# Review: Fourier Transform

# The convolution theorem

The Fourier transform of the convolution of two functions is the pro du ct of their Fourier transforms:

$$\mathcal{F}\{g * h\} = \mathcal{F}\{g\}\mathcal{F}\{h\}$$

The Fourier transform of the product of two functions is the convolu tion of the two Fourier transforms:

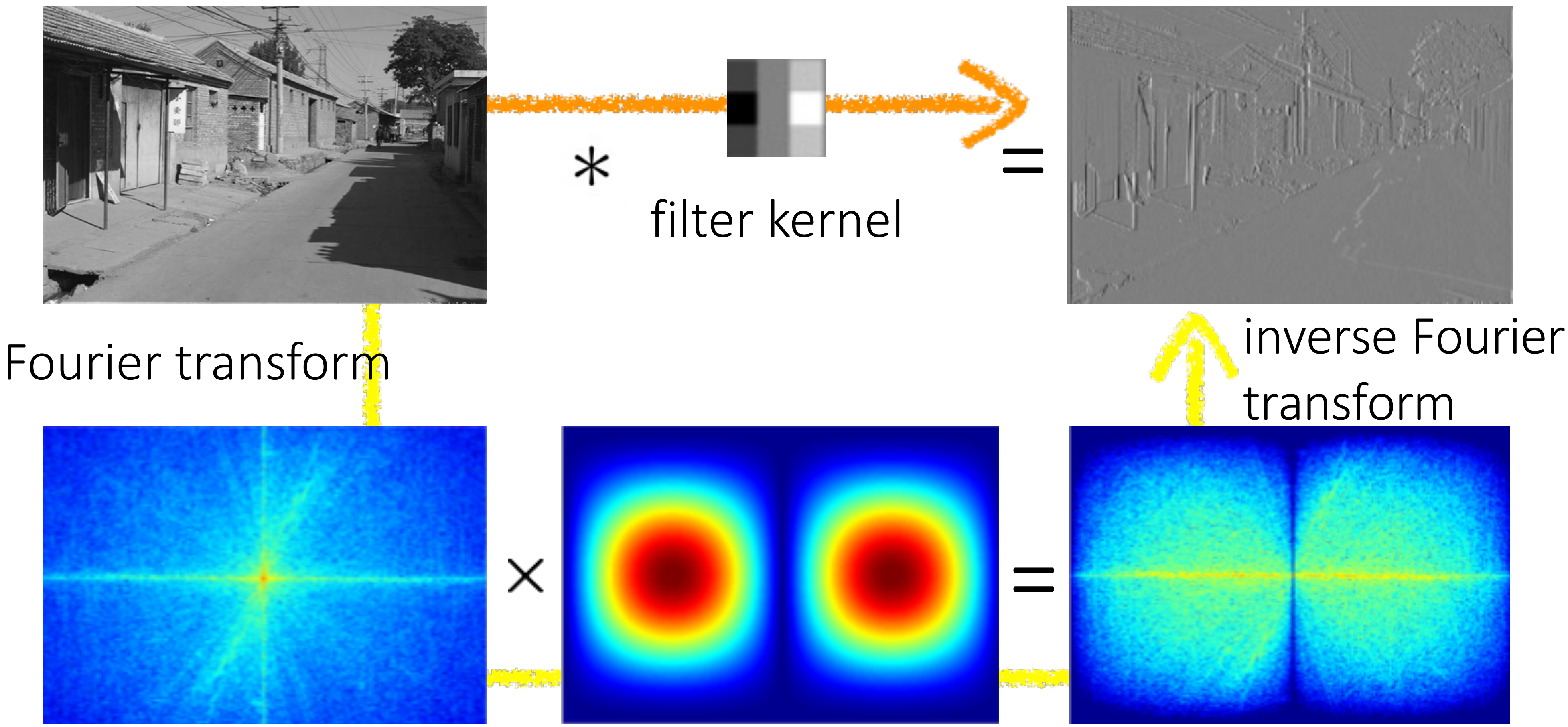
$$\mathcal{F}\{g \cdot h\} = \mathcal{F}\{g\} * \mathcal{F}\{h\}$$

Convolution in spatial domain is equivalent to multiplication in frequency domain!

$$g * h = \mathcal{F}^{-1}\left\{\mathcal{F}\{g\} \cdot \mathcal{F}\{h\}\right\}$$

$$g \cdot h = \mathcal{F}^{-1}\left\{\mathcal{F}\{g\} * \mathcal{F}\{h\}\right\}$$

# Spatial domain filtering

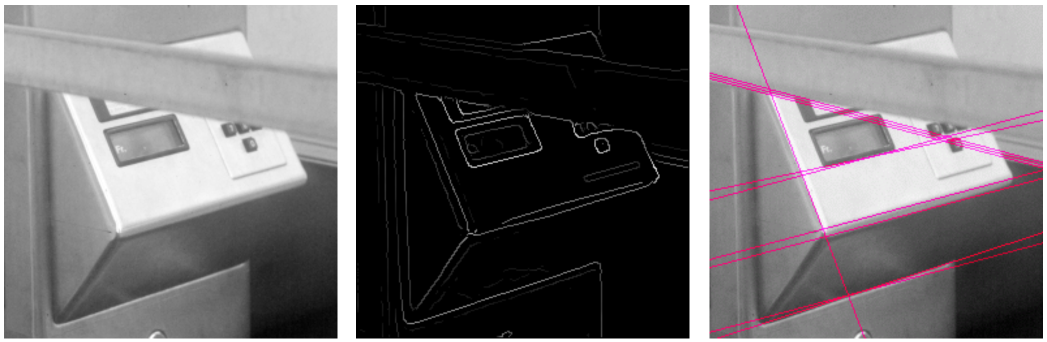



# Frequency domain filtering

# Hough transform

# Fitting a model

- Want to associate a model with observed features



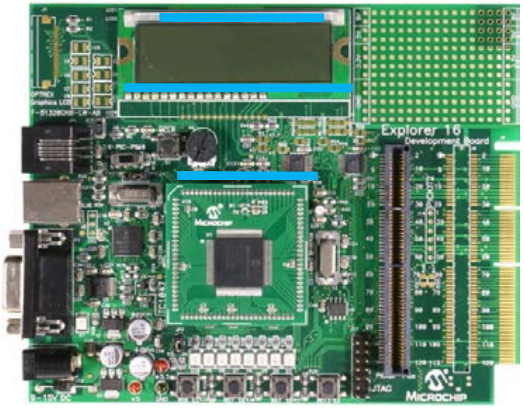
[Fig from Marszalek & Schmid, 2007]

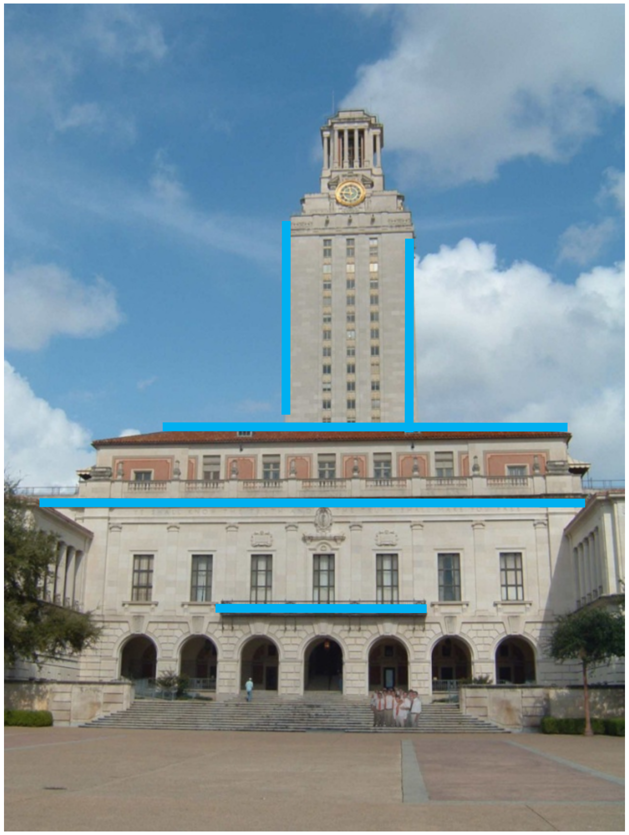
For example, the model could be a line, a circle, or an arbitrary shape.

# Example: Line fitting

- Why fit lines?

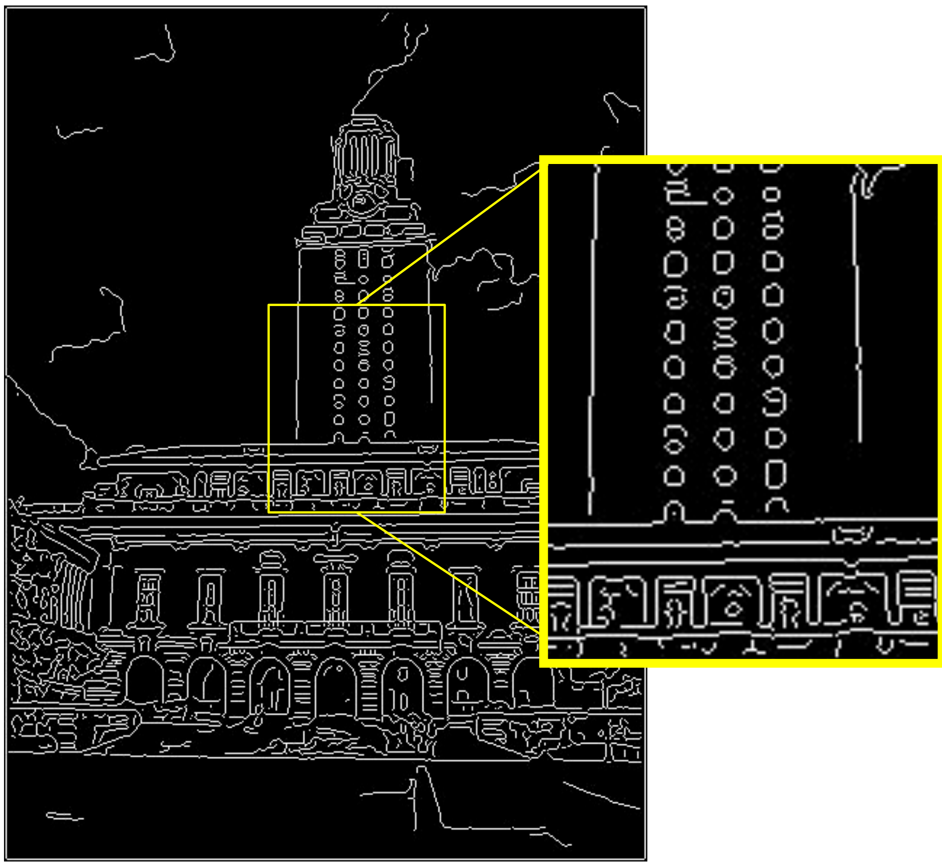
*Many objects characterized by presence of straight lines*



- Wait, why aren’t we done just by running edge detection?

# Difficulty of line fitting



- **Extra** edge points (clutter), multiple models:
  - which points go with which line, if any?
- Only some parts of each line detected, and some parts are **missing:**
  - how to find a line that bridges missing evidence?
- **Noise** in measured edge points, orientations:
  - how to detect true underlying parameters?

# Voting

It's not feasible to check all possible models or all combinations of features (e.g. edge pixels) by fitting a model to each possible subset.

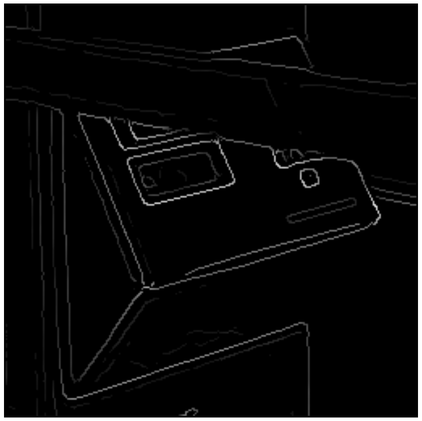
*Voting* is a general technique where we let the features vote for all models that are compatible with it.

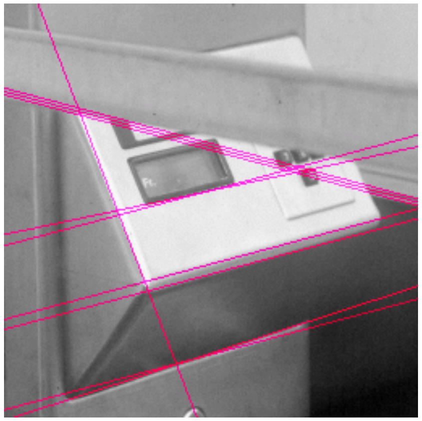
1. Cycle through features, each casting votes for model parameters.
2. Look for model parameters that receive a lot of votes.

# Fitting lines

- Given points that belong to a line, what is the line?
- How many lines are there?
- Which points belong to which lines?

- **Hough Transform** is a voting technique that can be used to answer all of these
  - Main idea:
  - 1. Record all possible lines on which each edge point lies.
  - 2. Look for lines that get many votes.





# Slope intercept form

variables

$$y = mx + b$$

parameters

variables

$$y - mx = b$$

parameters

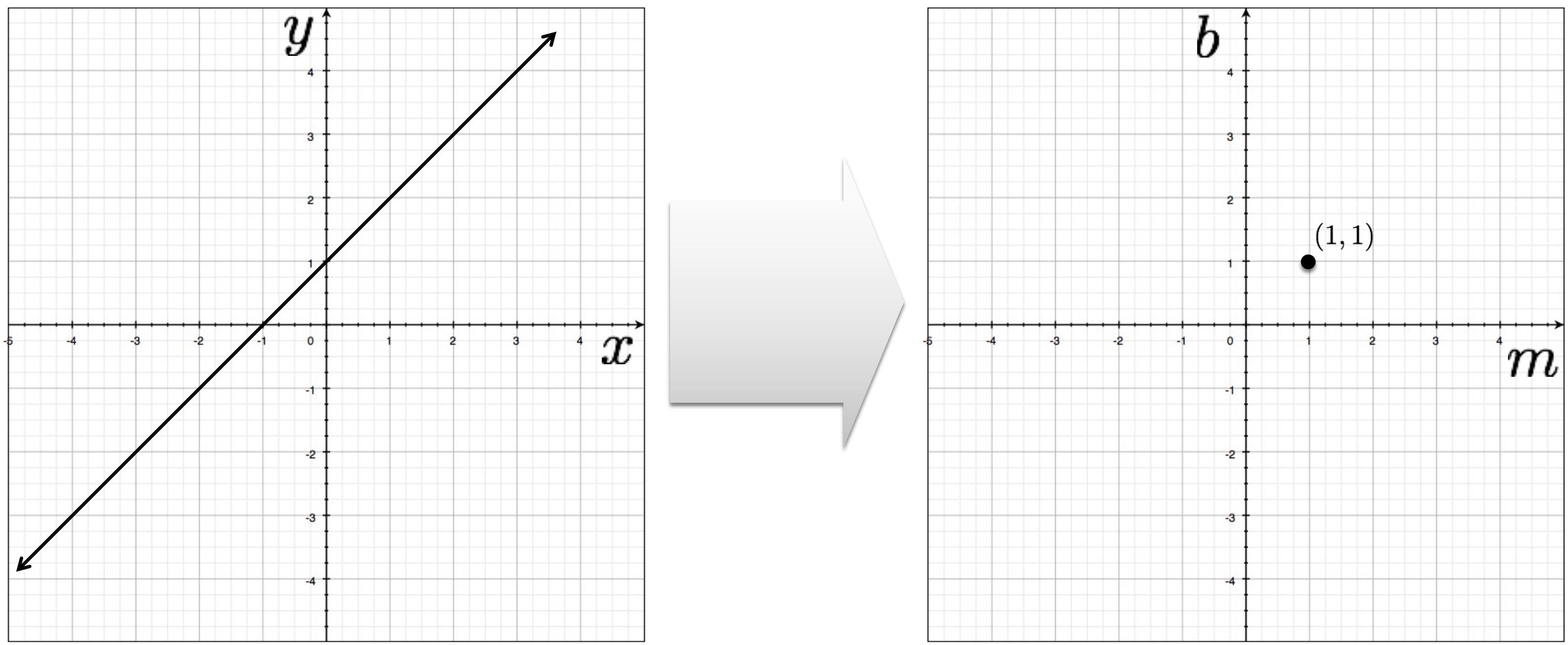



Image space

Parameter space

# Slope intercept form

variables

$$y = mx + b$$

parameters

variables

$$y - mx = b$$

parameters

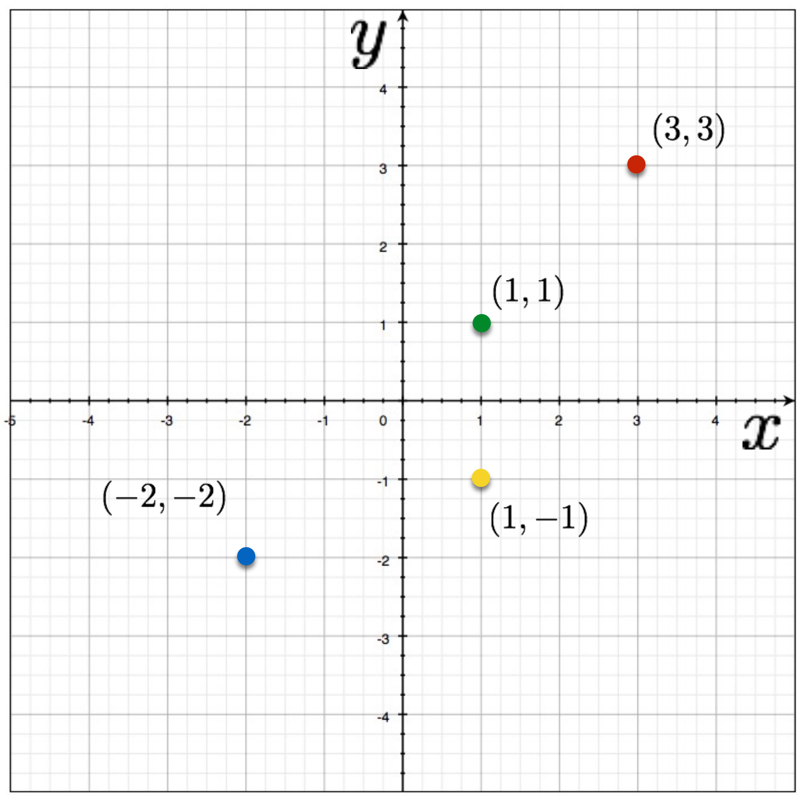
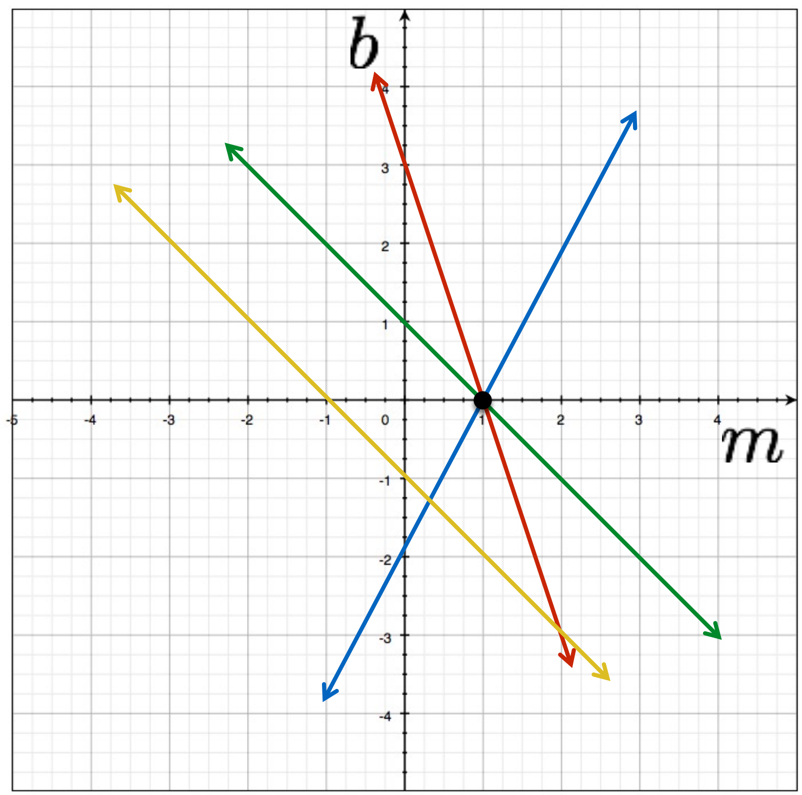



Image space




Parameter space

# Line Detection by Hough Transform

**Algorithm:**

1.Quantize Parameter Space $(m,c)$

2.Create Accumulator Array $A(m,c)$

3.Set $A(m,c)=0 \quad \forall m,c$

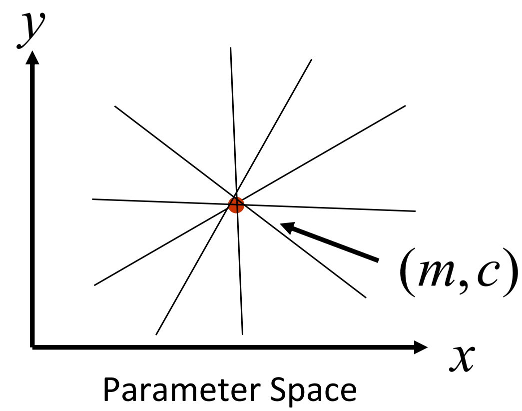
4.For each image edge $(x_i, y_i)$

For each element in $A(m,c)$

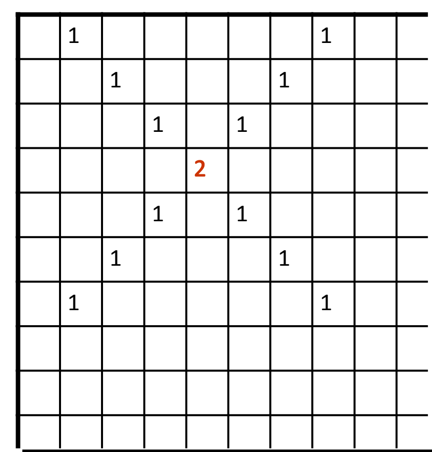
If $(m,c)$ lies on the line: $c = -x_i m + y_i$

Increment $A(m,c) = A(m,c) + 1$

5. Find local maxima in $A(m,c)$




$A(m,c)$



The space of m is huge! The space of c is huge!

# Better Parameterization

Use normal form:

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

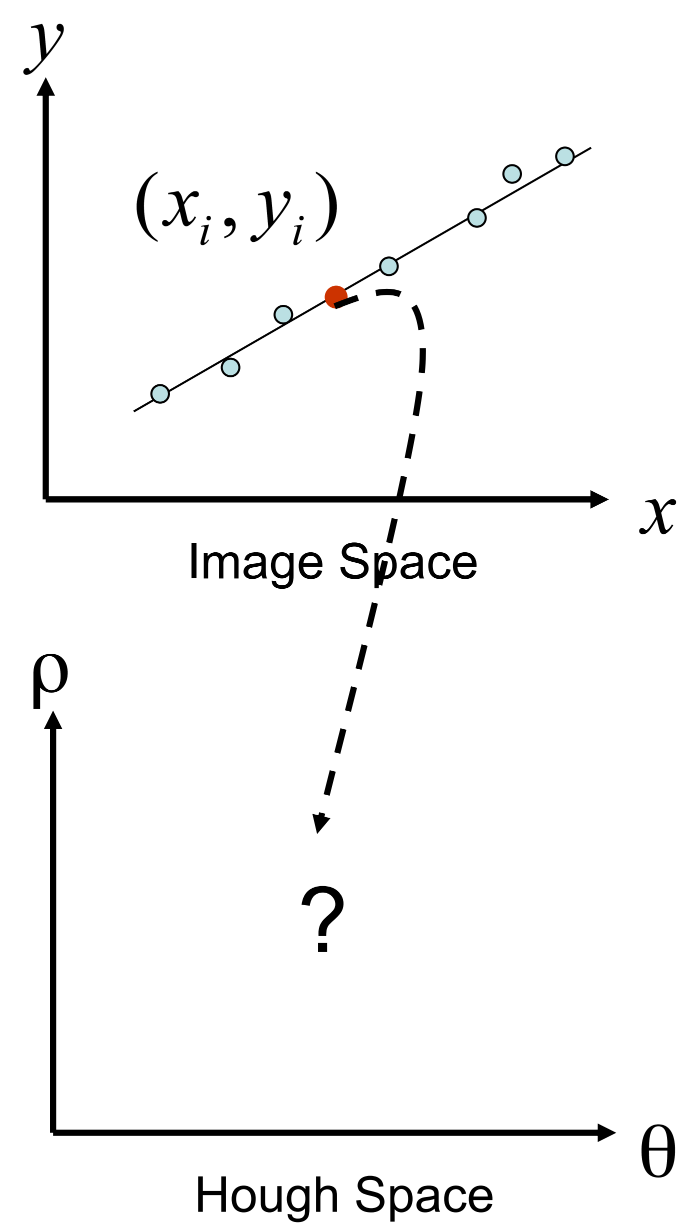
Given points $(x_i, y_i)$ find $(\rho, \theta)$

Hough Space Sinusoid

$$0 \leq \theta \leq 2\pi$$

$$0 \leq \rho \leq \rho_{max}$$

(Finite Accumulator Array Size)



# Image and parameter space

variables

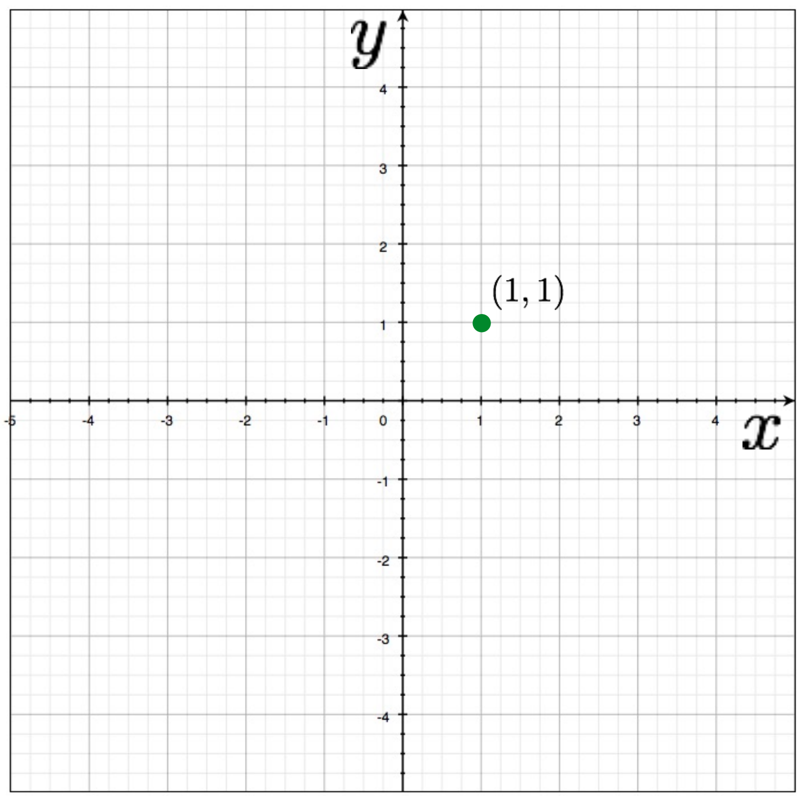
$$y = mx + b$$

parameters

parameters

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

variables




a point becomes a wave


Image space

Parameter space

# Image and parameter space

variables

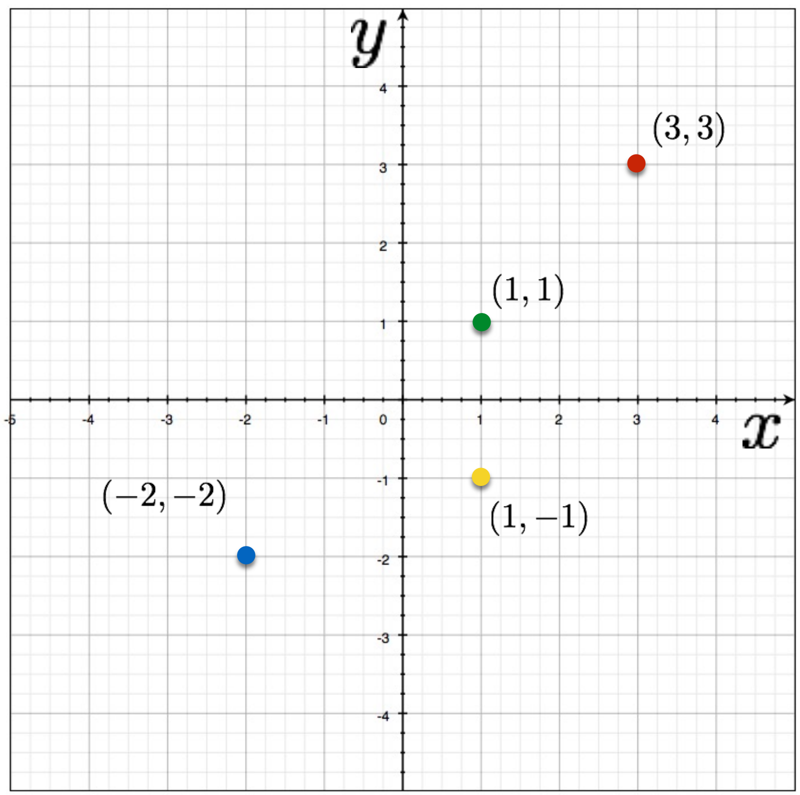
$$y = mx + b$$

parameters

parameters

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

variables




four points become ?

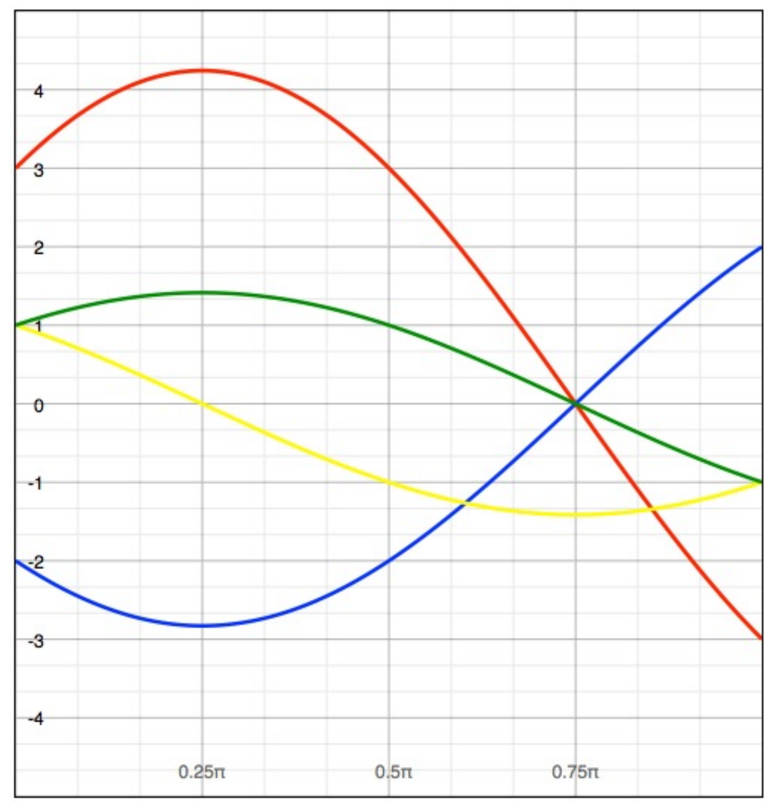



Image space

Parameter space

# Implementation

1. Initialize accumulator H to all zeros

2. For each edge point (x,y) in the image

    For $\theta$ = 0 to 360

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

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

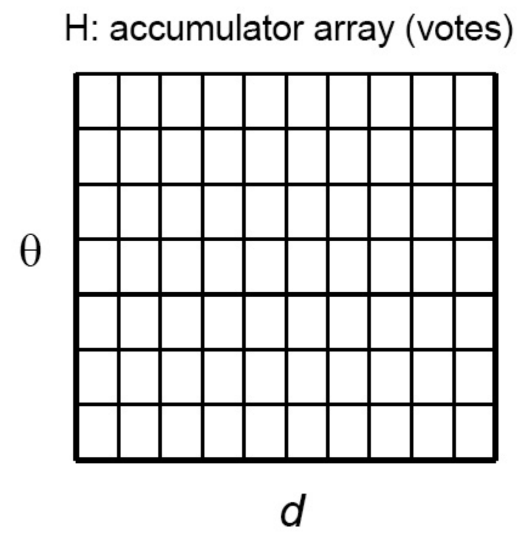
    end

end

3. Find the value(s) of $(\theta, \rho)$ where $H(\theta, \rho)$ is a local maximum

4. The detected line in the image is given by

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




cv2.HoughLines(), cv2.HoughLinesP()

https://opencv24-python-tutorials.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_houghlines/py_houghlines.html

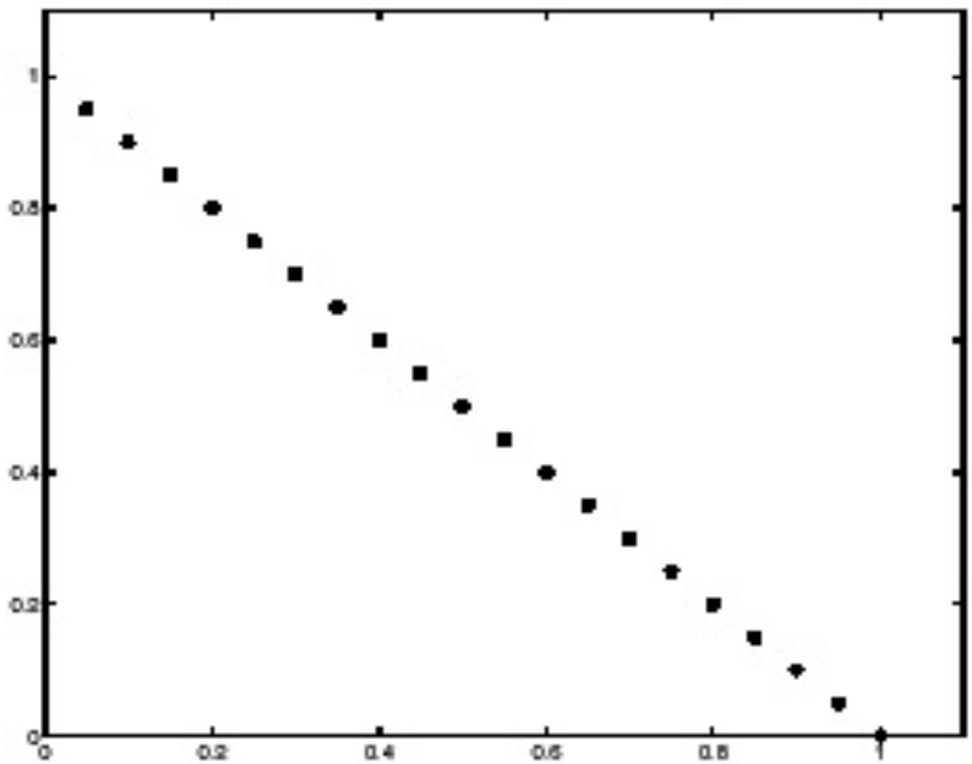
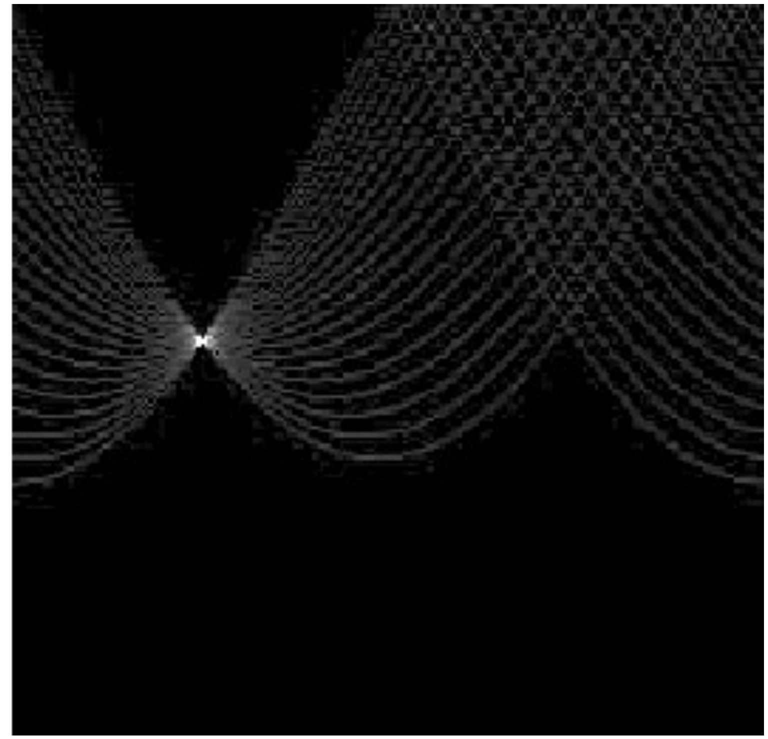


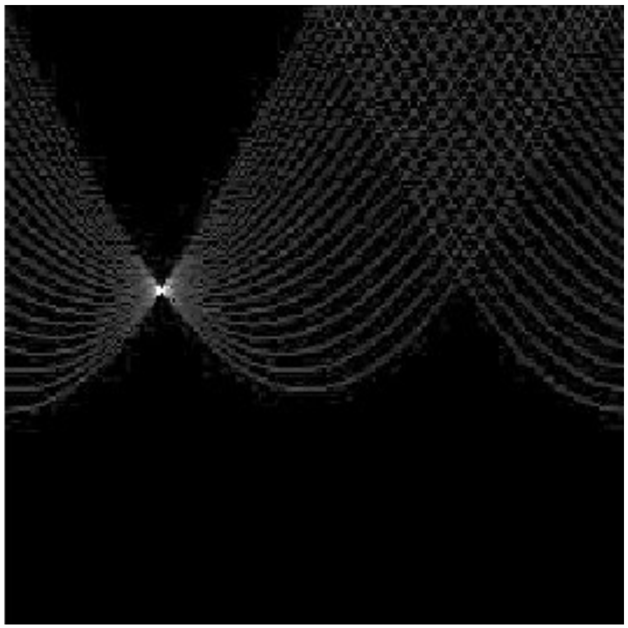

Image space



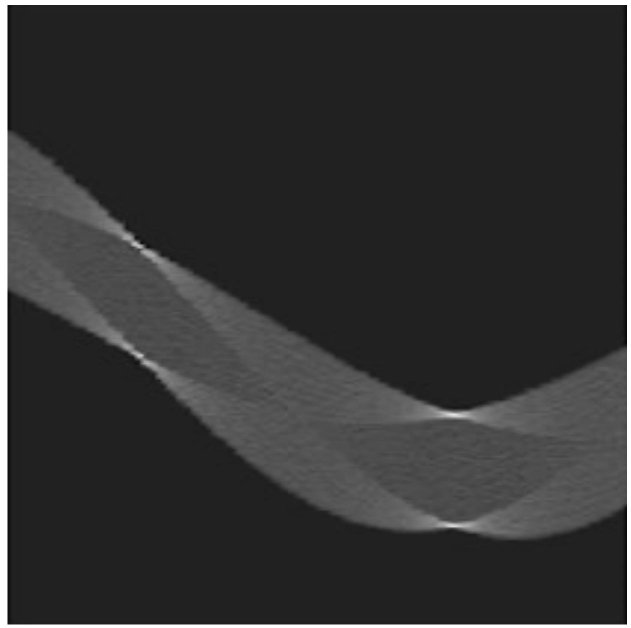
Votes

# Basic shapes

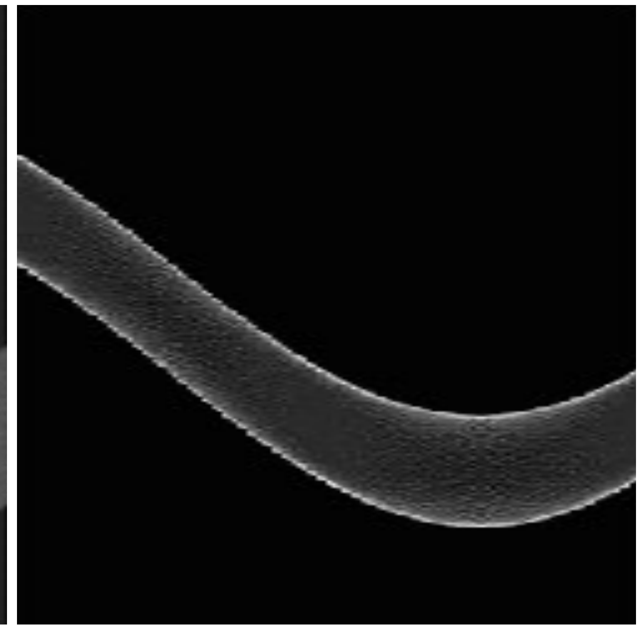
(in parameter space)



line



rectangle



circle

In practice, measurements are noisy…

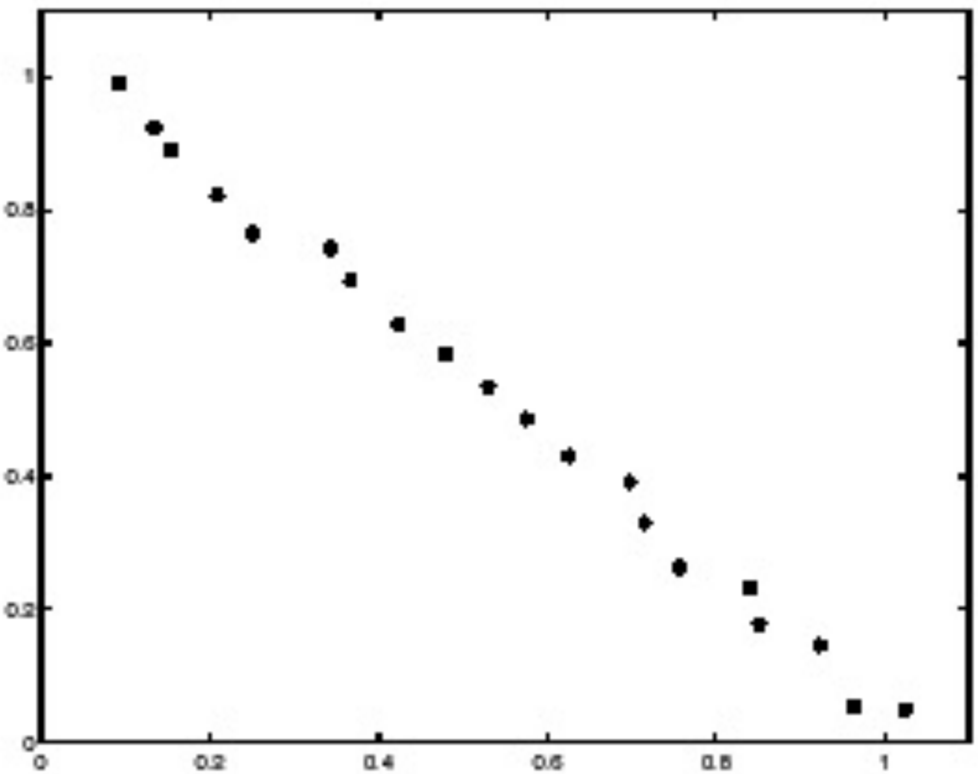
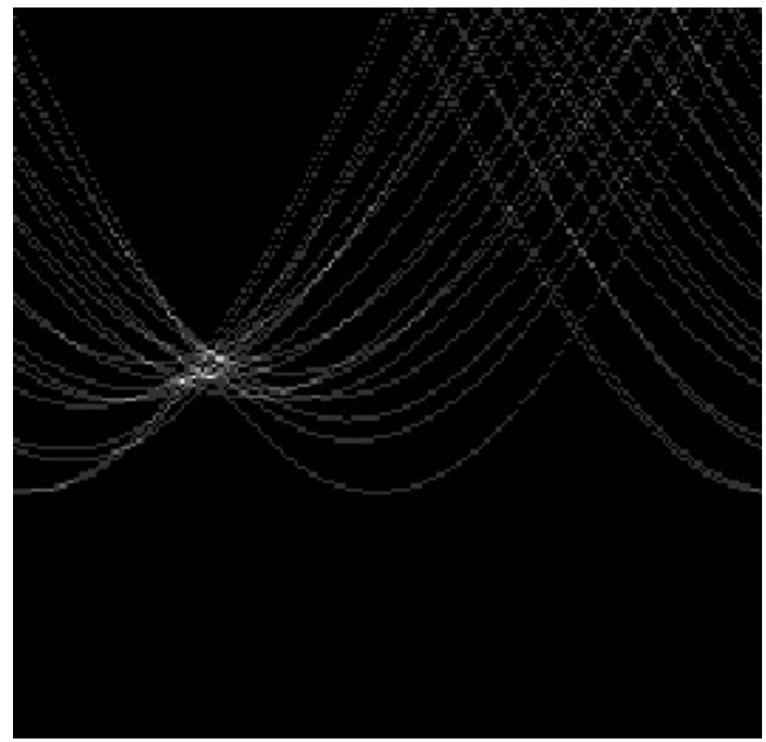



Image space



Votes

# Too much noise ...

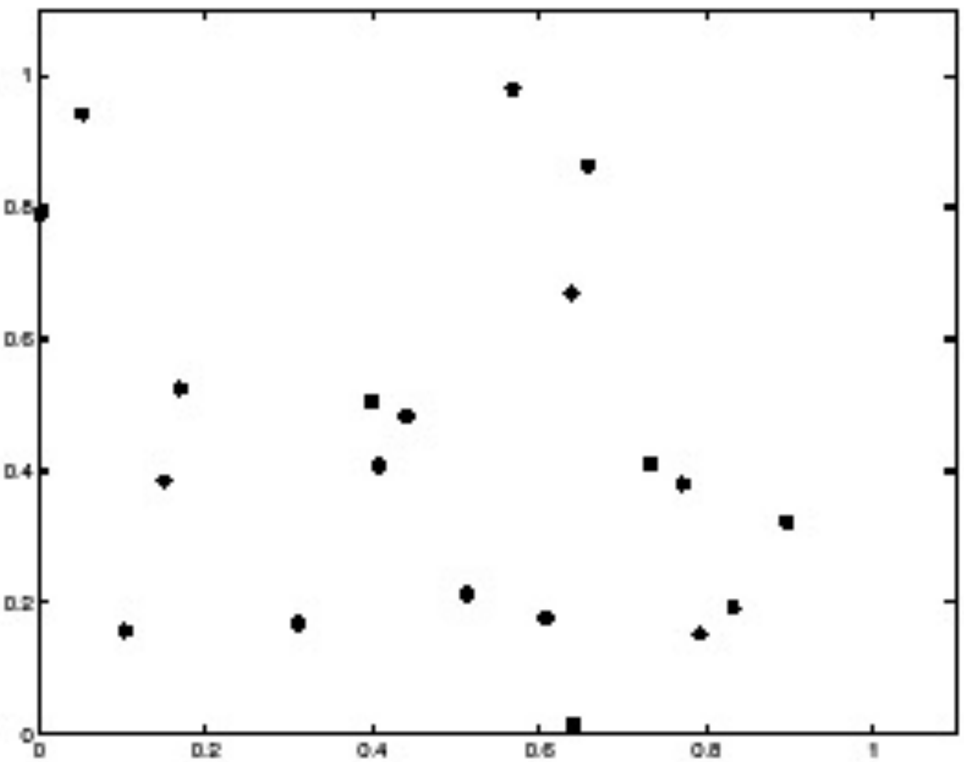
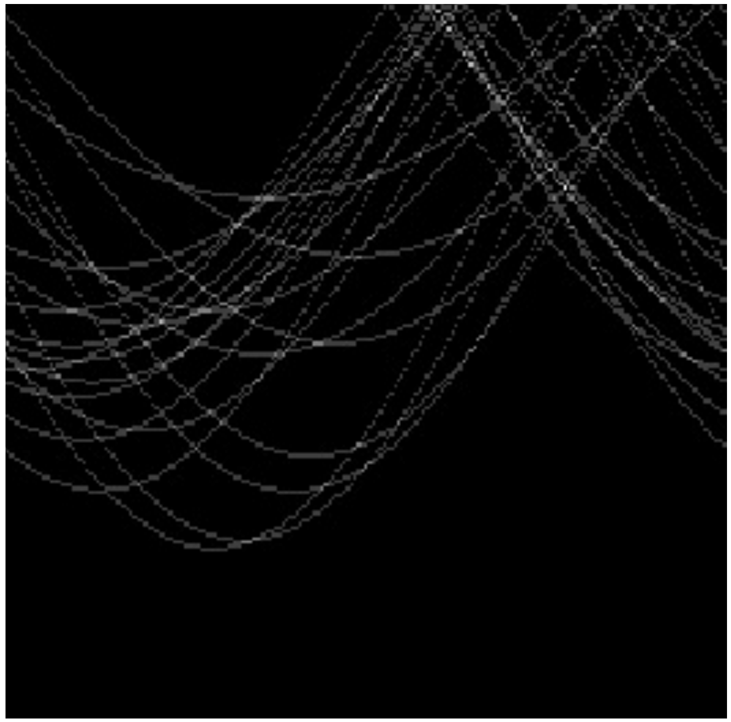



Image space

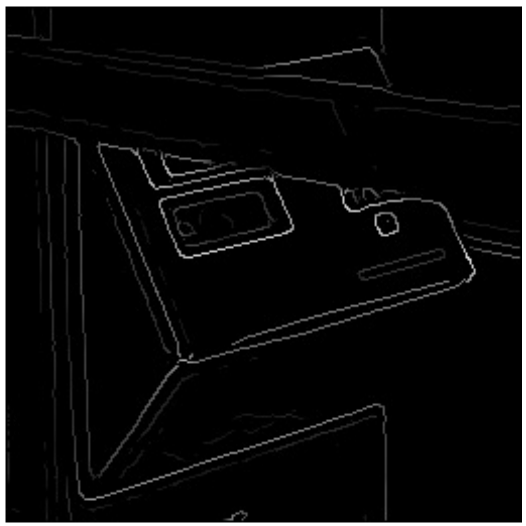


Votes

See Demo

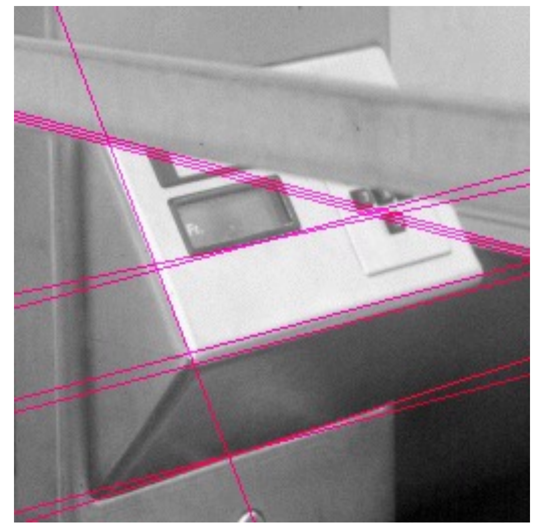
# Real-world example

Original

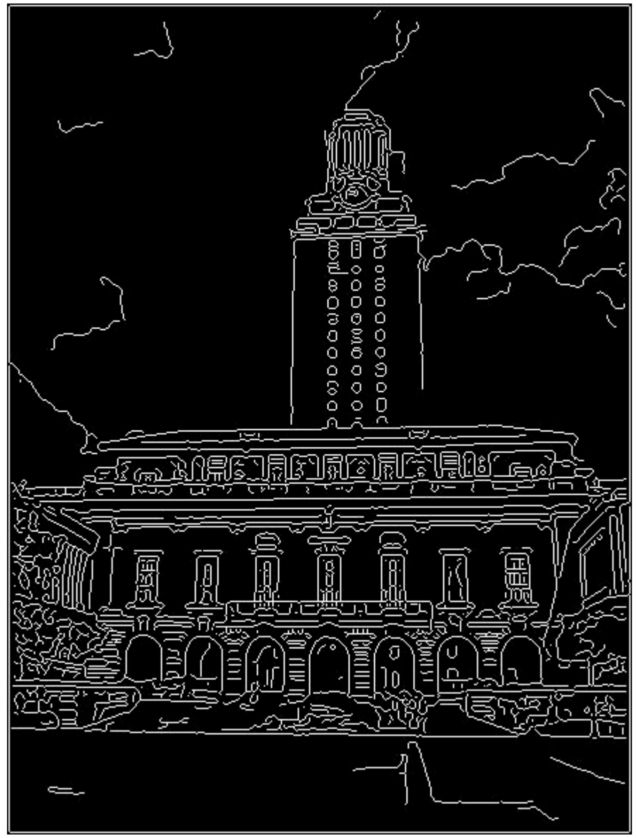


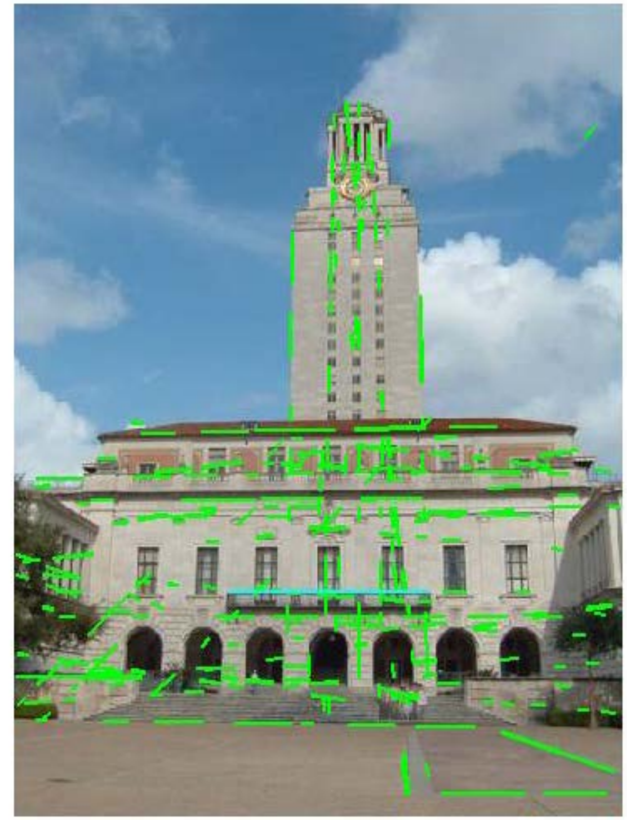
Edges

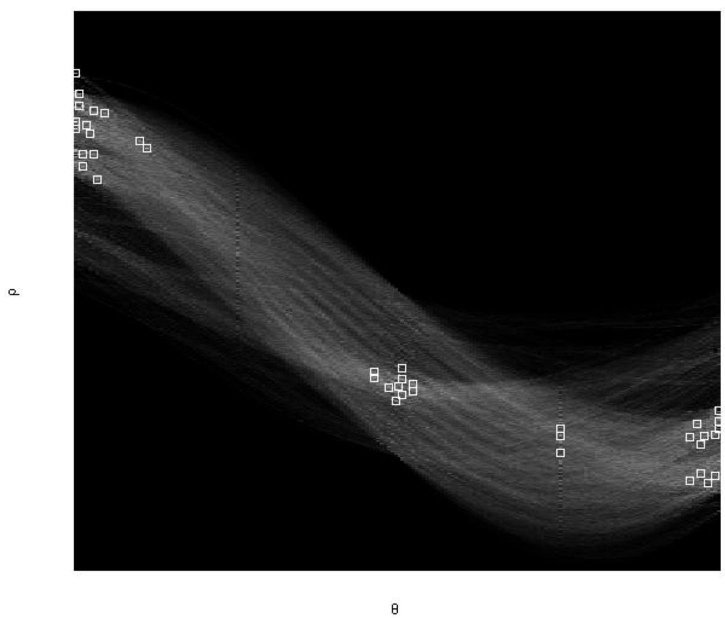
parameter space

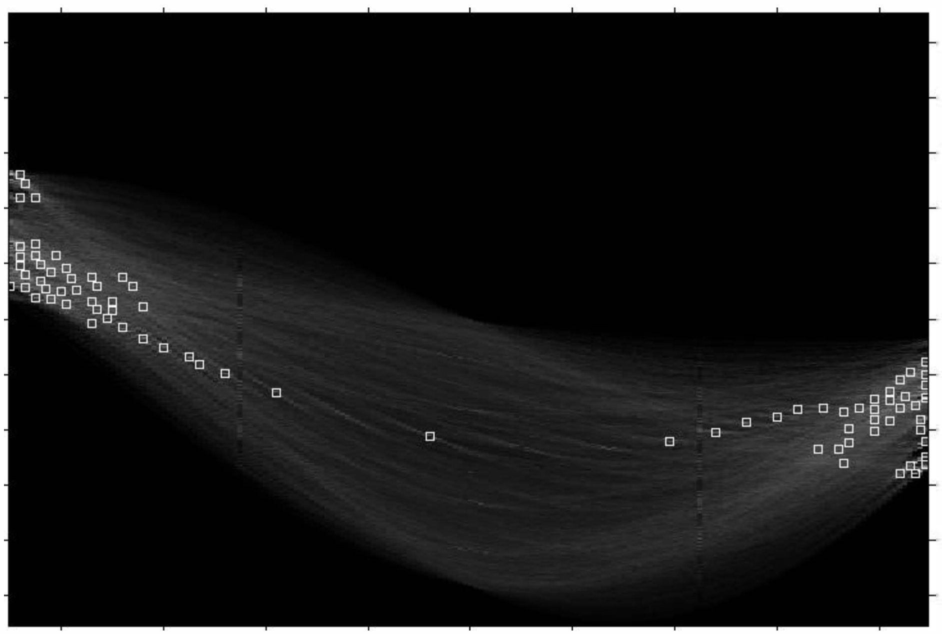


Hough Lines



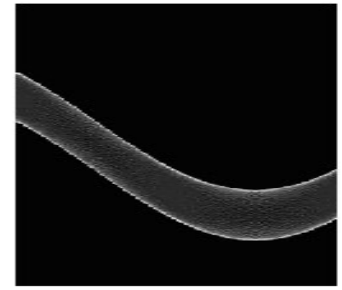



ρ
θ



**See Demo**

# Hough Circles



# Let’s assume radius known

parameters

$$(x-a)^2+(y-b)^2=r^2$$

variables

parameters

$$(x-a)^2+(y-b)^2=r^2$$

variables

*What is the dimension of the parameter space?*

parameters

$$(x - a)^2 + (y - b)^2 = r^2$$

variables

parameters

$$(x - a)^2 + (y - b)^2 = r^2$$

variables

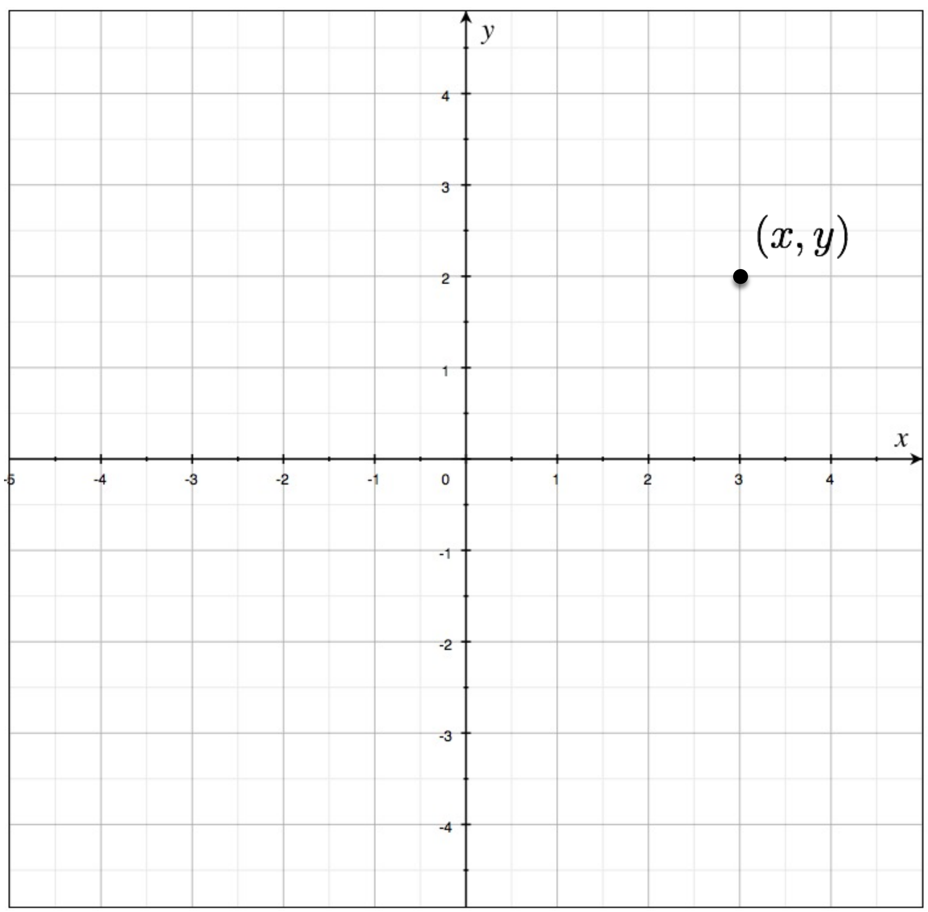
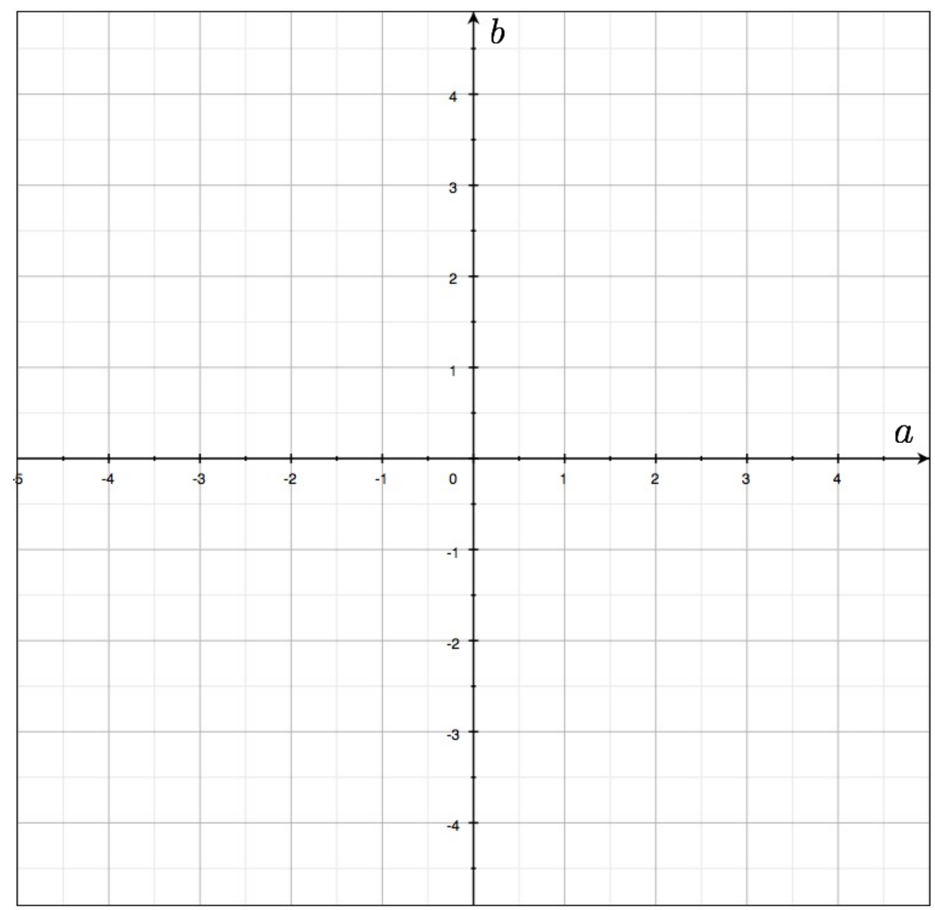



Image space




Parameter space

*What does a point in image space correspond to in parameter space?*

parameters

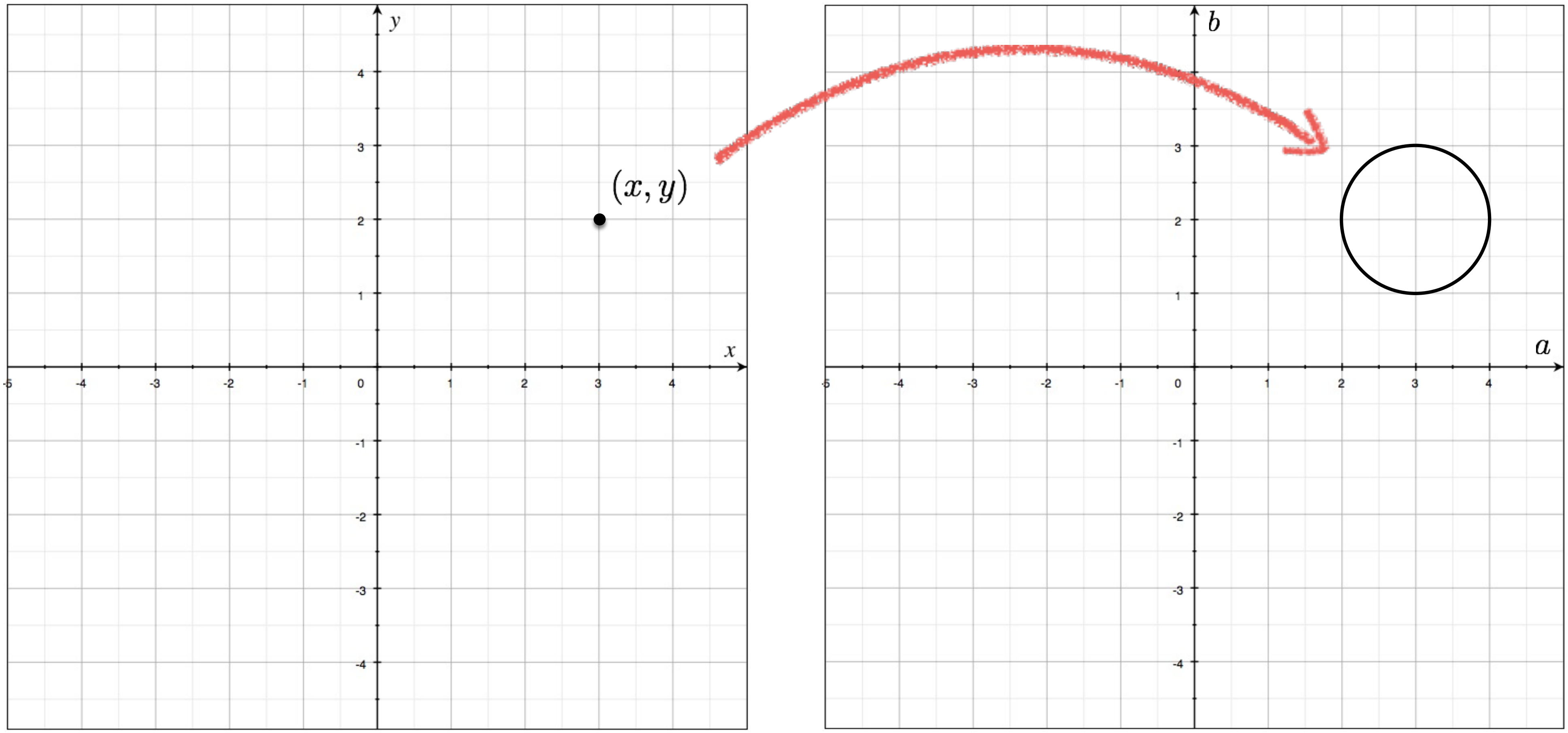
$$(x-a)^2+(y-b)^2=r^2$$

variables

parameters

$$(x-a)^2+(y-b)^2=r^2$$

variables



parameters

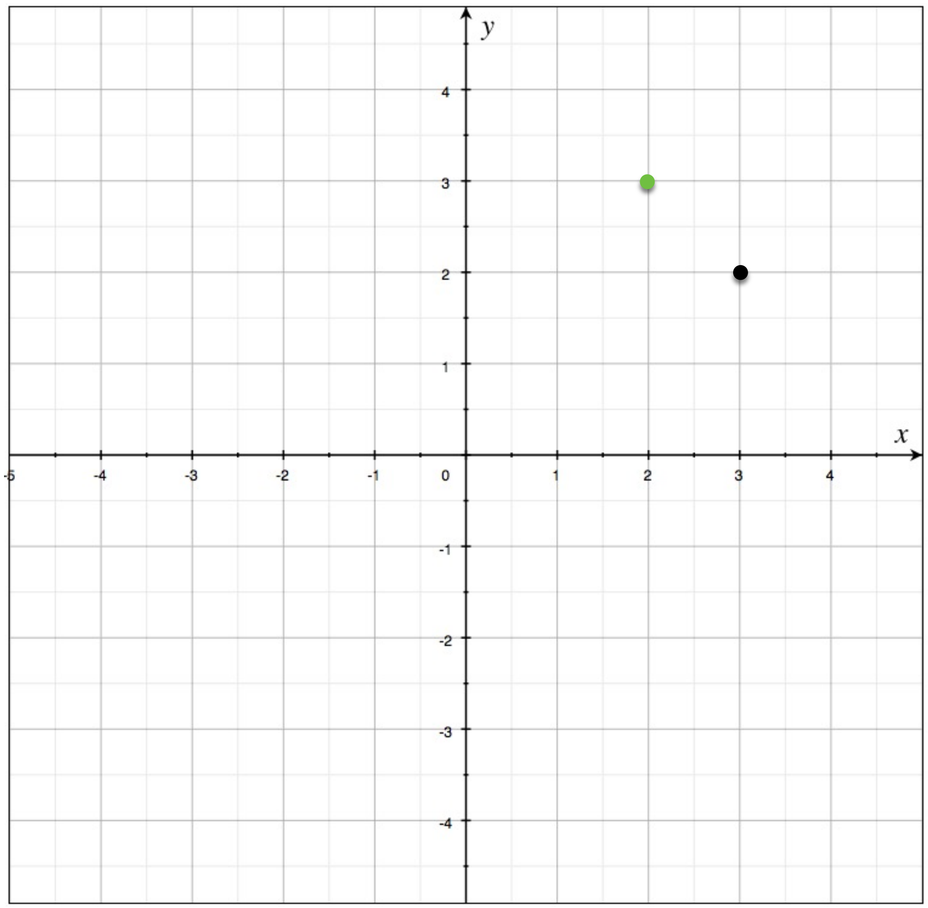
$$(x-a)^2+(y-b)^2=r^2$$

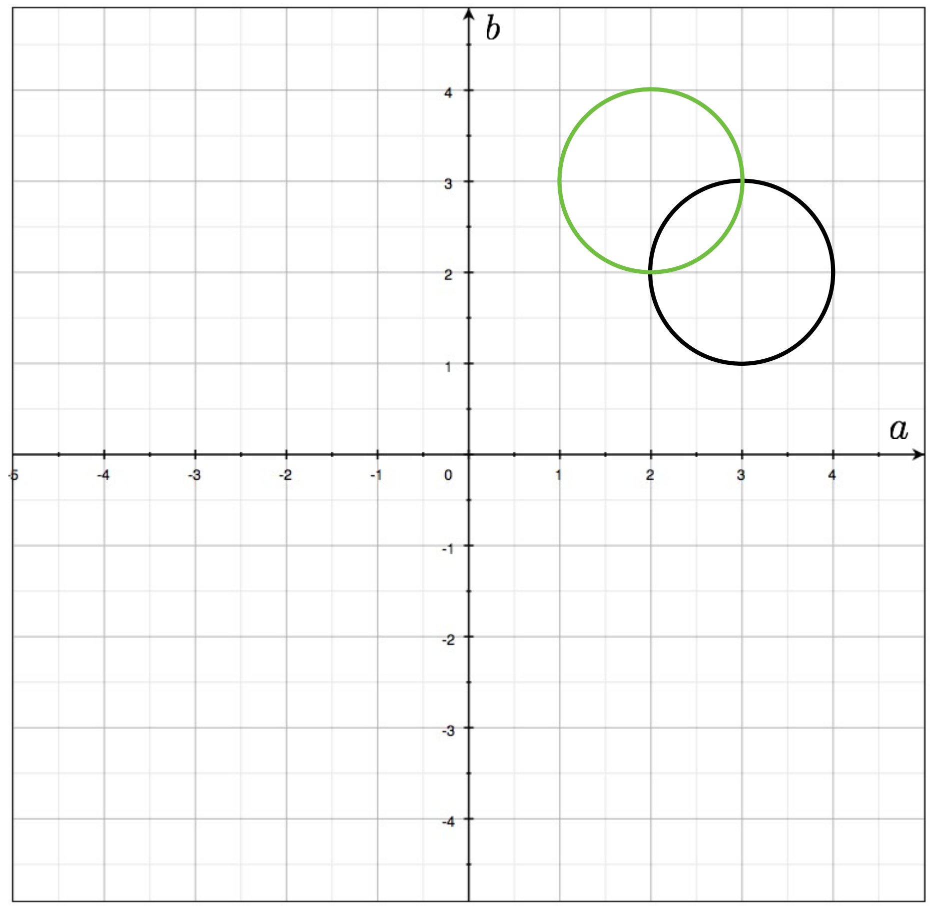
variables

parameters

$$(x-a)^2+(y-b)^2=r^2$$

variables

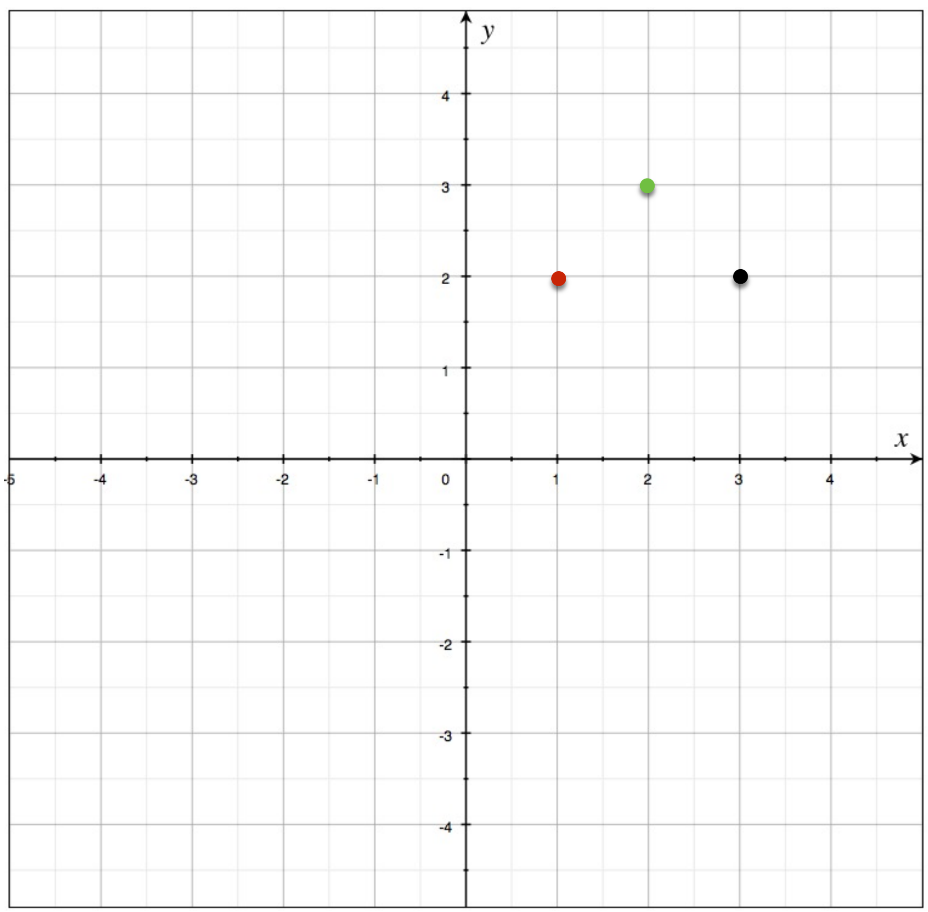


parameters

$$(x - a)^2 + (y - b)^2 = r^2$$

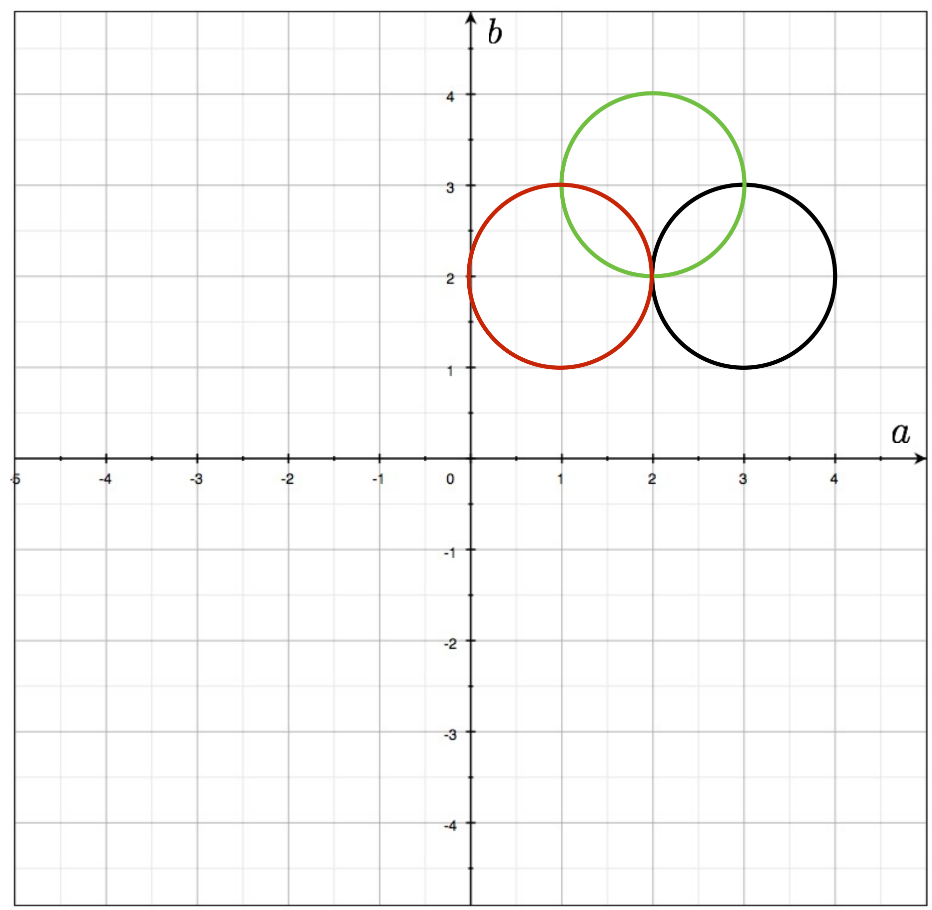
variables




parameters

$$(x - a)^2 + (y - b)^2 = r^2$$

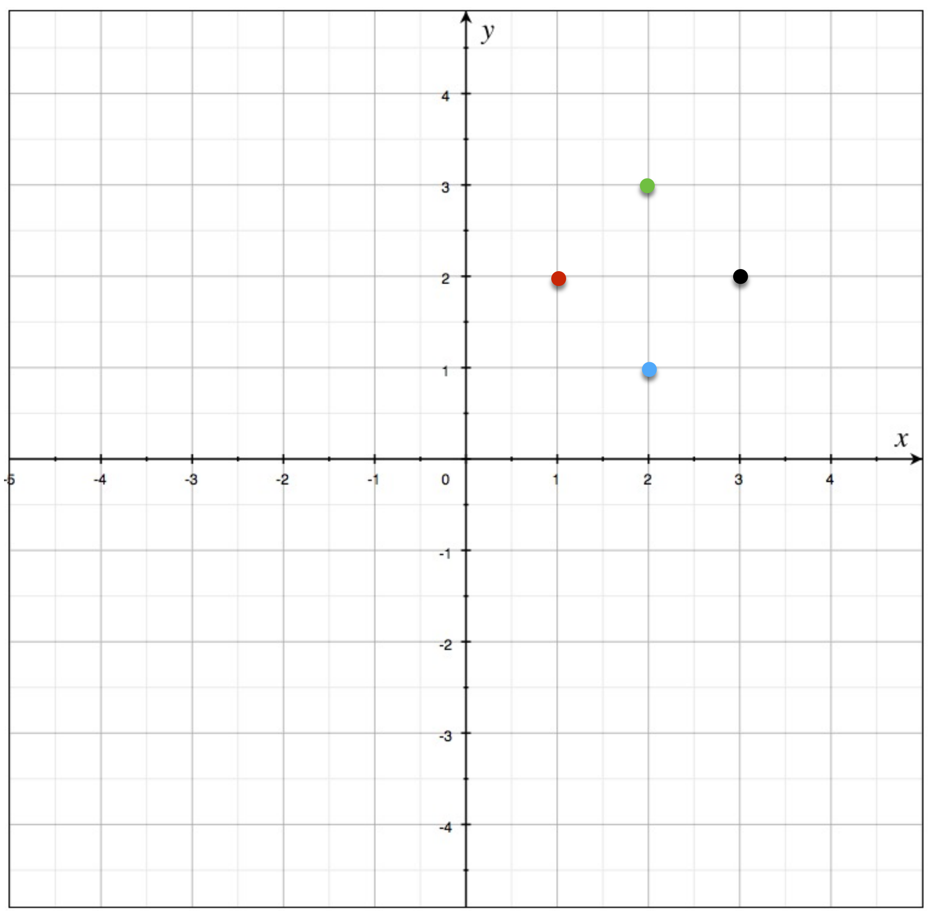
variables



parameters

$$(x-a)^2+(y-b)^2=r^2$$

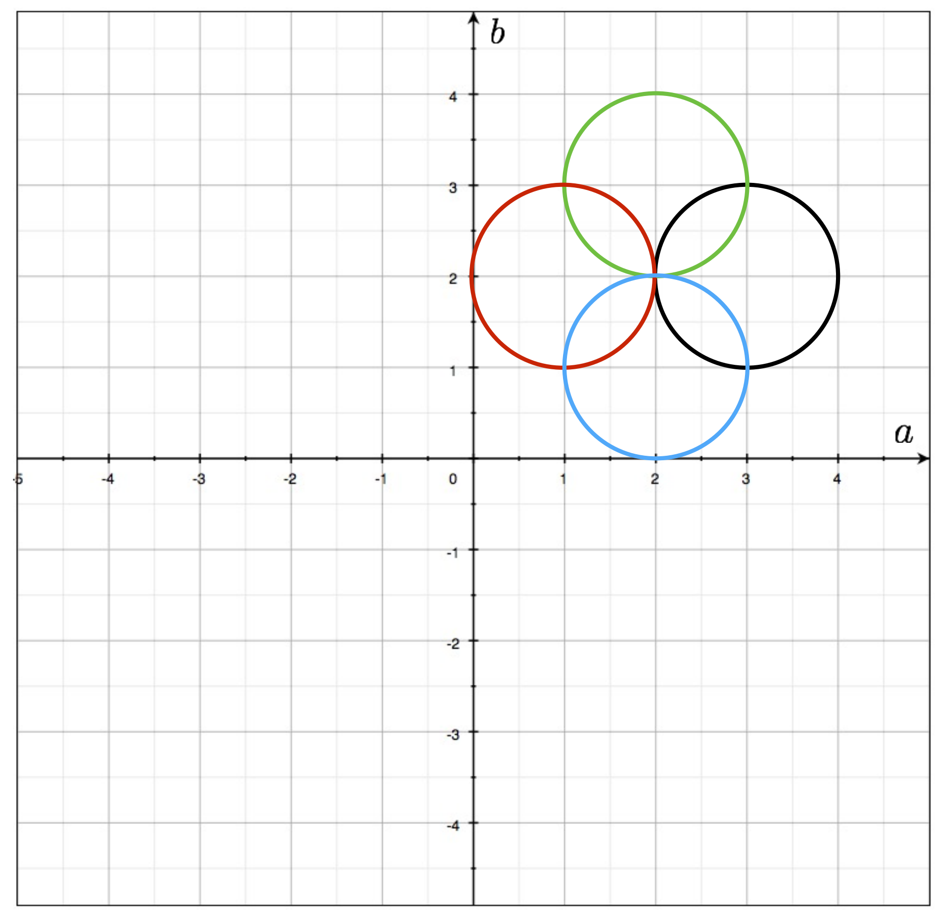
variables




parameters

$$(x-a)^2+(y-b)^2=r^2$$

variables



# What if radius is unknown?

parameters

$$(x-a)^2+(y-b)^2=r^2$$

variables

parameters

$$(x-a)^2+(y-b)^2=r^2$$

variables

# What if radius is unknown?

parameters

$$(x - a)^2 + (y - b)^2 = r^2$$

variables

parameters

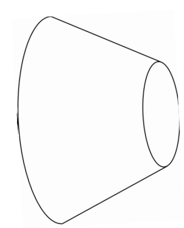
$$(x - a)^2 + (y - b)^2 = r^2$$

variables

If radius is not known: 3D Hough Space!

Use Accumulator array $A(a, b, r)$

Surface shape in Hough space is complicated

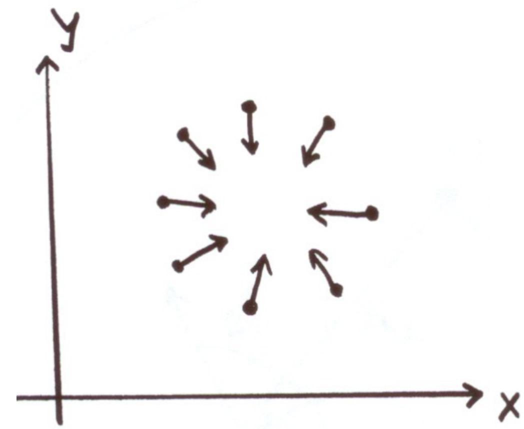


# Using Gradient Information

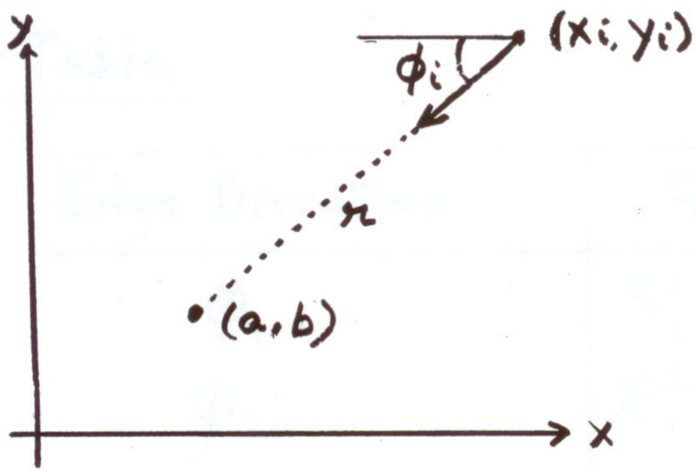
Gradient information can save lot of computation:

Edge Location $(x_i, y_i)$

Edge Direction $\phi_i$




Assume radius is known:




$$a = x - r\cos\phi$$

$$b = y - r\sin\phi$$

*Need to increment only one point in accumulator!*

parameters

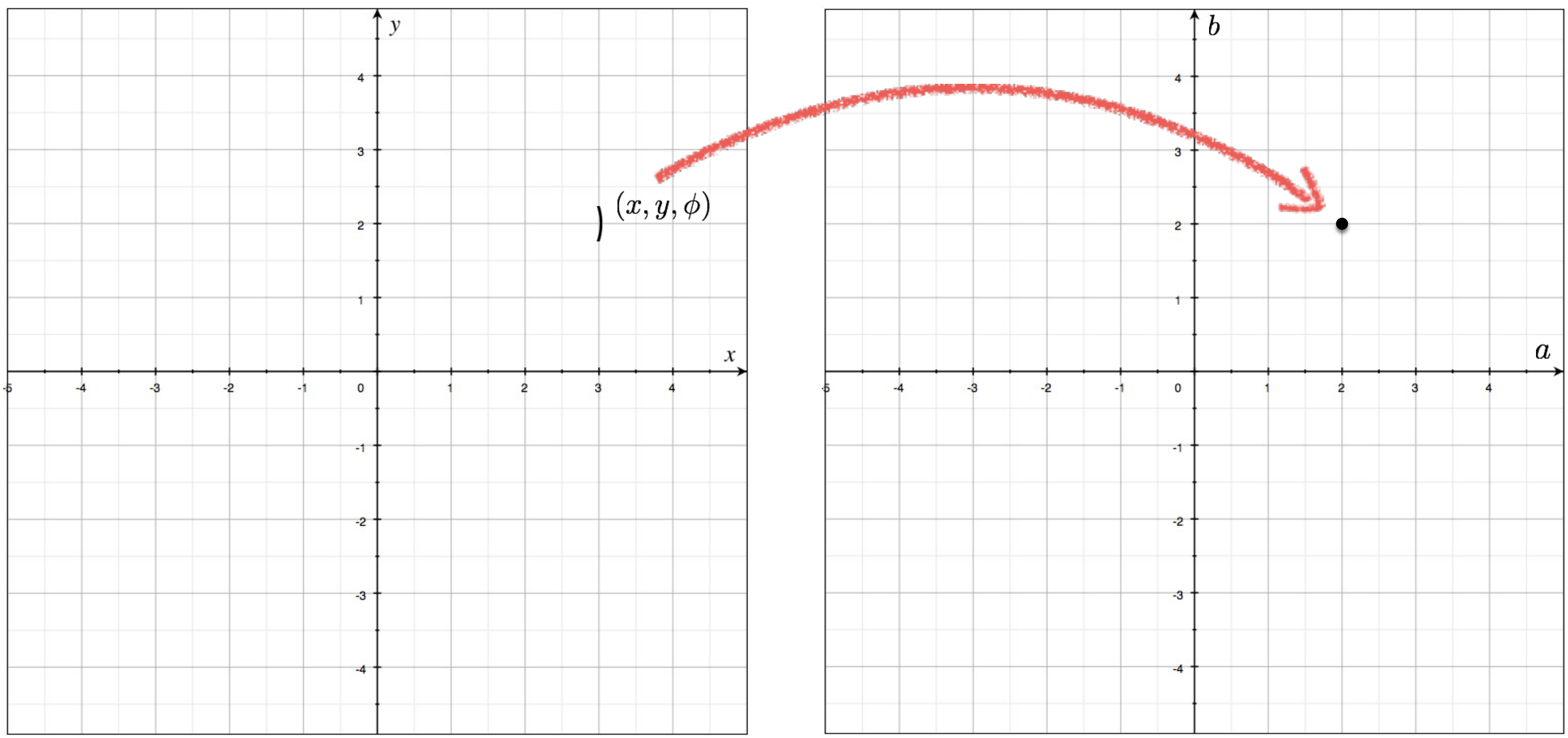
$$(x-a)^2+(y-b)^2=r^2$$

variables

parameters

$$(x-a)^2+(y-b)^2=r^2$$

variables



parameters

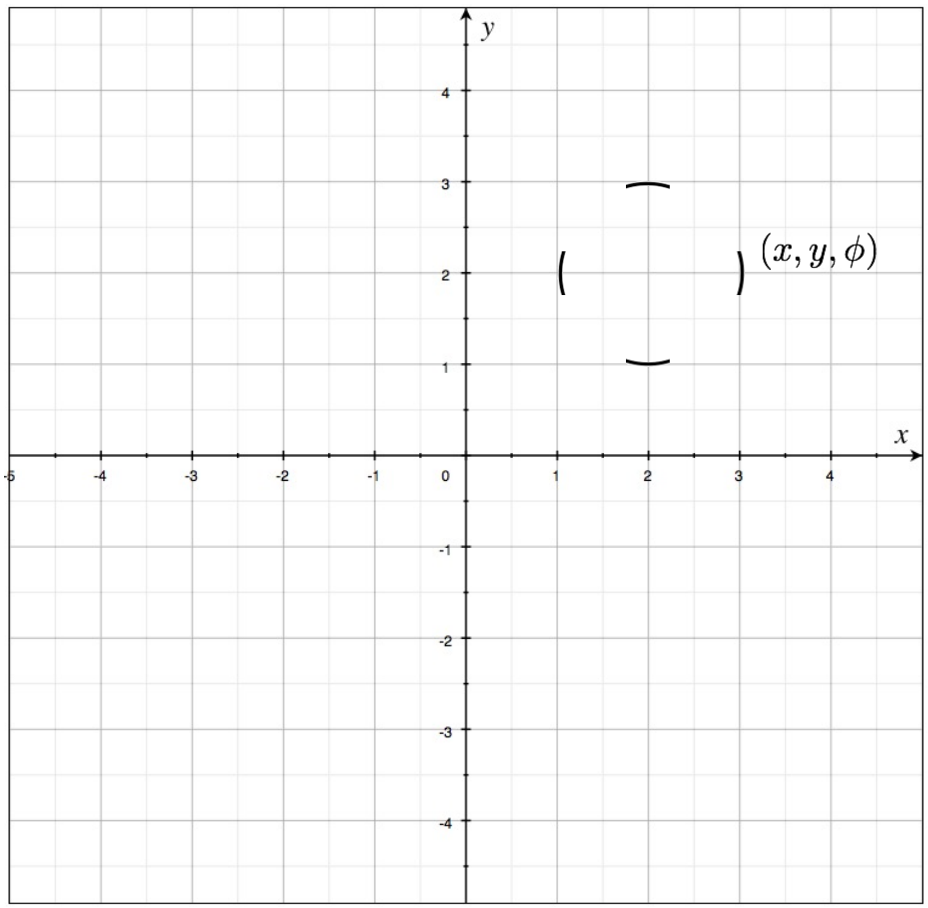
$$(x-a)^2+(y-b)^2=r^2$$

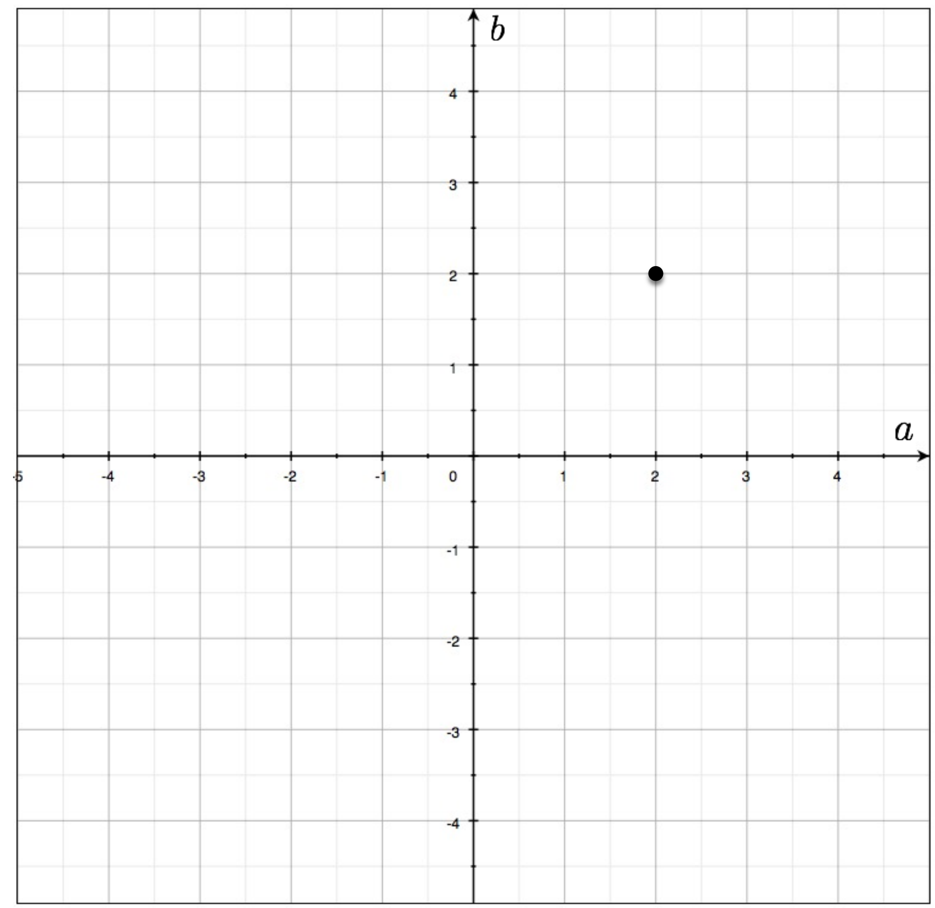
variables

parameters

$$(x-a)^2+(y-b)^2=r^2$$

variables



# Hough transform for circles

For every edge pixel $(x,y)$ :

For each possible radius value $r$:

For each possible gradient direction $\vartheta$:

*// or use estimated gradient at (x,y)*

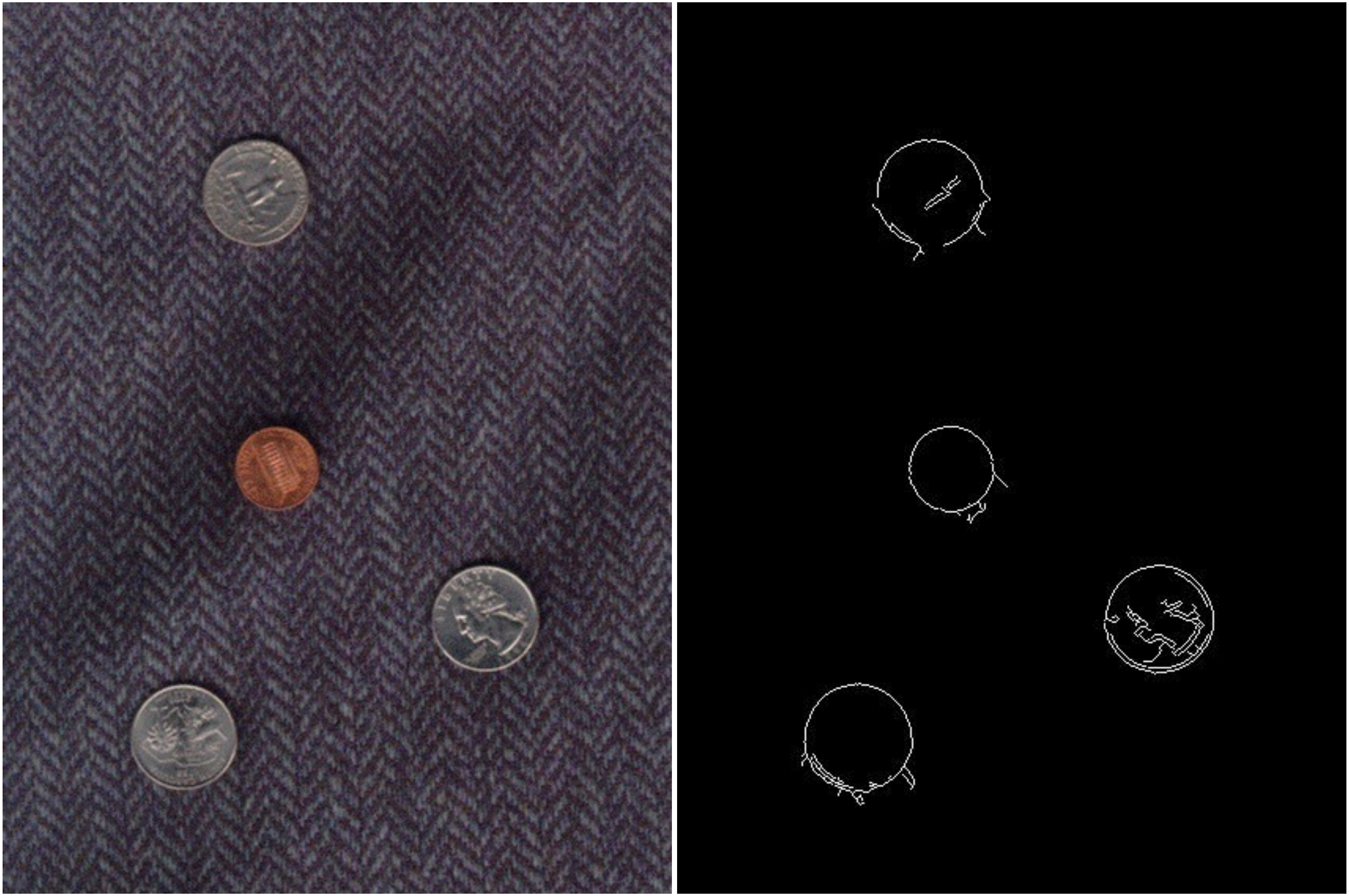
$a = x - r\cos(\vartheta)$ // column

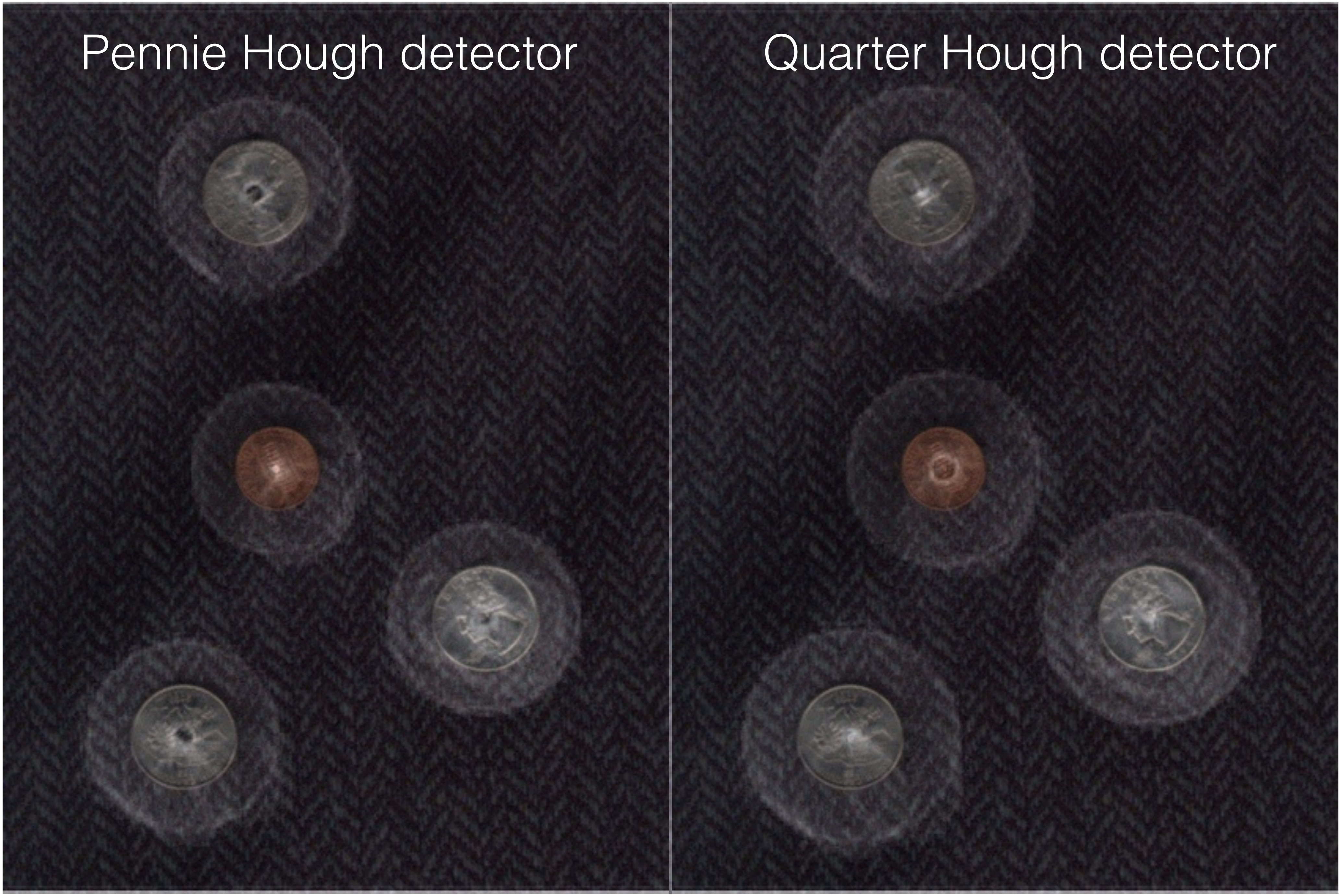
$b = y + r\sin(\vartheta)$ // row

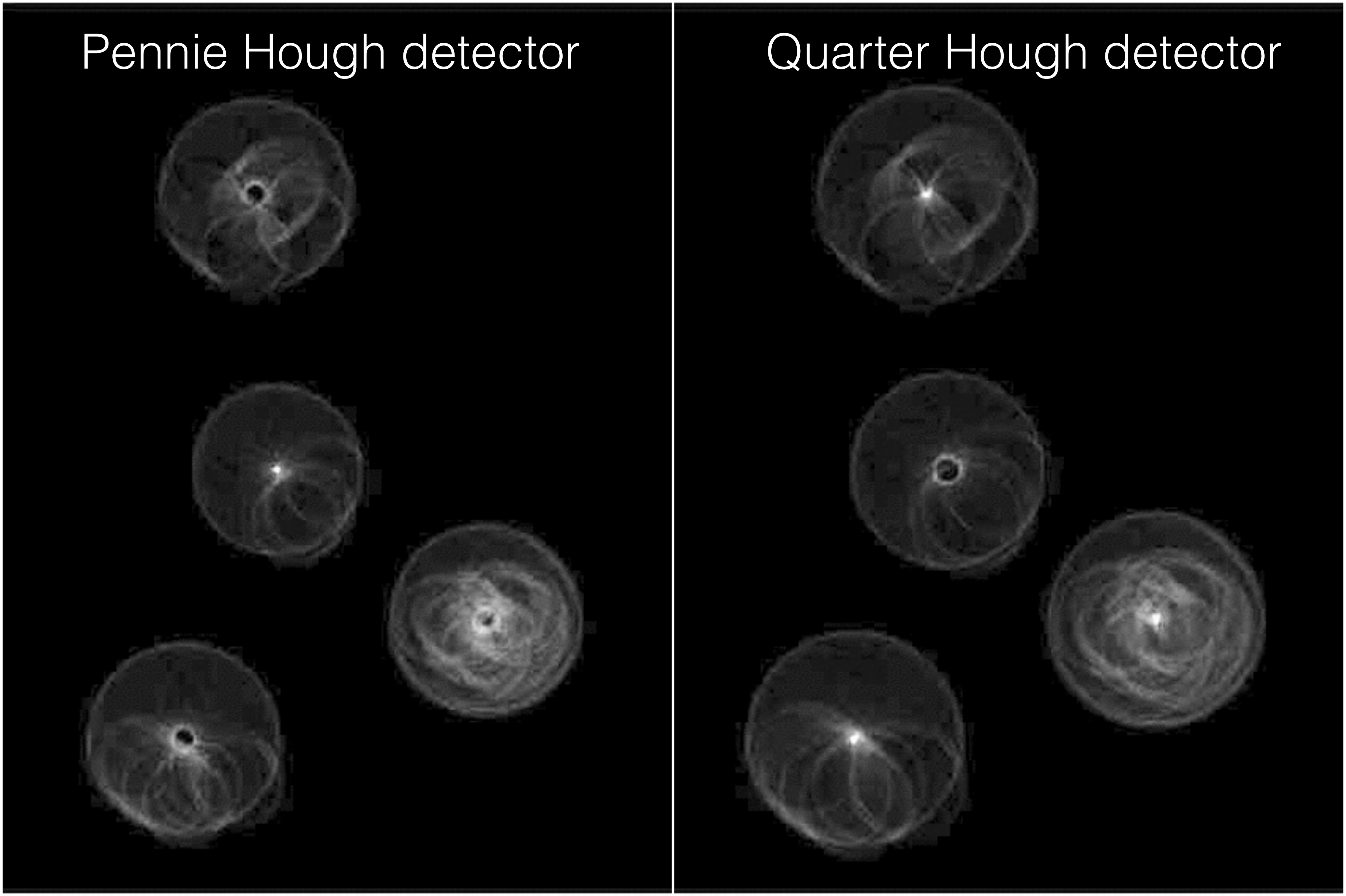
$H[a,b,r] \mathrel{+}= 1$

end

end



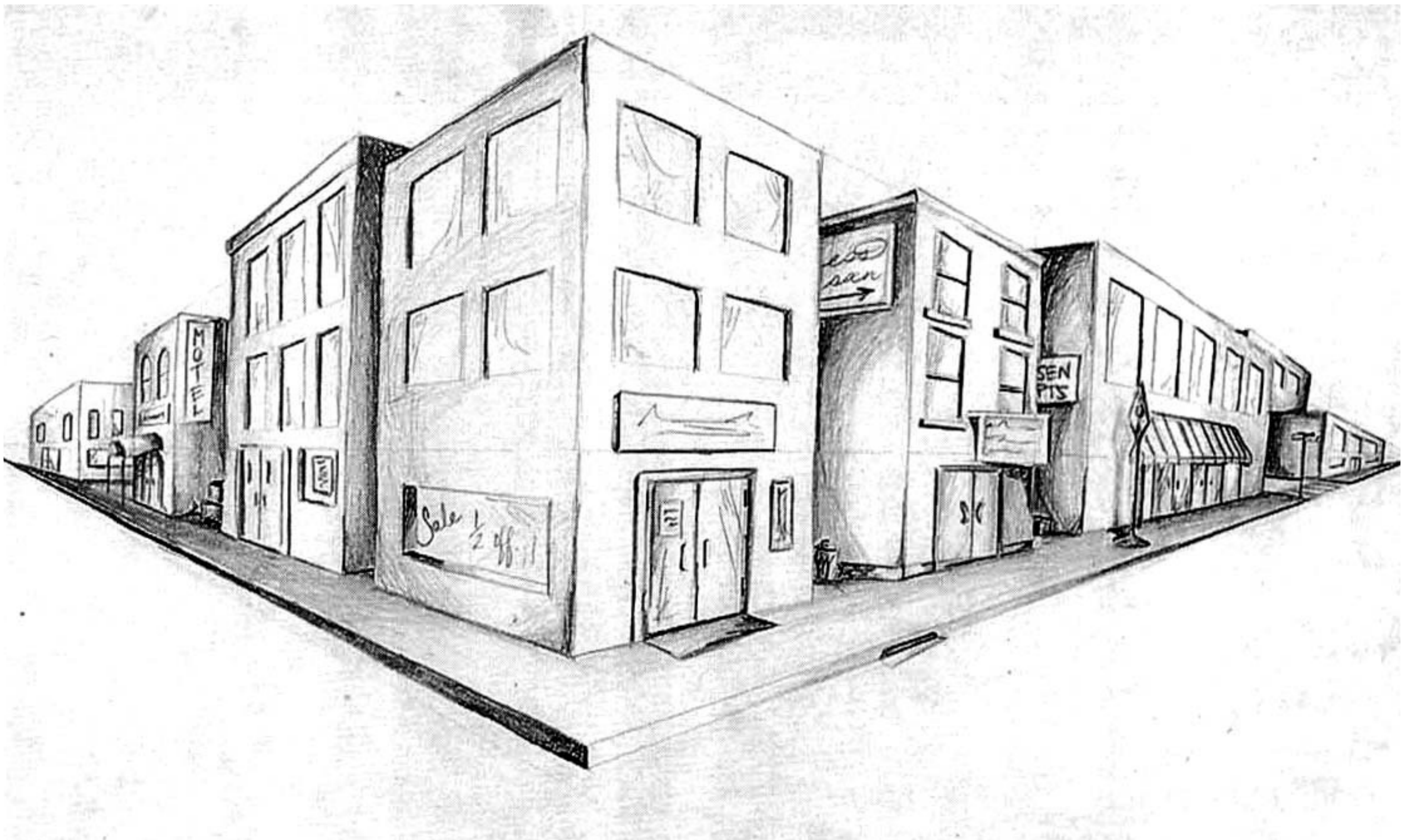

Pennie Hough detector
Quarter Hough detector


Pennie Hough detector
Quarter Hough detector

# Hough transform: pros and cons

- <u>Pros</u>
  - All points are processed independently, so can cope with occlusion
  - Some robustness to noise: noise points unlikely to contribute consistently to any single bin
  - Can detect multiple instances of a model in a single pass
- <u>Cons</u>
  - Complexity of search time increases exponentially with the number of model parameters
  - Non-target shapes can produce spurious peaks in parameter space
  - Quantization: hard to pick a good grid size

# Detecting corners




Computer Vision
Fall 2022, Lecture 5

# Why detect corners?

# Why detect corners?

Image alignment (homography, fundamental matrix)

3D reconstruction

Motion tracking

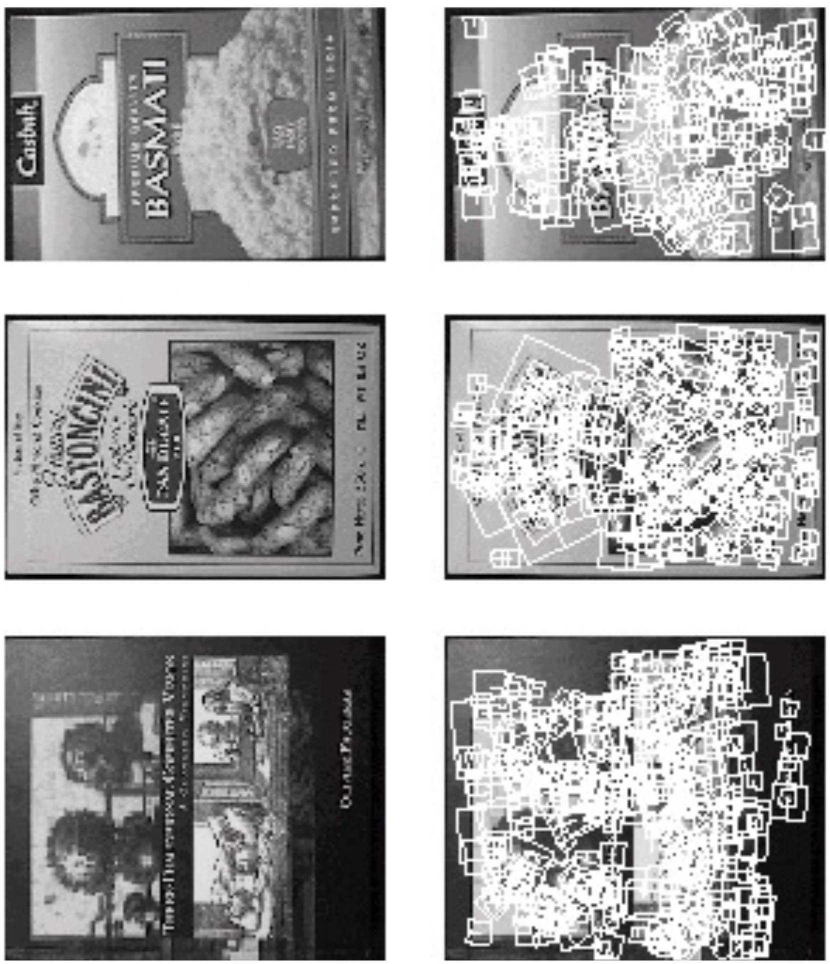
Object recognition

Indexing and database retrieval

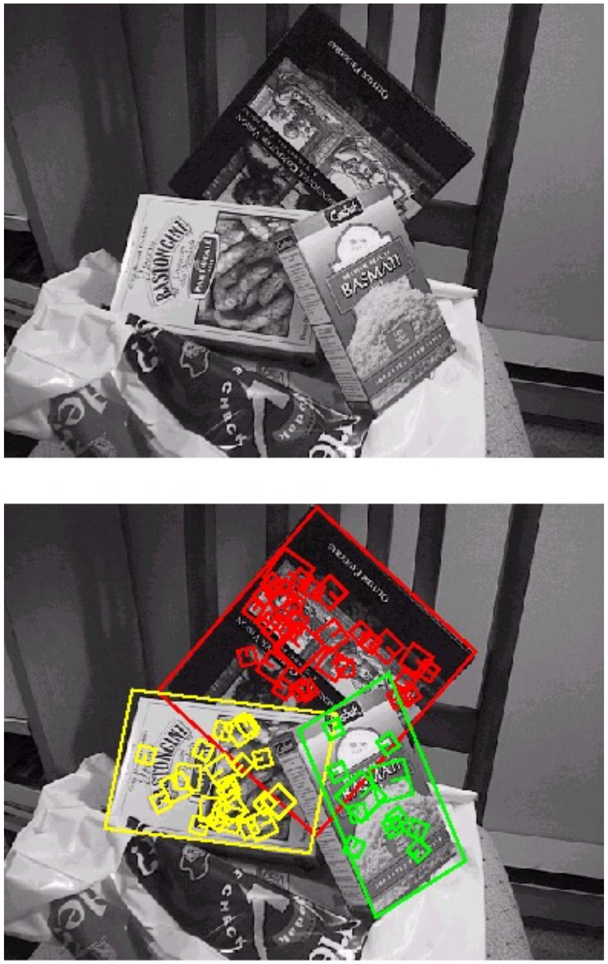
Robot navigation

# Planar object instance recognition

Database of planar objects



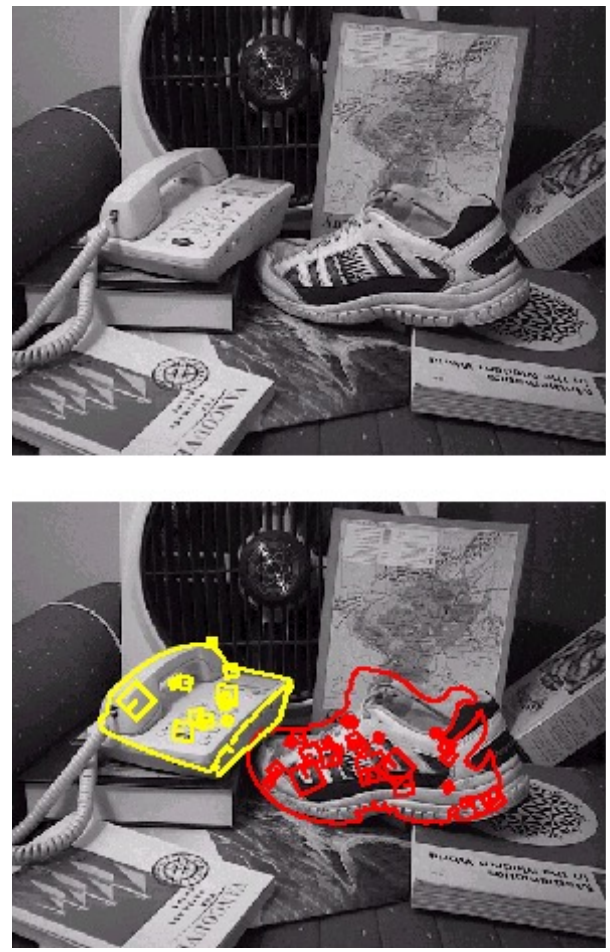
Instance recognition

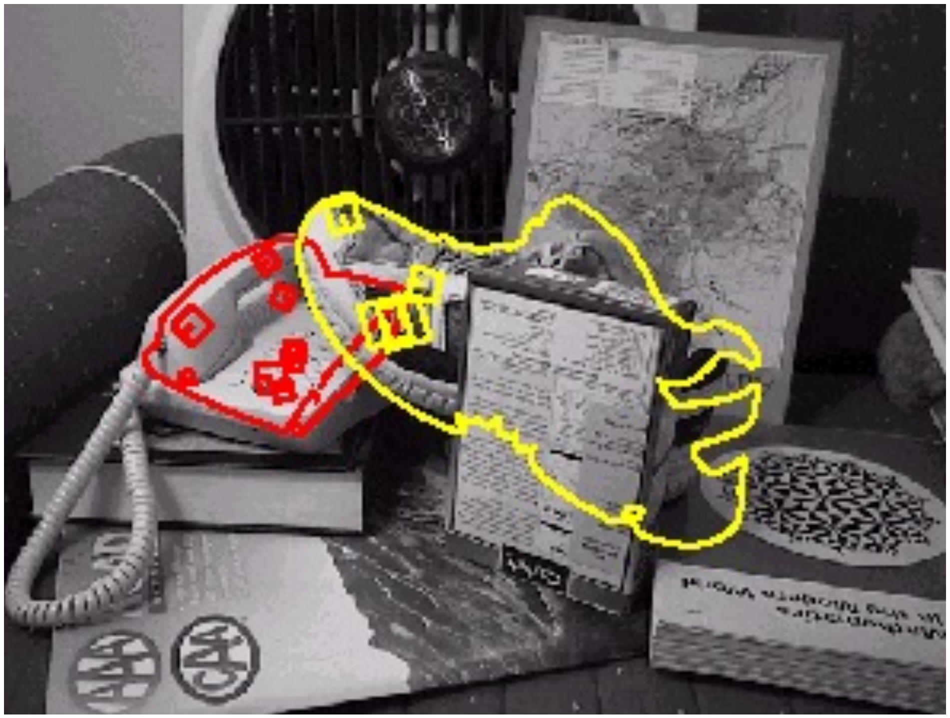


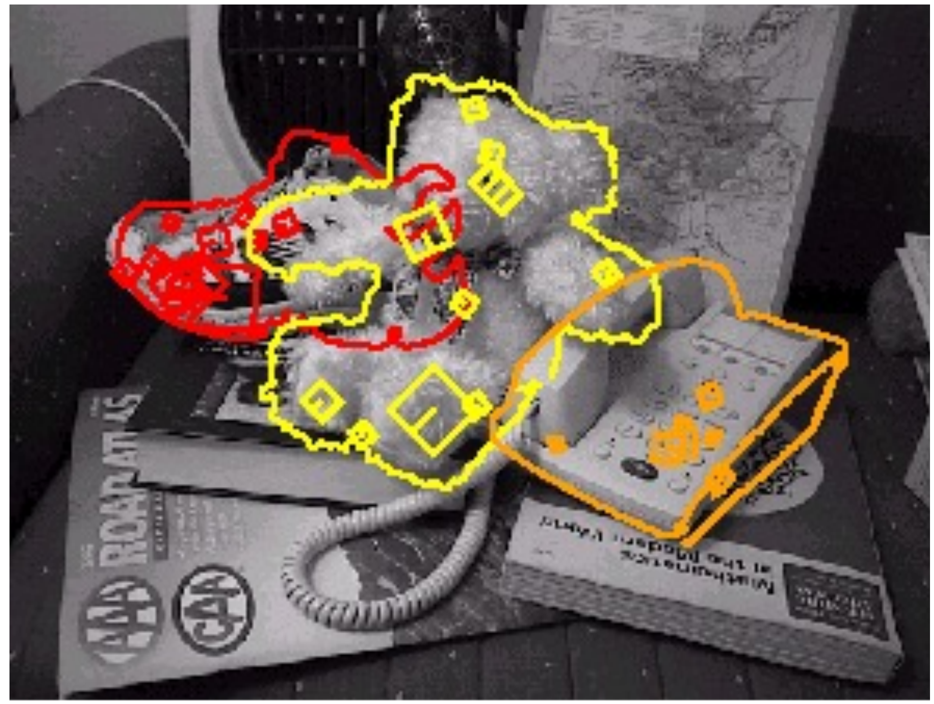
# 3D object recognition

Database of 3D objects

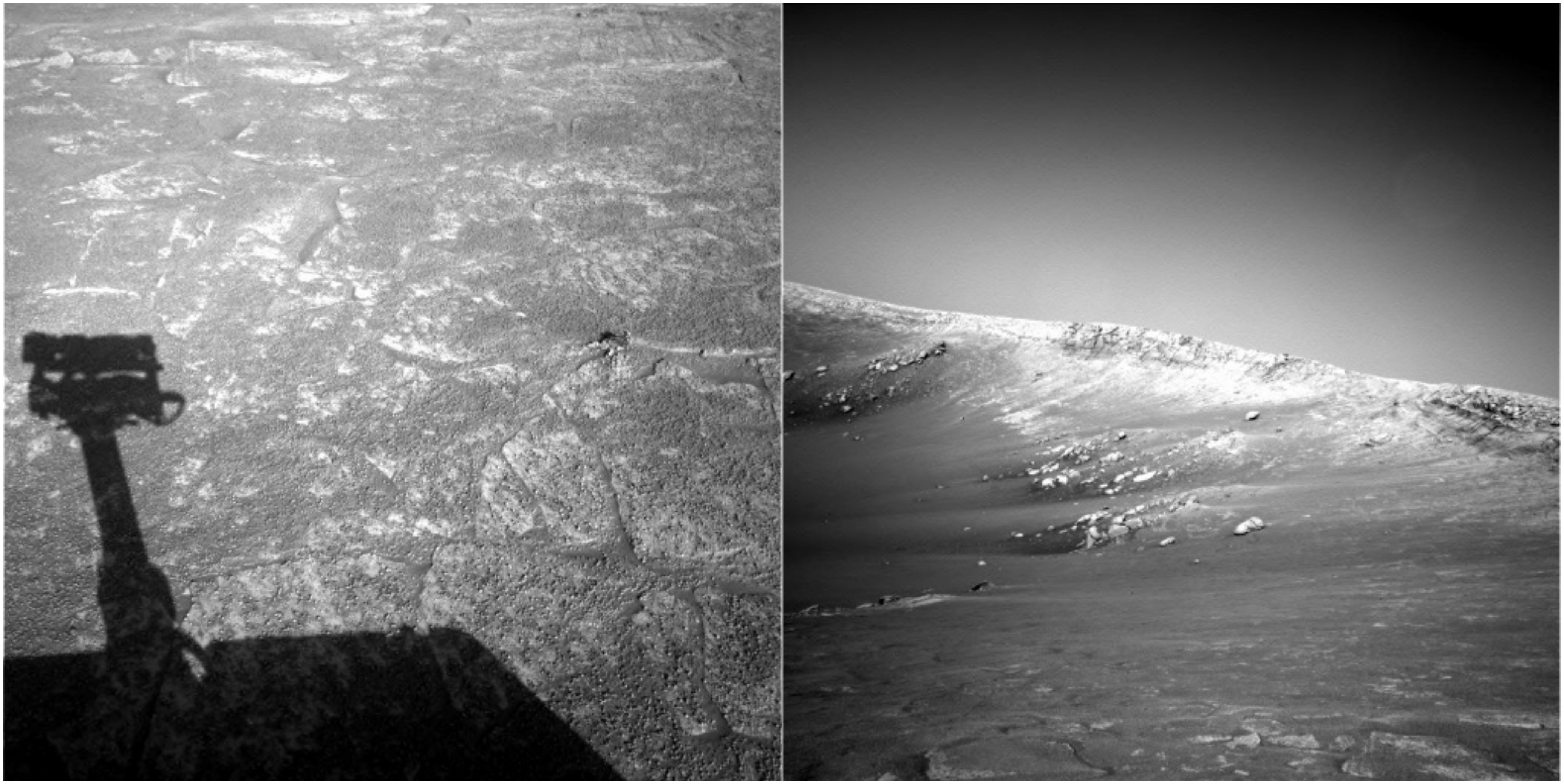
3D objects recognition





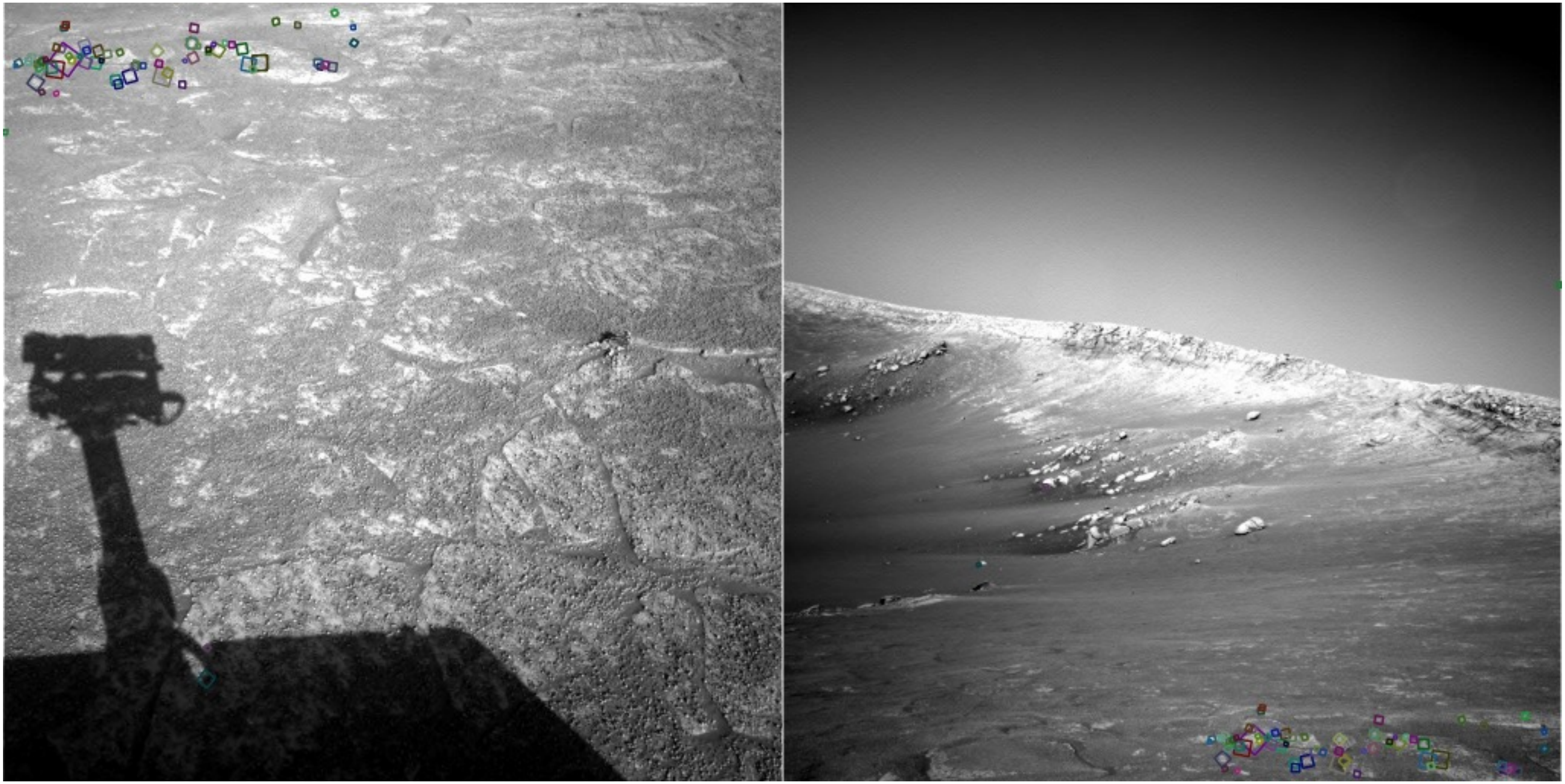


Recognition under occlusion

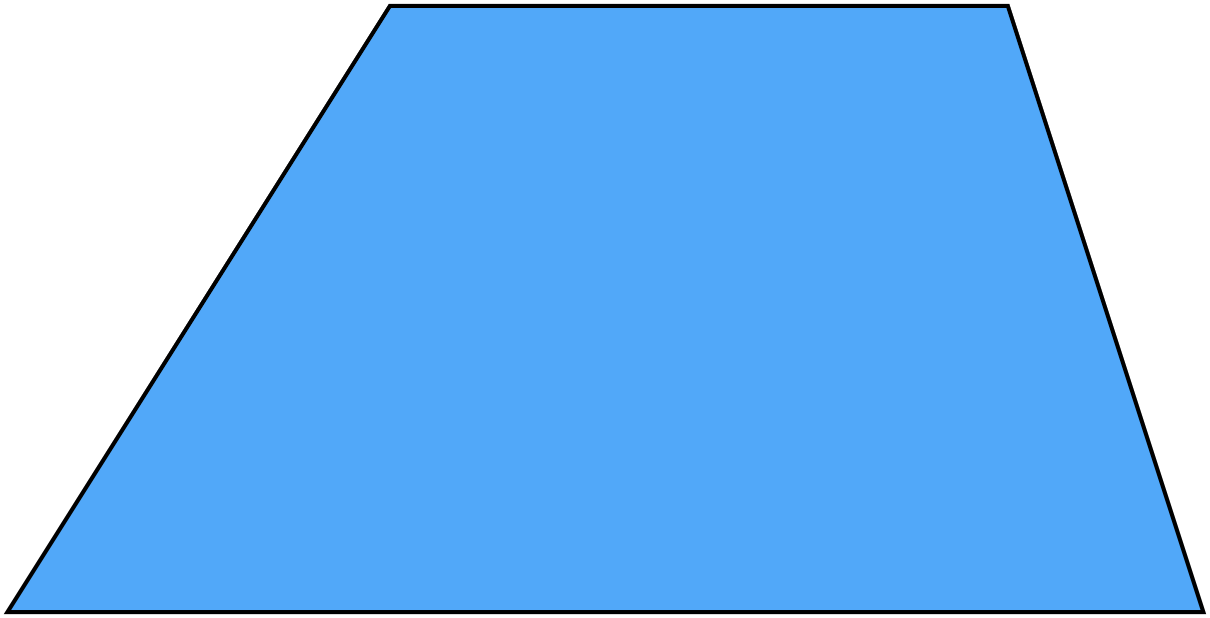


NASA Mars Rover images

*Where are the corresponding points?*

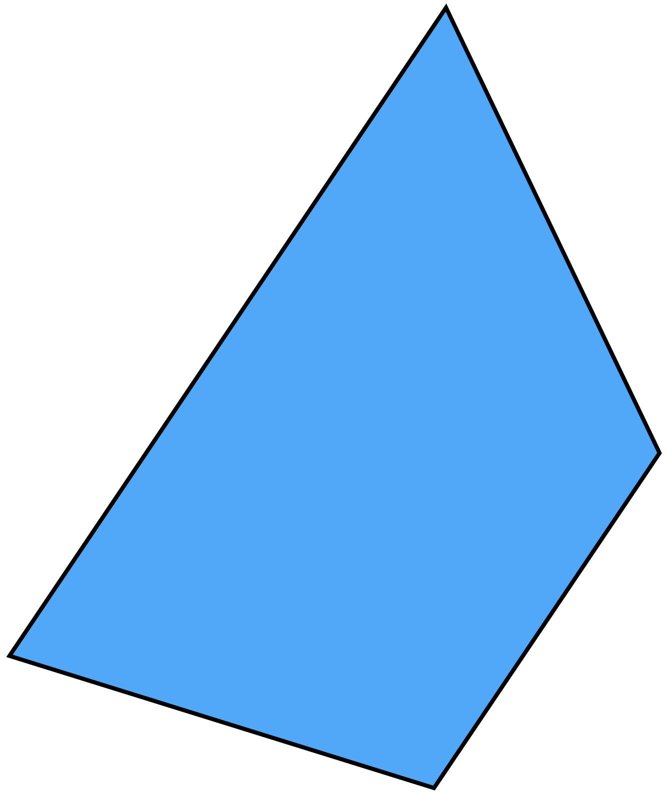


NASA Mars Rover images



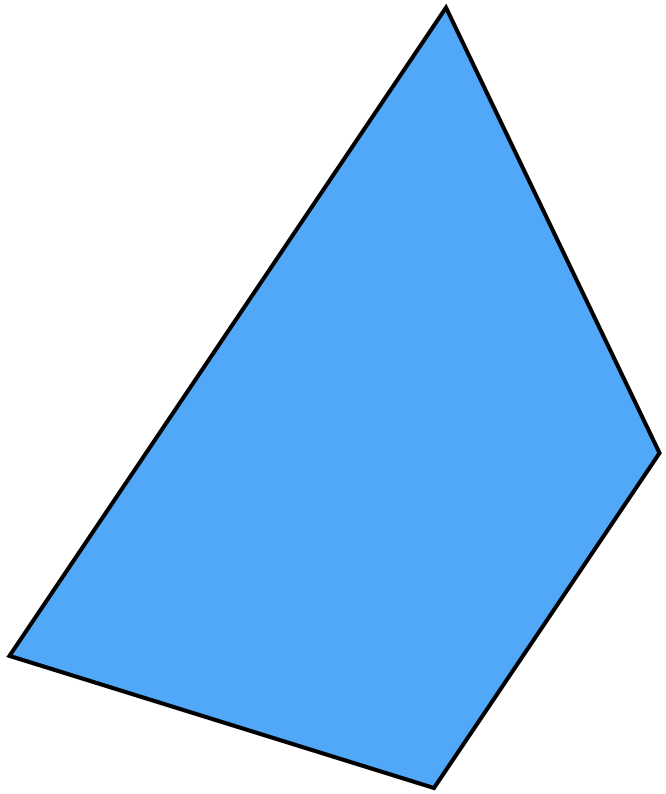
Pick a point in the image.
Find it again in the next image.

*What type of feature would you select?*



Pick a point in the image.
Find it again in the next image.

*What type of feature would you select?*

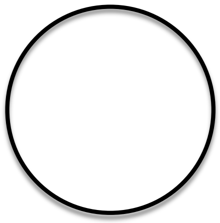


Pick a point in the image.
Find it again in the next image.

*What type of feature would you select?*

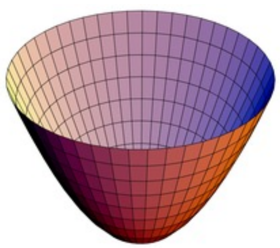
a corner

# Visualizing quadratics



Equation of a circle

$$1 = x^2 + y^2$$



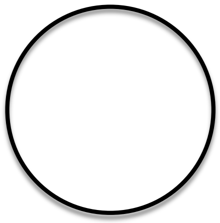
Equation of a 'bowl' (paraboloid)

$$f(x, y) = x^2 + y^2$$

*If you slice the bowl at*

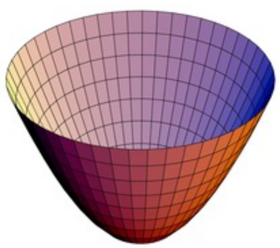
$$f(x, y) = 1$$

*what do you get?*



Equation of a circle

$$1 = x^2 + y^2$$



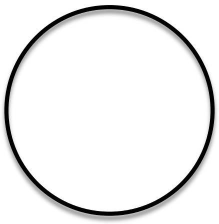
Equation of a 'bowl' (paraboloid)

$$f(x, y) = x^2 + y^2$$

*If you slice the bowl at*

$$f(x, y) = 1$$

*what do you get?*



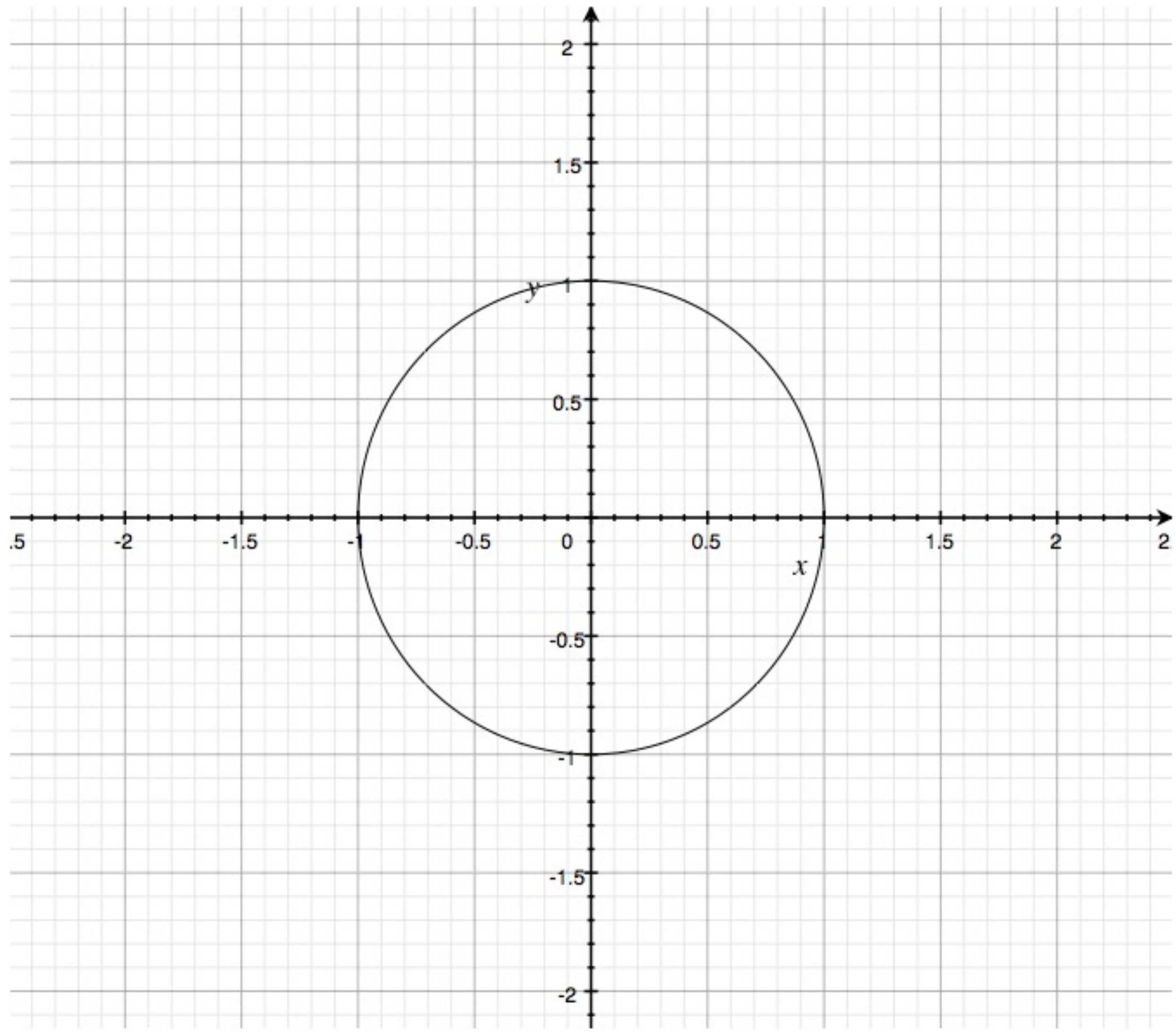
$$f(x, y) = x^2 + y^2$$

can be written in matrix form like this…

$$f(x, y) = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$f(x,y) = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

'sliced at 1'



*What happens if you* ***increase*** *coefficient on* ***x****?*

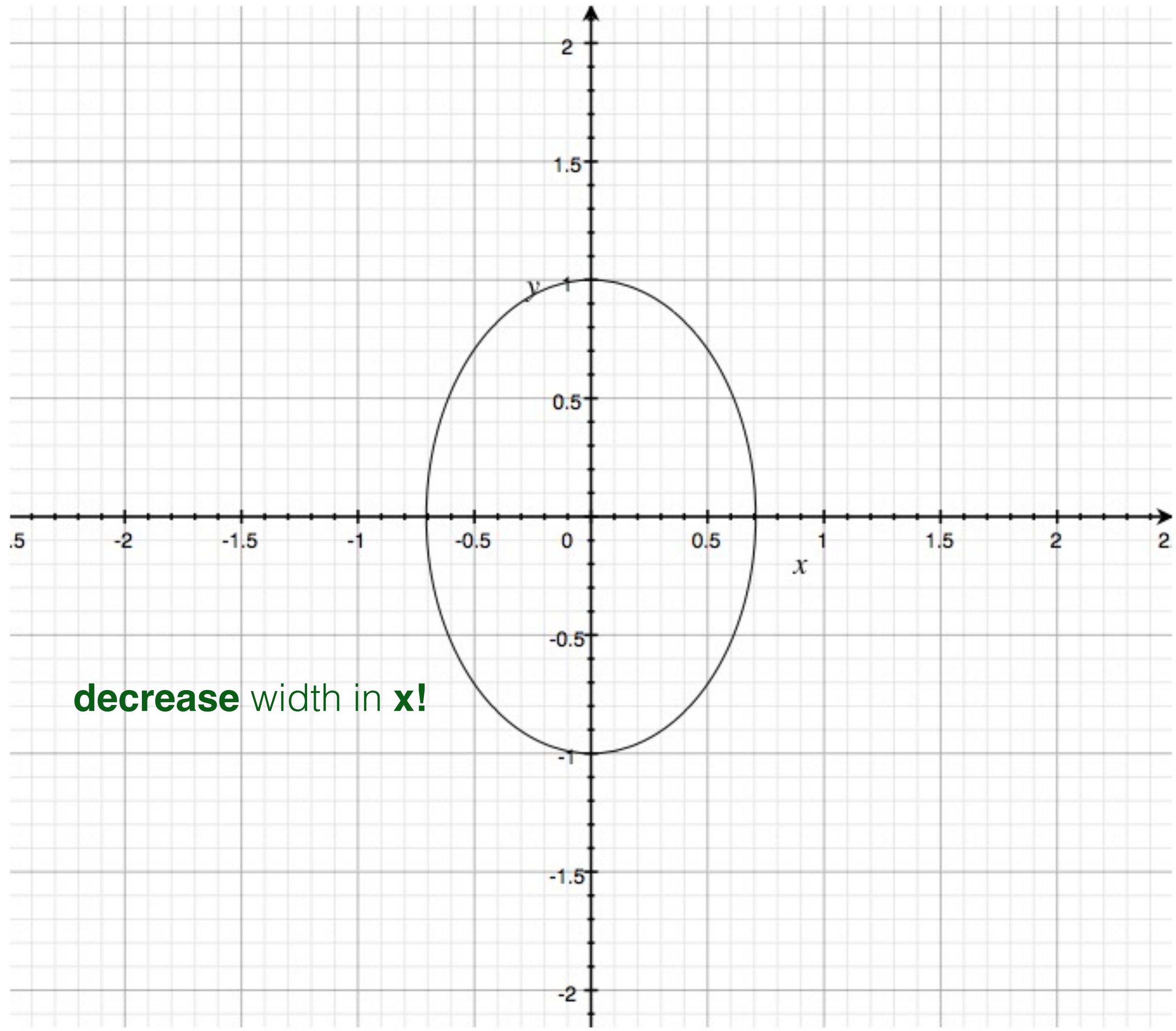
$$f(x,y) = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

and slice at 1

*What happens if you **increase** coefficient on **x**?*

$$f(x,y) = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

and slice at 1



*What happens if you* ***increase*** *coefficient on* ***y****?*

$$f(x,y) = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

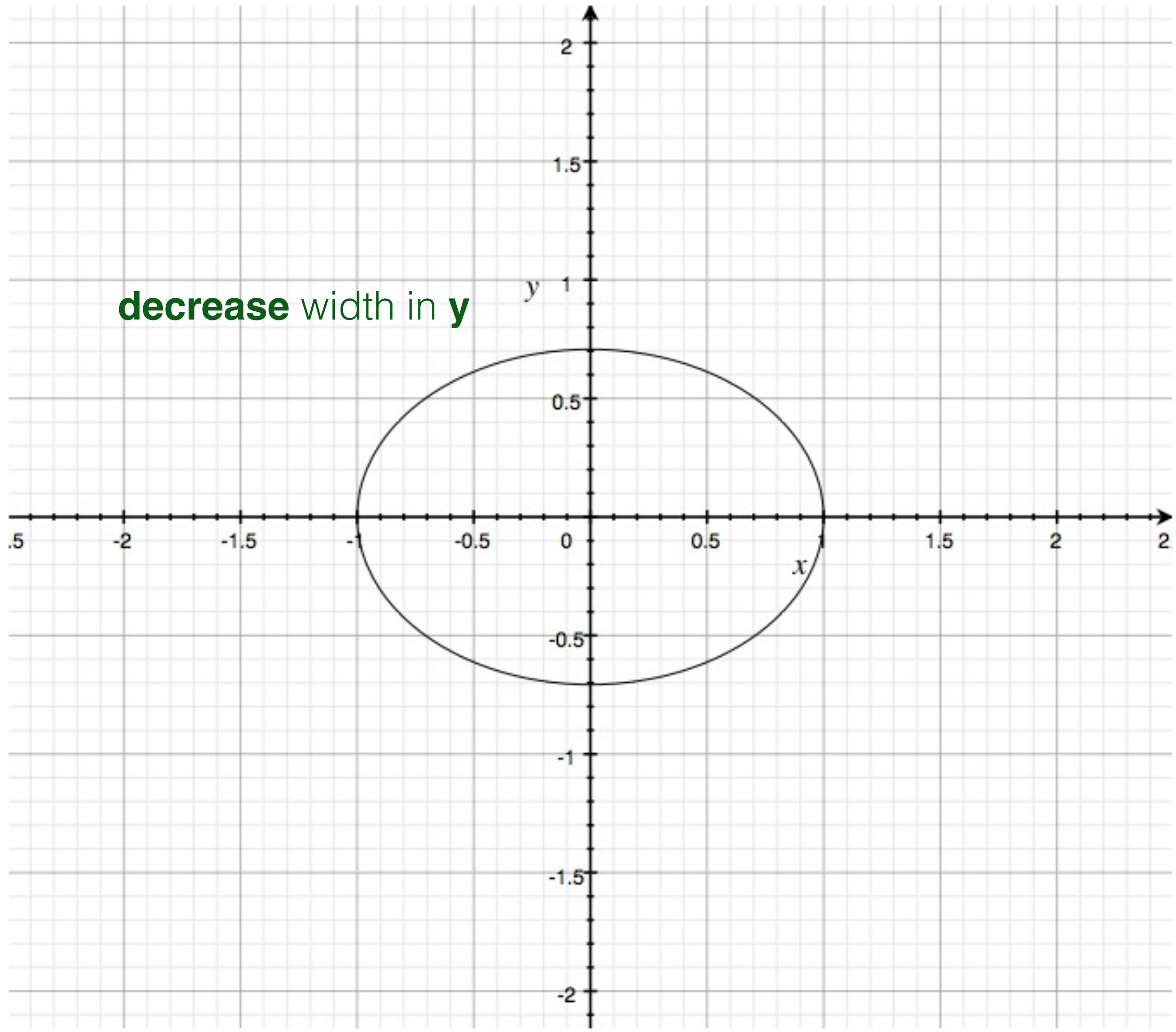
and slice at 1

*What happens if you **increase** coefficient on **y**?*

$$f(x,y) = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

and slice at 1

**decrease** width in **y**



$$f(x, y) = x^2 + y^2$$

can be written in matrix form like this…

$$f(x, y) = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

*What’s the shape?*
*What are the eigenvectors?*
*What are the eigenvalues?*

$$f(x,y) = x^2 + y^2$$

can be written in matrix form like this…

$$f(x,y) = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

**Result of Singular Value Decomposition (SVD)**

eigenvectors

eigenvalues along diagonal

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}^{\top}$$

axis of the 'ellipse slice'

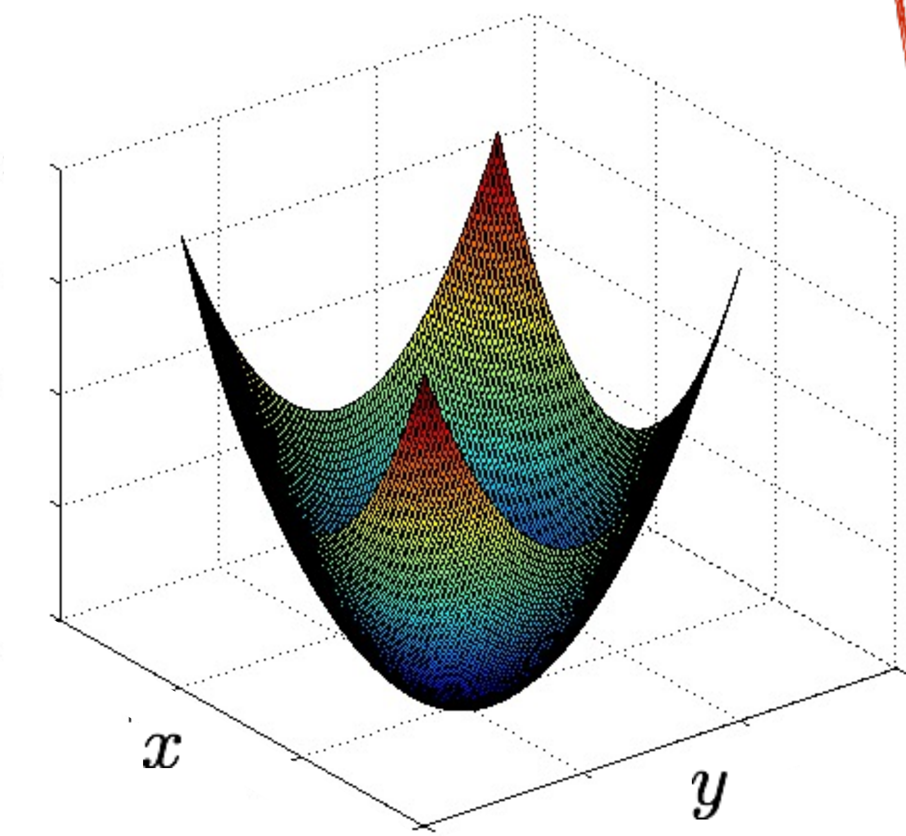
Inverse sqr of length of the quadratic along the axis

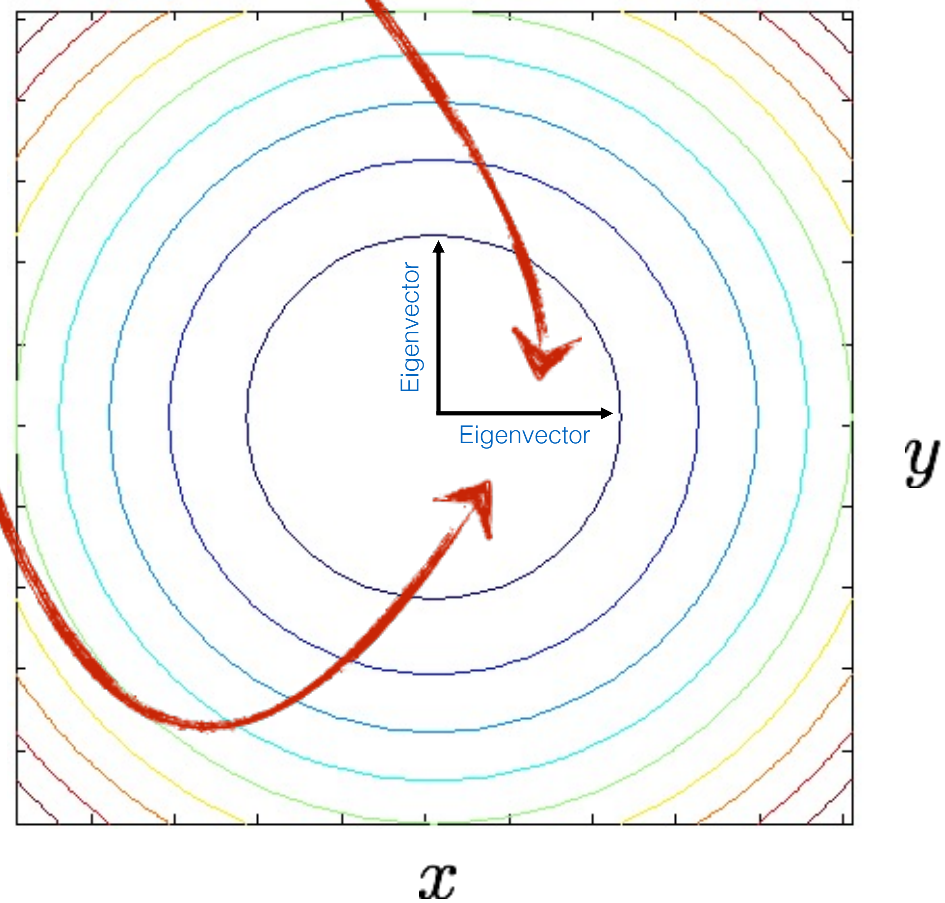
Eigenvectors Eigenvalues

$$\mathbf{A} = \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} = \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}^T$$

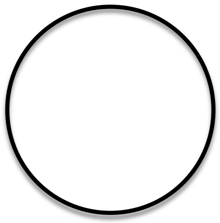
Eigenvectors

Inverse sqr of the size of the axis



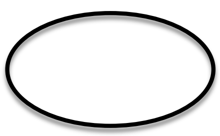



Recall:



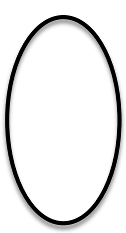
$$f(x,y) = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

you can smash this bowl in the **y** direction



$$f(x,y) = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

you can smash this bowl in the **x** direction



$$f(x,y) = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 4 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

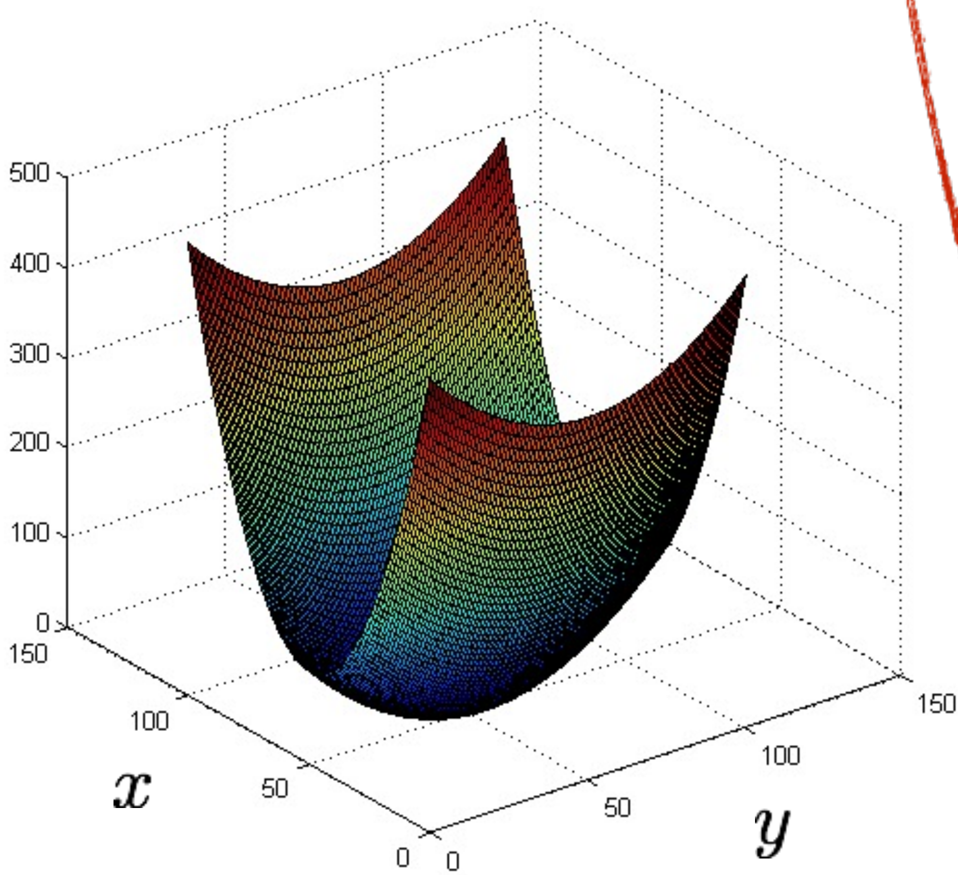
Eigenvalues

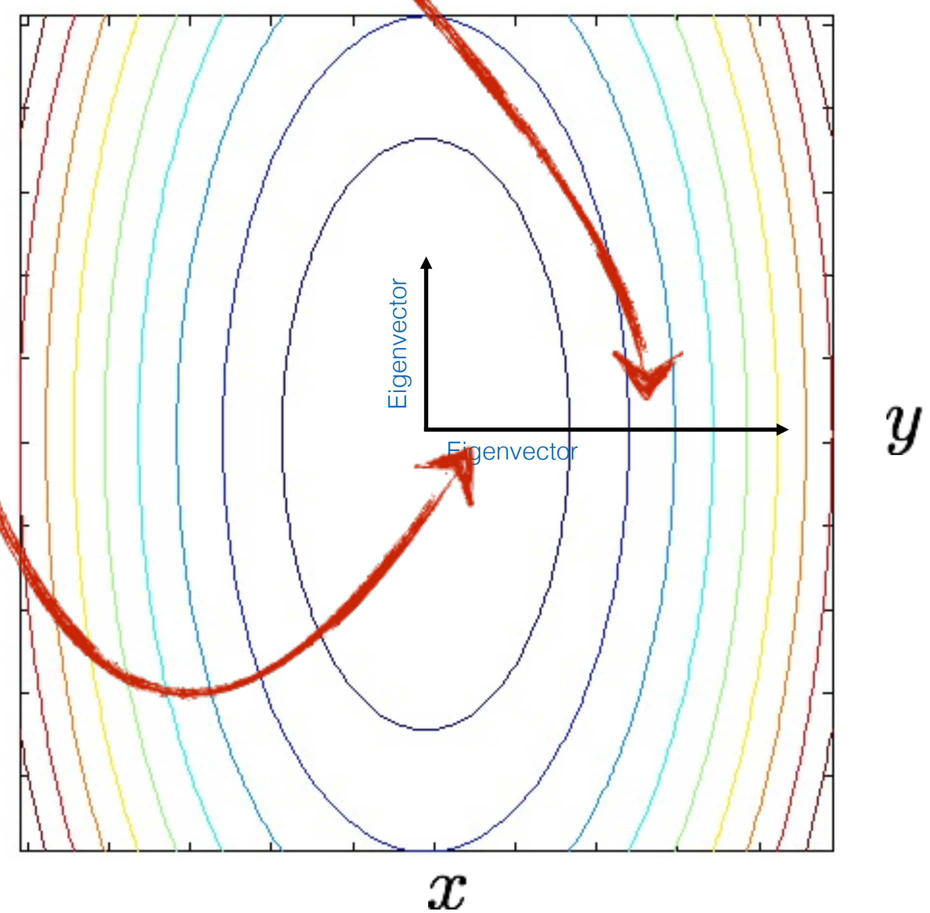
$$\mathbf{A} = \begin{bmatrix} 4 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 4 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}^T$$

Eigenvectors

Eigenvectors

Inverse sqr of length of axis

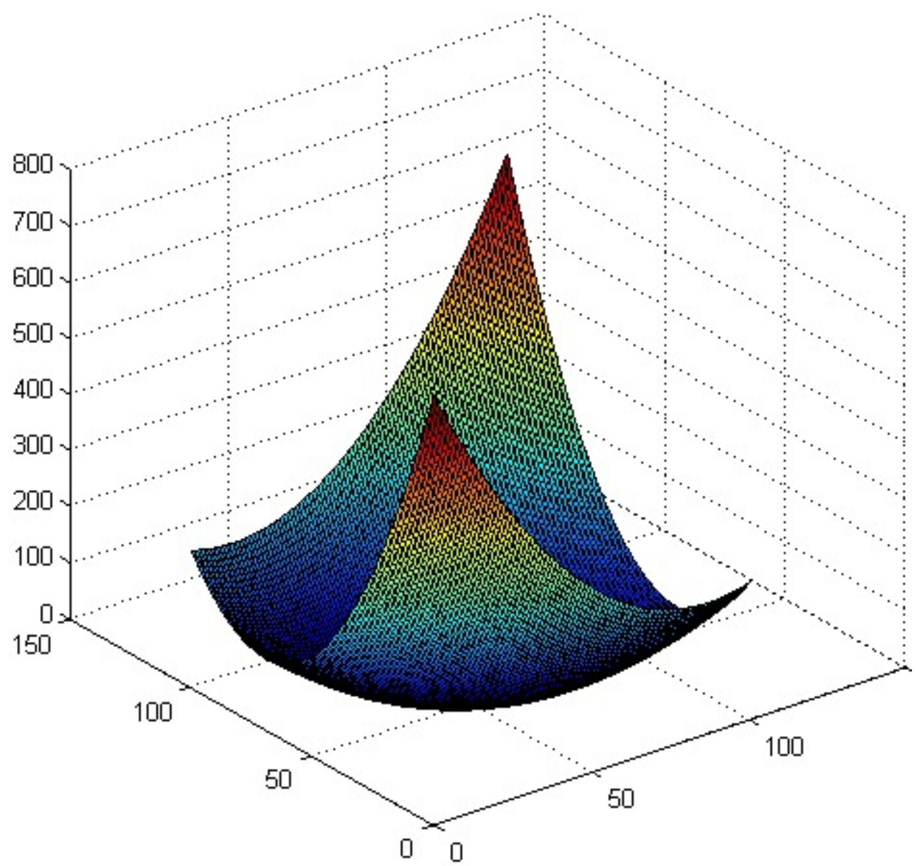


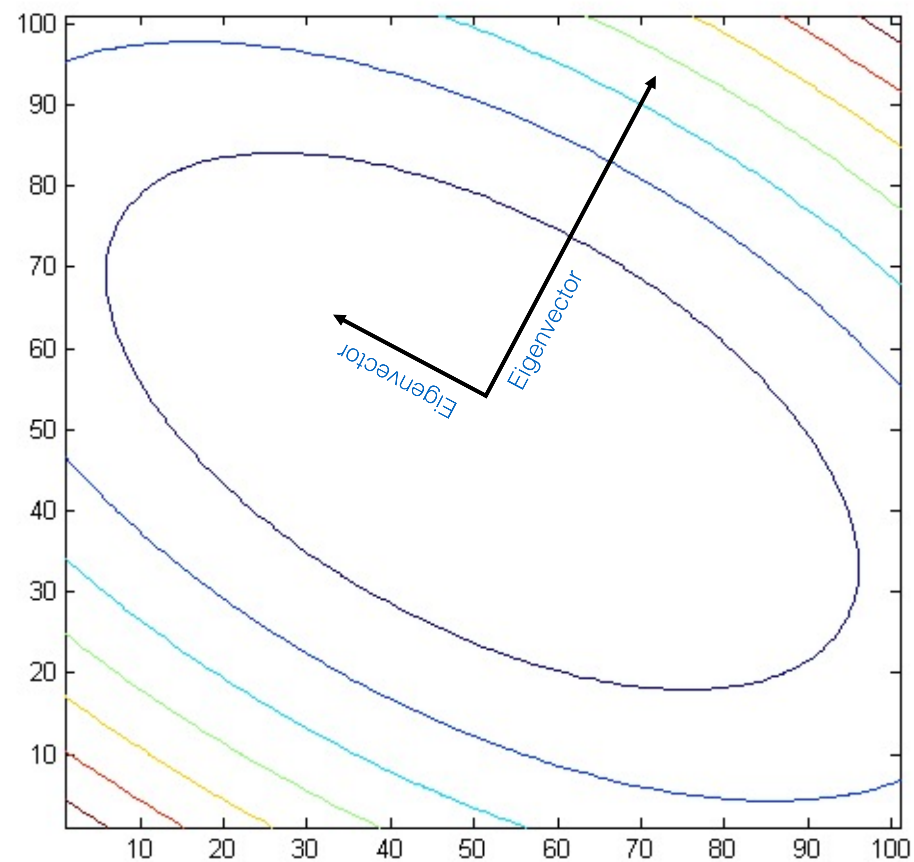



Eigenvalues

$$\mathbf{A} = \begin{bmatrix} 3.25 & 1.30 \\ 1.30 & 1.75 \end{bmatrix} = \begin{bmatrix} 0.50 & -0.87 \\ -0.87 & -0.50 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 4 \end{bmatrix} \begin{bmatrix} 0.50 & -0.87 \\ -0.87 & -0.50 \end{bmatrix}^T$$

Eigenvectors Eigenvectors



$$\mathbf{A} = \begin{bmatrix}7.75 & 3.90\\ 3.90 & 3.25\end{bmatrix} = \underset{\text{Eigenvectors}}{\begin{bmatrix}0.50 & -0.87\\ -0.87 & -0.50\end{bmatrix}} \overset{\text{Eigenvalues}}{\begin{bmatrix}1 & 0\\ 0 & 10\end{bmatrix}} \underset{\text{Eigenvectors}}{\begin{bmatrix}0.50 & -0.87\\ -0.87 & -0.50\end{bmatrix}^{T}}$$