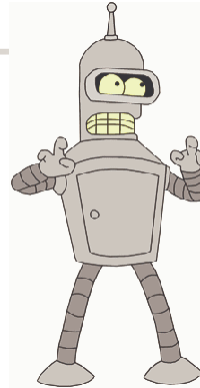


Methods for Extracting Line Segments from Sensor Data: RANSAC & Hough Transform

Joe Souto
Intro to Mobile Robotics



Outline

- Introduction
- RANSAC Algorithm
- Example problem
- Challenges in RANSAC
- Hough Transform
- Demonstration



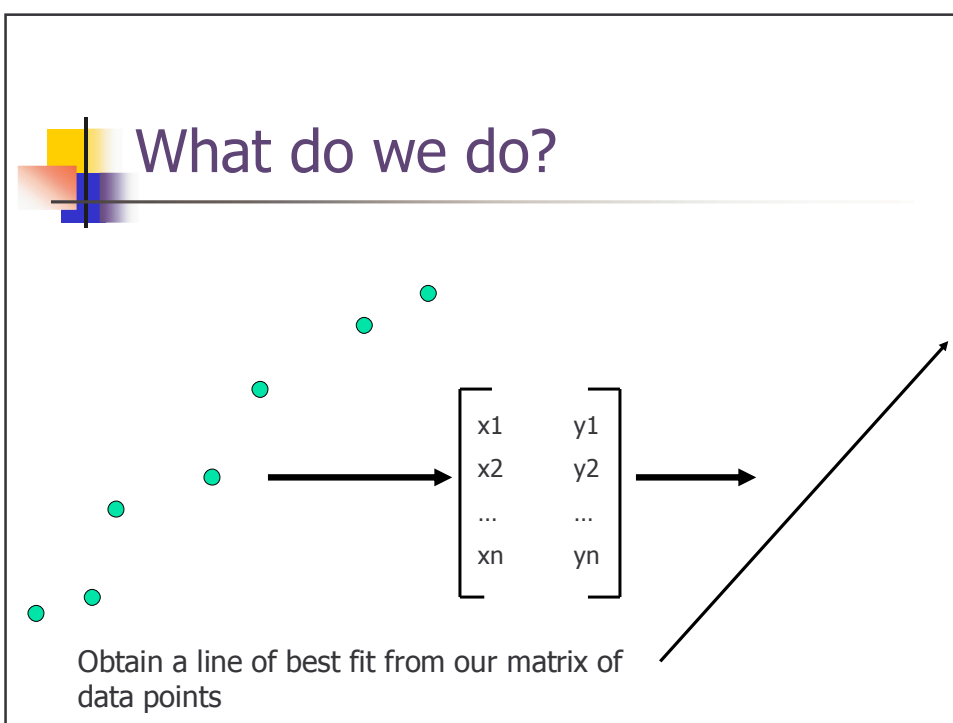
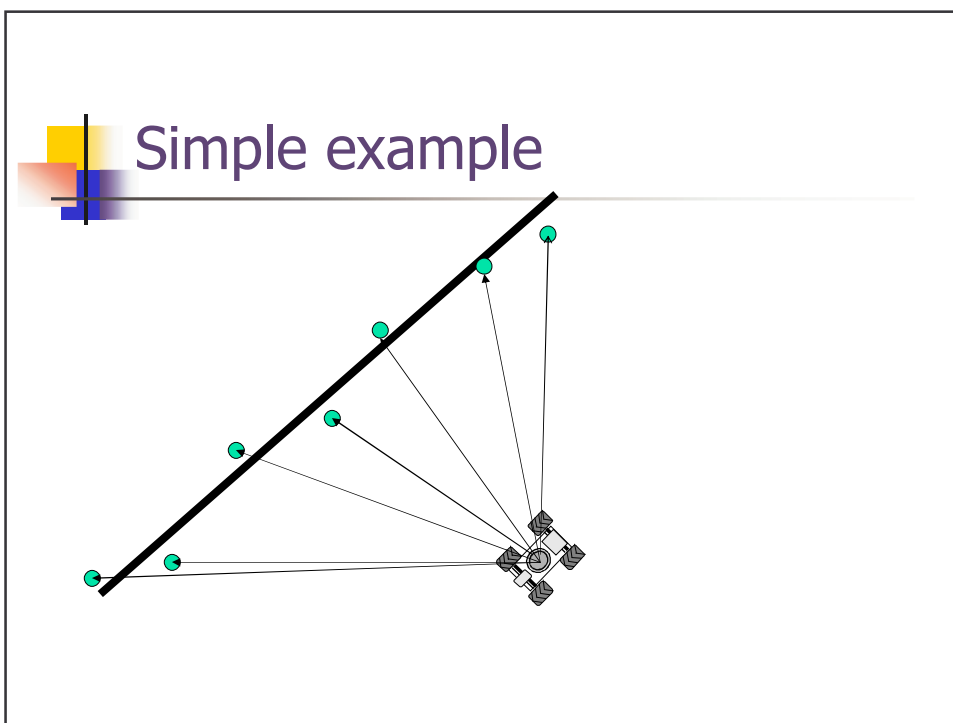
Purpose

- Given a set of data points, we want a robust method to identify lines which represent features in the environment
 - Must filter sensor data
 - Take remaining useful measurements and obtain parameters of interest



In a Perfect World

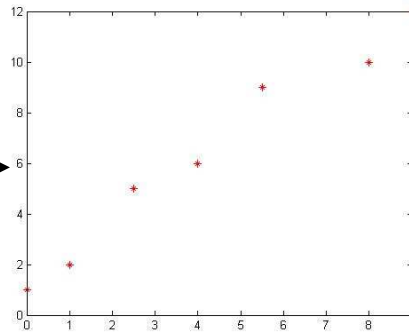
- All data points would roughly correspond to obstacles with no outliers
- We could use a Least Squares fit to obtain a line that perfectly matches a feature in the environment



How? Our Friend Matlab

We plot our data points in Matlab

$$\begin{bmatrix} x1 & y1 \\ x2 & y2 \\ \dots & \dots \\ xn & yn \end{bmatrix}$$



Matlab Continued

- We want a line of the form:
 $y = ax + b$

$$\begin{bmatrix} x1 & 1 \\ x2 & 1 \\ \dots & \dots \\ xn & 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} y1 \\ y2 \\ \dots \\ yn \end{bmatrix}$$

→ $Ax = b$

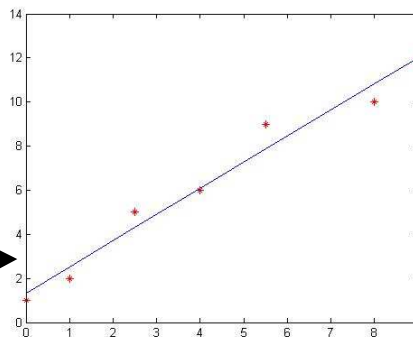
- A is overdetermined due to noise, so we solve with the pseudo-inverse

Matlab Continued

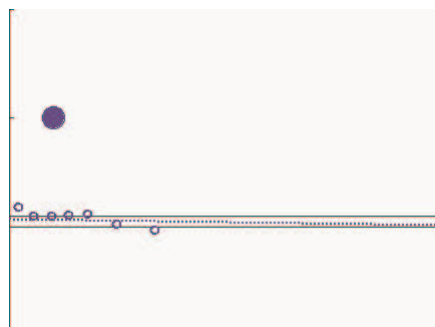
$$A^T A x = A^T b$$
$$x = (A^T A)^{-1} A^T b \longrightarrow \begin{bmatrix} a \\ b \end{bmatrix}$$

$x[1]$ is the slope
 $x[2]$ is the intercept

We can now plot the
line in Matlab



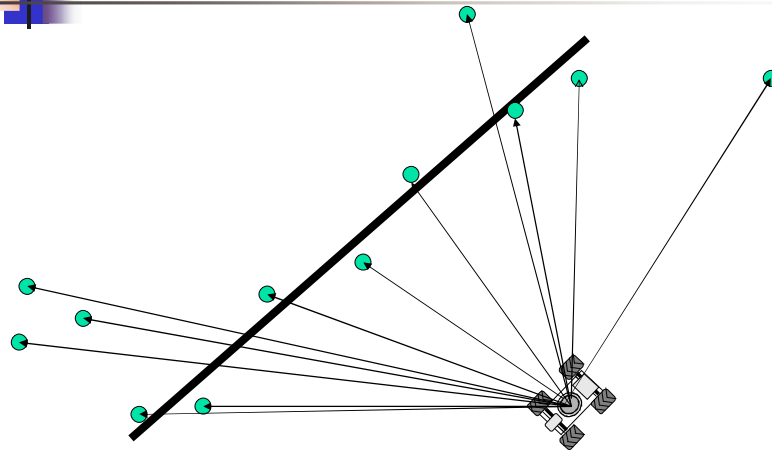
Simulation



Least Squares Wall Follower
Random Noise

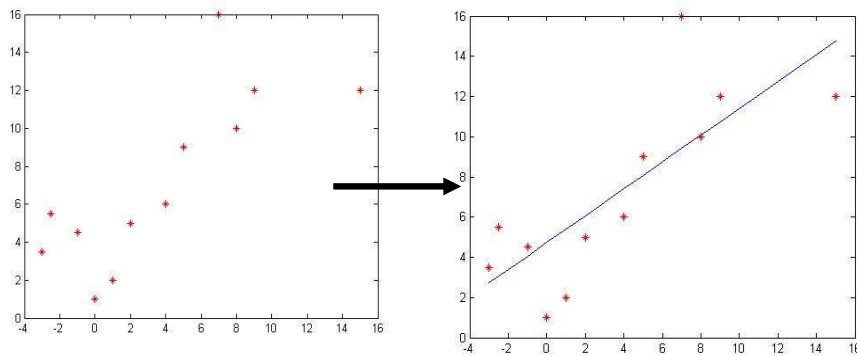
BUT! The
reality is
more
complex

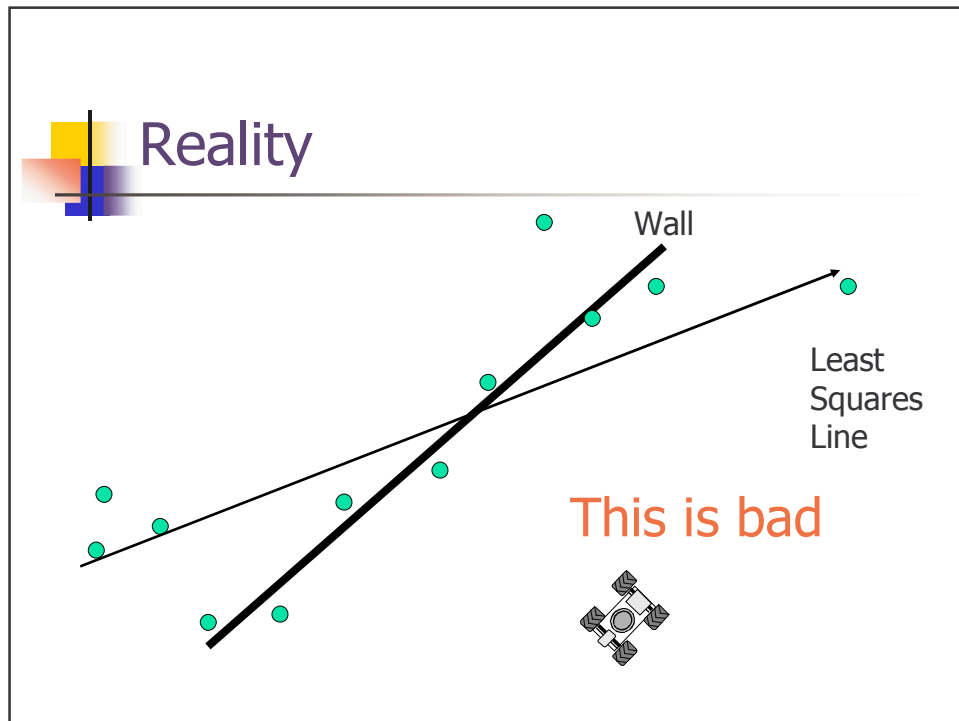
The Reality



Does Least Squares still work?

Plot points in Matlab, do $x = (A^T A)^{-1} A^T b$, etc etc





- ## How do we solve this problem?
- As shown, large outliers cause big problems with estimating obstacles
 - We need an algorithm that will filter out these bad data points
 - We can then fit a line to filtered set of points that is a reasonable approximation of where that line is in reality



Method #1 - RANSAC

- RANSAC = “Random Sample Consensus”
- What does this mean?
 - We will randomly sample points from our data set
 - Make a line model based on that data
 - See how many other data points fit that model (“Consensus”)
- Used with sensor data obtained from LRF or Sonar

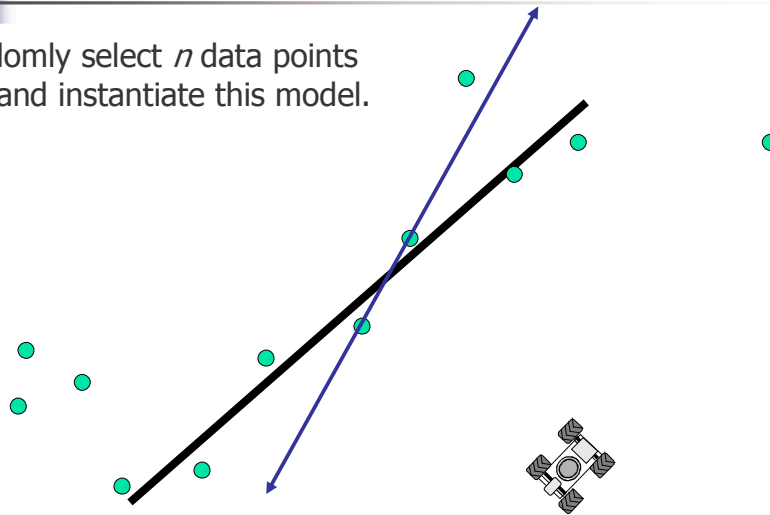


RANSAC – Basic Algorithm

- Given a feature model M that requires a minimum of n data points to instantiate its free parameters, and a set of data points P such that $|P| \geq n$:
- Randomly select n data points from P and instantiate this model.
 - Use the instantiated model M_1 to determine the subset S_1^* of points P that are within some error tolerance of M_1 . The set S_1^* is called the consensus set of S_1 .
 - If $|S_1^*|$ is greater than a threshold t use S_1^* to compute a new model M_1^* .
 - If $|S_1^*| < t$, randomly select a new subset S_2 and repeat the above process.
 - If after some predetermined number of trials no consensus set with t or more members has been found, either solve the model with the largest set found or terminate in failure.

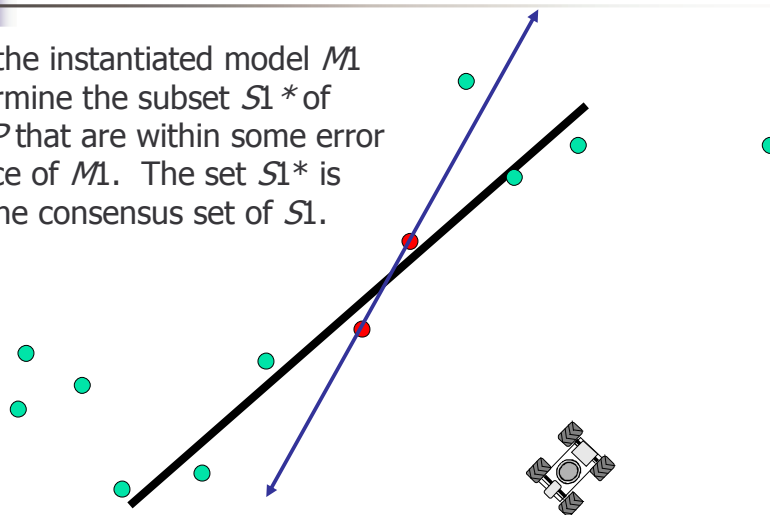
Let's Do an Example

1) Randomly select n data points from P and instantiate this model.



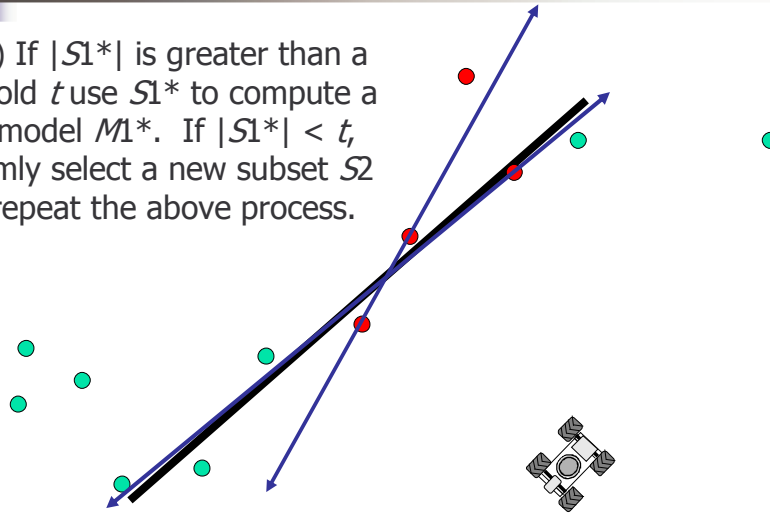
Let's Do an Example

2) Use the instantiated model M_1 to determine the subset S_1^* of points P that are within some error tolerance of M_1 . The set S_1^* is called the consensus set of S_1 .

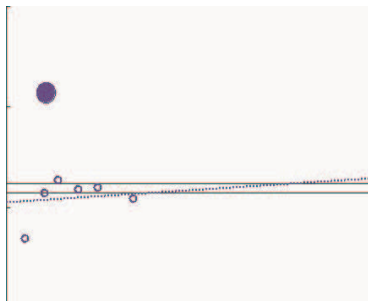


Let's Do an Example

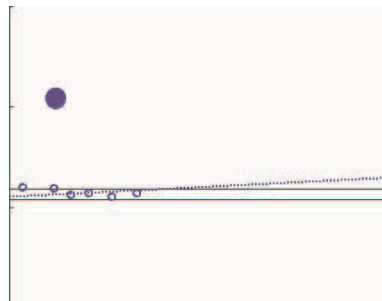
3 & 4) If $|S1^*|$ is greater than a threshold t use $S1^*$ to compute a new model $M1^*$. If $|S1^*| < t$, randomly select a new subset $S2$ and repeat the above process.



Simulations



Least Squares Wall Follower
Random Noise & 20% Outliers



RANSAC Wall Follower
Random Noise & 20% Outliers



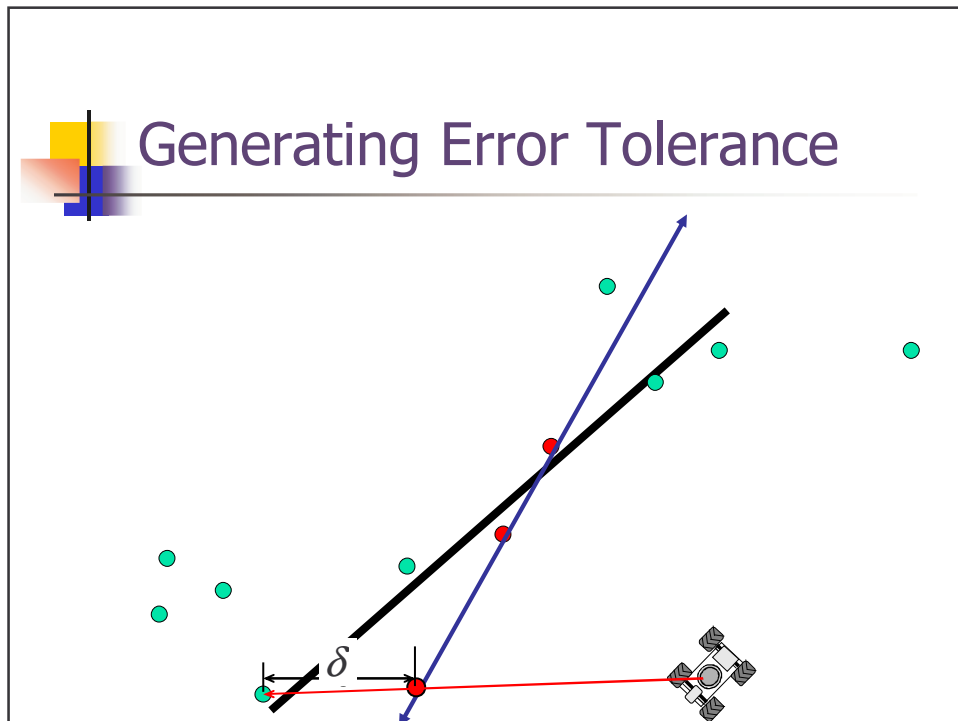
Issues We Need To Deal With

- 1) Generating the error tolerance
- 2) Number of iterations to perform
- 3) Choosing a good threshold



How to Generate Error Tolerance

- Use sensor/feature models
 - Assume estimated feature model is correct
 - Infer perfect, hypothetical sensor measurements corresponding to the estimated feature model
 - Estimate the distance δ of the inferred measurement from the actual measurement. If $\delta \leq \delta_{\max}$ accept, otherwise reject.
 - One possible metric for δ would be the standard deviation from our noise model



- ## Determining Iterations
- How many times must we select a subset of P to find a consensus set?
 - Depends upon the expected number of trials (k) required to select a subset of n good data points.
 - Let w be the probability that any selected data point is within the error tolerance of the model. Then...



Determining Iterations

Let $E(k)$ be the expected value of k , and set $b = w^n$, we have the following:

$$E(k) = b + 2*(1 - b)*b + 3*(1 - b)^2*b \dots \\ + i*(1 - b)^{i-1}*b + \dots$$

- With some math, it can be shown that:

$$E(k) = 1/b = w^{-n}$$



Determining Iterations

- Sample values of $E(k)$ for corresponding values of n and w

w	$n = 1$	2	3	4	5	6
0.9	1.1	1.2	1.4	1.5	1.7	1.9
0.8	1.3	1.6	2.0	2.4	3.0	3.8
0.7	1.4	2.0	2.9	4.2	5.9	8.5
0.6	1.7	2.8	4.6	7.7	13	21
0.5	2.0	4.0	8.0	16	32	64
0.4	2.5	6.3	16	39	98	244
0.3	3.3	11	37	123	412	—
0.2	5.0	25	125	625	—	—



Choosing a Threshold

- We want to capture as much valid data as possible. Realistically, we want our threshold, t , to be as large as possible
- We can estimate a value of t based on an assumption of how much of the data should belong to our particular model
- A simple majority of the available information can often be enough

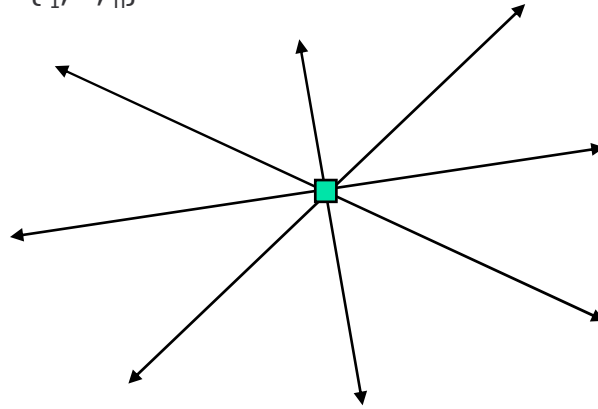


Method #2: Hough Transform

- Used on segmented images from a Camera
- A line in the image can be parameterized by 2 variables
$$y = a_i x + b_i$$
- Each edge pixel (x,y) corresponds to a family of lines $L(x,y) = \{l_1, \dots, l_n\}$
- Pixel (x,y) votes for each $l_i \in L(x,y)$
- Edge pixels that form a line will each place one vote for the same (a_i, b_i) - along with lots of other lines
- Lines that are in the image will receive more votes than ones that are not

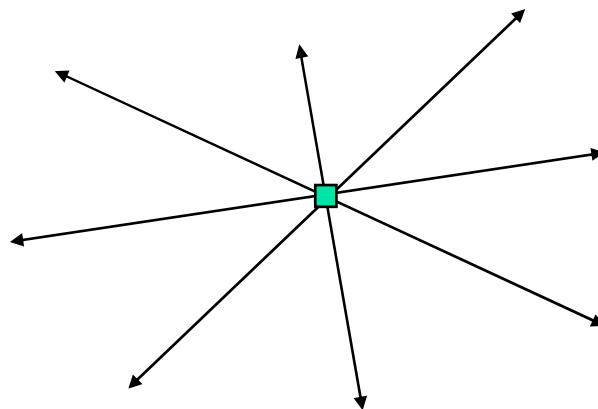
Hough Transform

- Each edge pixel (x,y) corresponds to a family of lines $L(x,y) = \{l_1, \dots, l_n\}$



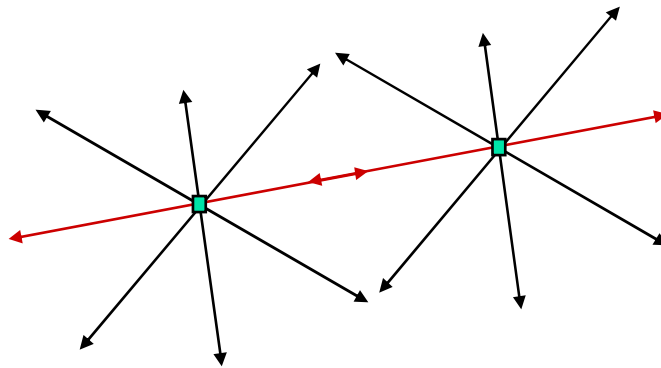
Hough Transform

- Pixel (x,y) votes for each $l_i \in L(x,y)$



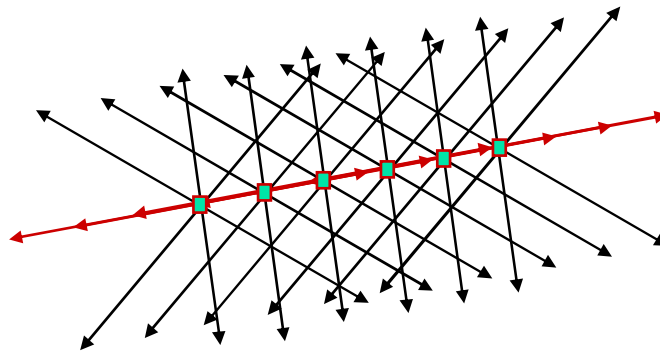
Hough Transform

- Edge pixels that form a line will each place one vote for the same (a_i, b_i) – along with lots of other lines



Hough Transform

- Lines that are in the image will receive more votes than ones that are not





Hough Transform

- Problem reduces to peak detection
 - Find lines with most votes
- However, we have a problem

$$a \in [-\infty, \infty]$$

$$b \in [-\infty, \infty]$$

- Infinite space is bad, we need to discretize the space we are working with



Solution: Polar Coordinates

- Represent the lines with ρ and θ instead of x and y

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

- This reduces our possible parameter ranges to the following:

$$\rho \in [0, \sqrt{im_h^2 + im_w^2}]$$

$$\theta \in [0, 2\pi]$$

- Where im_h = image height, im_w = image width



Hough Transform Algorithm

1. Take as input an edge image $E(i,j)$
2. Define a resolution $d\theta$ and $d\rho$ for θ and ρ , respectively
3. Construct an accumulator array $A(m,n)=0$ where $m=1/d\rho*\rho_{max}$ and $n=2\pi/d\theta$

$$\rho_{max} \in [0, \sqrt{im_h^2 + im_w^2}]$$

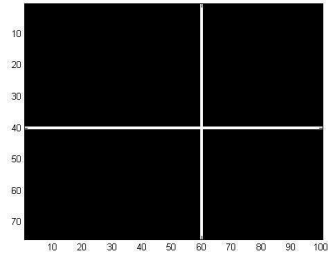


Hough Transform Algorithm

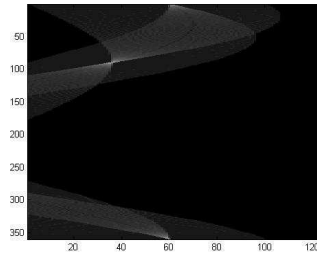
4. For each pixel $E(i,j) \neq 0$ do
 For $\theta = d\theta:d\theta:2*\pi$
 1. $\rho = i*\cos(\theta) + j*\sin(\theta)$
 2. if $\rho < 0$, continue
 3. Round ρ to the nearest $d\rho$ value
 4. $A(\rho, \theta)++$
5. Threshold $A(\rho, \theta)$ to find all relevant lines



Sample Peak Detection



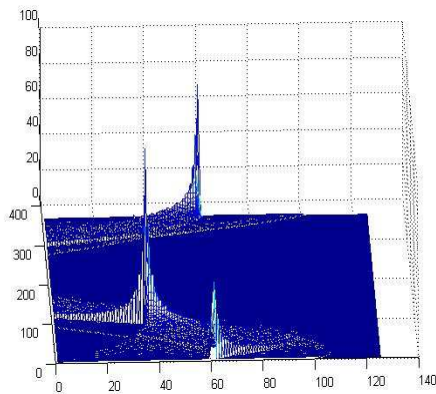
$E(i, j)$



$A(\rho, \theta)$



Sample Peak Detection



- Hough will give us information like this
- Peaks correspond to points with most votes
- Set threshold to obtain lines and remove other data



What's the use of all this?

- My Senior Project:
- Simulated Autonomous Highway
- Used Hough Transform to follow track lanes



Robot Videos



More Robot Videos

Even More Robot Videos



Further Discussion

- What did we do with Hough?
 - Extrapolate intersection of track lines
 - Calculate our angle of orientation with respect to this center point
 - Steer robot towards the center