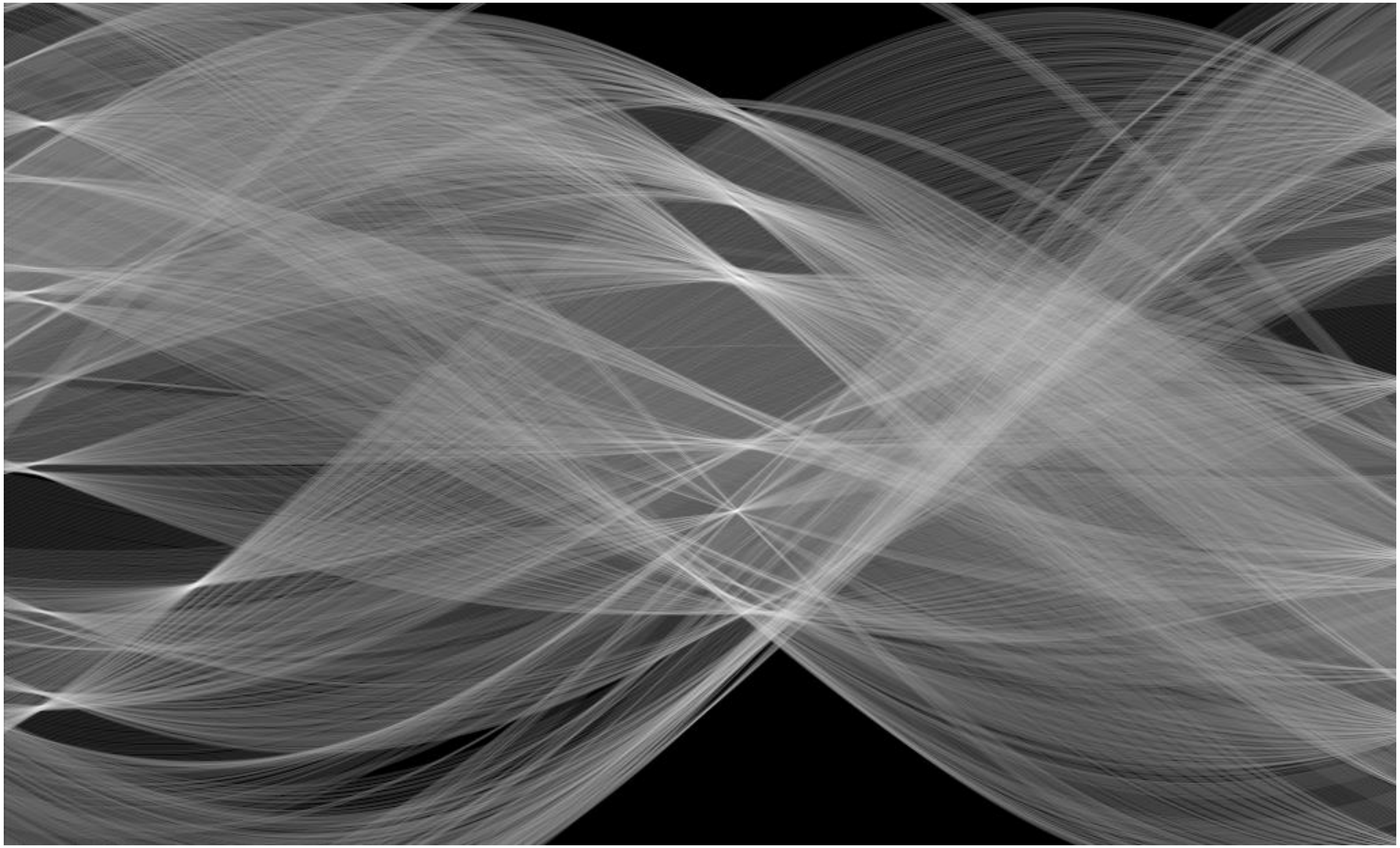
# Hough transform



http://www.cs.cmu.edu/~16385/

16-385 Computer Vision
Spring 2020, Lecture 4

# Course announcements

- Homework 1 posted on course website.
  - Due on February 5th at 23:59.
  - This homework is in Matlab.

- First theory quiz will be posted tonight and will be due on February 3rd, at 23:59.

- From here on, all office hours will be at Smith Hall 200.
  - Conference room next to the second floor restrooms.

# Overview of today’s lecture

Leftover from lecture 3:

- Frequency-domain filtering.
- Revisiting sampling.

New in lecture 4:

- Finding boundaries.
- Line fitting.
- Line parameterizations.
- Hough transform.
- Hough circles.
- Some applications.

# Slide credits

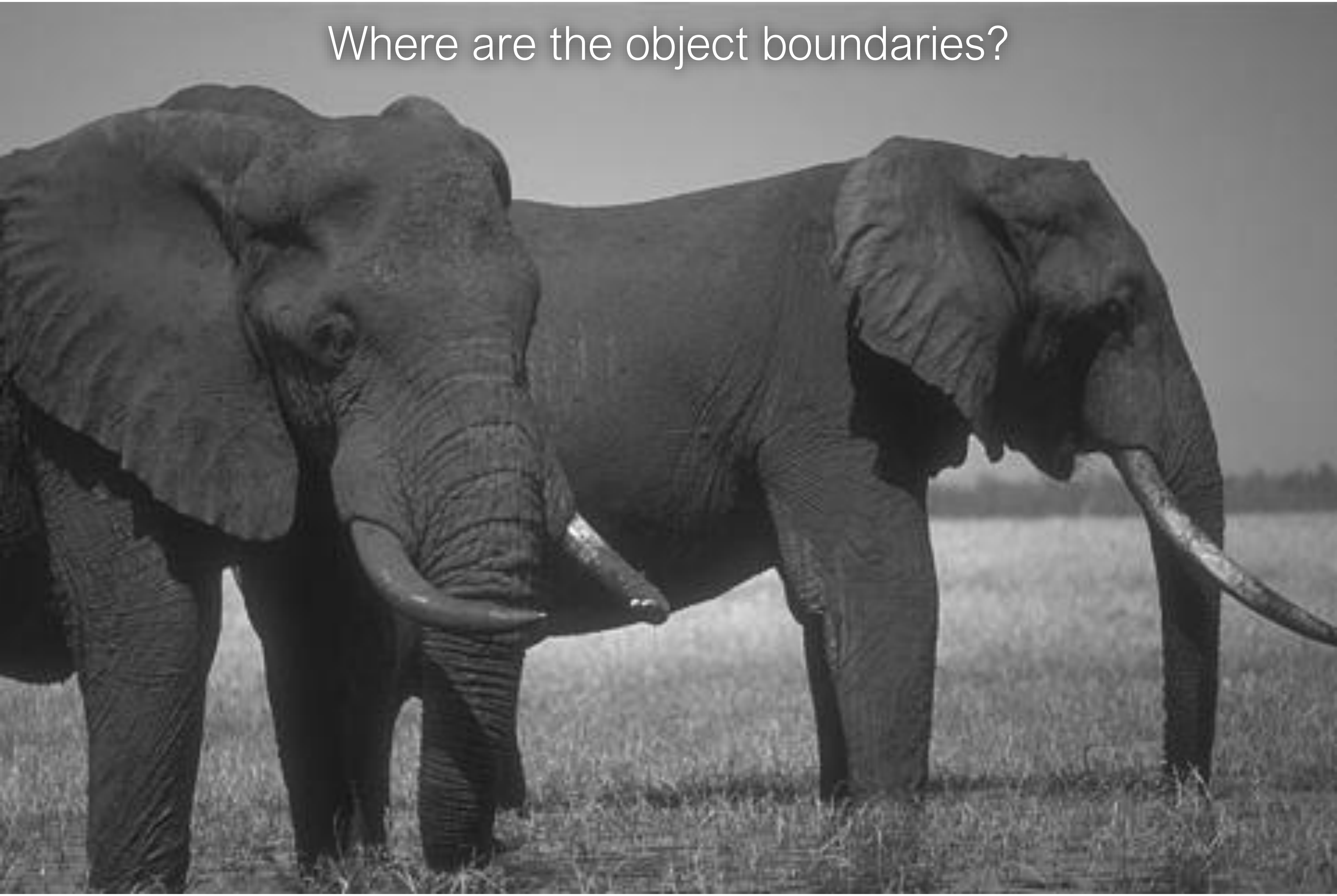
Most of these slides were adapted from:

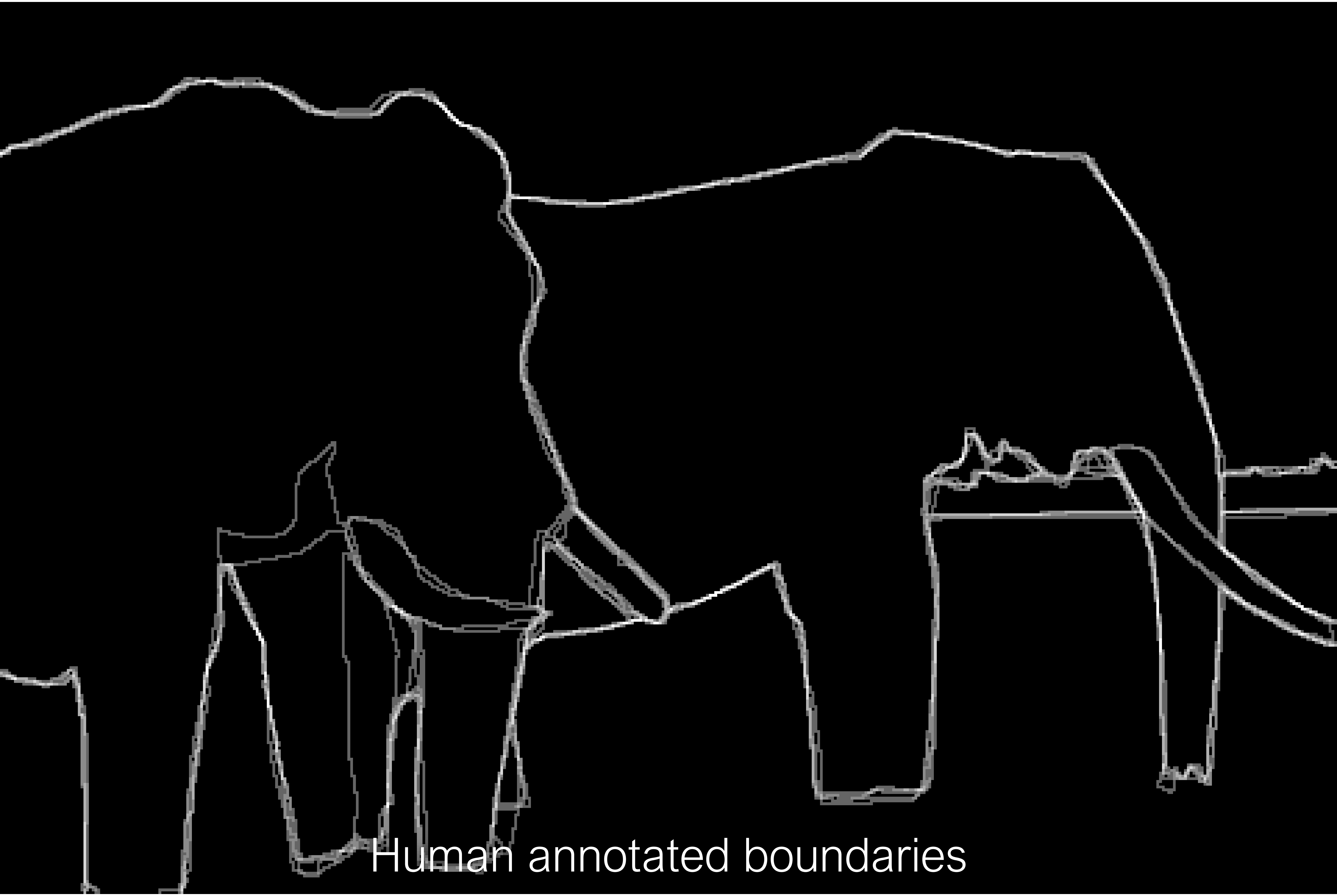
- Kris Kitani (15-463, Fall 2016).

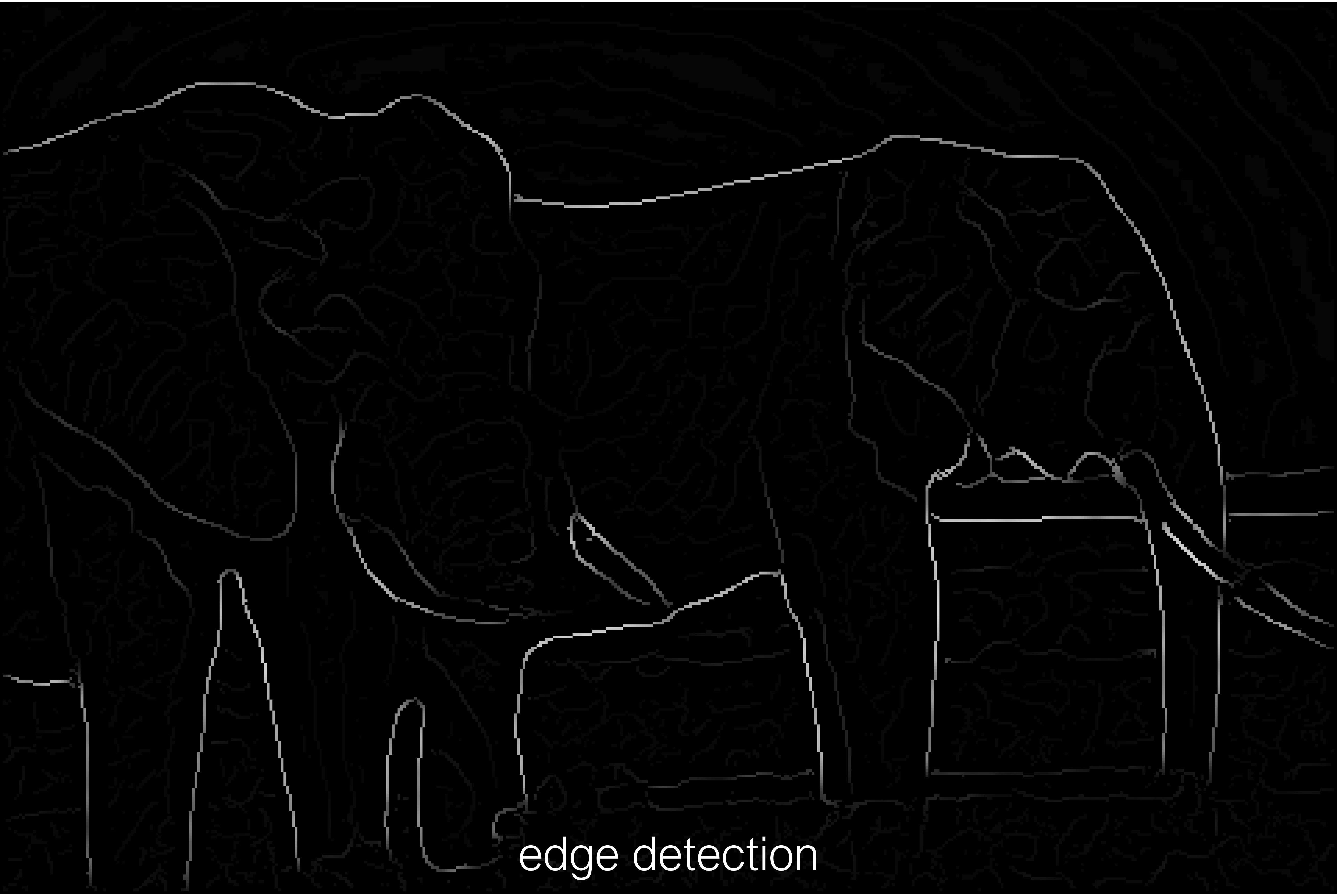
Some slides were inspired or taken from:

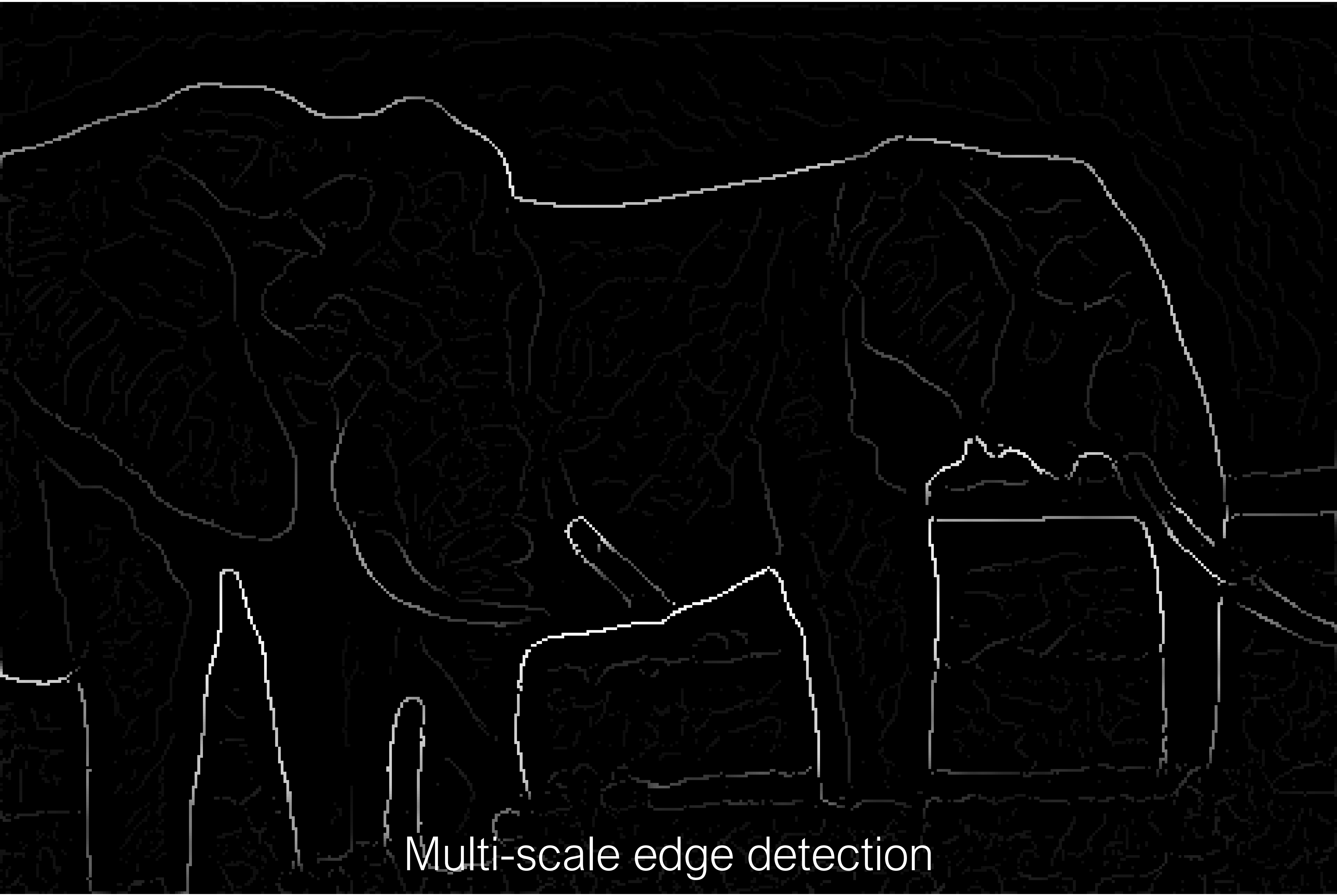
- Fredo Durand (MIT).
- James Hays (Georgia Tech).

# Finding boundaries

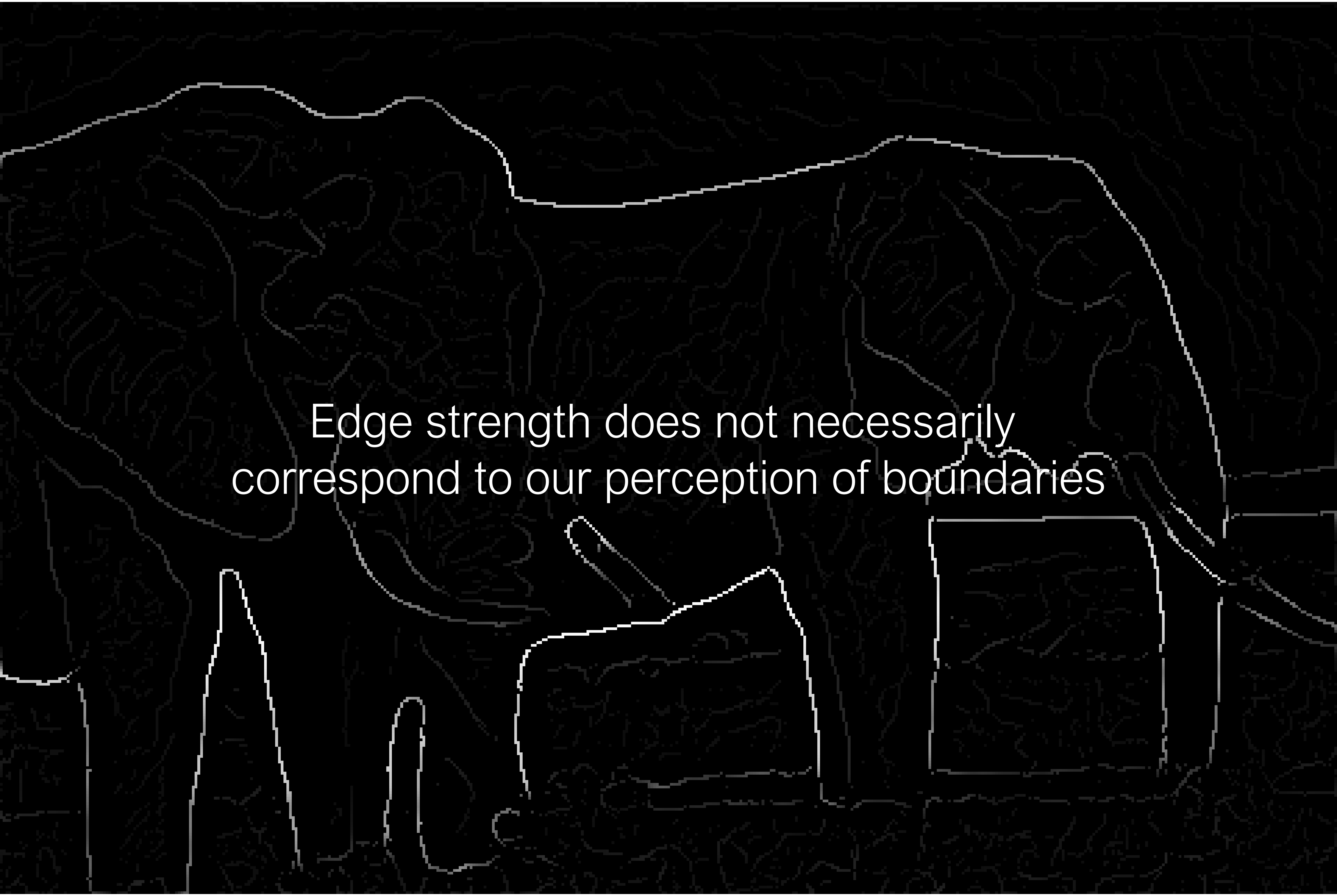

Where are the object boundaries?


Human annotated boundaries


edge detection

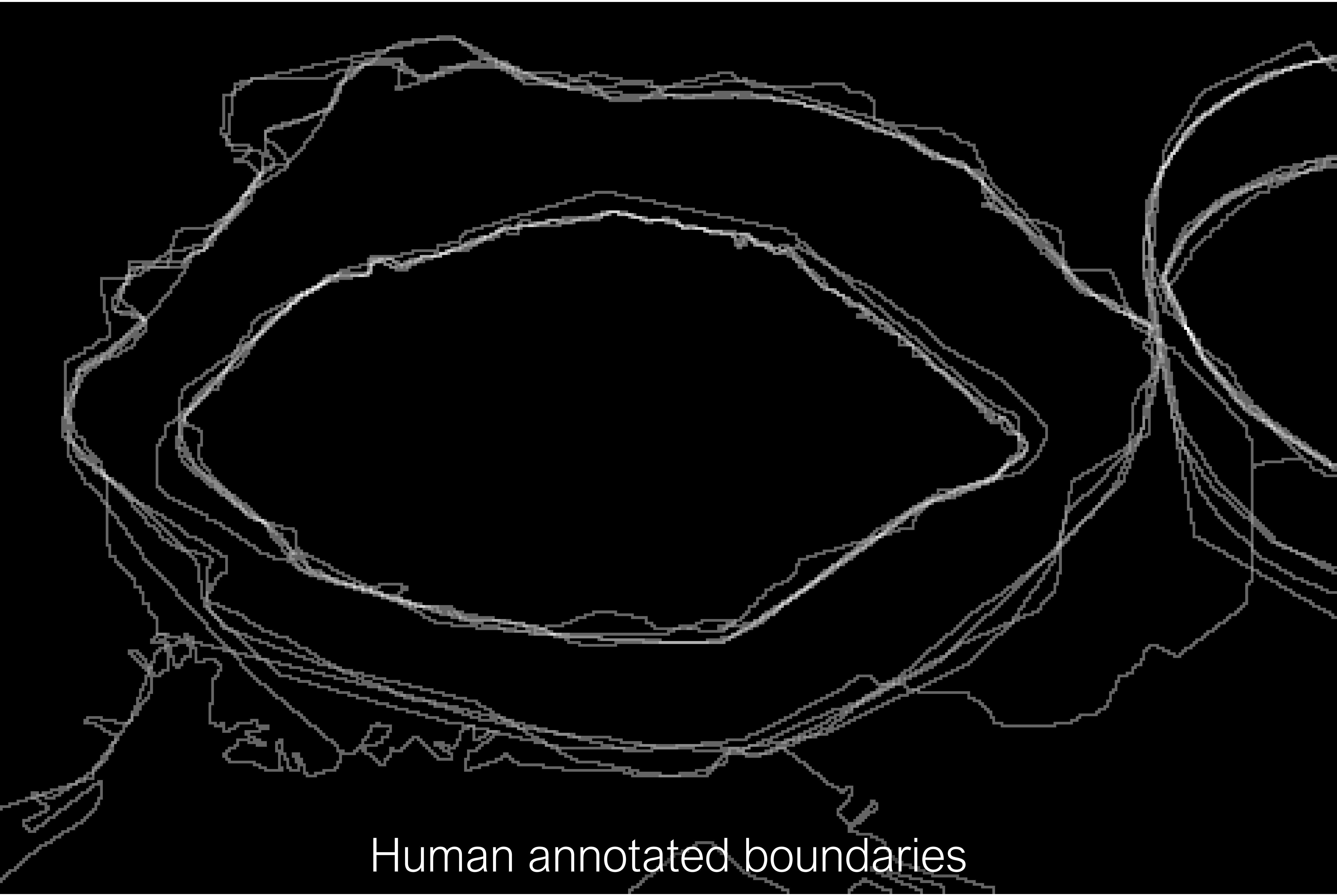


Multi-scale edge detection



Edge strength does not necessarily correspond to our perception of boundaries


Where are the object boundaries?


Human annotated boundaries

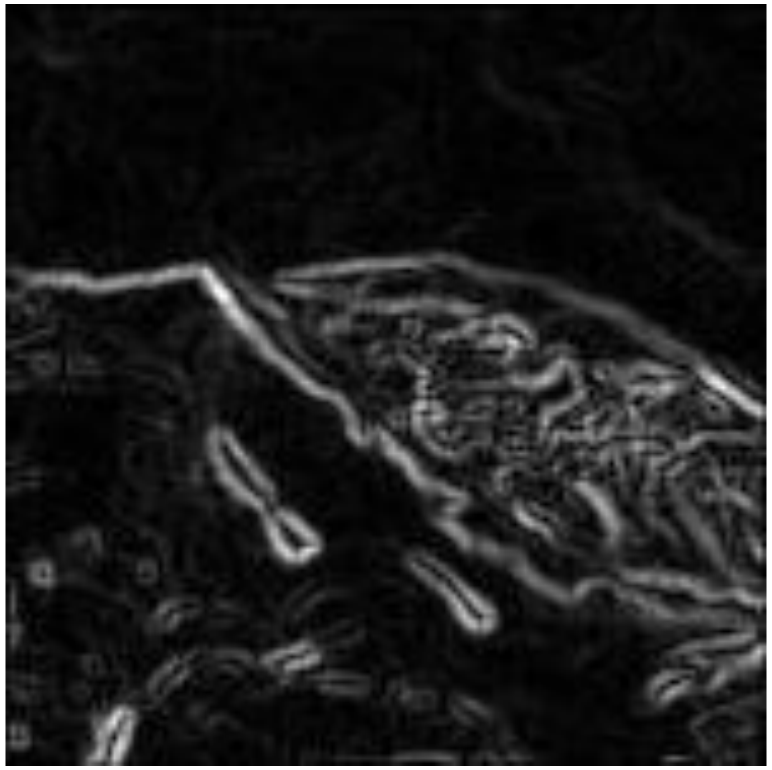

edge detection

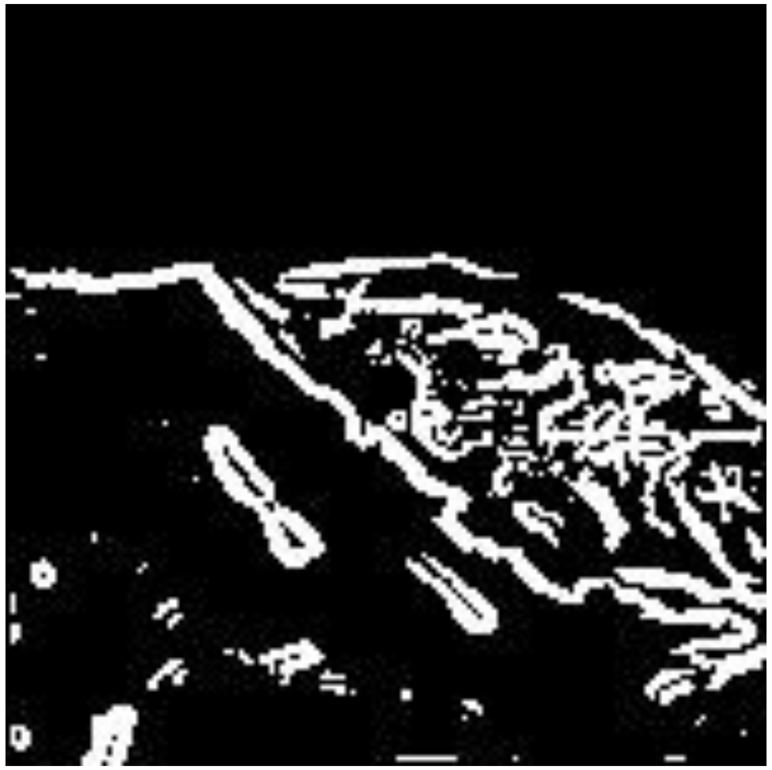

Defining boundaries are hard for us too

# Where is the boundary of the mountain top?

# Lines are hard to find


Original image

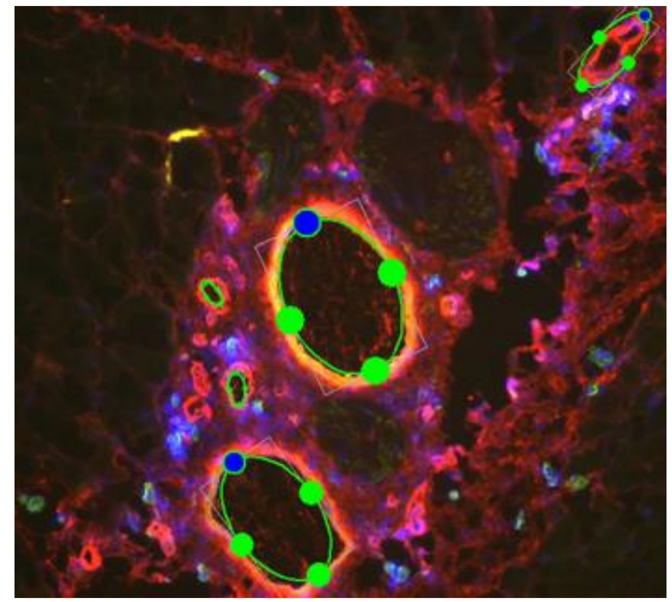

Edge detection


Thresholding

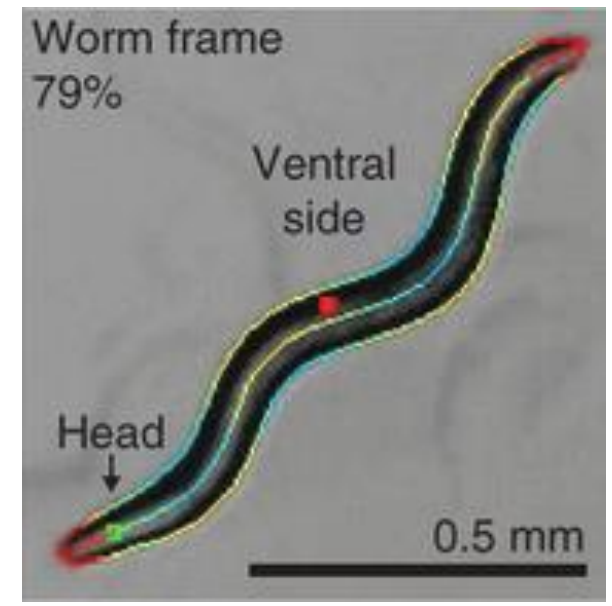
Noisy edge image
Incomplete boundaries

# Applications

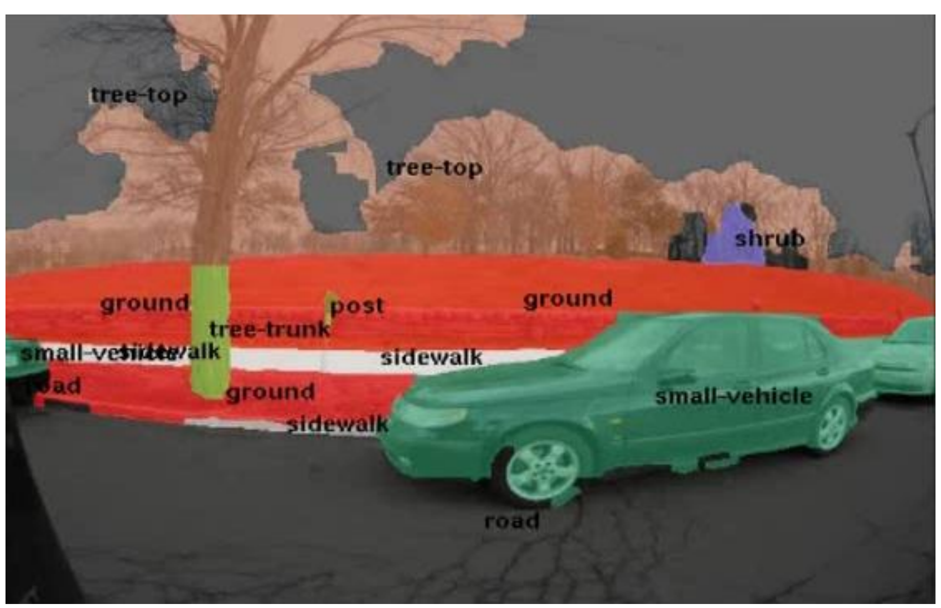
Autonomous Vehicles
(lane line detection)



tissue engineering
(blood vessel counting)

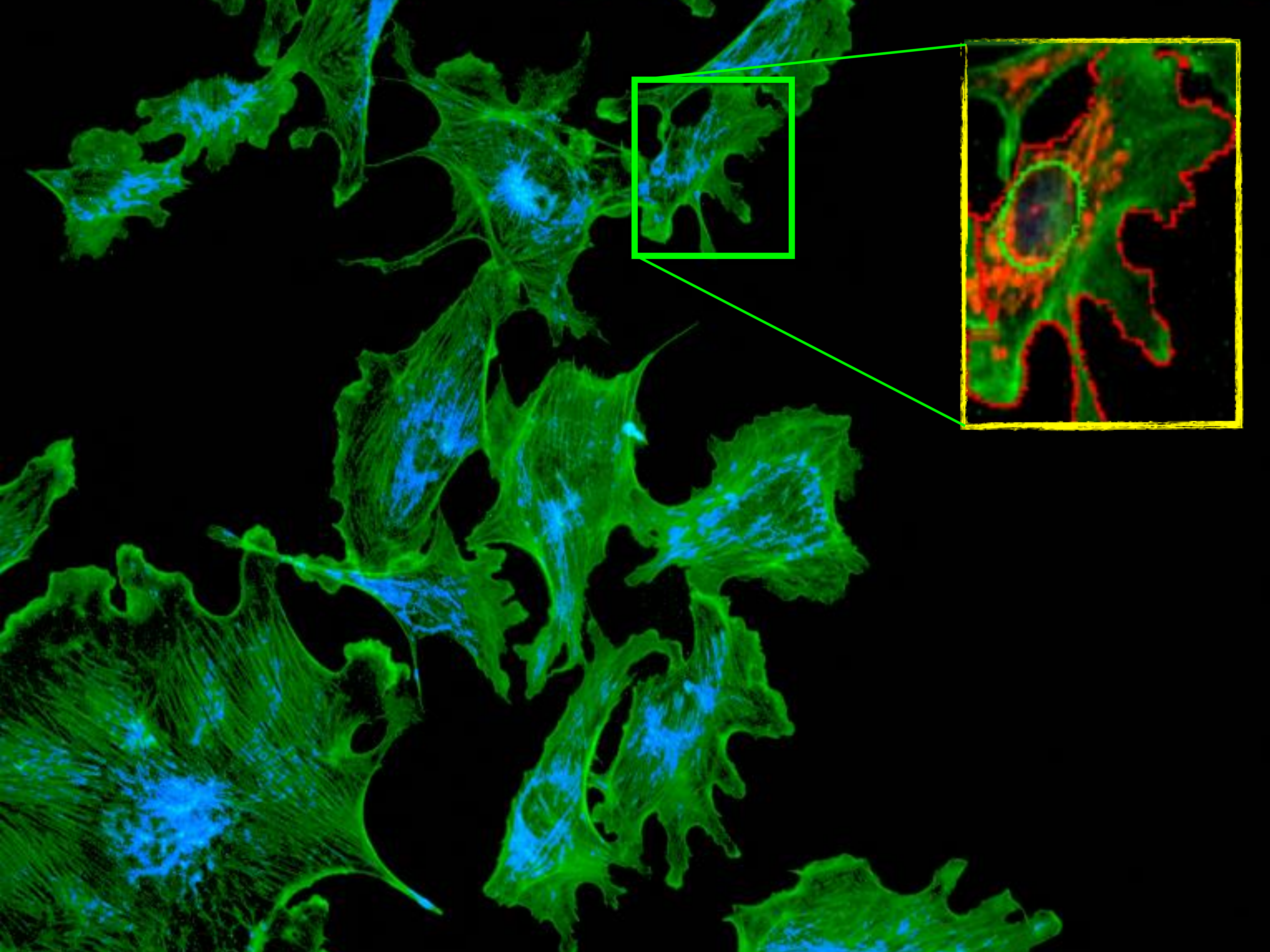



behavioral genetics
(earthworm contours)




Autonomous Vehicles
(semantic scene segmentation)

Computational Photography
(image inpainting)



# Line fitting

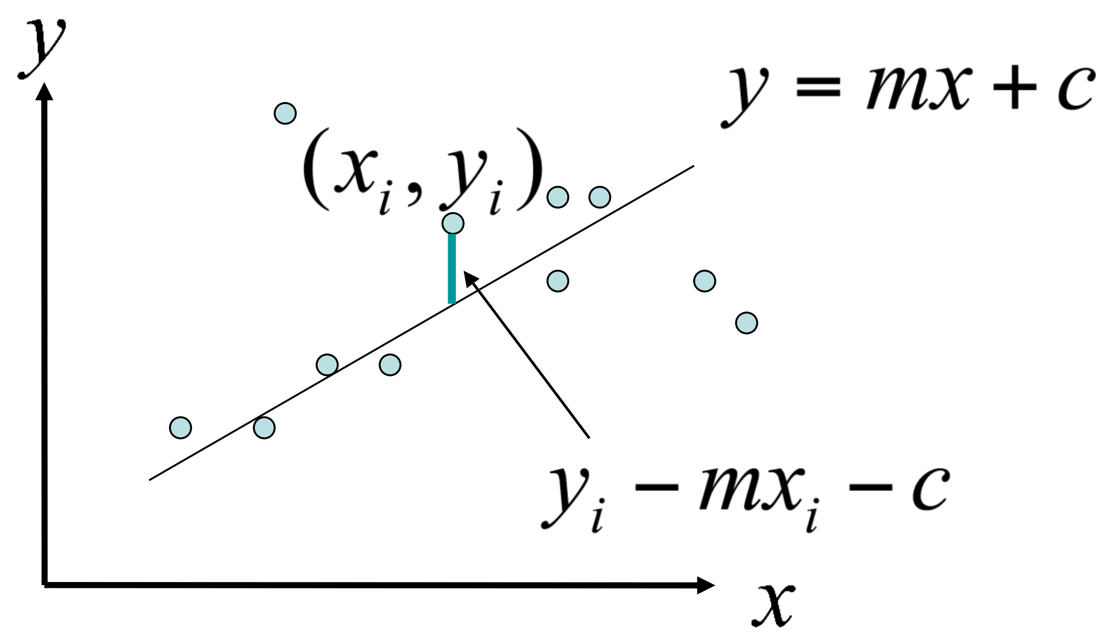
# Line fitting

Given: Many $(x_i, y_i)$ pairs

Find: Parameters $(m, c)$

Minimize: Average square distance:

$$E = \sum_i \frac{(y_i - mx_i - c)^2}{N}$$




*How can we solve this minimization?*

# Line fitting

Given: Many $(x_i, y_i)$ pairs

Find: Parameters $(m, c)$

Minimize: Average square distance:

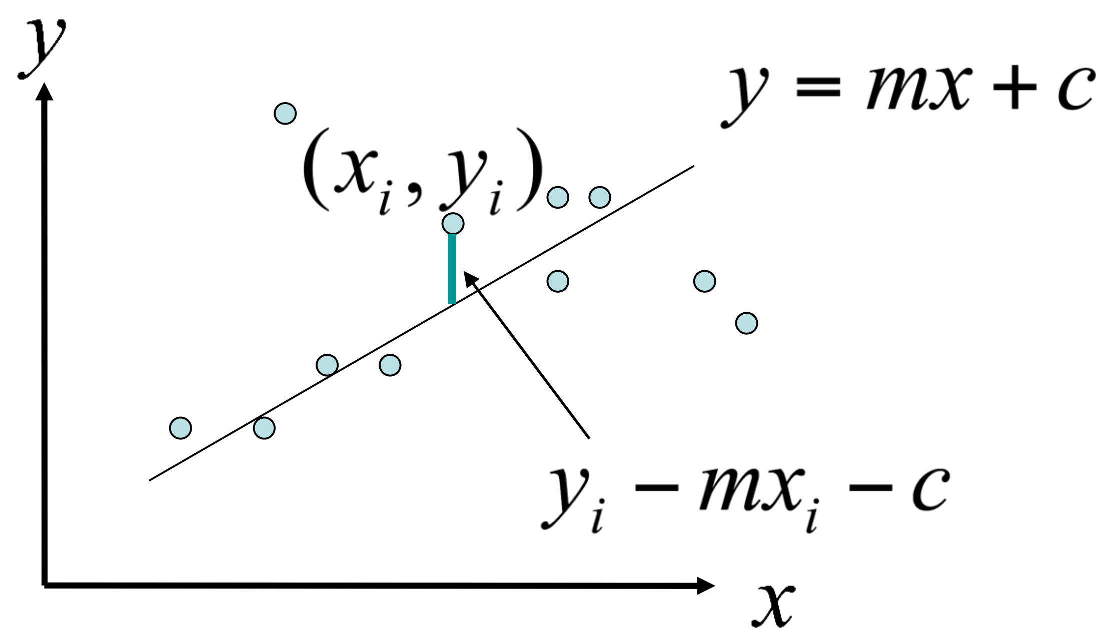
$$E = \sum_i \frac{(y_i - mx_i - c)^2}{N}$$

Using:

$$\frac{\partial E}{\partial m} = 0 \quad \& \quad \frac{\partial E}{\partial c} = 0$$

Note:

$$\bar{y} = \frac{\sum_i y_i}{N} \qquad \bar{x} = \frac{\sum_i x_i}{N}$$




$$c = \bar{y} - m\bar{x}$$

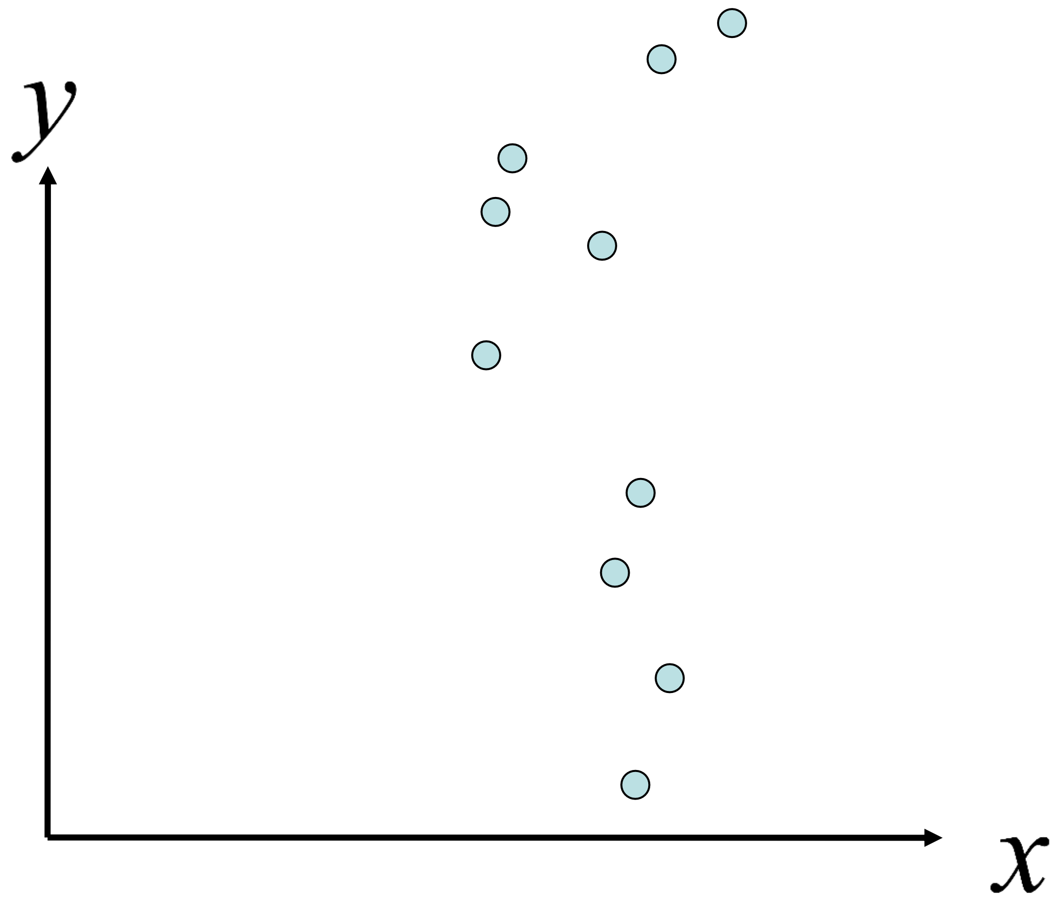
$$m = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2}$$

*What are some problems with the approach?*

# Problems with parameterizations

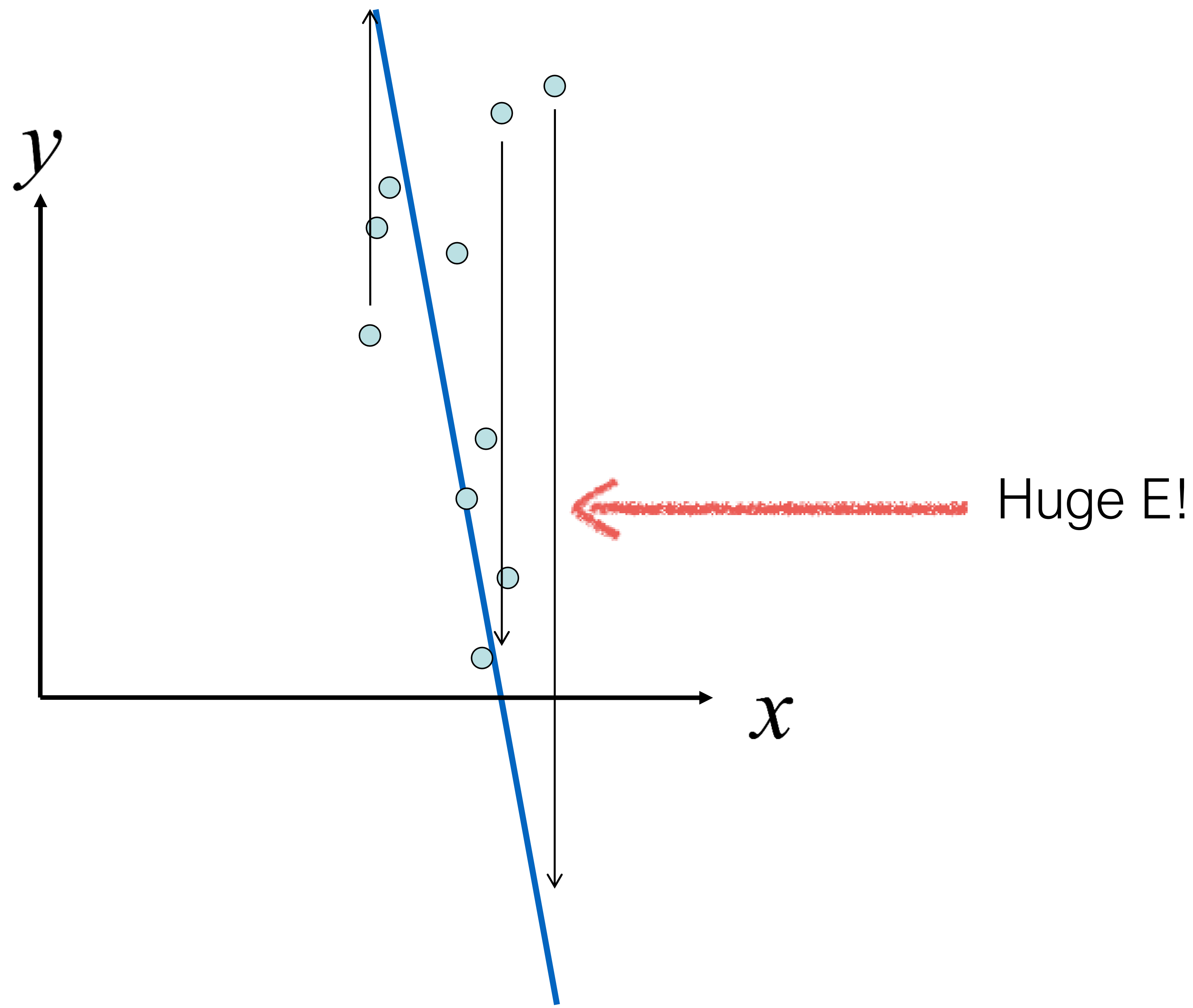
Where is the line that minimizes E?

$$\boxed{E = \sum_{i=1}^{n} (y_i - mx_i - b)^2}$$



# Problems with parameterizations

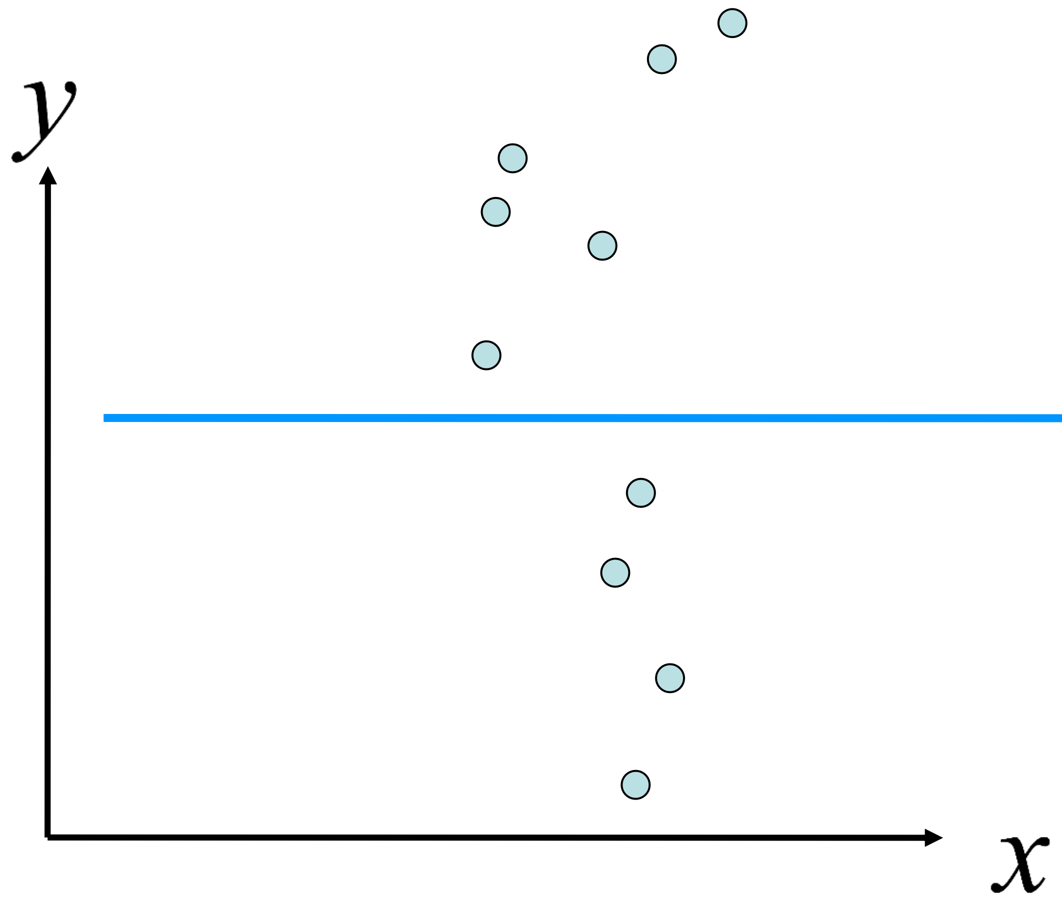
Where is the line that minimizes E? $\boxed{E = \sum_{i=1}^{n} (y_i - m x_i - b)^2}$



# Problems with parameterizations

Where is the line that minimizes E?

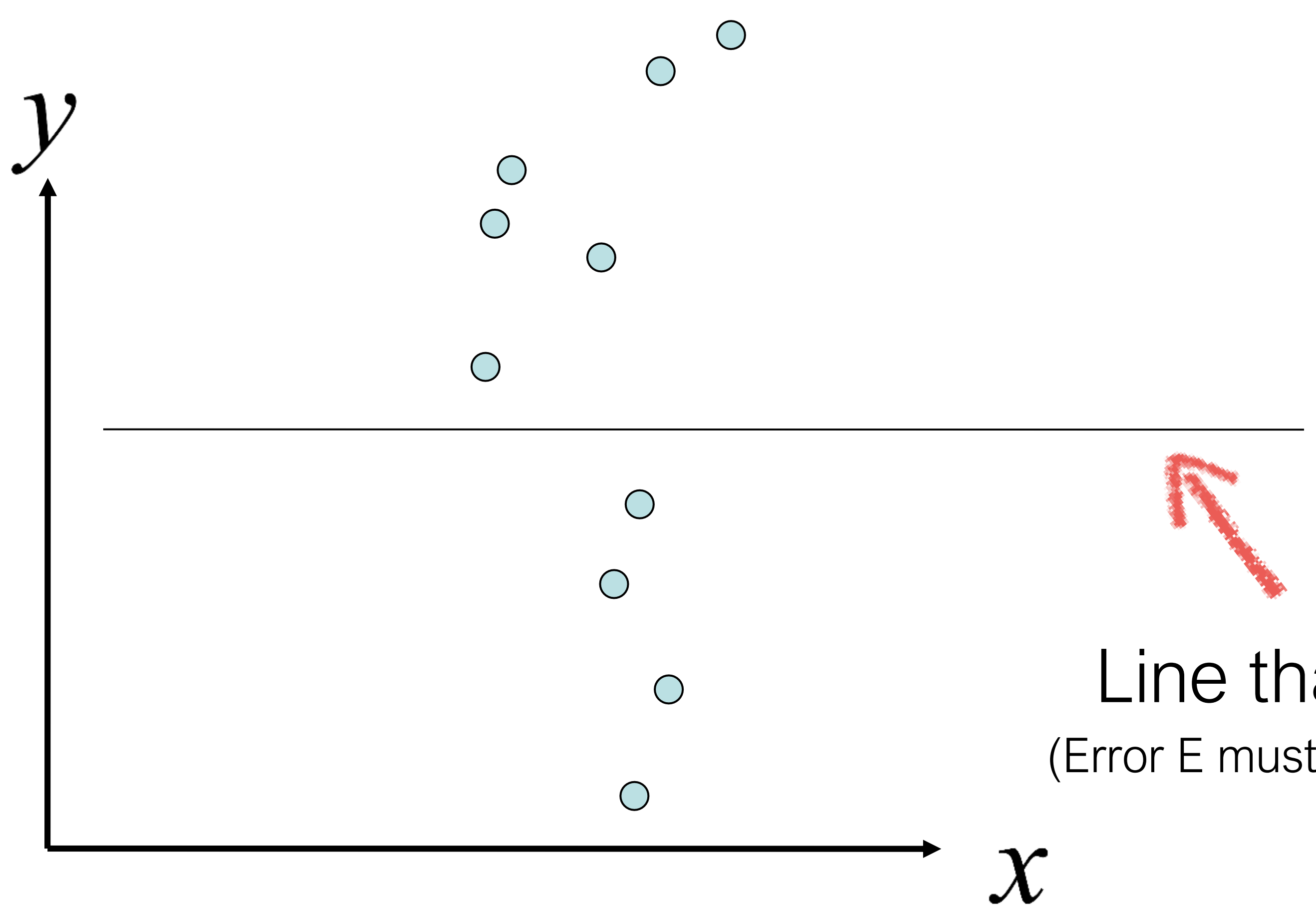
$$\boxed{E = \sum_{i=1}^{n} (y_i - m x_i - b)^2}$$




Line that minimizes E!!

# Problems with parameterizations

Where is the line that minimizes E?

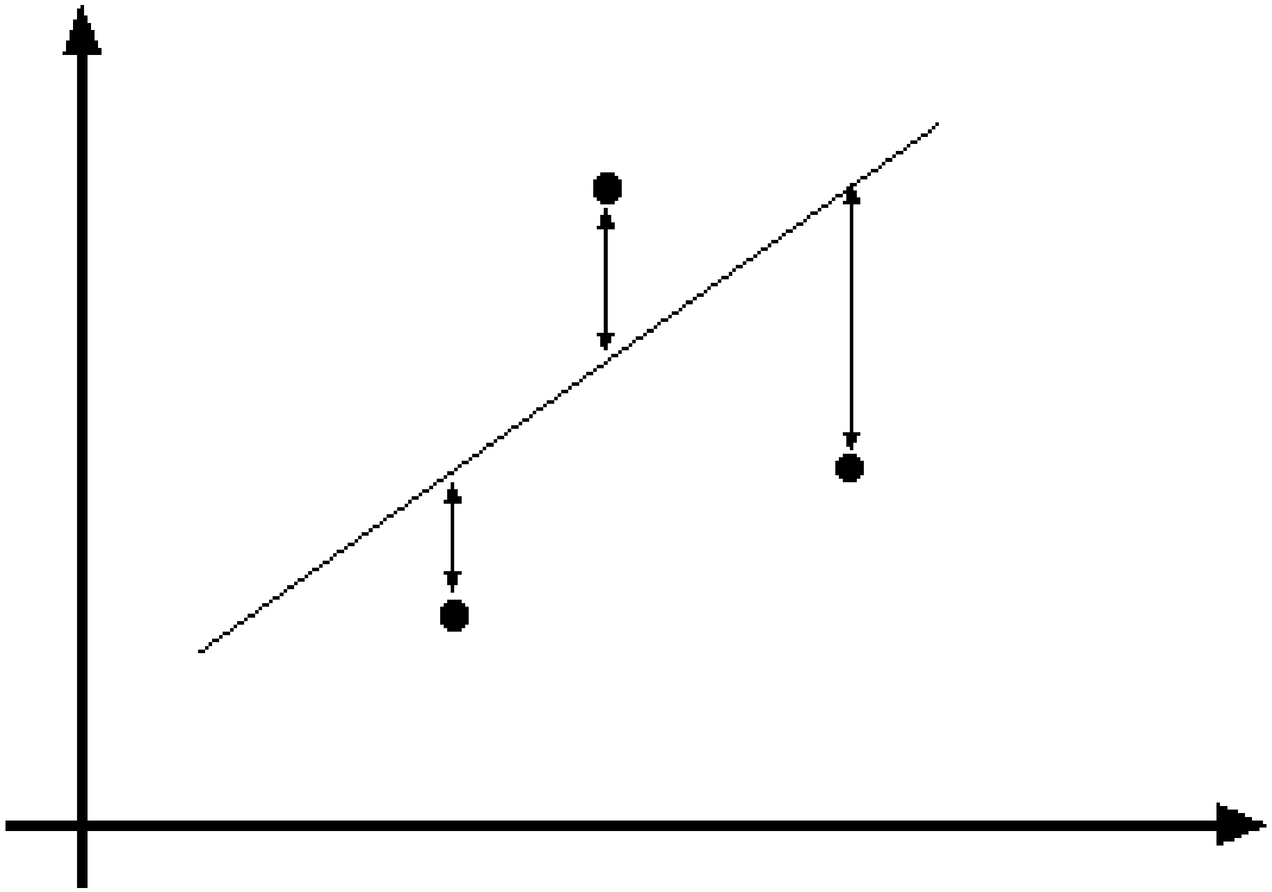


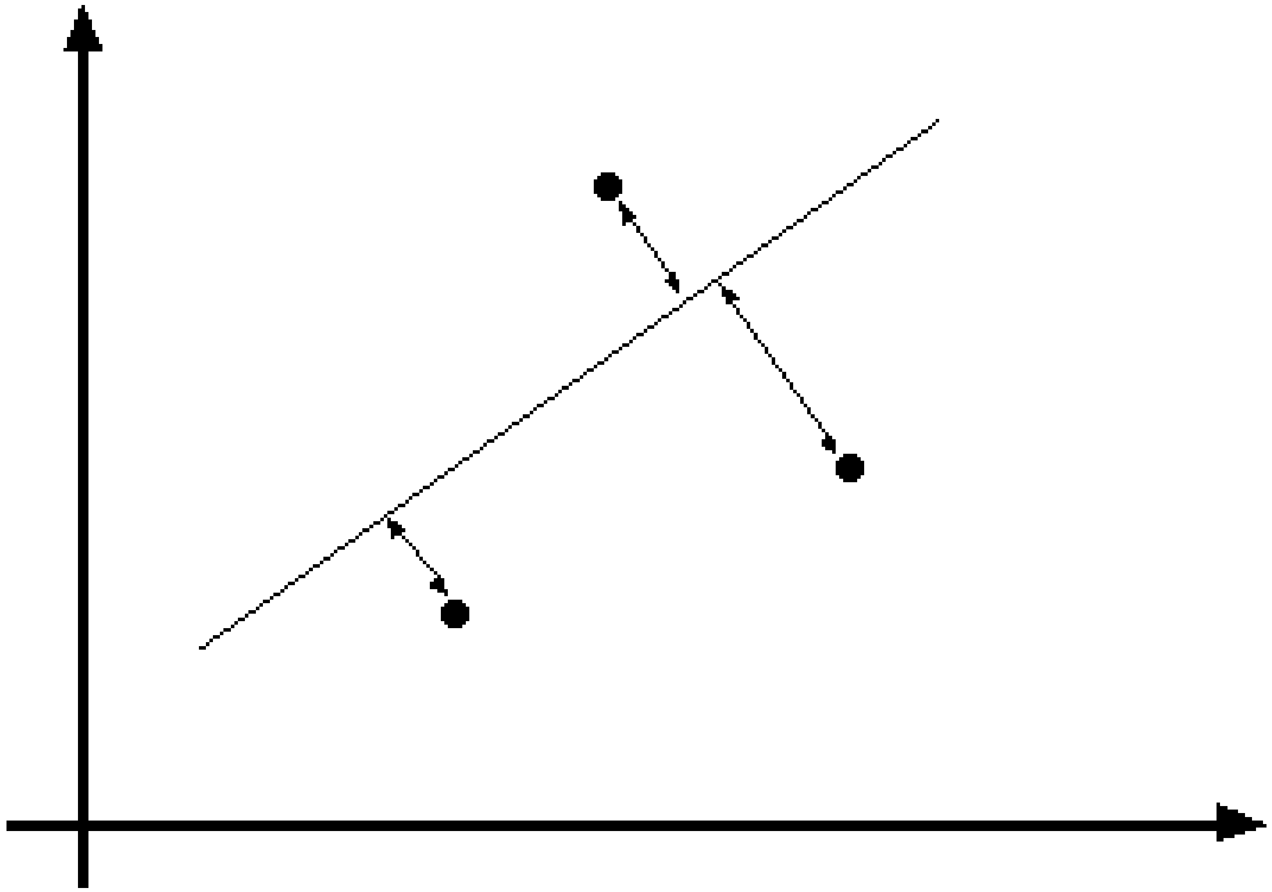

Line that minimizes E!!

(Error E must be formulated carefully!)

*How can we deal with this?*

Line fitting is easily setup as a maximum likelihood problem
… but choice of model is important

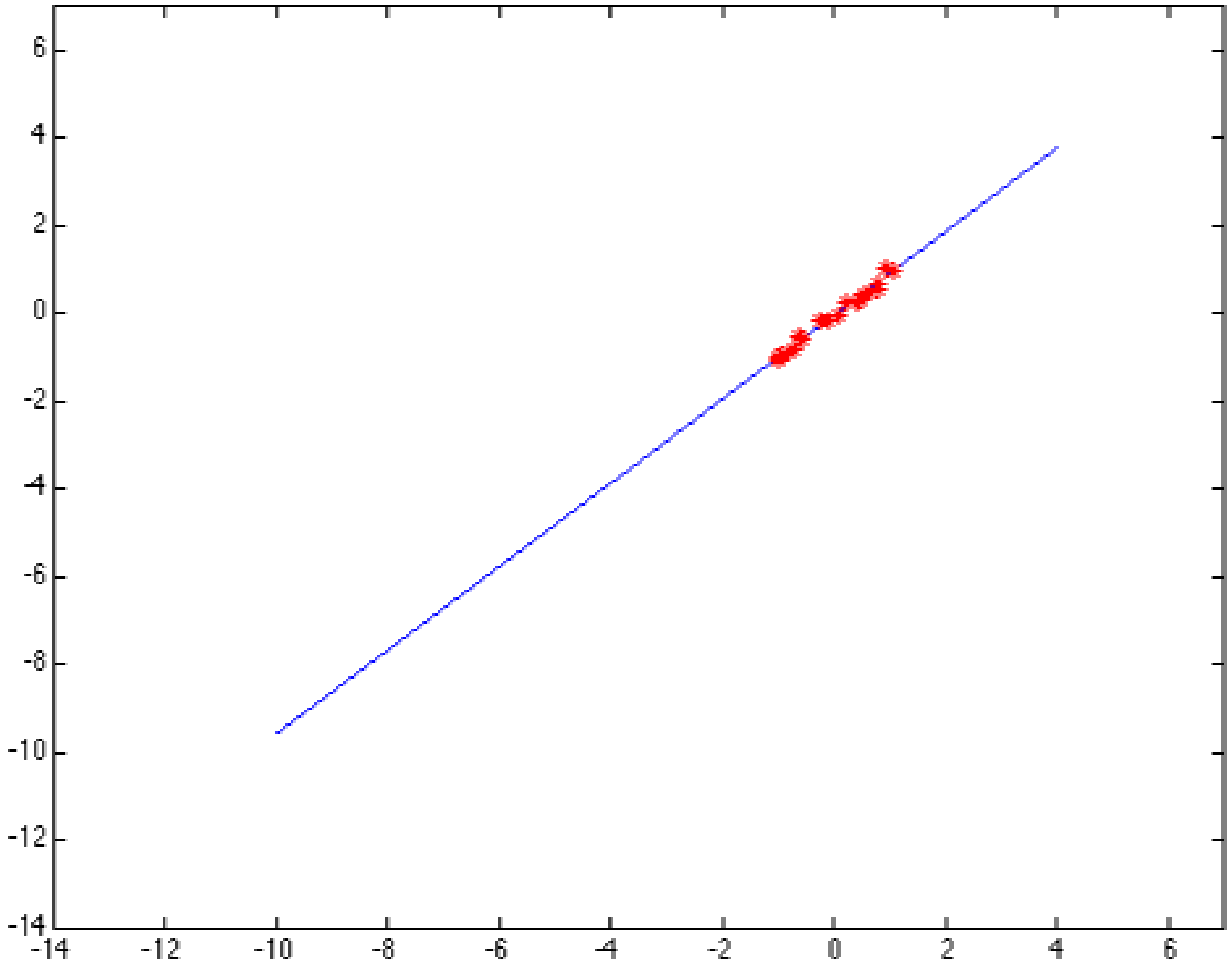




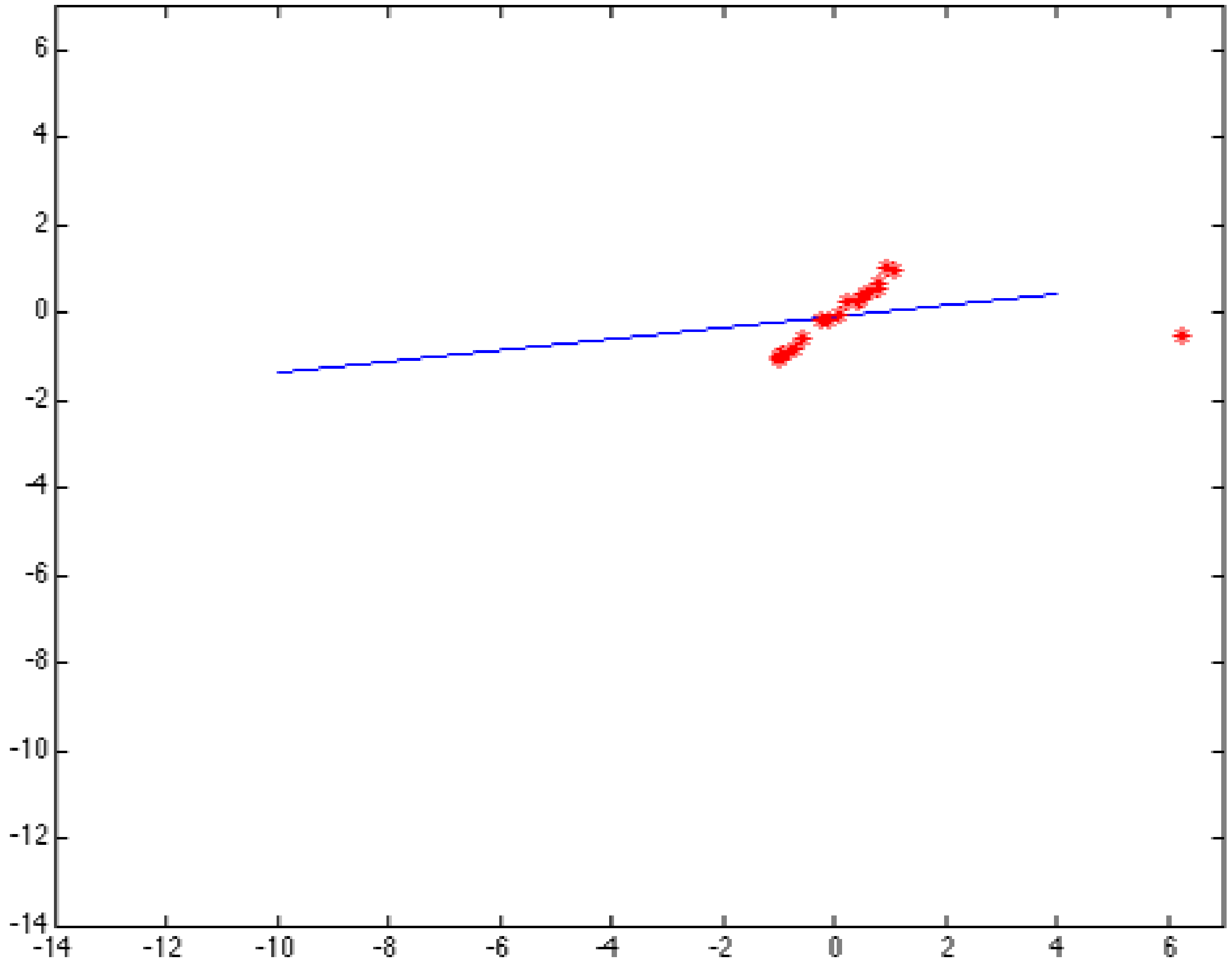
$$\boxed{E = \sum_{i=1}^{n} (y_i - mx_i - b)^2}$$

*What optimization are we solving here?*

# Problems with noise




Least-squares error fit




Squared error heavily penalizes outliers

# Model fitting is difficult because…

- **Extraneous data:** clutter or multiple models
  - We do not know what is part of the model?
  - Can we pull out models with a few parts from much larger amounts of background clutter?
- **Missing data:** only some parts of model are present
- **Noise**

- Cost:
  - It is not feasible to check all combinations of features by fitting a model to each possible subset

*So what can we do?*

# Line parameterizations

# Slope intercept form

$$y = mx + b$$

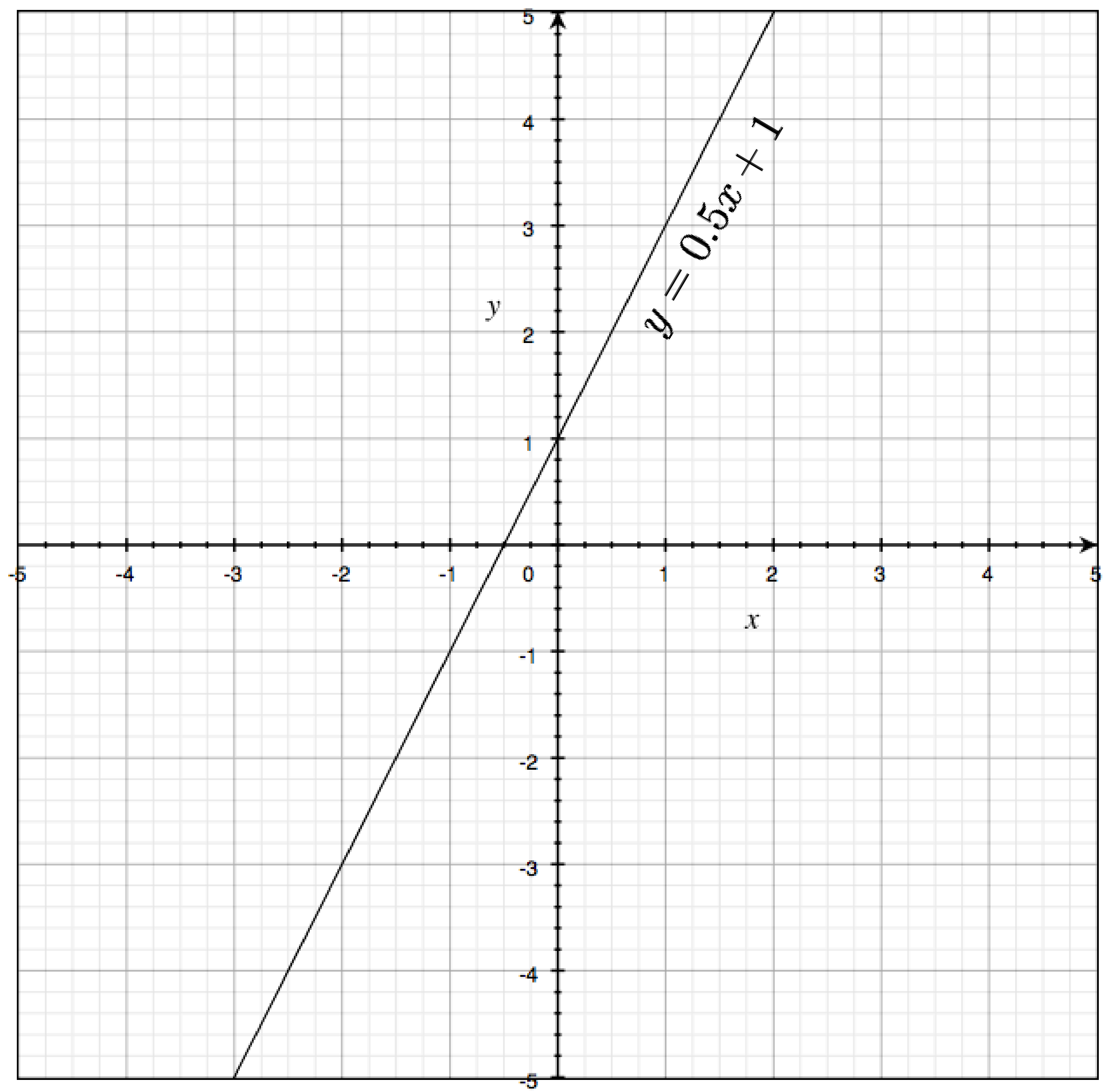
slope y-intercept

*What are m and b?*

# Slope intercept form

$$y = mx + b$$

slope y-intercept



# Double intercept form

$$\frac{x}{a} + \frac{y}{b} = 1$$

x-intercept

y-intercept

*What are x and y?*

# Double intercept form

$$\frac{x}{a} + \frac{y}{b} = 1$$

x-intercept

y-intercept

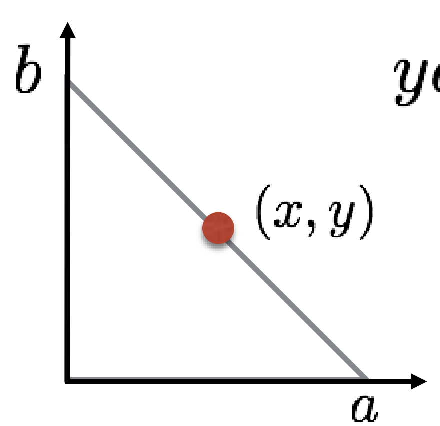
Derivation:

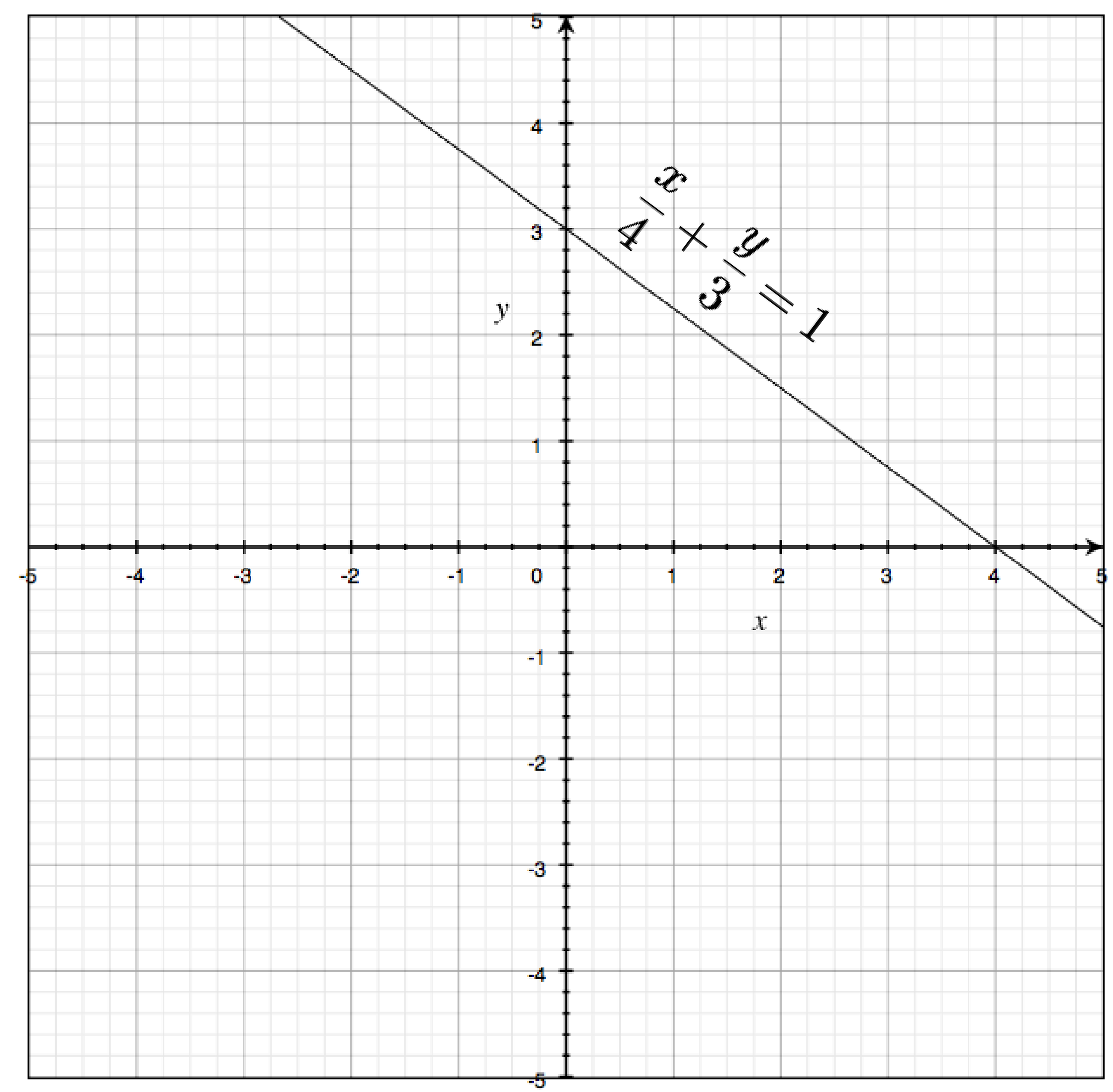
(Similar slope) $$\frac{y-b}{x-0} = \frac{0-y}{a-x}$$

$$ya + yx - ba + bx = -yx$$

$$ya + bx = ba$$

$$\frac{y}{b} + \frac{x}{a} = 1$$



# Normal Form

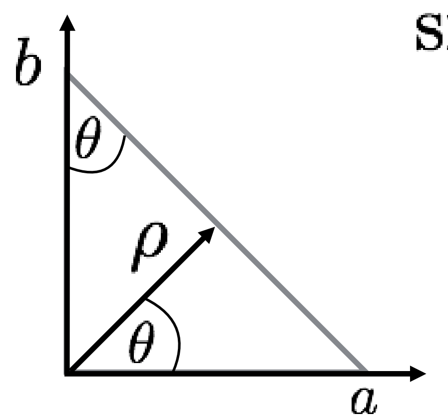
$$x \cos\theta + y \sin\theta = \rho$$

*What are rho and theta?*

# Normal Form

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

Derivation:

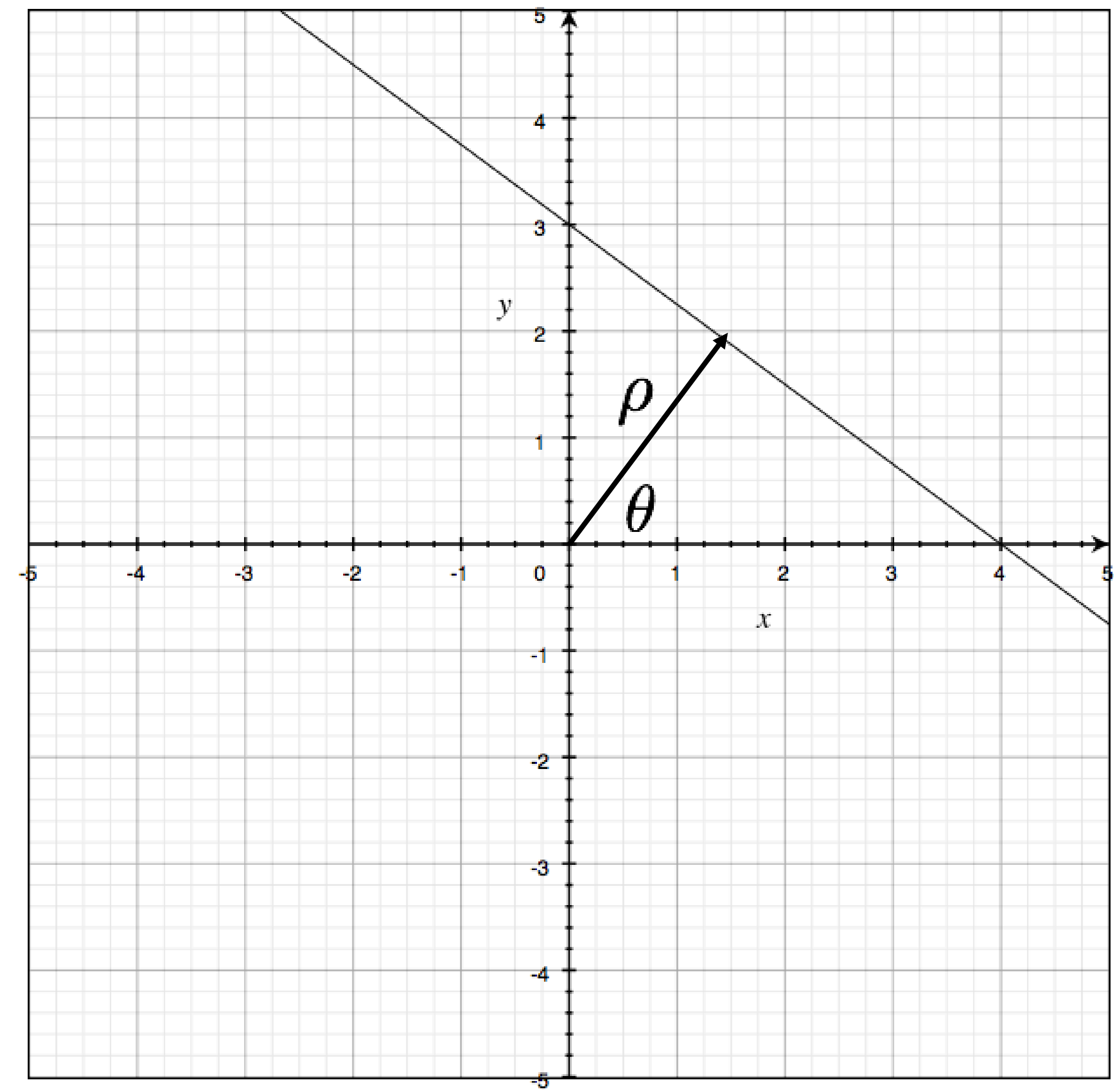



$$\cos \theta = \frac{\rho}{a} \rightarrow a = \frac{\rho}{\cos \theta}$$

$$\sin \theta = \frac{\rho}{b} \rightarrow b = \frac{\rho}{\sin \theta}$$

plug into: $\frac{x}{a} + \frac{y}{b} = 1$

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



# Hough transform

# Hough transform

- Generic framework for detecting a parametric model
- Edges don’t have to be connected
- Lines can be occluded
- Key idea: edges **vote** for the possible models

# Image and parameter space

variables

$$y = mx + b$$

parameters

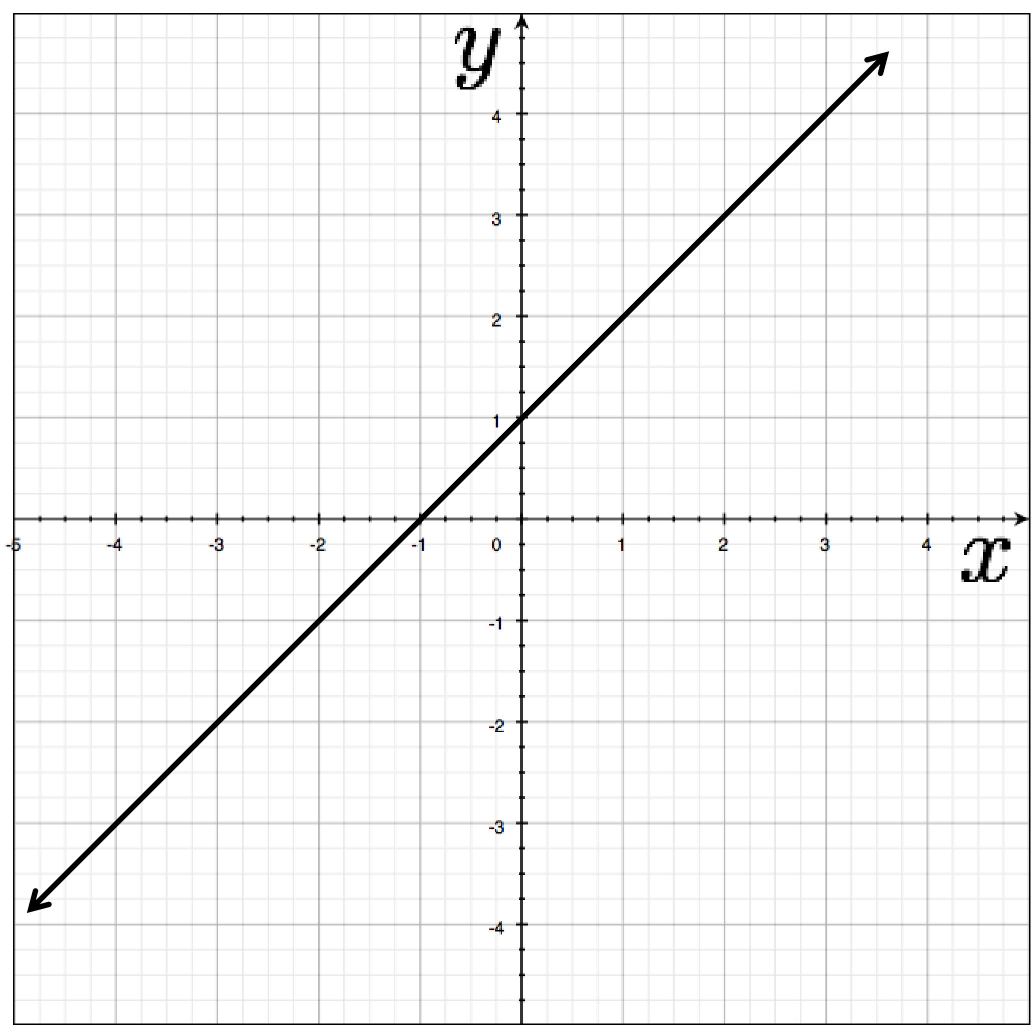



Image space

# Image and parameter space

variables

$$y = mx + b$$

parameters

variables

$$y - mx = b$$

parameters

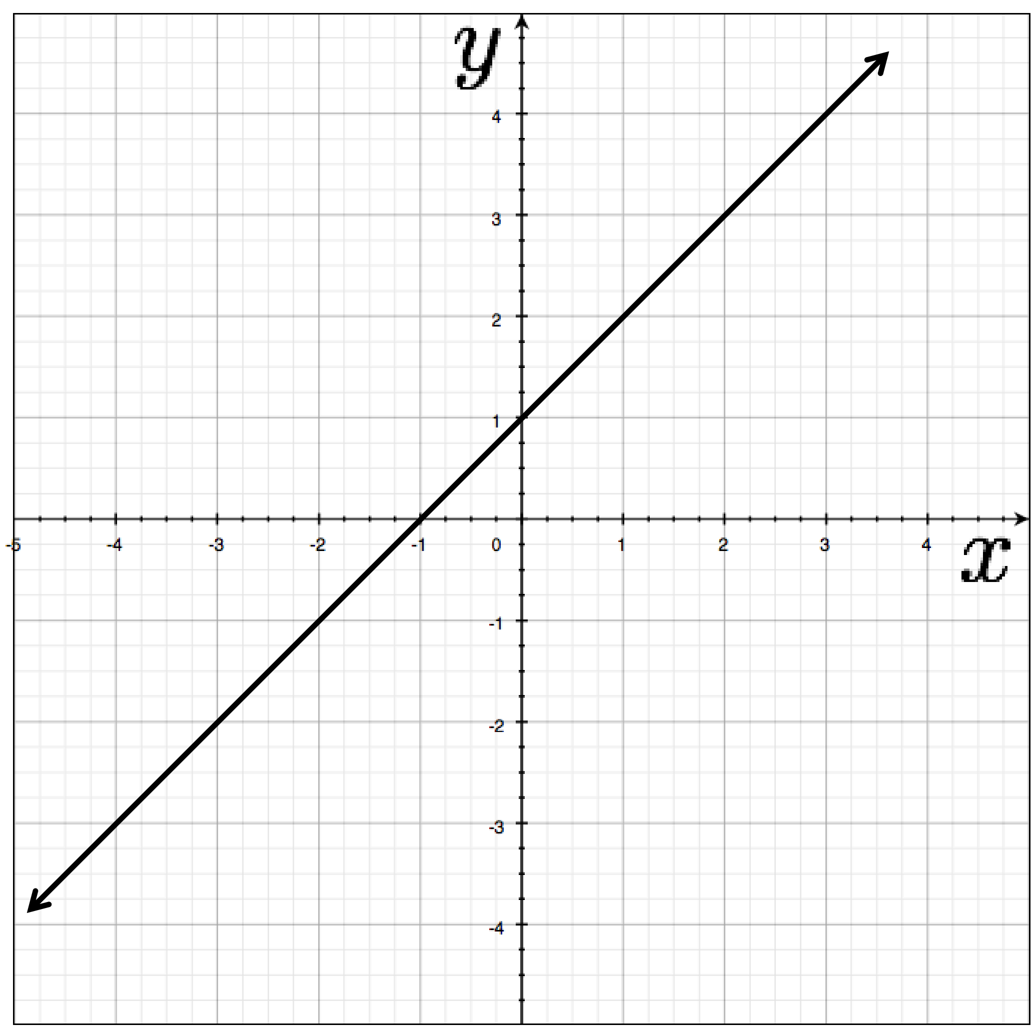
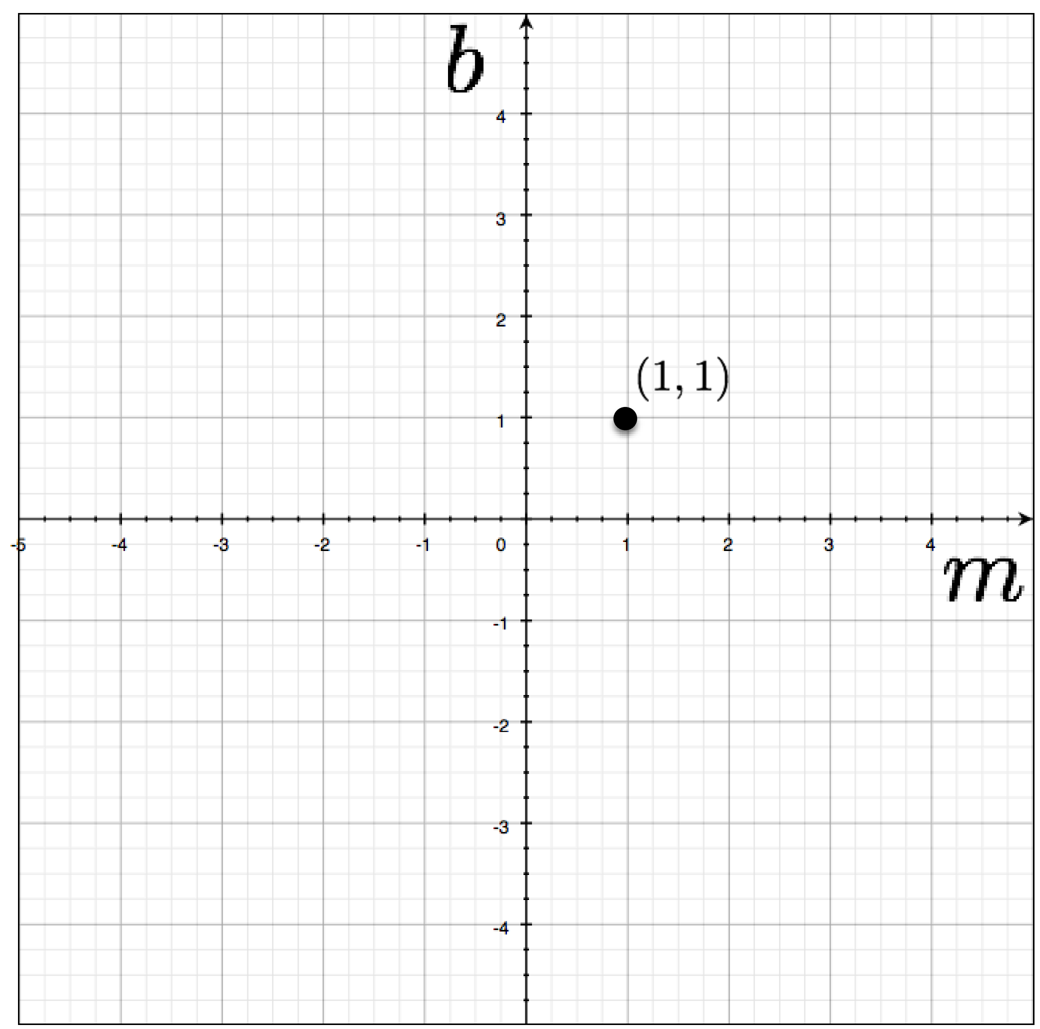



Image space

a line becomes a point




Parameter space

# Image and parameter space

variables

$$y = mx + b$$

parameters

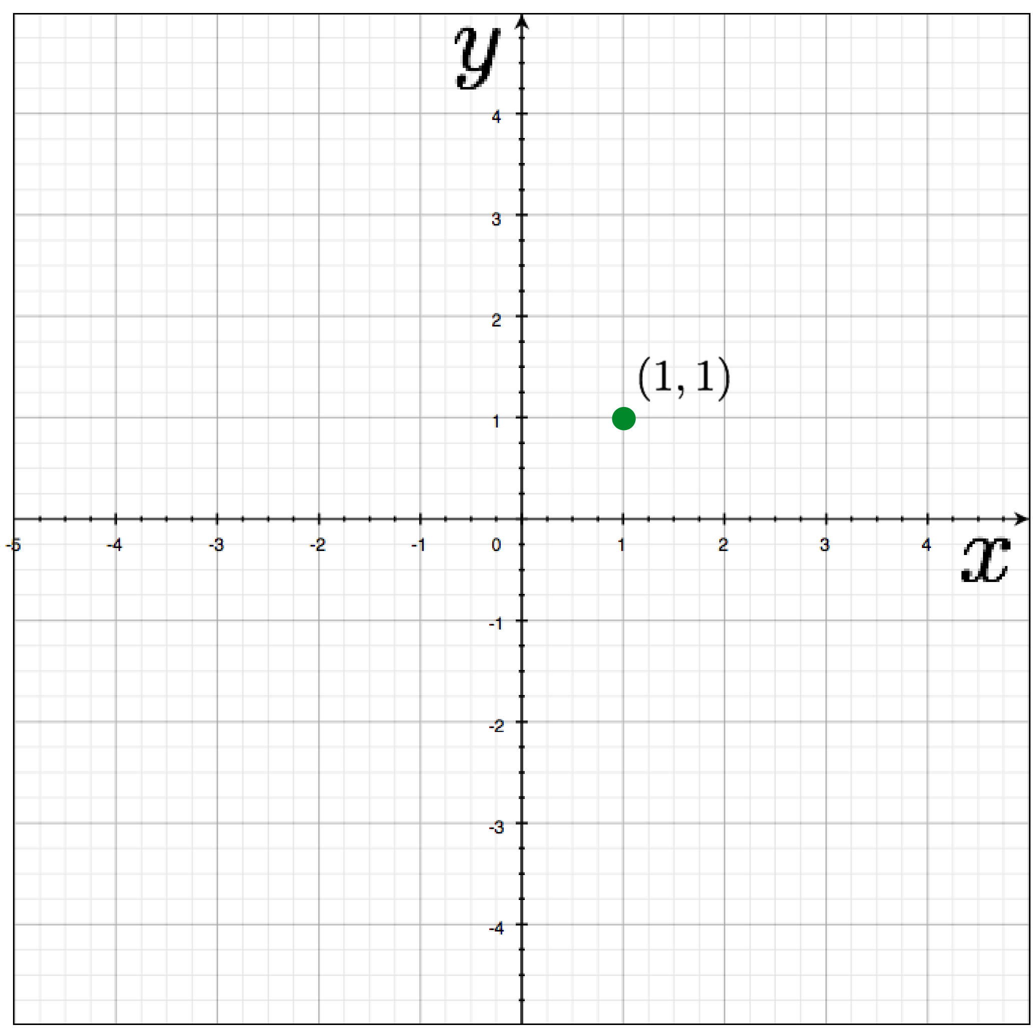



Image space

*What would a point in image space become in parameter space?*

# Image and parameter space

variables

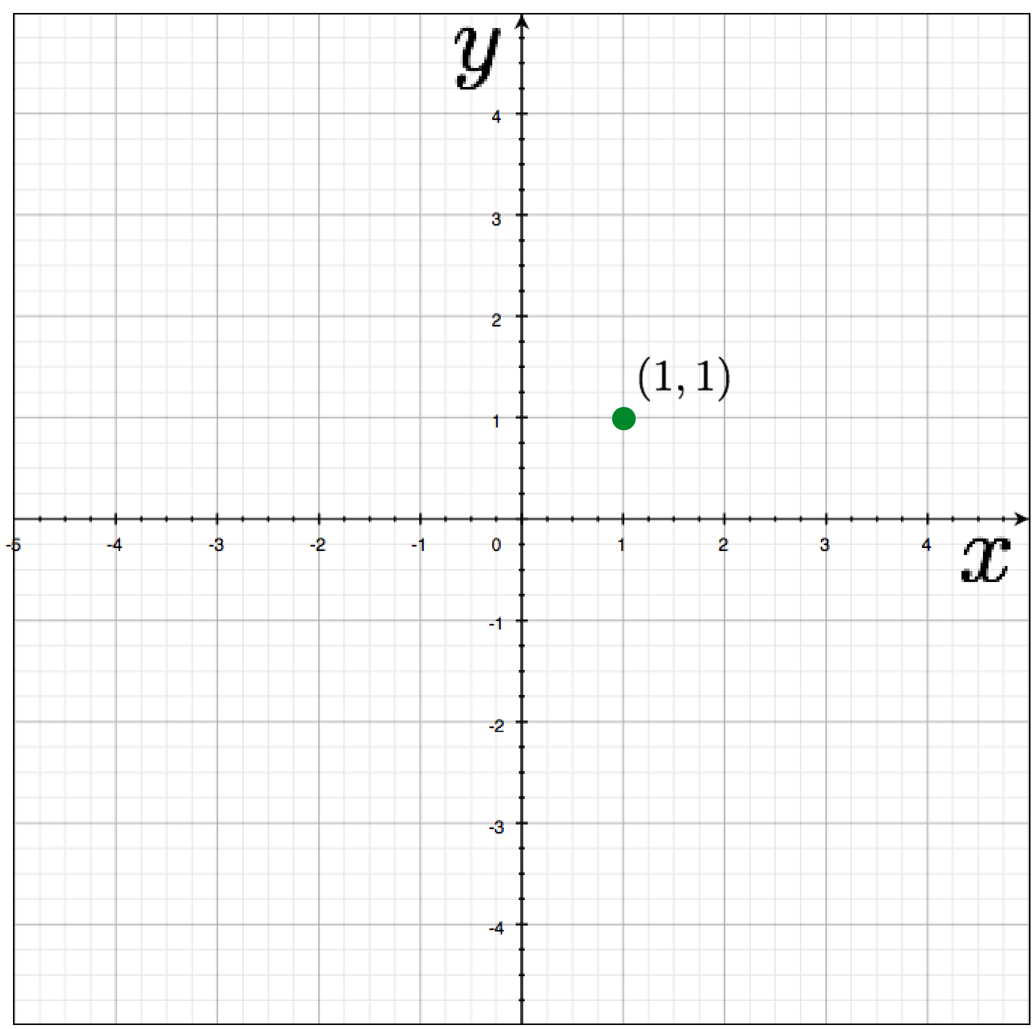
$$y = mx + b$$

parameters

variables

$$y - mx = b$$

parameters




a point becomes a line

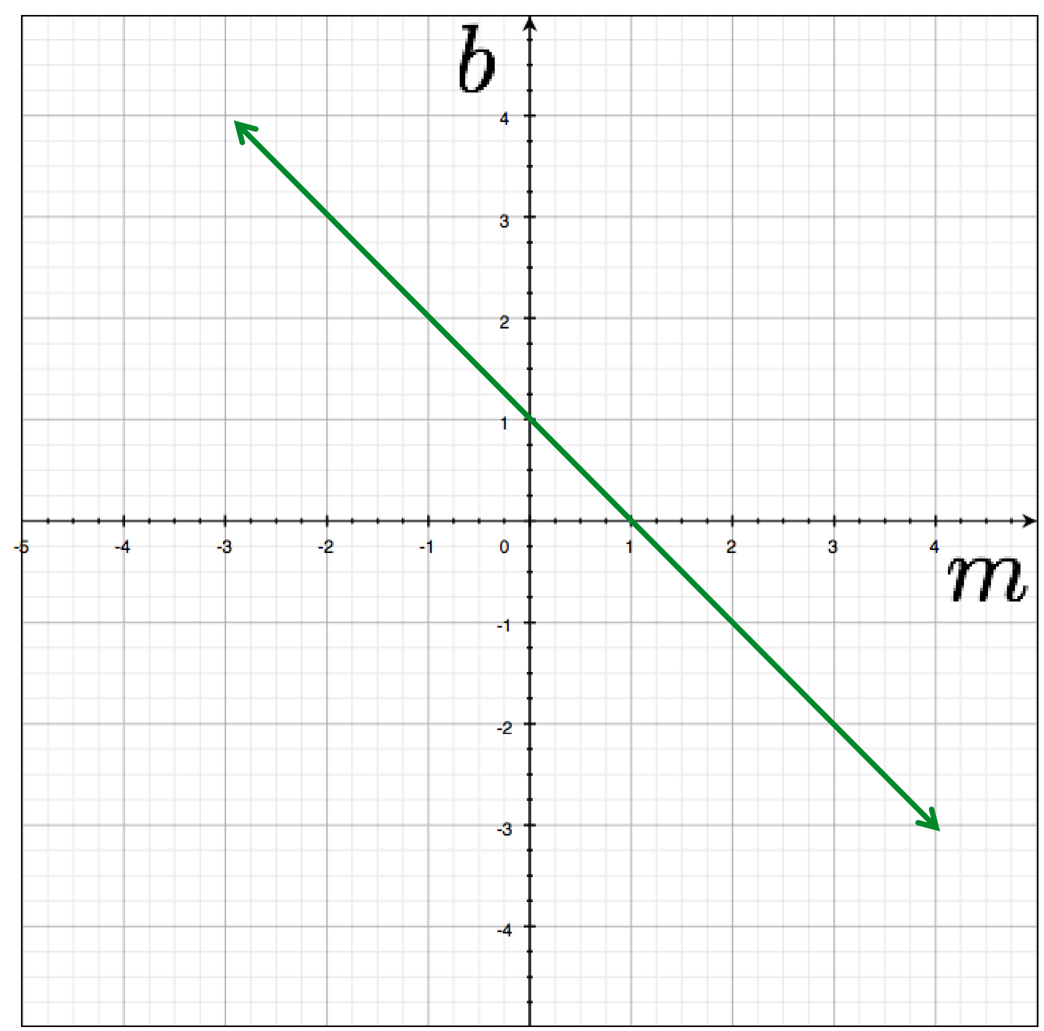



Image space

Parameter space

# Image and parameter space

variables

$$y = mx + b$$

parameters

variables

$$y - mx = b$$

parameters

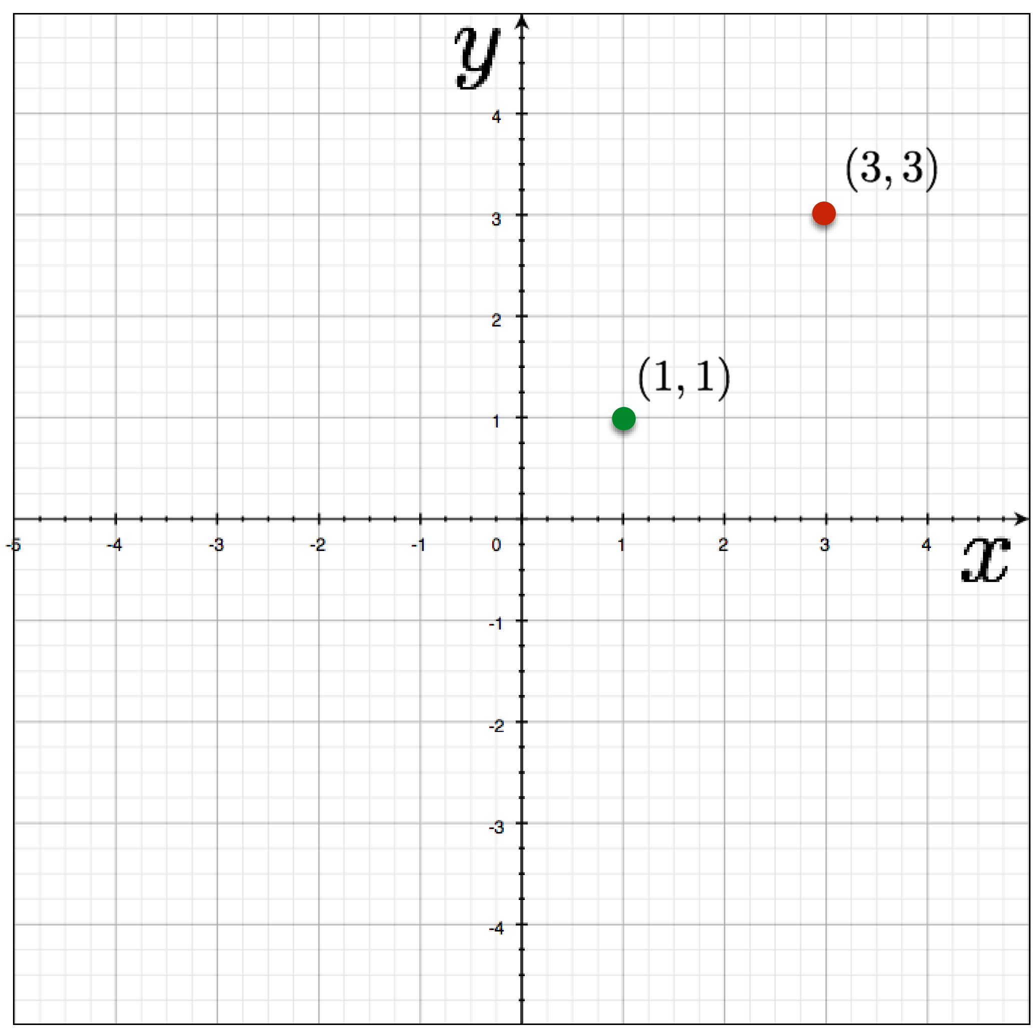
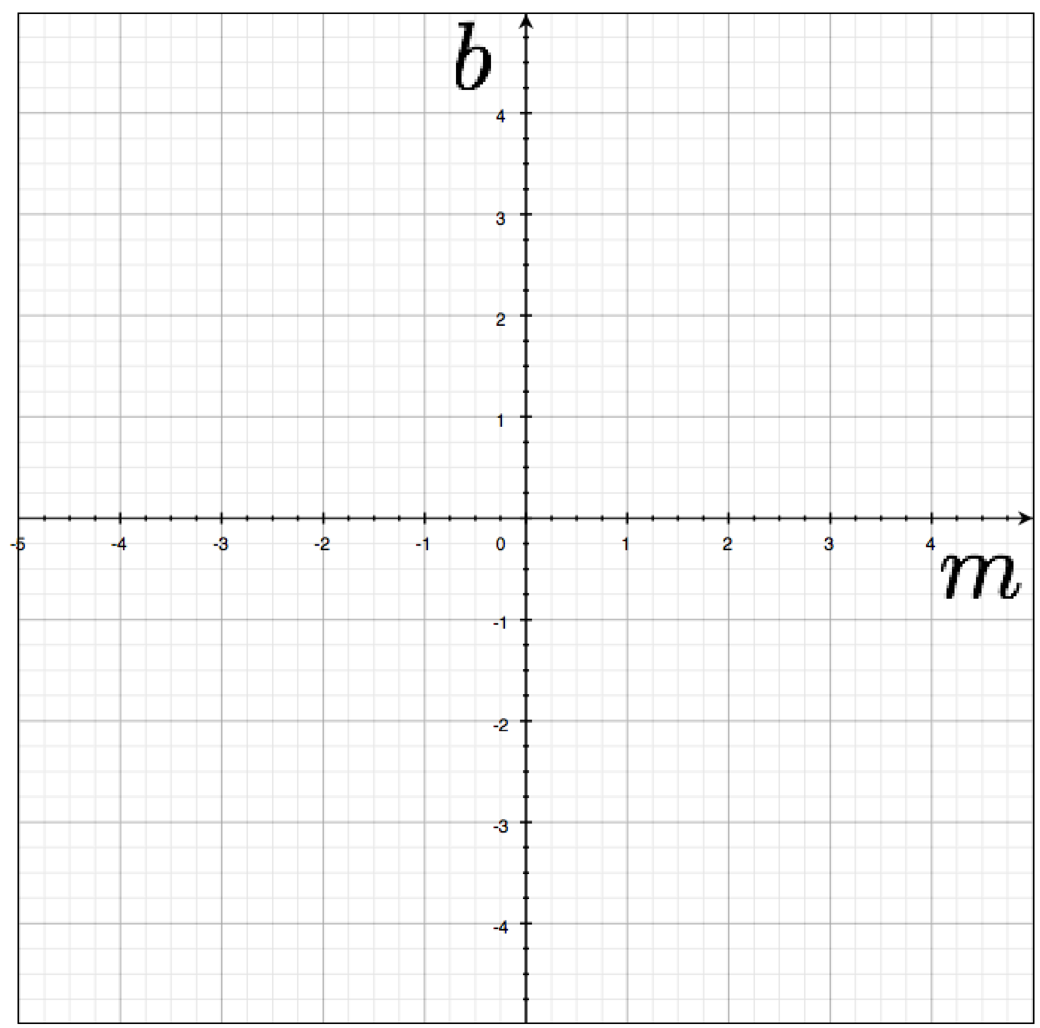



Image space

two points become ?




Parameter space

# Image and parameter space

variables

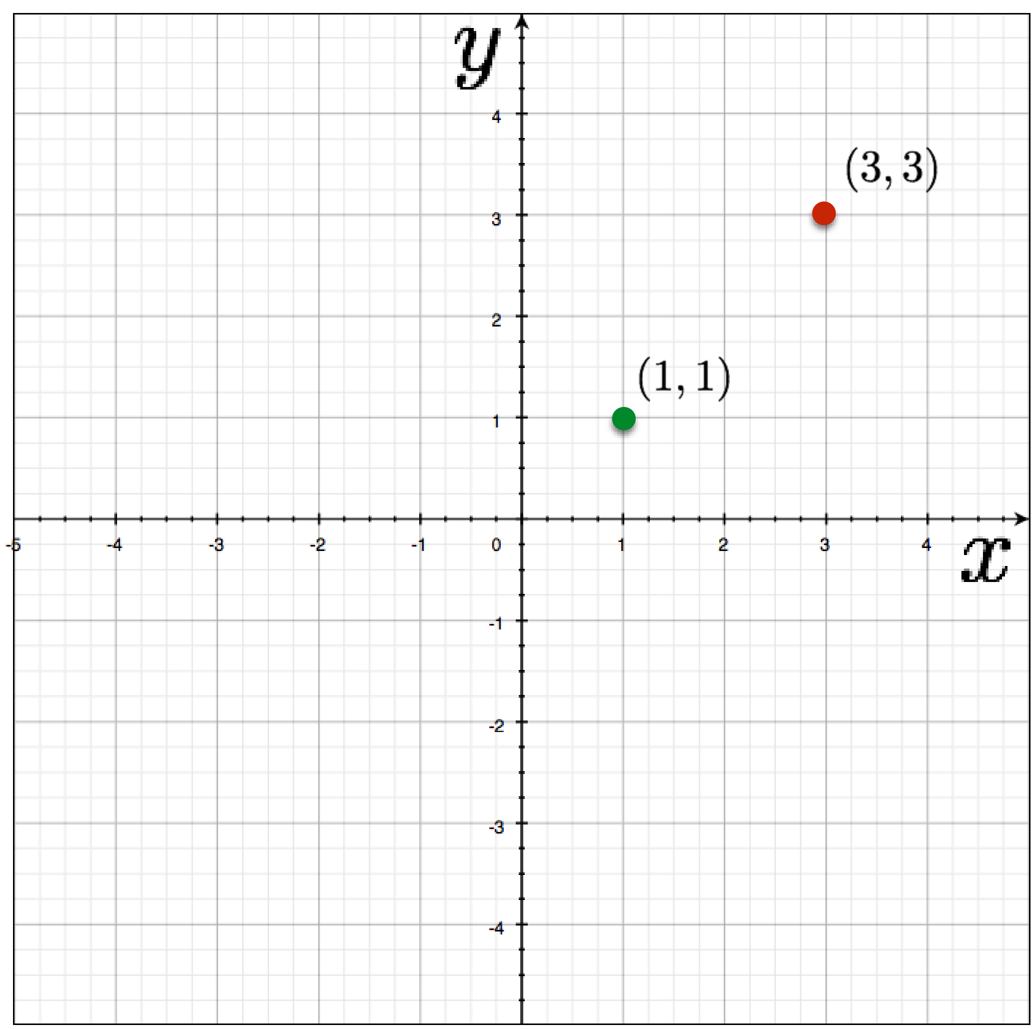
$$y = mx + b$$

parameters

variables

$$y - mx = b$$

parameters




two points become ?

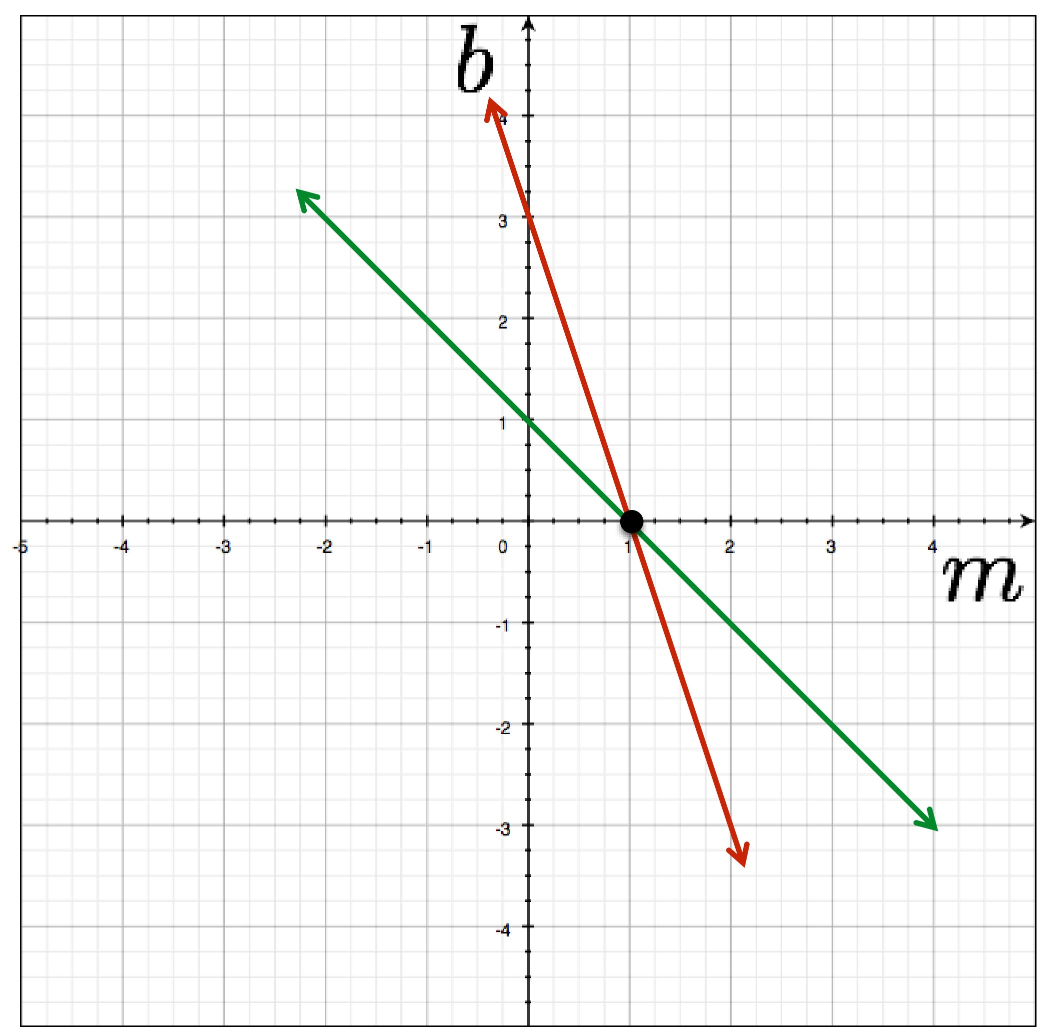



Image space

Parameter space

# Image and parameter space

variables

$$y = mx + b$$

parameters

variables

$$y - mx = b$$

parameters

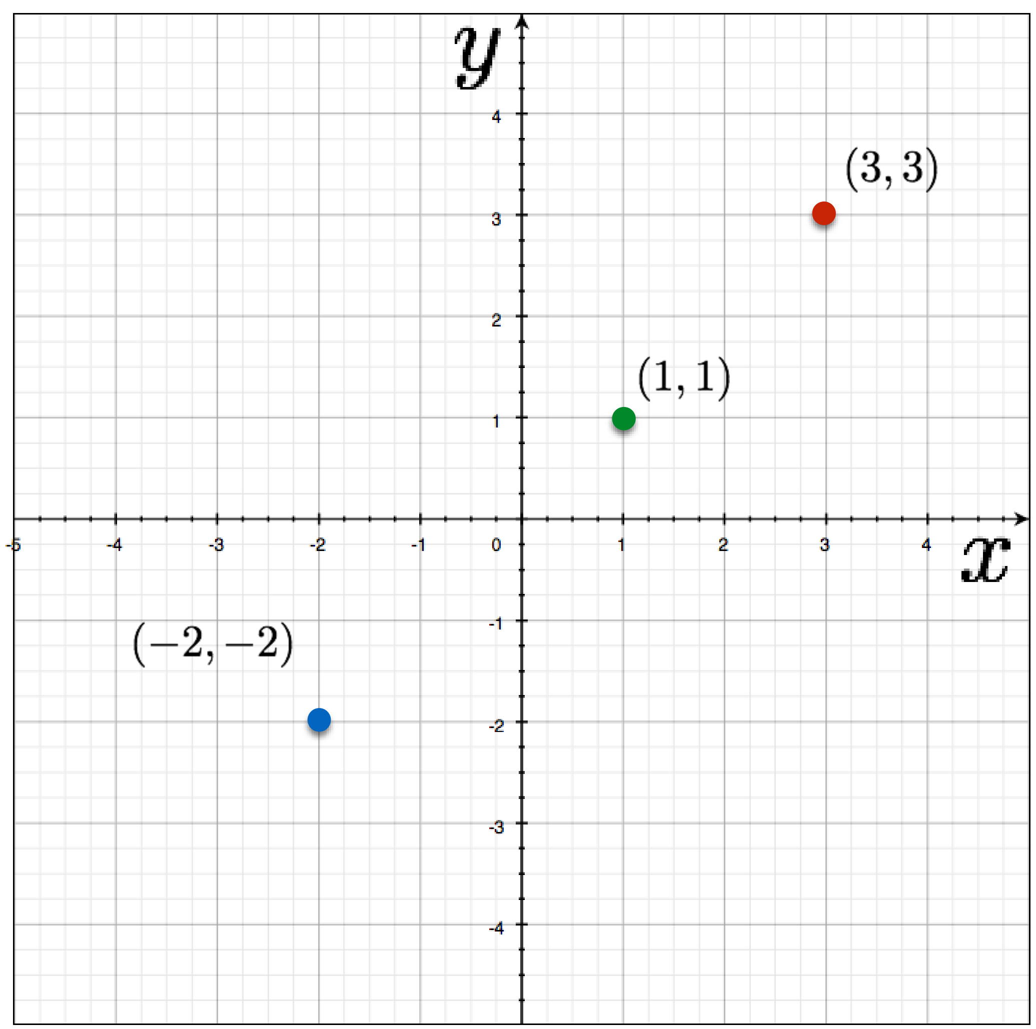
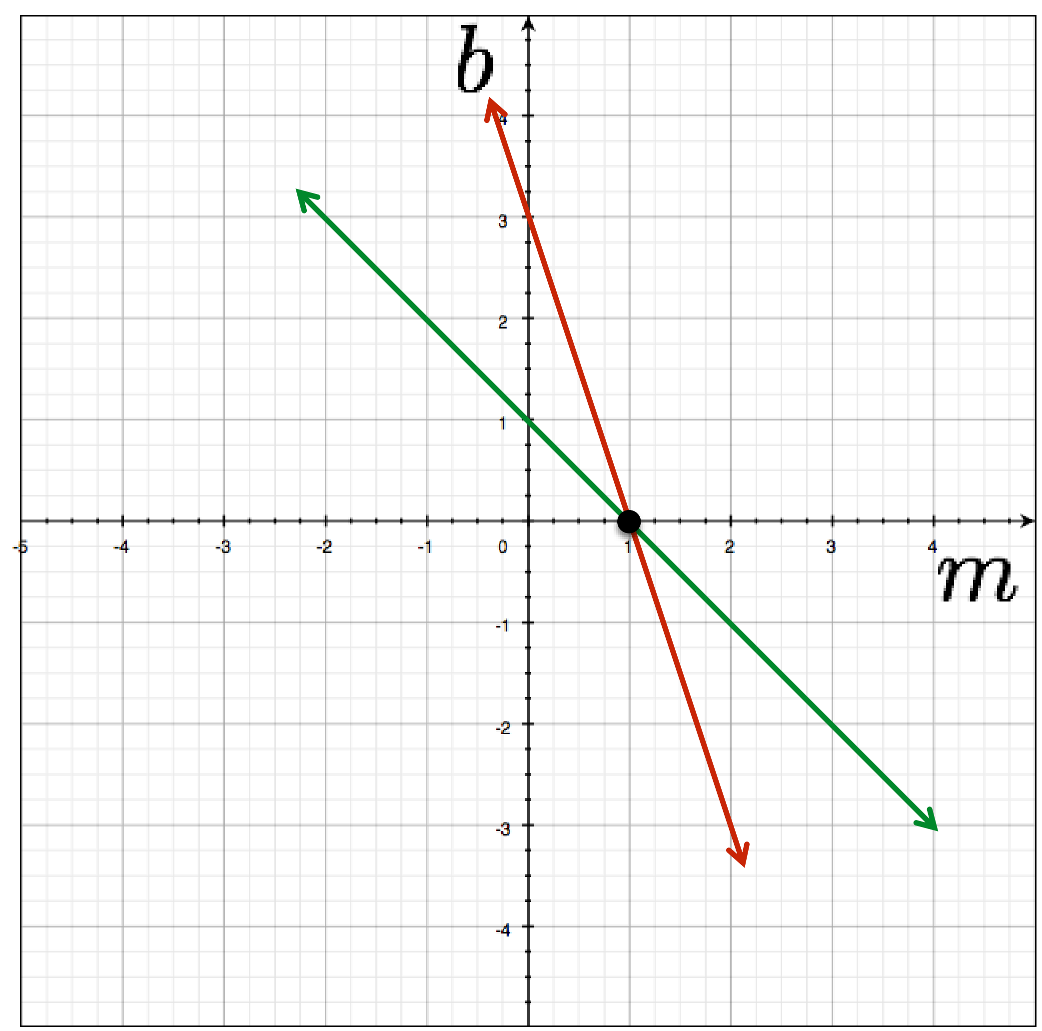



Image space

three points become ?




Parameter space

# Image and parameter space

variables

$$y = mx + b$$

parameters

variables

$$y - mx = b$$

parameters

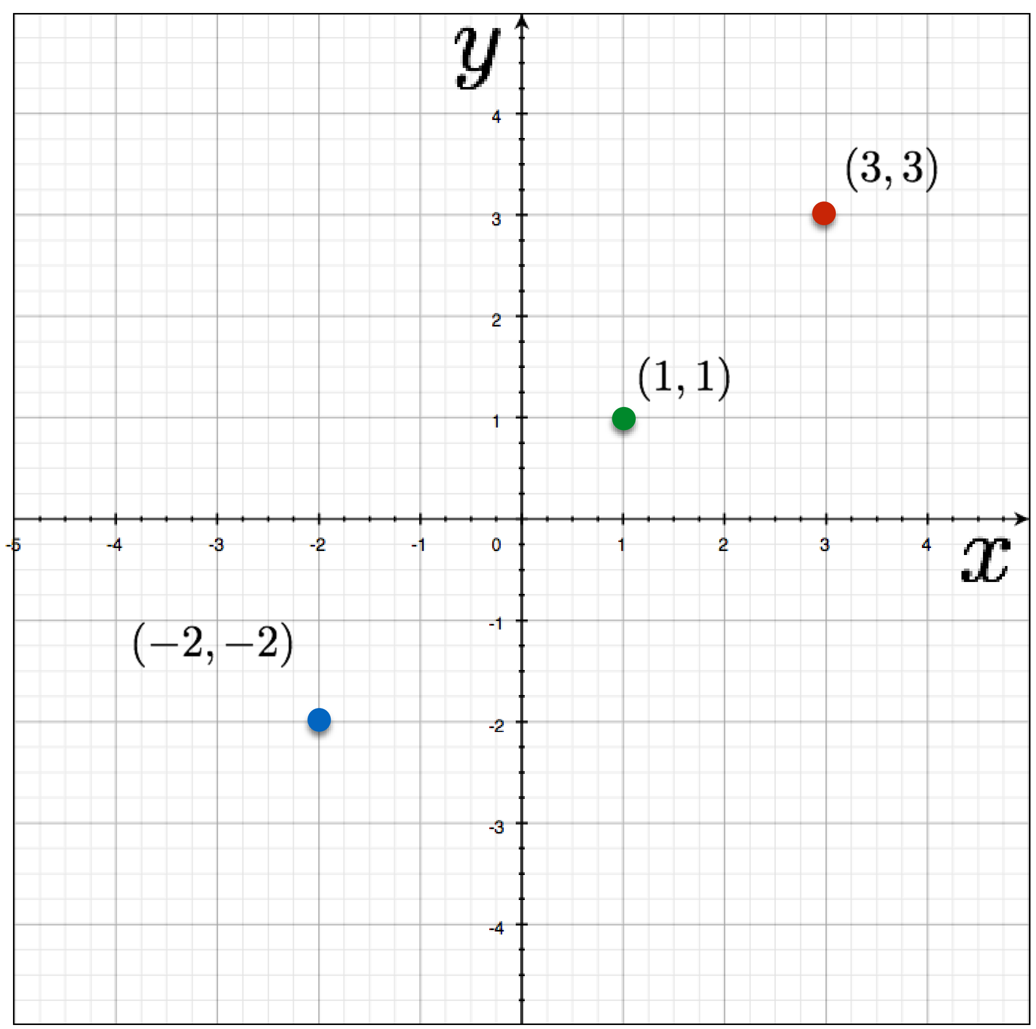
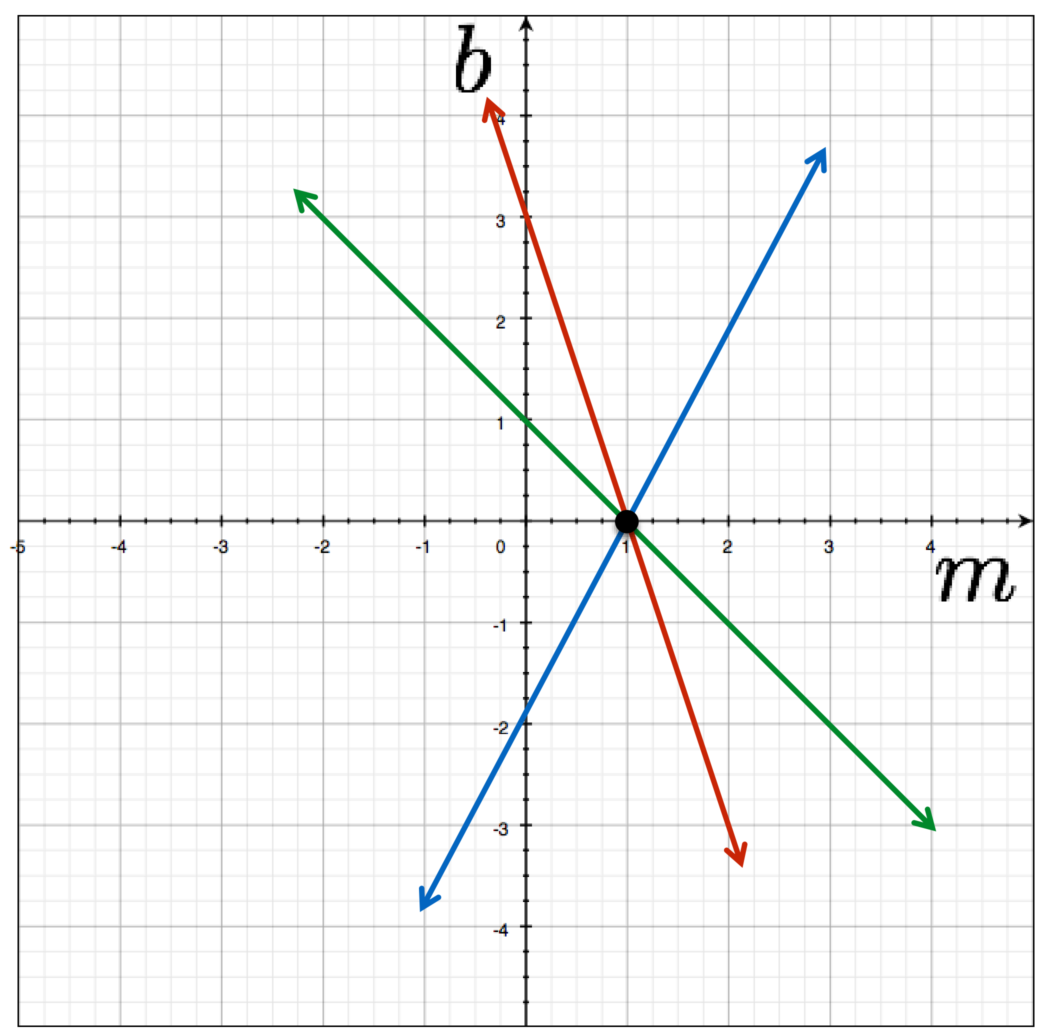



Image space

three points
become
?




Parameter space

# Image and parameter space

variables

$$y = mx + b$$

parameters

variables

$$y - mx = b$$

parameters

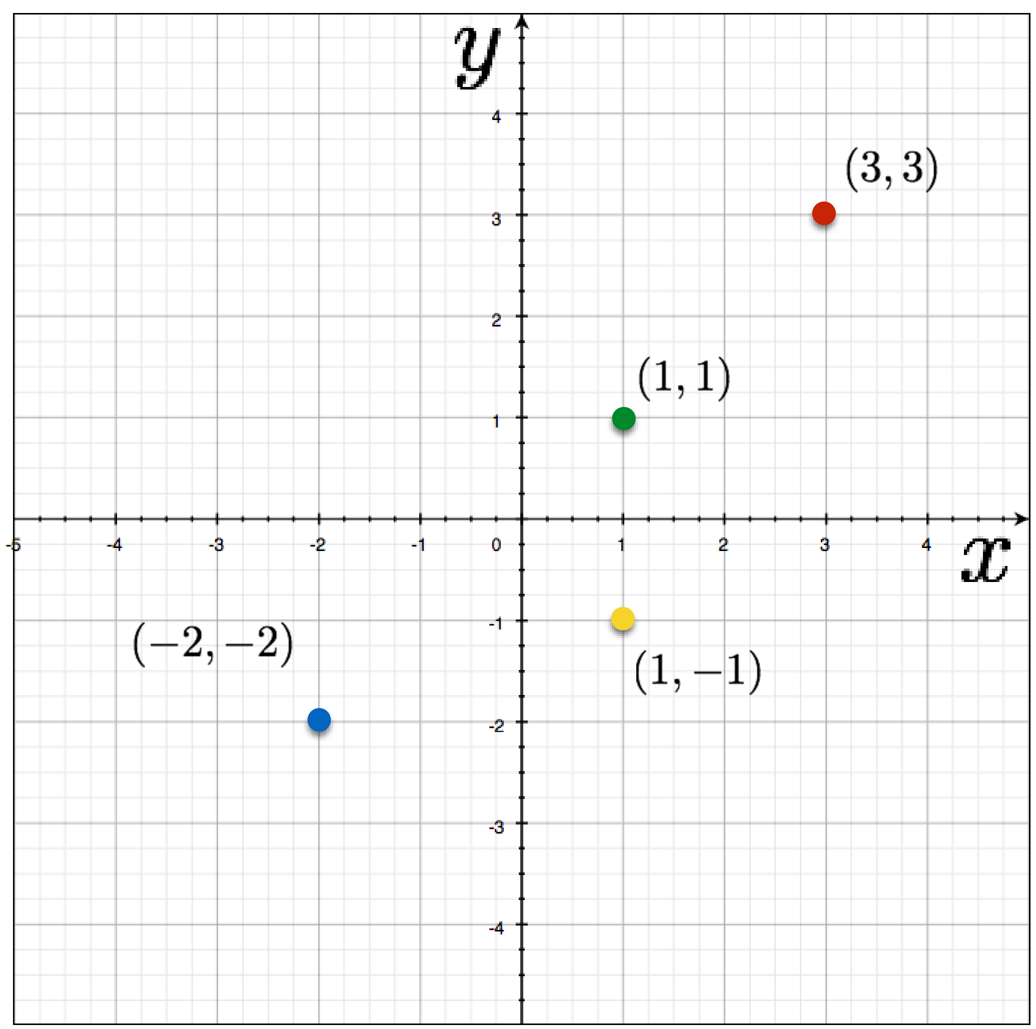
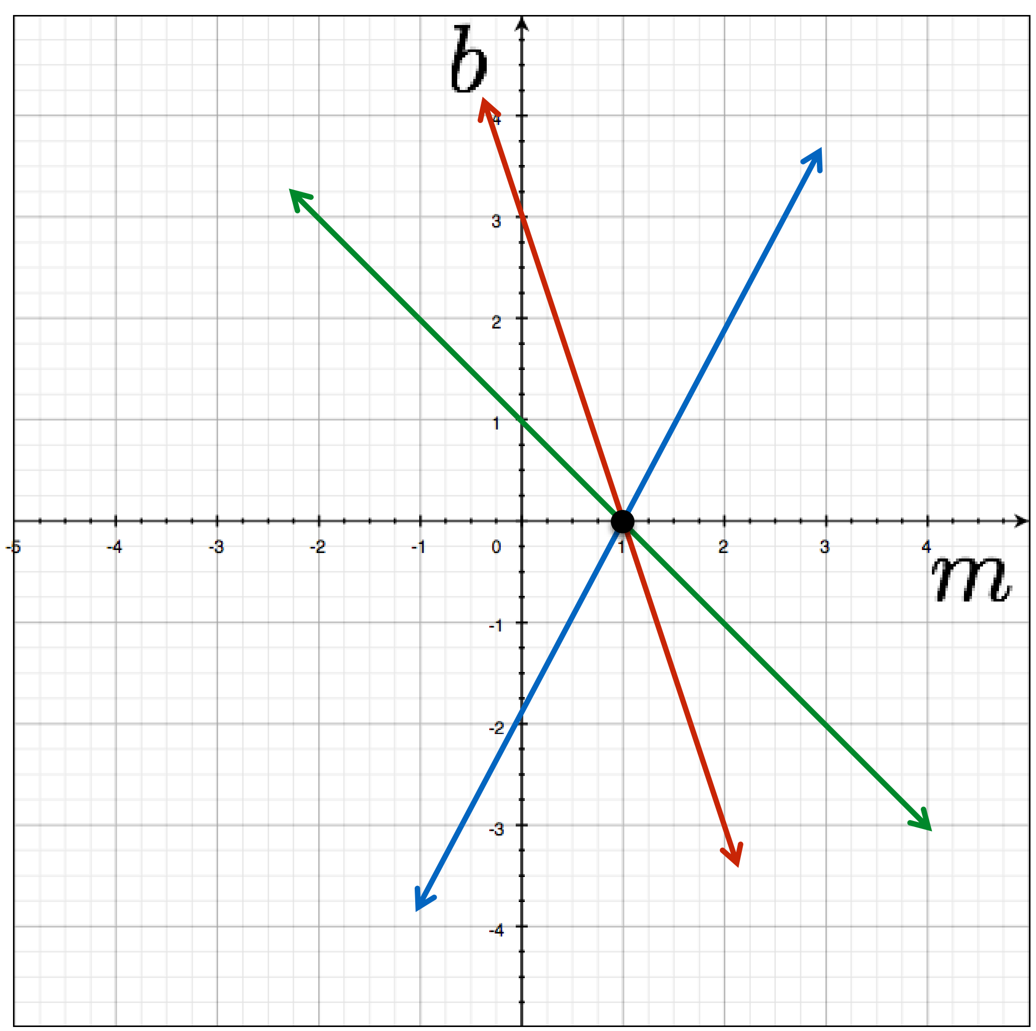



Image space


Parameter space

# Image and parameter space

variables

$$y = mx + b$$

parameters

variables

$$y - mx = b$$

parameters

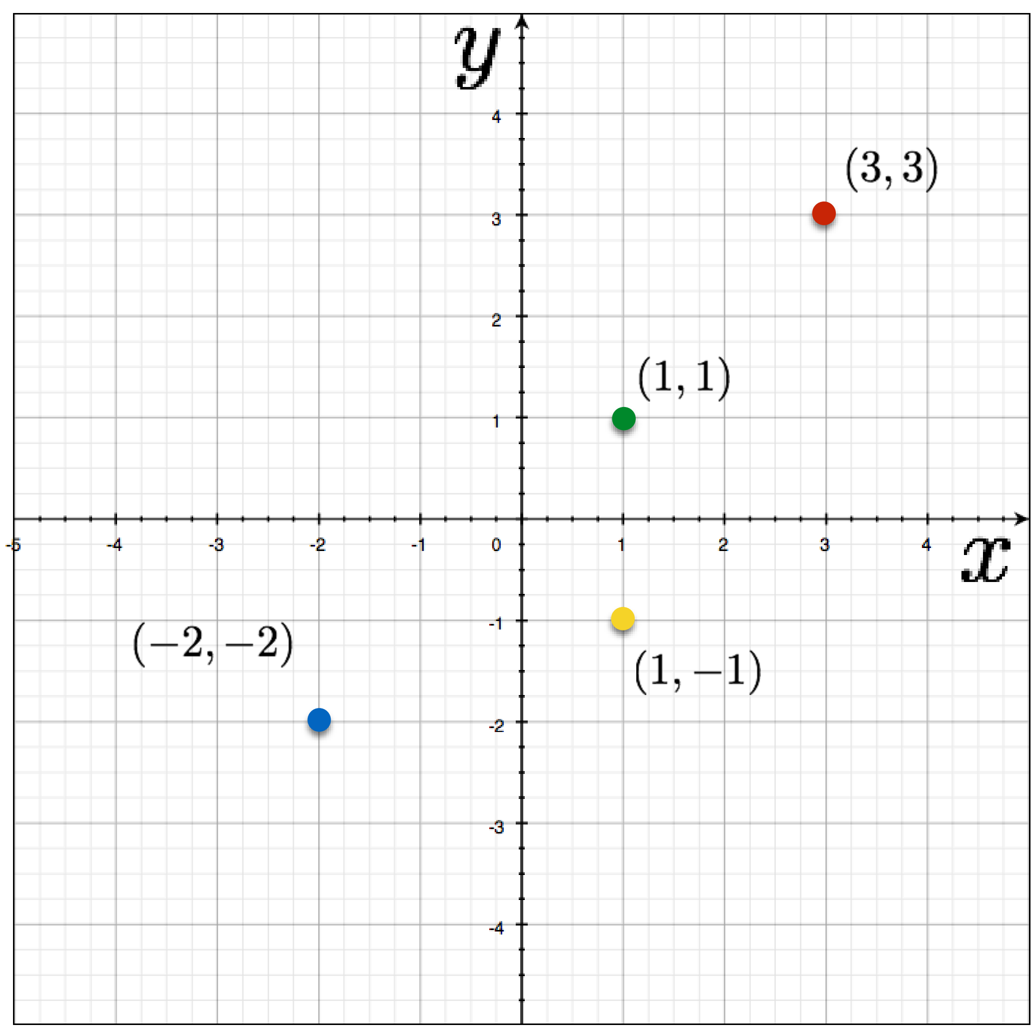
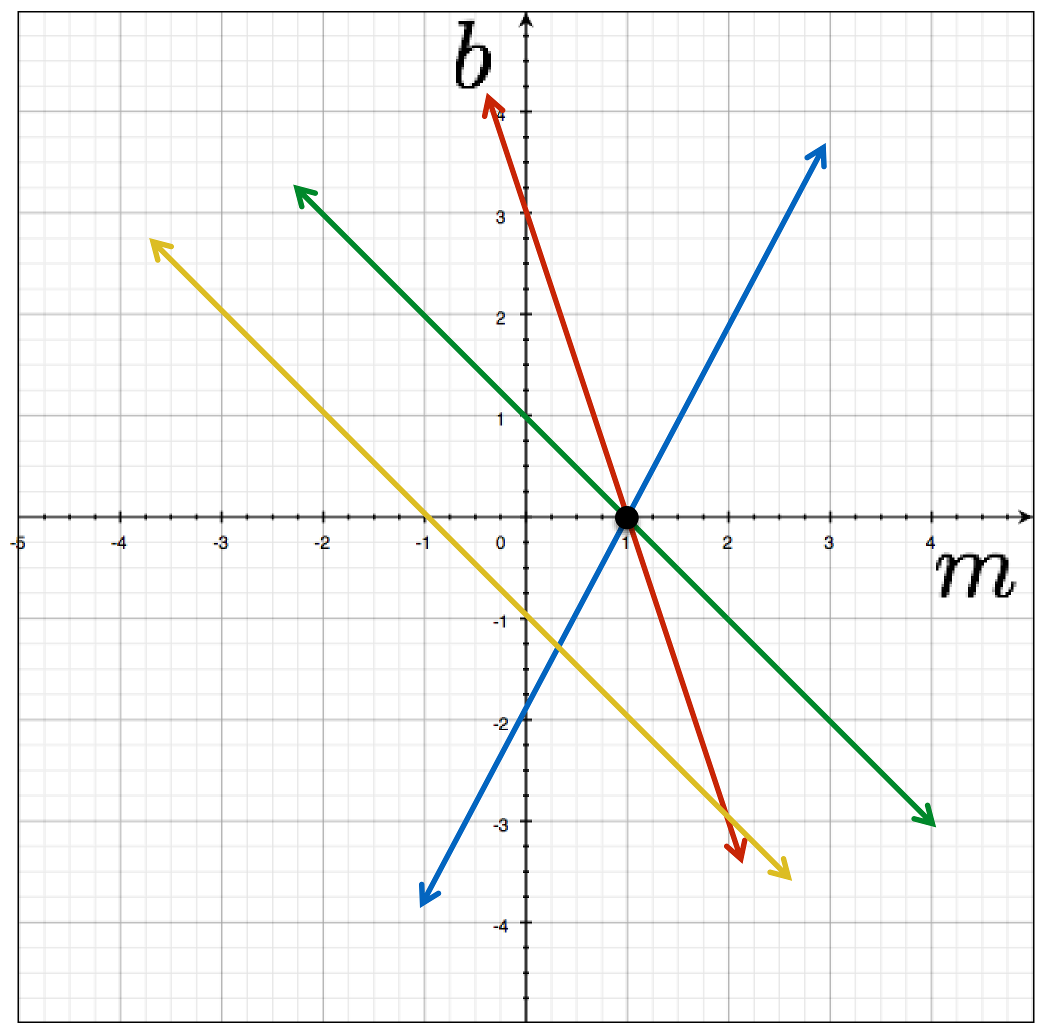



Image space

four points become ?




Parameter space

# How would you find the best fitting line?

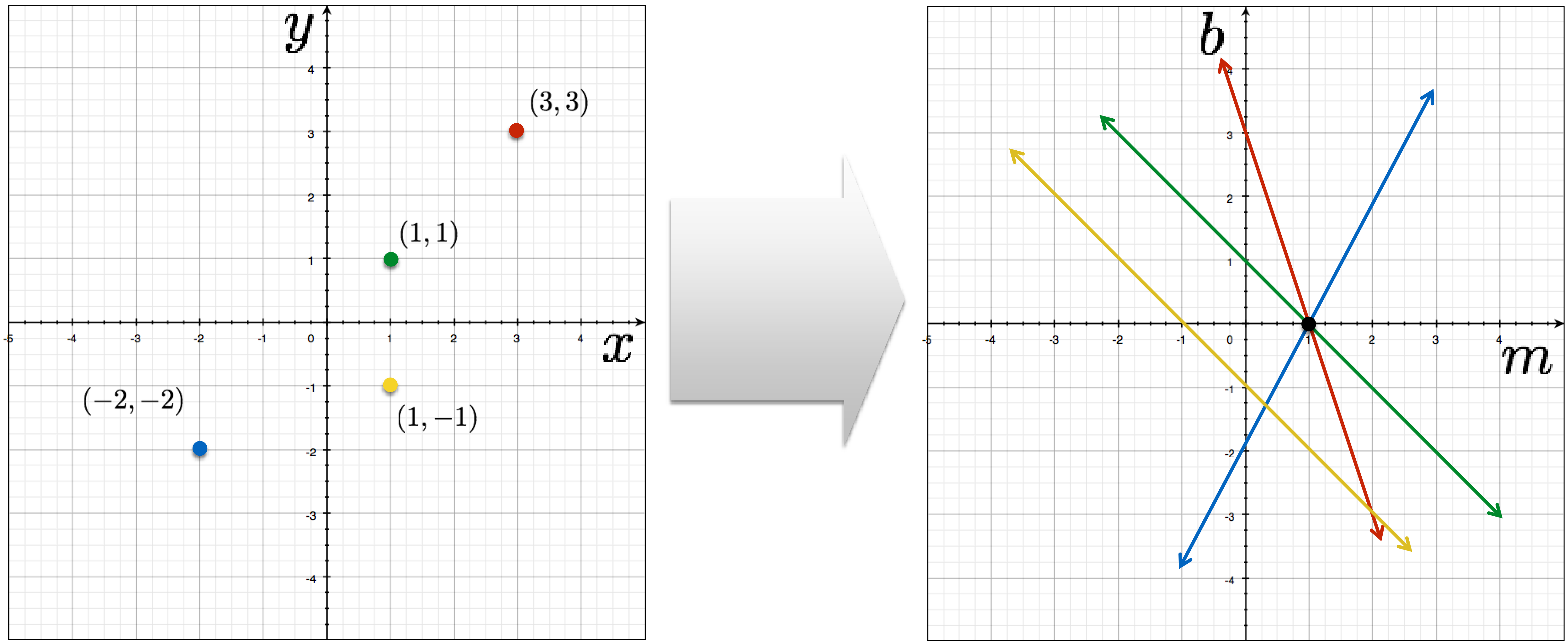



Image space

Parameter space

*Is this method robust to measurement noise?*

*Is this method robust to outliers?*

# Line Detection by Hough Transform

**Algorithm:**

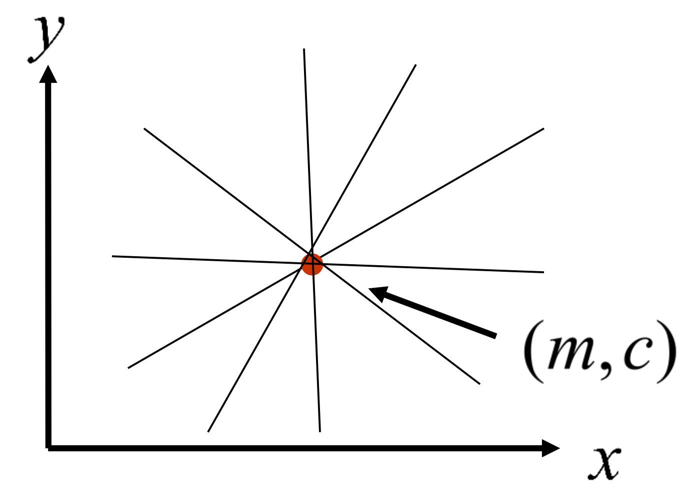
1.Quantize Parameter Space $(m, c)$

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 $A(m, c)$



Parameter Space

$A(m, c)$

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 1 | | | | | | 1 | | |
| | | 1 | | | | 1 | | | |
| | | | 1 | | 1 | | | | |
| | | | | 2 | | | | | |
| | | | 1 | | 1 | | | | |
| | | 1 | | | | 1 | | | |
| | 1 | | | | | | 1 | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |

# Problems with parameterization

*How big does the accumulator need to be for the parameterization* $(m, c)$*?*

$A(m, c)$

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 1 | | | | | | 1 | | |
| | | 1 | | | | 1 | | | |
| | | | 1 | | 1 | | | | |
| | | | | 2 | | | | | |
| | | | 1 | | 1 | | | | |
| | | 1 | | | | 1 | | | |
| | 1 | | | | | | 1 | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |

# Problems with parameterization

*How big does the accumulator need to be for the parameterization* $(m,c)$*?*

$A(m,c)$

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 1 | | | | | | 1 | | |
| | | 1 | | | | 1 | | | |
| | | | 1 | | 1 | | | | |
| | | | | 2 | | | | | |
| | | | 1 | | 1 | | | | |
| | | 1 | | | | 1 | | | |
| | 1 | | | | | | 1 | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |

The space of m is huge!

$$-\infty \leq m \leq \infty$$

The space of c is huge!

$$-\infty \leq c \leq \infty$$

# Better Parameterization

Use normal form:

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

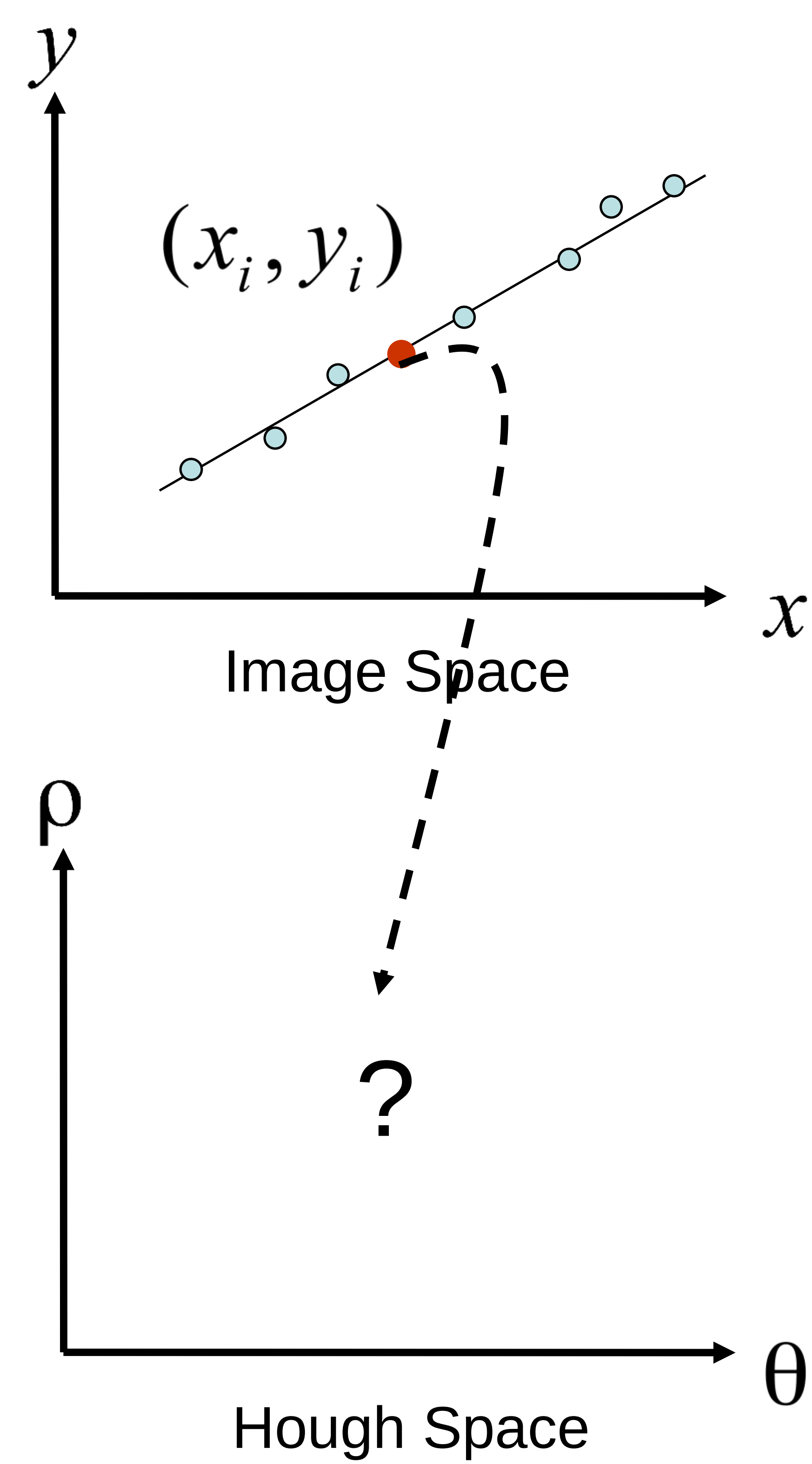
Given points $(x_i, y_i)$ find $(\rho, \theta)$

Hough Space Sinusoid

$$0 \le \theta \le 2\pi$$

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

(Finite Accumulator Array Size)



# Image and parameter space

variables

$$y = mx + b$$

parameters

parameters

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

variables

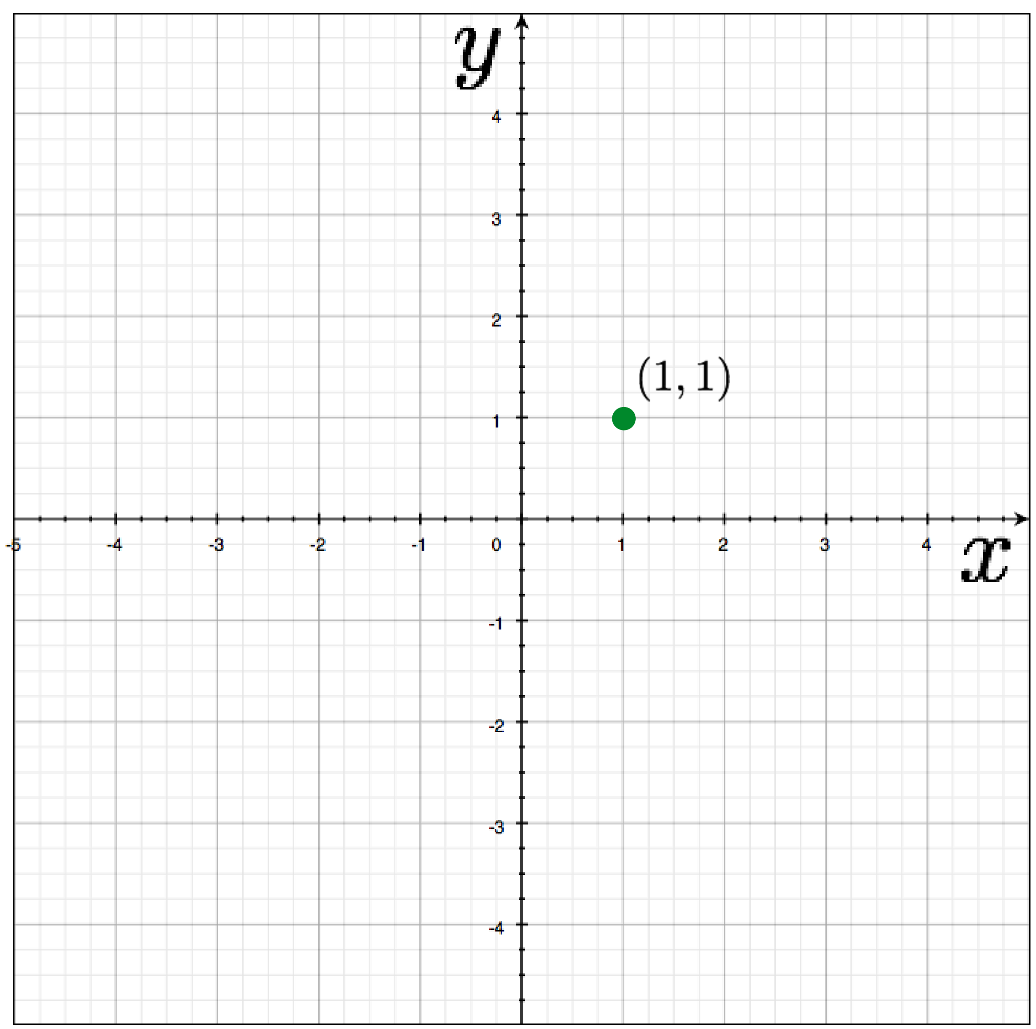
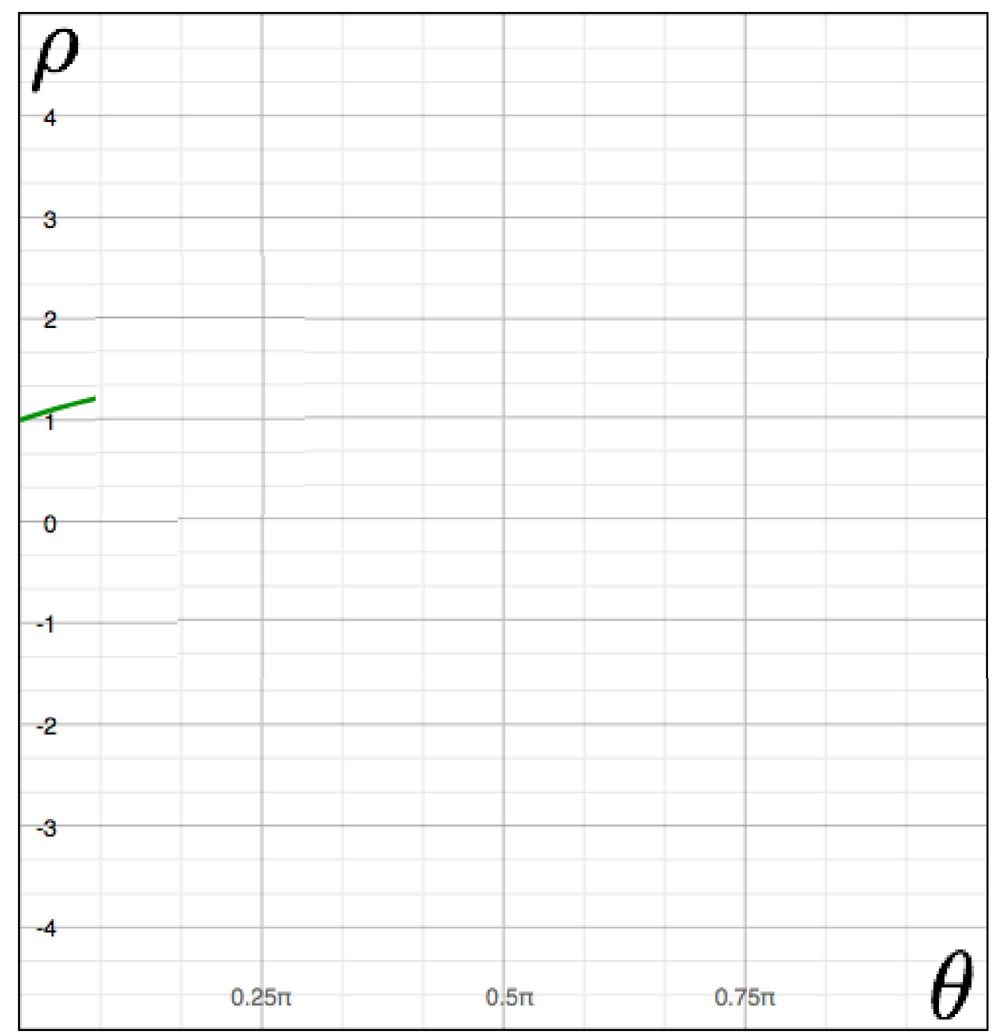



Image space

a point becomes?




Parameter space

# Image and parameter space

variables

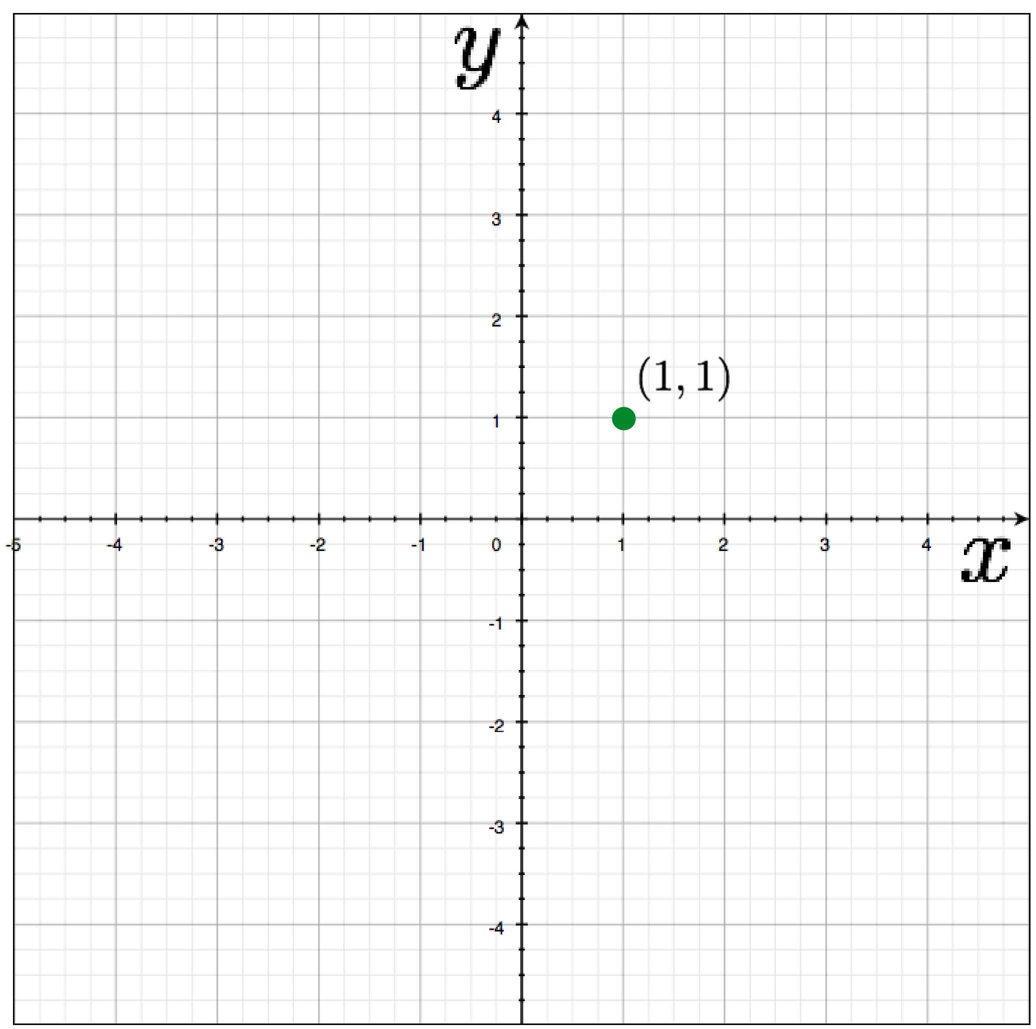
$$y = mx + b$$

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

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

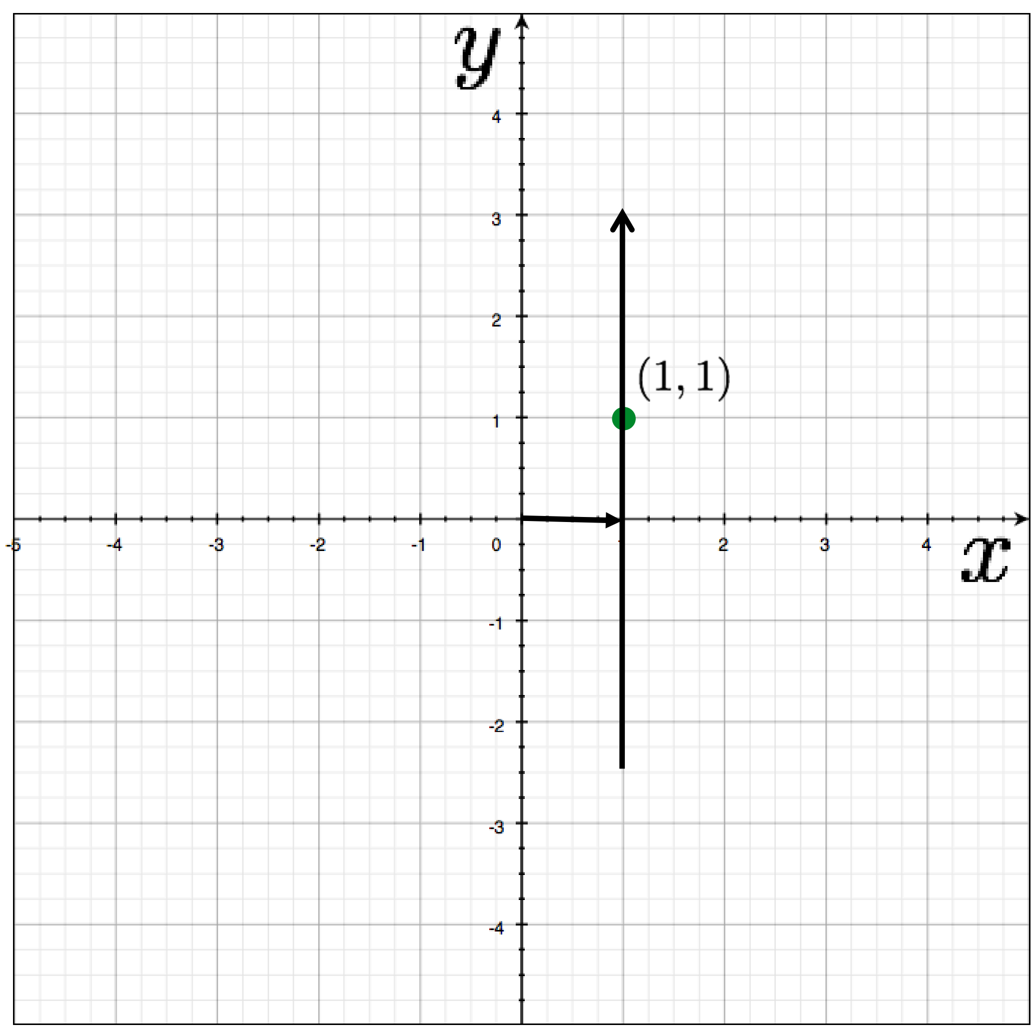
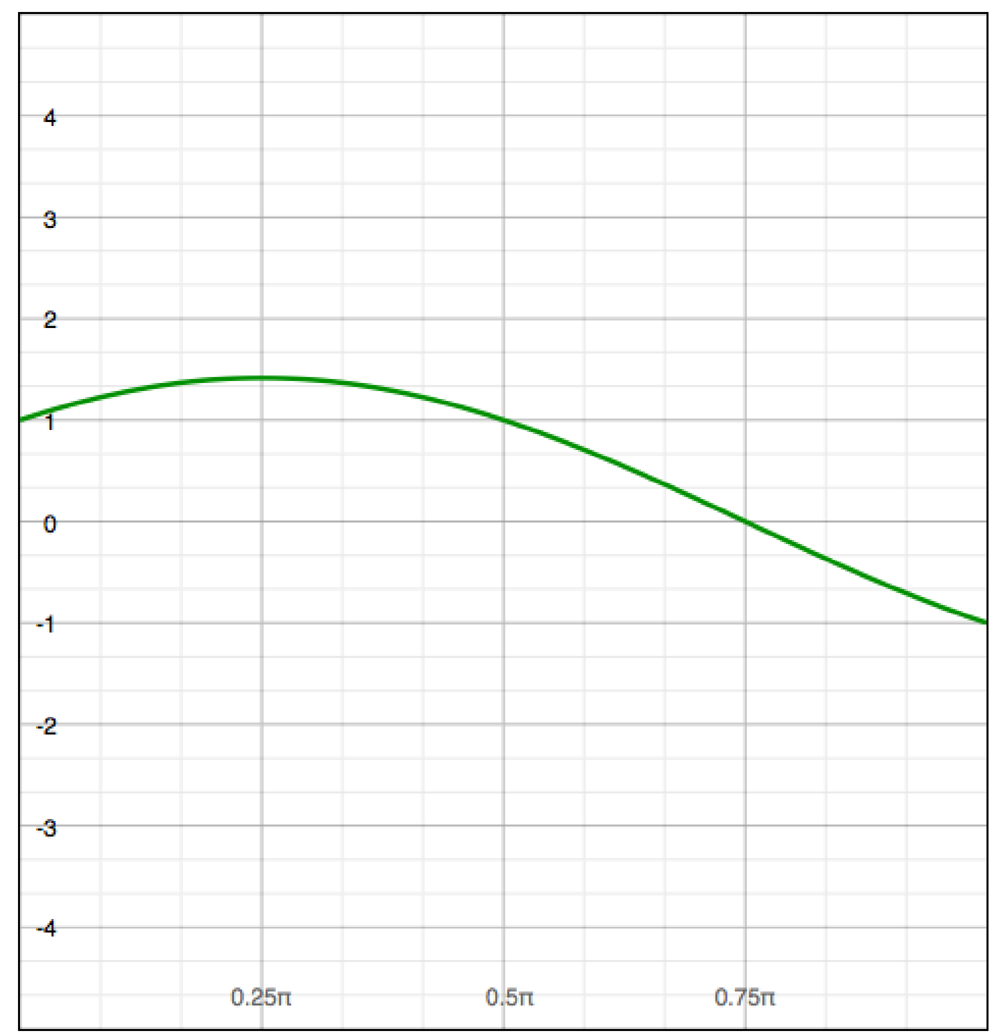



Image space

a line becomes?




Parameter space

# Image and parameter space

variables

$$y = mx + b$$

parameters

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

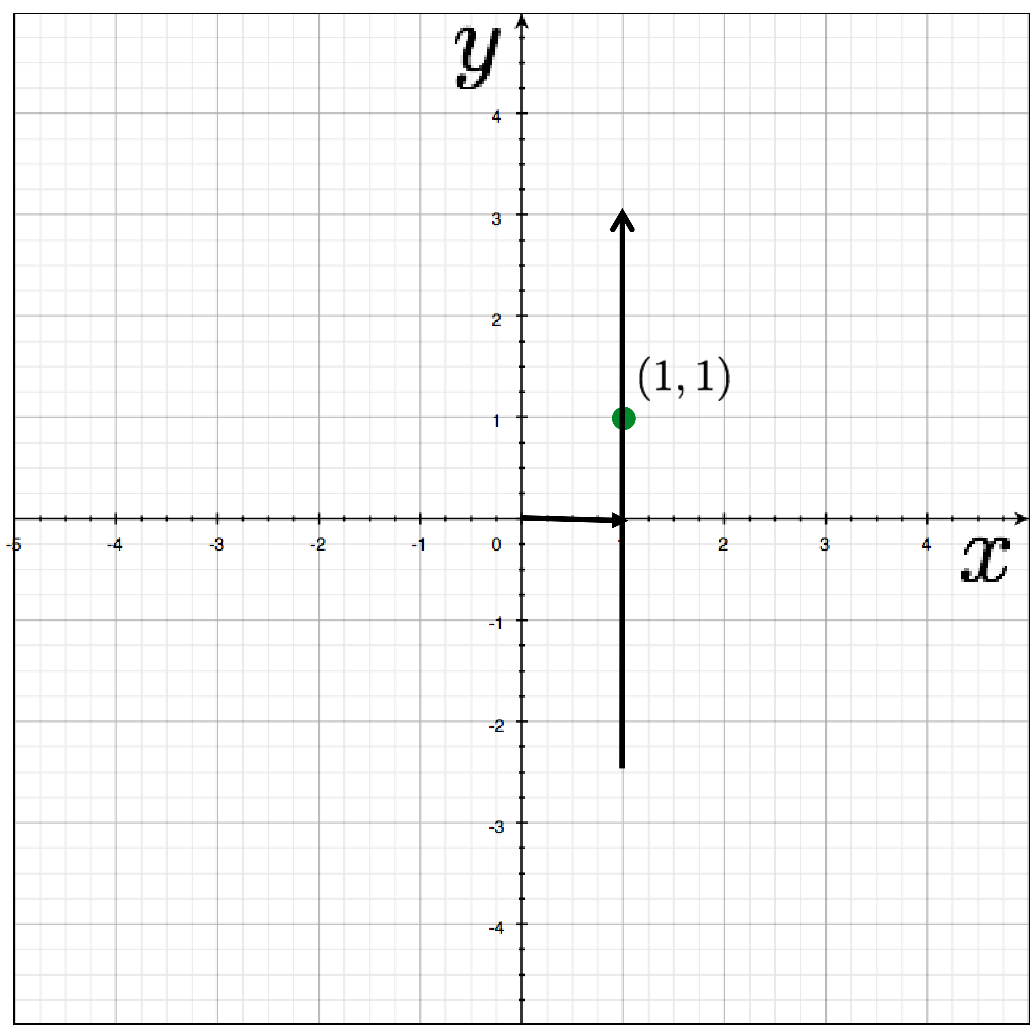
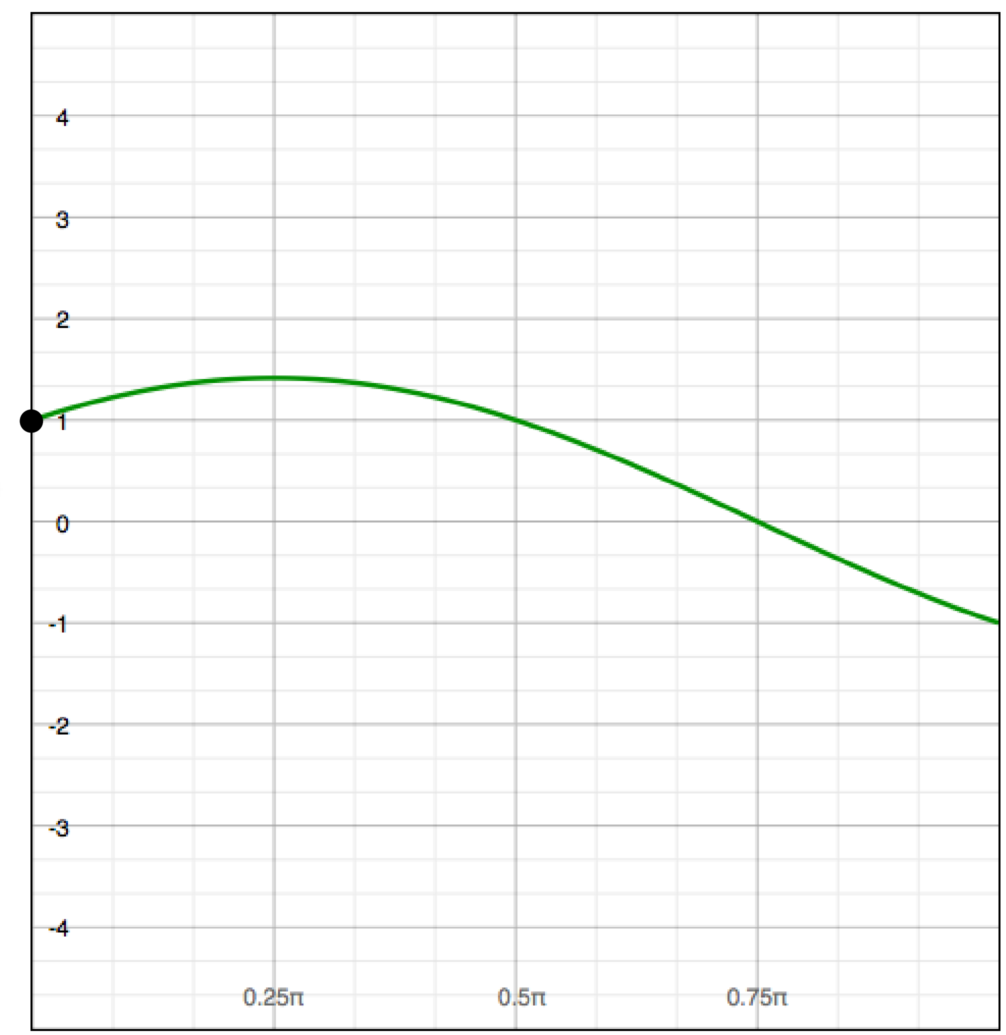



Image space

a line becomes a point




Parameter space

# Image and parameter space

variables

$$y = mx + b$$

parameters

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

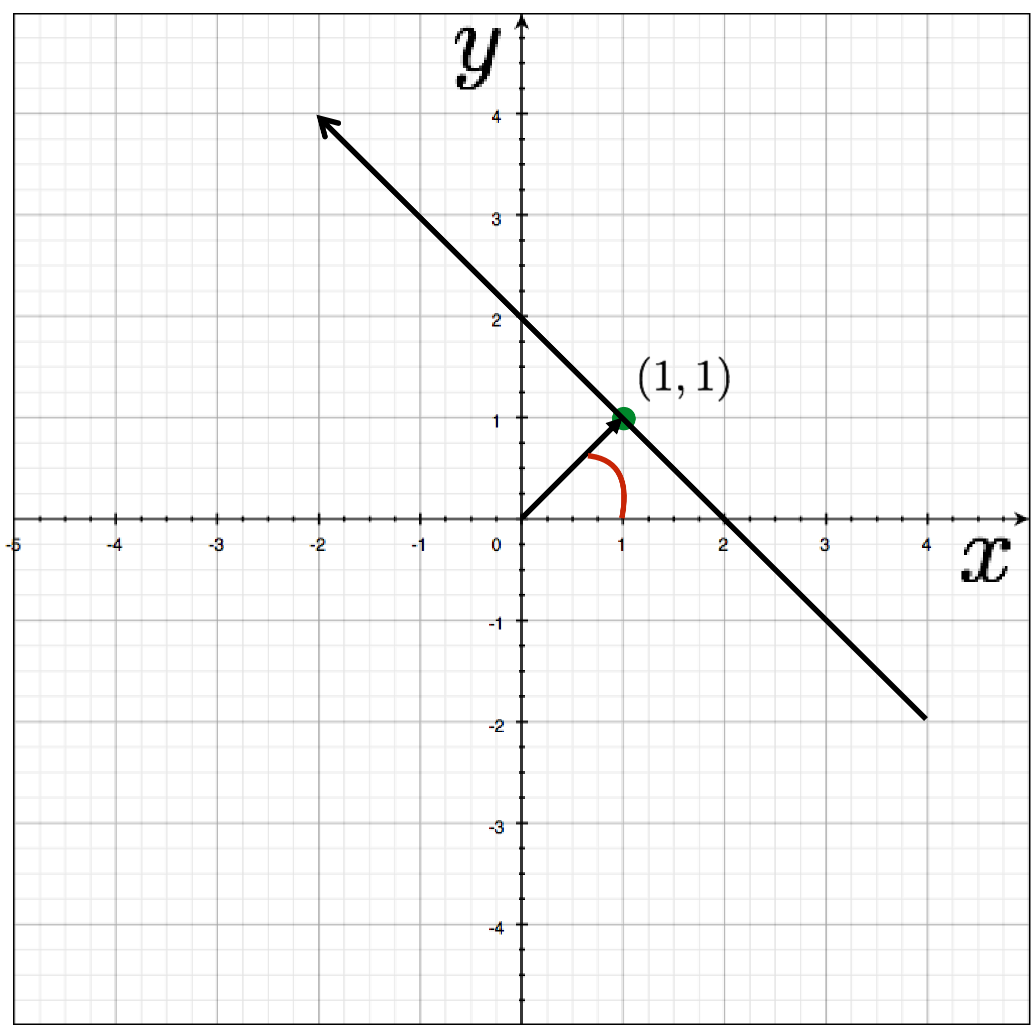
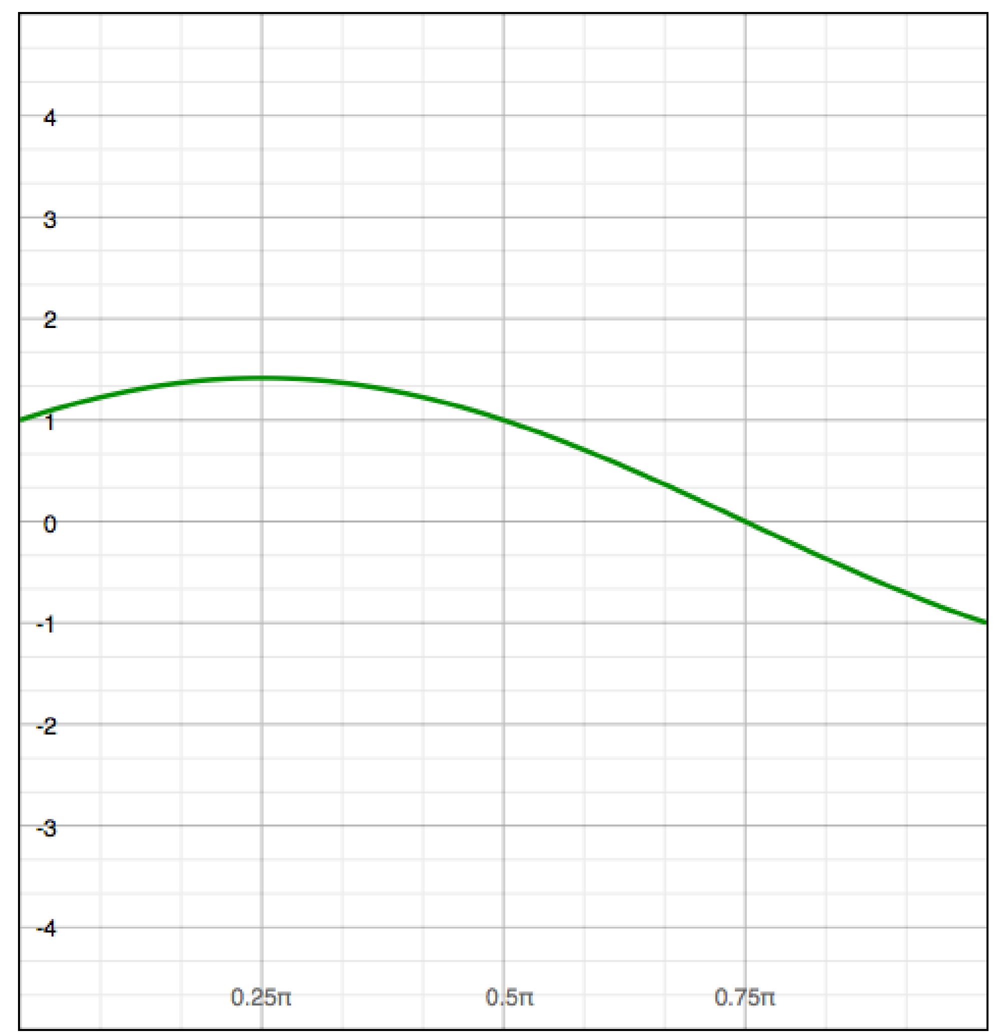



Image space

a line becomes?




Parameter space

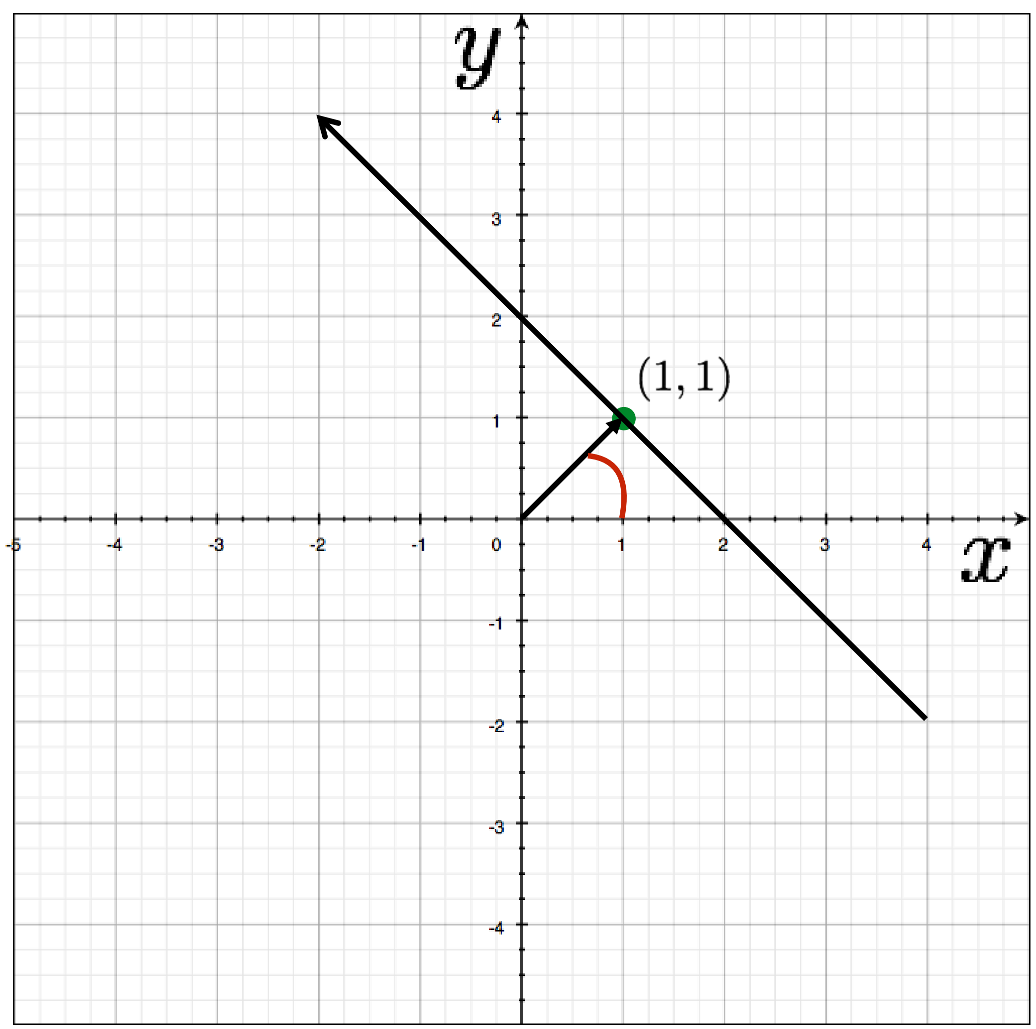
# Image and parameter space

variables

$$y = mx + b$$

parameters

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




a line becomes a point

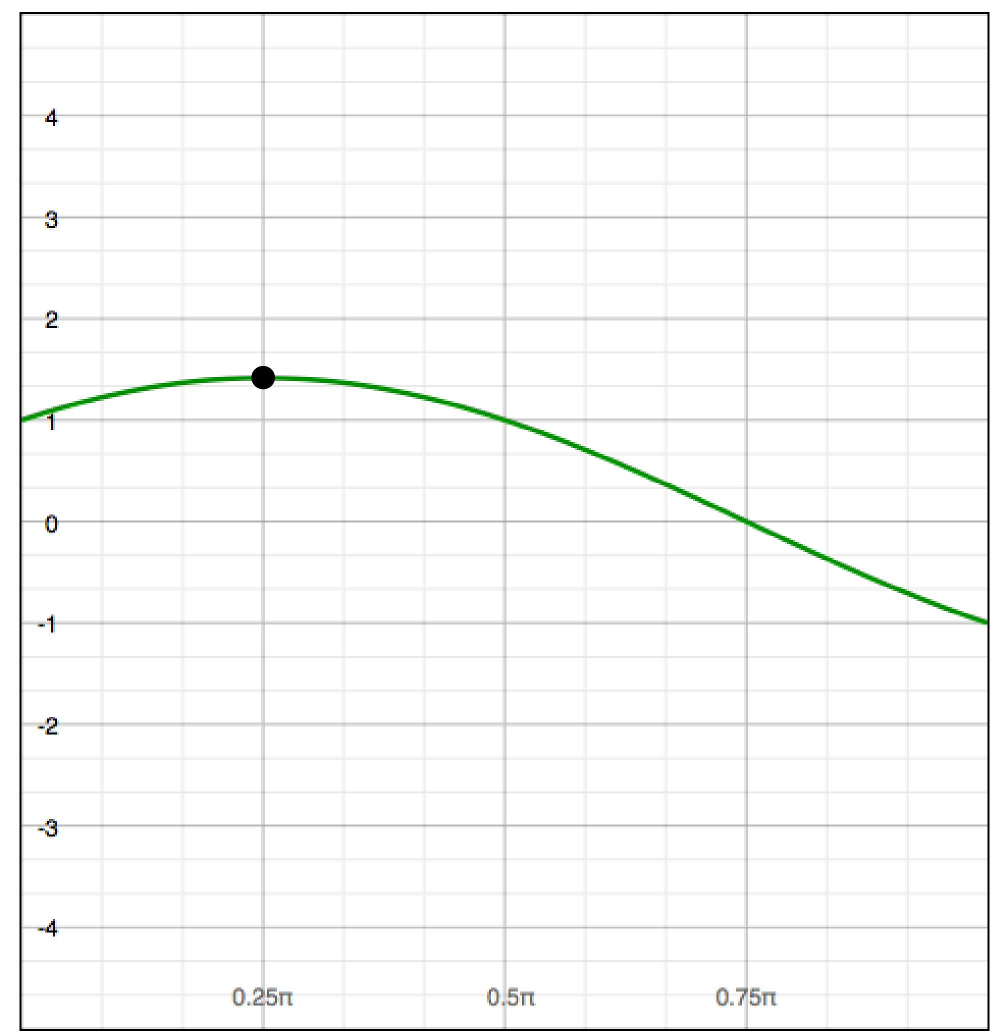



Image space

Parameter space

# Image and parameter space

variables

$$y = mx + b$$

parameters

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

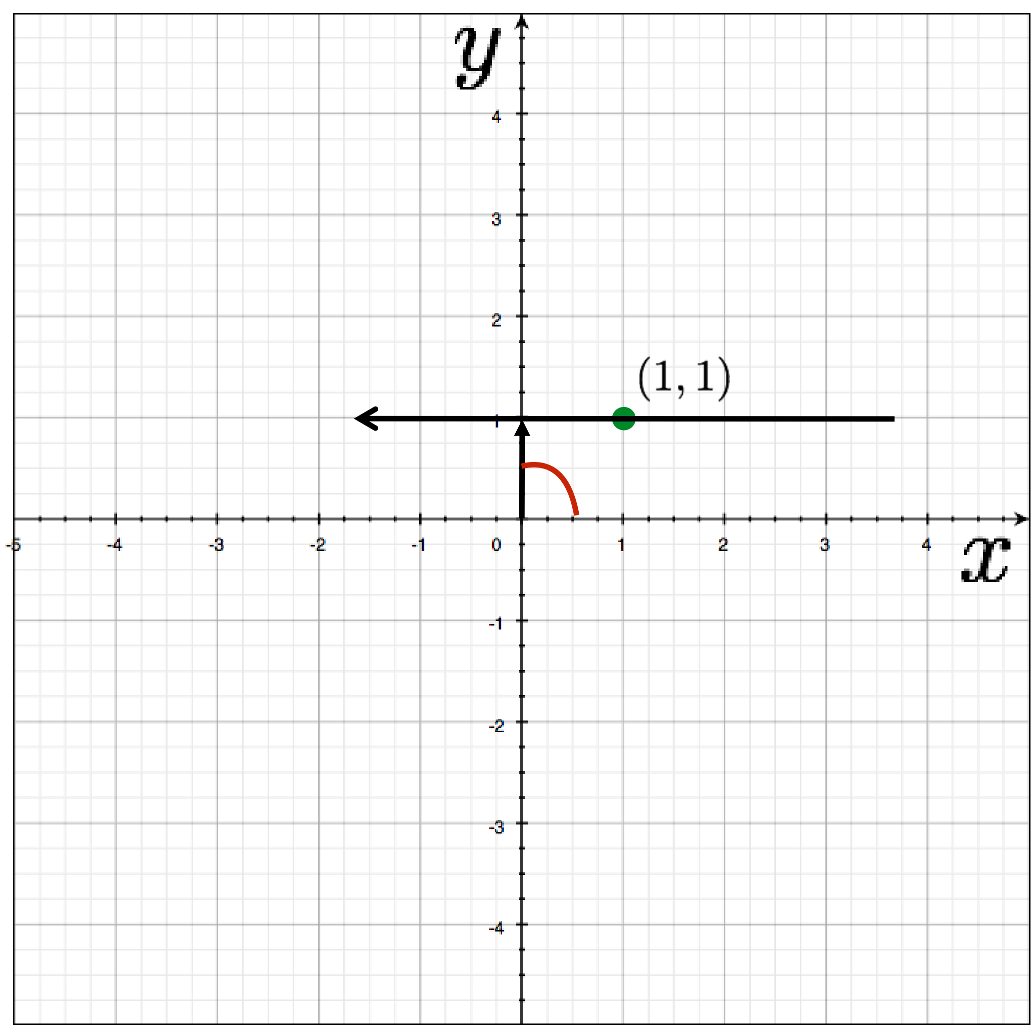
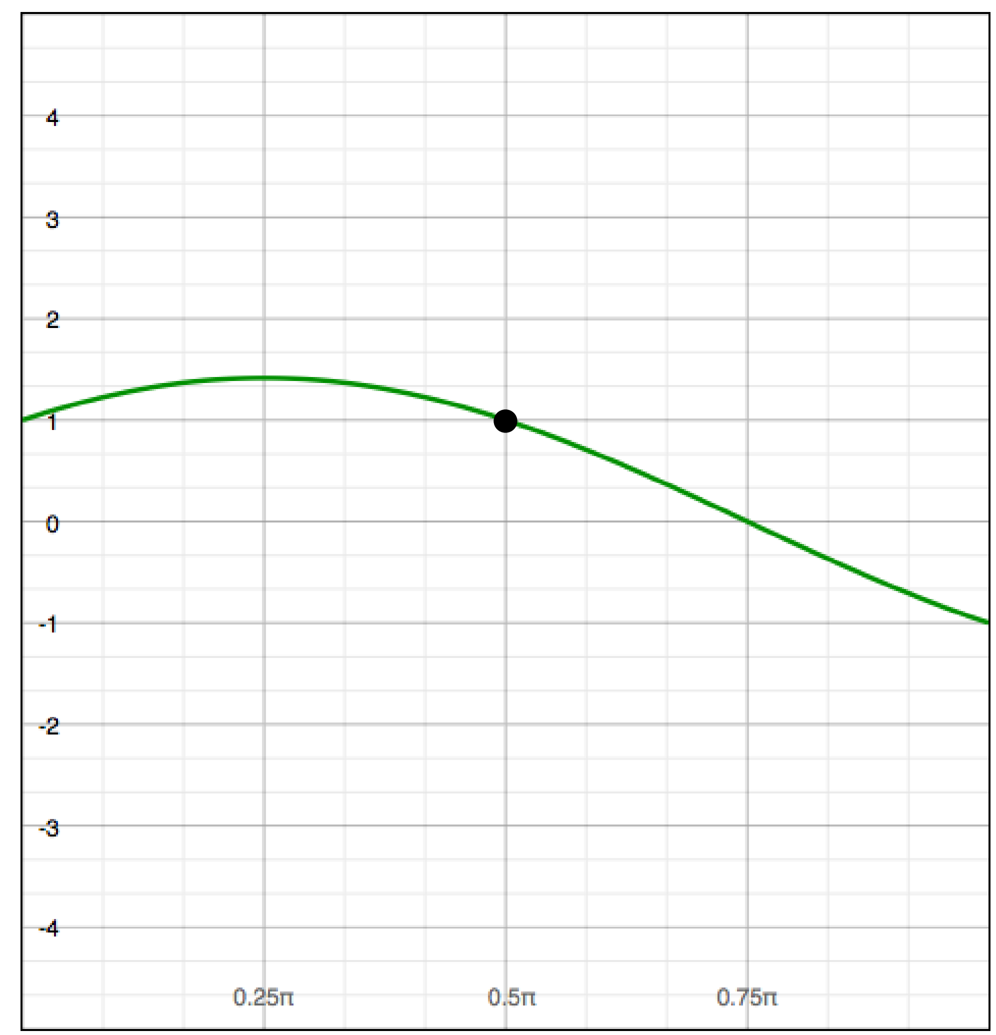



Image space

a line becomes a point




Parameter space

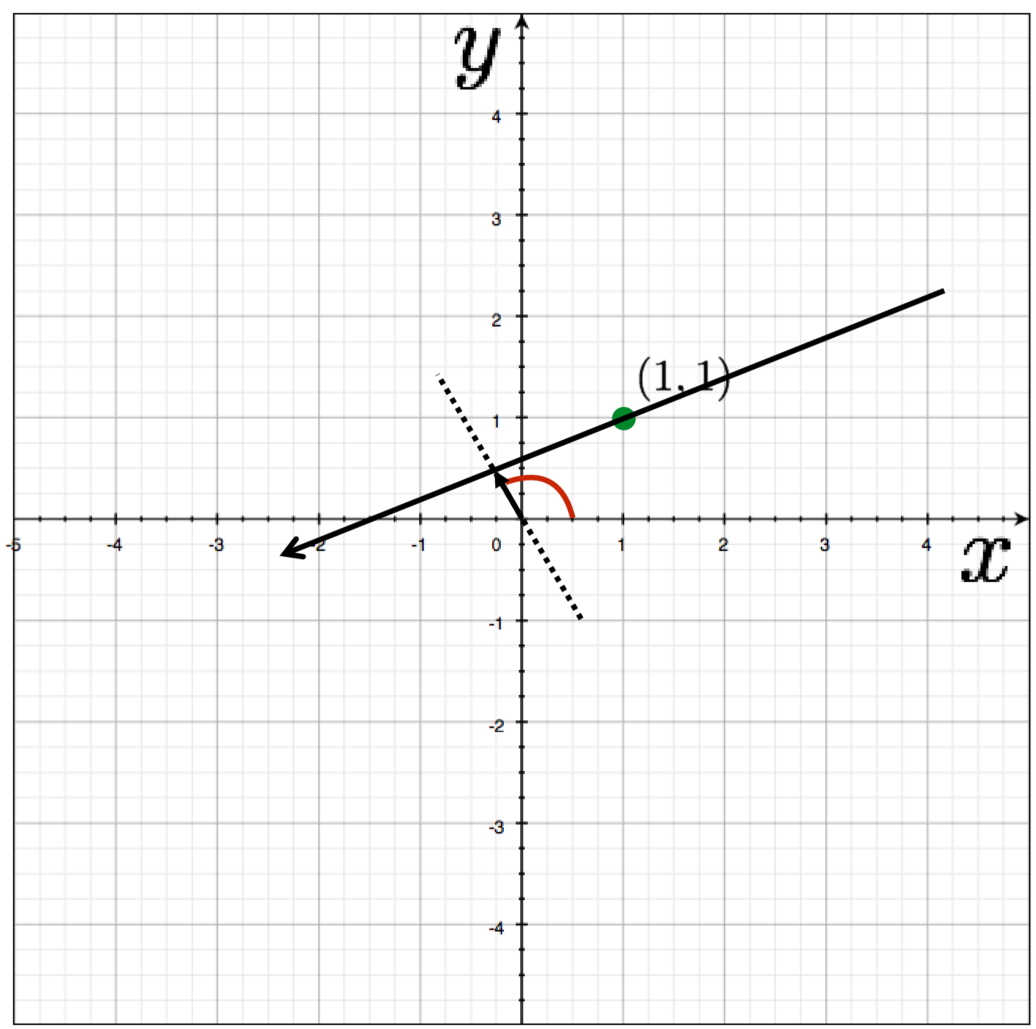
# Image and parameter space

variables

$$y = mx + b$$

parameters

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




a line becomes a point

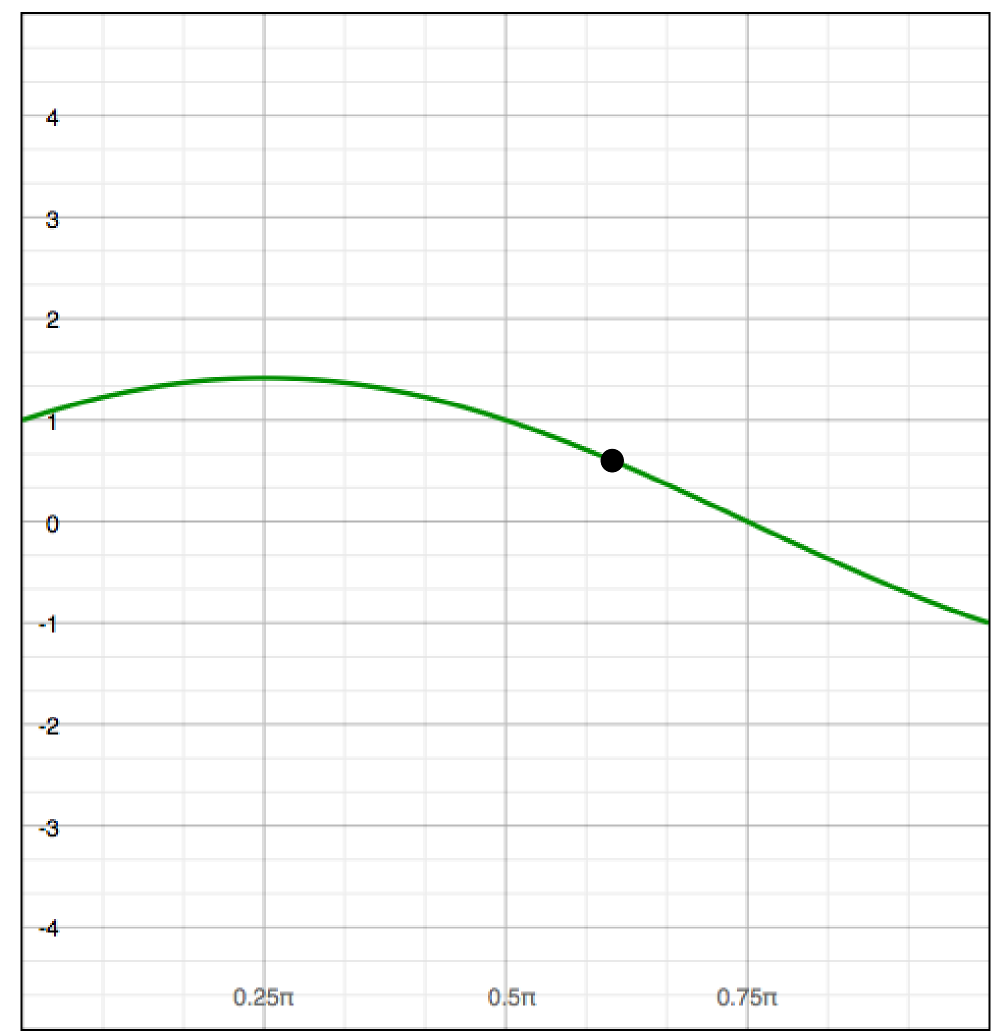



Image space

Parameter space

# Image and parameter space

variables

$$y = mx + b$$

parameters

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

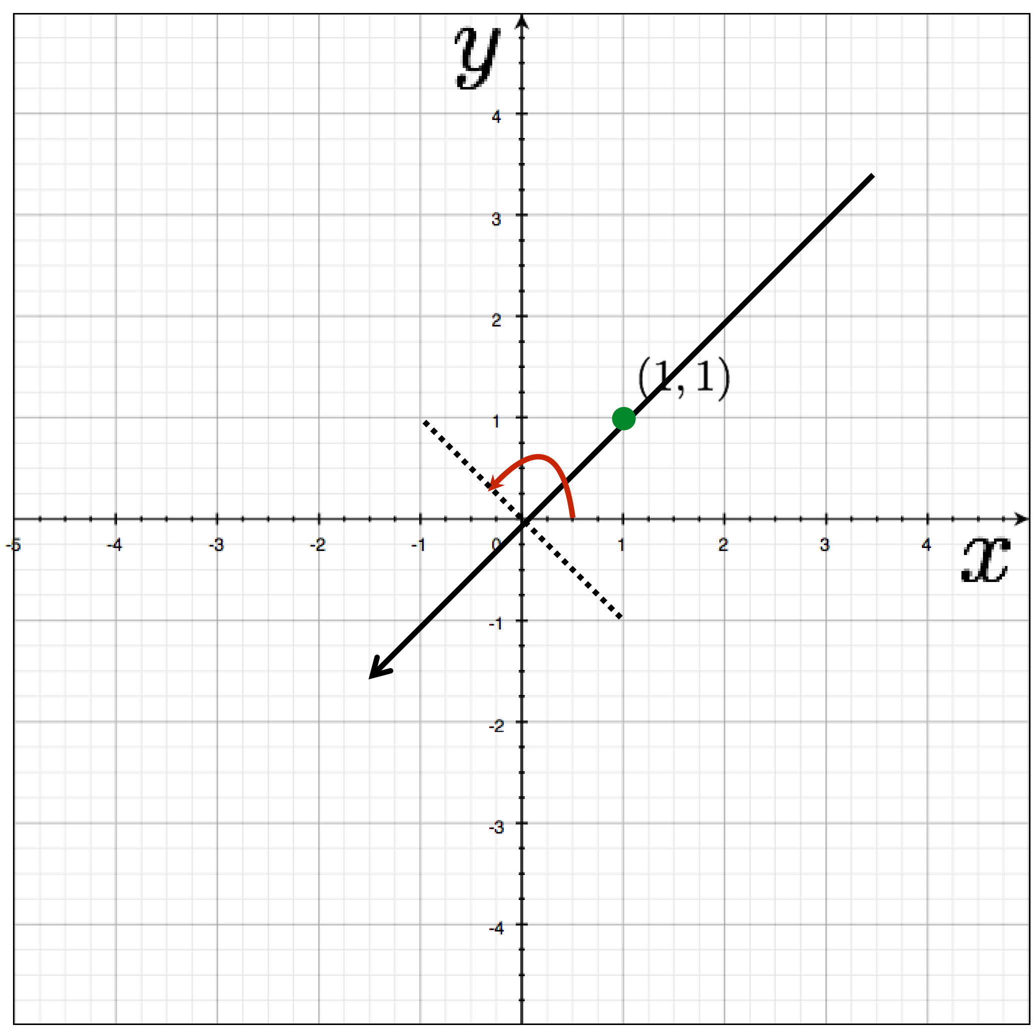
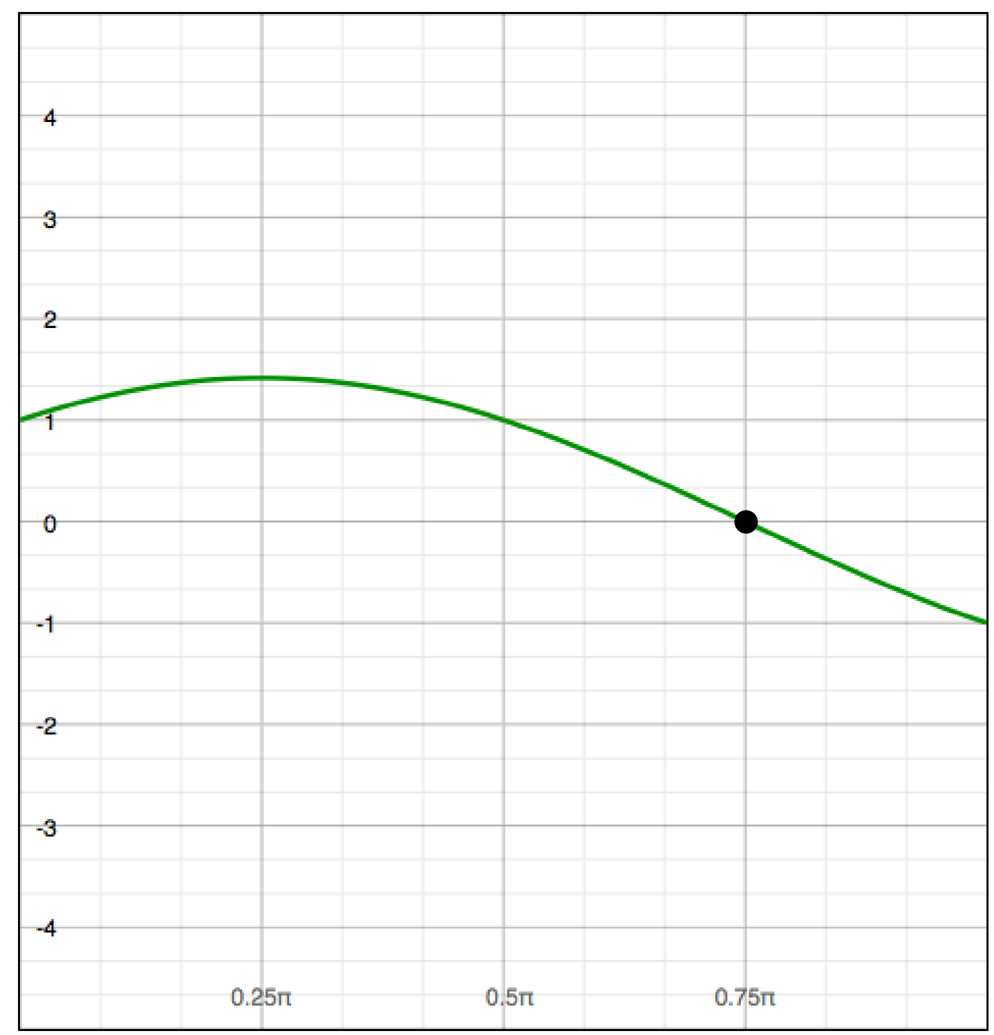



Image space

a line becomes a point




Parameter space

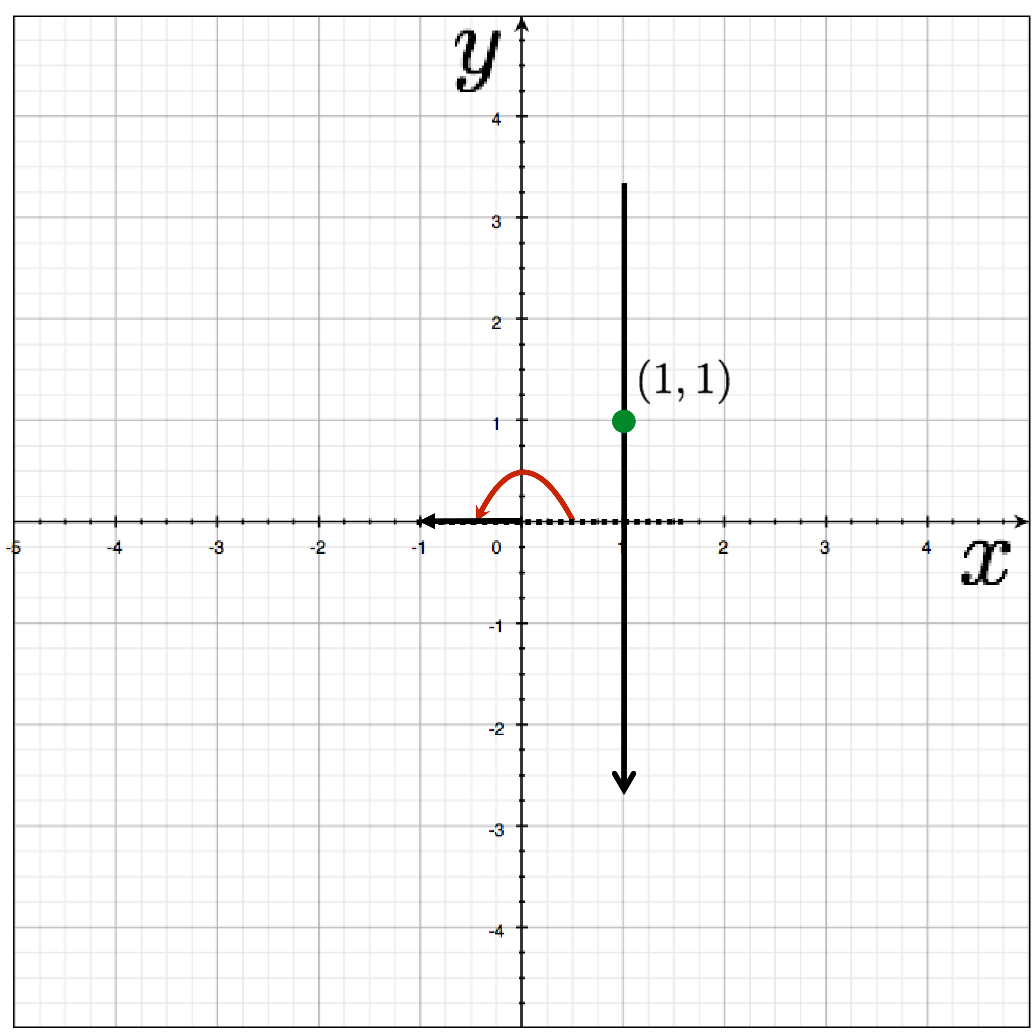
# Image and parameter space

variables

$$y = mx + b$$

parameters

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




a line becomes a point

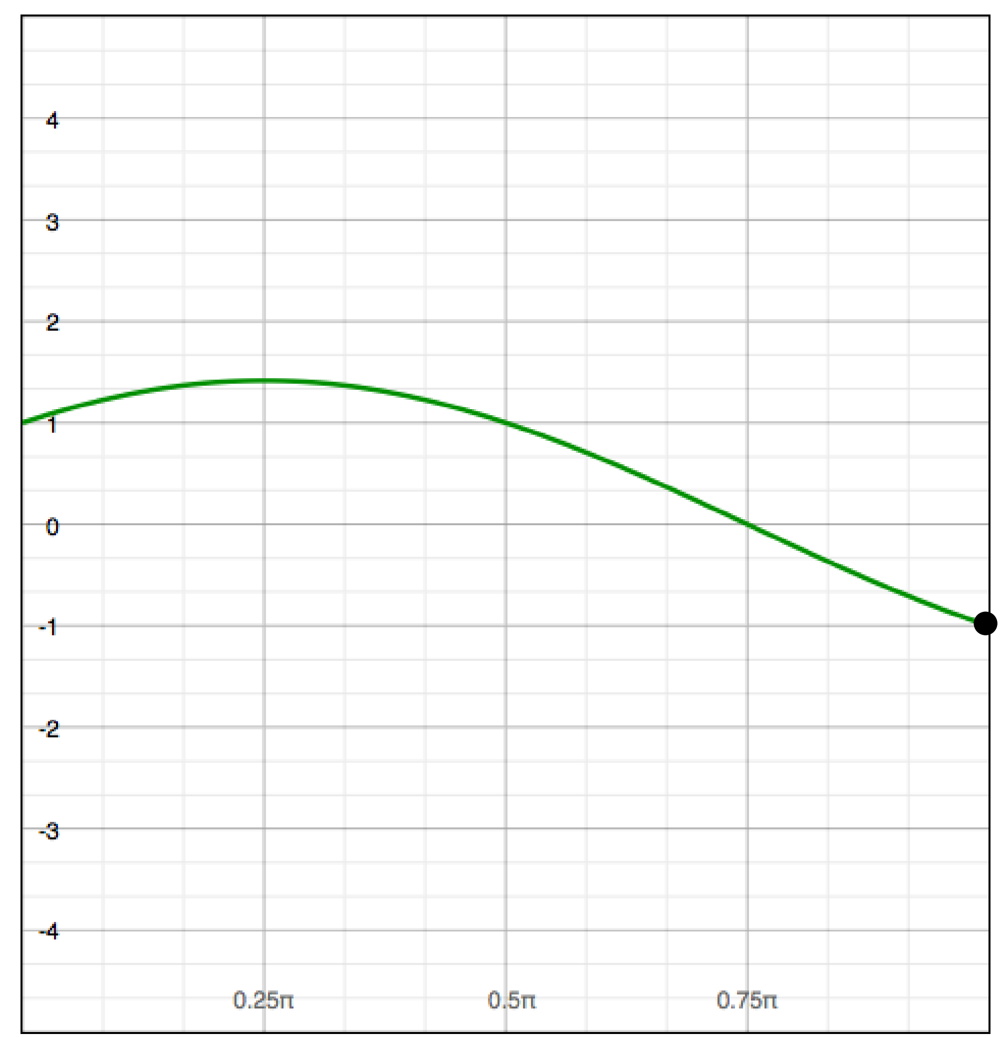



Image space

Parameter space

# Image and parameter space

variables

$$y = mx + b$$

parameters

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

Wait …why is rho negative?

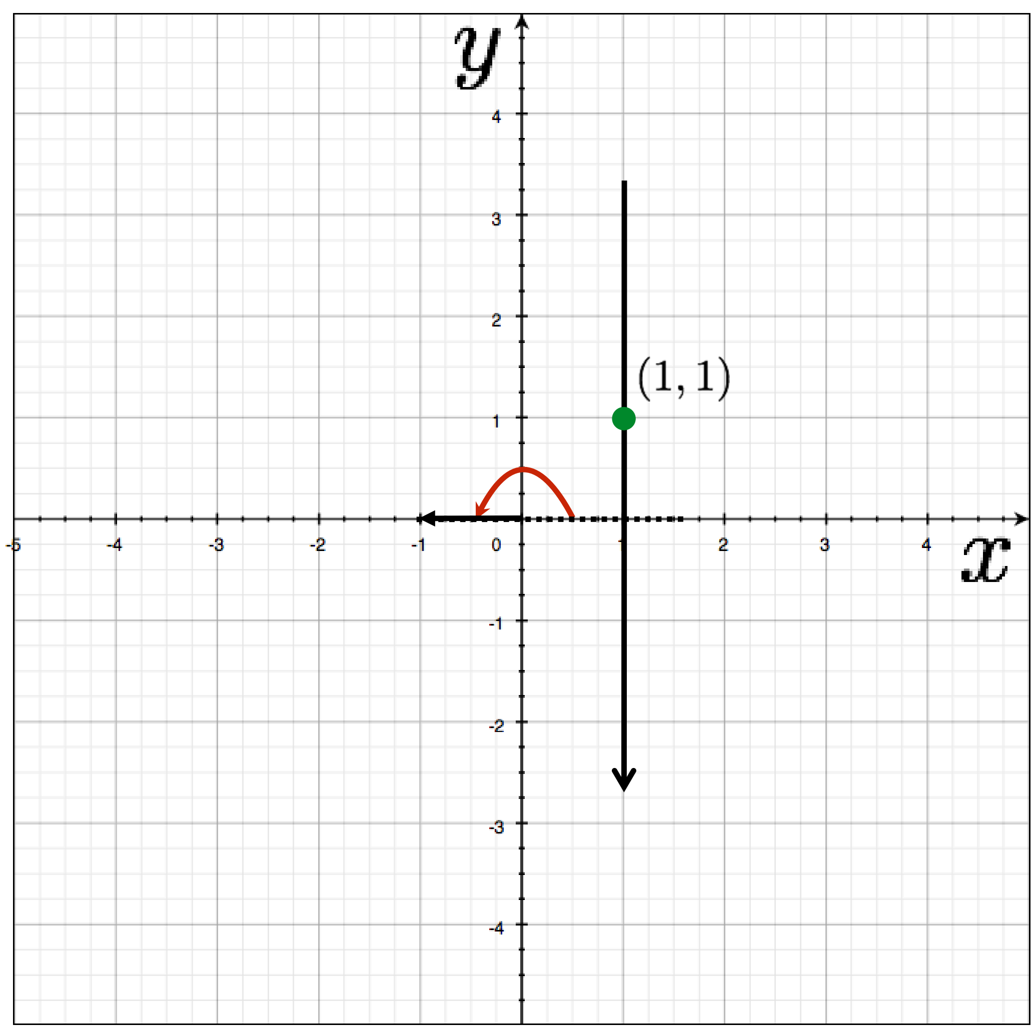
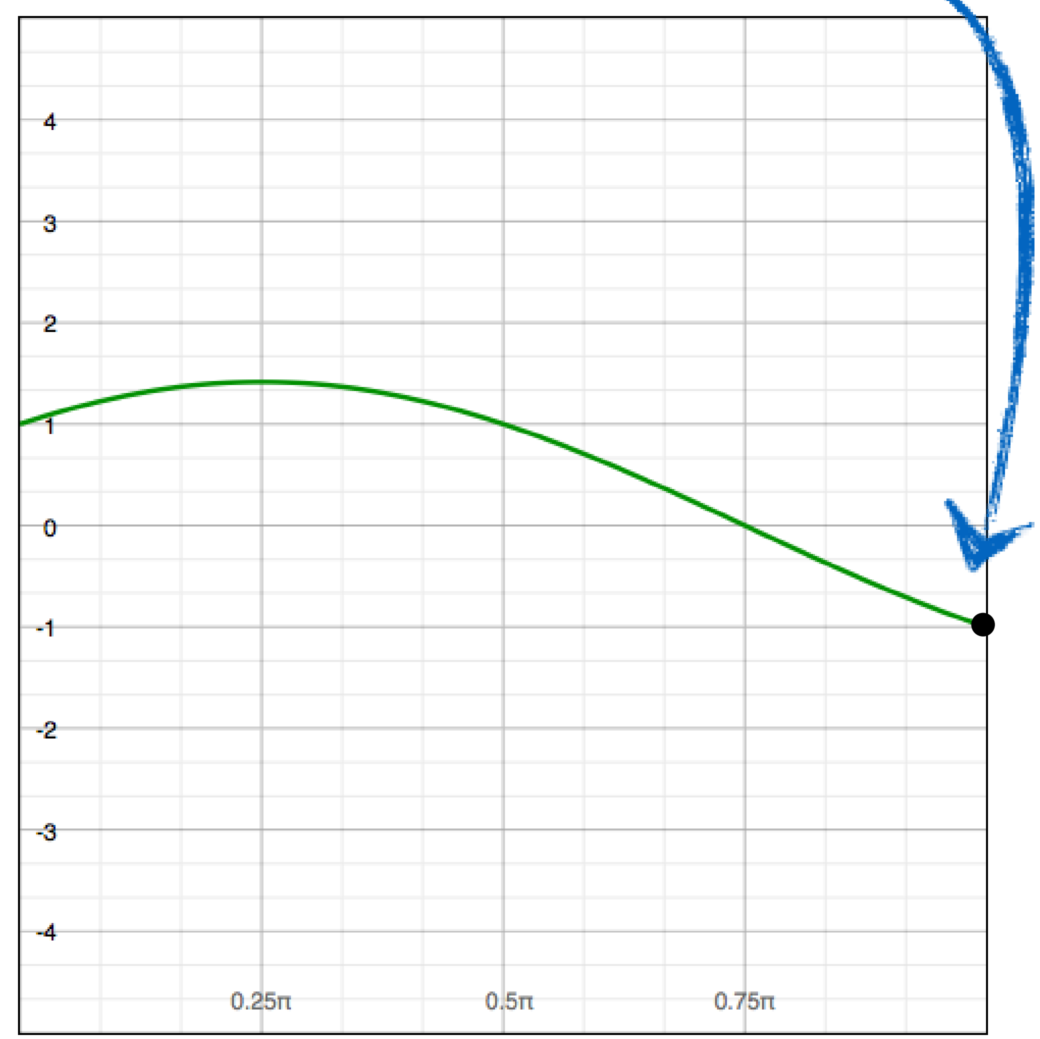



Image space

a line becomes a point




Parameter space

# Image and parameter space

variables

$$y = mx + b$$

parameters

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

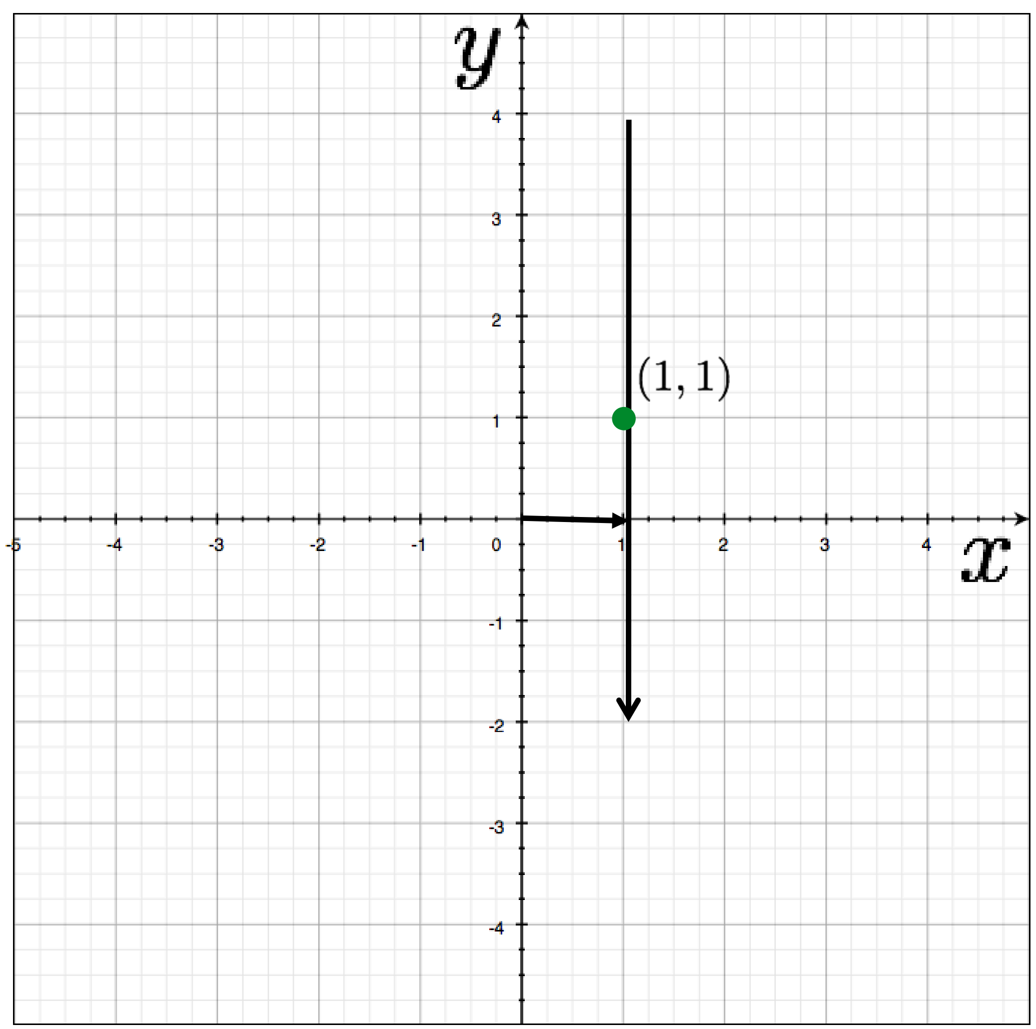
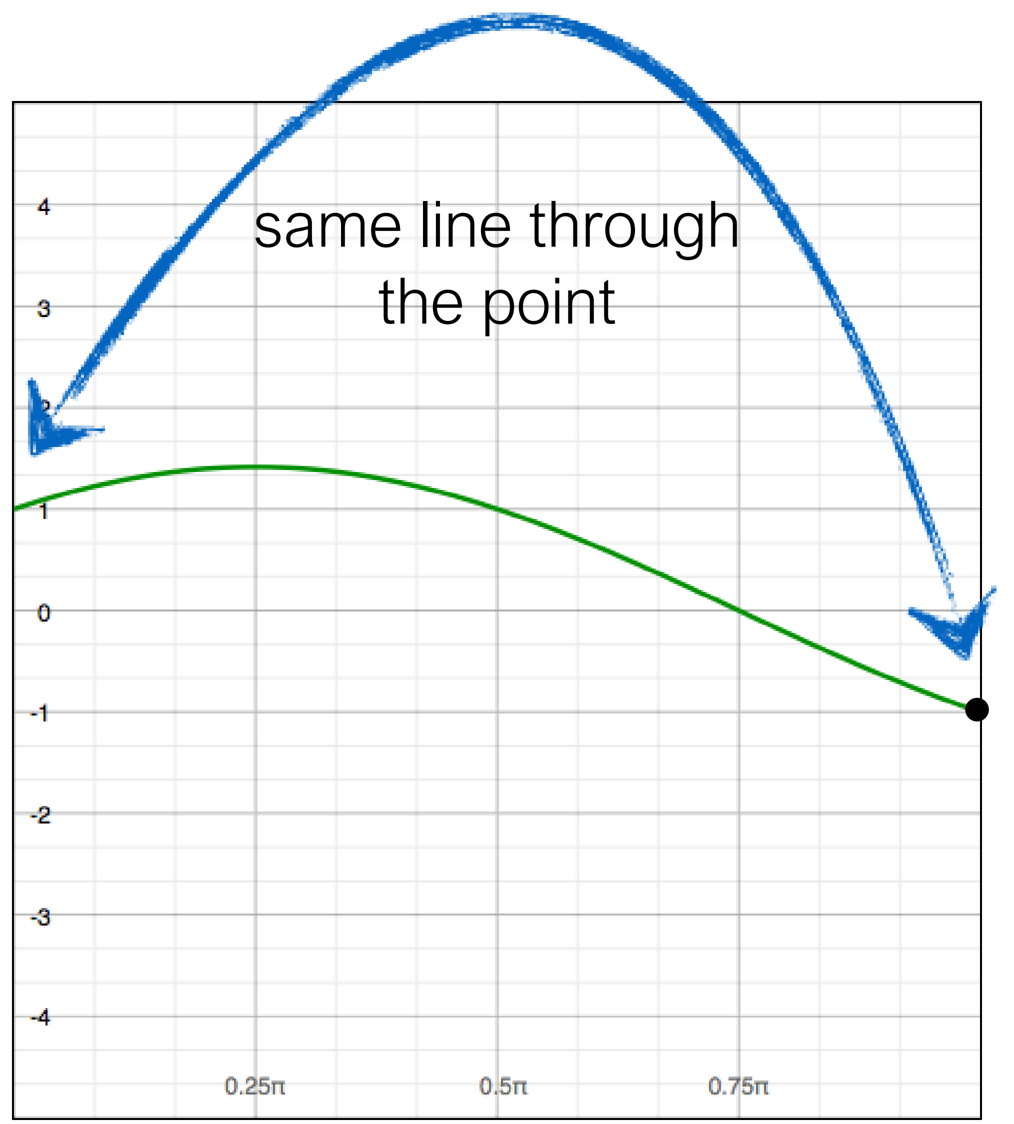



Image space

a line becomes a point

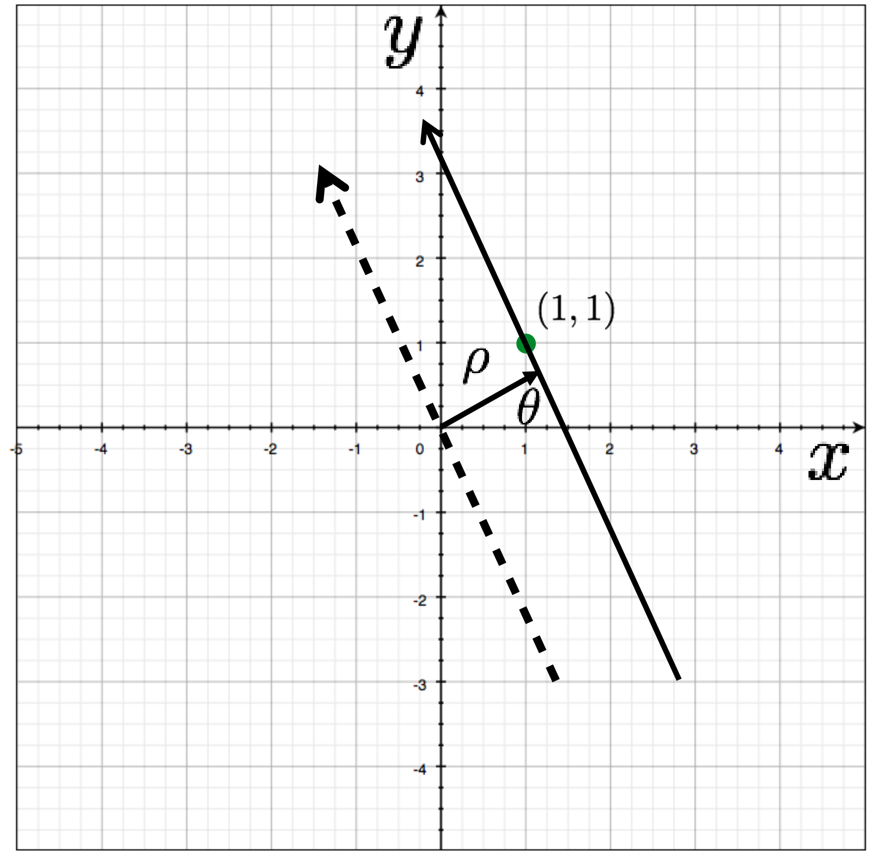



Parameter space

There are two ways to write the same line:

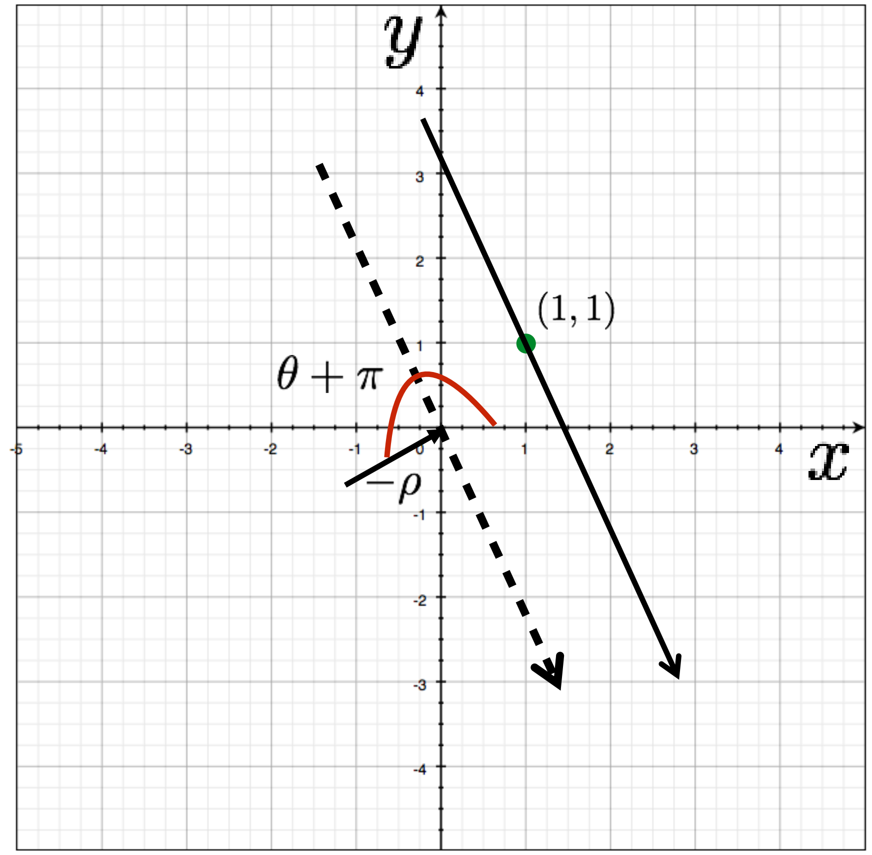
Positive rho version:

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




Negative rho version:

$$x \cos(\theta + \pi) + y \sin(\theta + \pi) = -\rho$$




Recall:

$$\sin(\theta) = -\sin(\theta + \pi)$$

$$\cos(\theta) = -\cos(\theta + \pi)$$

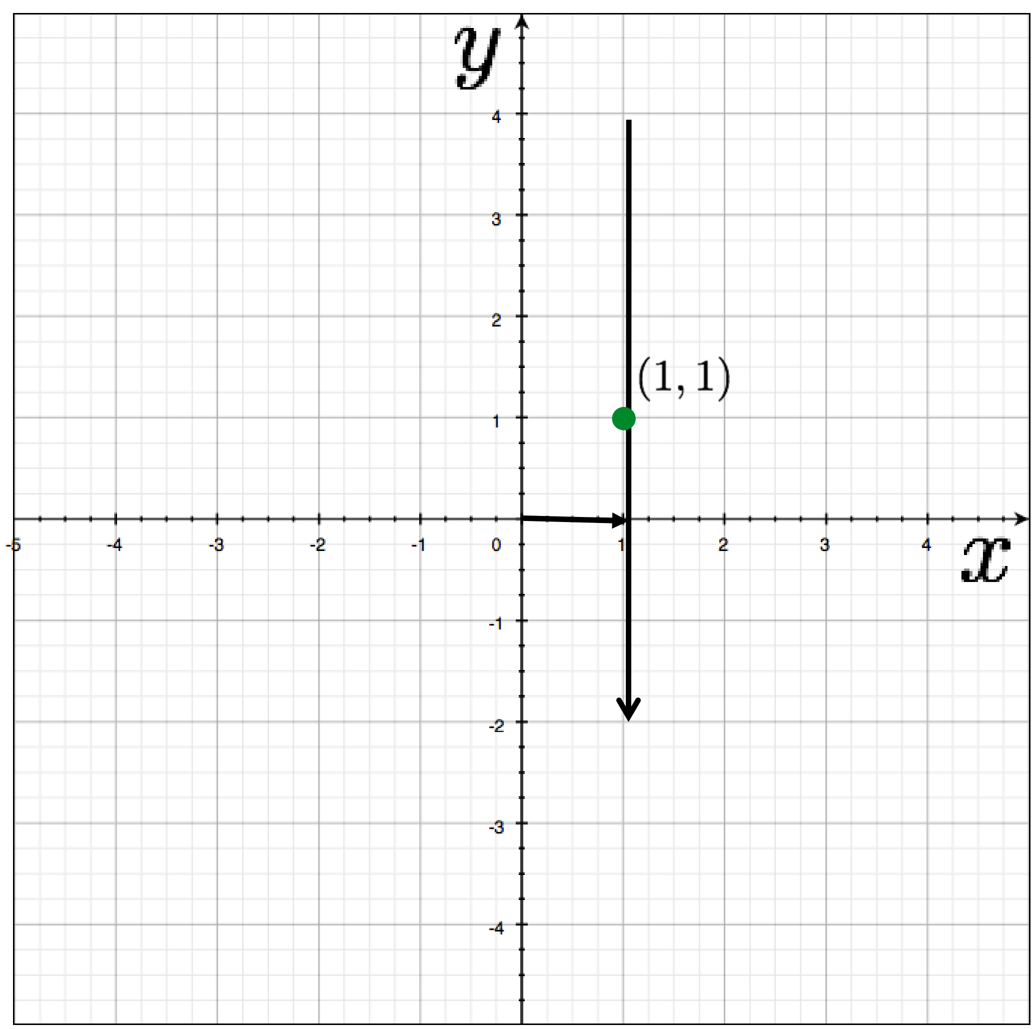
# Image and parameter space

variables

$$y = mx + b$$

parameters

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




a line becomes a point

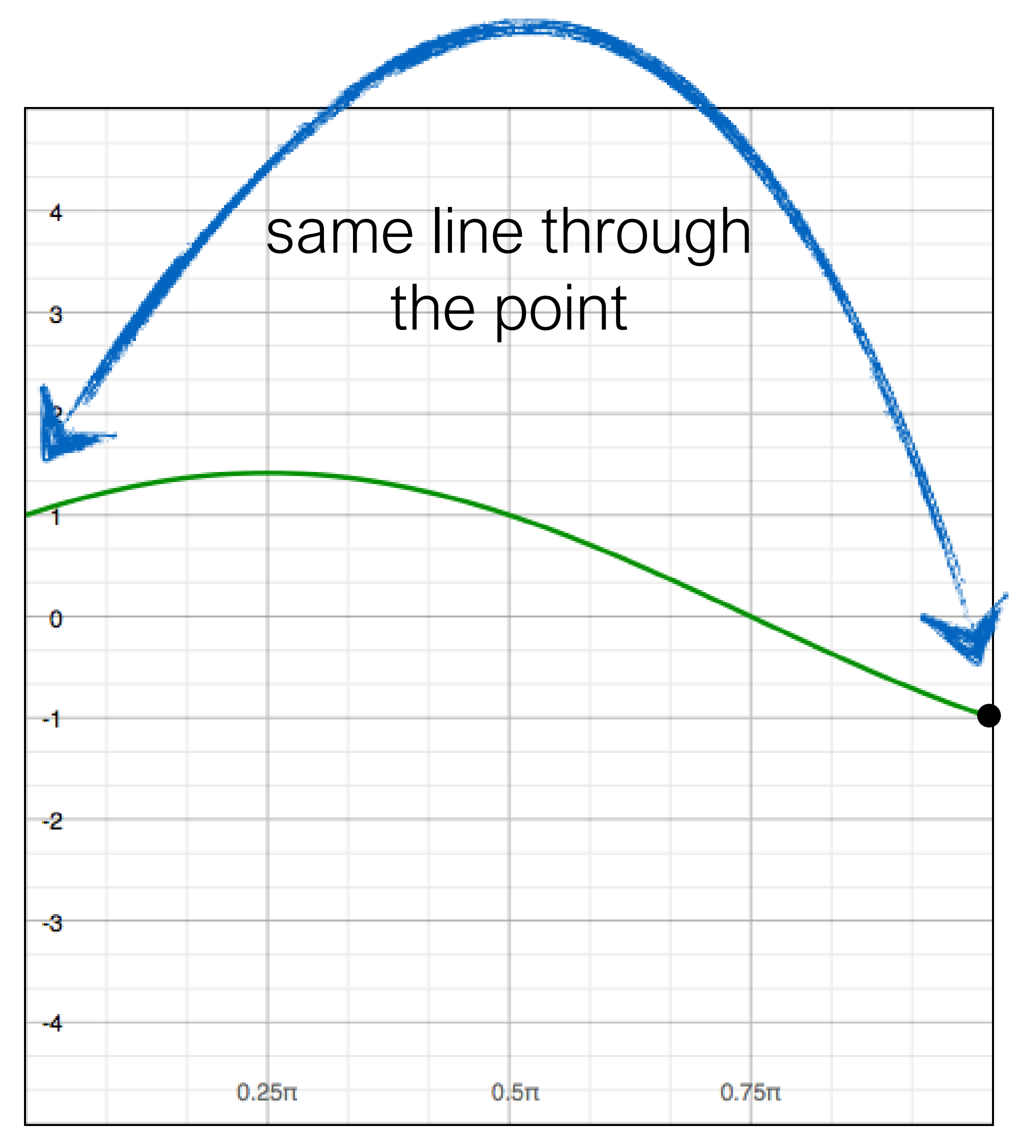



Image space

Parameter space

# Image and parameter space

variables

$$y = mx + b$$

parameters

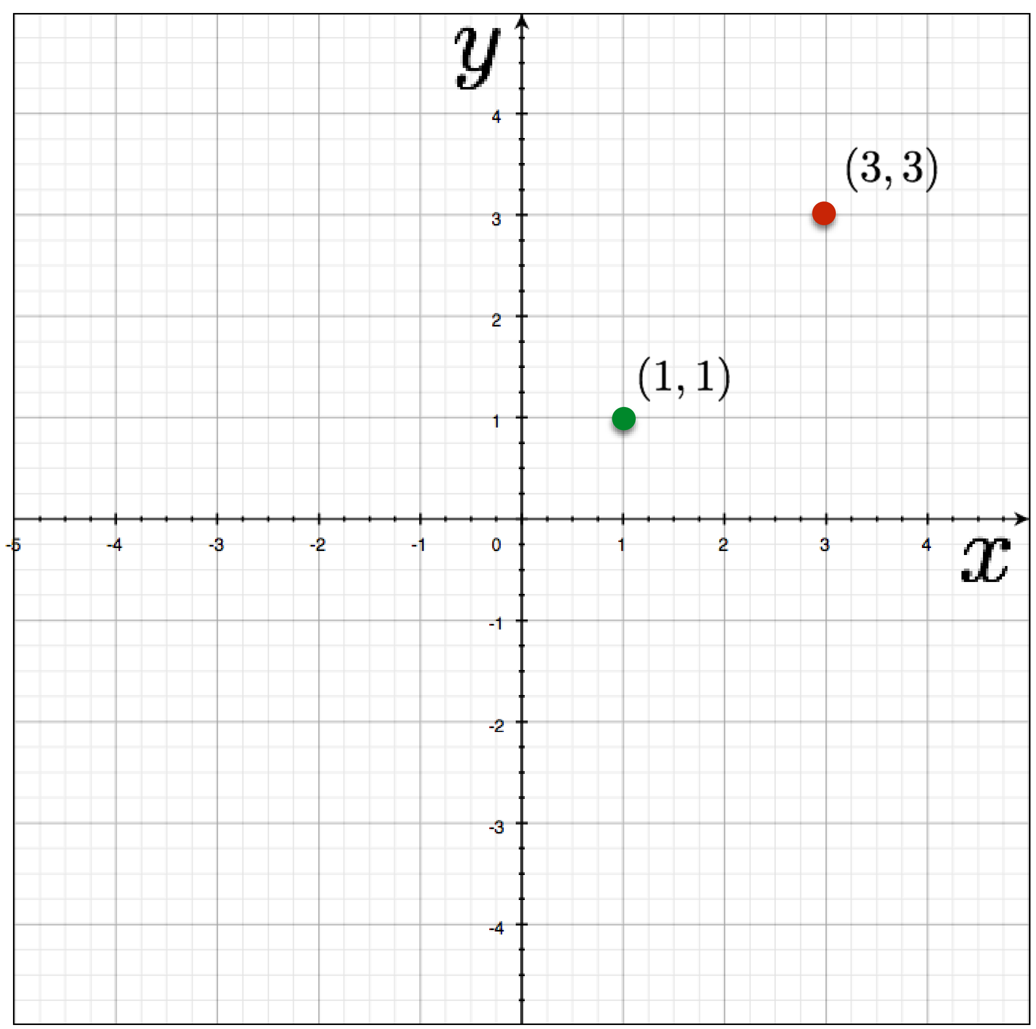
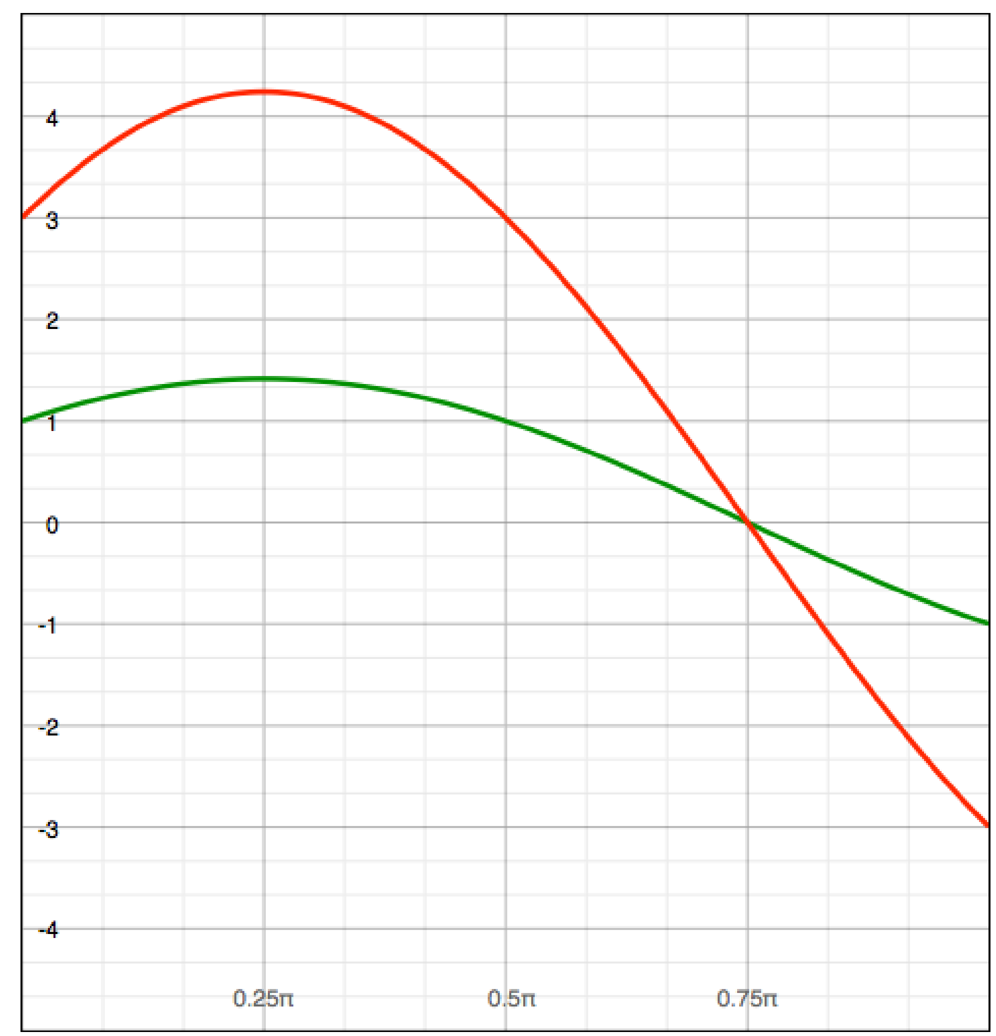



Image space

two points become ?




Parameter space

# Image and parameter space

variables

$$y = mx + b$$

parameters

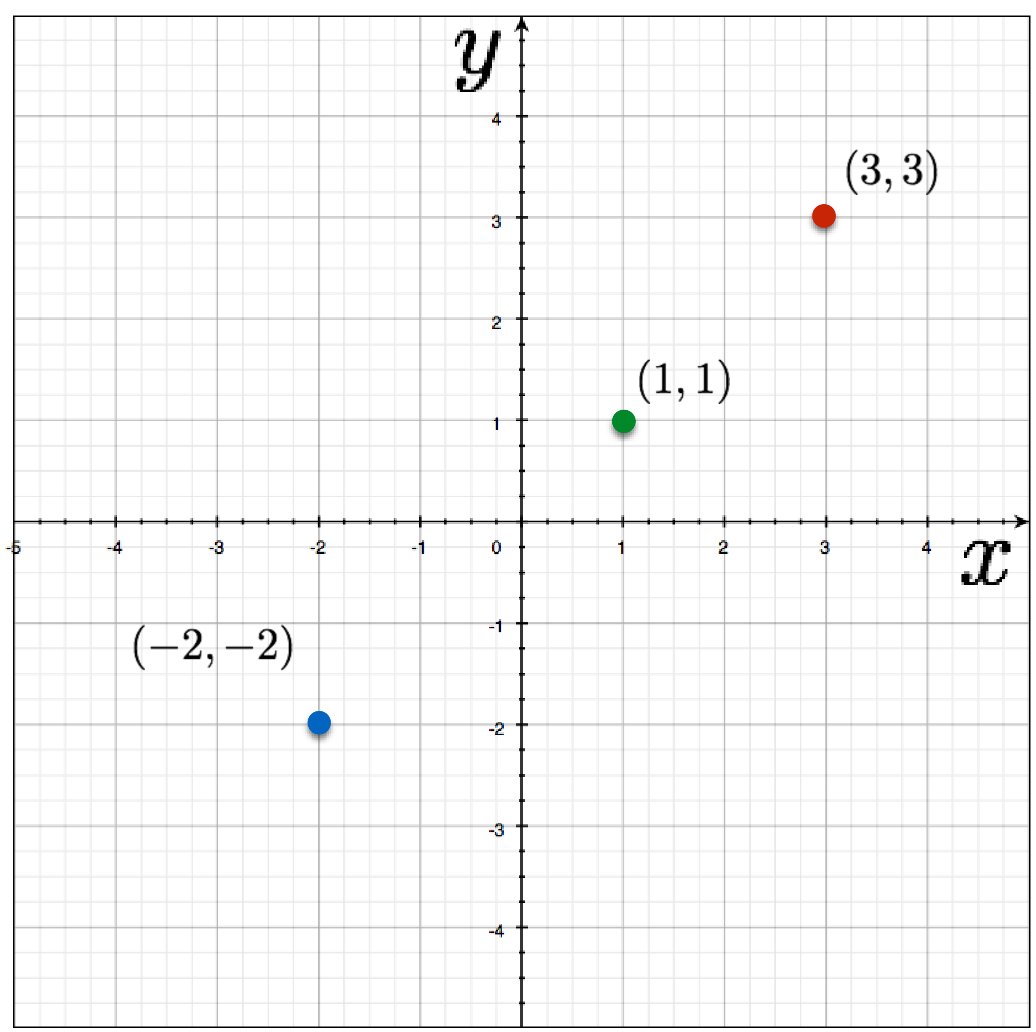
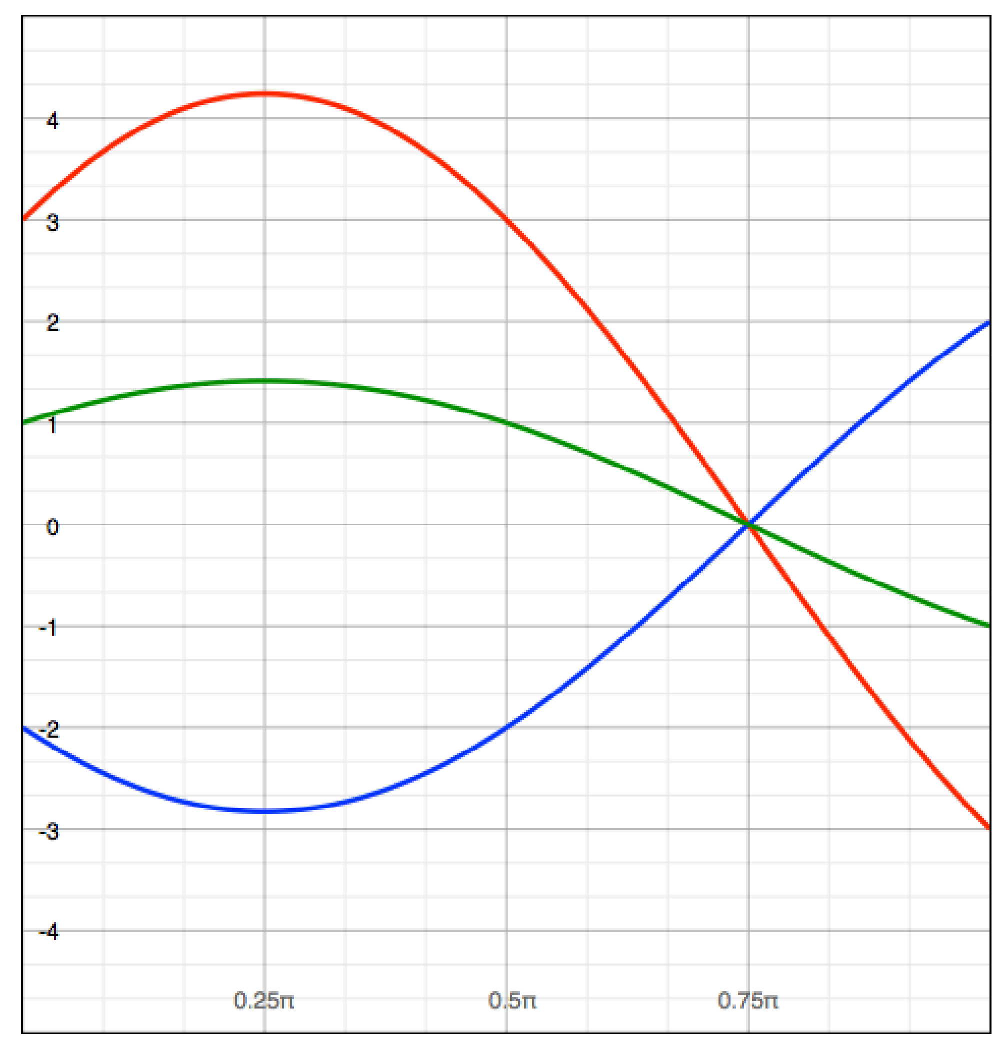



Image space

three points
become
?




Parameter space

# Image and parameter space

variables

$$y = mx + b$$

parameters

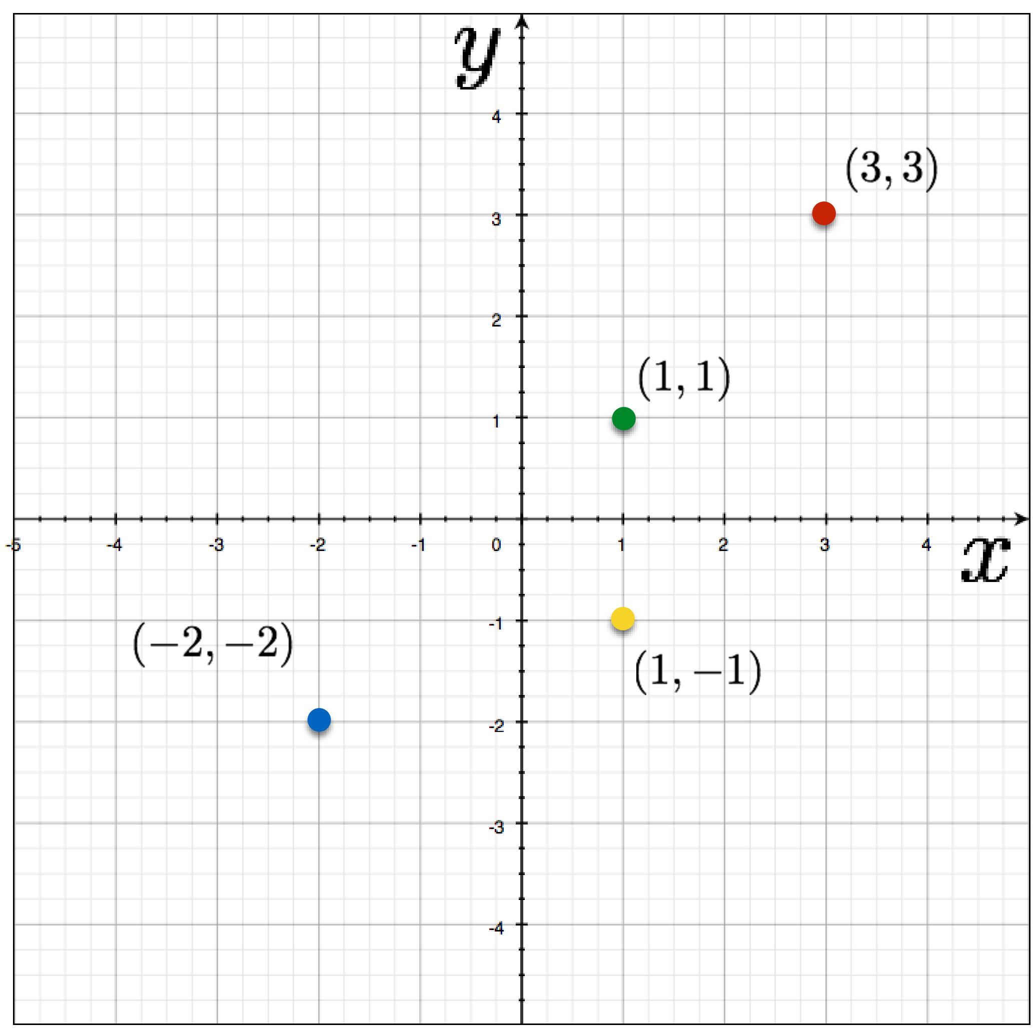
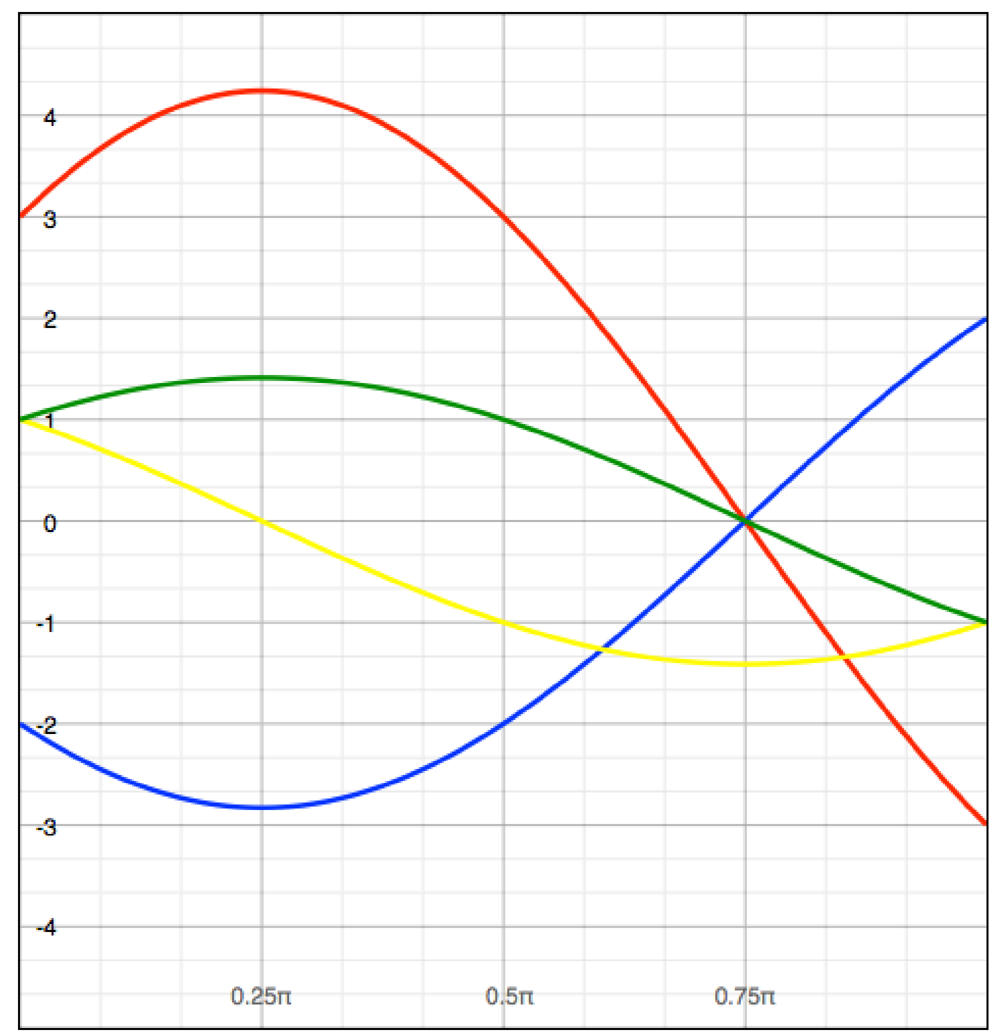



Image space

four points become ?




Parameter space

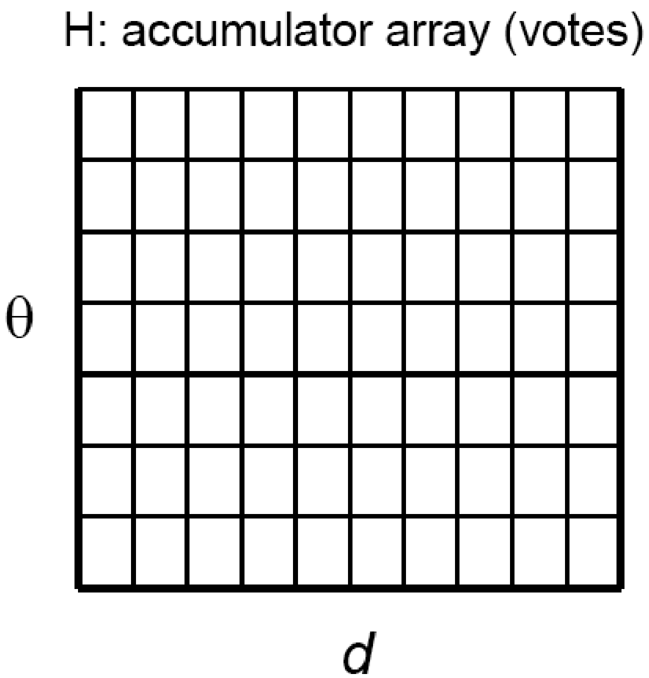
# Implementation

```
1. Initialize accumulator H to all zeros

2. For each edge point (x,y) in the image
       For θ = 0 to 180
          ρ = x cos θ + y sin θ
          H(θ, ρ) = H(θ, ρ) + 1
       end
   end

3. Find the value(s) of (θ, ρ) where H(θ, ρ)
   is a local maximum

4. The detected line in the image is given by
    ρ = x cos θ + y sin θ
```




NOTE: Watch your coordinates. Image origin is top left!

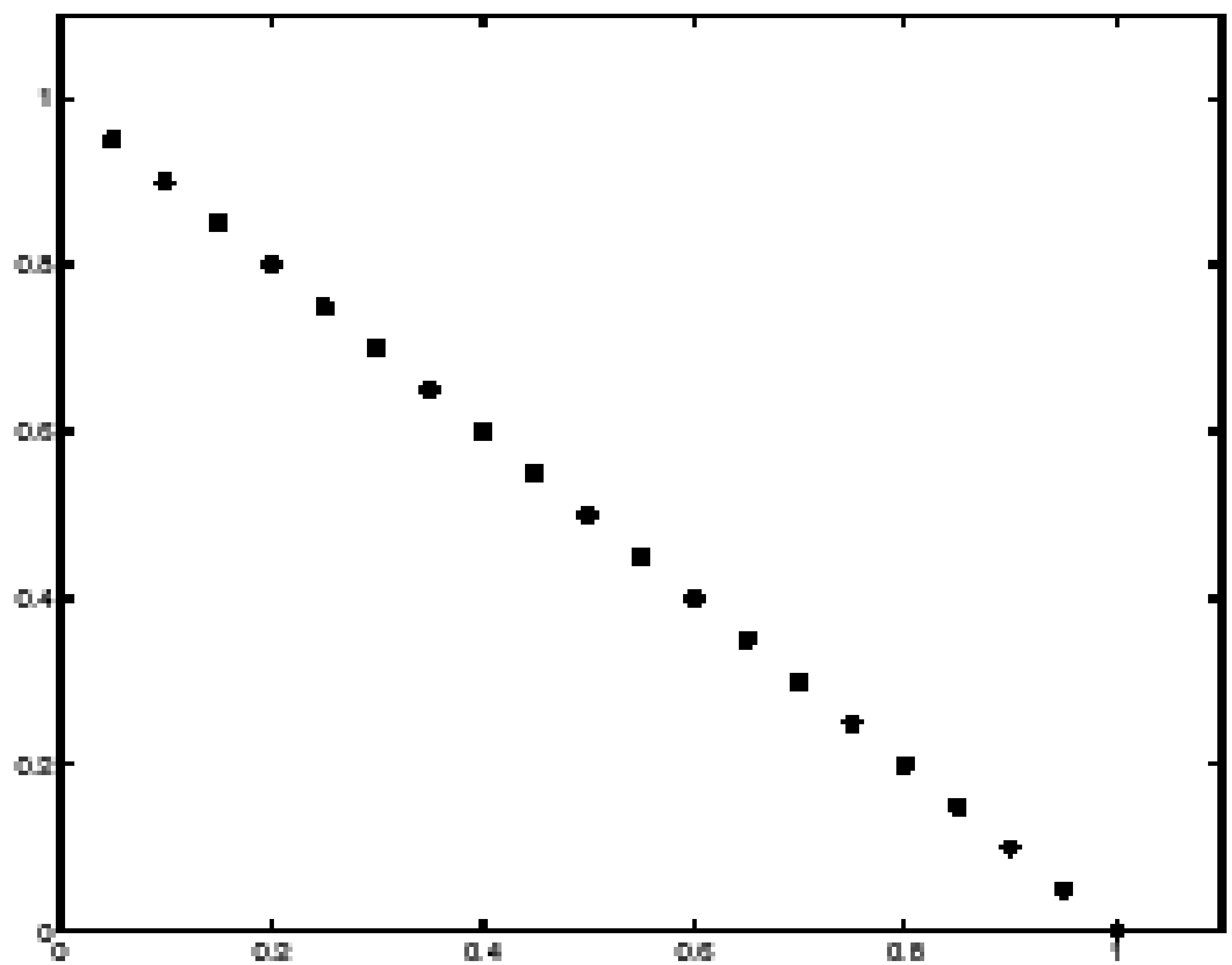
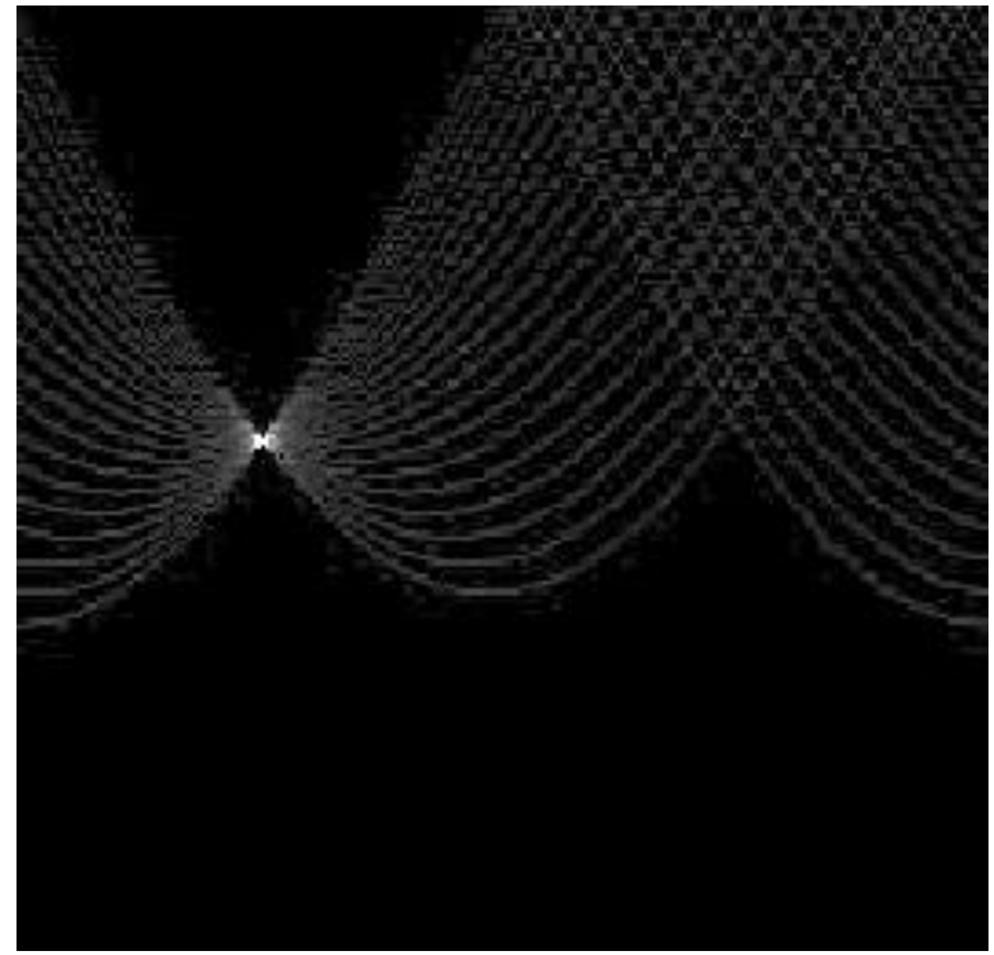


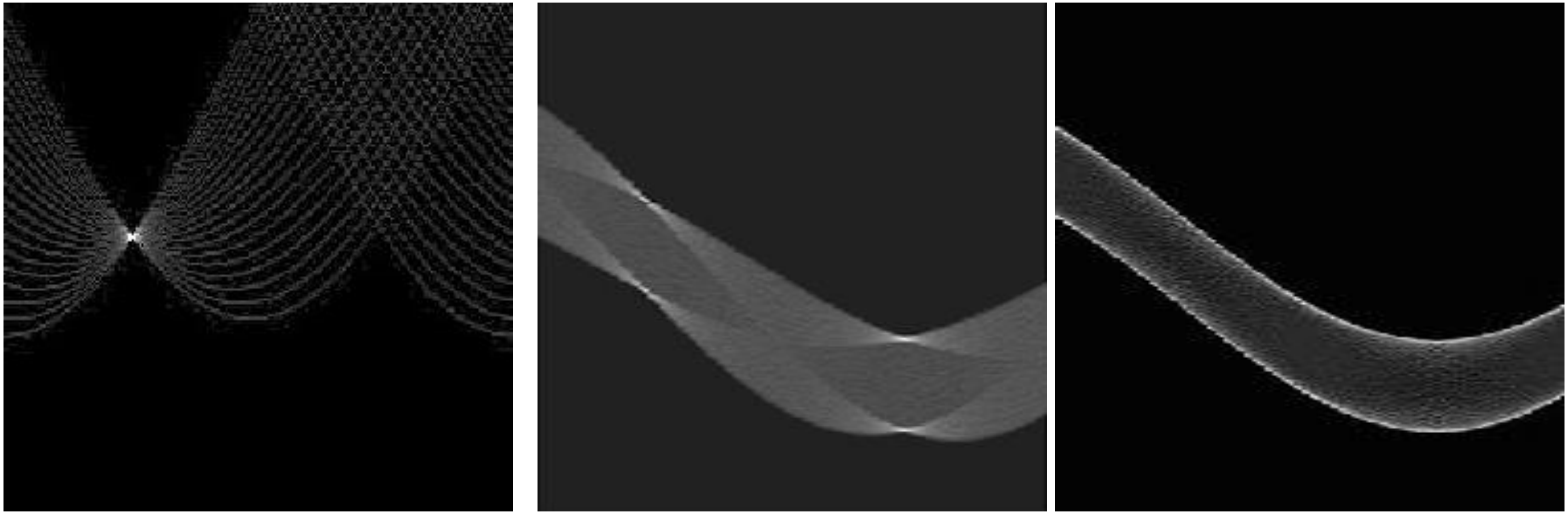

Image space



Votes

# Basic shapes

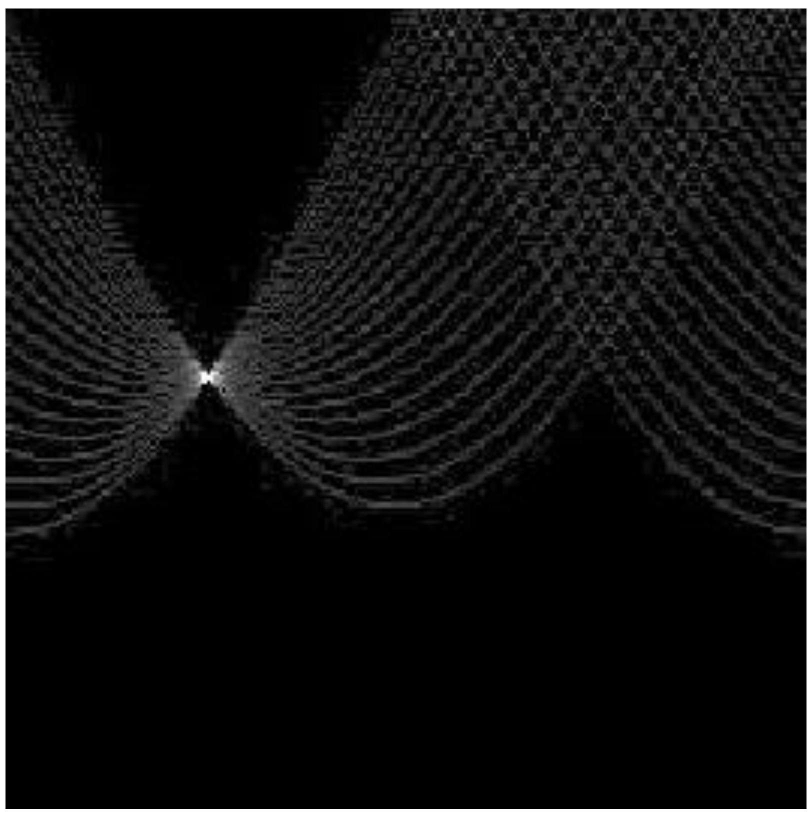
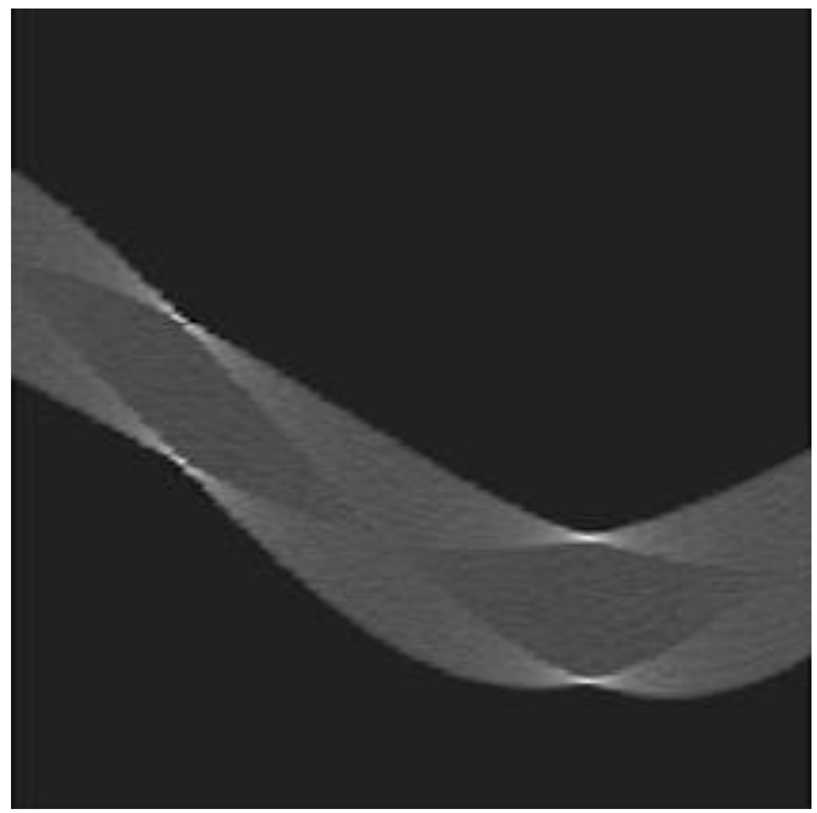
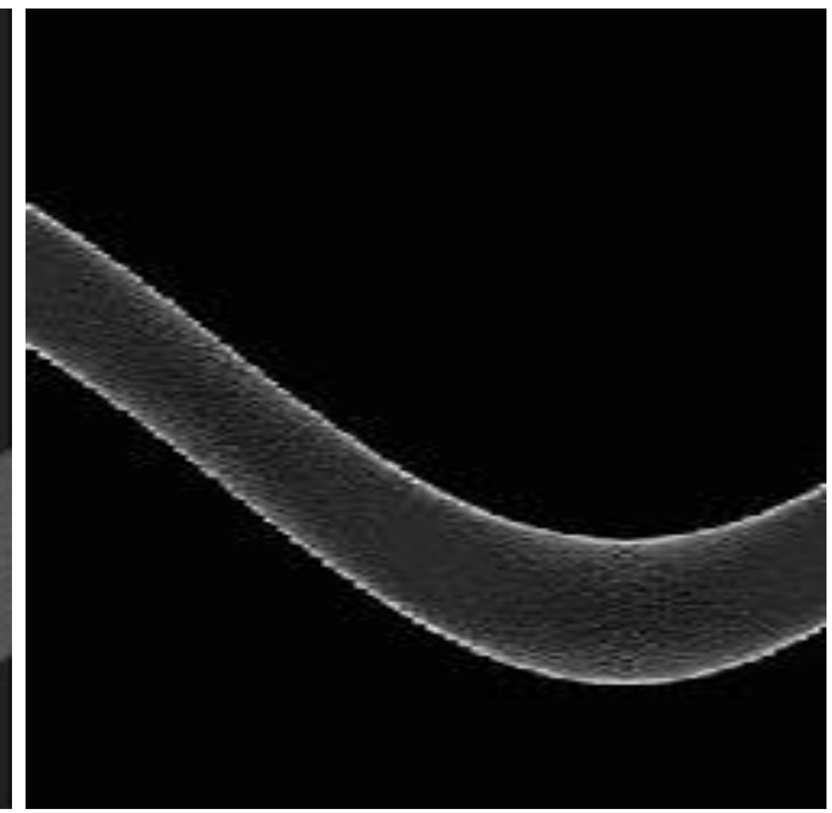
(in parameter space)



*can you guess the shape?*

# Basic shapes

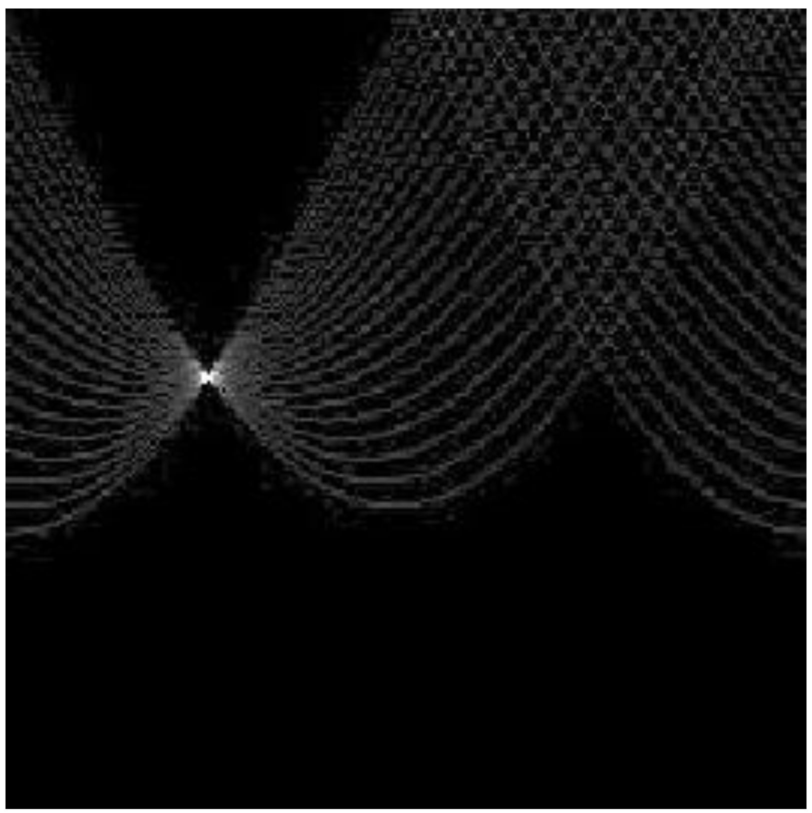
(in parameter space)

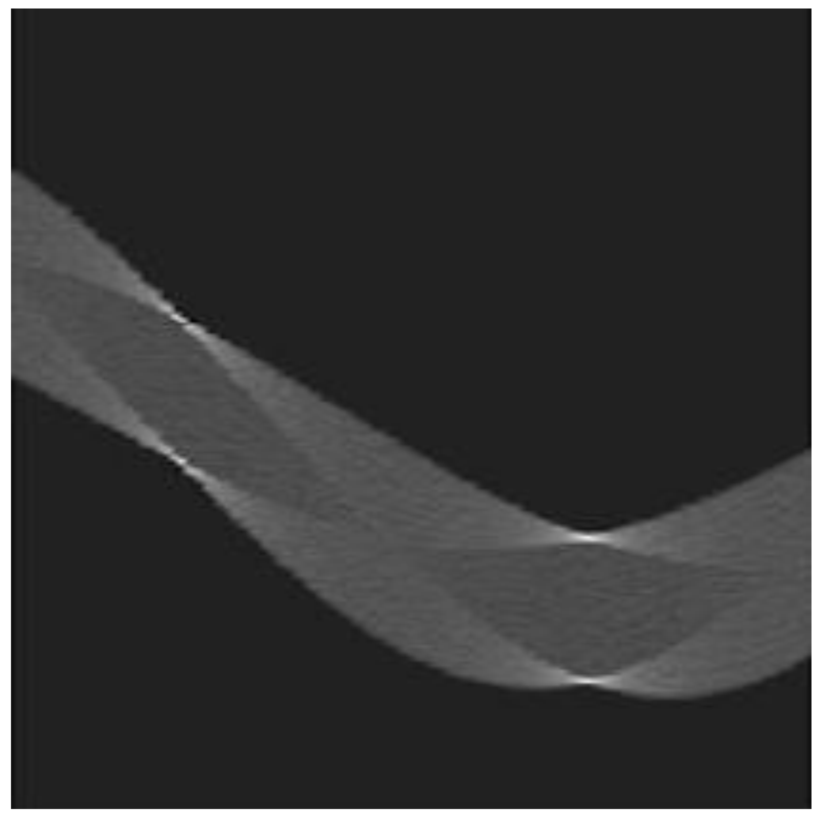
line

# Basic shapes

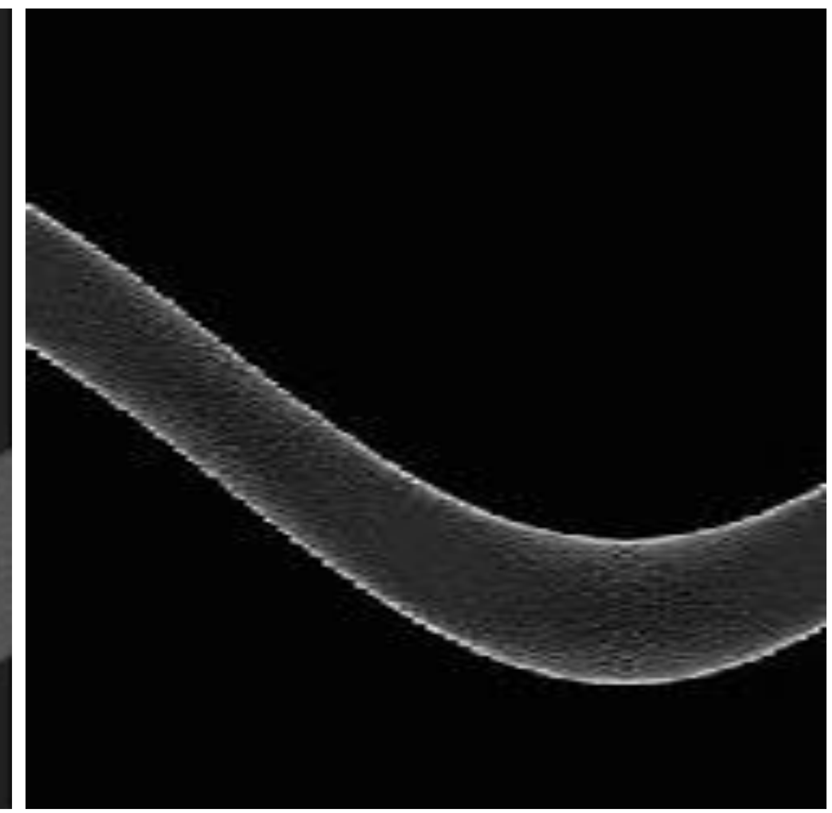
(in parameter space)



line

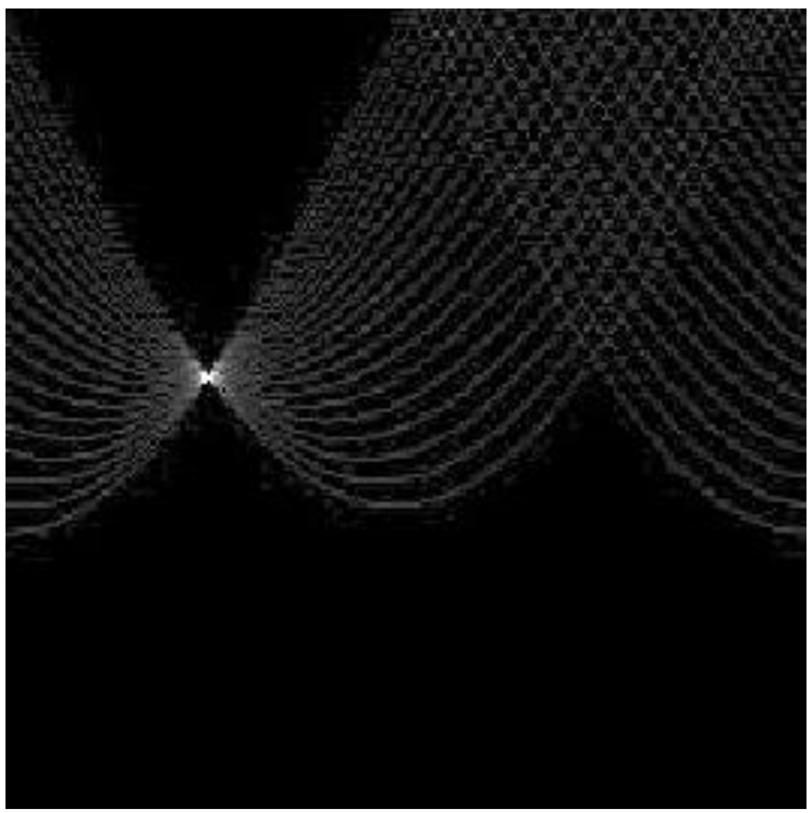


rectangle

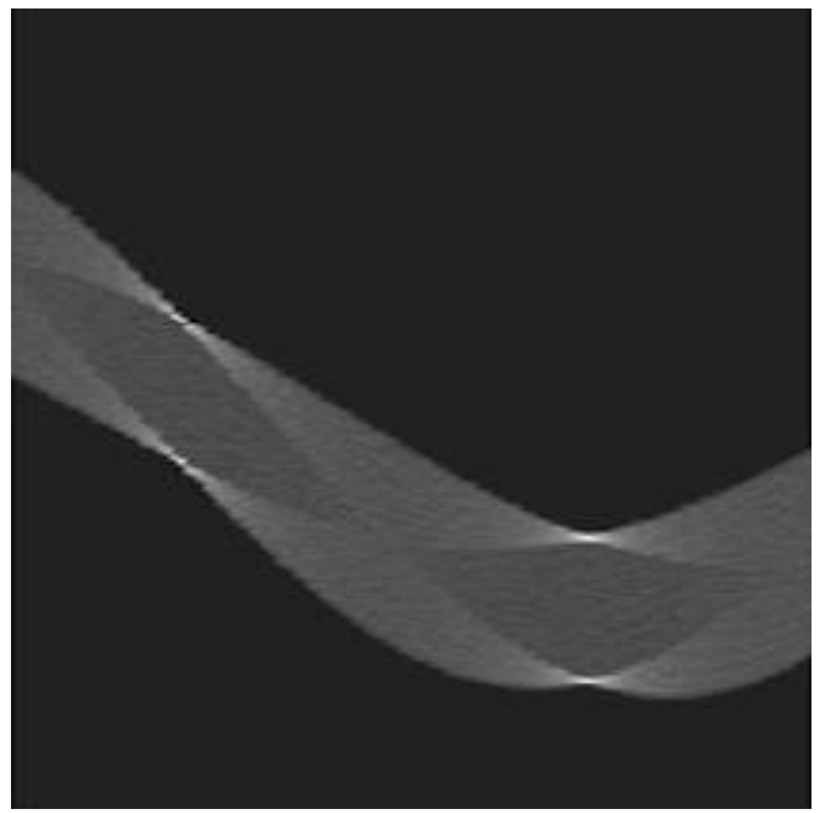


# Basic shapes

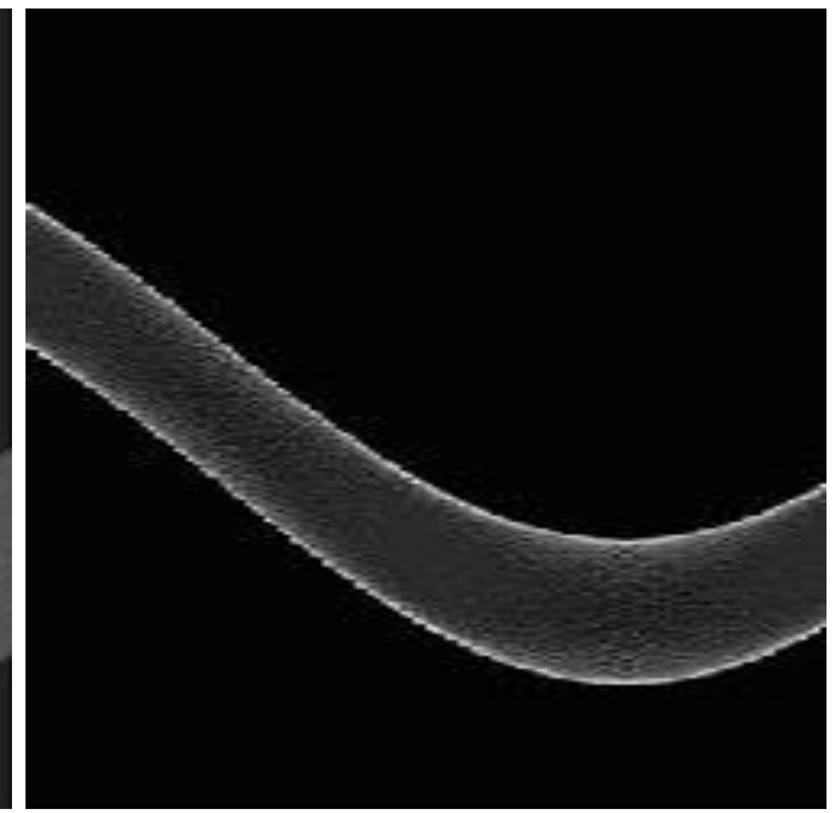
(in parameter space)



line

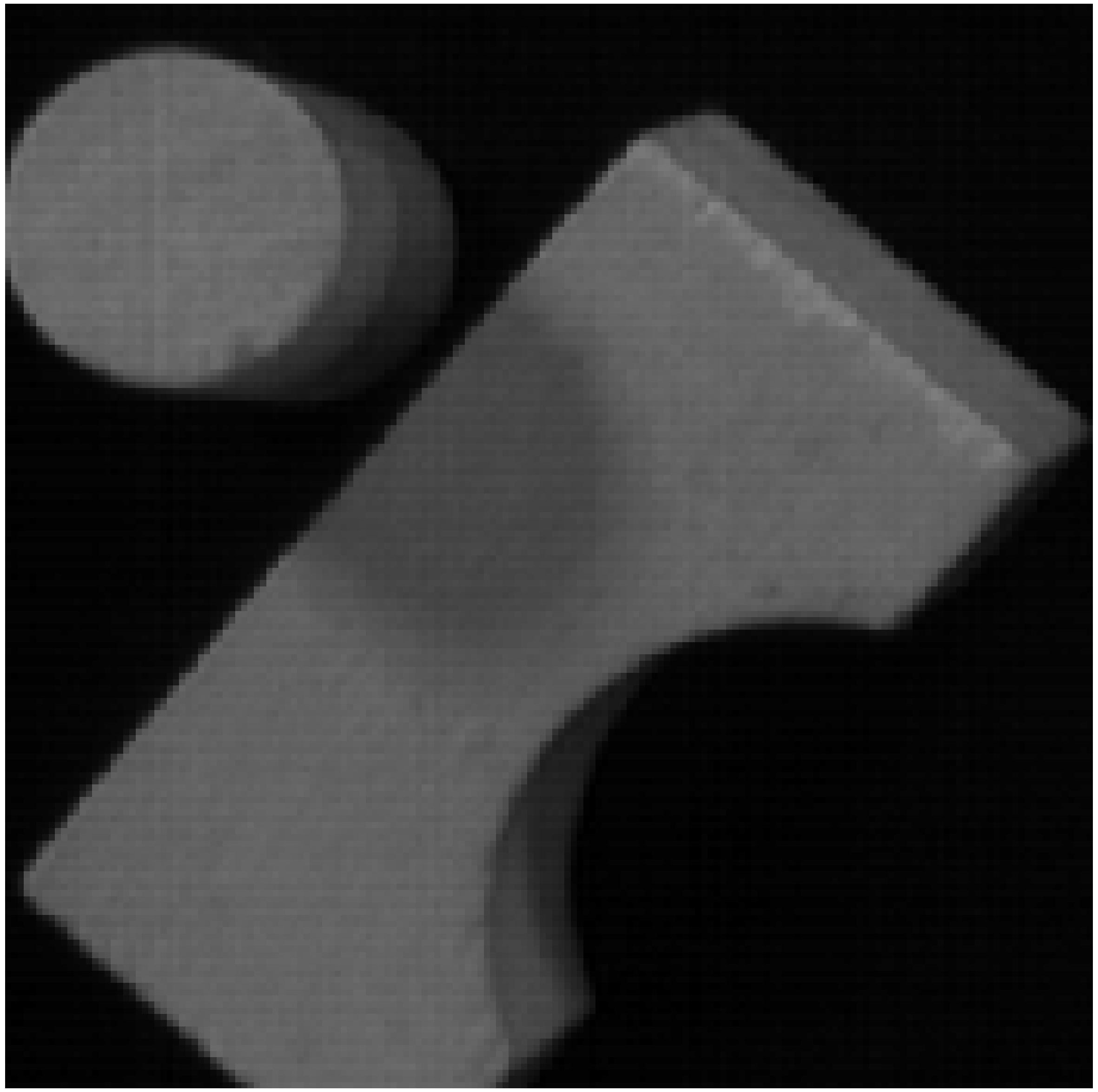


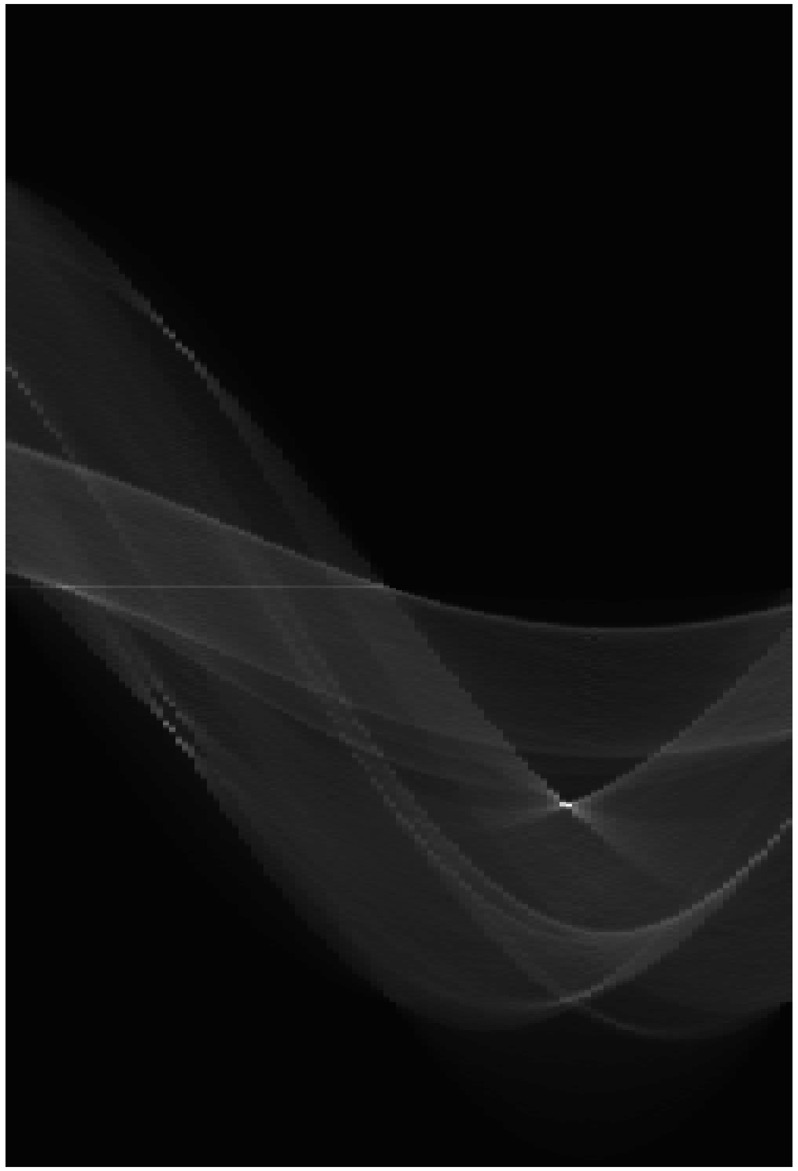
rectangle



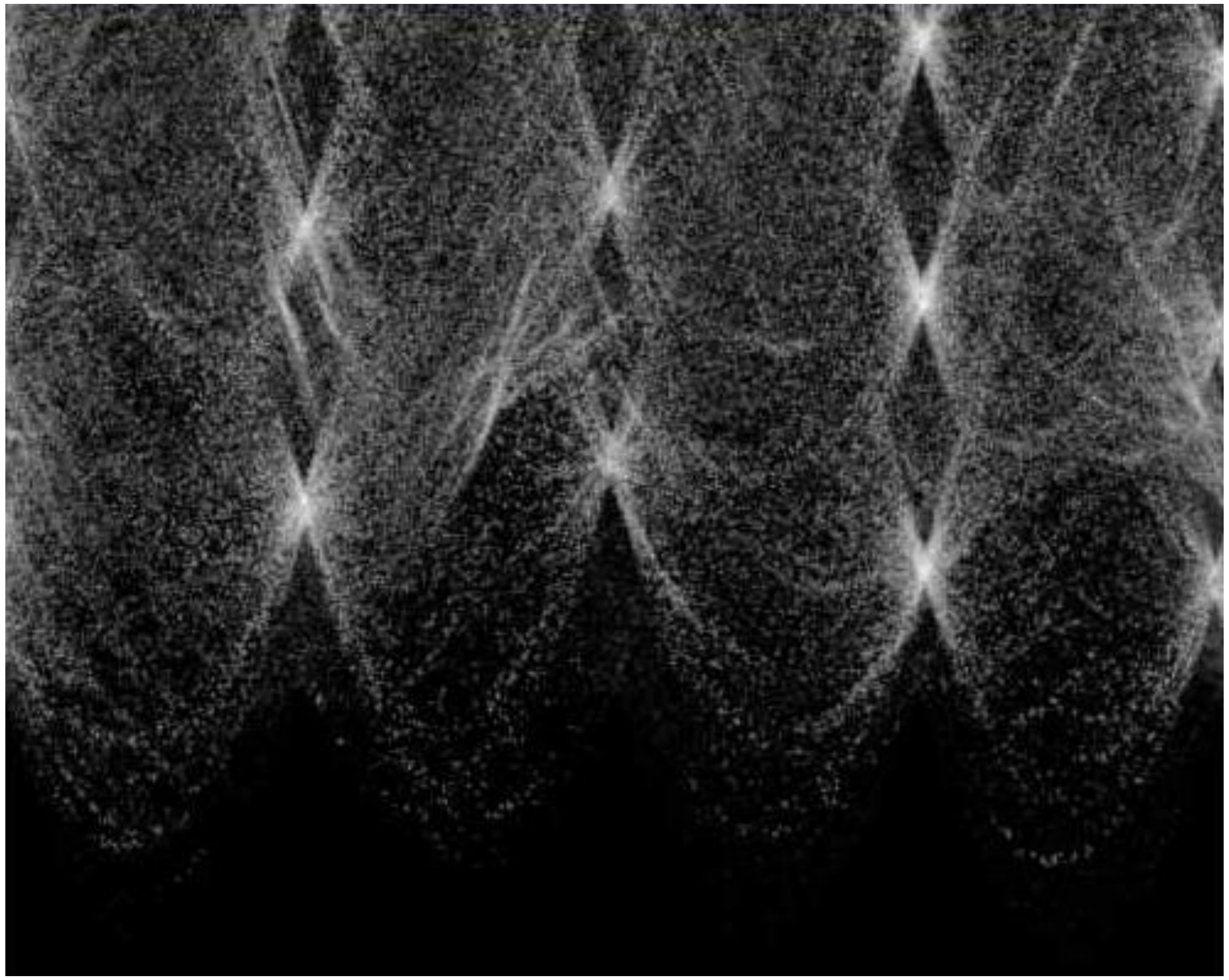
circle

# Basic Shapes





# More complex image



# In practice, measurements are noisy…

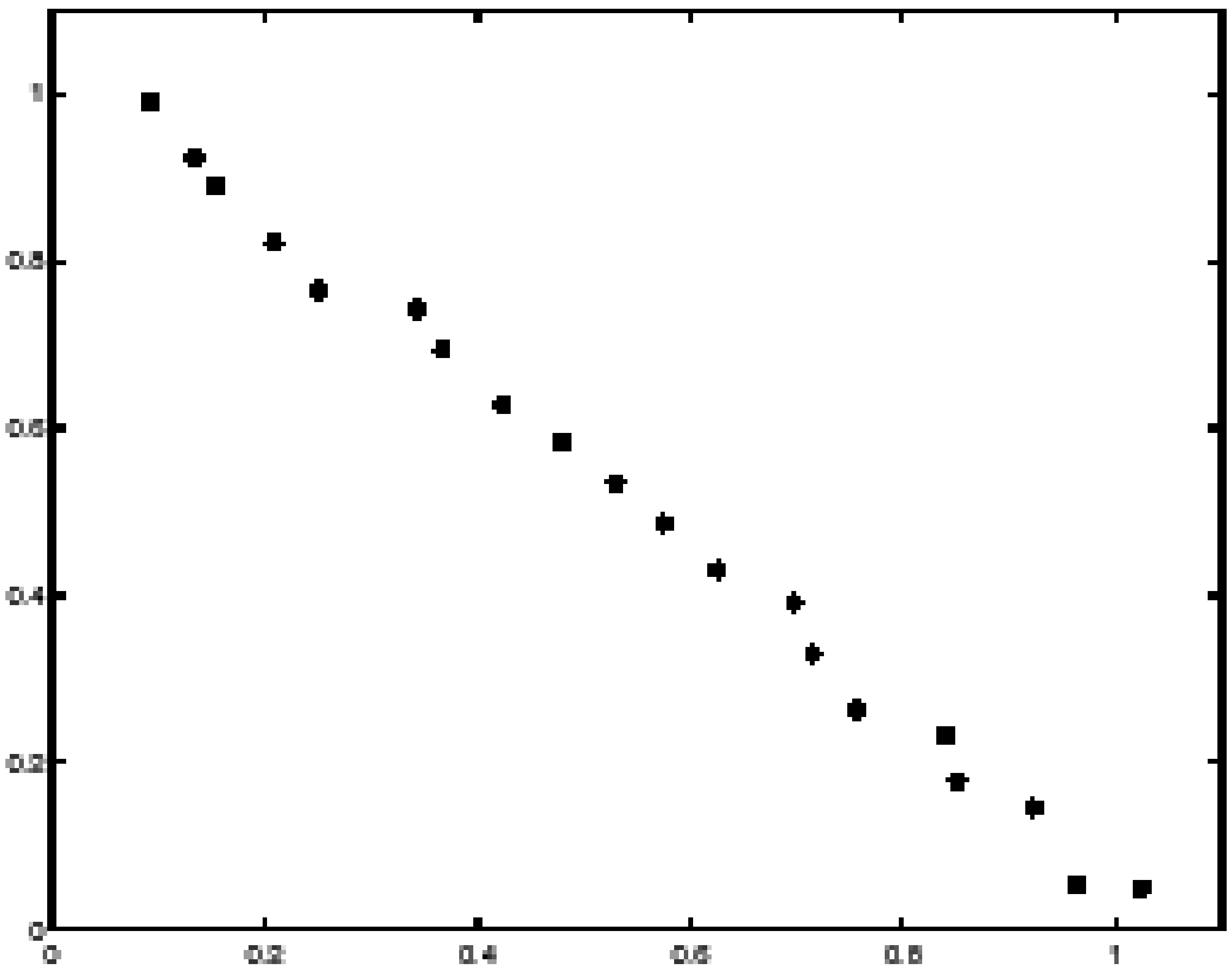
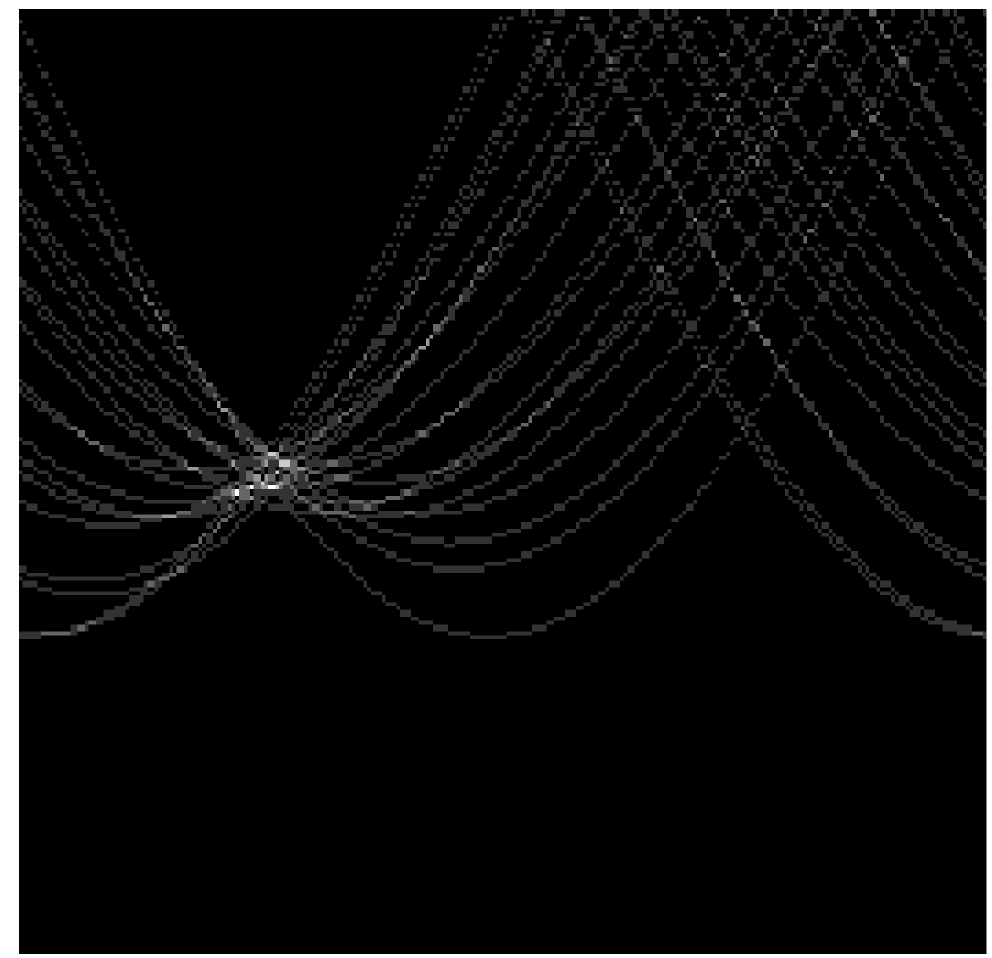



Image space



Votes

# Too much noise …

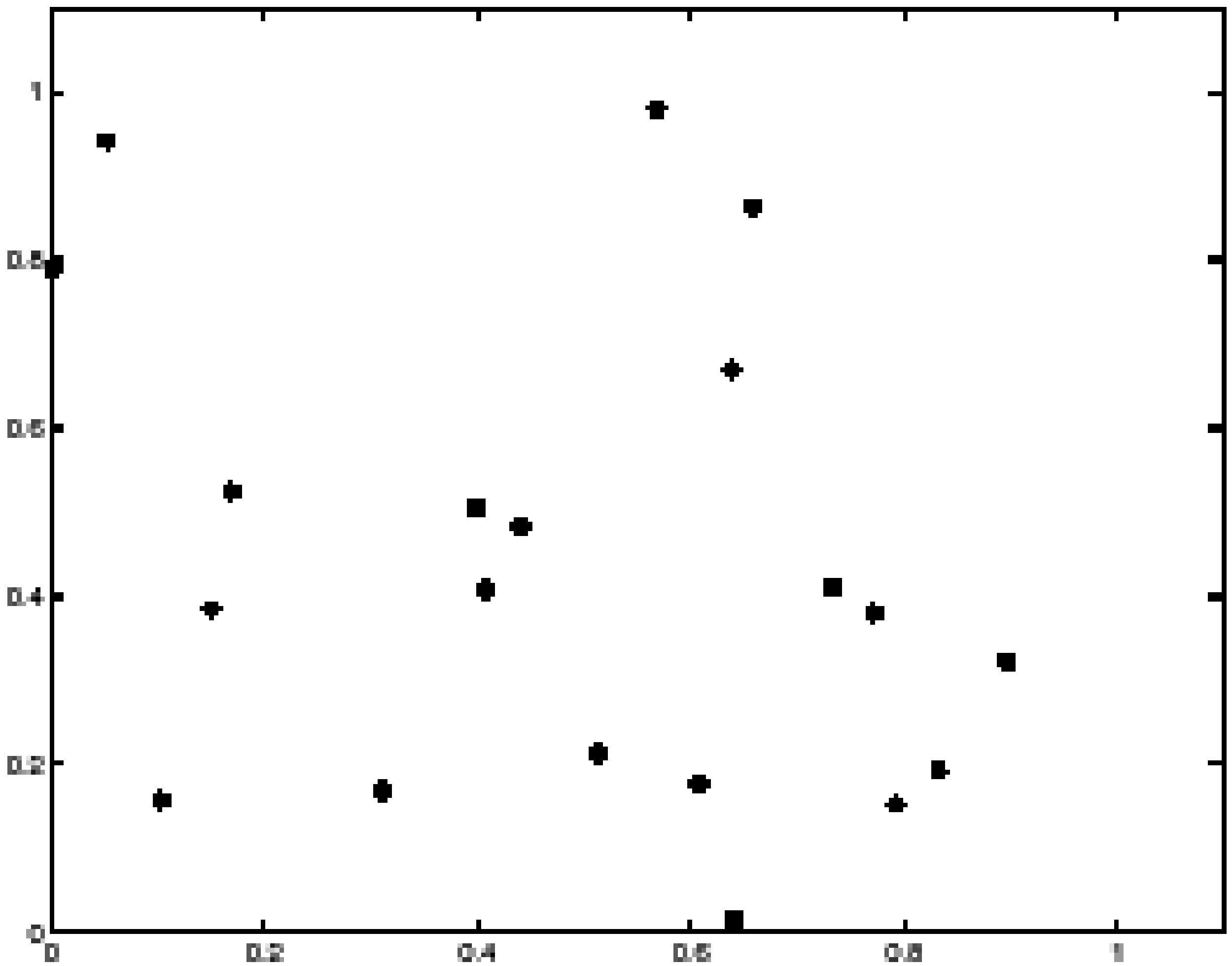
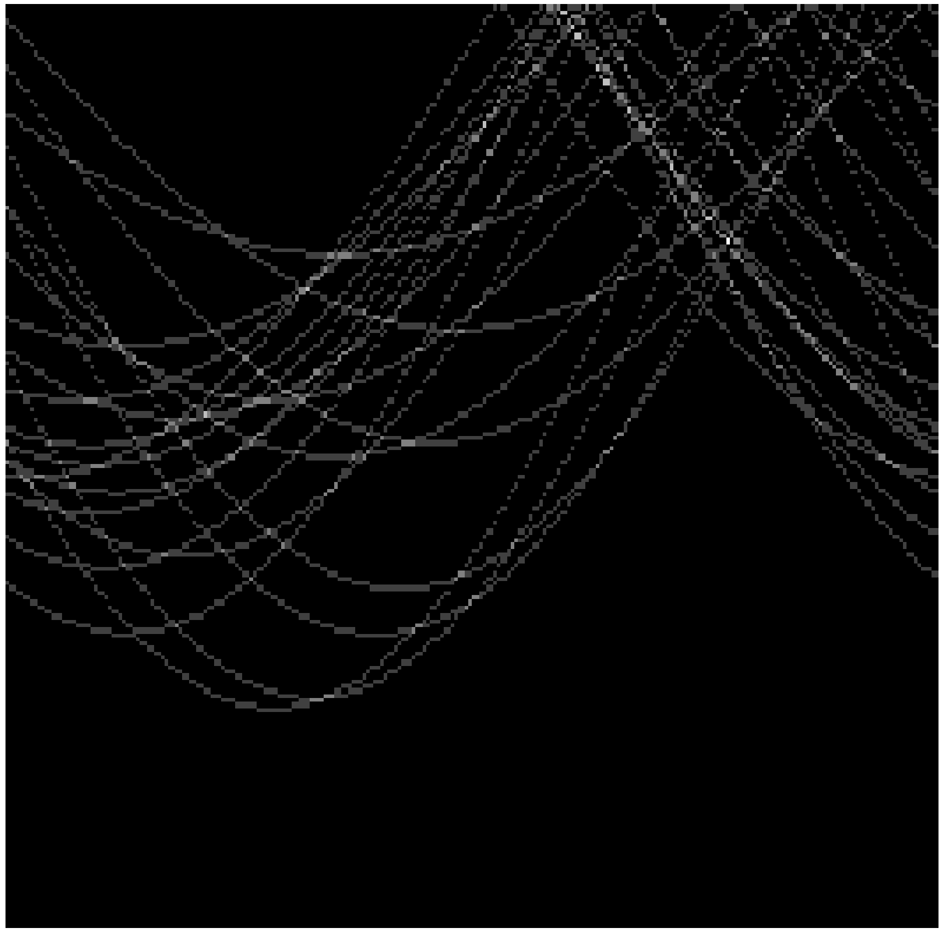
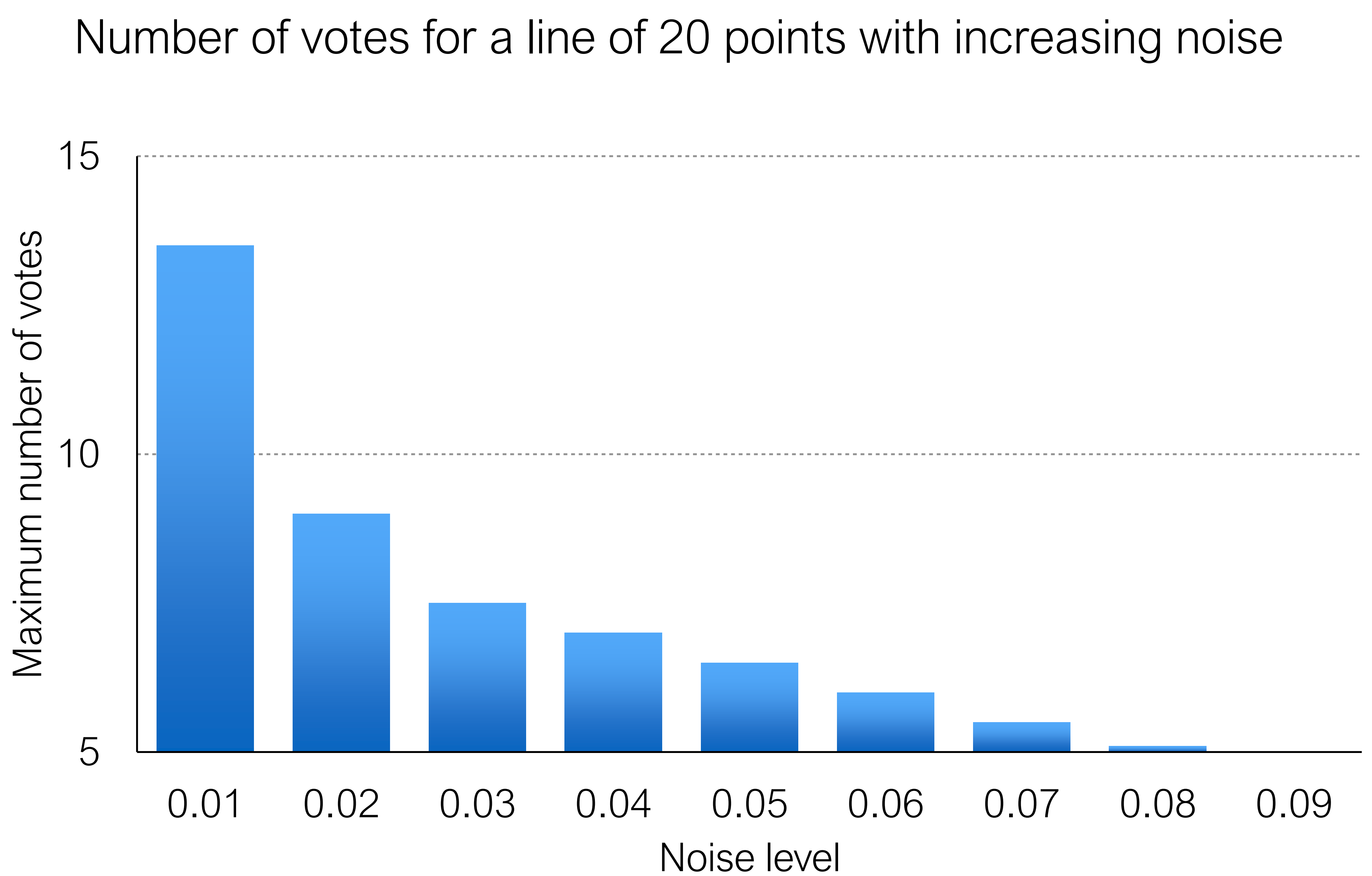



Image space



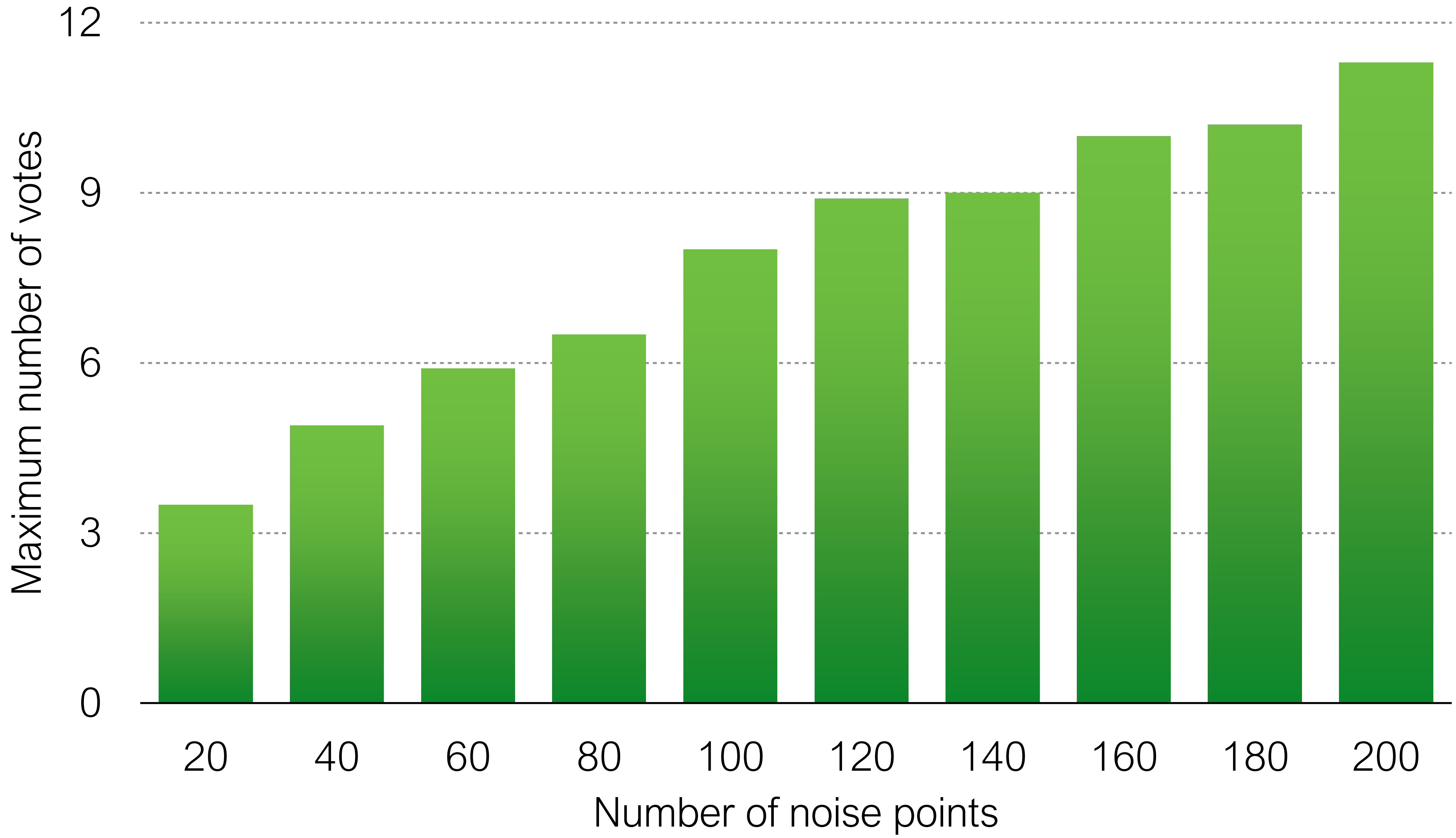
Votes

# Effects of noise level




More noise, fewer votes (in the right bin)

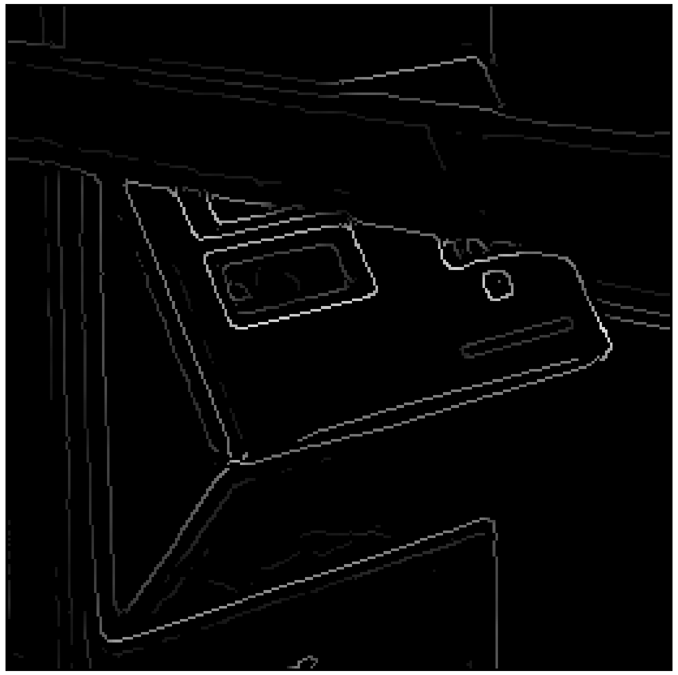
# Effect of noise points



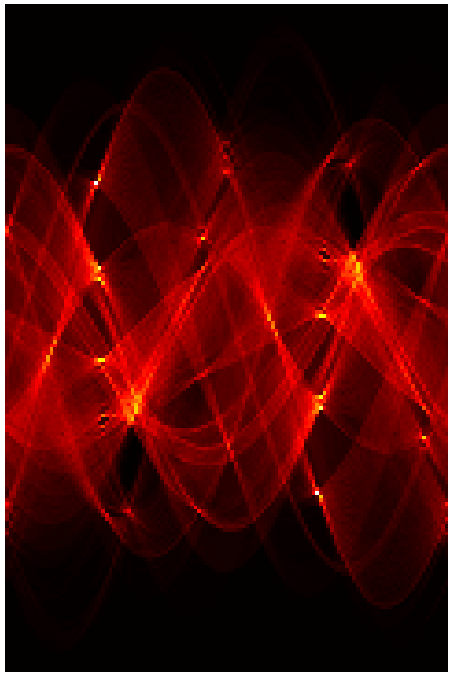

More noise, more votes (in the wrong bin)

# Real-world example

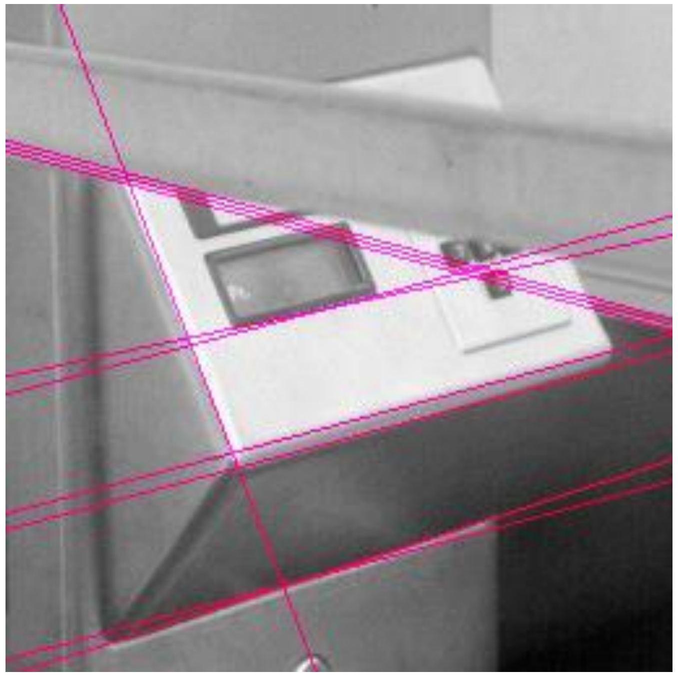
Original



Edges



parameter space



Hough Lines

# 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

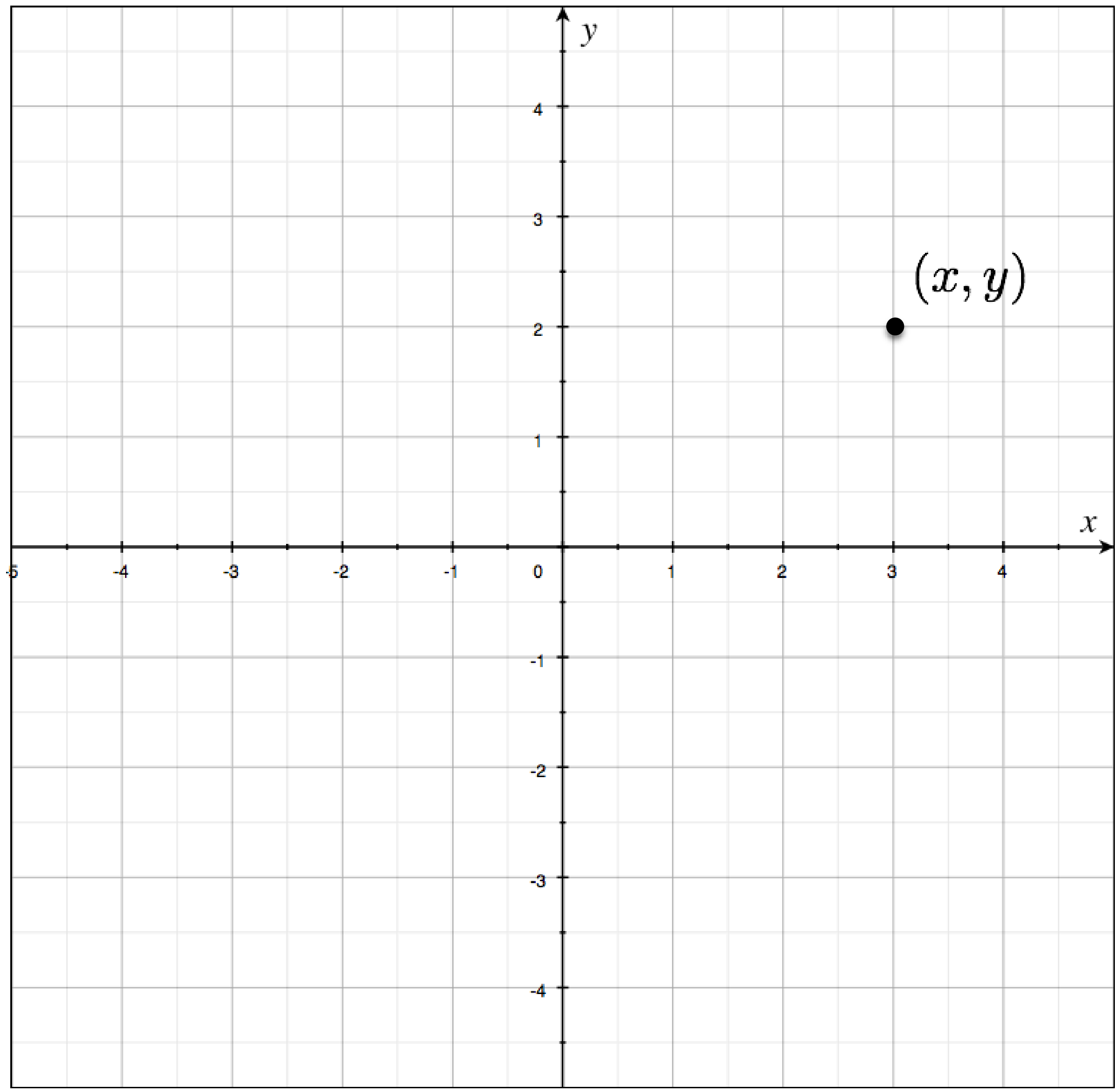
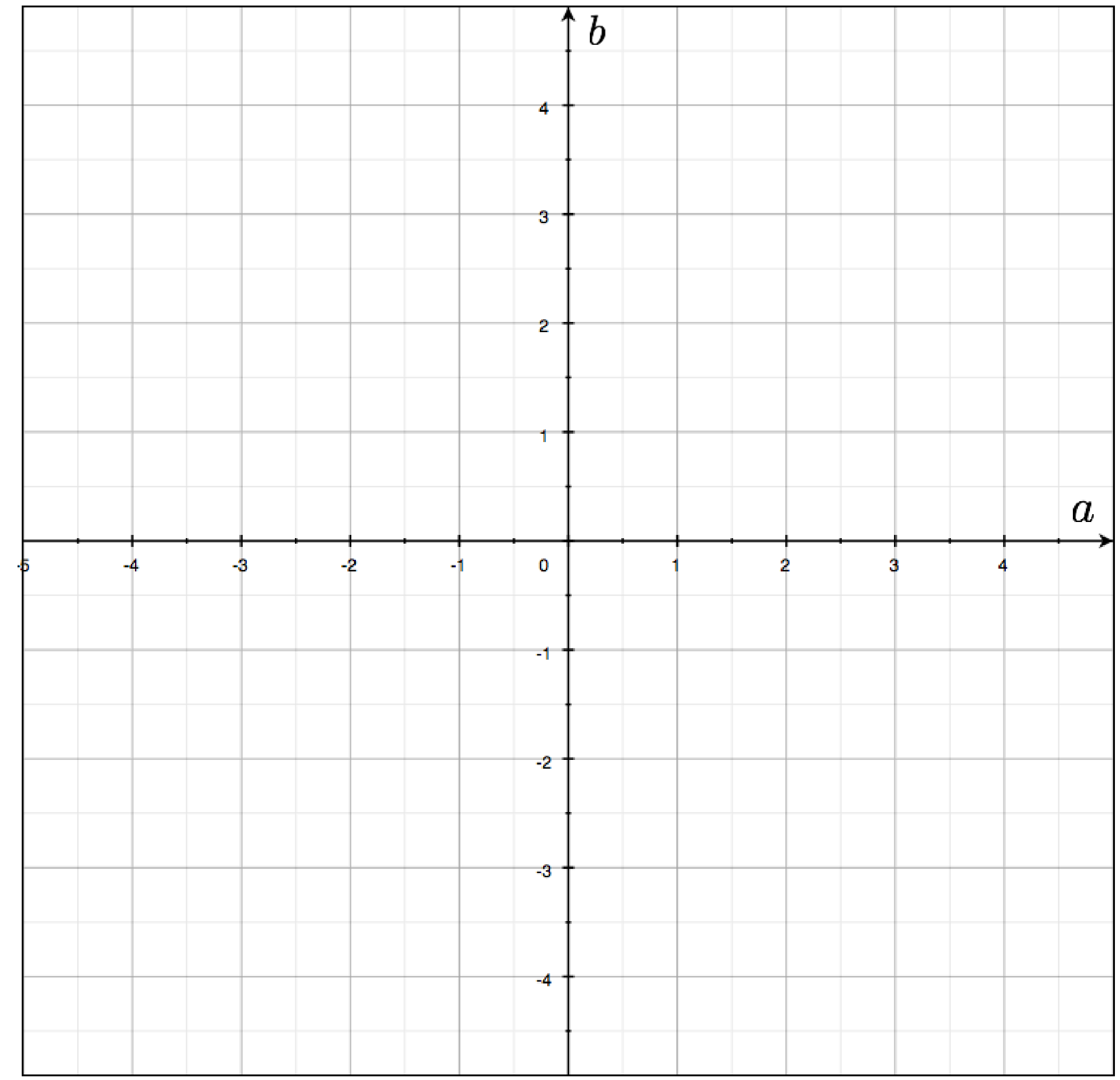



Image space




Parameter space

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

parameters

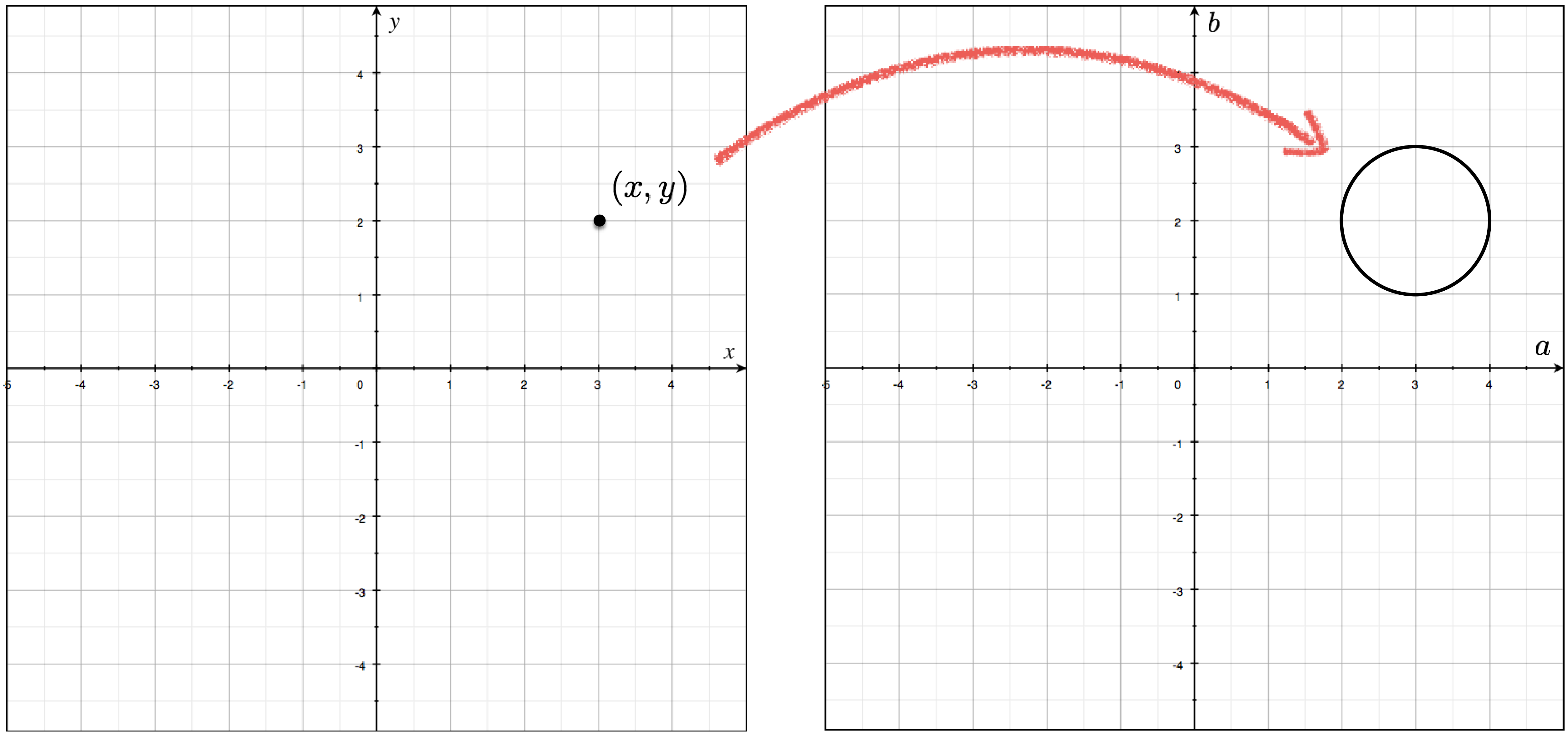
$$(x-a)^2+(y-b)^2=r^2$$

variables

parameters

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

variables



parameters

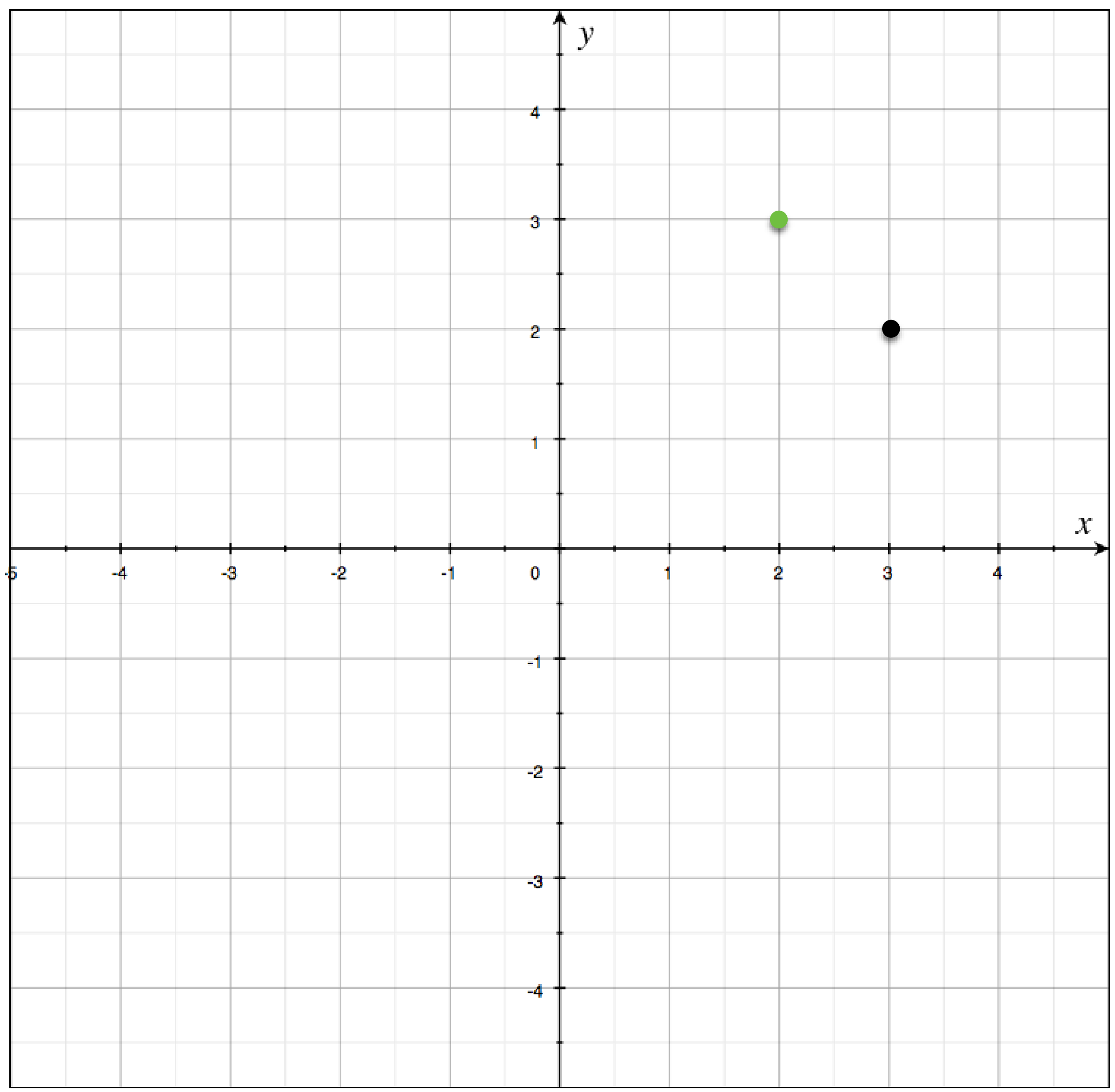
$$(x-a)^2+(y-b)^2=r^2$$

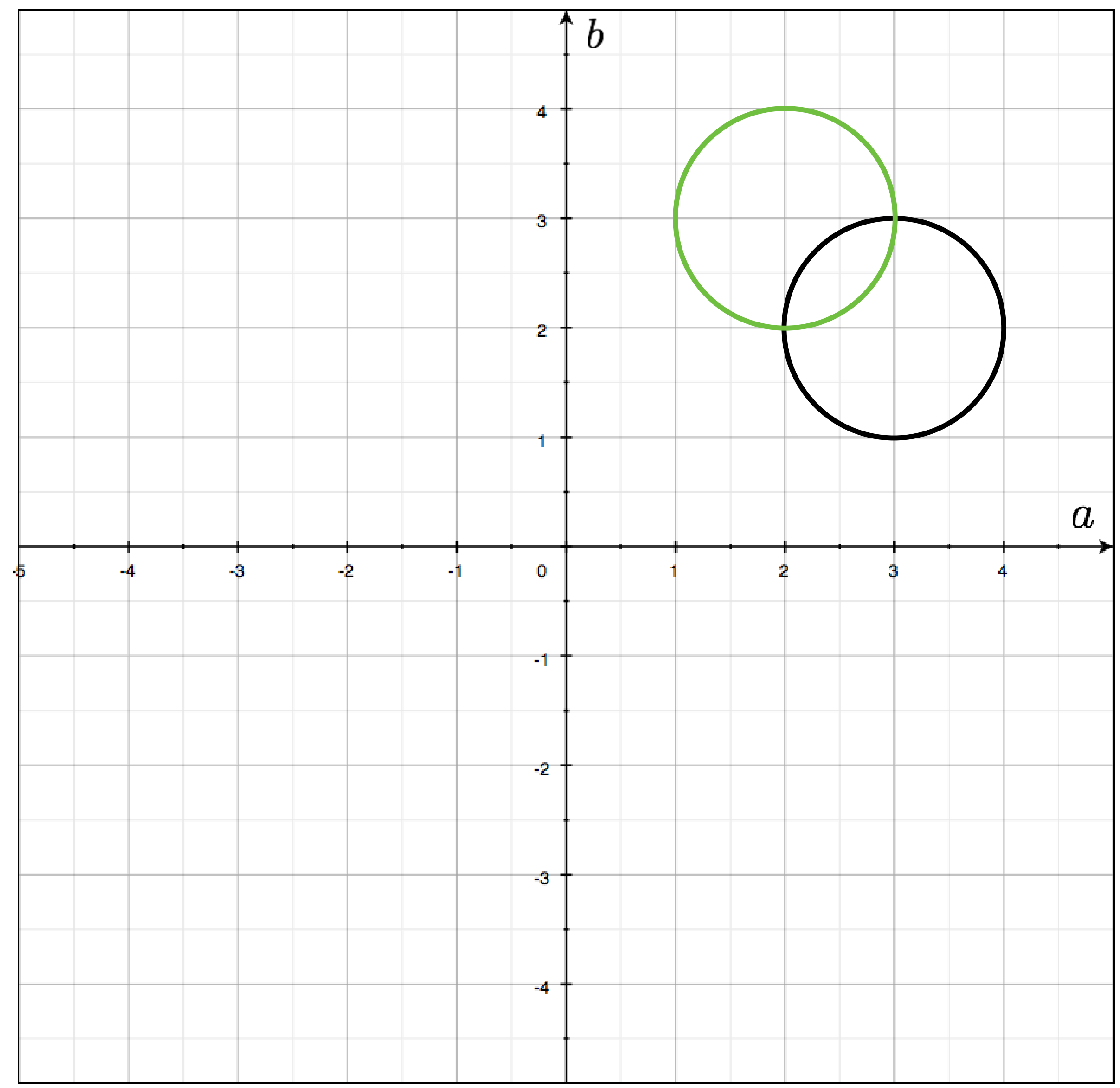
variables

parameters

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

variables

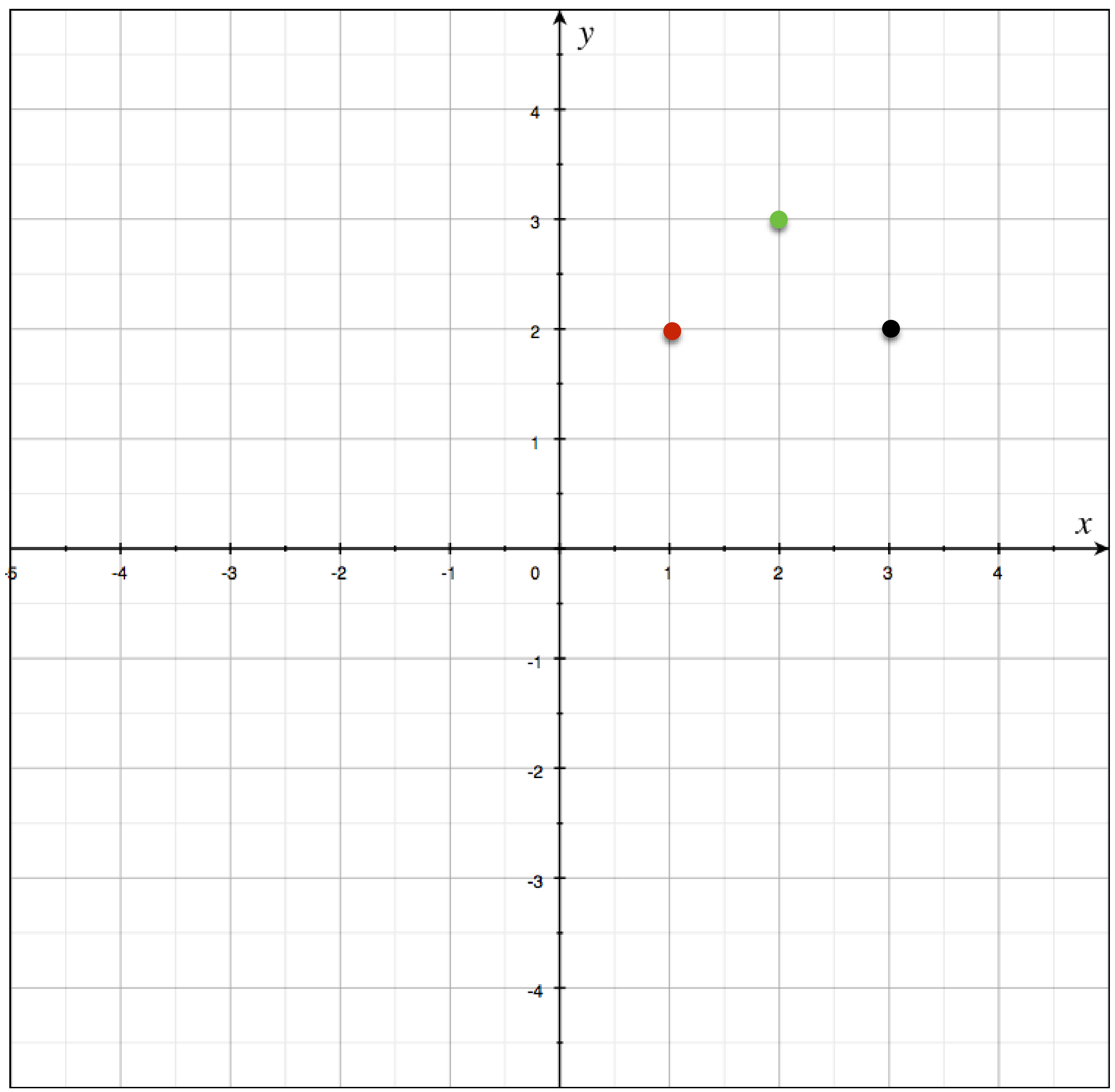


parameters

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

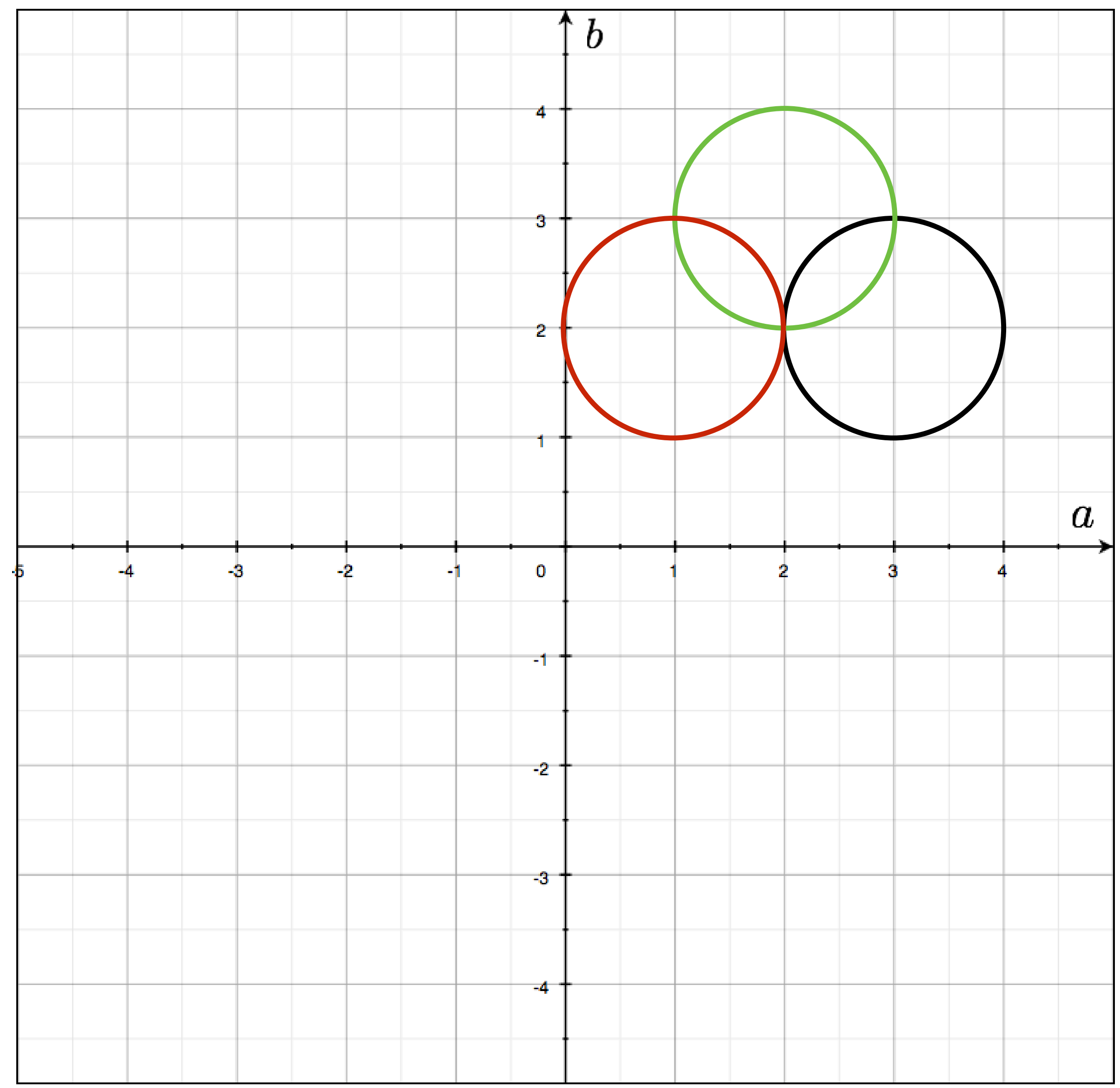
variables




parameters

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

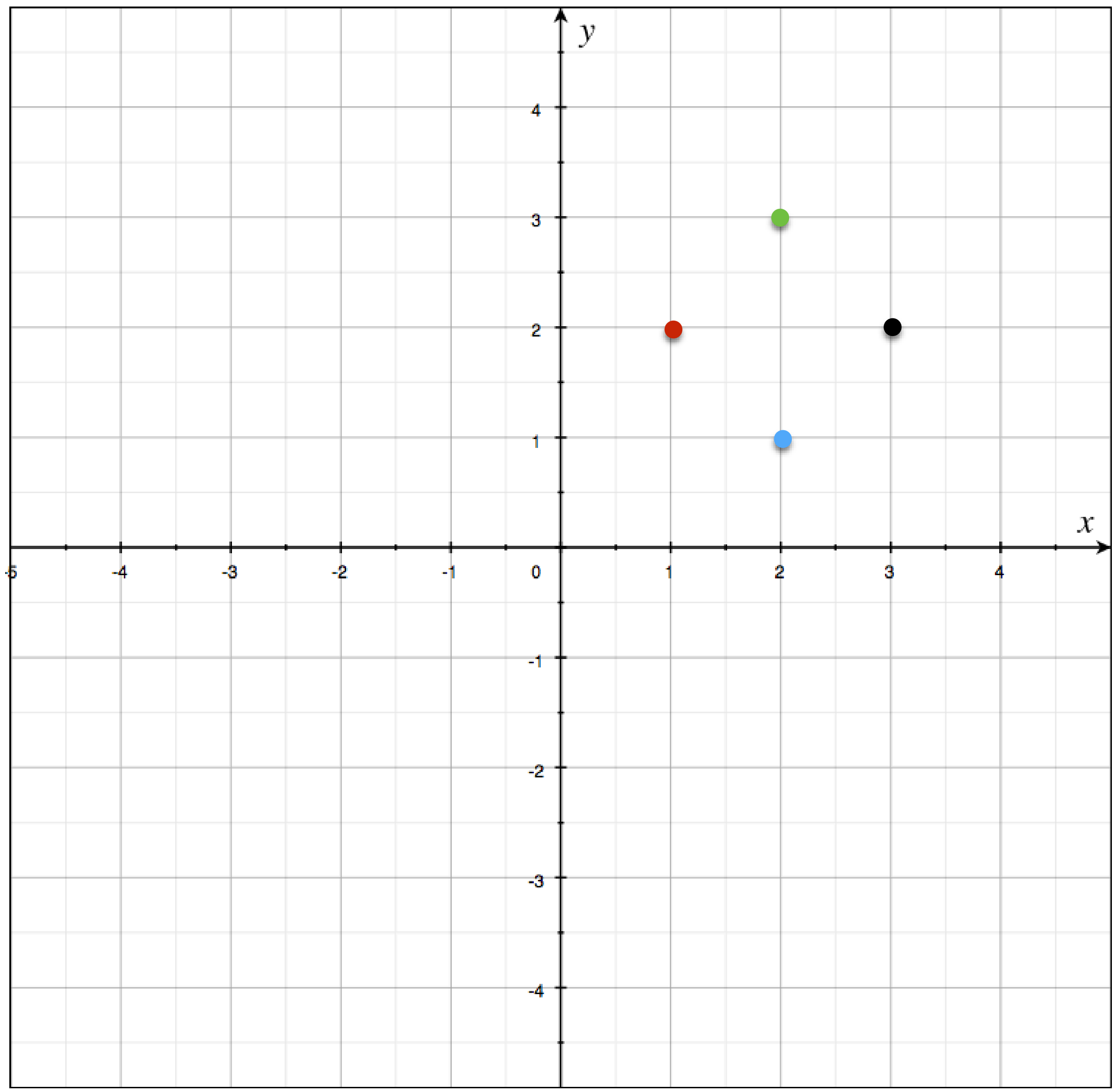
variables



parameters

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

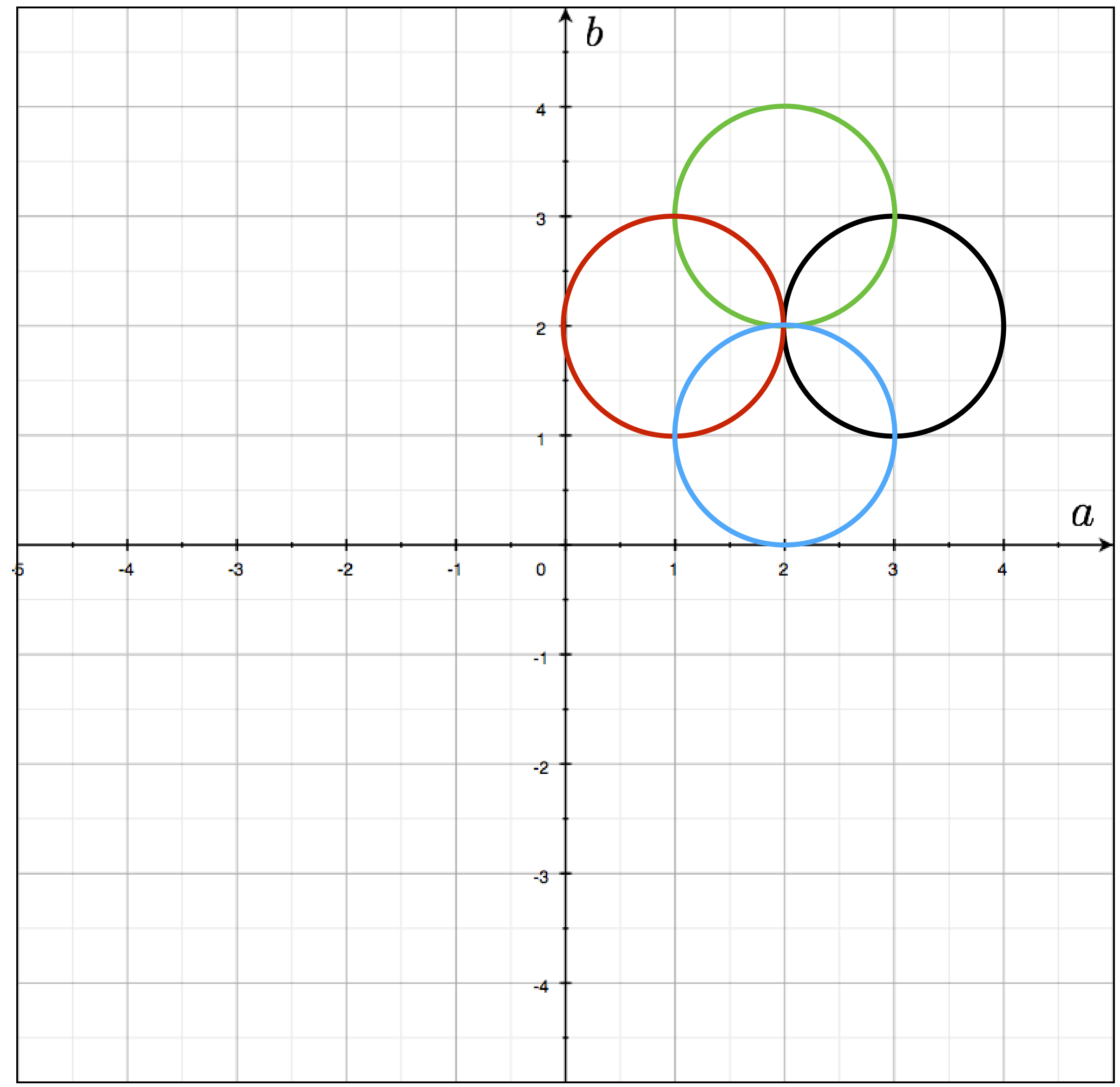
variables




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

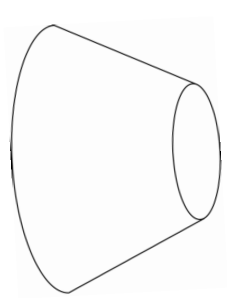
$$(x-a)^2+(y-b)^2=r^2$$

variables

If radius is not known: 3D Hough Space!

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

Surface shape in Hough space is complicated

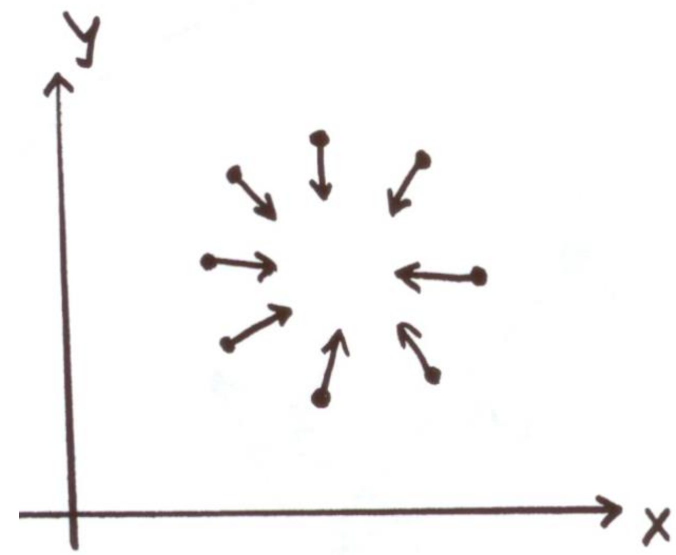


# Using Gradient Information

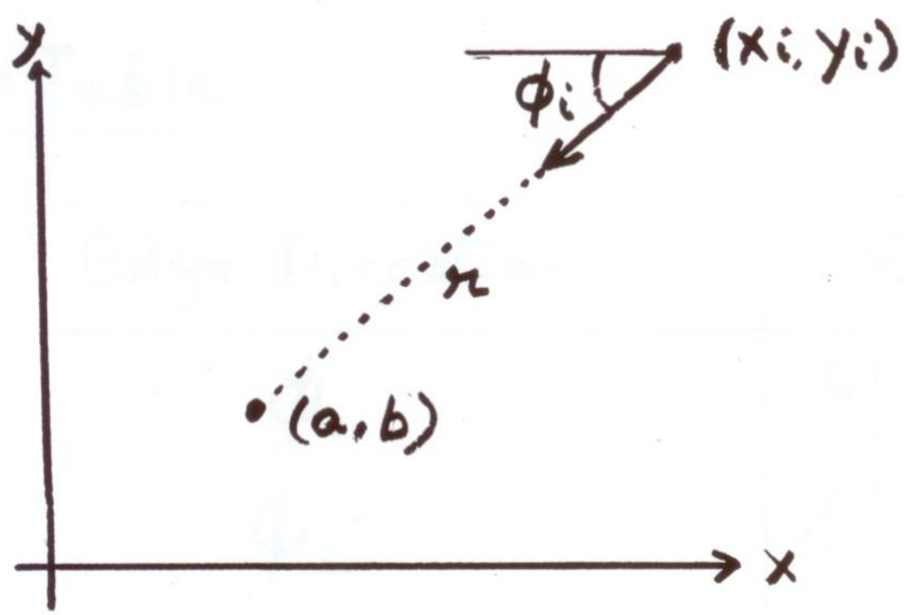
Gradient information can save lot of computation:

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

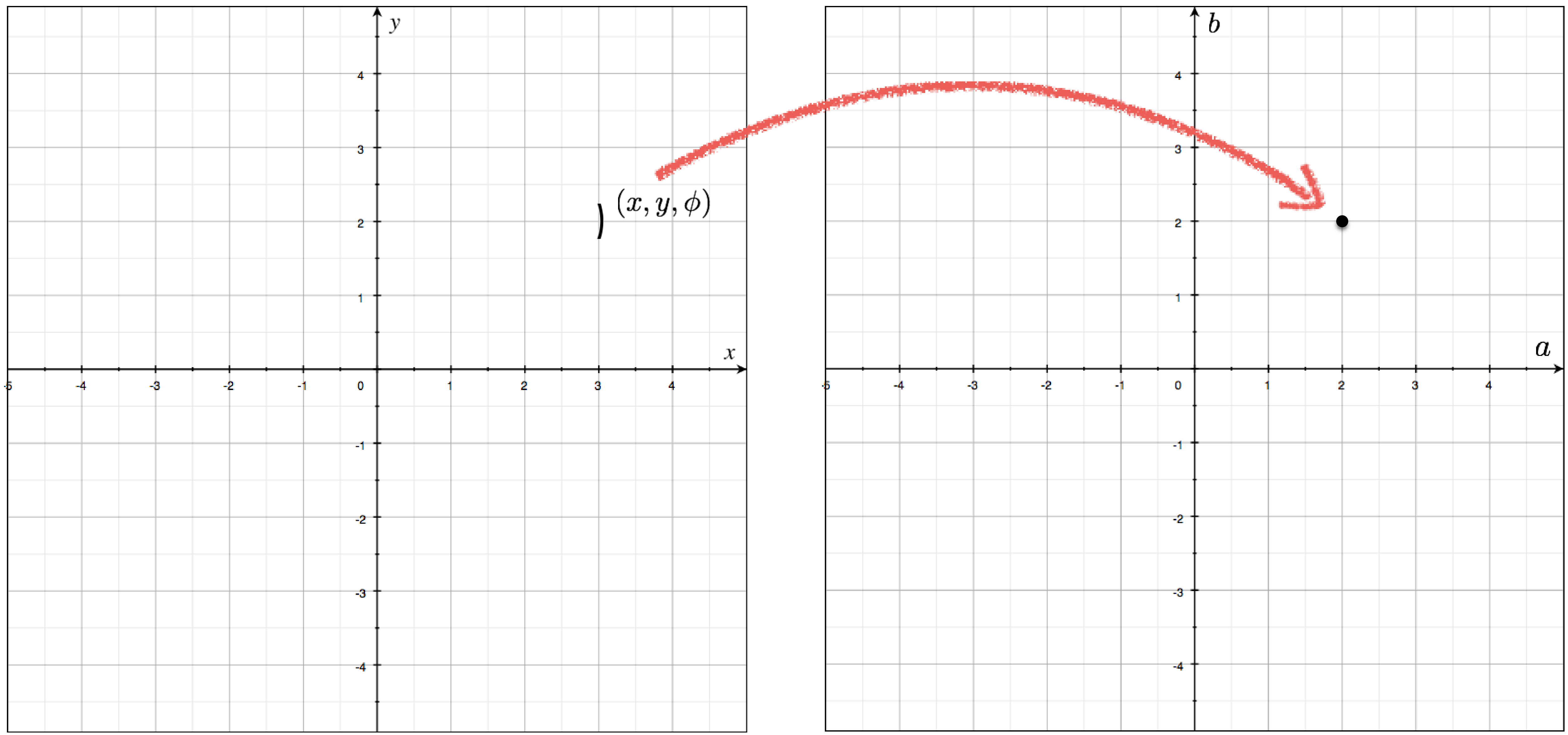
$$(x - a)^2 + (y - b)^2 = r^2$$

variables

parameters

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

variables



parameters

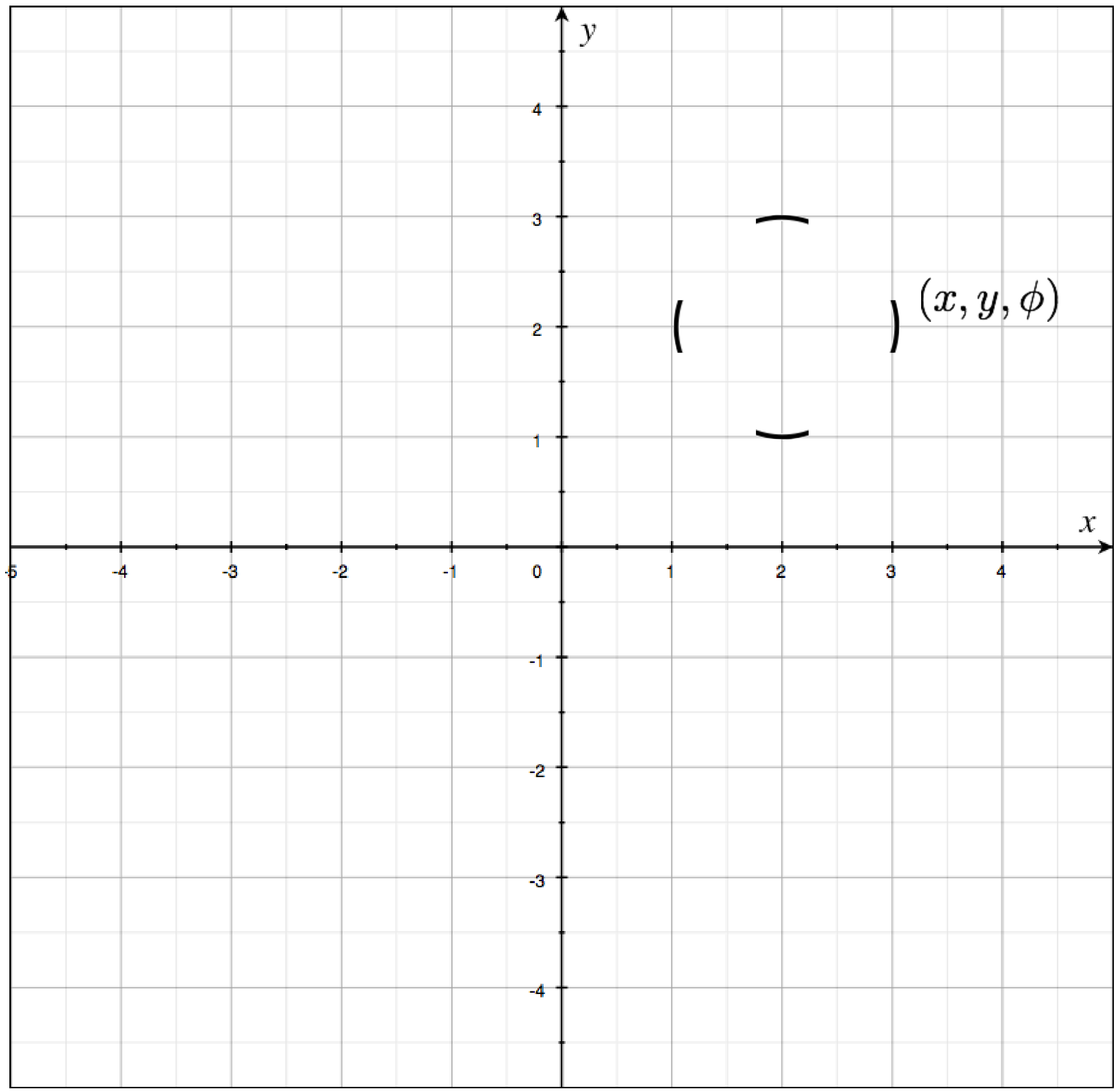
$$(x-a)^2+(y-b)^2=r^2$$

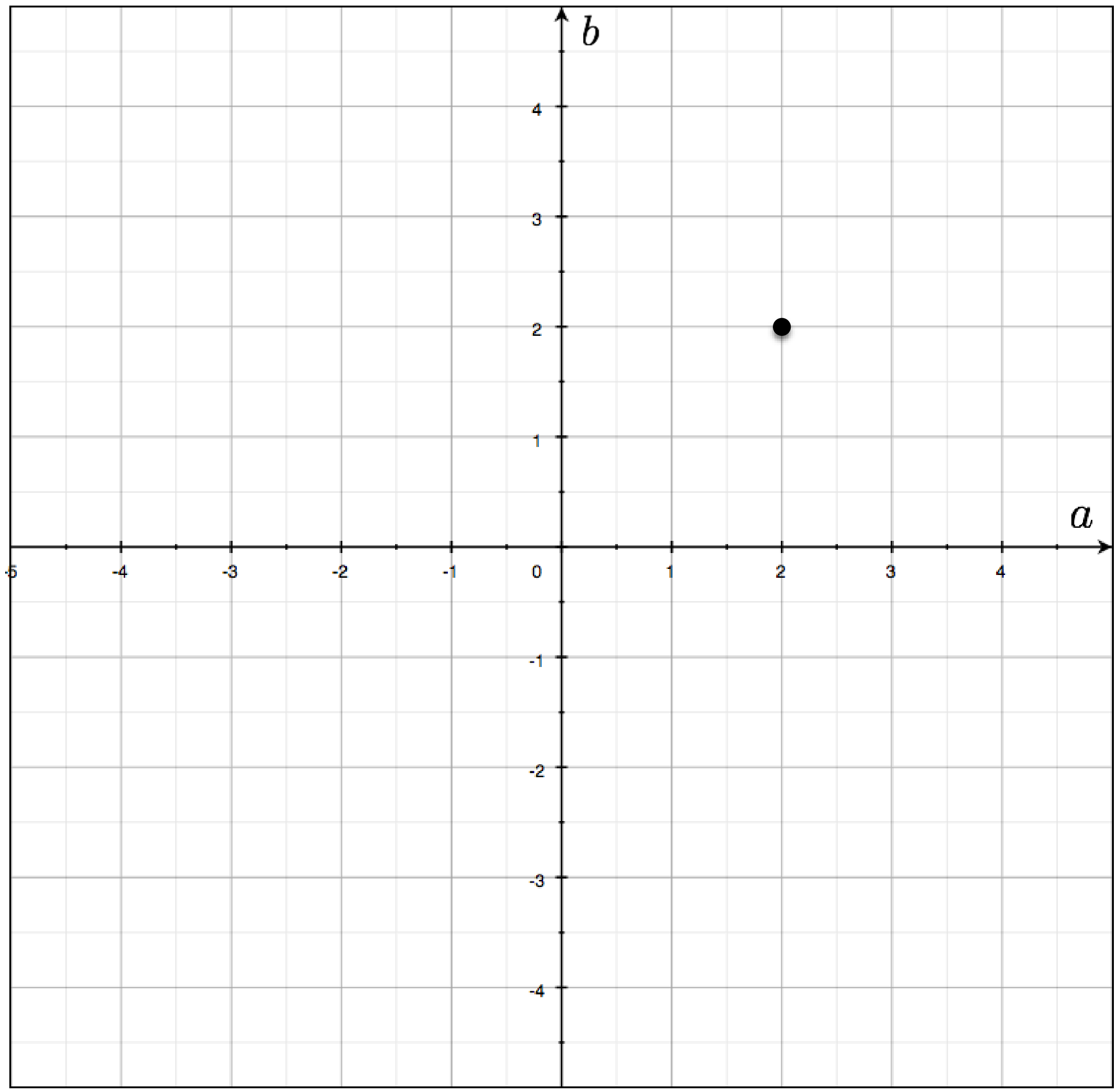
variables

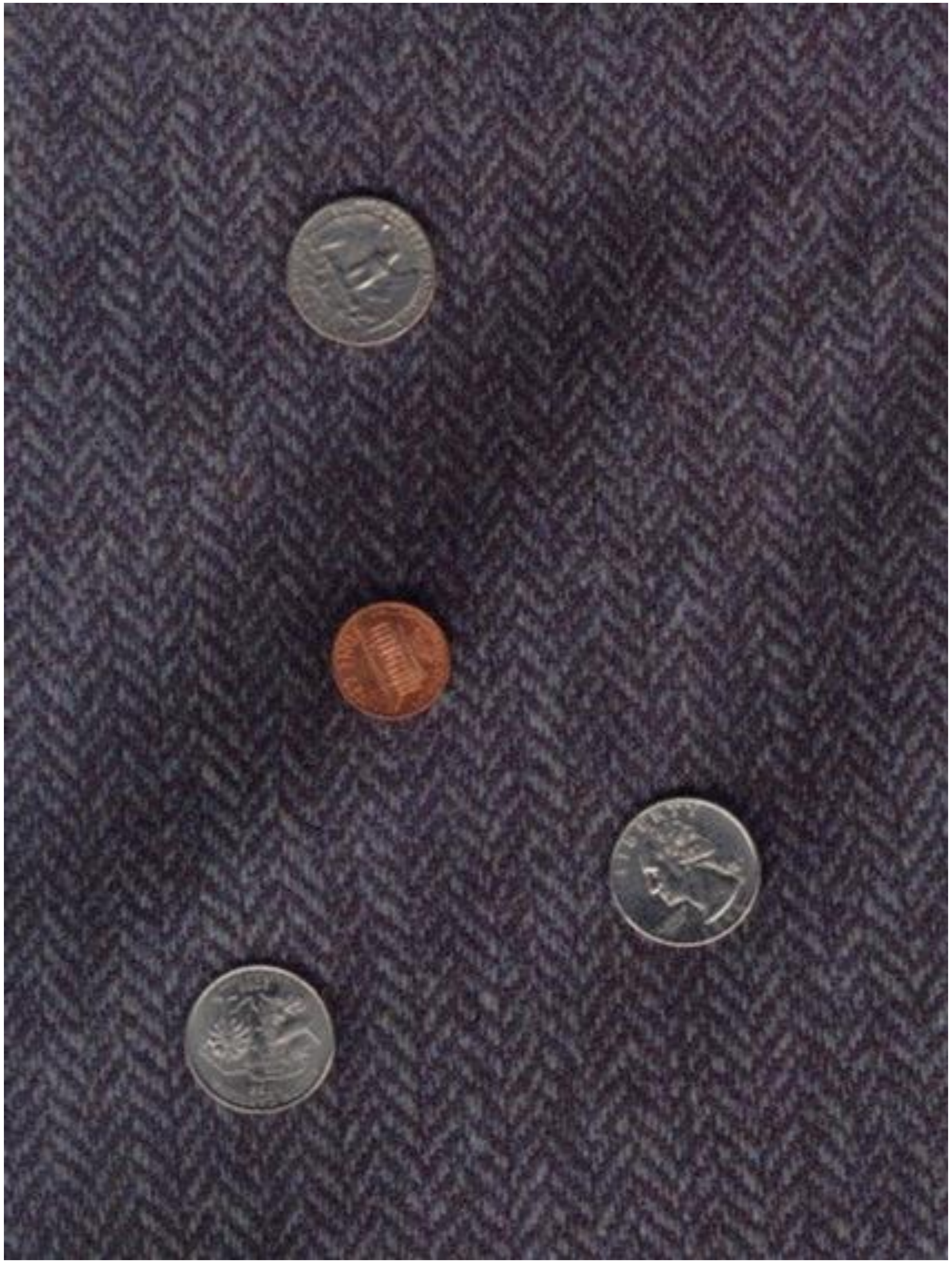
parameters

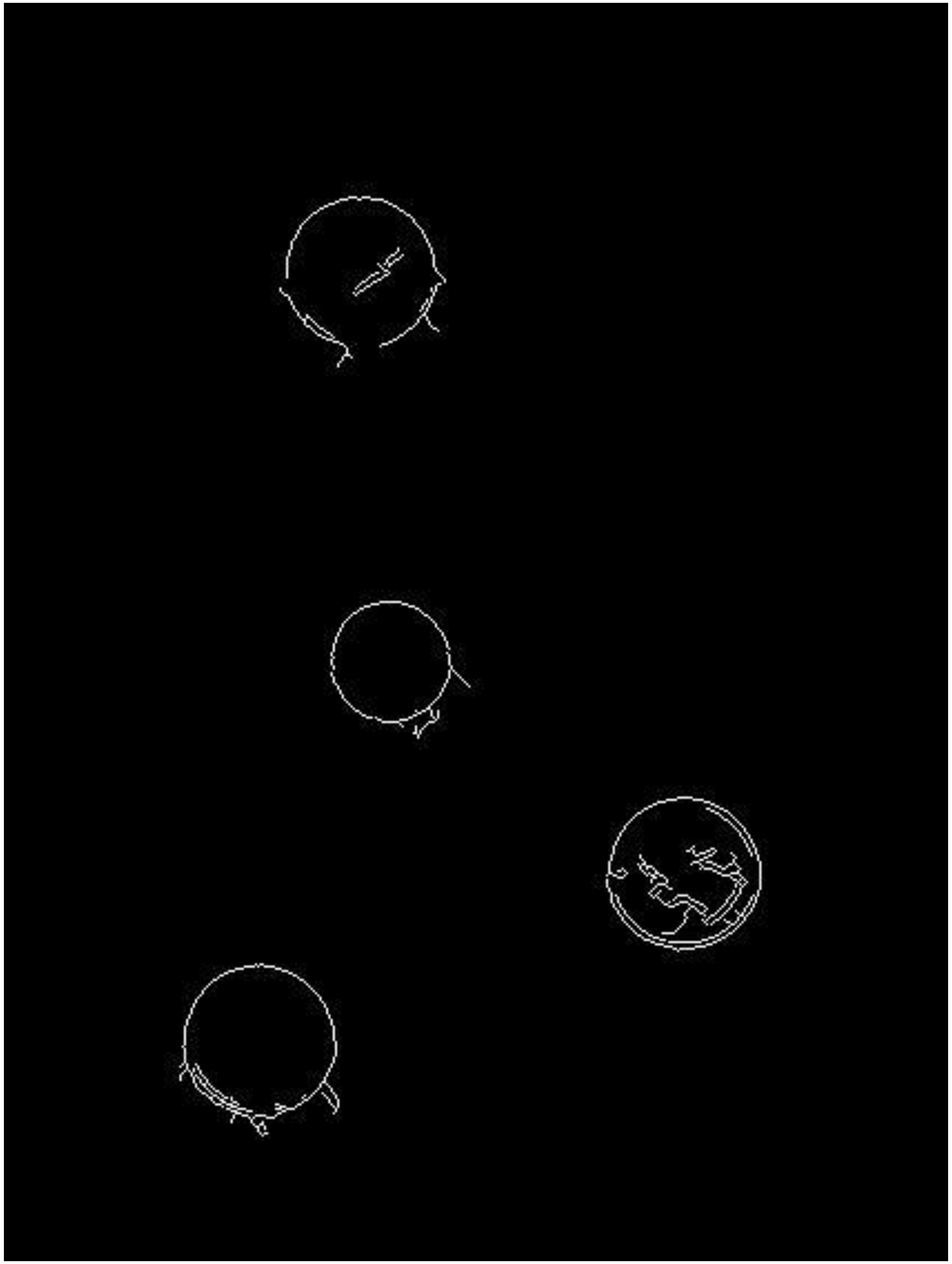
$$(x-a)^2+(y-b)^2=r^2$$

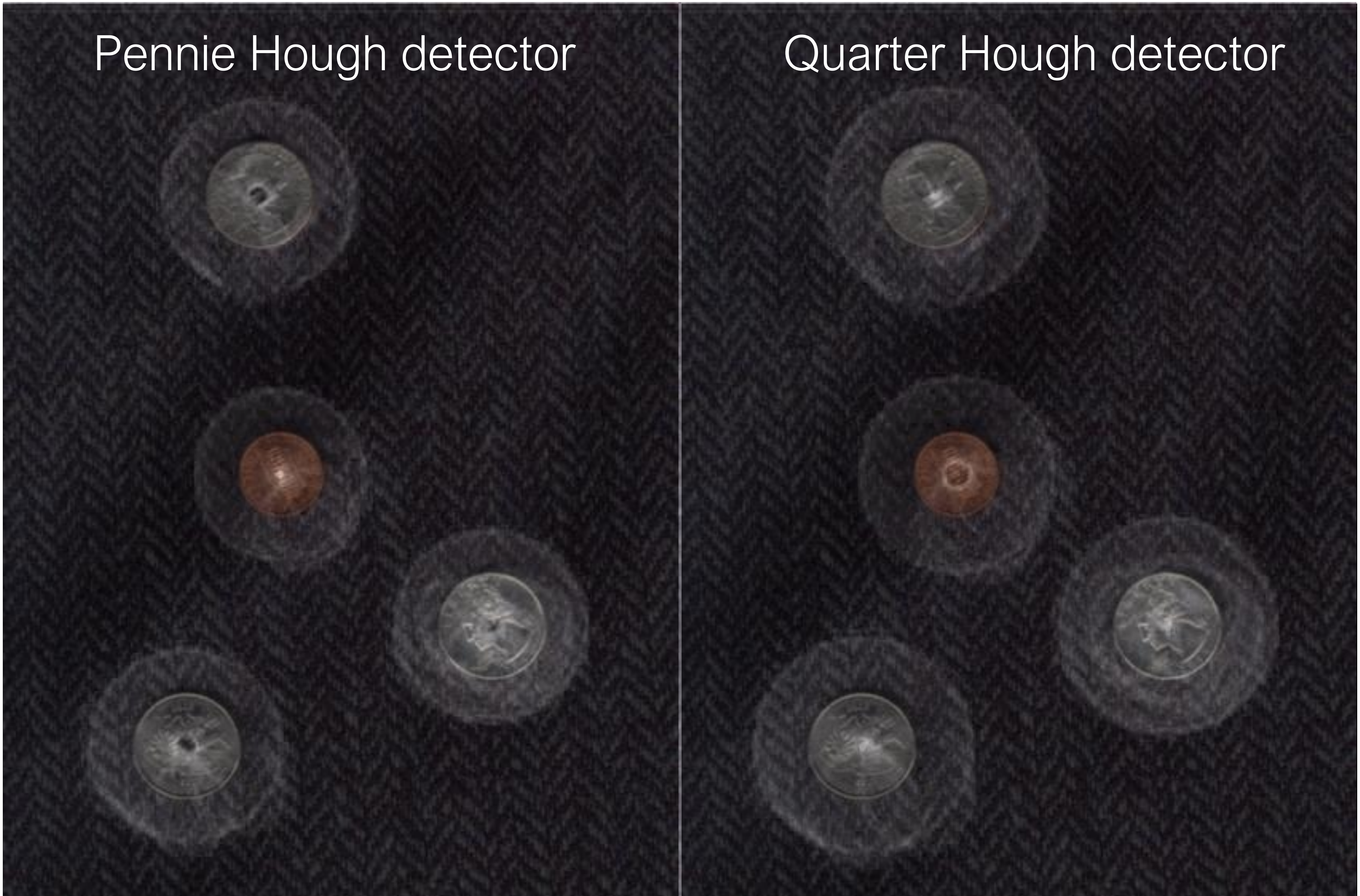
variables

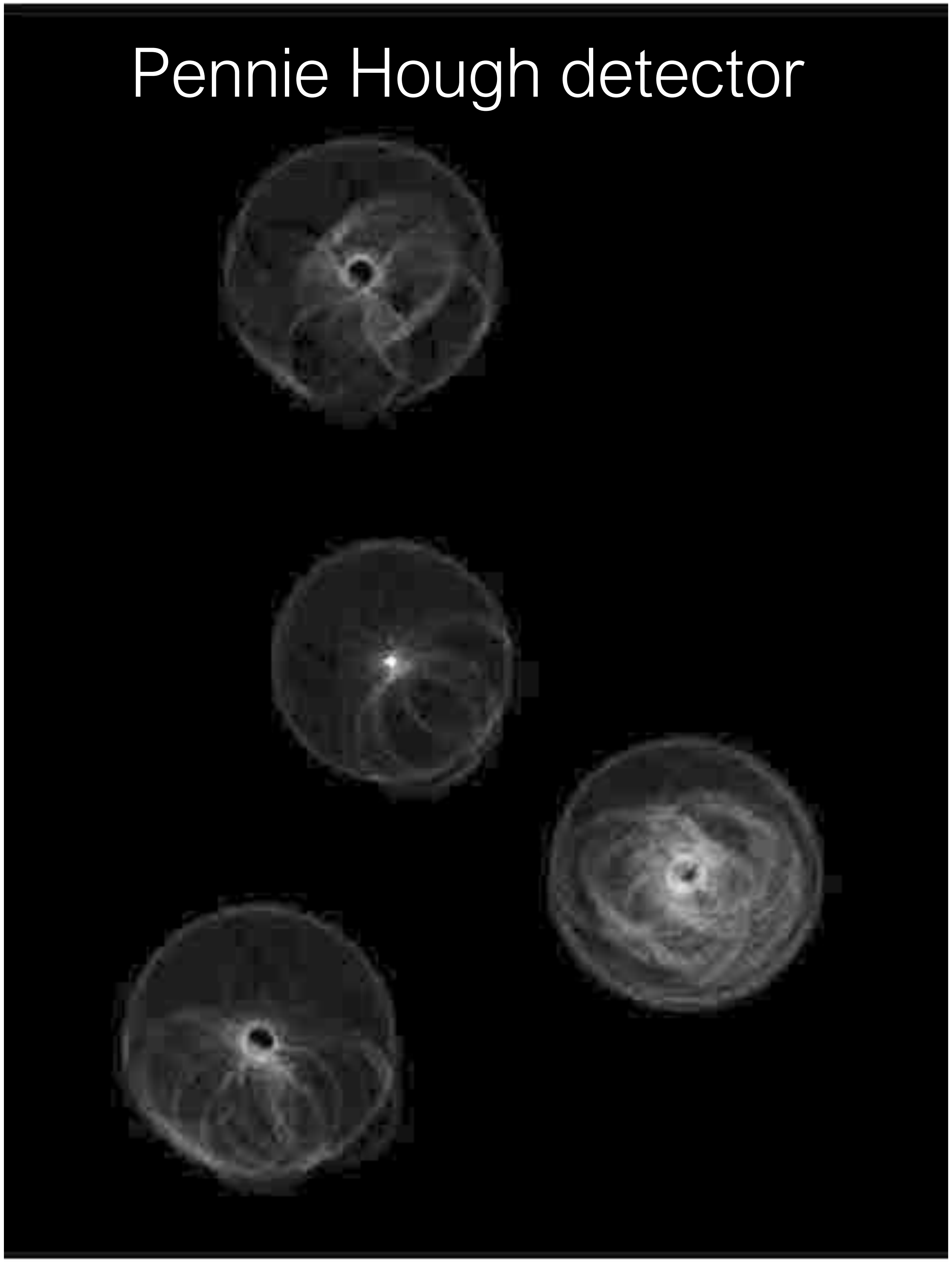
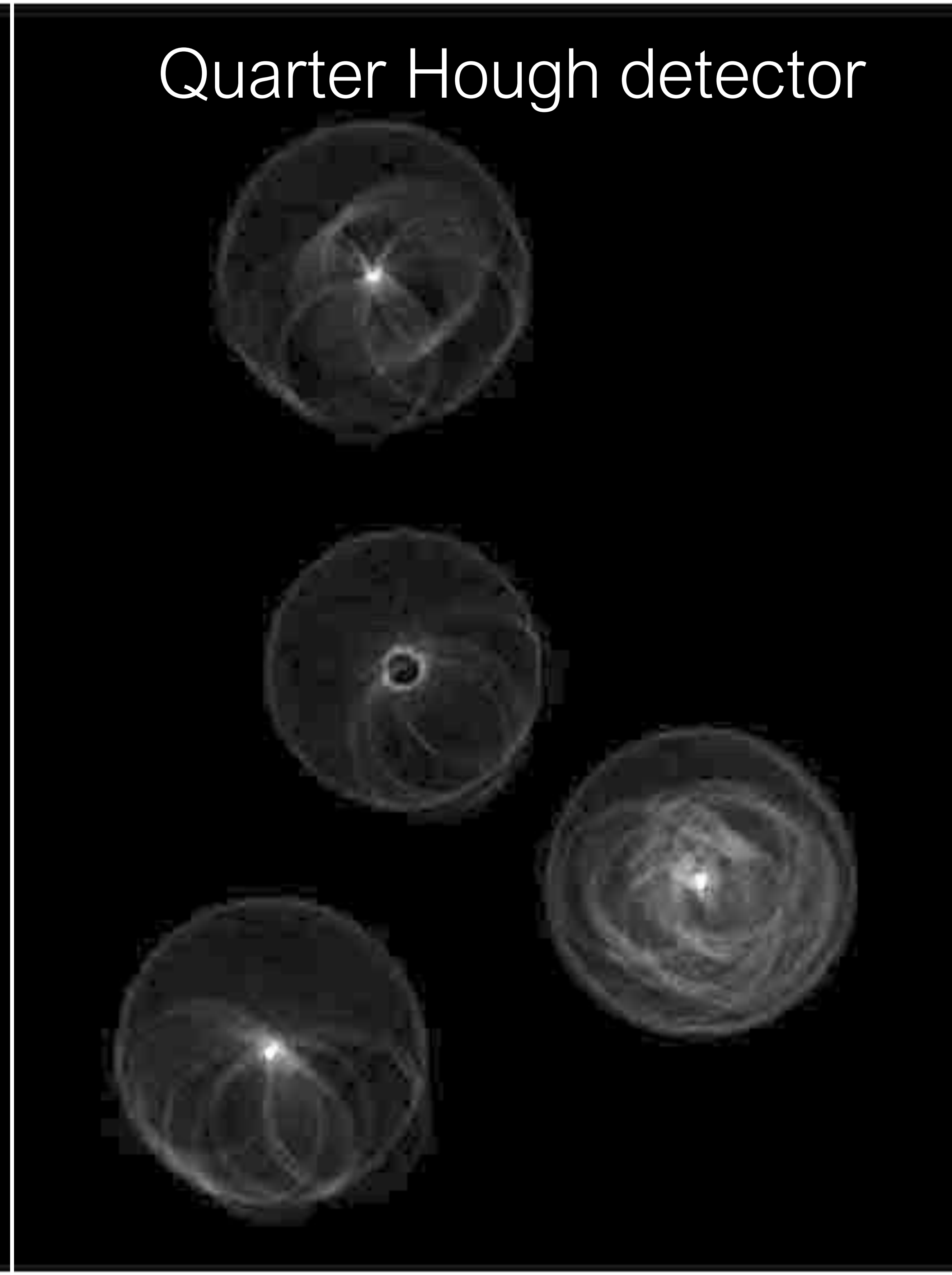







Pennie Hough detector
Quarter Hough detector


Pennie Hough detector

Quarter Hough detector

# The Hough transform …

Deals with occlusion well? 

Detects multiple instances? 

Robust to noise? 

Good computational complexity? 

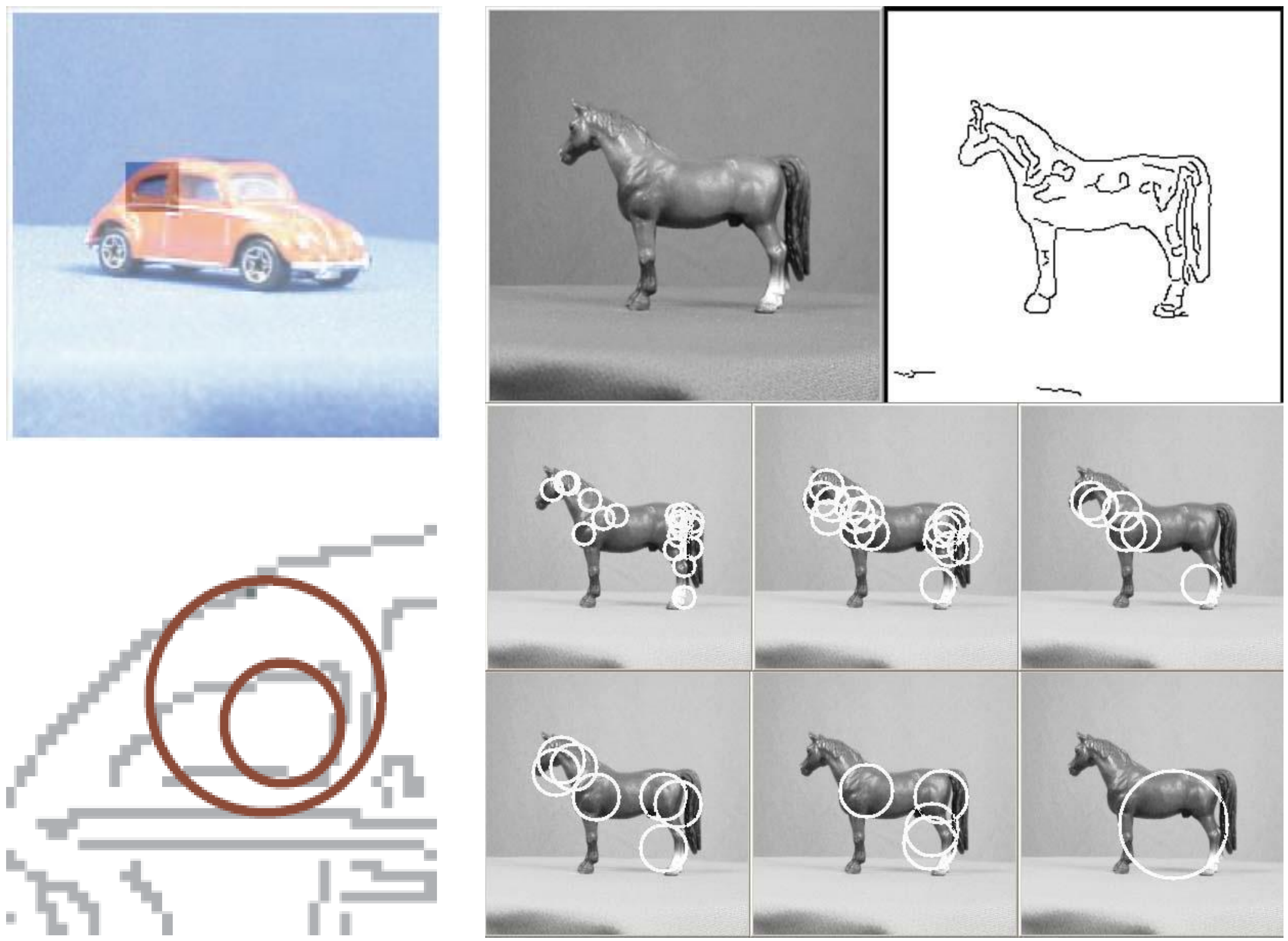
Easy to set parameters? 

Can you use Hough Transforms for other objects, beyond lines and circles?

Do you have to use edge detectors to vote in Hough Space?

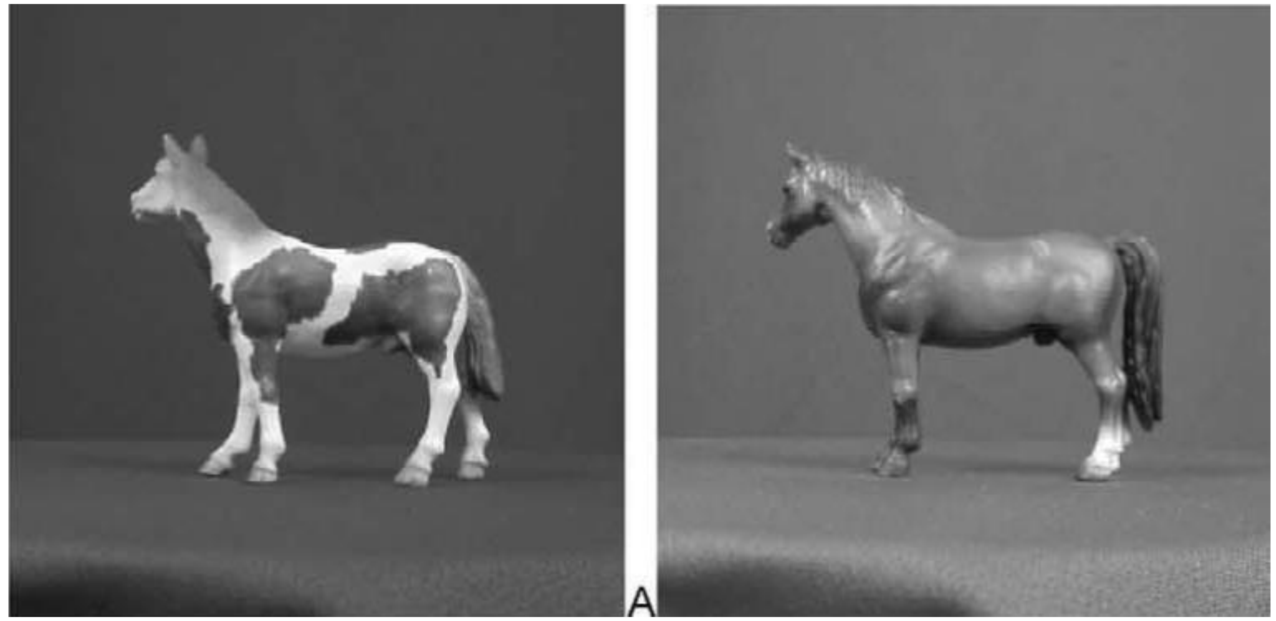
# Application of Hough transforms

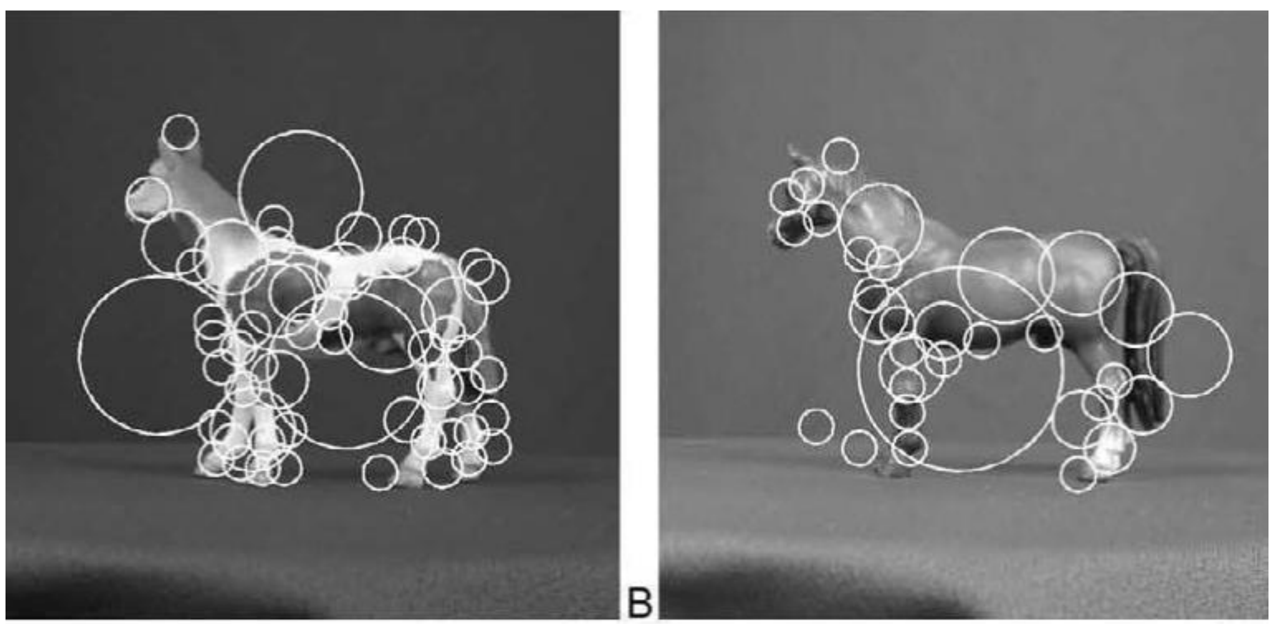
# Detecting shape features



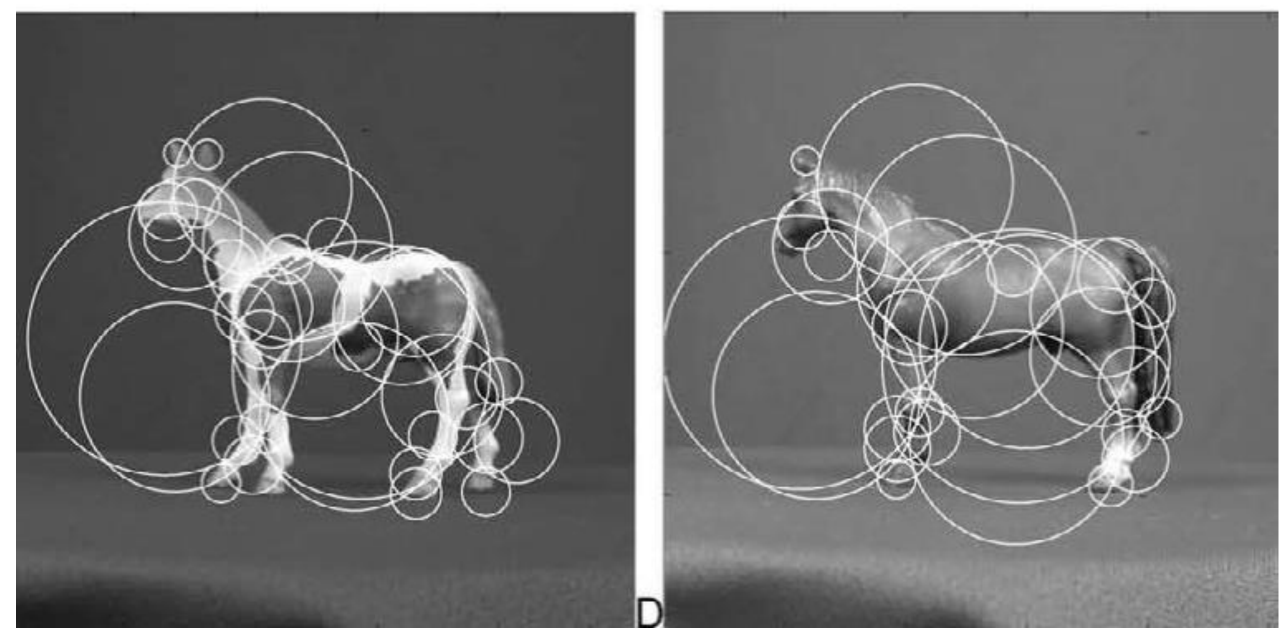
F. Jurie and C. Schmid, Scale-invariant shape features for recognition of object categories, CVPR 2004

Original images



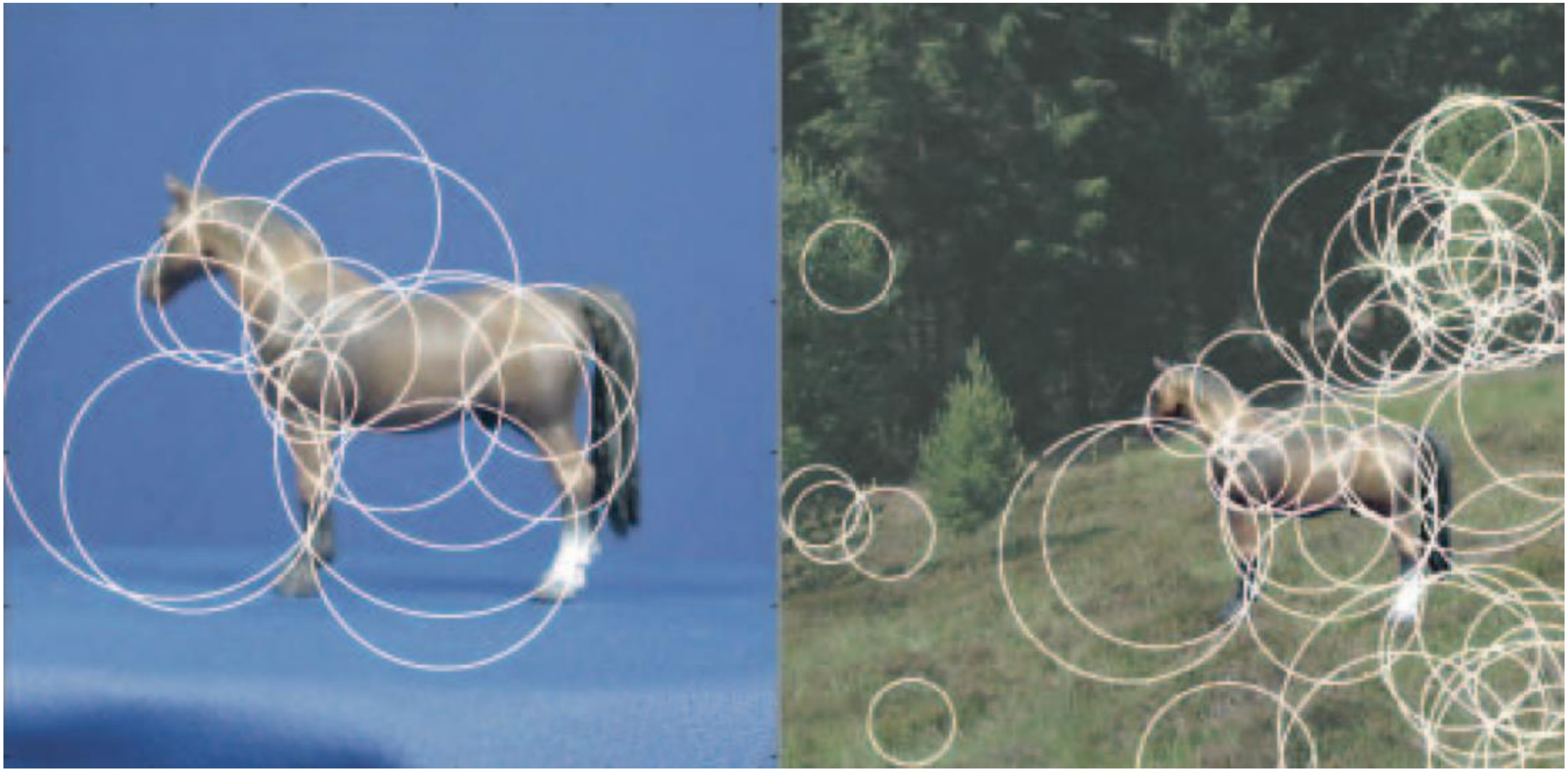




Laplacian circles




Hough-like circles

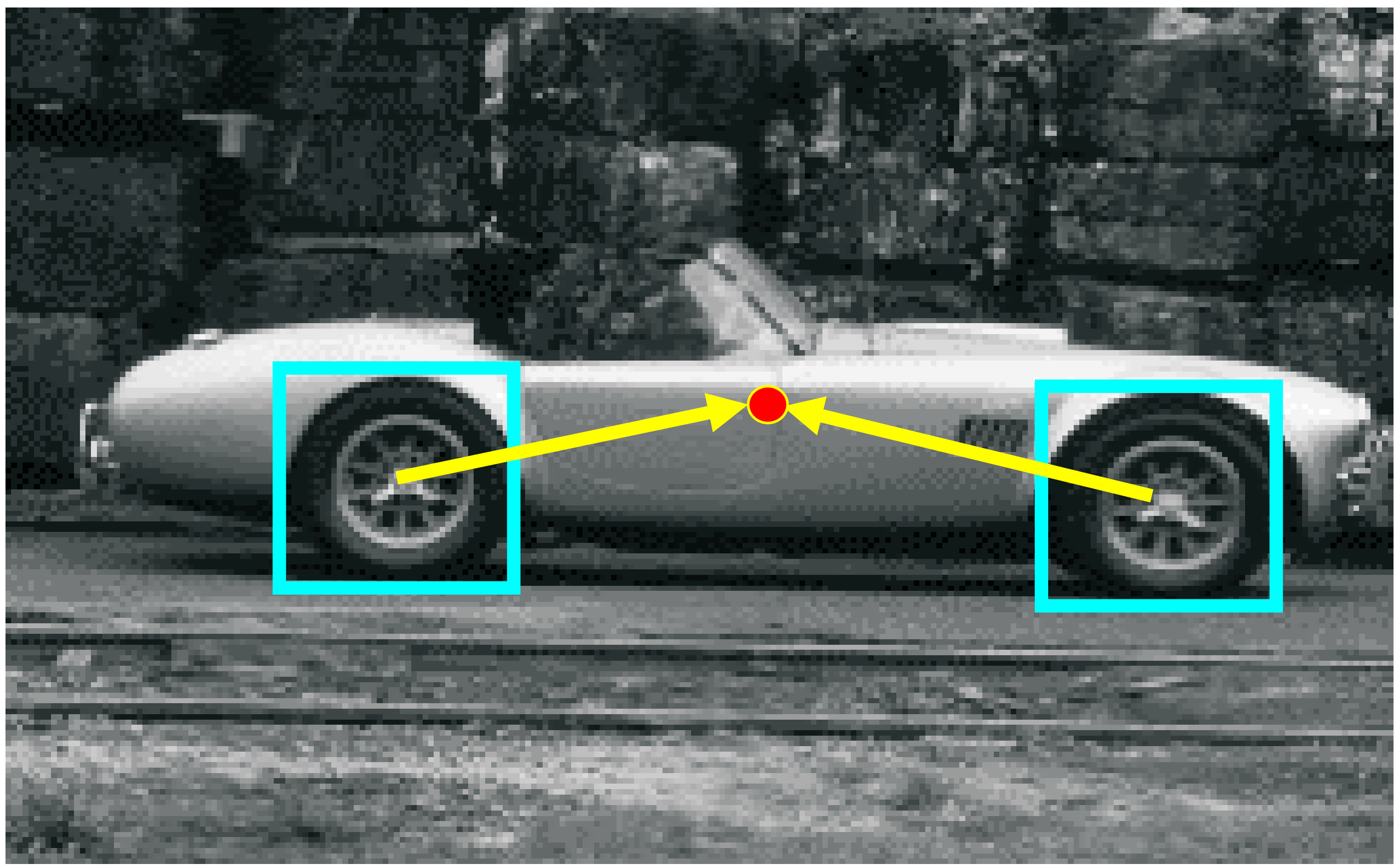
*Which feature detector is more consistent?*



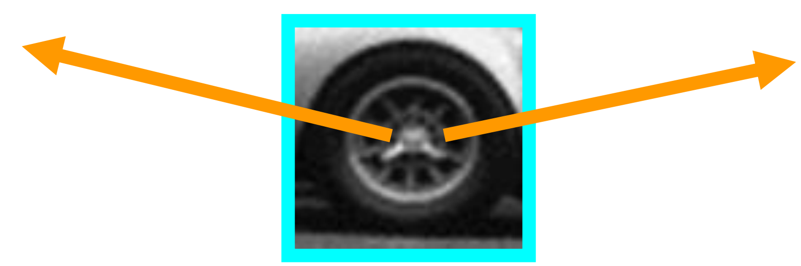
Robustness to scale and clutter

# Object detection

Index displacements by “visual codeword”



training image



visual codeword with
displacement vectors

B. Leibe, A. Leonardis, and B. Schiele, Combined Object Categorization and Segmentation with an Implicit Shape Model,
ECCV Workshop on Statistical Learning in Computer Vision 2004

# References

Basic reading:
- Szeliski textbook, Sections 4.2, 4.3.