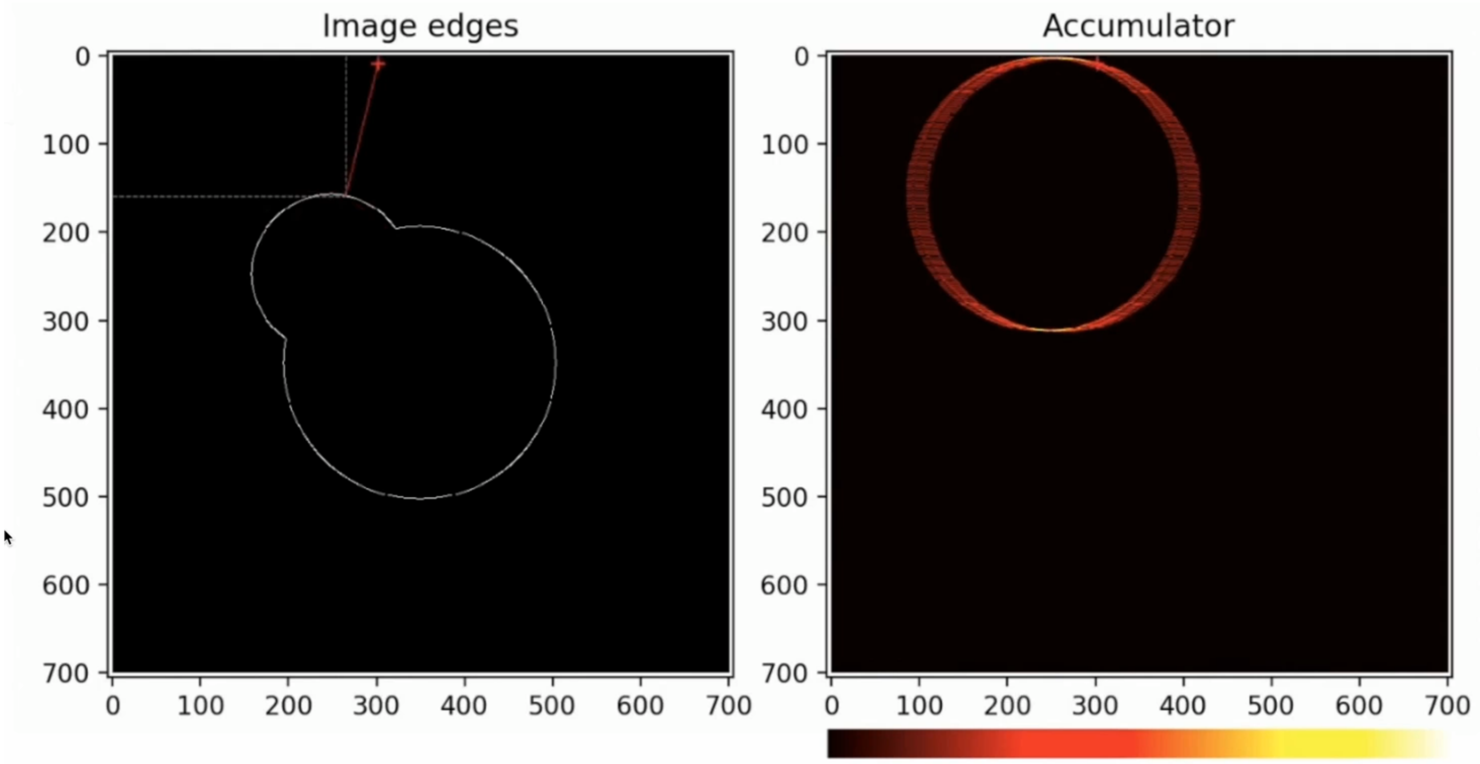
$$\boxed{\begin{aligned} &\forall r, \quad \begin{cases} x_0 = x \pm r\ cos\ \angle G \\ y_0 = y \pm r\ sin\ \angle G \end{cases} \\ &H(x_0, y_0, r) = H(x_0, y_0, r) + 1; \end{aligned}}$$

(3)

2. In the parameter space, any element $\boxed{H(x_0, y_0, r) > T_h}$ represents a circle with radius r located at (x_0, y_0) in the image.



Circle Detection Algorithm



https://www.youtube.com/watch?v=Ltqt24SQQoI&t=205s

Summary: Shape Recognition via Hough Transform

- Image space is transformed into a parameter space
- Voting procedure is carried out in the parameter space.
- Object candidates are obtained as local maxima.

Source: https://stackoverflow.com/questions/22274930/detect-objects-similar-to-circles/22279177

Next Lecture

Image Segmentation