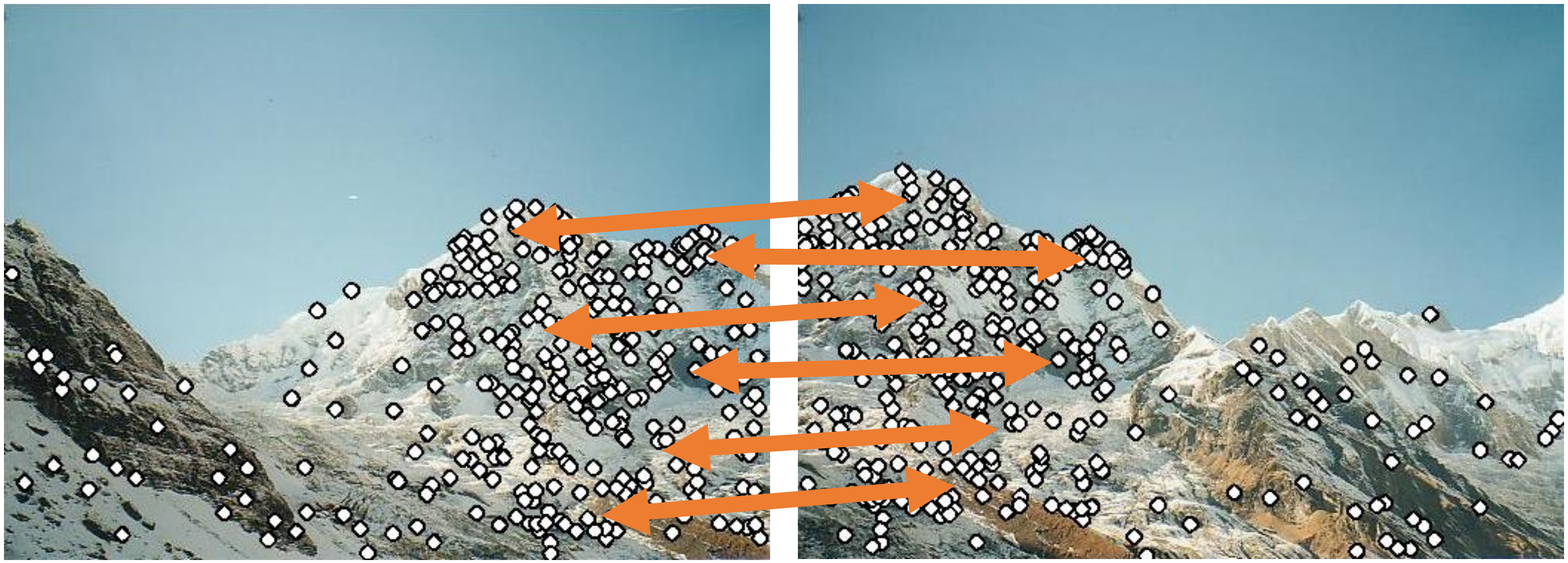
Lecture 11: Transforms and Fitting

Administrative

- HW2 due 1 week from yesterday, Wednesday 2/19 11:59pm

- HW3 released, due 2 weeks from tomorrow, Friday 2/28 11:59pm

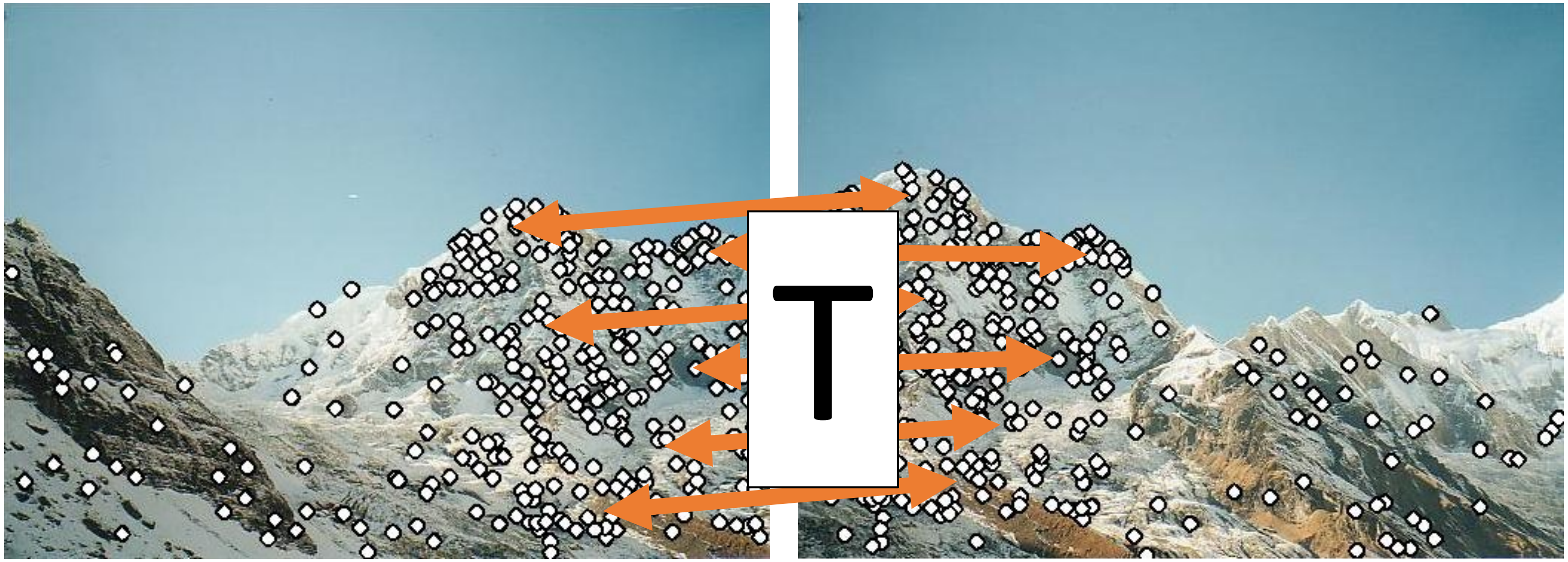
Last Class



1. How do we find distinctive / easy to locate features? *(Harris/Laplacian of Gaussian)*
2. How do we describe the regions around them? *(Normalize window, use histogram of gradient orientations)*

Our Goal

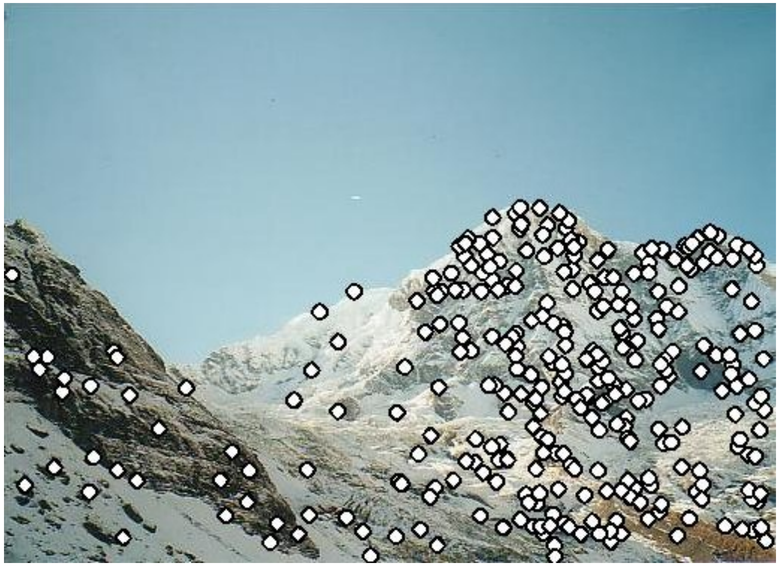
Solving for a Transformation



3: Solve for transformation T (e.g. such that **p1 ≡ T p2**) that fits the matches well

Remember: Human vs Computer

You, with your gigantic brain, see:



The computer sees:

097	097	097	097	097	097	097	097	096	097	097	096	096	096
100	100	100	100	100	100	101	101	102	101	100	100	100	099
105	105	105	105	105	105	105	103	102	102	101	103	104	105
109	109	109	109	109	110	107	118	145	132	120	112	106	103
113	113	113	112	112	113	110	129	160	160	164	162	157	151
118	117	118	123	119	118	112	125	142	134	135	139	139	175
123	121	125	162	166	157	149	153	160	151	150	146	137	168
127	127	125	168	147	117	139	135	126	147	147	149	156	160
133	130	150	179	145	132	160	134	150	150	111	145	126	121
138	134	179	185	141	090	166	117	120	153	111	153	114	126
144	151	188	178	159	154	172	147	159	170	147	185	105	122
152	157	184	183	142	127	141	133	137	141	131	147	144	147
130	147	185	180	139	131	154	121	140	147	107	147	120	128
035	102	194	175	149	140	179	128	146	168	096	163	101	125

You should expect **noise** (not at quite the right pixel) and **outliers** (random matches)

Today

- How do we fit **models** (i.e., a parameteric representation of data that's smaller than the data) to data?
- How do we handle:
  - **Noise** – least squares / total least squares
  - **Outliers** – RANSAC (random sample consensus)
  - **Multiple models** – Hough Transform (can also make RANSAC handle this with some effort)

Running Example: Lines

- We'll handle lines as our **models** today since you should be familiar with them
- Next class will cover more complex models. I promise we'll eventually stitch images together
- You can apply today's techniques on next class's models

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 candidate model, how well does it fit the data?

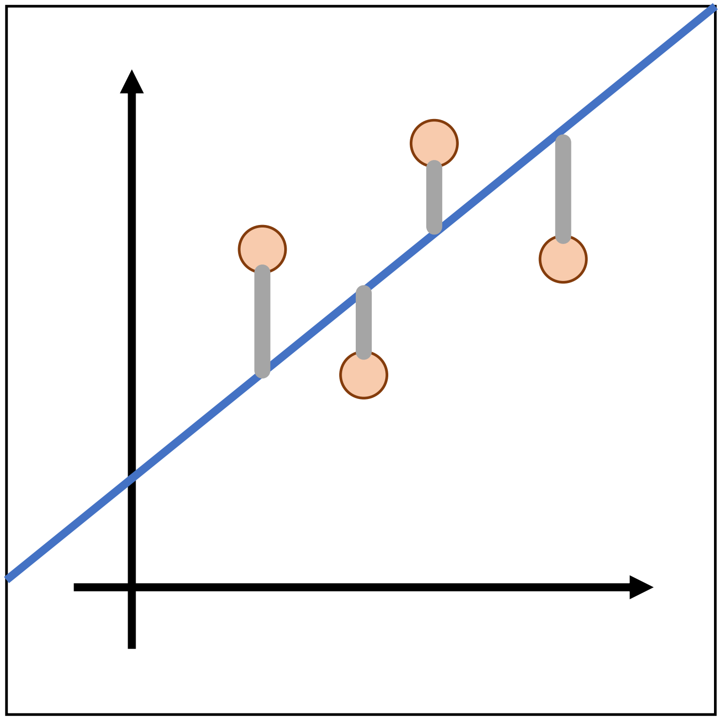
Example: Least-Squares Regression

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

$$\sum_{i=1}^{k}(y_i - \boldsymbol{w}^T\boldsymbol{x_i})^2 \longrightarrow \|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{w}\|_2^2$$

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

Solving Least Squares

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

$$\frac{\partial}{\partial \boldsymbol{w}} \|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{w}\|_2^2 = 2\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} - 2\boldsymbol{X}^T\boldsymbol{Y}$$

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

$$\boldsymbol{0} = 2\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} - 2\boldsymbol{X}^T\boldsymbol{Y}$$

$$\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} = \boldsymbol{X}^T\boldsymbol{Y}$$

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

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

(Derivation for the Curious)

$$\|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{w}\|_2^2 = (\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{w})^T(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{w})$$

$$= \boldsymbol{Y}^{\boldsymbol{T}}\boldsymbol{Y} - \mathbf{2}\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{Y} + (\boldsymbol{X}\boldsymbol{w})^{\boldsymbol{T}}\boldsymbol{X}\boldsymbol{w}$$

$$\frac{\partial}{\partial \boldsymbol{w}} (\boldsymbol{X}\boldsymbol{w})^{\boldsymbol{T}}(\boldsymbol{X}\boldsymbol{w}) = 2\left(\frac{\partial}{\partial \boldsymbol{w}}\boldsymbol{X}\boldsymbol{w}^T\right)\mathbf{Xw} = \mathbf{2X^TXw}$$

$$\frac{\partial}{\partial \boldsymbol{w}} \|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{w}\|_2^2 = 0 - 2\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{Y} + 2\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}\boldsymbol{w}$$

$$= 2\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}\boldsymbol{w} - 2\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{Y}$$

Two Solutions for Finding W

In One Go

Implicit form
(normal equations)

$$X^T X w = X^T Y$$

Explicit form
(don't do this)

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

Iteratively

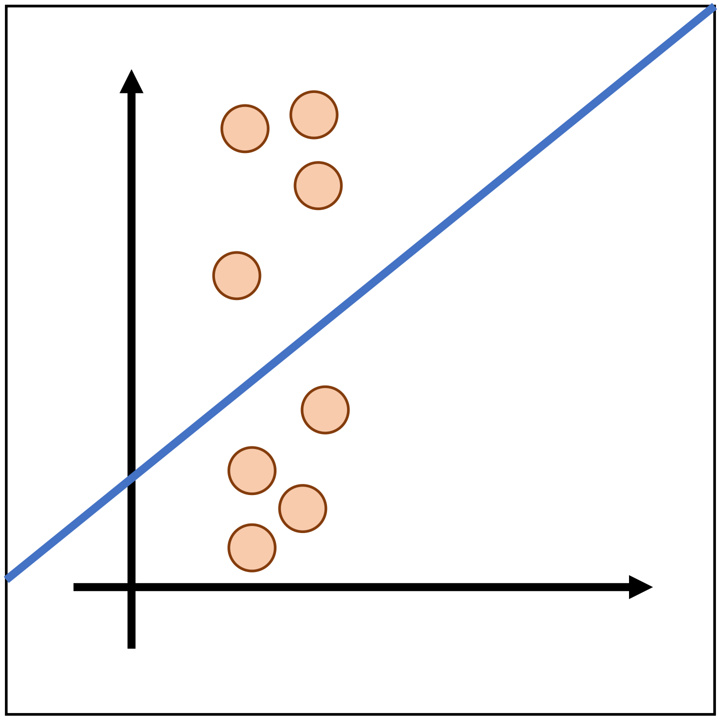
Recall: gradient is also direction that makes function go up the most.
What could we do?

$$w_0 = 0$$

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

What’s the Problem?

- Vertical lines impossible! (y = mx + b)
- Not rotationally invariant: the line will change depending on orientation of points



Alternate Formulation

Recall: $ax + by + c = 0$

$$\boldsymbol{l}^T \boldsymbol{p} = 0$$

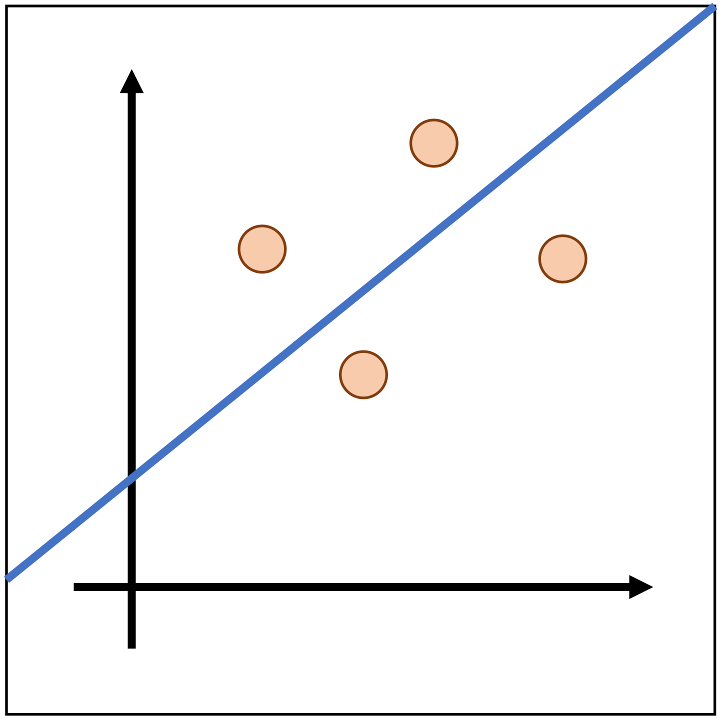
$$\boldsymbol{l} \equiv [a, b, c] \quad \boldsymbol{p} \equiv [x, y, 1]$$

Can always rescale **l.**

Pick a,b,d such that

$$\|\boldsymbol{n}\|_2^2 = \|[a, b]\|_2^2 = 1$$

$$d = -c$$



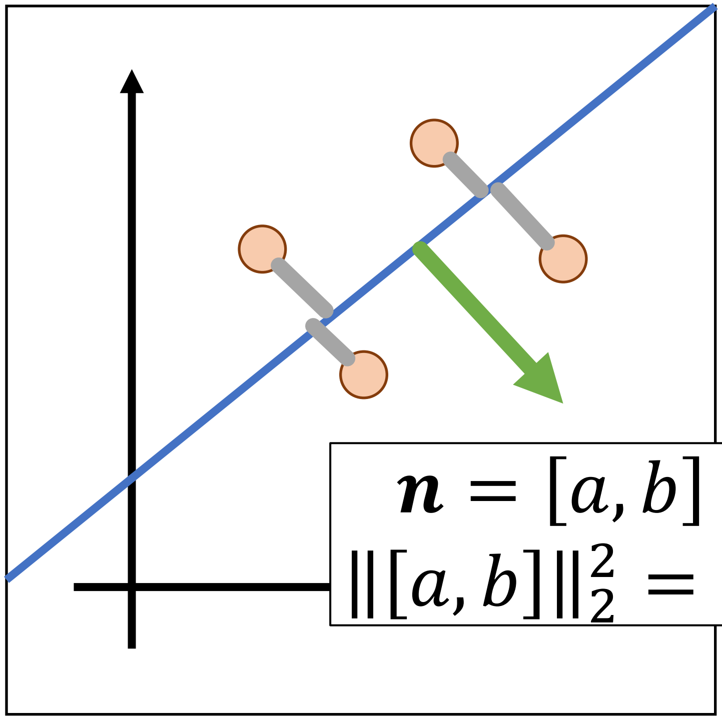
Alternate Formulation

Now: $ax + by - d = 0$

$\boldsymbol{n}^T[x, y] - d = 0$

Point to line distance:

$$\frac{\boldsymbol{n}^T[x, y] - d}{\|\boldsymbol{n}\|_2^2} = \boldsymbol{n}^T[x, y] - d$$



$\boldsymbol{n} = [a, b]$

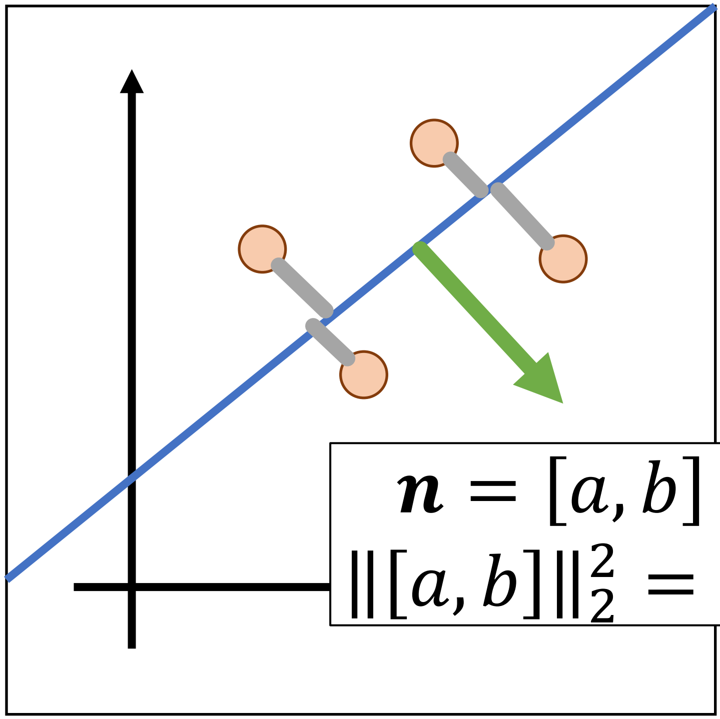
$\|[a, b]\|_2^2 = 1$

Total Least Squares

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

Model: $(\mathbf{n},d)$, $||n||^2 = 1$
$\mathbf{n}^T[x_i,y_i]-d=0$

Objective function:
$(\mathbf{n}^T[x_i,y_i]-d)^2$



$$\boldsymbol{n} = [a, b]$$
$$\|[a, b]\|_2^2 = 1$$

Total Least Squares Setup

Figure out objective first, then figure out ||n||=1

$$\sum_{i=1}^{k} \left(\boldsymbol{n}^T[x, y] - d\right)^2 \longrightarrow \|\boldsymbol{X}\boldsymbol{n} - \boldsymbol{1}d\|_2^2$$

$$\boldsymbol{X} = \begin{bmatrix} x_1 & y_1 \\ \vdots & \vdots \\ x_k & y_k \end{bmatrix} \quad \boldsymbol{1} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \quad \boldsymbol{n} = \begin{bmatrix} a \\ b \end{bmatrix} \quad \boldsymbol{\mu} = \frac{1}{k}\boldsymbol{1}^T\boldsymbol{X}$$

The mean / center of mass of the points: we'll use it later

Solving Total Least Squares

$$\|\boldsymbol{Xn} - \mathbf{1}d\|_2^2 = (\boldsymbol{Xn} - \mathbf{1}d)^T(\boldsymbol{Xn} - \mathbf{1}d)$$

$$= (\boldsymbol{Xn})^{\boldsymbol{T}}(\boldsymbol{Xn}) - 2d\mathbf{1}^{\boldsymbol{T}}\boldsymbol{Xn} + d^{\mathbf{2}}\mathbf{1}^{\boldsymbol{T}}\mathbf{1}$$

First solve for d at optimum (set to 0)

$$\frac{\partial}{\partial d}\|\boldsymbol{Xn} - \mathbf{1}d\|_2^2 = 0 - 2\mathbf{1}^{\boldsymbol{T}}\boldsymbol{Xn} + 2dk$$

$$0 = -2\mathbf{1}^{\boldsymbol{T}}\boldsymbol{Xn} + 2dk \longrightarrow 0 = -\mathbf{1}^{\boldsymbol{T}}\boldsymbol{Xn} + dk$$

$$\longrightarrow d = \frac{1}{k}\mathbf{1}^{\boldsymbol{T}}\boldsymbol{Xn} = \boldsymbol{\mu n}$$

Solving Total Least Squares

$$\|\boldsymbol{X}\boldsymbol{n} - \mathbf{1}d\|_2^2 = \|\boldsymbol{X}\boldsymbol{n} - \mathbf{1}\boldsymbol{\mu}\boldsymbol{n}\|_2^2 \qquad d = \boldsymbol{\mu}\boldsymbol{n}$$

$$= \|(\boldsymbol{X} - \mathbf{1}\boldsymbol{\mu})\,\boldsymbol{n}\|_2^2$$

Objective is then:

$$\arg\min_{\|\boldsymbol{n}\|=1} \left\|(\boldsymbol{X} - \mathbf{1}\boldsymbol{\mu})\,\boldsymbol{n}\right\|_2^2$$

Recall: Homogenous Least Squares

$$\underset{\|\boldsymbol{v}\|_2^2=1}{\arg\min} \|\boldsymbol{A}\boldsymbol{v}\|_2^2 \longrightarrow$$ Eigenvector corresponding to smallest eigenvalue of A^TA

Why do we need $||v||^2 = 1$ or some other constraint?

Applying it in our case:

$$\boldsymbol{n} = \text{smallest\_eigenvec}((\boldsymbol{X} - \boldsymbol{1}\boldsymbol{\mu})^{\boldsymbol{T}}(\boldsymbol{X} - \boldsymbol{1}\boldsymbol{\mu}))$$

Note: technically homogeneous only refers to $||Av||=0$ but it's common shorthand in computer vision to refer to the specific problem of $||v||=1$

Connection to ML

Matrix we take the eigenvector of looks like:

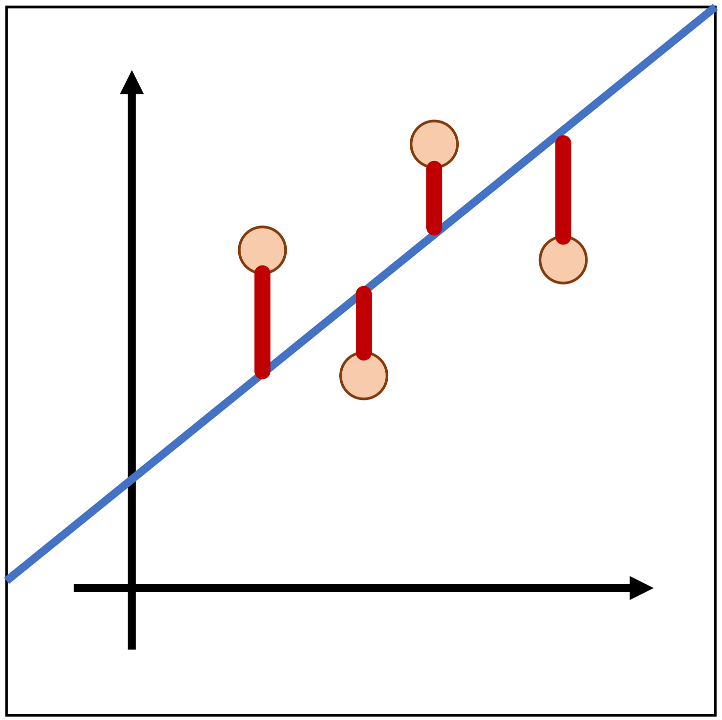
$$(\boldsymbol{X} - \mathbf{1}\boldsymbol{\mu})^T(\boldsymbol{X} - \mathbf{1}\boldsymbol{\mu}) = \begin{bmatrix} \sum_i (x_i - \mu_x)^2 & \sum_i (x_i - \mu_x)(y_i - \mu_y) \\ \sum_i (x_i - \mu_x)(y_i - \mu_y) & \sum_i (y_i - \mu_y)^2 \end{bmatrix}$$

This is a scatter matrix or scalar multiple of the covariance matrix. We're doing PCA, but taking the least principal component to get the normal.

Note: If you don't know PCA, just ignore this slide; it's to help build connections to people with a background in data science/ML.

Least Squares vs Total Least Squares

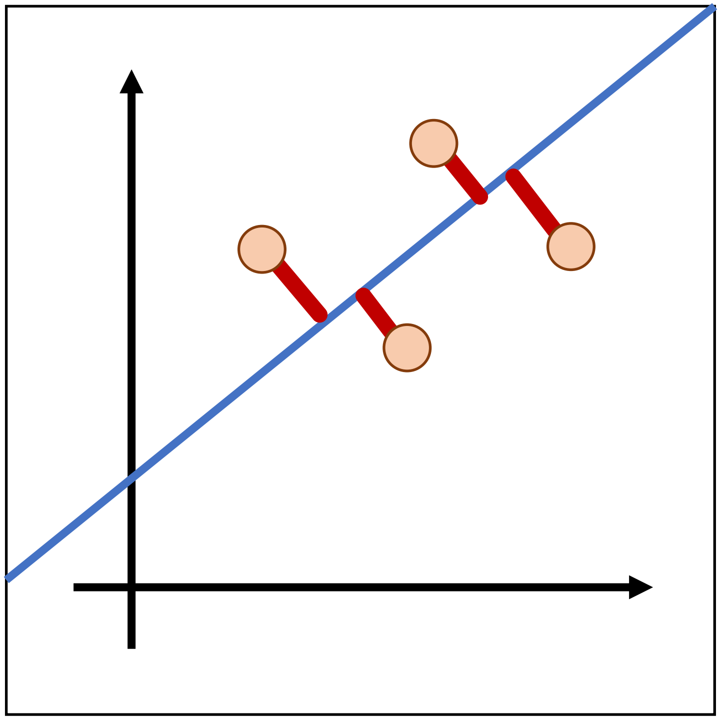
Least Squares:
Minimize error
of predictions



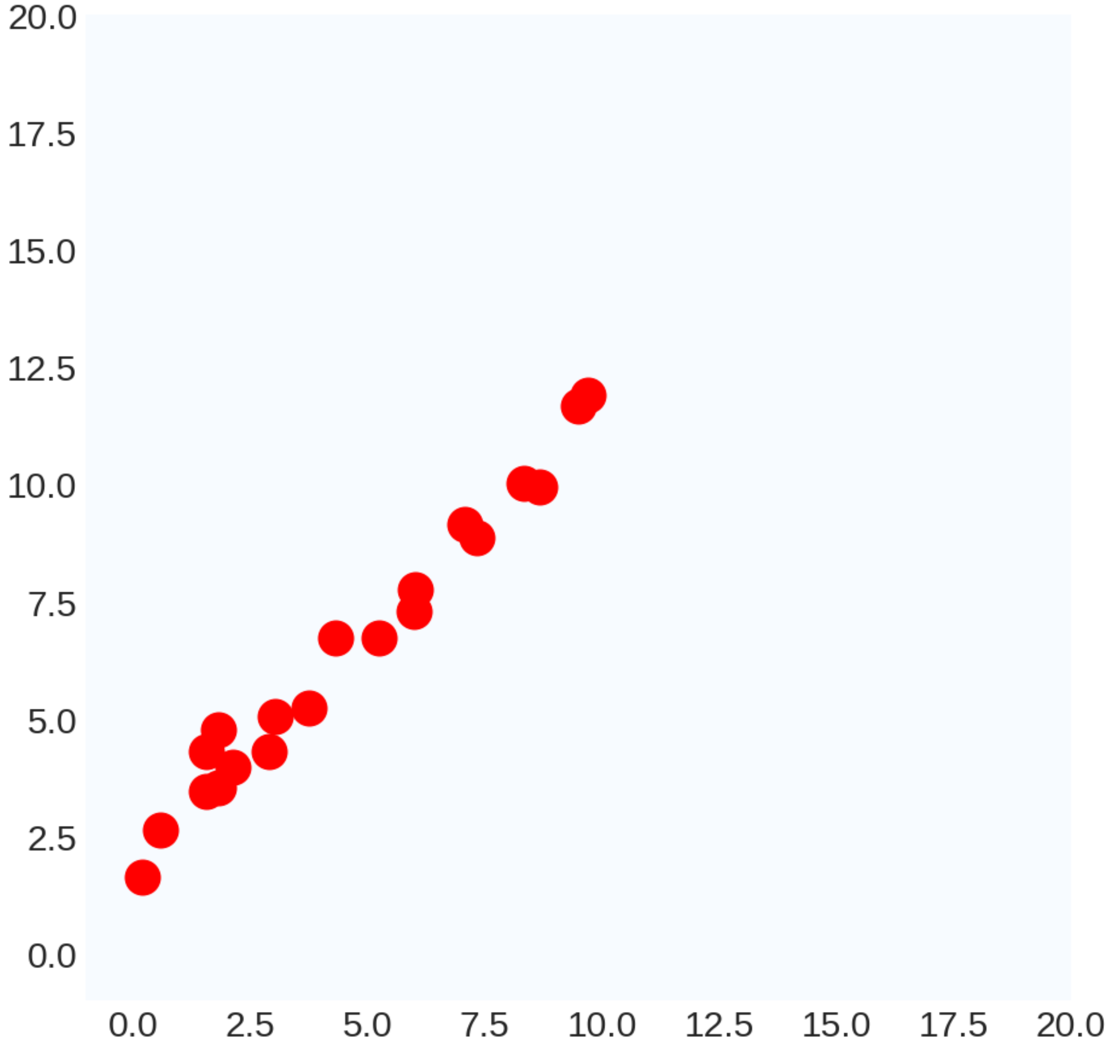
Predicted
Variables

Observed variables

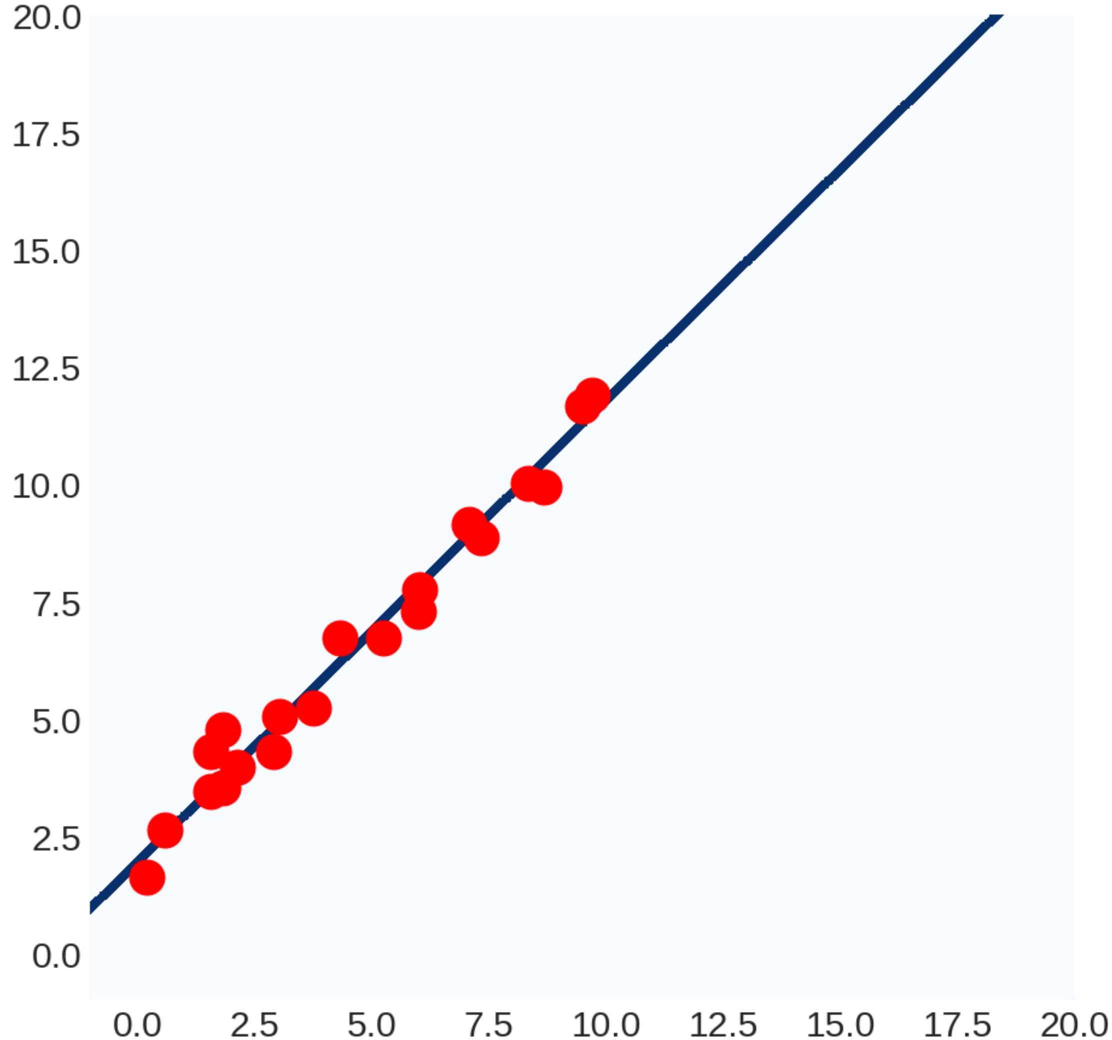
Total Least Squares
Find line that best
matches points in plane



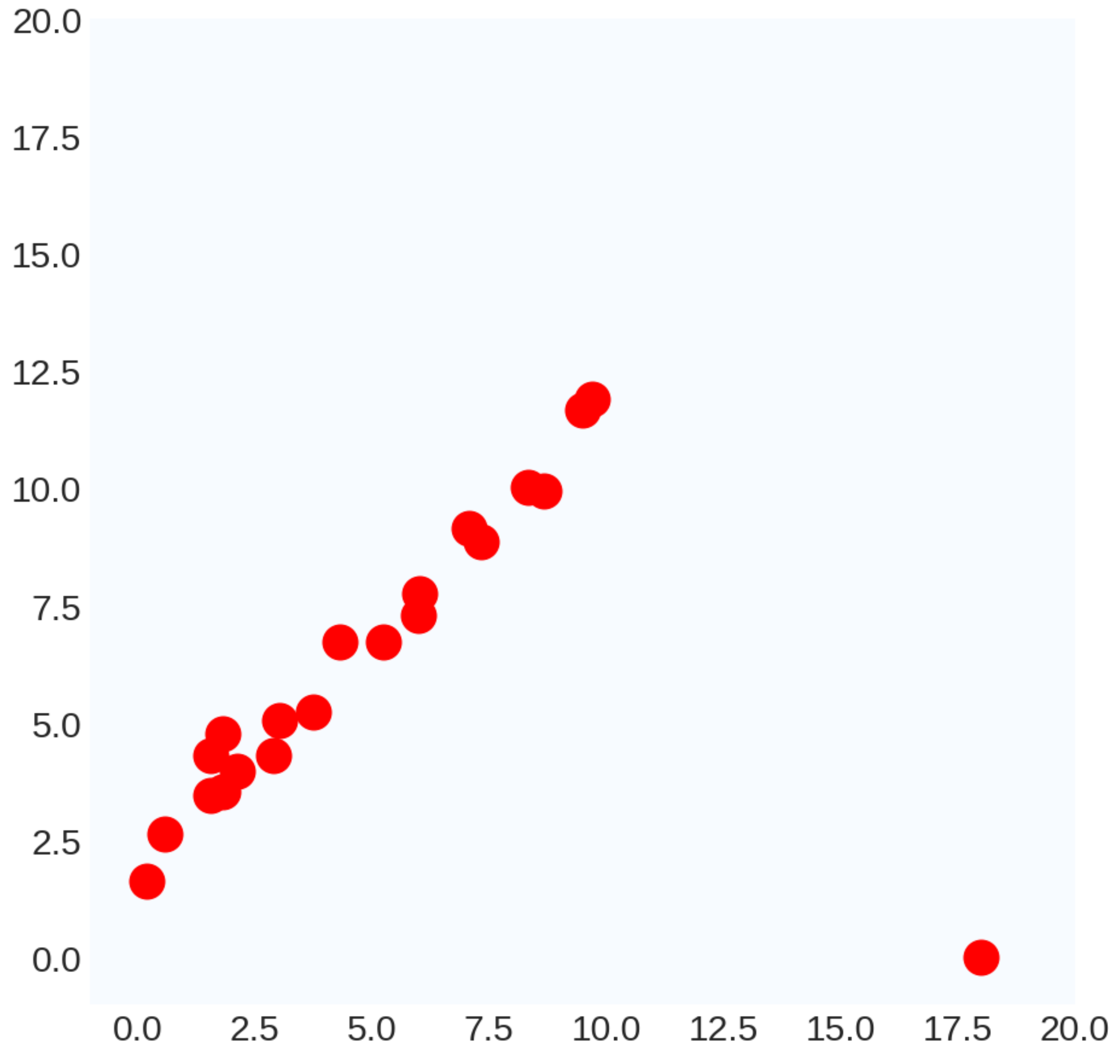
Running Least Squares



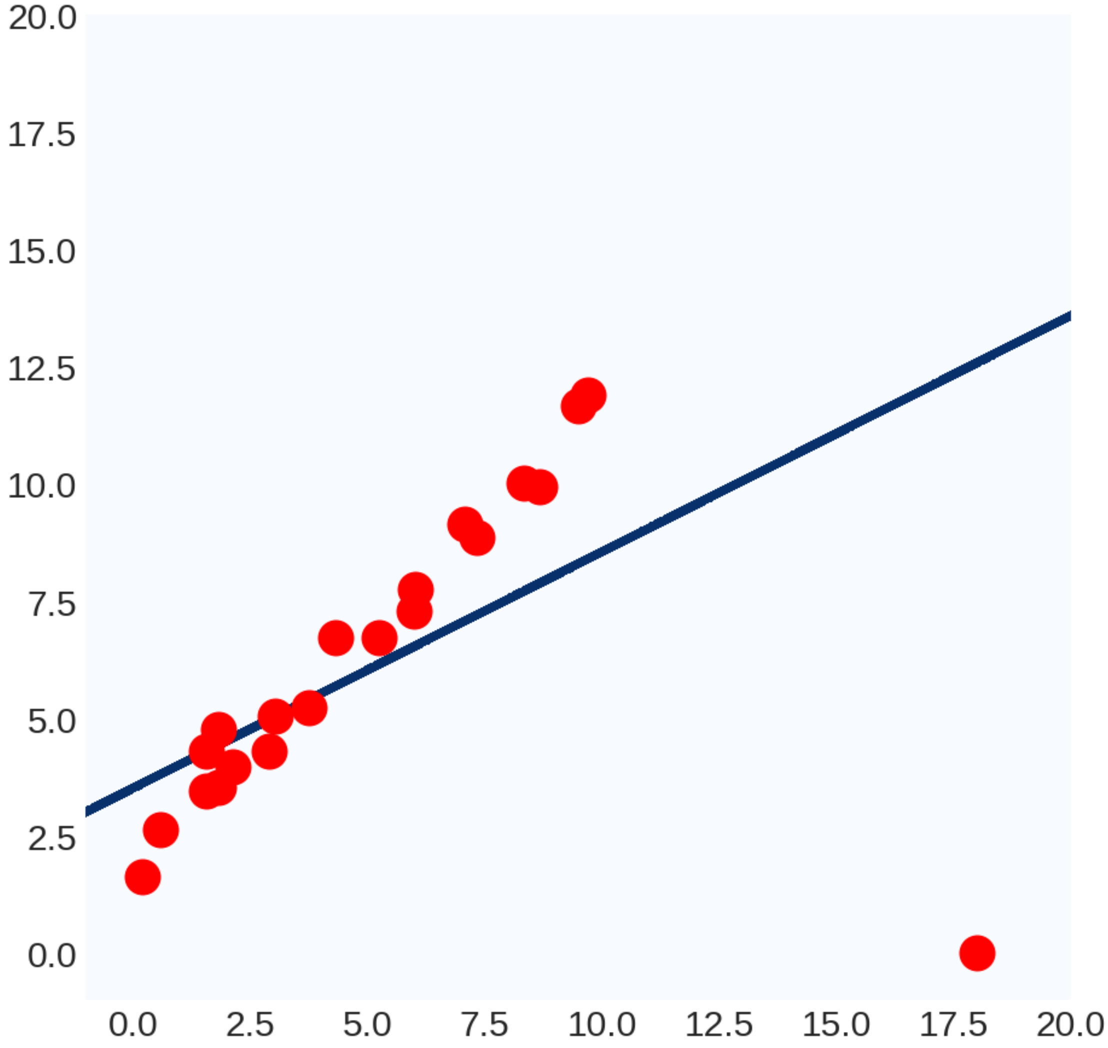
Running Least Squares



Running Least Squares



Running Least Squares



Running Least Squares

Way to think of it #1:

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

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

Way to think of it #2:

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

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

Common Fixes

Replace Least-Squares objective

Let $\boldsymbol{E} = \boldsymbol{Y} - \boldsymbol{X}\boldsymbol{W}$

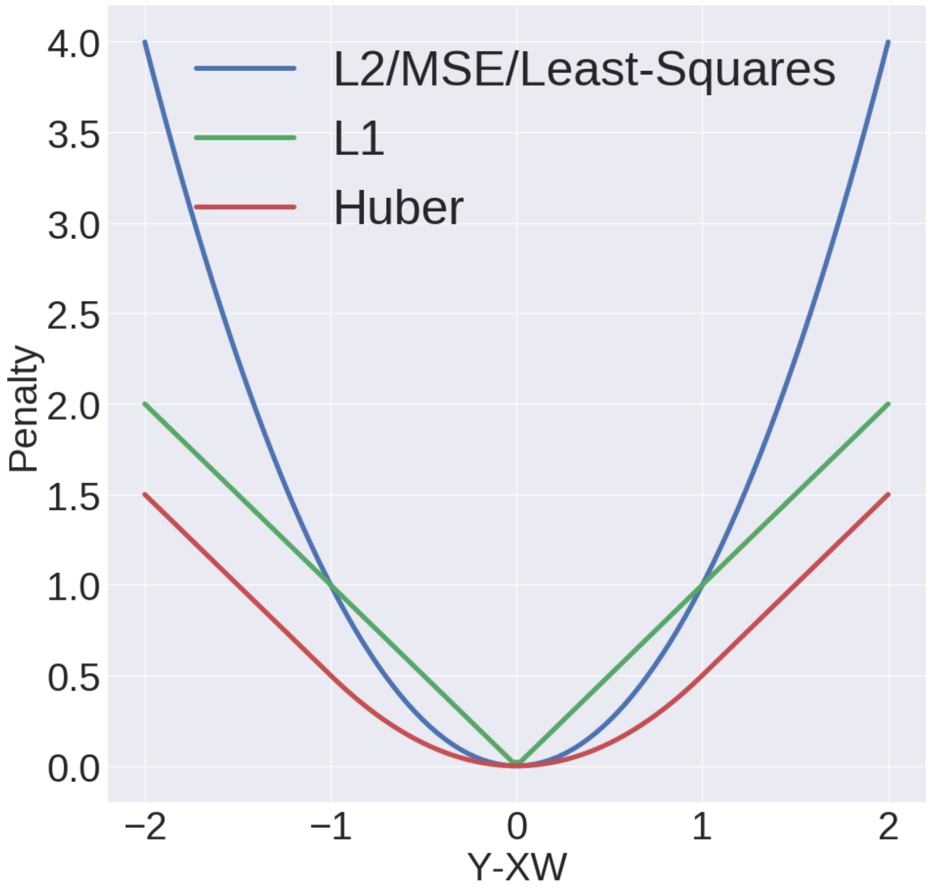
LS/L2/MSE: $\boldsymbol{E}_i^2$

L1: $|\boldsymbol{E}_i|$

Huber:

$|\boldsymbol{E}_i| \leq \delta$: $\frac{1}{2}\boldsymbol{E}_i^2$

$|\boldsymbol{E}_i| > \delta$: $\delta\left(|\boldsymbol{E}_i| - \frac{\delta}{2}\right)$

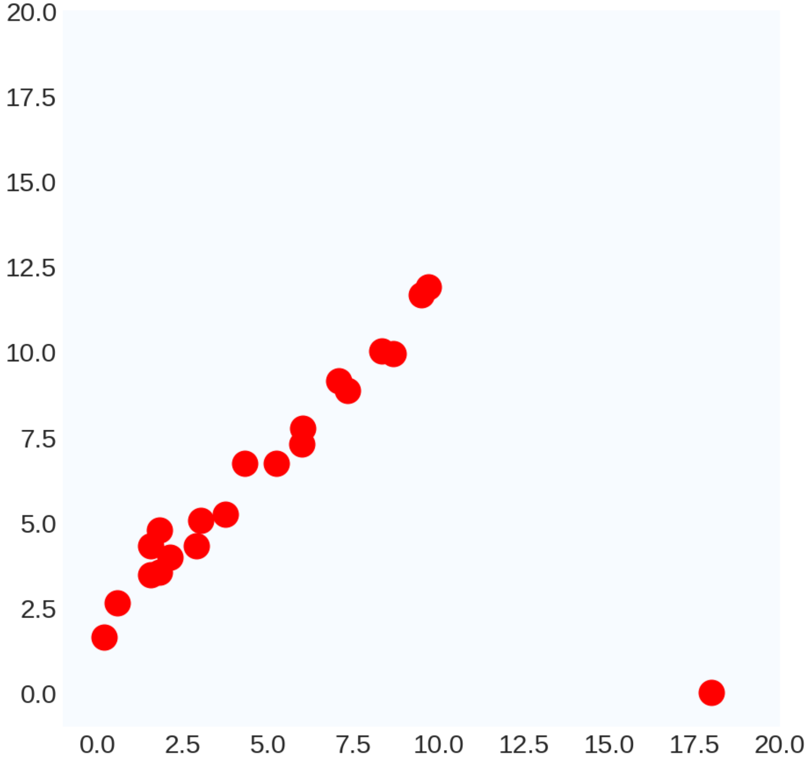


Issues with Common Fixes

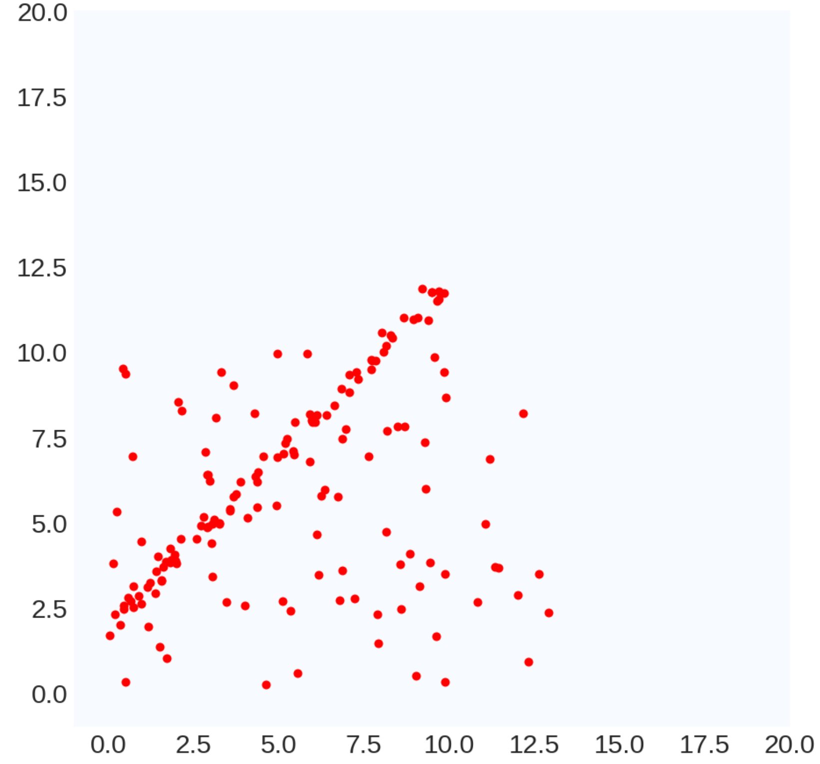
- Usually complicated to optimize:
  - Often no closed form solution
  - Typically not something you could write yourself
  - Sometimes not convex (local optimum is not necessarily a global optimum)
- Not simple to extend more complex objectives to things like total-least squares
- Typically don’t handle a ton of outliers (e.g., 80% outliers)

Outliers in Computer Vision

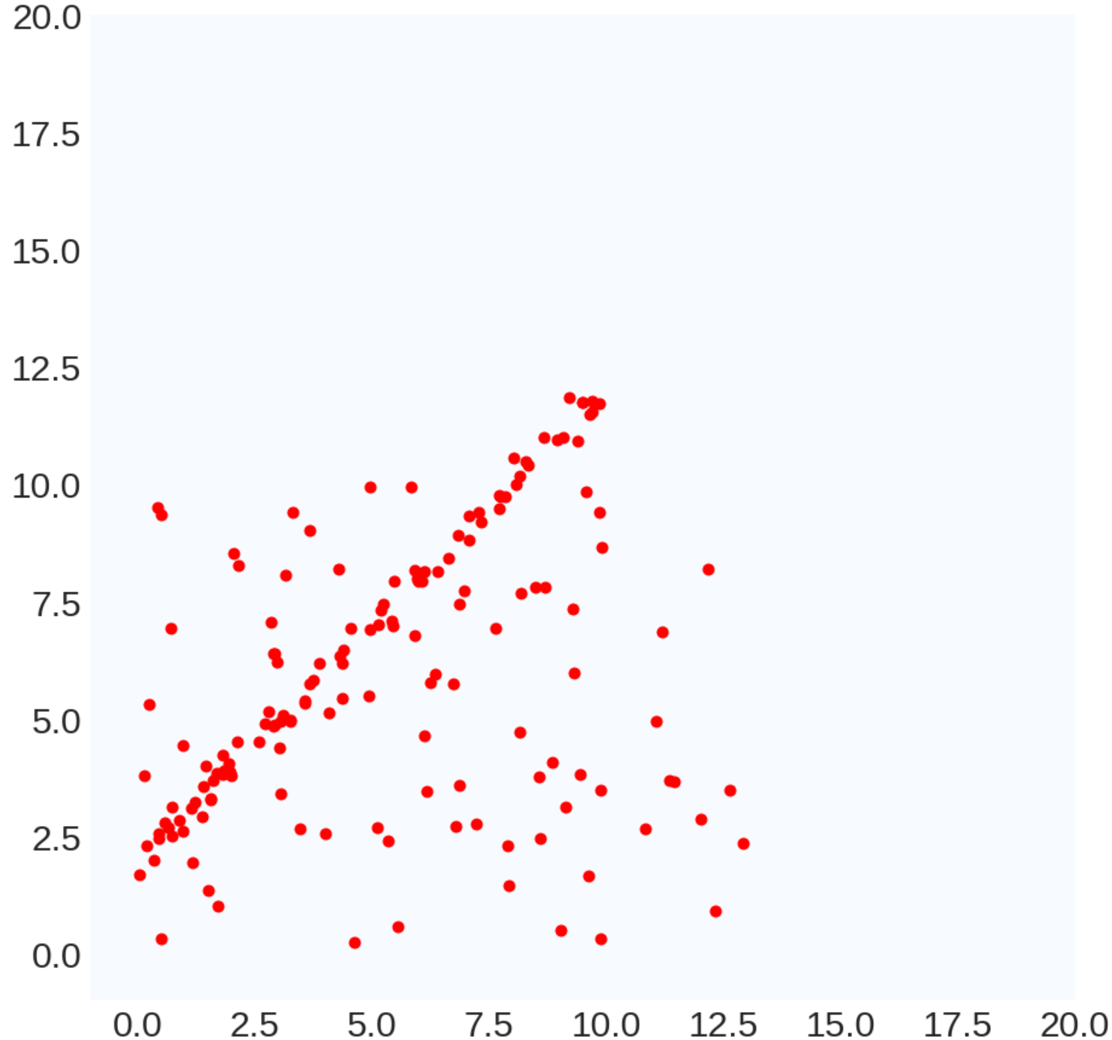
Single outlier:
rare



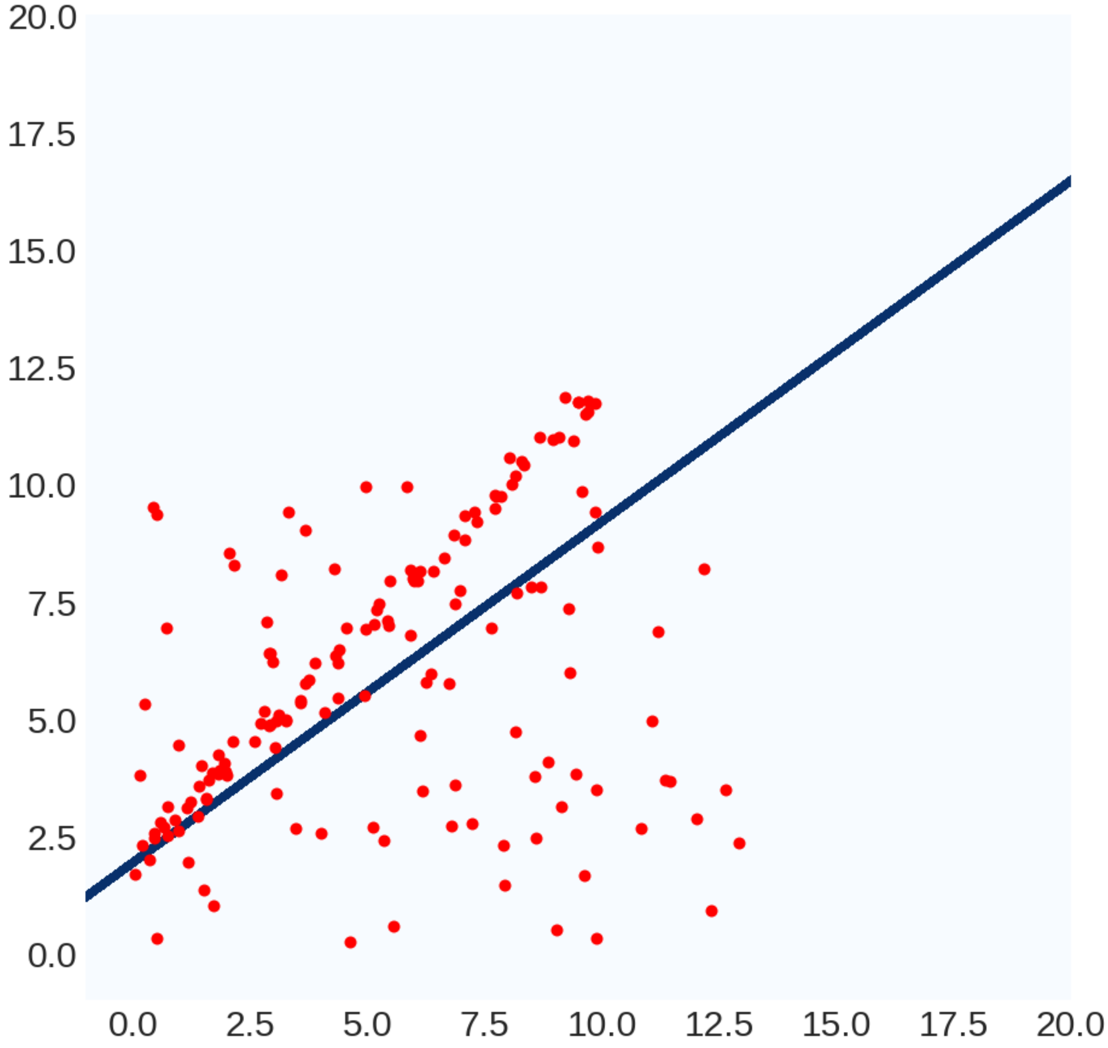
Many outliers:
common



Running Least Squares



Running Least Squares



A Simple but Clever Idea

- *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)

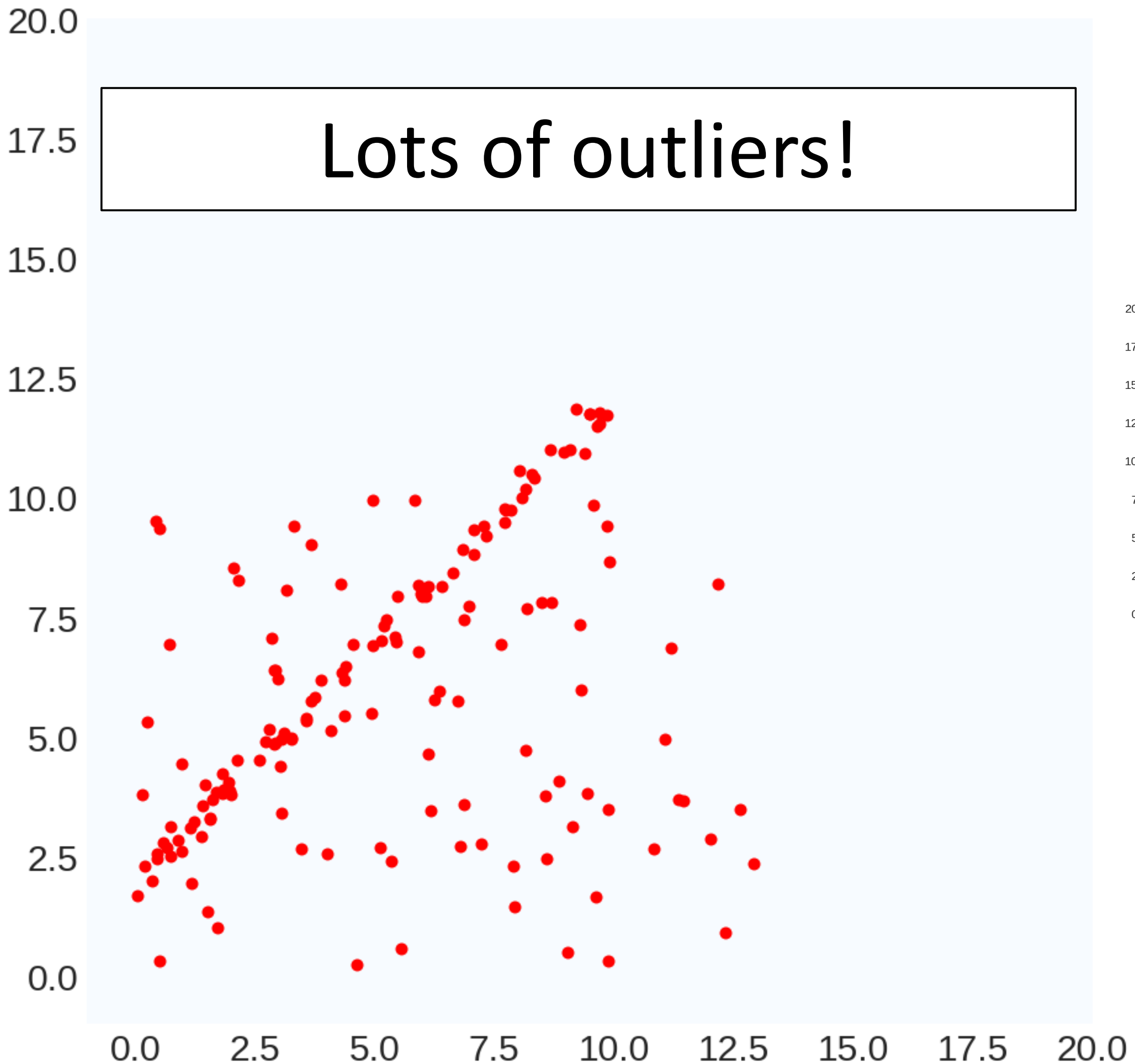
M. A. Fischler, R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. Comm. of the ACM, Vol 24, pp 381-395, 1981.

RANSAC for Lines

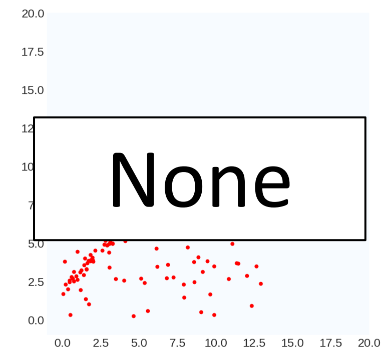
```
bestLine, bestCount = None, -1
for trial in range(numTrials):
        subset = pickPairOfPoints(data)
        line = totalLeastSquares(subset)
        E = linePointDistance(data,line)
        inliers = E < threshold
        if #inliers > bestCount:
                bestLine, bestCount = line, #inliers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

Running RANSAC

Trial #1



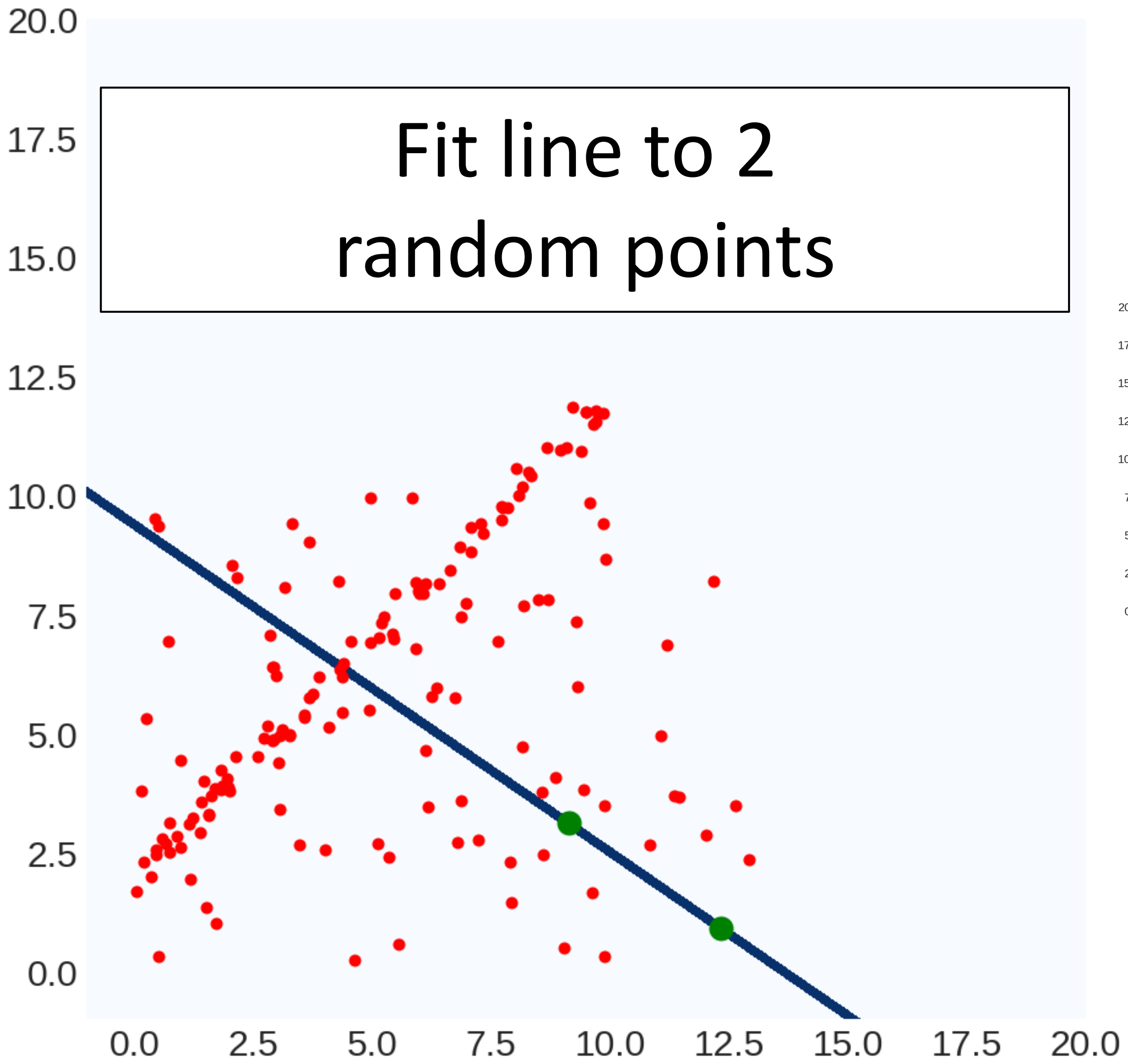
Best Model:



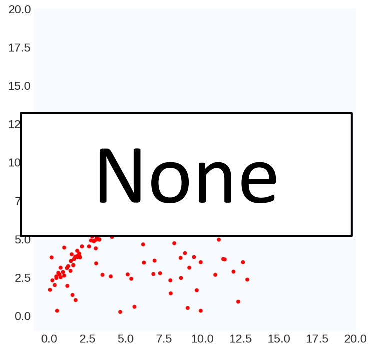
Best Count: -1

Running RANSAC

Trial #1



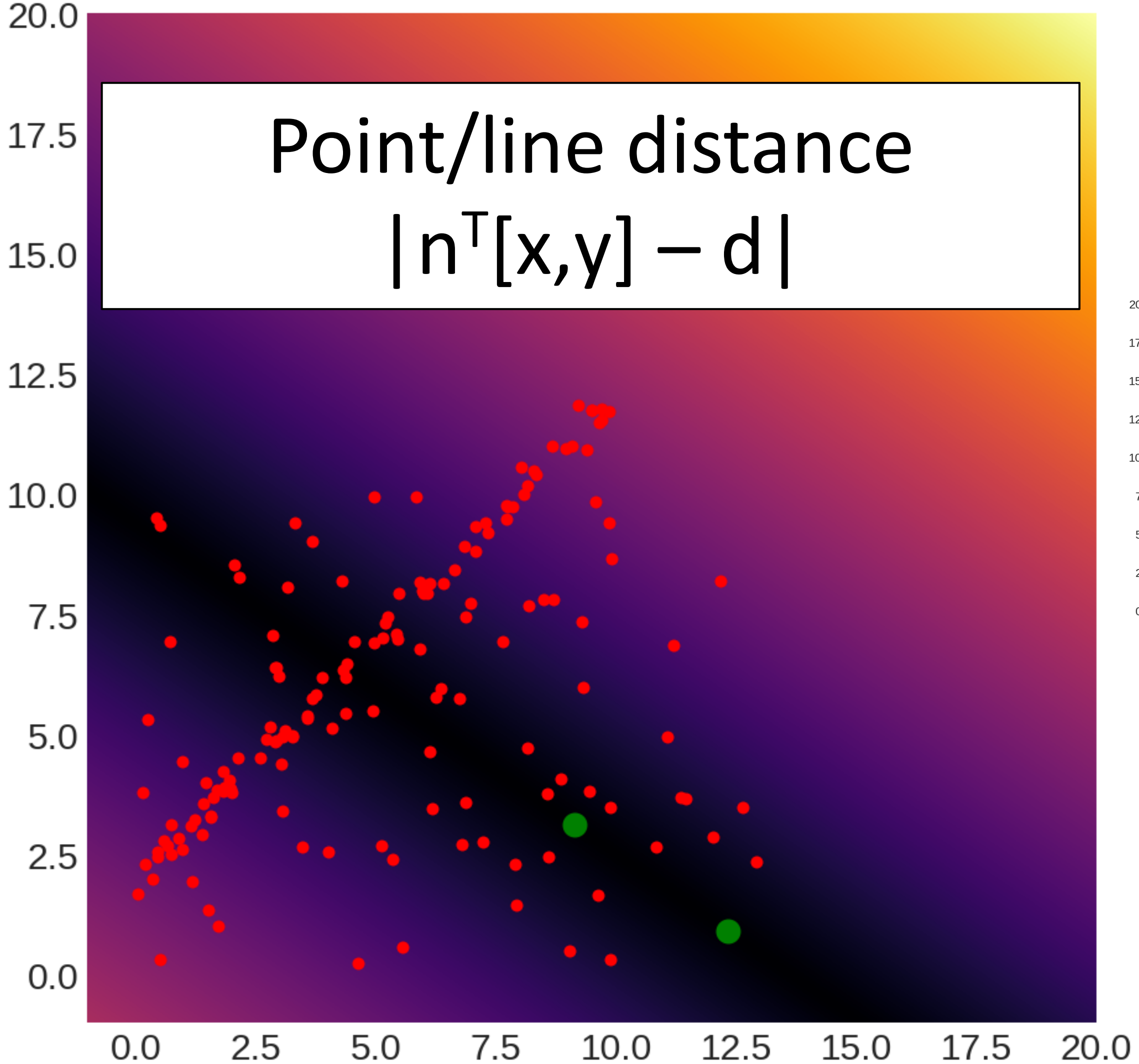
Best Model:



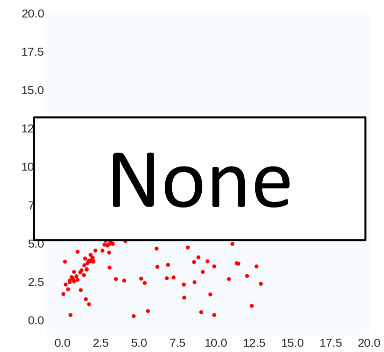
Best Count: -1

Running RANSAC

Trial #1



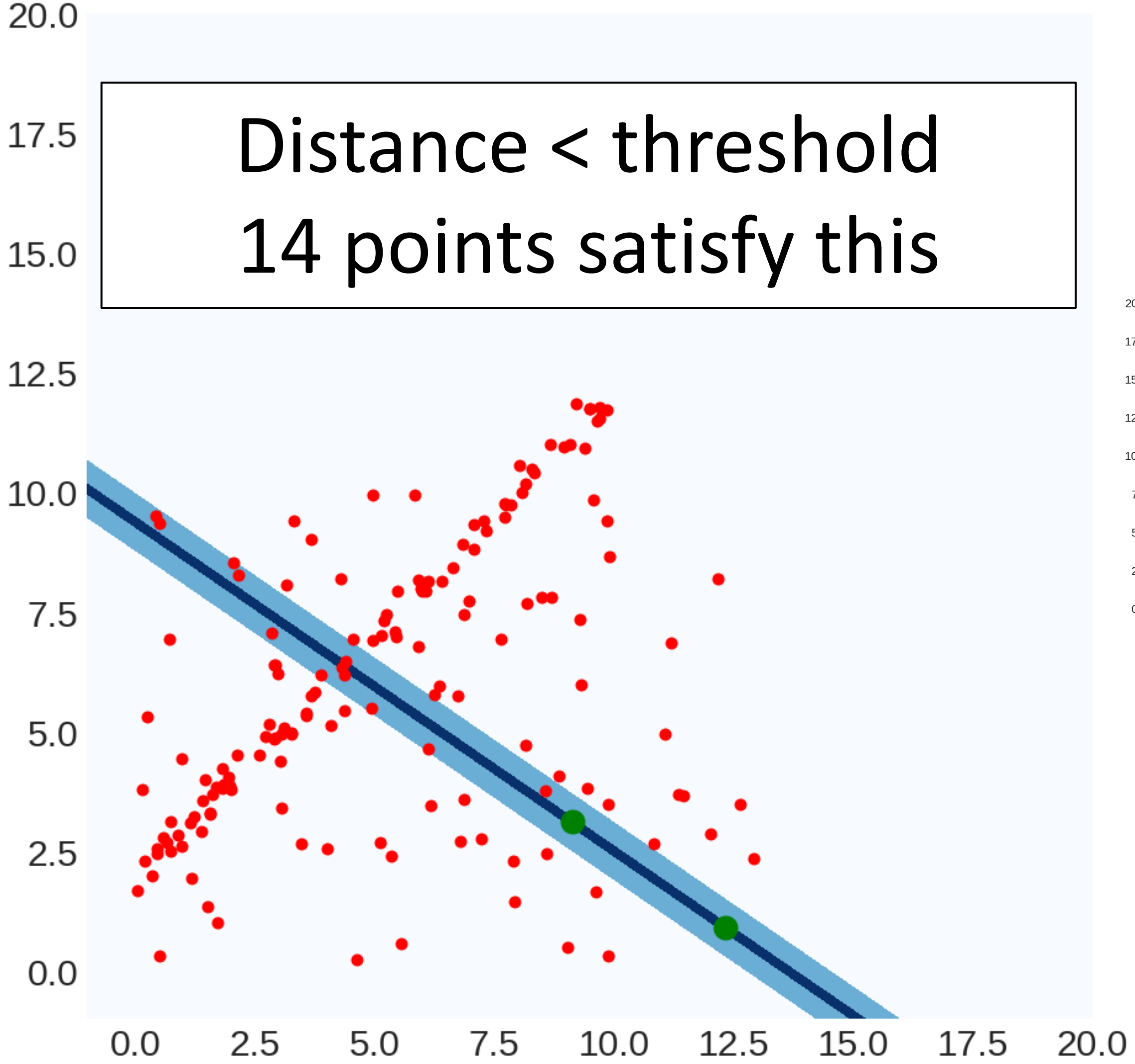
Best Model:



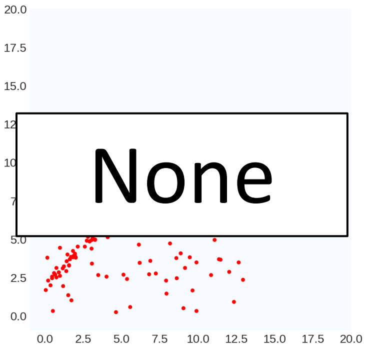
Best Count: -1

Running RANSAC

Trial #1



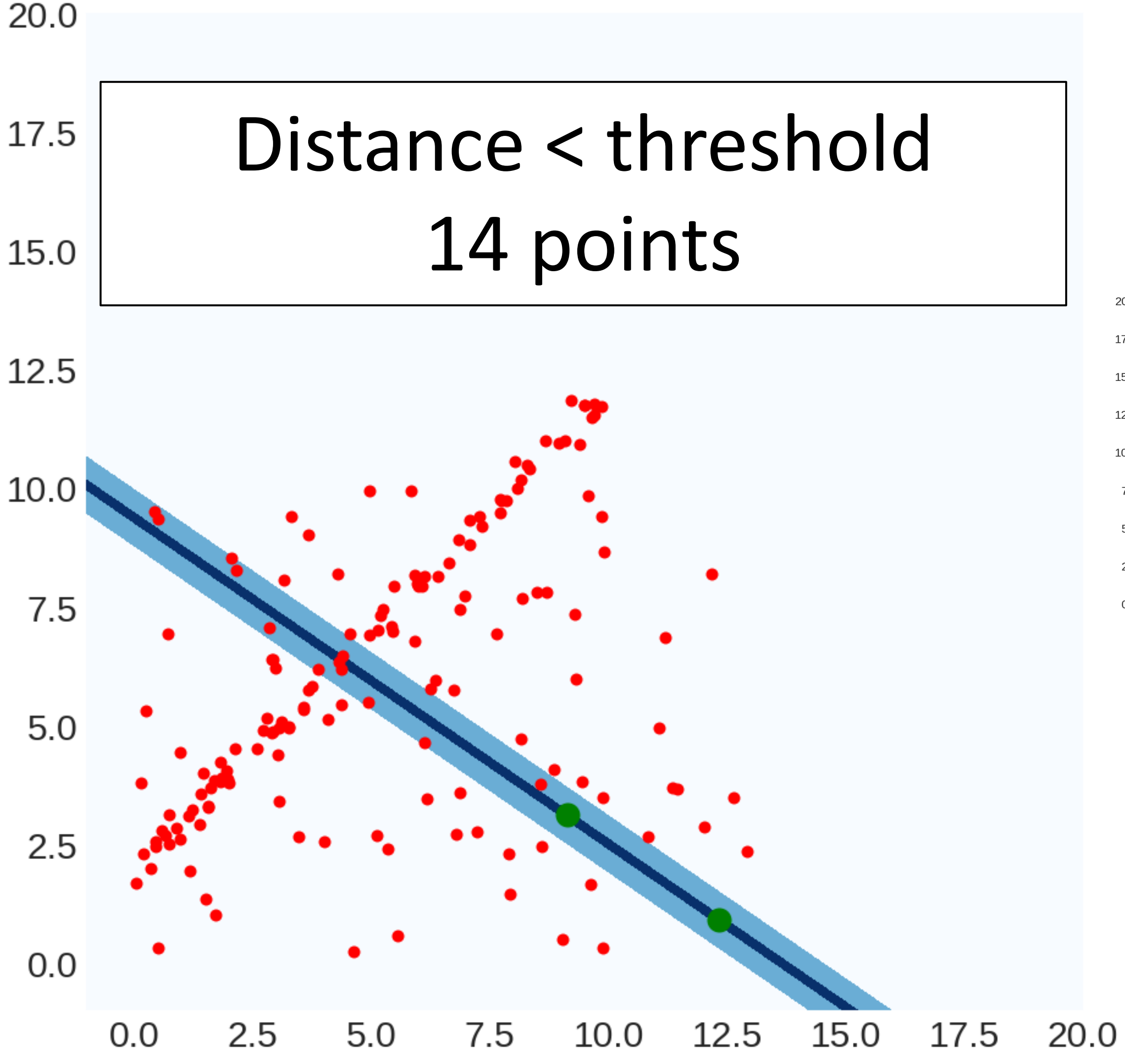
Best Model:



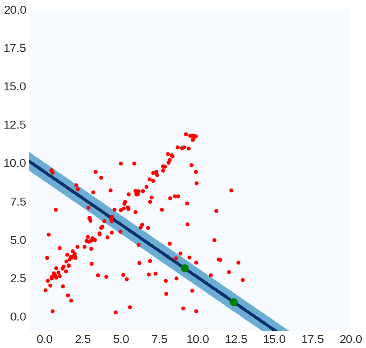
Best Count: -1

Running RANSAC

Trial
#1



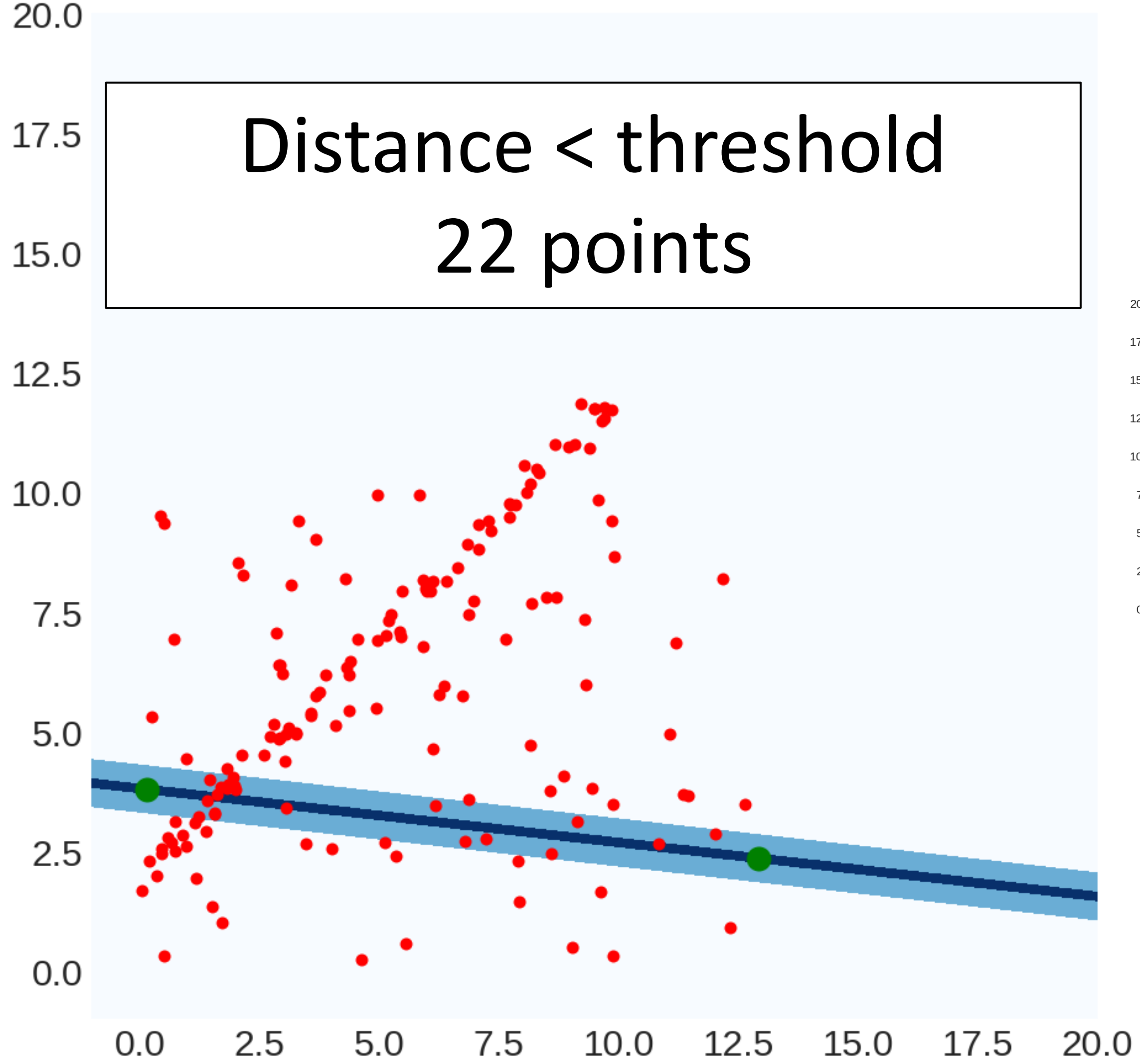
Best
Model:



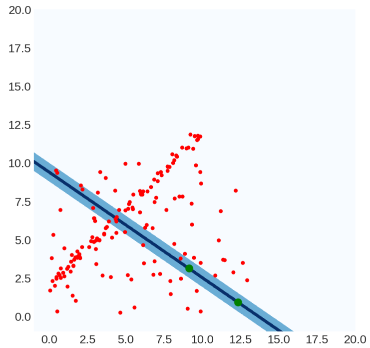
Best
Count:
14

Running RANSAC

Trial #2



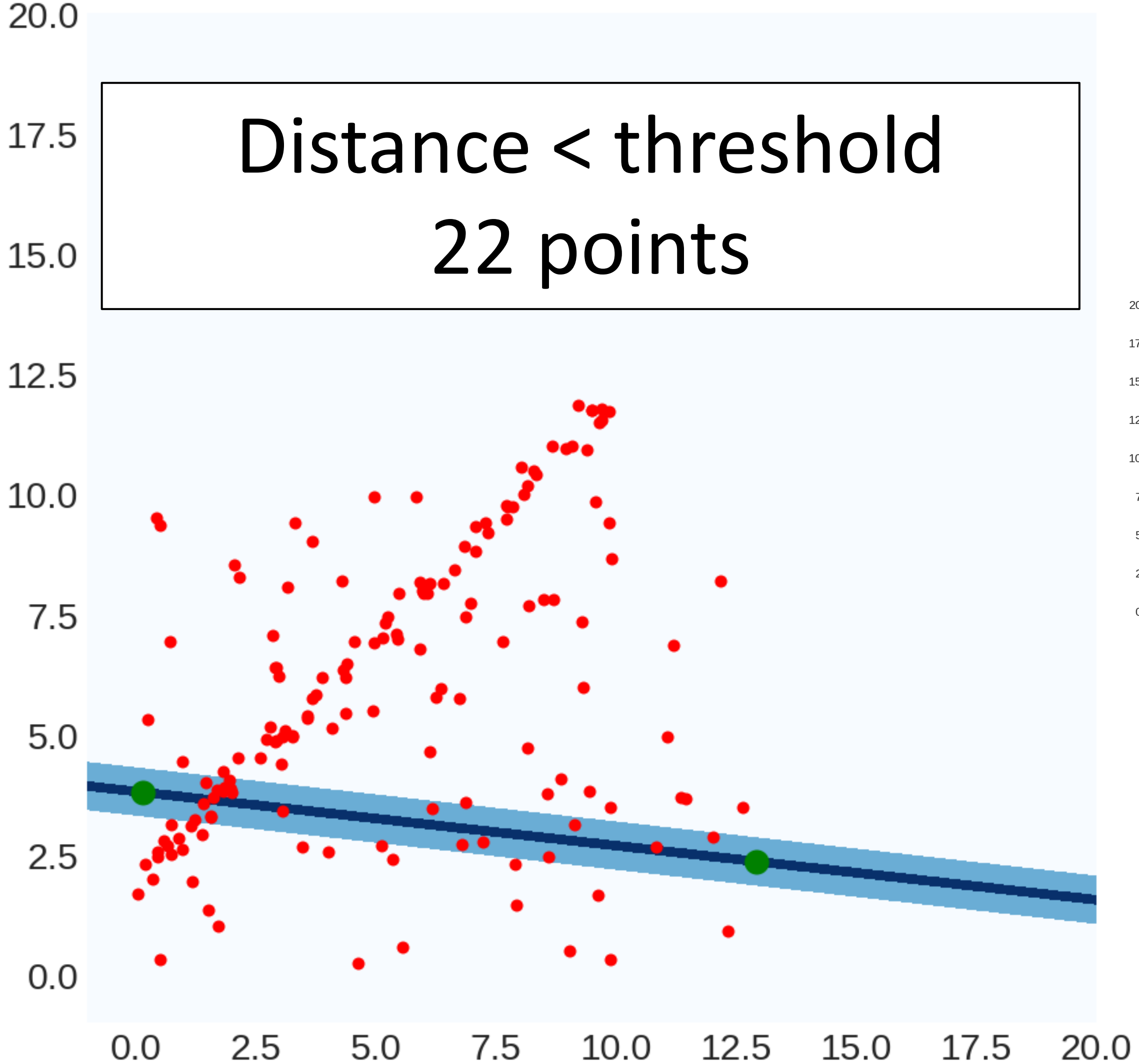
Best Model:



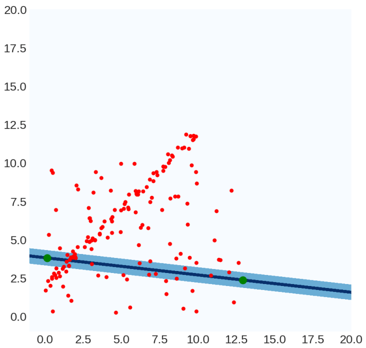
Best Count: 14

Running RANSAC

Trial #2



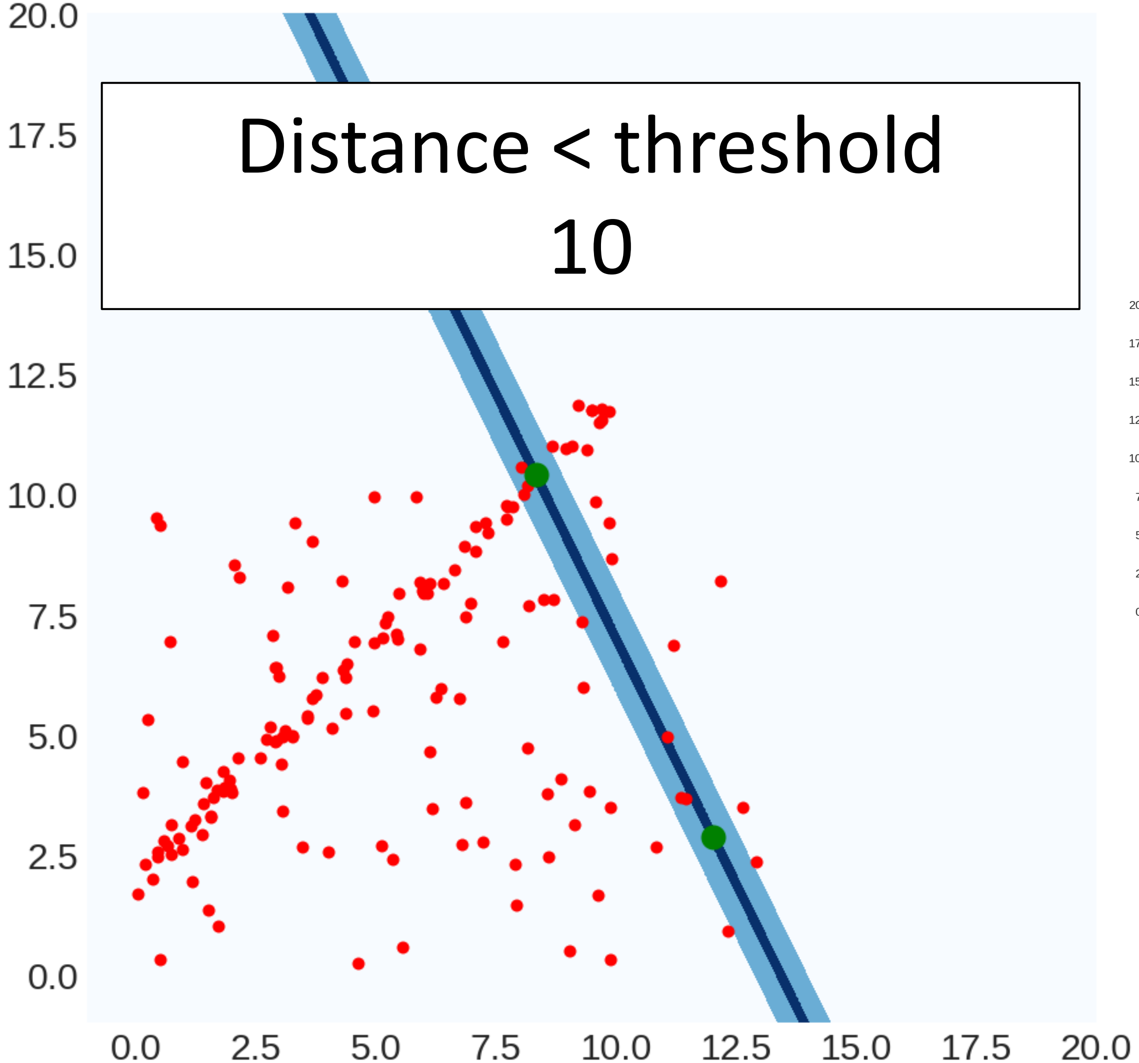
Best Model:



Best Count: 22

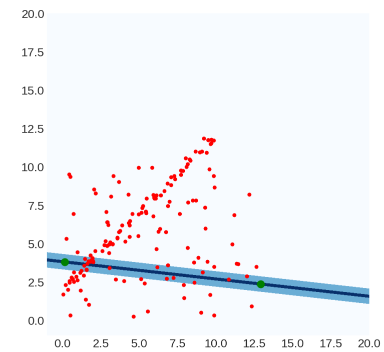
Running RANSAC

Trial #3



Distance < threshold
10

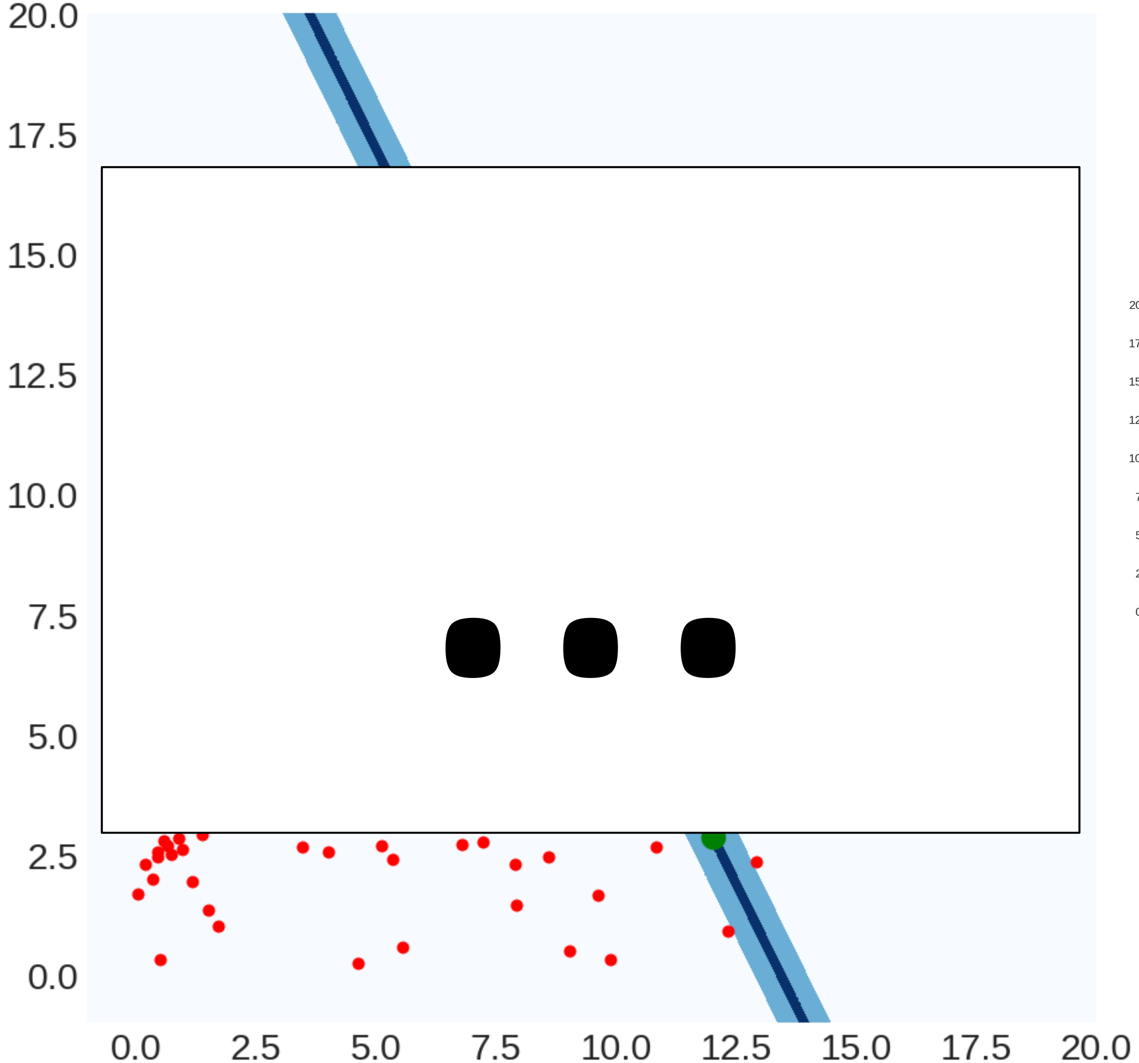
Best Model:



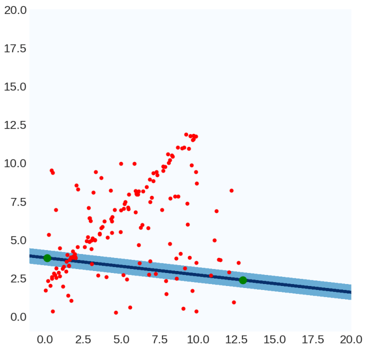
Best Count: 22

Running RANSAC

Trial #3



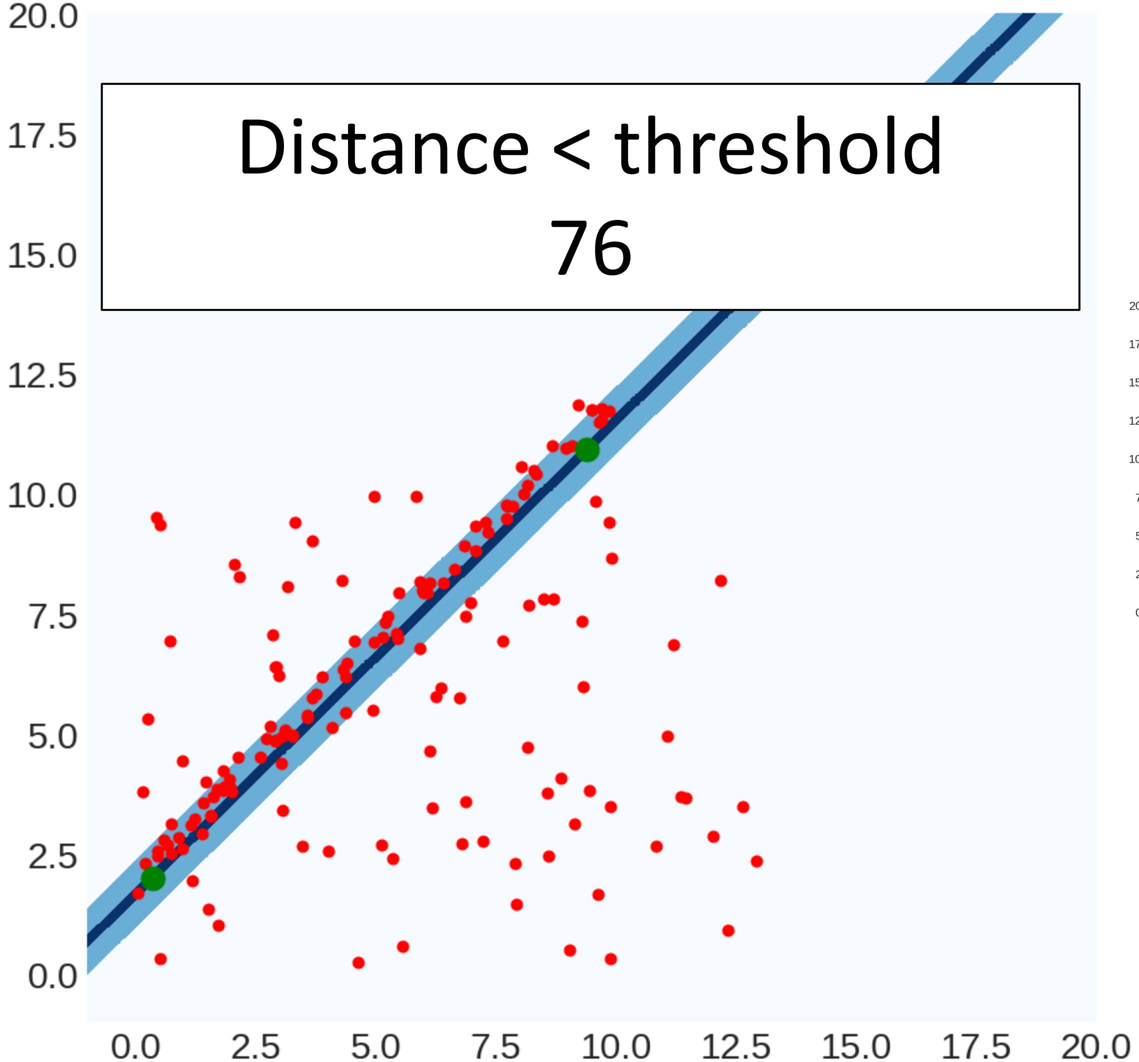
Best Model:



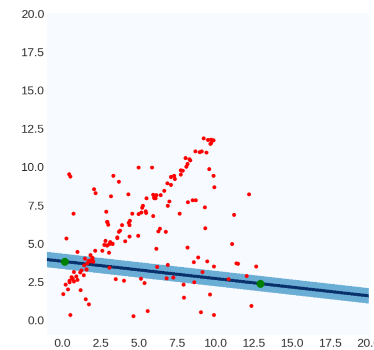
Best Count: 22

Running RANSAC

Trial #9



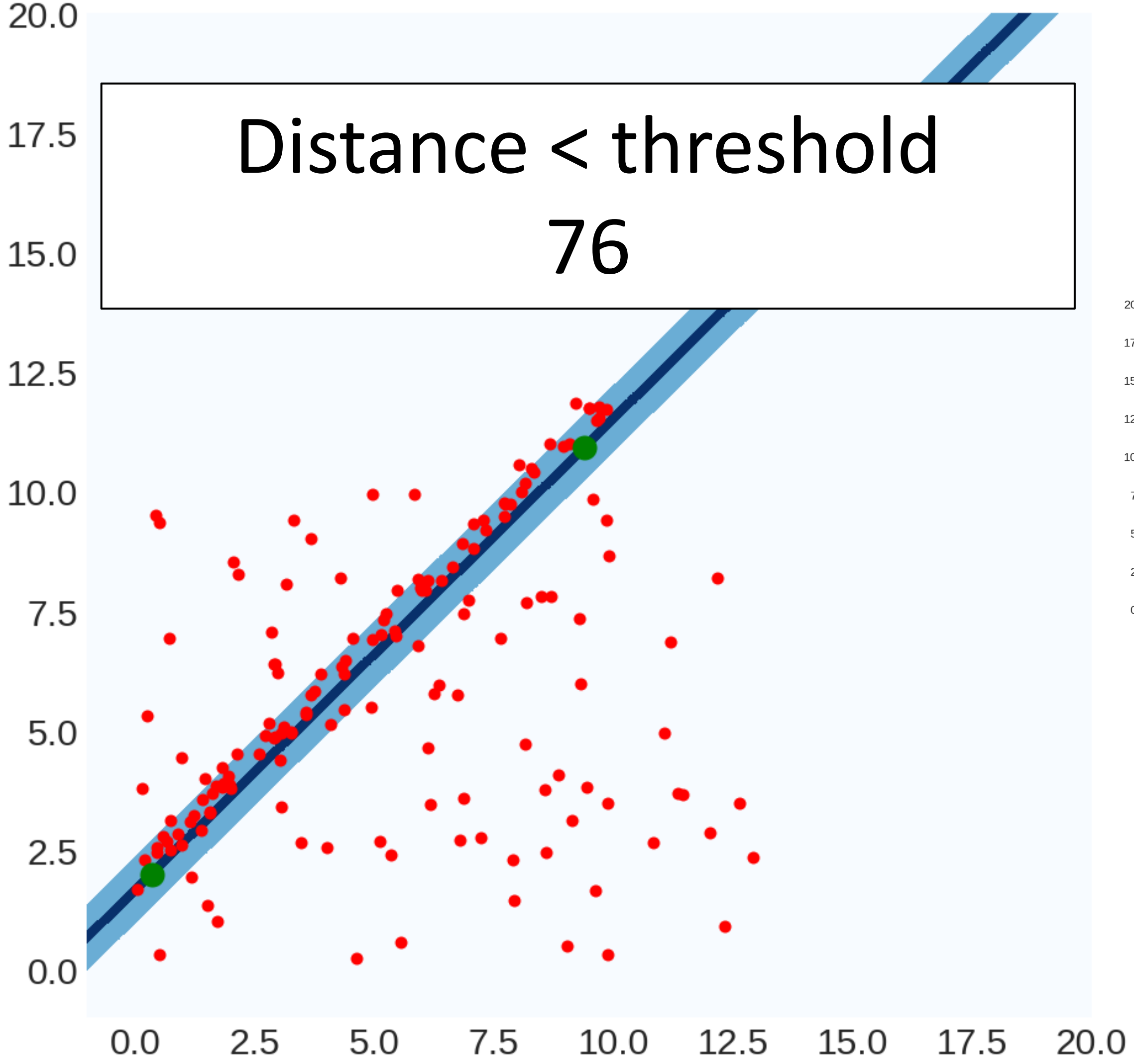
Best Model:



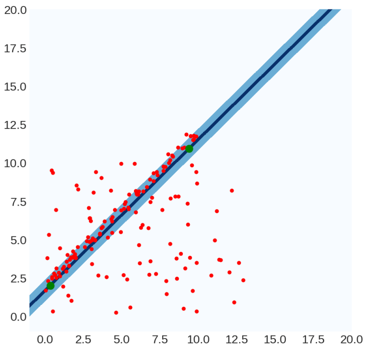
Best Count: 22

Running RANSAC

Trial #9



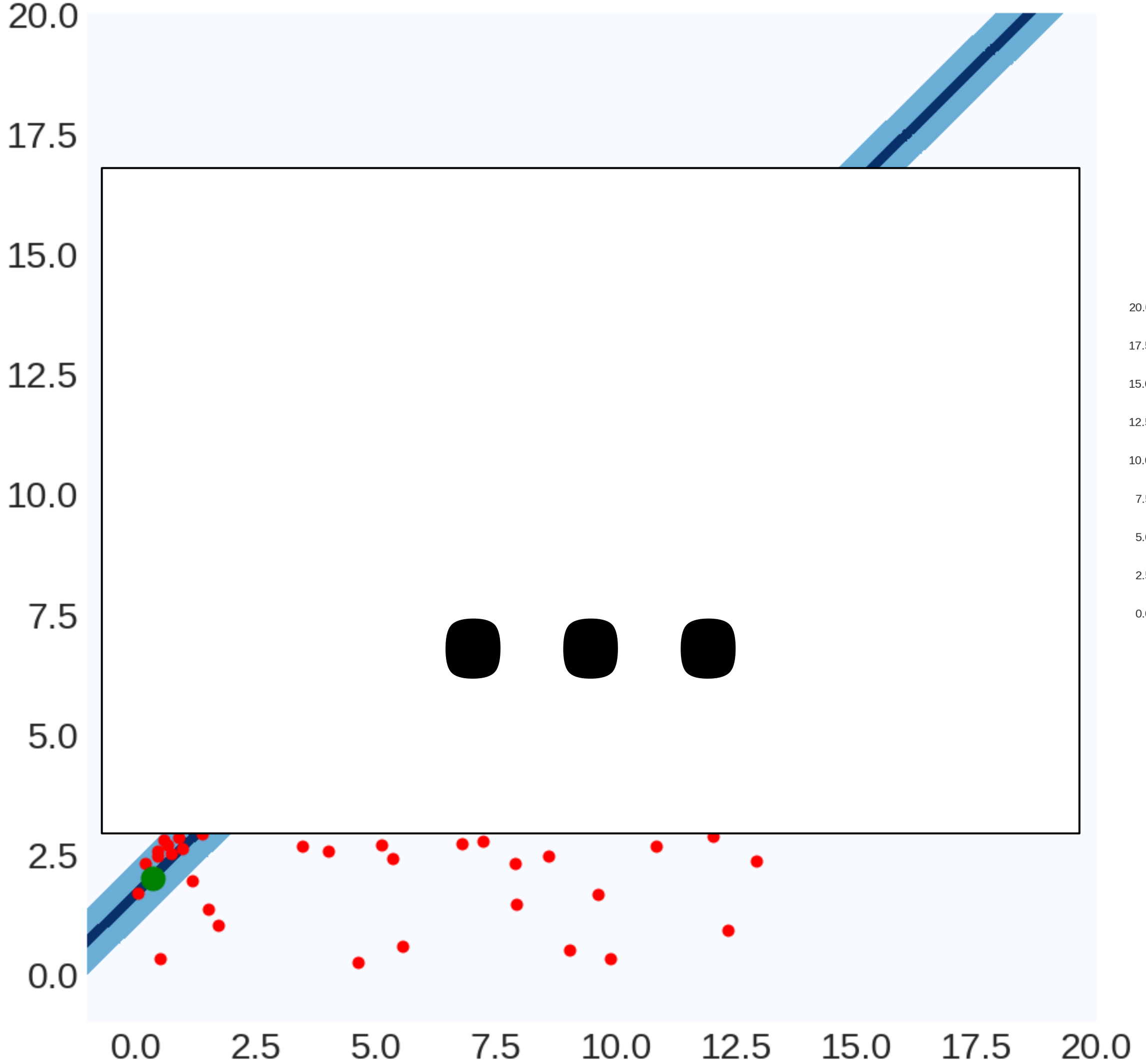
Best Model:



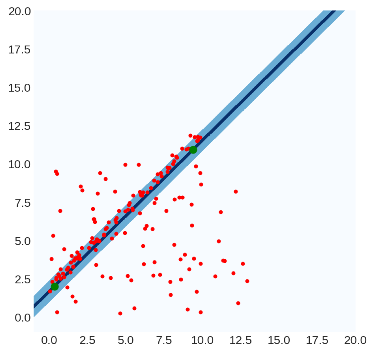
Best Count: 76

Running RANSAC

Trial #9



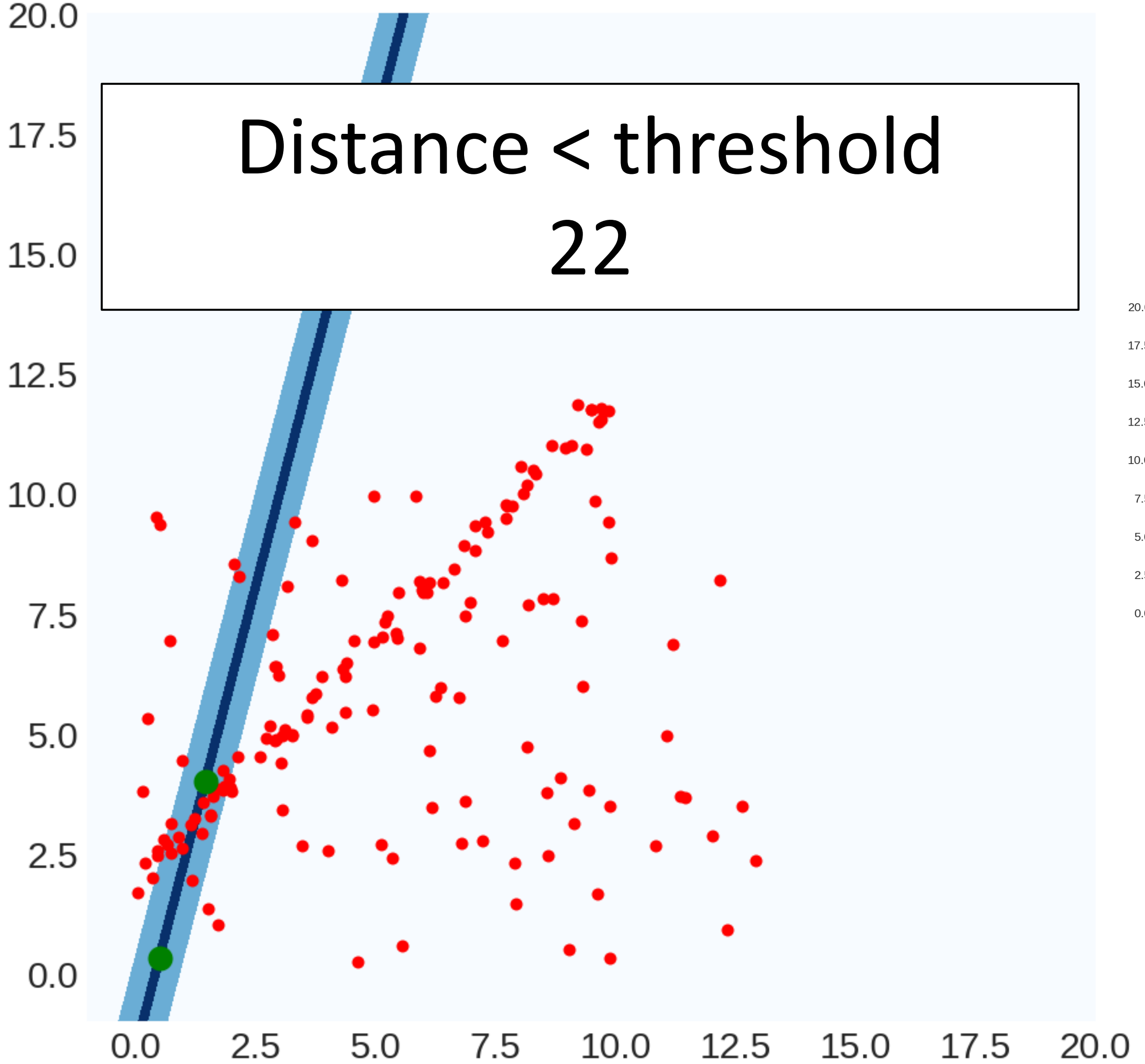
Best Model:



Best Count: 76

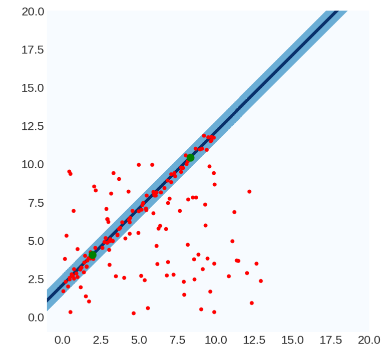
Running RANSAC

Trial #100



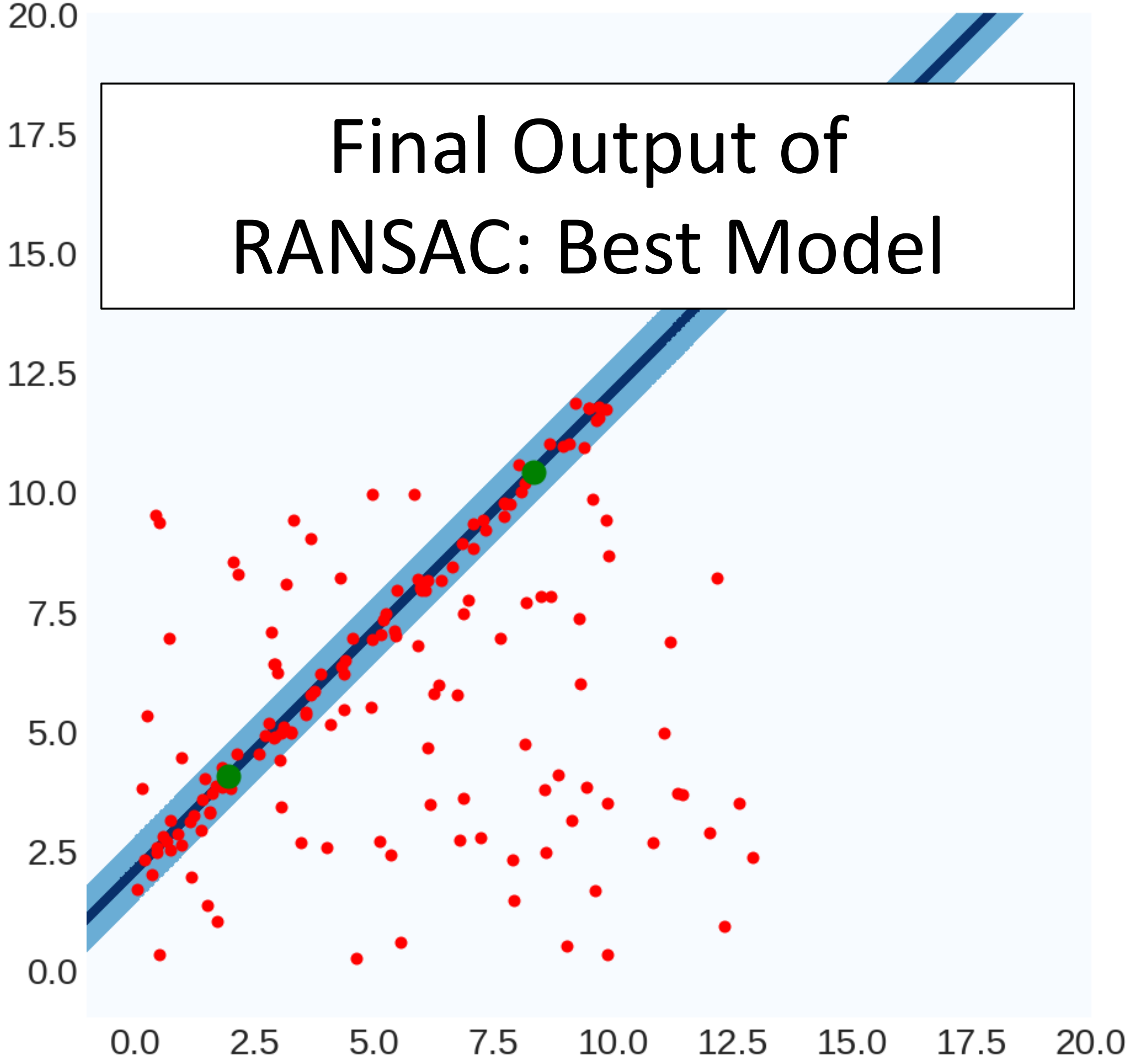
Distance < threshold
22

Best Model:



Best Count: 85

Running RANSAC



RANSAC in General

```
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)
```

(often refit on the inliers for best model)

RANSAC: How Many Trials?

Suppose that:

r: Fraction of outliers (e.g. 80%)

s: Number of points we pick per set (e.g. 2)

N: Number of times we run RANSAC (e.g. N=500)

What's the probability of picking a sample set with no outliers?

$$\approx (1 - r)^s$$ **(4%)**

What's the probability of picking a sample set with some outliers?

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

RANSAC: How Many Trials?

Suppose that:

r: Fraction of outliers (e.g. 80%)

s: Number of points we pick per set (e.g. 2)

N: Number of times we run RANSAC (e.g. N=500)

What's the probability of picking a sample set with some outliers?

$$1-(1-r)^s \qquad \textbf{(96\%)}$$

What's the probability of picking only sample sets some outliers?

$$(1-(1-r)^s)^N \qquad \textbf{(}\mathbf{10^{-7}}\textbf{\% N=500)}$$

(13% N=50)

What's the probability of picking any set with no outliers?

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

RANSAC: How Many Trials

RANSAC fails to fit a line with 80% outliers after trying only 500 times

P(Failure):
1 / 731,784,961

P($157M Jackpot):
1 / 302,575,350

Death by vending machine

P(Death):
≈1 / 112,000,000

Odds/Jackpot amount from 2/7/2019 megamillions.com, unfortunate demise odds from livescience.com

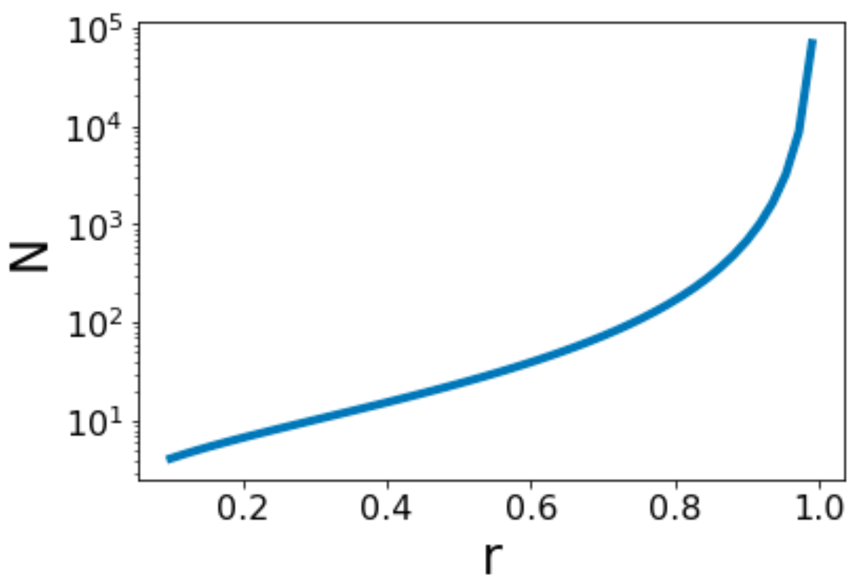
RANSAC: How Many Trials

Suppose that:
r: Fraction of outliers (e.g. 80%)
s: Number of points we pick per set (e.g. 2)
N: Number of times we run RANSAC (e.g. N=500)
C: Chance that we find a set with no outliers (e.g. 99.9%)

What's the probability of picking any set with no outliers?

$$C \geq 1 - (1 - (1 - r)^s)^N$$

$$N \geq \frac{\log(1 - T)}{\log(1 - (1 - r)^s)}$$



RANSAC: How Many Trials

Suppose that:

r: Fraction of outliers (e.g. 80%)

s: Number of points we pick per set (e.g. 2)

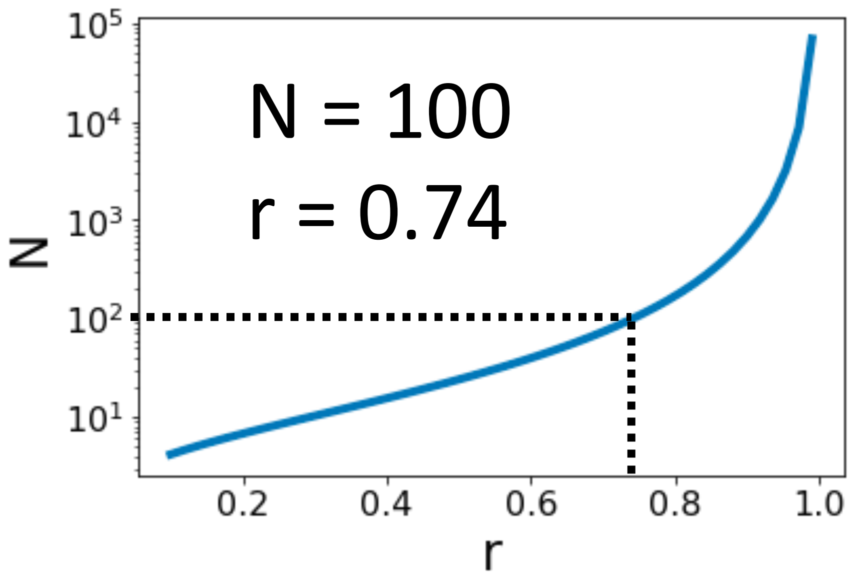
N: Number of times we run RANSAC (e.g. N=500)

C: Chance that we find a set with no outliers (e.g. 99.9%)

What's the probability of picking any set with no outliers?

$$C \geq 1 - (1 - (1 - r)^s)^N$$

$$N \geq \frac{\log(1 - T)}{\log(1 - (1 - r)^s)}$$



RANSAC: How Many Trials

Suppose that:

r: Fraction of outliers (e.g. 80%)

s: Number of points we pick per set (e.g. 2)

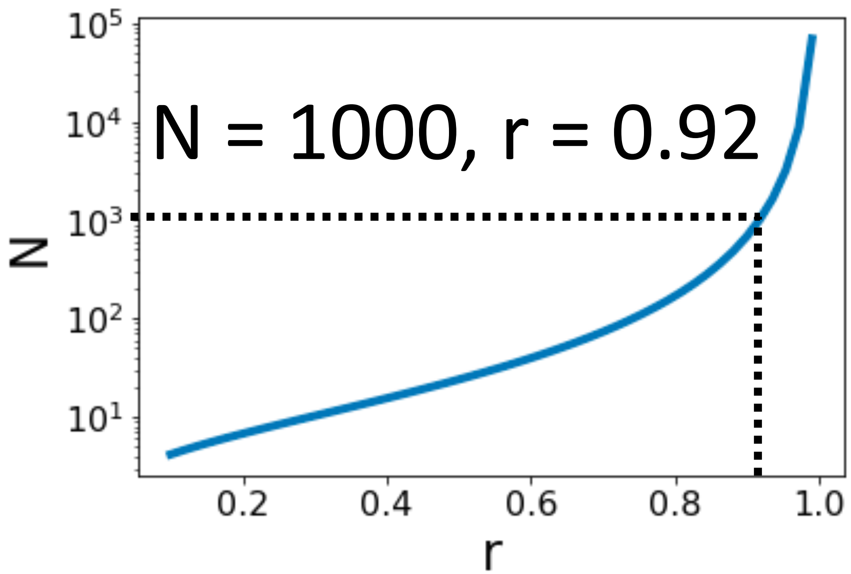
N: Number of times we run RANSAC (e.g. N=500)

C: Chance that we find a set with no outliers (e.g. 99.9%)

What's the probability of picking any set with no outliers?

$$C \geq 1 - (1 - (1 - r)^s)^N$$

$$N \geq \frac{\log(1 - T)}{\log(1 - (1 - r)^s)}$$



RANSAC: How Many Trials

Suppose that:

r: Fraction of outliers (e.g. 80%)

s: Number of points we pick per set (e.g. 2)

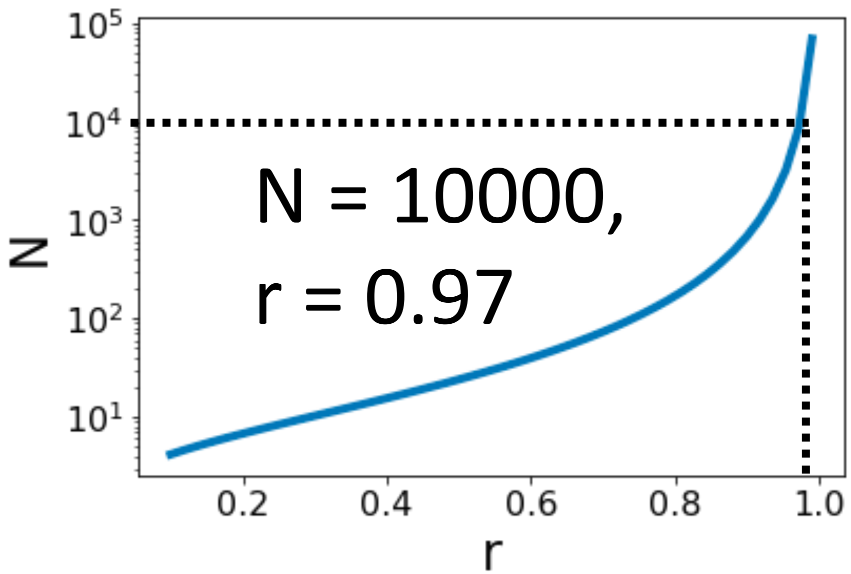
N: Number of times we run RANSAC (e.g. N=500)

C: Chance that we find a set with no outliers (e.g. 99.9%)

What's the probability of picking any set with no outliers?

$$C \geq 1 - (1 - (1 - r)^s)^N$$

$$N \geq \frac{\log(1 - T)}{\log(1 - (1 - r)^s)}$$



RANSAC: Subset Size

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

- **Why intuitively?**
- You'll find out more precisely in homework 3.

RANSAC: Inlier Threshold

- Common sense; there’s no magical threshold

RANSAC: Pros and Cons

Pros

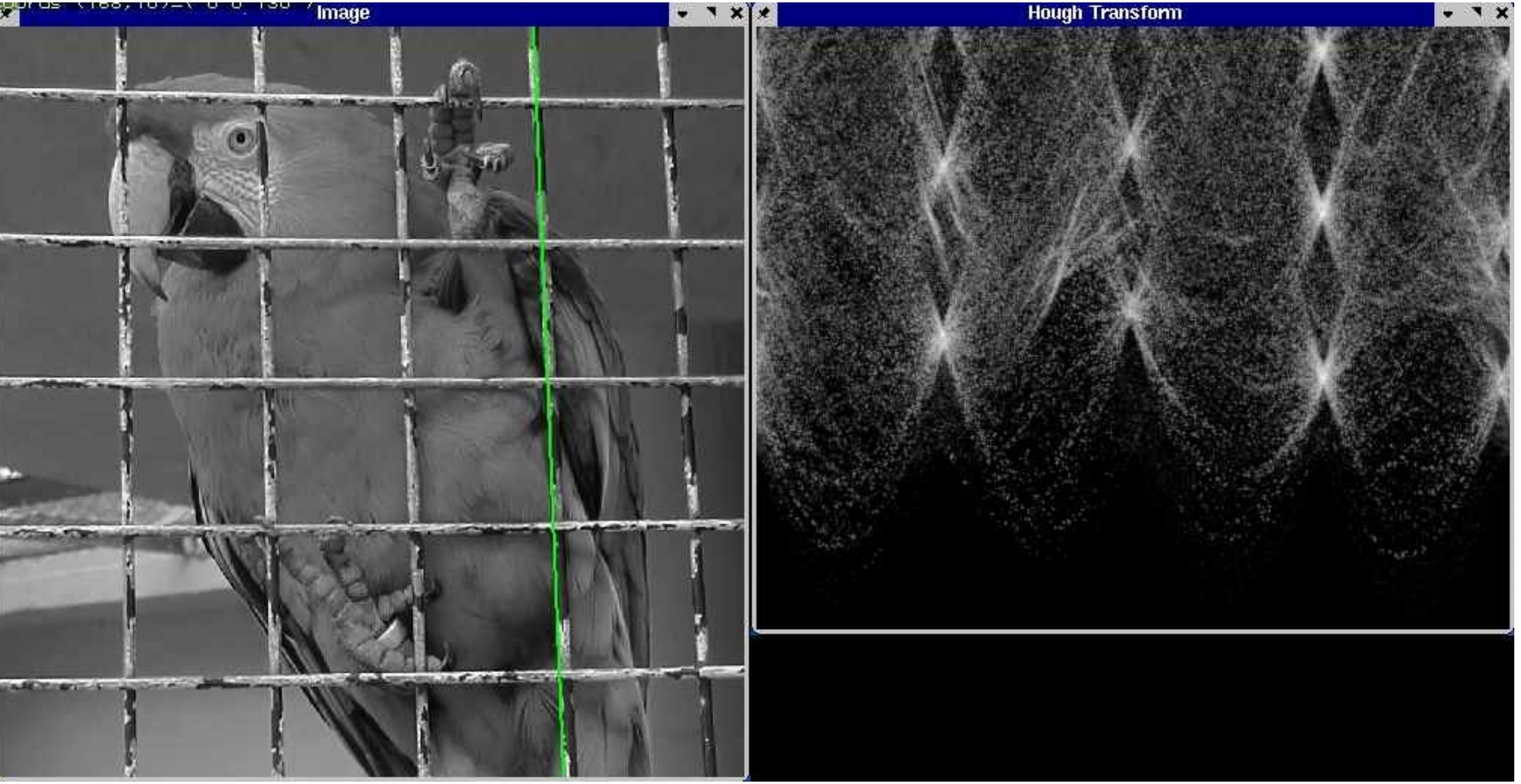
1. Ridiculously simple
2. Ridiculously effective
3. Works in general

Cons

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

Slide credit: S. Lazebnik

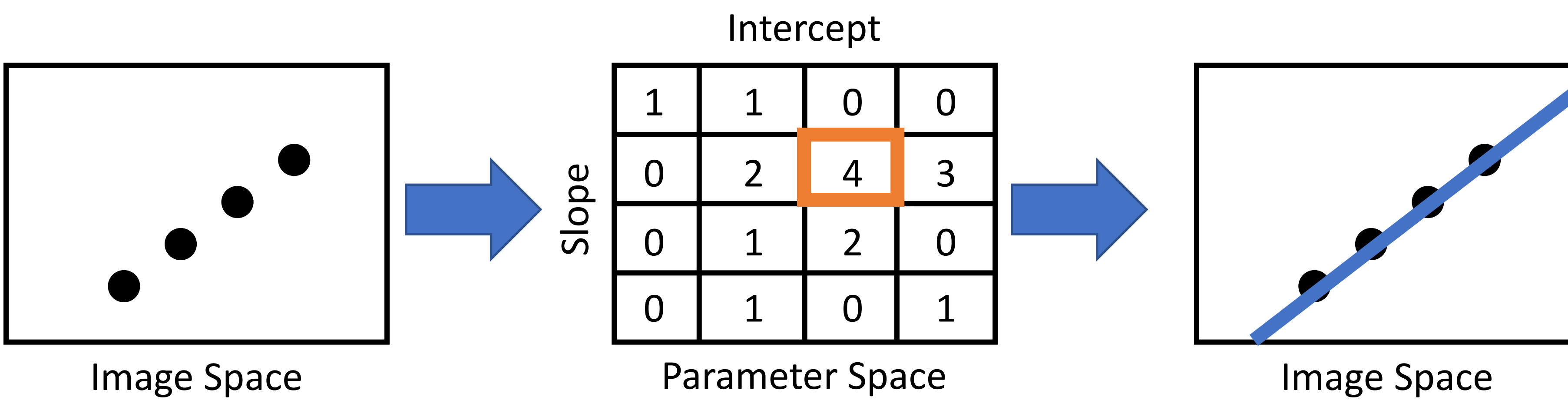
Hough Transform



Slide credit: S. Lazebnik

Hough Transform

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



P.V.C. Hough, *Machine Analysis of Bubble Chamber Pictures,* Proc. Int. Conf. High Energy Accelerators and Instrumentation, 1959

Slide credit: S. Lazebnik

Hough Transform

Line in image = point in parameter space

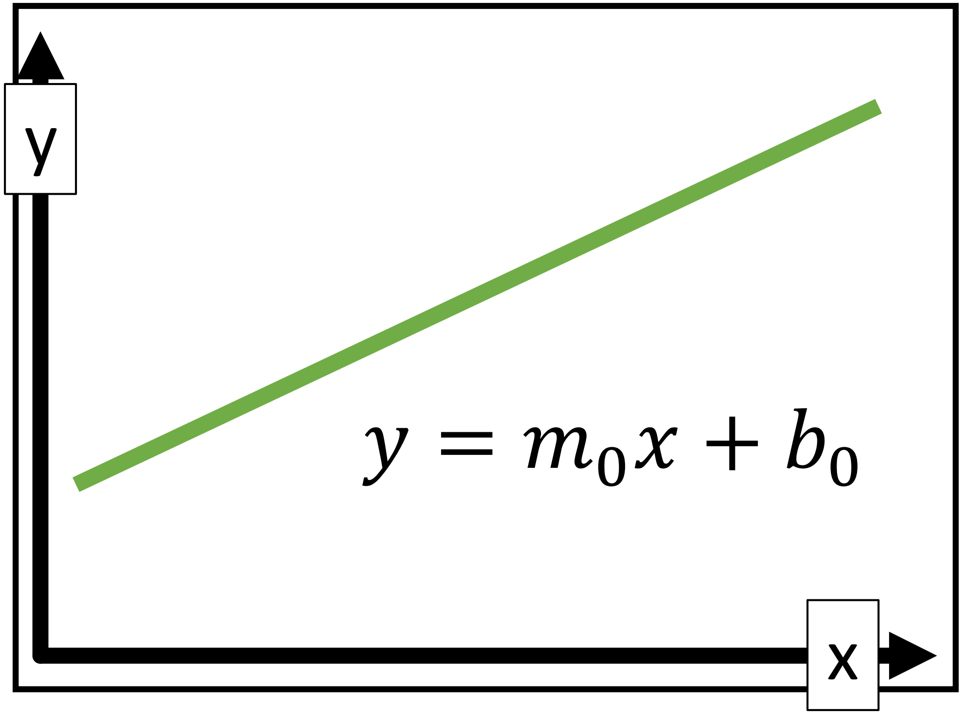
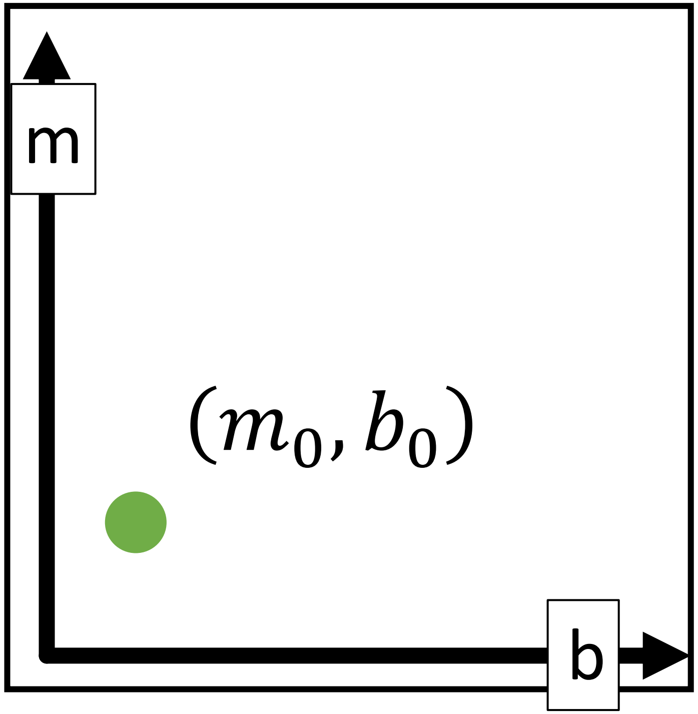


Image Space



Parameter Space

Diagram is remake of S. Seitz Slides; these are illustrative and values may not be real

Hough Transform

Point in image = line in parameter space

All lines through the point: $b = x_0 m + y_0$

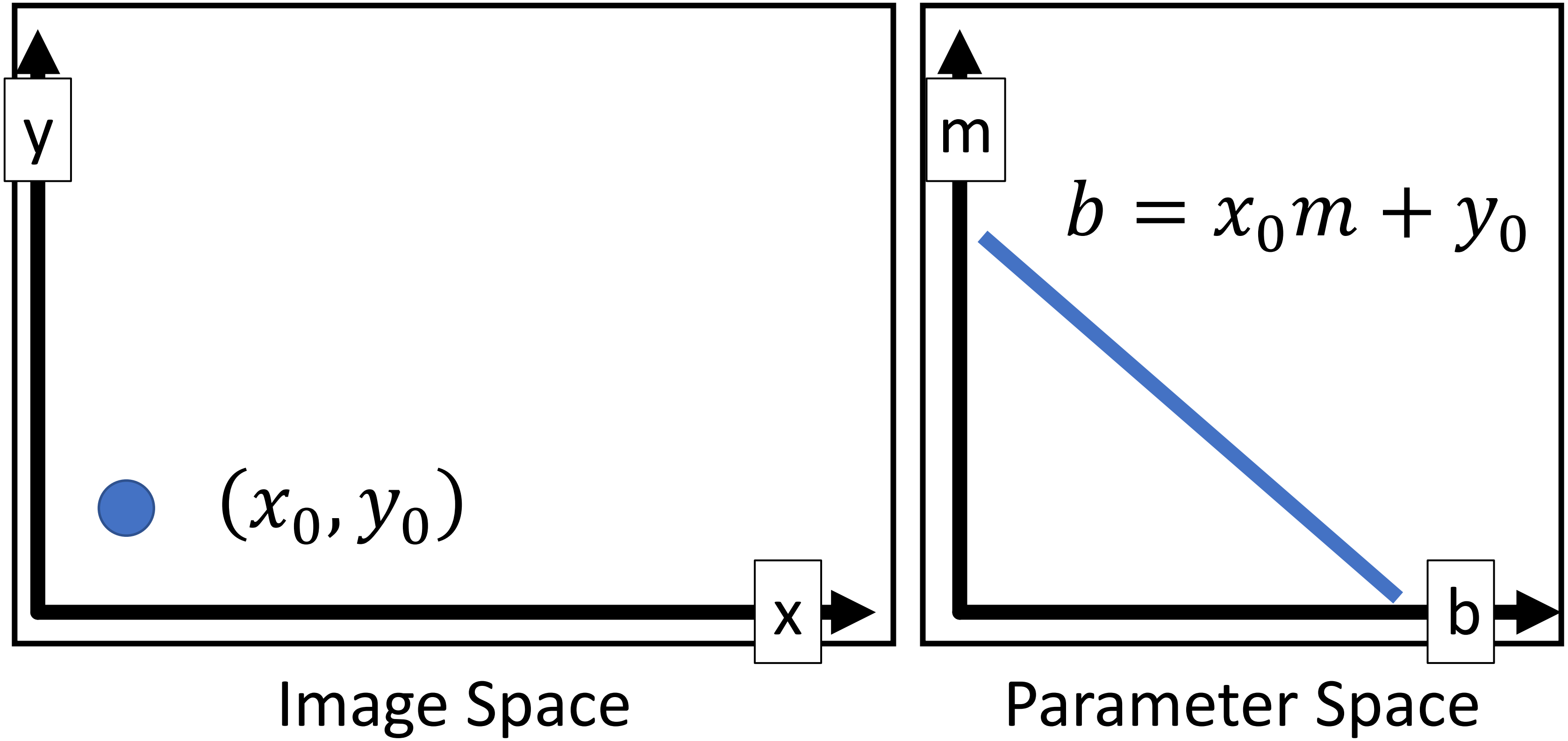


Diagram is remake of S. Seitz Slides; these are illustrative and values may not be real

Hough Transform

Point in image = line in parameter space

All lines through the point: $b = x_1 m + y_1$

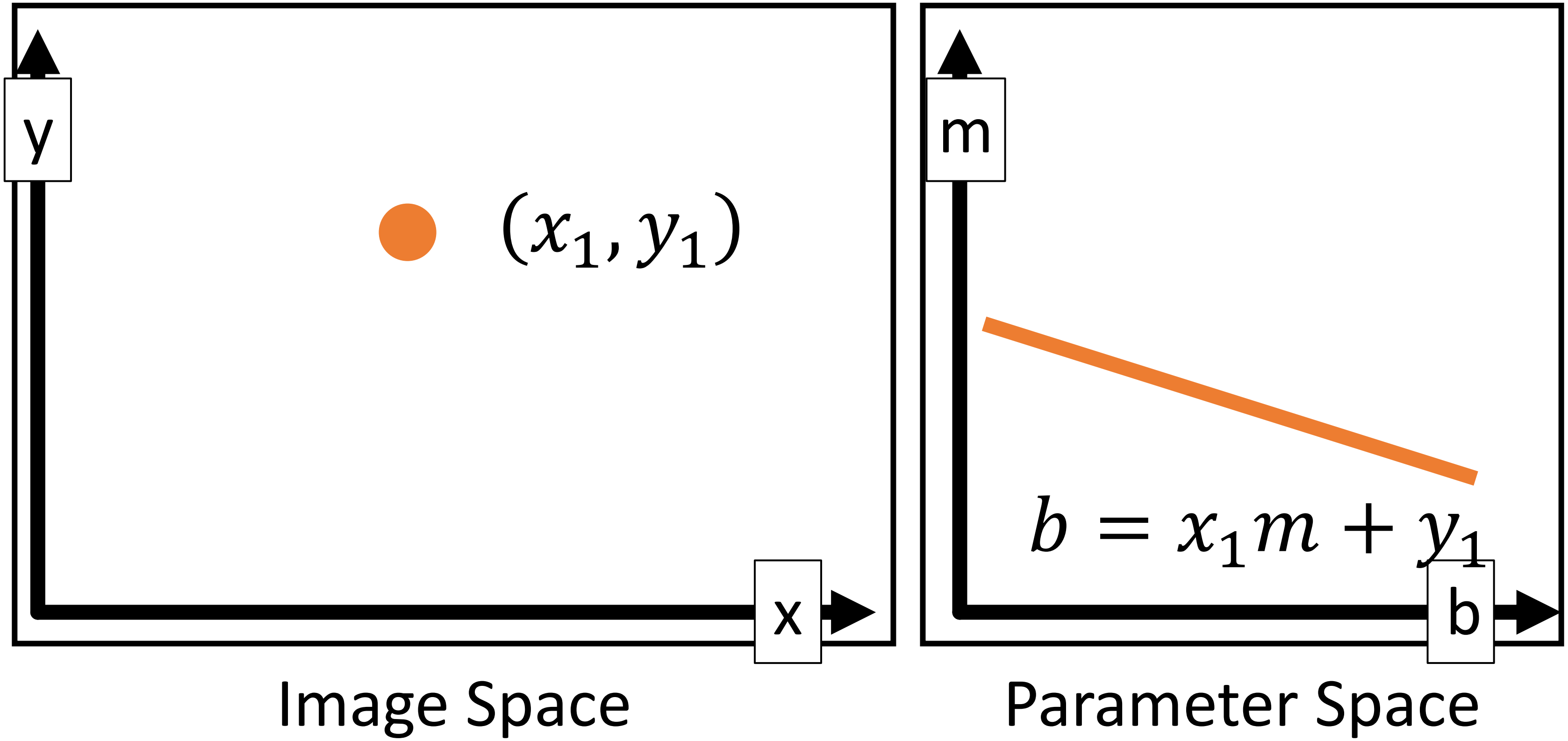
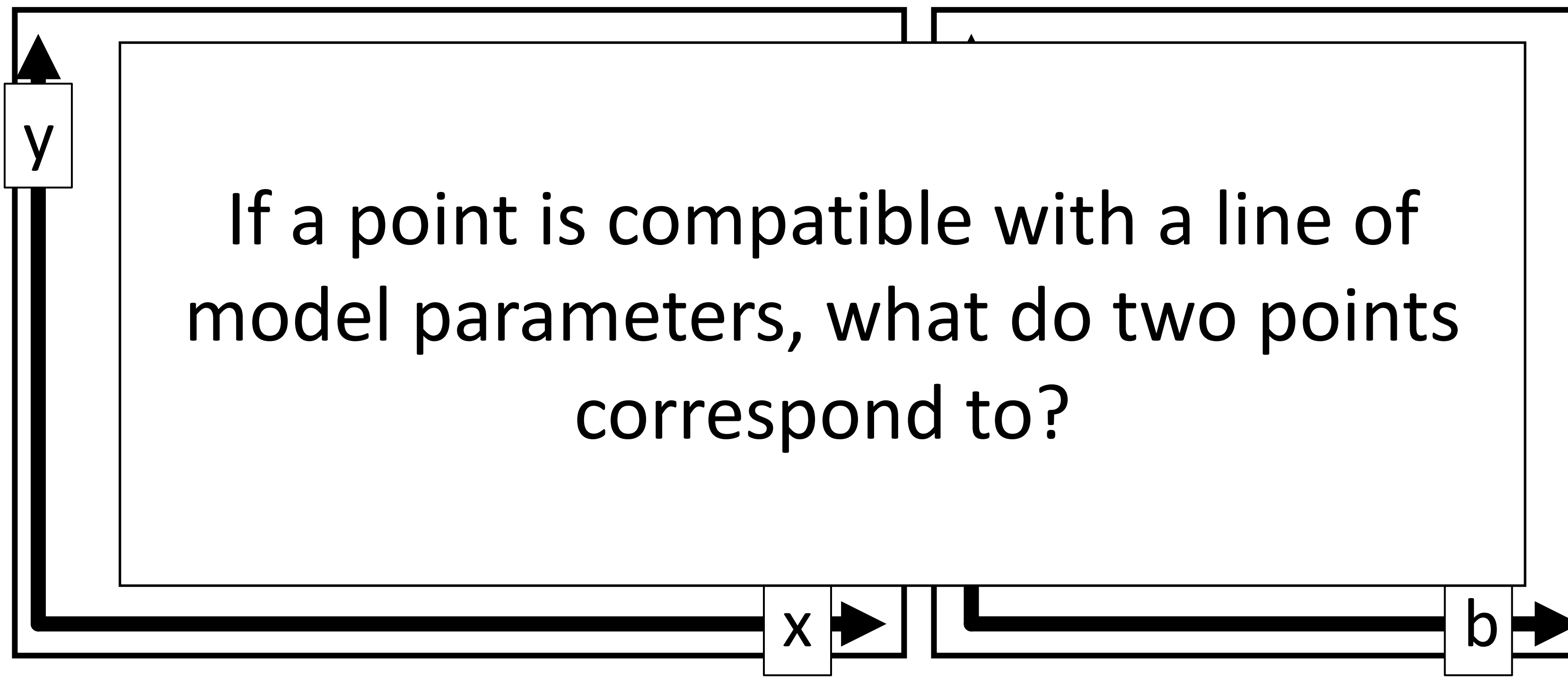


Diagram is remake of S. Seitz Slides; these are illustrative and values may not be real

Hough Transform

Point in image = line in parameter space

All lines through the point: $b = x_1 m + y_1$



If a point is compatible with a line of model parameters, what do two points correspond to?

Image Space

Parameter Space

Diagram is remake of S. Seitz Slides; these are illustrative and values may not be real

Hough Transform

Line through two points in image = intersection of two lines in parameter space (i.e., solutions to both equations)

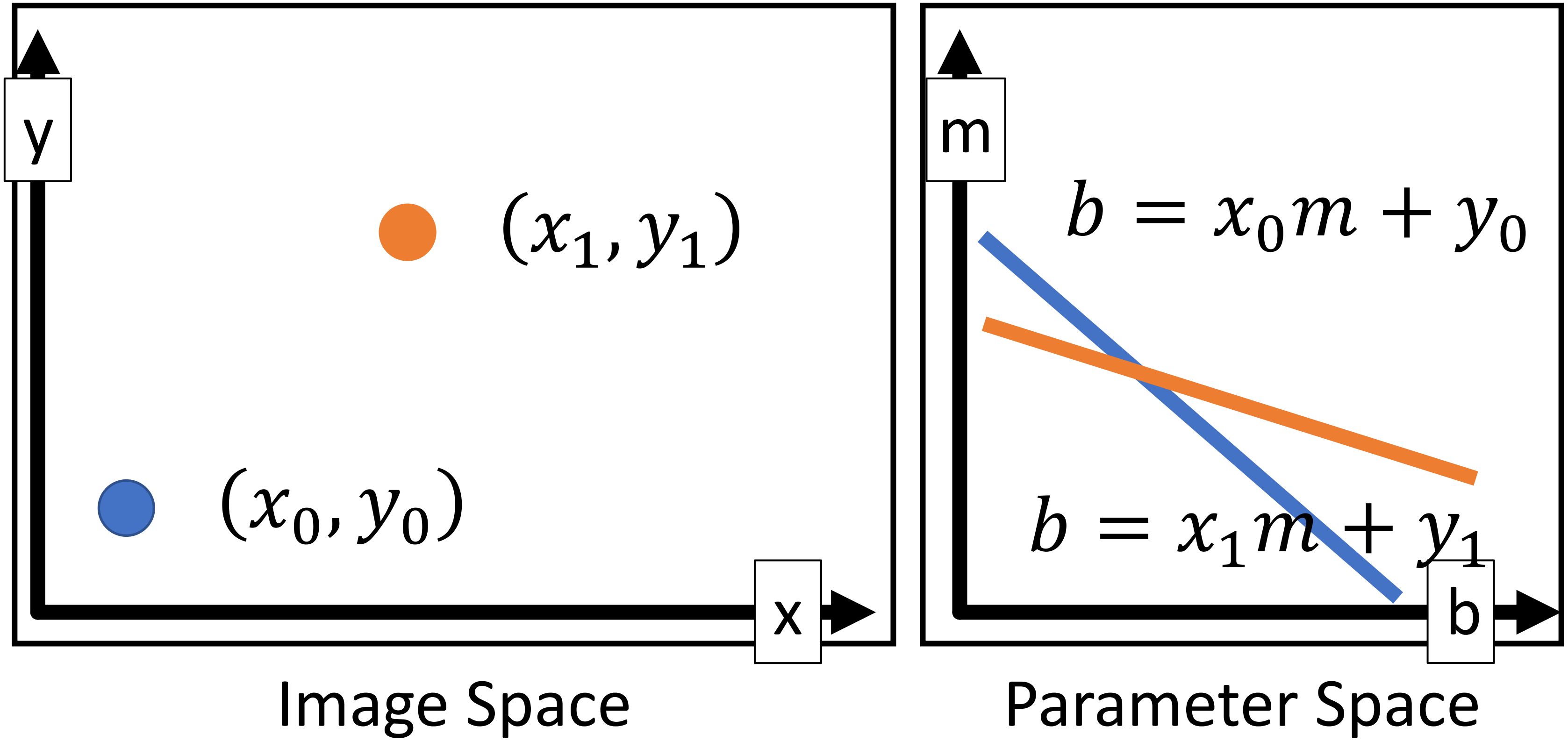


Diagram is remake of S. Seitz Slides; these are illustrative and values may not be real

Hough Transform

Line through two points in image = intersection of two lines in parameter space (i.e., solutions to both equations)

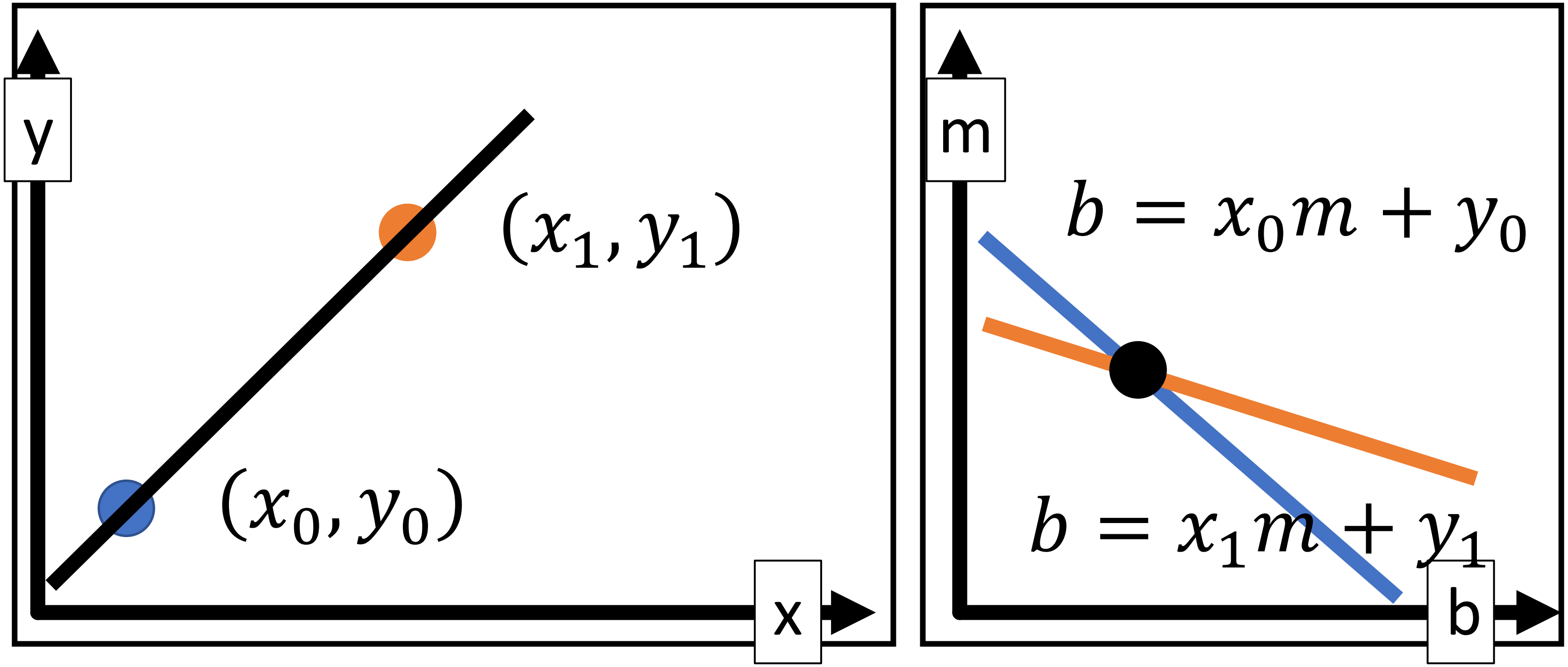


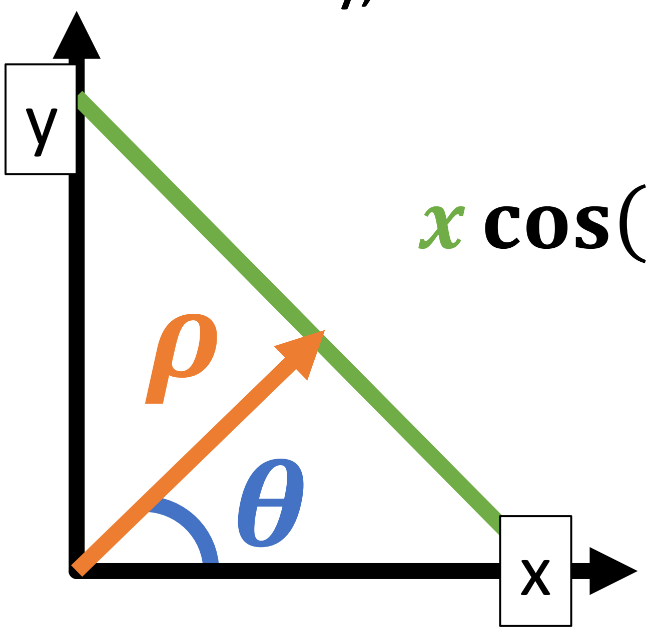
Image Space

Parameter Space

Diagram is remake of S. Seitz Slides; these are illustrative and values may not be real

Hough Transform

- *Recall*: m, b space is awful
- ax+by+c=0 is better, but *unbounded*
- Trick: write lines using angle + offset (normally a mediocre way, but makes things bounded)



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

Diagram is remake of S. Seitz Slides; these are illustrative and values may not be real

Hough Transform Algorithm

Remember: $x\cos(\theta) + y\sin(\theta) = \rho$

```
Accumulator H = zeros(?,?)
For x,y in detected_points:
    For θ in range(0,180,?):
        ρ = x cos(θ) + y sin(θ)
        H[θ, ρ] += 1
#any local maxima (θ, ρ) of H is a line
#of the form ρ = x cos(θ) + y sin(θ)
```

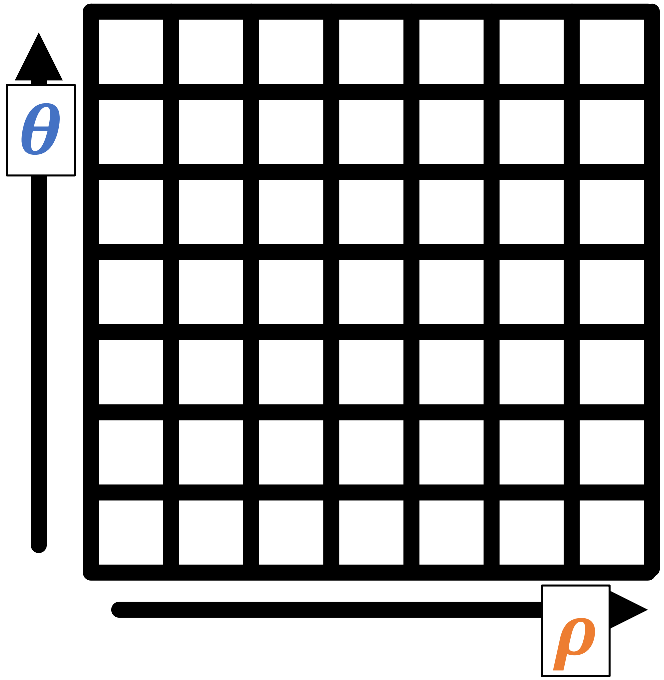


Diagram is remake of S. Seitz Slides; these are illustrative and values may not be real

Hough Transform: Example

Points (x,y) -> sinusoids

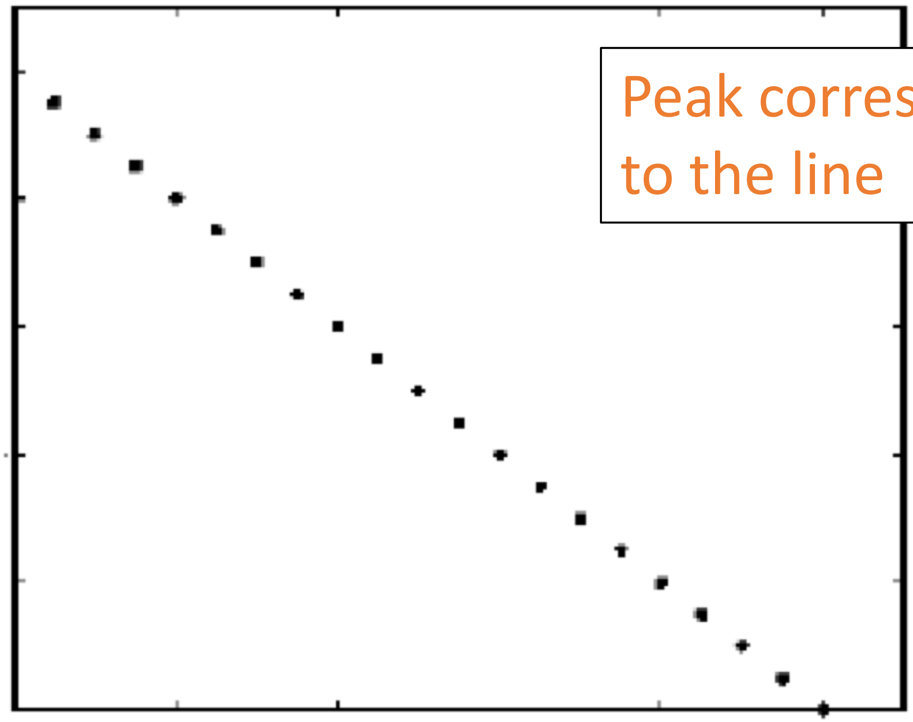
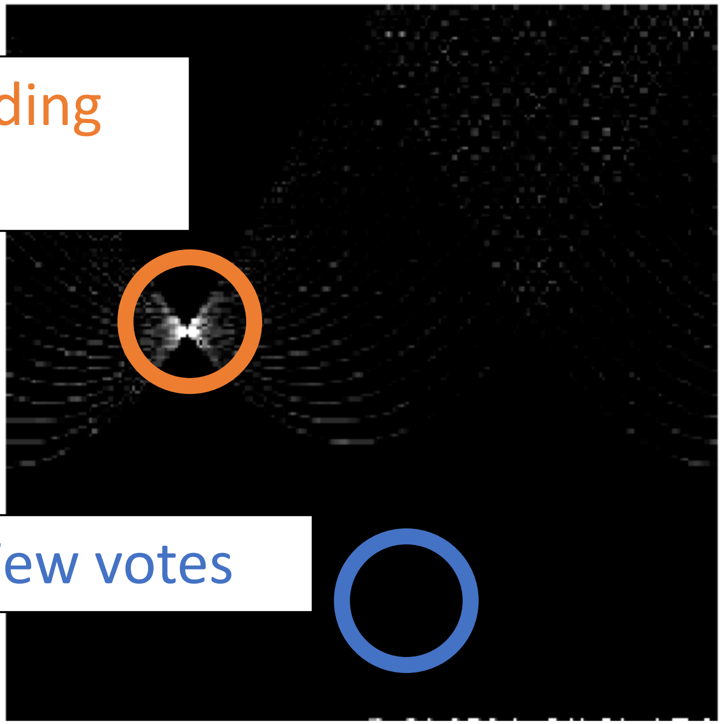


Image Space

Peak corresponding to the line

Few votes



Parameter Space

Slide Credit: S. Lazebnik

Hough Transform: Example

Pros

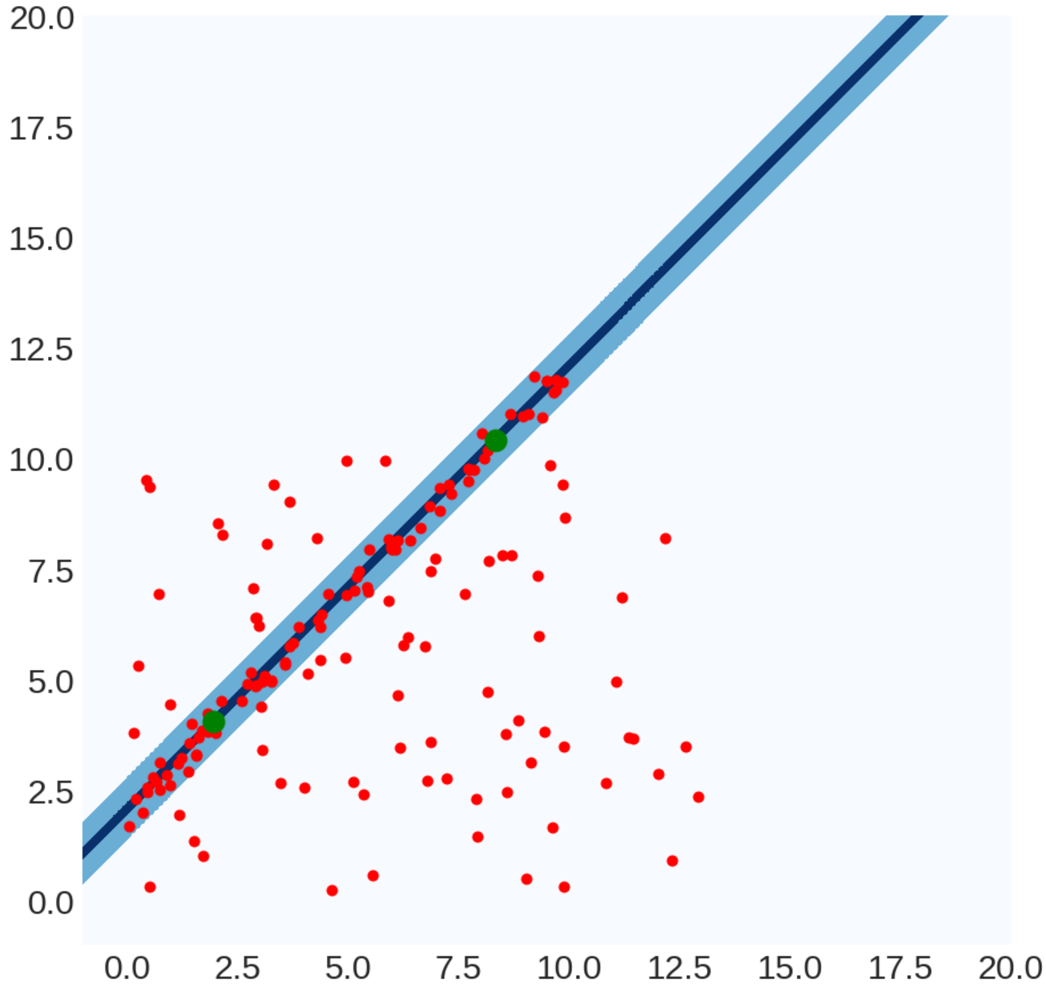
1. Handles **multiple** models
2. Some robustness to noise
3. In principle, general

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)

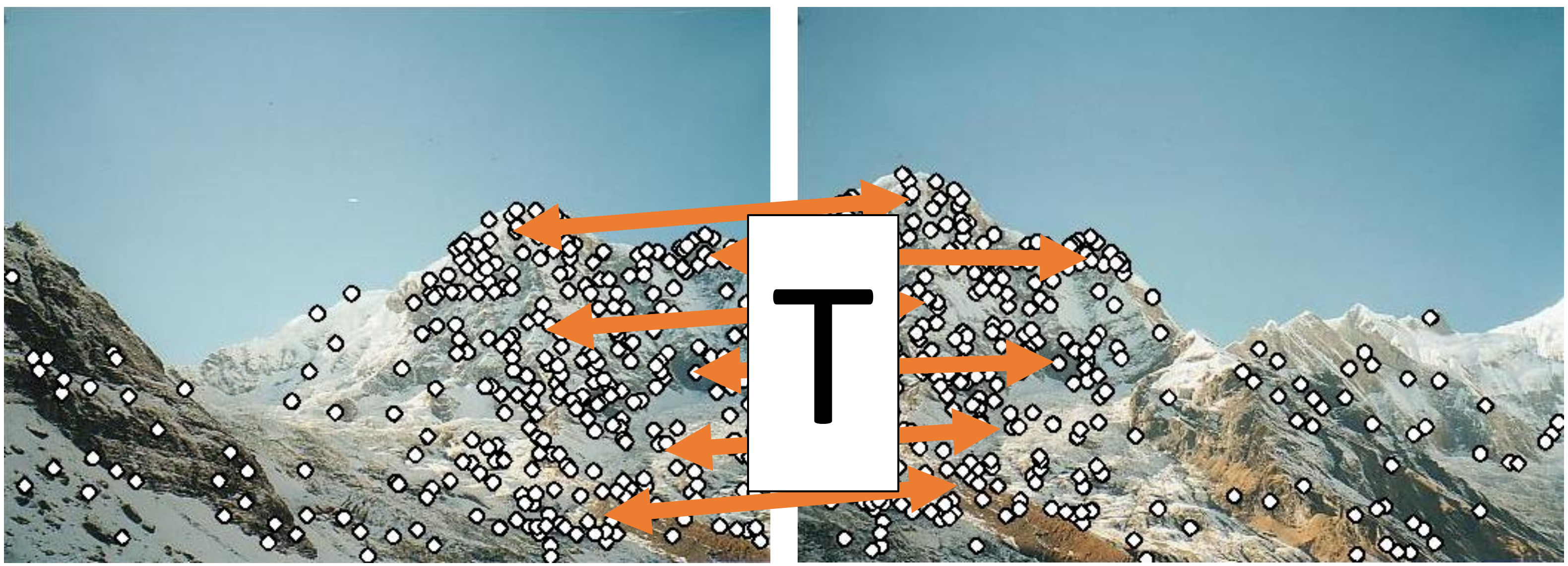
Slide Credit: S. Lazebnik

Today: Fitting Lines



Next Time: Fitting More Complex Transforms

Solving for a Transformation



3: Solve for transformation T