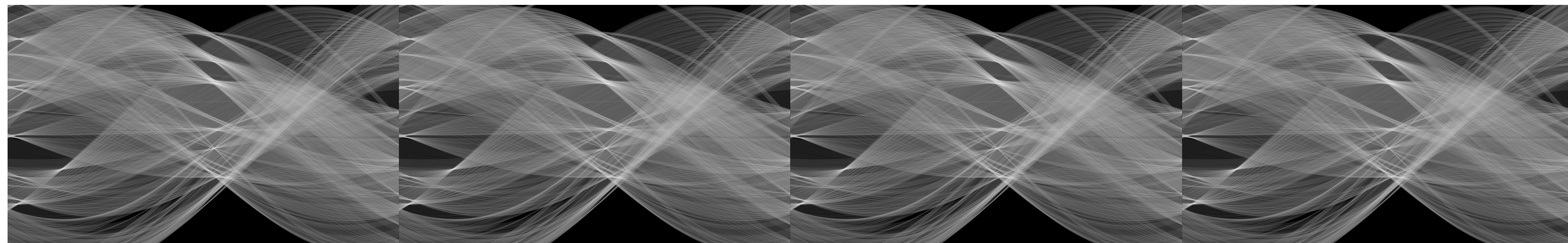




# CPSC 425: Computer Vision



**Image Credit:** Ioannis (Yannis) Gkioulekas (CMU)

**Lecture 15:** Hough Transform

# Menu for Today (October 30, 2024)

## Topics:

- **Hough** Transform
- Transformation Space Voting
- **Line** Detection

## Readings:

- **Today's** Lecture: Szeliski 7.4, Forsyth & Ponce 10.1

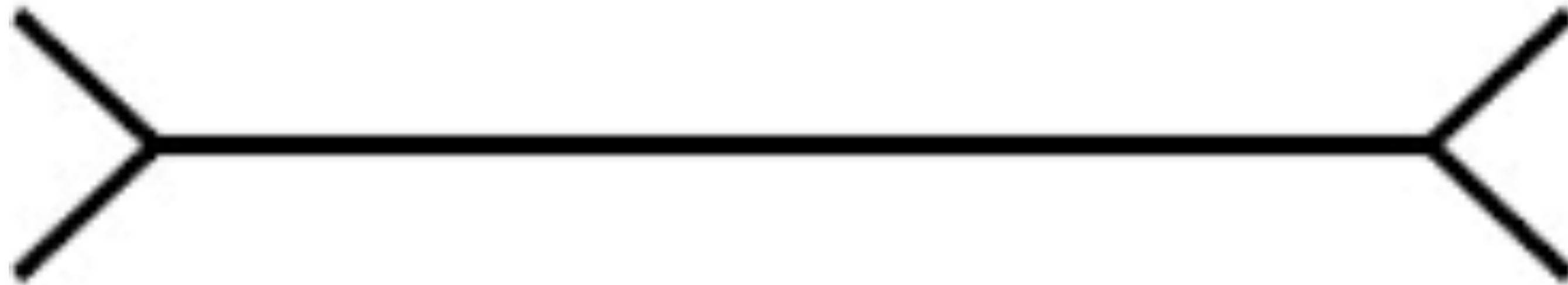
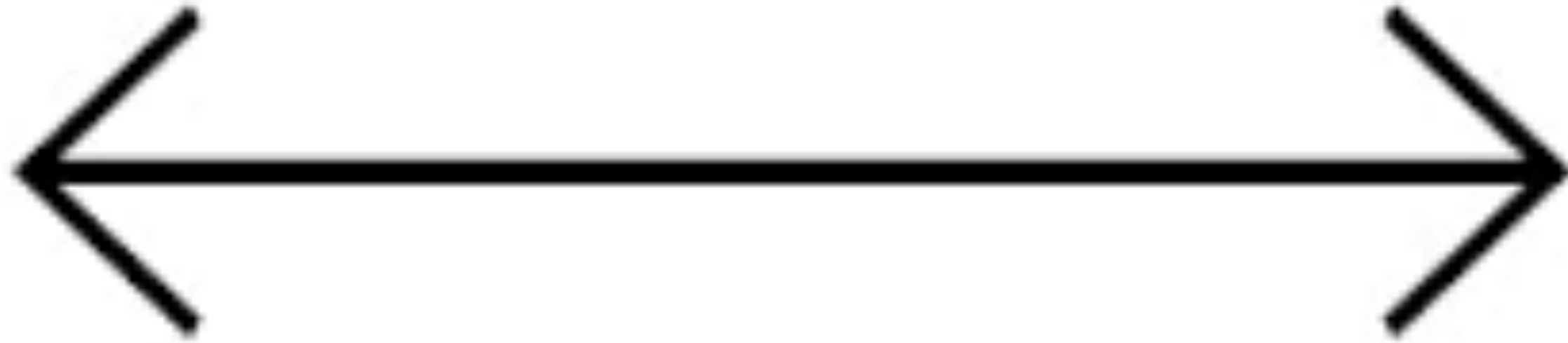
## Reminders:

- **Assignment 4:** RANSAC and Panorama Stitching
- **Midterm** grading is on the way (**Gradescope**)

# Learning **Goals**

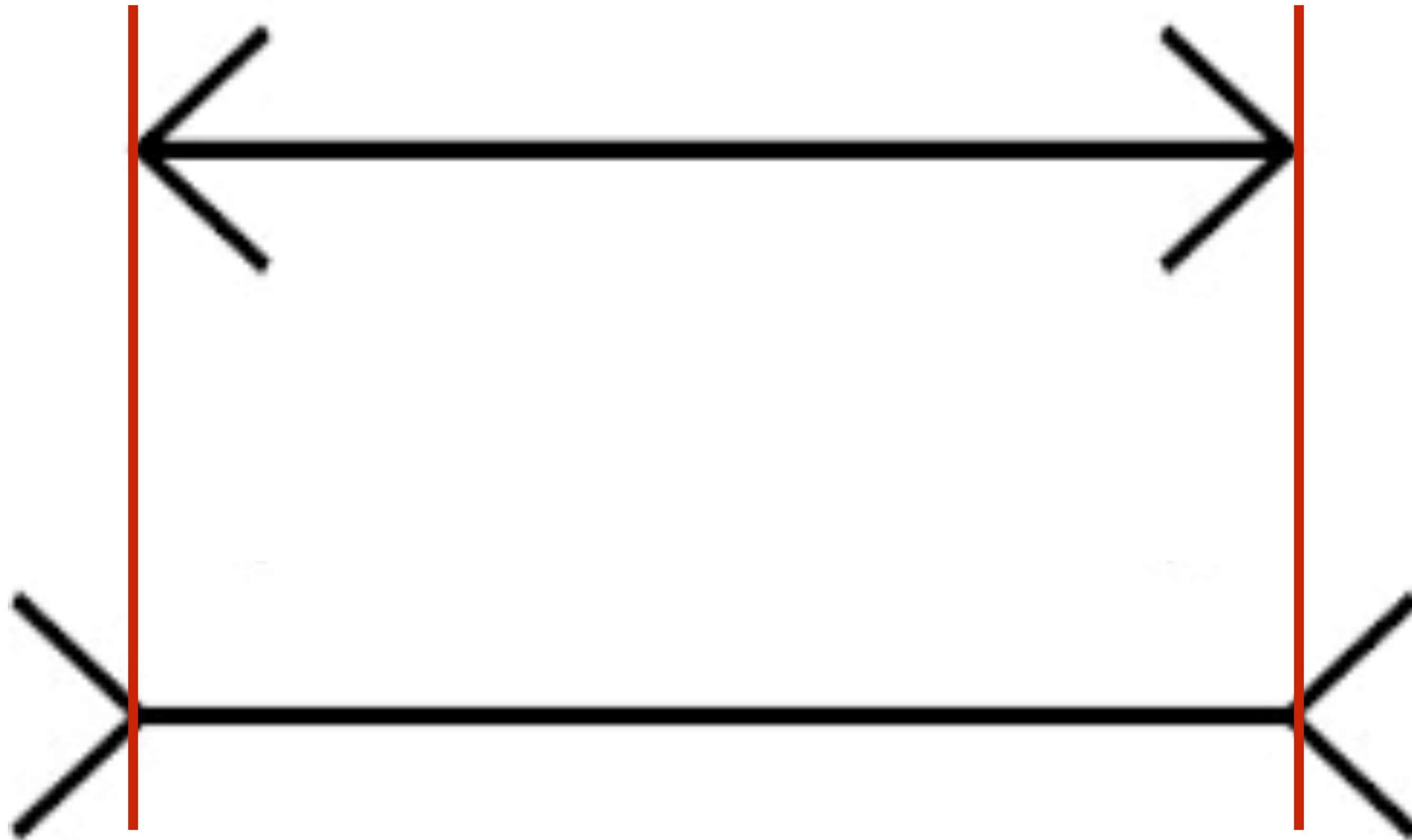
1. How to get **multiple** hypothesis
2. **Voting**-based strategies are useful

# Today's **“fun”** Example: Müller-Lyer Illusion

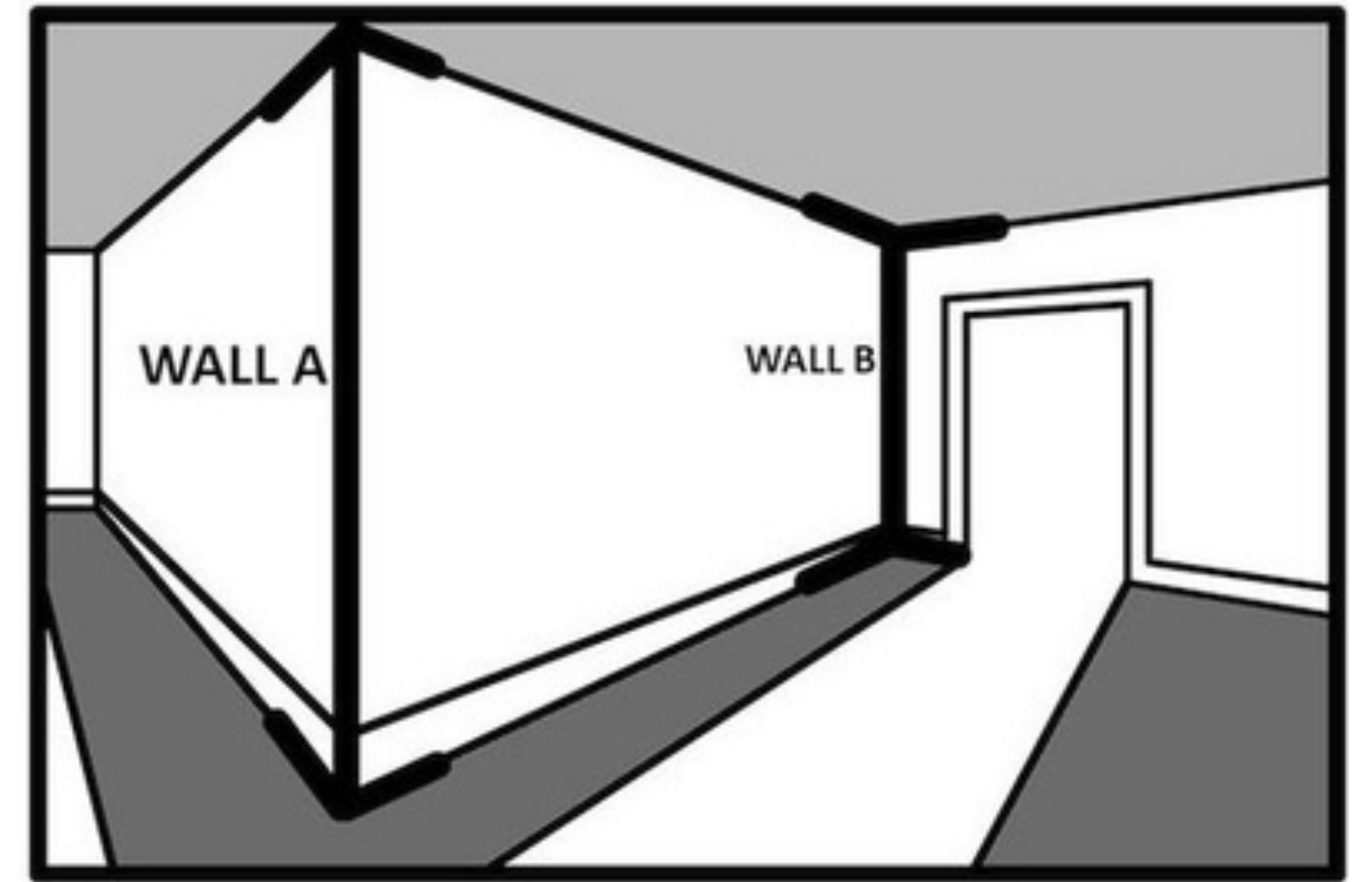
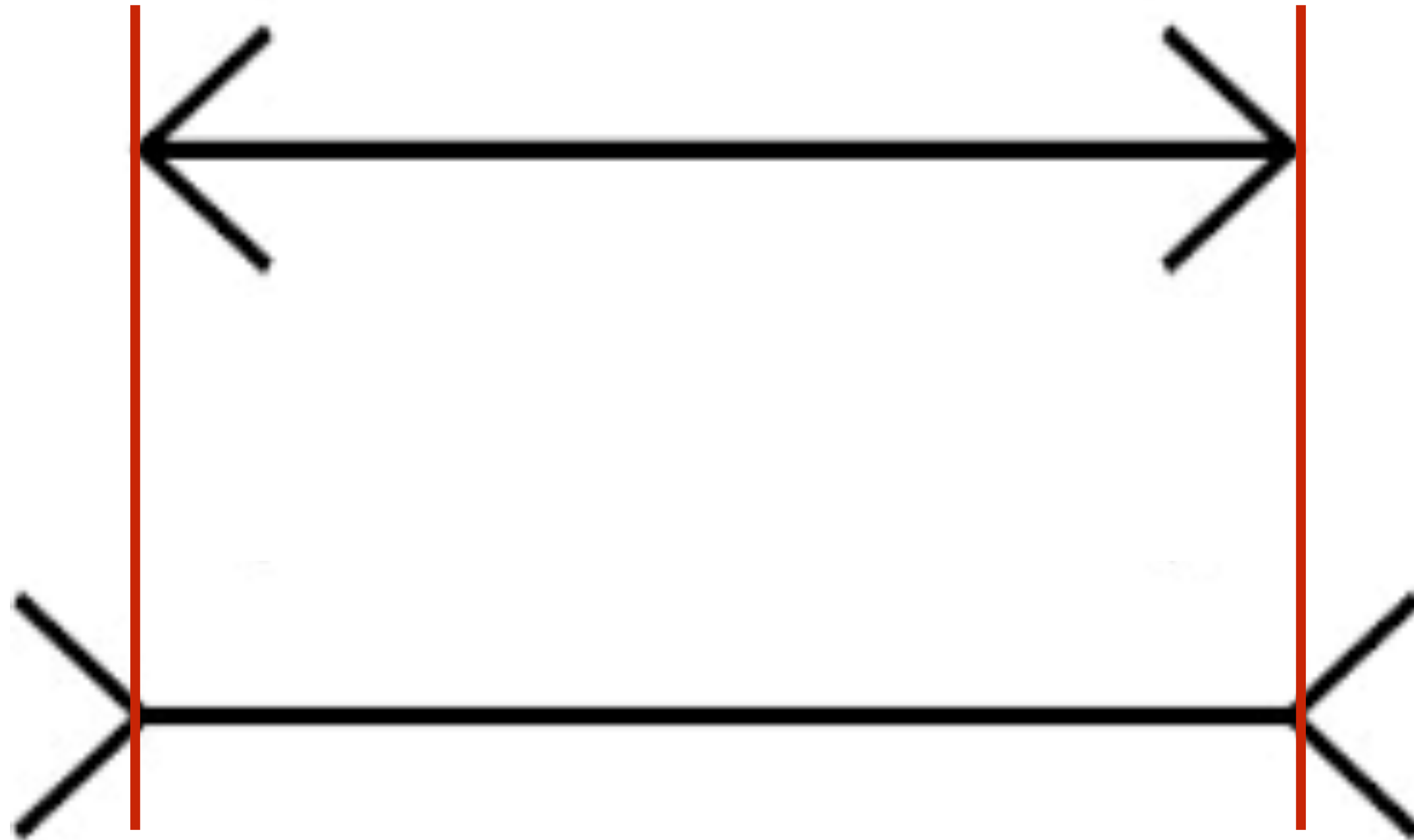




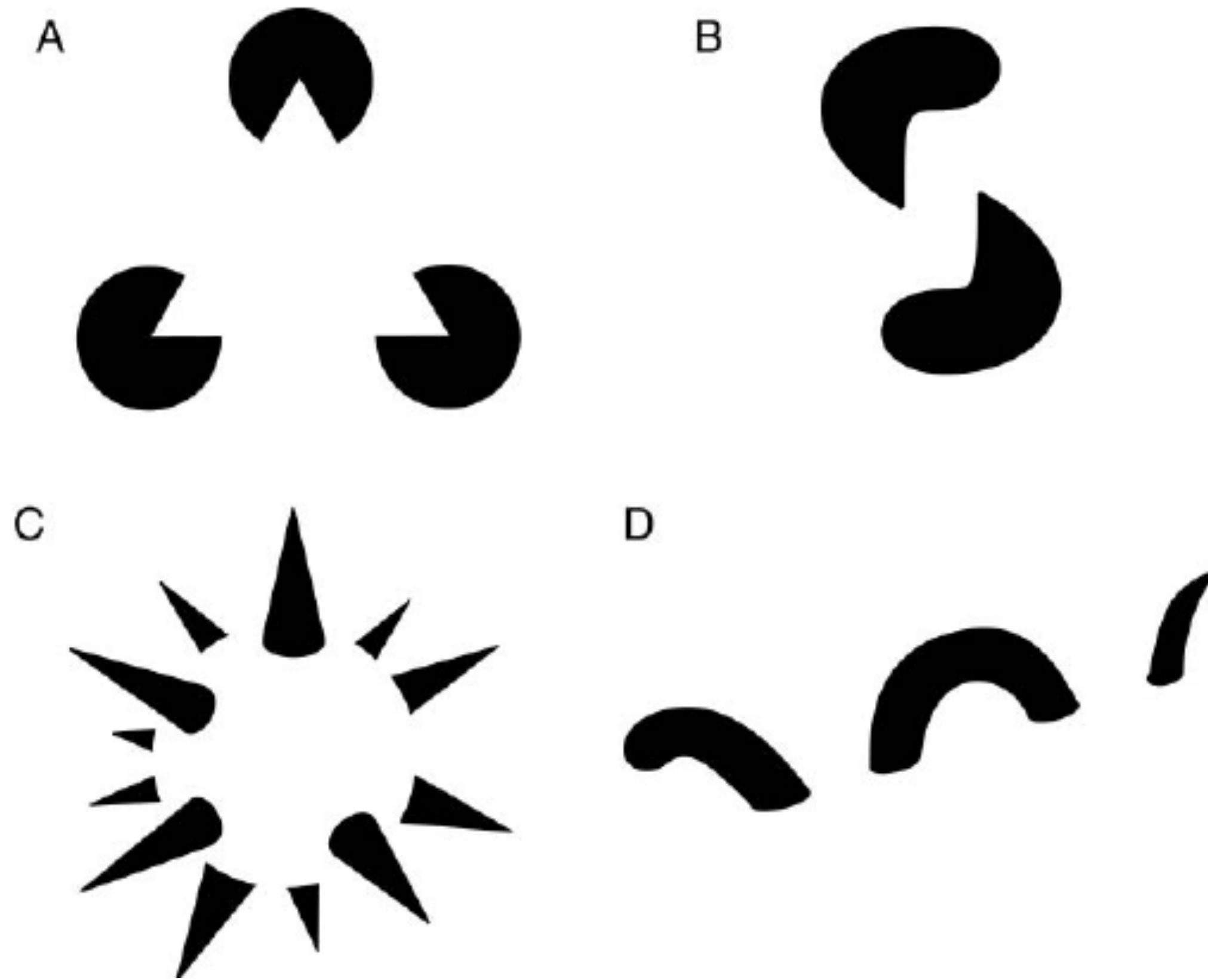
# Today's **“fun”** Example: Müller-Lyer Illusion



# Today's "fun" Example: Müller-Lyer Illusion

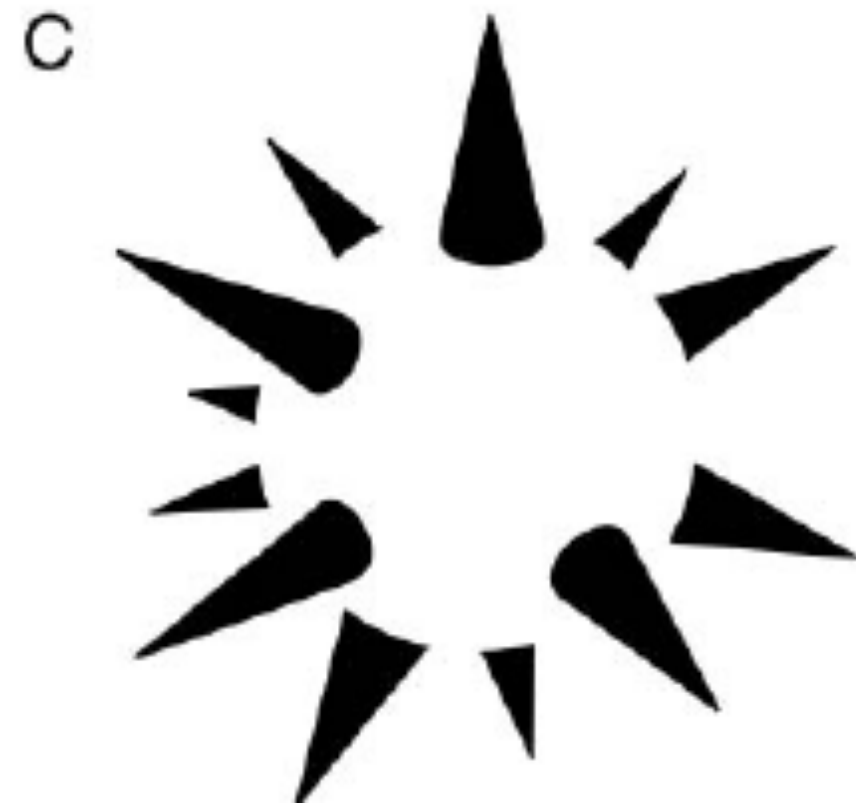
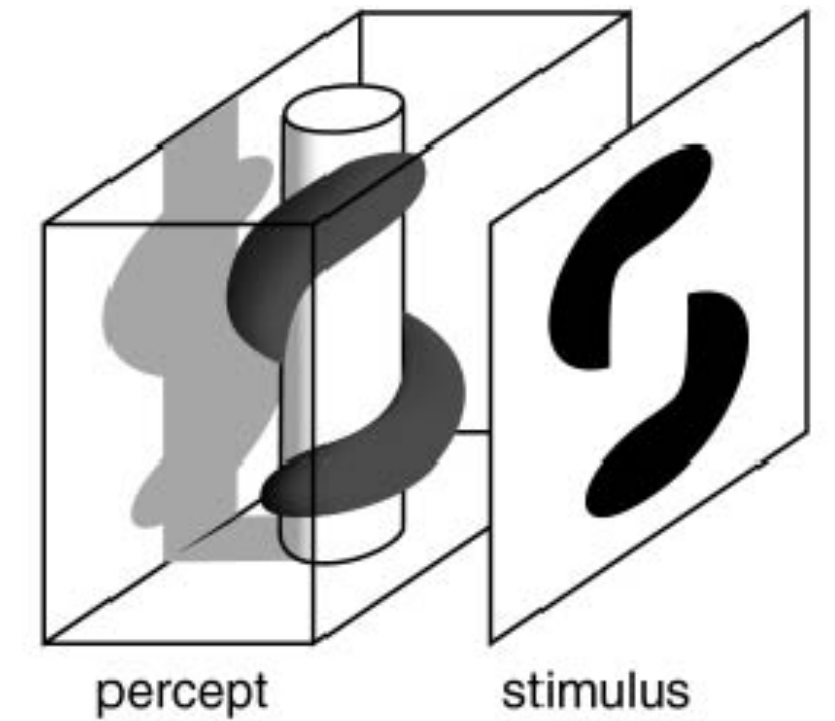
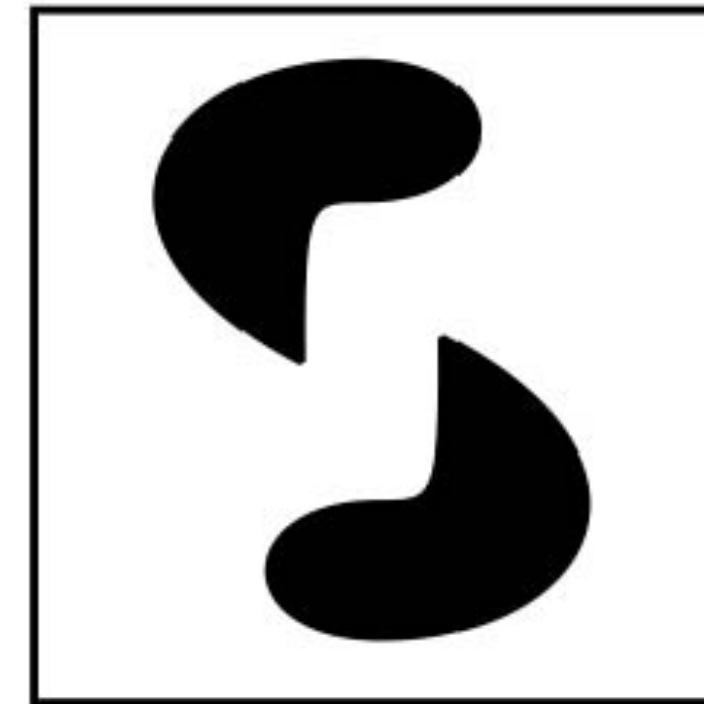
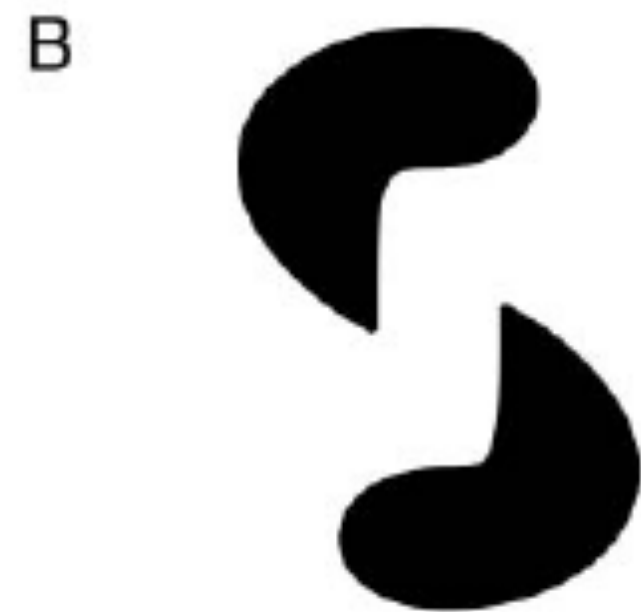
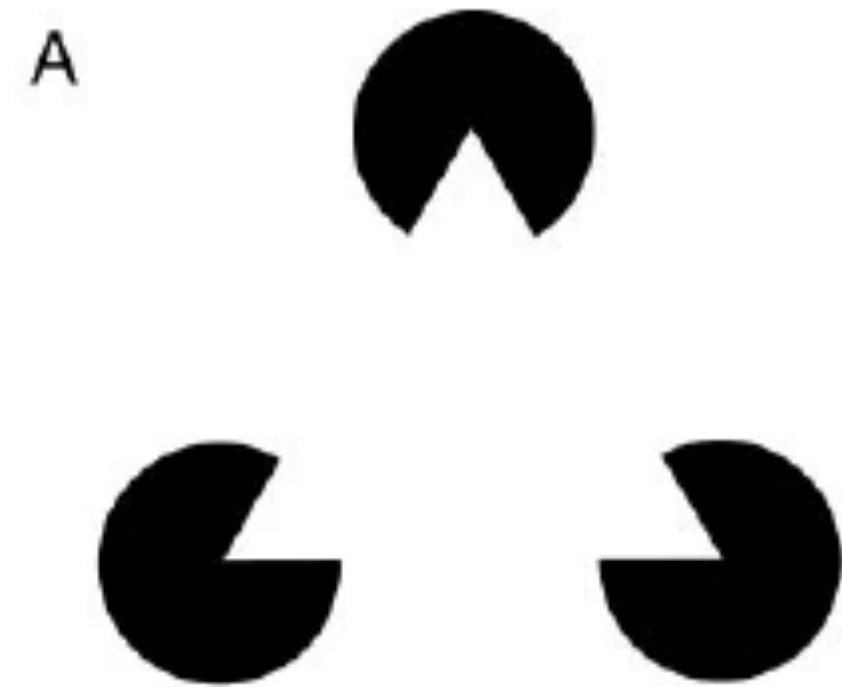


# Today's “fun” Example: Tse's Volumetric Illusions



- A.** Kanizsa triangle
- B.** Tse's volumetric worm
- C.** Idesawa's spiky sphere
- D.** Tse's "sea monster"

# Today's "fun" Example: Tse's Volumetric Illusions

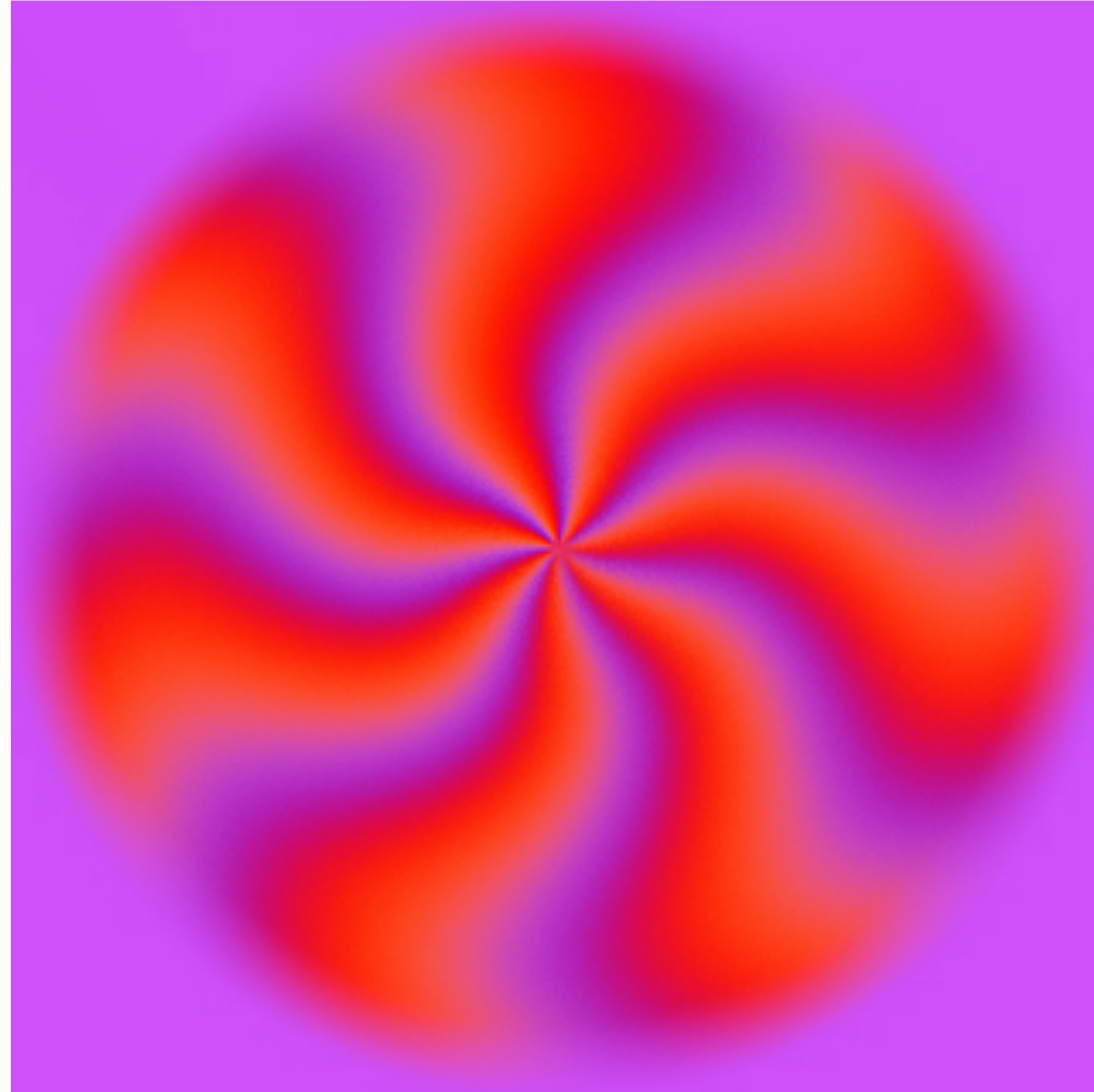


**Figure credit:** Steve Lehar

Today's **“fun”** Example: FedEx

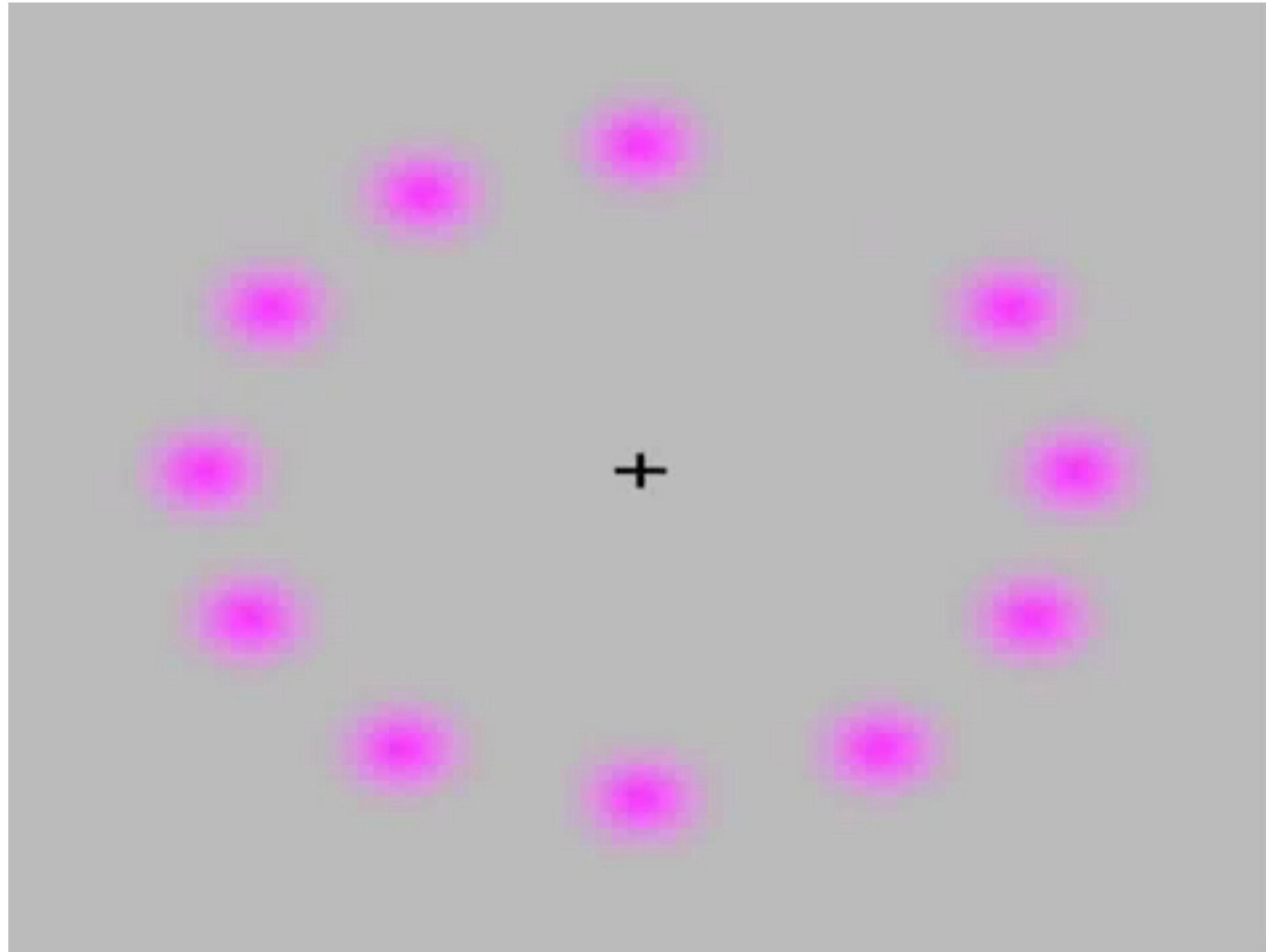
The FedEx logo is displayed in a large, bold, sans-serif font. The word "Fed" is in a dark purple color, and the word "Ex" is in a bright orange color. The letters are closely spaced and have a clean, modern appearance.

# Today's “**fun**” Example: Motion Illusion

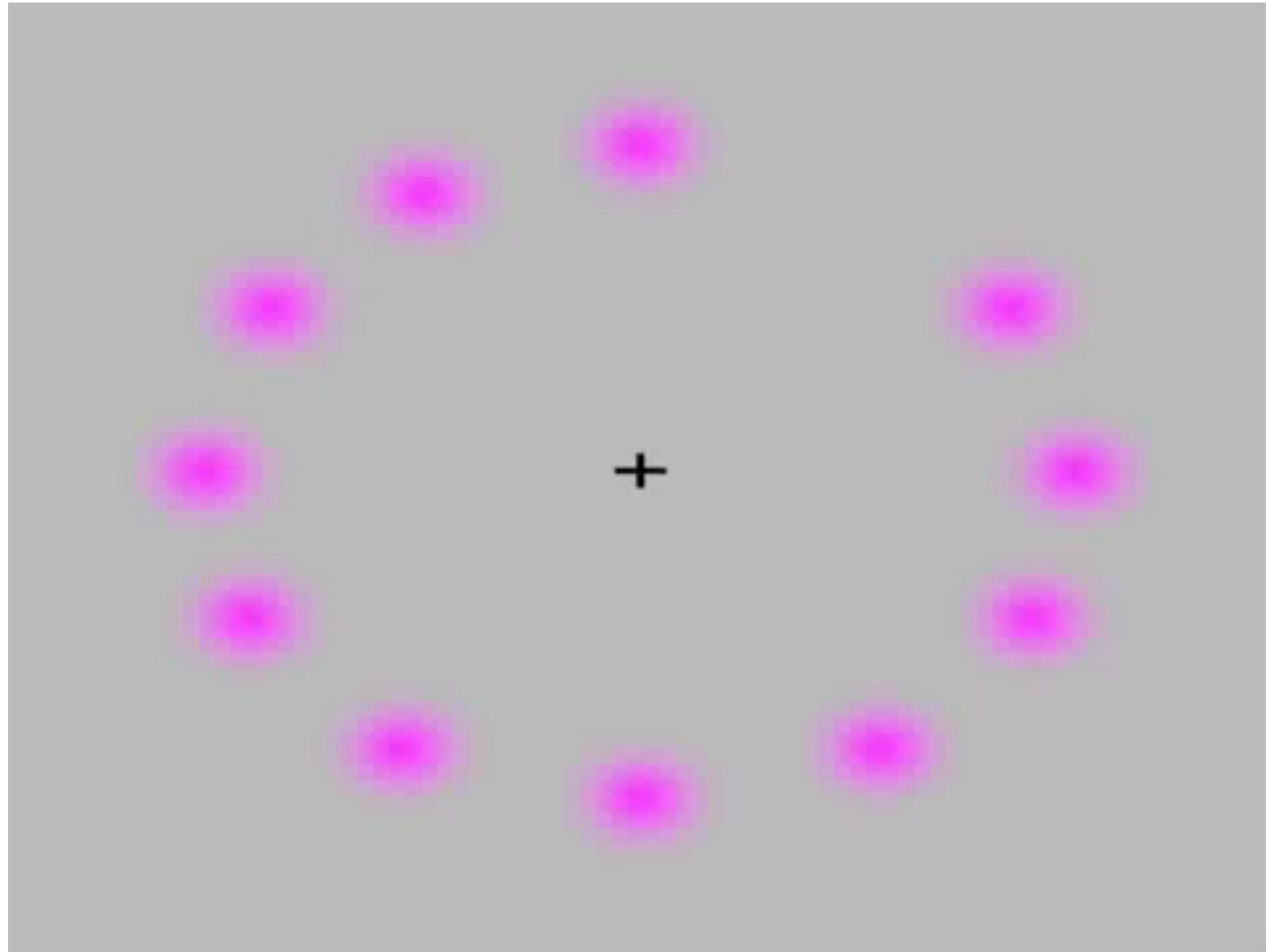




# Today's “**fun**” Example: Lilac Chaser (a.k.a. Pac-Man) Illusion



# Today's “**fun**” Example: Lilac Chaser (a.k.a. Pac-Man) Illusion





# Image **Alignment**

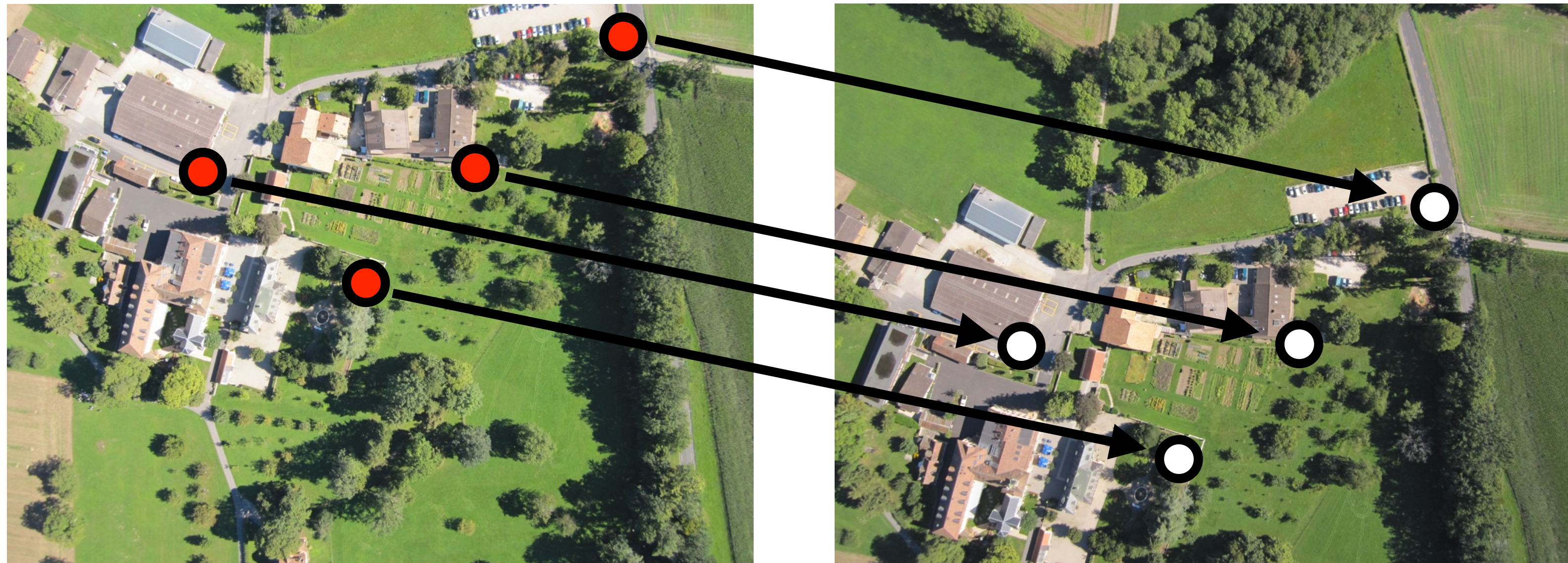
**Aim:** Warp one image to align with another using a 2D transformation





# Image Alignment

**Step 1:** Find correspondences (matching points) across two images



$$\mathbf{u} = \mathbf{H}\mathbf{x}$$

2 points for Similarity

3 for Affine

4 for Homography



# Image Alignment

**Step 2:** Compute the transformation to align the two images



# RANSAC (RANDOM SAMPLE CONSENSUS)

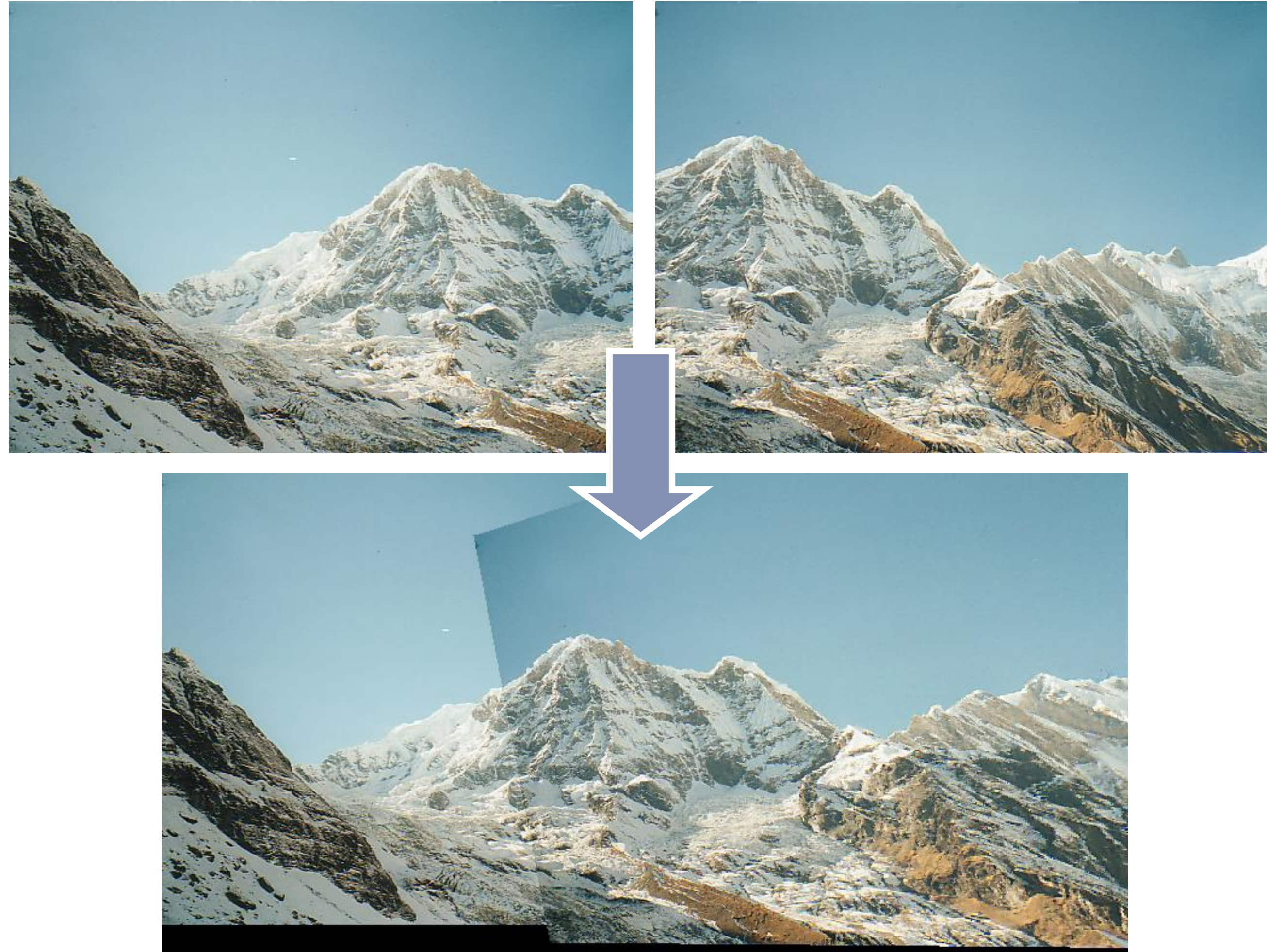
1. Randomly choose minimal subset of data points necessary to fit model (a **sample**)
2. Points within some distance threshold,  $t$ , of model are a **consensus set**.  
Size of consensus set is model's **support**
3. Repeat for  $N$  samples; model with biggest support is most robust fit
  - Points within distance  $t$  of best model are inliers
  - Fit final model to all inliers

RANSAC is very useful for variety of applications



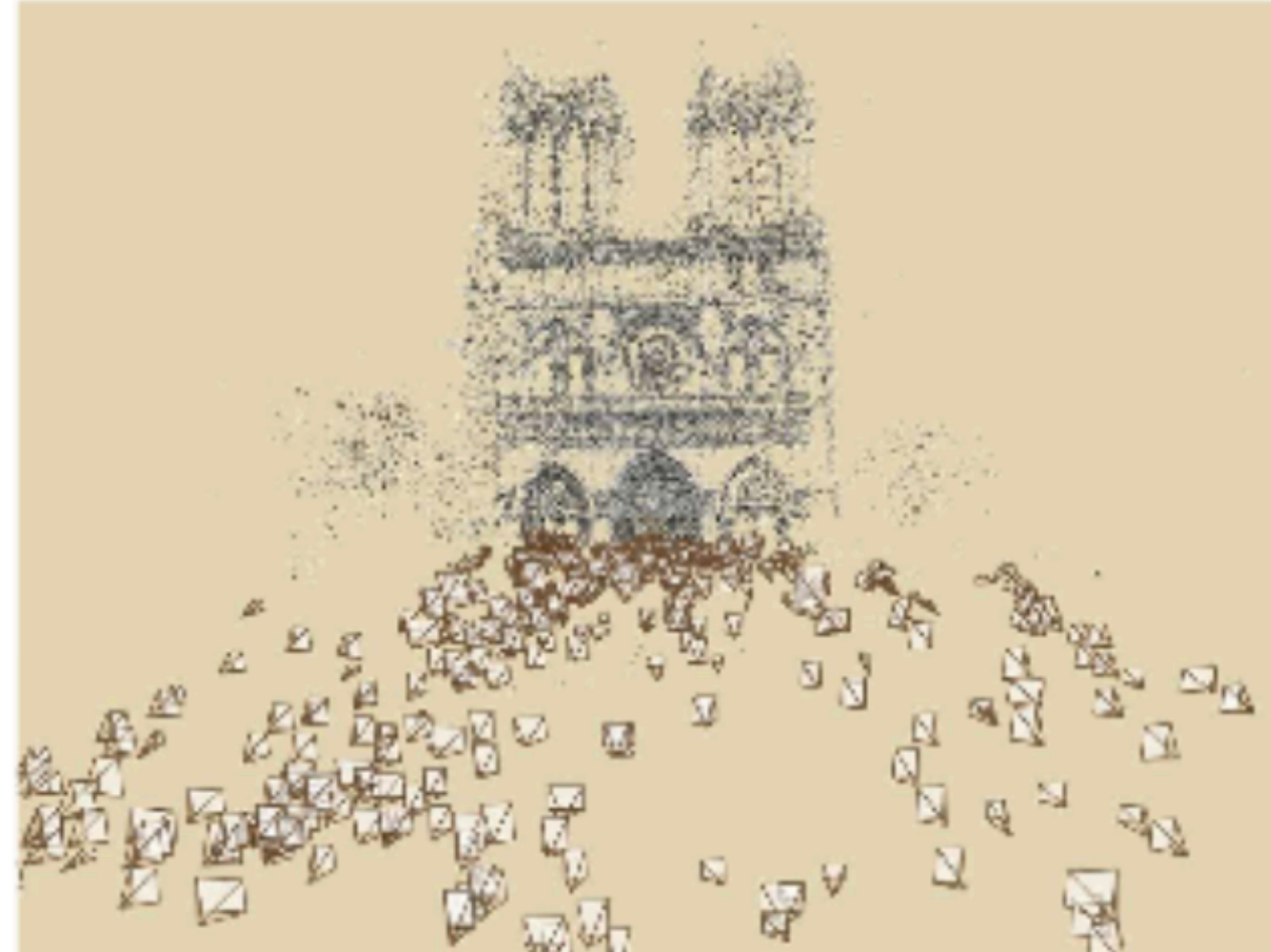
# 2-view **Rotation** Estimation

Final rotation estimation





# Example: Photo Tourism



**Figure credit:** Snavely et al. 2006

Takes as input unstructured collections of photographs and reconstructs each photo's viewpoint and a sparse 3D model of the scene

Uses both SIFT and RANSAC

# Example: Photo Tourism



[ Agarwal, Furukawa, Snavely, Curless, Seitz, Szeliski, 2010 ]



# Example: Photo Tourism



[ Agarwal, Furukawa, Snavely, Curless, Seitz, Szeliski, 2010 ]

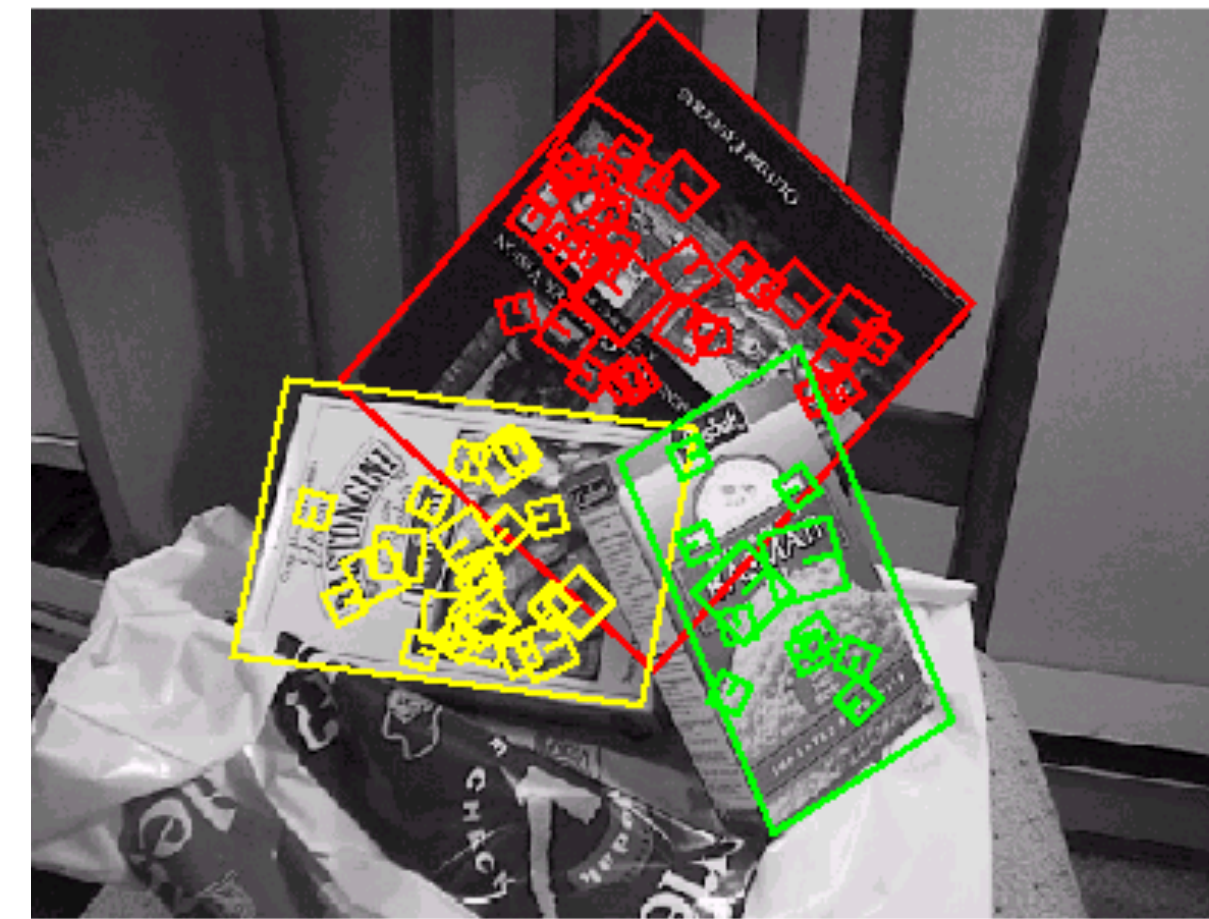


# Object Instance Recognition

Database of planar objects



Instance recognition





# Discussion of RANSAC

## Advantages:

- General method suited for a wide range of model fitting problems
- Easy to implement and easy to calculate its failure rate

## Disadvantages:

- Only handles a moderate percentage of outliers without cost blowing up
- Many real problems have high rate of outliers (but sometimes selective choice of random subsets can help)
- Hard to deal with multiple solutions (e.g., object detection with many objects)

The **Hough transform** can handle high percentage of outliers

# Learning **Goals**

1. How to get **multiple** hypothesis
2. **Voting**-based strategies are useful

# Hough Transform: Motivation

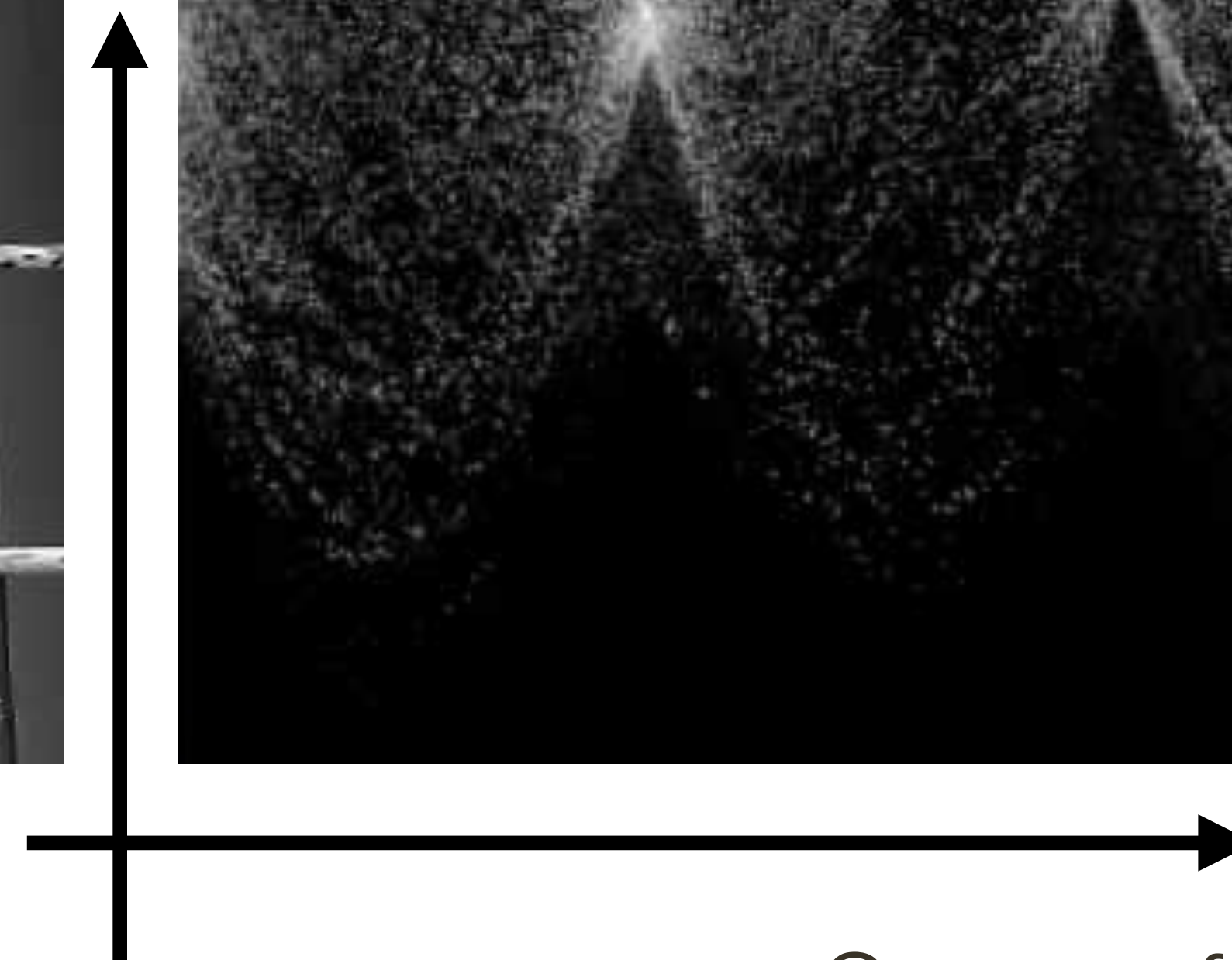
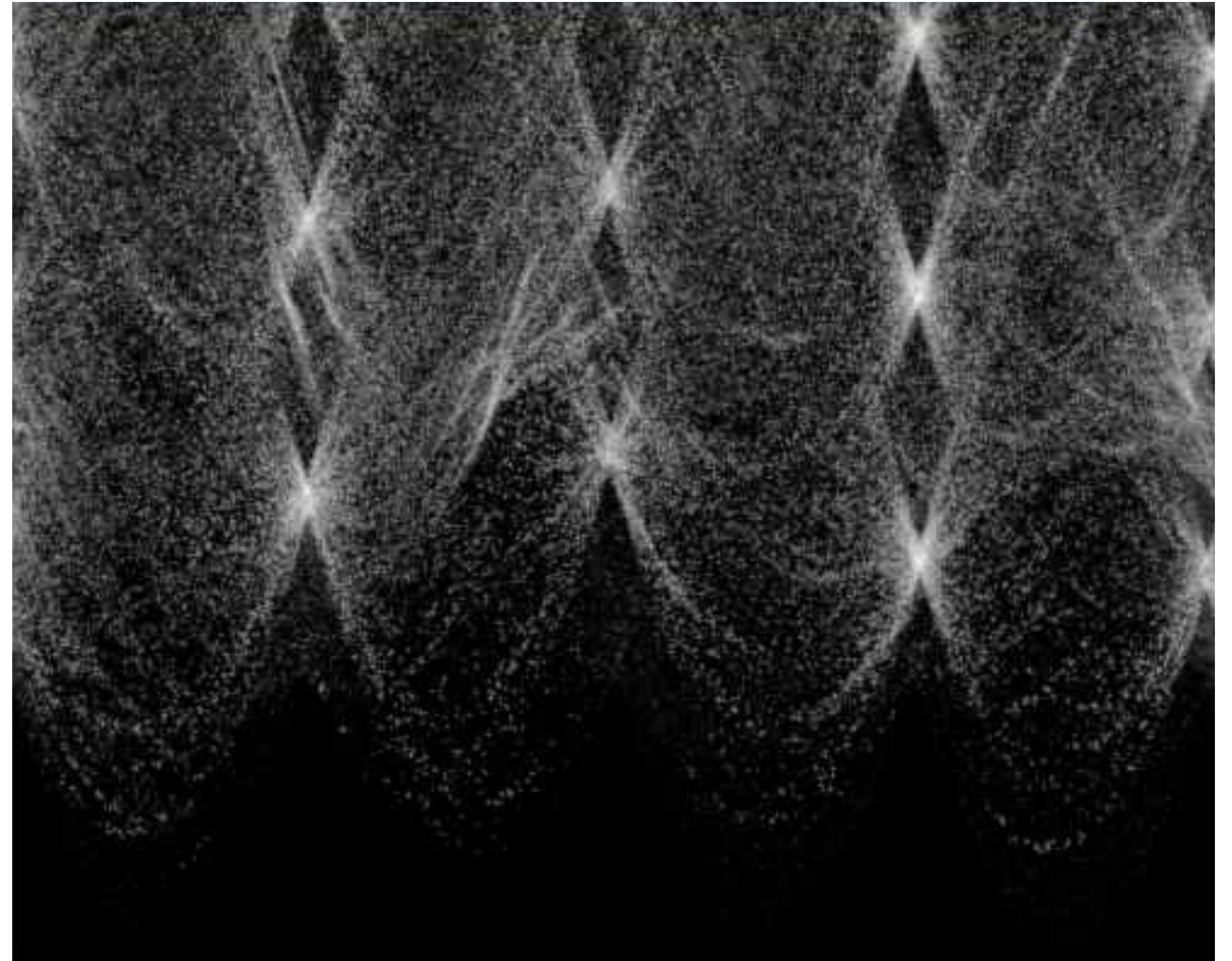


How to find lines in this image?



# Hough Transform: Motivation

Votes / Probability Distribution



Space of 2D Image Lines



# Hough Transform

Idea of **Hough transform**:

- For each token / data point vote for all models to which it could belong
- Return models that get many votes / distribution of possible models

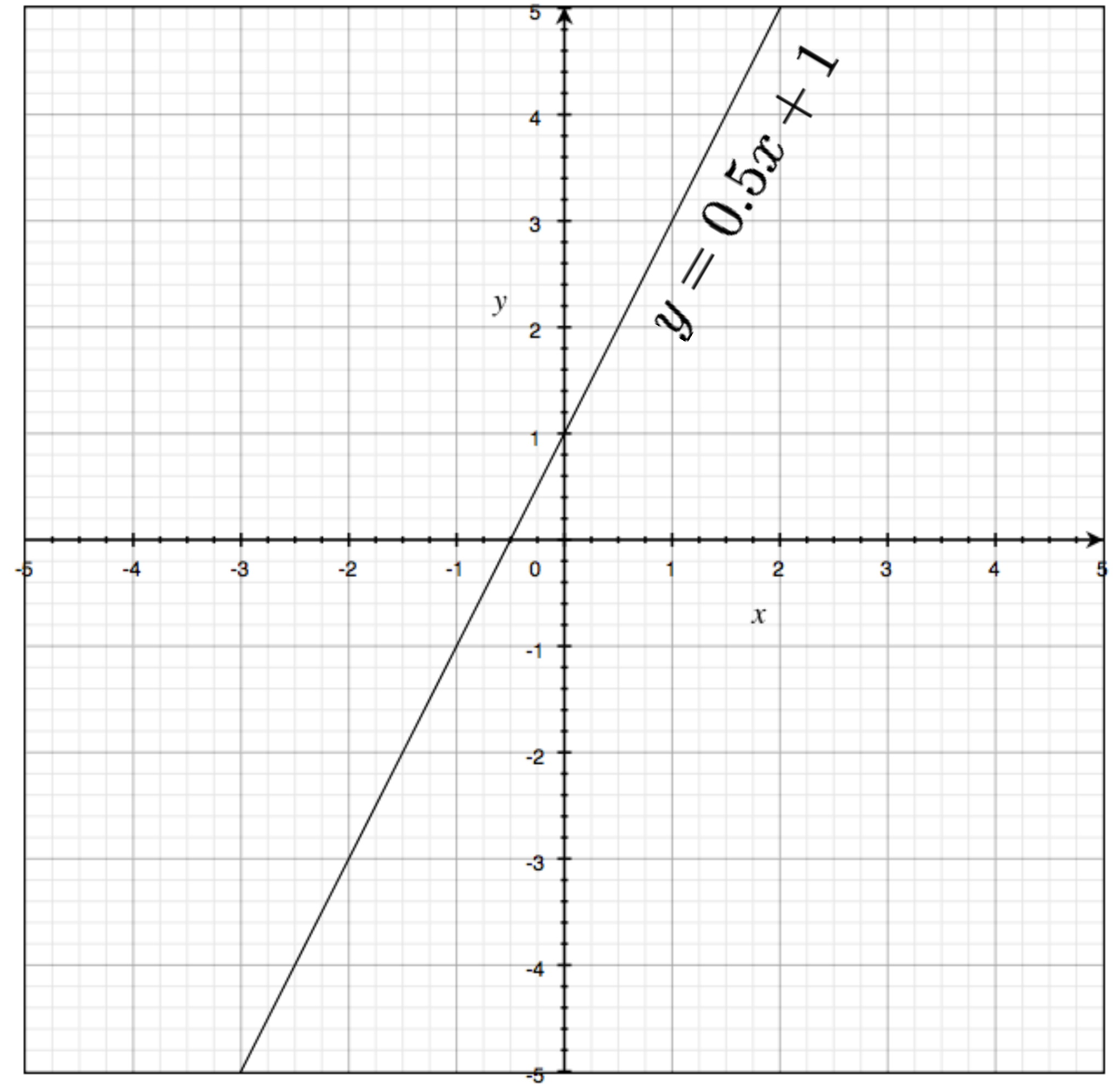
**Example:** For each point, vote for all lines that could pass through it; the true lines will pass through many points and so receive many votes

c.f. **RANSAC** which optimizes a **single hypothesis** by maximizing the number of inliers (though modifications exist to find multiple instances of a model)

# Lines: Slope intercept form

$$y = mx + b$$

↑ slope      ↑ y-intercept



# Hough Transform: Image and Parameter Space

variables

$$y = mx + b$$

parameters

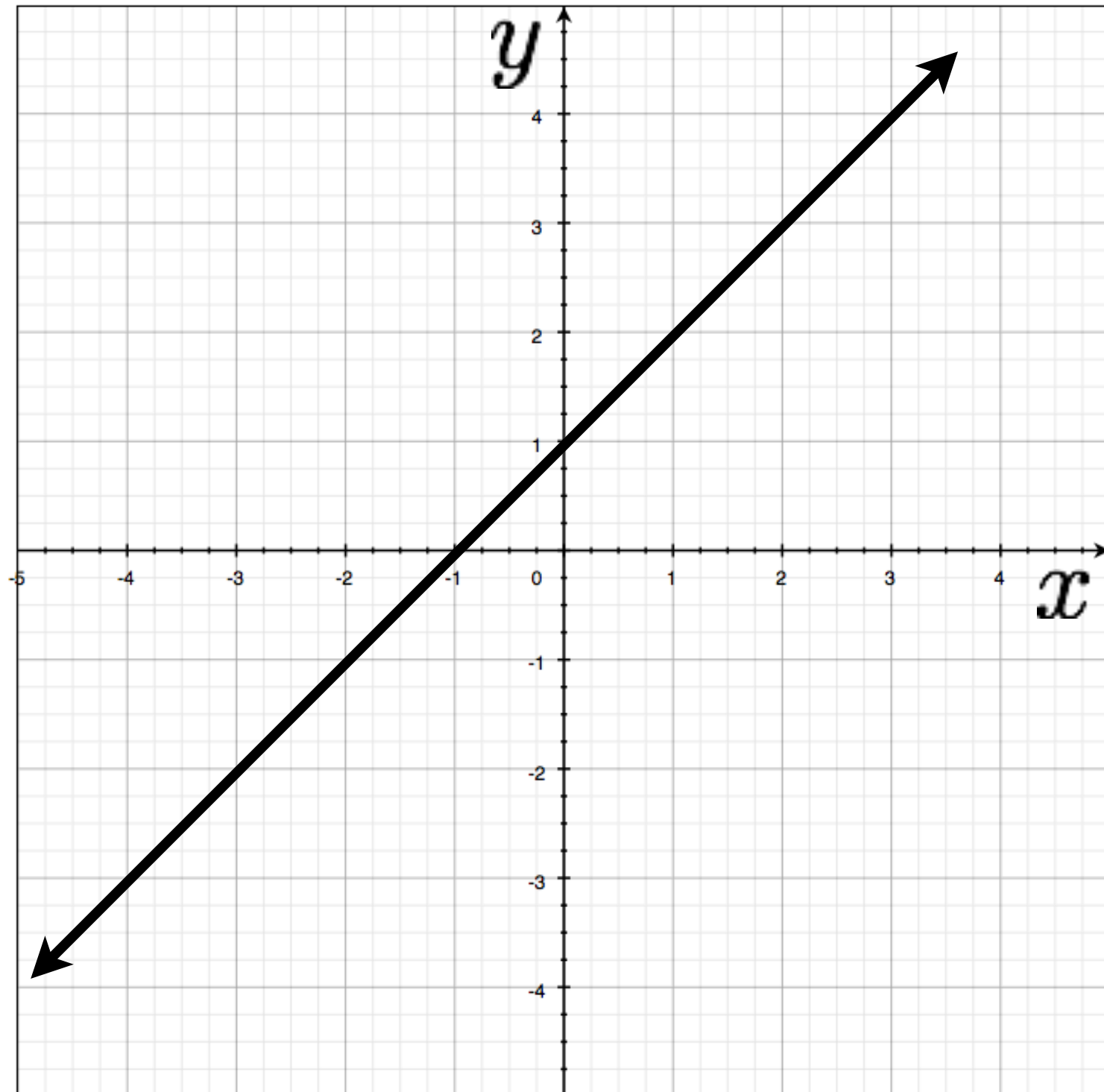


Image space



# Hough Transform: Image and Parameter Space

variables

$$y = mx + b$$

parameters

variables

$$y - mx = b$$

parameters

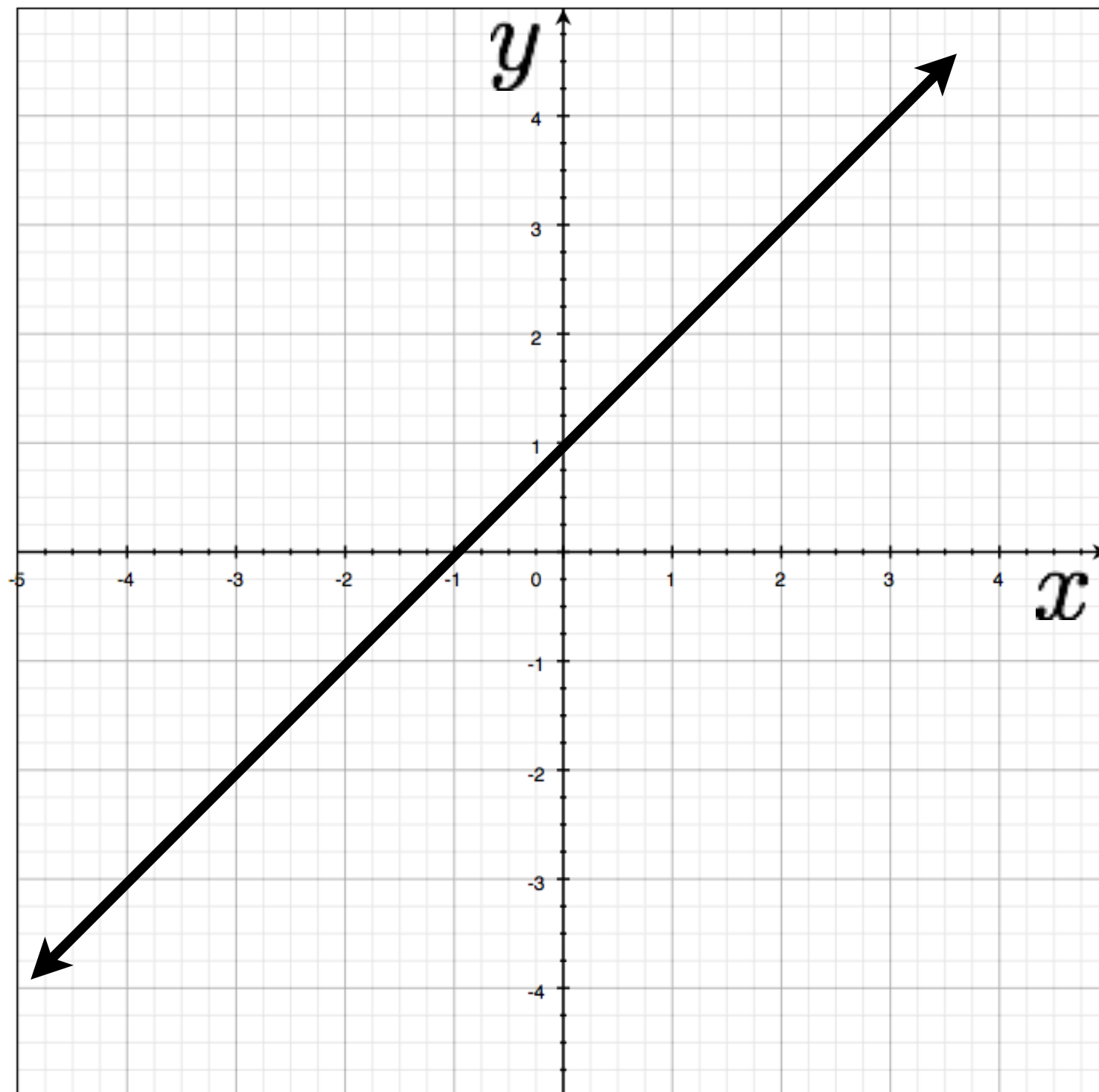
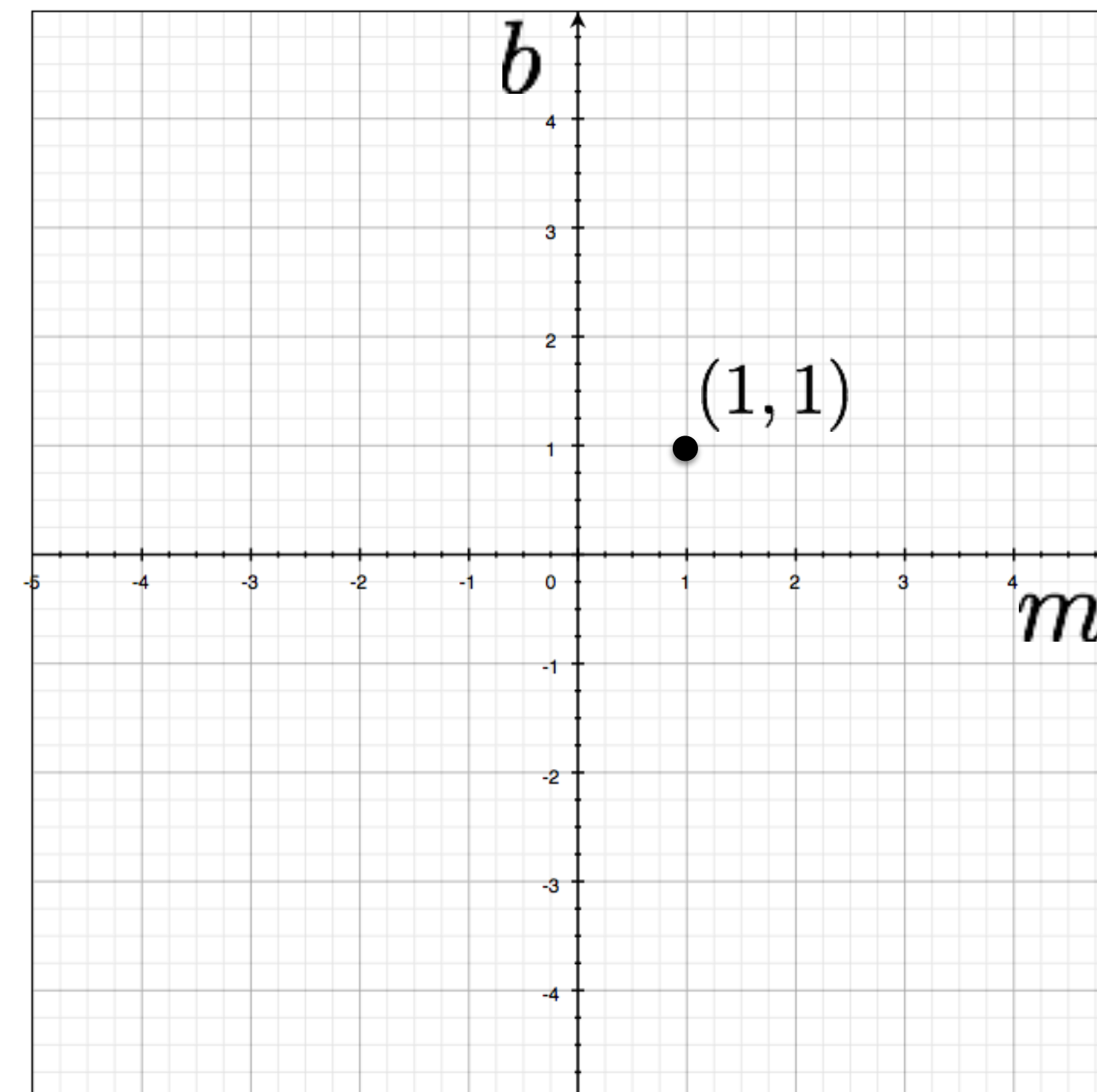


Image space

a line  
becomes a  
point



Parameter space

# Hough Transform: Image and Parameter Space

variables

$$y = mx + b$$

parameters

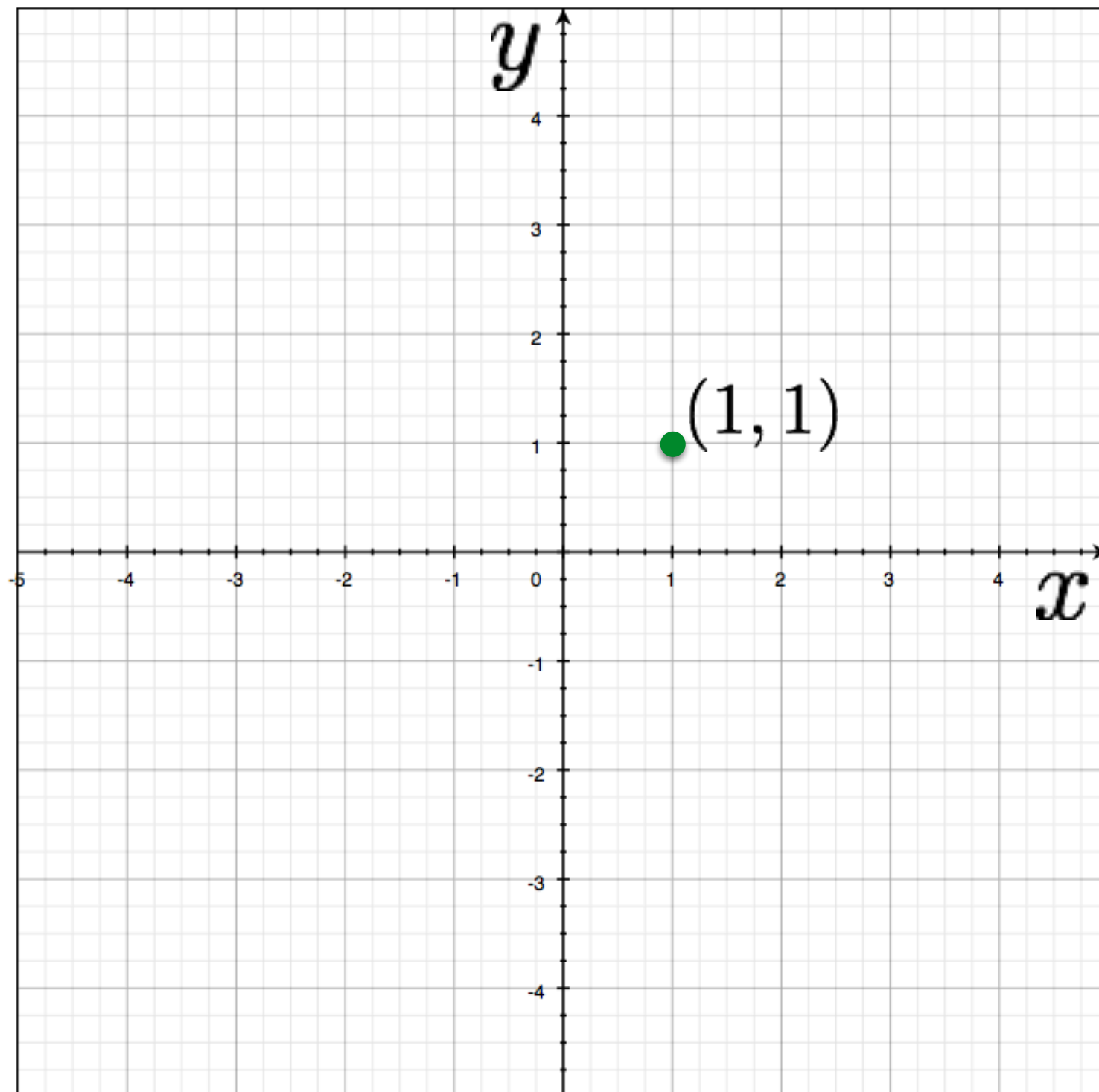


Image space

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

# Hough Transform: Lines

variables

$$y = mx + b$$

parameters

variables

$$y - mx = b$$

parameters

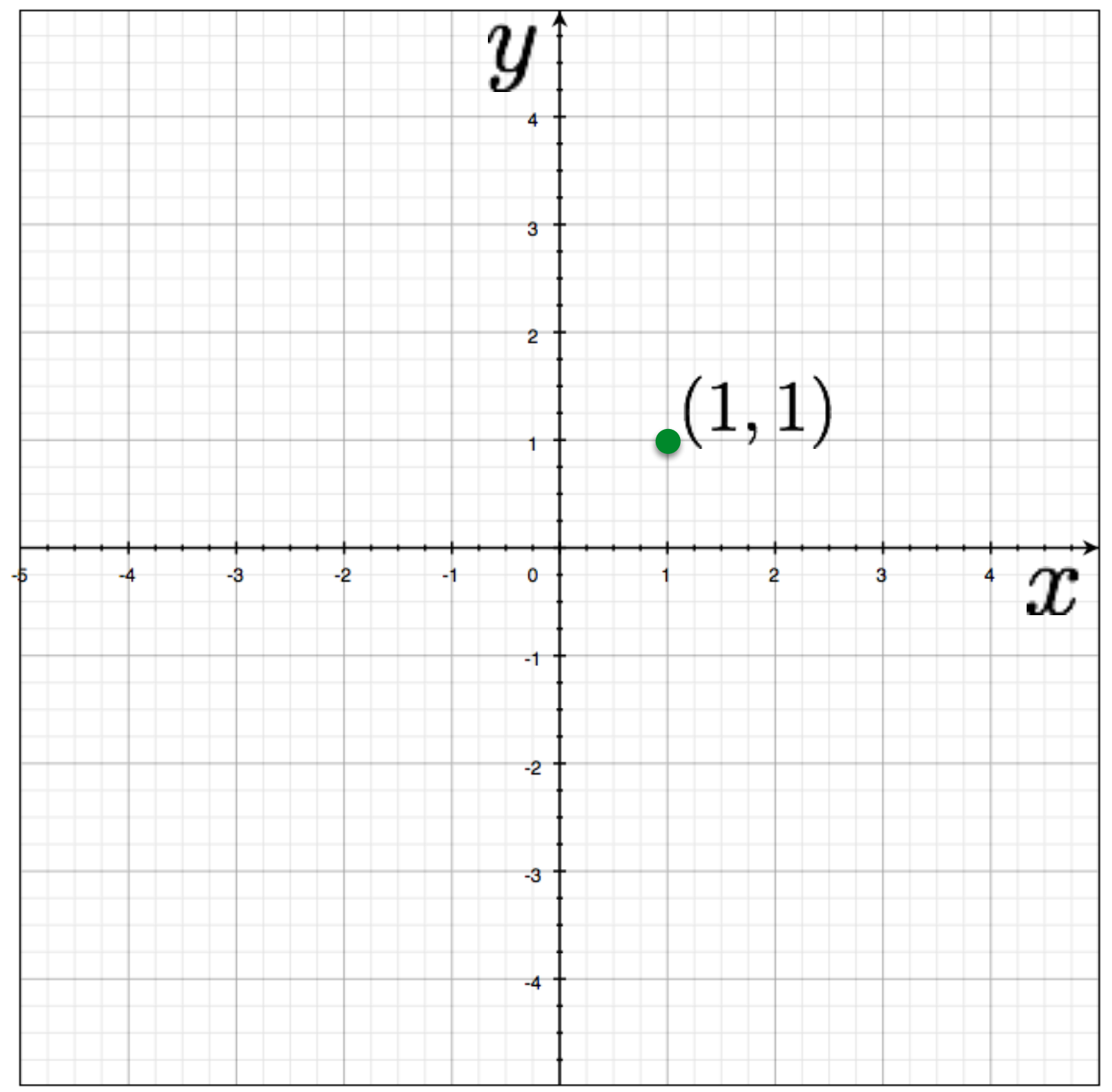
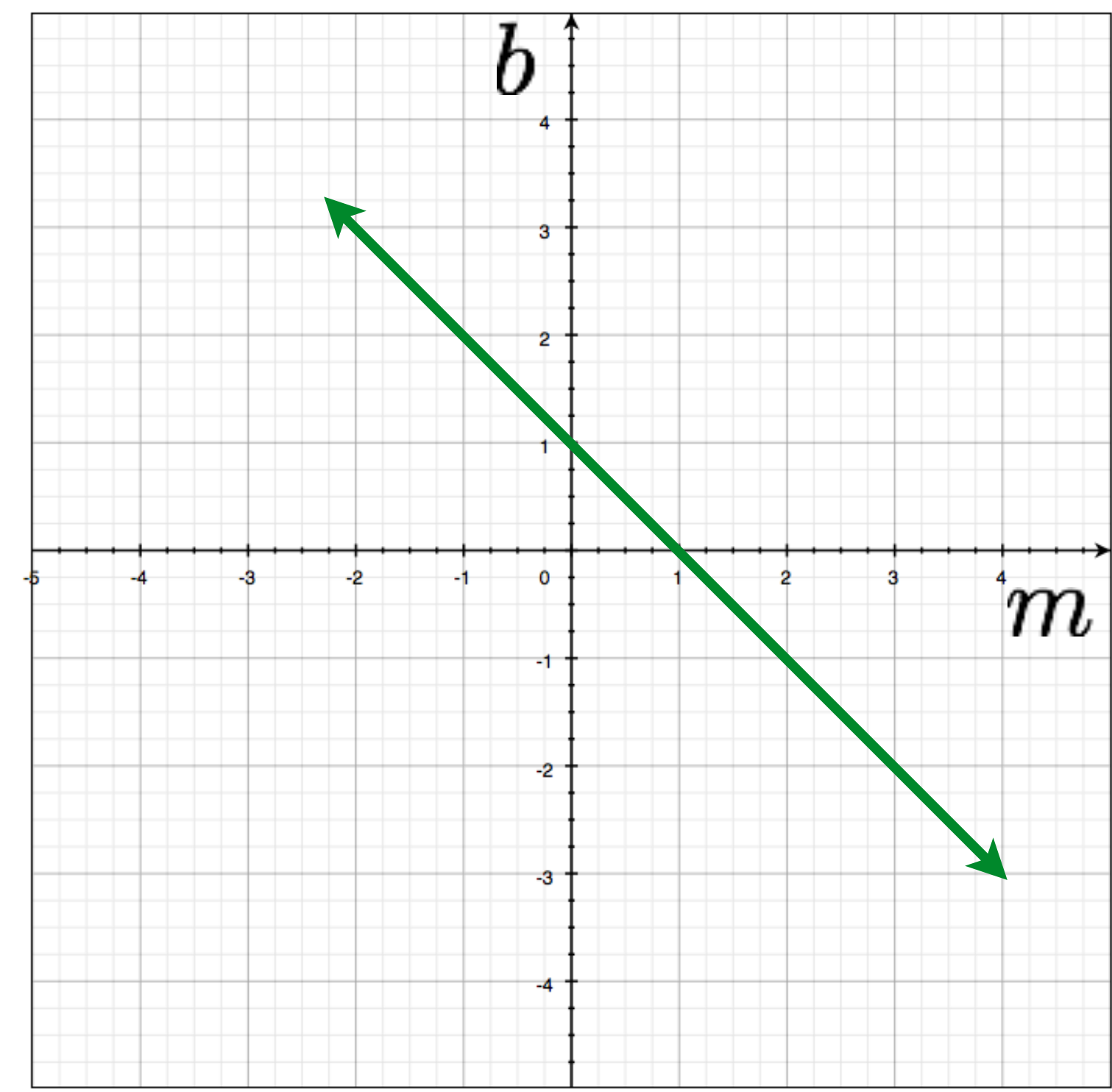


Image space

a point becomes a line



Parameter space

# Hough Transform: Lines

variables

$$y = mx + b$$

parameters

variables

$$y - mx = b$$

parameters

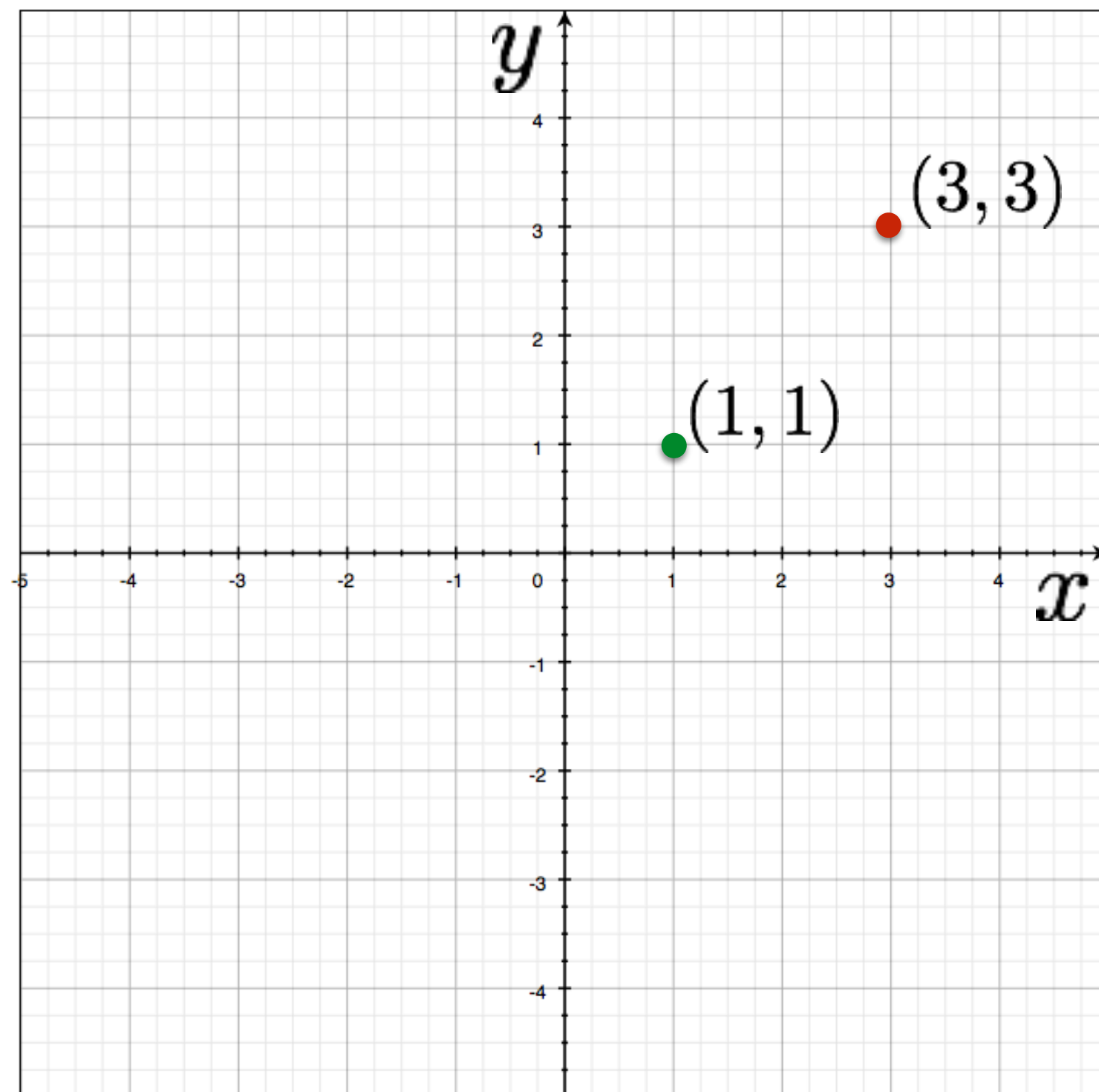
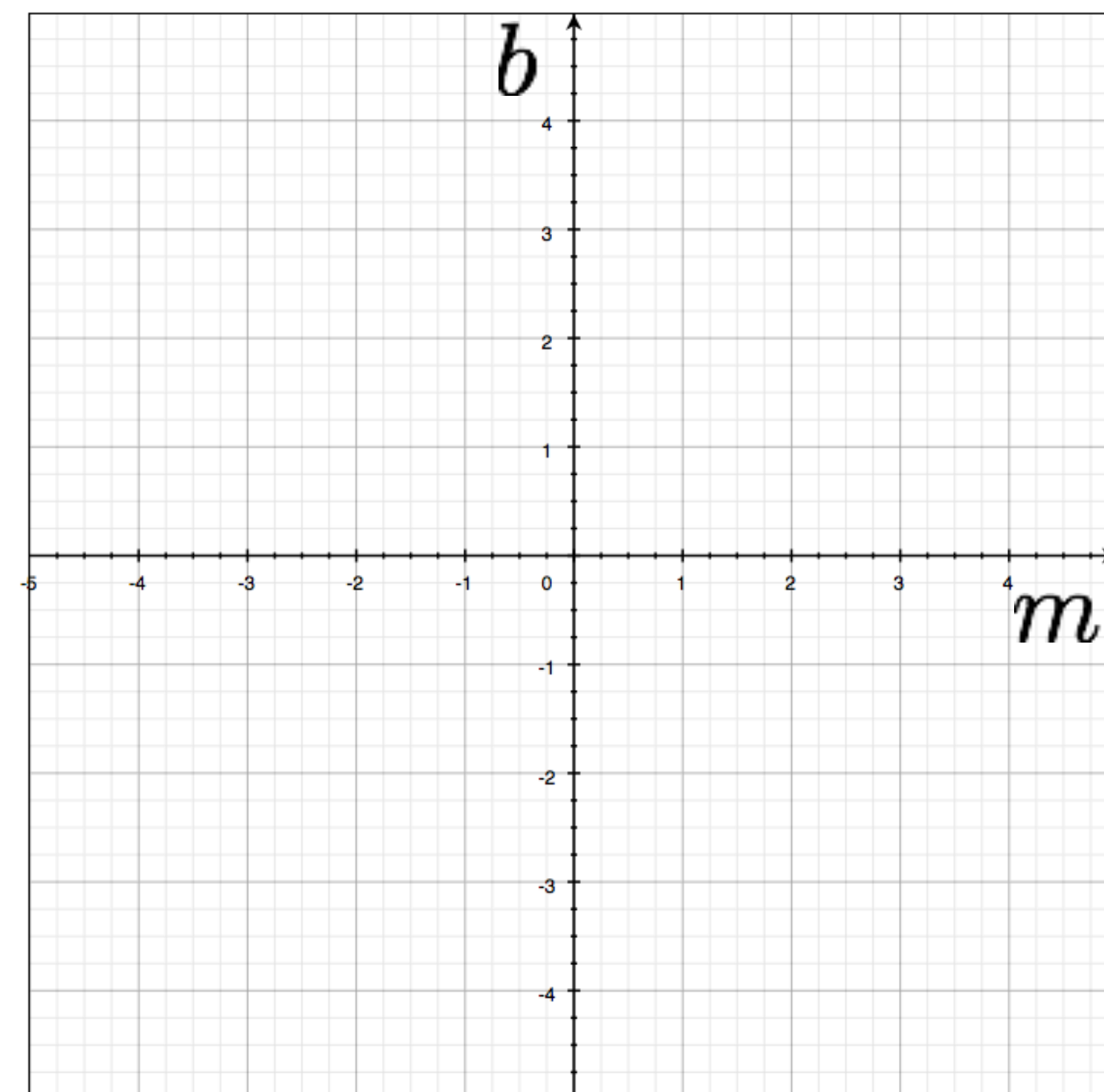


Image space



Parameter space

# Hough Transform: Lines

variables

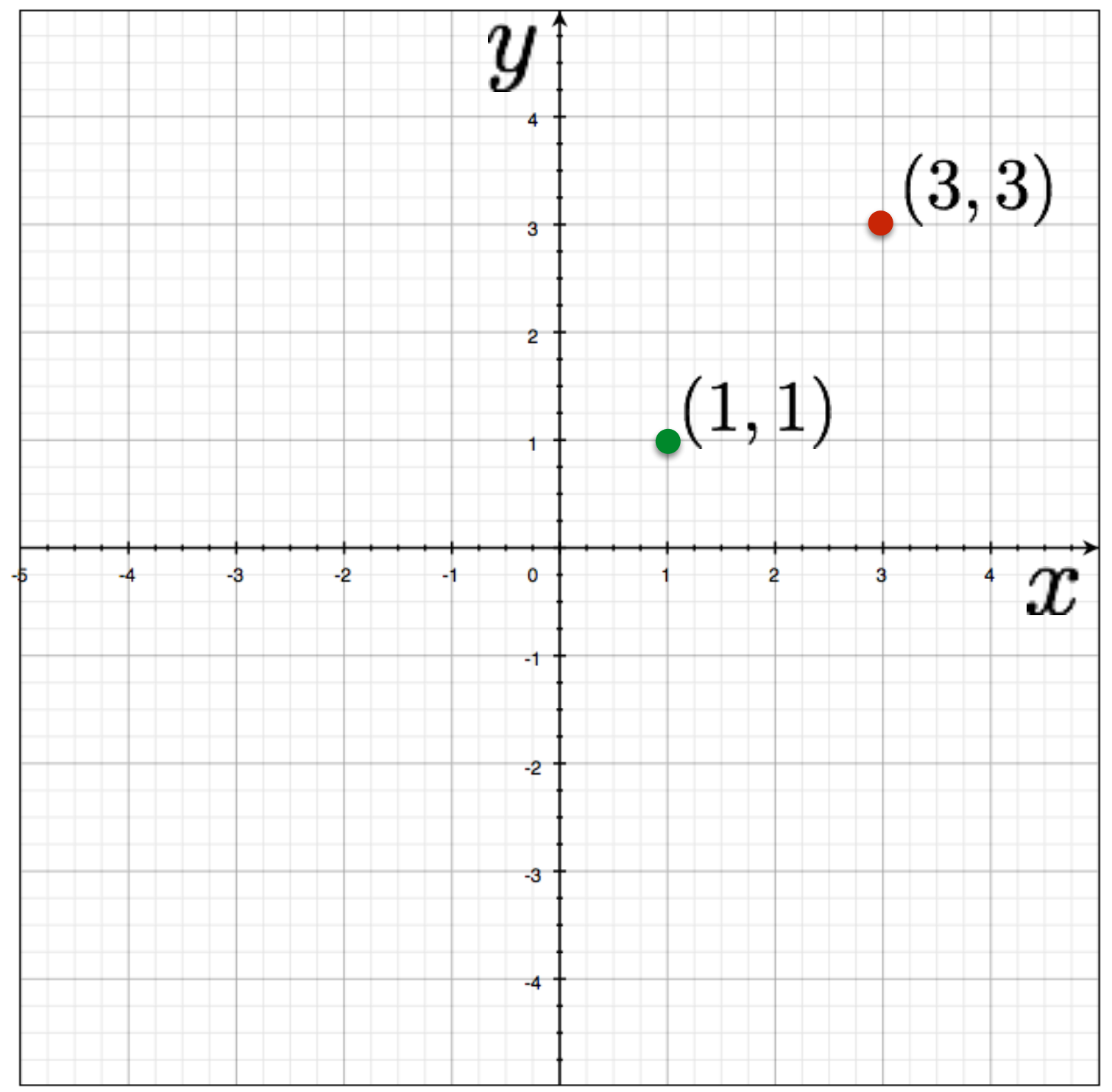
$$y = mx + b$$

parameters

variables

$$y - mx = b$$

parameters



two points?

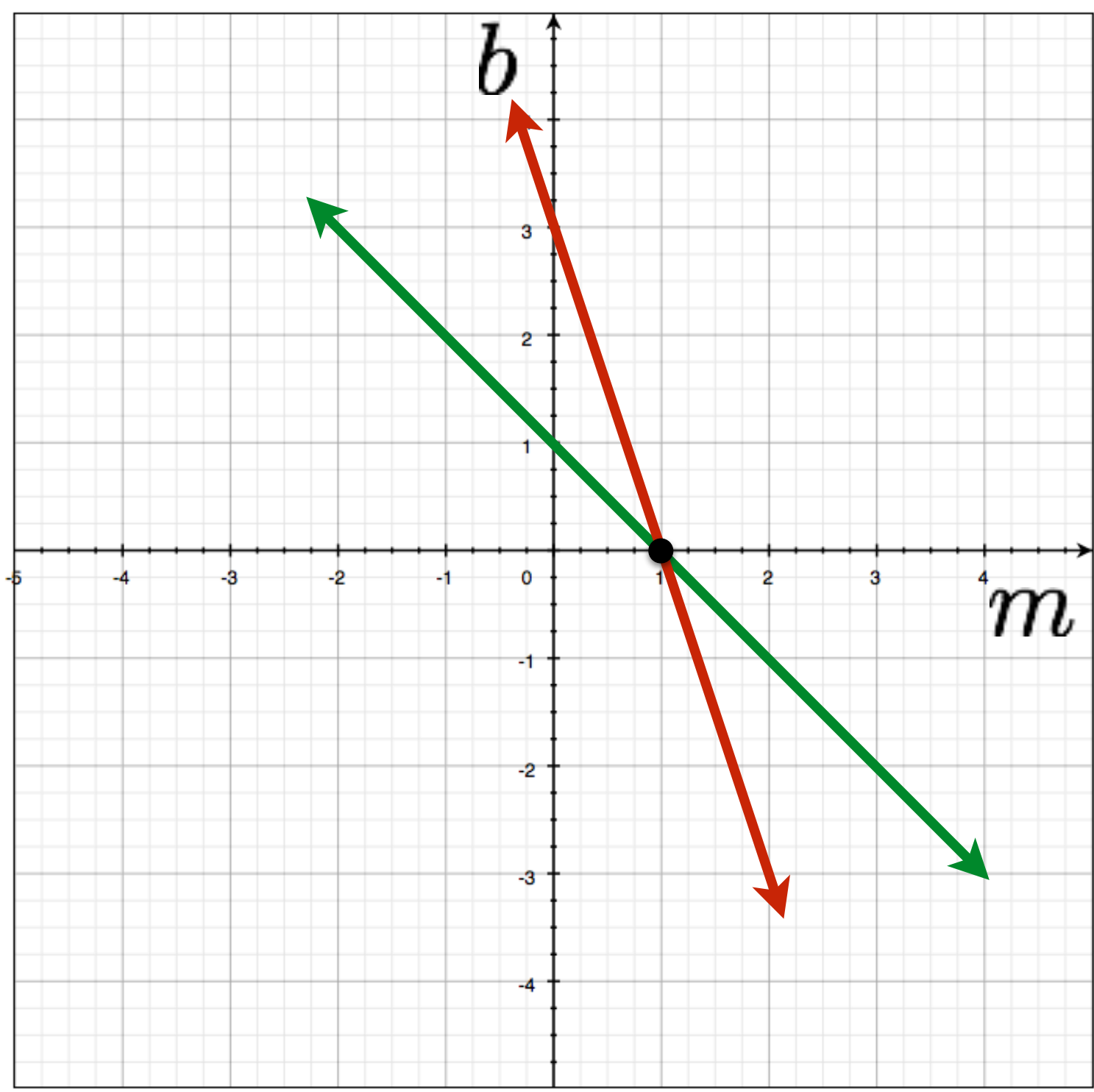


Image space

Parameter space

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

# Hough Transform: Lines

variables

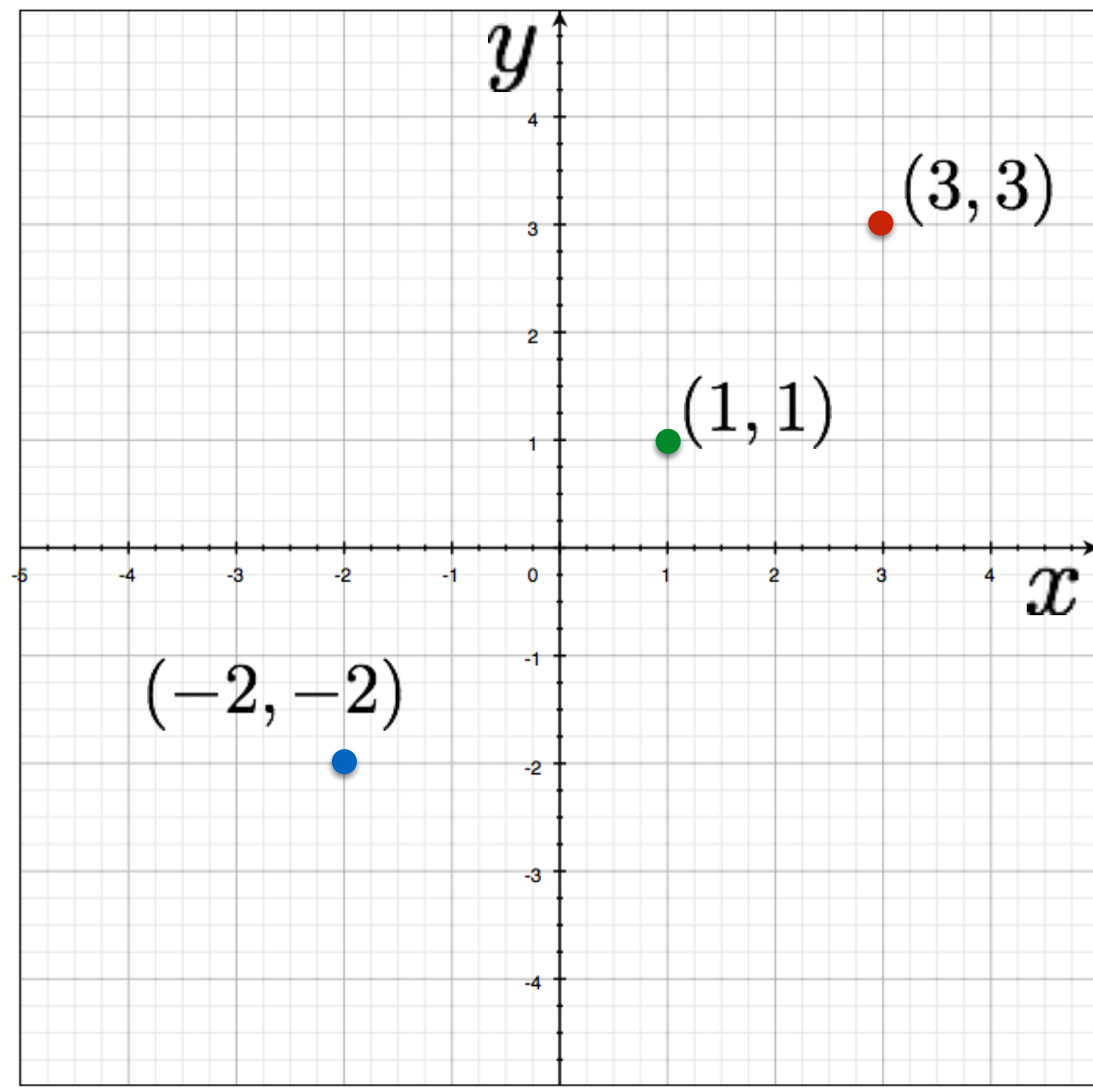
$$y = mx + b$$

parameters

variables

$$y - mx = b$$

parameters



three points?

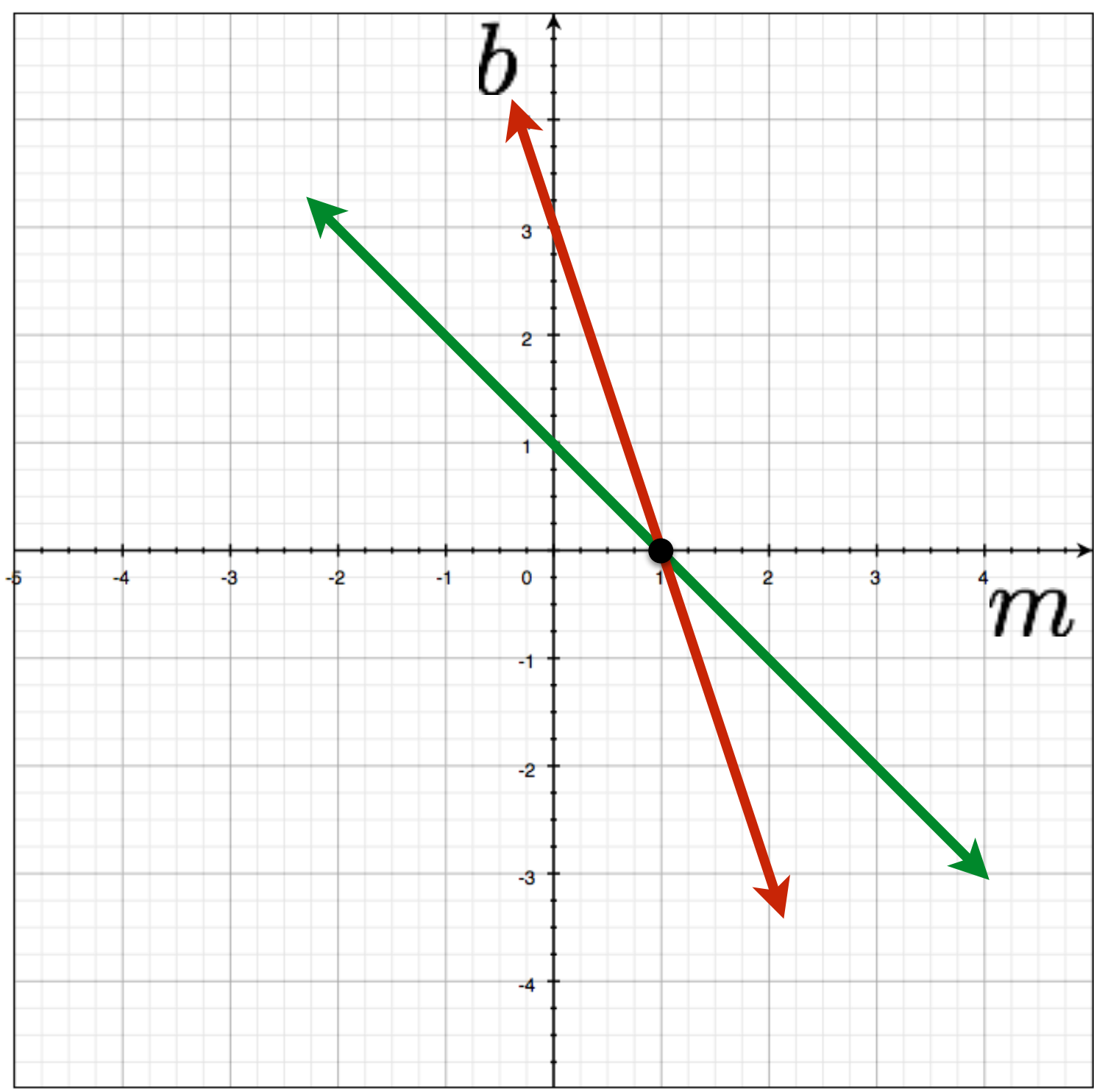


Image space

Parameter space

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)



# Hough Transform: Lines

variables

$$y = mx + b$$

parameters

variables

$$y - mx = b$$

parameters

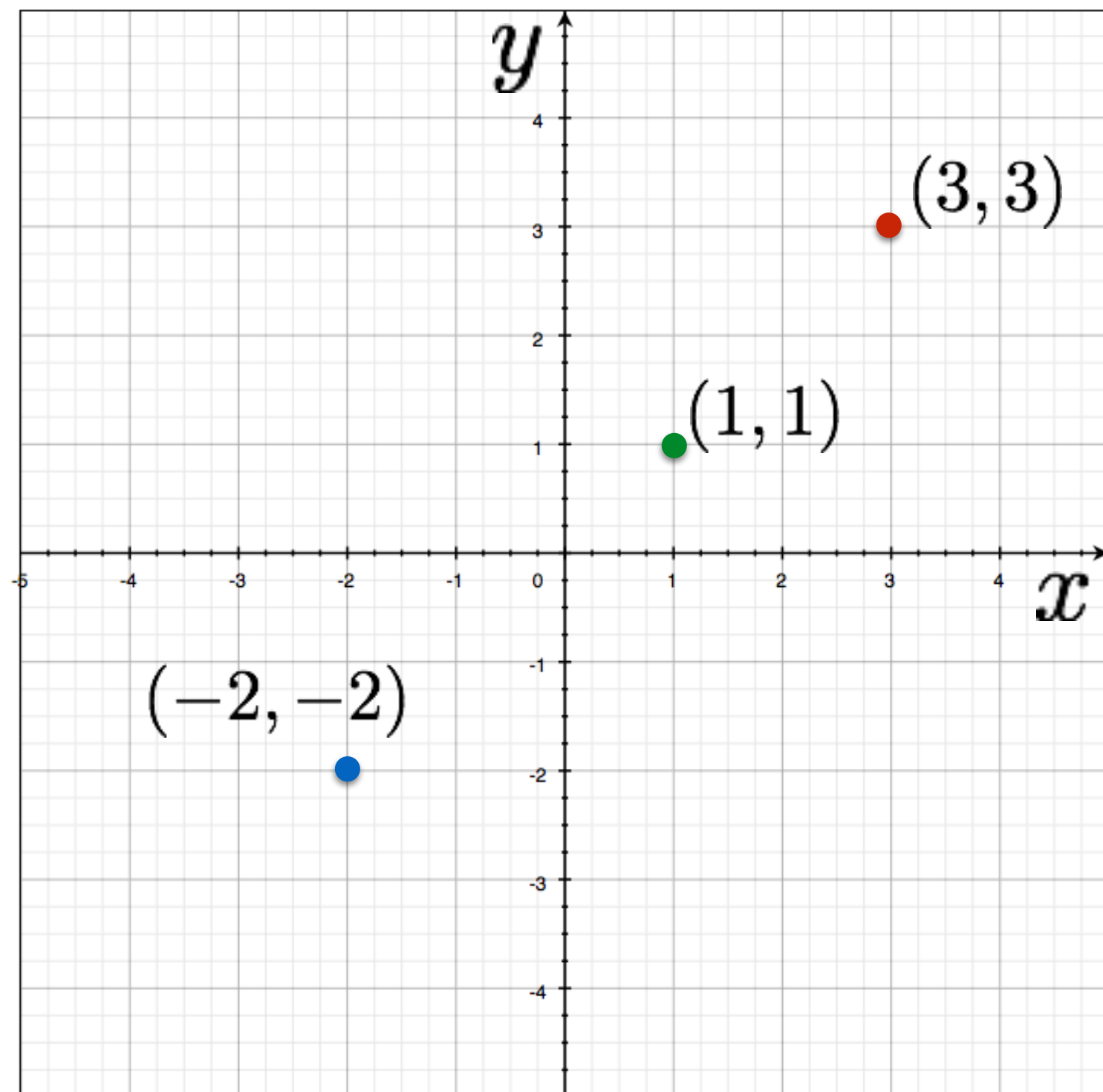
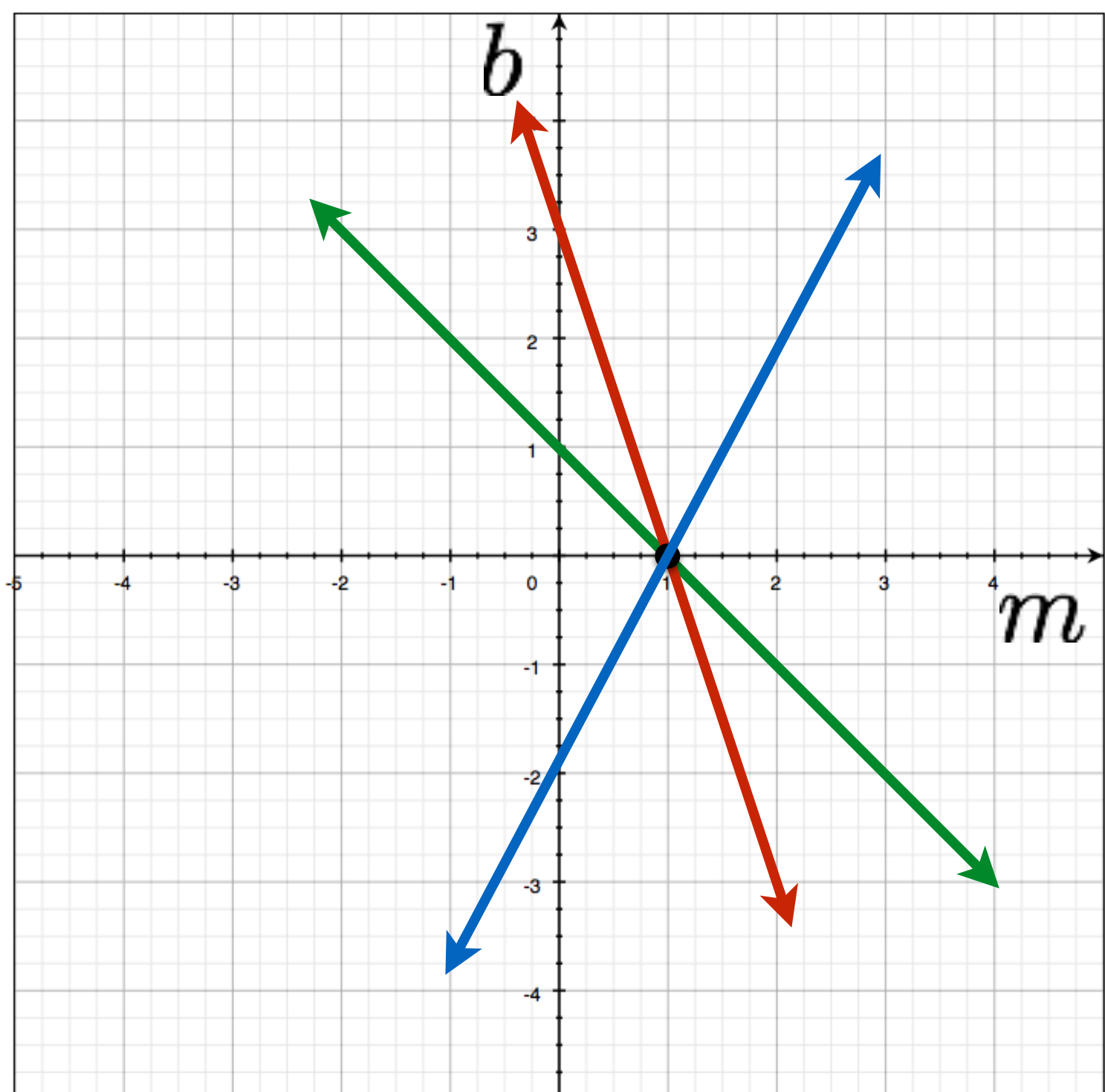


Image space

three points?



Parameter space

# Hough Transform: Lines

variables

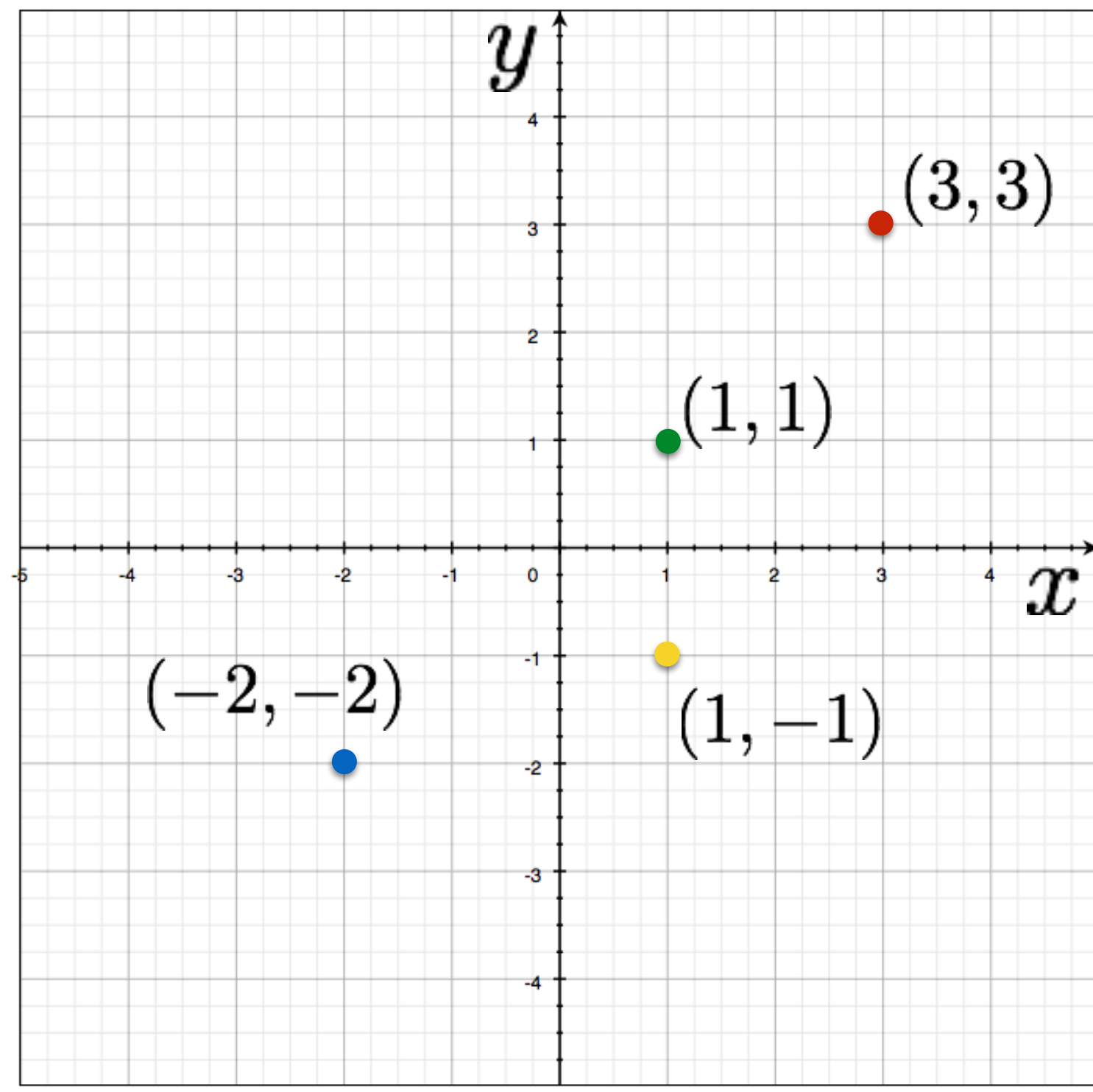
$$y = mx + b$$

parameters

variables

$$y - mx = b$$

parameters



four points?

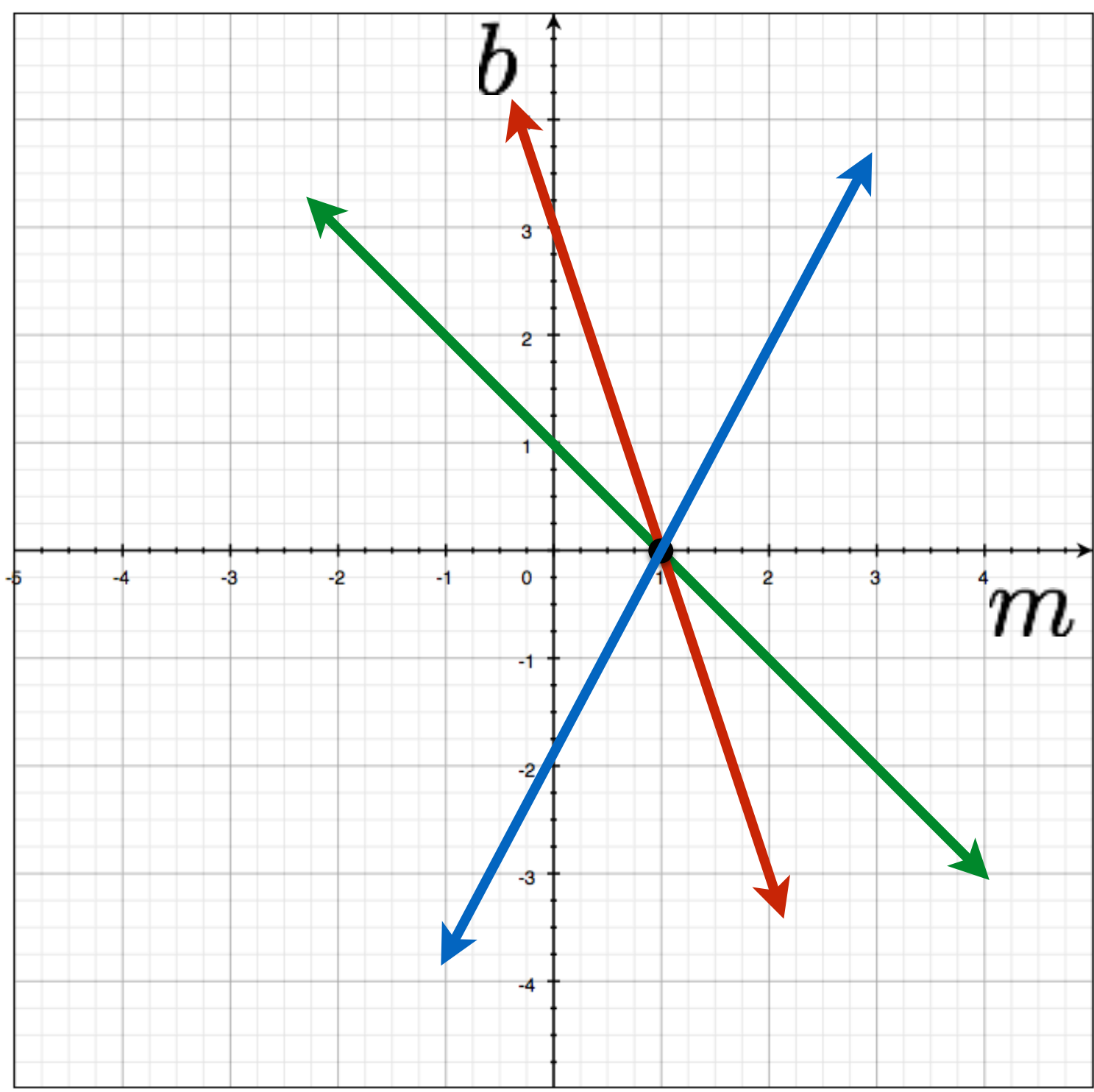


Image space

Parameter space



# Hough Transform: Lines

variables

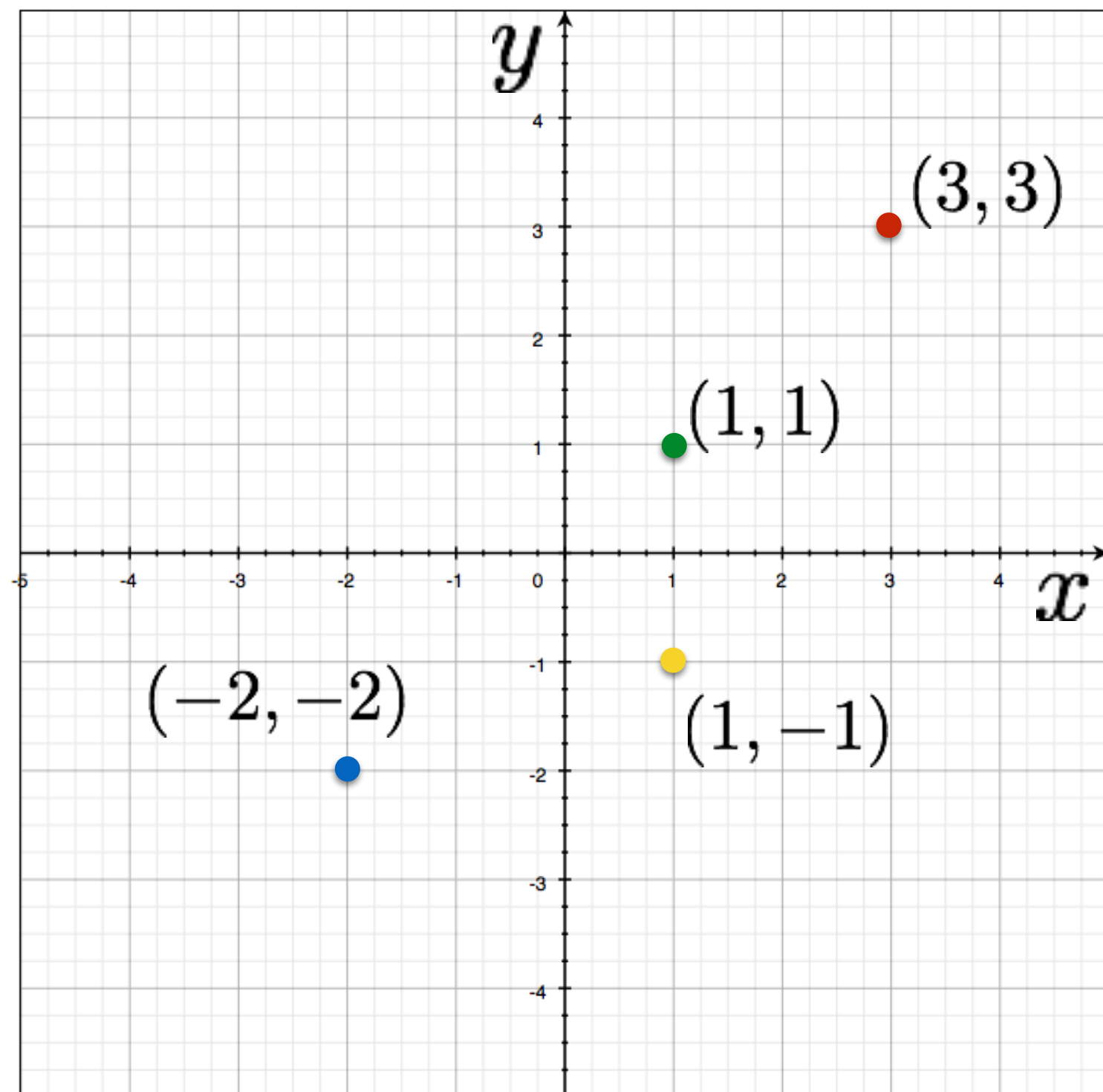
$$y = mx + b$$

parameters

variables

$$y - mx = b$$

parameters



four points?

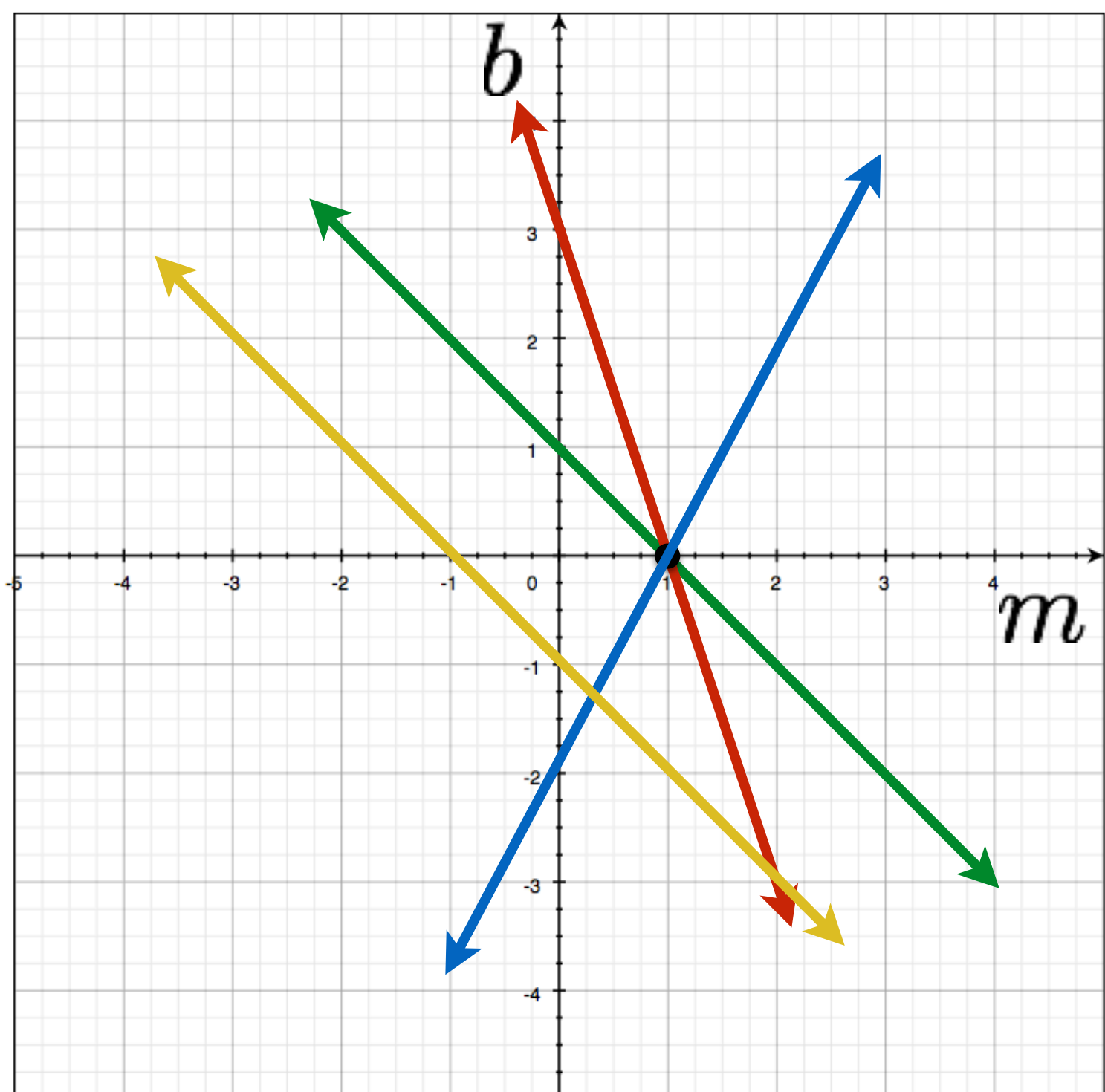


Image space

Parameter space

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

# Hough Transform: Lines

How would you find the best fitting line?

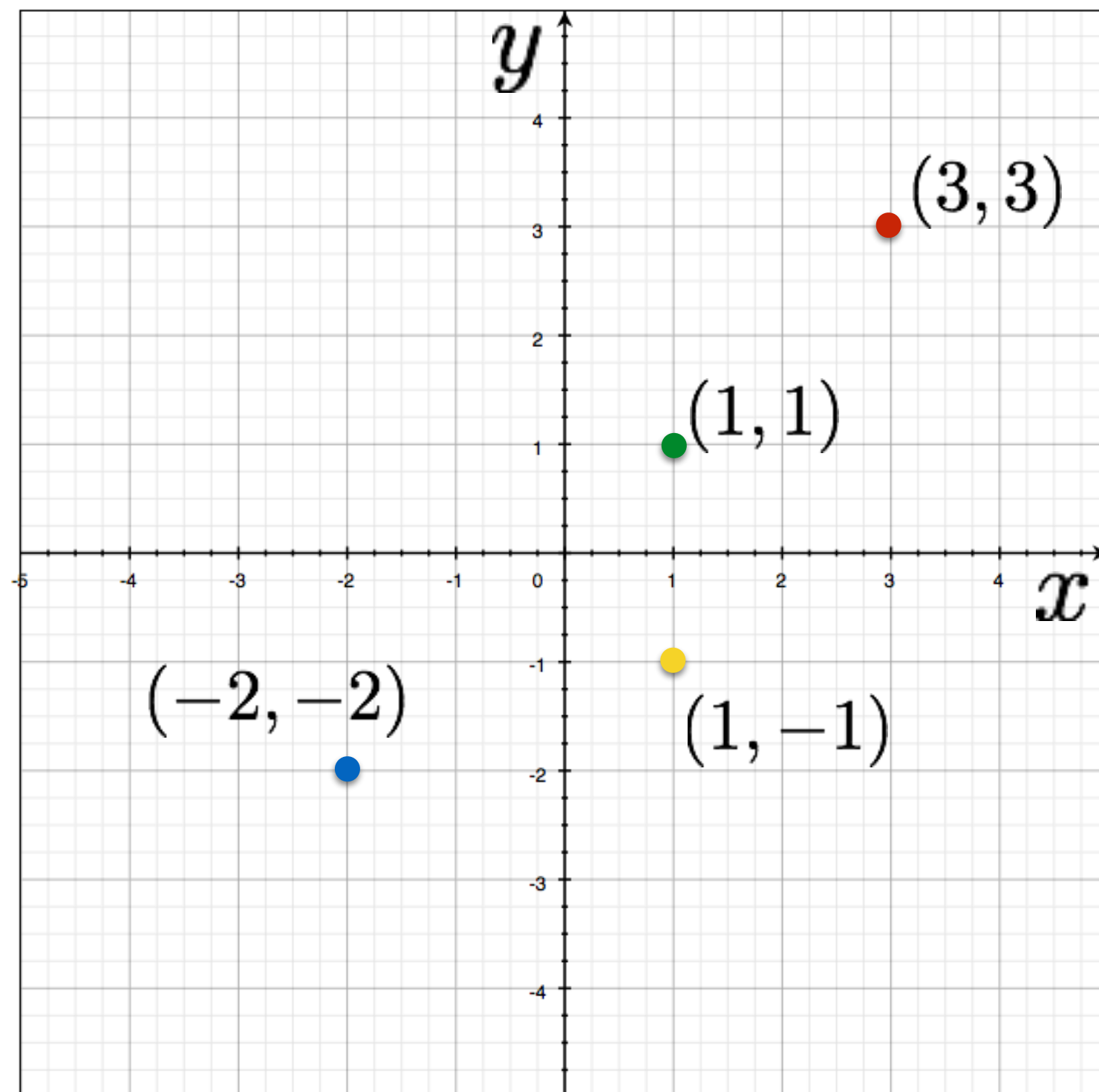
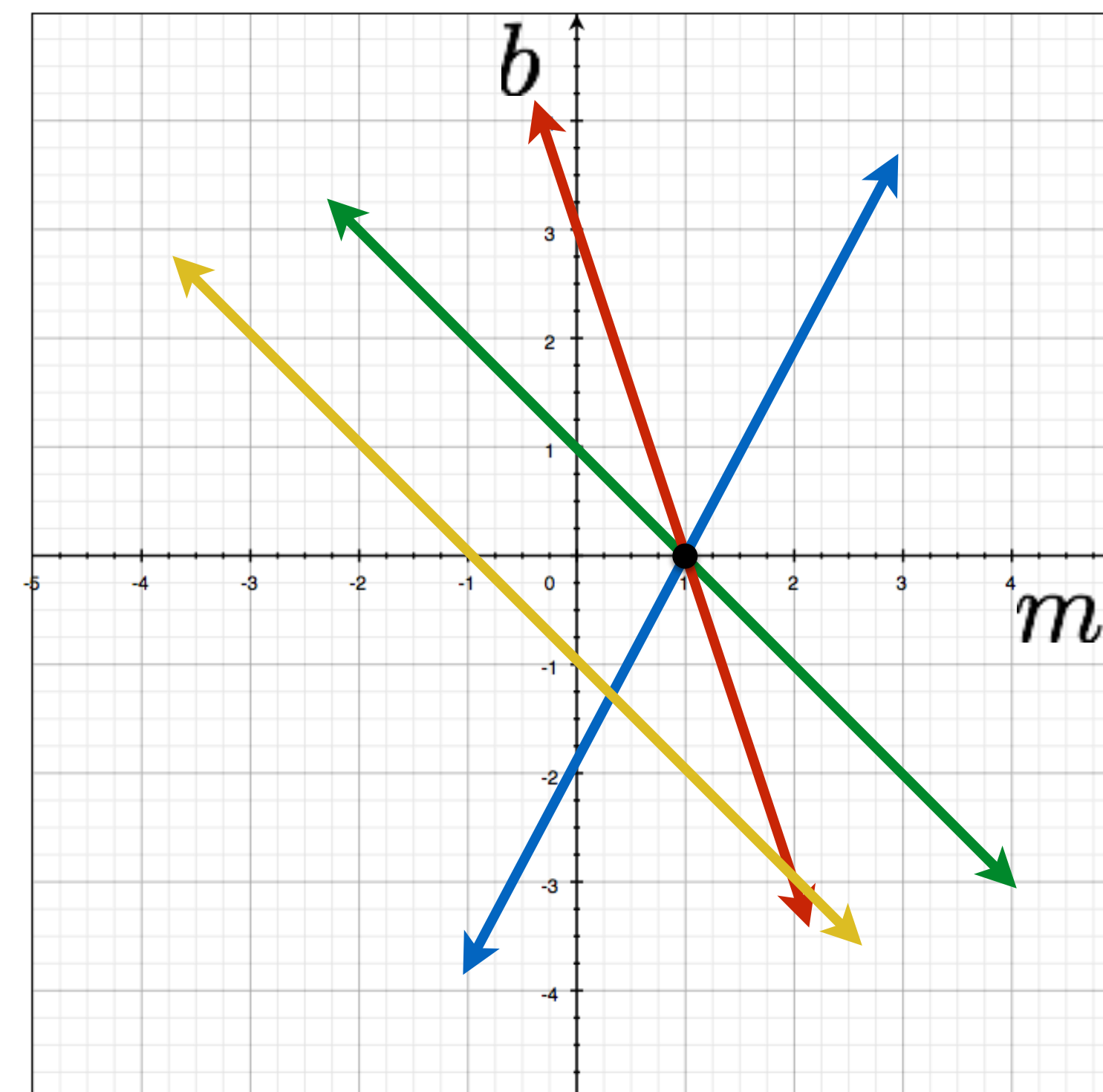


Image space



Parameter space

# Hough Transform: Lines

Is this method robust to measurement noise? clutter?

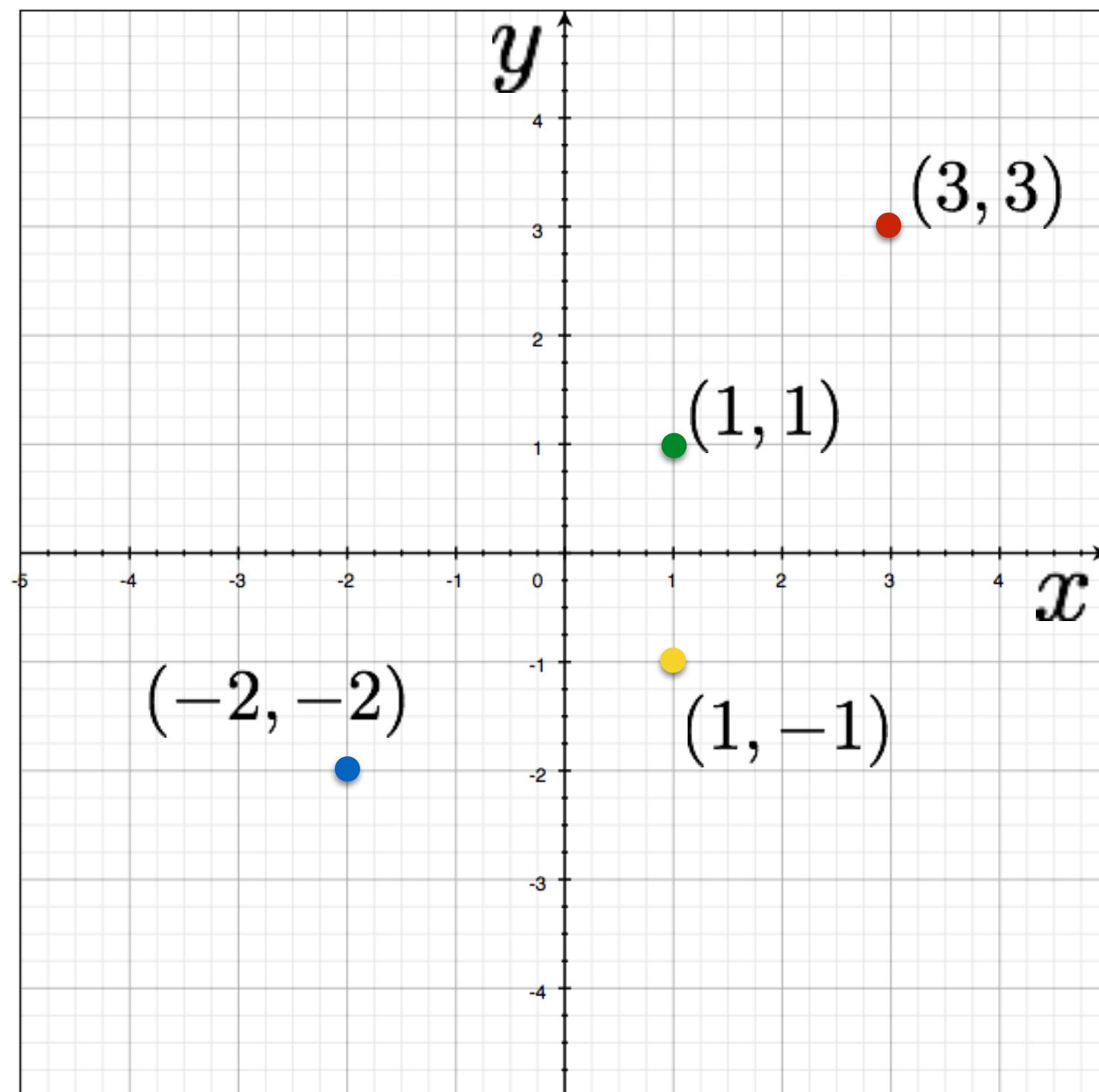
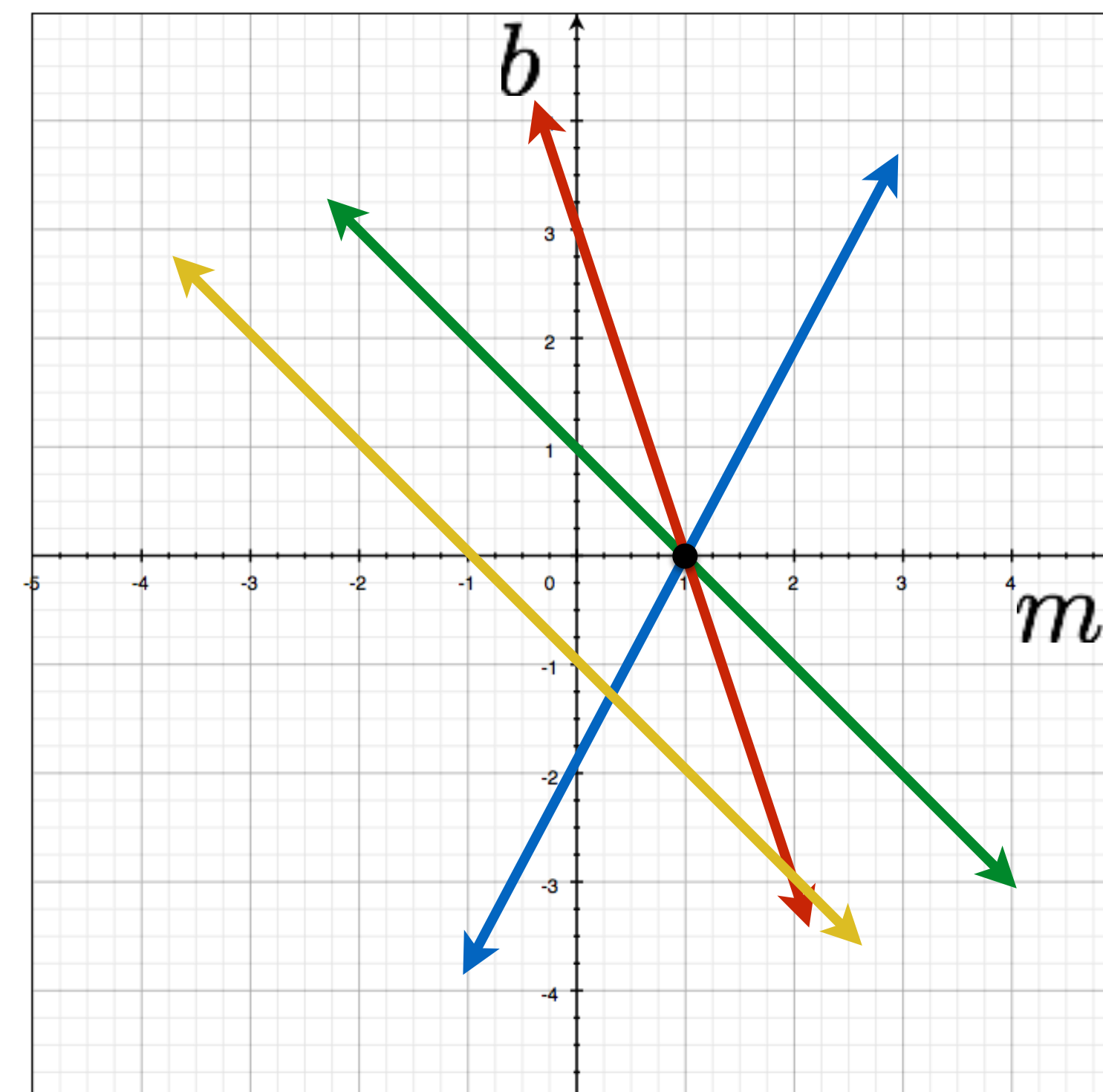


Image space



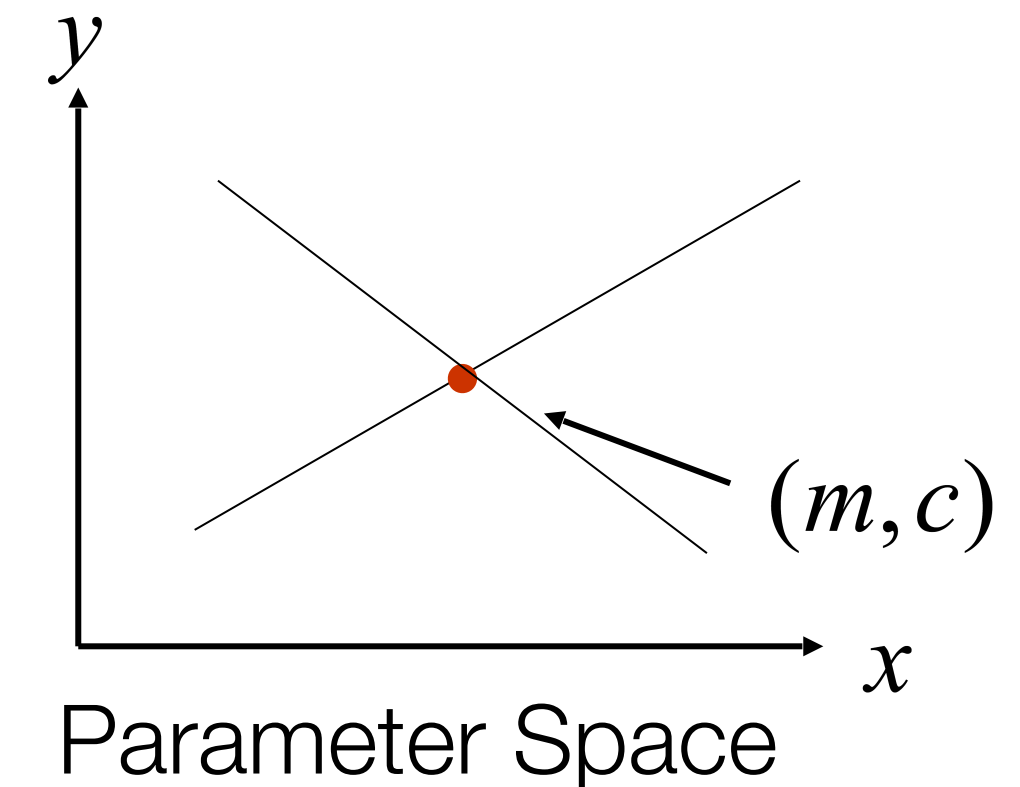
Parameter space



# 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  $A(m, c)$



$A(m, c)$

1					1		
	1				1		
		1	1				
			2				
		1	1				
	1				1		
1						1	

# Problems with **Parametrization**

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 **Parametrization**

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!

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

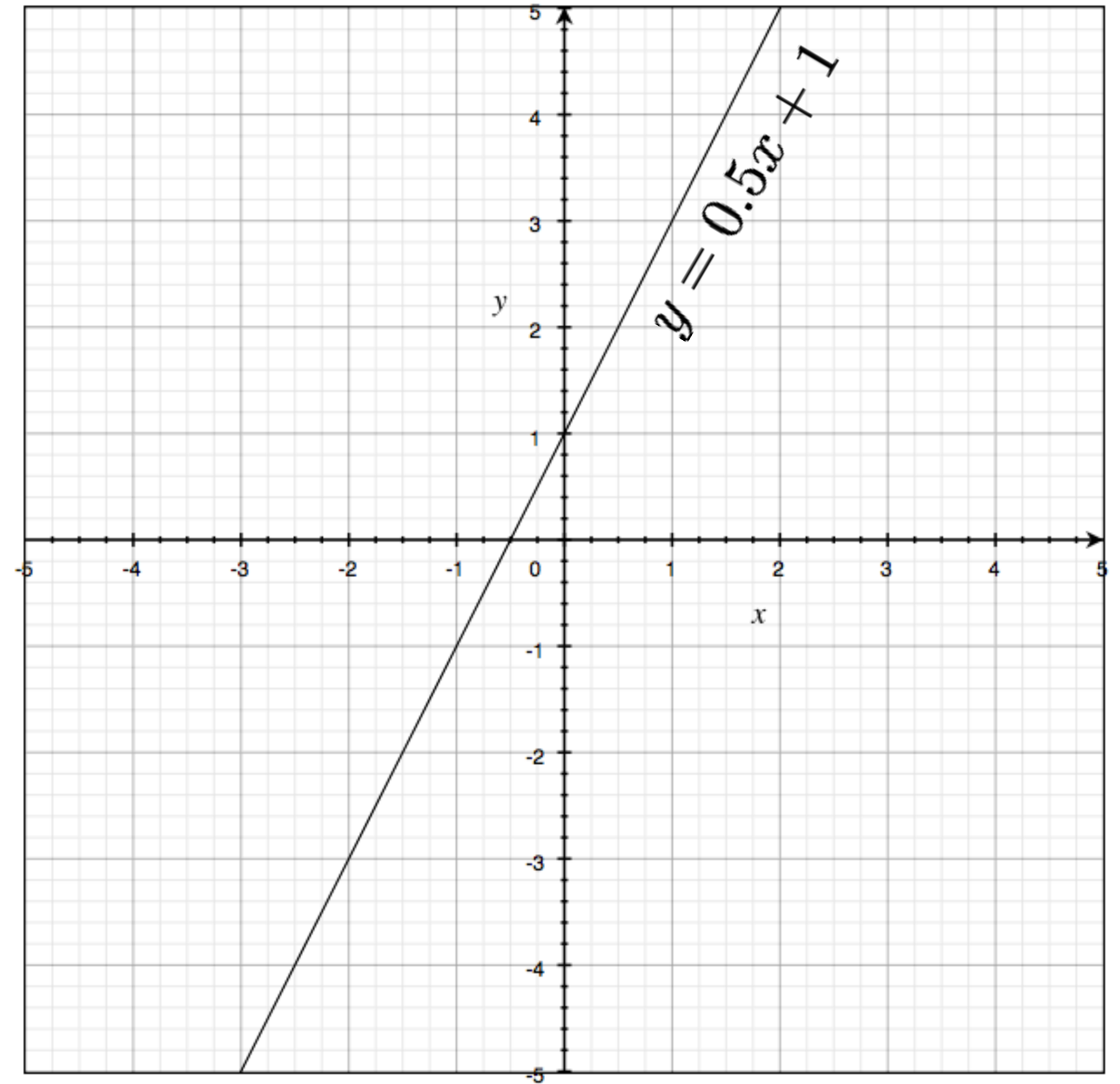


# Lines: Slope intercept form

$$y = mx + b$$

↑  
slope

↑  
y-intercept



# Lines: Normal form

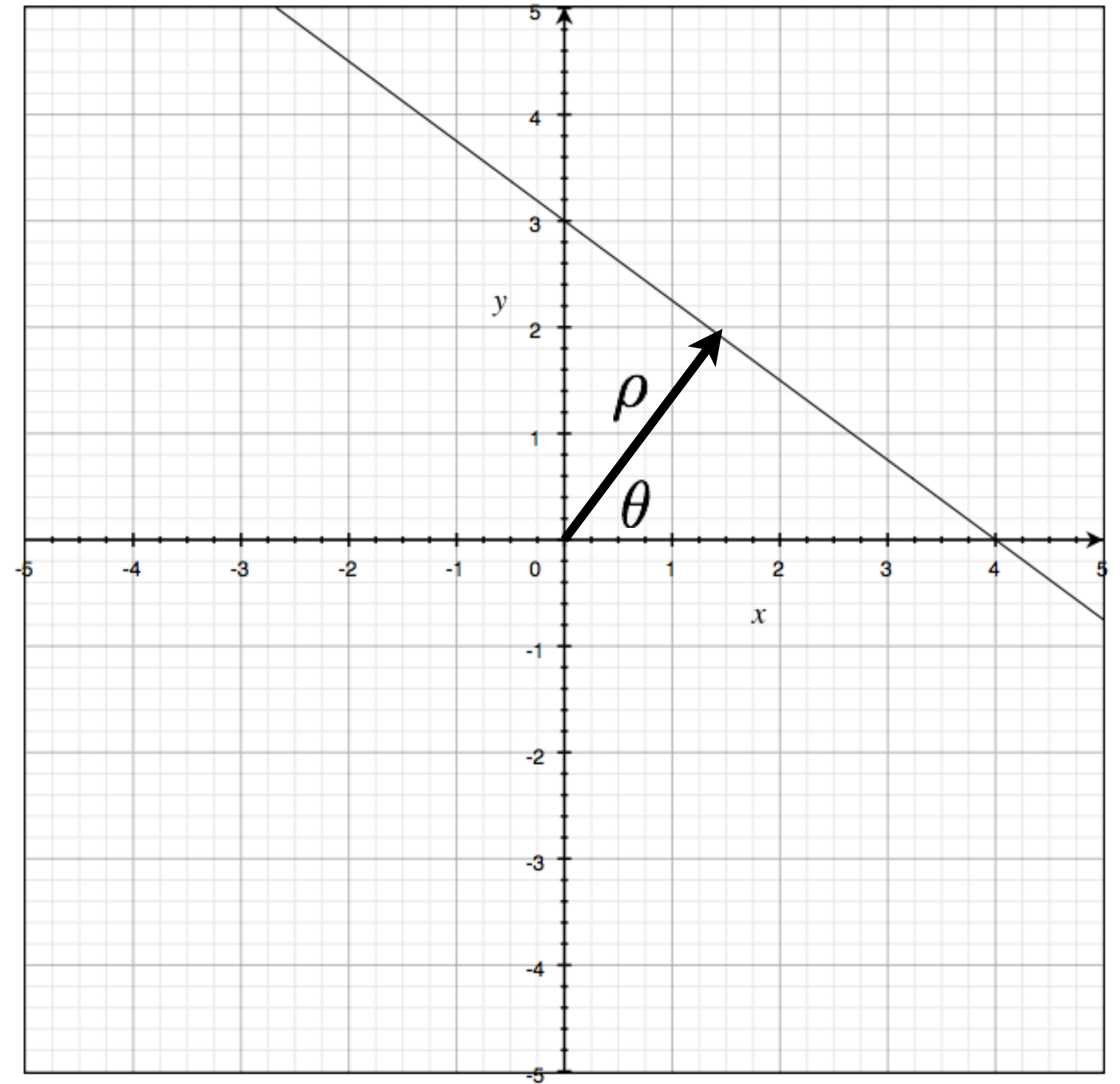
$$x \sin \theta + y \cos \theta = \rho$$

**Forsyth/Ponce convention**

$$x \sin \theta + y \cos \theta + r = 0$$

$$r \geq 0$$

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



# Hough Transform: Lines

$$y = mx + b$$

variables

parameters

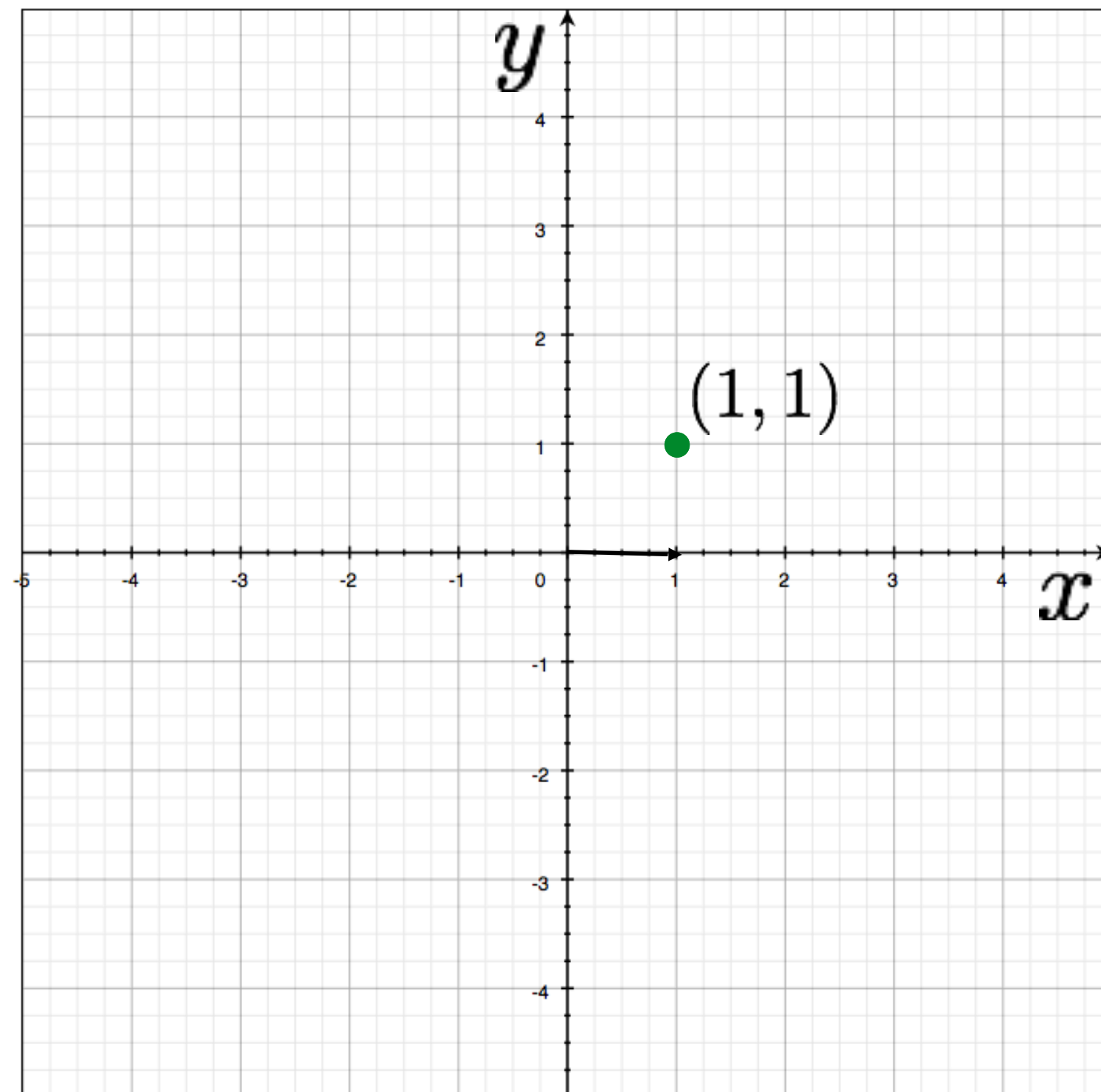


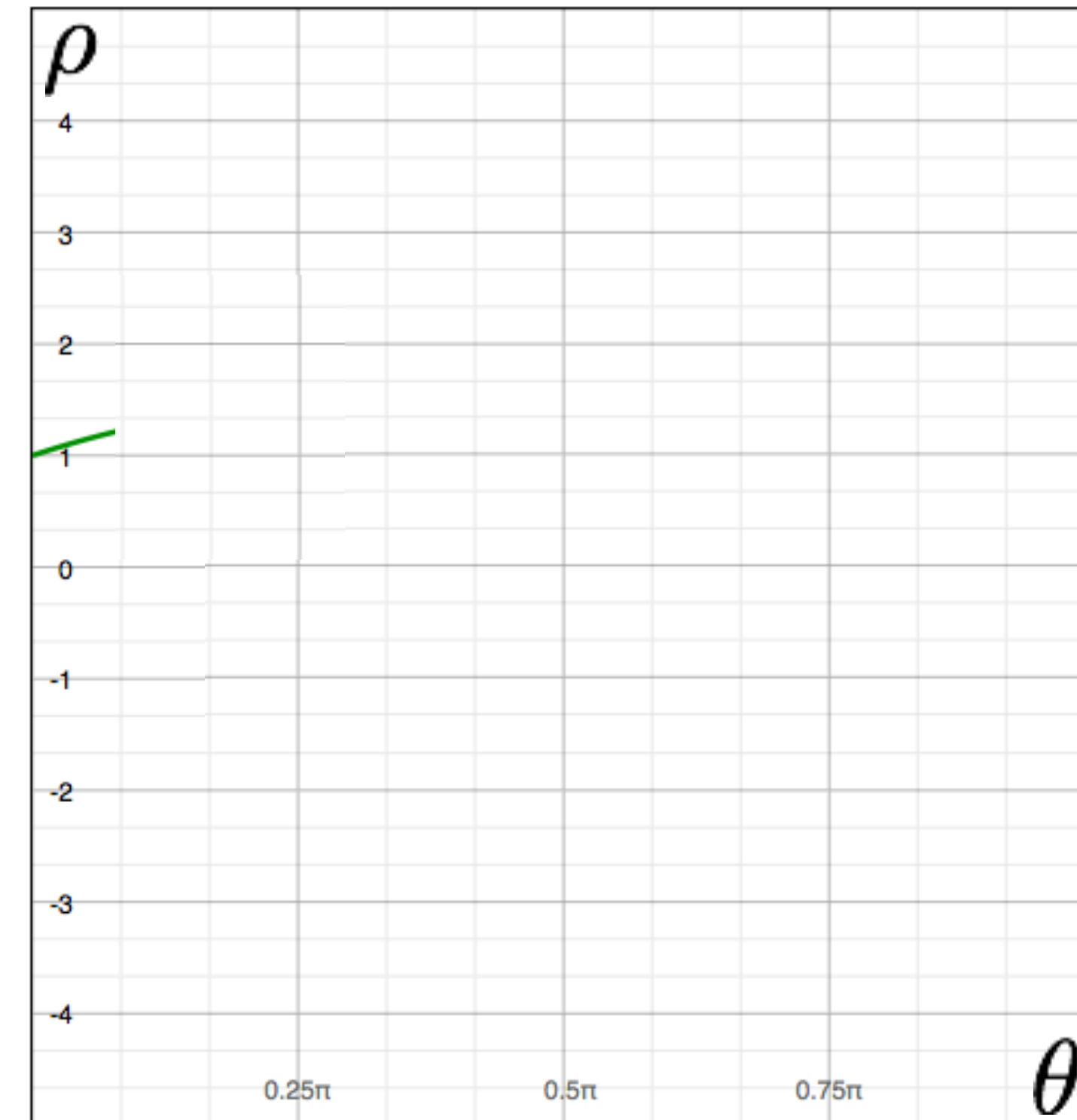
Image space



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

parameters

variables



Parameter space



# Hough Transform: Lines

variables

$$y = mx + b$$

parameters

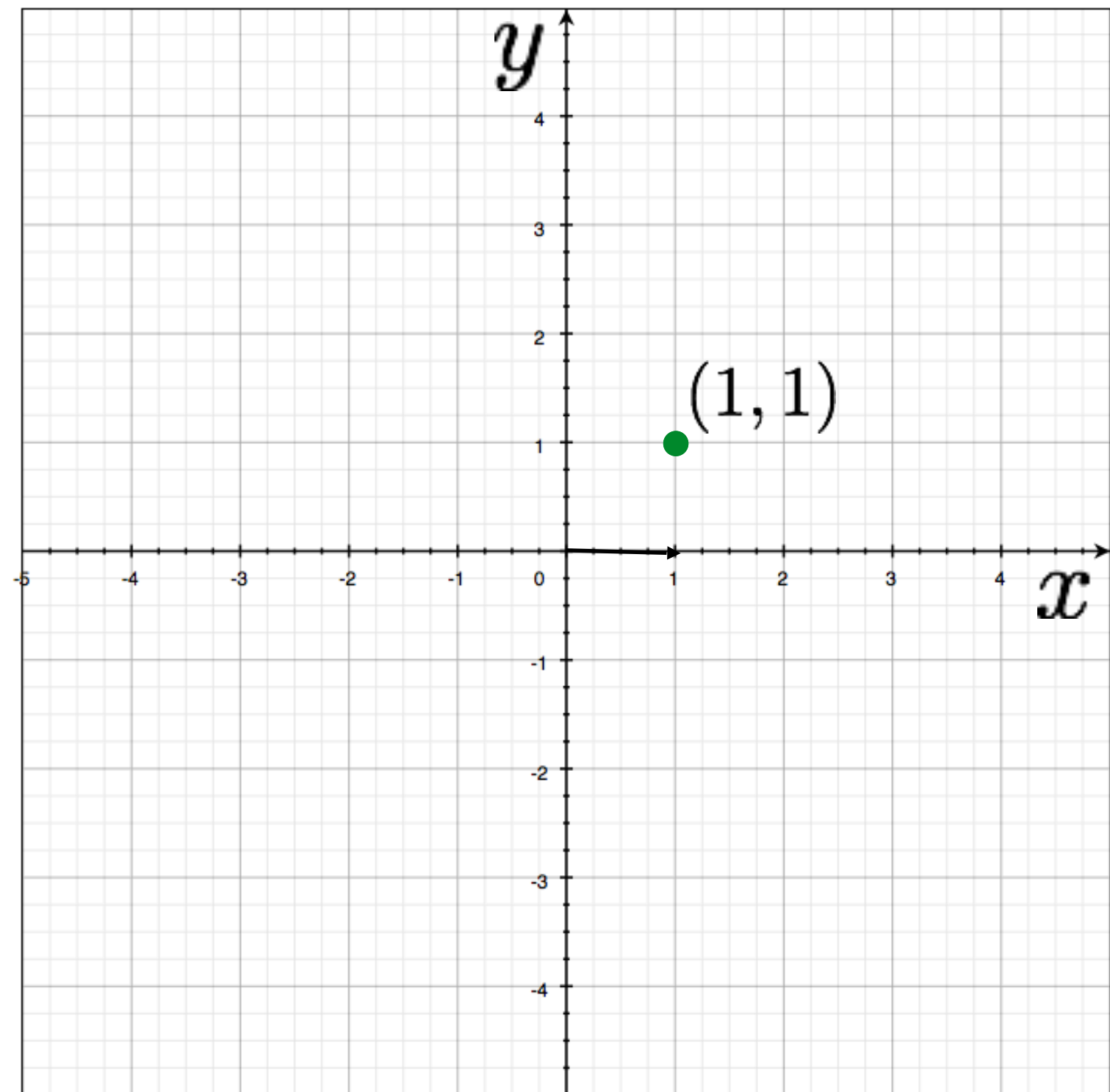


Image space

a point becomes a wave

parameters

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

variables



Parameter space

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

# Hough Transform: Lines

variables

$$y = mx + b$$

parameters

parameters

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

variables

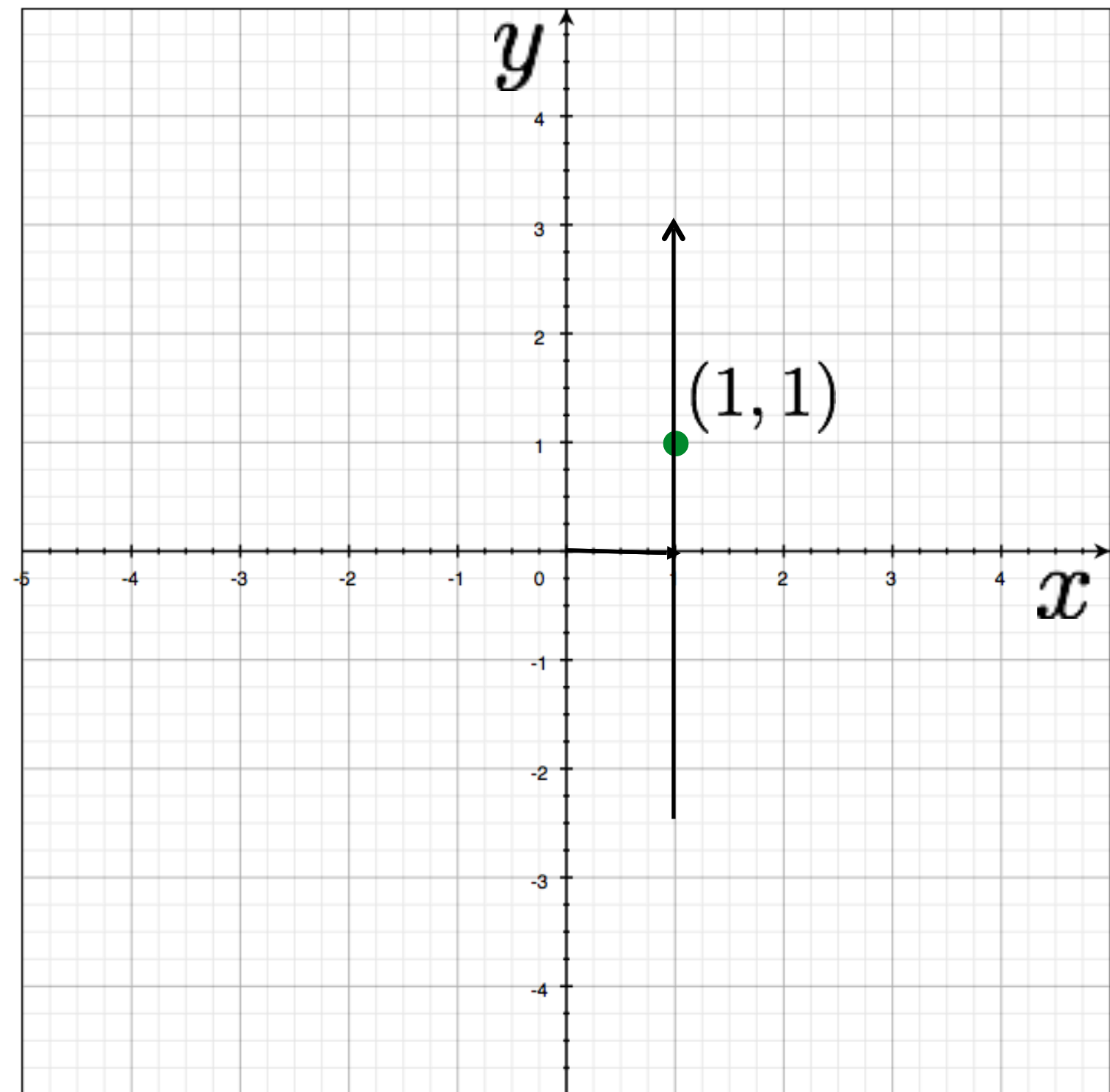
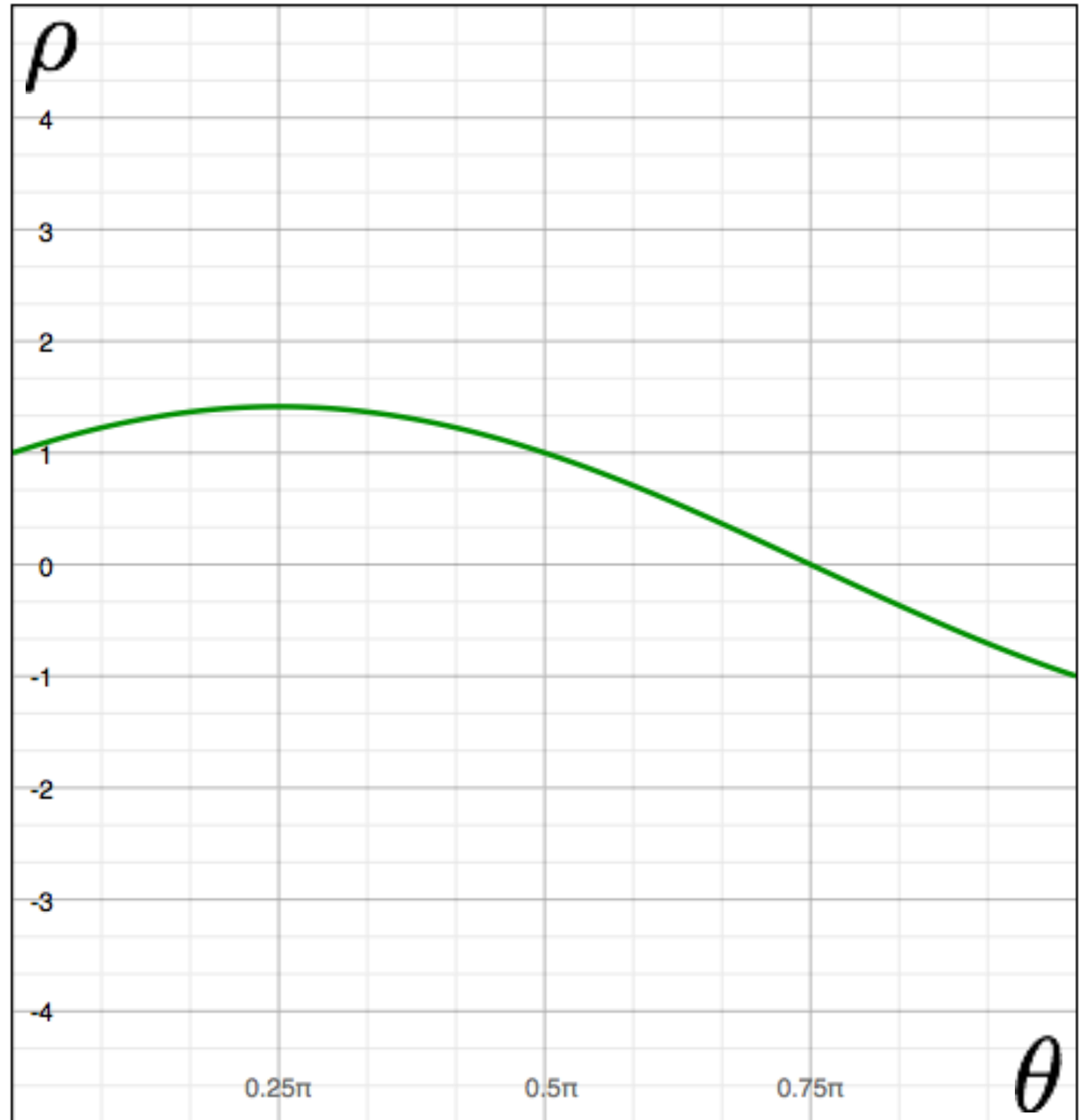


Image space

a line becomes?



Parameter space

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

# Hough Transform: Lines

variables

$$y = mx + b$$

parameters

parameters

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

variables

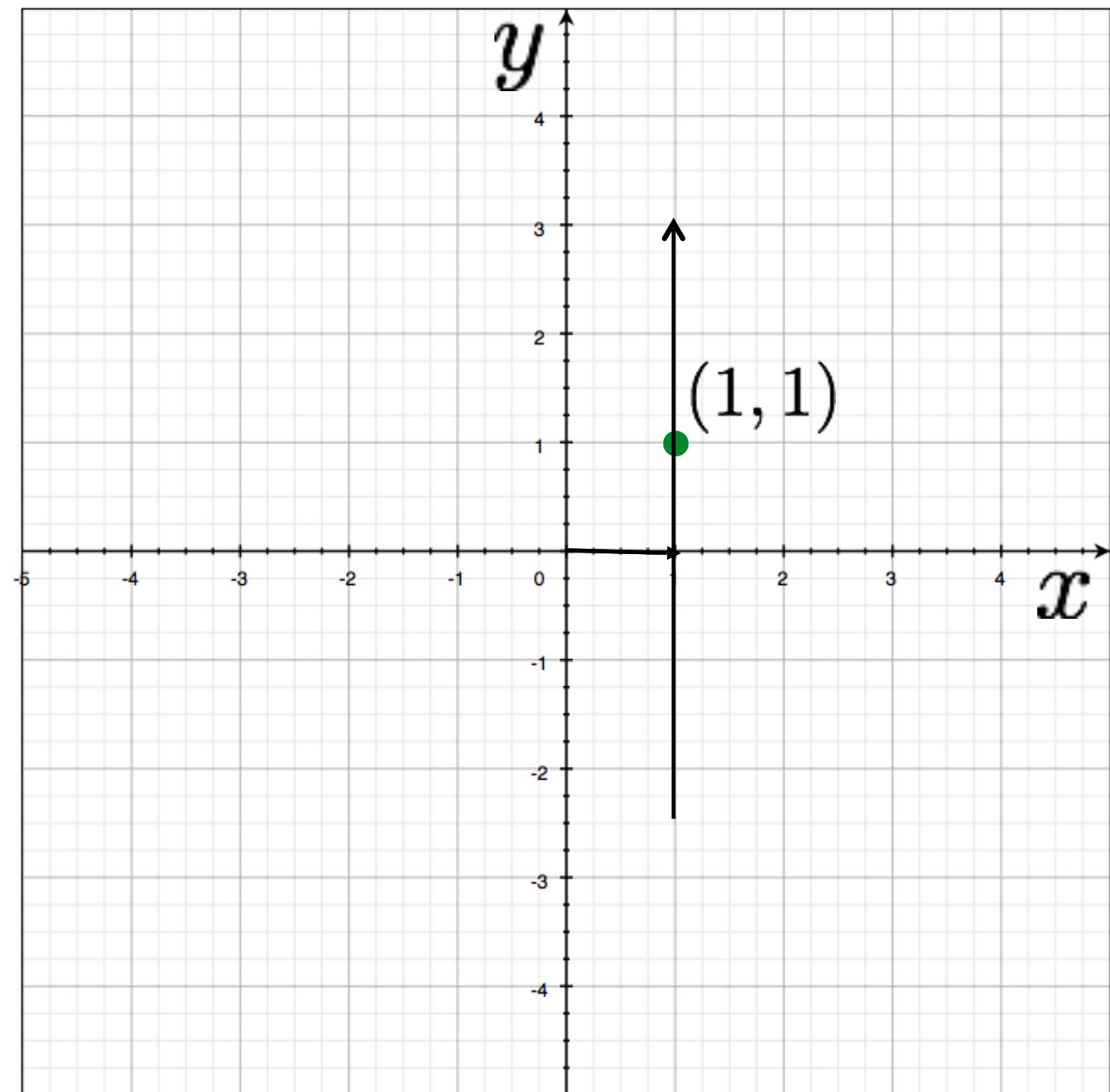


Image space

a line becomes a point



Parameter space



# Hough Transform for Lines (switching to books notation)

**Idea:** Each point votes for the lines that pass through it

— A line is the set of points,  $(x, y)$ , such that

$$x \sin \theta + y \cos \theta + r = 0$$

— Different choices of  $\theta, r$  give different lines

# Hough Transform for Lines (switching to books notation)

**Idea:** Each point votes for the lines that pass through it

— A line is the set of points,  $(x, y)$ , such that

$$x \sin \theta + y \cos \theta + r = 0$$

— Different choices of  $\theta, r$  give different lines

— For any  $(x, y)$  there is a one parameter family of lines through this point. Just let  $(x, y)$  be constants and for each value of  $\theta$  the value of  $r$  will be determined

— Each point enters votes for each line in the family

— If there is a line that has lots of votes, that will be the line passing near the points that voted for it

# Hough Transform: Lines

variables

$$y = mx + b$$

parameters

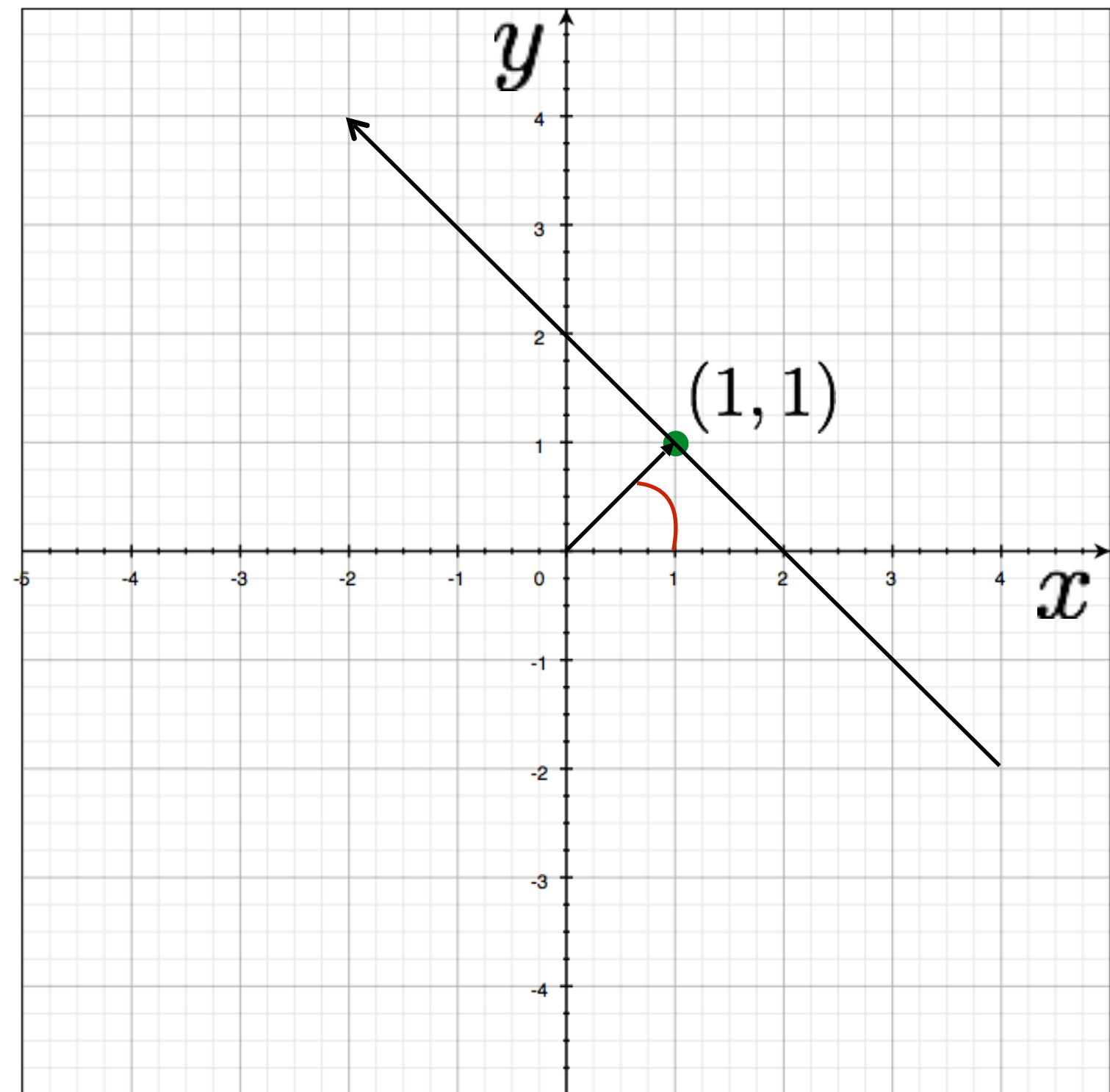


Image space

a line becomes?

parameters

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

variables



Parameter space



# Hough Transform: Lines

variables

$$y = mx + b$$

parameters

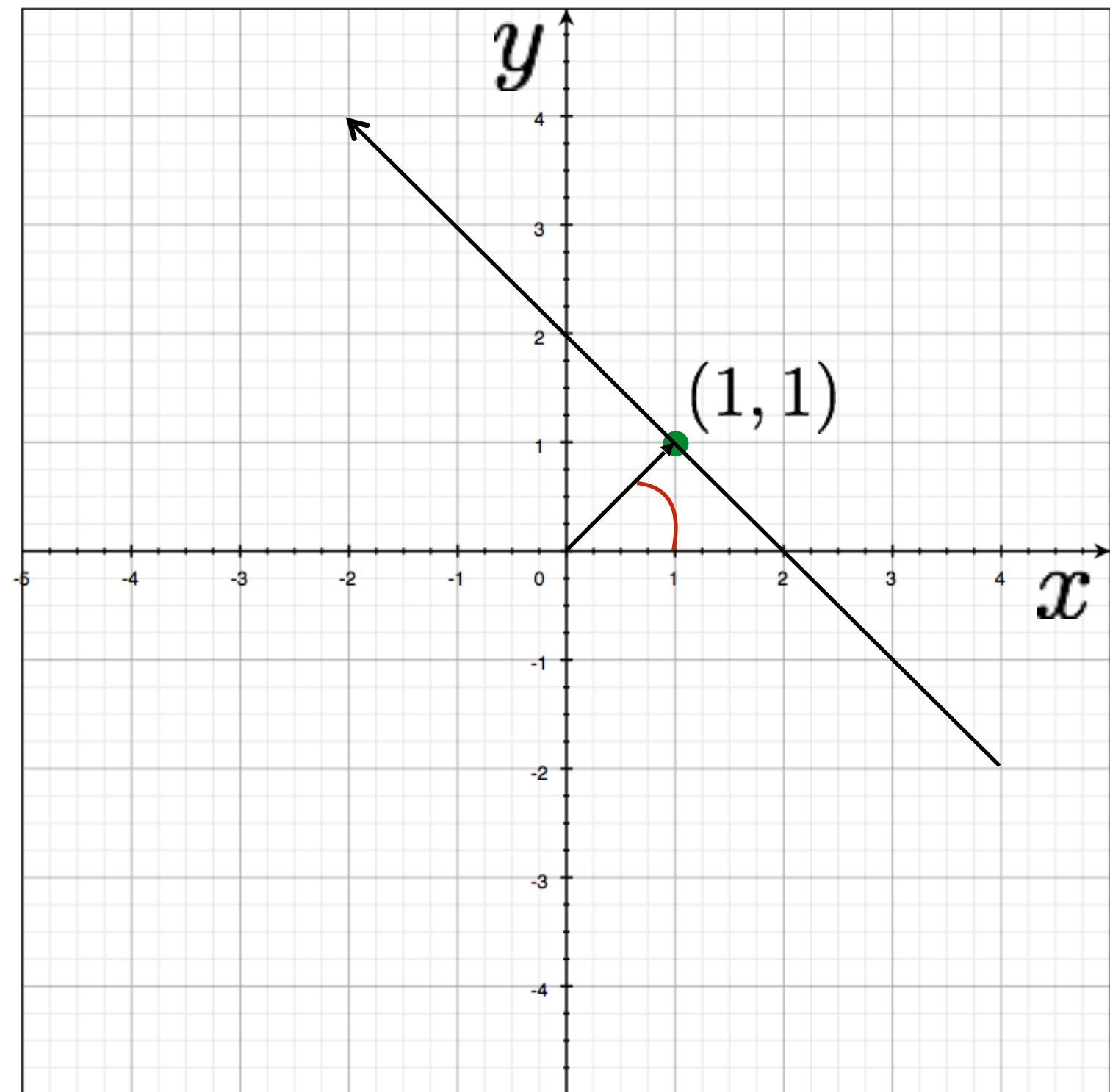


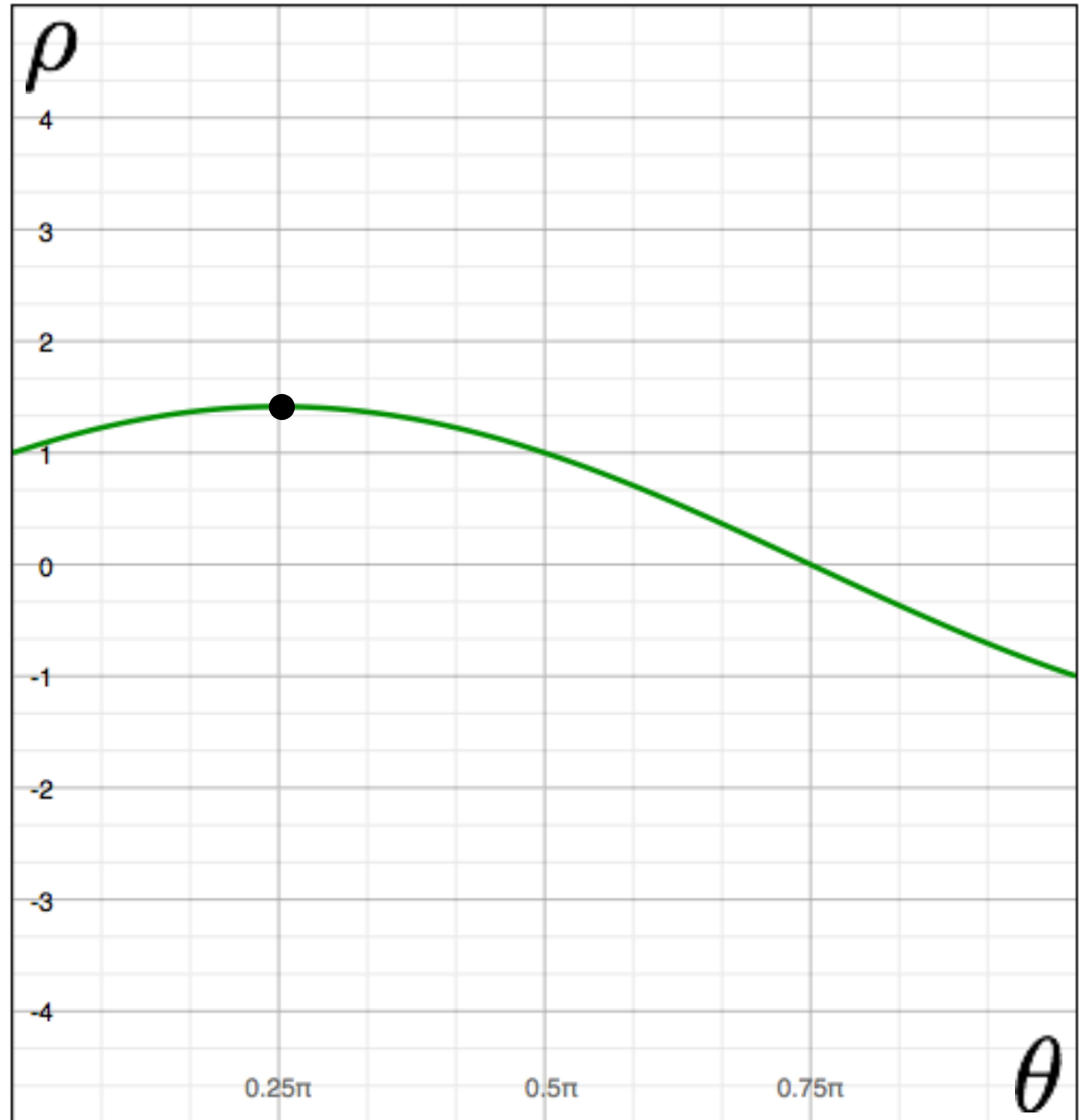
Image space

a line becomes a point

parameters

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

variables



Parameter space

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

# Hough Transform: Lines

variables

$$y = mx + b$$

parameters

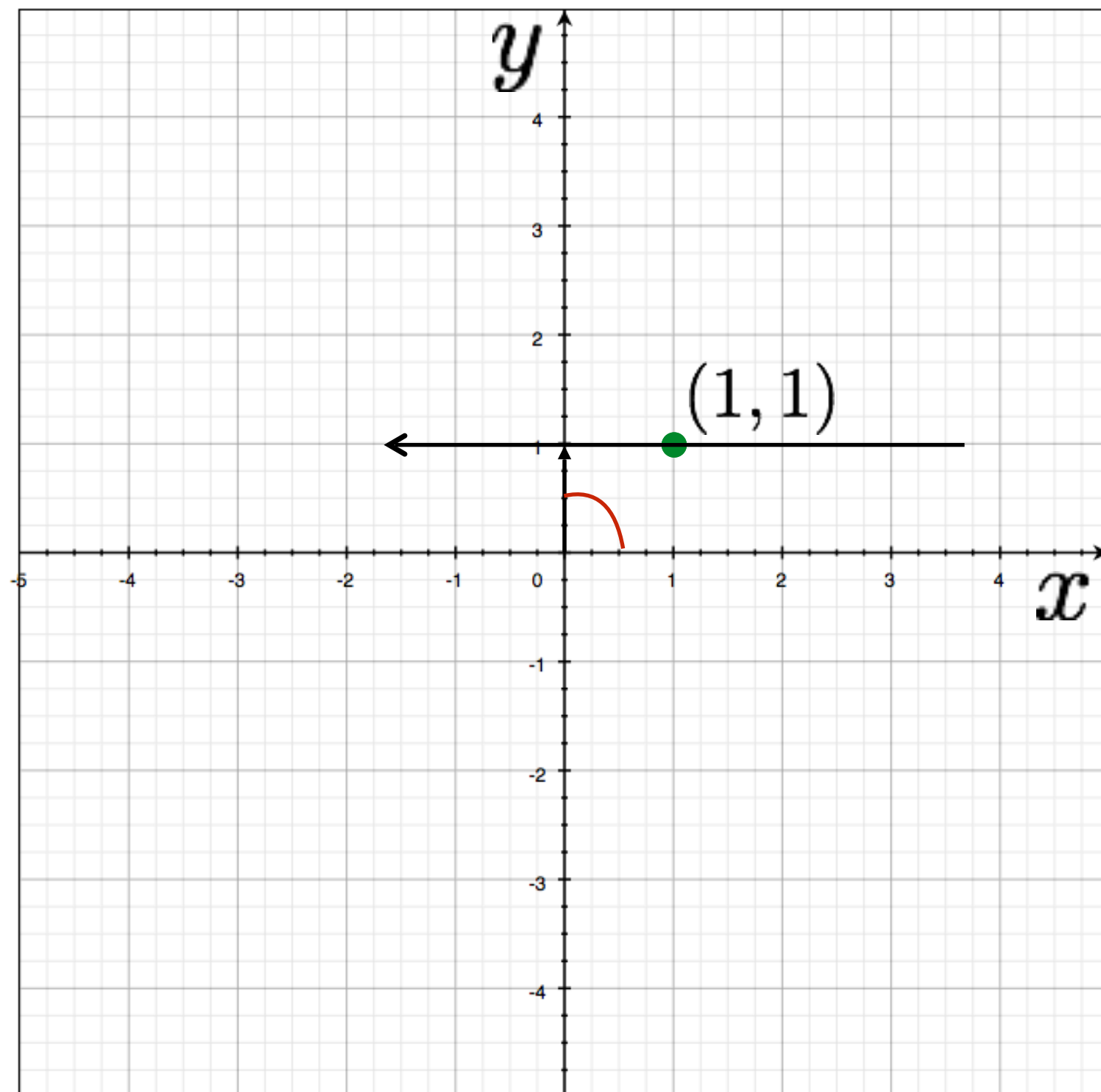


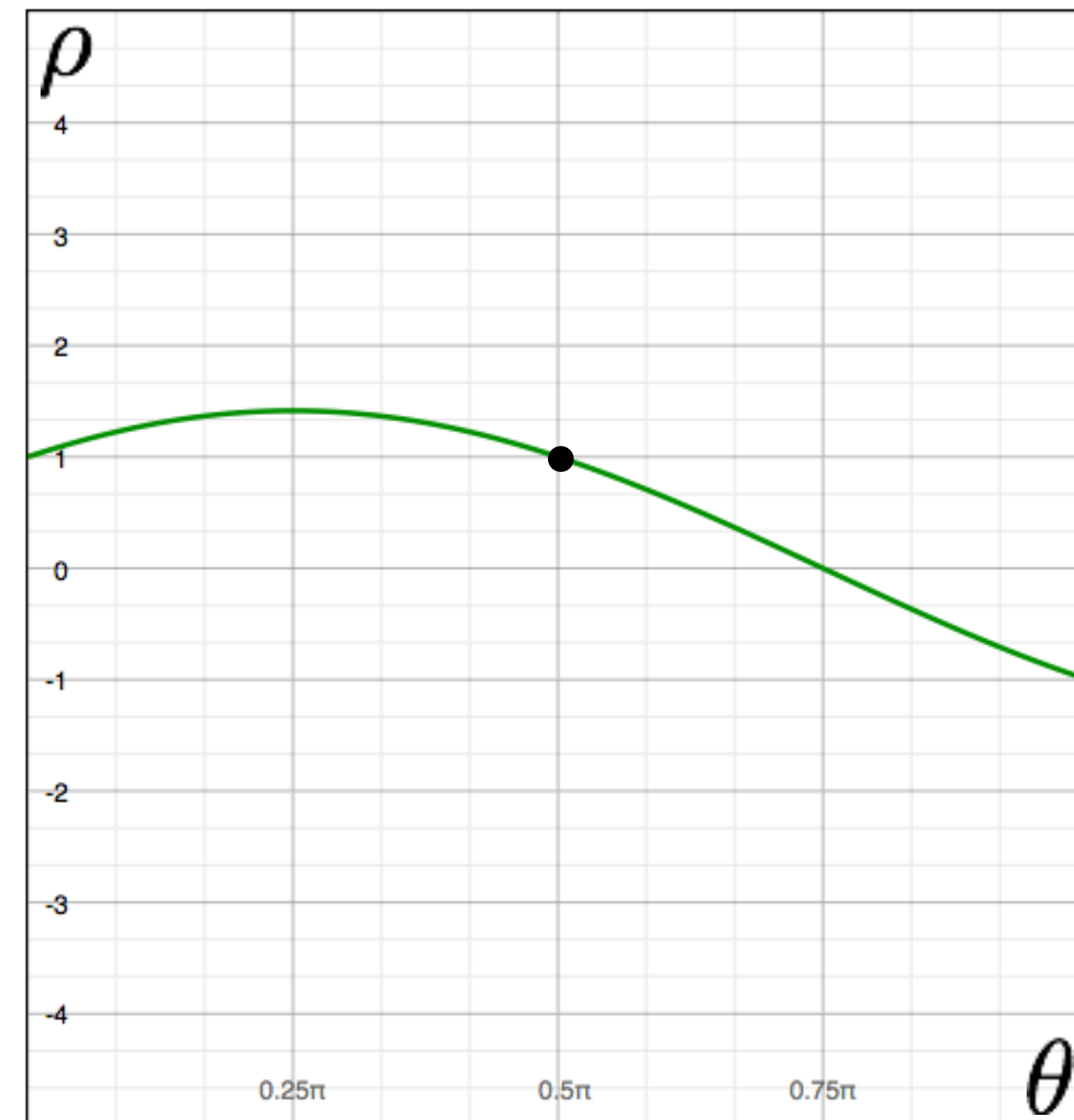
Image space

a line becomes a point

parameters

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

variables



Parameter space

# Hough Transform: Lines

variables

$$y = mx + b$$

parameters

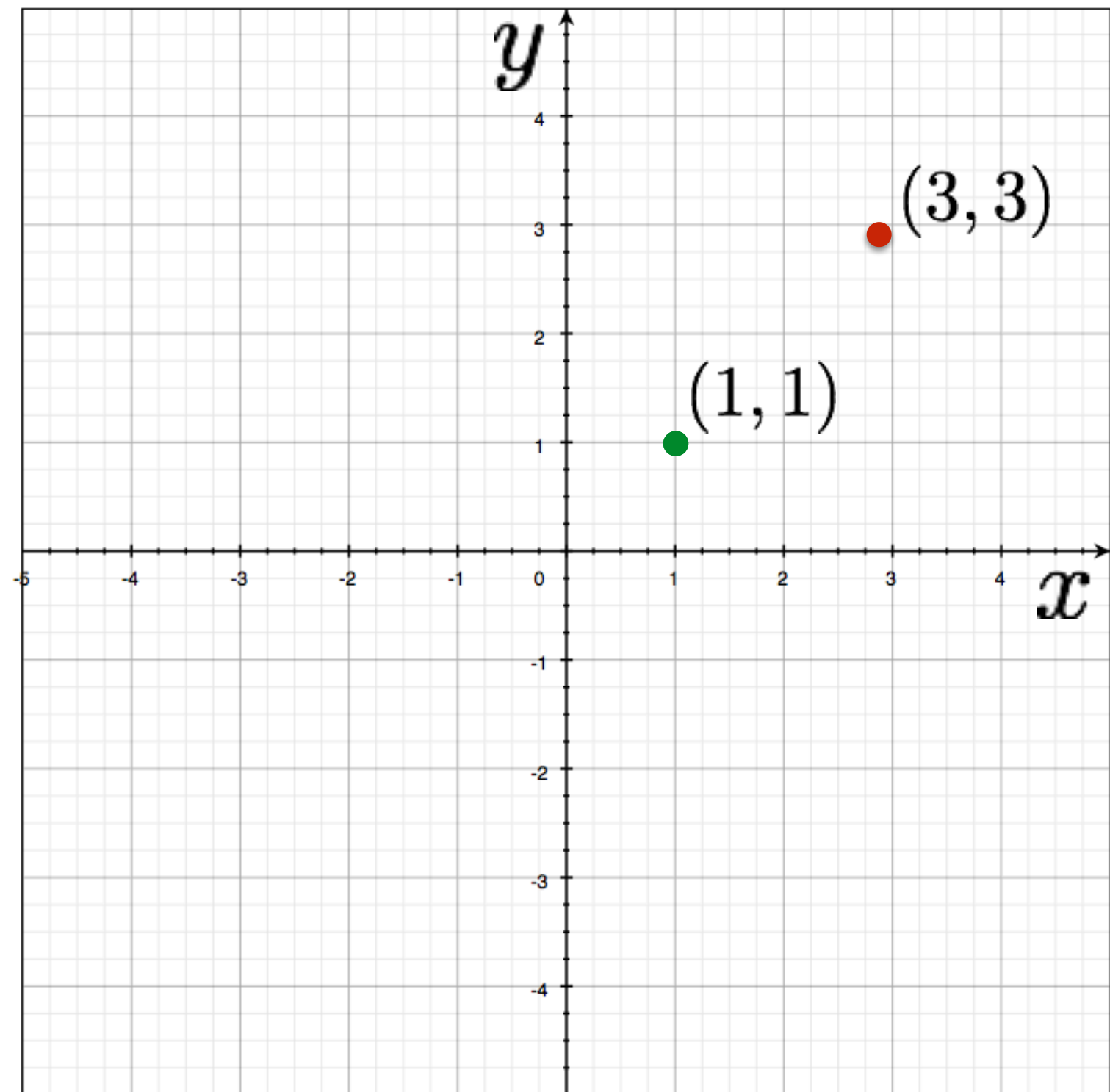


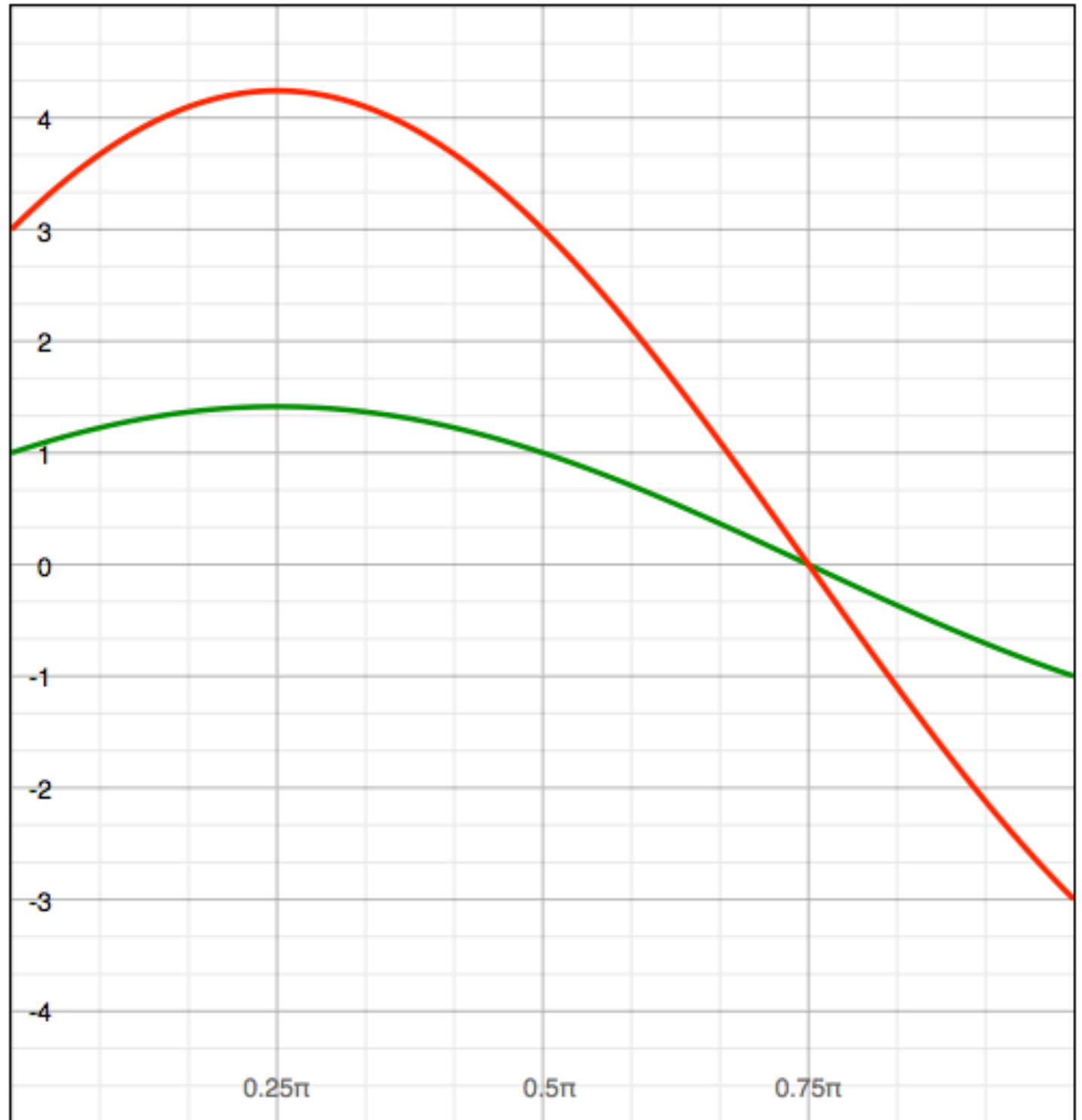
Image space

two points become?

parameters

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

variables



Parameter space



# Hough Transform: Lines

variables

$$y = mx + b$$

parameters

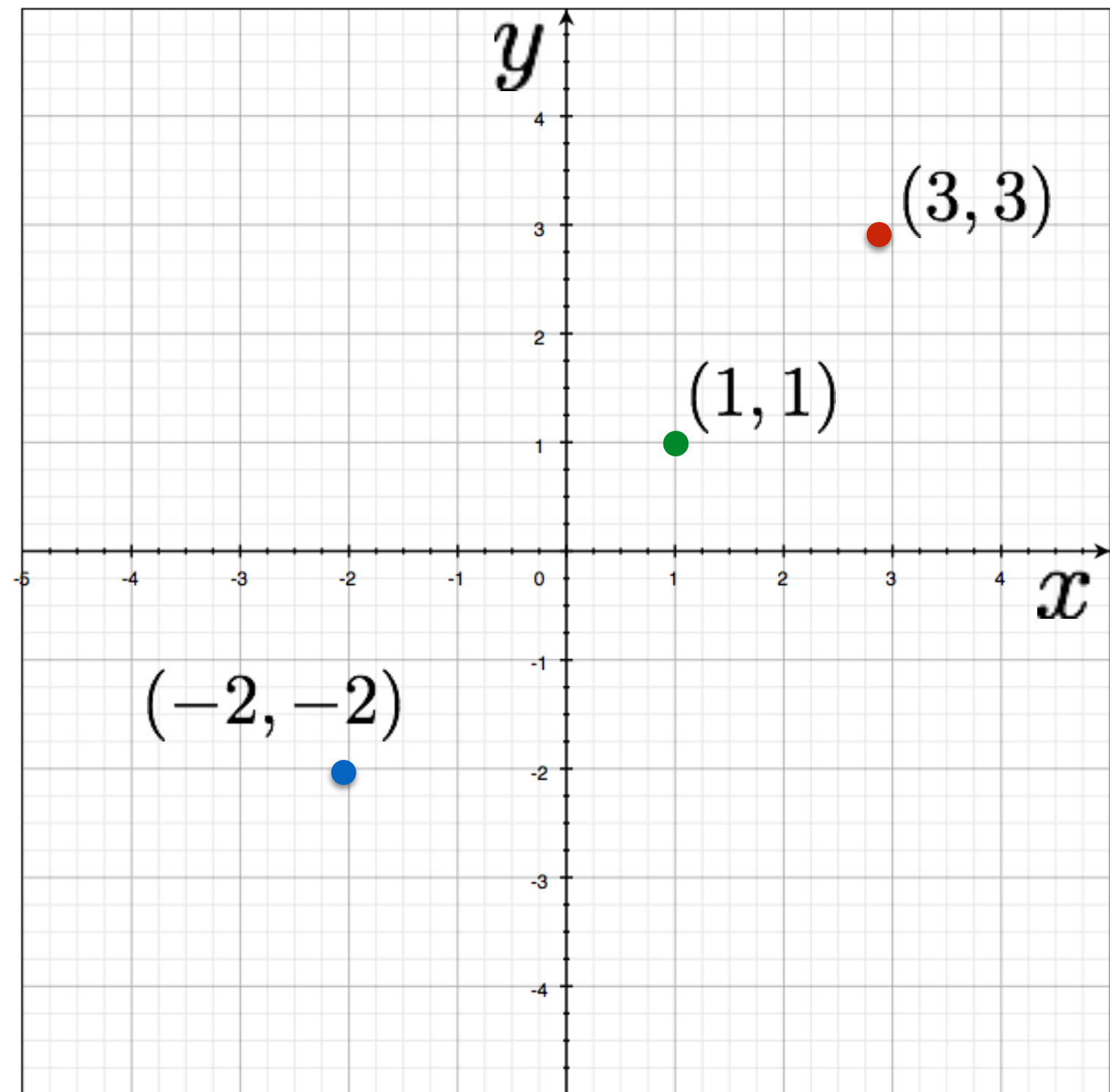


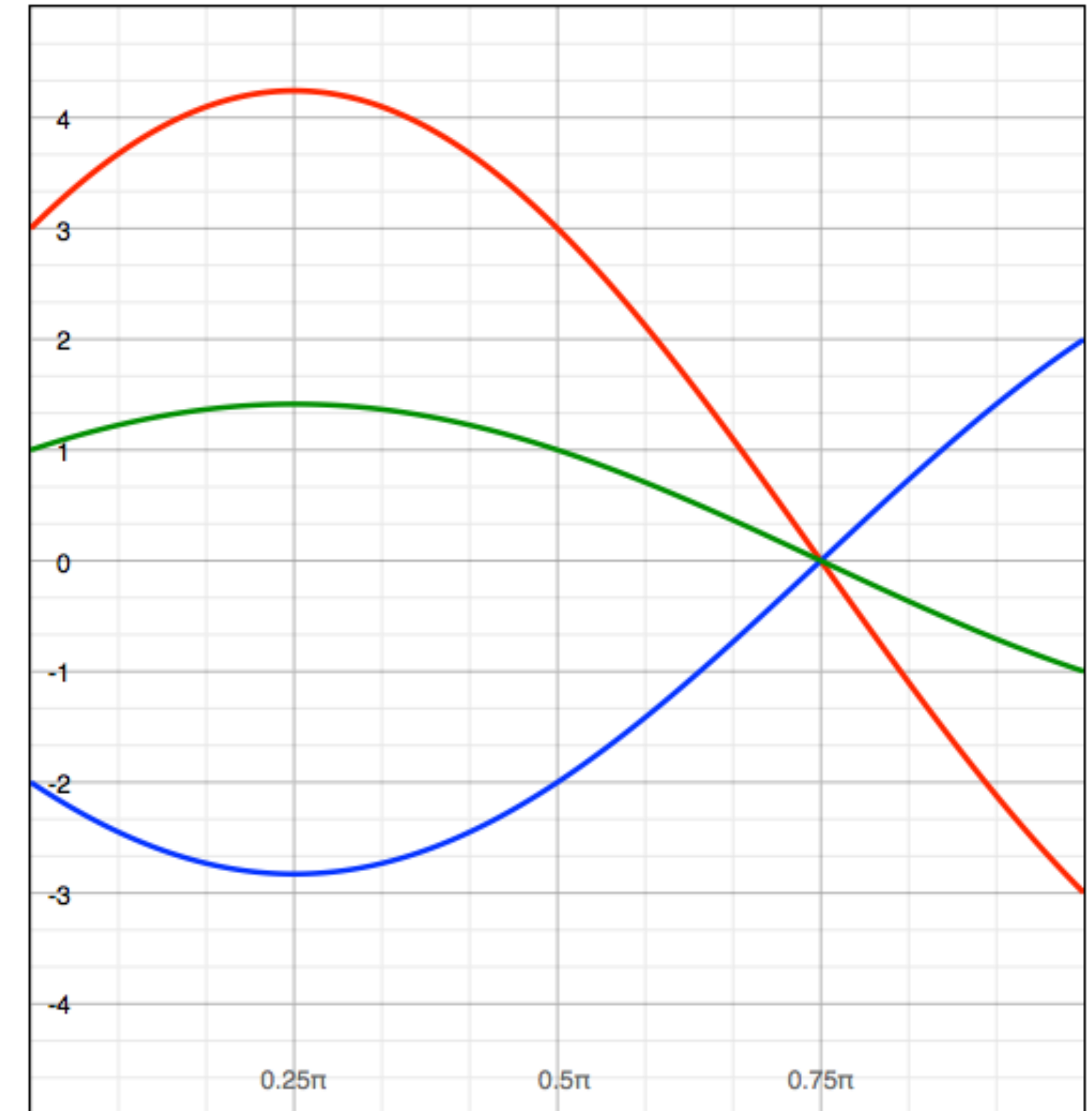
Image space

three points become?

parameters

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

variables



Parameter space

# Hough Transform: Lines

variables

$$y = mx + b$$

parameters

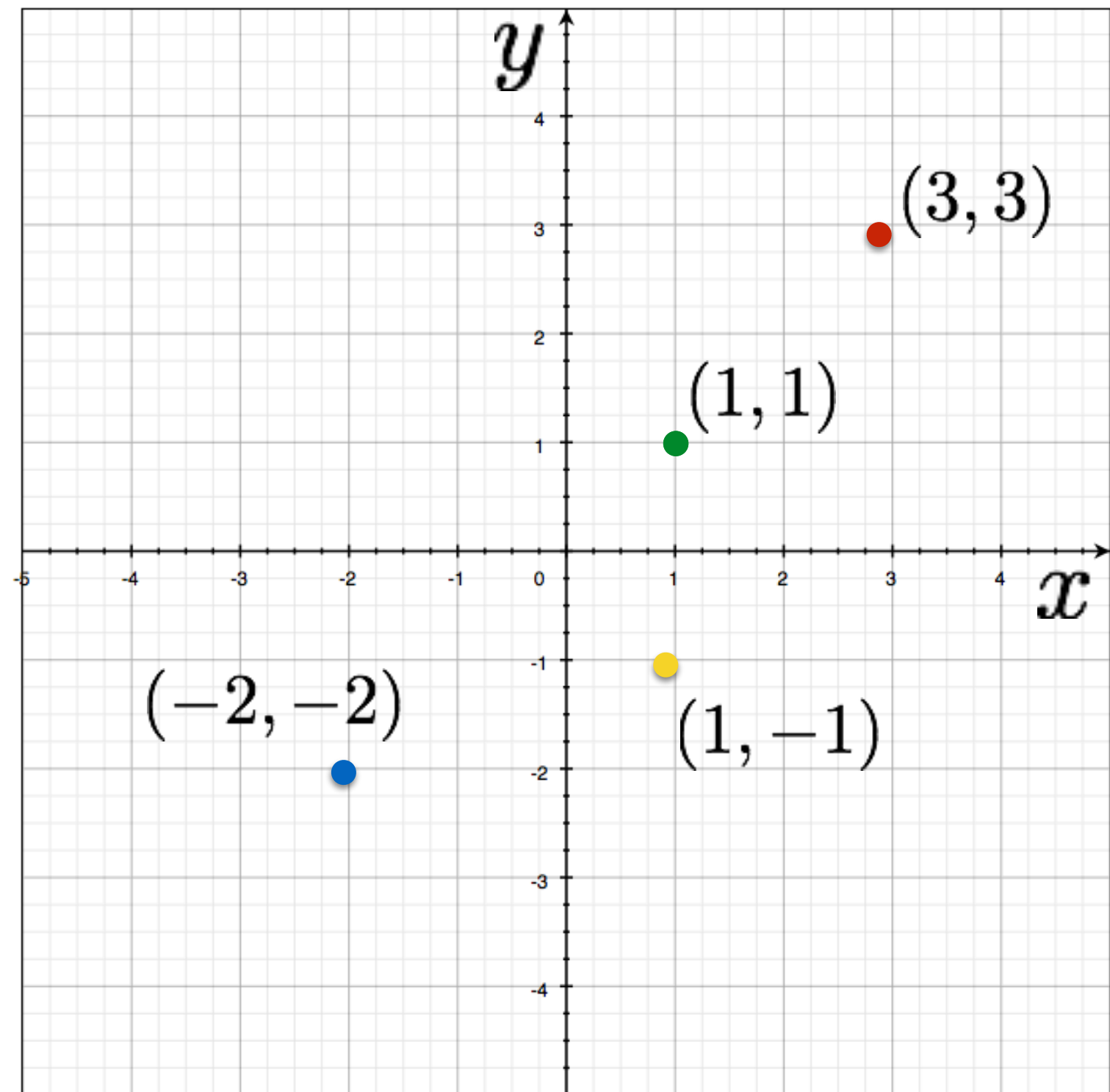


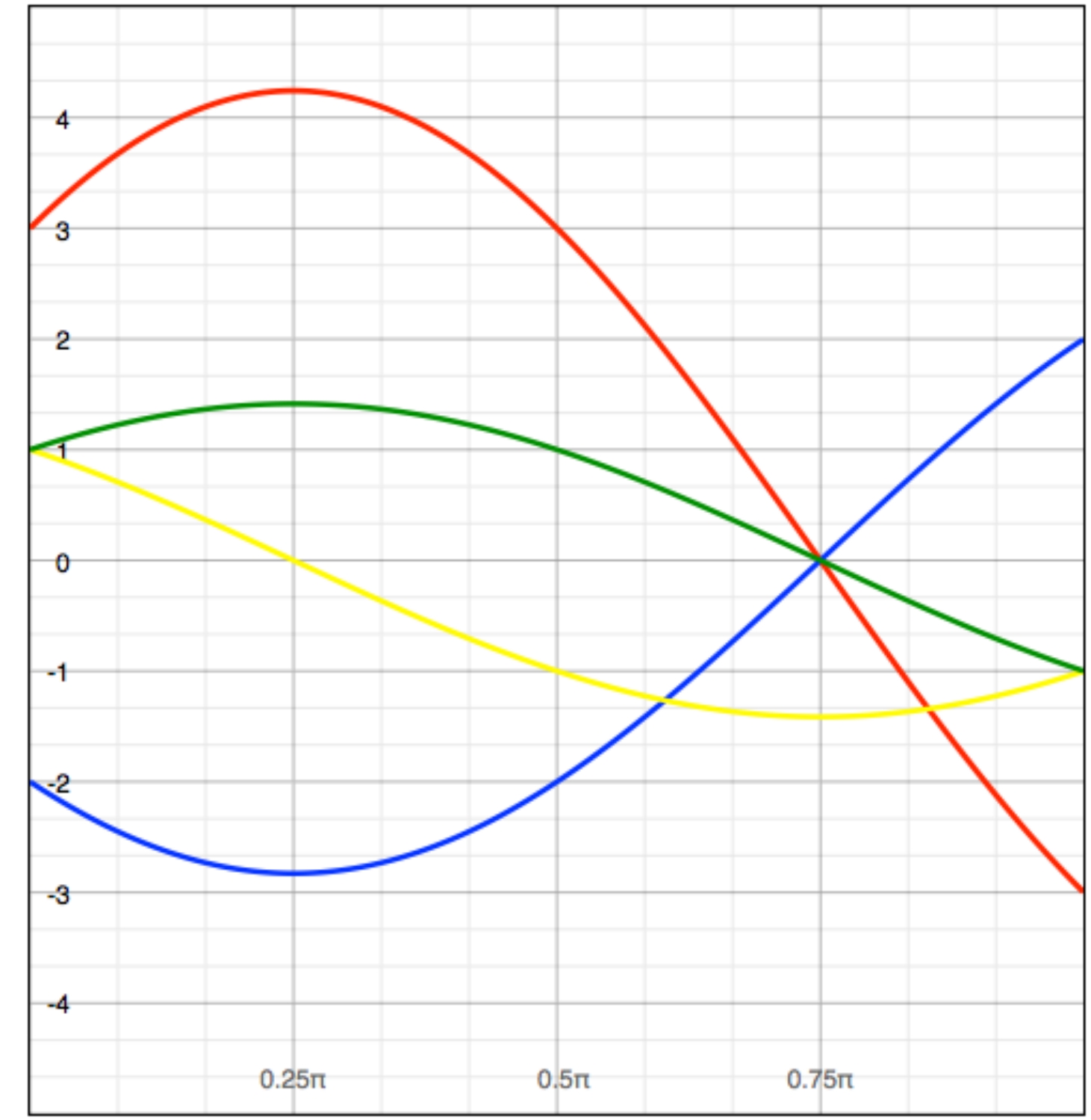
Image space

four points become?

parameters

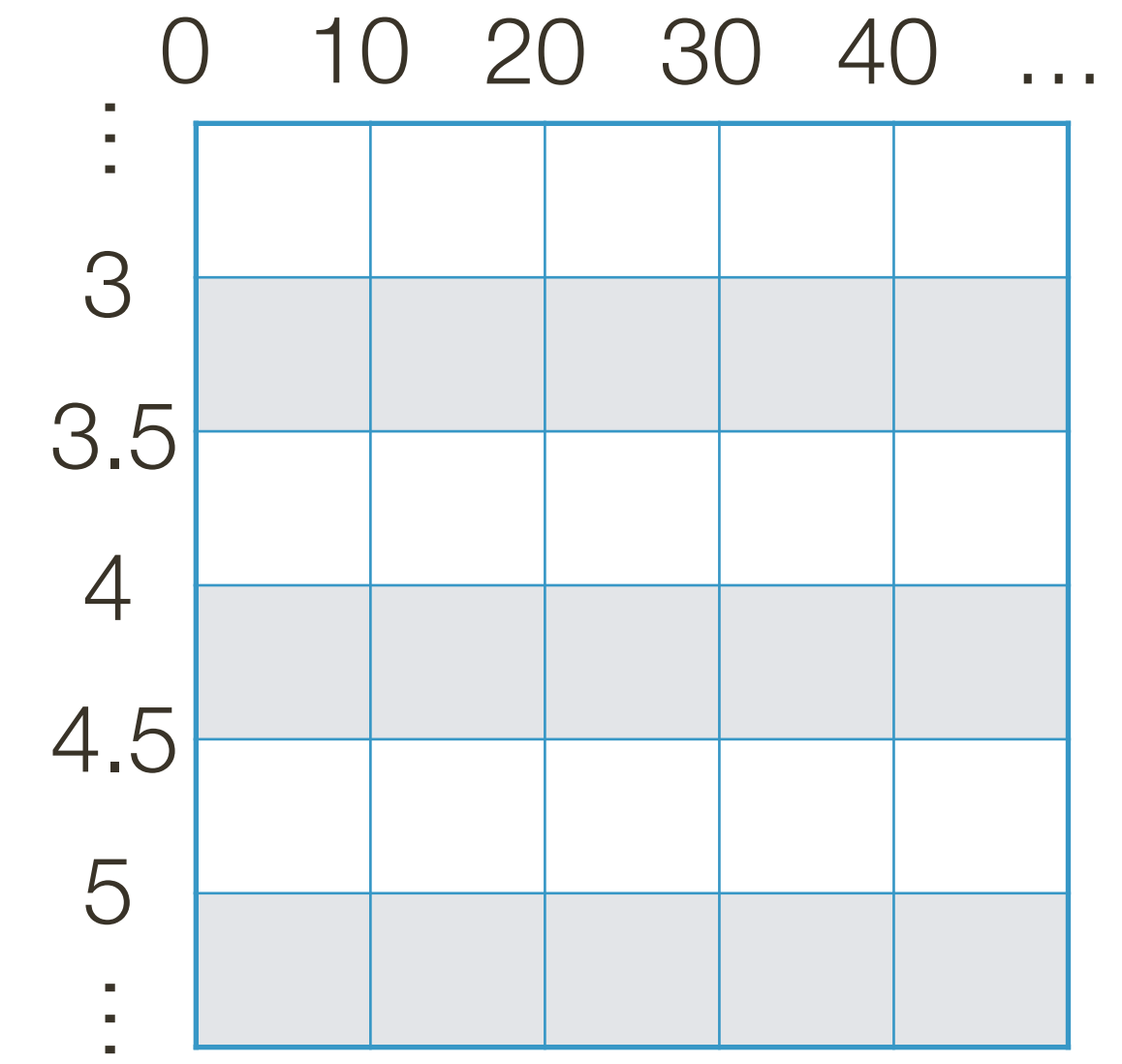
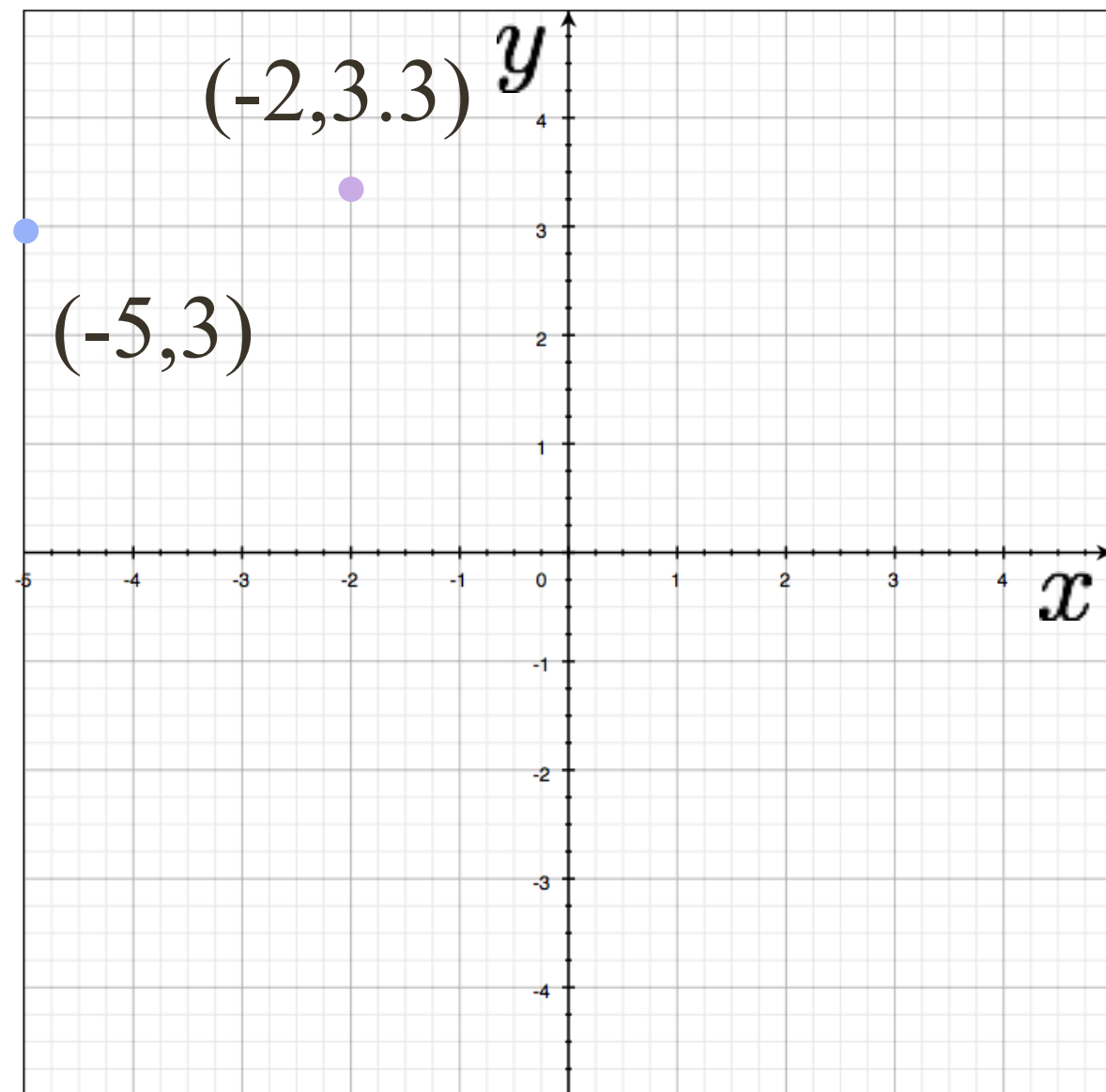
$$x \sin \theta + y \cos \theta = \rho$$

variables



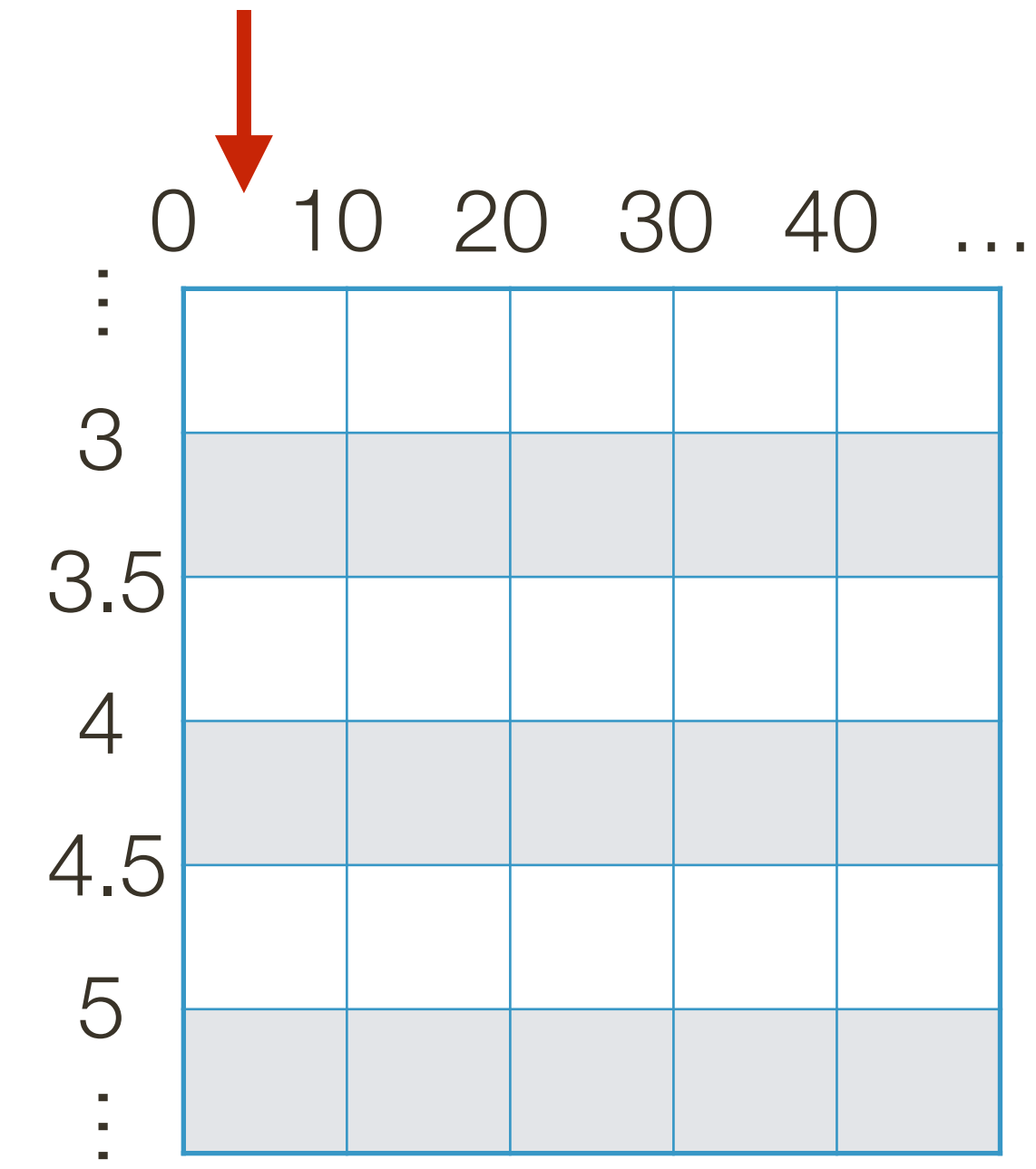
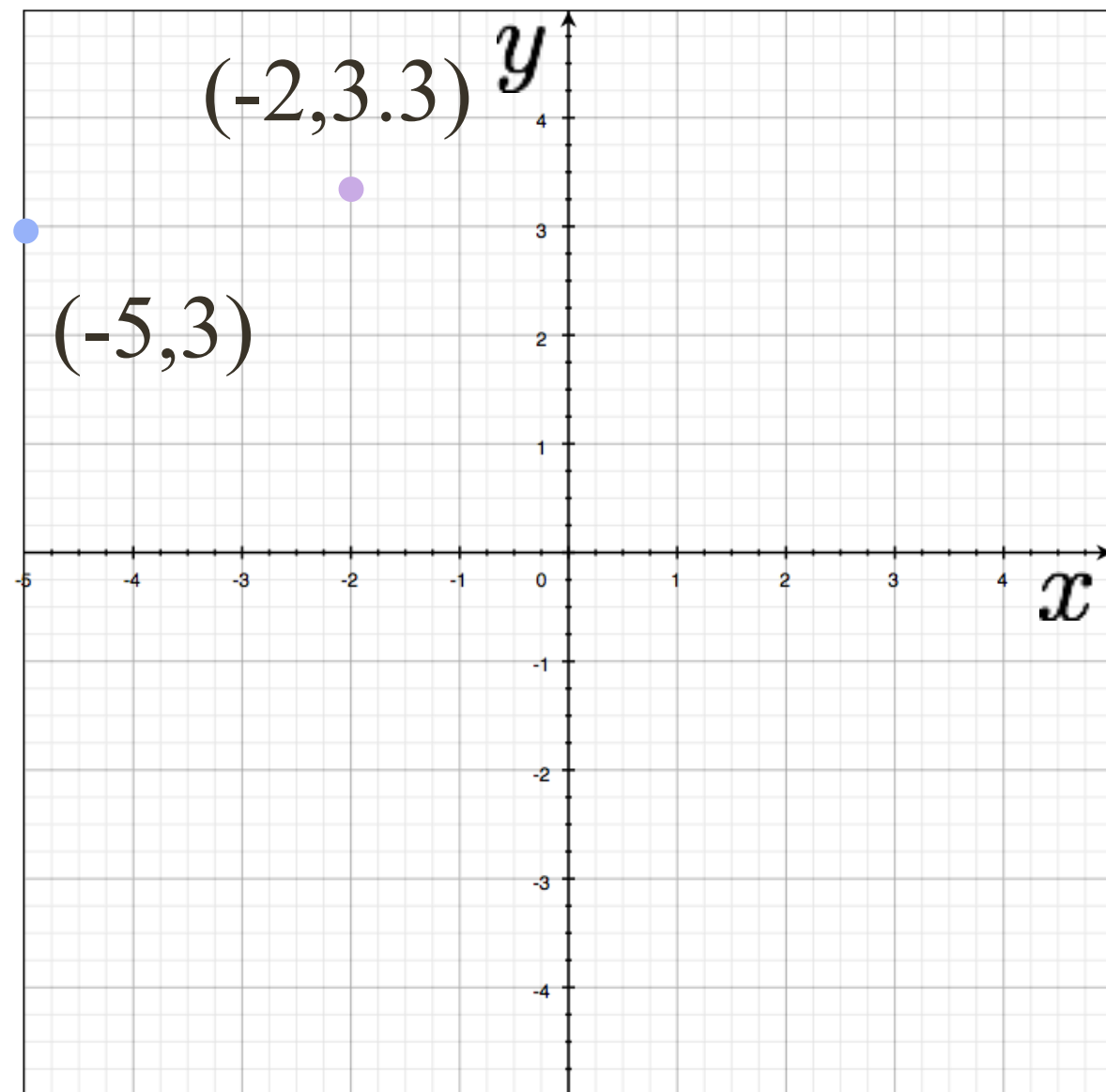
Parameter space

# Example: Hough Transform for Lines



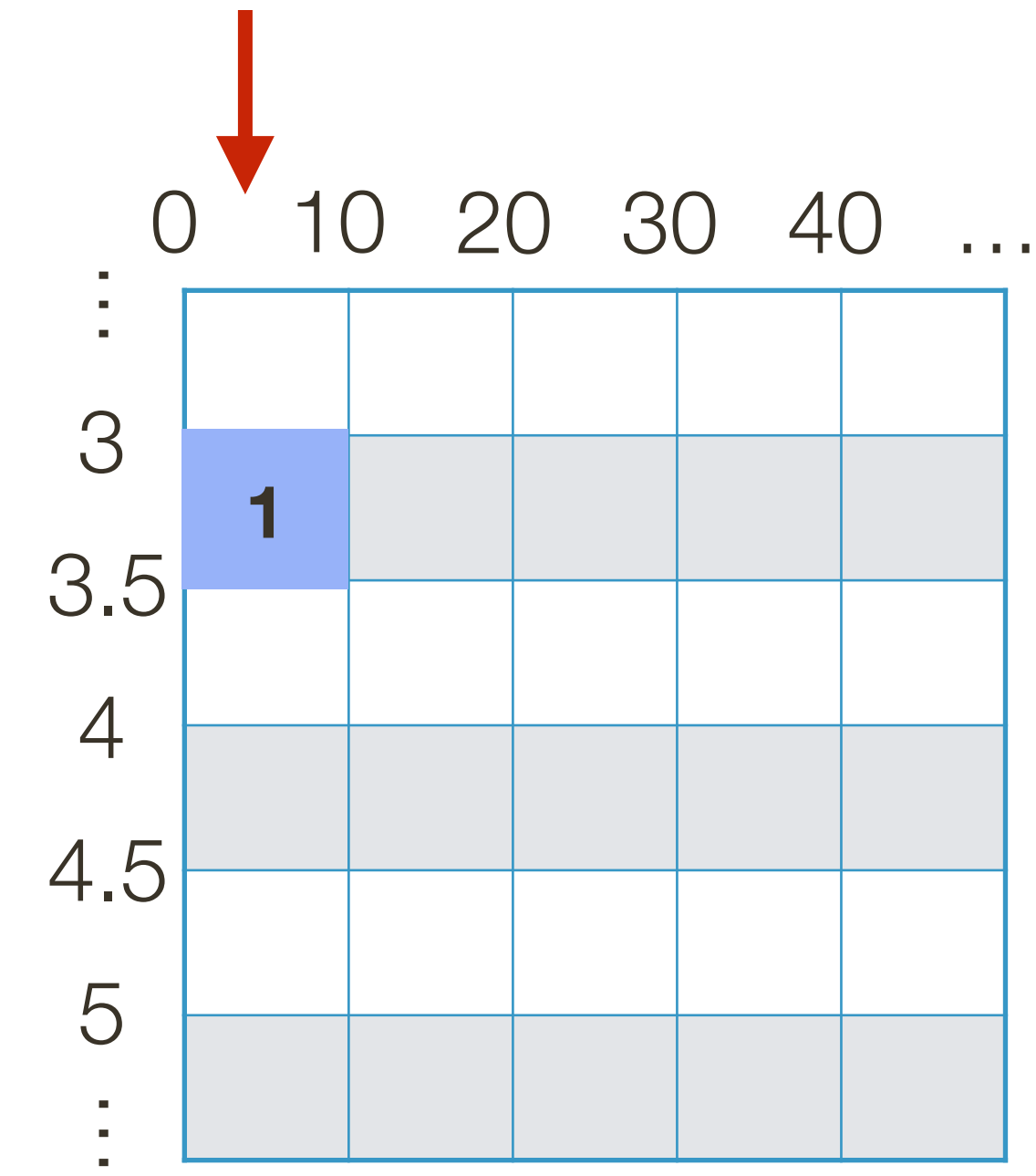
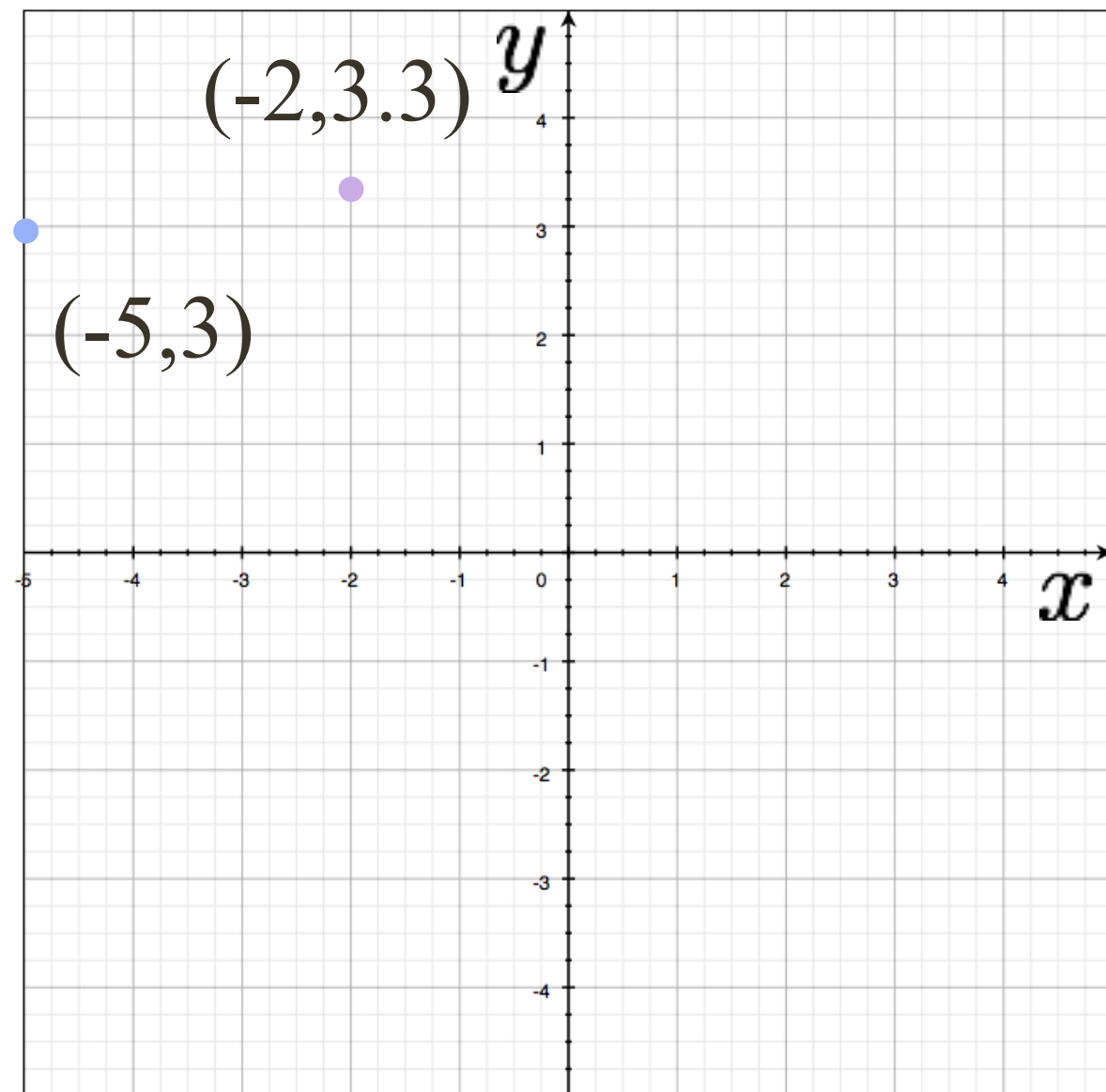


# Example: Hough Transform for Lines



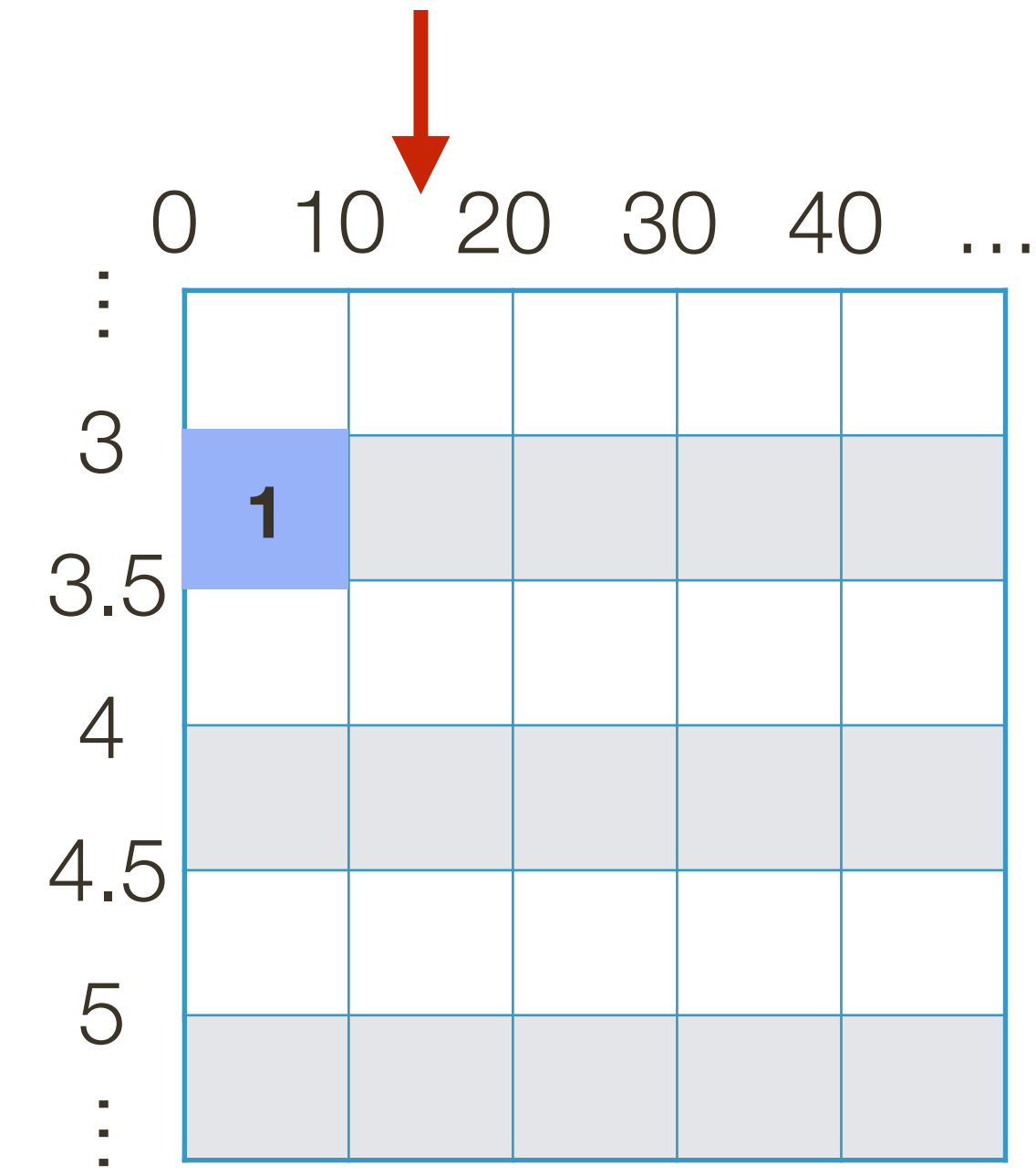
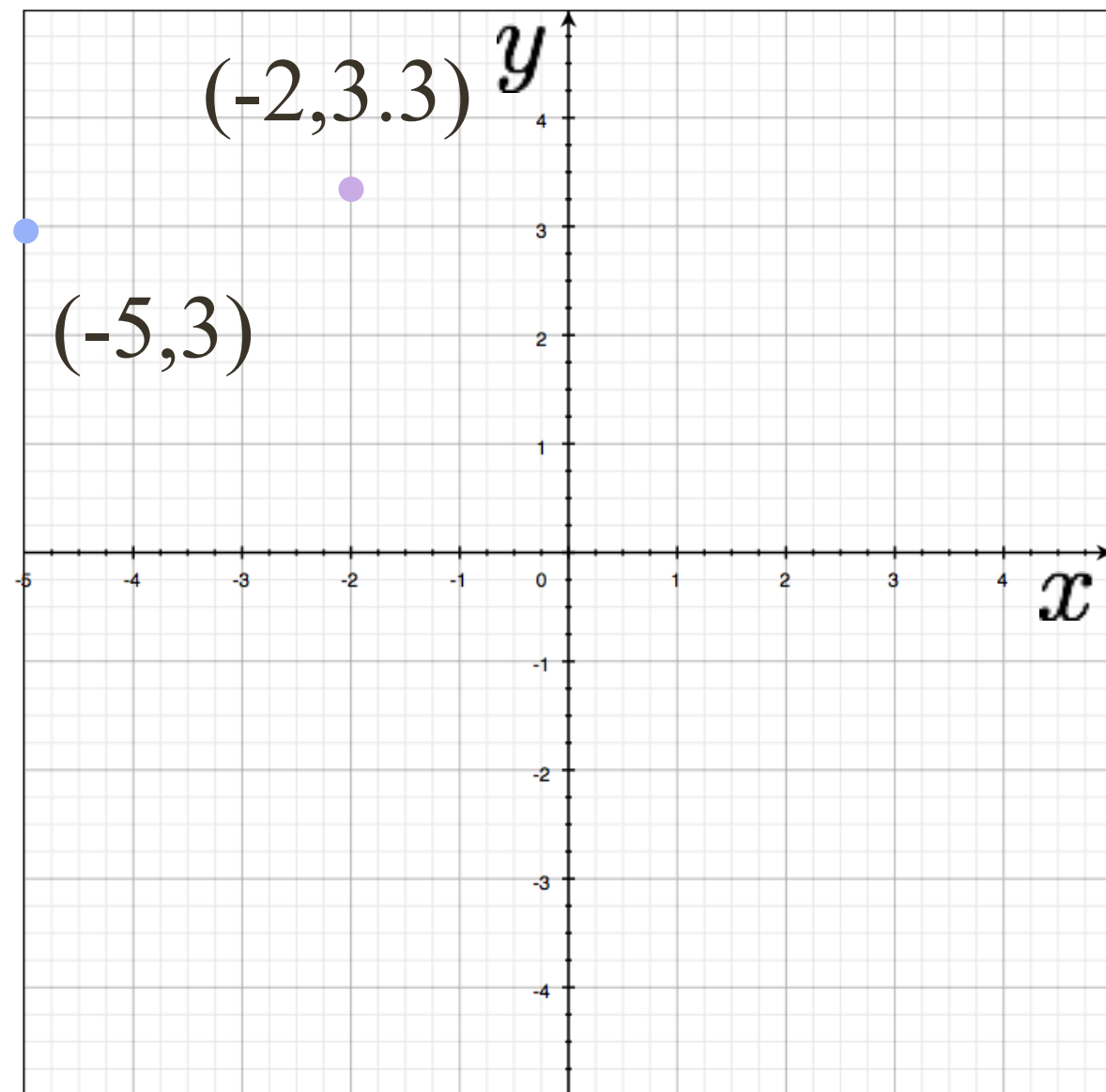
$$-5 \sin(5^\circ) - 3 \cos(5^\circ) + r = 0 \Rightarrow r = 3.42$$

# Example: Hough Transform for Lines



$$-5 \sin(5^\circ) - 3 \cos(5^\circ) + r = 0 \Rightarrow r = 3.42$$

# Example: Hough Transform for Lines

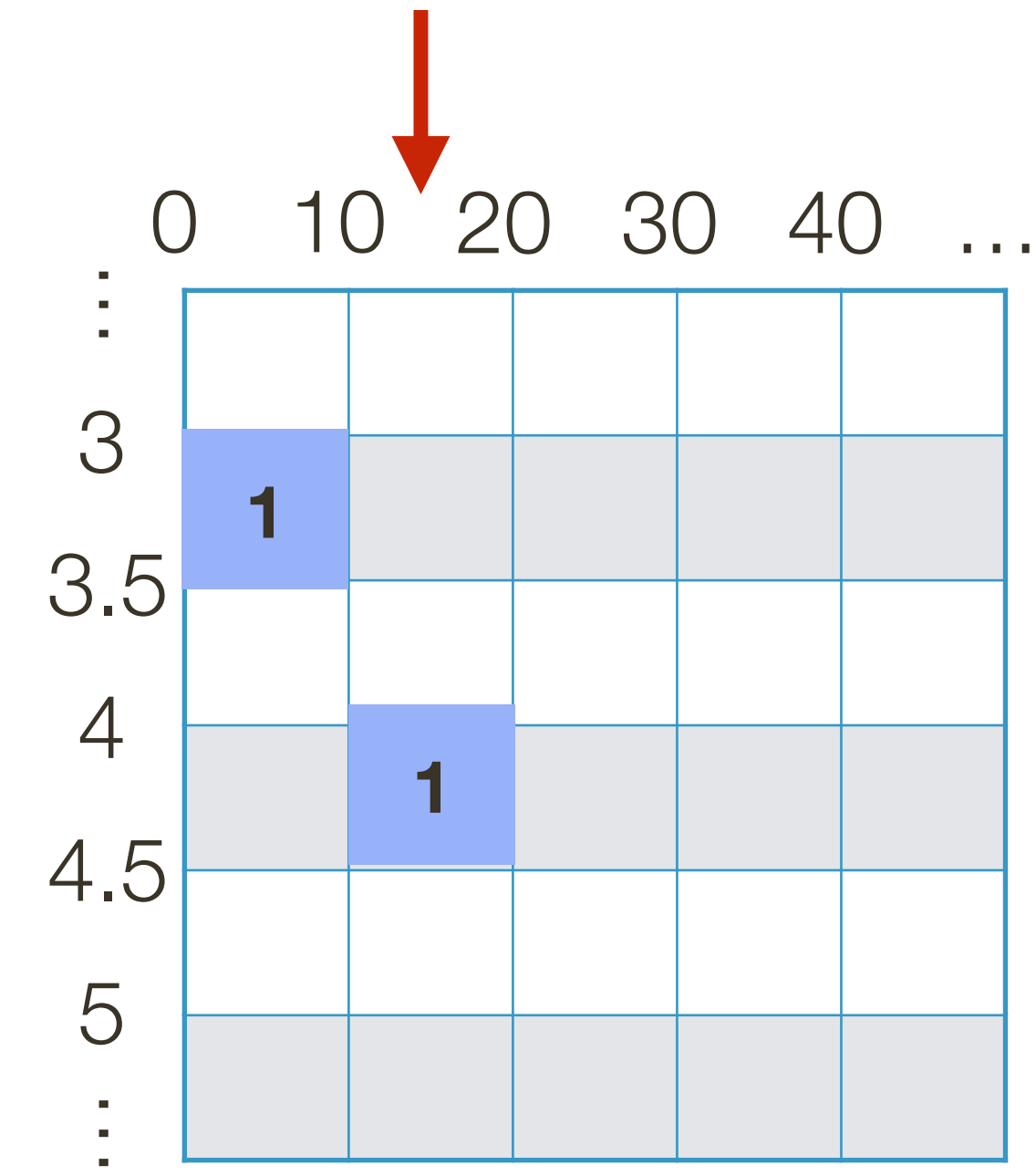
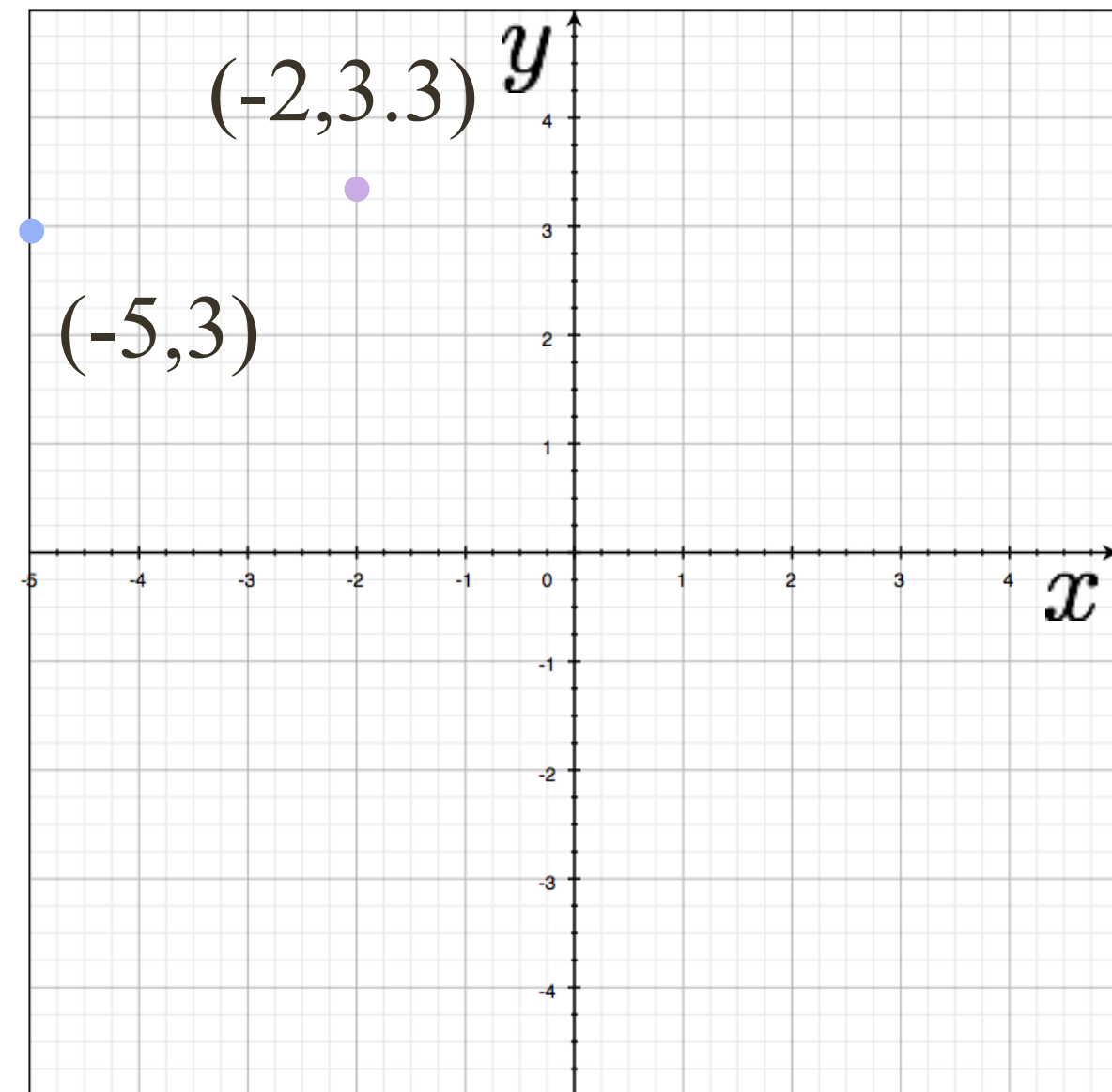


$$-5 \sin(5^\circ) - 3 \cos(5^\circ) + r = 0 \Rightarrow r = 3.42$$

$$-5 \sin(15^\circ) - 3 \cos(15^\circ) + r = 0 \Rightarrow r = 4.18$$



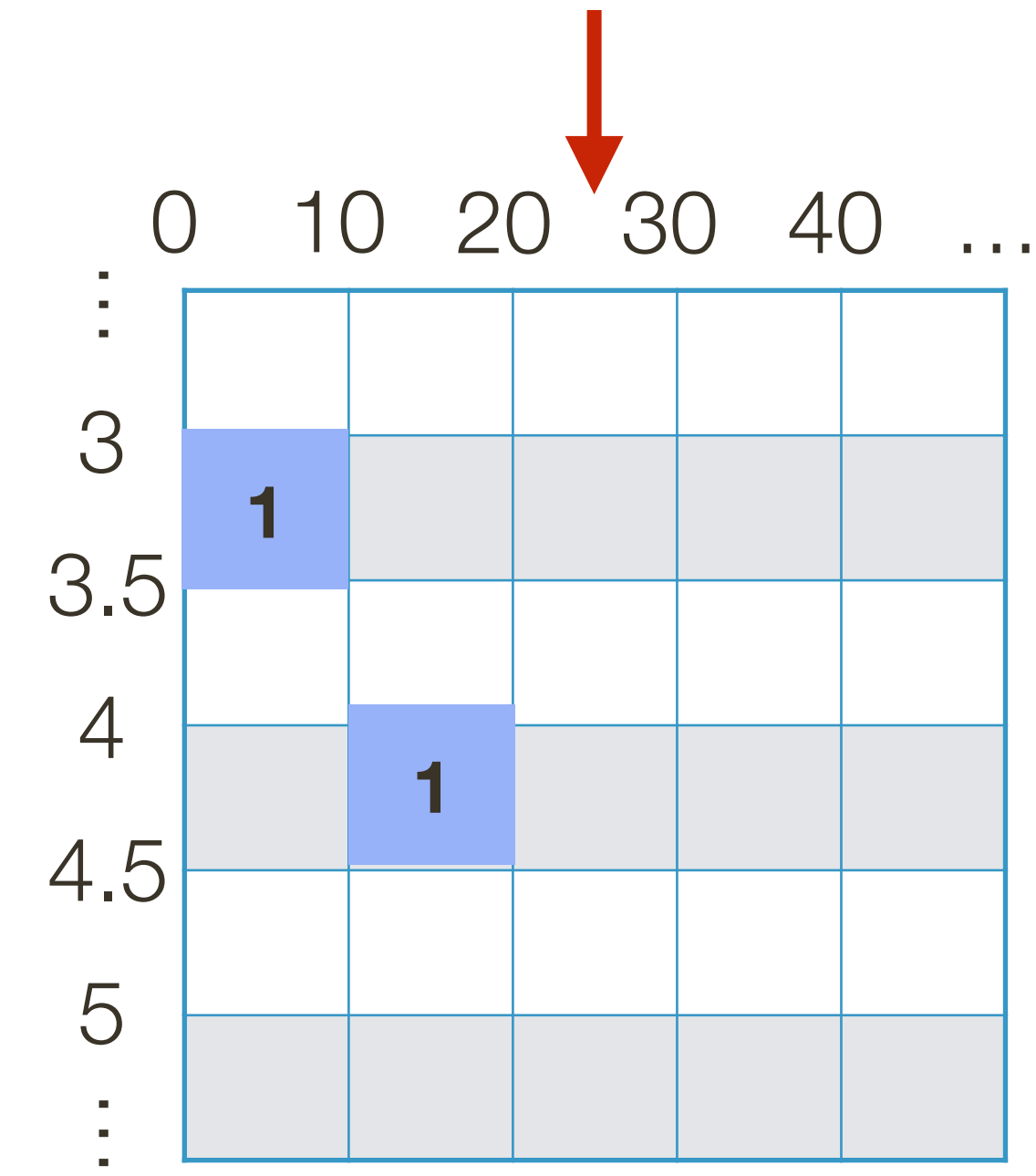
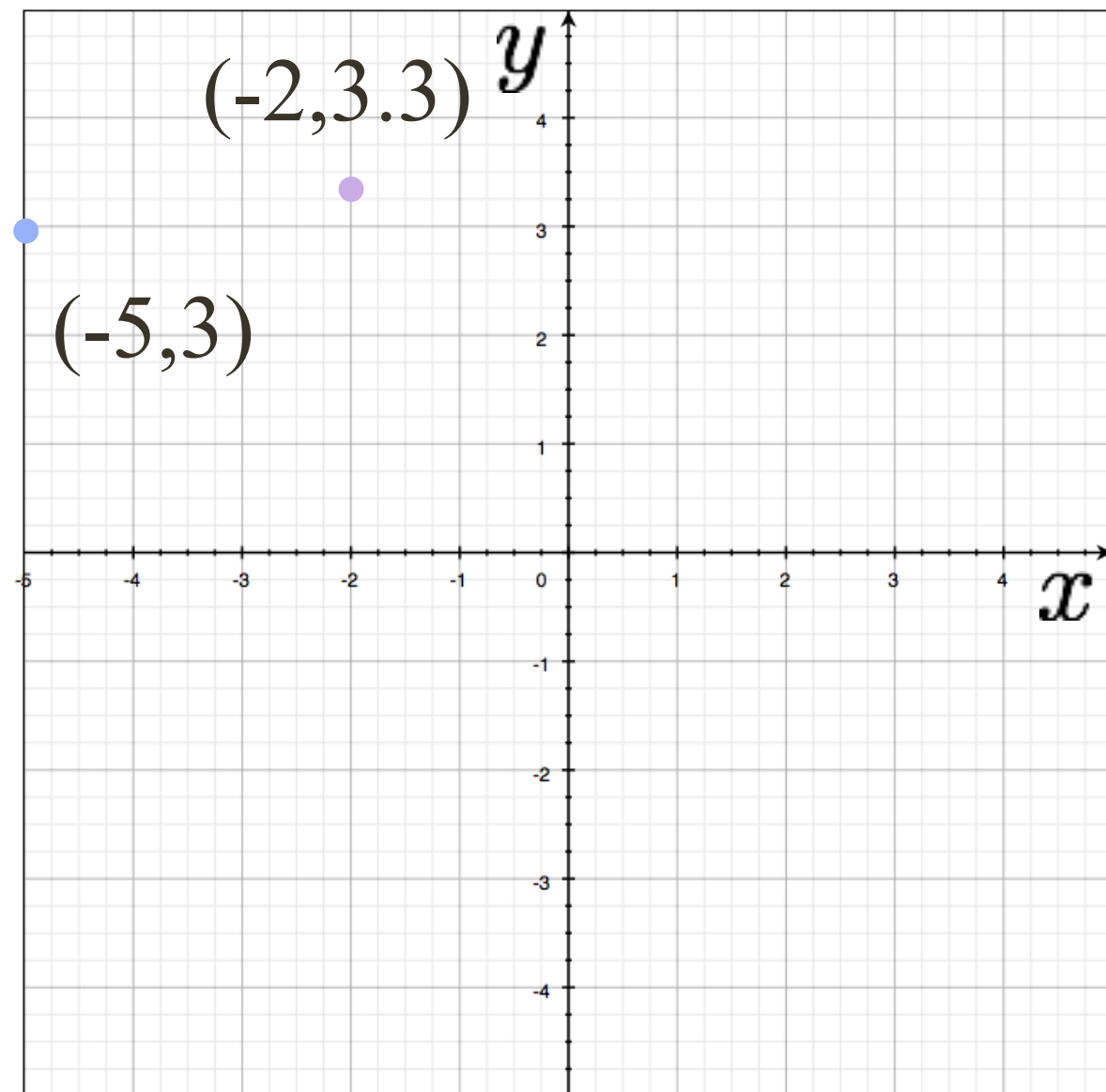
# Example: Hough Transform for Lines



$$-5 \sin(5^\circ) - 3 \cos(5^\circ) + r = 0 \Rightarrow r = 3.42$$

$$-5 \sin(15^\circ) - 3 \cos(15^\circ) + r = 0 \Rightarrow r = 4.18$$

# Example: Hough Transform for Lines

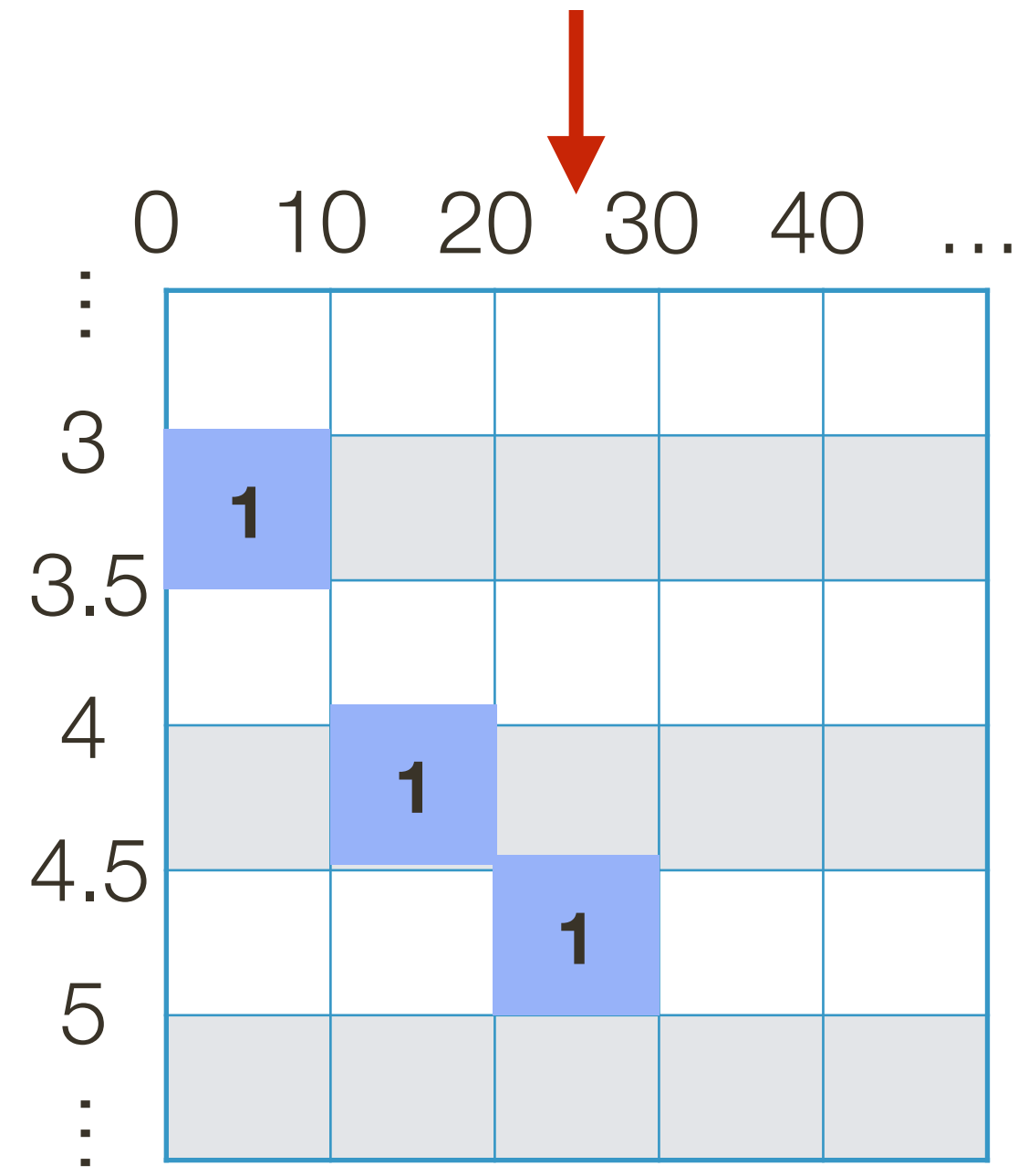
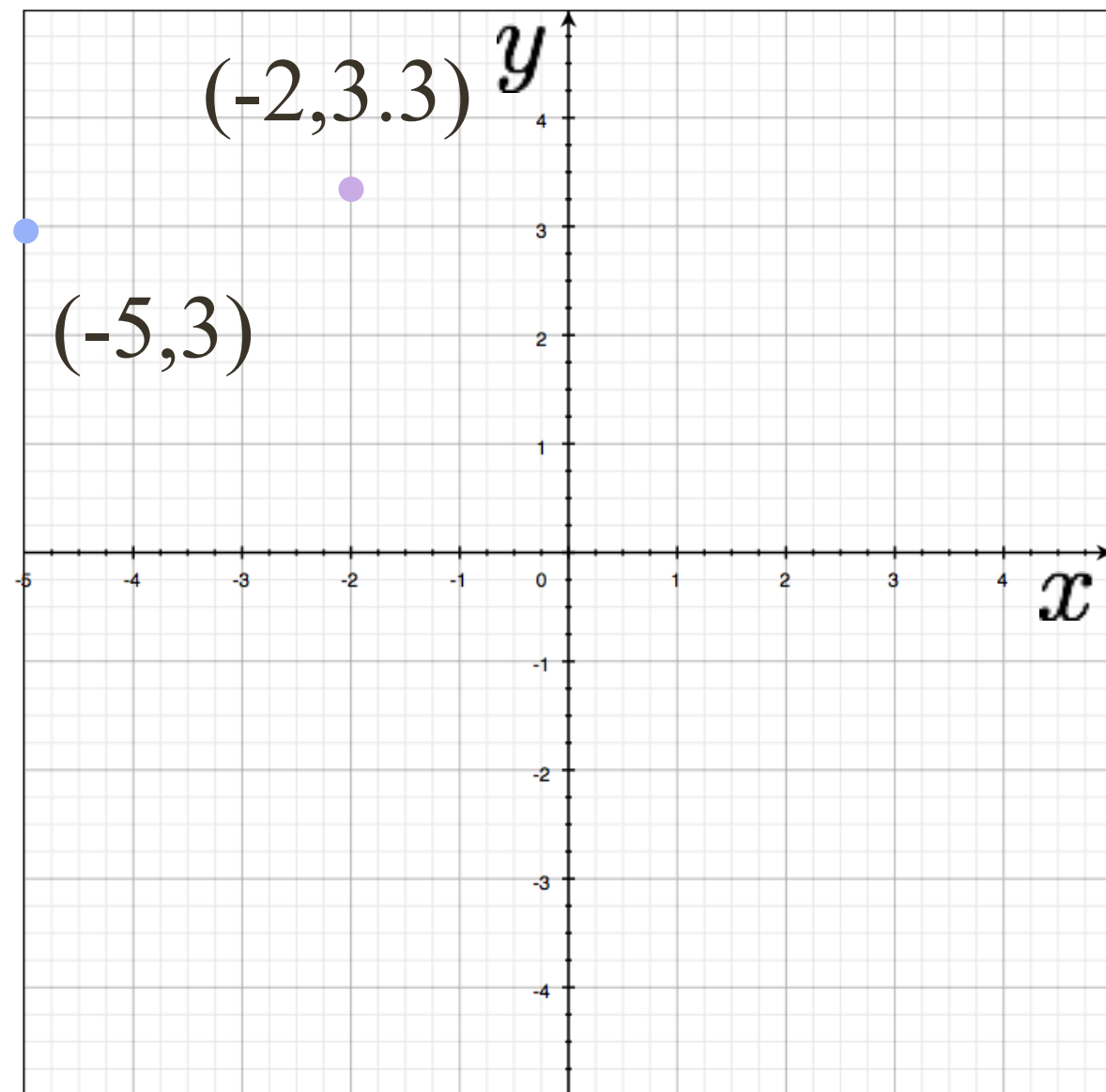


$$-5 \sin(5^\circ) - 3 \cos(5^\circ) + r = 0 \Rightarrow r = 3.42$$

$$-5 \sin(15^\circ) - 3 \cos(15^\circ) + r = 0 \Rightarrow r = 4.18$$

$$-5 \sin(25^\circ) - 3 \cos(25^\circ) + r = 0 \Rightarrow r = 4.83$$

# Example: Hough Transform for Lines

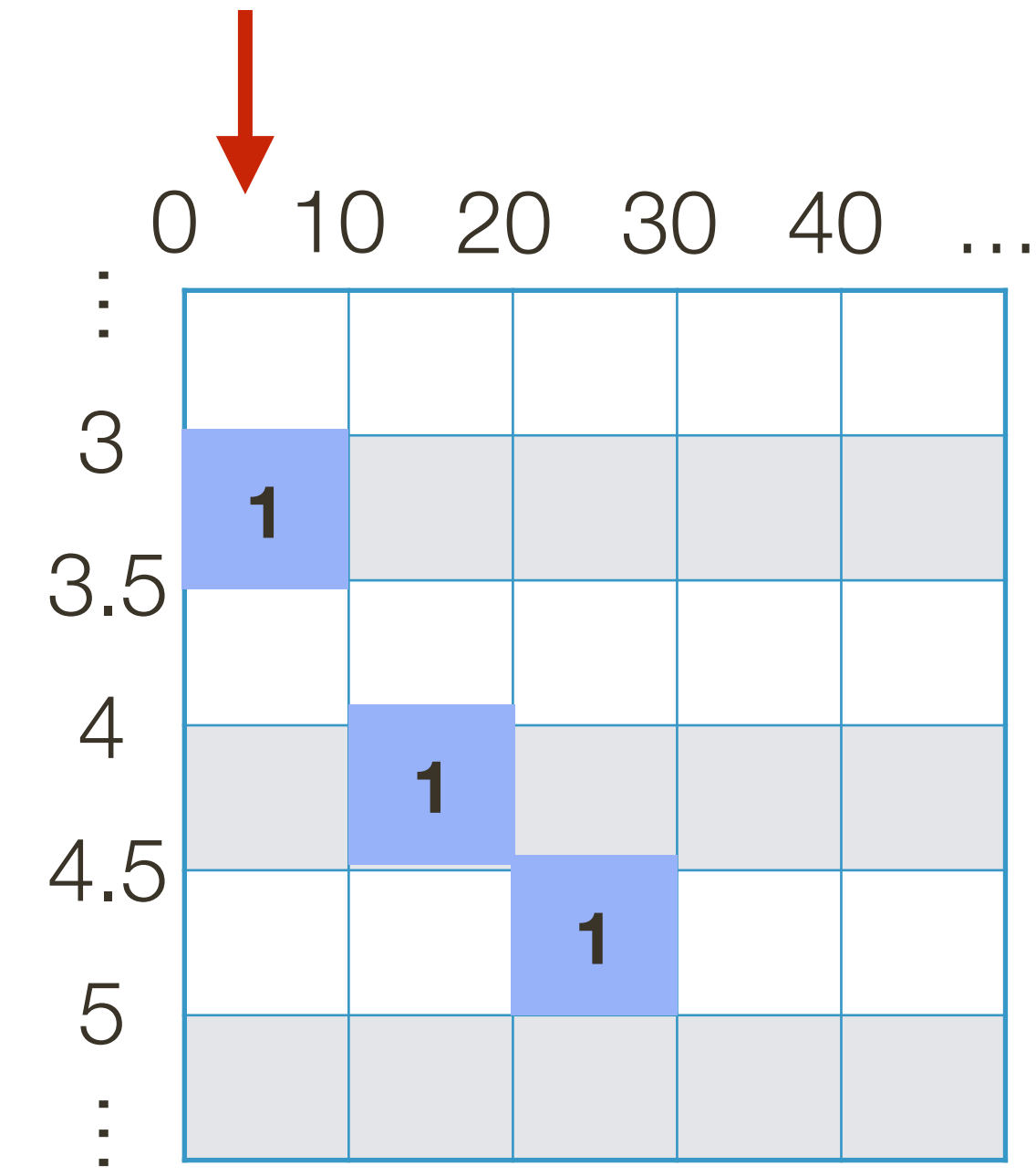
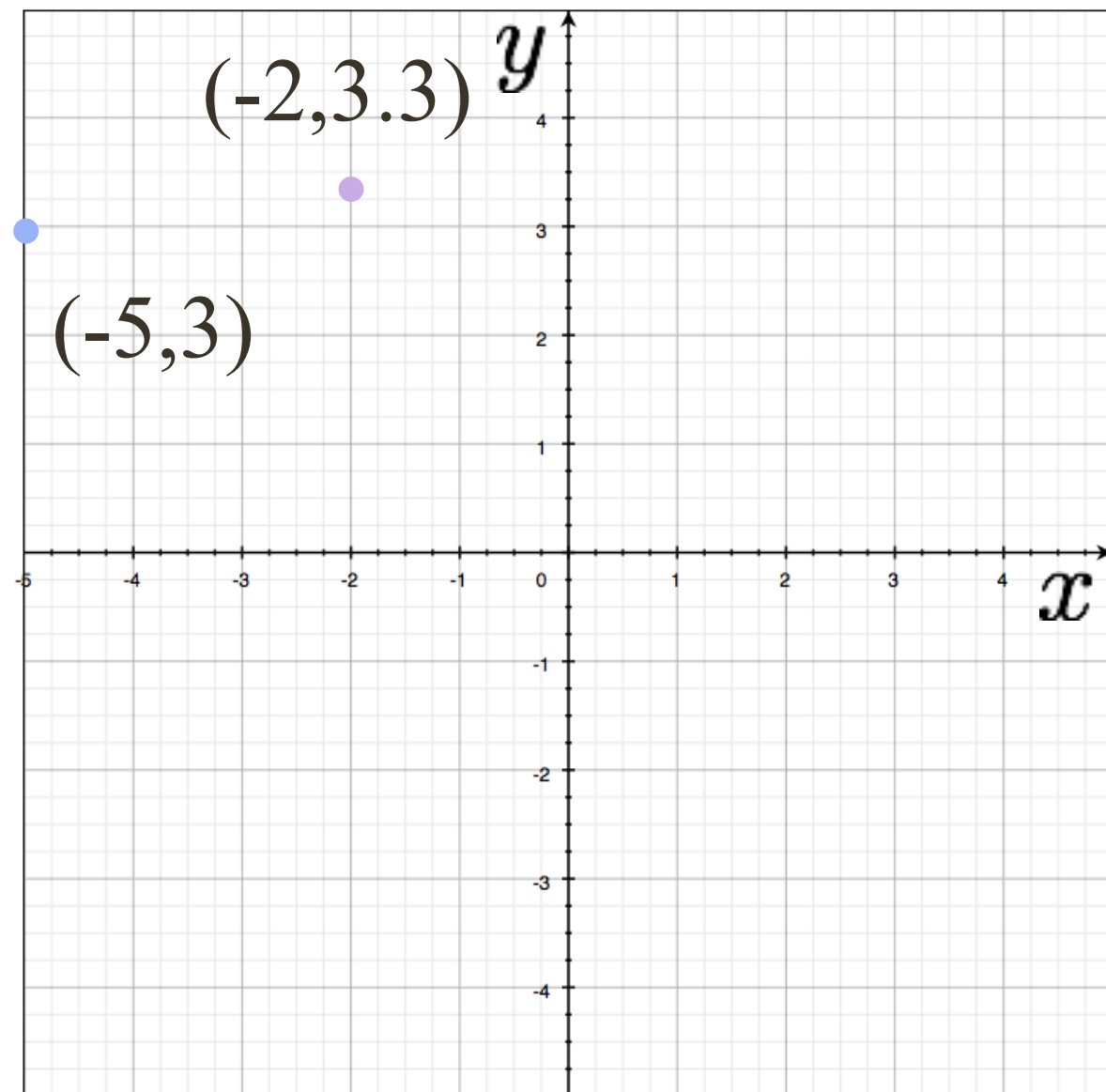


$$-5 \sin(5^\circ) - 3 \cos(5^\circ) + r = 0 \Rightarrow r = 3.42$$

$$-5 \sin(15^\circ) - 3 \cos(15^\circ) + r = 0 \Rightarrow r = 4.18$$

$$-5 \sin(25^\circ) - 3 \cos(25^\circ) + r = 0 \Rightarrow r = 4.83$$

# Example: Hough Transform for Lines



$$-5 \sin(5^\circ) - 3 \cos(5^\circ) + r = 0 \Rightarrow r = 3.42$$

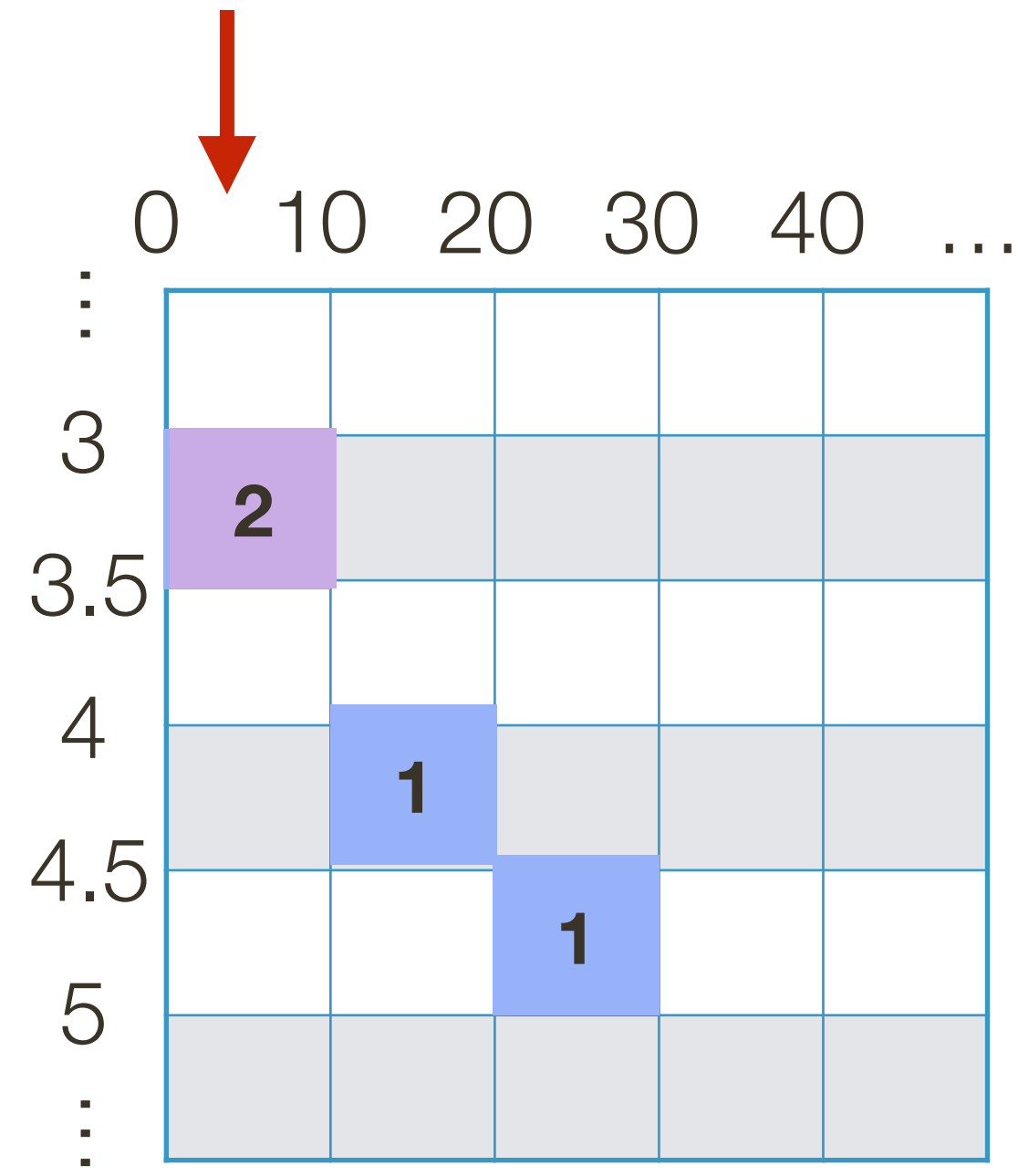
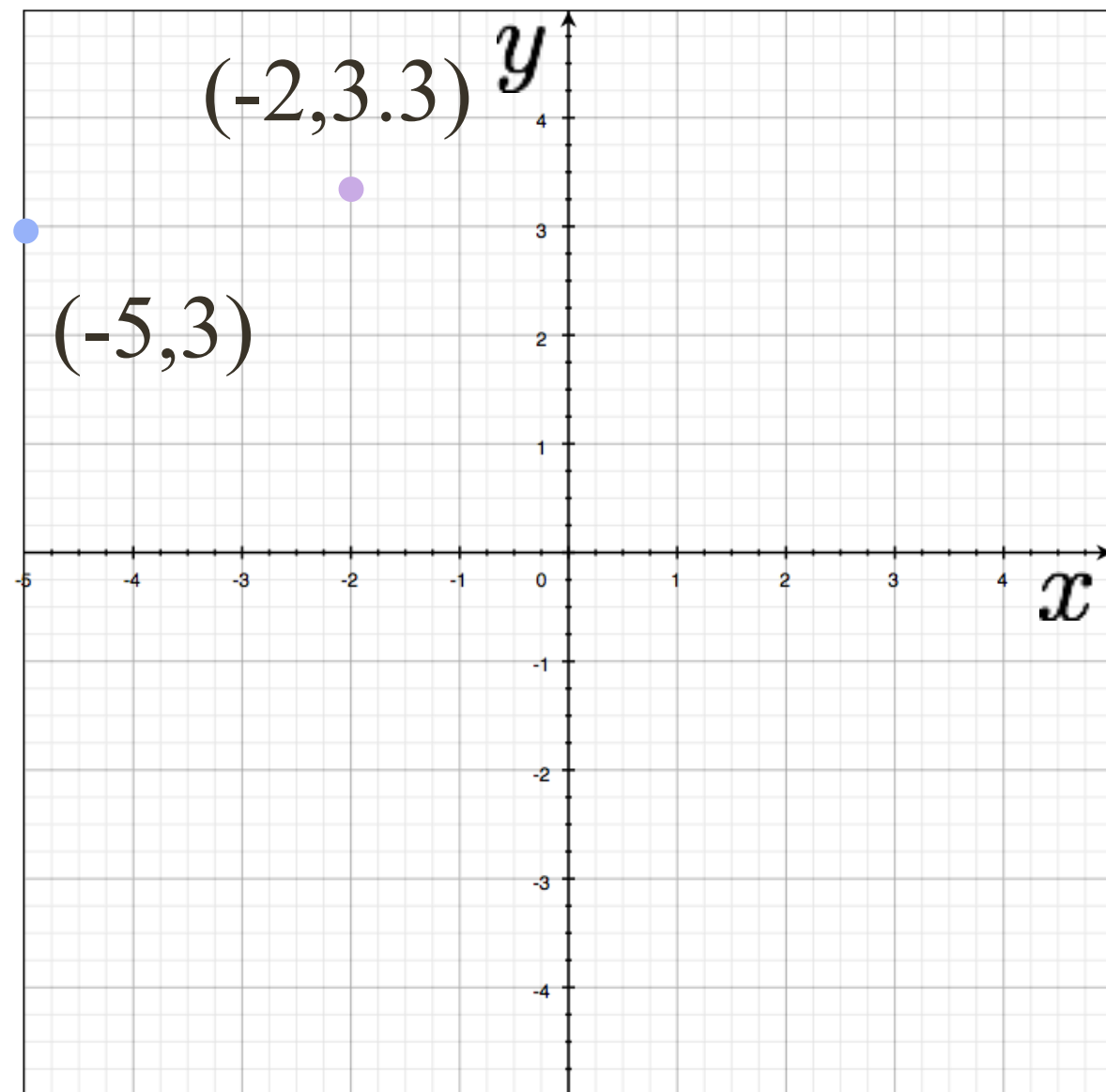
$$-5 \sin(15^\circ) - 3 \cos(15^\circ) + r = 0 \Rightarrow r = 4.18$$

$$-5 \sin(25^\circ) - 3 \cos(25^\circ) + r = 0 \Rightarrow r = 4.83$$

$$-2 \sin(5^\circ) - 3.3 \cos(5^\circ) + r = 0 \Rightarrow r = 3.46$$



# Example: Hough Transform for Lines



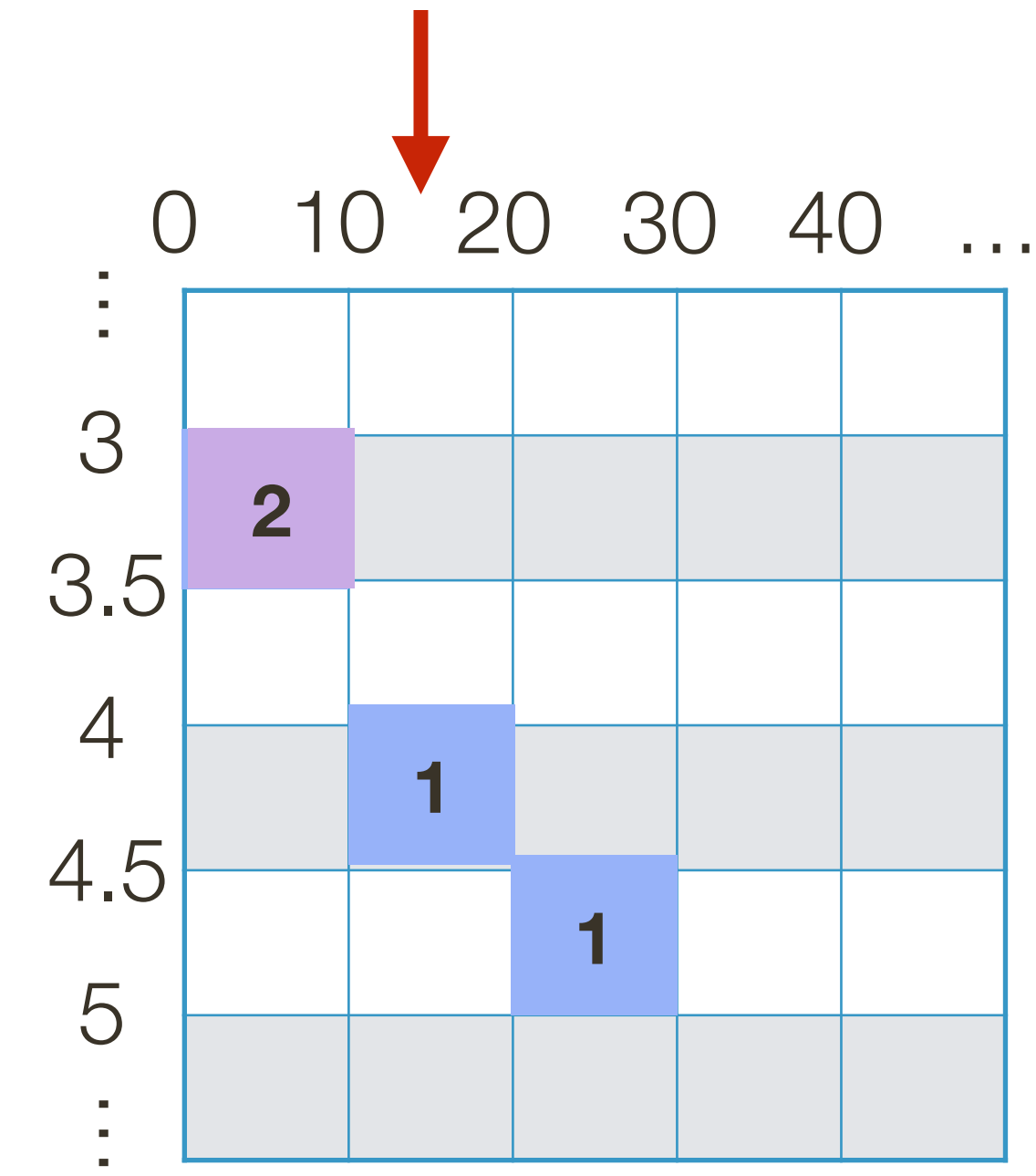
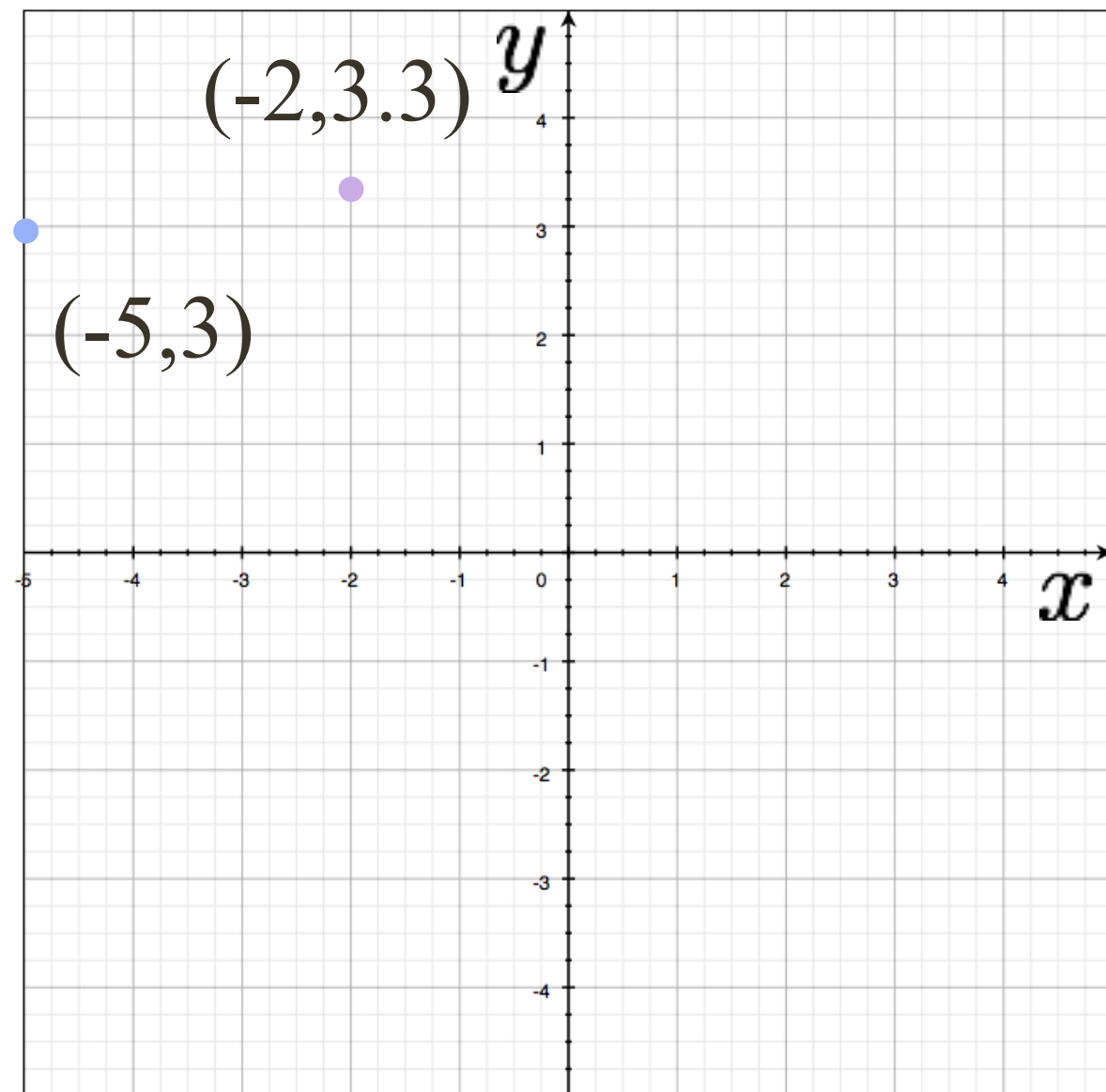
$$-5 \sin(5^\circ) - 3 \cos(5^\circ) + r = 0 \Rightarrow r = 3.42$$

$$-5 \sin(15^\circ) - 3 \cos(15^\circ) + r = 0 \Rightarrow r = 4.18$$

$$-5 \sin(25^\circ) - 3 \cos(25^\circ) + r = 0 \Rightarrow r = 4.83$$

$$-2 \sin(5^\circ) - 3.3 \cos(5^\circ) + r = 0 \Rightarrow r = 3.46$$

# Example: Hough Transform for Lines



$$-5 \sin(5^\circ) - 3 \cos(5^\circ) + r = 0 \Rightarrow r = 3.42$$

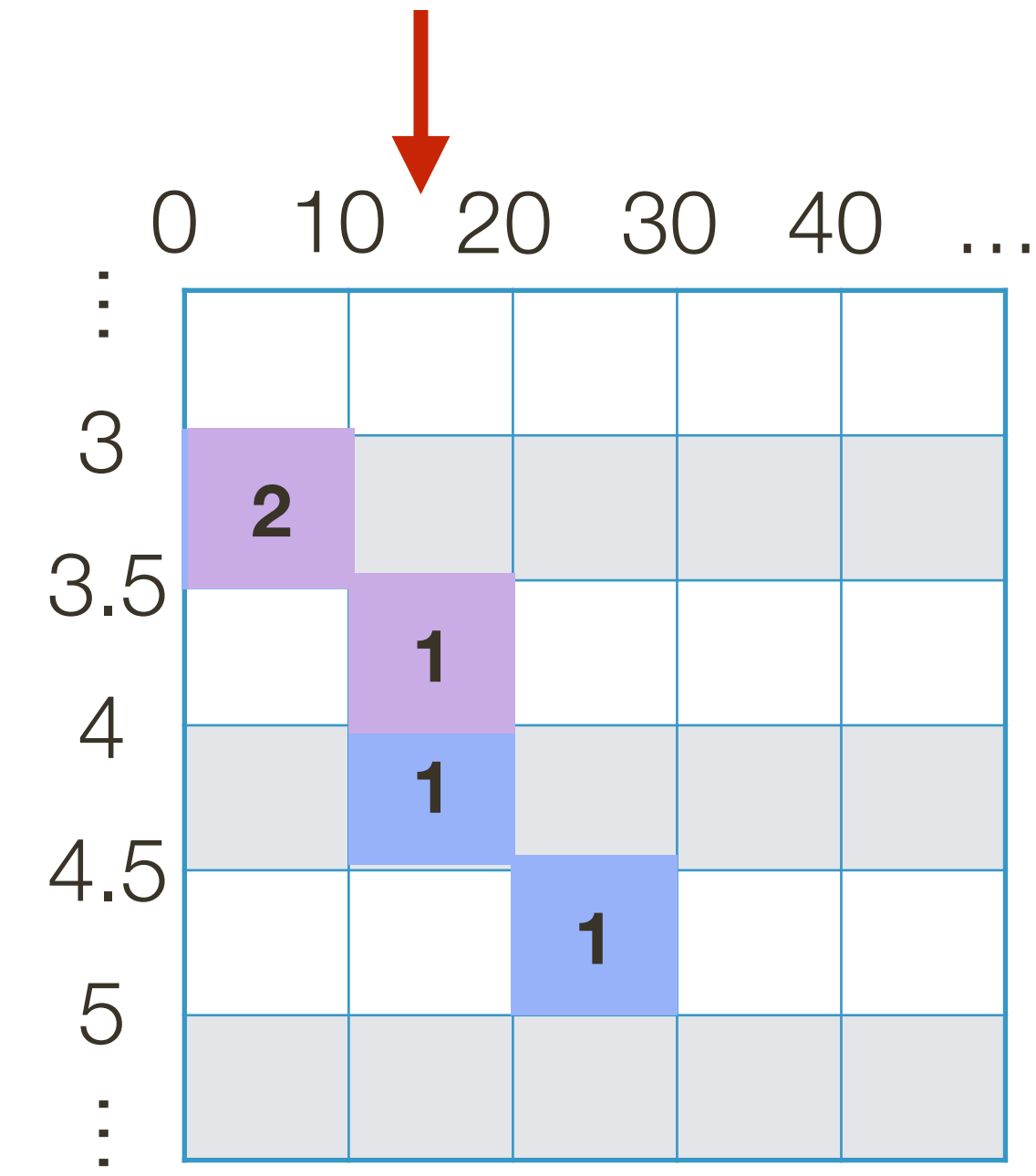
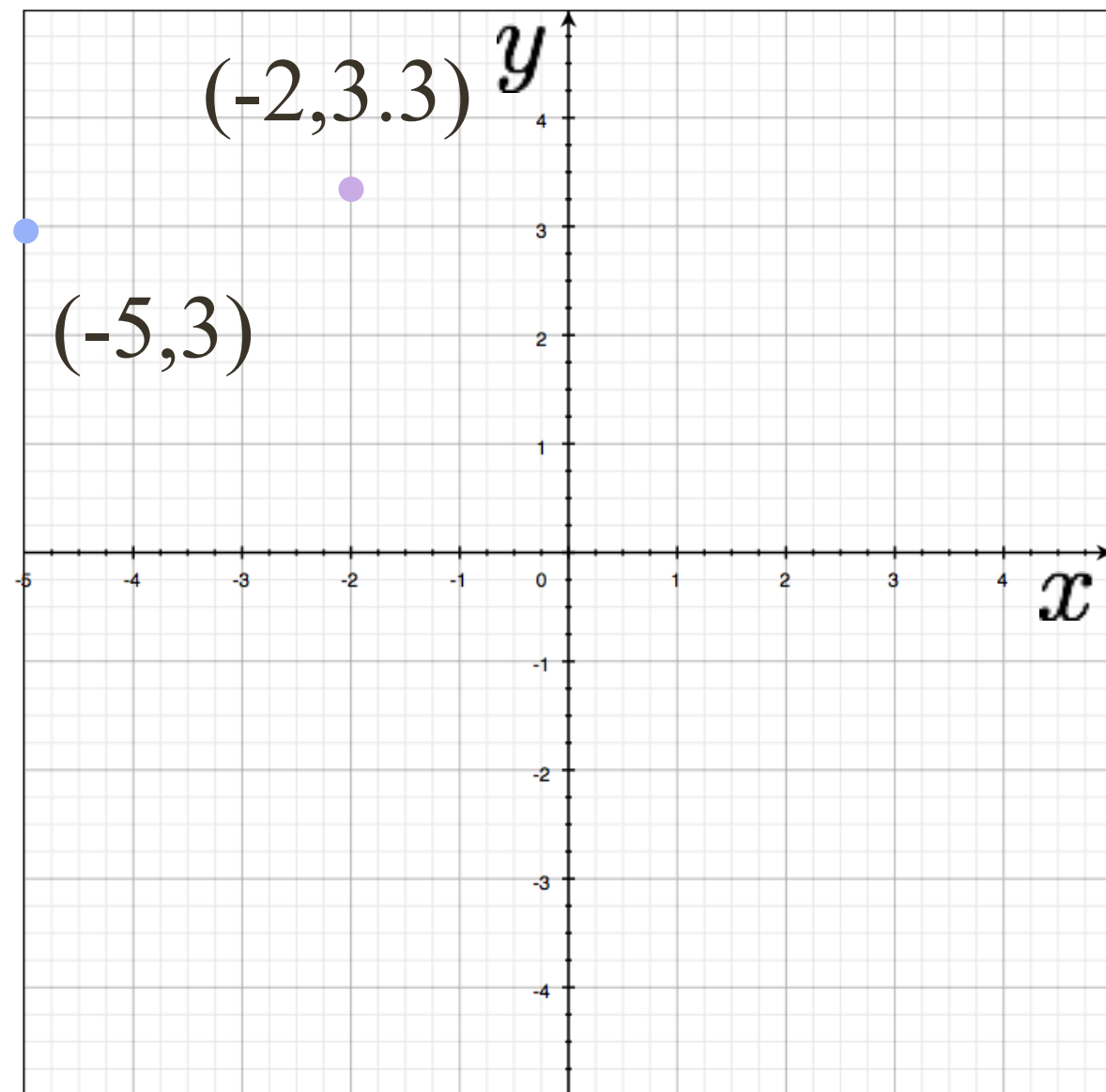
$$-5 \sin(15^\circ) - 3 \cos(15^\circ) + r = 0 \Rightarrow r = 4.18$$

$$-5 \sin(25^\circ) - 3 \cos(25^\circ) + r = 0 \Rightarrow r = 4.83$$

$$-2 \sin(5^\circ) - 3.3 \cos(5^\circ) + r = 0 \Rightarrow r = 3.46$$

$$-2 \sin(15^\circ) - 3.3 \cos(15^\circ) + r = 0 \Rightarrow r = 3.71$$

# Example: Hough Transform for Lines



$$-5 \sin(5^\circ) - 3 \cos(5^\circ) + r = 0 \Rightarrow r = 3.42$$

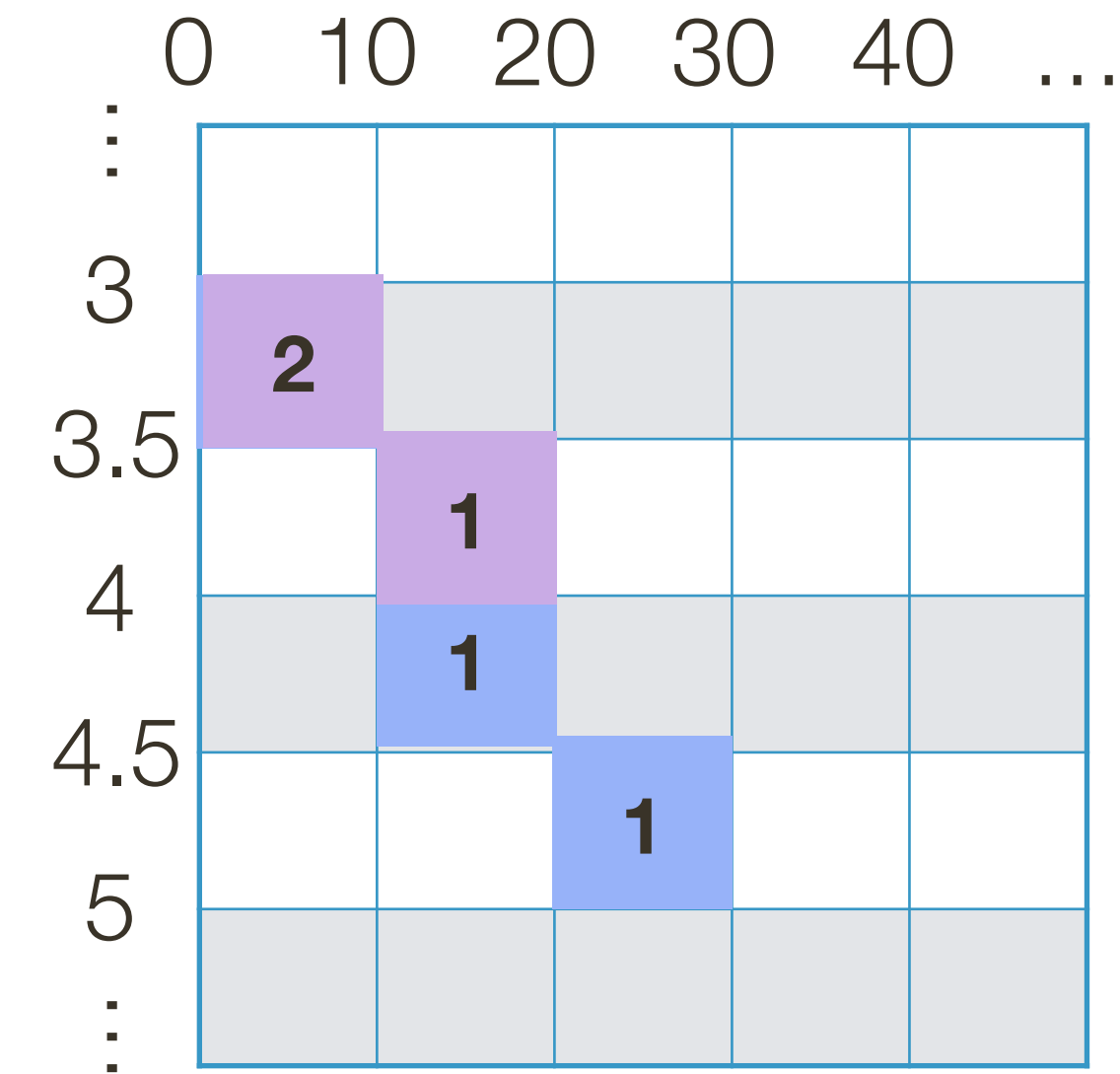
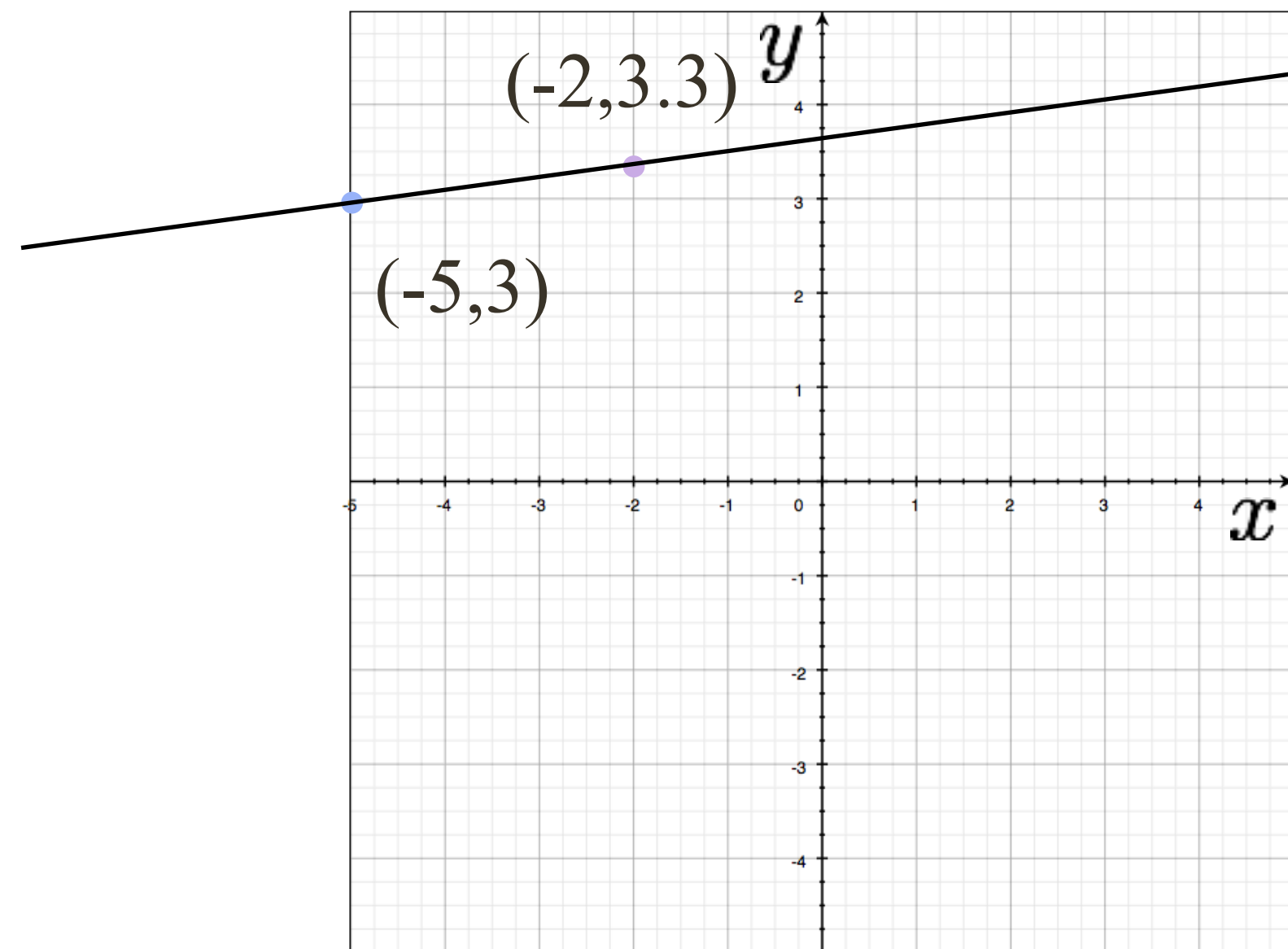
$$-5 \sin(15^\circ) - 3 \cos(15^\circ) + r = 0 \Rightarrow r = 4.18$$

$$-5 \sin(25^\circ) - 3 \cos(25^\circ) + r = 0 \Rightarrow r = 4.83$$

$$-2 \sin(5^\circ) - 3.3 \cos(5^\circ) + r = 0 \Rightarrow r = 3.46$$

$$-2 \sin(15^\circ) - 3.3 \cos(15^\circ) + r = 0 \Rightarrow r = 3.71$$

# Example: Hough Transform for Lines



$$-5 \sin(5^\circ) - 3 \cos(5^\circ) + r = 0 \Rightarrow r = 3.42$$

$$-5 \sin(15^\circ) - 3 \cos(15^\circ) + r = 0 \Rightarrow r = 4.18$$

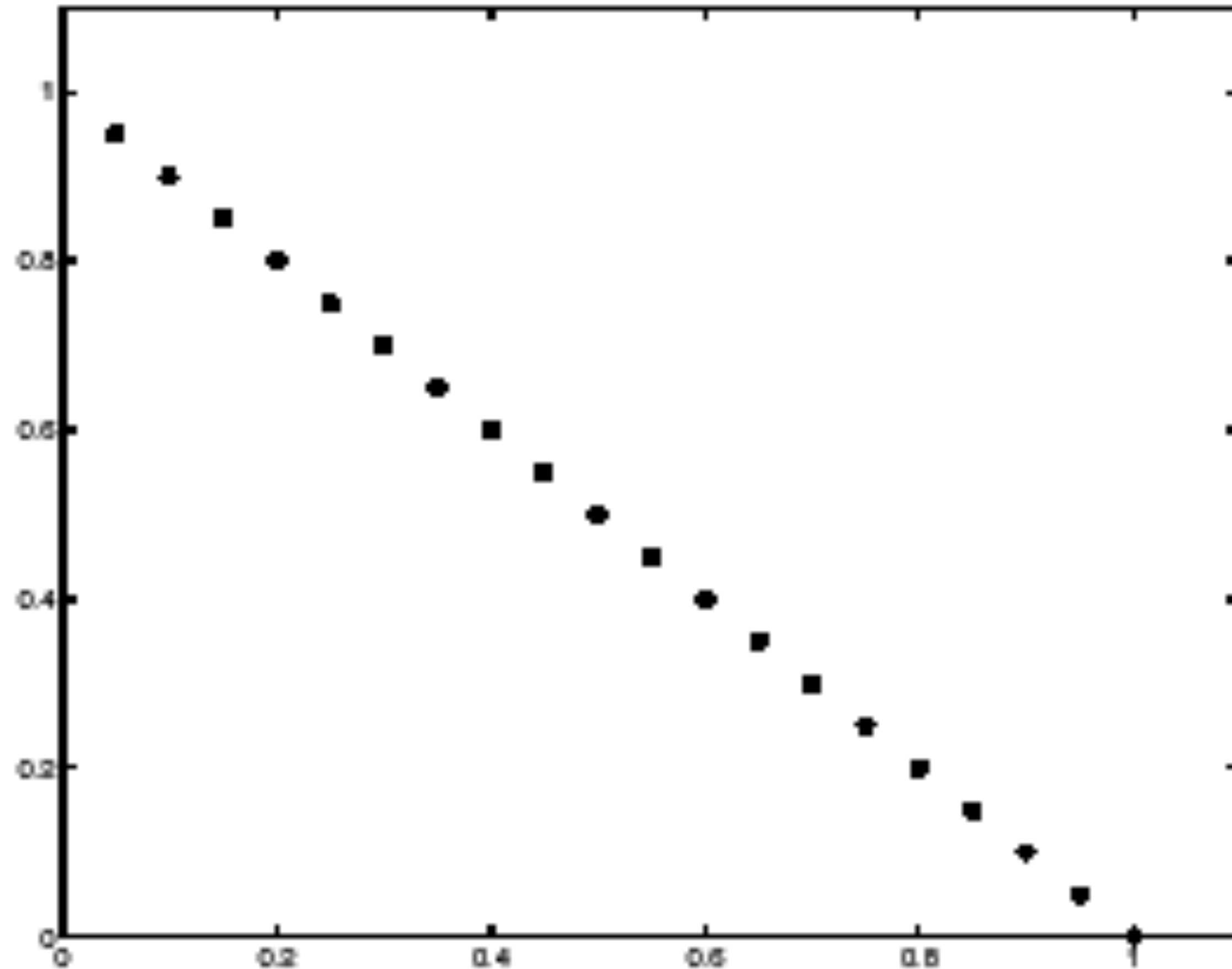
$$-5 \sin(25^\circ) - 3 \cos(25^\circ) + r = 0 \Rightarrow r = 4.83$$

$$-2 \sin(5^\circ) - 3.3 \cos(5^\circ) + r = 0 \Rightarrow r = 3.46$$

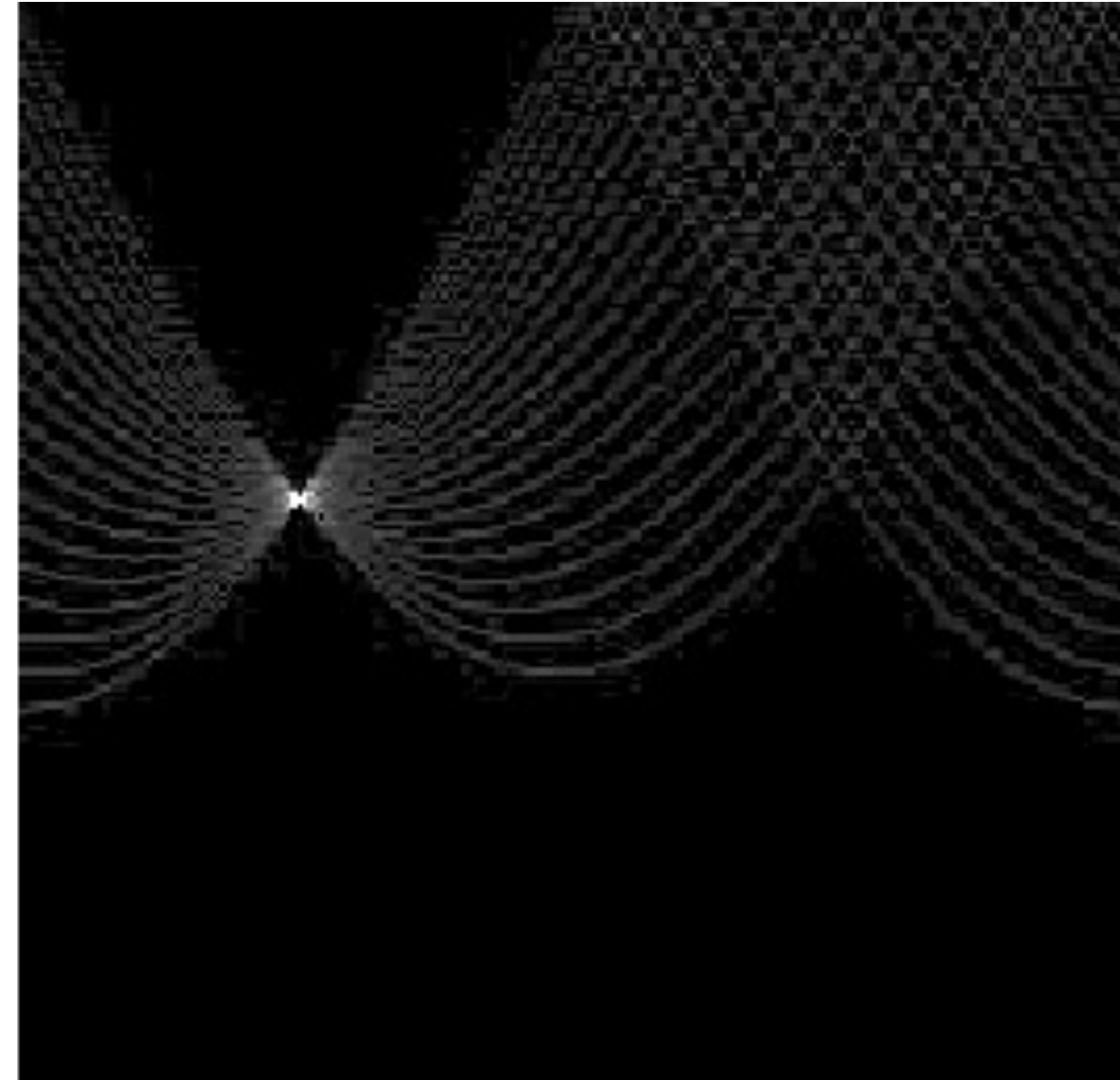
$$-2 \sin(15^\circ) - 3.3 \cos(15^\circ) + r = 0 \Rightarrow r = 3.71$$



# Example: Clean Data



**Tokens**

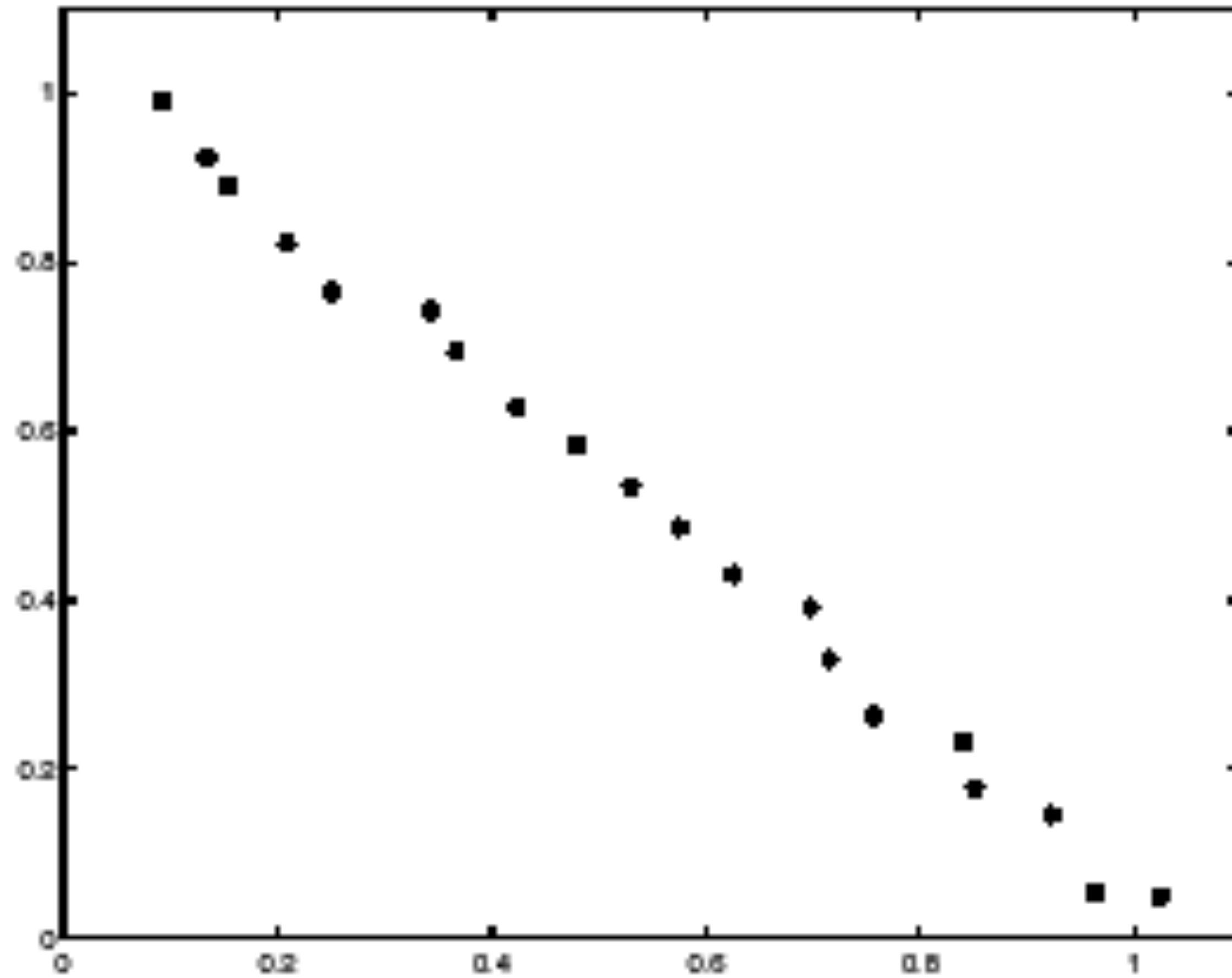


**Votes**

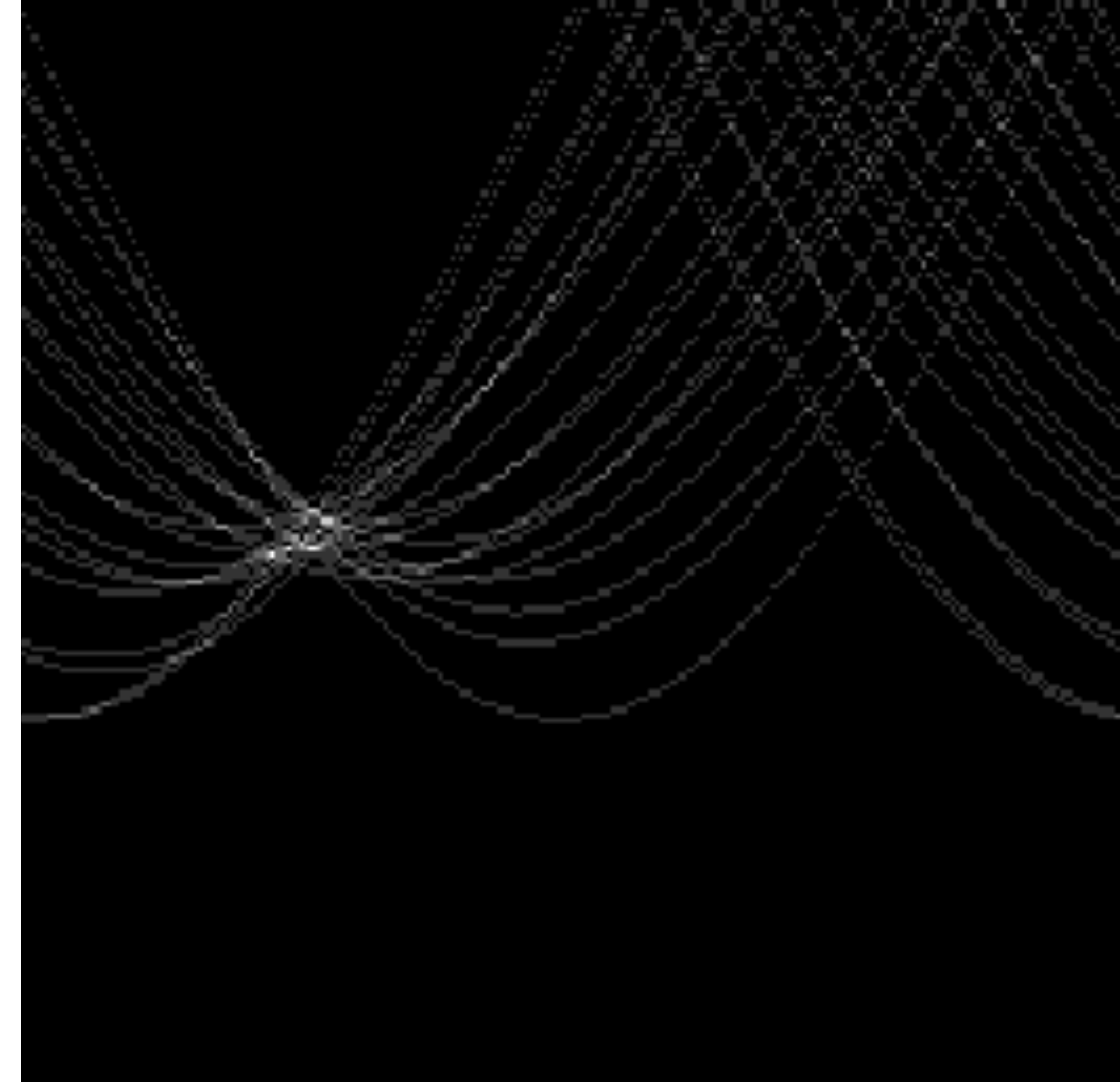
Horizontal axis is  $\theta$   
Vertical Axis is  $r$

Forsyth & Ponce (2nd ed.) Figure 10.1 (Top)

# Example: Some Noise



**Tokens**

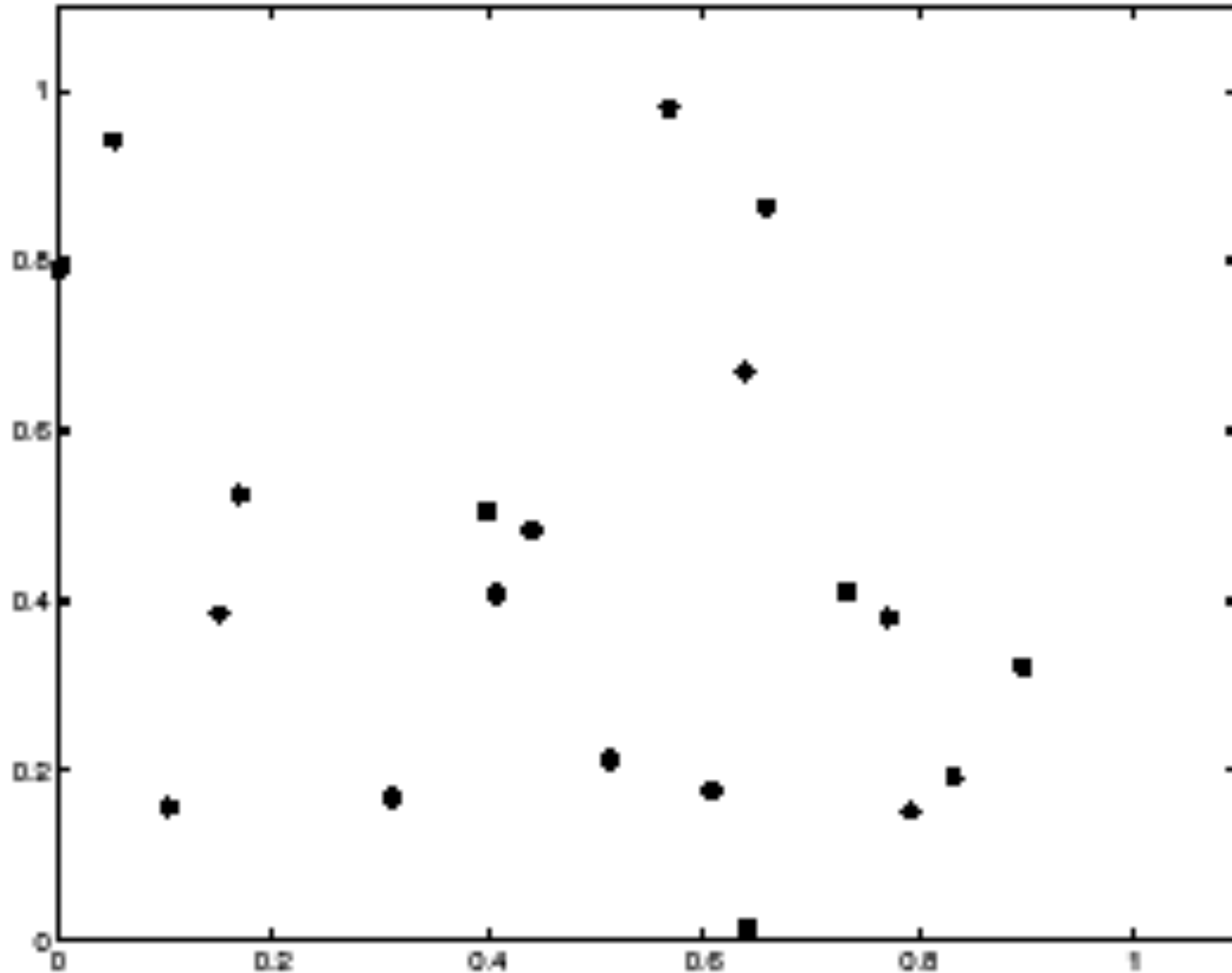


**Votes**

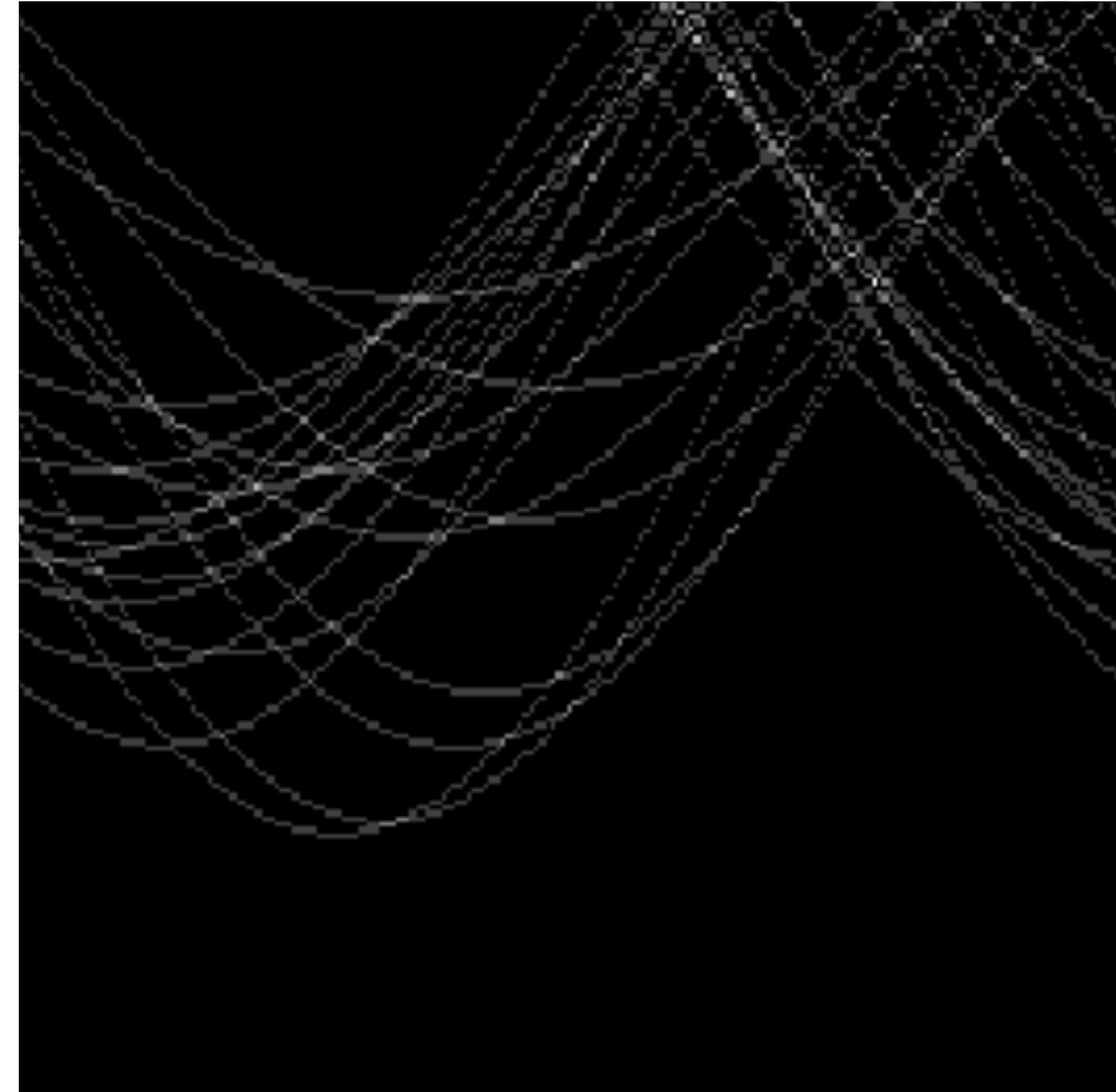
Horizontal axis is  $\theta$   
Vertical Axis is  $r$

Forsyth & Ponce (2nd ed.) Figure 10.1 (Bottom)

# Example: Too Much Noise



**Tokens**



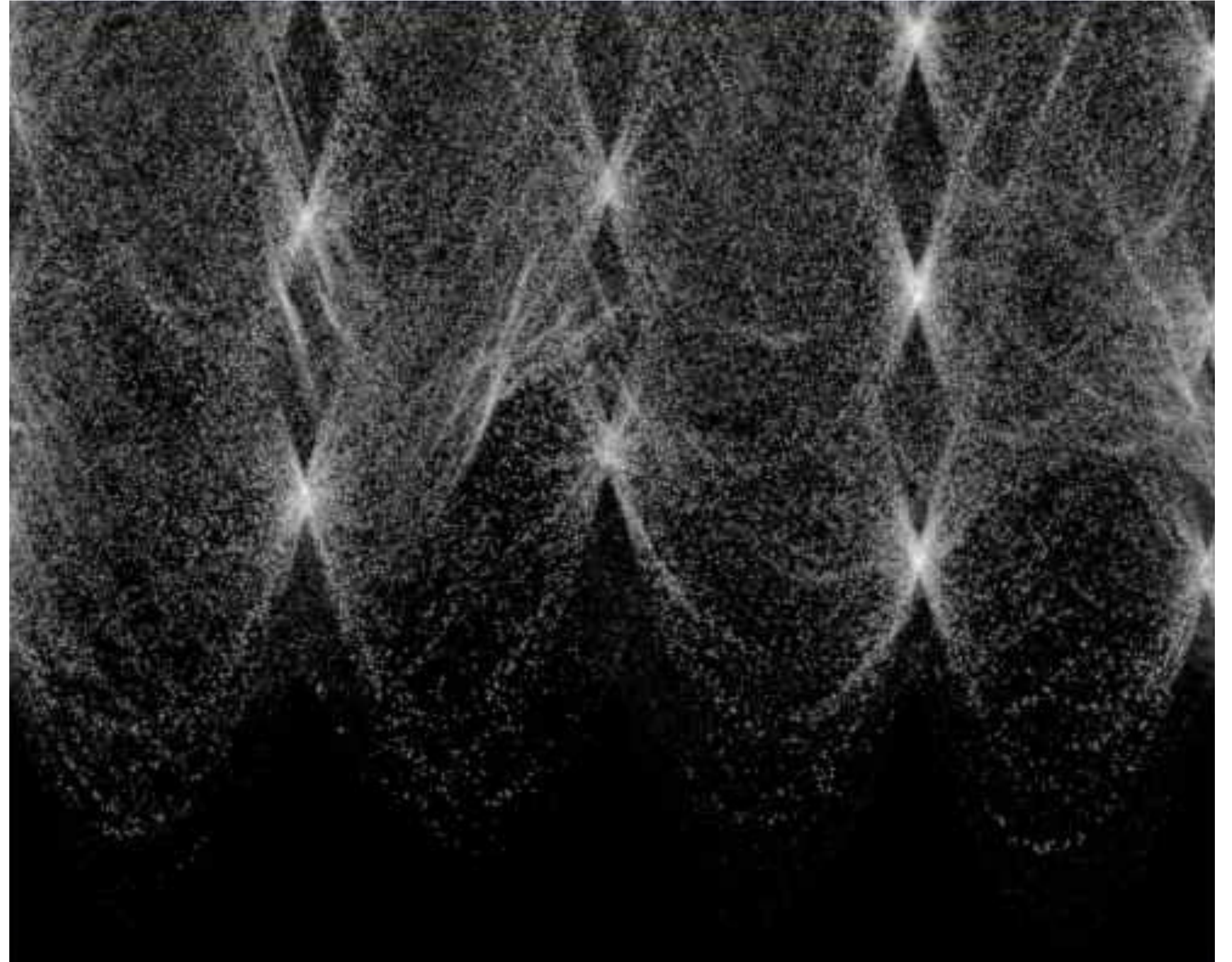
**Votes**

Horizontal axis is  $\theta$   
Vertical Axis is  $r$

Forsyth & Ponce (2nd ed.) Figure 10.2



# Real World **Example**

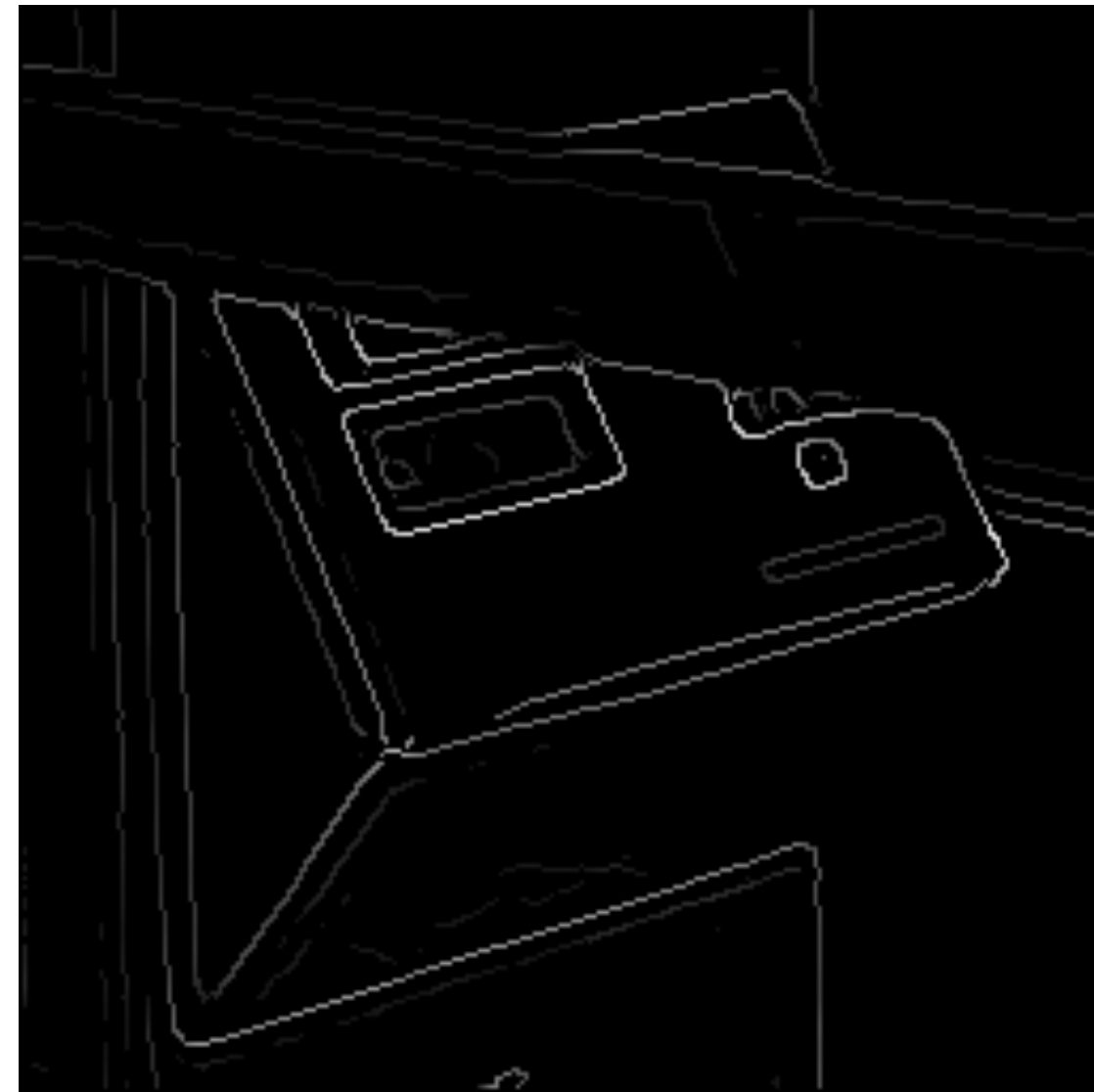




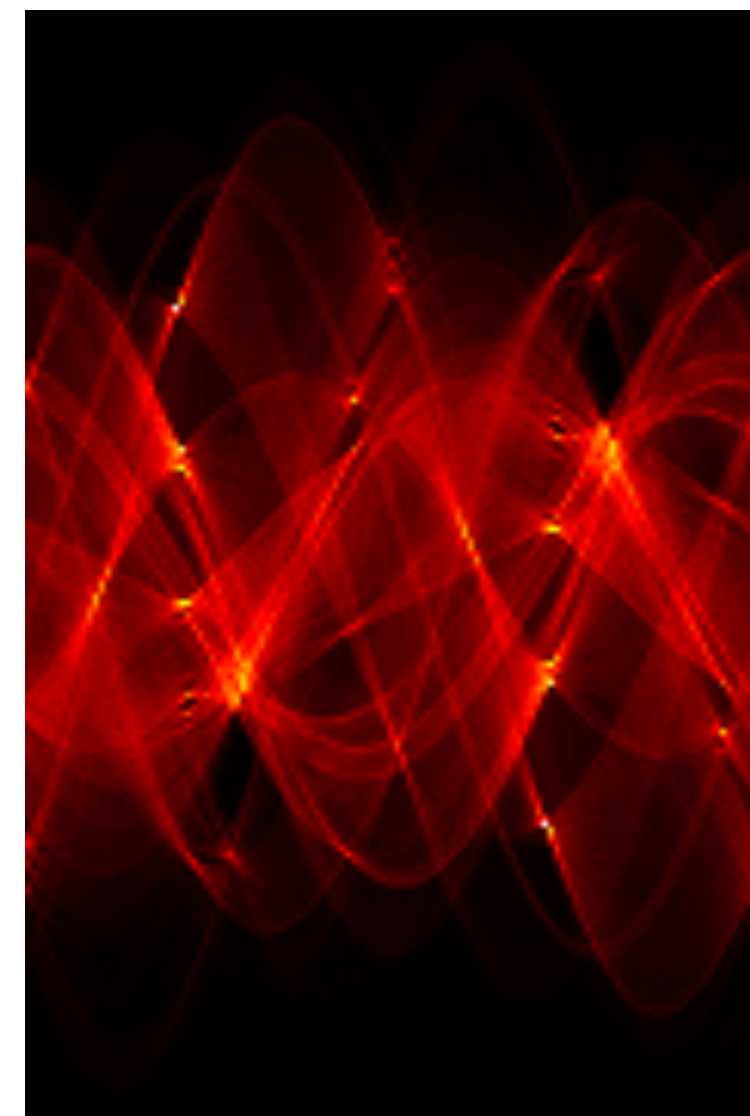
# Real World **Example**



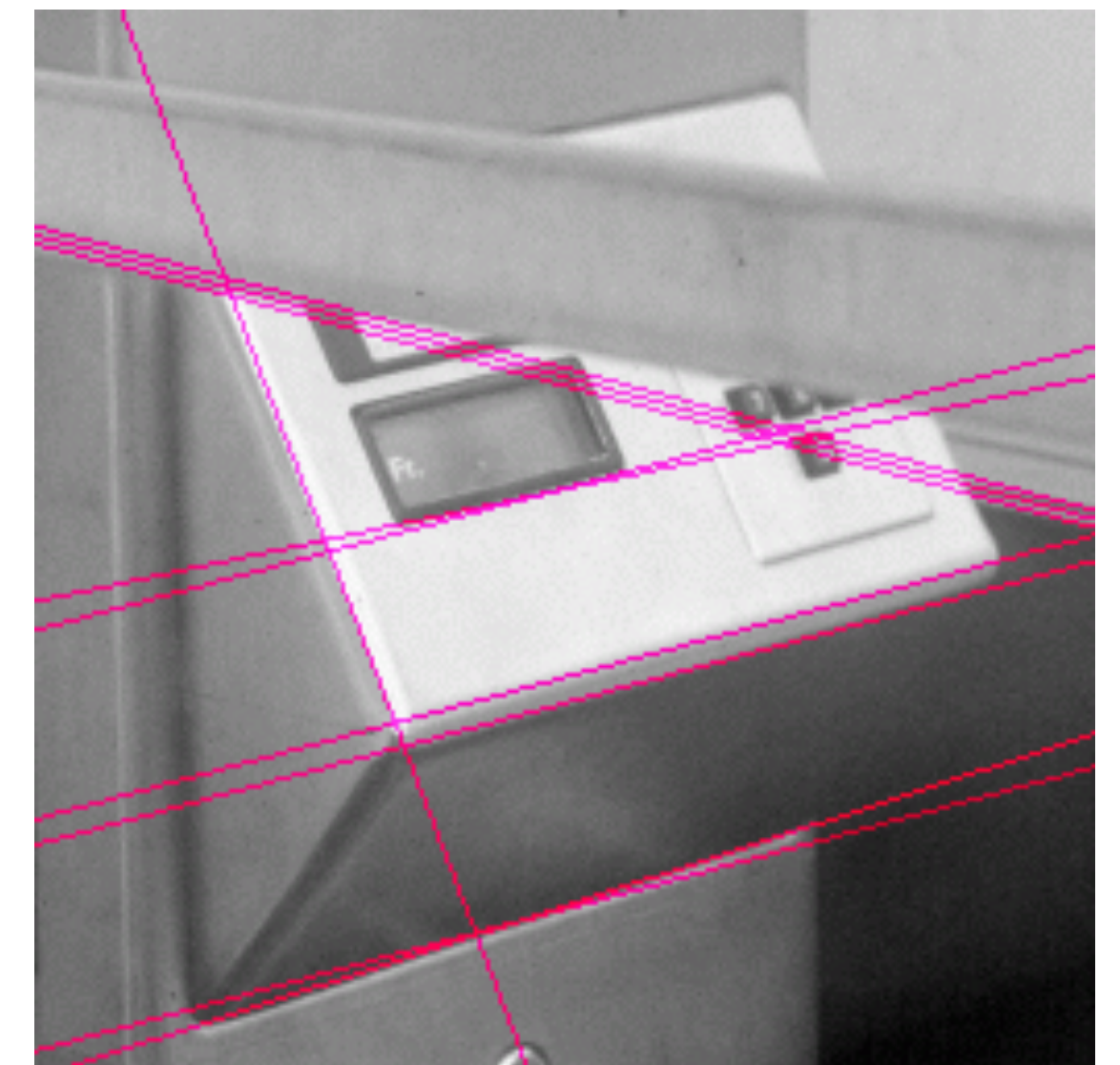
Original



Edges



Parameter  
space



Hough Lines

# Mechanics of Hough Transform

1. Construct a quantized array to represent  $\theta$  and  $r$
2. For each point, render curve  $(\theta, r)$  into this array adding one vote at each cell

## Difficulties:

— How big should the cells be? (too big, and we merge quite different lines; too small, and noise causes lines to be missed)

## How many lines?

- Count the peaks in the Hough array
- Treat adjacent peaks as a single peak

# Some **Practical Details** of Hough Transform

It is best to **vote** for the two closest bins in each dimension, as the locations of the bin boundaries are arbitrary

- This means that peaks are “blurred” and noise will not cause similar votes to fall into separate bins

Can use a **hash table** rather than an array to store the votes

- This means that no effort is wasted on initializing and checking empty bins
- It avoids the need to predict the maximum size of the array, which can be non-rectangular

# Hough Transform: Transformation Space Voting

Sometimes a single point / measurement can vote on the entire transformation

e.g., **SIFT** keypoint matches with **location, scale and orientation** vote on the 4 parameters of a **similarity transform** (x,y,s,theta)

In this case, the votes of each sample can be seen as a **distribution in the parameter space** of the transformation

This can be effective in preventing noise in the distribution, e.g., edge detections with orientation can vote on **single lines** rather than **all lines** that pass through a point.

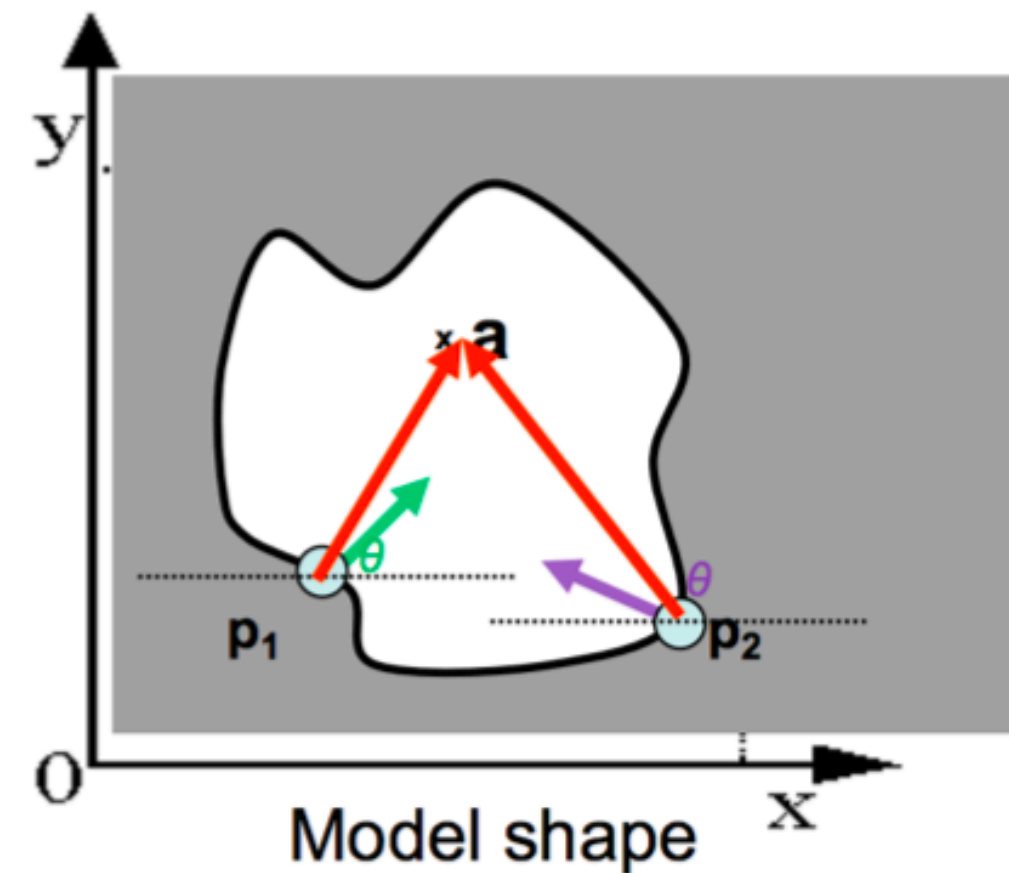


# Generalized Hough Transform

What if we want to detect an **arbitrary** geometric shape?

# Generalized Hough Transform

What if we want to detect an **arbitrary** geometric shape?



	...
	...
⋮	

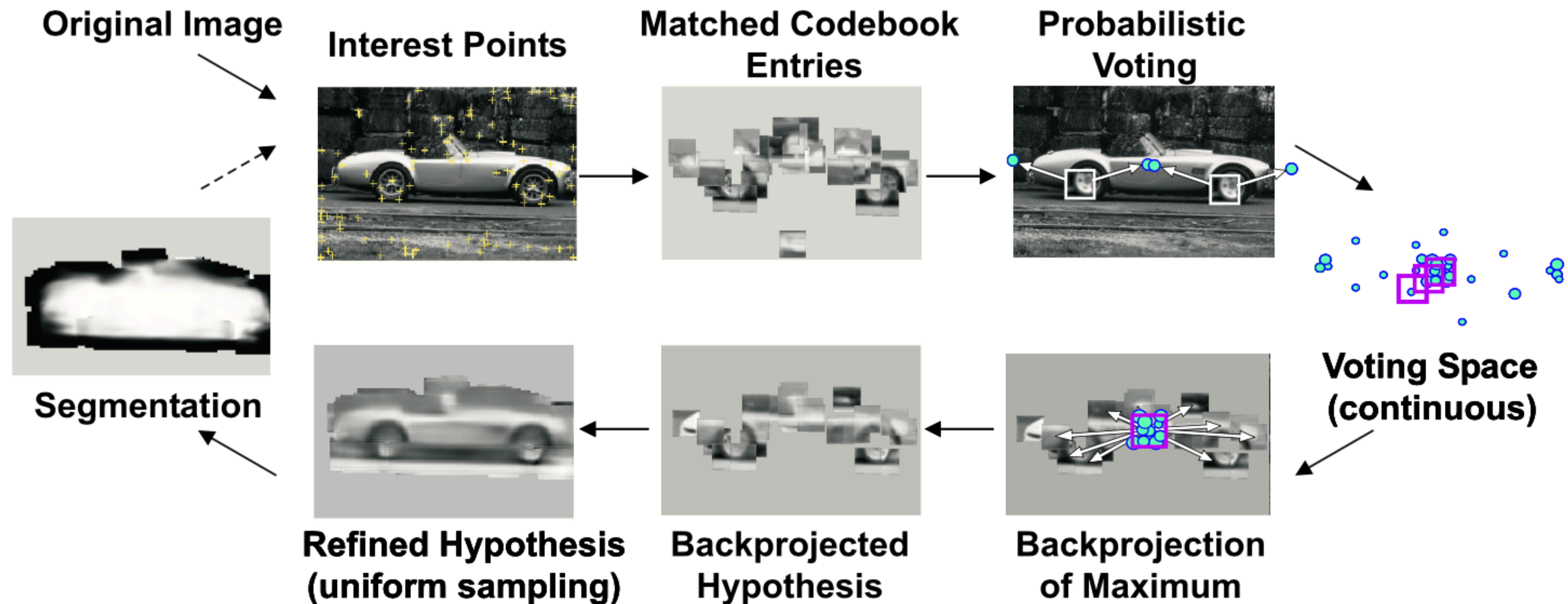
## Offline procedure:

At each boundary point, compute displacement vector:  $\mathbf{r} = \mathbf{a} - \mathbf{p}_i$ .

Store these vectors in a table indexed by gradient orientation  $\theta$ .

# Example 1: Object Recognition — Implicit Shape Model

Combined object detection and segmentation using an implicit shape model. Image patches cast weighted votes for the object centroid.



# Example 1: Object Recognition — Implicit Shape Model

## Basic Idea:

- Find **interest points/keypoints** in an image (e.g., SIFT Keypoint detector or Corners)
- **Match patch** around each interest point to a training patch (e.g., SIFT Descriptor)
- **Vote** for object center given that training instances
- Find the patches that voted for the peaks (**back-project**)



# Example 1: Object Recognition — Implicit Shape Model

“Training” images of cows

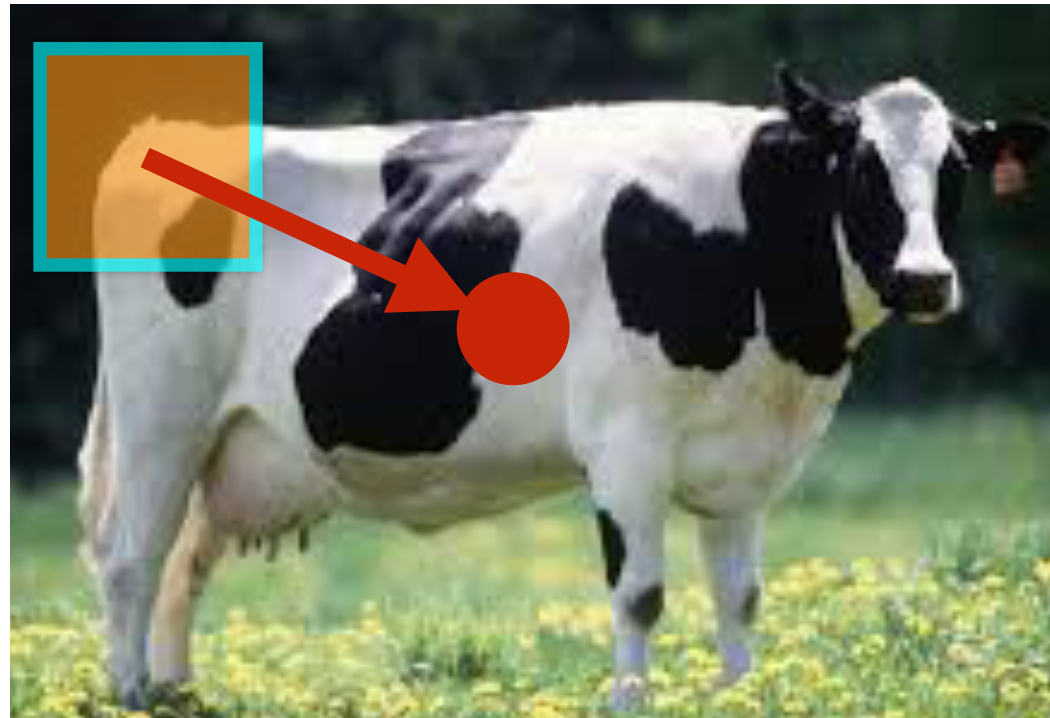


Image Index	Keypoint Index	Keypoint Detection (4D)	Keypoint Description (128D)	Offset to Centroid
Image 1	1	[x, y, s, Theta]	[...]	[x,y] →

# Example 1: Object Recognition — Implicit Shape Model

“Training” images of cows

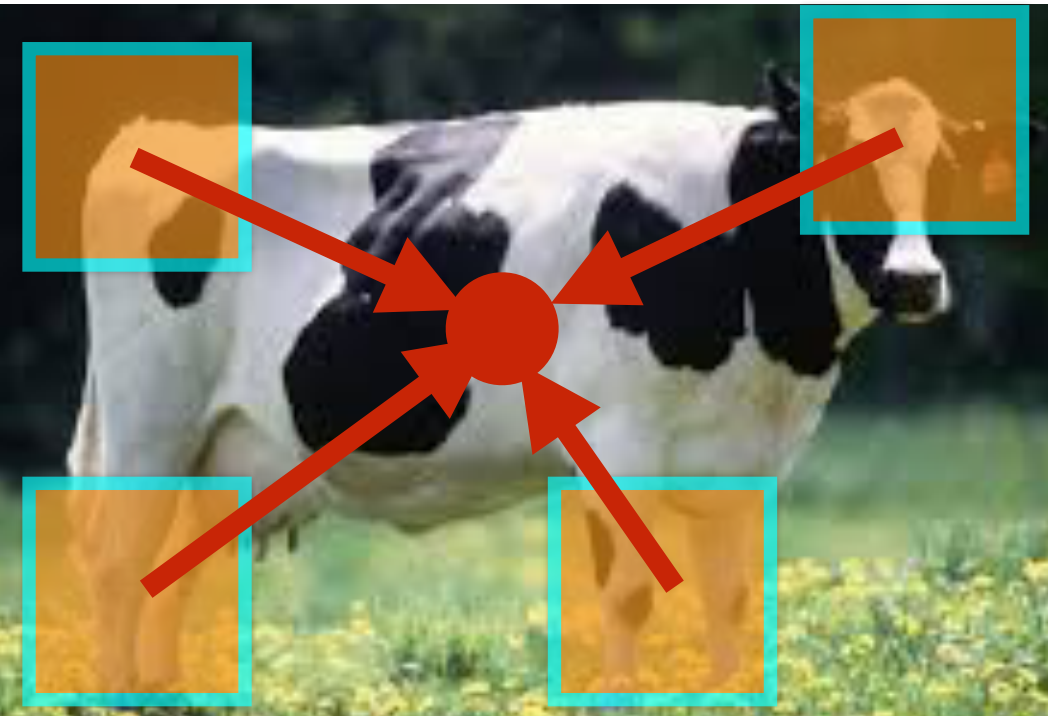


Image Index	Keypoint Index	Keypoint Detection (4D)	Keypoint Description (128D)	Offset to Centroid
Image 1	1	[x, y, s, Theta]	[...]	[x,y] →
Image 1	2	[x, y, s, Theta]	[...]	[x,y]
....	....	....	...	...
Image 1	265	[x, y, s, Theta]	[...]	[x,y]

\* Slide from Sanja Fidler



# Example 1: Object Recognition – Implicit Shape Model

“Training” images of cows

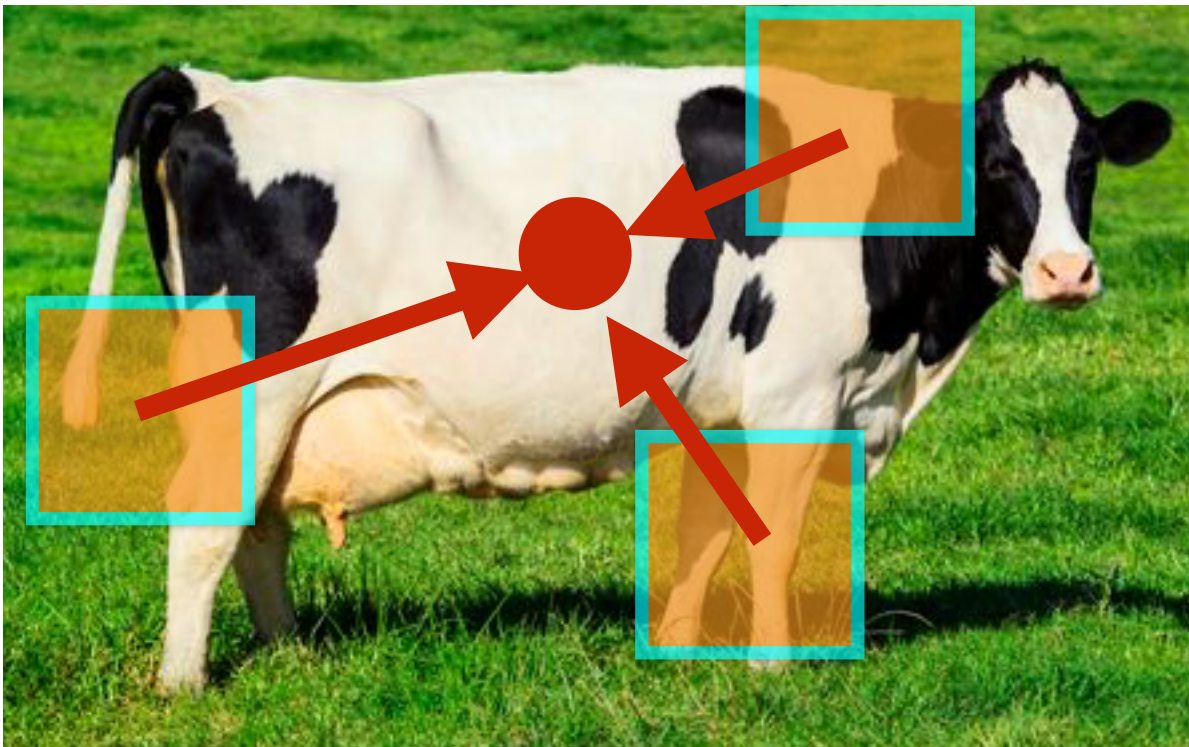
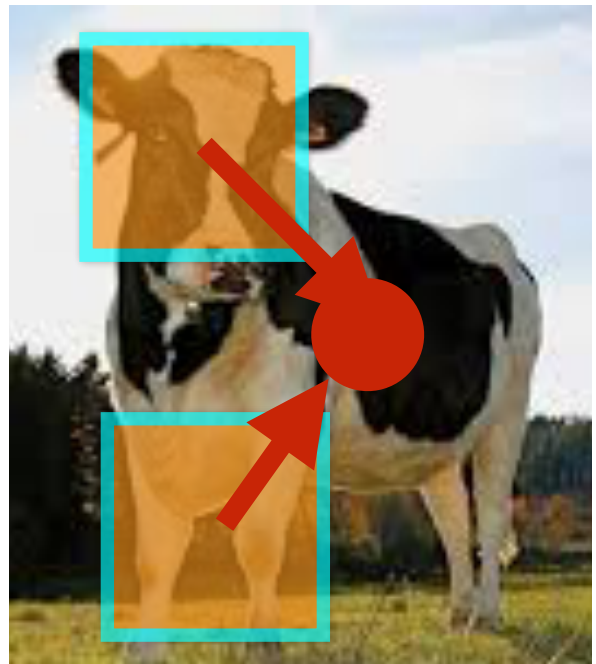
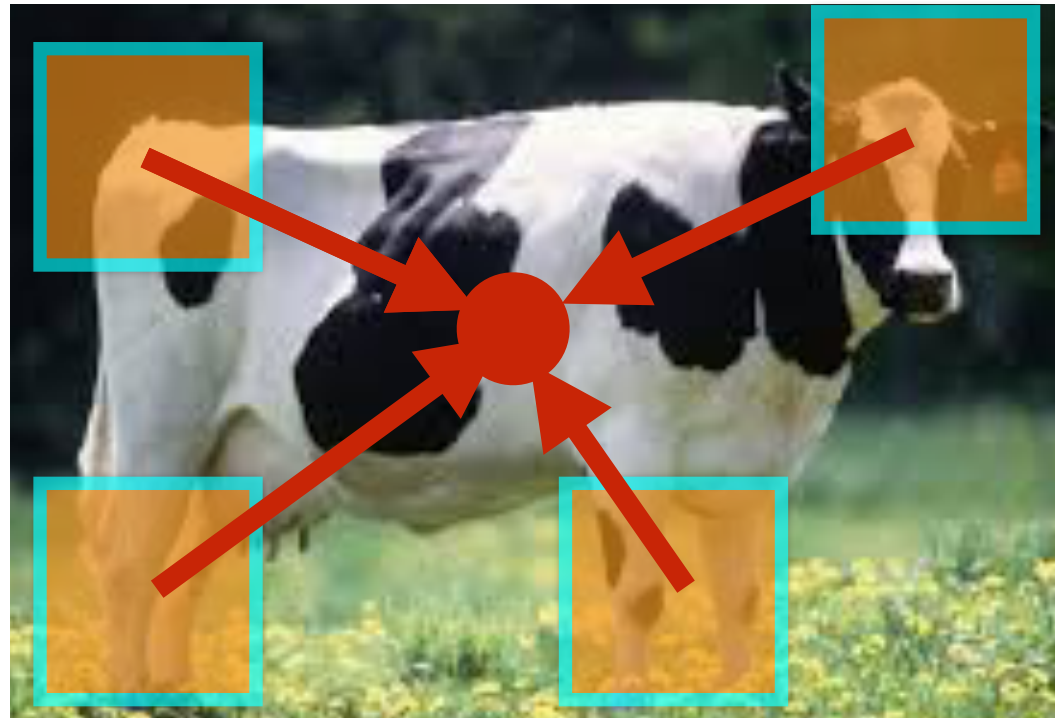


Image Index	Keypoint Index	Keypoint Detection (4D)	Keypoint Description (128D)	Offset to Centroid
Image 1	1	[x, y, s, Theta]	[...]	[x,y] →
Image 1	2	[x, y, s, Theta]	[...]	[x,y]
....	....	....	...	...
Image 1	265	[x, y, s, Theta]	[...]	[x,y]
<hr/>				
Image 2	1	[x, y, s, Theta]	[...]	[x,y]
Image 2	2	[x, y, s, Theta]	[...]	[x,y]
...	...	...	...	...
Image 2	645	[x, y, s, Theta]	[...]	[x,y]
<hr/>				
Image K	1	[x, y, s, Theta]	[...]	[x,y]
Image K	2	[x, y, s, Theta]	[...]	[x,y]
...	...	...	...	...
Image K	134	[x, y, s, Theta]	[...]	[x,y]

\* Slide from Sanja Fidler



# Example 1: Object Recognition — Implicit Shape Model

“**Training**” images of cows



“**Testing**” image





# Example 1: Object Recognition — Implicit Shape Model

“Training” images of cows



“Testing” image



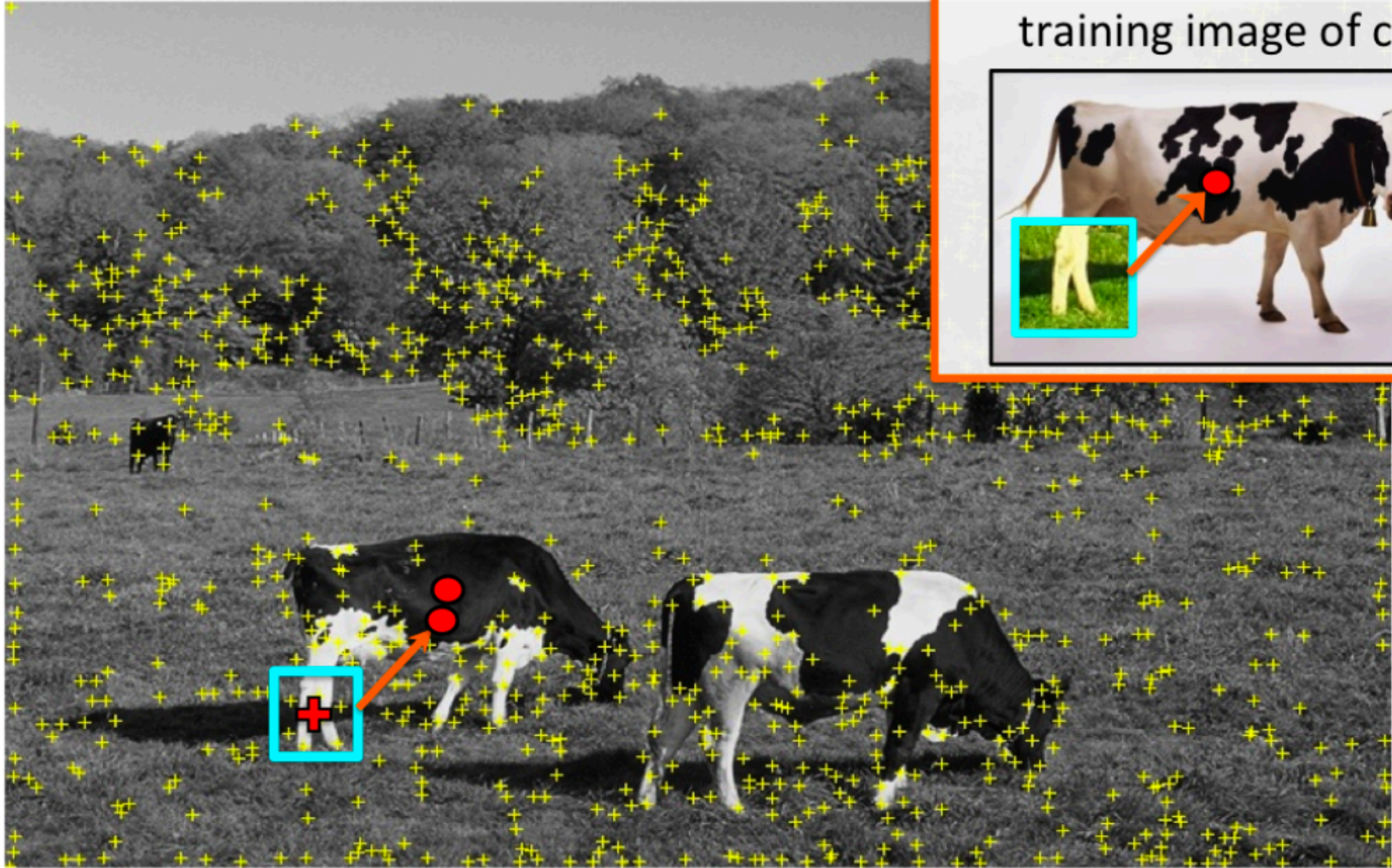
Vote for center of object



# Example 1: Object Recognition — Implicit Shape Model

“Training” images of cows

“Testing” image



Vote for center of object

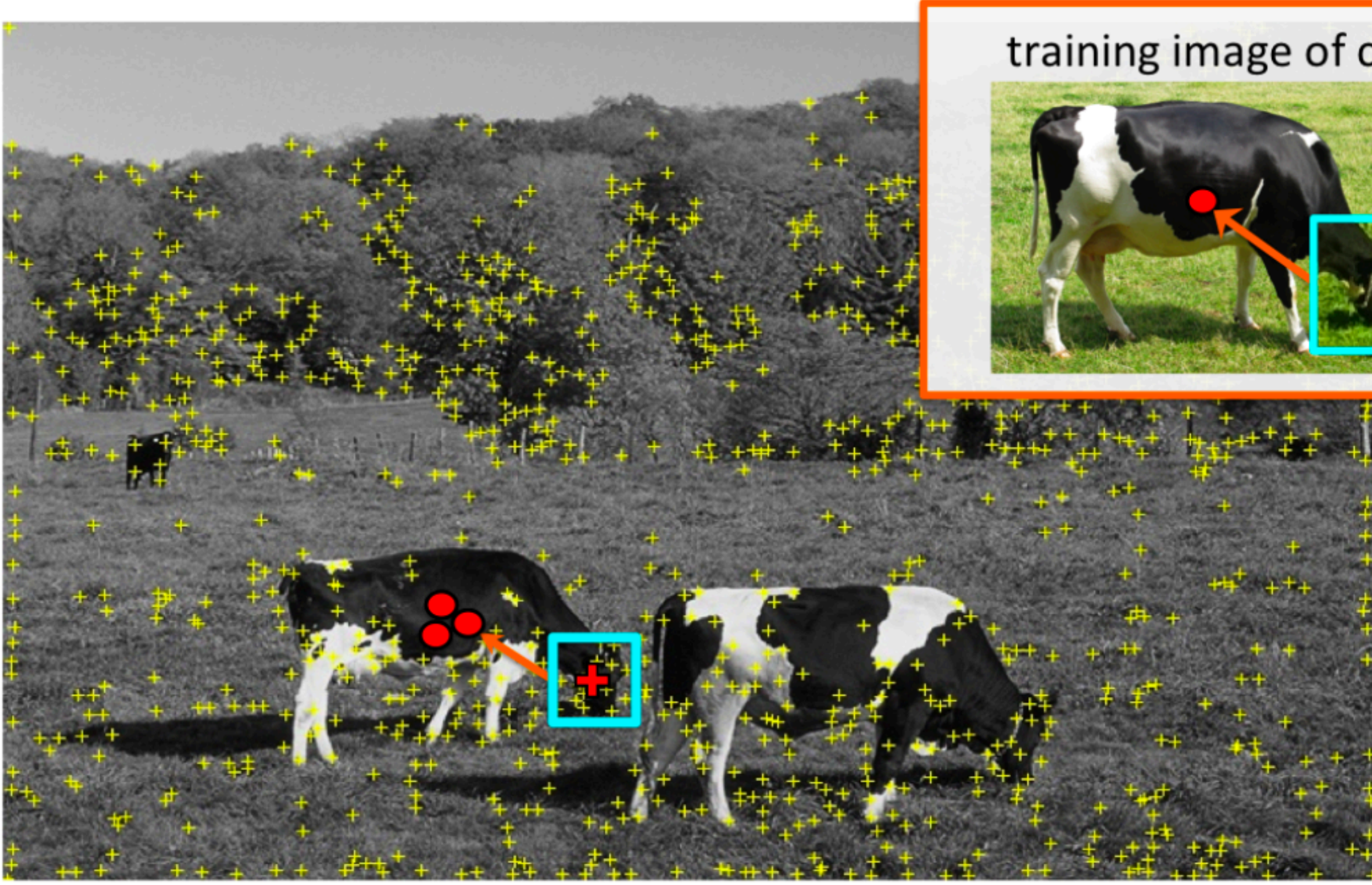
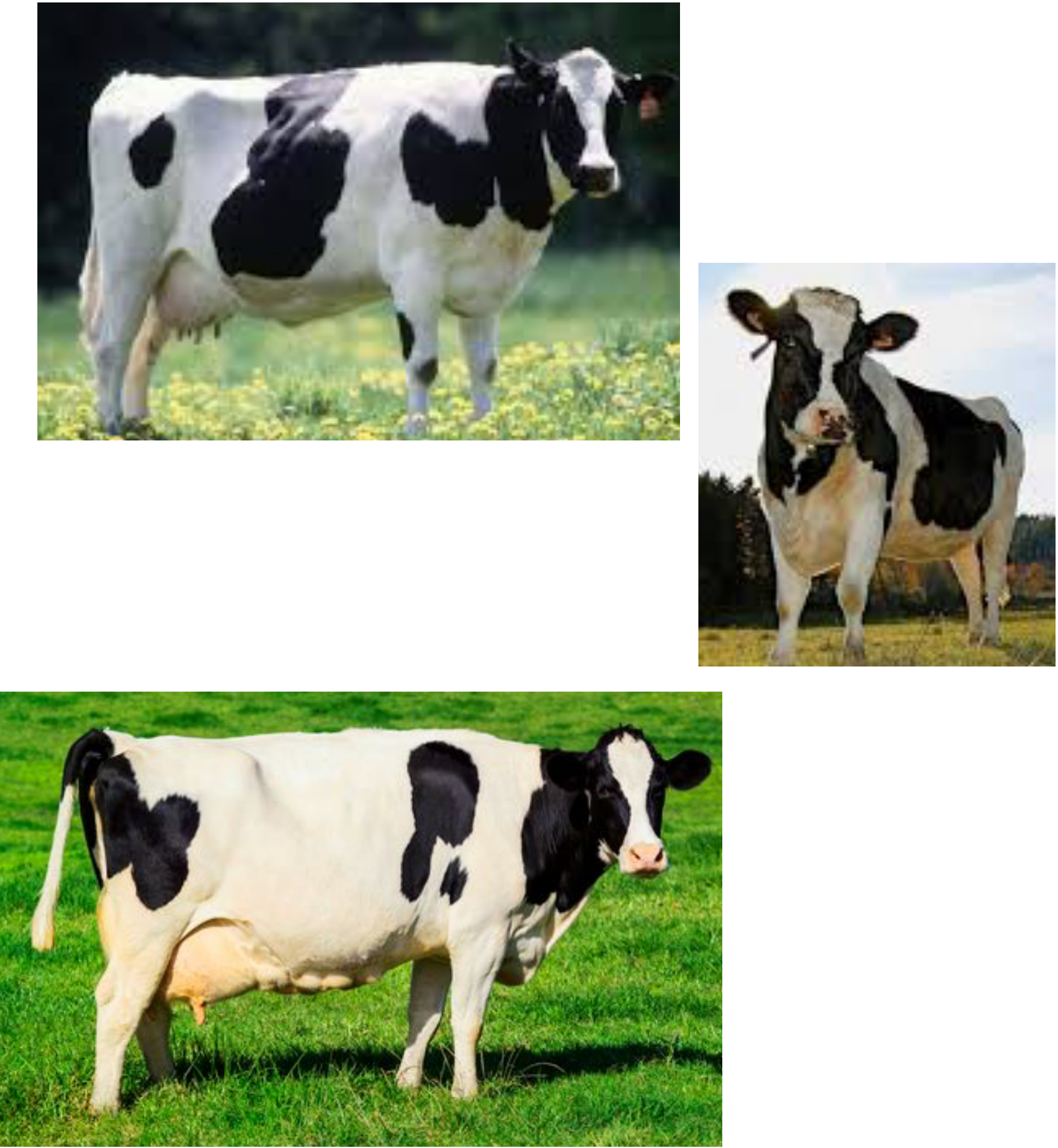
\* Slide from Sanja Fidler



# Example 1: Object Recognition — Implicit Shape Model

“Training” images of cows

“Testing” image



Vote for center of object

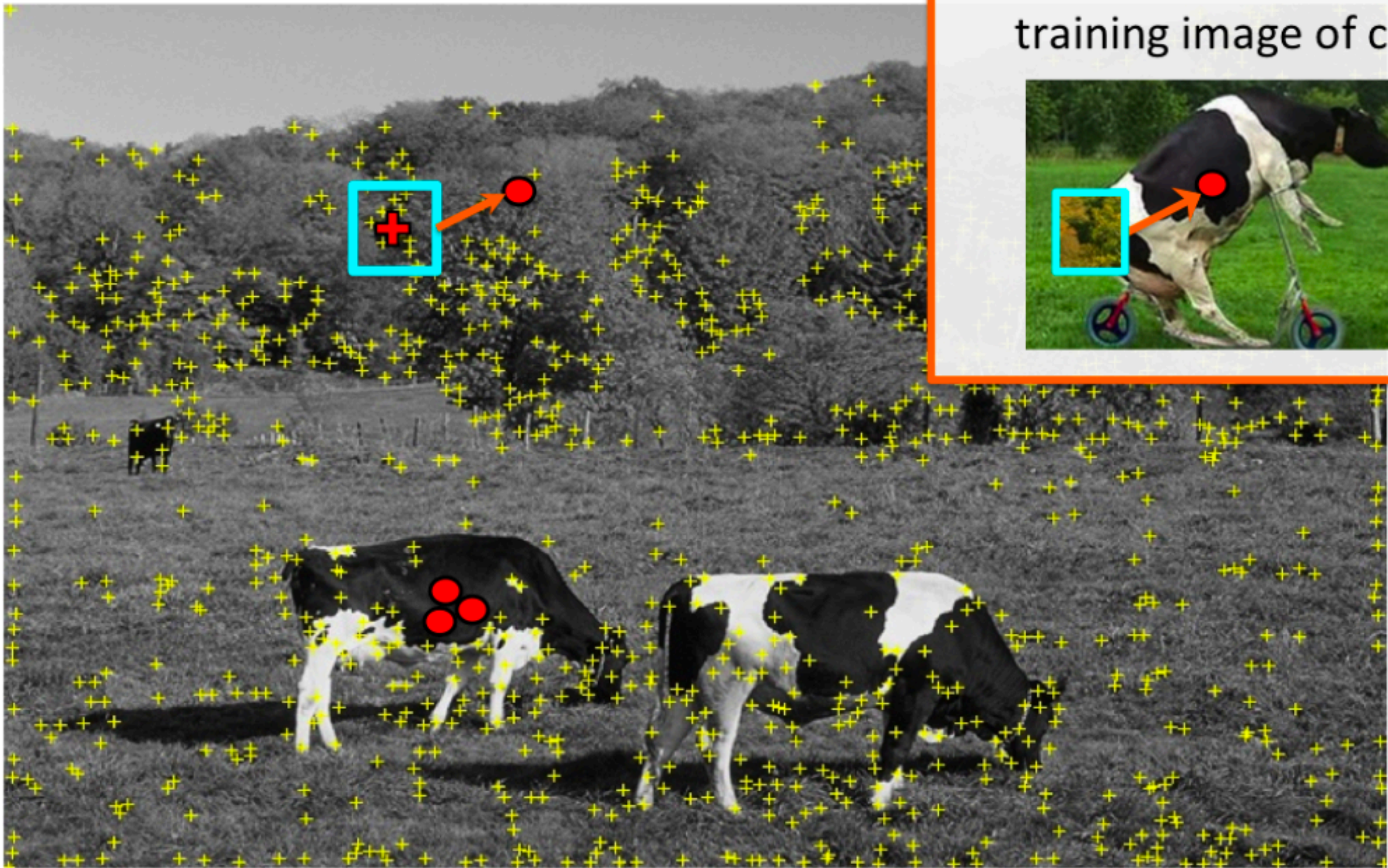
\* Slide from Sanja Fidler



# Example 1: Object Recognition — Implicit Shape Model

“Training” images of cows

“Testing” image



of course sometimes wrong votes are bound to happen

\* Slide from Sanja Fidler

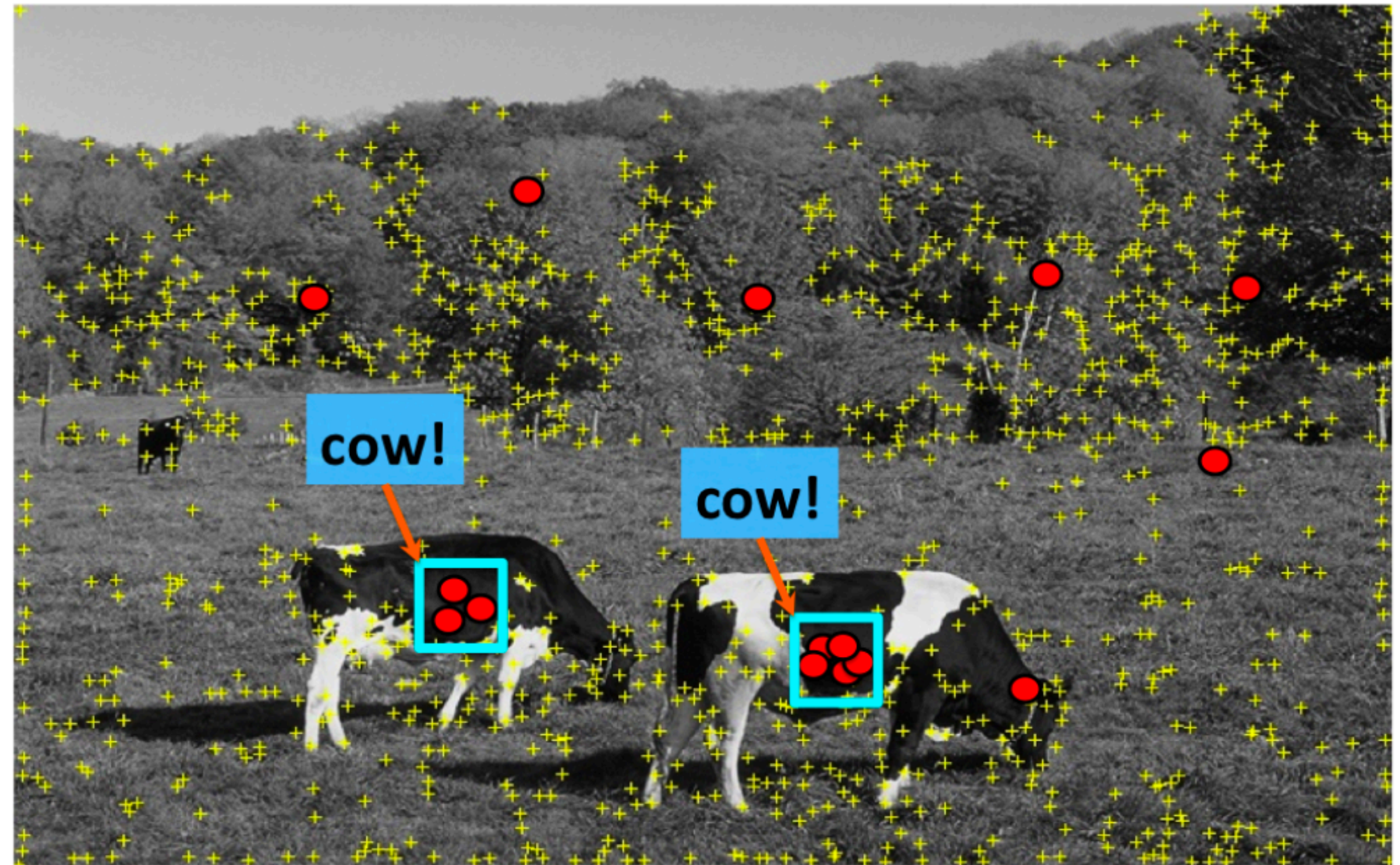


# Example 1: Object Recognition — Implicit Shape Model

“**Training**” images of cows



“**Testing**” image



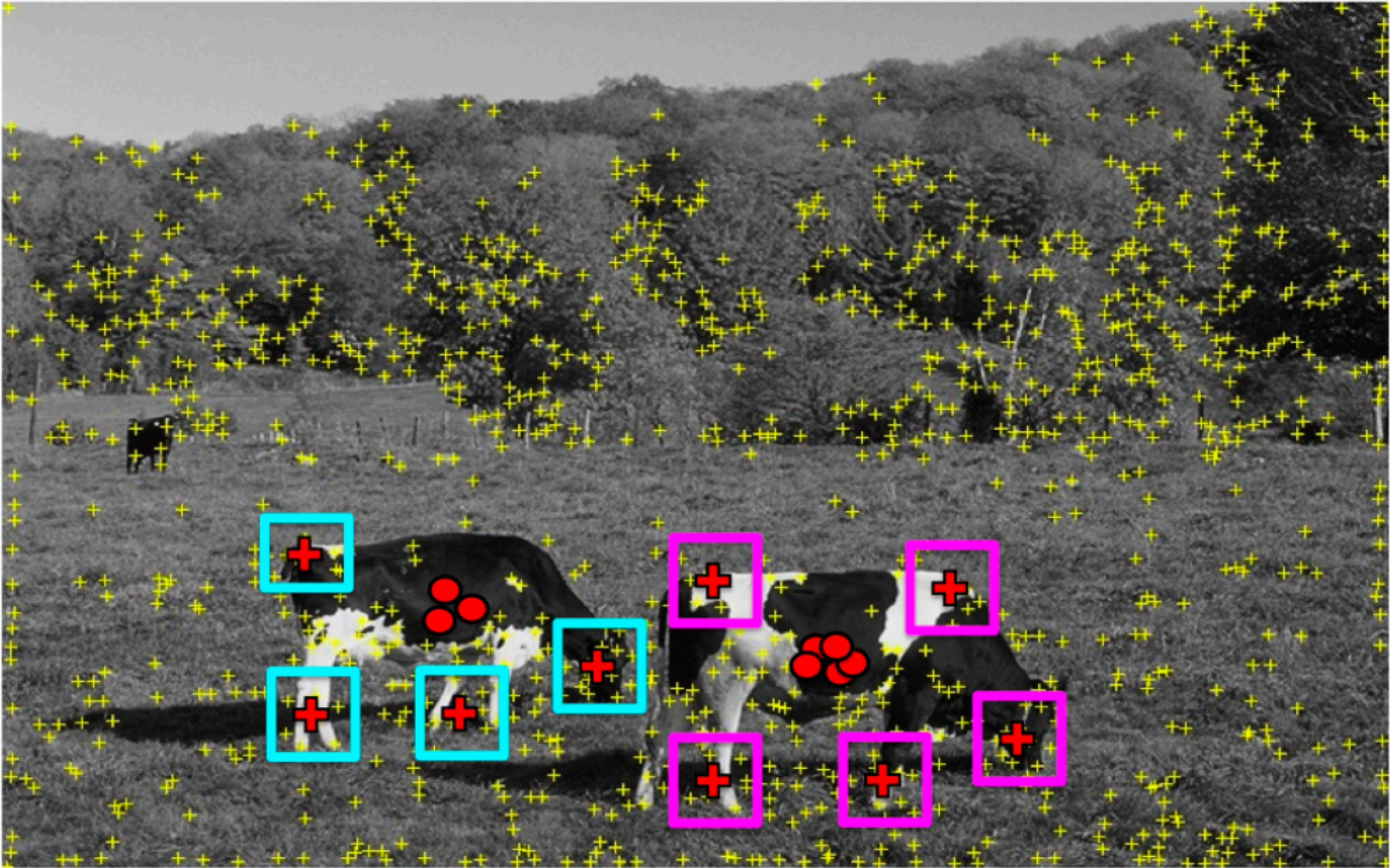
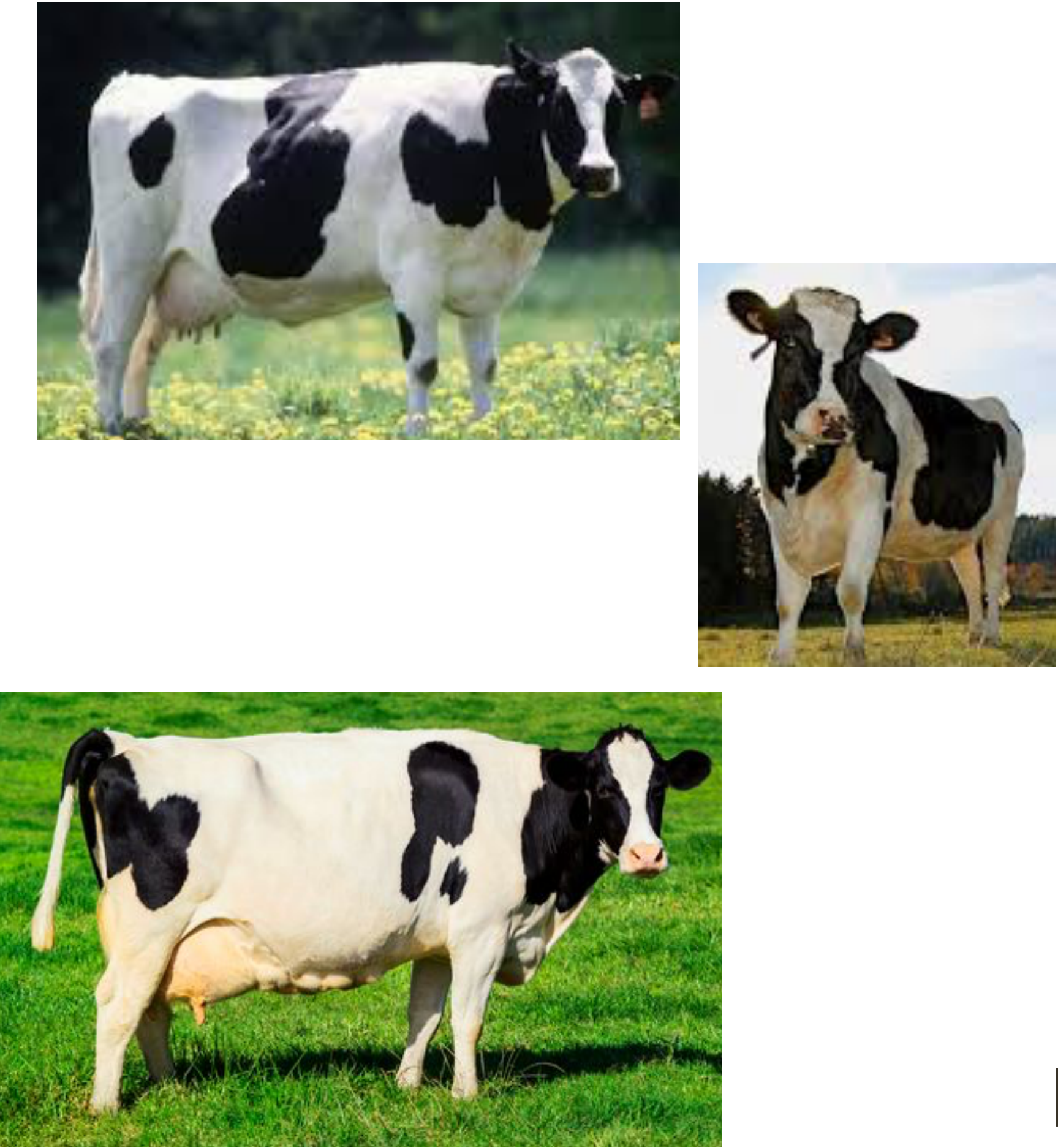
That's ok. We want only **peaks** in voting space.



# Example 1: Object Recognition — Implicit Shape Model

“Training” images of cows

“Testing” image



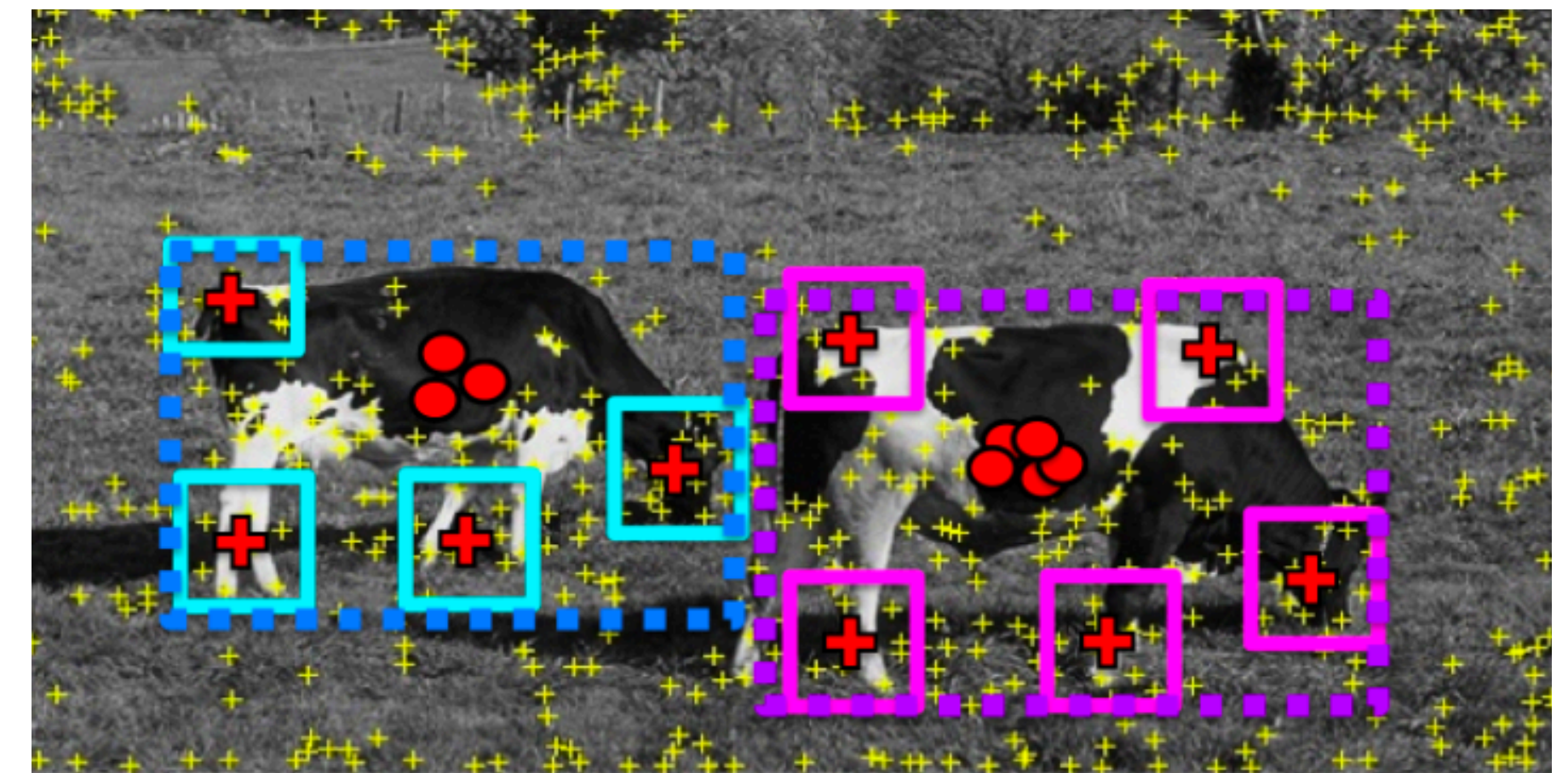
Find patches that voted for the peaks (back-project)

\* Slide from Sanja Fidler



# Example 1: Object Recognition — Implicit Shape Model

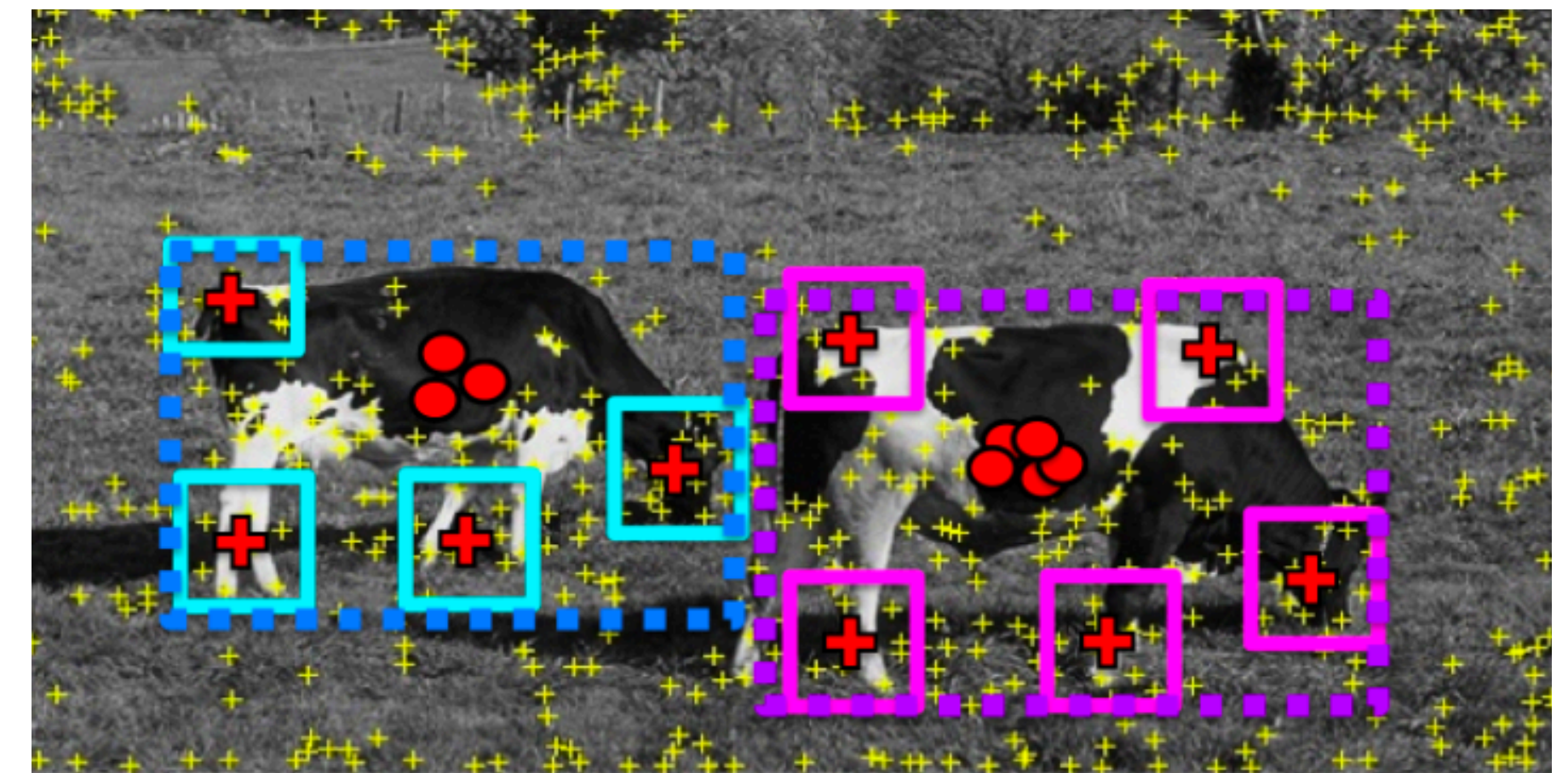
Image Index	Keypoint Index	Keypoint Detection (4D)	Keypoint Description (128D)	Offset to Centroid
Image 1	1	[x, y, s, Theta]	[...]	[x,y]
Image 1	2	[x, y, s, Theta]	[...]	[x,y]
....	....	....	...	...
Image 1	265	[x, y, s, Theta]	[...]	[x,y]
<hr/>				
Image 2	1	[x, y, s, Theta]	[...]	[x,y]
Image 2	2	[x, y, s, Theta]	[...]	[x,y]
...	...	...	...	...
Image 2	645	[x, y, s, Theta]	[...]	[x,y]
<hr/>				
Image K	1	[x, y, s, Theta]	[...]	[x,y]
Image K	2	[x, y, s, Theta]	[...]	[x,y]
...	...	...	...	...
Image K	134	[x, y, s, Theta]	[...]	[x,y]





# Example 1: Object Recognition — Implicit Shape Model

Image Index	Keypoint Index	Keypoint Detection (4D)	Keypoint Description (128D)	Offset to Centroid
Image 1	1	[x, y, s, Theta]	[...]	[x,y]
Image 1	2	[x, y, s, Theta]	[...]	[x,y]
....	....	....	...	...
Image 1	265	[x, y, s, Theta]	[...]	[x,y]
<hr/>				
Image 2	1	[x, y, s, Theta]	[...]	[x,y]
Image 2	2	[x, y, s, Theta]	[...]	[x,y]
...	...	...	...	...
Image 2	645	[x, y, s, Theta]	[...]	[x,y]
<hr/>				
Image K	1	[x, y, s, Theta]	[...]	[x,y]
Image K	2	[x, y, s, Theta]	[...]	[x,y]
...	...	...	...	...
Image K	134	[x, y, s, Theta]	[...]	[x,y]



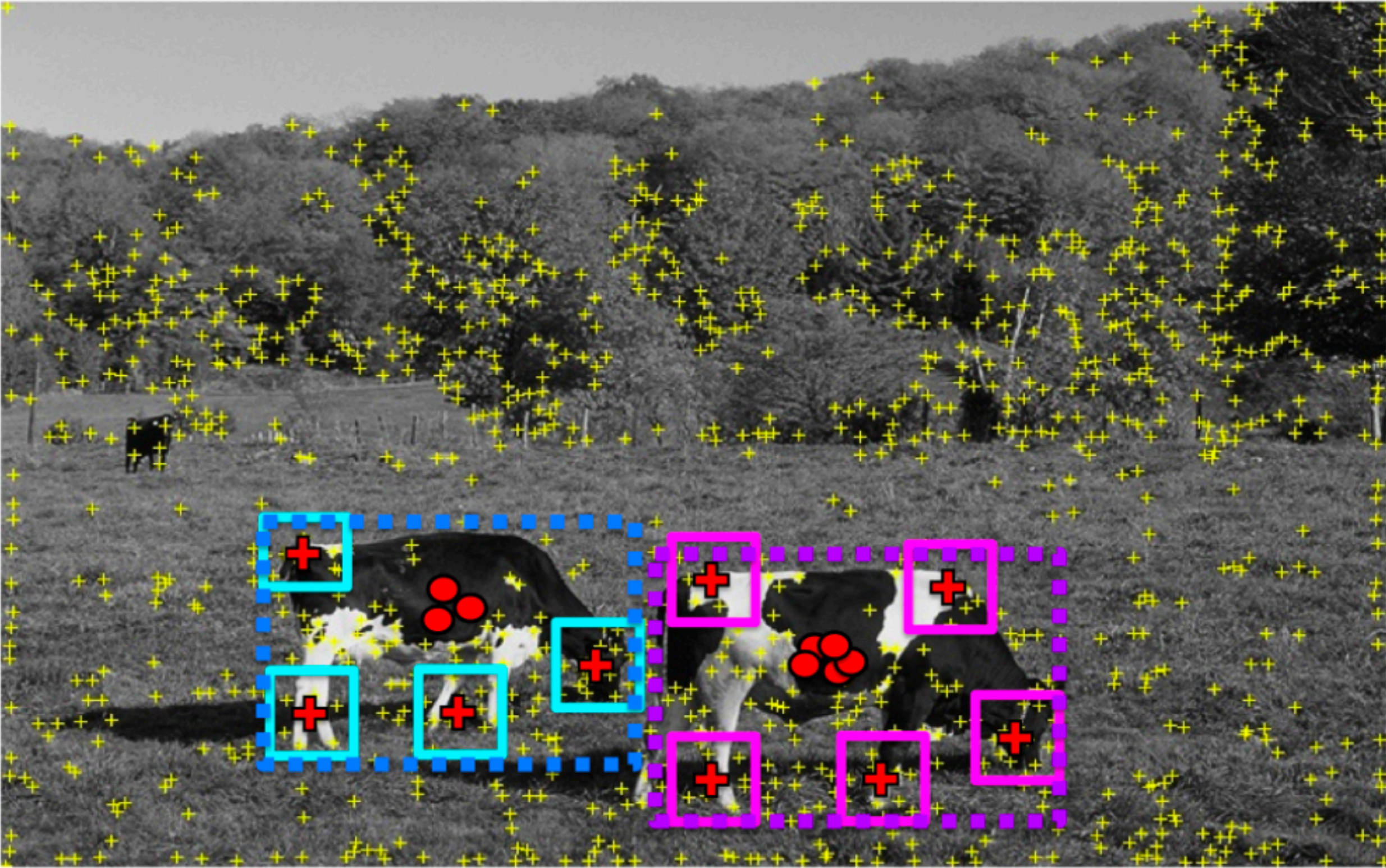


# Example 1: Object Recognition — Implicit Shape Model

“Training” images of cows

“Testing” image

box around patches = object



Find objects based on the back projected patches

\* Slide from Sanja Fidler

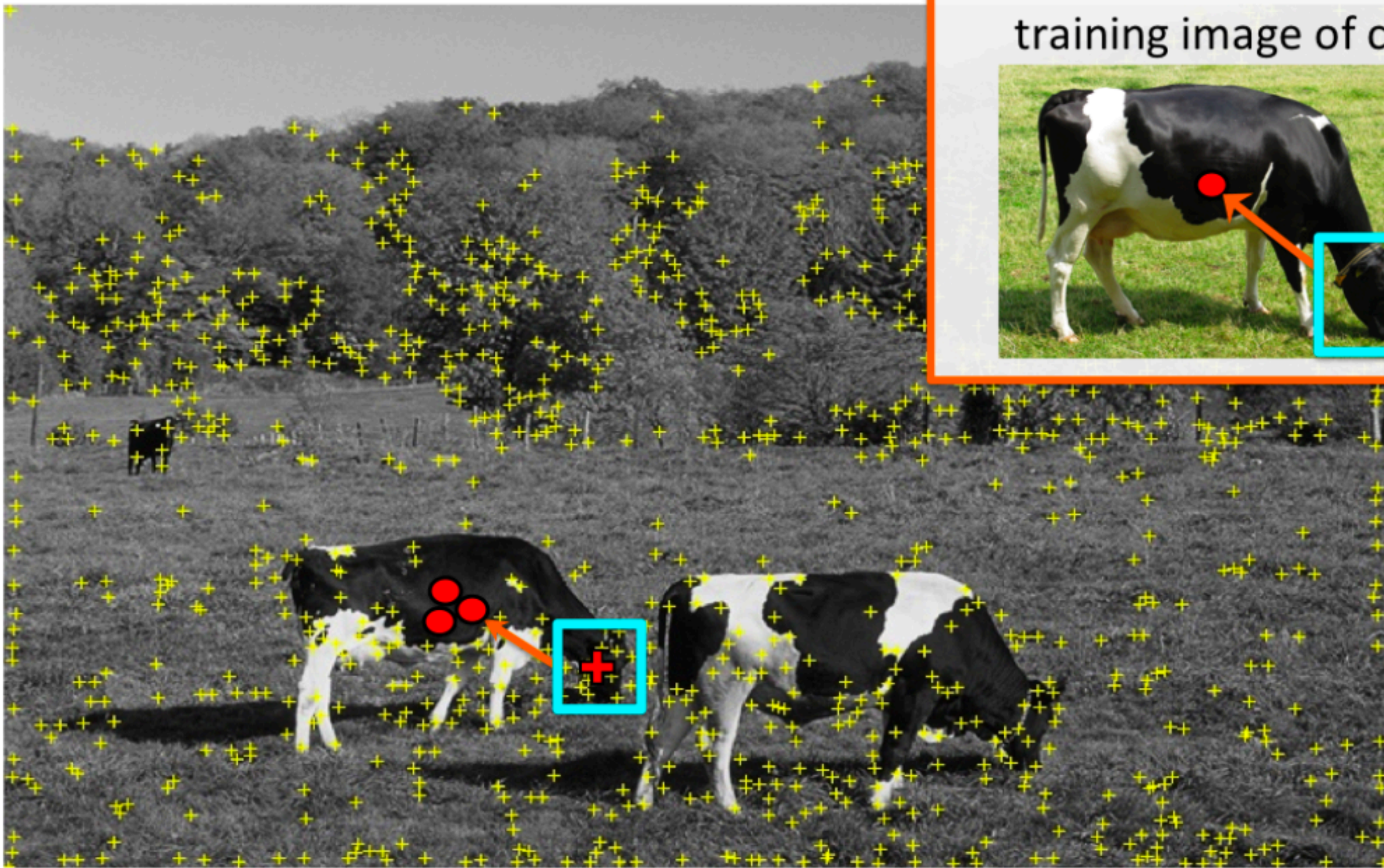


# Example 1: Object Recognition — Implicit Shape Model

“Training” images of cows

“Testing” image

Really easy ... but slow ... how do we make it fast?



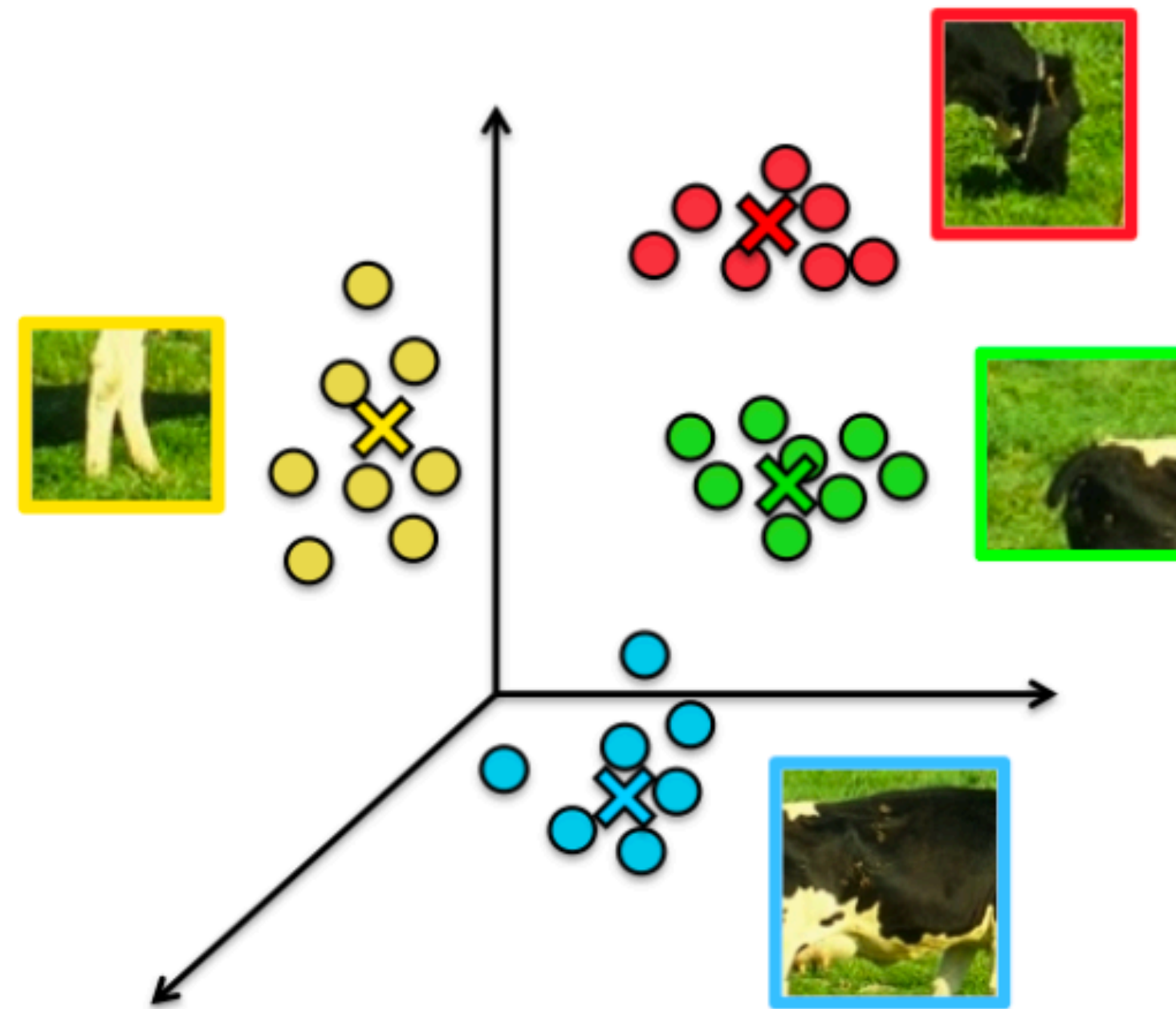
We need to match a patch around each yellow keypoint to all patches in all training images (**slow**)

\* Slide from Sanja Fidler



# Visual Words

- **Visual vocabulary** (we saw this for retrieval)
- Compare each patch to a small set of visual words (clusters)

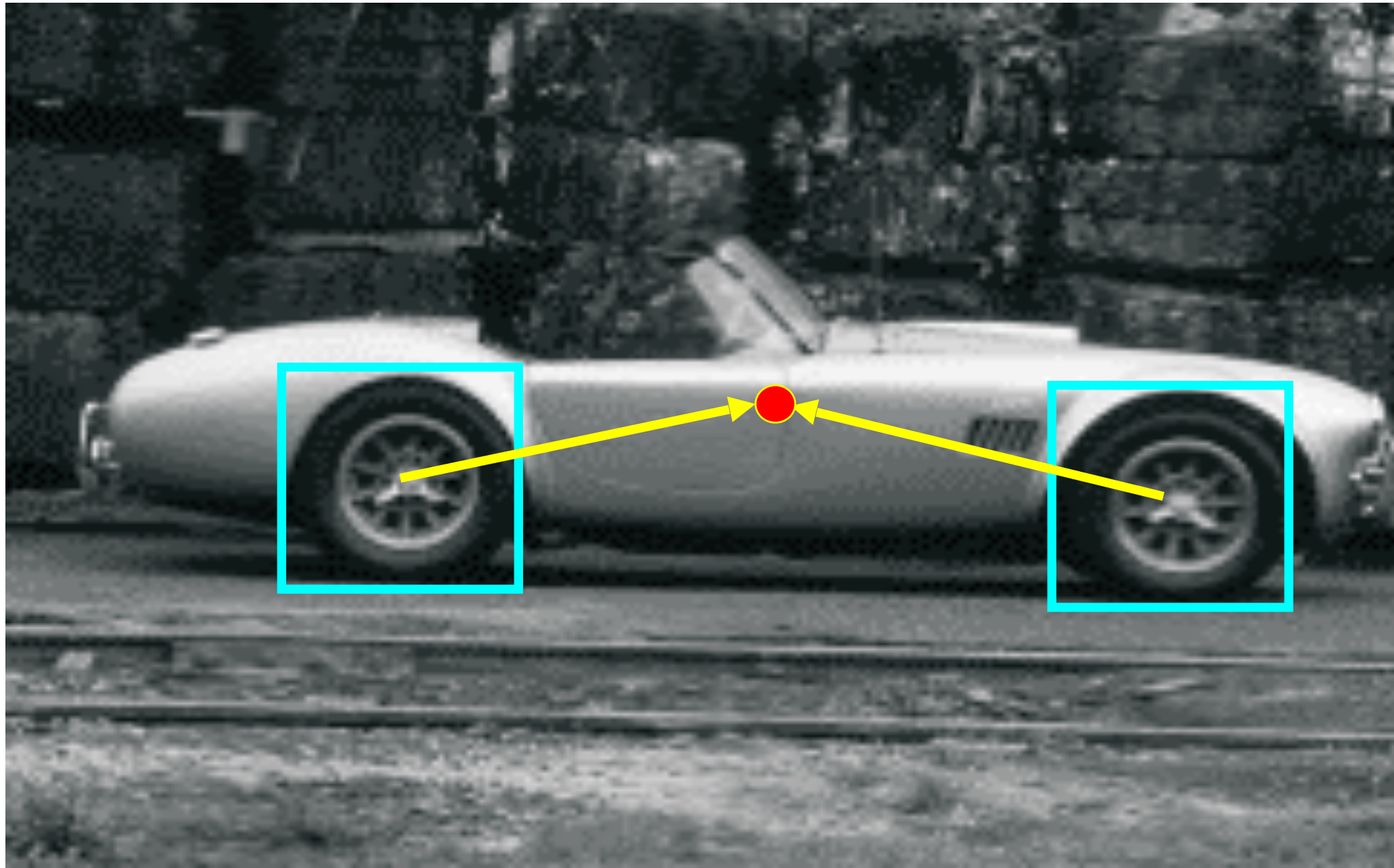


**Visual words (visual codebook)!**

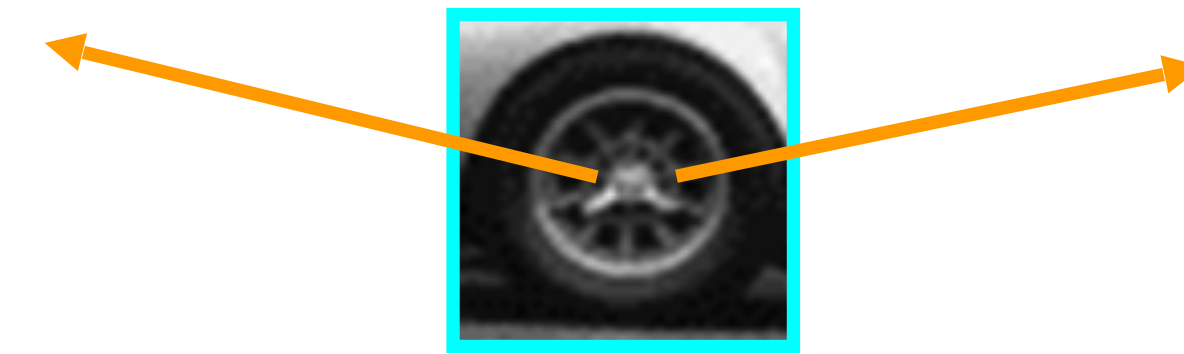


# Example 1: Object Recognition — Implicit Shape Model

Index displacements by “visual codeword”



training image



visual codeword with displacement vectors



# Example 1: Object Recognition — Implicit Shape Model

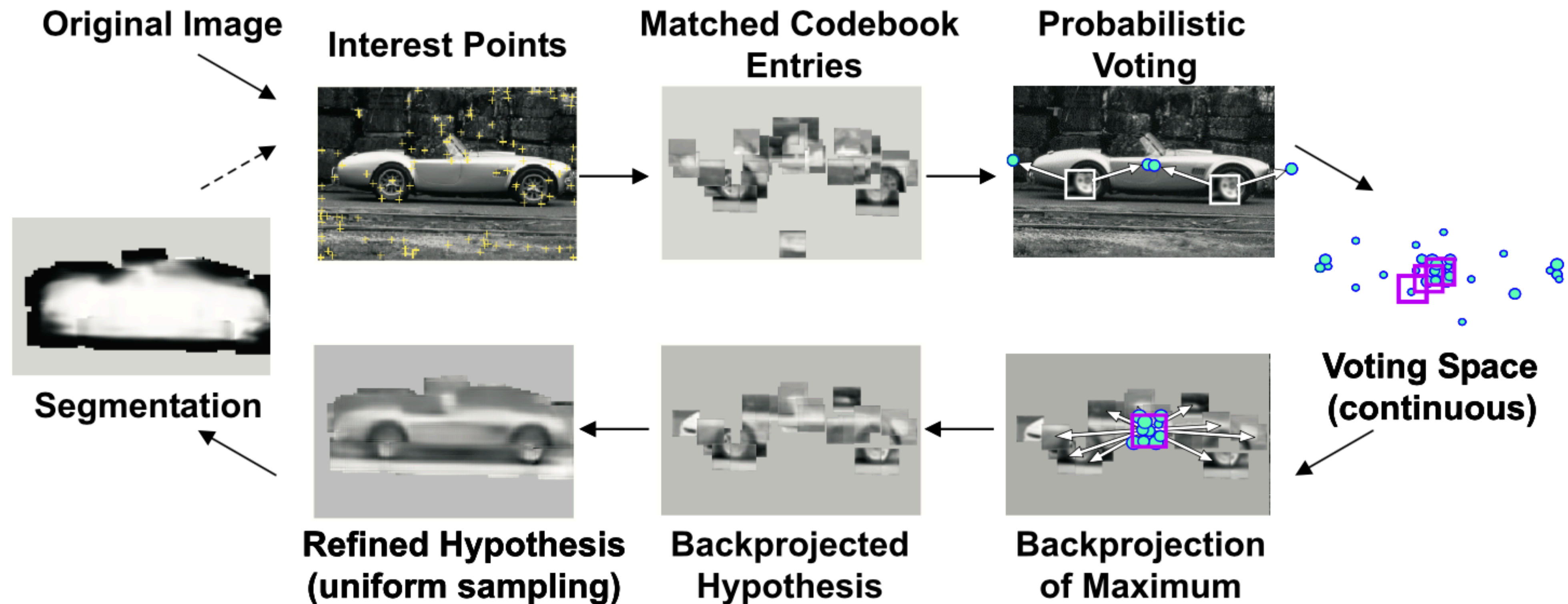


B. Leibe, A. Leonardis, and B. Schiele, Combined Object Categorization and Segmentation with an Implicit Shape Model, ECCV Workshop on Statistical Learning in Computer Vision 2004



# Inferring Other Information: **Segmentation**

Combined object detection and segmentation using an implicit shape model. Image patches cast weighted votes for the object centroid.





# Example 1: Object Recognition — Implicit Shape Model

“Training” images of cows

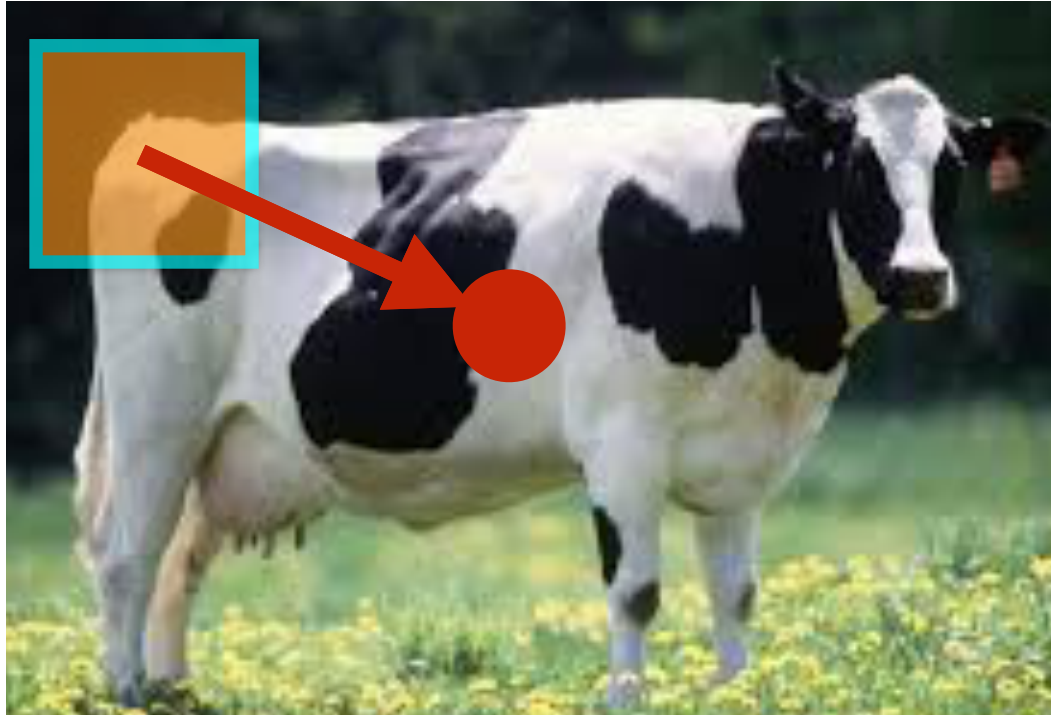


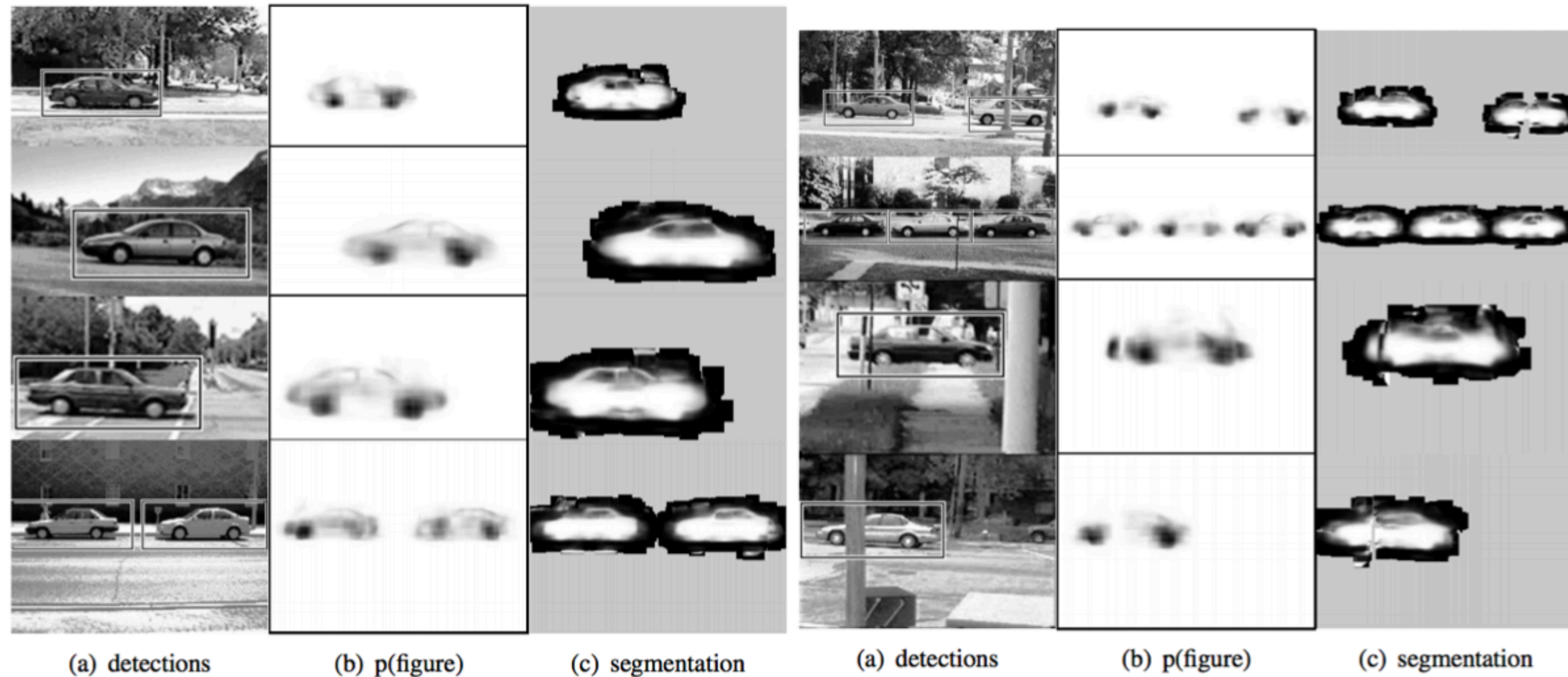
Image Index	Keypoint Index	Keypoint Detection (4D)	Keypoint Description (128D)	Offset to Centroid	Segment
Image 1	1	[x, y, s, Theta]	[...]	[x,y]	
Image 1	2	[x, y, s, Theta]	[...]	[x,y]	
....	....	....	...	...	
Image 1	265	[x, y, s, Theta]	[...]	[x,y]	
Image 2	1	[x, y, s, Theta]	[...]	[x,y]	
Image 2	2	[x, y, s, Theta]	[...]	[x,y]	
...	...	...	...	...	
Image 2	645	[x, y, s, Theta]	[...]	[x,y]	
Image K	1	[x, y, s, Theta]	[...]	[x,y]	
Image K	2	[x, y, s, Theta]	[...]	[x,y]	
...	...	...	...	...	
Image K	134	[x, y, s, Theta]	[...]	[x,y]	

\* Slide from Sanja Fidler



# Inferring Other Information: **Segmentation**

**Idea:** When back-projecting, back-project labeled segmentations per training patch



[Source: B. Leibe]

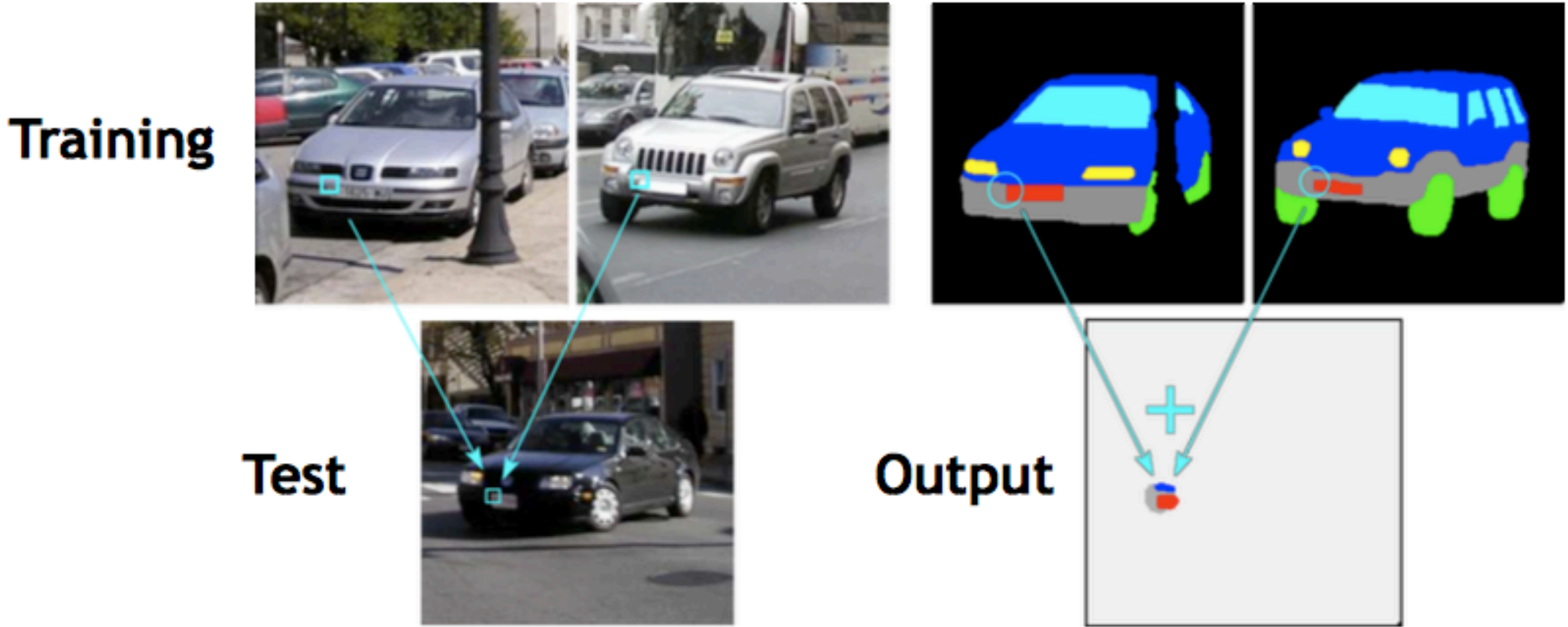
# Inferring Other Information: **Segmentation**



[Source: B. Leibe]



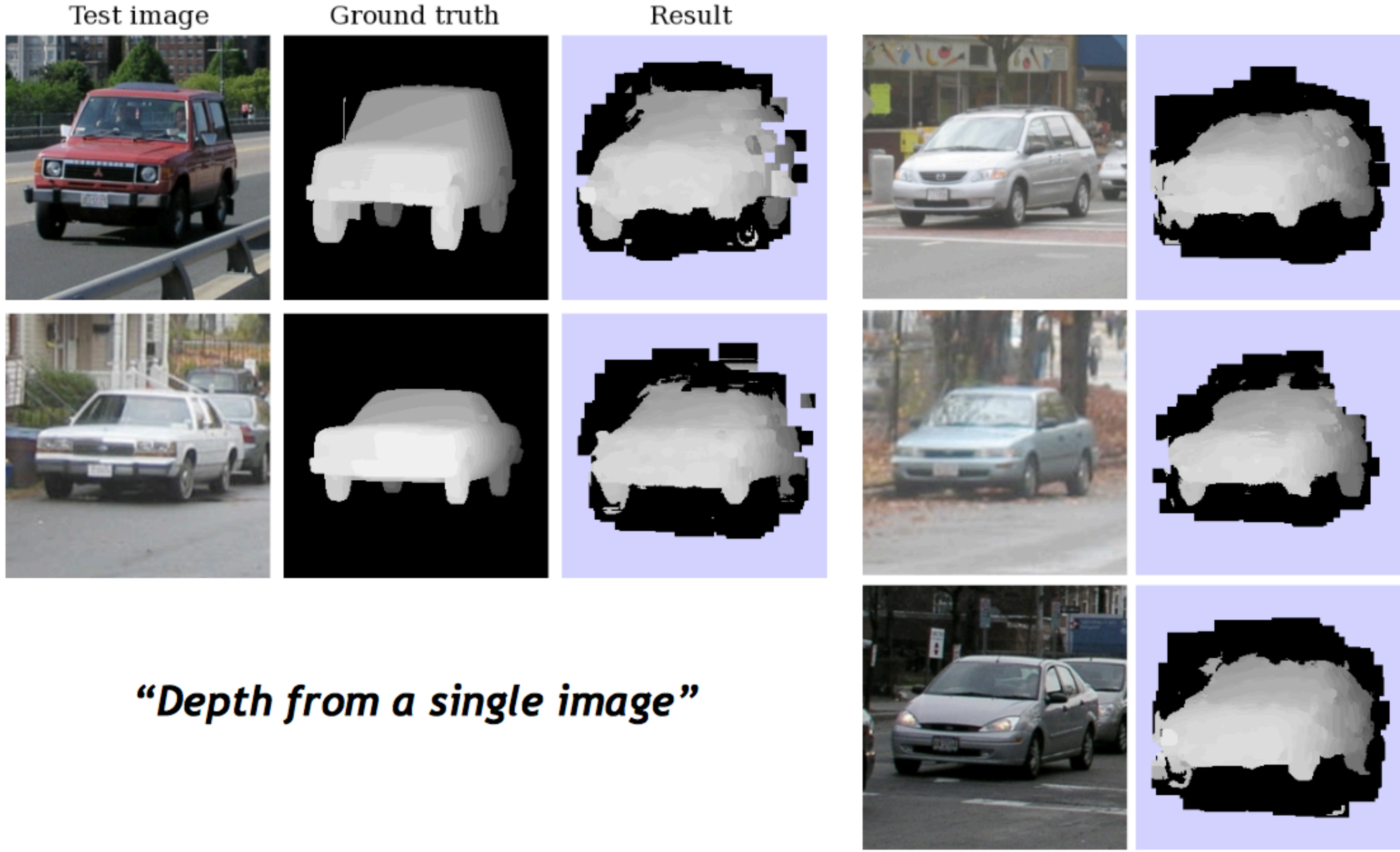
# Inferring Other Information: **Part Labels**



\* Slide from Sanja Fidler



# Inferring Other Information: **Depth**



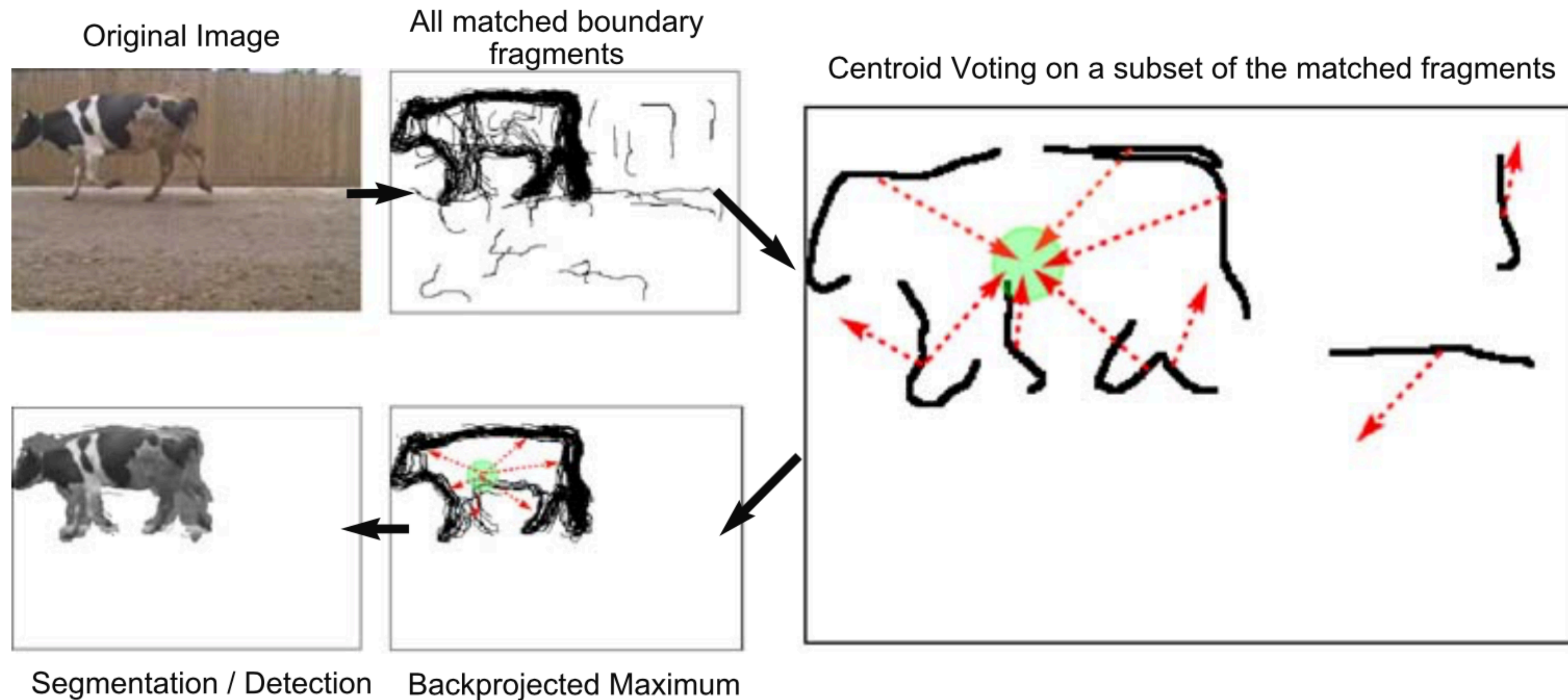
***“Depth from a single image”***

\* Slide from Sanja Fidler



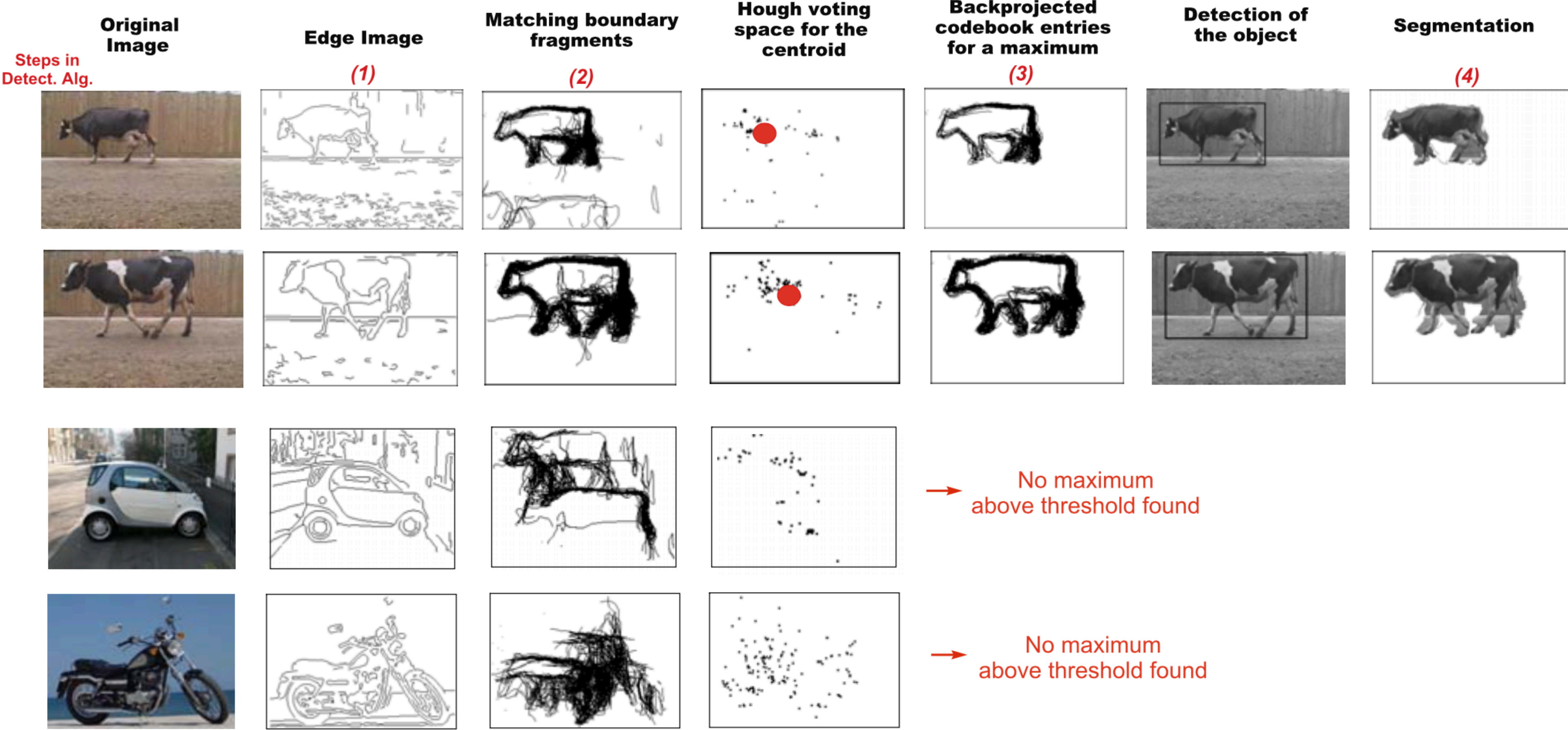
# Example 2: Object Recognition — Boundary Fragments

**Boundary fragments** cast weighted votes for the object centroid. Also obtains an estimate of the object's contour.



# Example 2: Object Recognition – Boundary Fragments

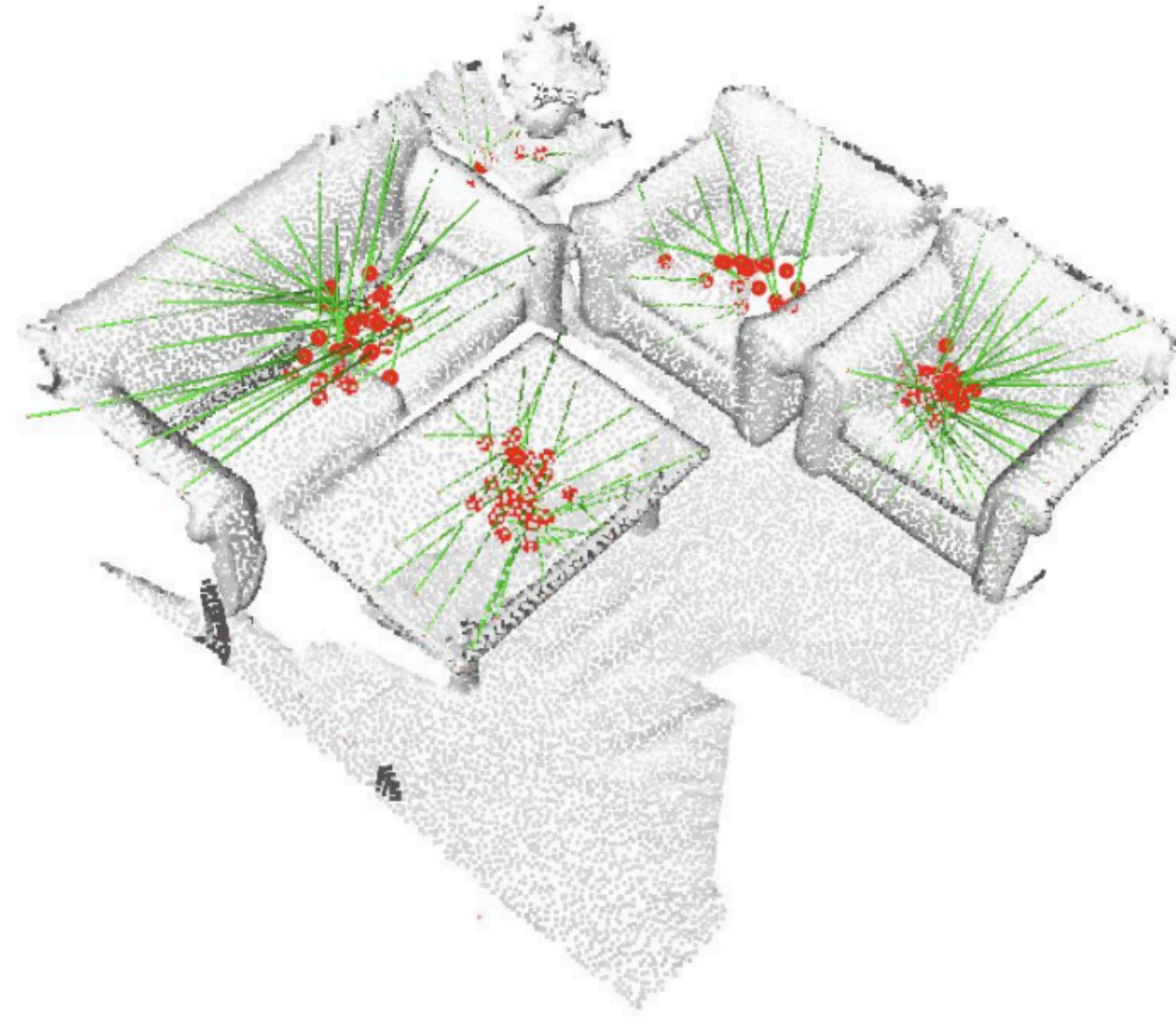
**Boundary fragments** cast weighted votes for the object centroid. Also obtains an estimate of the object's contour.





# Example 3: Deep Hough Voting

Voting from input point cloud



3D detection output

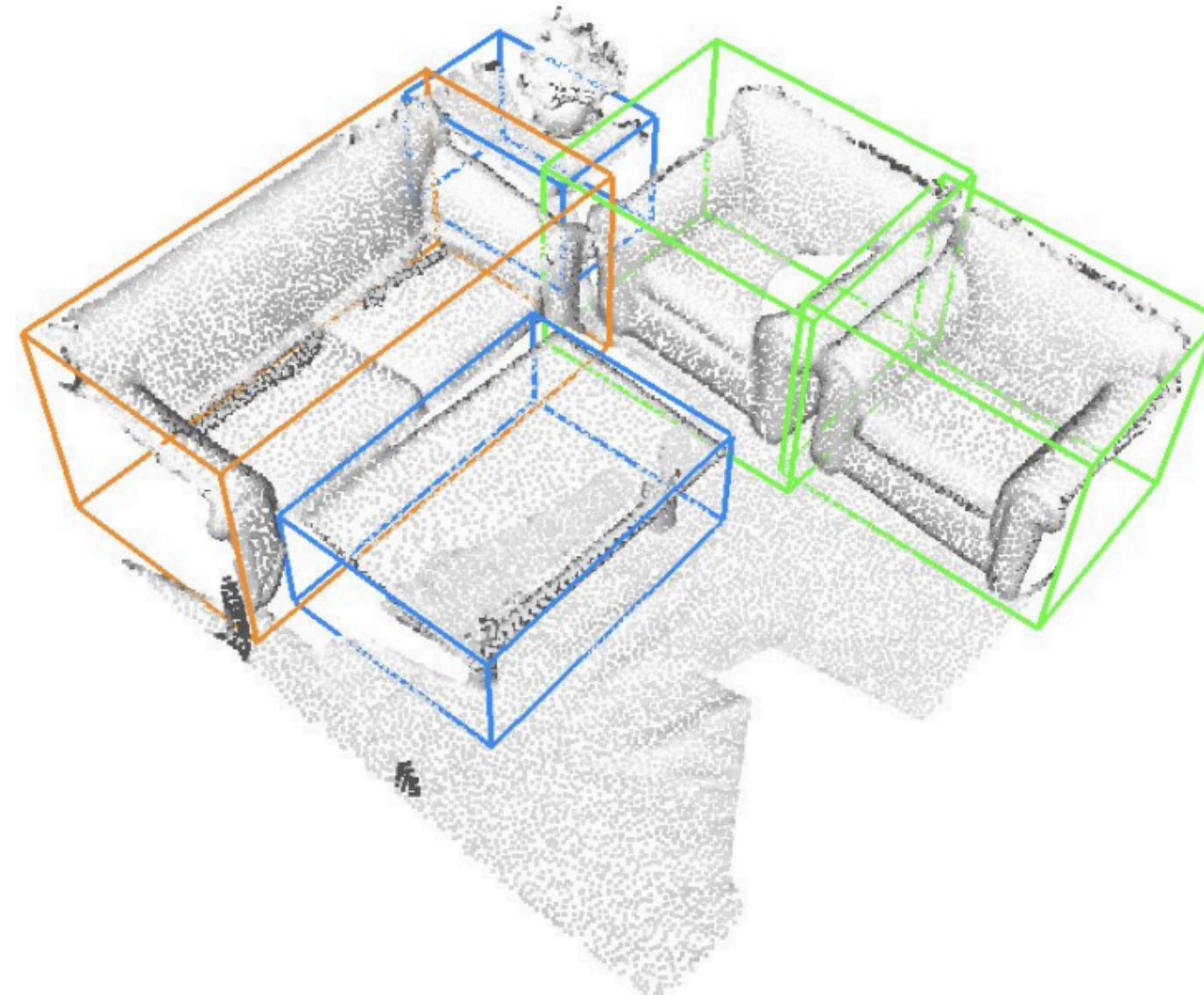


Figure 1. **3D object detection in point clouds with a deep Hough voting model.** Given a point cloud of a 3D scene, our VoteNet votes to object centers and then groups and aggregates the votes to predict 3D bounding boxes and semantic classes of objects.



# Example 3: Deep Hough Voting

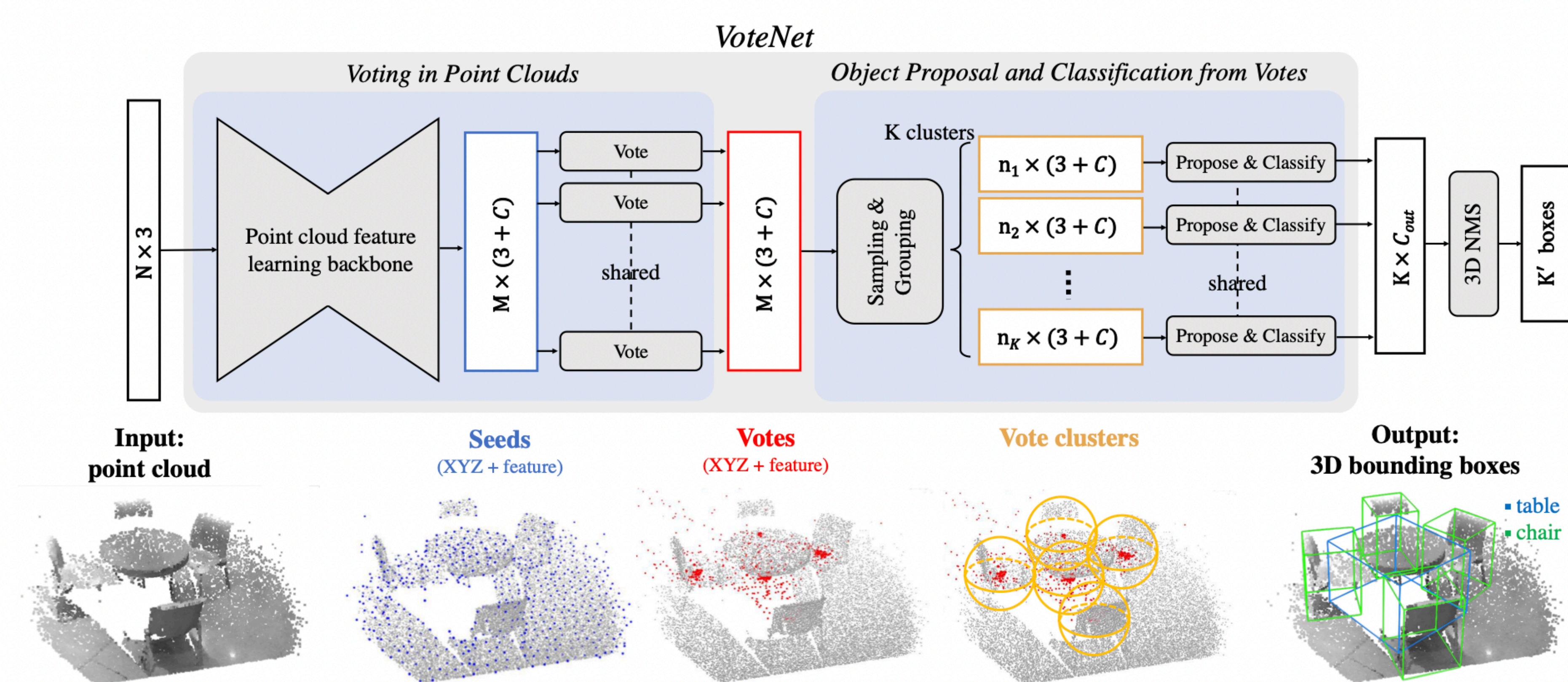


Figure 2. **Illustration of the VoteNet architecture** for 3D object detection in point clouds. Given an input point cloud of  $N$  points with XYZ coordinates, a backbone network (implemented with PointNet++ [36] layers) subsamples and learns deep features on the points and outputs a subset of  $M$  points but extended by  $C$ -dim features. This subset of points are considered as seed points. Each seed independently generates a vote through a voting module. Then the votes are grouped into clusters and processed by the proposal module to generate the final proposals. The classified and NMSed proposals become the final 3D bounding boxes output. Image best viewed in color.



# Summary of Hough Transform

## Idea of **Hough transform**:

- For each token vote for all models to which the token could belong
- Return models that get many votes

e.g., For each point, vote for all lines that could pass through it; the true lines will pass through many points and so receive many votes

## **Advantages:**

- Can handle high percentage of outliers: each point votes separately
- Can detect multiple instances of a model in a single pass

## **Disadvantages:**

- Search time increases exponentially with the number of model parameters
- Can be tricky to pick a good bin size