CS283: Robotics Spring 2025: Perception

Sören Schwertfeger

ShanghaiTech University

RANGE SENSING

Range Sensing

- Color/ gray scale cameras: do NOT measure the distance to the object
- Range sensing: get the distance to the object
- Basic principles:
  - <u>Time of flight</u>
    - Sound/ Ultrasound (in air, underwater)
    - Light (Based on Phase or based on time)
      - Single rotating laser beam (LRF; e.g. Sick)
      - Multiple rotating laser beams (3D LRF; e.g. Velodyne)
      - Solid state laser (e.g. Intel RealSense L515)
      - LED light & imager (ToF camera, e.g. Kinect 2)
    - Radio Waves (Radar)

- <u>Projected Pattern</u>
  - Single laser (Triangulation)
  - 2D pattern (e.g. Kinect 1)

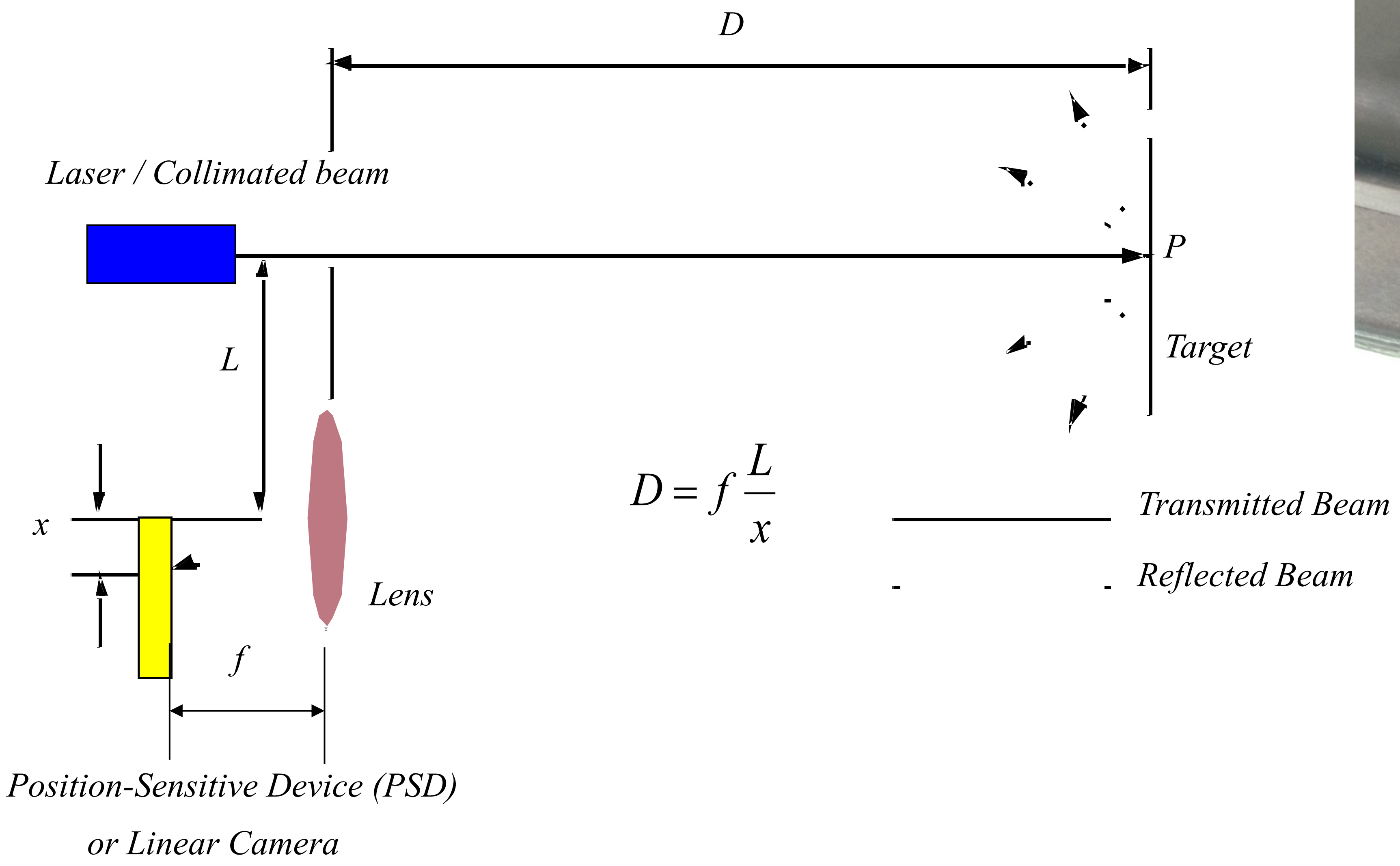
- <u>Stereo Vision</u>
  - Passive
  - Active with pattern (e.g. Intel RealSense D435)

RANGE SENSING: PROJECTED PATTERN

Triangulation Ranging

- Geometrical properties of the image to establish a distance measurement
- e.g. project a well defined light pattern (e.g. point, line) onto the environment.
  - reflected light is than captured by a photo-sensitive line or matrix (camera) sensor device
  - simple triangulation allows to establish a distance.
- e.g. size of an captured object is precisely known
  - triangulation without light projecting

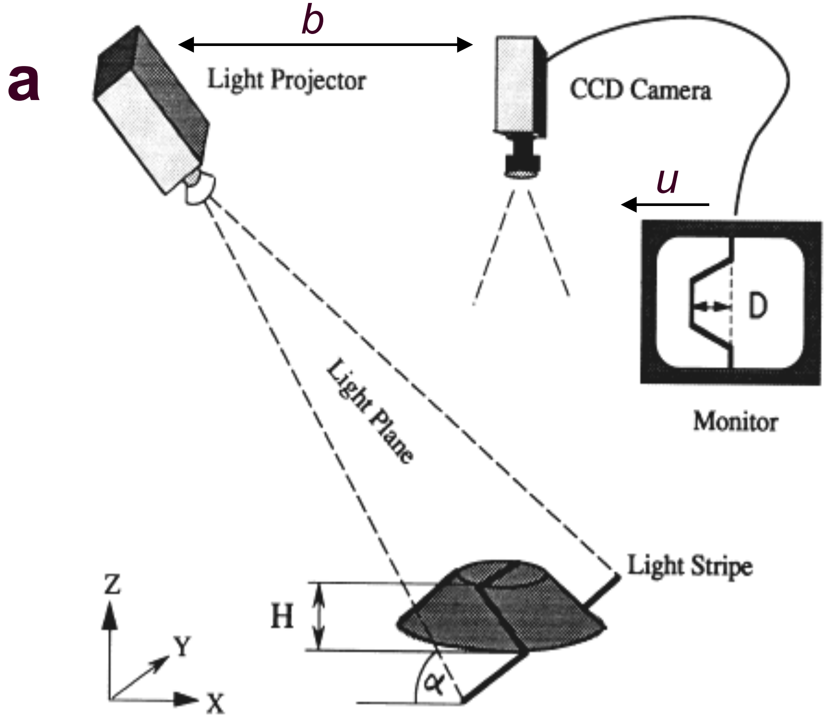
Laser Triangulation (1D)

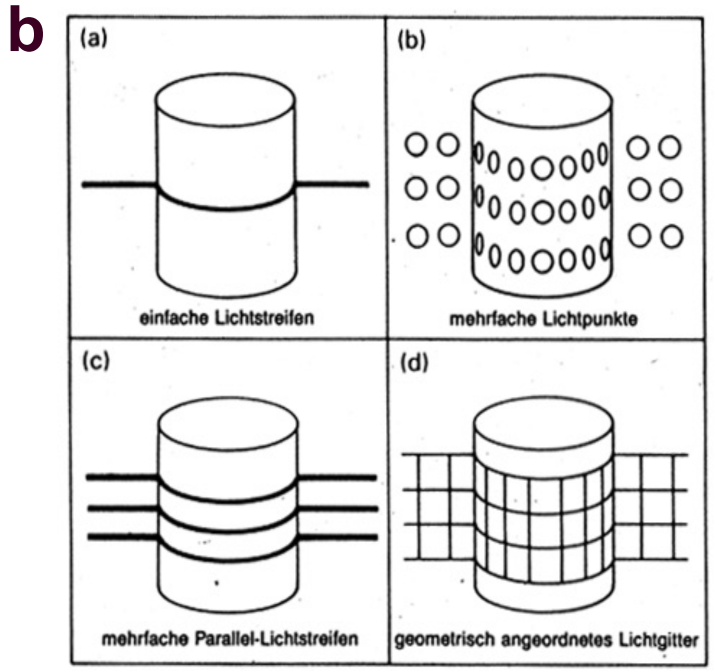


$$D = f \frac{L}{x}$$

Cliff sensor on Turtle-bot

Structured Light (vision, 2 or 3D): Structured Light





- Eliminate the correspondence problem by projecting structured light on the scene.
- Slits of light or emit collimated light (possibly laser) by means of a rotating mirror.
- Light perceived by camera
- Range to an illuminated point can then be determined from simple geometry.

Structured Light (vision, 2 or 3D)

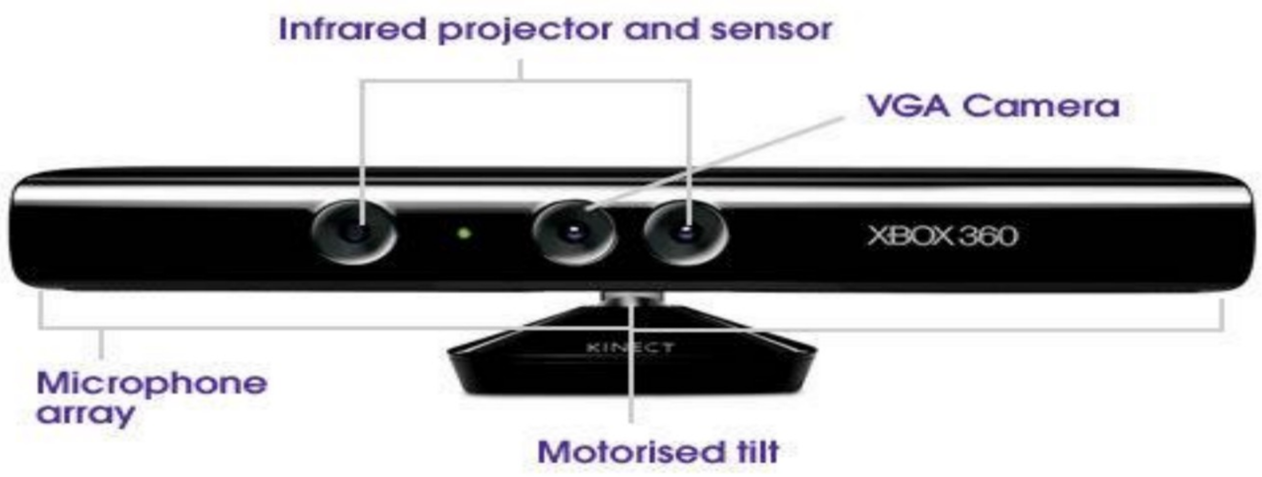
- Baseline length *b*:
  - the smaller b is the more compact the sensor can be.
  - the larger b is the better the range resolution is.

Note: for large b, the chance that an illuminated point is not visible to the receiver increases.

- Focal length f:
  - larger focal length f can provide
    - either a larger field of view
    - or an improved range resolution
  - however, large focal length means a larger sensor head

PrimeSense Cameras

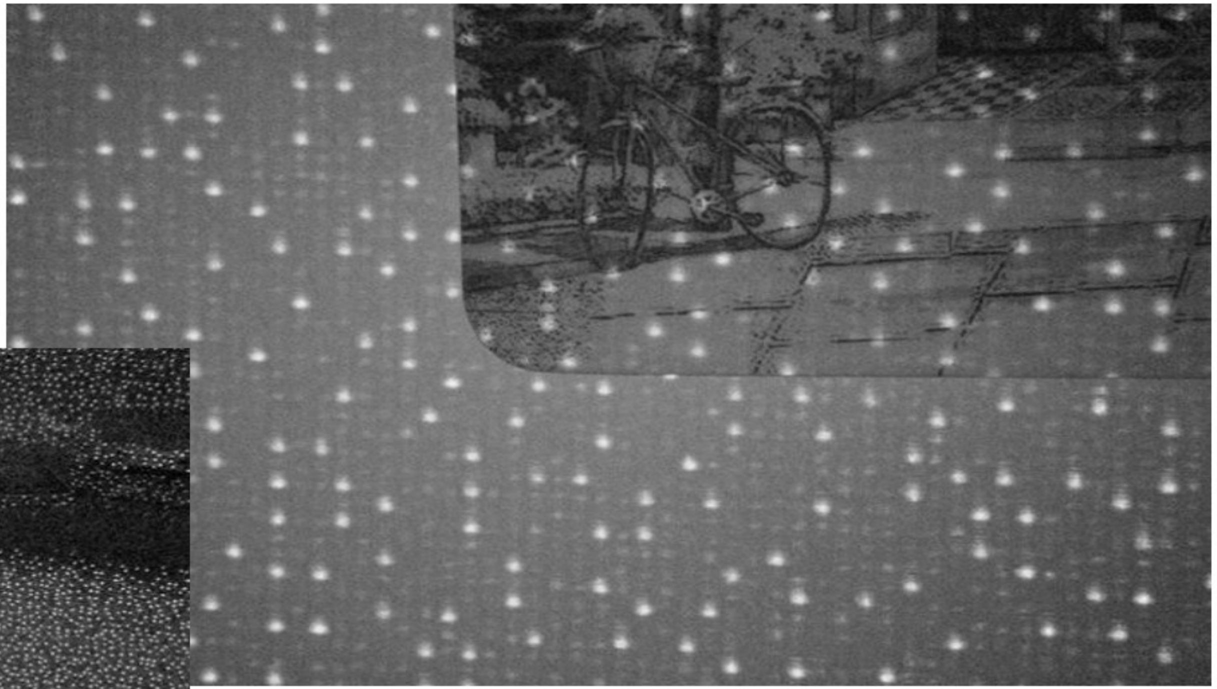
- Devices: Microsoft Kinect and Asus Xtion
- Developed by Israeli company PrimeSense in 2010



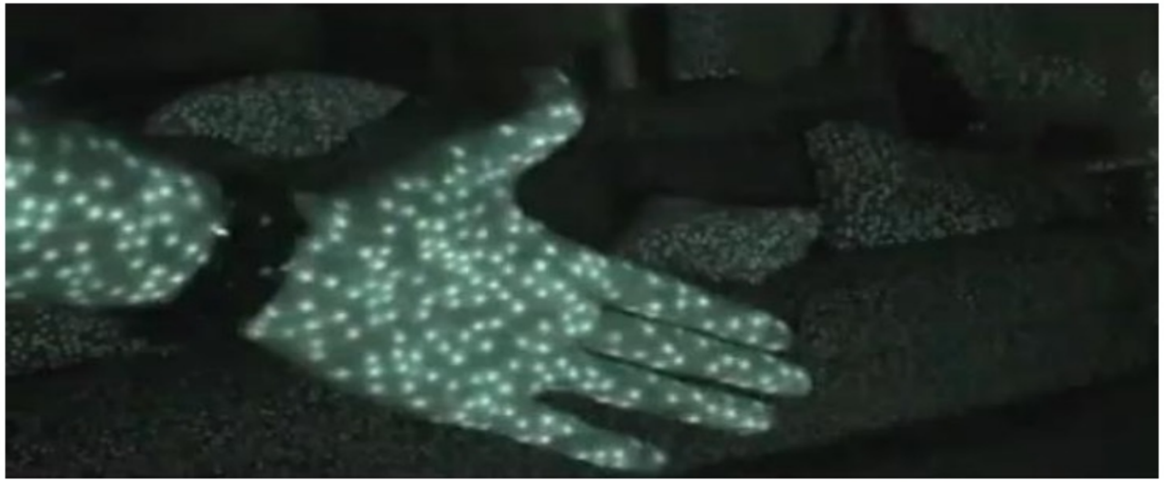
- Components:
  - IR camera (640 x 480 pixel)
  - IR Laser projector
  - RGB camera (640 x 480 or 1280 x 1024)
  - Field of View (FoV):
    - 57.5 degrees horizontally,
    - 43.5 degrees vertically

IR Pattern

Sensitive to infrared light
=>
Does NOT work outdoors!



Depth Map



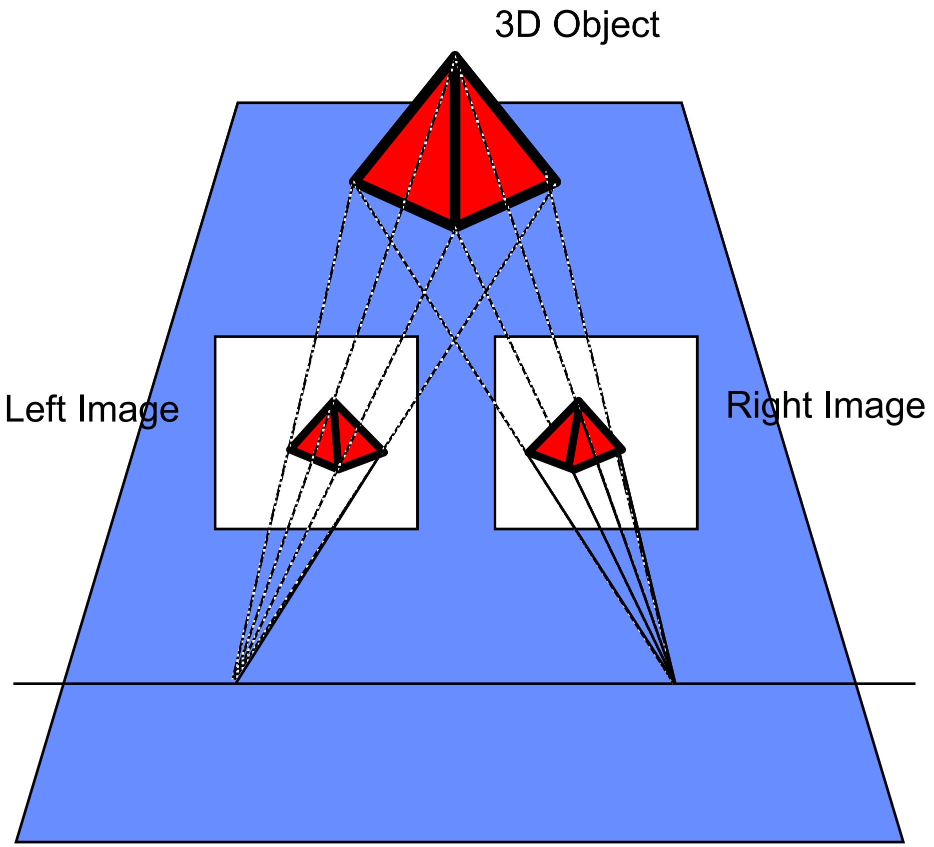
Microsoft Kinect: Depth Computation (1)

- **Depth from Stereo**
  - The Kinect uses an infrared projector and an infrared sensor; it does not use its RGB camera for depth computation
  - The technique of analyzing a known pattern is structured light
  - The IR projector projects a pseudo-random pattern across the surface of the room.
  - The direction of each speckle of the patter is known (from pre calibration during manufacturing) and is hardcoded into the memory of the Kinect
  - By measuring the position of each speckle in the IR image, its depth can be computed


RANGE SENSING: STEREO VISION

Stereo Cameras

- Theory will be covered in detail in Vision Lecture
- Estimate depth by using 2 cameras

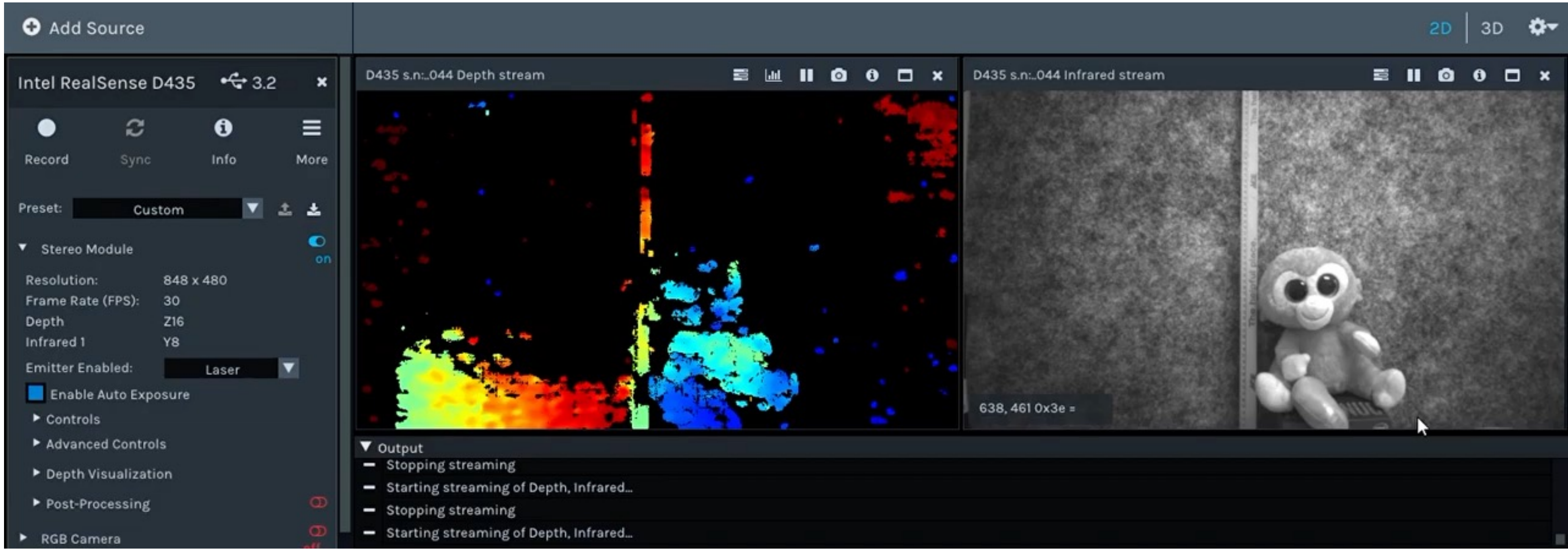


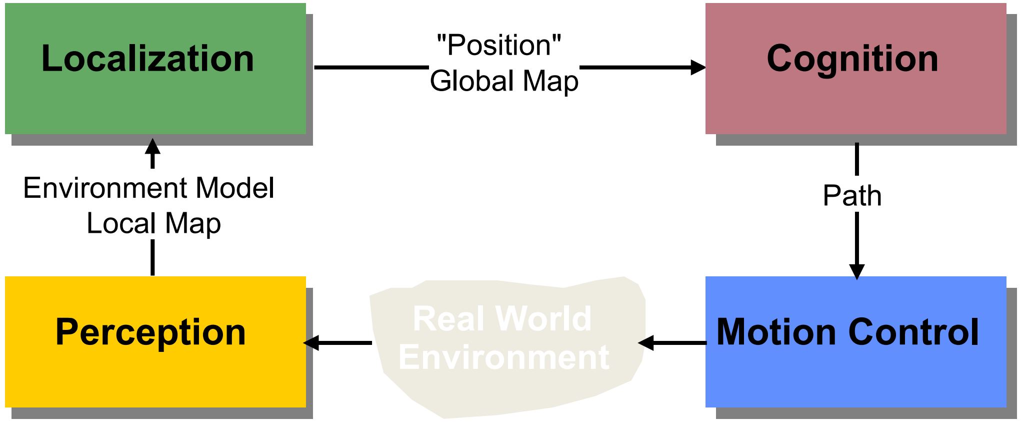
Stereo example: ZED camera

- Dual 4MP Camera @15Hz (lower resolution => higher fps)
- Up to 20m distance
- Passive Sensor
- Doesn’t work on single color surfaces (e.g. white wall)!

Intel RealSense D435

- Stereo Infrared – works indoors and outdoors
- Active pattern (e.g. for white wall) – only works indoors!
- Depth resolution: 1280 × 720
- Depth Field of View (FOV): 86° × 57° (±3°)
- RGB: 1920 × 1080
- Small and lightweight
- With IMU





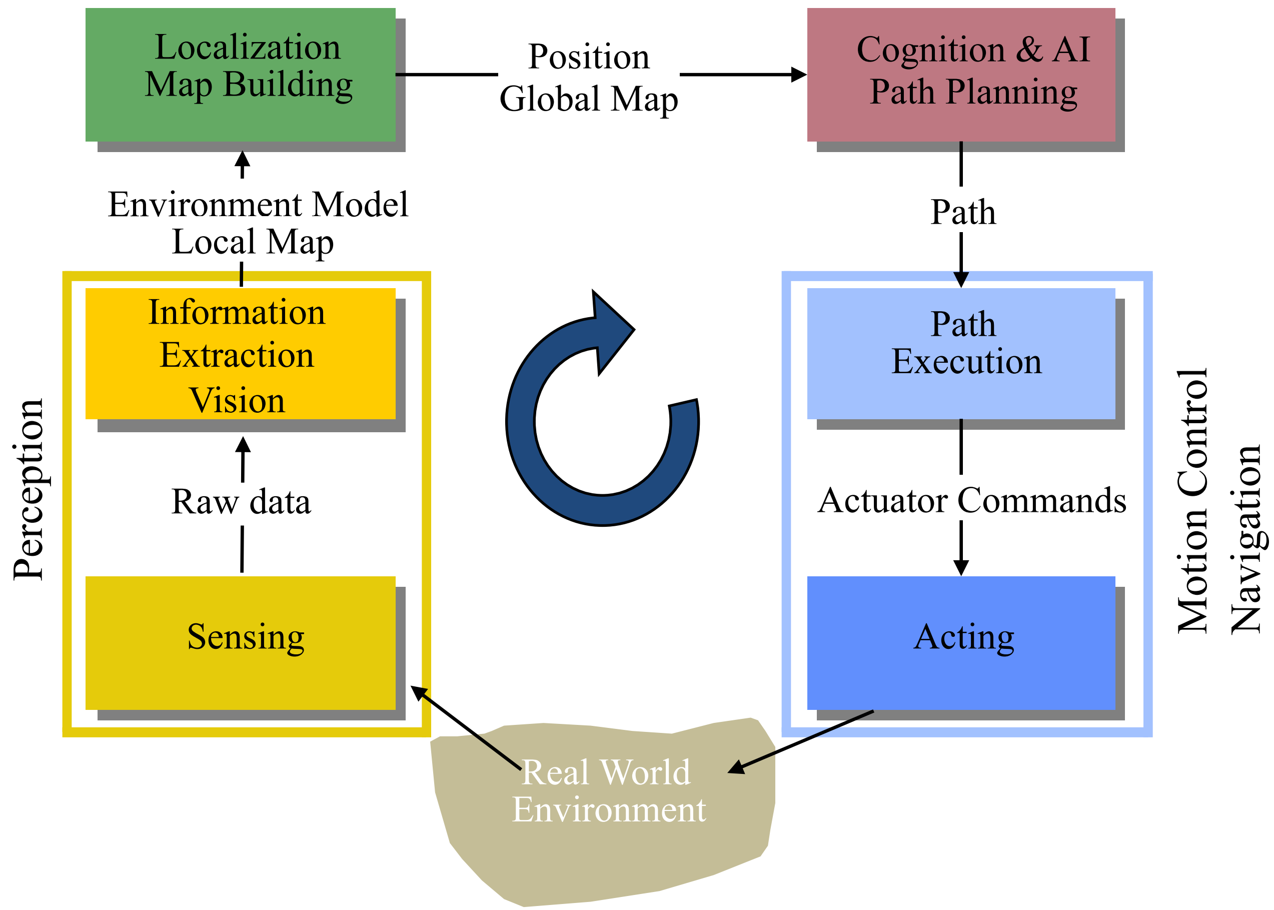
PERCEPTION

Line extraction from laser scans

Vision

Sildes from Roland Siegwart and Davide Scaramuzza, ETH Zurich

General Control Scheme for Mobile Robot Systems



LINE EXTRACTION

Split and merge
Linear regression
RANSAC
Hough-Transform

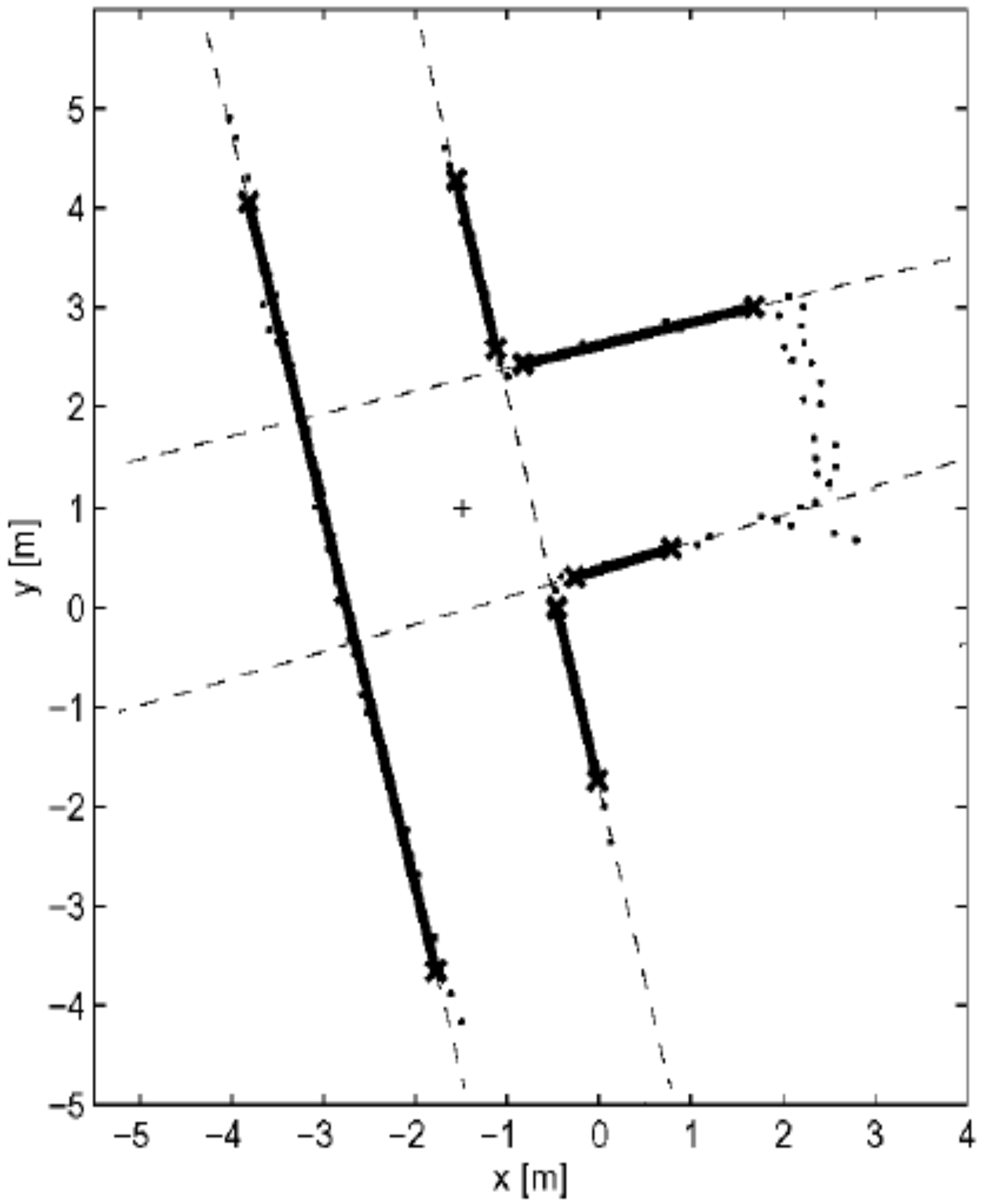
Line Extraction: Motivation

- Laser Range Scan
  - Example: 360 deg – black points
  - Example: dashed lines: desired line extractions

- Use detected lines for:
  - Scan registration (find out transform between frames of two consecutive LRF scans – change due to robot motion)

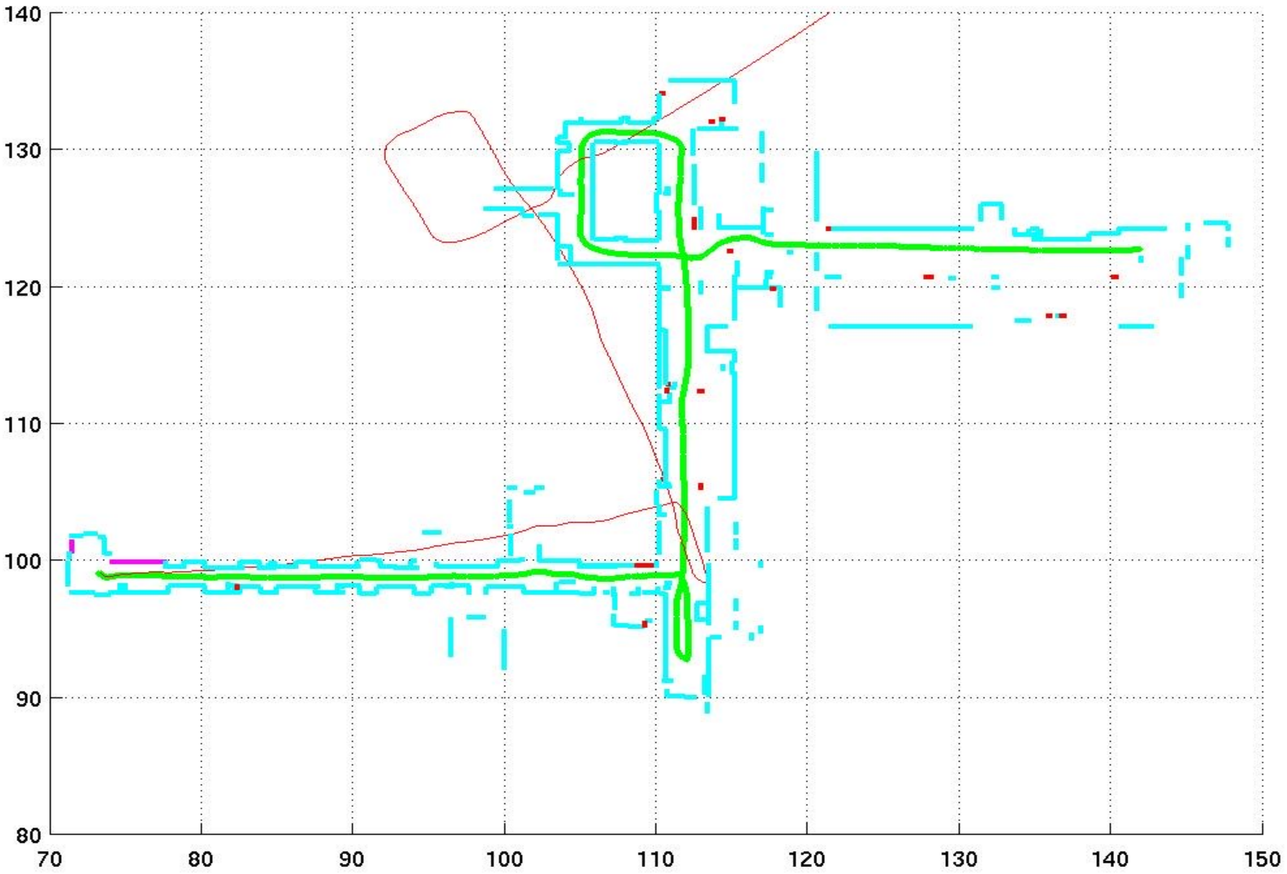
  OR

  - Mapping using line representation



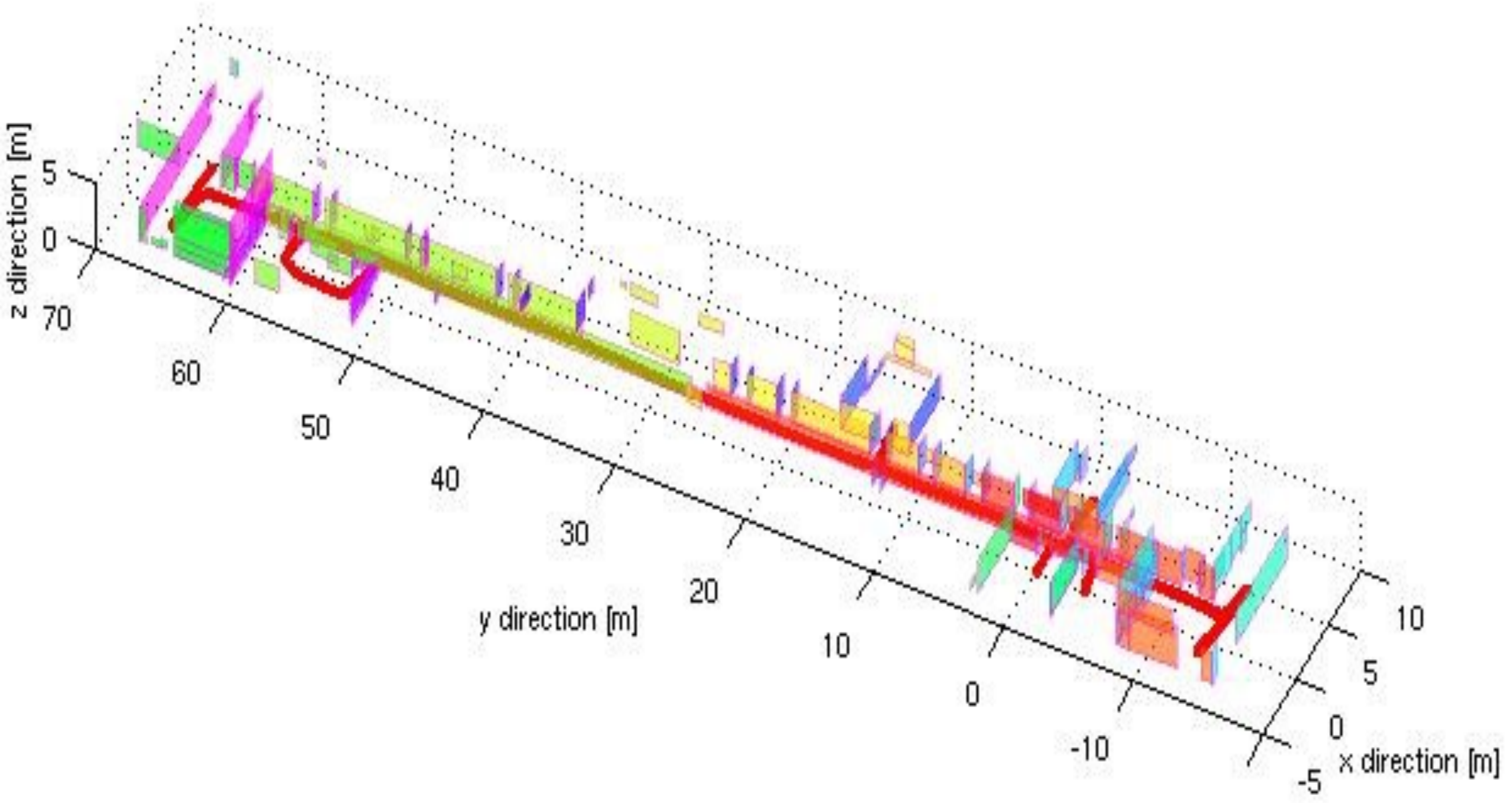
Line Extraction: Motivation

- Map of hallway built using line segments



Line Extraction: Motivation

- Map of the hallway built using orthogonal planes constructed from line segments

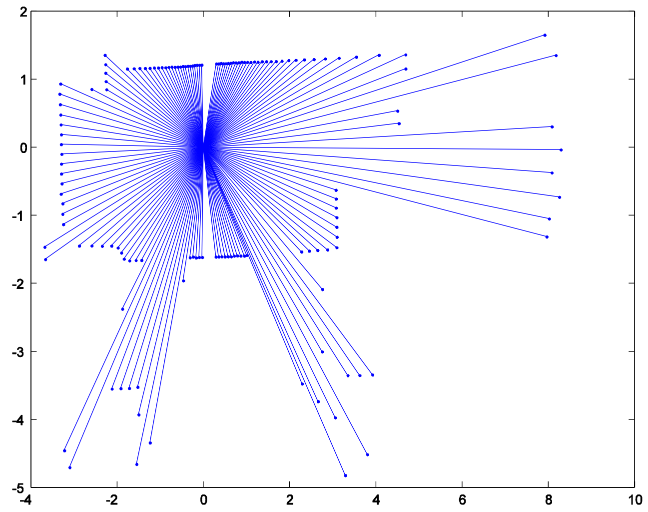


Line Extraction: Motivation

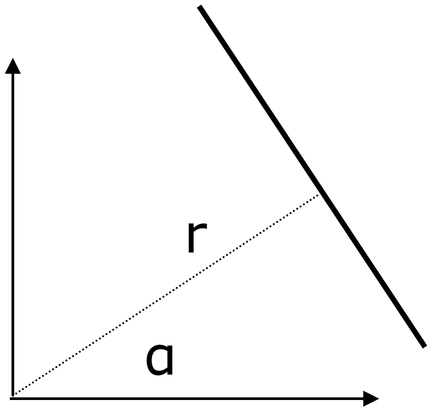
- Why laser scanner:
  - Dense and accurate range measurements
  - High sampling rate, high angular resolution
  - Good range distance and resolution.

- Why line segment:
  - The simplest geometric primitive
  - Compact, requires less storage
  - Provides rich and accurate information
  - Represents most office-like environment.

Line Extraction: The Problem



- Scan point in polar form: (ρ_i, θ_i)
- Assumptions:
  - Gaussian noise
  - Negligible angular uncertainty
- Line model in polar form:
  - $x \cos \alpha + y \sin \alpha = r$
  - $-\pi < \alpha <= \pi$
  - $r >= 0$



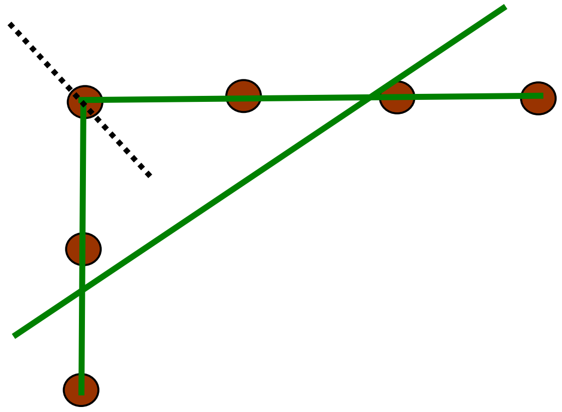
Line Extraction: The Problem (2)

- Three main problems:
  - How many lines ?
  - Which points belong to which line ?
    - This problem is called SEGMENTATION
  - Given points that belong to a line, how to estimate the line parameters ?
    - This problem is called LINE FITTING

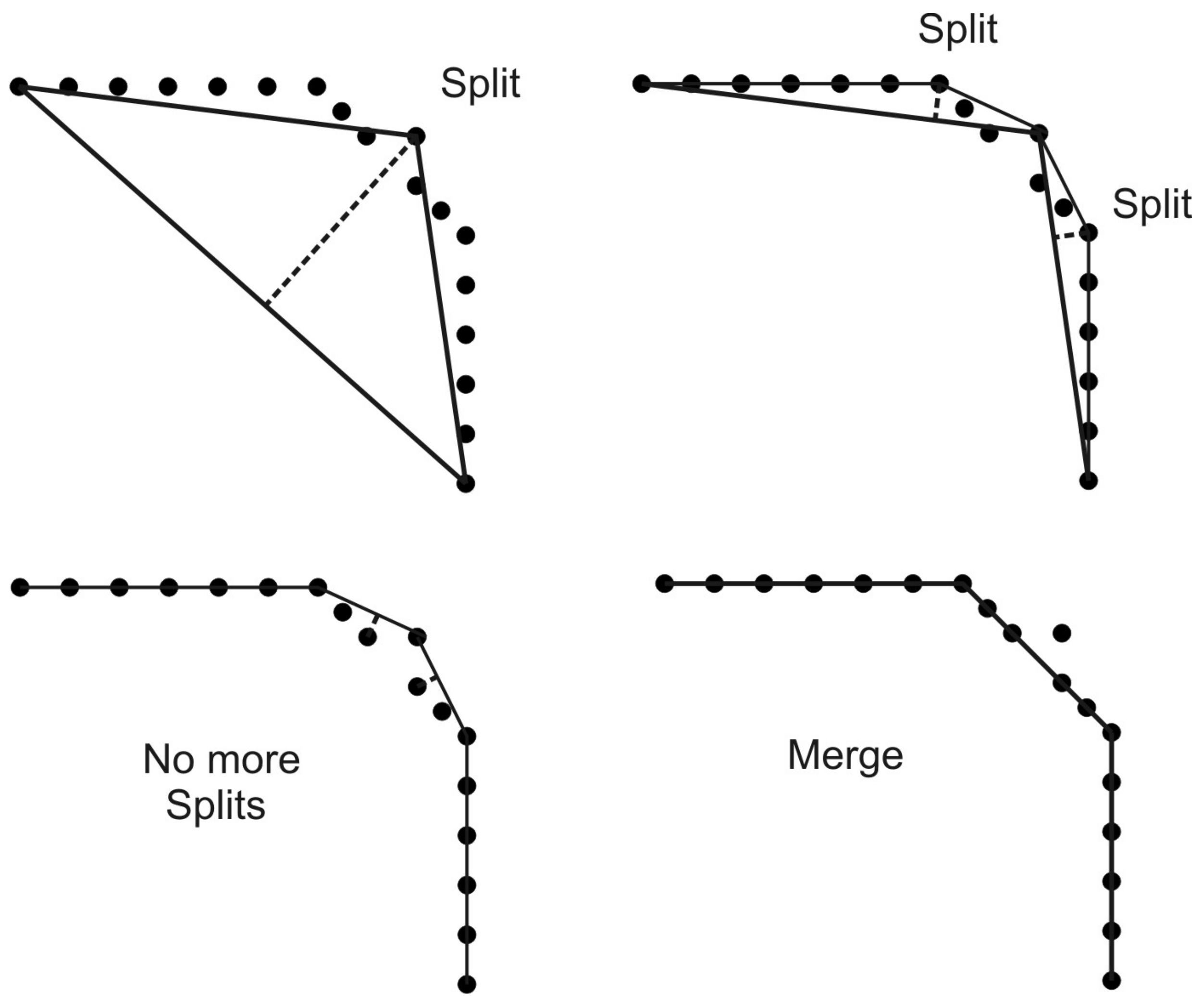
- The Algorithms we will see:
  1. Split and merge
  2. Linear regression
  3. RANSAC
  4. Hough-Transform

Algorithm 1: Split-and-Merge (standard)

- The most popular algorithm which is originated from computer vision.
- A recursive procedure of fitting and splitting.
- A slightly different version, called Iterative-End-Point-Fit, simply connects the end points for line fitting.



Algorithm 1: Split-and-Merge (Iterative-End-Point-Fit)

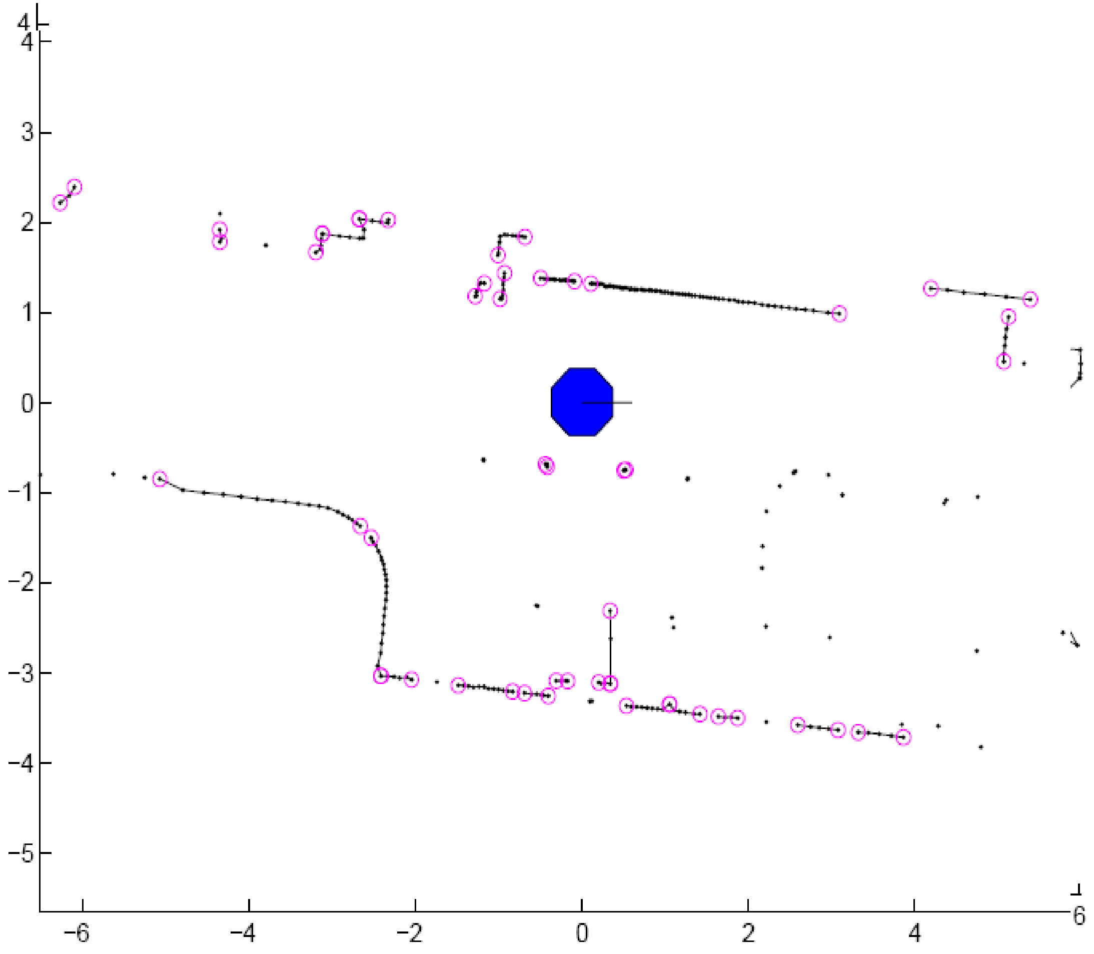


Algorithm 1: Split-and-Merge

Algorithm 1: *Split-and-Merge*

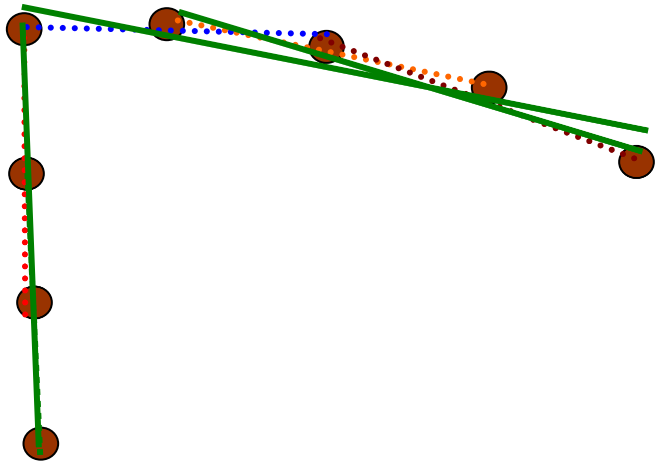
1. Initial: set s_1 consists of N points. Put s_1 in a list L
2. Fit a line to the next set s_i in L
3. Detect point P with maximum distance d_P to the line
4. If d_P is less than a threshold, continue (go to step 2)
5. Otherwise, split s_i at P into s_{i1} and s_{i2}, replace s_i in L by s_{i1} and s_{i2}, continue (go to 2)
6. When all sets (segments) in L have been checked, merge collinear segments.

Algorithm 1: Split-and-Merge: Example application



Algorithm 2: Line-Regression

- Uses a “sliding window” of size *Nf*
- The points within each “sliding window” are fitted by a segment
- Then adjacent segments are merged if their line parameters are close



$Nf = 3$

Algorithm 2: Line-Regression

Algorithm 2: *Line-Regression*

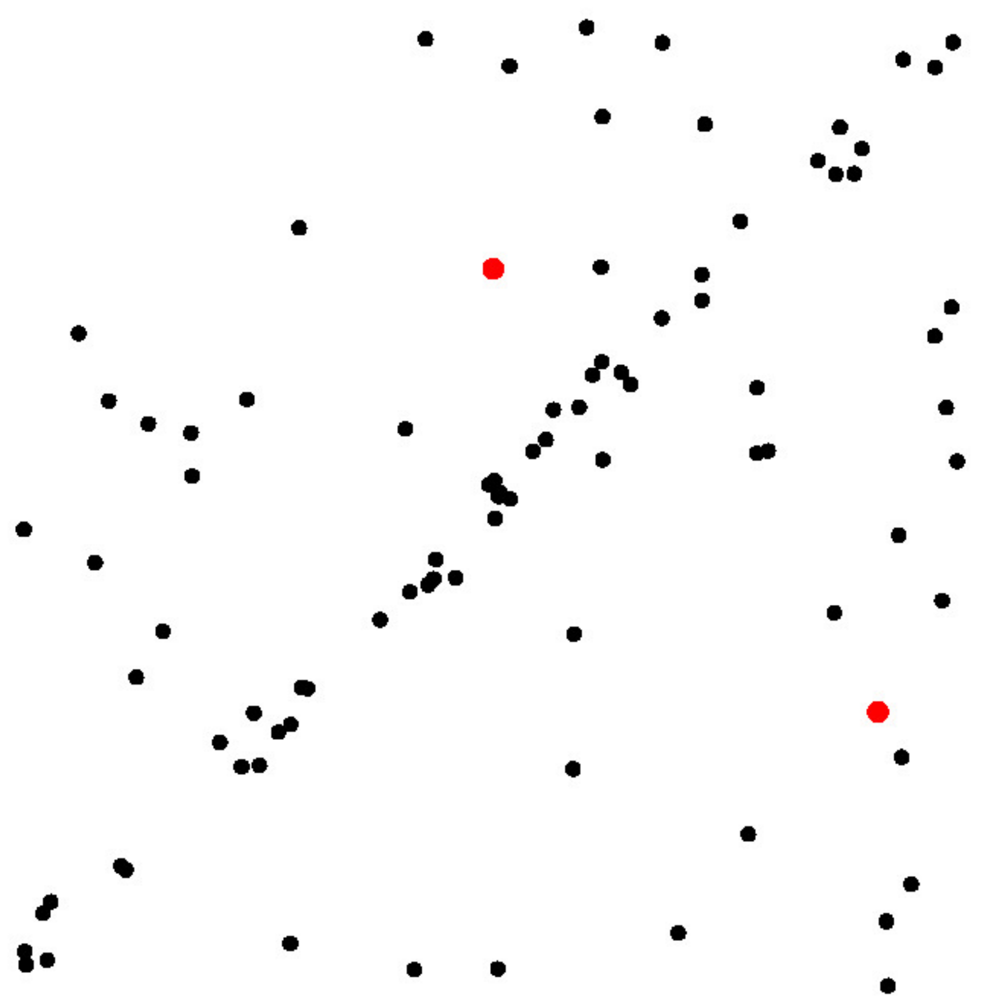
1. Initialize sliding window size N_f
2. Fit a line to every N_f consecutive points (a window)
3. Compute a line fidelity array, each is the sum of Mahalanobis distances between every three adjacent windows
4. Construct line segments by scanning the fidelity array for consecutive elements having values less than a threshold, using an AHC algorithm
5. Merge overlapped line segments and recompute line parameters for each segment

Algorithm 3: RANSAC

- Acronym: **<u>Ran</u>**dom **<u>Sa</u>**mple **<u>C</u>**onsensus.
- Generic & robust fitting algorithm of models with outliers
  - Outliers: points which do not satisfy a model
- RANSAC: apply to any problem where:
  - identify the inliers
  - which satisfy a predefined mathematical model.
- Typical robotics applications:
  - line extraction from 2D range data (sonar or laser);
  - plane extraction from 3D range data
  - structure from motion
- RANSAC:
  - iterative method & non-deterministic
- <u>Drawback: A nondeterministic method, results are different between runs.</u>

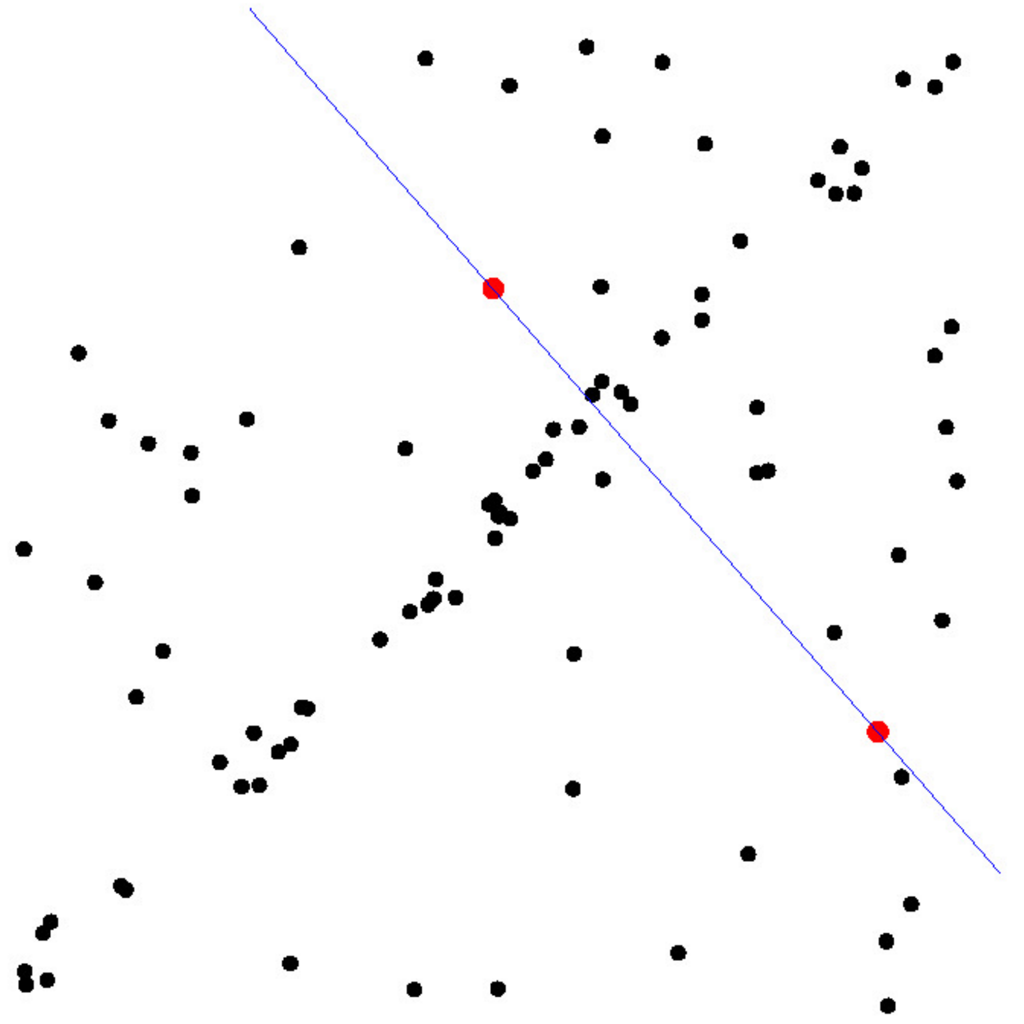
Algorithm 3: RANSAC

Algorithm 3: RANSAC



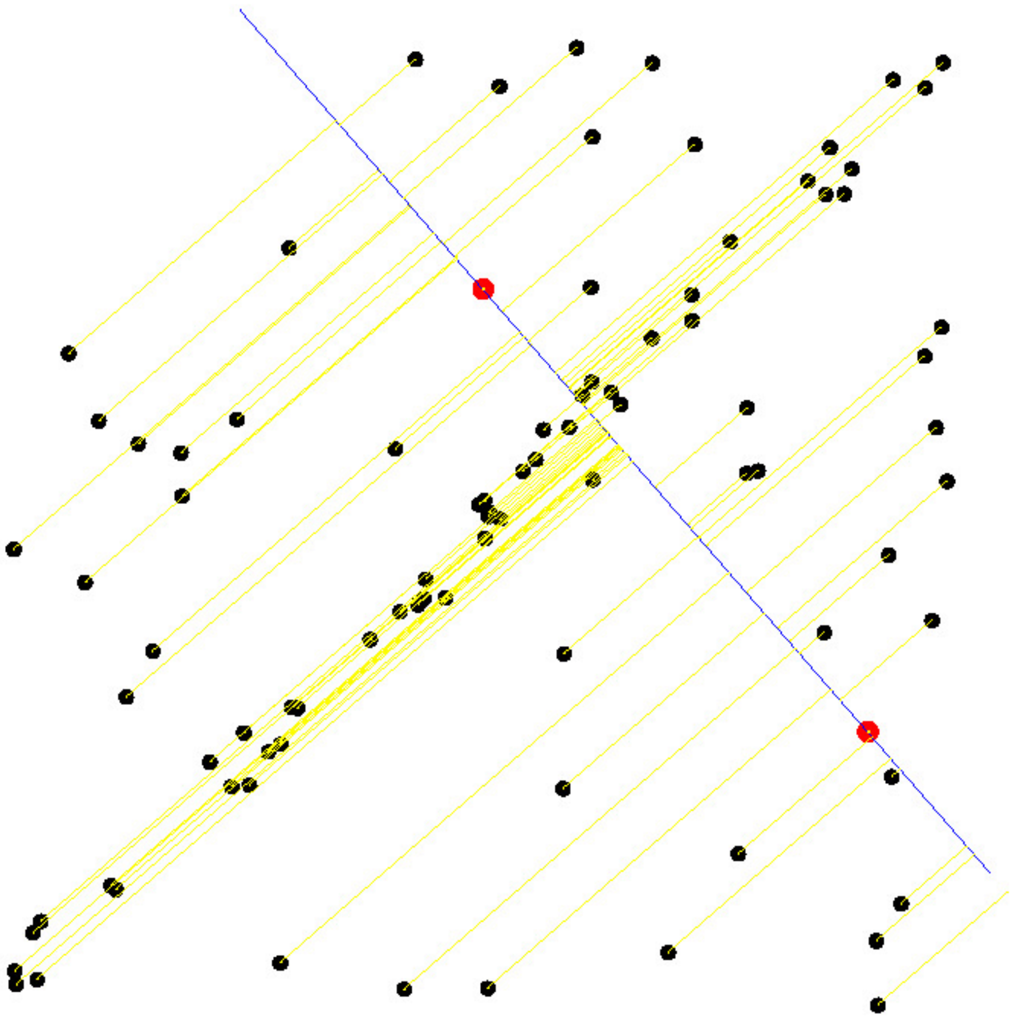
- **Select sample of 2 points at random**

Algorithm 3: RANSAC



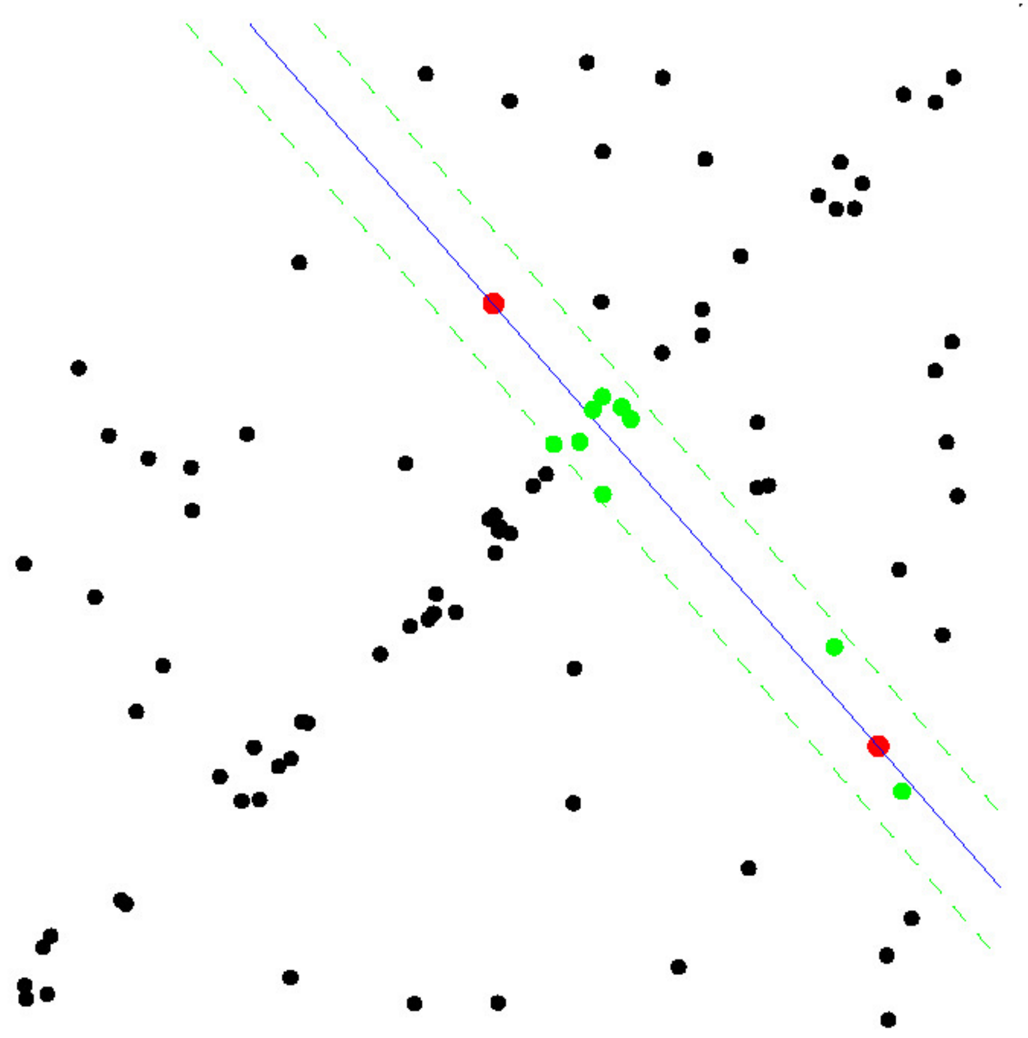
- Select sample of 2 points at random
- **Calculate model parameters that fit the data in the sample**

RANSAC



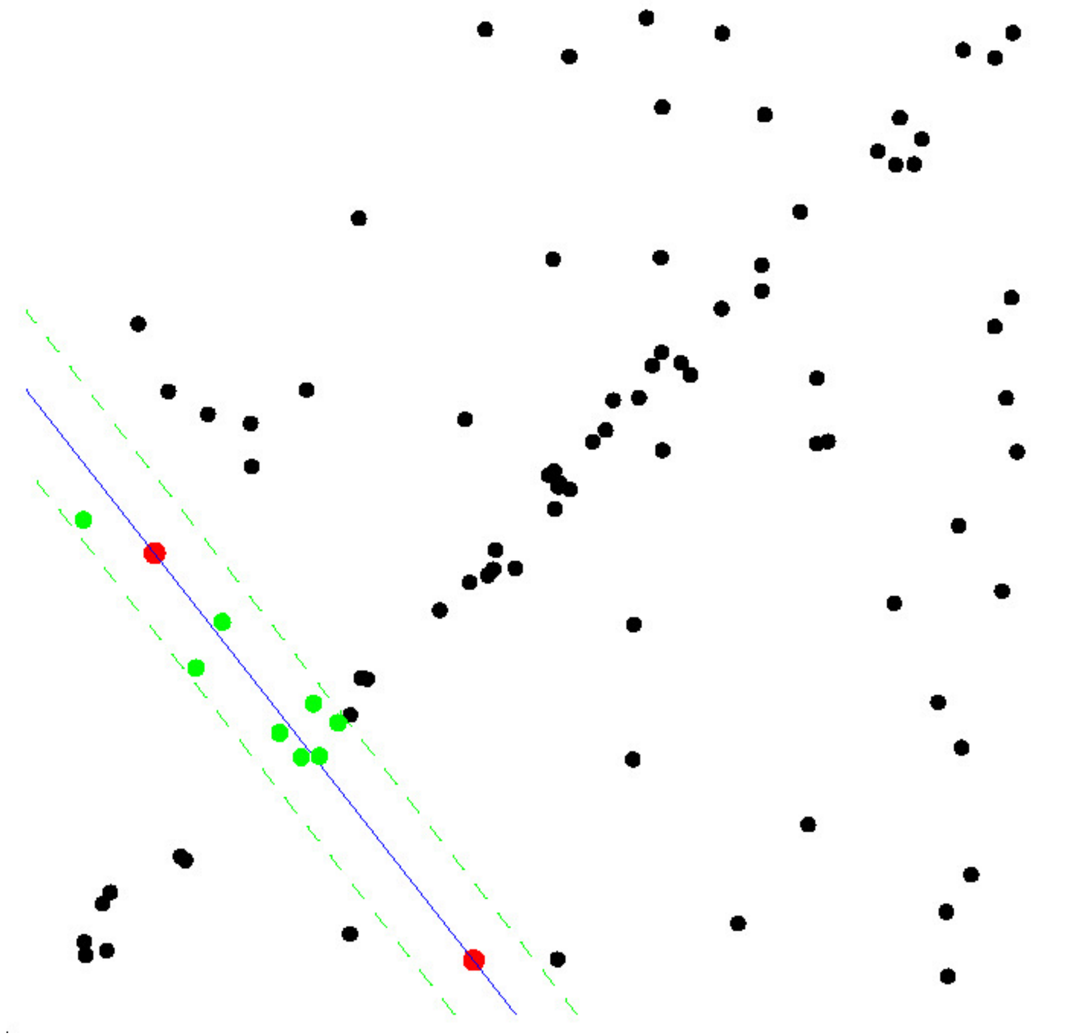
- Select sample of 2 points at random
- Calculate model parameters that fit the data in the sample
- **Calculate error function for each data point**

Algorithm 3: RANSAC



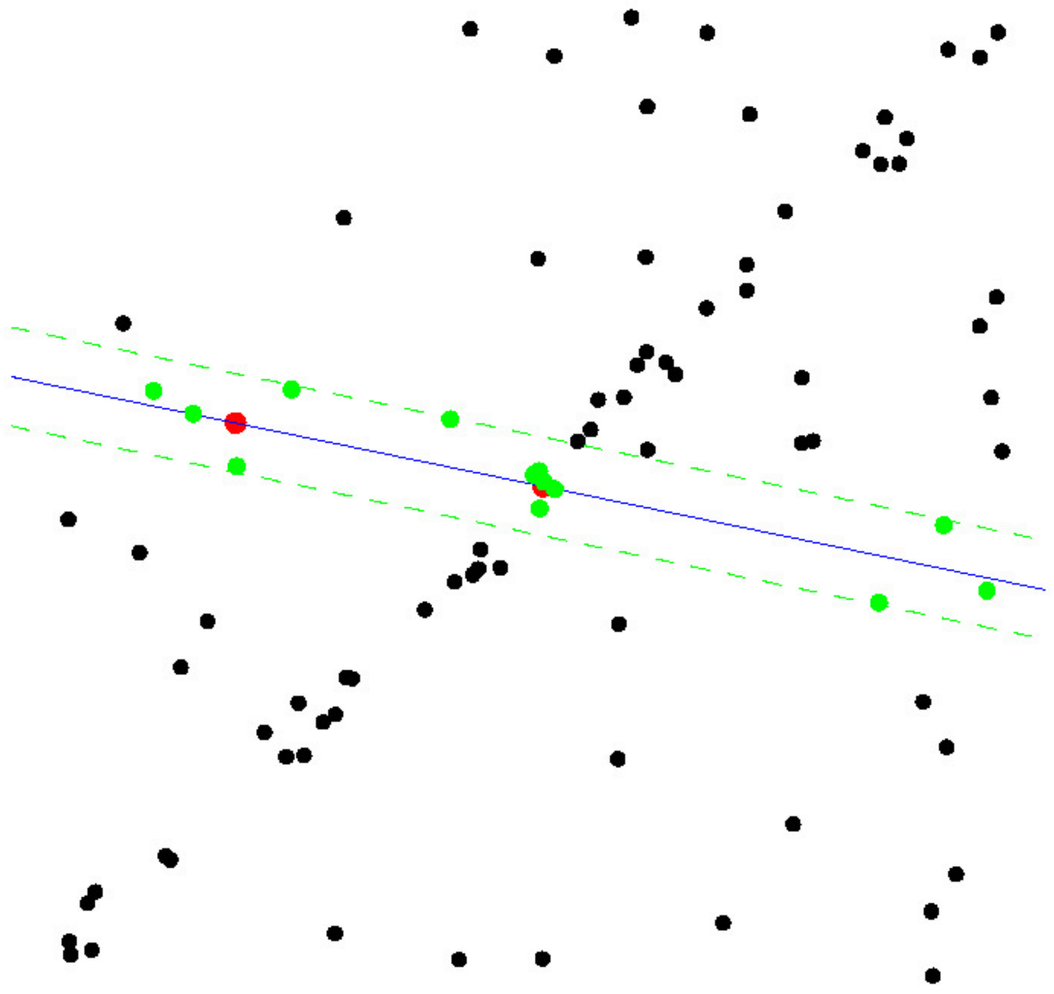
- Select sample of 2 points at random
- Calculate model parameters that fit the data in the sample
- Calculate error function for each data point
- **Select data that support current hypothesis**

Algorithm 3: RANSAC



- Select sample of 2 points at random
- Calculate model parameters that fit the data in the sample
- Calculate error function for each data point
- Select data that support current hypothesis
- **Repeat sampling**

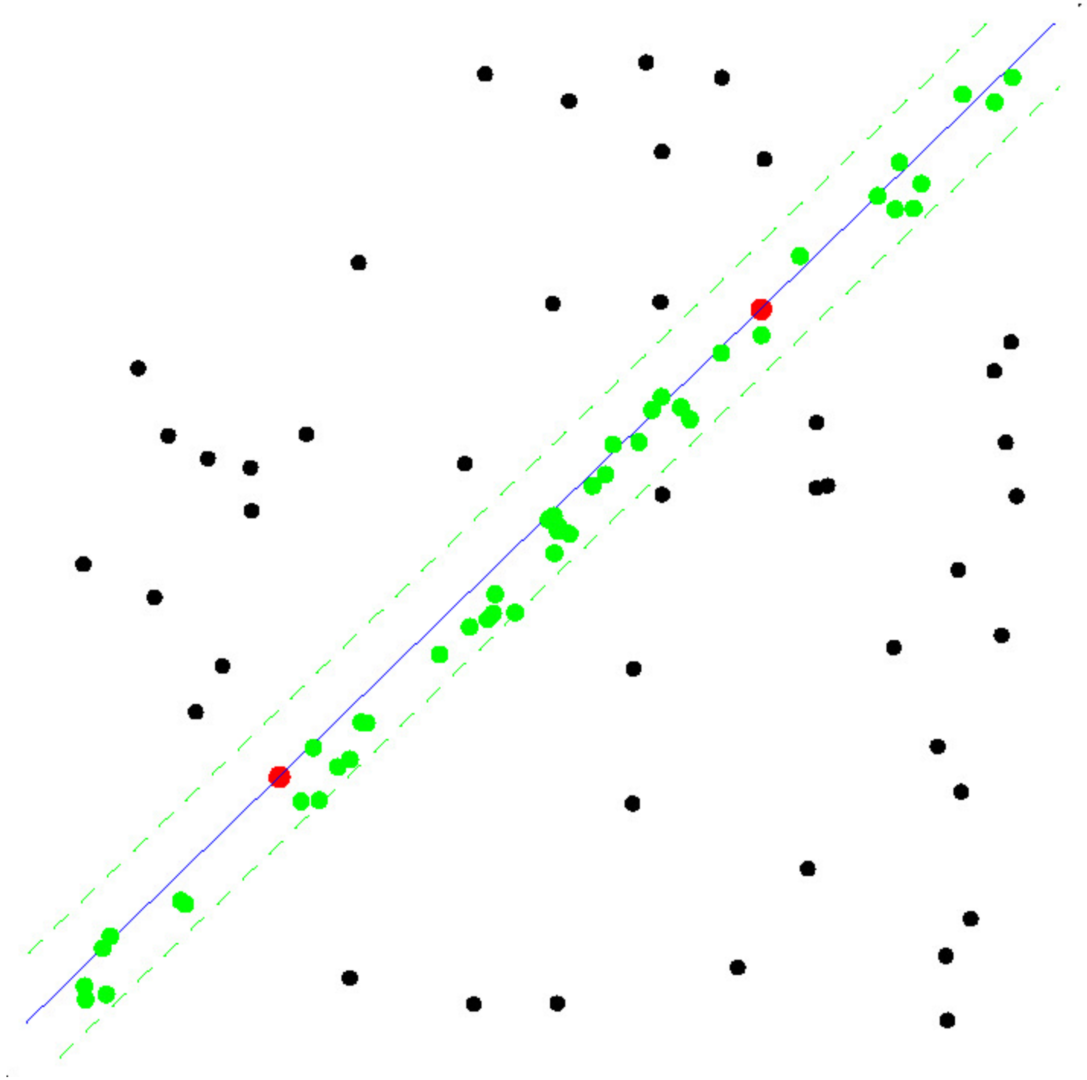
Algorithm 3: RANSAC



- Select sample of 2 points at random
- Calculate model parameters that fit the data in the sample
- Calculate error function for each data point
- Select data that support current hypothesis
- **Repeat sampling**

Algorithm 3: RANSAC

ALL-INLIER SAMPLE



Algorithm 3: RANSAC

Algorithm 4: *RANSAC*

1. Initial: let A be a set of N points
2. **repeat**
3. Randomly select a sample of 2 points from A
4. Fit a line through the 2 points
5. Compute the distances of all other points to this line
6. Construct the inlier set (i.e. count the number of points with distance to the line $< d$)
7. Store these inliers
8. **until** Maximum number of iterations k reached
9. The set with the maximum number of inliers is chosen as a solution to the problem

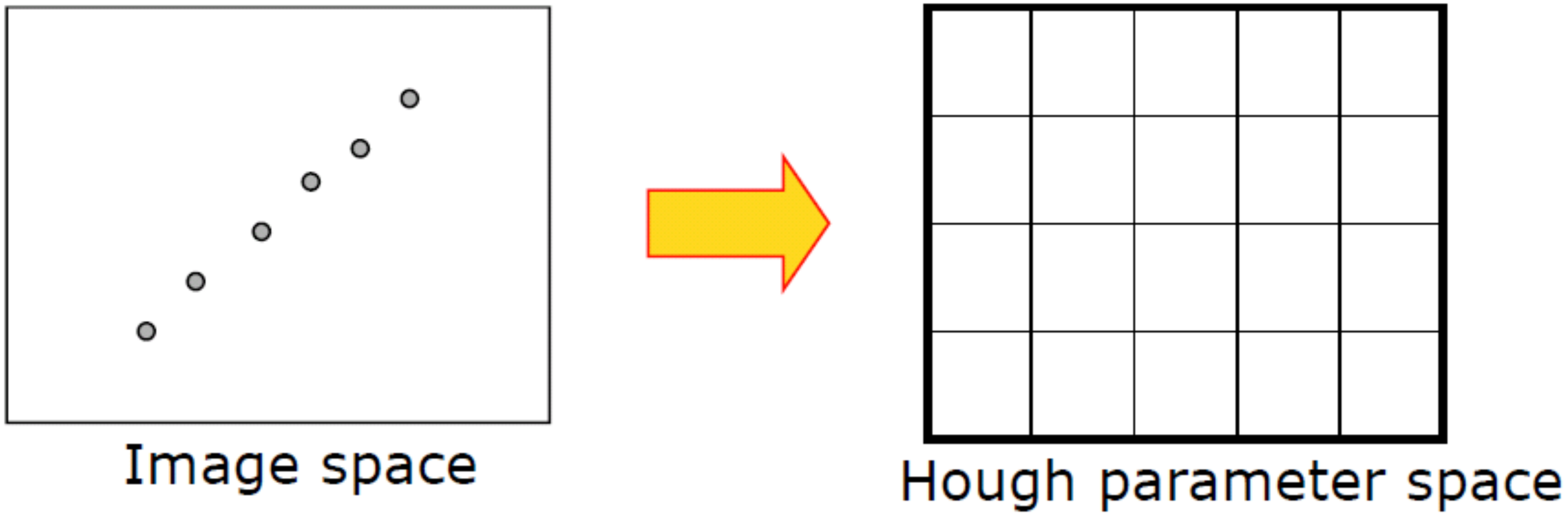
Algorithm 3: RANSAC

How many iterations does RANSAC need?

- Because we cannot know in advance if the observed set contains the maximum number of inliers, the ideal would be to check all possible combinations of 2 points in a dataset of N points.
- The number of combinations is given by N(N-1)/2, which makes it computationally unfeasible if N is too large. For example, in a laser scan of 360 points we would need to check all 360*359/2= 64,620 possibilities!
- Do we really need to check all possibilities or can we stop RANSAC after iterations? The answer is that indeed we do not need to check all combinations but just a subset of them if we have a rough estimate of the percentage of inliers in our dataset
- This can be done in a probabilistic way

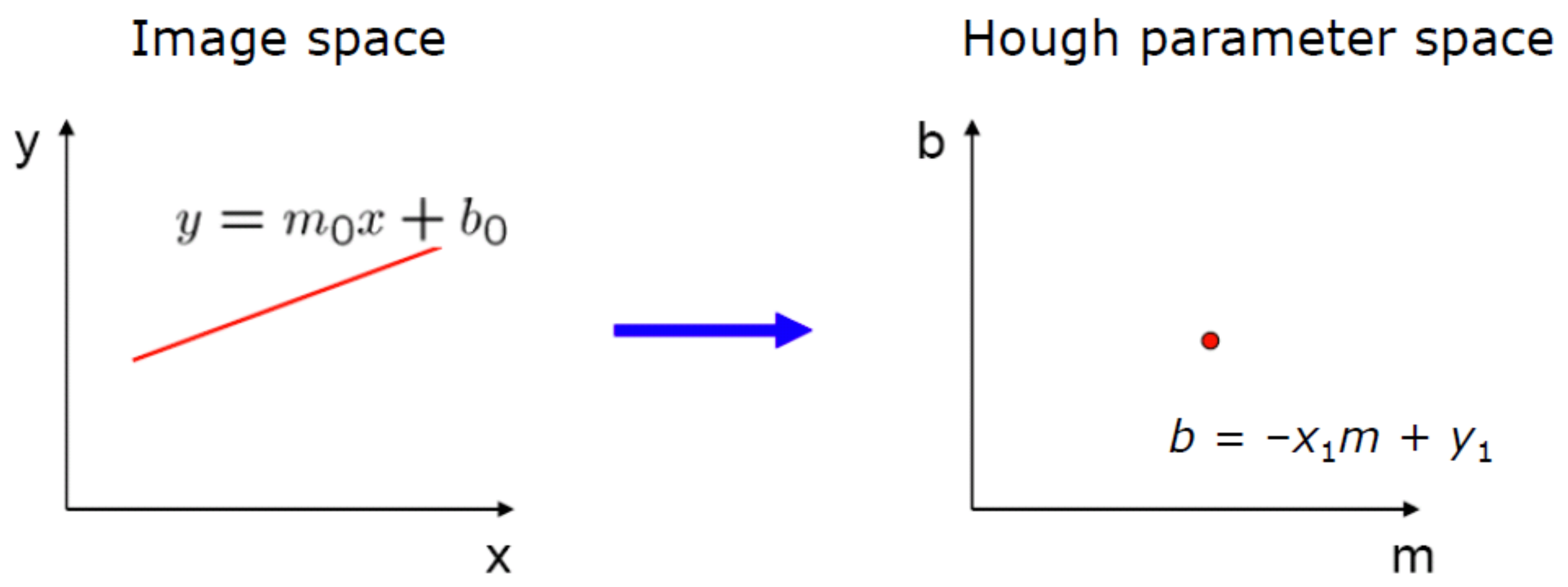
Algorithm 4: Hough-Transform

- Hough Transform uses a voting scheme



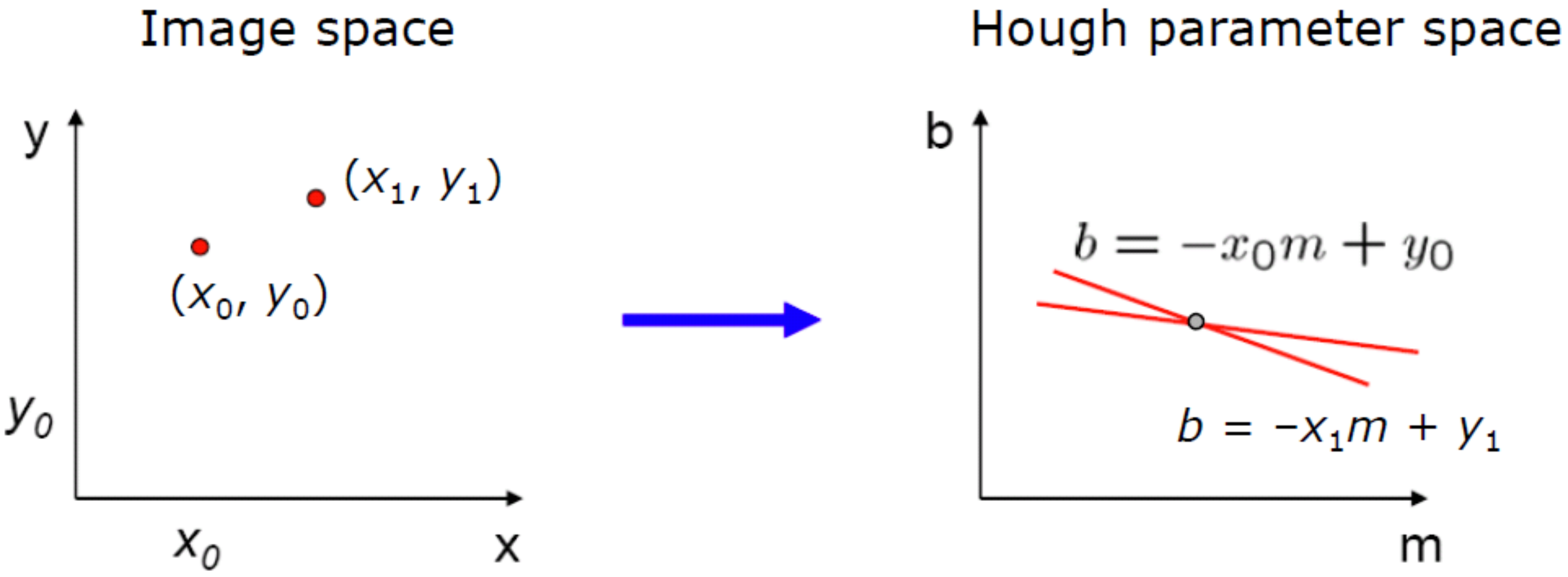
Algorithm 4: Hough-Transform

- A line in the image corresponds to a point in Hough space



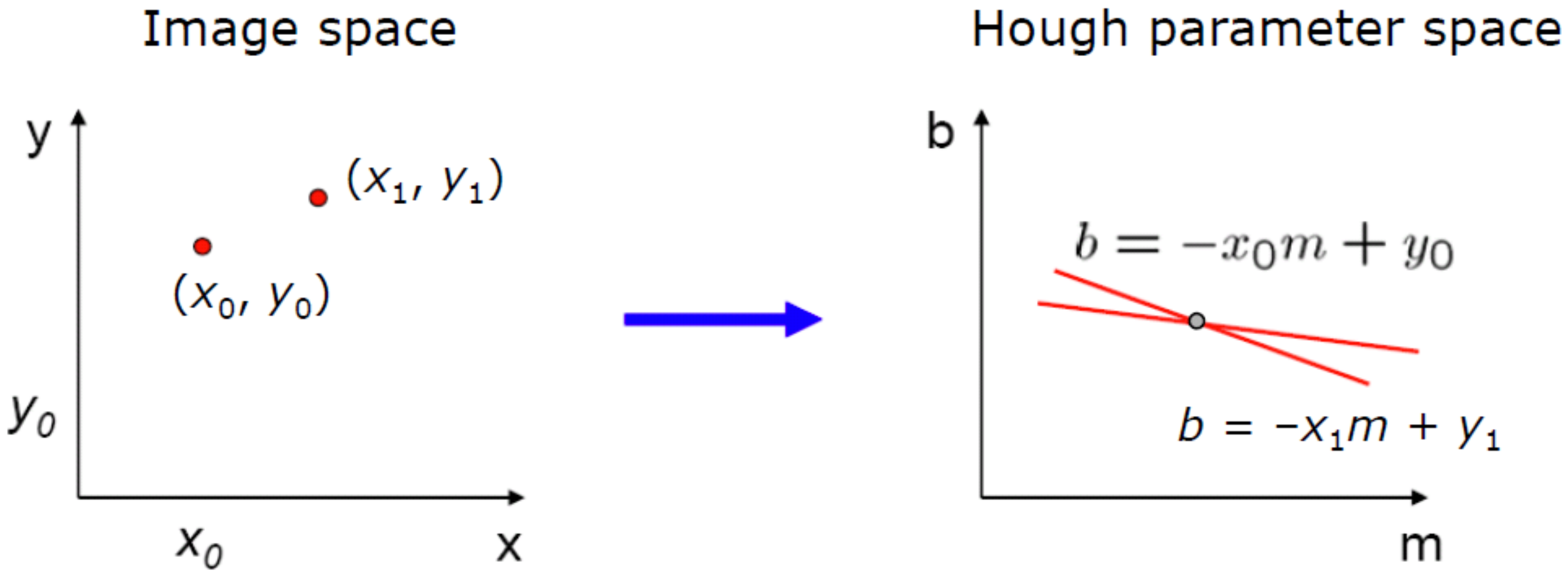
Algorithm 4: Hough-Transform

- What does a point (x_0, y_0) in the image space map to in the Hough space?



Algorithm 4: Hough-Transform

- Where is the line that contains both (x_0, y_0) and (x_1, y_1)?
  - It is the intersection of the lines $b = –x_0m + y_0$ and $b = –x_1m + y_1$



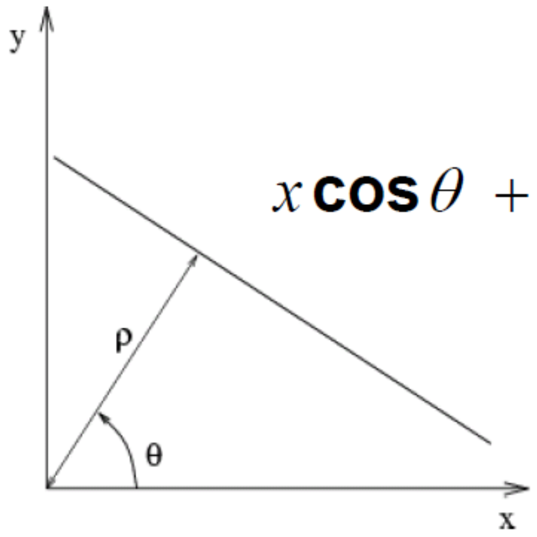
Algorithm 4: Hough-Transform

- Problems with the (m,b) space:
  - Unbounded parameter domain
  - Vertical lines require infinite m

Algorithm 4: Hough-Transform

- Problems with the (m,b) space:
  - Unbounded parameter domain
  - Vertical lines require infinite m

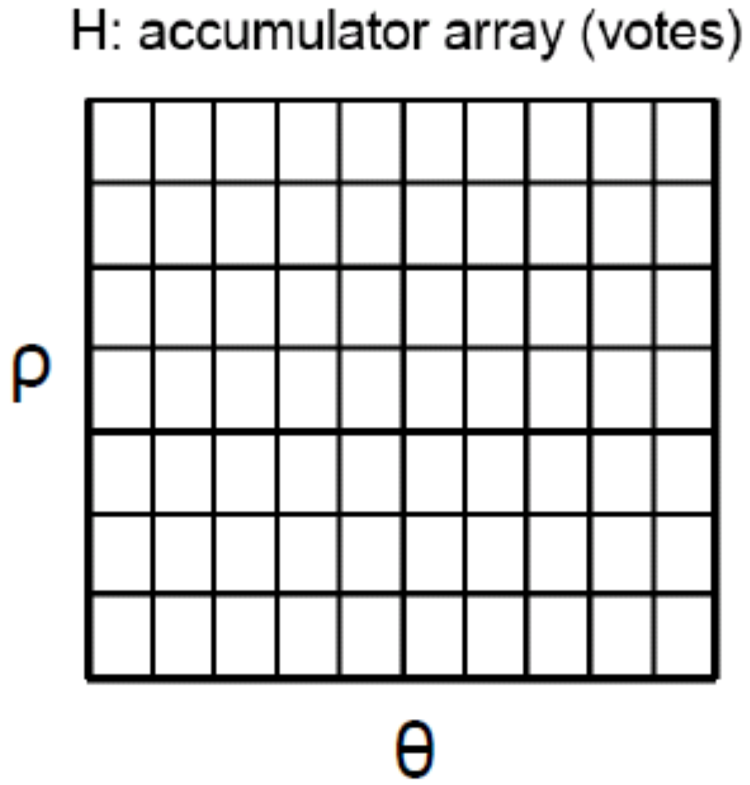
- Alternative: polar representation



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

Each point will add a sinusoid in the (θ,ρ) parameter space

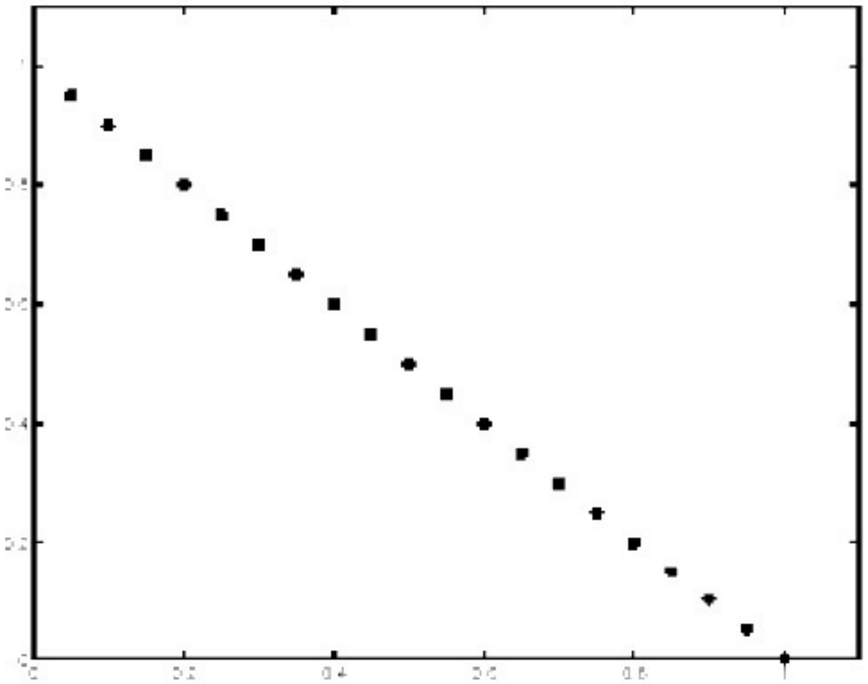
Algorithm 4: Hough-Transform



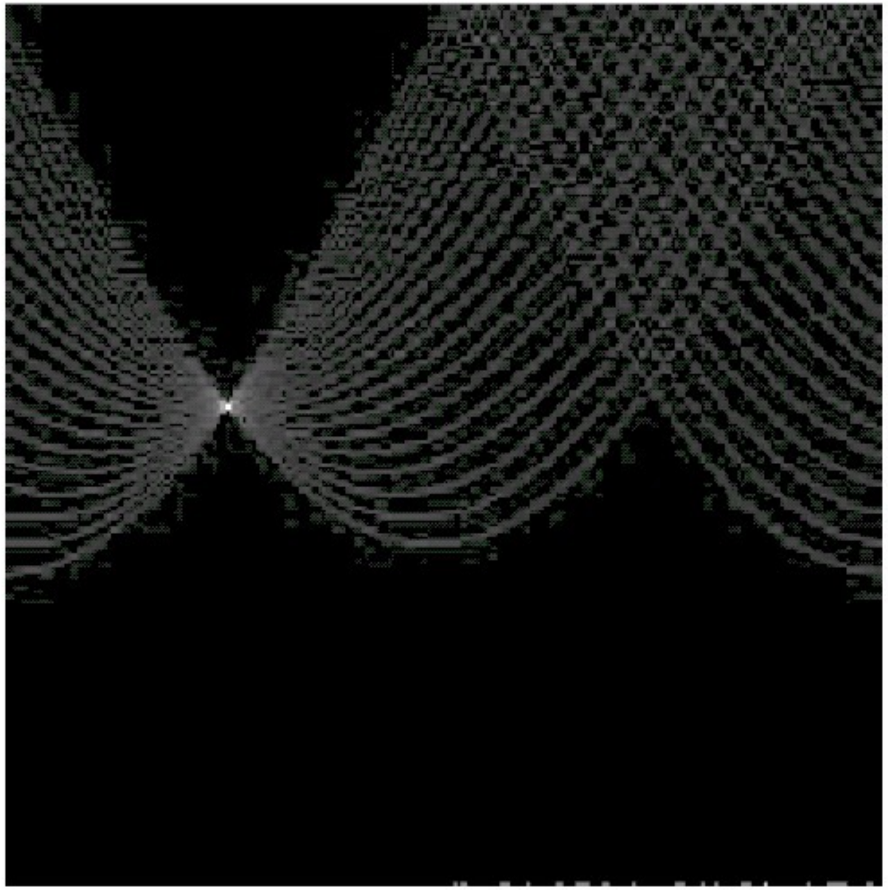
1. Initialize accumulator H to all zeros
2. For each edge point (x,y) in the image
   - For $\theta = 0$ to 180 (with a step size of e.g. 18)
     - $\rho = x \cos\theta + y \sin\theta$
     - $H(\theta, \rho) = H(\theta, \rho) + 1$
   - end

   end
3. Find the values of (θ, ρ) where $H(\theta, \rho)$ is a local maximum
4. The detected line in the image is given by $\rho = x \cos\theta + y \sin\theta$

Algorithm 4: Hough-Transform

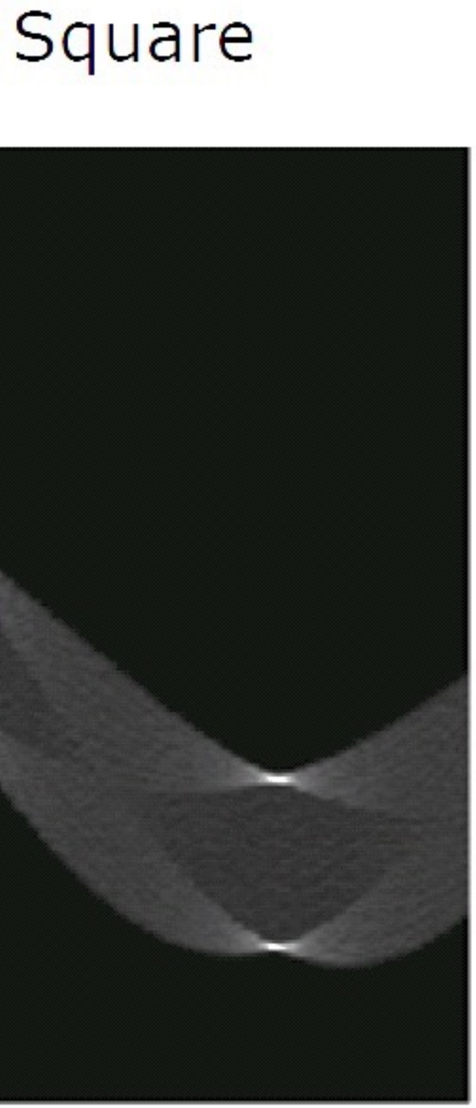


features

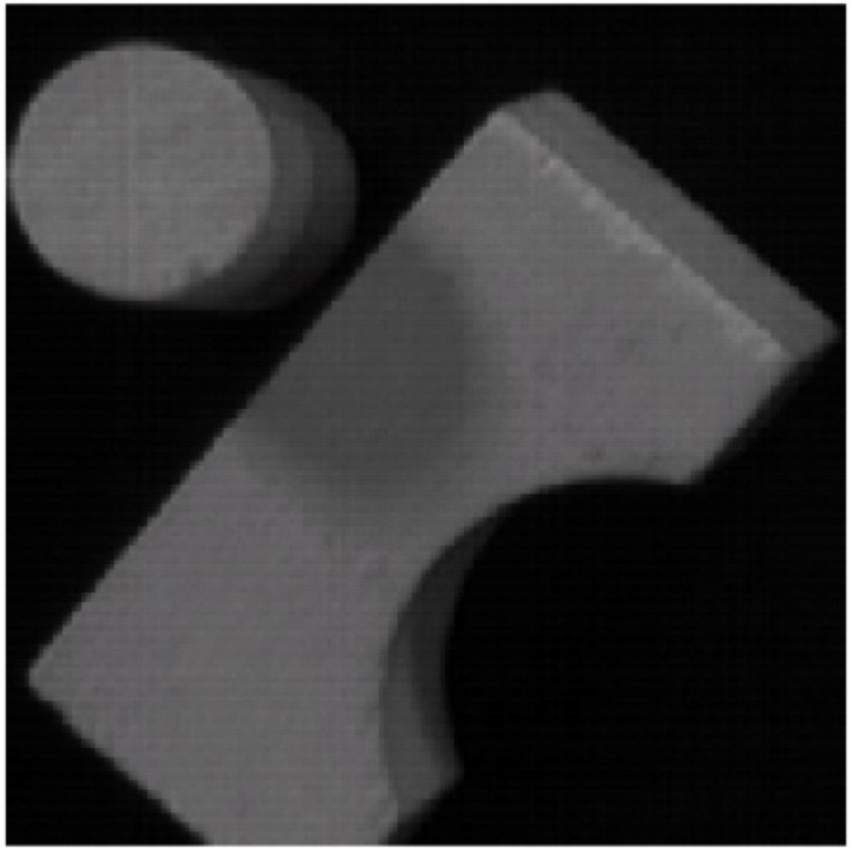


votes

Algorithm 4: Hough-Transform

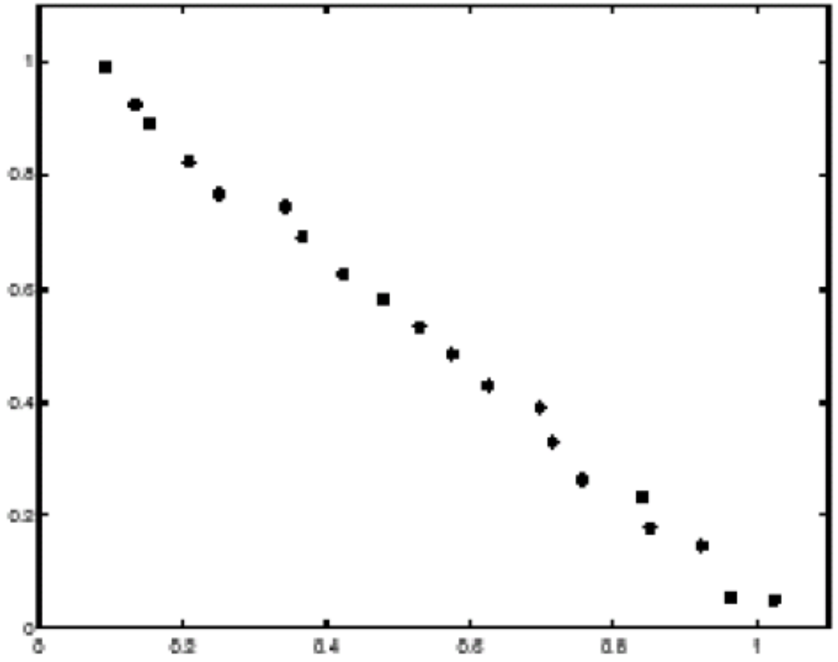


Algorithm 4: Hough-Transform

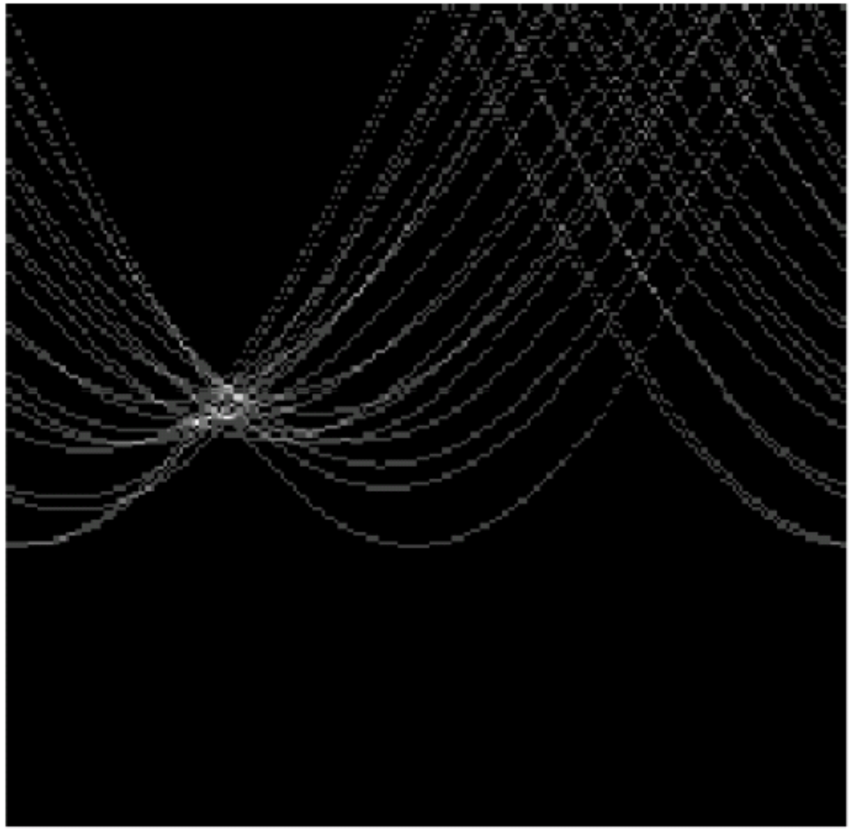


Algorithm 4: Hough-Transform

Effect of Noise



features

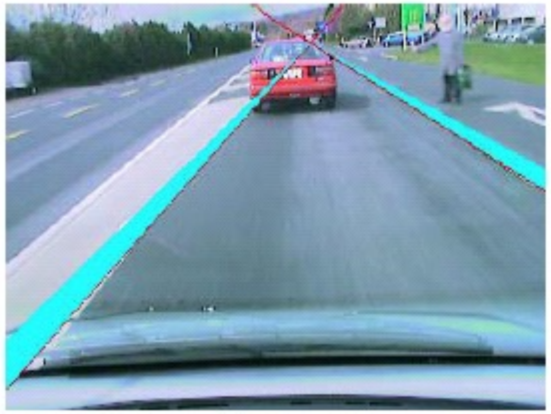


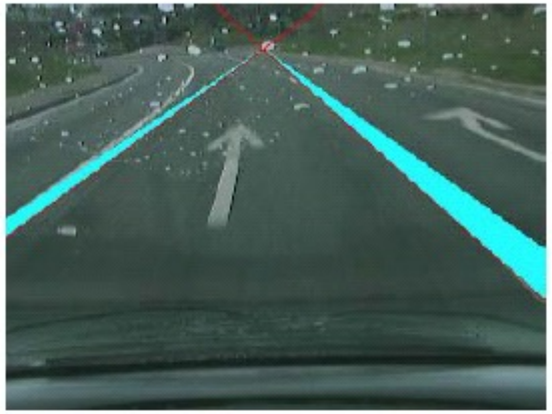
votes

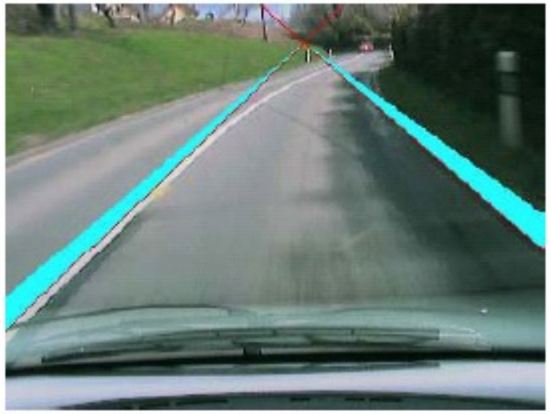
- Peak gets fuzzy and hard to locate

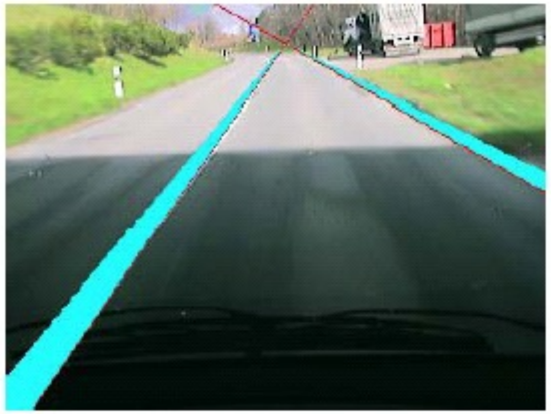
Algorithm 4: Hough-Transform

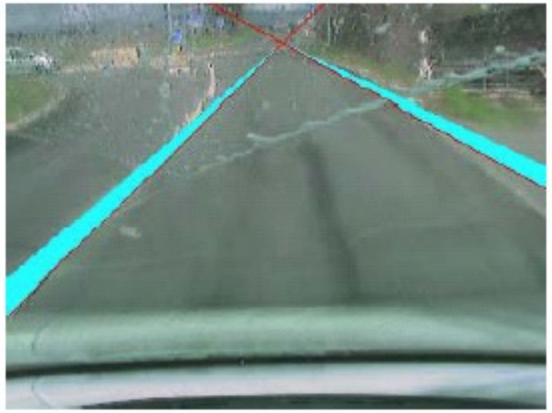
Application: Lane detection

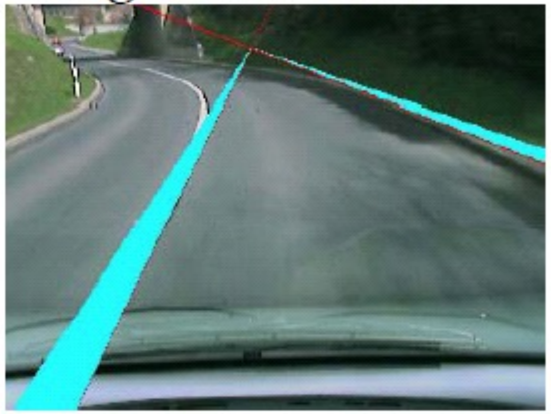

Inner city traffic


Ground signs

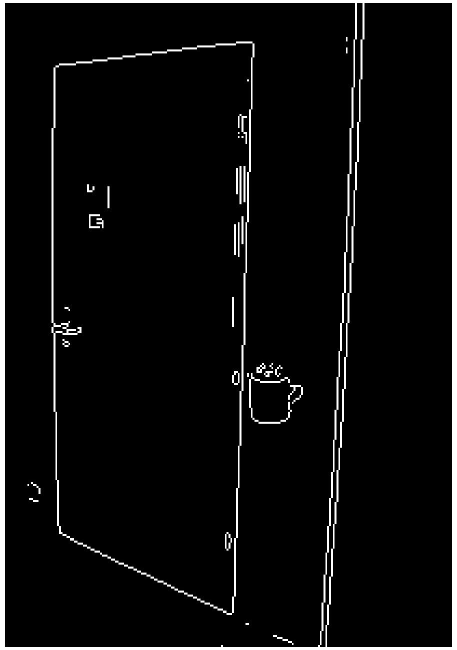

Country-side lane

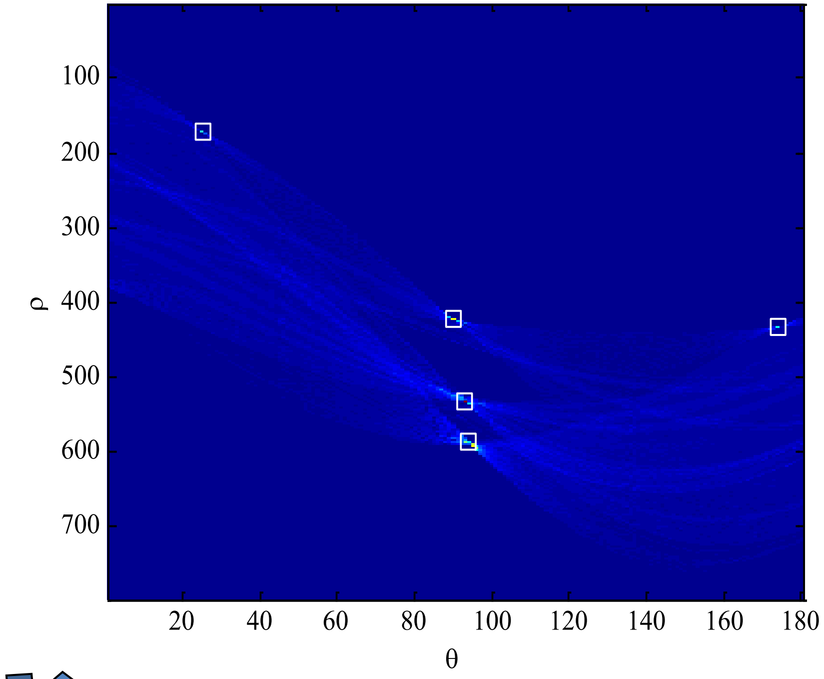

Tunnel exit


Obscured windscreen

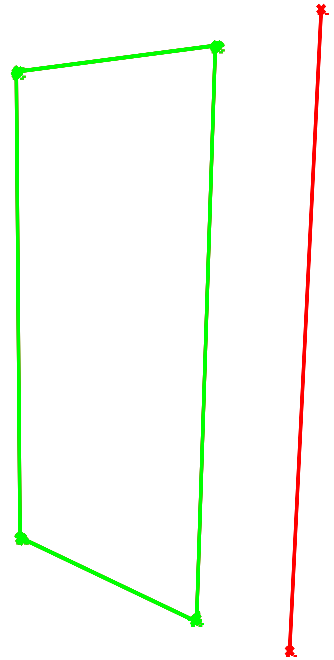

High curvature

Example – Door detection using Hough Transform





Hough Transform



Hough Transform: other features

Lines:

$$p = (d, \upsilon)$$

$$g(x,y,p) := x \cdot \cos(\upsilon) + y \cdot \sin(\upsilon) - d$$

Circles:

$$p = (x_0, y_0, r)$$

$$g(x,y,p) := (x - x_0)^2 + (y - y_0)^2 - r^2$$

Ellipses:

$$p = (x_0, y_0, a, b, \psi)$$

$$g(x,y,p) := \frac{\left[(x - x_0)\cdot\cos(\psi) + (y - y_0)\cdot\sin(\psi)\right]^2}{a^2} + \frac{\left[(y - y_0)\cdot\cos(\psi) - (x - x_0)\cdot\sin(\psi)\right]^2}{b^2} - 1$$

Hough Transform

- Advantages
  - Noise and background clutter do not impair detection of local maxima
  - Partial occlusion and varying contrast are minimized
- Negatives
  - Requires time and space storage that increases exponentially with the dimensions of the parameter space

Comparison **Line Detection**

- Deterministic methods perform better with laser scans
  - Split-and-merge, Line-Regression, Hough transform
  - Make use of the sequencing property of scan points.

- Nondeterministic methods can produce high False Positives
  - RANSAC
  - Do not use the sequencing property
  - But it can cope with outliers

- Overall:
  - Split-and-merge is the fastest, best real-time application