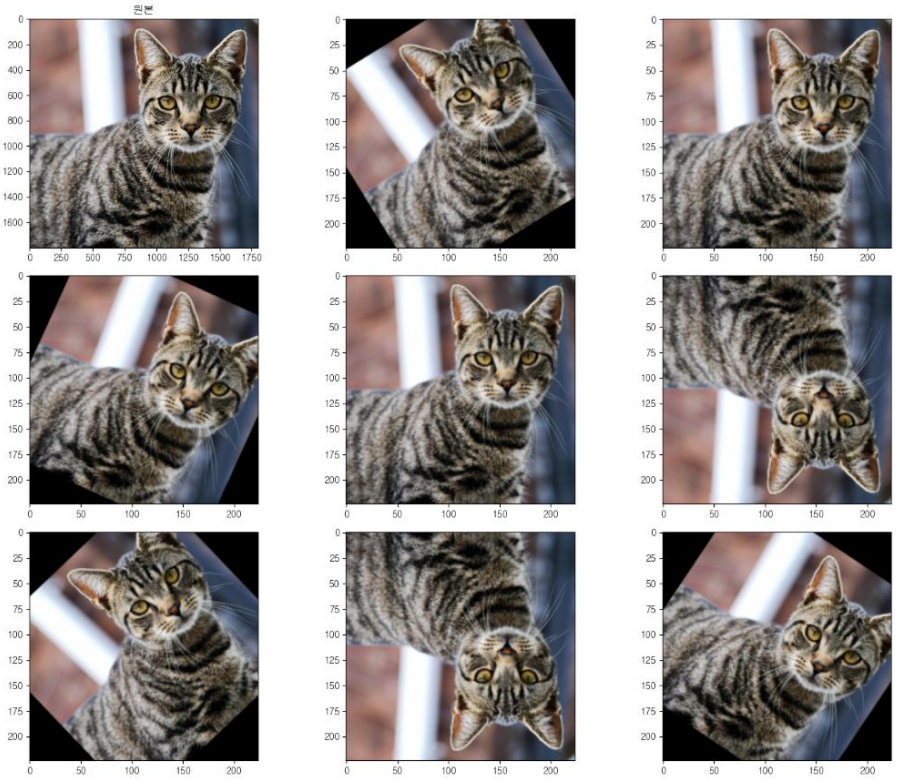
CS-GY 6643
Computer Vision

Lecture 5: Image transforms

Prof. Erdem Varol



Today's menu

Announcements

Model fitting

- RANSAC
- Hough transform

Coding lab

Announcements

Project 1 due Oct 4

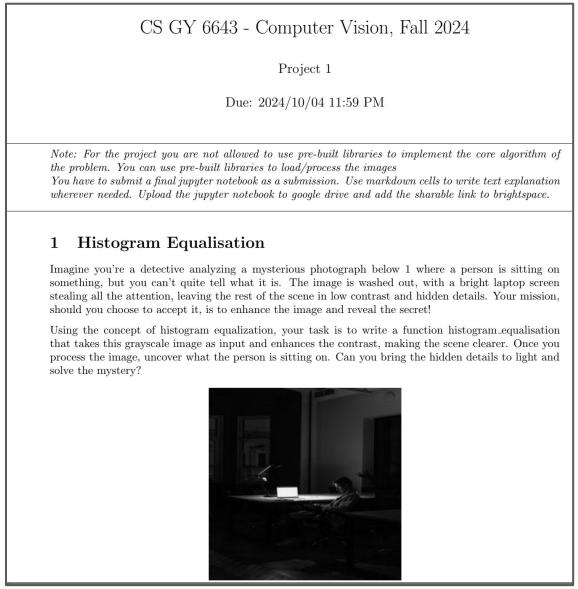
15 % of total course grade.

A python notebook with executed outputs + a pdf write up of each step explained to be submitted on brightspace.

You are encouraged to discuss with other students and look online for examples.

Project must be submitted individually (**We will check for blatant code copying**).

1 point will be deducted for every hour the project is late (rounded down).



CS GY 6643 - Computer Vision, Fall 2024

Project 1

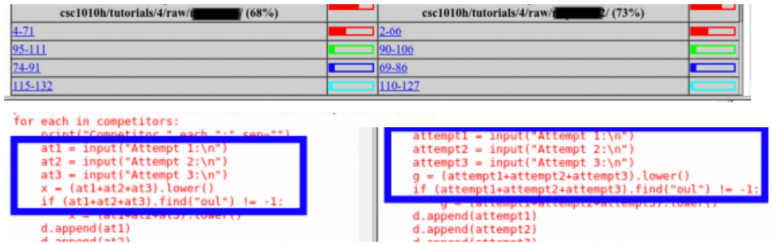
Due: 2024/10/04 11:59 PM

Note: For the project you are not allowed to use pre-built libraries to implement the core algorithm of the problem. You can use pre-built libraries to load/process the images
You have to submit a final jupyter notebook as a submission. Use markdown cells to write text explanation wherever needed. Upload the jupyter notebook to google drive and add the sharable link to brightspace.

1 Histogram Equalisation

Imagine you're a detective analyzing a mysterious photograph below 1 where a person is sitting on something, but you can't quite tell what it is. The image is washed out, with a bright laptop screen stealing all the attention, leaving the rest of the scene in low contrast and hidden details. Your mission, should you choose to accept it, is to enhance the image and reveal the secret!

Using the concept of histogram equalization, your task is to write a function histogram_equalisation that takes this grayscale image as input and enhances the contrast, making the scene clearer. Once you process the image, uncover what the person is sitting on. Can you bring the hidden details to light and solve the mystery?



Final project proposal due Oct 10

Additional office hours to discuss project directions

Friday 10/5 - 10a-1pm

Monday 10/7 - 10am-1pm

Appointment booking @
https://calendar.app.google/GHgsCYBdyBJmw2m97

20% of your final project grade (4 out of 20 pts)

Proposal template here:

https://www.overleaf.com/read/rhdrqnmsyzpv#c1b706

(Please copy this template in your own account)

Must complete as a team - one person may submit on brightspace.

Computer Vision CS-GY 6643 - Final Project - Project Title

Jane Doe, jd1234@nyu.edu,John Doe, jd1235@nyu.edu, Bob Doe, bd1234@nyu.edu, Mary Doe, md1234@nyu.edu

1 Introduction and background

Introduce the problem statement, importance, background on how others have tried to tackle it. Cite references (Bommasani et al. 2021; Han et al. 2021). If we were to solve this problem, what can we gain scientifically and methodologically? Why has this problem been difficult to solve before and why do you think now is the time to solve it (e.g. before we didn't have the right type of sensors or computing resources, now we do). Briefly introduce how your proposed approach is going to overcome the previous obstacles.

2 Datasets

Describe the datasets that will be used for this study. Generate some figures to illustrate how they look like, what kind of noise we expect to see, class imbalance, missing and incomplete features etc.

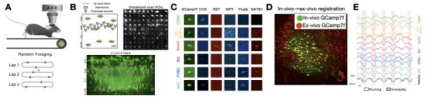


Figure and caption

3 Methods

Introduce the mathematical formulation of the problem (if applicable) or describe the model architecture using diagrams.

4 Evaluation metrics

Which metrics will you use to measure the success of the method. Are there literature evidence to suggest these are appropriate metrics?

5 Baseline methods

Which methods will you show as baselines?

6 Preliminary Results

Show how a prototype of the proposed method can work on the datasets either with real data or using a toy example. Also, show that you are able to run a baseline method here.

7 Final project plan

What did you learn from your preliminary experiments? How will this affect how you will cast your model for the final project? It is important to demonstrate here that based on your preliminary results, your final project is something you can execute in the the next two months.

8 Author contributions

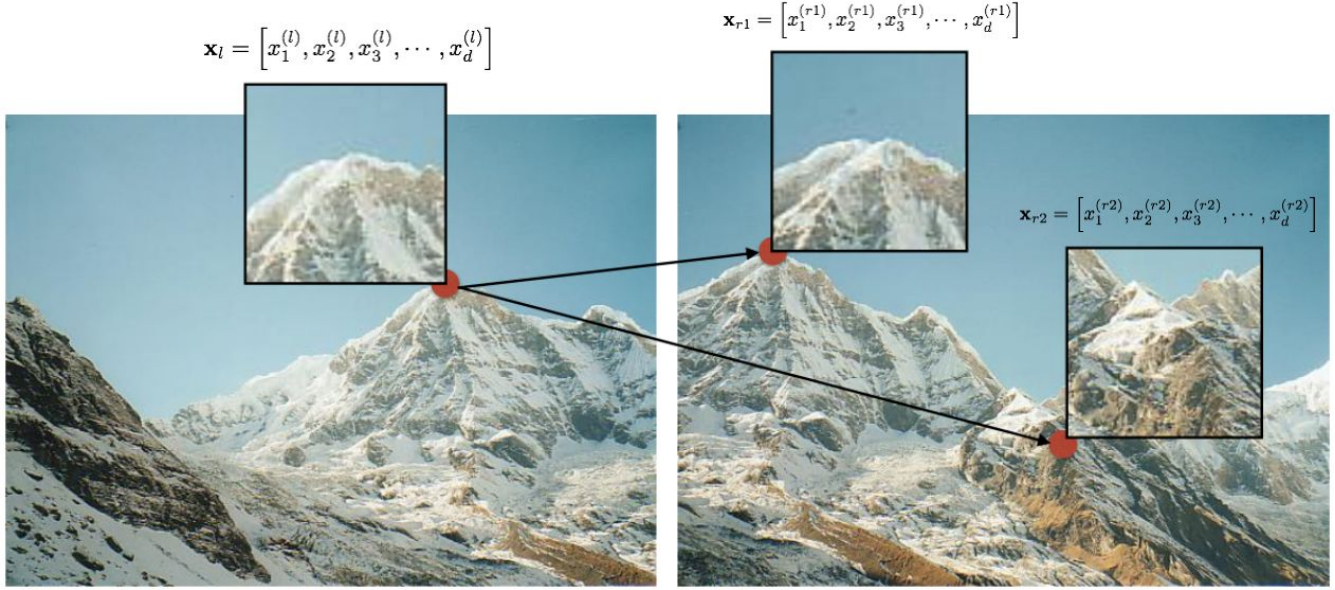
You must explicitly state who in the team is going to execute which parts of the project.

Image transforms

Recall example from previous lecture

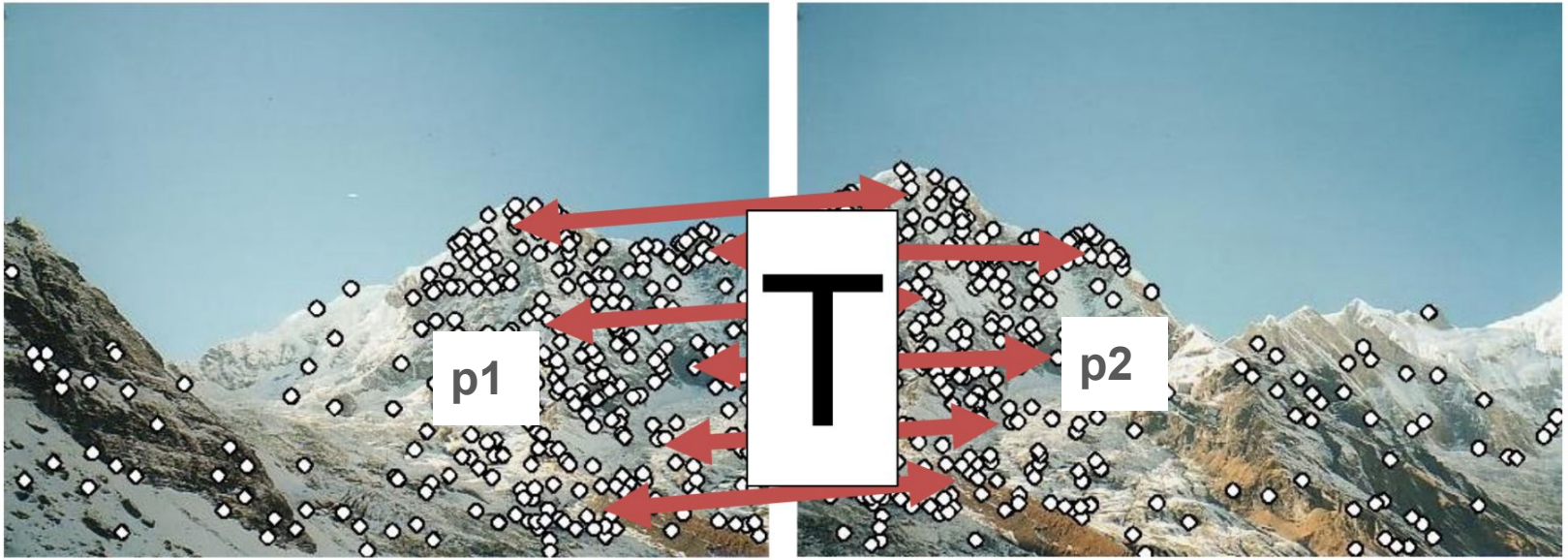
Given these images: how do you align them?

Recap of last lecture



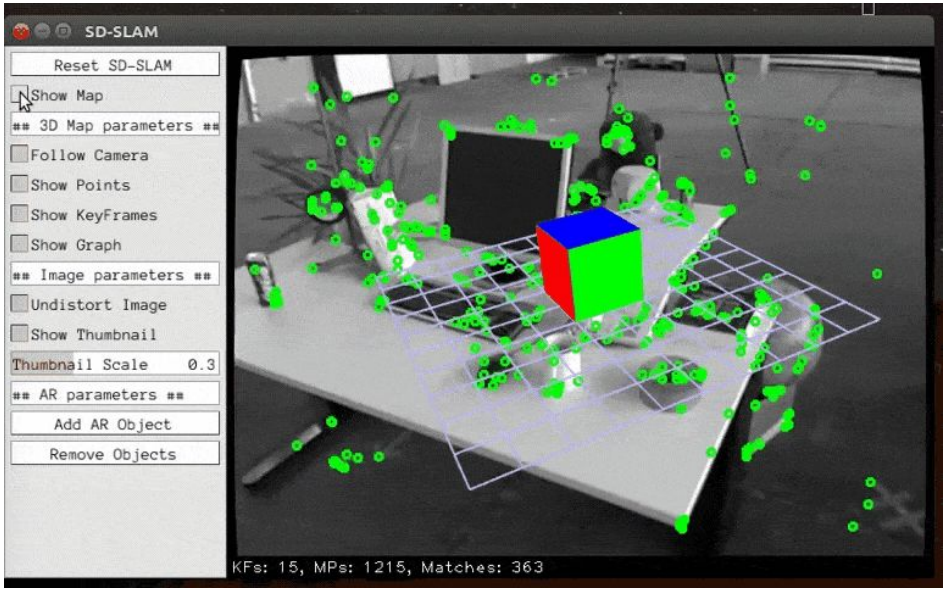
1) Detect “important” points - e.g. Harris corner detector
2) Describe the information round them - e.g. SIFT
3) Match points with similar descriptions

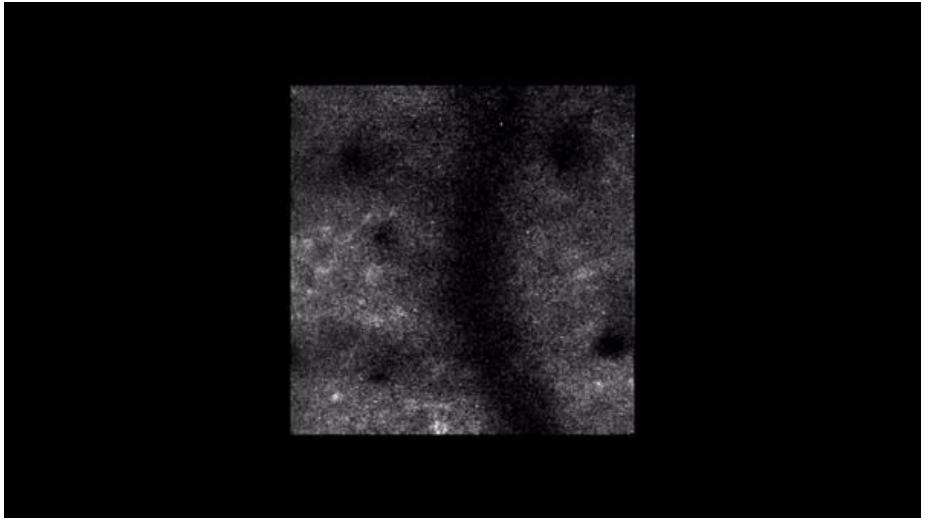
Today



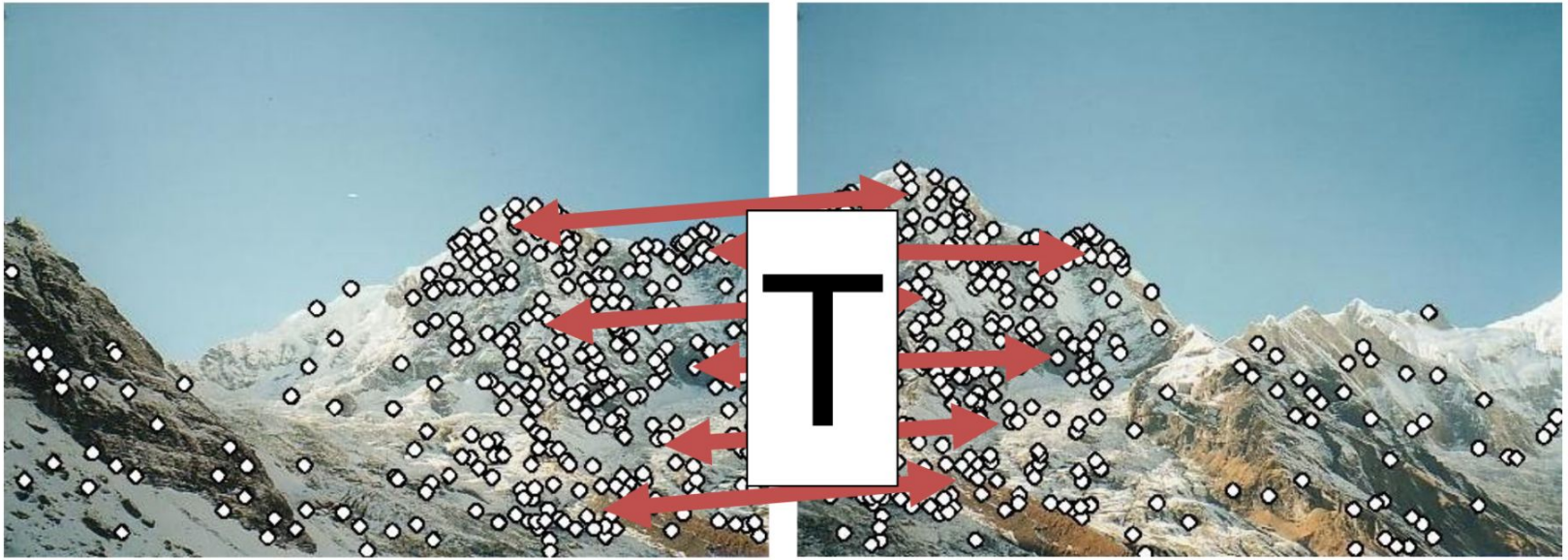
4) Find transformation T such that p1 = Tp2

Some applications - scene reconstruction, multimodal image alignment





How do we go about finding T? Model fitting!



Abstractly: find a **model** that explains the **data** with minimum **error**.

Model Fitting

Need three ingredients

Data: what data are we trying to explain with a model?

Model: what's the compressed, parametric form of the data?

Objective function: given a prediction, how do we evaluate how correct it is?

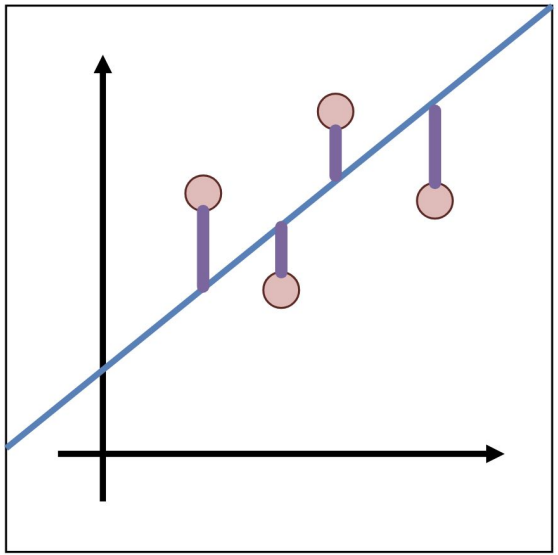
Example: Least-Squares

Fitting a line to data

Data: (x_1,y_1), (x_2,y_2), …, (x_k,y_k)

Model: (m,b) $y_i=mx_i+b$
Or ($\mathbf{w}$) $y_i = \mathbf{w}^T\mathbf{x}_i$

Objective function:
$(y_i - \mathbf{w}^T\mathbf{x}_i)^2$



Least-Squares Setup

$$\sum_{i=1}^{k} \left(y_i - \boldsymbol{w}^T \boldsymbol{x}_i\right)^2 \quad \rightarrow \quad \|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{w}\|_2^2$$

$$\boldsymbol{Y} = \begin{bmatrix} y_1 \\ \vdots \\ y_k \end{bmatrix} \qquad \boldsymbol{X} = \begin{bmatrix} x_1 & 1 \\ \vdots & 1 \\ x_k & 1 \end{bmatrix} \qquad \boldsymbol{w} = \begin{bmatrix} m \\ b \end{bmatrix}$$

Solving Least-Squares

$$\|Y - Xw\|_2^2$$

$$\frac{\partial}{\partial w}\|Y - Xw\|_2^2 = 2X^TXw - 2X^TY$$

Recall: derivative is 0 at a maximum / minimum. Same is true about gradients.

$$\mathbf{0} = 2X^TXw - 2X^TY$$

$$X^TXw = X^TY$$

$$w = (X^TX)^{-1}X^TY$$

Aside: **0** is a vector of 0s. **1** is a vector of 1s.

Two Solutions to Getting W

In One Go

Analytical solution*:

$$\boldsymbol{w} = \left(\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}\right)^{-\mathbf{1}}\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{Y}$$

Iteratively

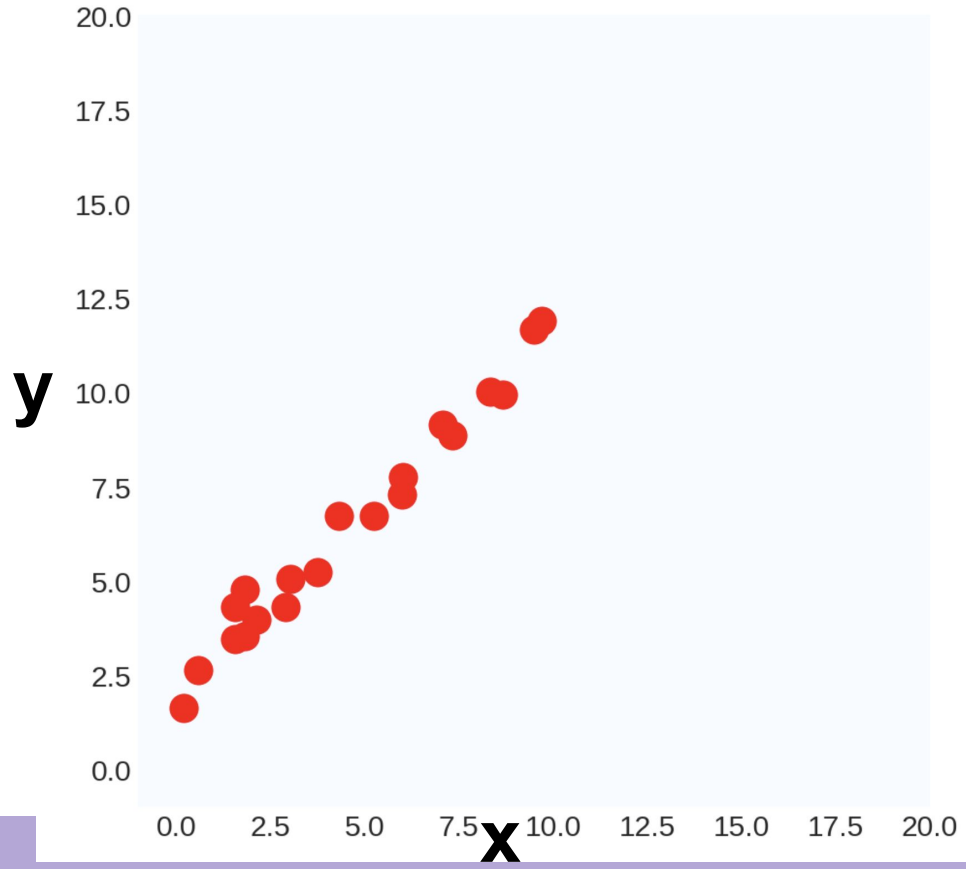
Gradient descent:

$$\boldsymbol{w_0} = \mathbf{0}$$

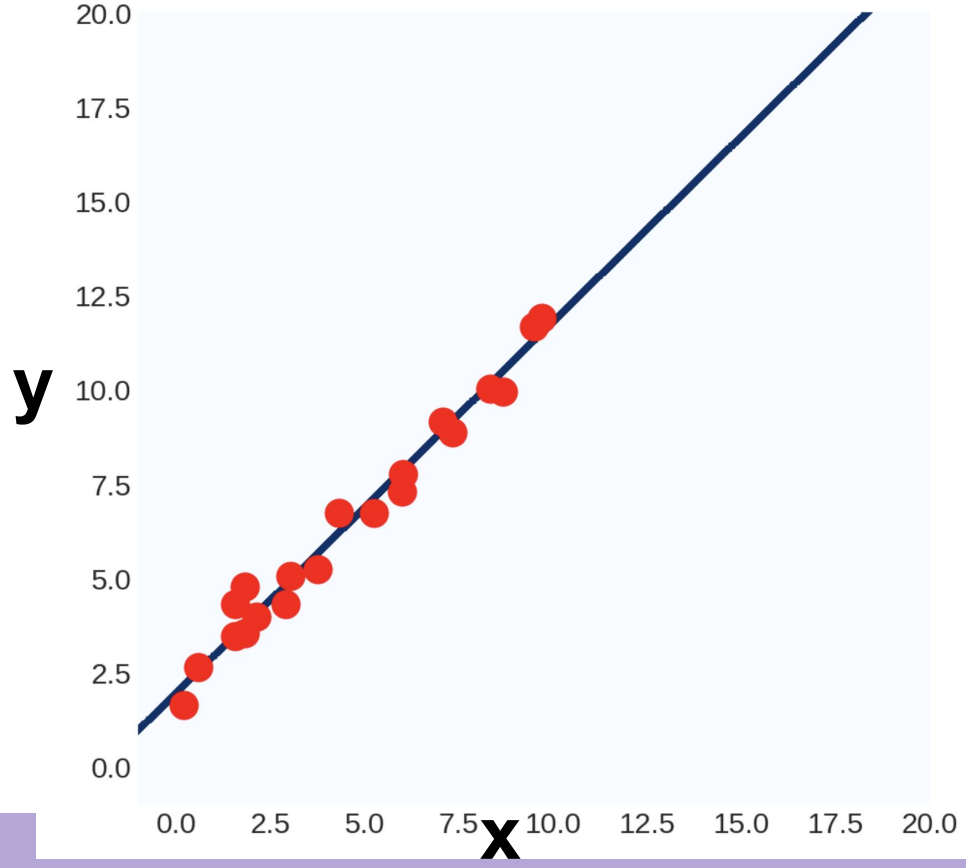
$$\boldsymbol{w_{i+1}} = \boldsymbol{w_i} - \boldsymbol{\gamma}\left(\frac{\partial}{\partial \boldsymbol{w}}\|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{w}\|_2^2\right)$$

*Technically one step of a Newton method

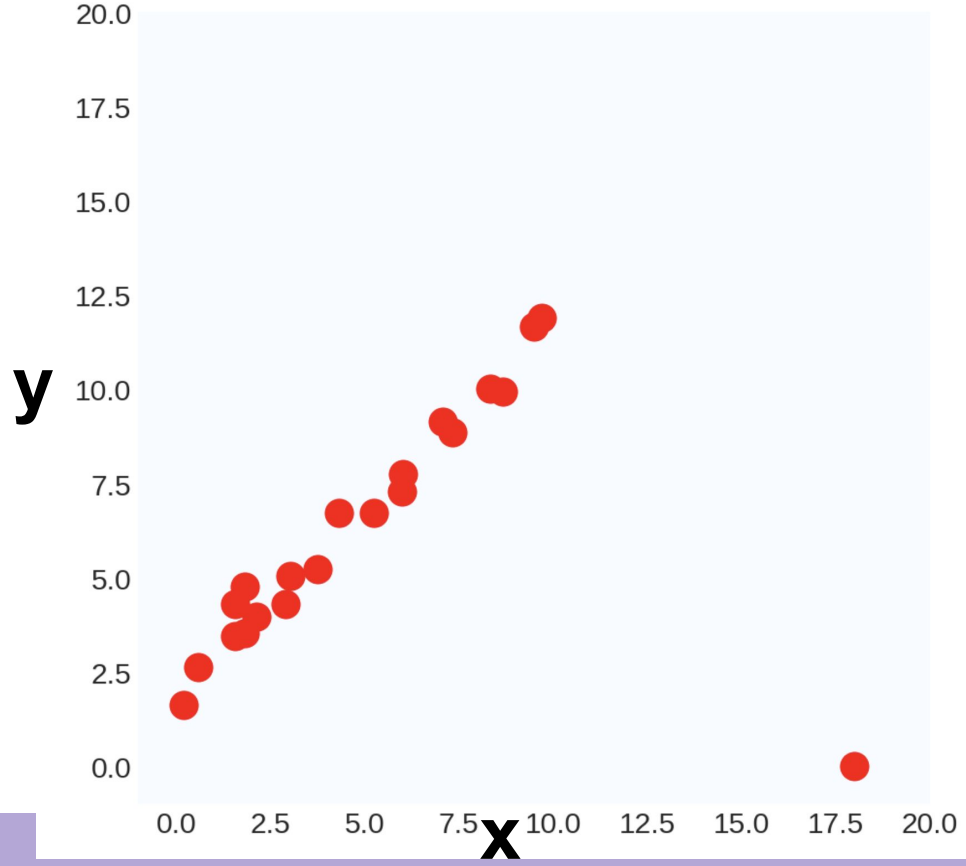
1D data example x ↦y



Model fit: $w = (X^T X)^{-1} X^T Y$



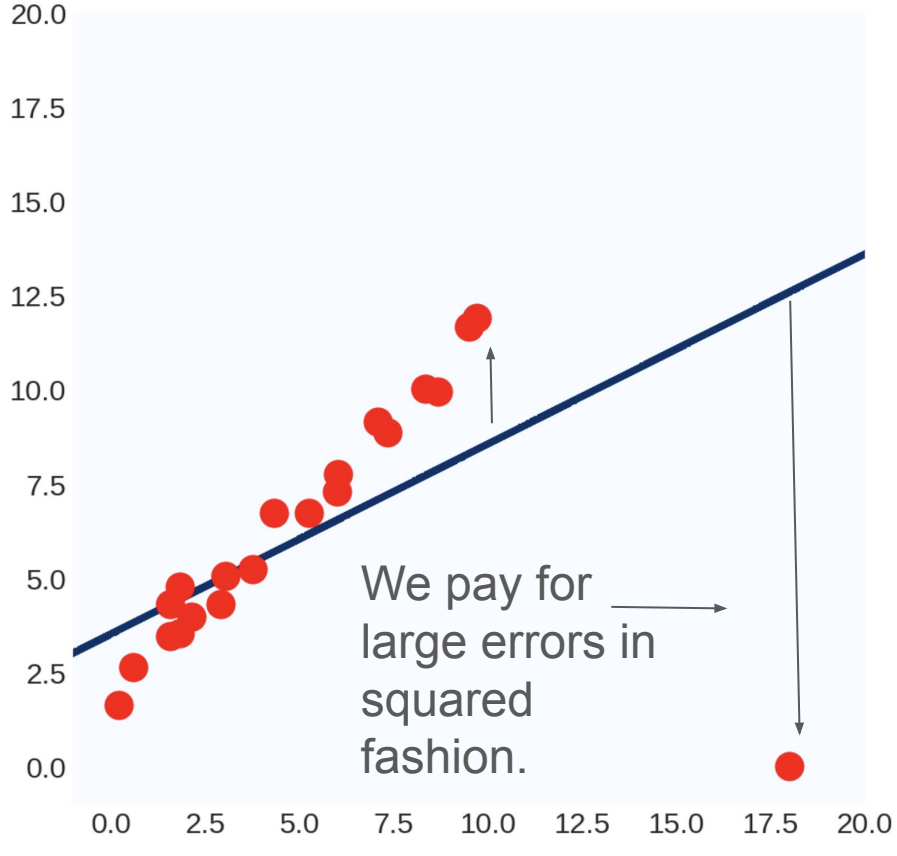
What happens if we have outliers in our data?



Model fit $w = (X^T X)^{-1} X^T Y$ is skewed.

Why?

- One big error offsets many good fits.



$$\|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{w}\|_2^2$$

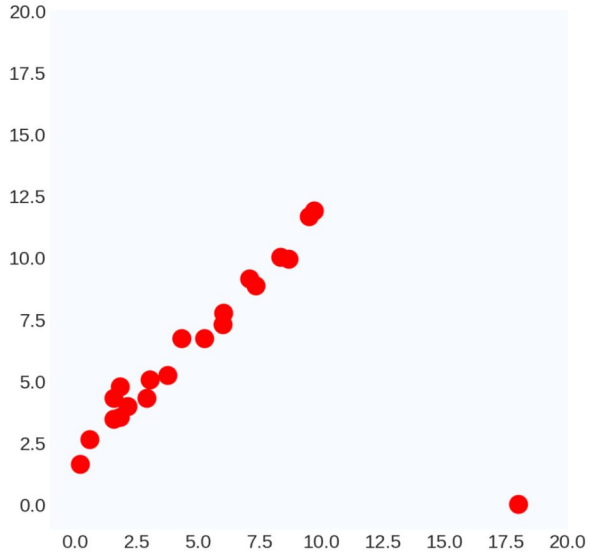
100^2 >> 10^2: least-squares prefers having no large errors, even if the model is useless overall

$$\boldsymbol{w} = \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{Y}$$

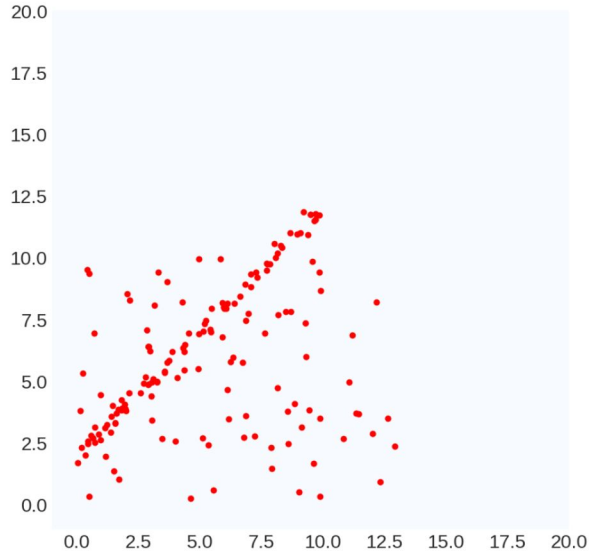
Weights are a linear transformation of the output variable: can manipulate W by manipulating Y.

Outliers in Computer Vision

Single outlier: *rare*



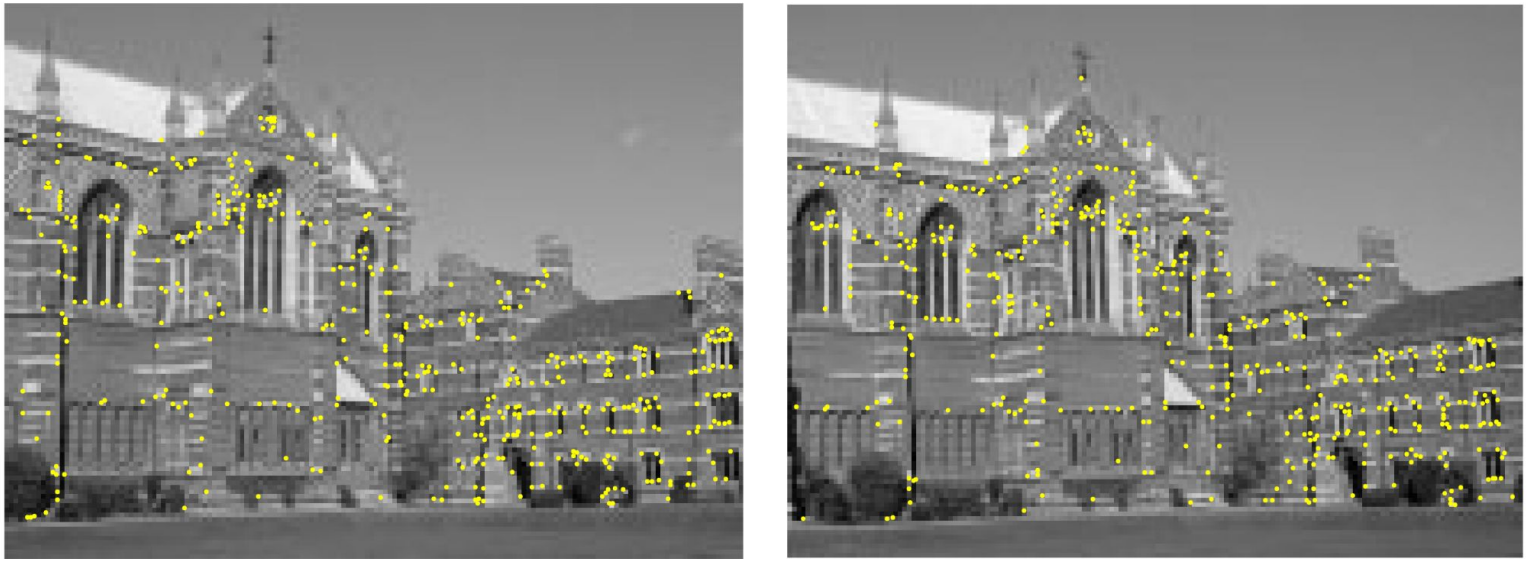
Many outliers: *common*



Brief flashback to our original problem for demonstration (with a slightly different image pair)

We want to align these two images.

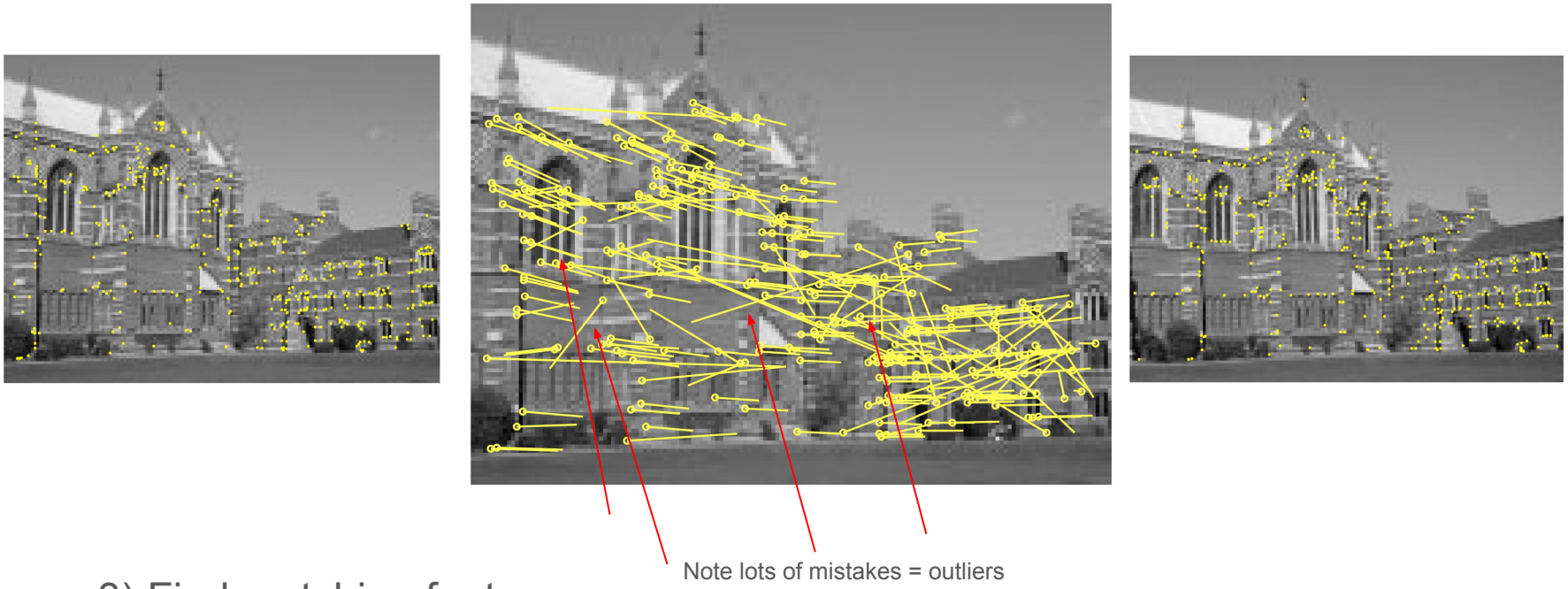
Brief flashback to our original problem for demonstration (with a slightly different image pair)



≈ 500 corner features found in each image

1) Detect corners, 2) SIFT features for each corner

Brief flashback to our original problem for demonstration (with a slightly different image pair)



3) Find matching features

Least squares - minimize big errors at all costs. Not suitable for outliers…

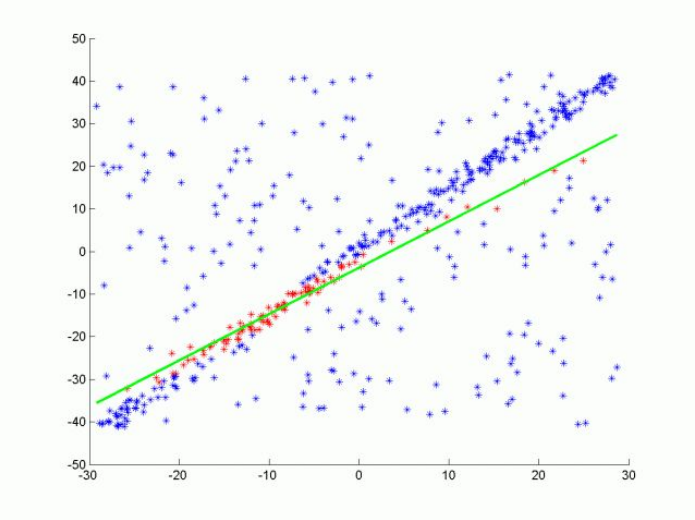
- *What we really want*: model explains **many** points **“well”**
- *Least Squares*: model makes as few big mistakes as possible over the entire dataset

- *New objective*: find model for which error is “small” for as many data points as possible
- *Method*: RANSAC (**RA**ndom **SA**mple **C**onsensus)

Fischler 1981
https://dl.acm.org/doi/abs/10.1145/358669.358692

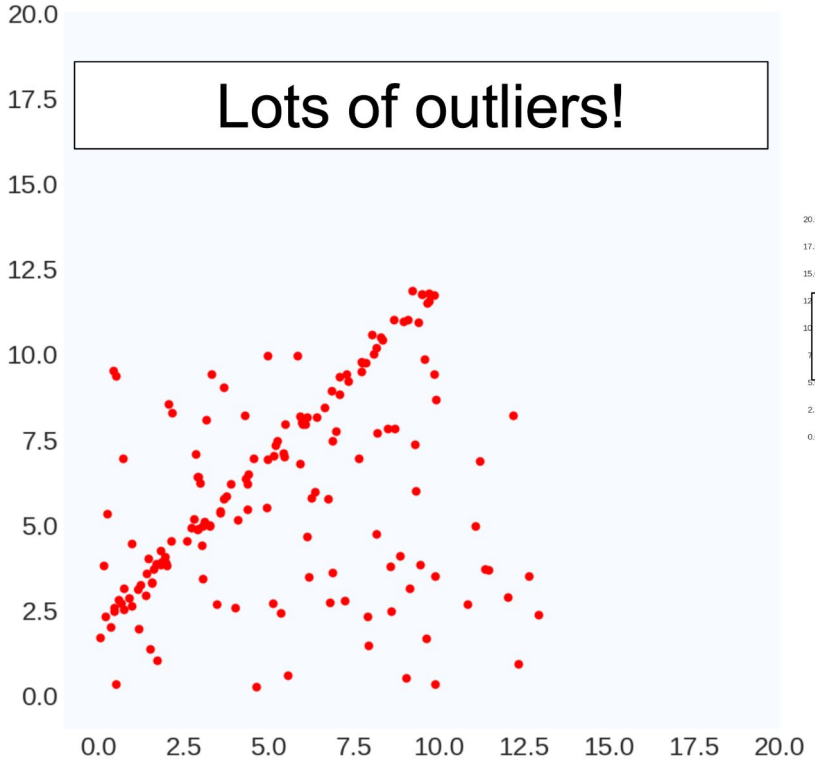
Back to 1D example - RANSAC

```
bestLine, bestCount = None, -1
for trial in range(numTrials):
        subset = pickPairOfPoints(data)
        line =       LeastSquares(subset)
        E = linePointDistance(data,line)
        inliers = E < threshold
        if #inliers > bestCount:
                bestLine, bestCount = line, #inliers
```

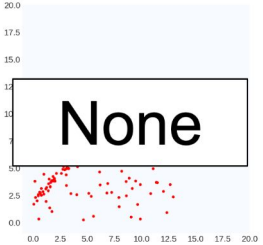


RANSAC steps

Trial #1



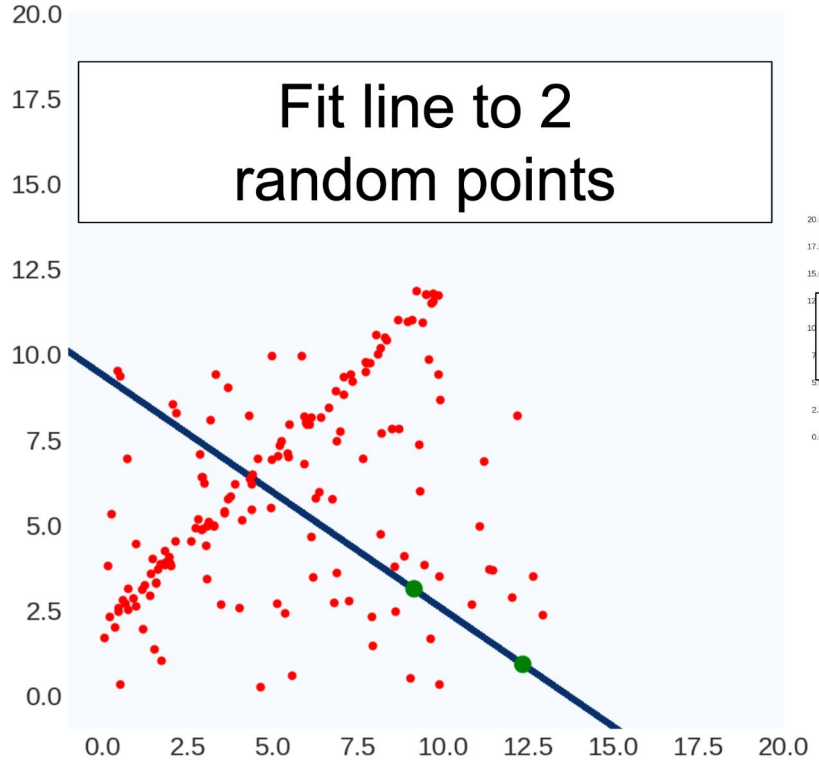
Best Model:



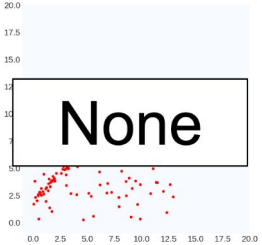
Best Count: -1

RANSAC steps

Trial #1



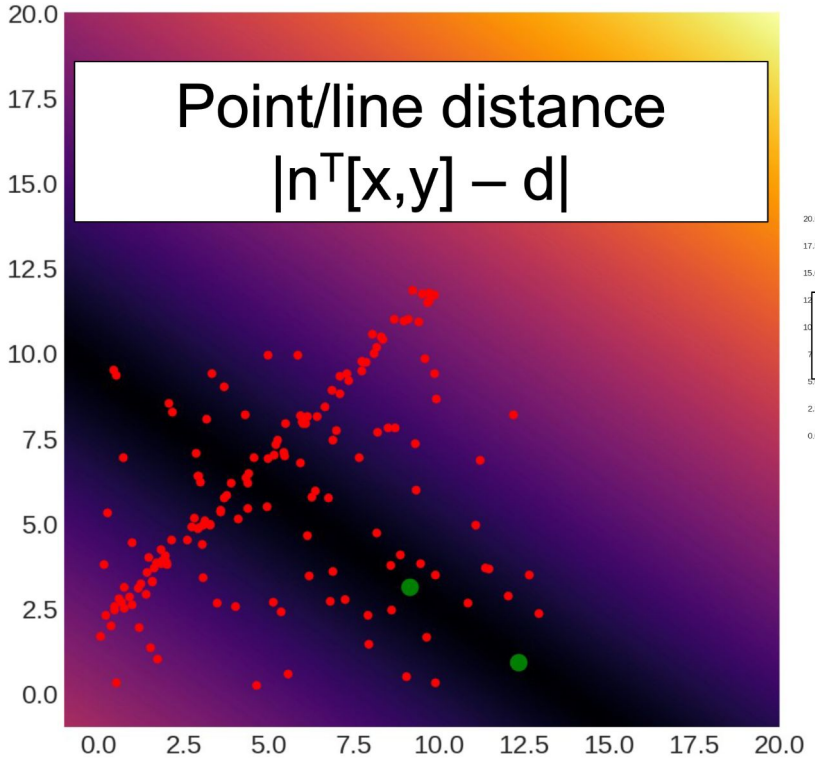
Best Model:



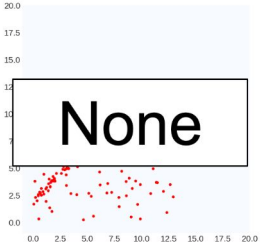
Best Count: -1

RANSAC steps

Trial #1



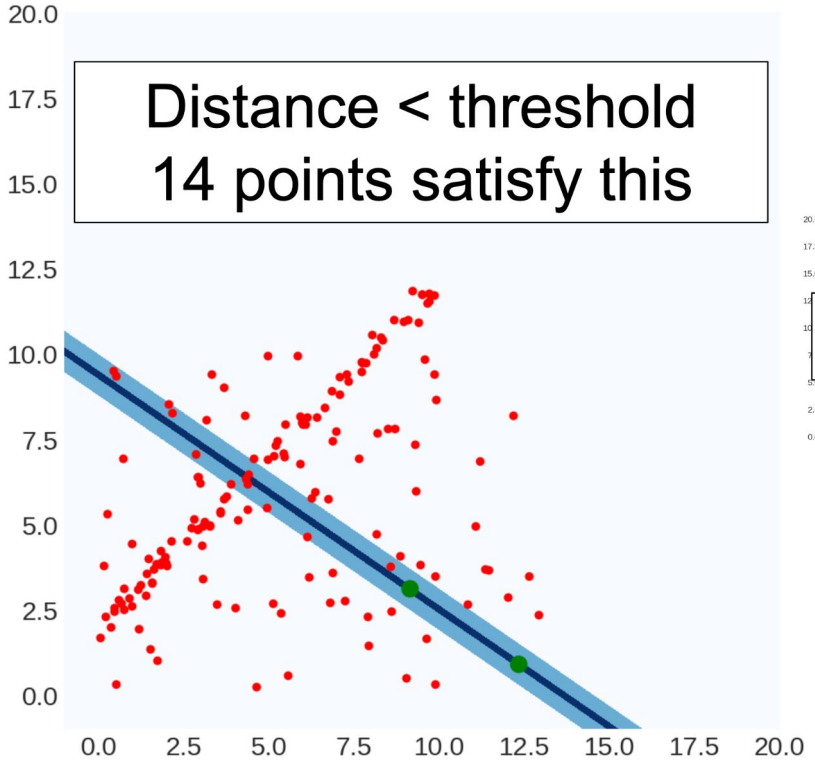
Best Model:



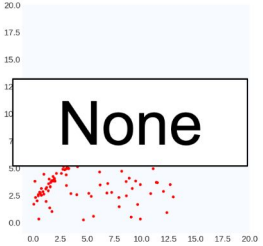
Best Count: -1

RANSAC steps

Trial #1



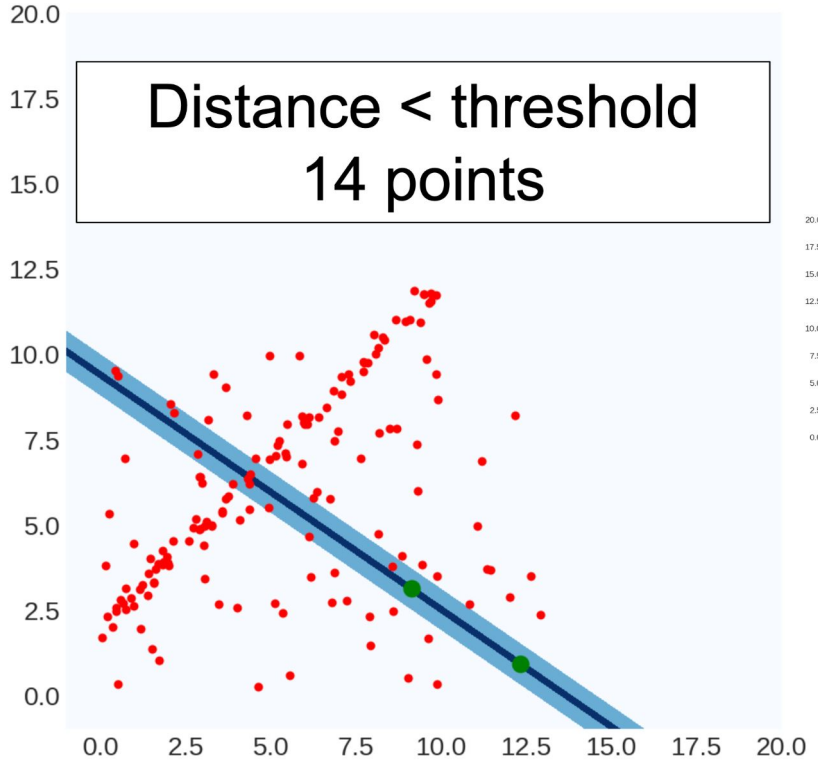
Best Model:



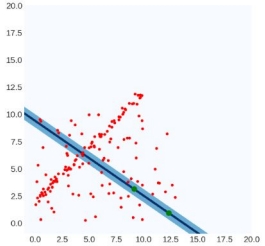
Best Count: -1

RANSAC steps

Trial #1



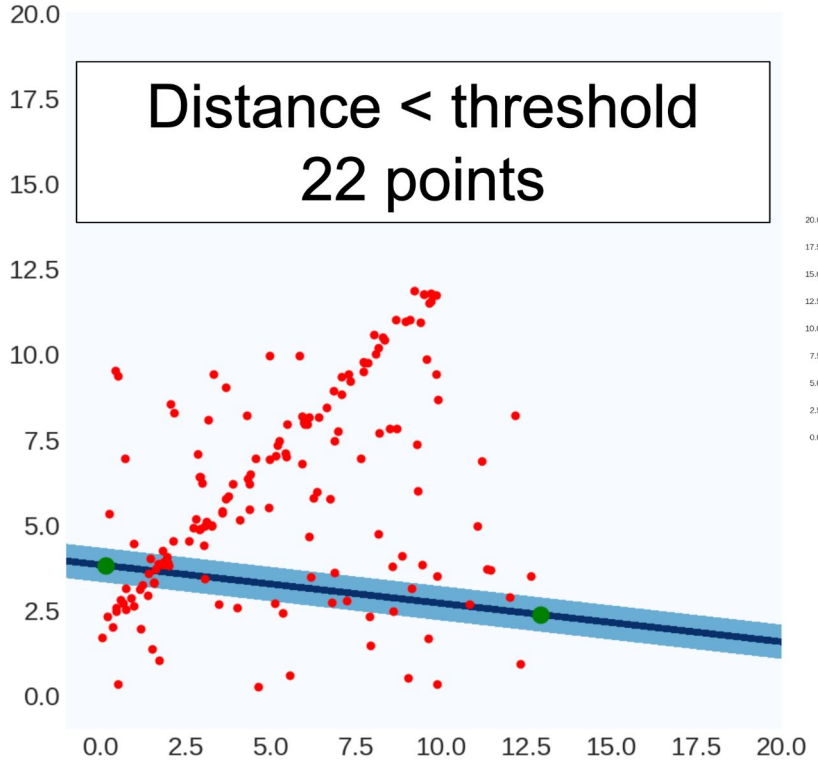
Best Model:



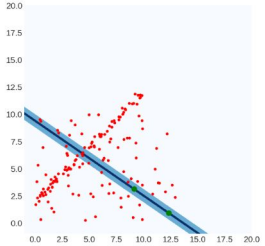
Best Count: 14

RANSAC steps

Trial #2



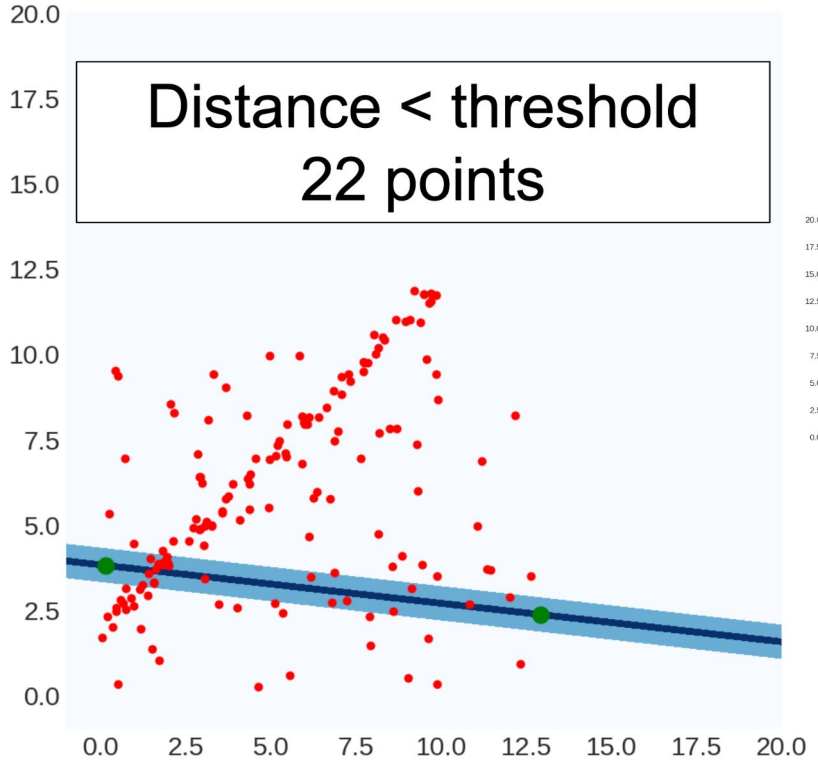
Best Model:



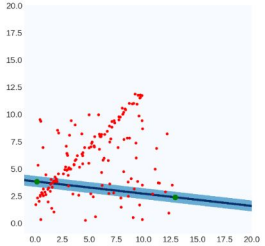
Best Count: 14

RANSAC steps

Trial #2



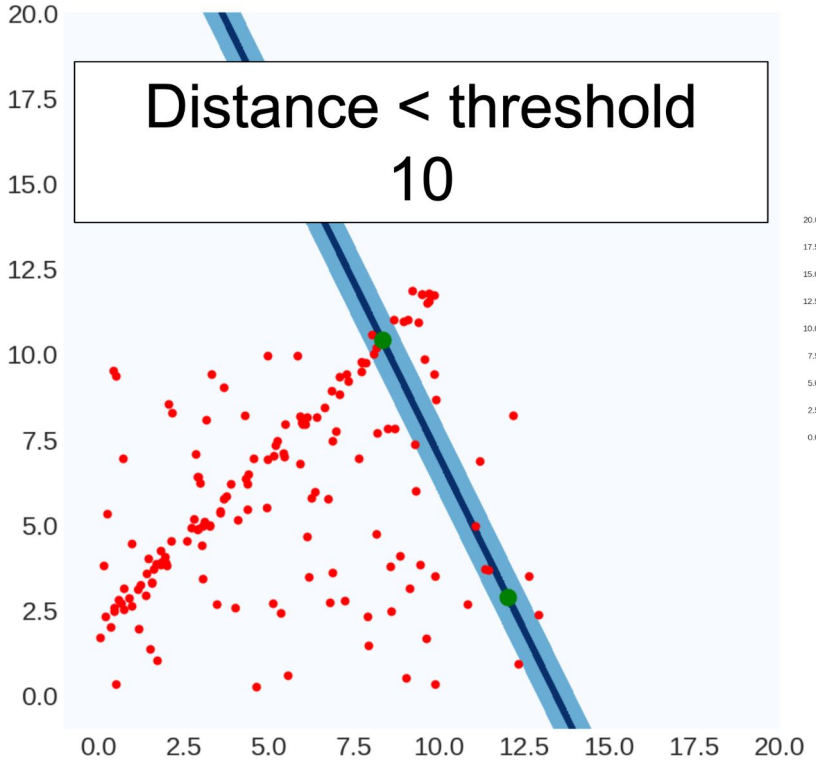
Best Model:



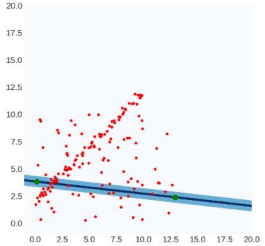
Best Count: 22

RANSAC steps

Trial #3



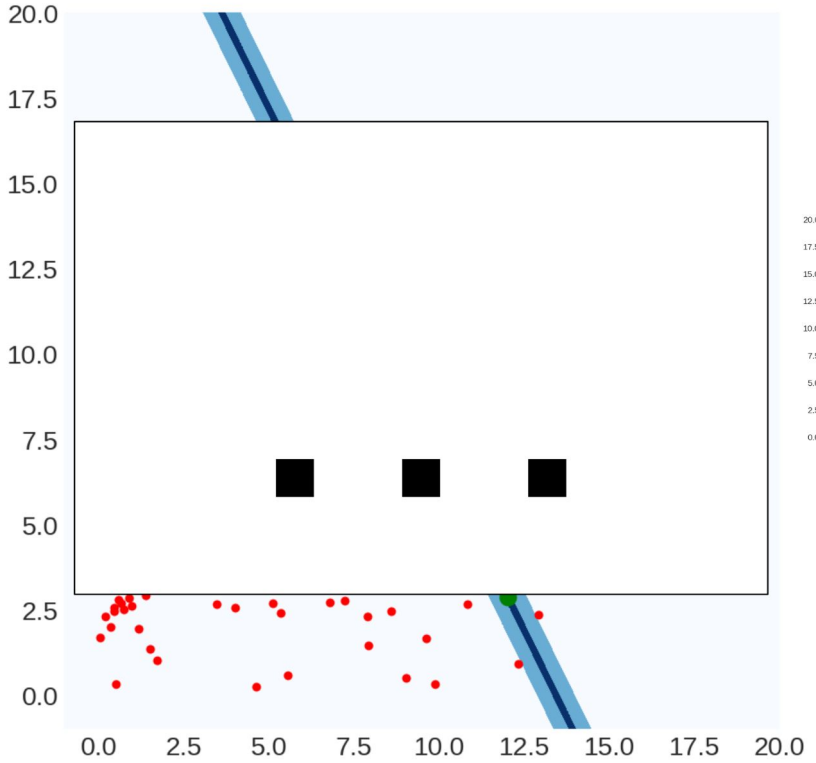
Best Model:



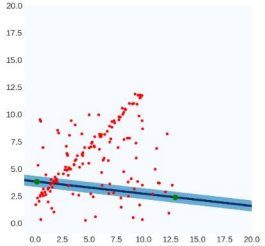
Best Count: 22

RANSAC steps

Trial #3



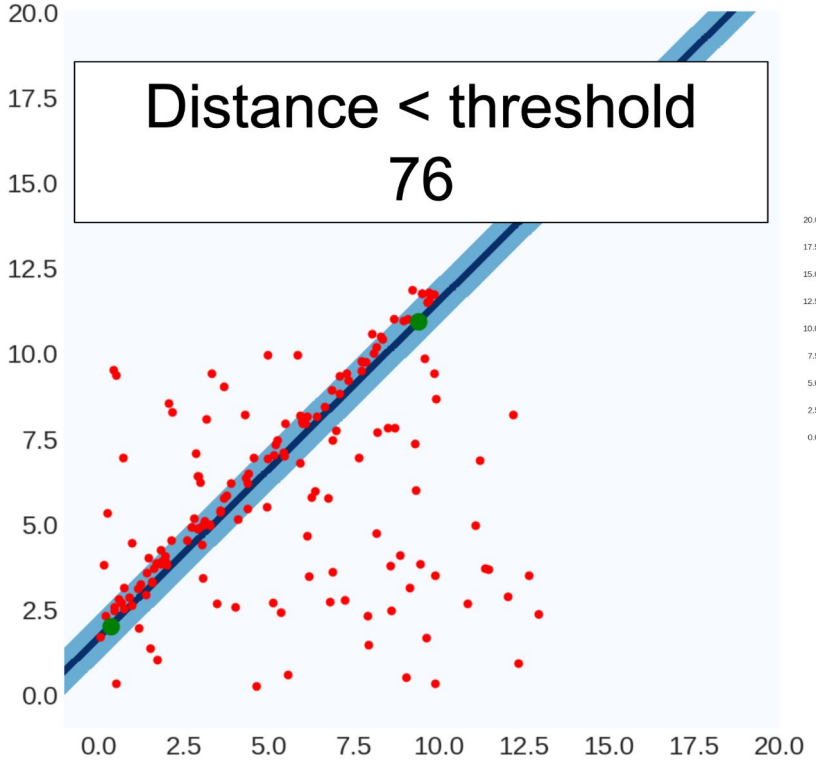
Best Model:



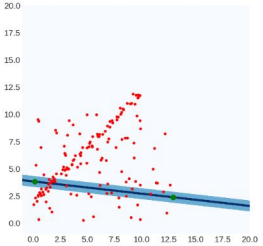
Best Count: 22

RANSAC steps

Trial #9



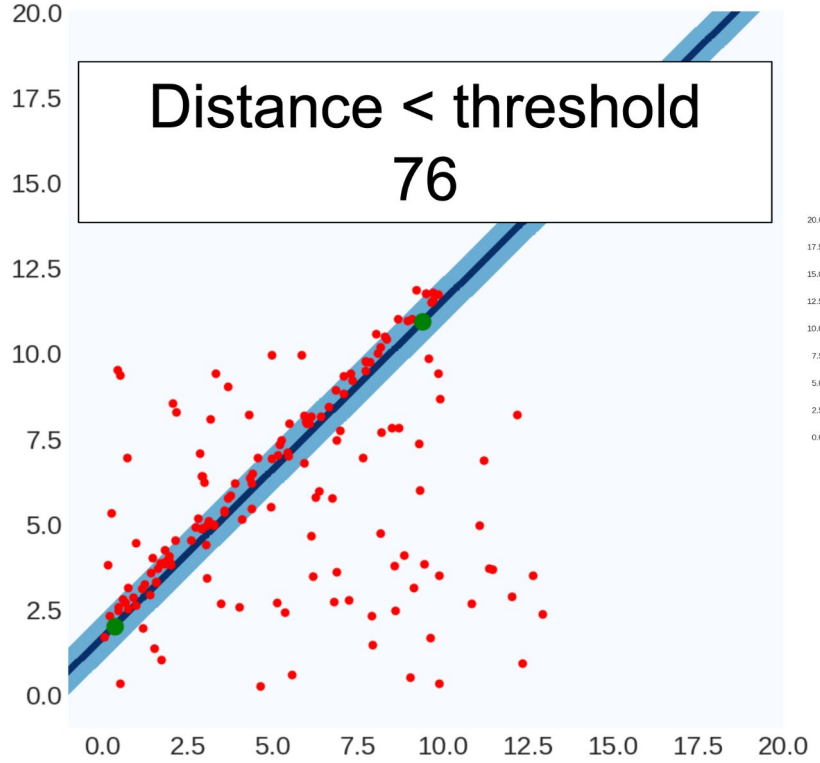
Best Model:



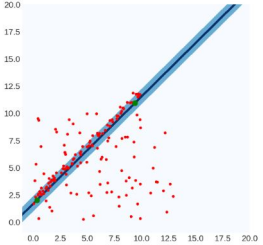
Best Count: 22

RANSAC steps

Trial #9



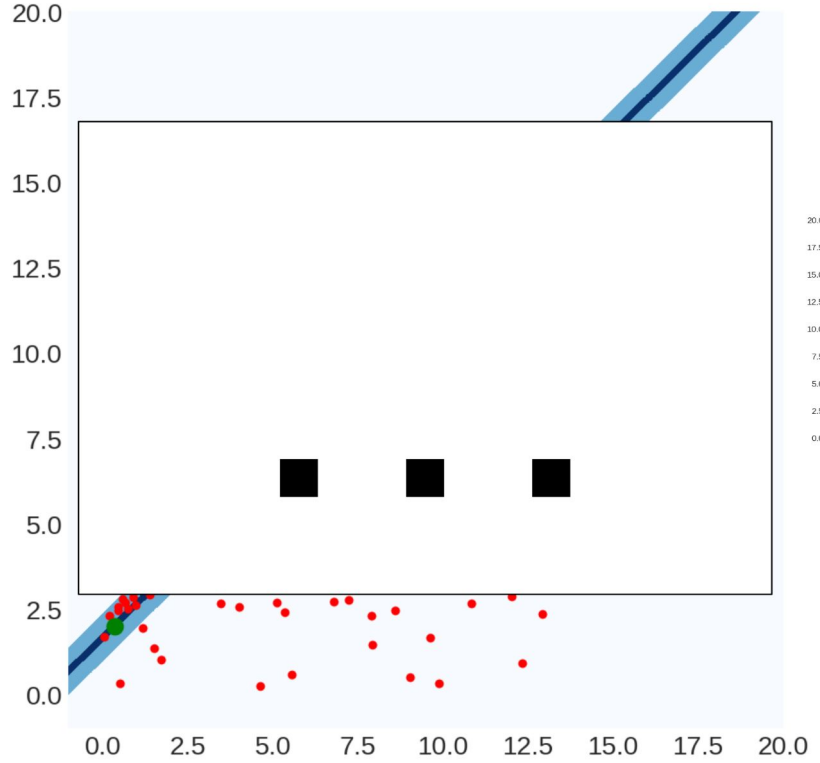
Best Model:



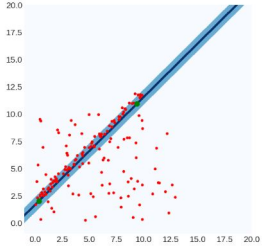
Best Count: 76

RANSAC steps

Trial #9



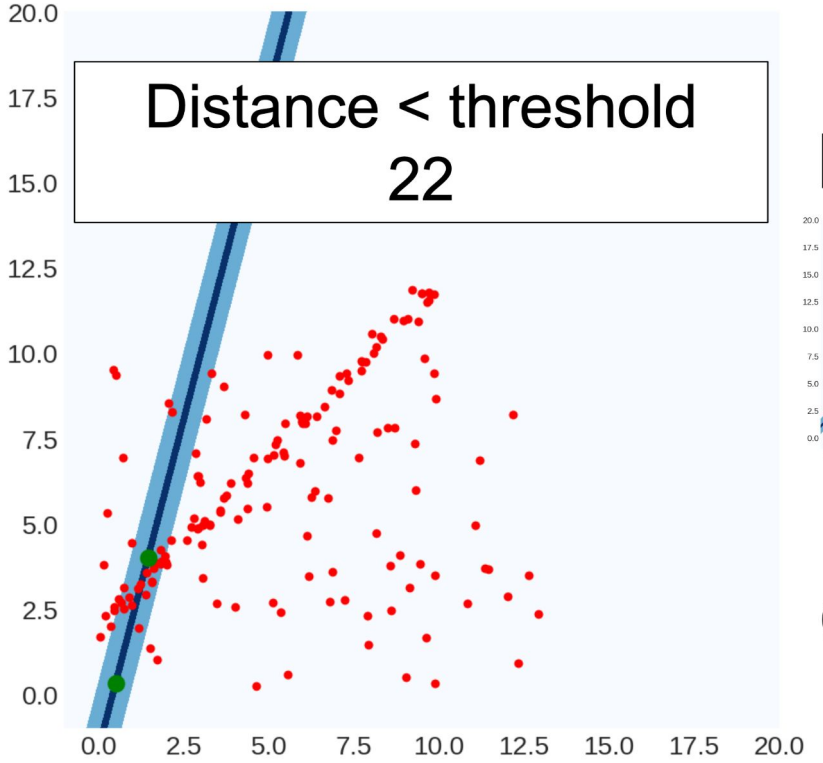
Best Model:



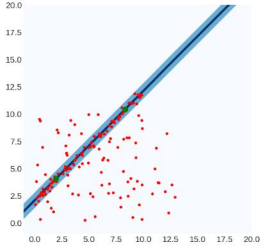
Best Count: 76

RANSAC steps

Trial #100

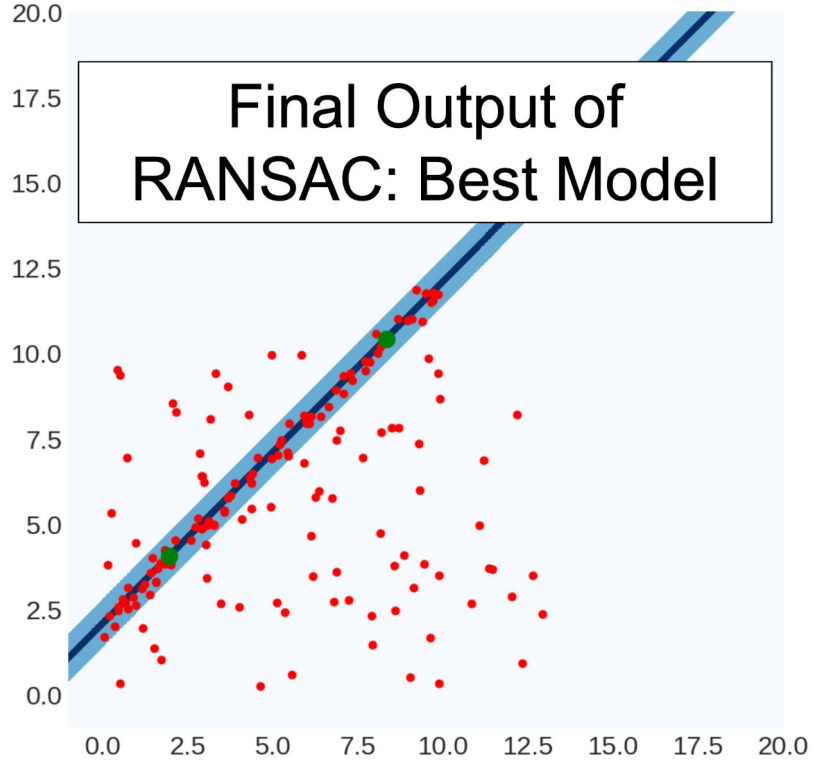


Best Model:



Best Count: 85

RANSAC steps



RANSAC and its key parameters

```
best, bestCount = None, -1
for trial in range(NUM_TRIALS):
        subset = pickSubset(data,SUBSET_SIZE)
        model = fitModel(subset)
        E = computeError(data,line)
        inliers = E < THRESHOLD
        if #(inliers) > bestCount:
                best, bestCount = model, #(inliers)
Refit on the inliers for best model (keep these!)
```

Analysis

r is the fraction of outliers (e.g., 80%)
Suppose we pick *s* points (e.g., 2)
we run RANSAC *N* times (e.g., 500)

What's the probability that **s** points we selected in each trial are NOT outliers?

Analysis

r is the fraction of outliers (e.g., 80%)

Suppose we pick *s* points (e.g., 2)

we run RANSAC *N* times (e.g., 500)

What's the probability that **s** points we selected in each trial are NOT outliers?

$(1-r)^s$

Analysis

r is the fraction of outliers (e.g., 80%)

Suppose we pick *s* points (e.g., 2)

we run RANSAC *N* times (e.g., 500)

What's the probability that **s** points we selected in each trial are NOT outliers?

$(1-r)^s$

What's the probability that any of the **s** points we selected in each trial are outliers?

Analysis

r is the fraction of outliers (e.g., 80%)

Suppose we pick *s* points (e.g., 2)

we run RANSAC *N* times (e.g., 500)

What's the probability that **s** points we selected in each trial are NOT outliers?

$(1-r)^s$

What's the probability that any of the **s** points we selected in each trial are outliers?

Analysis

r is the fraction of outliers (e.g., 80%)

Suppose we pick *s* points (e.g., 2)

we run RANSAC *N* times (e.g., 500)

What's the probability that **s** points we selected in each trial are NOT outliers?

$(1-r)^s$

What's the probability that any of the **s** points we selected in each trial are outliers?

$1-(1-r)^s$

Analysis

Suppose
r is the fraction of outliers (e.g., 80%)
we pick *s* points (e.g., 2)
we run RANSAC *N* times (e.g., 500)

What's the probability that **s** points we selected in each trial are NOT outliers?

$(1-r)^s$

What's the probability that any of the **s** points we selected in each trial are outliers?

$1-(1-r)^s$

What's the probability that we select outliers in all N trials?

Analysis

Suppose
r is the fraction of outliers (e.g., 80%)
we pick *s* points (e.g., 2)
we run RANSAC *N* times (e.g., 500)

What's the probability that **s** points we selected in each trial are NOT outliers?

$(1-r)^s$

What's the probability that any of the **s** points we selected in each trial are outliers?
$1-(1-r)^s$

What's the probability that we select outliers in all N trials?

$(1-(1-r)^s)^N$

Analysis

Suppose
r is the fraction of outliers (e.g., 80%)
we pick *s* points (e.g., 2)
we run RANSAC *N* times (e.g., 500)

What's the probability that **s** points we selected in each trial are NOT outliers?

$(1-r)^s$

What's the probability that any of the **s** points we selected in each trial are outliers?

$1-(1-r)^s$

What's the probability that we select outliers in all N trials?

$(1-(1-r)^s)^N$

What's the probability that we pick any set with all inliers?

$1-(1-(1-r)^s)^N$

Analysis

Suppose
r is the fraction of outliers (e.g., 80%)
we pick *s* points (e.g., 2)
we run RANSAC *N* times (e.g., 500)

What's the probability that **s** points we selected in each trial are NOT outliers?

$(1-r)^s$ 4%

What's the probability that any of the **s** points we selected in each trial are outliers?

$1-(1-r)^s$ 96%

What's the probability that we select outliers in all N trials?

$(1-(1-r)^s)^N$ For N=50, 13% — for N=500, 10^{-7}%

What's the probability that we pick any set with all inliers?

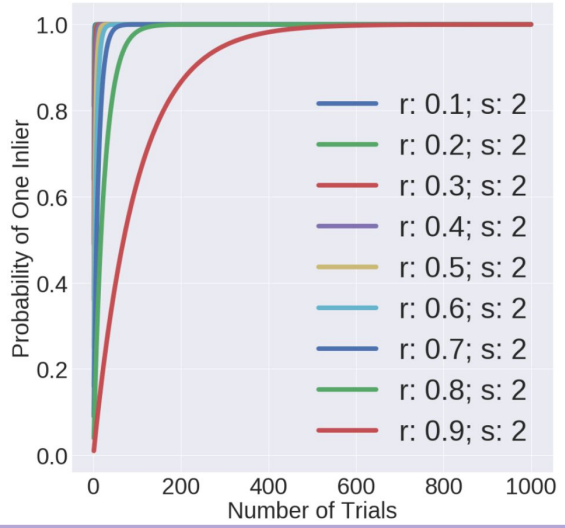
$1-(1-(1-r)^s)^N$ For N=50, 86% — for N=500, ~99.99999%

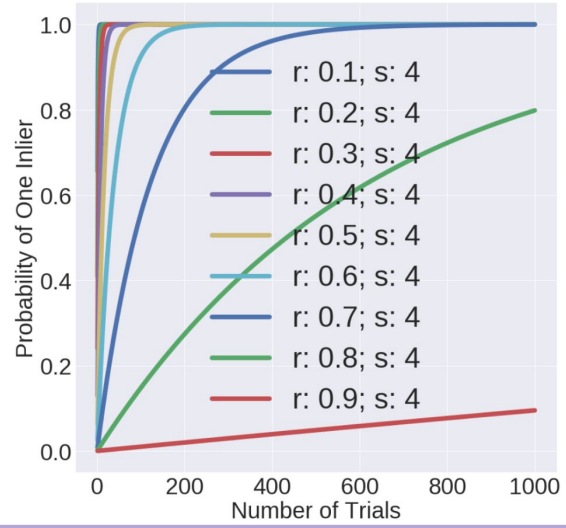
Need more trials if more points are used for fitting or the number of outliers are greater

r is the fraction of outliers (e.g., 80%)

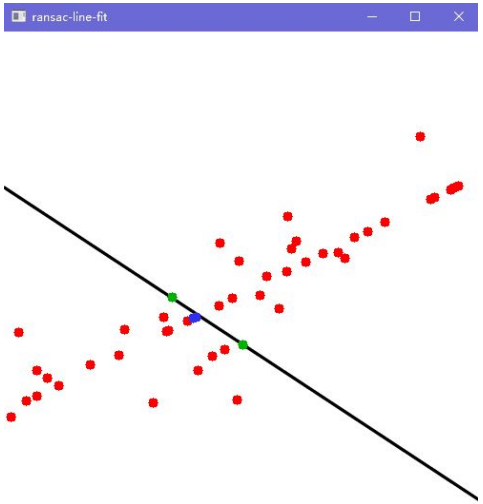
Suppose we pick *s* points (e.g., 2)

we run RANSAC *N* times (e.g., 500)





RANSAC subset size - analysis

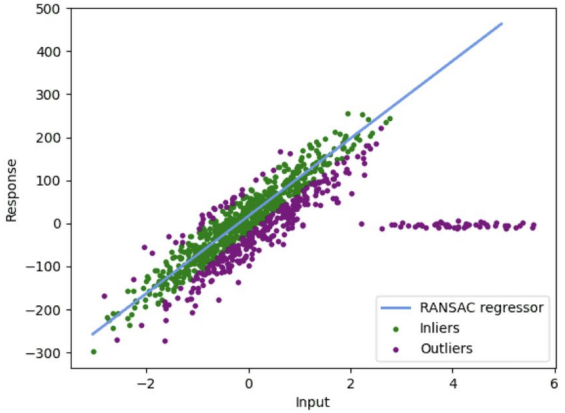


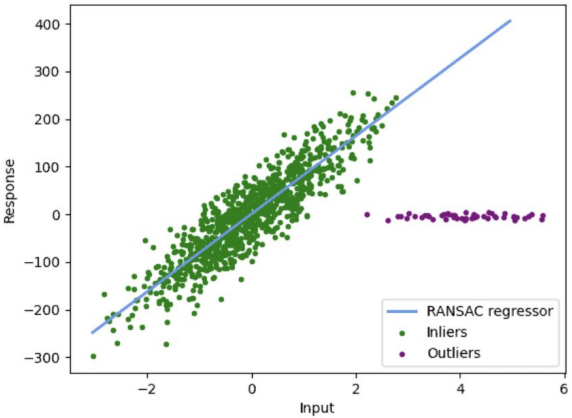
- Always the smallest possible set for fitting the model.
- Minimum number for lines: 2 data points
- Minimum number for 3D planes: **how many?**

RANSAC threshold - analysis

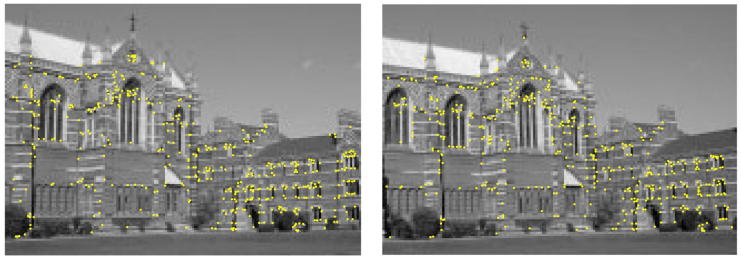
Threshold should reflect the inherent noise within the inliers.

Usually unknown but can be tuned in training.





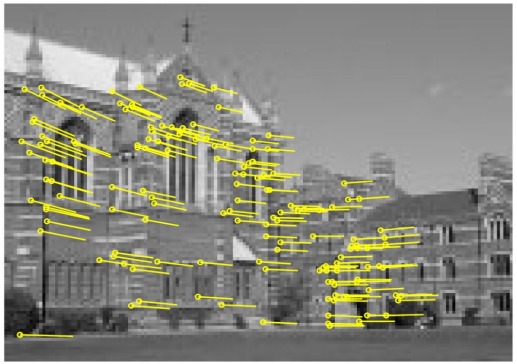
Fitting an example model with RANSAC



≈ 500 corner features found in each image

268 matched corners

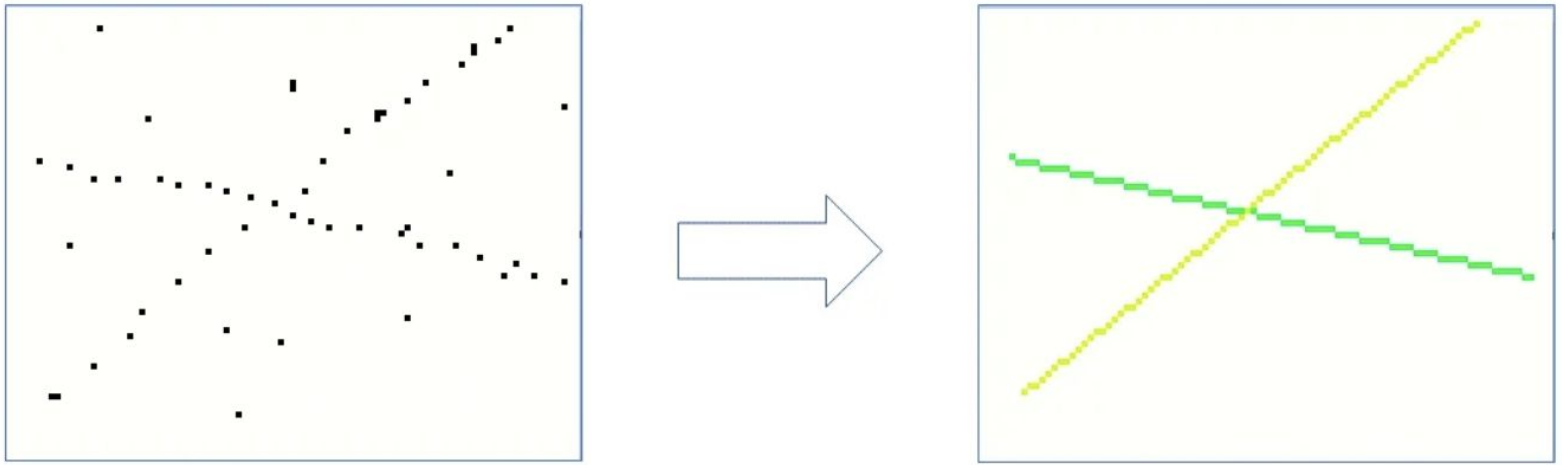
RANSAC
with s=4



151 **inliers**

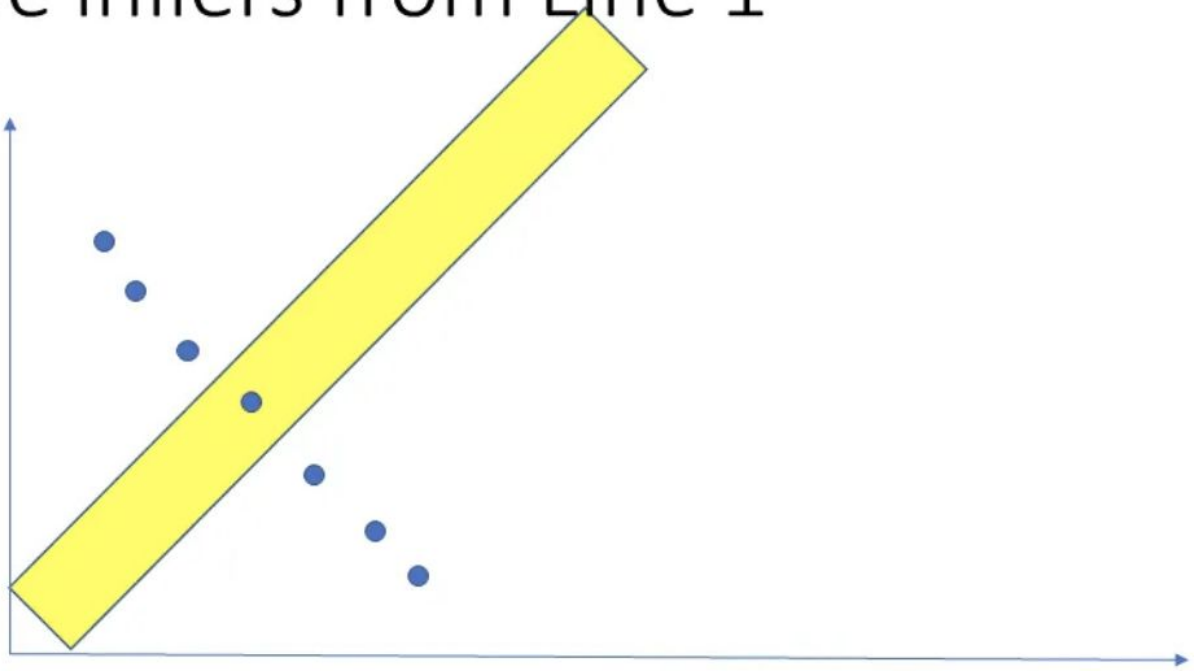
117 **outliers**

RANSAC applications - line fitting

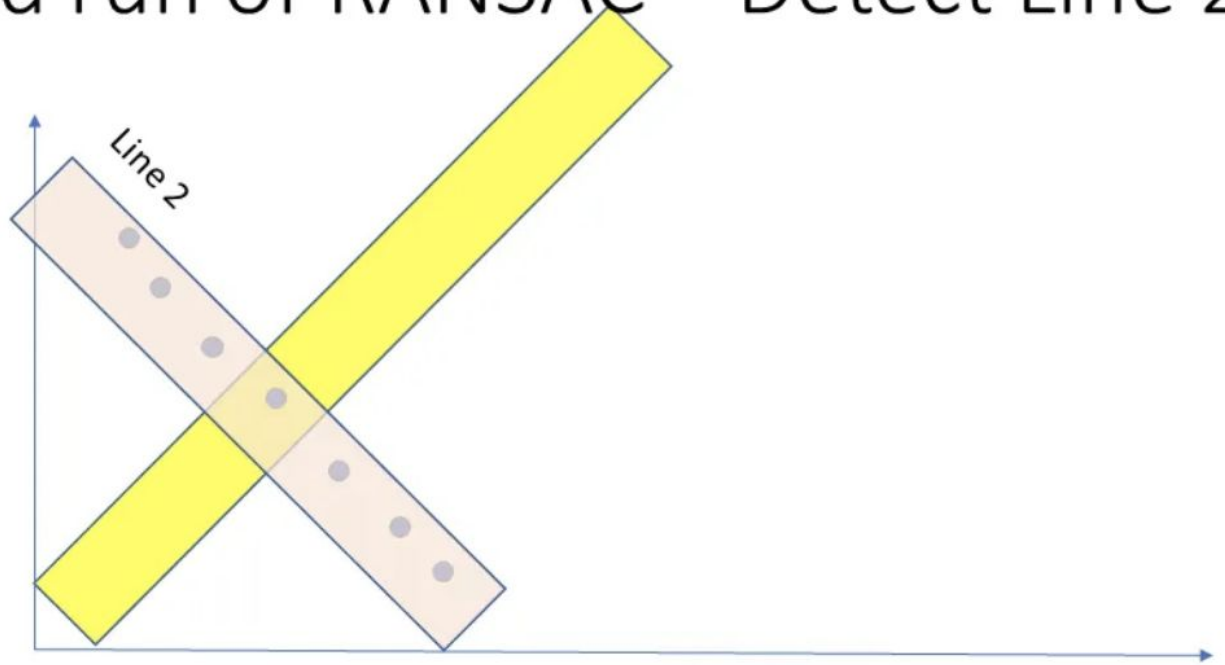


Given noisy image with corner or edge detections, can we detect lines?

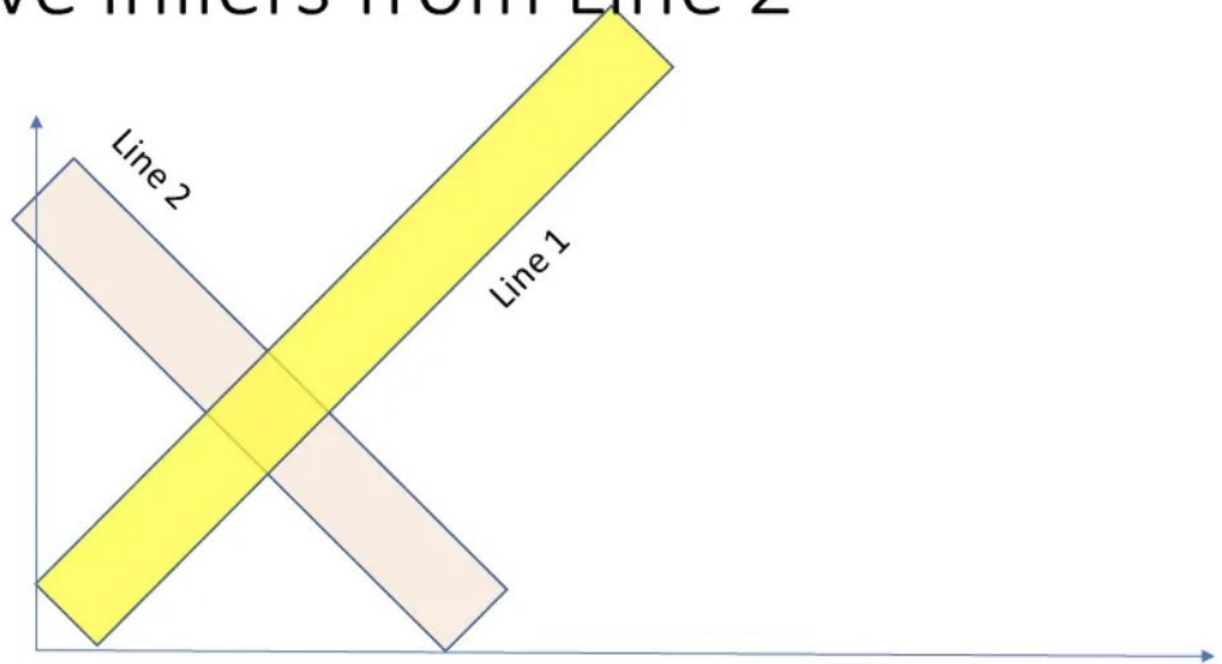
Remove inliers from Line 1



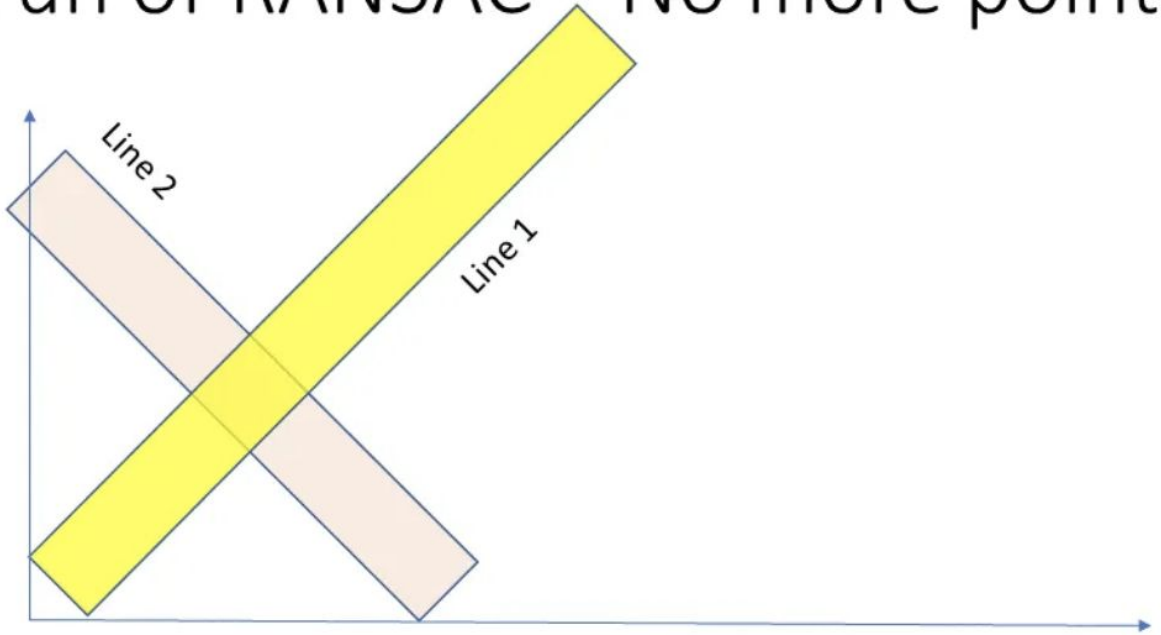
Second run of RANSAC – Detect Line 2



Remove inliers from Line 2



Third run of RANSAC – No more points left



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

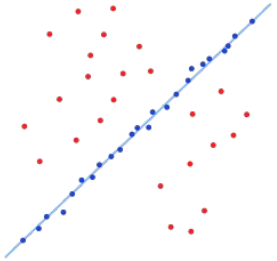
— Many real problems have high rate of outliers (but sometimes selective choice of random subsets can help)

The Hough transform can handle high percentage of outliers

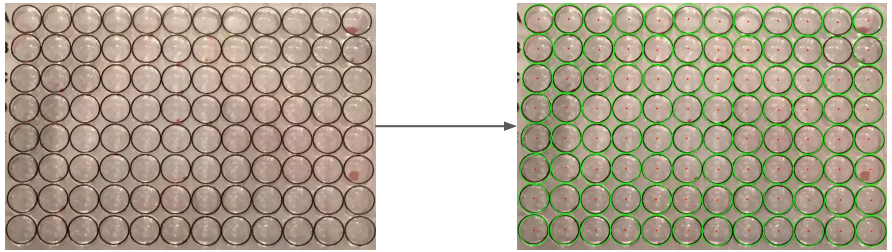
Suppose we want to fit a model to a set of tokens

— e.g. A line fits well to a set of points. This is unlikely to be due to chance, so we represent the points as a line.

— e.g. A 3D model can be scaled, rotated and translated to closely fit a set of points or line segments. If it fits well, the object is recognized.



Line finding



Circle finding

Challenges

Extraneous data: clutter or multiple models

— We do not know what is part of the model

— Can we fit models with a few parts when there is significant background clutter?

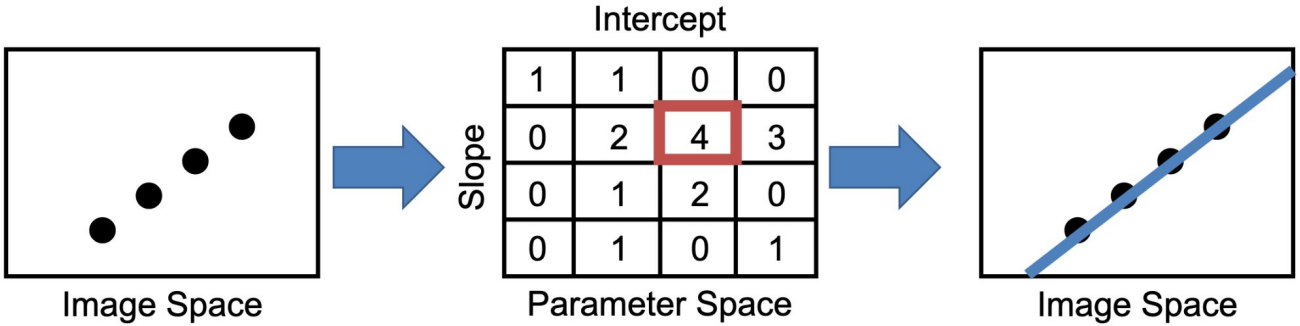
Missing data: only some parts of model are present Noise

Computational cost:

— Not feasible to check all combinations of features by fitting a model to each possible subset

Hough Transform

1. Discretize space of parametric models
2. Each pixel votes for all compatible models
3. Find models compatible with many pixels



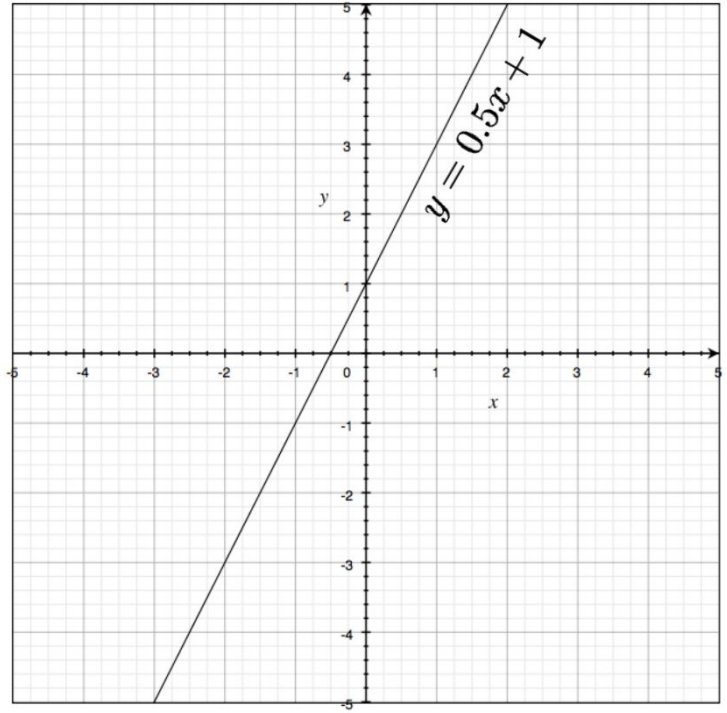
Example: For each point, vote for all lines that could pass through it; the true lines will pass through many points and so receive many votes

Duda & Hart 1972 https://dl.acm.org/doi/10.1145/361237.361242

Lines: Slope intercept form

$$y = mx + b$$

slope y-intercept



Hough Transform: Image and Parameter Space

variables

$$y = mx + b$$

parameters

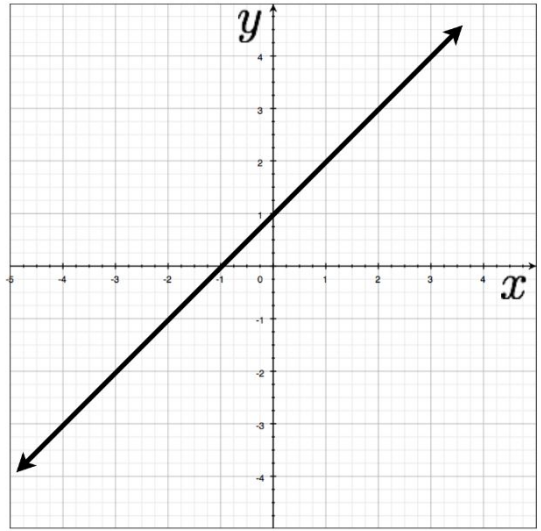


Image space

Hough Transform: Image and Parameter Space

variables

$$y = mx + b$$

parameters

variables

$$y - mx = b$$

parameters

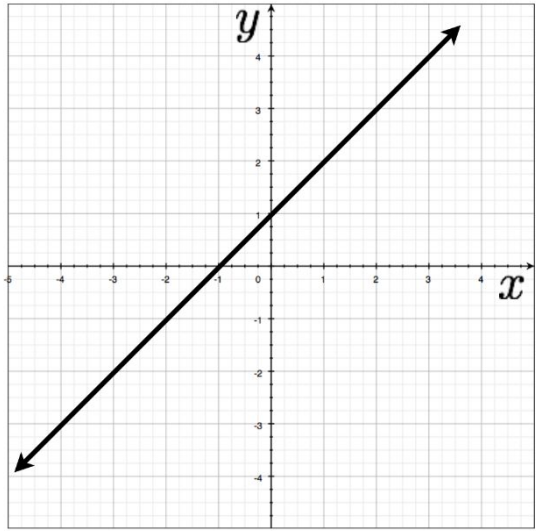
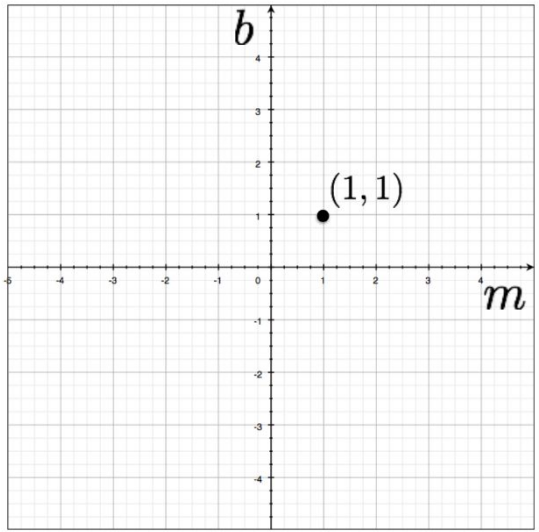


Image space

a line becomes a point



Parameter space

Hough Transform: Image and Parameter Space

variables

$$y = mx + b$$

parameters

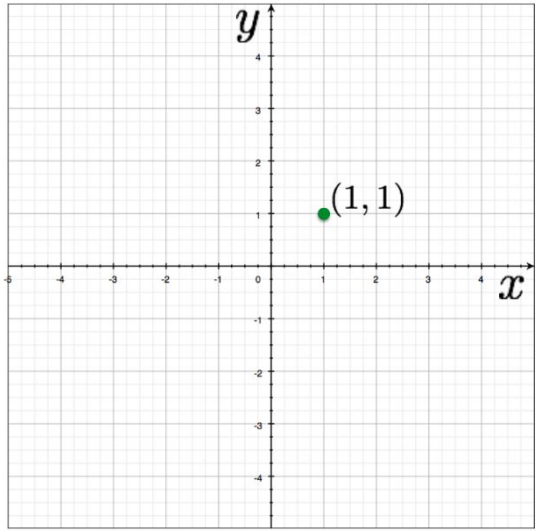


Image space

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

Hough Transform: Lines

variables

$$y = mx + b$$

parameters

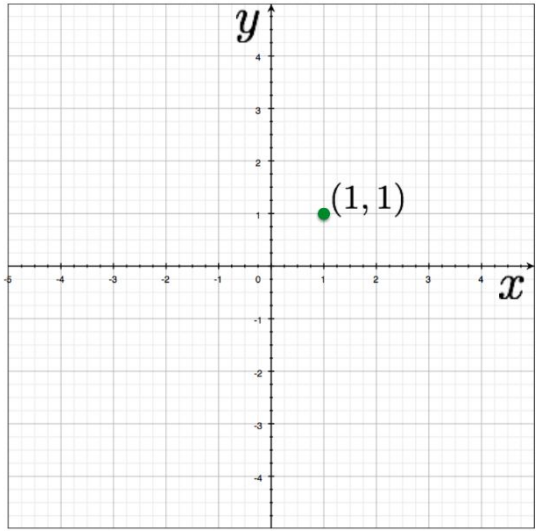


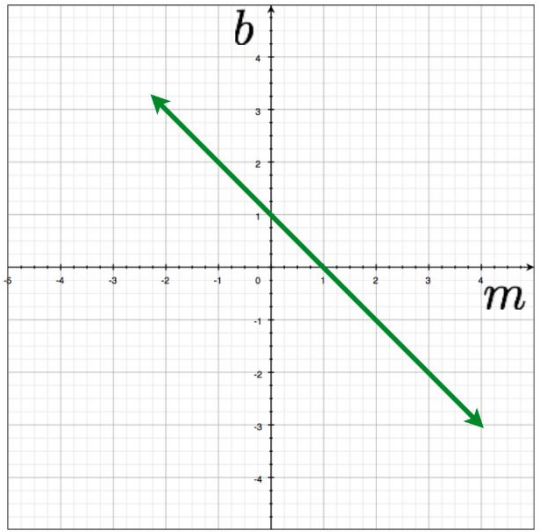
Image space

a point becomes a line

variables

$$y - mx = b$$

parameters



Parameter space

Hough Transform: Lines

variables

$$y = mx + b$$

parameters

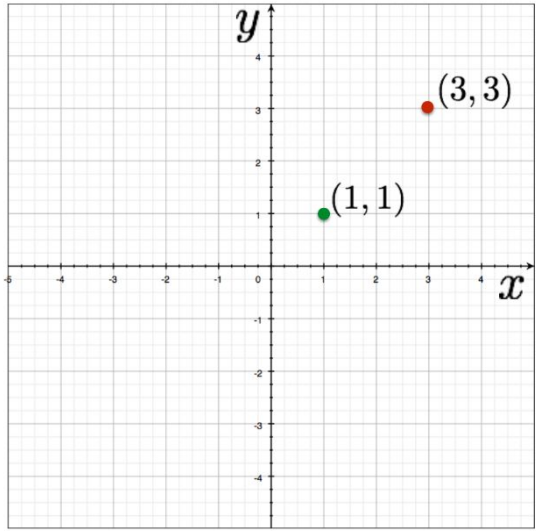


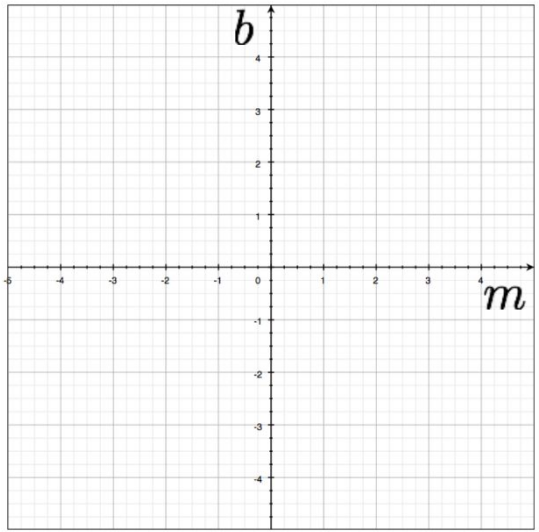
Image space

two points?

variables

$$y - mx = b$$

parameters



Parameter space

Hough Transform: Lines

variables

$$y = mx + b$$

parameters

variables

$$y - mx = b$$

parameters

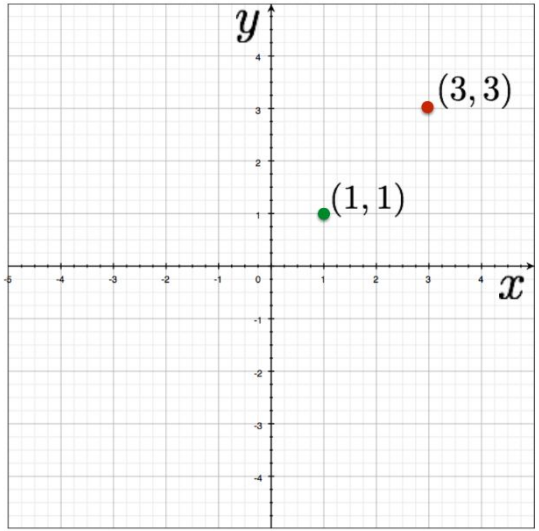
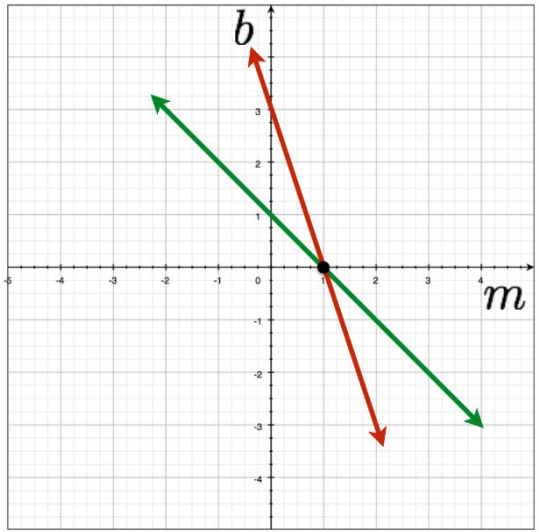


Image space

two points?



Parameter space

Hough Transform: Lines

variables

$$y = mx + b$$

parameters

variables

$$y - mx = b$$

parameters

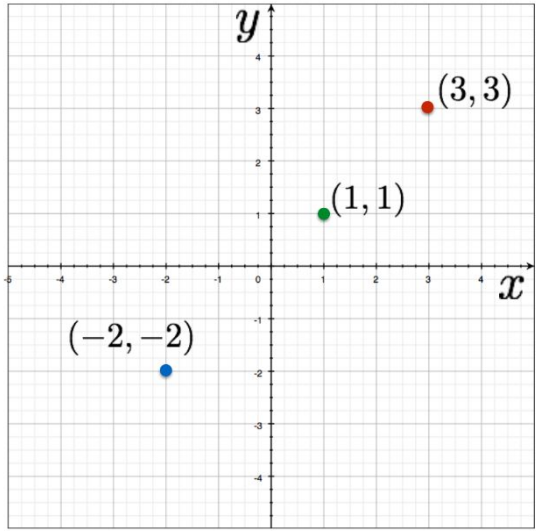
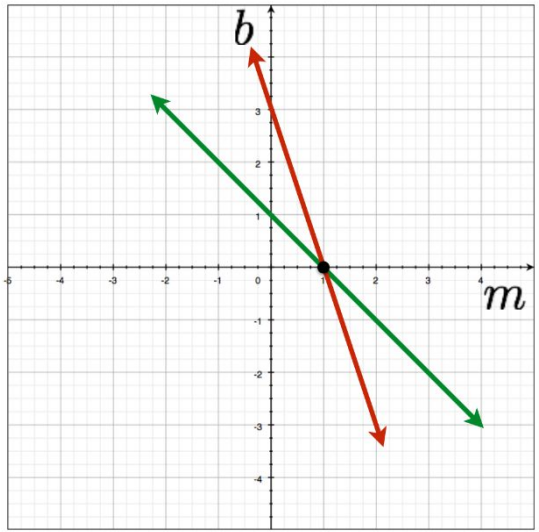


Image space

three points?



Parameter space

Hough Transform: Lines

variables

$$y = mx + b$$

parameters

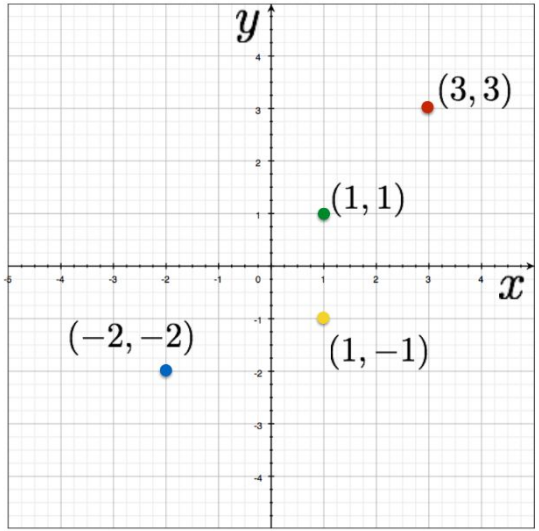


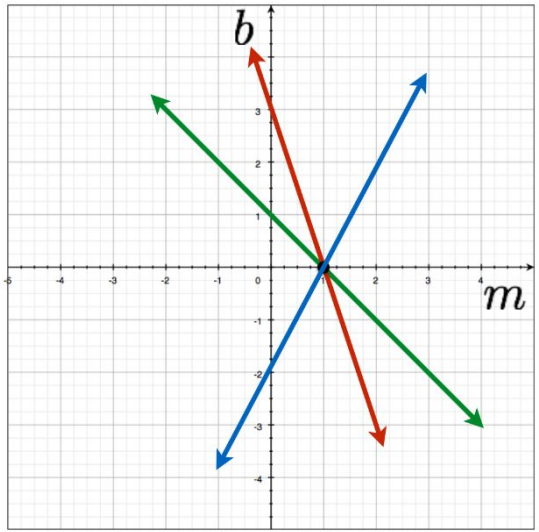
Image space

four points?

variables

$$y - mx = b$$

parameters



Parameter space

Hough Transform: Lines

variables

$$y = mx + b$$

parameters

variables

$$y - mx = b$$

parameters

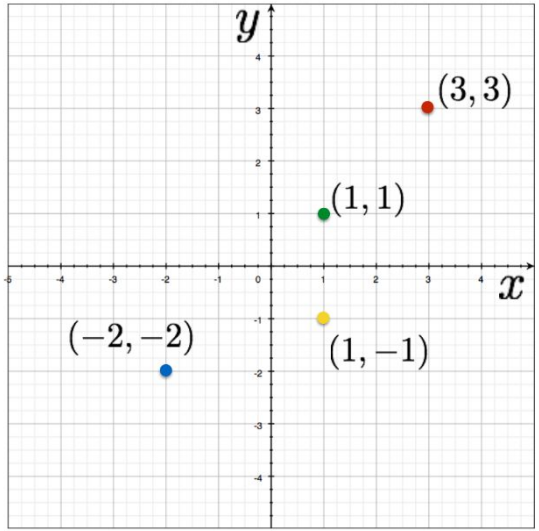
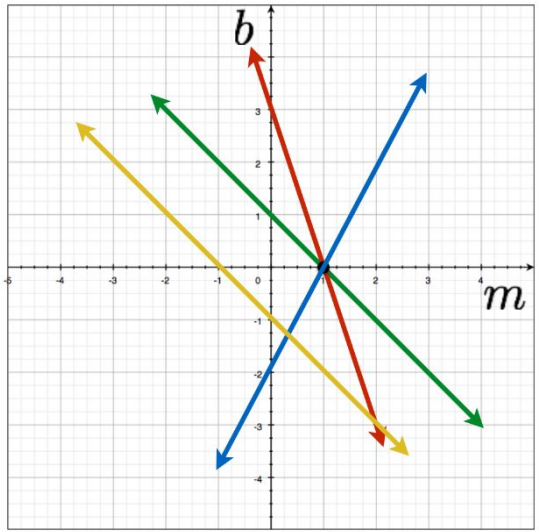


Image space

four points?



Parameter space

Hough Transform: Lines

How would you find the best fitting line?

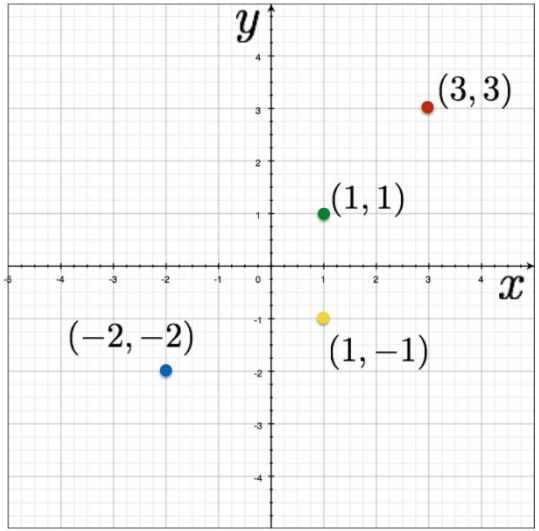
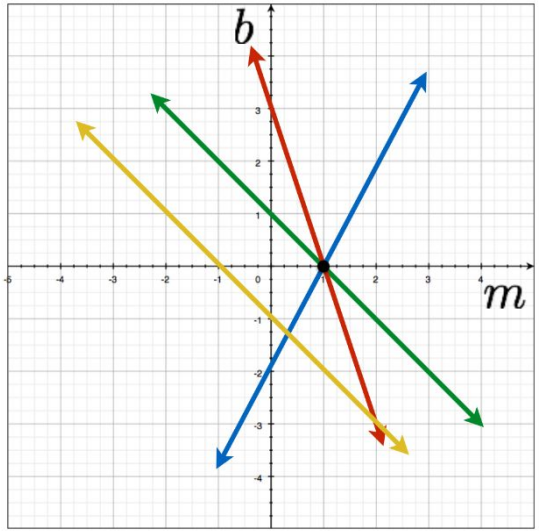


Image space



Parameter space

Hough Transform: Lines

Is this method robust to measurement noise? clutter?

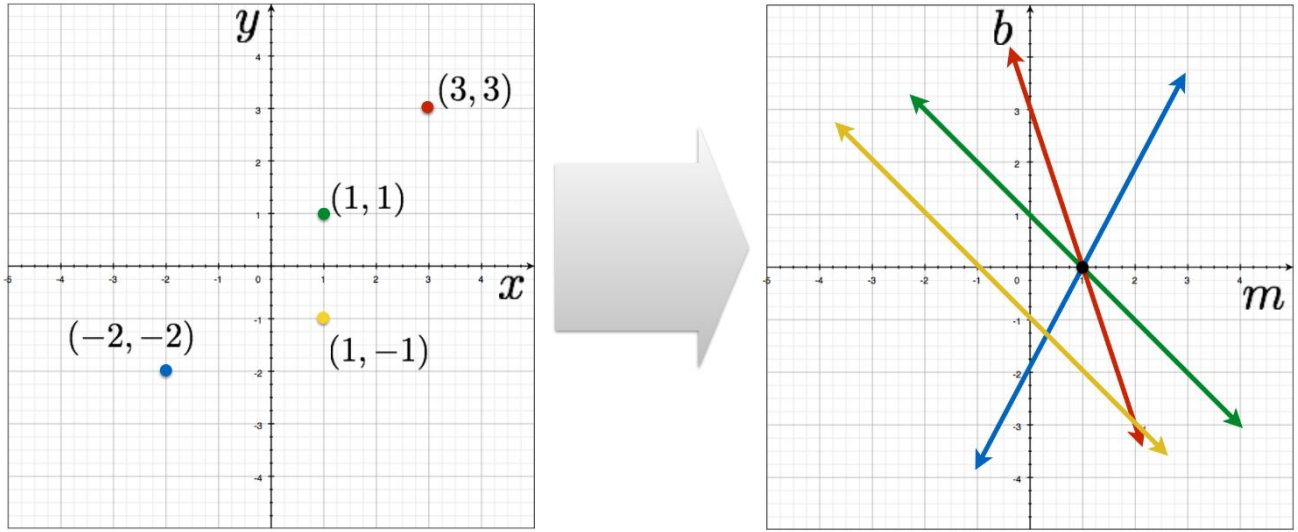


Image space

Parameter space

Line Detection by Hough Transform

Algorithm:

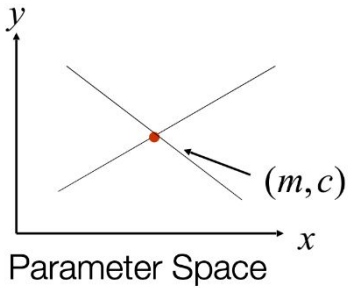
1. Quantize Parameter Space (m,c)

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 $A(m,c)$



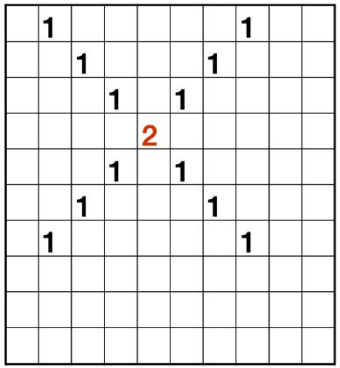
$A(m,c)$

	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)?

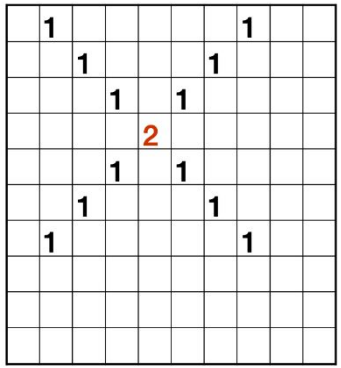
$A(m, c)$



Problems with **Parametrization**

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

$A(m, c)$



The space of m is huge!

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

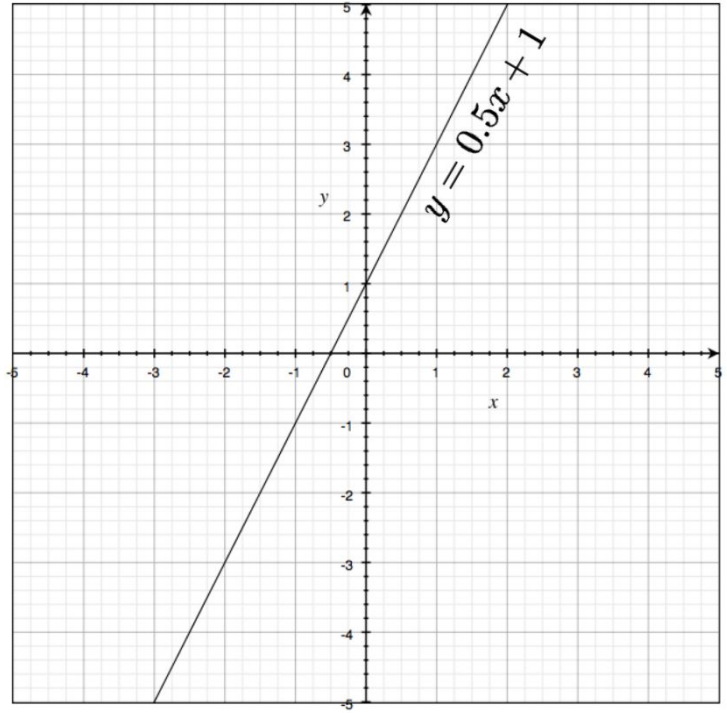
The space of c is huge!

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

Lines: Slope intercept form

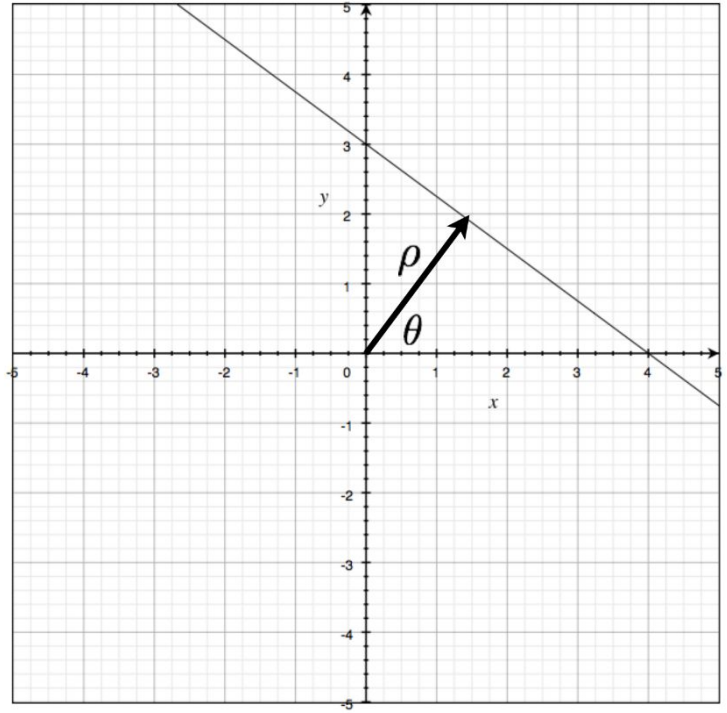
$$y = mx + b$$

slope y-intercept



Lines: Normal form

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



Book's convention

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

$$r \geq 0$$

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

Hough Transform: Lines

variables

$$y = mx + b$$

parameters

parameters

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

variables

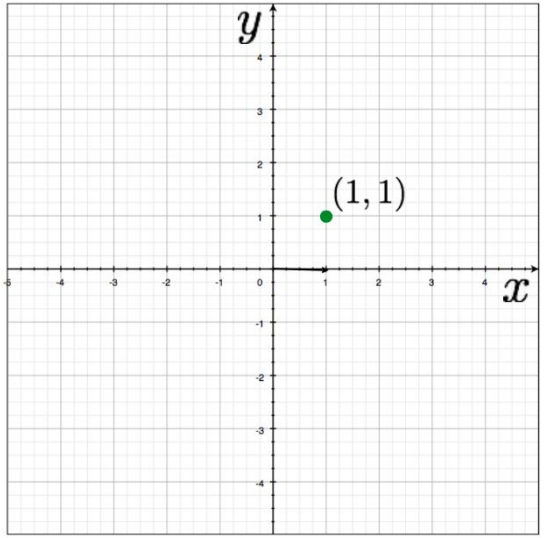
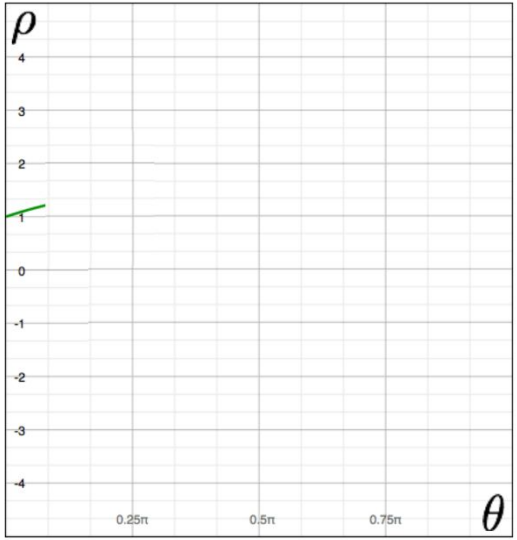


Image space

a point becomes?



Parameter space

Hough Transform: Lines

variables

$$y = mx + b$$

parameters

parameters

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

variables

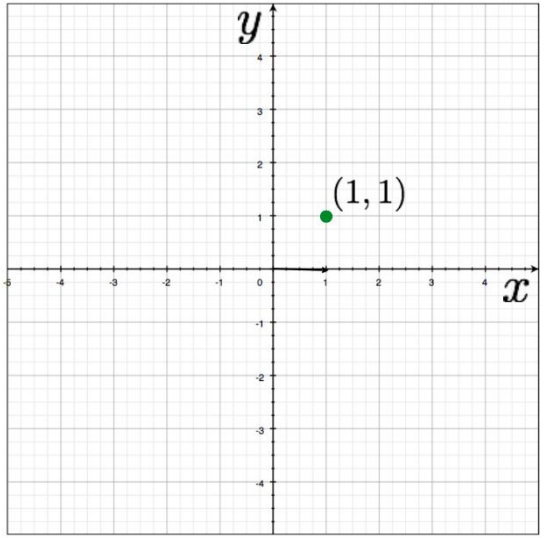
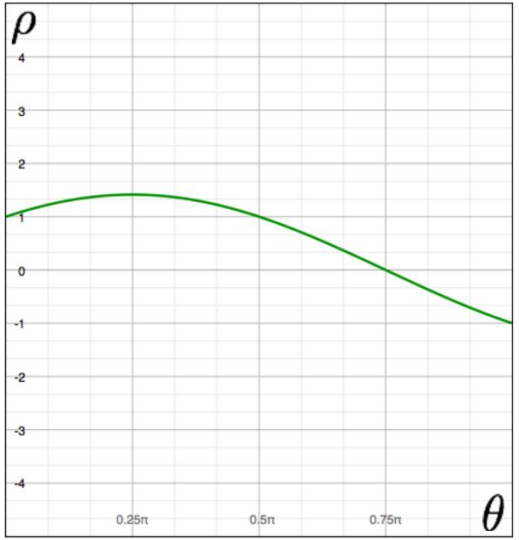


Image space

a point becomes a wave



Parameter space

Hough Transform: Lines

variables

$$y = mx + b$$

parameters

parameters

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

variables

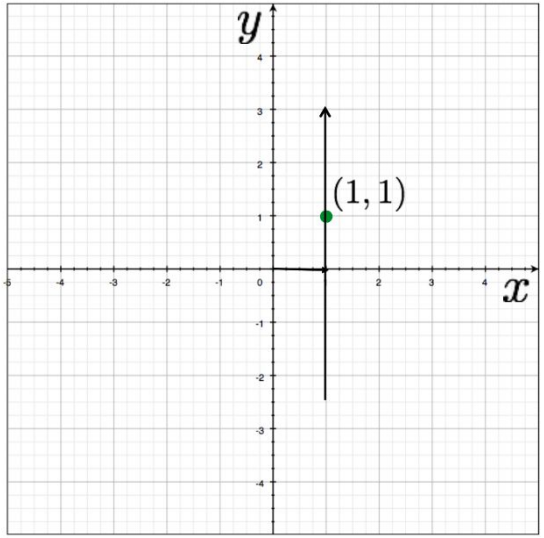
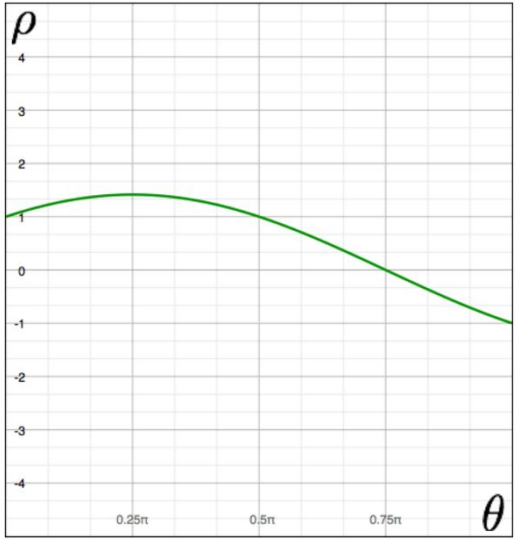


Image space

a line becomes?



Parameter space

Hough Transform: Lines

variables

$$y = mx + b$$

parameters

parameters

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

variables

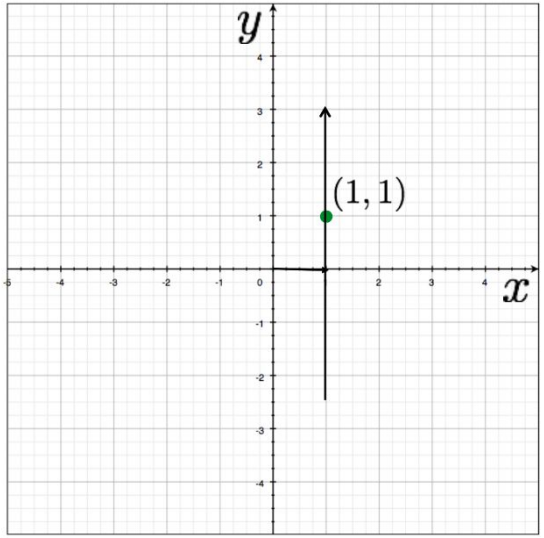


Image space

a line becomes a point

Parameter space

Hough Transform: Lines

variables

$$y = mx + b$$

parameters

parameters

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

variables

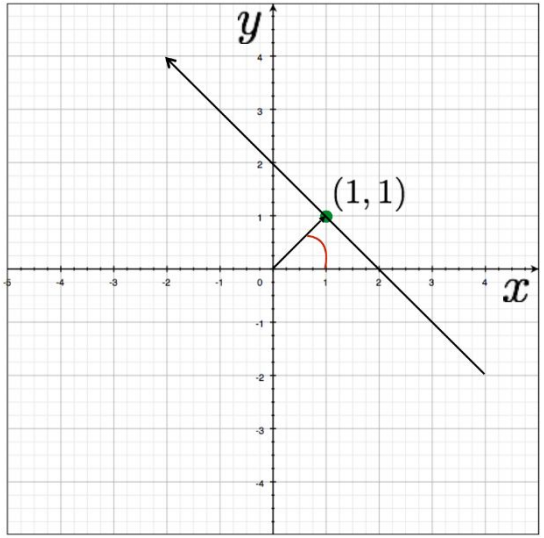
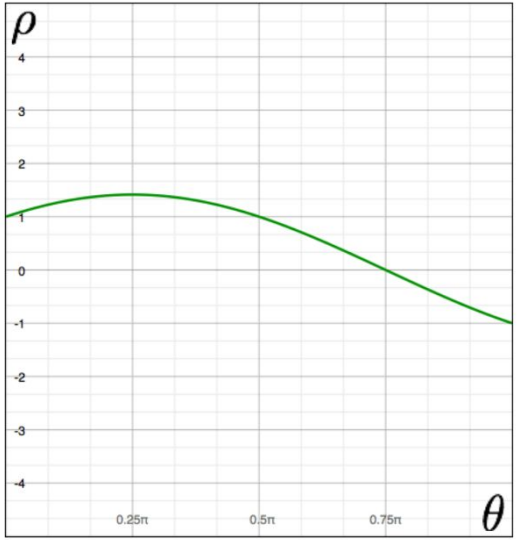


Image space

a line becomes?



Parameter space

Hough Transform: Lines

variables

$$y = mx + b$$

parameters

parameters

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

variables

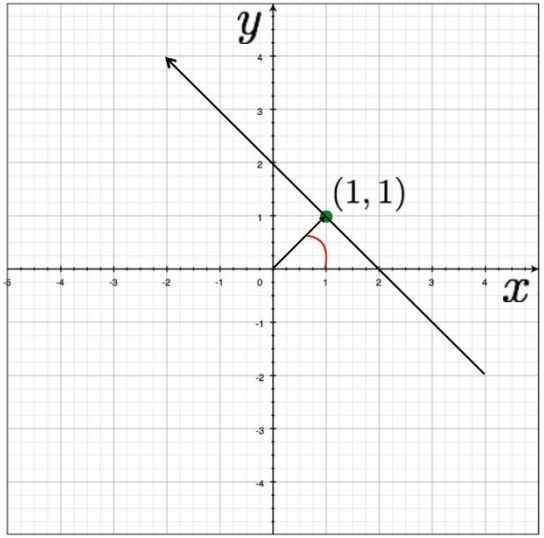
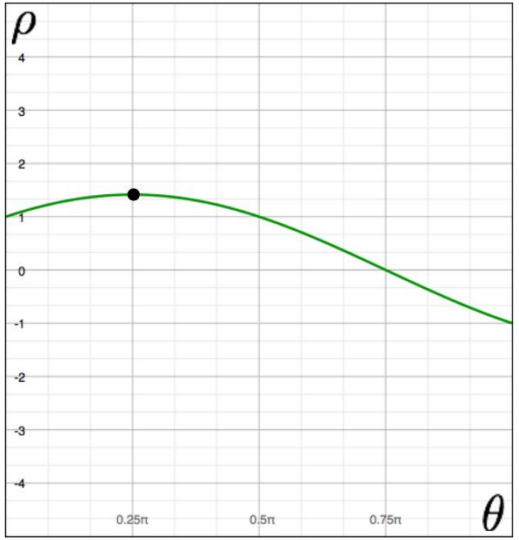


Image space

a line becomes a point



Parameter space

Hough Transform: Lines

variables

$$y = mx + b$$

parameters

parameters

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

variables

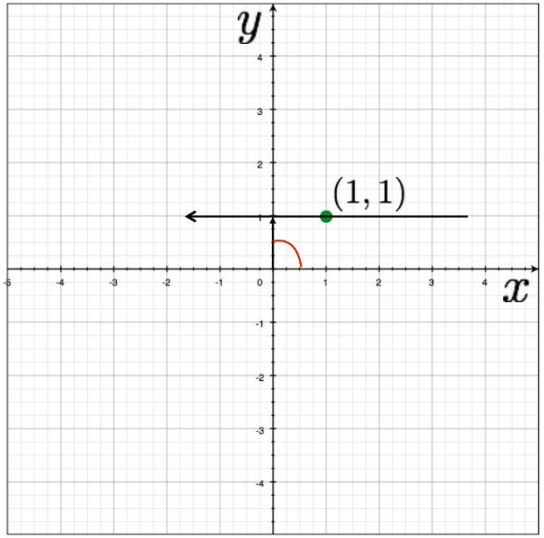
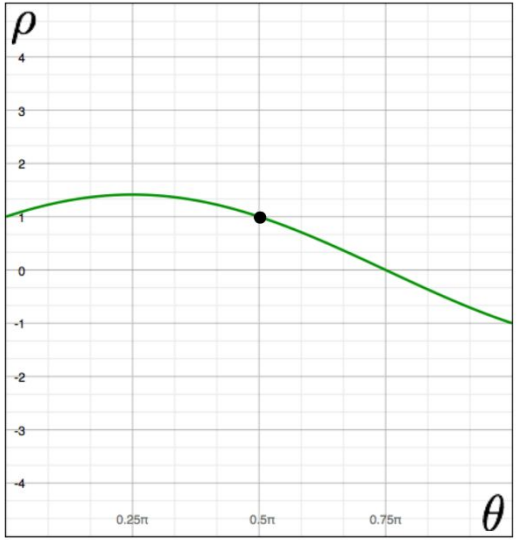


Image space

a line becomes a point



Parameter space

Hough Transform: Lines

variables

$$y = mx + b$$

parameters

parameters

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

variables

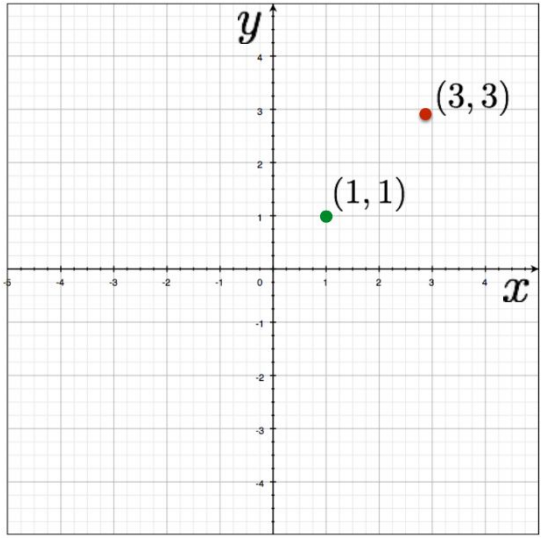
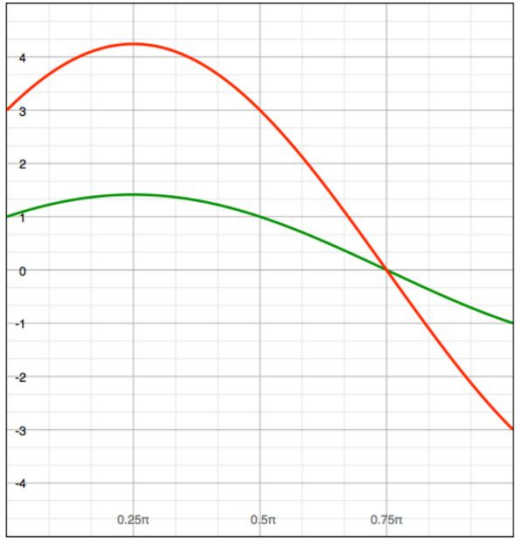


Image space

two points become?



Parameter space

Hough Transform: Lines

variables

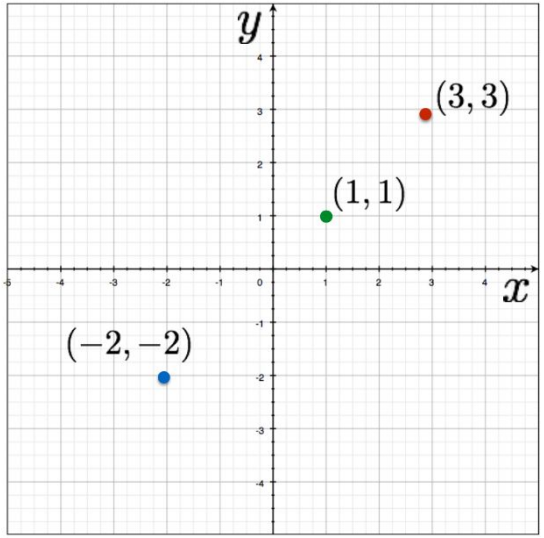
$$y = mx + b$$

parameters

parameters

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

variables



three points become?

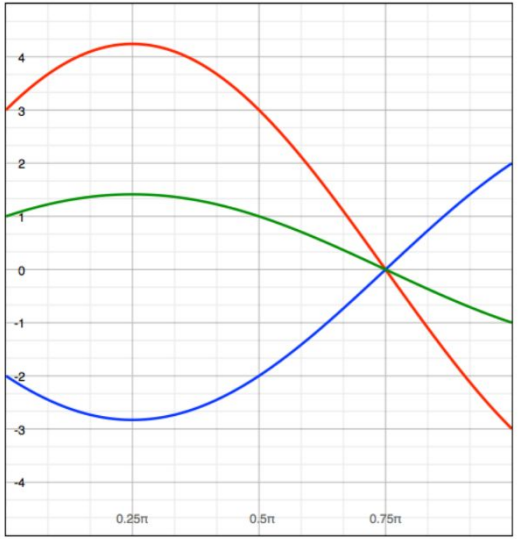


Image space

Parameter space

Hough Transform: Lines

variables

$$y = mx + b$$

parameters

parameters

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

variables

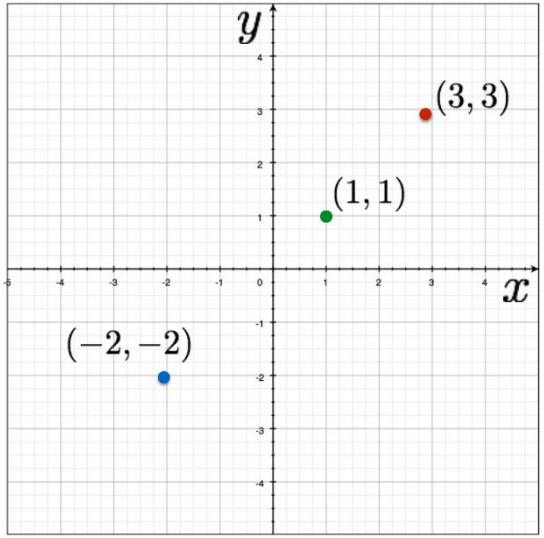
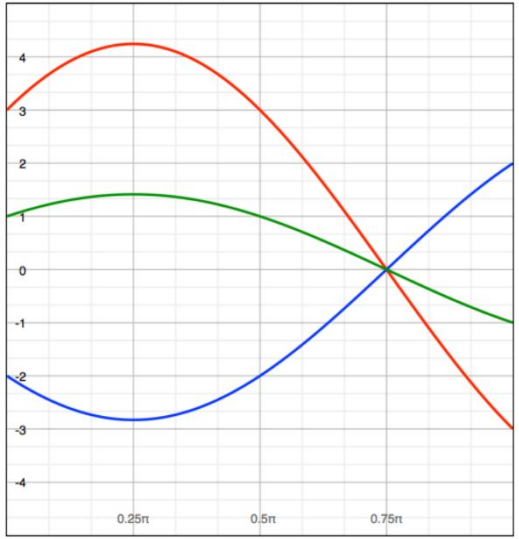


Image space

three points become?



Parameter space

Hough Transform: Lines

variables

$$y = mx + b$$

parameters

parameters

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

variables

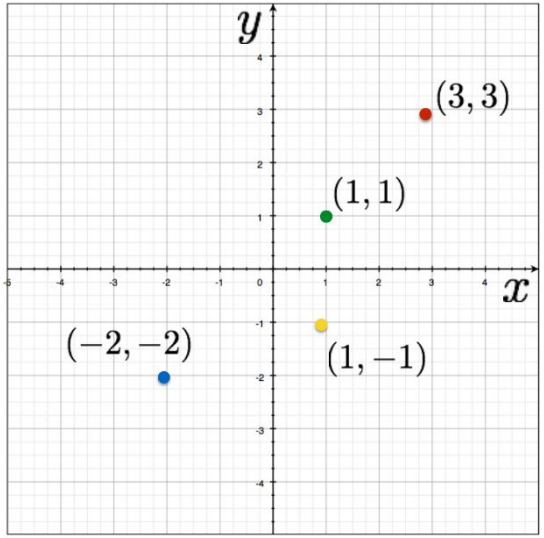
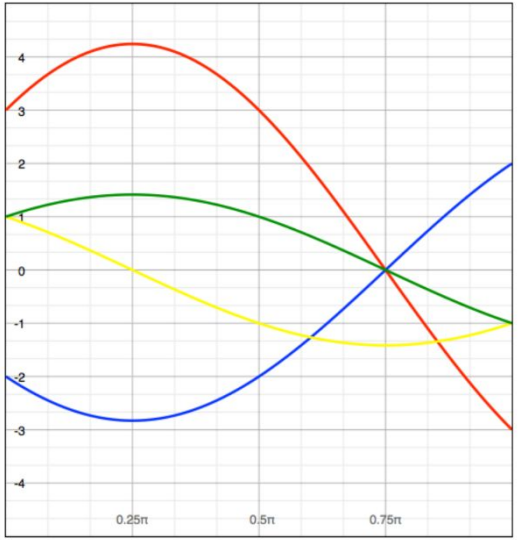


Image space

four points become?



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 θ, 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

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

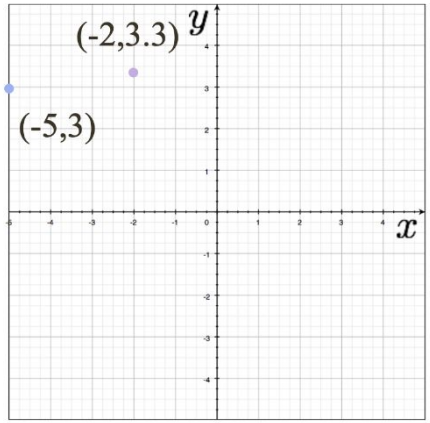
— Different choices of θ, r give different lines

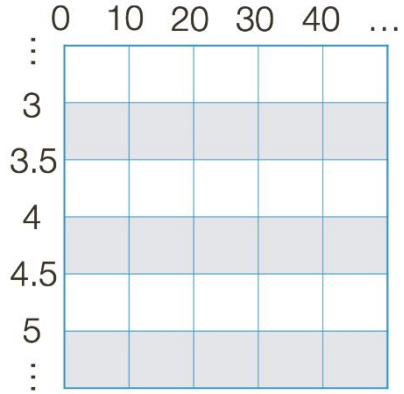
— For any (x, y) there is a one parameter family of lines through this point. Just let (x, y) be constants and for each value of θ the value of r will be determined

— Each point enters votes for each line in the family

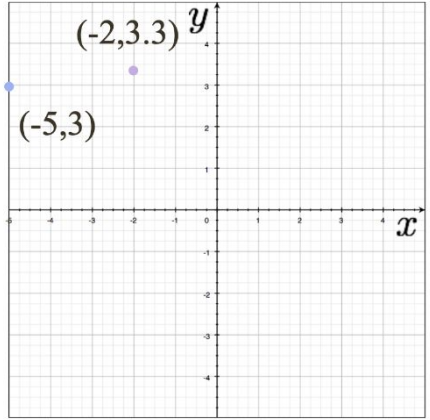
— If there is a line that has lots of votes, that will be the line passing near the points that voted for it

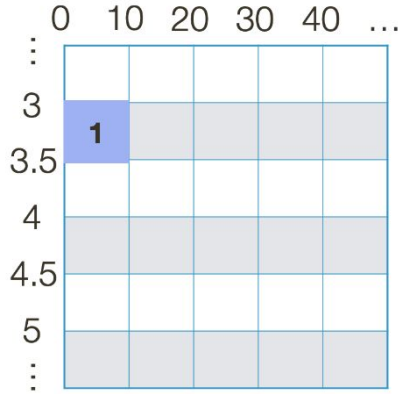
Example: Hough Transform for Lines





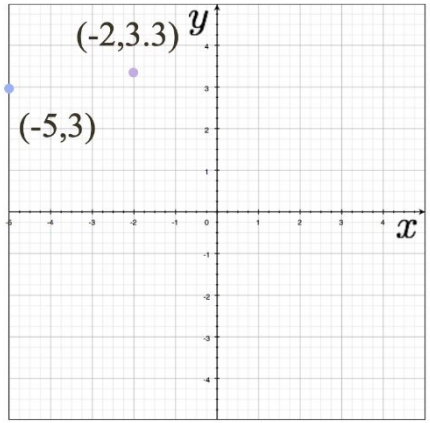
Example: Hough Transform for Lines

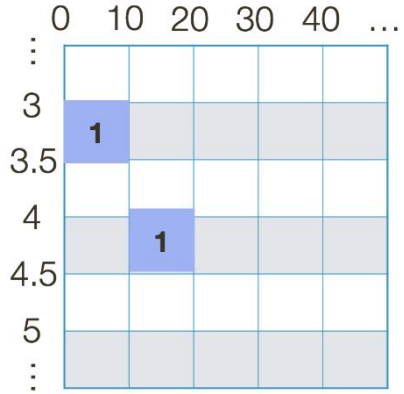




$$-5\sin(5°) - 3\cos(5°) + r = 0 => r = 3.42$$

Example: Hough Transform for Lines

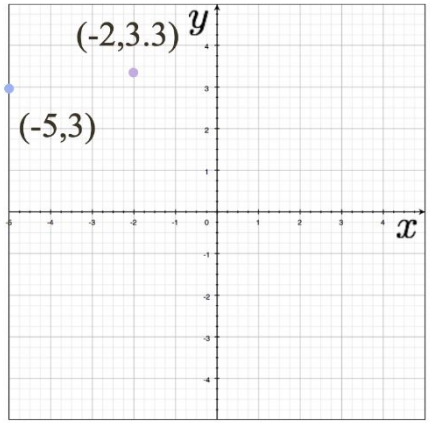


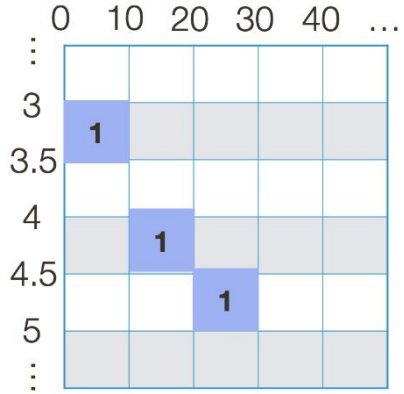


$$-5\sin(5°) - 3\cos(5°) + r = 0 => r = 3.42$$

$$-5\sin(15°) - 3\cos(15°) + r = 0 => r = 4.18$$

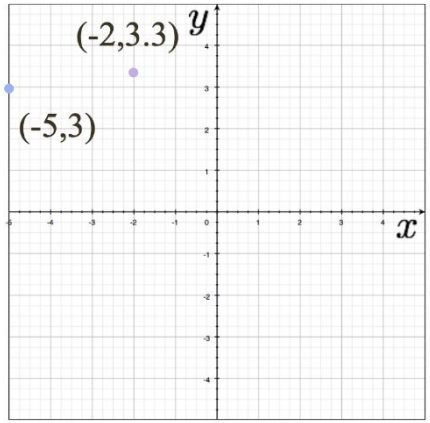
Example: Hough Transform for Lines

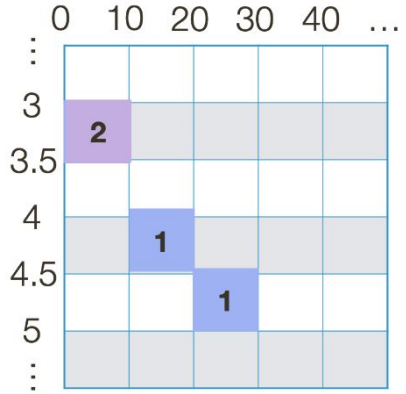




$$-5\sin(5°) - 3\cos(5°) + r = 0 => r = 3.42$$
$$-5\sin(15°) - 3\cos(15°) + r = 0 => r = 4.18$$
$$-5\sin(25°) - 3\cos(25°) + r = 0 => r = 4.83$$

Example: Hough Transform for Lines

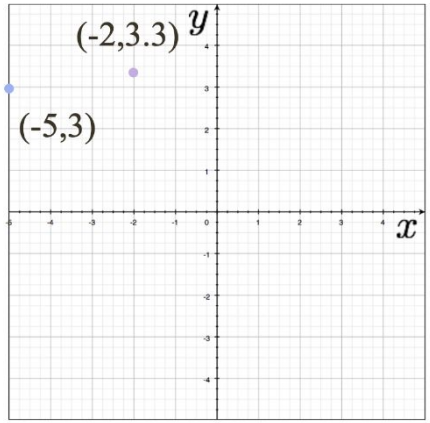


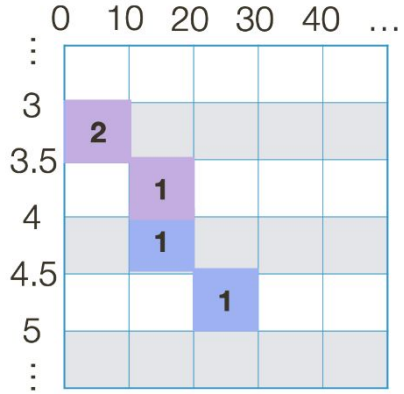


$$-5\sin(5°) - 3\cos(5°) + r = 0 => r = 3.42$$
$$-5\sin(15°) - 3\cos(15°) + r = 0 => r = 4.18$$
$$-5\sin(25°) - 3\cos(25°) + r = 0 => r = 4.83$$

$$-2\sin(5°) - 3.3\cos(5°) + r = 0 => r = 3.46$$

Example: Hough Transform for Lines

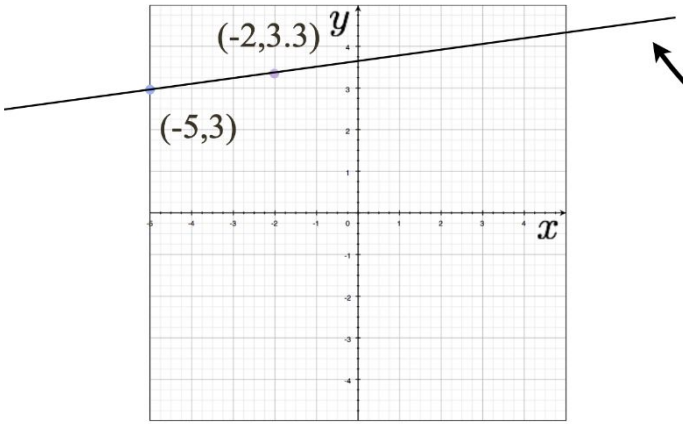


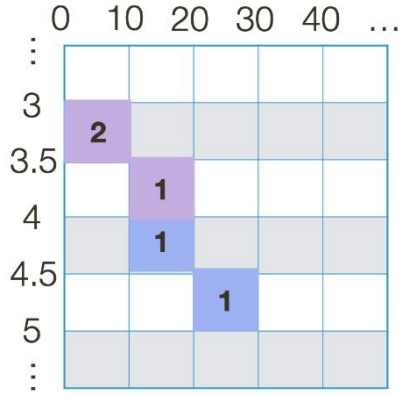


$$-5\sin(5°) - 3\cos(5°) + r = 0 => r = 3.42$$
$$-5\sin(15°) - 3\cos(15°) + r = 0 => r = 4.18$$
$$-5\sin(25°) - 3\cos(25°) + r = 0 => r = 4.83$$

$$-2\sin(5°) - 3.3\cos(5°) + r = 0 => r = 3.46$$
$$-2\sin(15°) - 3.3\cos(15°) + r = 0 => r = 3.71$$

Example: Hough Transform for Lines

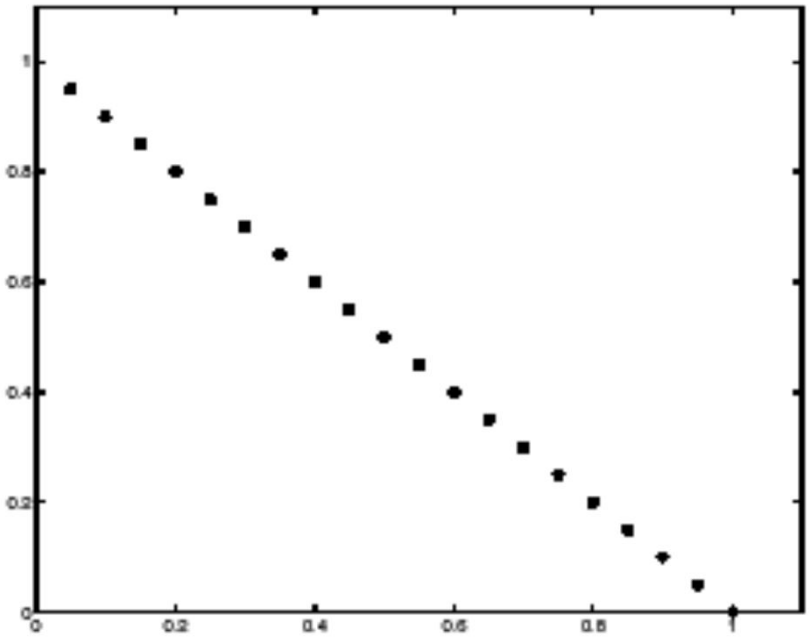




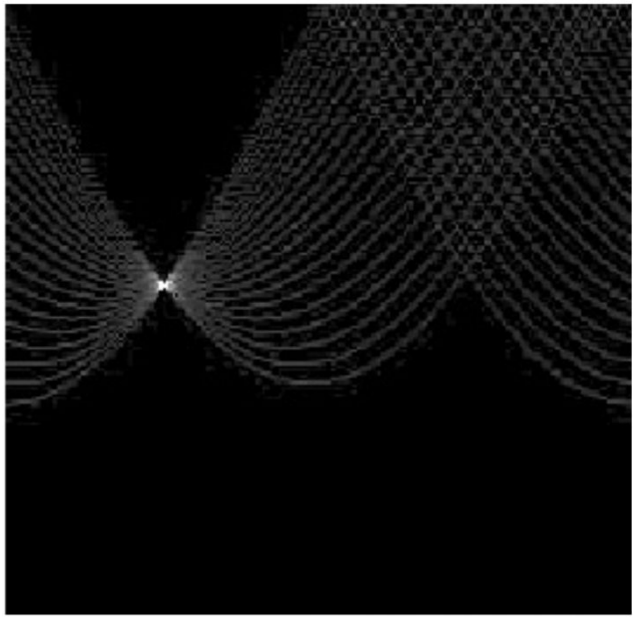
$$-5\sin(5°) - 3\cos(5°) + r = 0 => r = 3.42$$
$$-5\sin(15°) - 3\cos(15°) + r = 0 => r = 4.18$$
$$-5\sin(25°) - 3\cos(25°) + r = 0 => r = 4.83$$

$$-2\sin(5°) - 3.3\cos(5°) + r = 0 => r = 3.46$$
$$-2\sin(15°) - 3.3\cos(15°) + r = 0 => r = 3.71$$

Example: Clean Data



Tokens



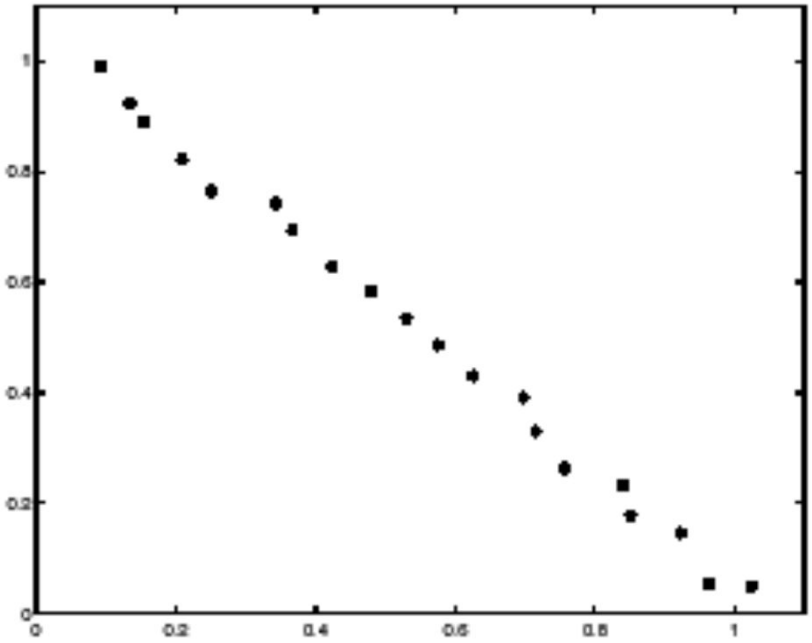
Votes

Horizontal axis is θ

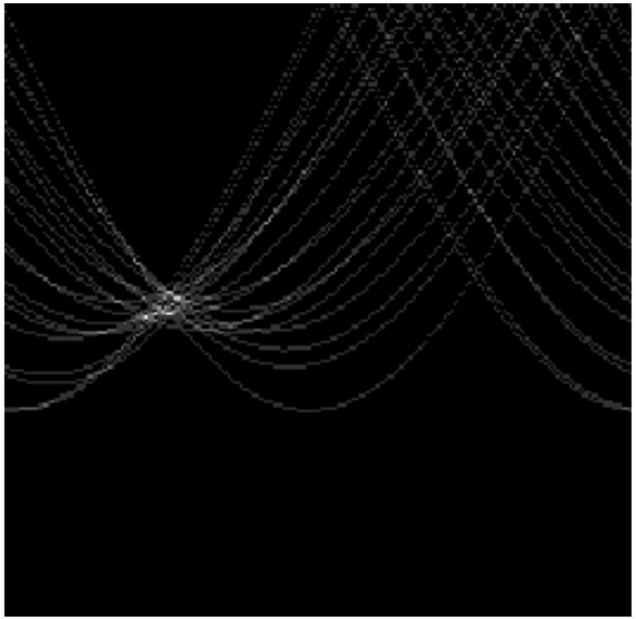
Vertical Axis is r

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

Example: Some Noise



Tokens



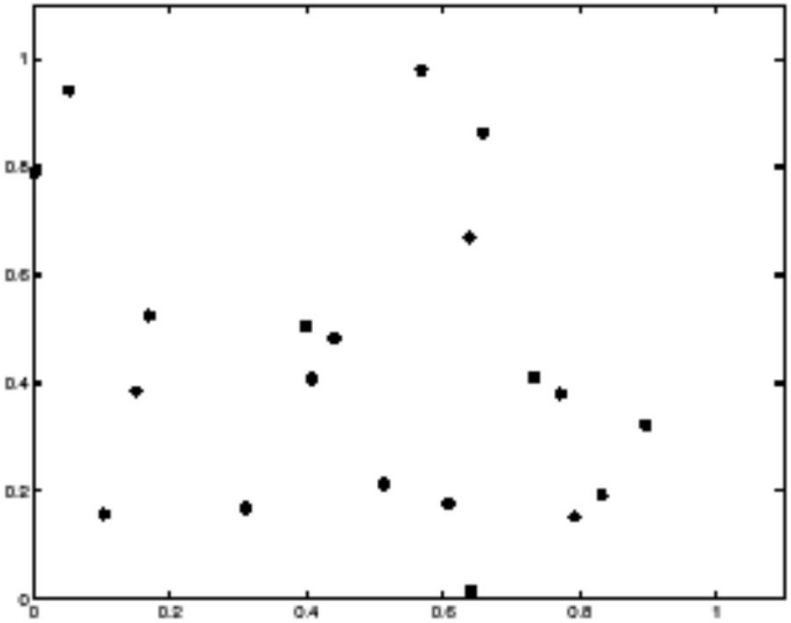
Votes

Horizontal axis is θ

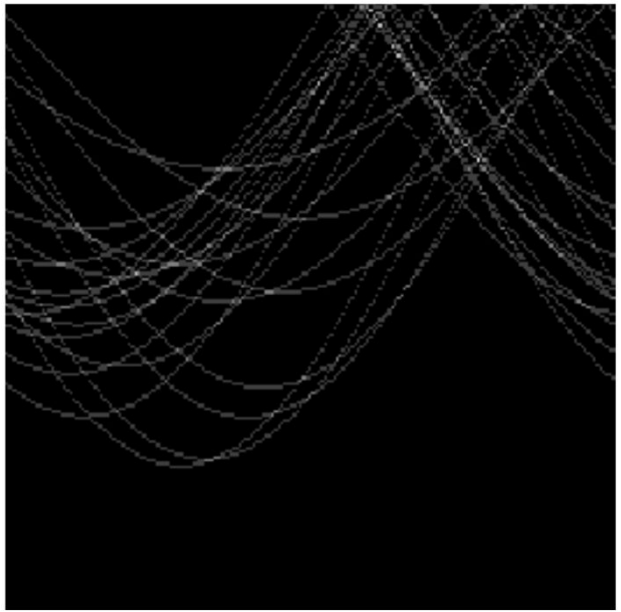
Vertical Axis is r

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

Example: Too Much Noise



Tokens



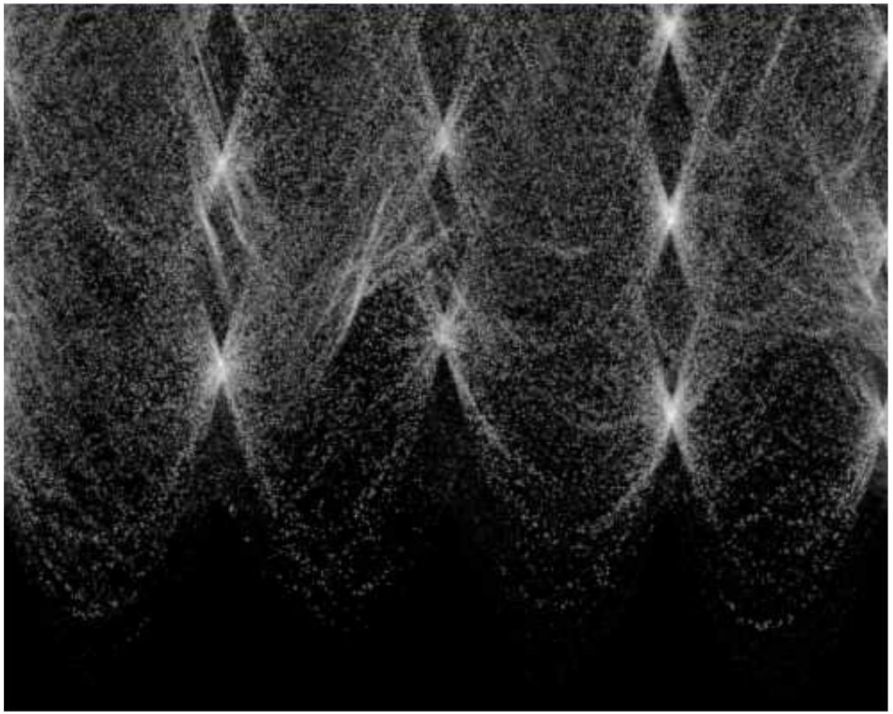
Votes

Horizontal axis is θ

Vertical Axis is r

Forsyth & Ponce (2nd ed.) Figure 10.2

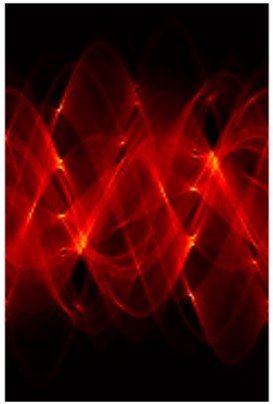
Real World **Example**



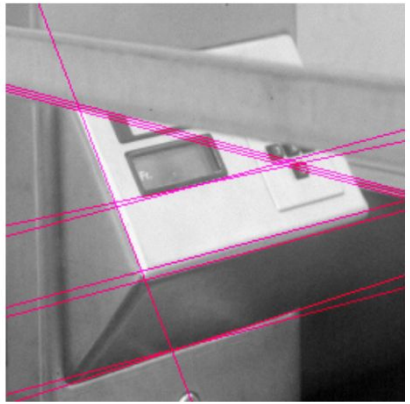
Real World **Example**

Original

Edges



Parameter space



Hough Lines

Mechanics of Hough Transform

1. Construct a quantized array to represent θ and r

2. For each point, render curve (θ, 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

Hough vs. RANSAC

Pros

1. Handles **multiple** models
2. Some robustness to noise
3. In principle, general

Cons

1. Have to bin ALL parameters: exponential in #params
2. Have to parameterize your space nicely
3. Details really, really important (a working version requires a lot more than what I showed you)

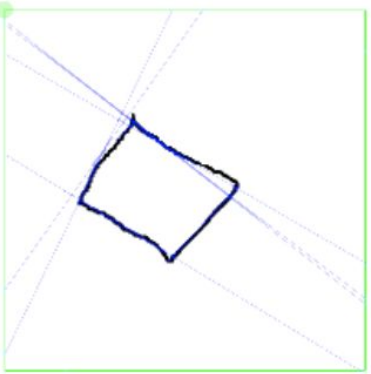
Pros

1. Ridiculously simple
2. Ridiculously effective
3. Works in general

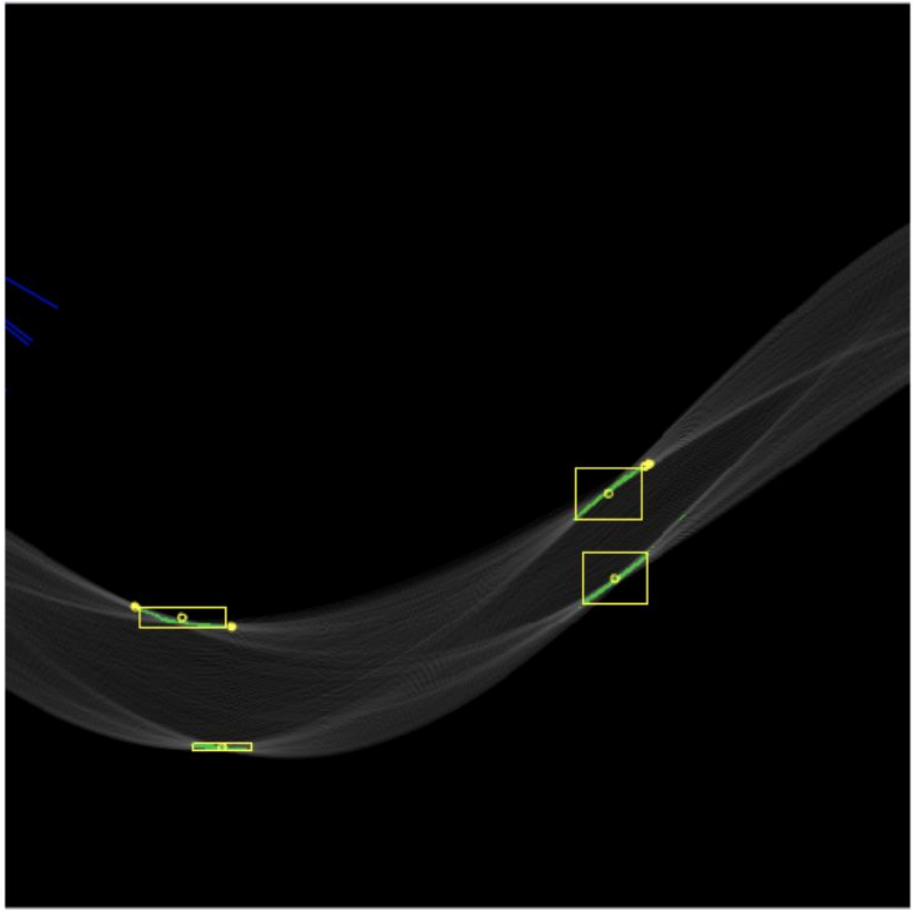
Cons

1. Have to tune parameters
2. No theory (so can't derive parameters via theory)
3. Not magic, especially with lots of outliers

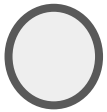
4. Can't handle multiple modes easily.



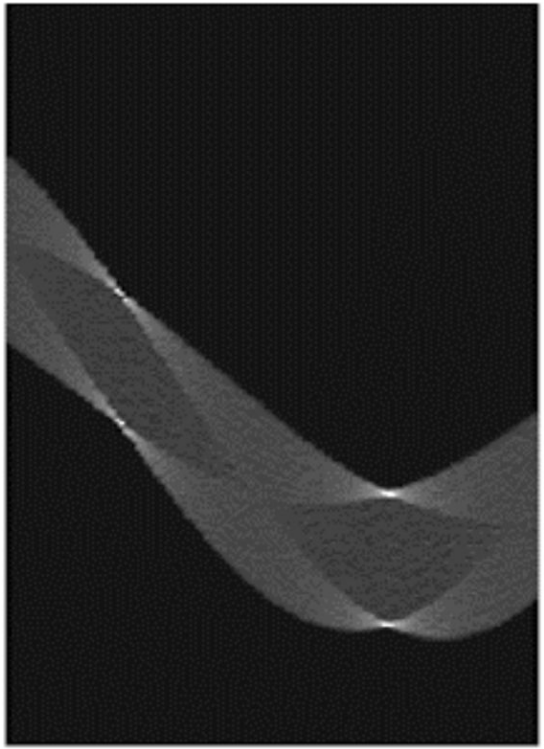
0
0
500
Thres: 30
0 2 (x=355, y=254, w=2, h=1)
1 6 (x=352, y=255, w=3, h=2)
2 1036 (x=315, y=257, w=37, h=28)
3 1008 (x=319, y=304, w=36, h=28)
4 2 (x=71, y=333, w=2, h=1)
5 528 (x=74, y=334, w=48, h=11)
6 3 (x=124, y=344, w=3, h=1)
7 128 (x=104, y=409, w=32, h=4)
Blobs: 8

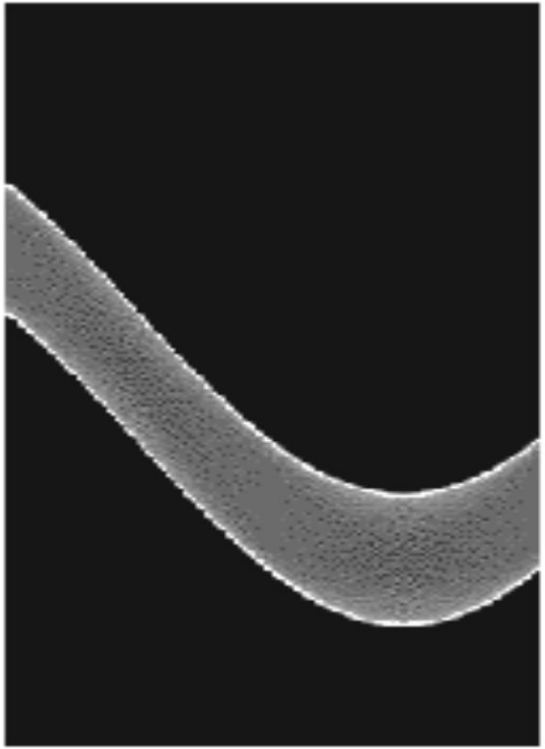


Square



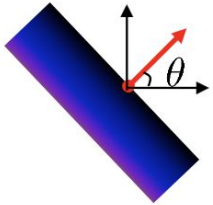
Circle





Incorporating image gradients

- Recall: when we detect an edge point, we also know its gradient direction
- But this means that the line is uniquely determined!



$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

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

- Modified Hough transform:

For each edge point (x,y)
    θ = gradient orientation at (x,y)
    ρ = x cos θ + y sin θ
    H(θ, ρ) = H(θ, ρ) + 1
end

Comparison with/without edge orientation

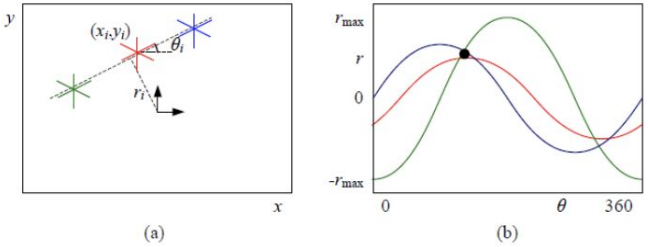


Figure 7.46 *Original Hough transform: (a) each point votes for a complete family of potential lines $r_i(\theta) = x_i \cos\theta + y_i \sin\theta$; (b) each pencil of lines sweeps out a sinusoid in (r, θ); their intersection provides the desired line equation.*

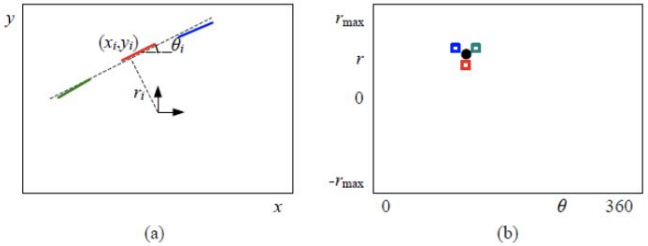


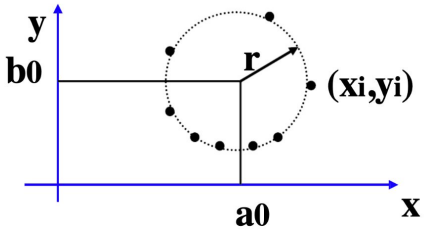
Figure 7.47 *Oriented Hough transform: (a) an edgel re-parameterized in polar (r, θ) coordinates, with $\hat{\mathbf{n}}_i = (\cos\theta_i, \sin\theta_i)$ and $r_i = \hat{\mathbf{n}}_i \cdot \mathbf{x}_i$; (b) (r, θ) accumulator array, showing the votes for the three edgels marked in red, green, and blue.*

Important:

Edge/line detection algorithms will also provide gradient orientation (remember Edge filter with derivative of Gaussian).

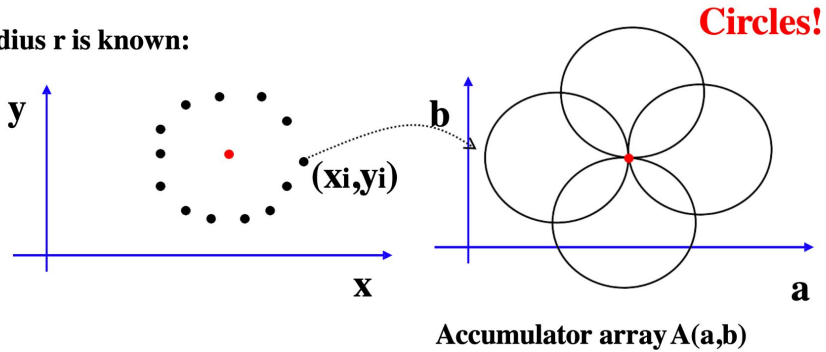
Now, we don't have to search through all orientations since the orientation Θ is known !

We can use Hough transforms to detect more complex shapes: circles



Equation of Circle: $(x_i - a_0)^2 + (y_i - b_0)^2 = r^2$

If radius r is known:



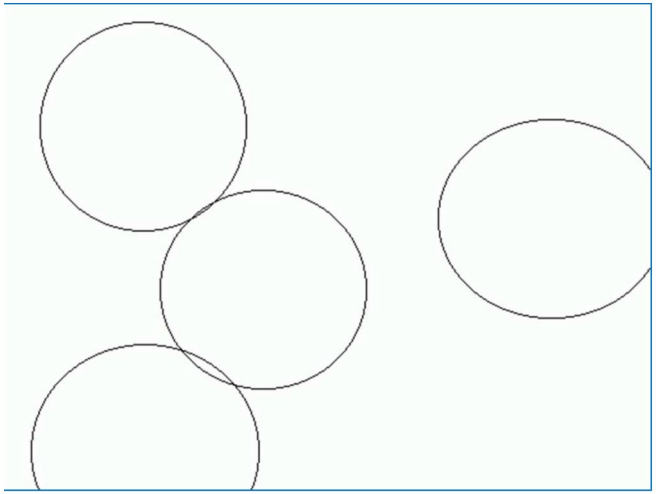
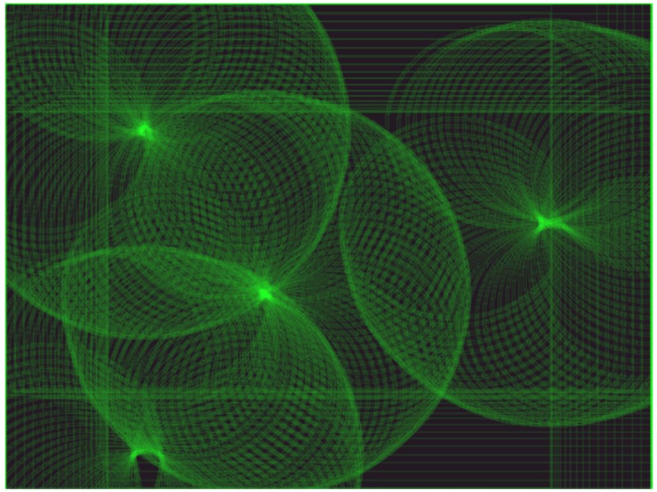
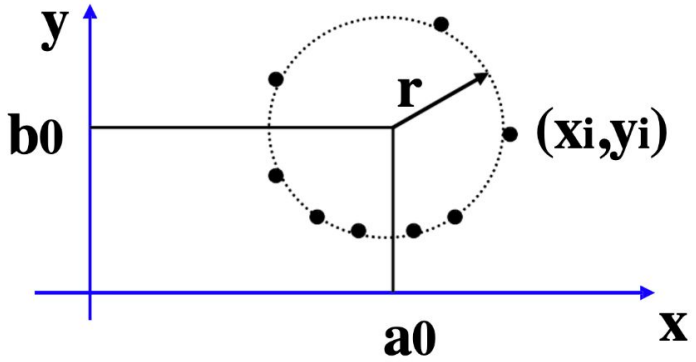


Image Space



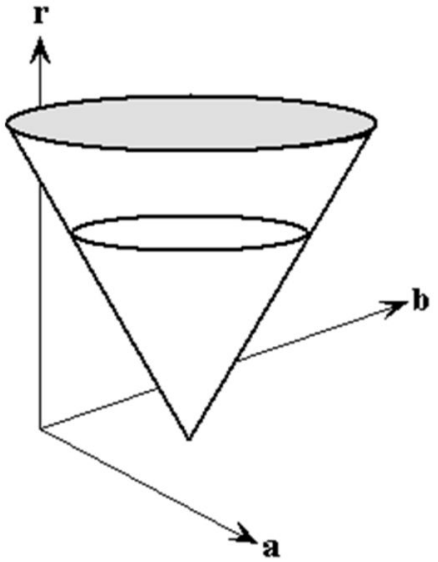
Hough Space with fixed r

Circle centers show accumulation of votes

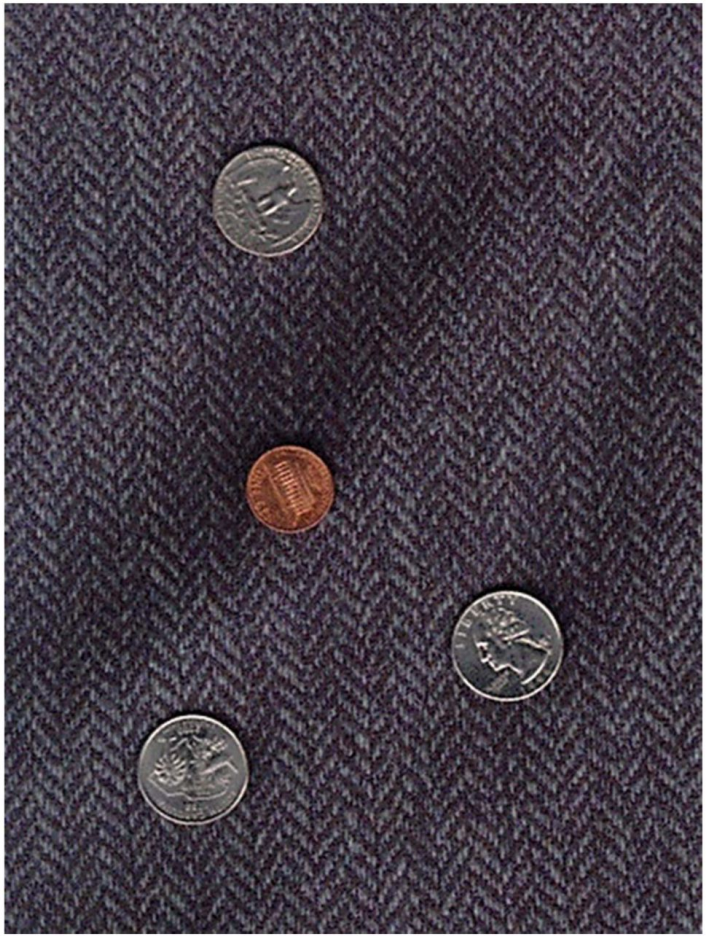


If r is not known
Use accumulator array A(a,b,r)

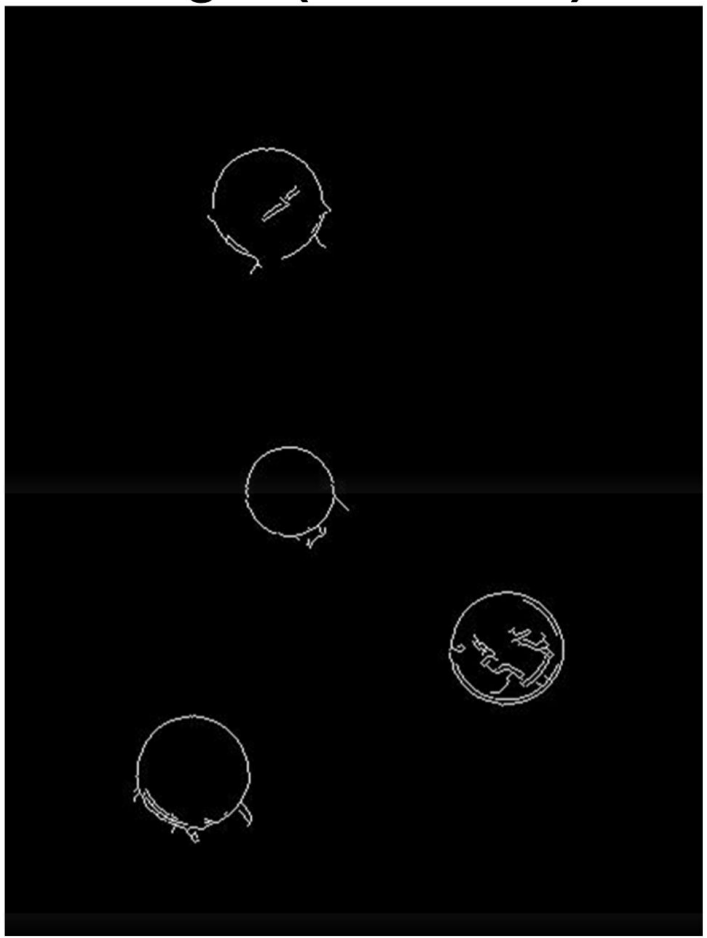
For each (xi,yi) increment A(a,b,r)
such that $(x_i - a)^2 + (y_i - b)^2 = r^2$

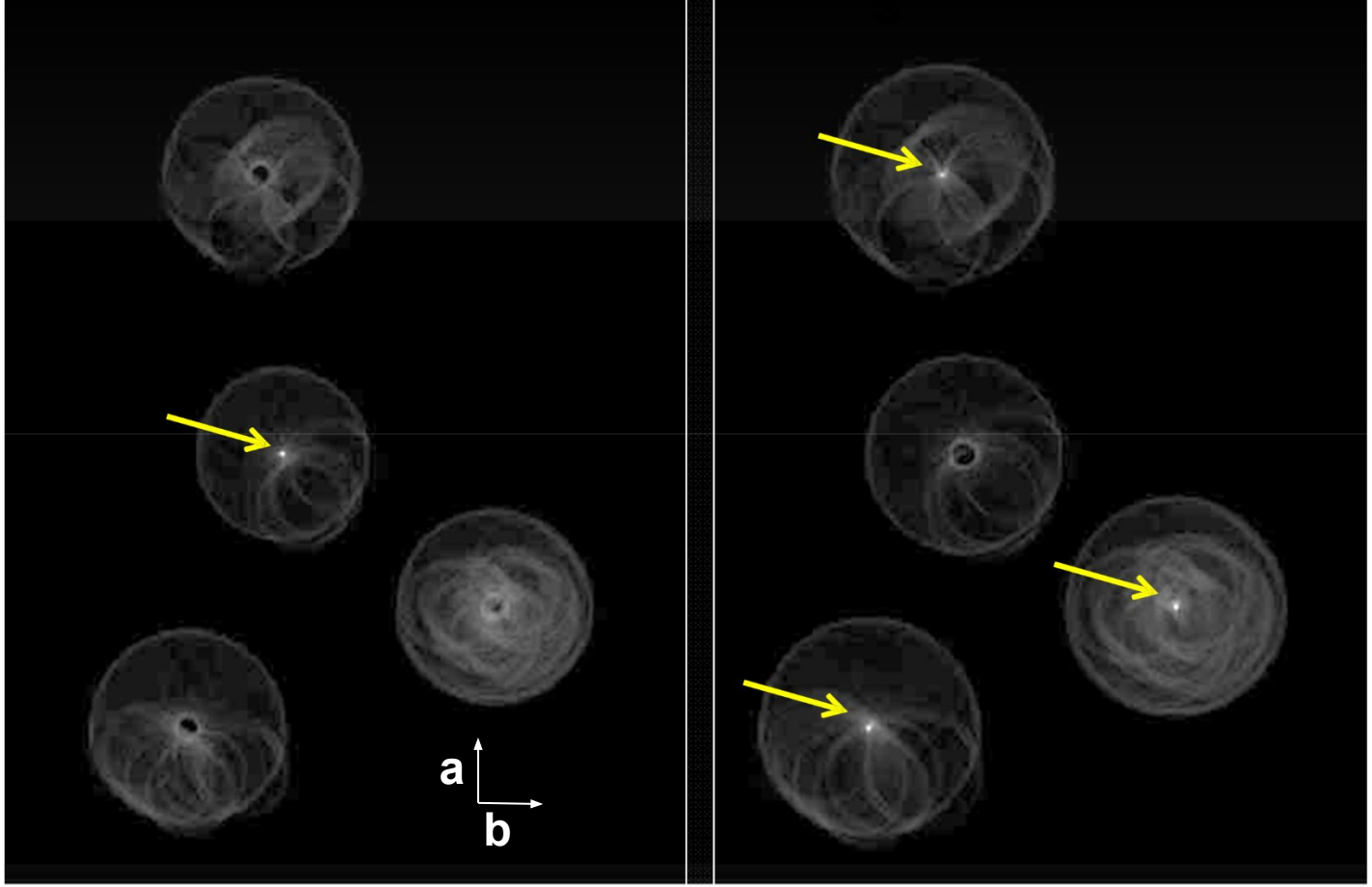


Original

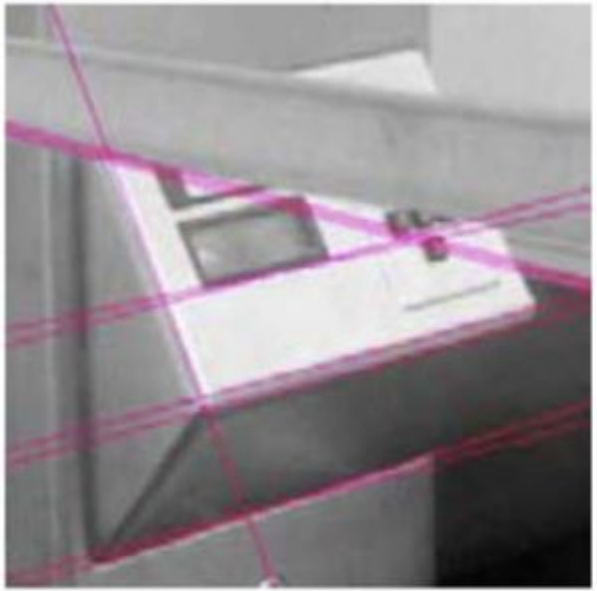


Edges (note noise)



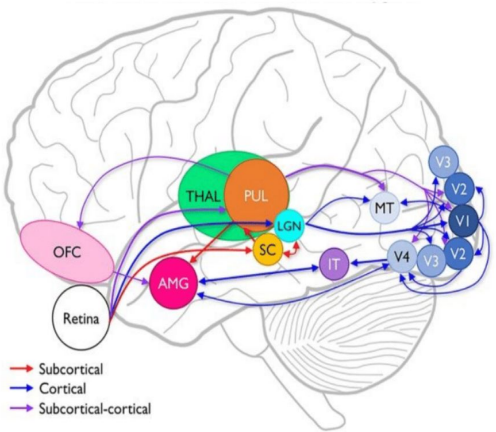


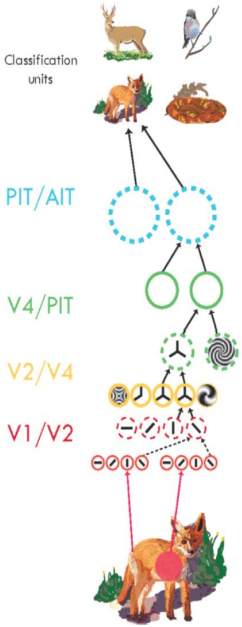
r=10 r=20



simple model: **lines**

simple model: **circles**



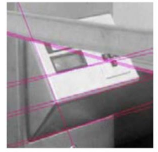


High-level vision (object, scene)

After midterm

Mid-level vision (segmentation)

Mid level features
Lecture 5-7



simple model: **lines**

simple model: **circles**

Low-level vision (Edge, corner)

Low level features

Lecture 3-4

243	239	240
242	239	218
243	242	123

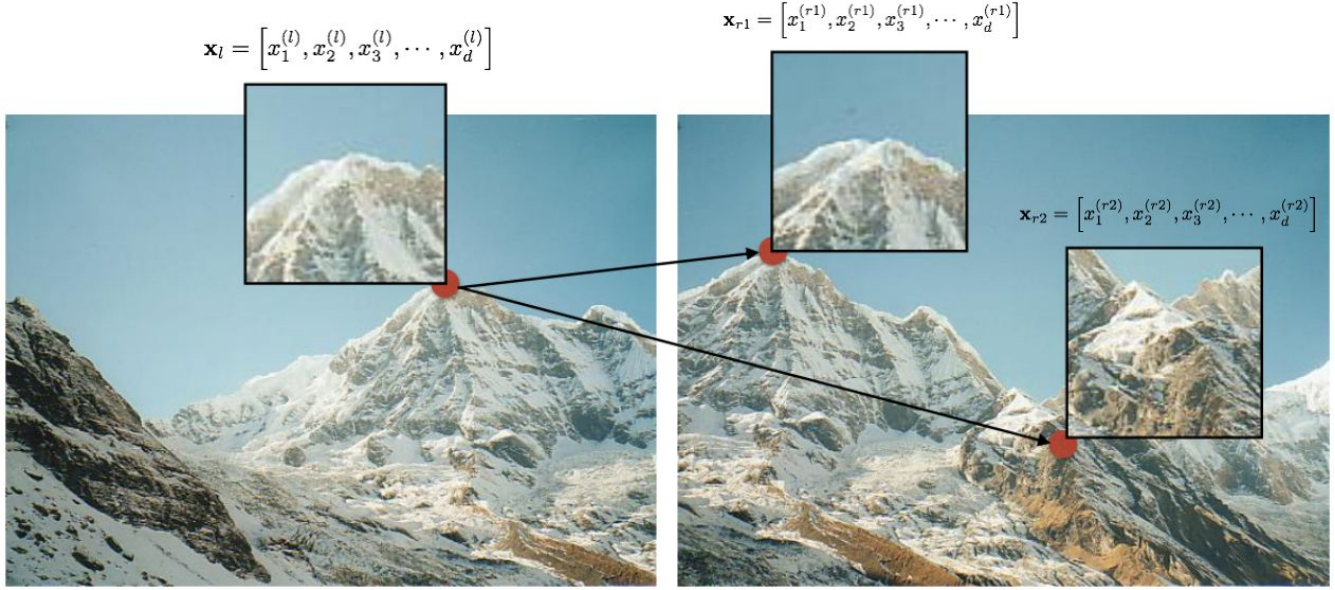
patches

Lecture 2

243 pixels

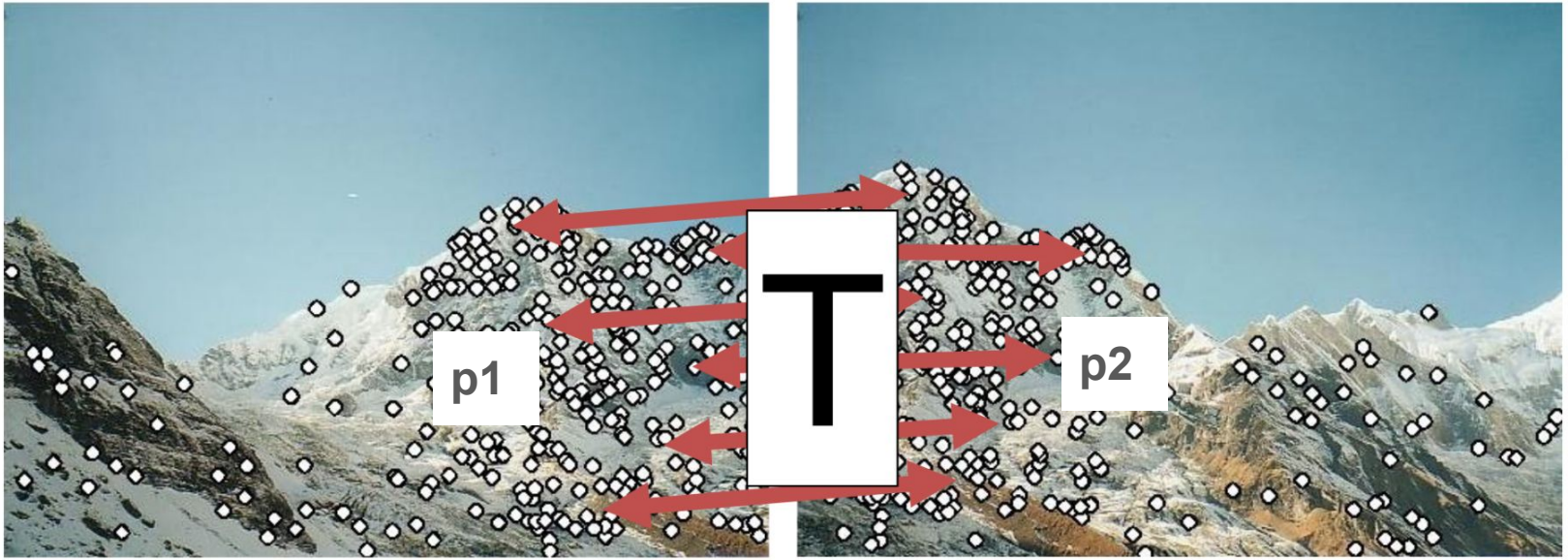
start

Tying it to our application goal - image stitching



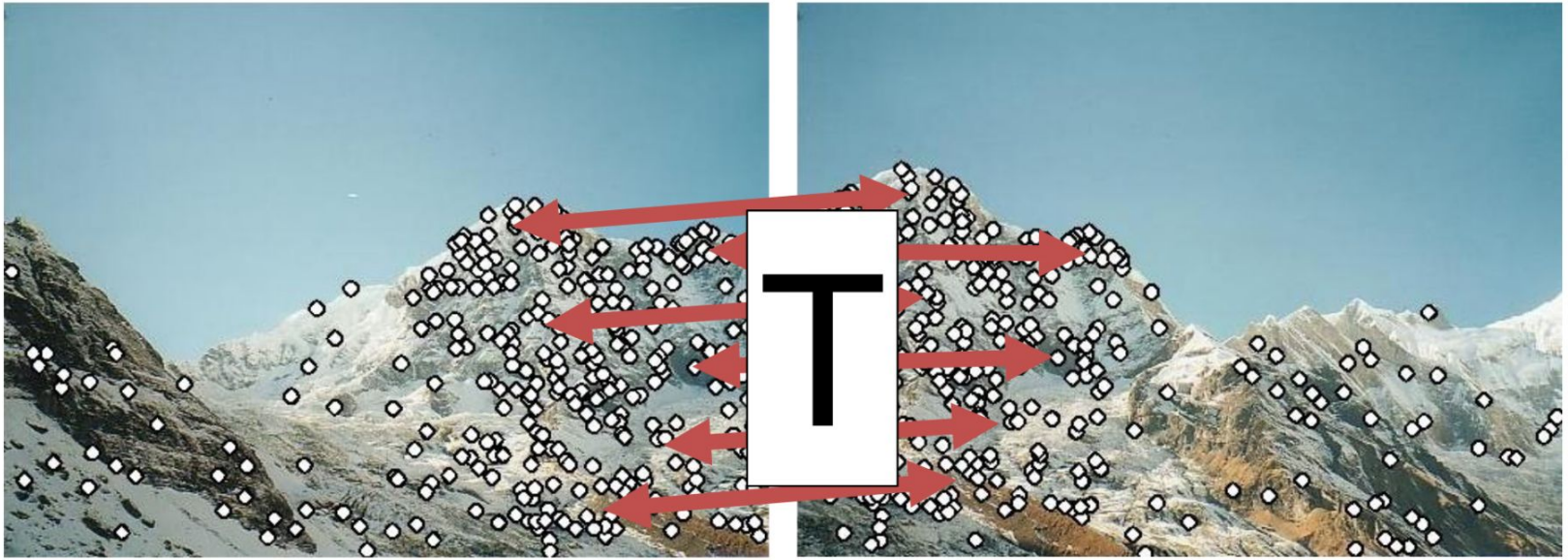
1) Detect “important” points - e.g. Harris corner detector
2) Describe the information round them - e.g. SIFT
3) Match points with similar descriptions - L2 distance

Next step is to fit a transformation model



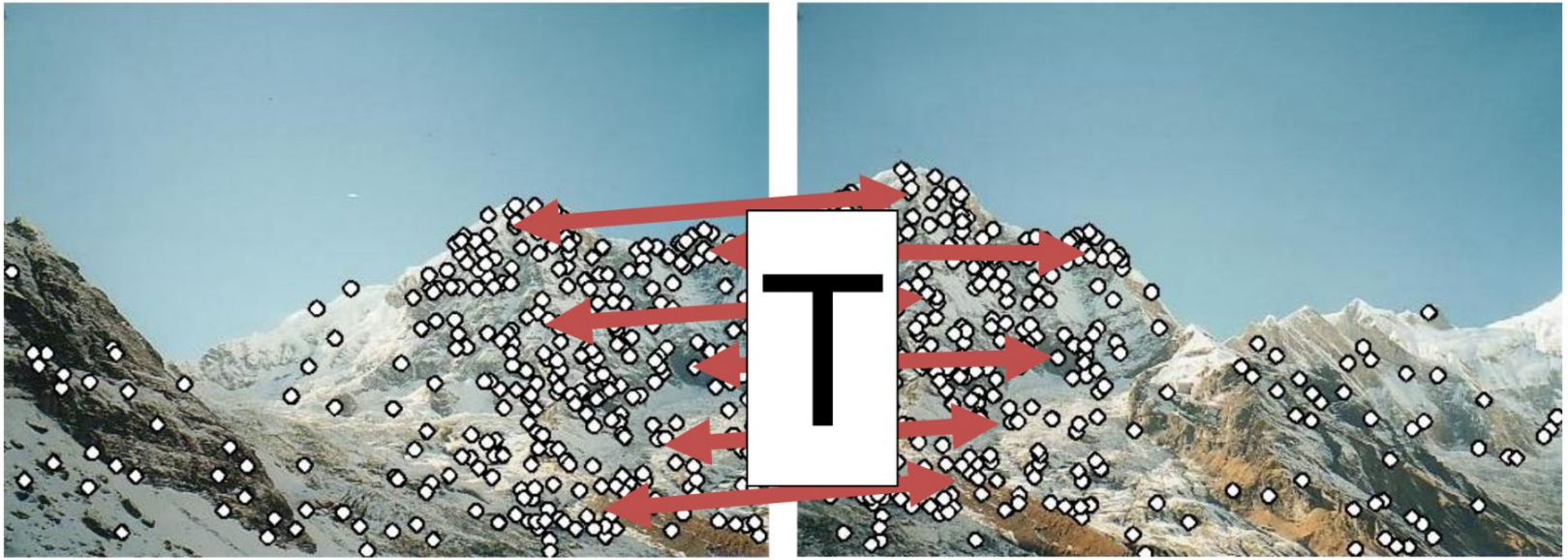
4) Find transformation T such that p1 = Tp2

Today we covered model fitting in a general sense



Abstractly: Fit models to **data** with minimum **error** and robust to outliers e.g. line or circle models.

Next lecture is the conclusion of this series



Fit **geometric transformation models** that align two images with minimum error and robust to outliers.

Next: code lab

Next week: image warping and stitching