Intelligent Robots Practice

Perception with Uncertainty

Chungbuk National University, Korea
Intelligent Robots Lab. (IRL)

Prof. Gon-Woo Kim

IRL
Intelligent Robots Lab.
Chungbuk National University

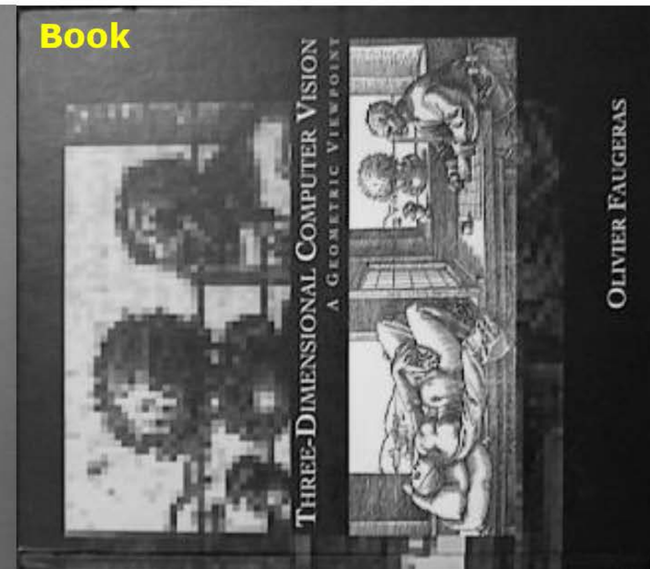
Contents

- From object recognition to scene/place recognition
- Probabilistic Primer
- Uncertainties
  - Representation + Propagation
  - Line extraction from a point cloud
    - Split-and-merge
    - Line-Regression
    - RANSAC
    - Hough Transform

From object recognition to scene/place recognition

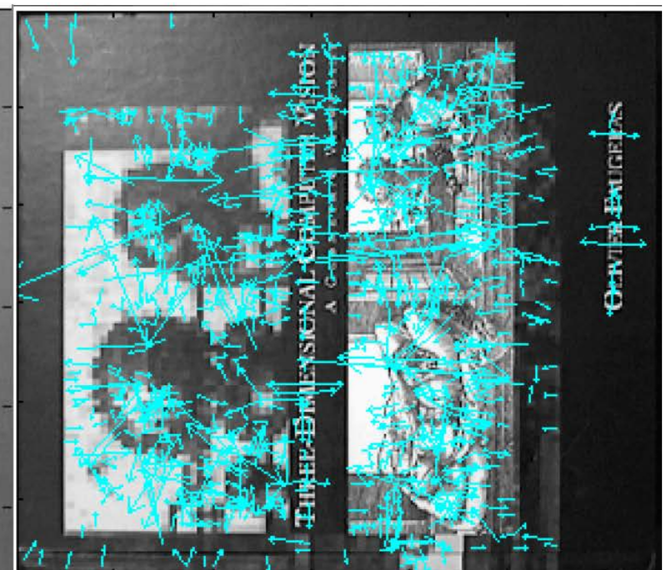
From object recognition to scene/place recognition

- Object Recognition
  - Q: Is this Book present in the Scene?



From object recognition to scene/place recognition

- Object Recognition
  - Q: Is this Book present in the Scene?



Extract keypoints in both images

From object recognition to scene/place recognition

- Object Recognition
  - Q: Is this Book present in the Scene?

Look for corresponding matches

Most of the Book's keypoints are present in the Scene

⇨ A: The Book is present in the Scene

From object recognition to scene/place recognition

- Object Recognition
  - Taking this a step further…
    - Find an object in an image

?

    - Find an object in multiple images

?

    - Find multiple objects in multiple images

?

From object recognition to scene/place recognition

- Bag of Words (BoW)
  - Extension to scene/place recognition
    - Is this image in my database?
    - Robot: Have I been to this place before?
      → "**loop closure**" problem, "**kidnapped robot**" problem
  - Use analogies from text retrieval
    - Visual Words
    - Vocabulary of Visual Words
    - "Bag of Words" (BOW) approach

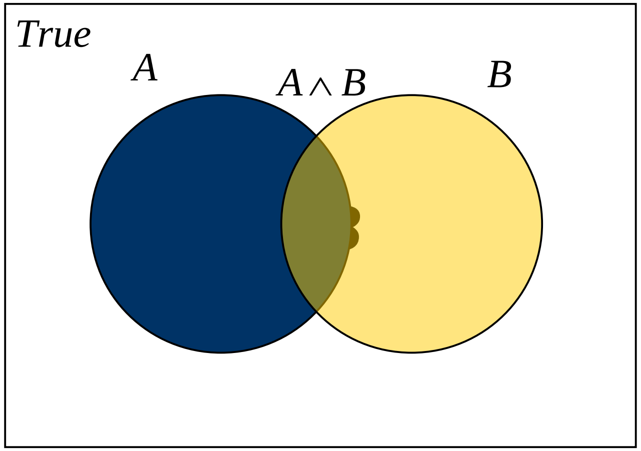
From object recognition to scene/place recognition

- Bag of Words (BoW)
  - Building the Visual Vocabulary

Probabilistic Primer

Probability

- Axioms of Probability Theory
  - Pr(A) denotes probability that proposition A is true.
    - Axiom1: $0 \le \Pr(A) \le 1$
    - Axiom2: $\Pr(True) = 1 \qquad \Pr(False) = 0$
    - Axiom3: $\Pr(A \vee B) = \Pr(A) + \Pr(B) - \Pr(A \wedge B)$



Probability

- Axioms of Probability Theory
  - Using the Axioms

$$
\begin{aligned}
\Pr(A \vee \neg A) &= \Pr(A) + \Pr(\neg A) - \Pr(A \wedge \neg A) \\
\Pr(True) &= \Pr(A) + \Pr(\neg A) - \Pr(False) \\
1 &= \Pr(A) + \Pr(\neg A) - 0 \\
\Pr(\neg A) &= 1 - \Pr(A)
\end{aligned}
$$

Probability

- Conditional probability
  - Conditional probability

$$p(x \mid y) = p(X = x \mid Y = y)$$

    - probability of X = x given Y = y

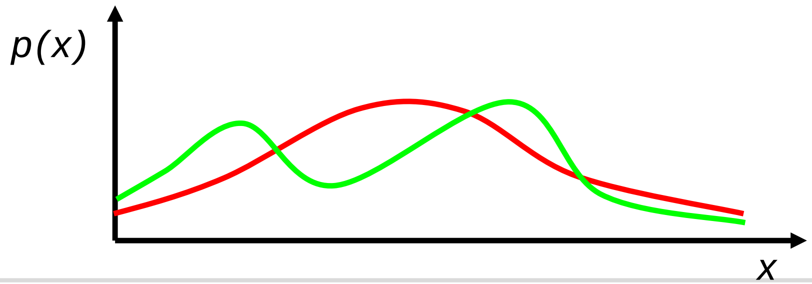
  - Joint probability

$$p(x, y) = p(X = x \text{ and } Y = y) = p(x \mid y)\,p(y) = p(y \mid x)\,p(x)$$

Probability

- Discrete Random Variables
  - X denotes a random variable.
  - X can take on a countable number of values in $\{x_1, x_2, \ldots, x_n\}$
  - $P(X=x_i)$, or $P(x_i)$, is the probability that the random variable X takes on value x_i.
  - $P(\cdot)$ is called probability mass function
  - E.g. $P(Room) = \langle 0.7, 0.2, 0.08, 0.02 \rangle$

- Continuous Random Variables
  - X takes on values in the continuum
  - $p(X=x)$, or $p(x)$, is a probability density function $\Pr(x \in (a,b)) = \int_a^b p(x)dx$



Probability

Joint and Conditional Probability

- $P(X=x \text{ and } Y=y) = P(x,y)$

- If X and Y are independent then

$$P(x,y) = P(x)\, P(y)$$

- $P(x \mid y)$ is the probability of x given y

$$P(x \mid y) = P(x,y) \,/\, P(y)$$

$$P(x,y) = P(x \mid y)\, P(y)$$

- If X and Y are independent then

$$P(x \mid y) = P(x)$$

Probability

- Law of Total Probability, Marginals

Discrete case

$$\sum_x P(x) = 1$$

$$P(x) = \sum_y P(x, y)$$

$$P(x) = \sum_y P(x \mid y) P(y)$$

Continuous case

$$\int p(x)\, dx = 1$$

$$p(x) = \int p(x, y)\, dy$$

$$p(x) = \int p(x \mid y)\, p(y)\, dy$$

Probability

- Bayes Formula

$$P(x, y) = P(x \mid y) P(y) = P(y \mid x) P(x)$$

$$\Rightarrow$$

$$\boxed{P(x \mid y) = \frac{P(y \mid x)\; P(x)}{P(y)} = \frac{\text{likelihood} \cdot \text{prior}}{\text{evidence}}}$$

Probability

- Bayes Formula
  - Normalization

$$P(x \mid y) = \frac{P(y \mid x)\ P(x)}{P(y)} = \eta\, P(y \mid x)\, P(x)$$

$$\eta = P(y)^{-1} = \frac{1}{\sum_x P(y \mid x) P(x)}$$

  - Algorithm:

$$\forall x : \mathrm{aux}_{x|y} = P(y \mid x)\ P(x)$$

$$\eta = \frac{1}{\sum_x \mathrm{aux}_{x|y}}$$

$$\forall x : P(x \mid y) = \eta\ \mathrm{aux}_{x|y}$$

Probability

- Conditioning
  - Law of total probability

$$P(x) = \int P(x, z)dz$$

$$P(x) = \int P(x \mid z)P(z)dz$$

$$P(x \mid y) = \int P(x \mid y, z)\, P(z \mid y)\, dz$$

  - Bayes Rule with Background Knowledge

$$P(x \mid y, z) = \frac{P(y \mid x, z)\; P(x \mid z)}{P(y \mid z)}$$

  - Conditional Independence

$$P(x, y \mid z) = P(x \mid z)P(y \mid z)$$

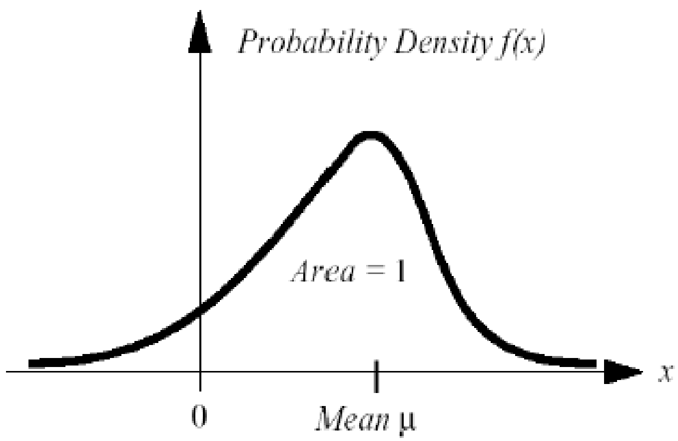
Uncertainties

Uncertainties

- Uncertainty Representation
  - Sensing in the real world is **always uncertain**
    - How can uncertainty be **represented** or quantified?
    - How does uncertainty **propagate**?
      i.e. given uncertain inputs into a system, what is the uncertainty in the output?
    - What is the merit of all this for mobile robotics?

Uncertainties

- Uncertainty Representation
  - Probability Density Function (PDF)
    - Use a Probability Density Function (PDF) to characterize the statistical properties of a variable X



$$\int_{-\infty}^{\infty} f(x)dx = 1$$

- Expected value of variable X:

$$\mu = E[X] = \int_{-\infty}^{\infty} xf(x)dx$$

- Variance of variable X

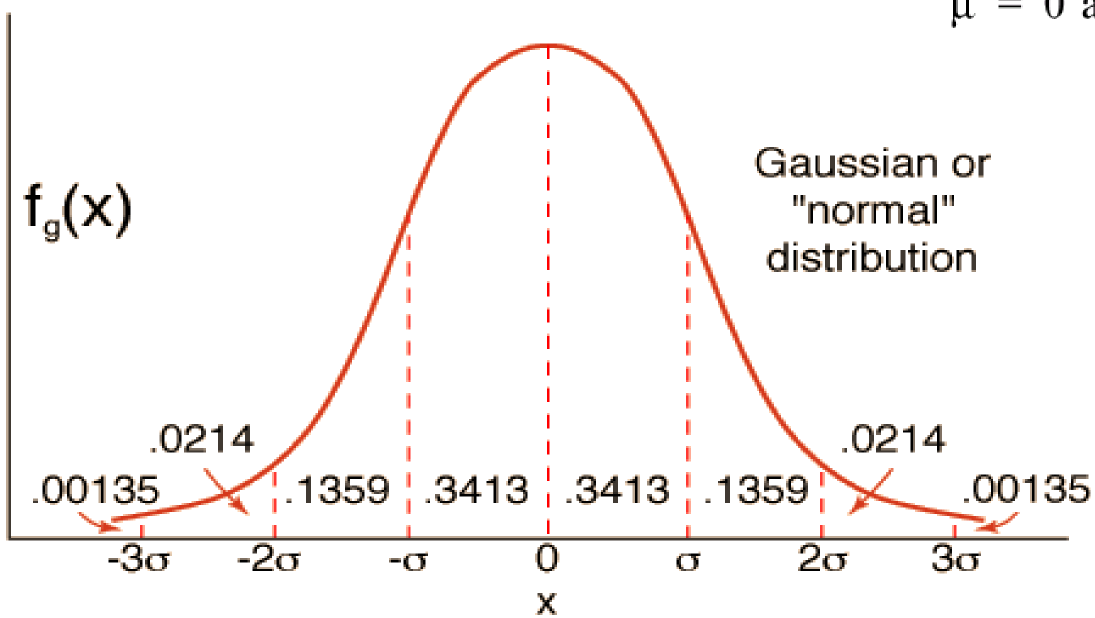
$$\sigma^2 = \int_{-\infty}^{\infty} (x-\mu)^2 f(x)dx$$

Uncertainties

- Uncertainty Representation
  - Probability Density Function (PDF)
    - Gaussian Distribution
      - Most common PDF for characterizing uncertainties: Gaussian

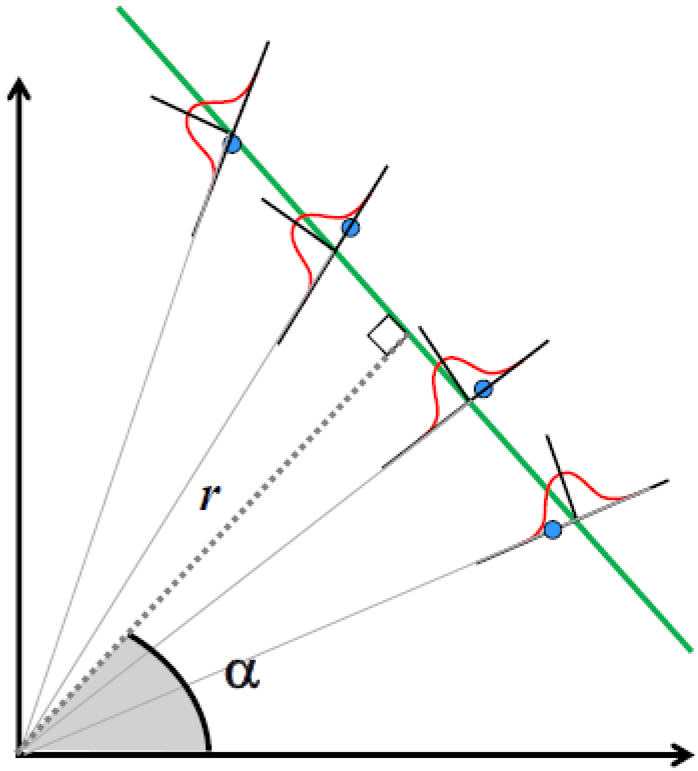
$$f(x) = \frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

$\mu = 0$ and $\sigma = 1$



Uncertainties

- Uncertainty Propagation
  - The Error Propagation Law
    - Imagine extracting a line based on point measurements with uncertainties.
    - Model parameters in polar coordinates [(r, α) uniquely identifies a line]
    - The question:
      - What is the **uncertainty of the extracted line** knowing the uncertainties of the measurement points that contribute to it ?

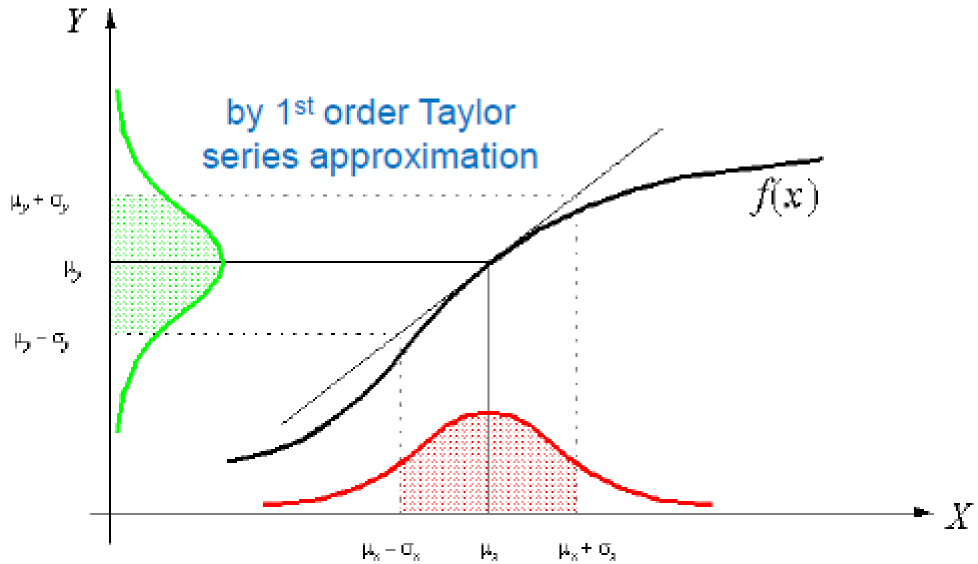


Uncertainties

Uncertainty Propagation

The Error Propagation Law

- 1D case of a nonlinear error propagation problem

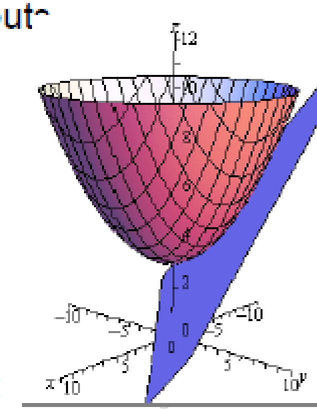


- It can be shown that the output covariance matrix C_Y is given by the error propagation law:

$$C_Y = F_X C_X F_X^T$$

- where
  - C_X: covariance matrix representing the input uncertainties
  - C_Y: covariance matrix representing the propagated uncertainties for the outputs
  - F_X: is the ***Jacobian*** matrix defined as:

$$F_X = \begin{bmatrix} \frac{\partial f_1}{\partial X_1} & \cdots & \frac{\partial f_1}{\partial X_n} \\ \vdots & \cdots & \vdots \\ \frac{\partial f_m}{\partial X_1} & \cdots & \frac{\partial f_m}{\partial X_n} \end{bmatrix}$$



  - Defines the orientation of the tangent line/plane/hyper-plane at a given point

Uncertainties

- Uncertainty Propagation
  - The Error Propagation Law
    - Line Extraction: Unweighted Least Sq.
      - Point-Line distance

$$\rho_i \cos(\theta_i - \alpha) - r = d_i$$

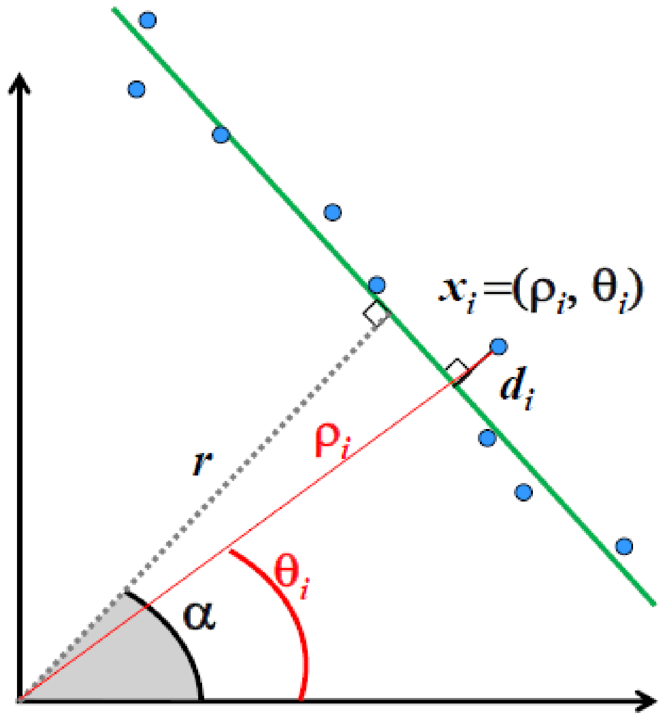
      - If each measurement is equally uncertain then sum of sq. errors:

$$S = \sum_i d_i^2 = \sum_i (\rho_i \cos(\theta_i - \alpha) - r)^2$$

      - Goal: minimize S when selecting (r, α)
        ⇨ solve the system

$$\frac{\partial S}{\partial \alpha} = 0 \qquad \frac{\partial S}{\partial r} = 0$$

      - **"Unweighted Least Squares"**



Intelligent Robots Lab.
Chungbuk National University

Uncertainties

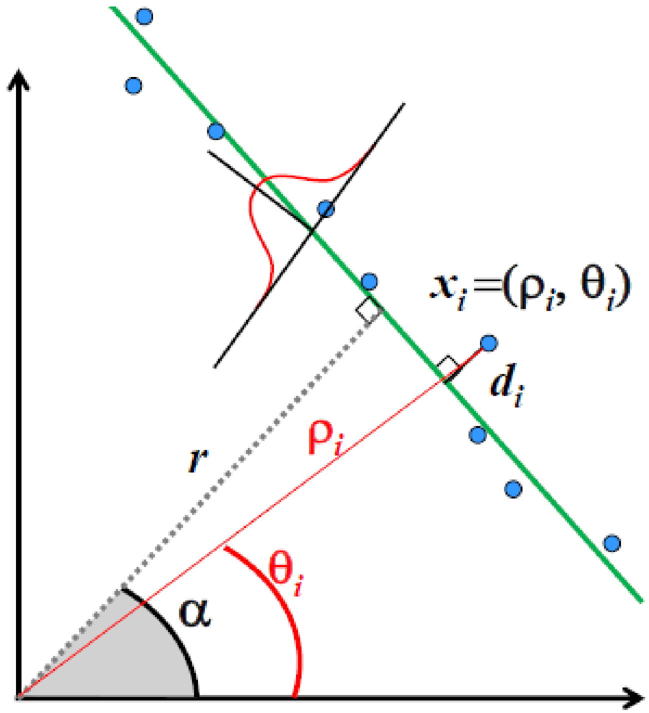
- Uncertainty Propagation
  - The Error Propagation Law
    - Line Extraction: Unweighted Least Sq.
      - Point-Line distance

$$\rho_i \cos(\theta_i - \alpha) - r = d_i$$

      - Each sensor measurement, may have its own, unique uncertainty

$$S = \sum w_i d_i^2 = \sum w_i (\rho_i \cos(\theta_i - \alpha) - r)^2$$

$$w_i = 1/\sigma_i^2$$



      - **Weighted Least Squares**

$$\frac{\partial S}{\partial \alpha} = 0 \qquad \frac{\partial S}{\partial r} = 0$$

Uncertainties

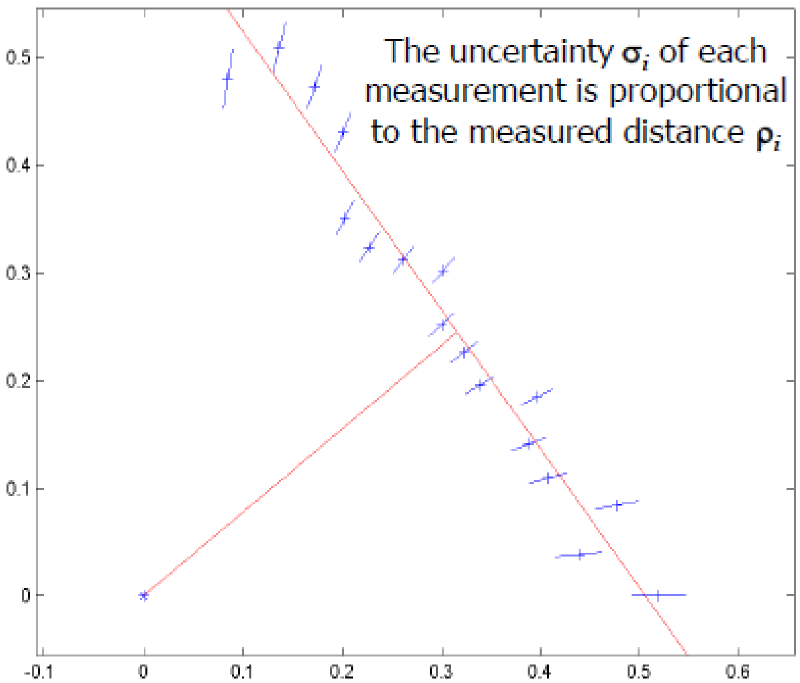
- Uncertainty Propagation
  - The Error Propagation Law
    - Line Extraction Uncertainty
      - Weighted least squares and solving the system:

$$\frac{\partial S}{\partial \alpha} = 0 \qquad \frac{\partial S}{\partial r} = 0$$

      - Gives the line parameters:

$$\alpha = \frac{1}{2}\operatorname{atan}\left(\frac{\sum w_i \rho_i^2 \sin 2\theta_i - \frac{2}{\Sigma w_i}\sum\sum w_i w_j \rho_i \rho_j \cos\theta_i \sin\theta_j}{\sum w_i \rho_i^2 \cos 2\theta_i - \frac{1}{\Sigma w_i}\sum\sum w_i w_j \rho_i \rho_j \cos(\theta_i + \theta_j)}\right)$$

$$r = \frac{\sum w_i \rho_i \cos(\theta_i - \alpha)}{\sum w_i}$$



      - If $\begin{cases} \rho_i \sim N(\hat{\rho}_i, \sigma_{\rho_i}{}^2) \\ \theta_i \sim N(\hat{\theta}_i, \sigma_{\theta_i}{}^2) \end{cases}$ what is the uncertainty in the line (r, α) ?

Uncertainties

- Uncertainty Propagation
  - The Error Propagation Law
    - Error Propagation: Line extraction

Assuming that ρ_i, θ_i are independent

The uncertainty of **each measurement** $x_i=(\rho_i, \theta_i)$ is described by the covariance matrix: $C_{x_i} = \begin{bmatrix} \sigma_{\rho_i}^2 & 0 \\ 0 & \sigma_{\theta_i}^2 \end{bmatrix}$

The uncertainty in the **line** (r, α) is described by the covariance matrix: $C_{\alpha r} = \begin{bmatrix} \sigma_\alpha^2 & \sigma_{\alpha r} \\ \sigma_{r\alpha} & \sigma_r^2 \end{bmatrix}$ **= ?**

Define:

$$C_x = \begin{bmatrix} diag(\sigma_\rho^2) & 0 \\ 0 & diag(\sigma_\theta^2) \end{bmatrix} = \begin{bmatrix} \cdots & 0 & 0 & . & 0 & 0 & . \\ . & \sigma_{\rho_i}^2 & 0 & . & 0 & 0 & . \\ . & 0 & \sigma_{\rho_{i+1}}^2 & . & 0 & 0 & . \\ . & . & . & \cdots & . & . & . \\ . & 0 & 0 & . & \sigma_{\theta_i}^2 & 0 & . \\ . & 0 & 0 & . & 0 & \sigma_{\theta_i+1}^2 & . \\ . & . & . & . & . & . & \cdots \end{bmatrix}_{2n \times 2n}$$

Error Propagation Law

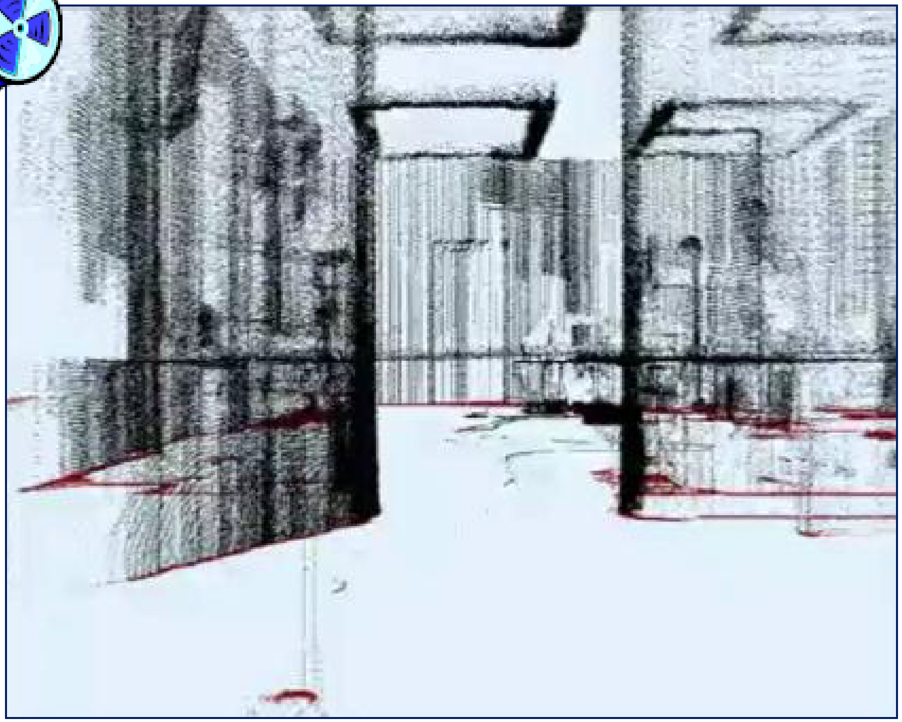
$$\boxed{C_{\alpha r} = F_{\rho\theta} C_x F_{\rho\theta}^T}$$

Jacobian:

$$F_{\rho\theta} = \begin{bmatrix} \cdots & \frac{\partial\alpha}{\partial\rho_i} & \frac{\partial\alpha}{\partial\rho_{i+1}} & \cdots & \frac{\partial\alpha}{\partial\theta_i} & \frac{\partial\alpha}{\partial\theta_{i+1}} & \cdots \\ \cdots & \frac{\partial r}{\partial\rho_i} & \frac{\partial r}{\partial\rho_{i+1}} & \cdots & \frac{\partial r}{\partial\theta_i} & \frac{\partial r}{\partial\theta_{i+1}} & \cdots \end{bmatrix}$$

Uncertainties

- Line Extraction from a point cloud
  - Extract lines from a point cloud (e.g. range scan)
    - Three main problems:
      - How many lines are there?
      - **Segmentation**: Which points belong to which line ?
      - **Line Fitting/Extraction**: Given points that belong to a line, how to estimate the line parameters ?
    - Algorithms we will see:
      1. Split-and-merge
      2. Linear regression
      3. RANSAC
      4. Hough-Transform



Uncertainties

- Line Extraction from a point cloud
  - Algorithm 1: Split-and-Merge (standard)
    - Popular algorithm, originates 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.

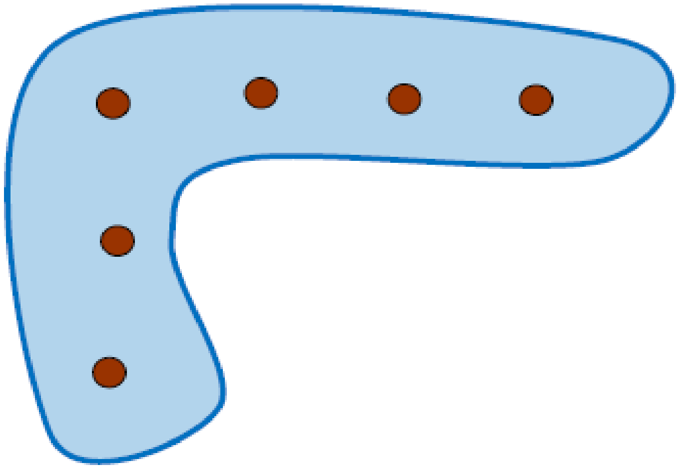
Initialise set **S** to contain all points

Split

- Fit a line to points in current set **S**
- Find the most distant point to the line
- If distance > threshold ⇨ split set & repeat with left and right point sets

Merge

- If two consecutive segments are close/collinear enough, obtain the common line and find the most distant point
- If distance <= threshold, merge both segments



Uncertainties

- Line Extraction from a point cloud
  - Algorithm 1: Split-and-Merge (standard)
    - Popular algorithm, originates 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.

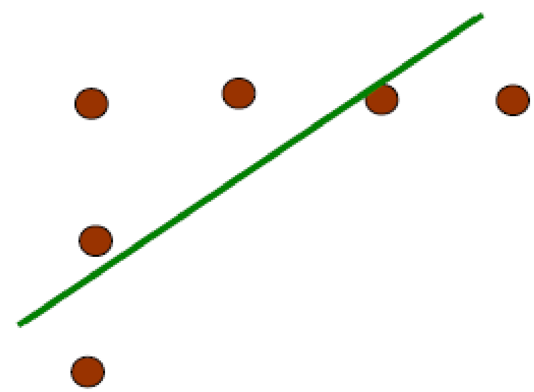
Initialise set **S** to contain all points

Split

- Fit a line to points in current set **S**
- Find the most distant point to the line
- If distance > threshold ⇨ split set & repeat with left and right point sets

Merge

- If two consecutive segments are close/collinear enough, obtain the common line and find the most distant point
- If distance <= threshold, merge both segments



Uncertainties

- Line Extraction from a point cloud
  - Algorithm 1: Split-and-Merge (standard)
    - Popular algorithm, originates 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.

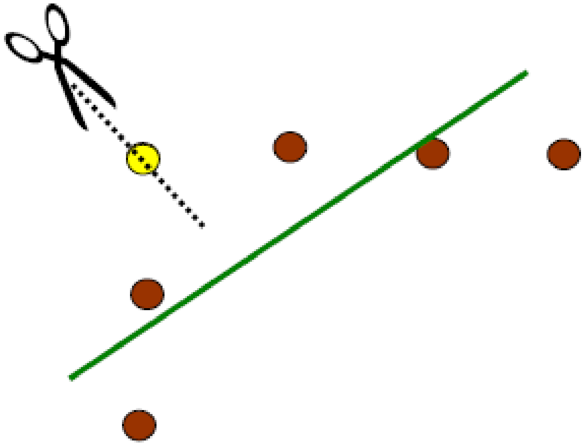
Initialise set **S** to contain all points

Split

- Fit a line to points in current set **S**
- Find the most distant point to the line
- If distance > threshold ⇨ split set & repeat with left and right point sets

Merge

- If two consecutive segments are close/collinear enough, obtain the common line and find the most distant point
- If distance <= threshold, merge both segments



Uncertainties

- Line Extraction from a point cloud
  - Algorithm 1: Split-and-Merge (standard)
    - Popular algorithm, originates 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.

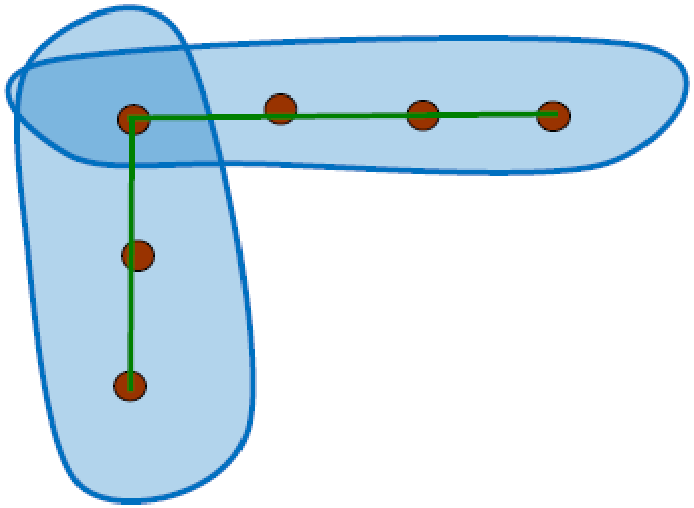
Initialise set **S** to contain all points

Split

- Fit a line to points in current set **S**
- Find the most distant point to the line
- If distance > threshold ⇨ split set & repeat with left and right point sets

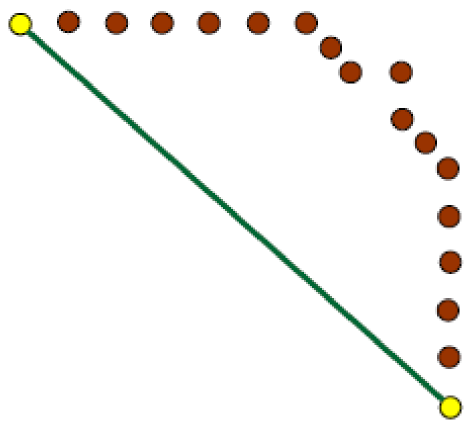
Merge

- If two consecutive segments are close/collinear enough, obtain the common line and find the most distant point
- If distance <= threshold, merge both segments



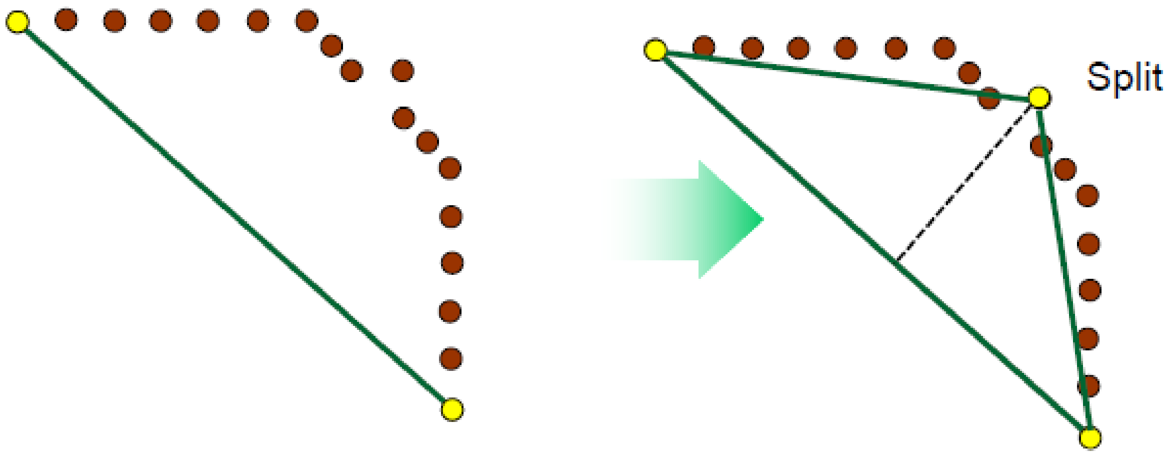
Uncertainties

- Line Extraction from a point cloud
  - Algorithm 1: Split-and-Merge (iterative end-point-fit)
    - Iterative end-point-fit: simply connects the end points for line fitting



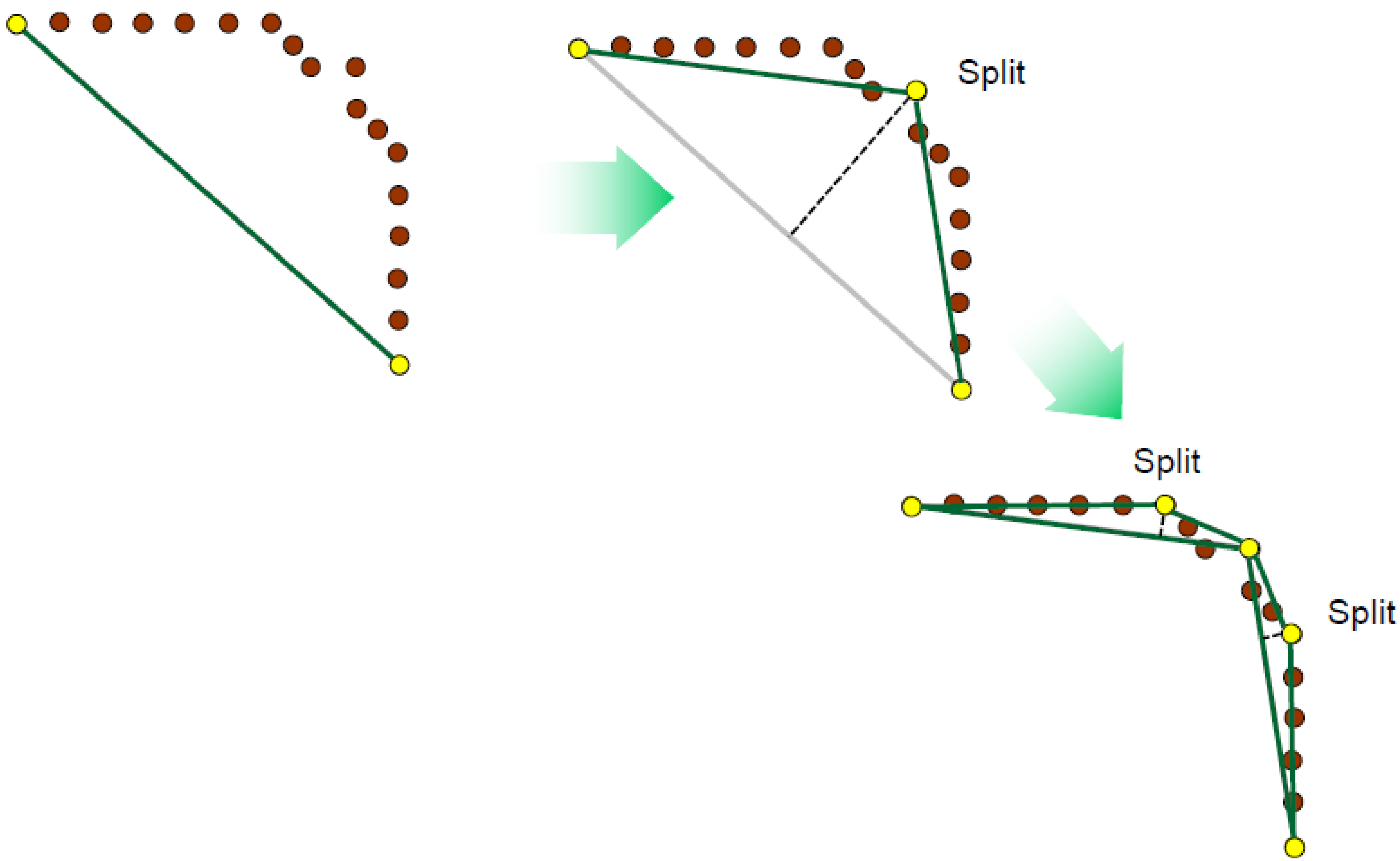
Uncertainties

- Line Extraction from a point cloud
  - Algorithm 1: Split-and-Merge (iterative end-point-fit)
    - Iterative end-point-fit: simply connects the end points for line fitting



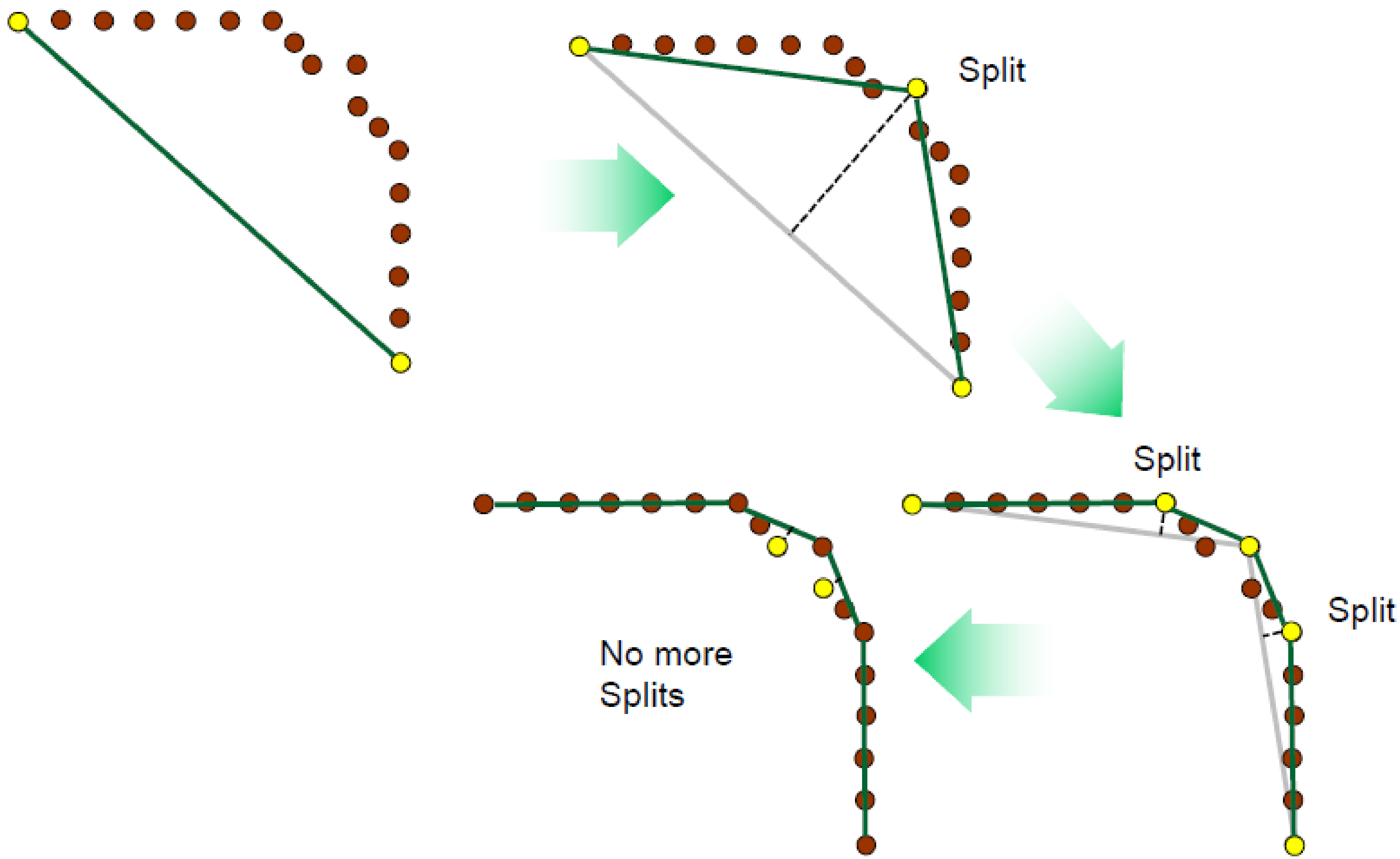
Uncertainties

- Line Extraction from a point cloud
  - Algorithm 1: Split-and-Merge (iterative end-point-fit)
    - Iterative end-point-fit: simply connects the end points for line fitting



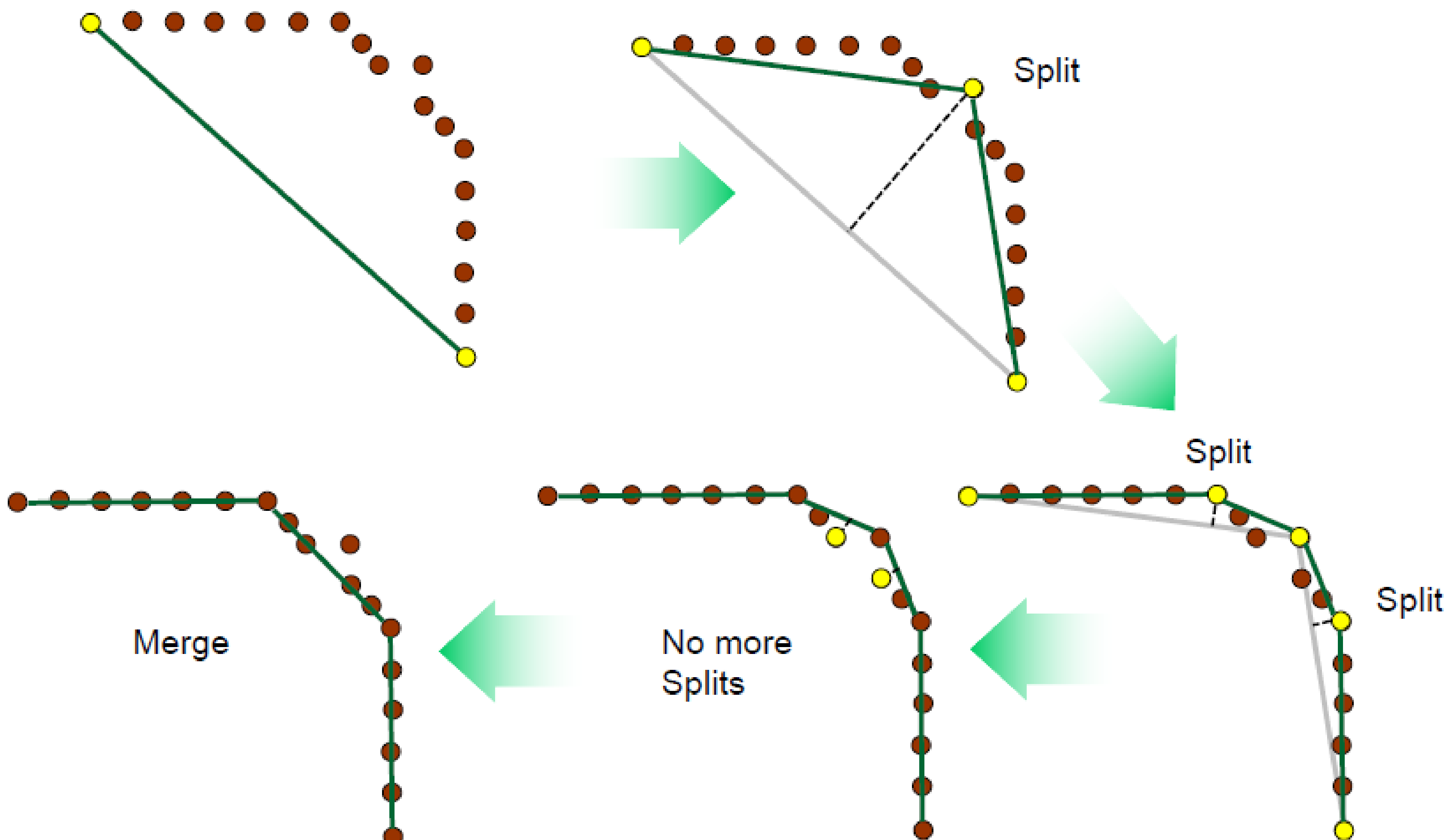
Uncertainties

- Line Extraction from a point cloud
  - Algorithm 1: Split-and-Merge (iterative end-point-fit)
    - Iterative end-point-fit: simply connects the end points for line fitting



Uncertainties

- Line Extraction from a point cloud
  - Algorithm 1: Split-and-Merge (iterative end-point-fit)
    - Iterative end-point-fit: simply connects the end points for line fitting



Uncertainties

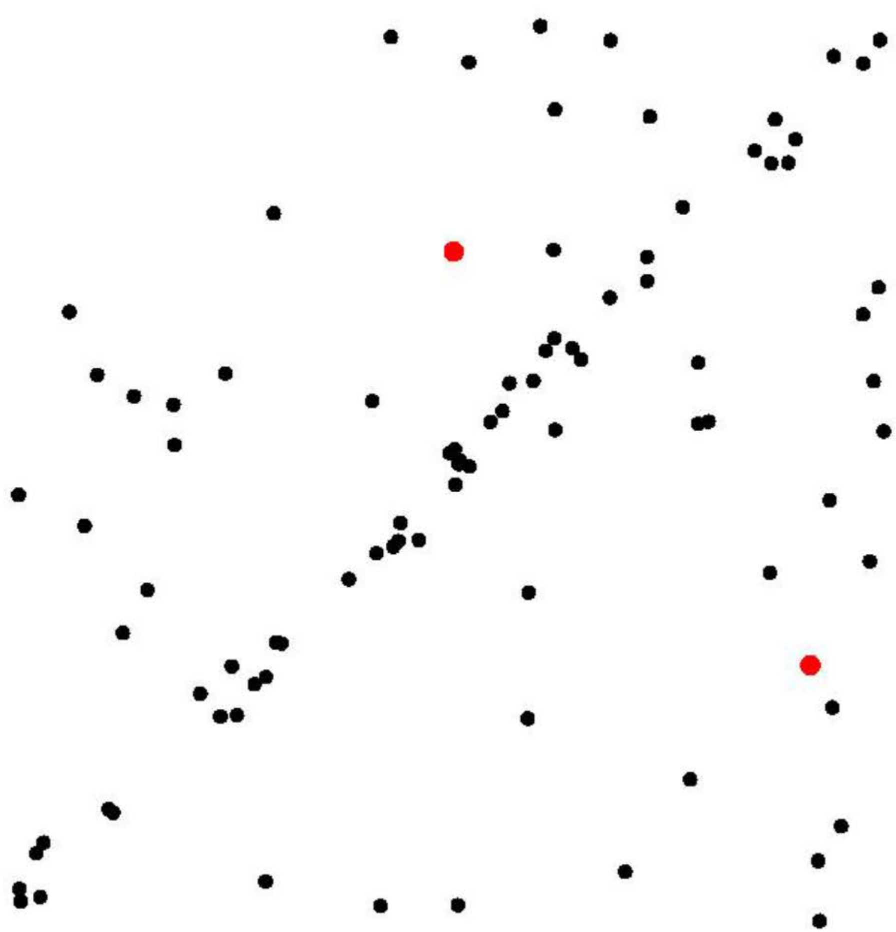
- Line Extraction from a point cloud
  - Algorithm 3: RANSAC
    - **RAN**dom **SA**mple **C**onsensus(RANSAC)
    - It is a generic and robust fitting algorithm of models in the presence of outliers (i.e. points which do not satisfy a model)
    - Generally applicable algorithm to any problem where the goal is to **identity the inliers which satisfy a predefined model**
    - Typical application in robotics
      - Line extraction from 2D range data
      - Plane extraction from 3D range data
      - Feature matching, structure from motion, etc
    - RANSAC is an **iterative** method and is **non-deterministic** in that the probability to find a set free of outliers increases as more iterations are used
    - Drawback: a **non-deterministic method, different results**

Uncertainties

- Line Extraction from a point cloud
  - Algorithm 3: RANSAC

Uncertainties

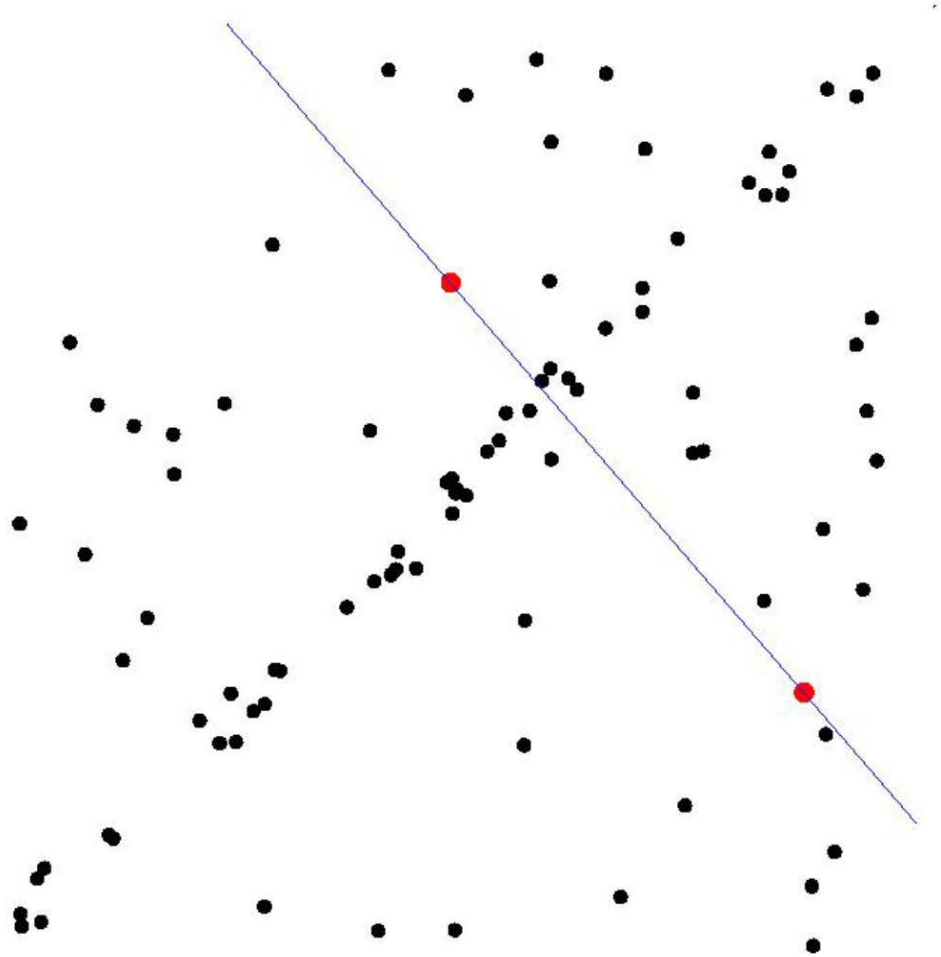
- Line Extraction from a point cloud
  - Algorithm 3: RANSAC



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

Uncertainties

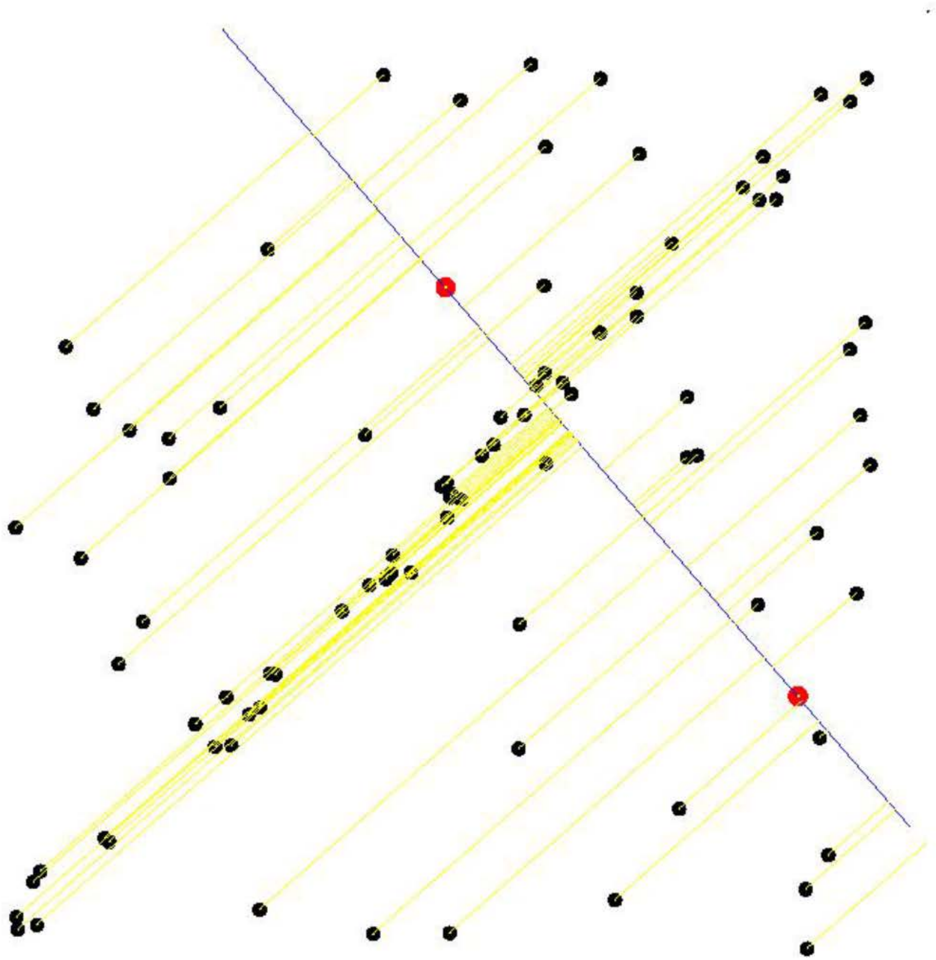
- Line Extraction from a point cloud
  - Algorithm 3: RANSAC



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

Uncertainties

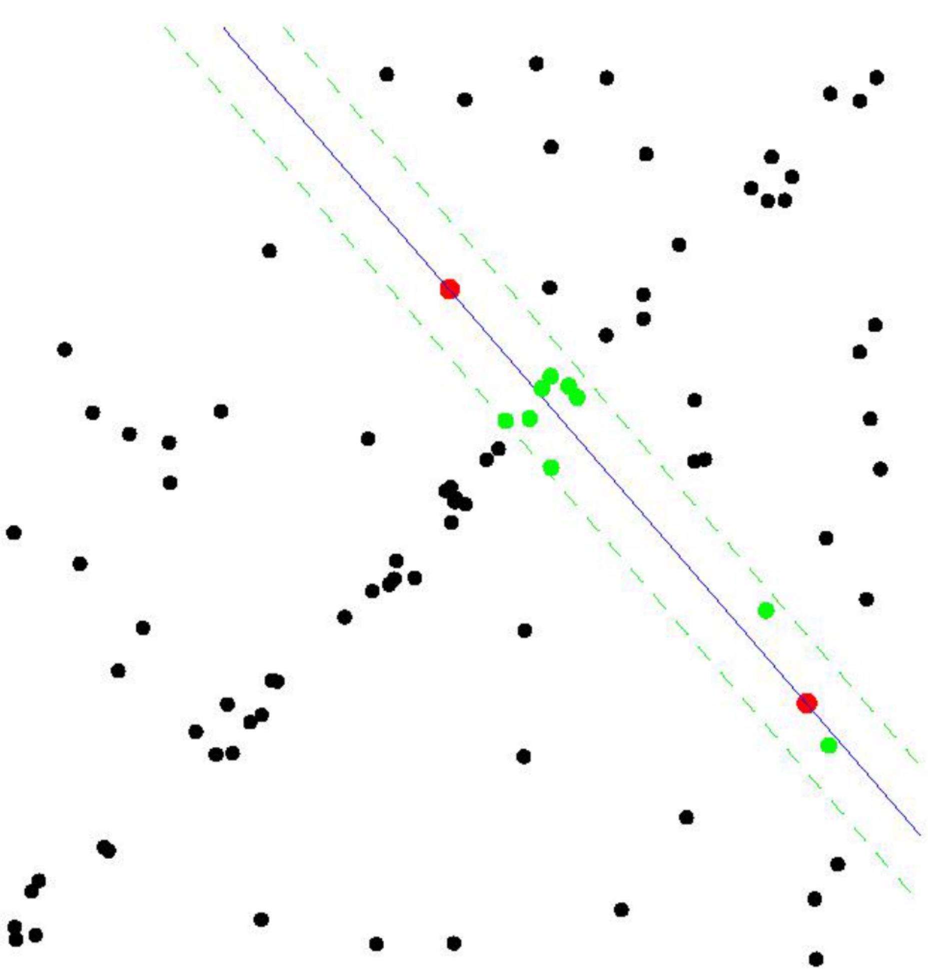
- Line Extraction from a point cloud
  - Algorithm 3: RANSAC



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

Uncertainties

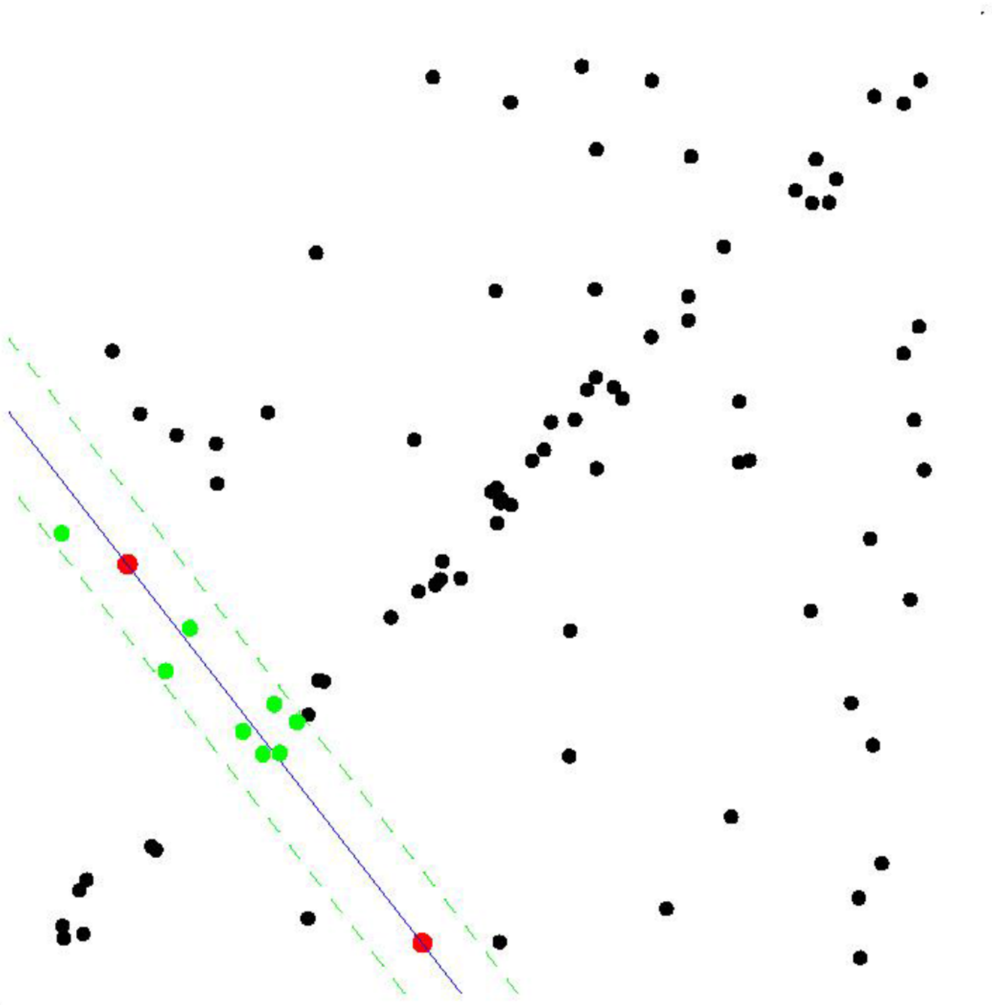
- Line Extraction from a point cloud
  - Algorithm 3: RANSAC



- 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**

Uncertainties

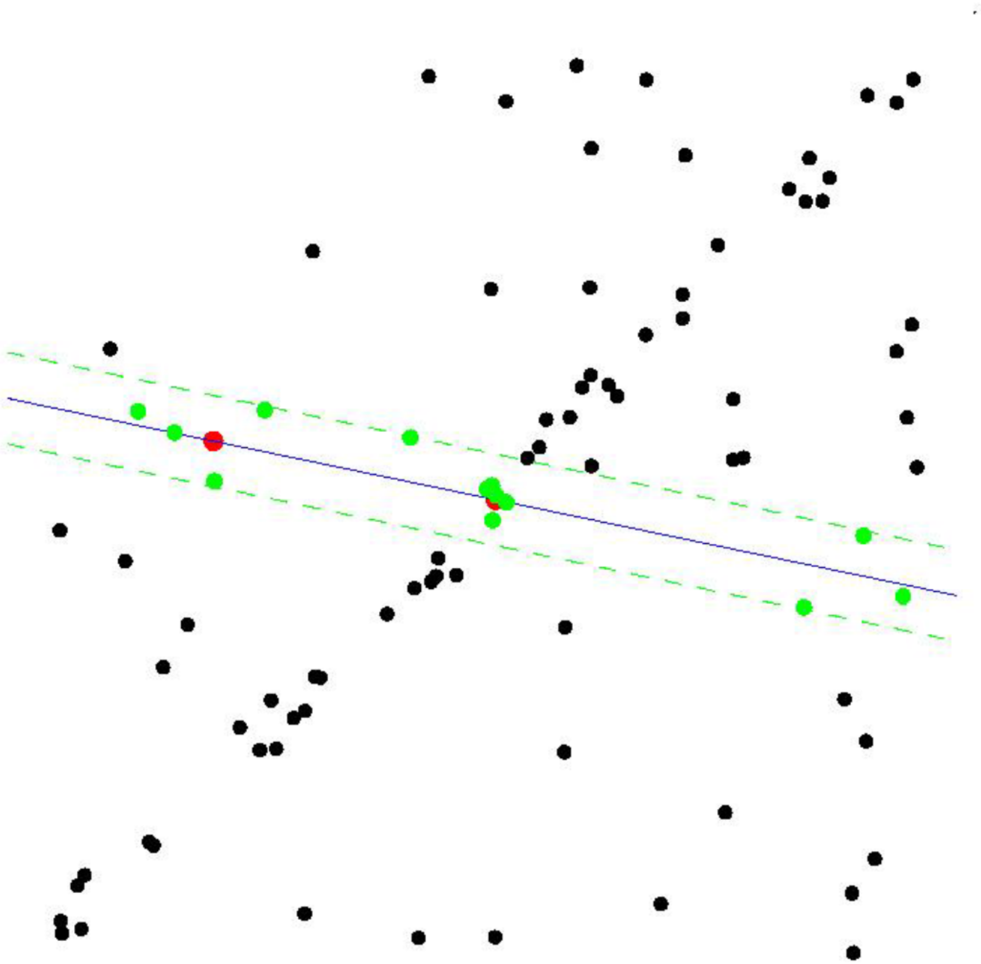
- Line Extraction from a point cloud
  - Algorithm 3: RANSAC



- 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**

Uncertainties

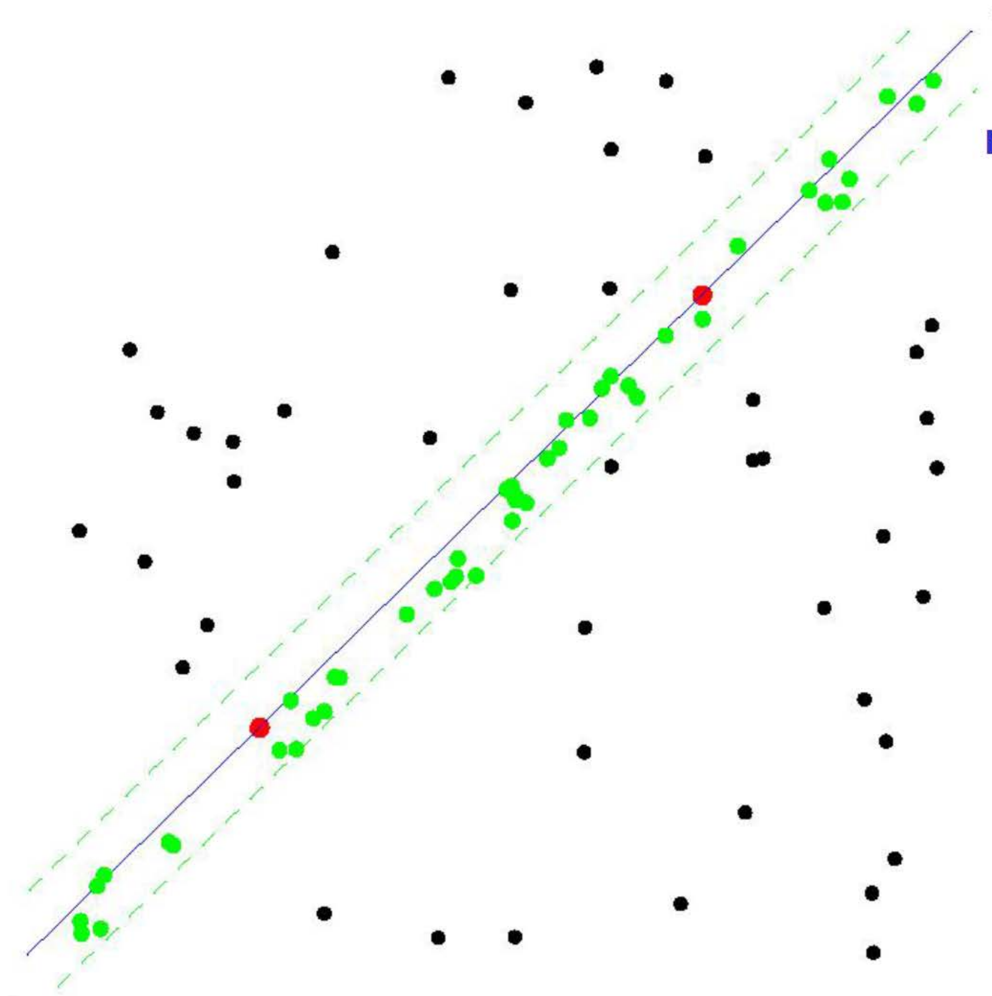
- Line Extraction from a point cloud
  - Algorithm 3: RANSAC



- 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**

Uncertainties

- Line Extraction from a point cloud
  - Algorithm 3: RANSAC



Set with the maximum number of inliers obtained within k iterations

Uncertainties

- Line Extraction from a point cloud
  - Algorithm 3: RANSAC

Algorithm *RANSAC* *(for line extraction from 2D range data)*

---

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

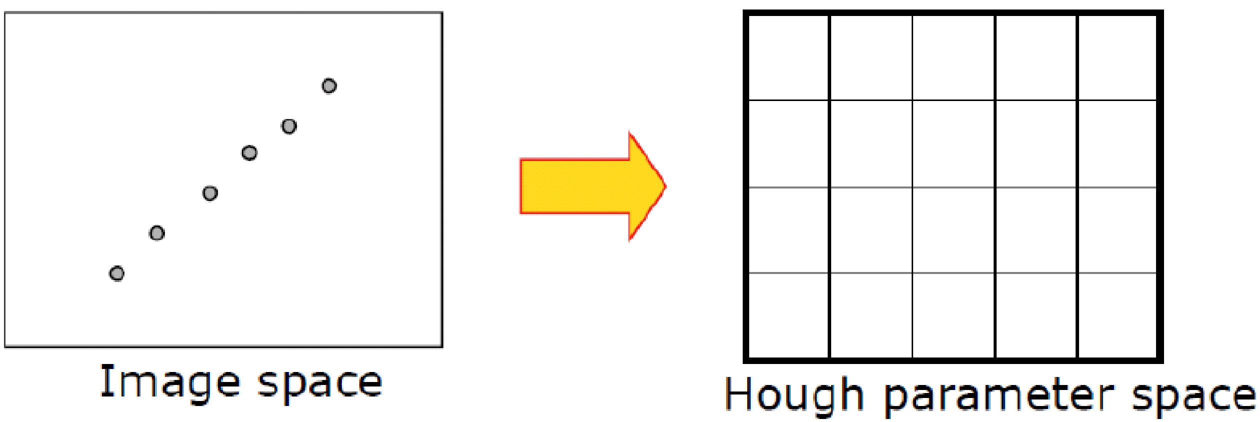
---

Uncertainties

- Line Extraction from a point cloud
  - Algorithm 3: RANSAC
    - How many iterations does RANSAC need?
      - We cannot know in advance if the observed set contains the max. no. inliers
        ⇨ ideally: check all possible combinations of 2 points in a dataset of **N** points.
      - No. all pairwise combinations: **N(N-1)/2**
        ⇨ computationally infeasible if **N** is too large.
        example: laser scan of **360** points ⇨ 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?
        Checking a subset of combinations is enough if we have a **rough** estimate of the percentage of inliers in our dataset
      - This can be done in a probabilistic way

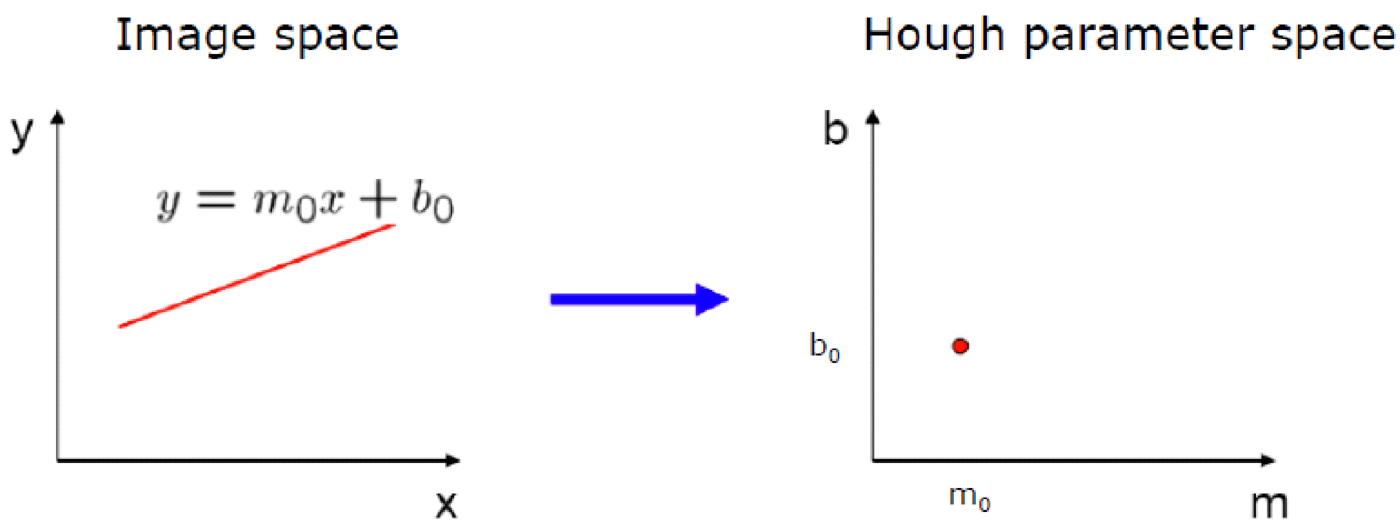
Uncertainties

- Line Extraction from a point cloud
  - Algorithm 4: Hough-Transform
    - Edges **vote** for plausible line locations
    - Map image space into Hough parameter space
    - Hough space parameterizes coordinate space w.r.t line characteristics
    - In practice, it's a discretized accumulator array
      - Comprising of voting bins



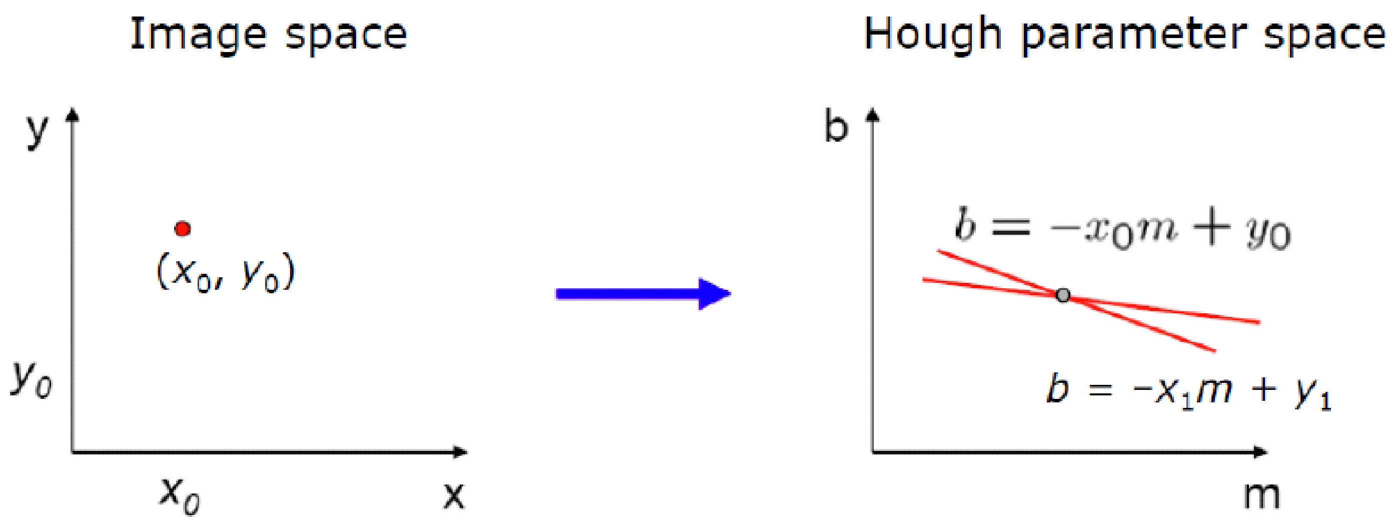
Uncertainties

- Line Extraction from a point cloud
  - Algorithm 4: Hough-Transform
    - A line in the image corresponds to a point in Hough space



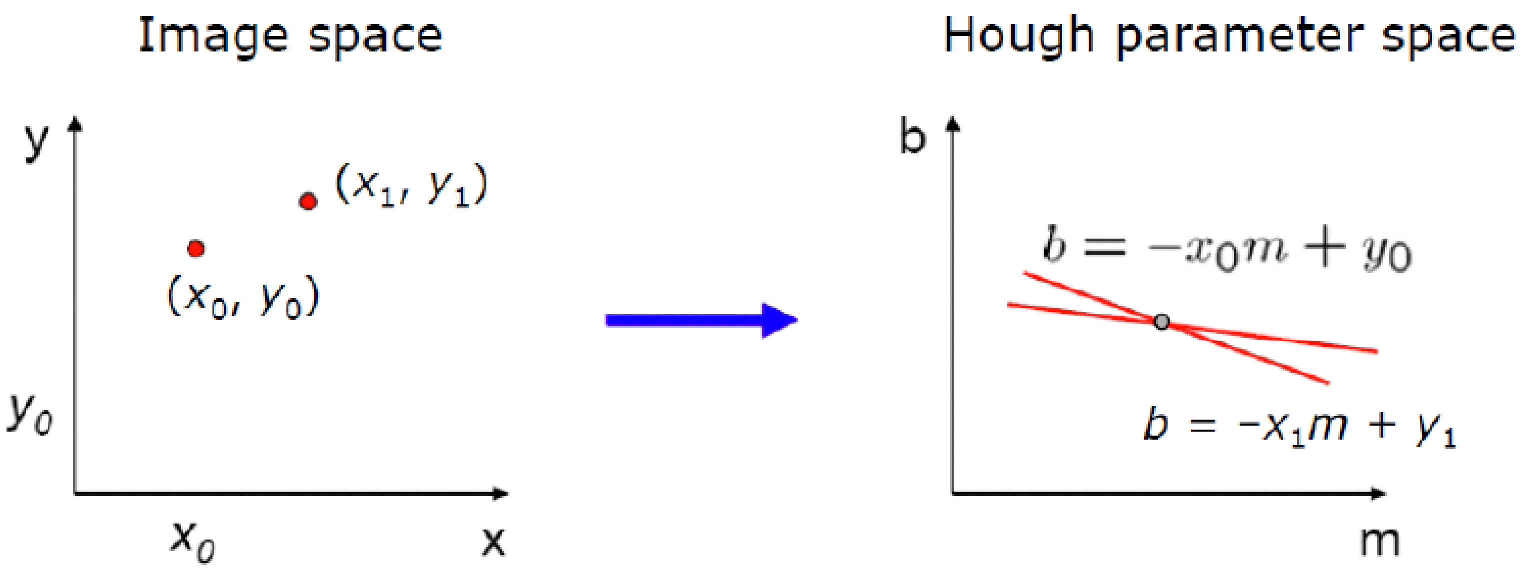
Uncertainties

- Line Extraction from a point cloud
  - Algorithm 4: Hough-Transform
    - What does a point (x_0, y_0) in the image space map to in the Hough space?



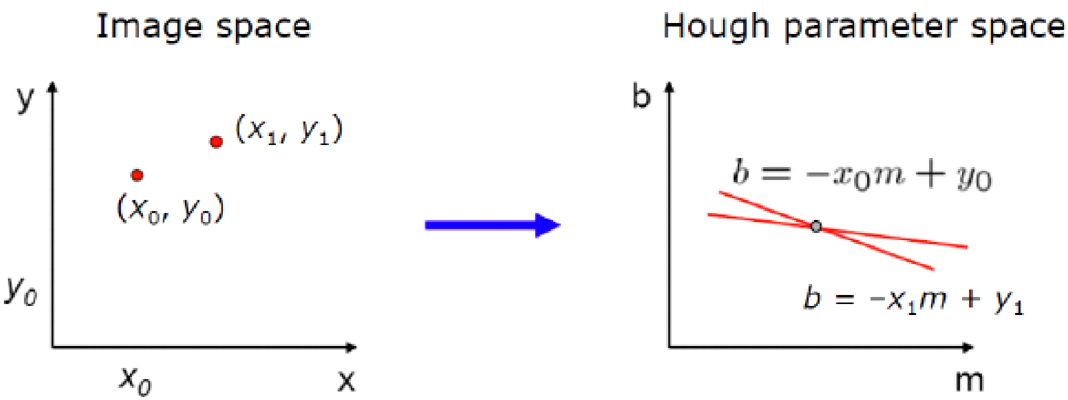
Uncertainties

- Line Extraction from a point cloud
  - 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 $b = -x_0 m + y_0$ and $b = -x_1 m + y_1$

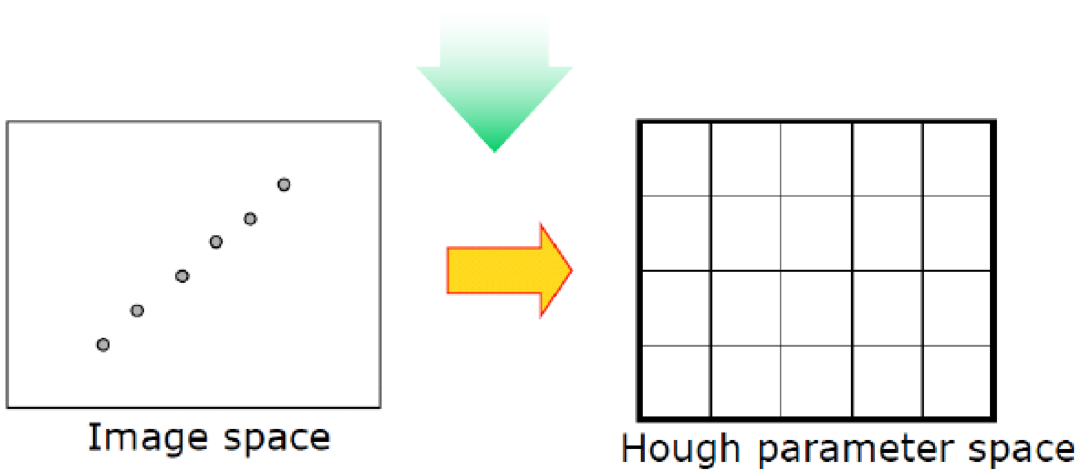


Uncertainties

- Line Extraction from a point cloud
  - Algorithm 4: Hough-Transform

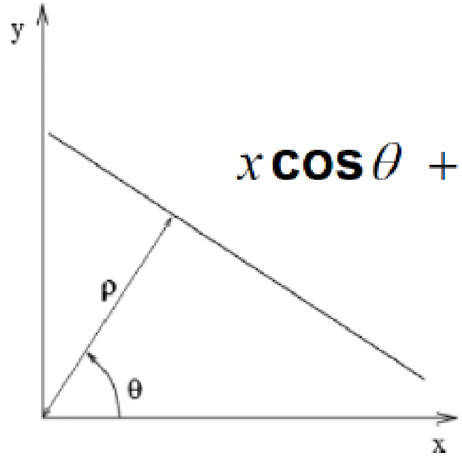


- Each point in image space, votes for line-parameters in Hough parameter space



Uncertainties

- Line Extraction from a point cloud
  - 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 in image space will map to a sinusoid in the (ρ,θ) parameter space

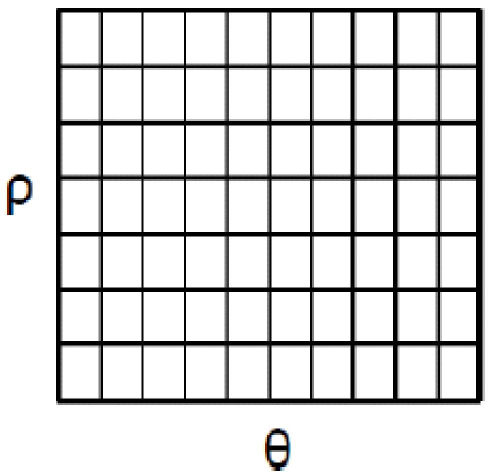
Uncertainties

- Line Extraction from a point cloud
  - Algorithm 4: Hough-Transform

1. Initialize accumulator H to all zeros
2. **for** each edge point (x,y) in the image
   - **for** all θ in [0,180]
     - Compute $\rho = x \cos \theta + y \sin \theta$
     - $H(\theta, \rho) = H(\theta, \rho) + 1$
   - **end**

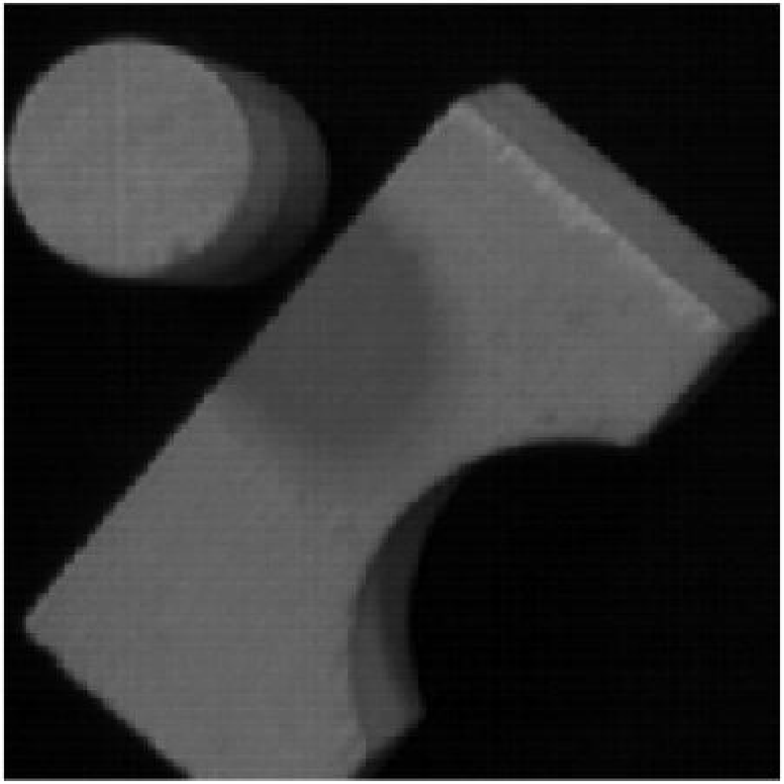
   end
3. Find the values of (θ, ρ) where $H(\theta, \rho)$ is a local maximum
4. The detected line in the image is given by: $\rho = x \cos \theta + y \sin \theta$

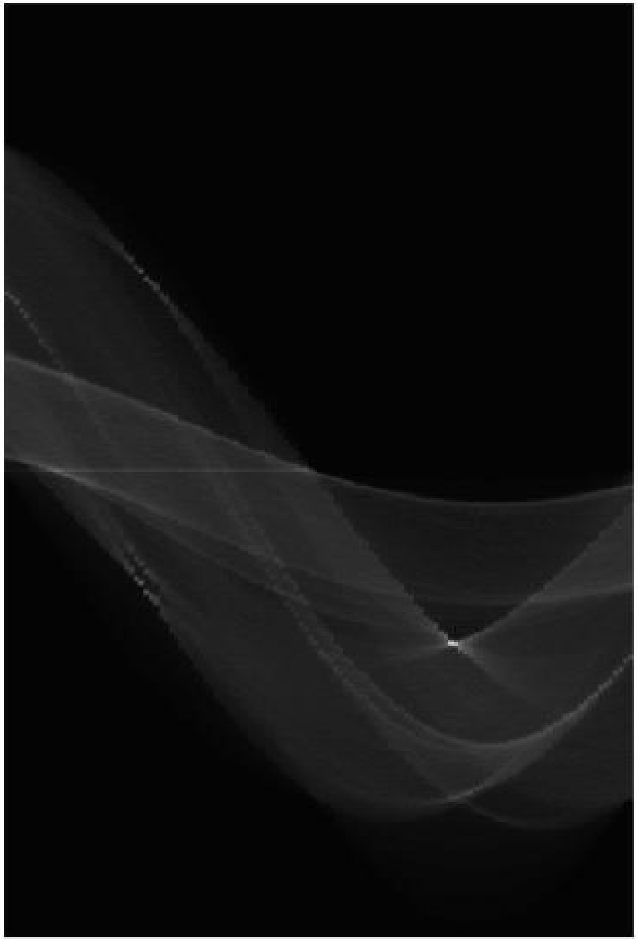
H: accumulator array (votes)



Uncertainties

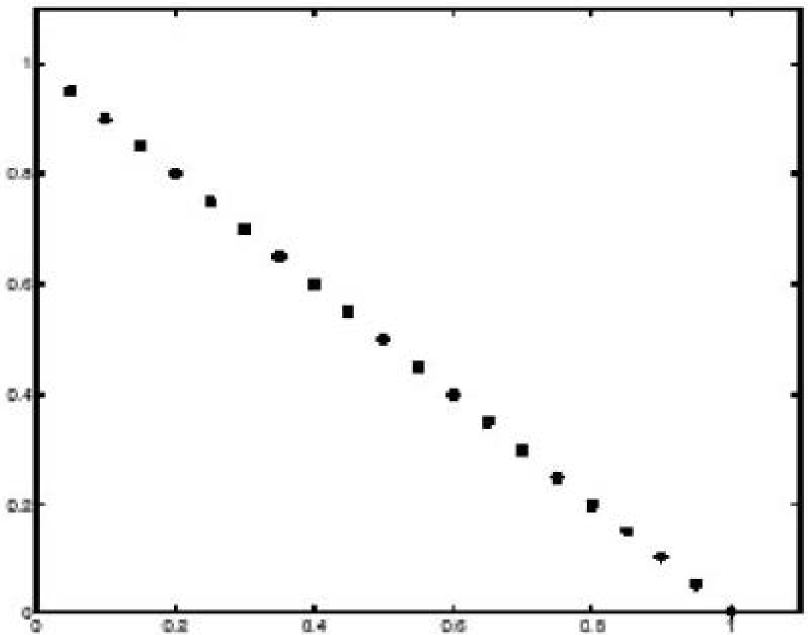
- Line Extraction from a point cloud
  - Algorithm 4: Hough-Transform



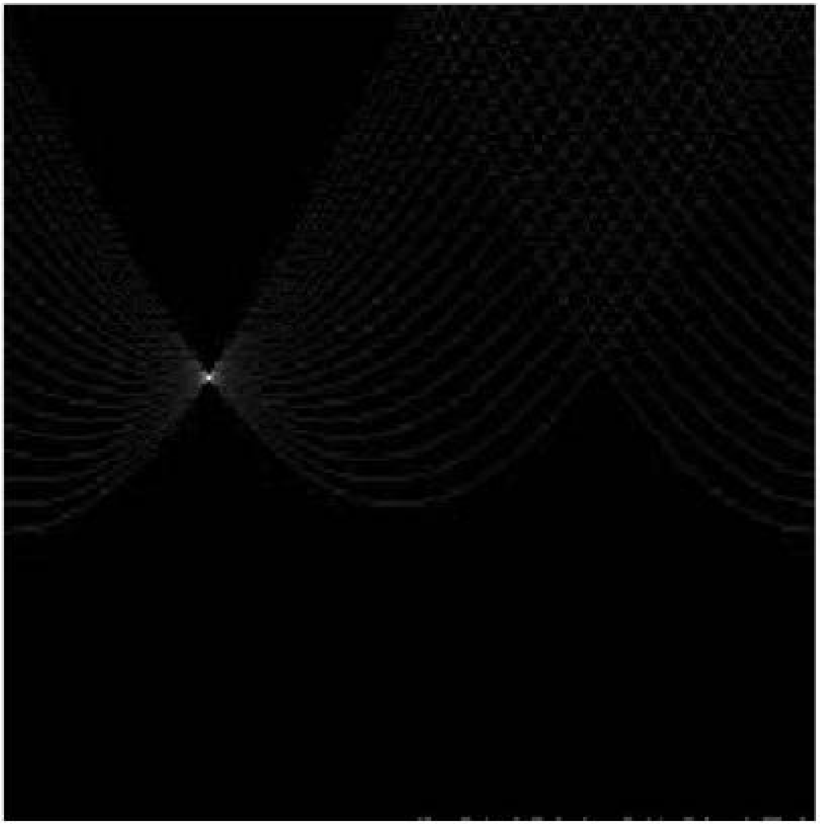


Uncertainties

- Line Extraction from a point cloud
  - Algorithm 4: Hough-Transform



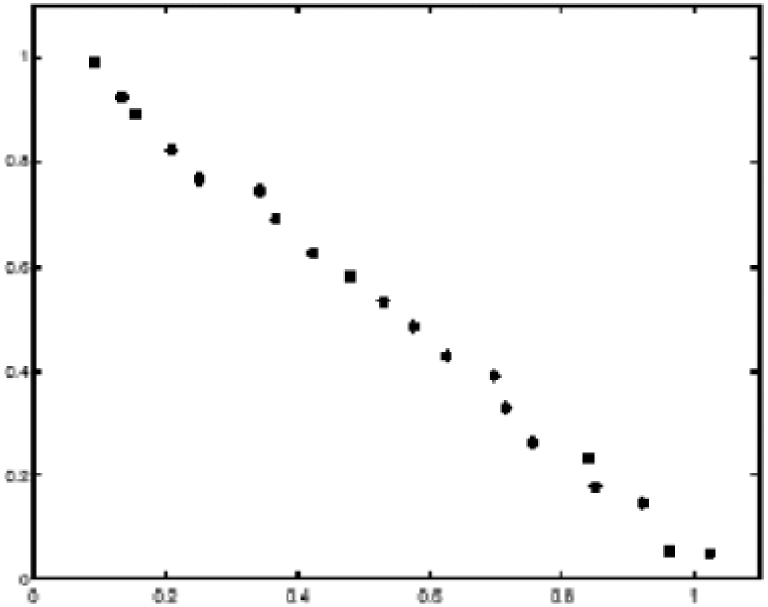
features



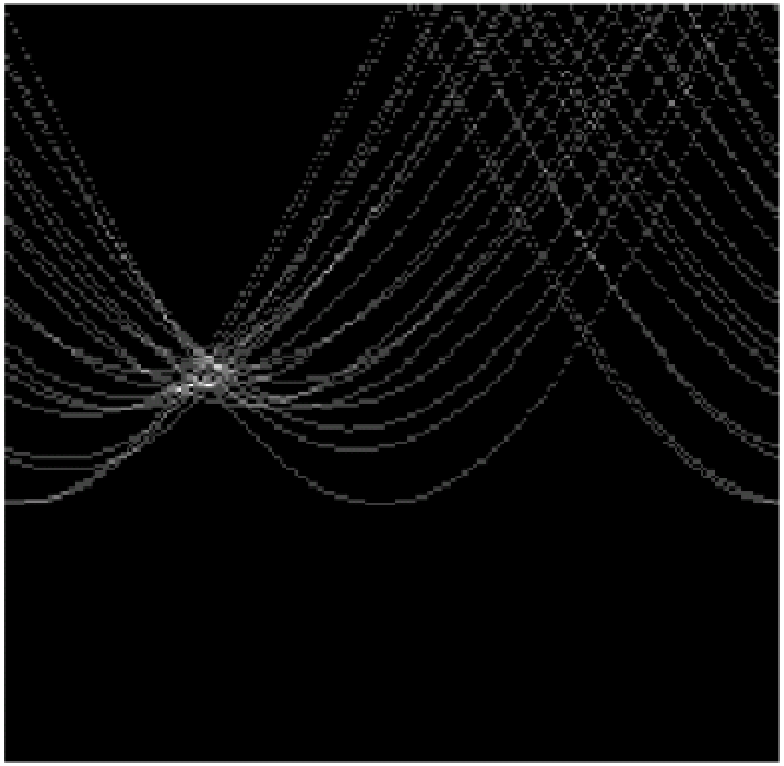
votes

Uncertainties

- Line Extraction from a point cloud
  - Algorithm 4: Hough-Transform
    - Effect of noise: peak gets fuzzy and hard to locate



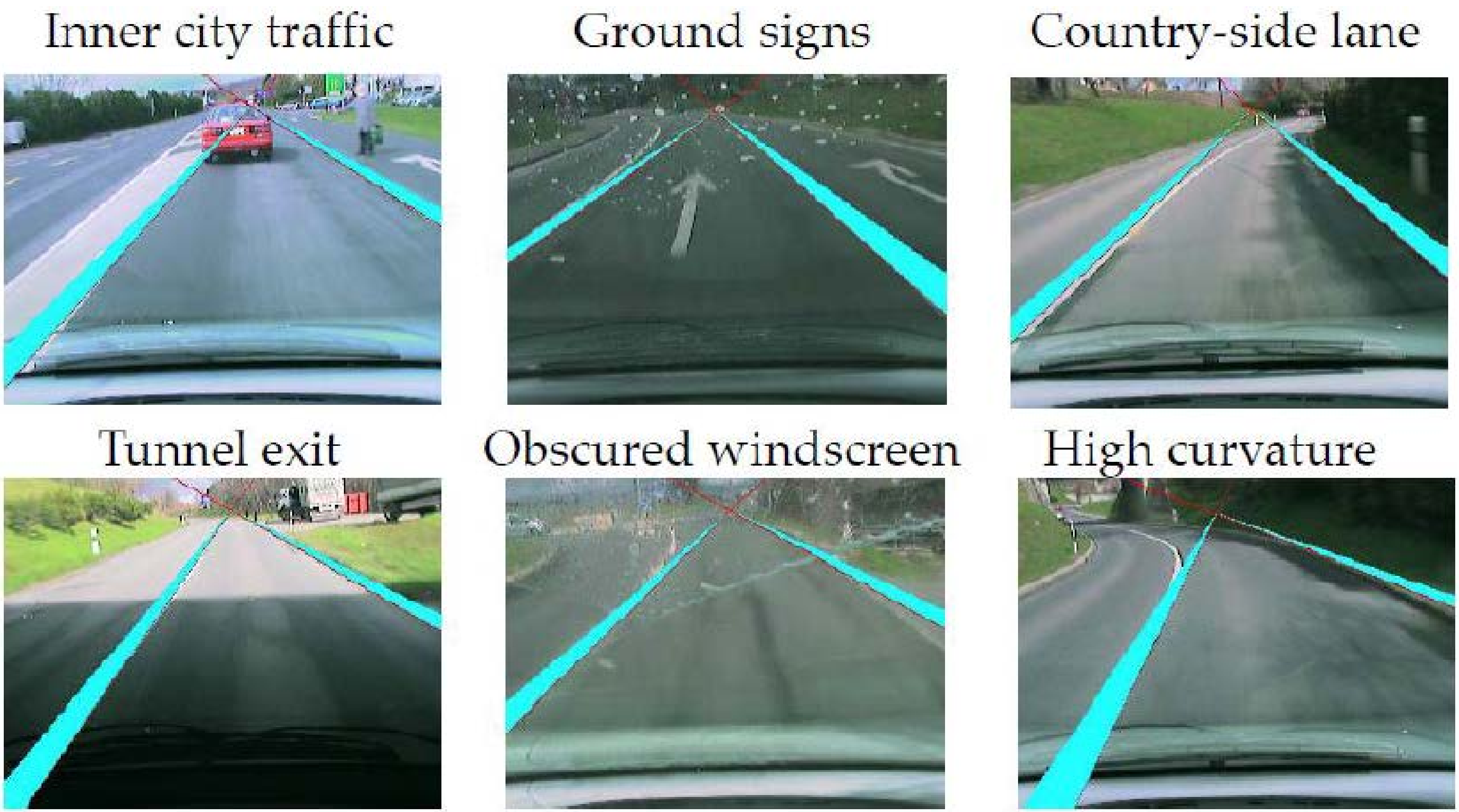
features



votes

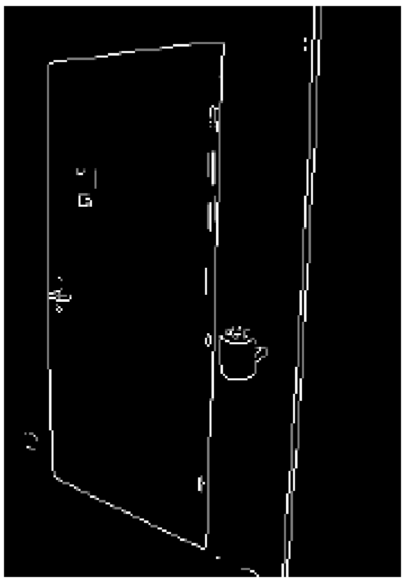
Uncertainties

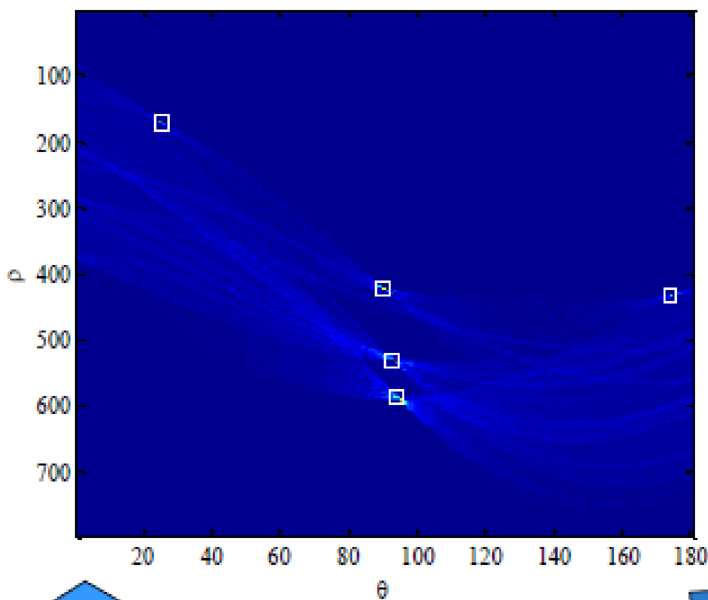
- Line Extraction from a point cloud
  - Algorithm 4: Hough-Transform
    - Example: Lane detection using HT



Uncertainties

- Line Extraction from a point cloud
  - Algorithm 4: Hough-Transform
    - Example: Door detection using HT





Hough Transform