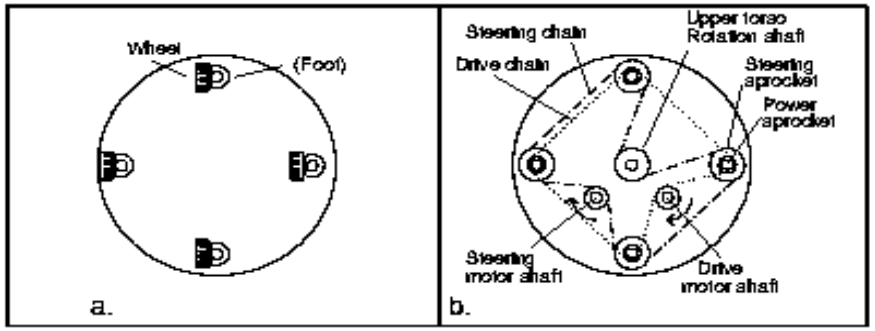
2. Sensor and Feature Modelling

Sensors for Mobile Robots

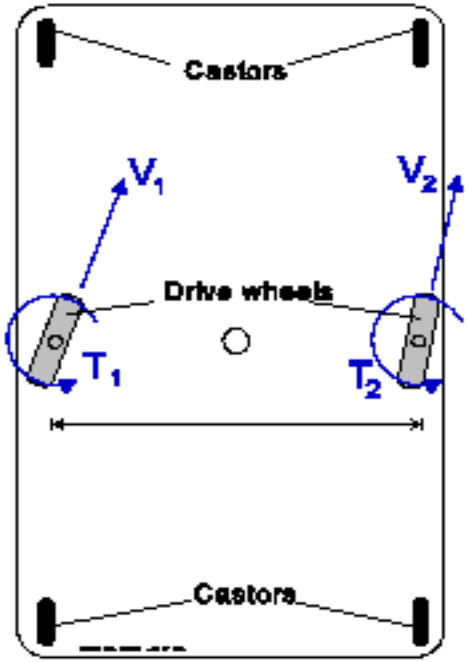
- **Contact sensors:** Bumpers
- **Internal sensors (Dead-reckoning)**
  - Wheel Encoders, Odometry
  - Accelerometers (spring-mounted masses)
  - Gyroscopes (spinning mass, laser light)
  - Compasses, inclinometers (earth magnetic field, gravity)
- **Proximity sensors**
  - Sonar (time of flight)
  - Radar (phase and frequency)
  - Laser range-finders (triangulation, ToF, phase)
  - Infrared (intensity)
- **Visual sensors:** Cameras
- **Satellite-based sensors:** GPS, Galileo?

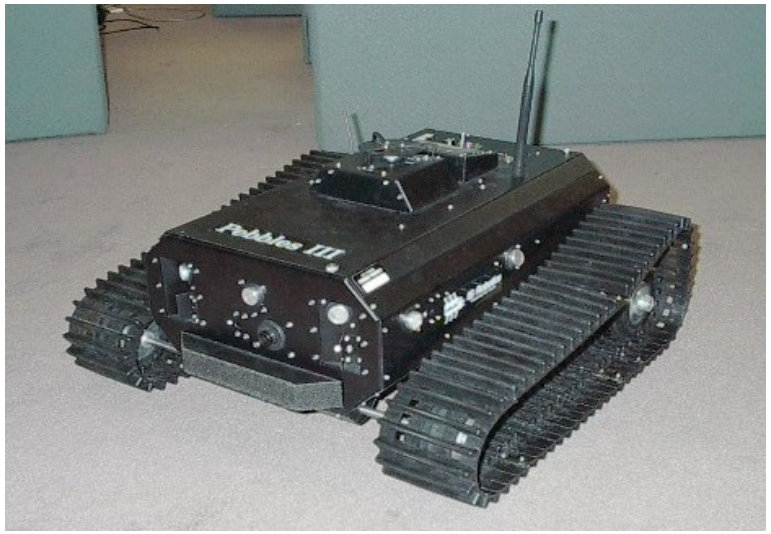
Internal Sensors: Odometry

Synchro-drive

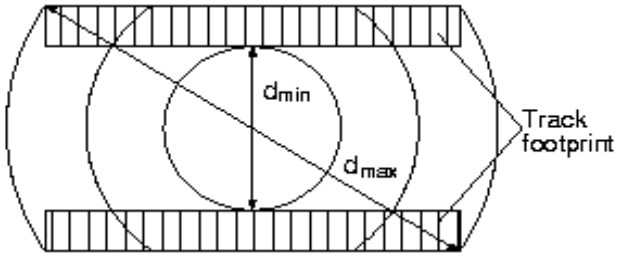


Multi-DoF vehicles





Tracked vehicles



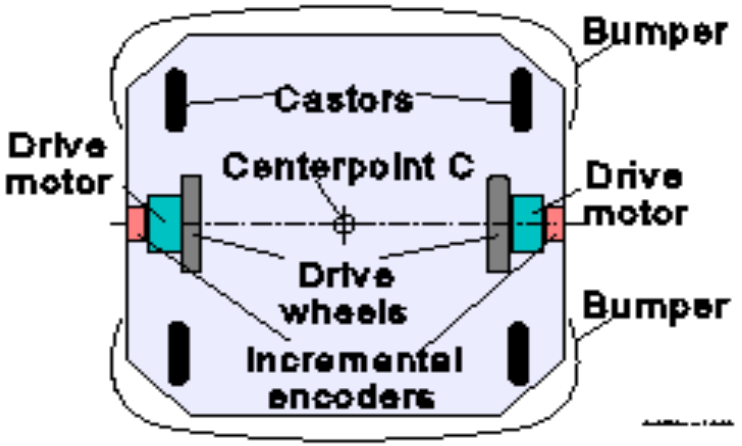
ROBOTICS
UNIVERSIDAD DE ZARAGOZA

Internal sensors: Odometry

Usual configuration: **Differential Drive**

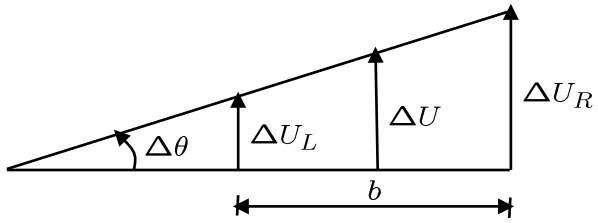
$$c_m = \frac{\pi D_n}{n C_e}$$

D_n = Nominal wheel diameter
C_e = Encoder resolution
n = Gear ratio

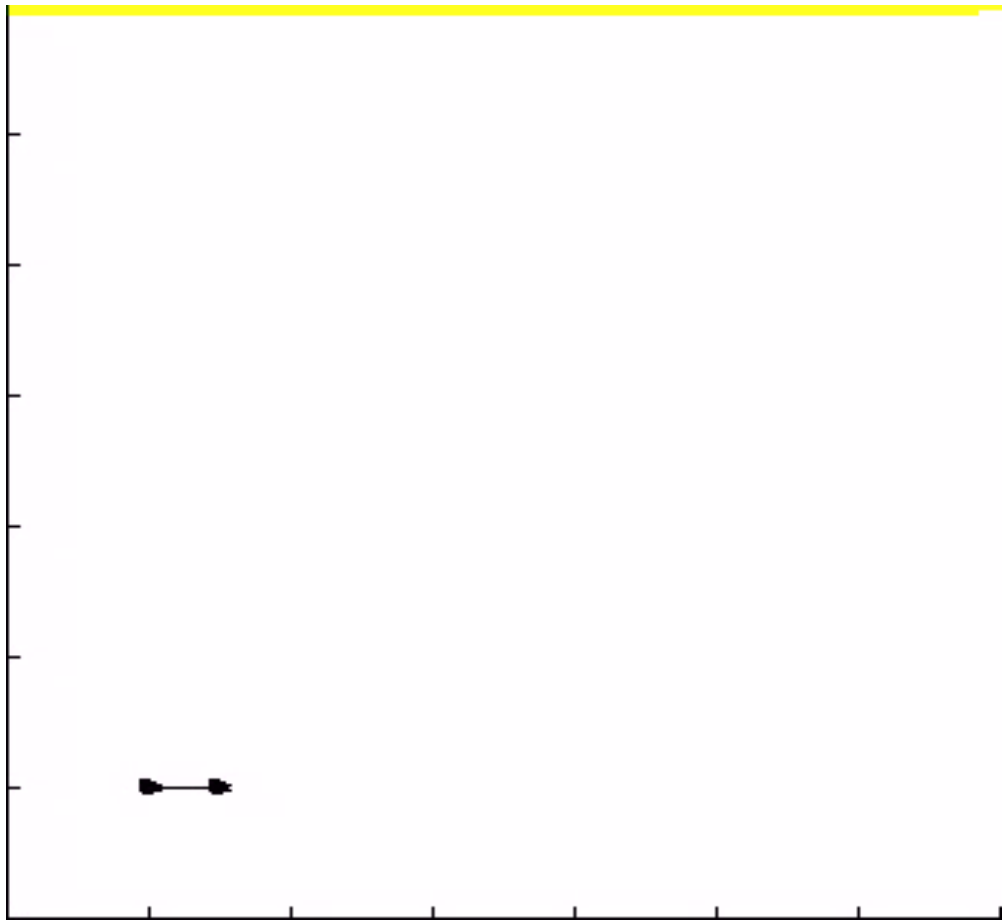


$$\left.\begin{array}{l} \Delta U_L = c_m N_L \\ \Delta U_R = c_m N_R \end{array}\right\} \Rightarrow \Delta U = \frac{\Delta U_R + \Delta U_L}{2} \;;\; \Delta\theta = \frac{\Delta U_R - \Delta U_L}{b}$$

$$\left\{\begin{array}{rcl} x_k &=& x_{k-1} + \Delta U \cos(\theta_{k-1} + \Delta\theta) \\ y_k &=& y_{k-1} + \Delta U \sin(\theta_{k-1} + \Delta\theta) \\ \theta_k &=& \theta_{k-1} + \Delta\theta \end{array}\right.$$

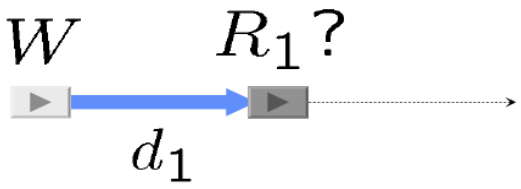


Odometry Drift



Localization using Odometry

- A robot moves in a 1D space:



Uncertainty in Robot Position

- Odometry:

$$d_1 = \hat{d}_1 + \tilde{d}_1$$

True value — Measured — Error

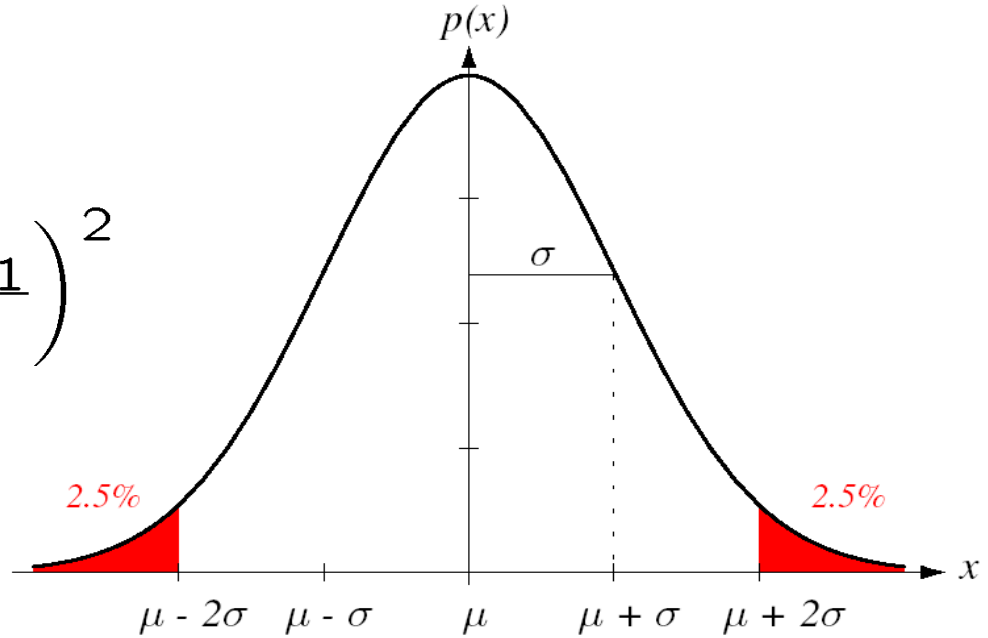
- Uncertainty model:

$$E(\tilde{d}_1) = \mu_1$$
$$\mathtt{Var}(\tilde{d}_1) = \sigma_1^2$$

Guassianity Assumption

$$\tilde{d}_1 \sim N(\mu_1, \sigma_1^2)$$

$$p(\tilde{d}_1) = \frac{1}{\sqrt{2\pi}\sigma_1} e^{-\frac{1}{2}\left(\frac{\tilde{d}_1 - \mu_1}{\sigma_1}\right)^2}$$



$$Pr\left\{ \mid \tilde{d}_1 - \mu_1 \mid \leq \sigma_1 \right\} \simeq 0.68$$
$$Pr\left\{ \mid \tilde{d}_1 - \mu_1 \mid \leq 2\sigma_1 \right\} \simeq 0.95$$
$$Pr\left\{ \mid \tilde{d}_1 - \mu_1 \mid \leq 3\sigma_1 \right\} \simeq 0.997$$

ROBOTICS
UNIVERSIDAD DE ZARAGOZA

Odometry

- Odometry error model:

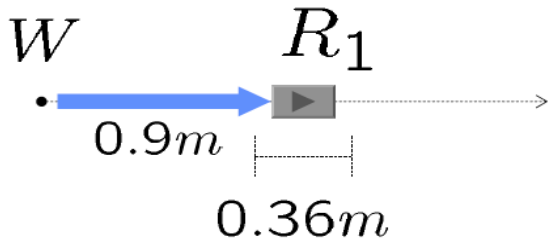
$$d_1 = \hat{d}_1 + \tilde{d}_1$$

$$\tilde{d}_1 \sim N(\mu_1, \sigma_1^2)$$

$$\mu_1 = 0$$

$$\sigma_1 = 0.1 \cdot \hat{d}_1$$

- Example: move 0.9m



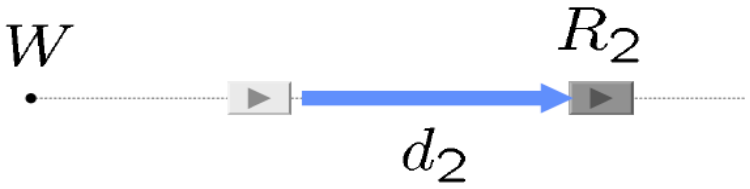
$$\hat{x}_{WR_1} = \hat{d}_1 = 0.9m$$

$$\sigma_{x_{WR_1}} = 0.09m$$

$$Pr\left\{ \mid \tilde{d}_1 \mid \leq 0.18m \right\} \simeq 0.95$$

Odometry

- Robot moves again 0.85m: $\hat{d}_2 = 0.85m$

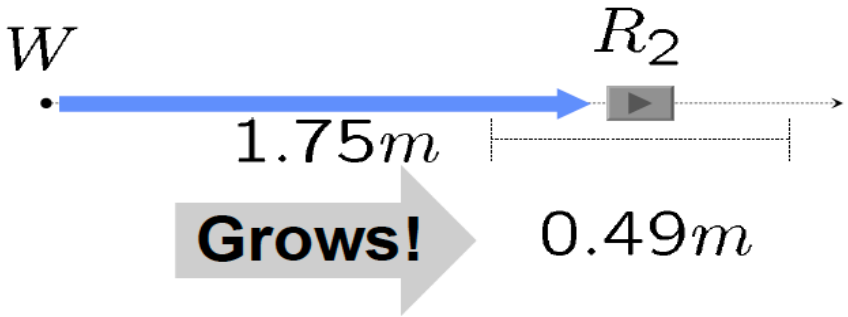


$$x_{WR_2} = \hat{d}_1 + \tilde{d}_1 + \hat{d}_2 + \tilde{d}_2$$

$$E(\tilde{d}_1 + \tilde{d}_2) = \mu_1 + \mu_2 = 0$$

$$\mathtt{Var}(\tilde{d}_1 + \tilde{d}_2) = \sigma_1^2 + \sigma_2^2 = 0.1^2\left(\hat{d}_1^2 + \hat{d}_2^2\right)$$

- New estimation:

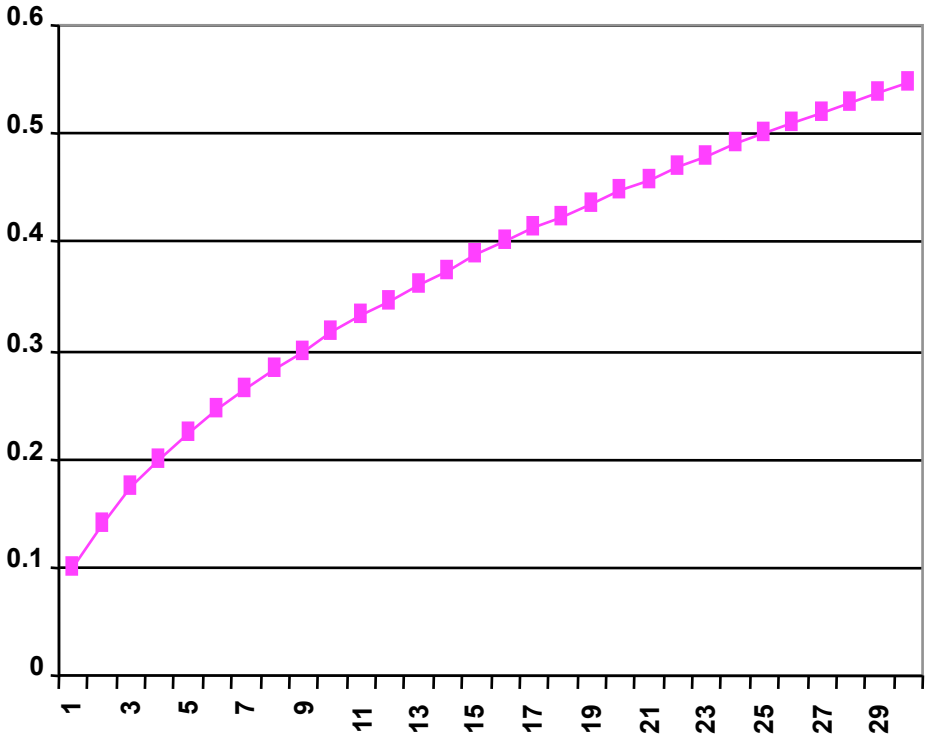


$$\hat{x}_{WR_2} = 1.75m$$

$$\sigma_{x_{WR_2}} \simeq 0.12m$$

Odometry

Odometry error grows unbounded



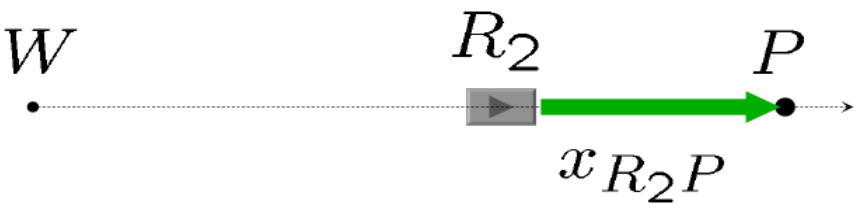
We are assuming error independence

Map-based Localization

- Assume that we know the exact location of a point P:

$$x_{WP} = 3m$$

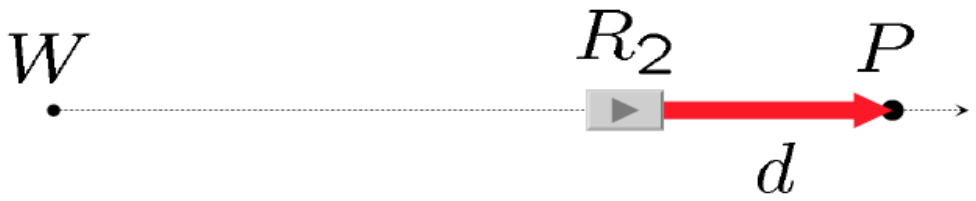
- We can predict the position of the point P relative to the robot position:



$$\begin{aligned} \hat{x}_{R_2P} &= -\hat{x}_{WR_2} + x_{WP} \\ &= 1.25m \\ \sigma_{x_{R_2P}} &= \sigma_{x_{WR_2}} \\ &\simeq 0.12m \end{aligned}$$

Environment Perception

- Assume that the robot has a sensor that measures the distance to the point, with some uncertainty, proportional to the distance:



Sensor measure:

$$\begin{aligned} \hat{d} &= 1.1m \\ \mu_d &= 0 \\ \sigma_d &= 0.01 \cdot \hat{d} \\ &= 0.011m \end{aligned}$$

Predicted value:

$$\begin{aligned} \hat{x}_{R_2P} &= 1.25m \\ \sigma_{x_{R_2P}} &\simeq 0.12m \end{aligned}$$

We can get a better estimation of robot position

Estimation Problem

- State to estimate: robot location

$$\begin{aligned} x &= x_{WR_2} \\ P &= \mathtt{Cov}(x) \end{aligned}$$

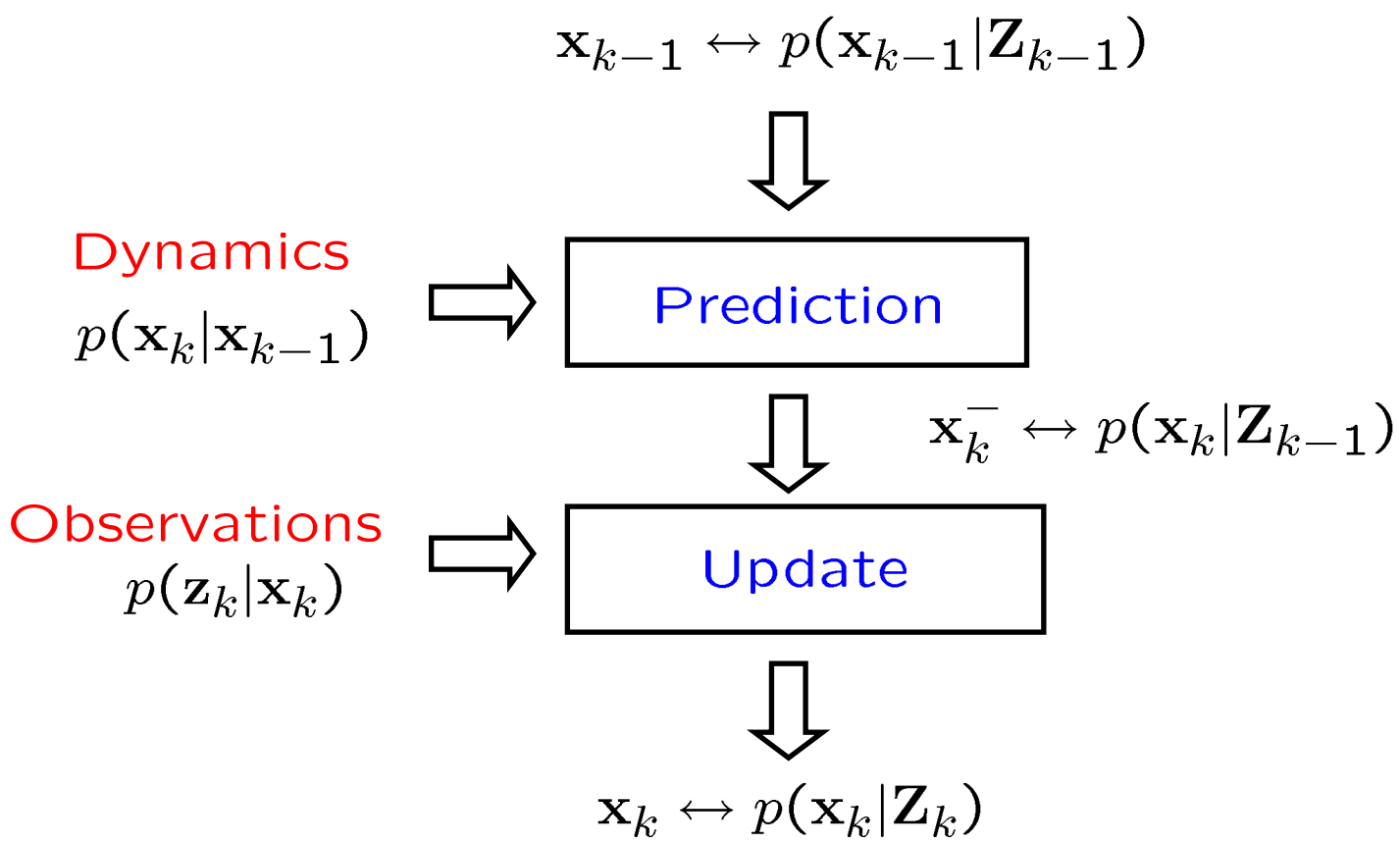
- Prediction: given by odometry

$$\begin{aligned} \hat{x}^- &= \hat{x}_{WR_2} = 1.75m \\ P^- &= \sigma^2_{x_{WR_2}} = 0.0144m^2 \end{aligned}$$

- Sensor measurement: distance to the point

$$\begin{aligned} z &= \hat{d} + \mu_d = 1.1m \\ R &= \sigma^2_d = 0.000121m^2 \end{aligned}$$

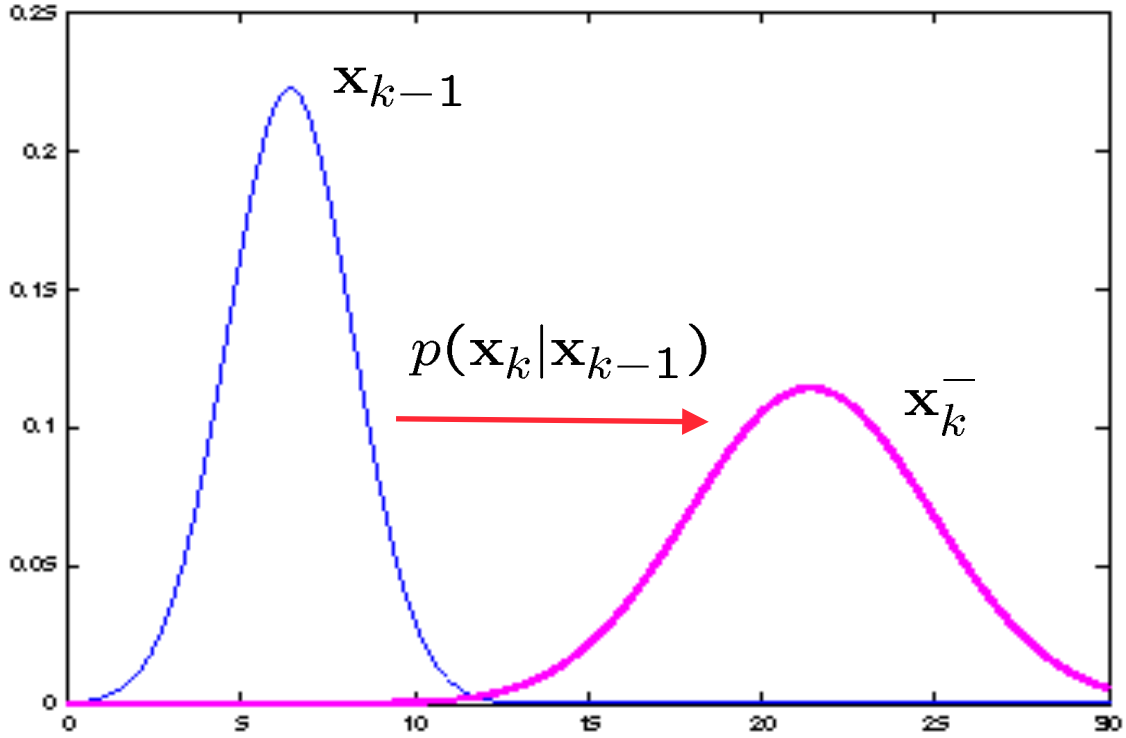
Kalman Filter



Nonlinear models: Extended / Unscented Kalman Filter (EKF / UKF)

1D Kalman Filter

- Prediction step:

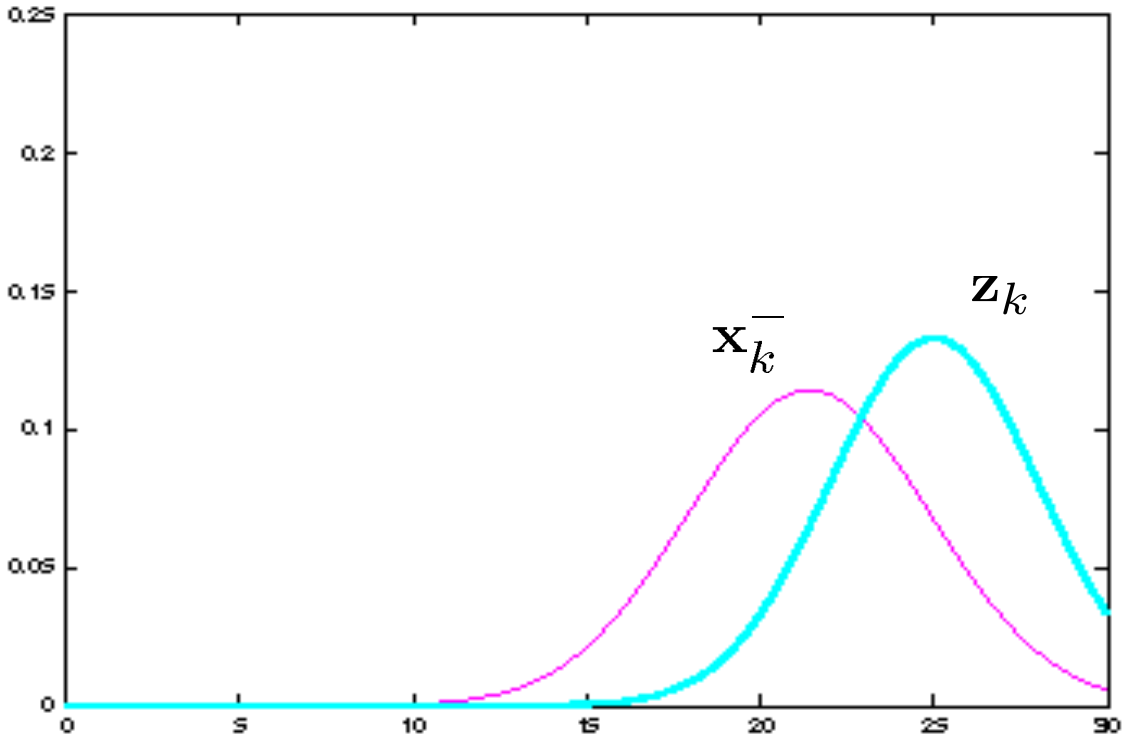


$$\bar{\mathbf{x}}_k = \mathbf{f}_k(\mathbf{x}_{k-1}) + \mathbf{v}_k \Rightarrow \begin{cases} \bar{\mathbf{x}}_k = \mathbf{F}_k\mathbf{x}_{k-1} + \mathbf{v}_k & ; \text{ linear} \\ \bar{\mathbf{x}}_k \simeq \mathbf{F}_k\mathbf{x}_{k-1} + \mathbf{v}_k & ; \text{ nonlinear} \end{cases}$$

$$\mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_k)$$

1D Kalman Filter

- Measurement:

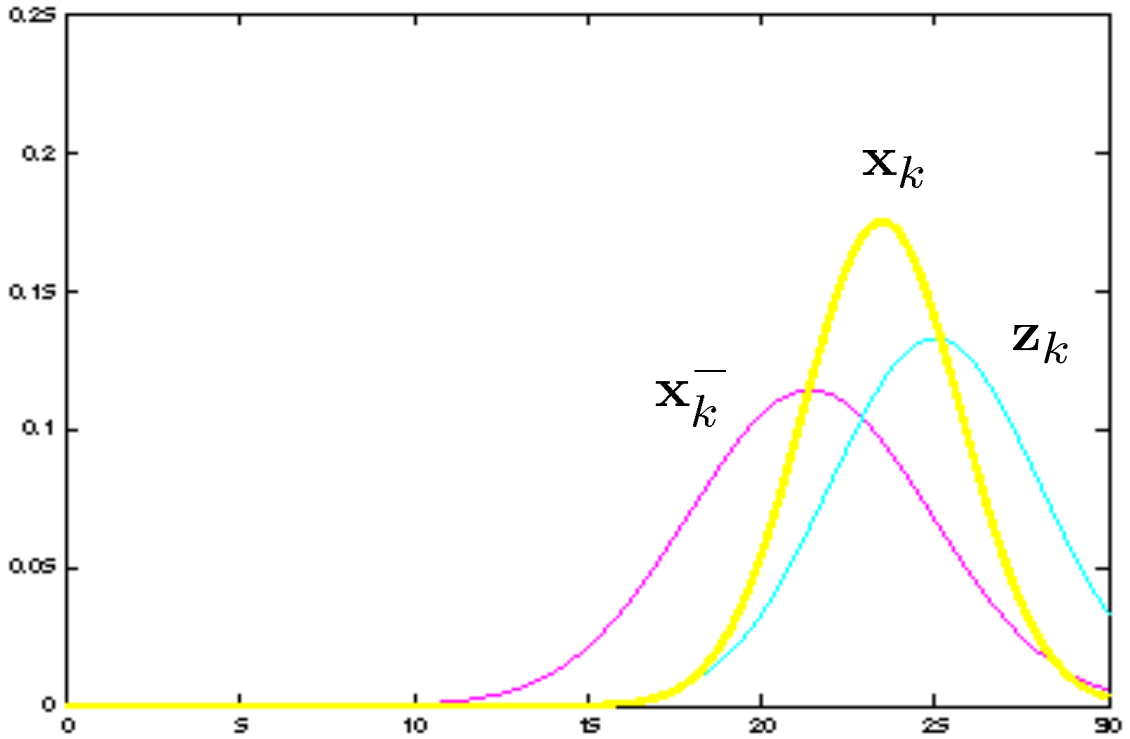


$$\mathbf{z}_k = \mathbf{h}_k(\mathbf{x}_k^-) + \mathbf{w}_k \Rightarrow \begin{cases} \mathbf{z}_k = \mathbf{H}_k\mathbf{x}_k^- + \mathbf{w}_k & ; \text{ linear} \\ \mathbf{z}_k \simeq \mathbf{H}_k\mathbf{x}_k^- + \mathbf{w}_k & ; \text{ nonlinear} \end{cases}$$

$$\mathbf{w}_k \sim \mathcal{N}(0, \mathbf{R}_k)$$

1D Kalman Filter

- Update step:



$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}_k\hat{\mathbf{x}}_k^-)$$

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_k^-$$

$$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}_k^T(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^T + \mathbf{R}_k)^{-1}$$

Kalman Filtering

- State to estimate: robot location

$$\begin{aligned} x &= x_{WR_2} \\ P &= \mathtt{Cov}(x) \end{aligned}$$

- Prediction: given by odometry

$$\begin{aligned} \widehat{x}^- &= \widehat{x}_{WR_2} = 1.75m \\ P^- &= \sigma^2_{x_{WR_2}} = 0.0144m^2 \end{aligned}$$

- Sensor measurement: distance to the point

$$\begin{aligned} z &= \widehat{d} + \mu_d = 1.1m \\ R &= \sigma^2_d = 0.000121m^2 \end{aligned}$$

Kalman Filtering

- Measurement equation:

$$z = d = -x_{WR_2} + x_{WP}$$

- Filter gain: $K = P^-(P^- + R)^{-1} = 0.991667241$

- Estimation update:

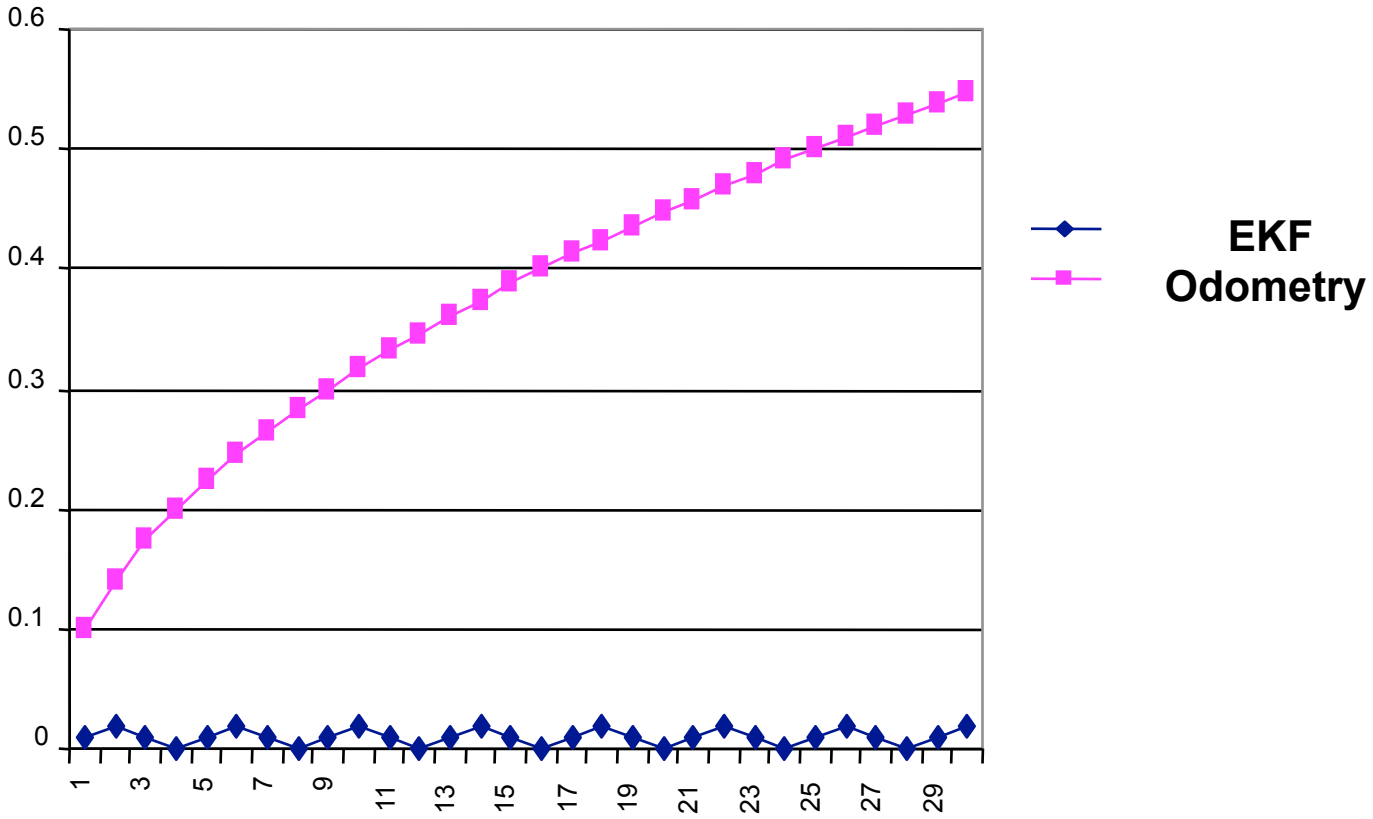
$$\begin{aligned} \hat{x} &= \hat{x}^- + K(1.1 - 1.25) \\ &= 1.899m \end{aligned} \qquad \begin{aligned} P &= (I - K)P^- \\ &= 0.000120m^2 \\ \sigma &= 0.011m \end{aligned}$$

- Compare with:

$$\hat{x}^- = 1.75m \qquad \sigma^- = 0.12m$$

Better Robot Estimation

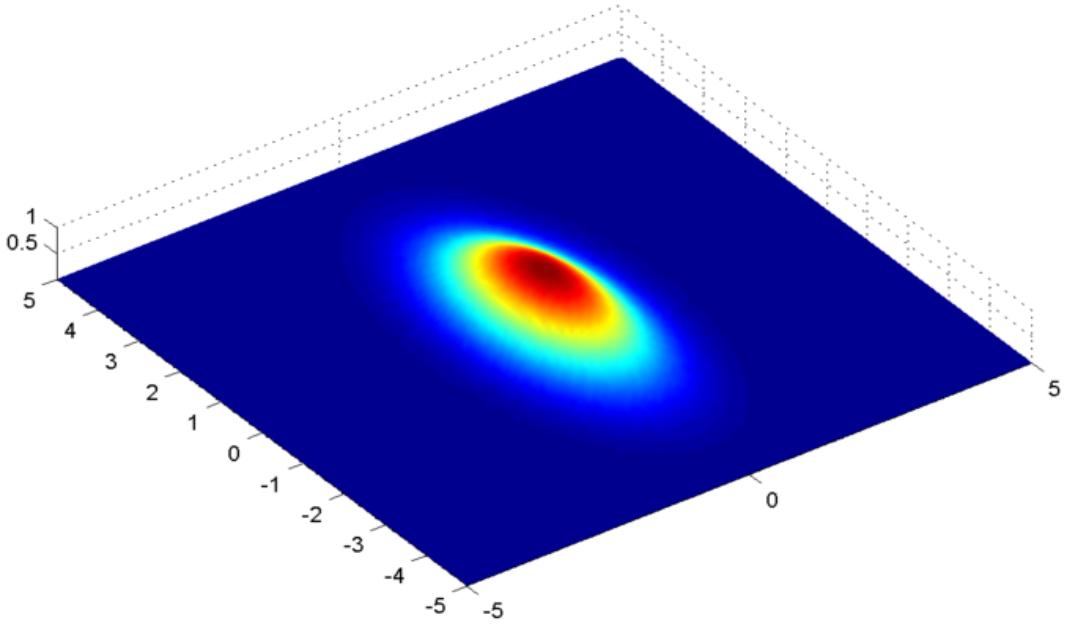
Bounded Robot Error

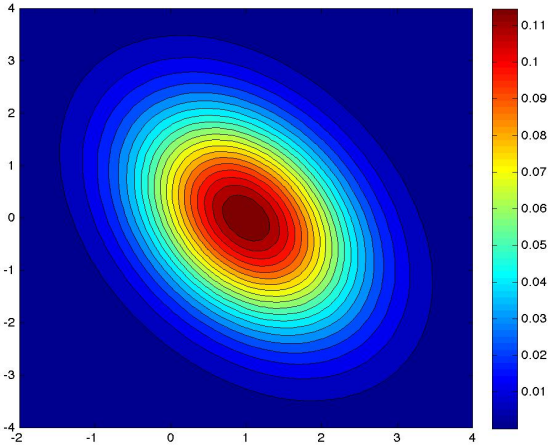


Moving close to point P

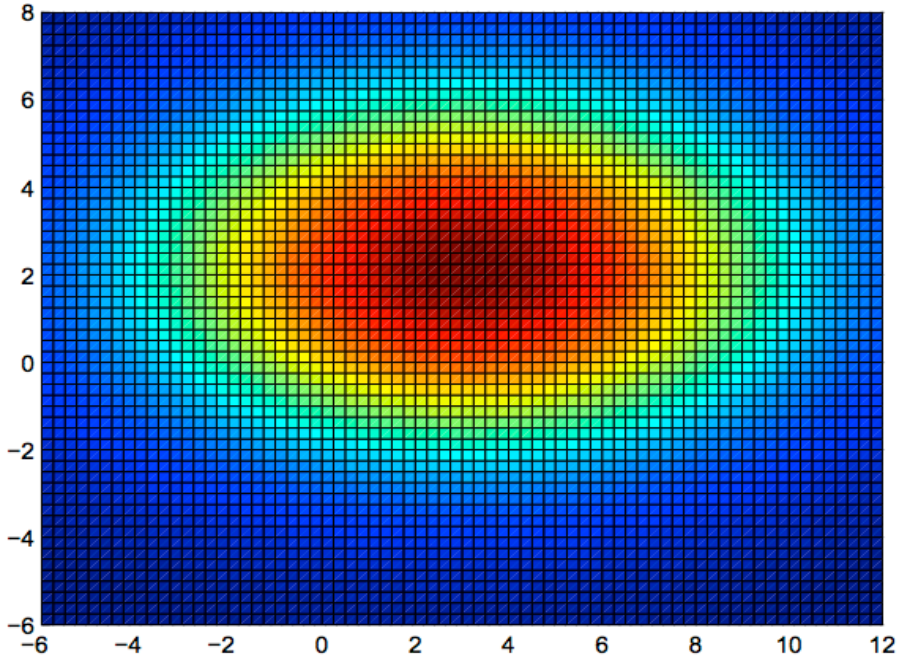
Multivariate Gaussian Uncertainty

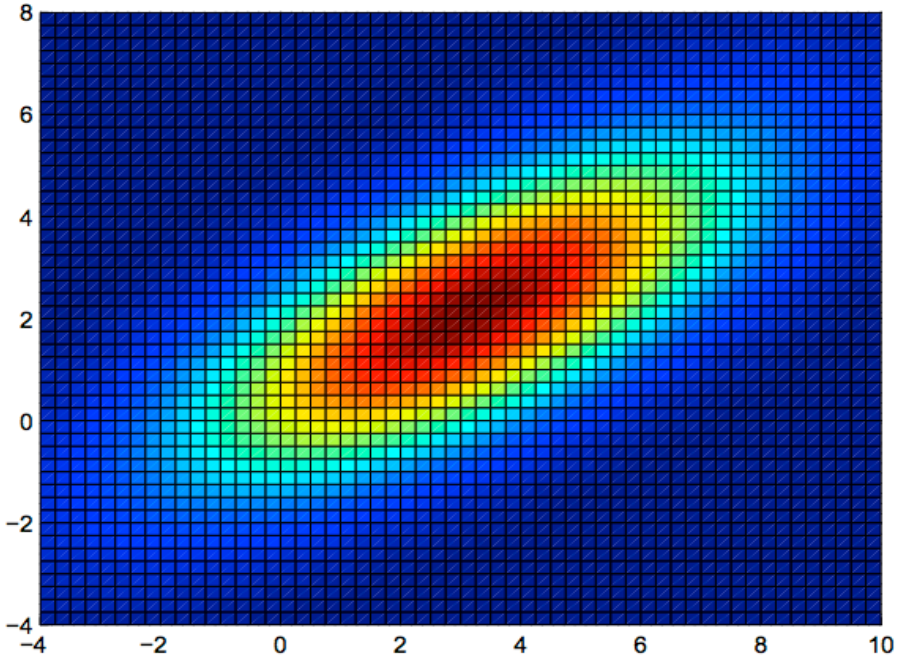
$$p(x;\mu,\Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$$





Correlation



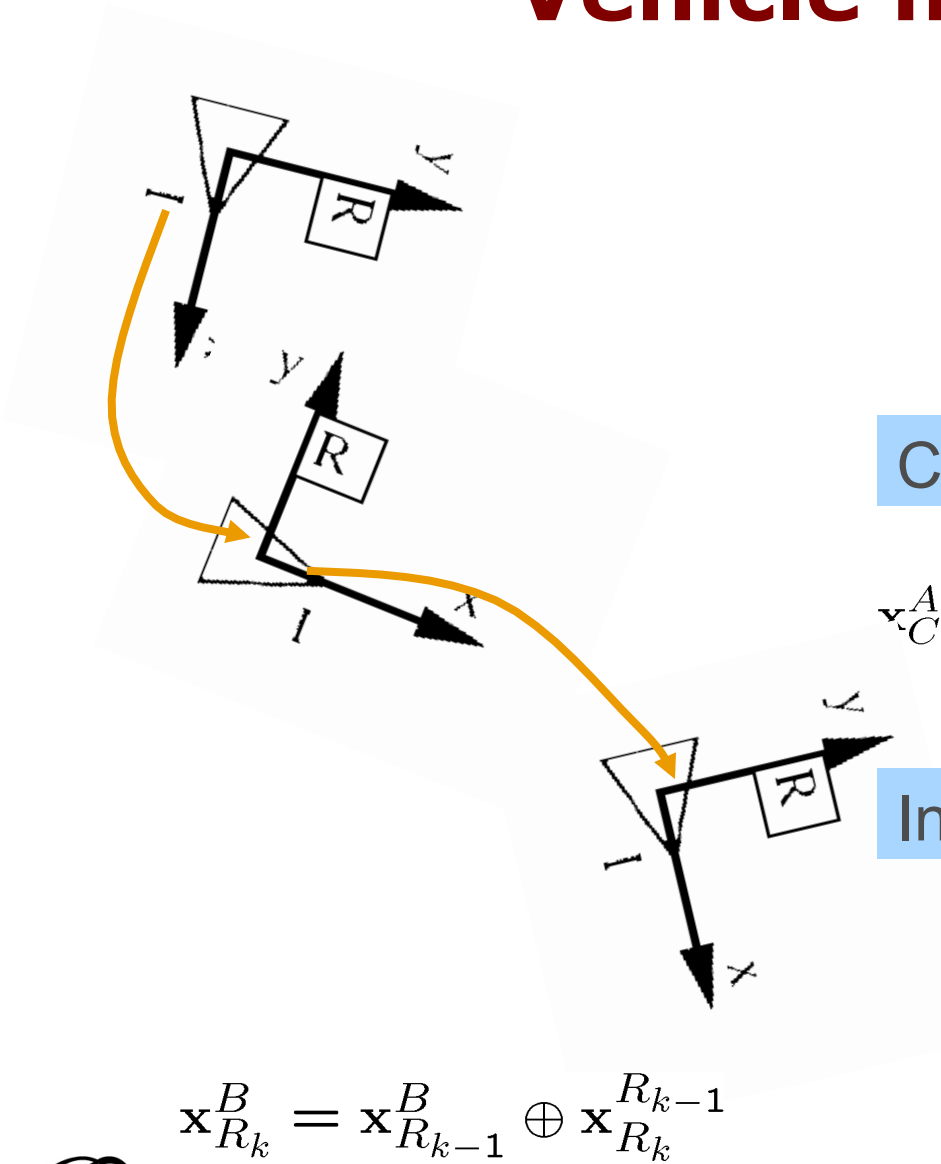


$$\Sigma = \begin{bmatrix} 25 & 0 \\ 0 & 9 \end{bmatrix}$$

$$\Sigma = \begin{bmatrix} 10 & 5 \\ 5 & 5 \end{bmatrix}$$

ROBOTICS
UNIVERSIDAD DE ZARAGOZA

Vehicle motion in 2D



$$\mathbf{x}_B^A = \begin{bmatrix} x_1 \\ y_1 \\ \phi_1 \end{bmatrix} \qquad \mathbf{x}_C^B = \begin{bmatrix} x_2 \\ y_2 \\ \phi_2 \end{bmatrix}$$

Composition:

$$\mathbf{x}_C^A = \mathbf{x}_B^A \oplus \mathbf{x}_C^B = \begin{bmatrix} x_1 + x_2 \cos\phi_1 - y_2 \sin\phi_1 \\ y_1 + x_2 \sin\phi_1 + y_2 \cos\phi_1 \\ \phi_1 + \phi_2 \end{bmatrix}$$

Inversion:

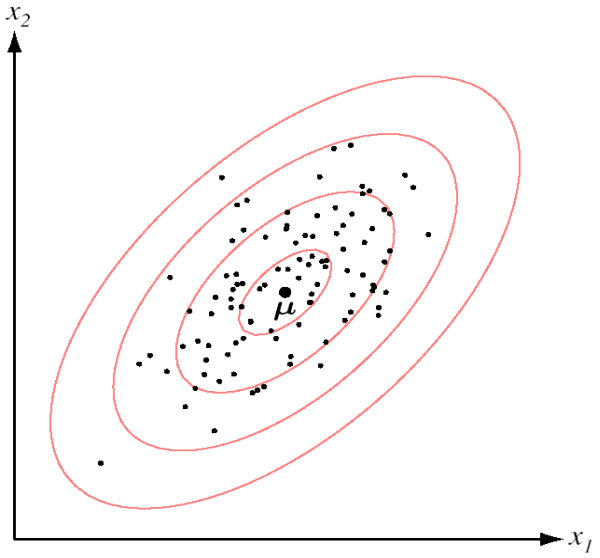
$$= \ominus \mathbf{x}_B^A = \begin{bmatrix} -x_1 \cos\phi_1 - y_1 \sin\phi_1 \\ x_1 \sin\phi_1 - y_1 \cos\phi_1 \\ -\phi_1 \end{bmatrix}$$

$$\mathbf{x}_{R_k}^B = \mathbf{x}_{R_{k-1}}^B \oplus \mathbf{x}_{R_k}^{R_{k-1}}$$

Odometry in 2D

Odometry model:

$$\begin{aligned} \mathbf{x}_{R_k}^{R_{k-1}} &= \hat{\mathbf{x}}_{R_k}^{R_{k-1}} + \mathbf{v}_k \\ E[\mathbf{v}_k] &= \mathbf{0} \\ E[\mathbf{v}_k \mathbf{v}_j^T] &= \delta_{kj} \mathbf{Q}_k \end{aligned}$$



Composition:

$$\begin{aligned} \hat{\mathbf{x}}_{R_k}^{B} &= \hat{\mathbf{x}}_{R_{k-1}}^{B} \oplus \hat{\mathbf{x}}_{R_k}^{R_{k-1}} \\ \mathbf{P}_{R_k} &\simeq J_1 \, \mathbf{P}_{R_{k-1}} J_1^T + J_2 \, \mathbf{Q}_k J_2^T \end{aligned}$$

$$J_1 = \left. \frac{\partial \left(\mathbf{x}_{R_{k-1}}^{B} \oplus \mathbf{x}_{R_k}^{R_{k-1}} \right)}{\partial \mathbf{x}_{R_{k-1}}^{B}} \right|_{(\hat{\mathbf{x}}_{R_{k-1}}^{B}, \hat{\mathbf{x}}_{R_k}^{R_{k-1}})}$$

$$J_2 = \left. \frac{\partial \left(\mathbf{x}_{R_{k-1}}^{B} \oplus \mathbf{x}_{R_k}^{R_{k-1}} \right)}{\partial \mathbf{x}_{R_k}^{R_{k-1}}} \right|_{(\hat{\mathbf{x}}_{R_{k-1}}^{B}, \hat{\mathbf{x}}_{R_k}^{R_{k-1}})}$$

Odometry in 2D

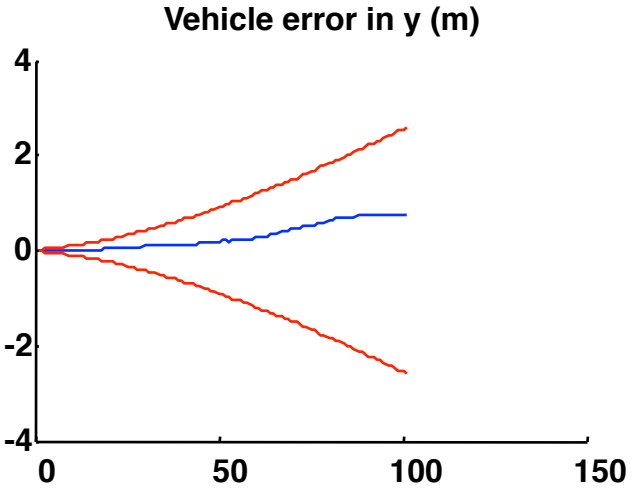
Jacobians:

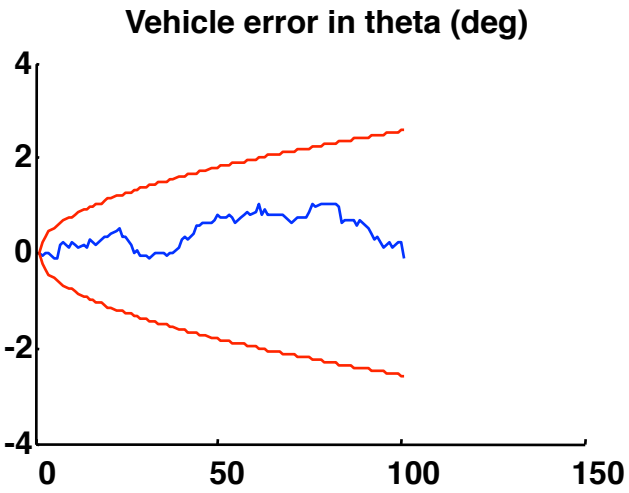
$$J_{1\oplus}\{\mathbf{x}_B^A, \mathbf{x}_C^B\} = \left.\frac{\partial\left(\mathbf{x}_B^A \oplus \mathbf{x}_C^B\right)}{\partial \mathbf{x}_B^A}\right|_{(\hat{\mathbf{x}}_B^A, \hat{\mathbf{x}}_C^B)} = \begin{bmatrix} 1 & 0 & -x_2 \sin\phi_1 - y_2 \cos\phi_1 \\ 0 & 1 & x_2 \cos\phi_1 - y_2 \sin\phi_1 \\ 0 & 0 & 1 \end{bmatrix}$$

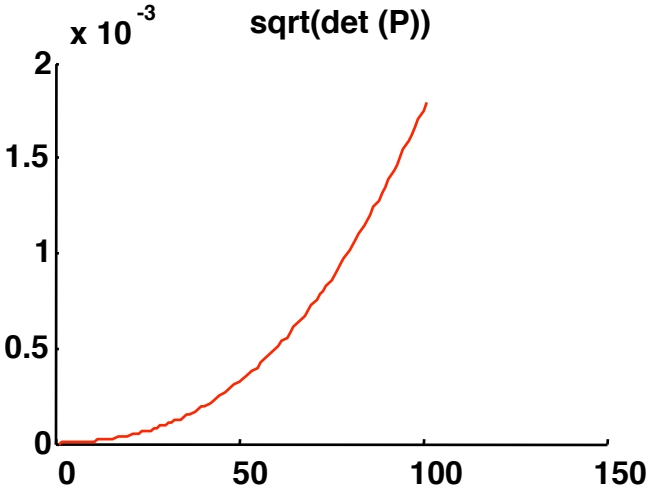
$$J_{2\oplus}\{\mathbf{x}_B^A, \mathbf{x}_C^B\} = \left.\frac{\partial\left(\mathbf{x}_B^A \oplus \mathbf{x}_C^B\right)}{\partial \mathbf{x}_C^B}\right|_{(\hat{\mathbf{x}}_B^A, \hat{\mathbf{x}}_C^B)} = \begin{bmatrix} \cos\phi_1 & -\sin\phi_1 & 0 \\ \sin\phi_1 & \cos\phi_1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$J_{\ominus}\{\mathbf{x}_B^A\} = \left.\frac{\partial\left(\ominus \mathbf{x}_B^A\right)}{\partial \mathbf{x}_B^A}\right|_{(\hat{\mathbf{x}}_B^A)} = \begin{bmatrix} -\cos\phi_1 & -\sin\phi_1 & -x_1 \sin\phi_1 - y_1 \cos\phi_1 \\ \sin\phi_1 & -\cos\phi_1 & x_1 \cos\phi_1 + y_1 \sin\phi_1 \\ 0 & 0 & -1 \end{bmatrix}$$

Dead-reckoning, moving forward



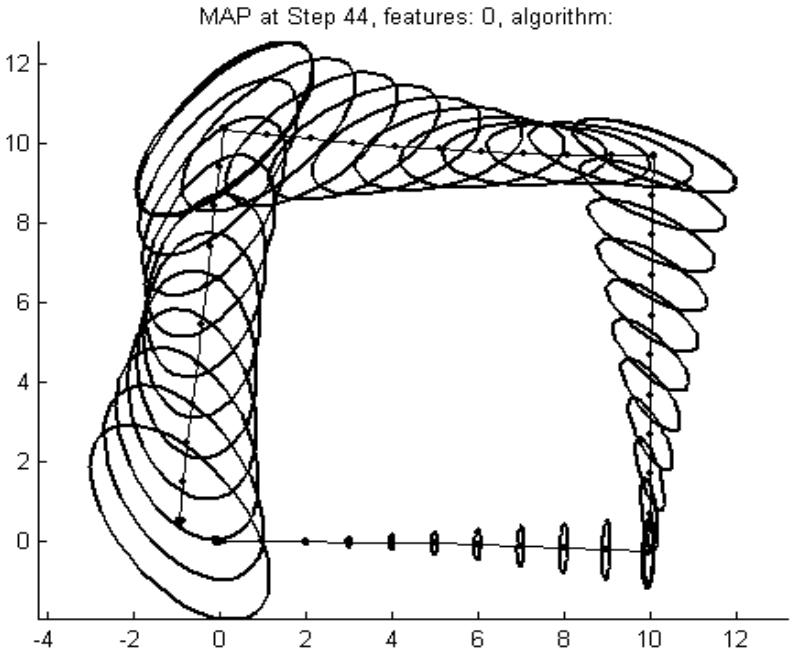


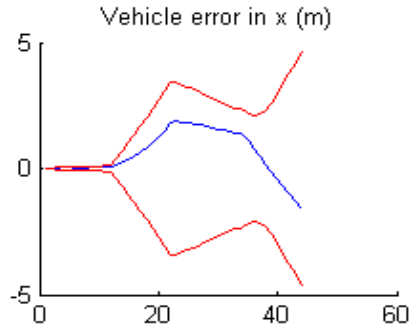


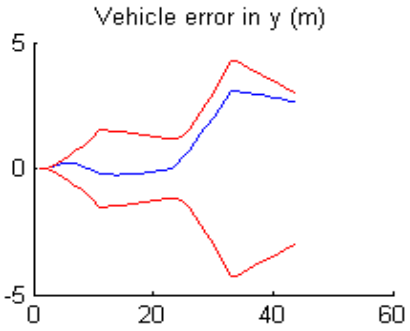
ROBOTICS
UNIVERSIDAD DE ZARAGOZA

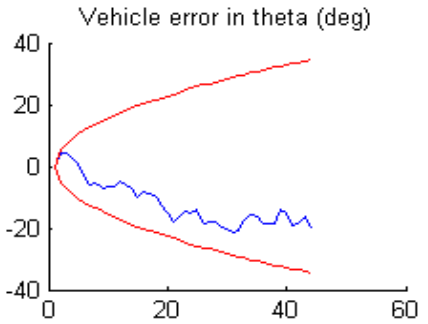
The need for SLAM

Error$\pm 2\sigma$ (prob. 0.95)





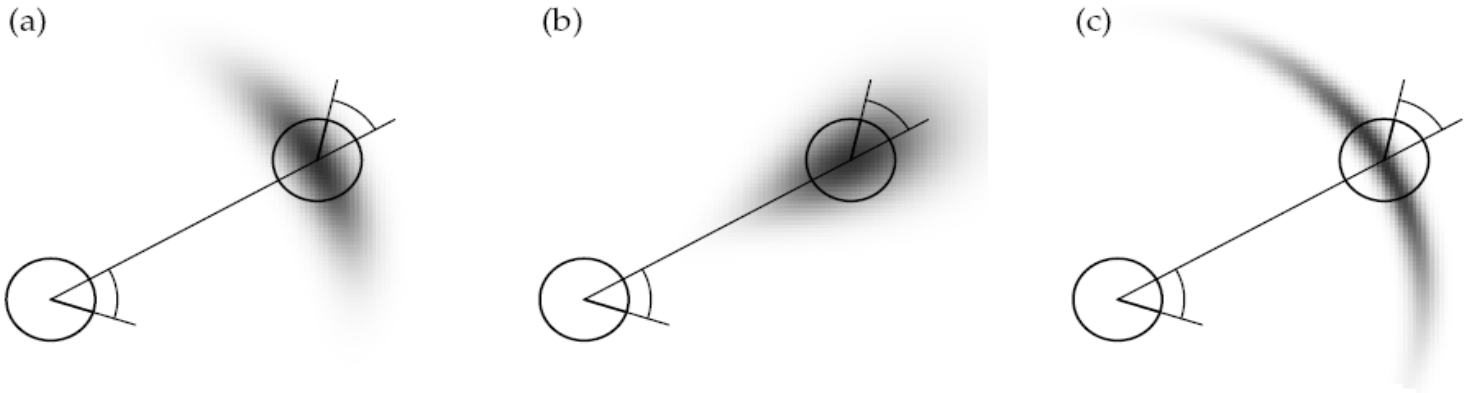




Odometry in 2D

$$\mathbf{x}_{R_k}^{R_{k-1}} = (0, 0, \phi_1)^t \oplus (x_1, 0, 0)^t \oplus (0, 0, \phi_2)^t$$

For different rotation and translational errors

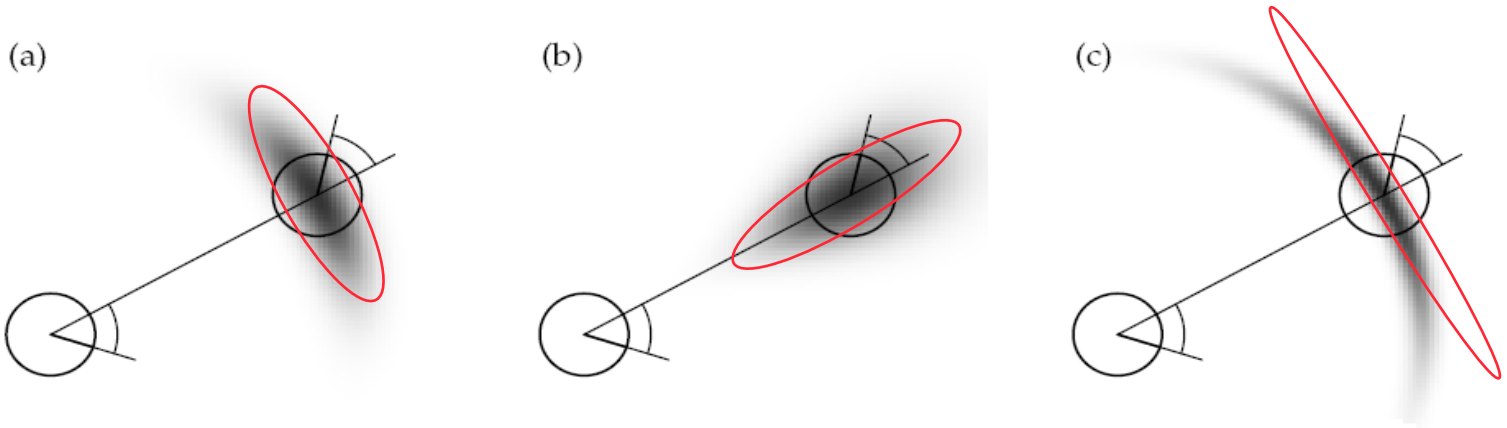


(image: Thrun, Burgard, Fox)

Odometry in 2D

$$\mathbf{x}_{R_k}^{R_{k-1}} = (0, 0, \phi_1)^t \oplus (x_1, 0, 0)^t \oplus (0, 0, \phi_2)^t$$

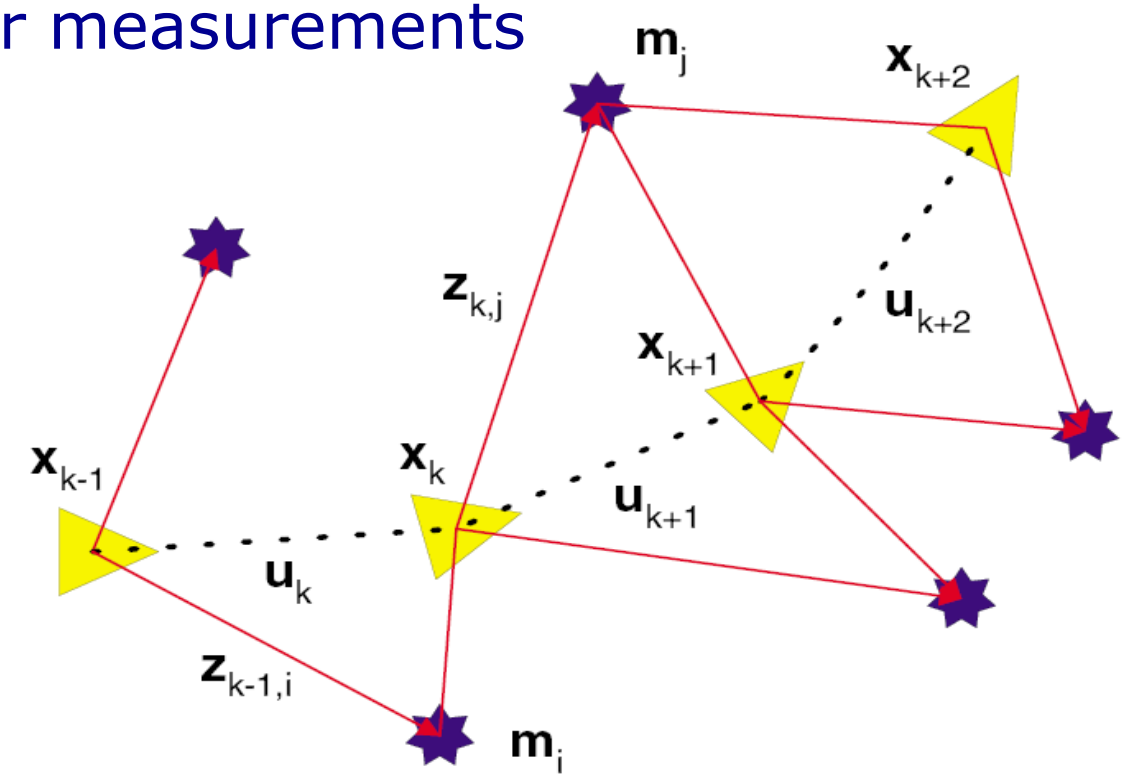
We are linearizing errors!



(image: Thrun, Burgard, Fox)

Localization and Mapping elements

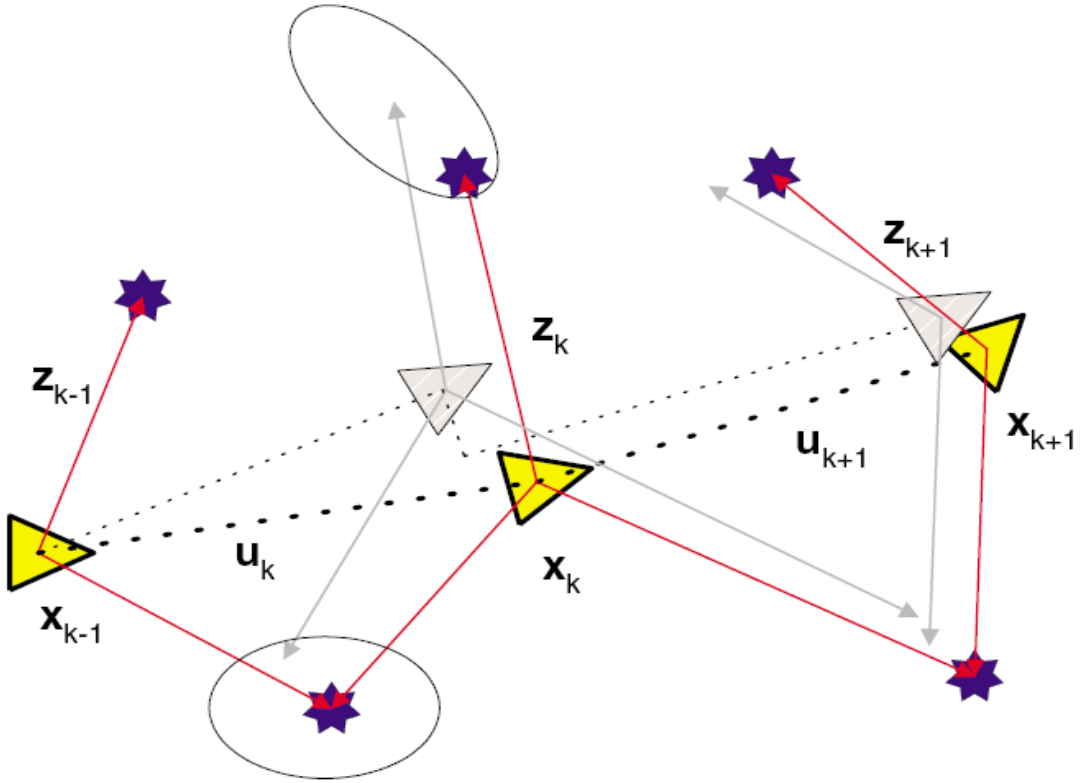
- M: environment features
- U: control inputs
- X: vehicle locations
- Z: sensor measurements



(image: Durrant-Whyte)

Map-based localization

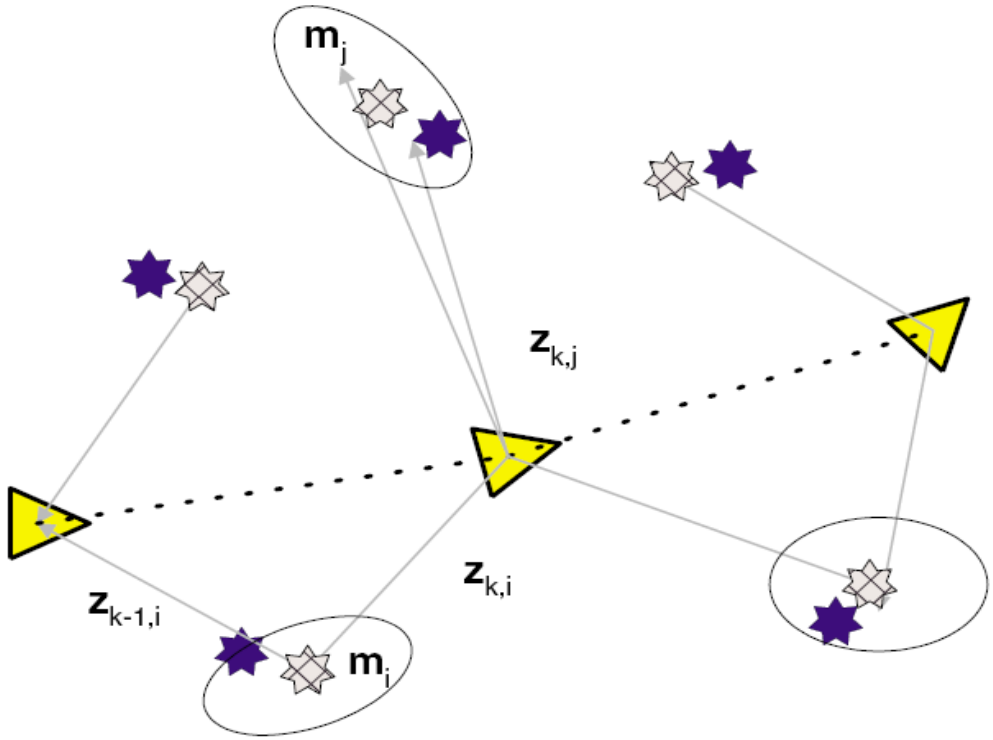
- Given M, U, Z
- Compute X



(image: Durrant-Whyte)

Mapping

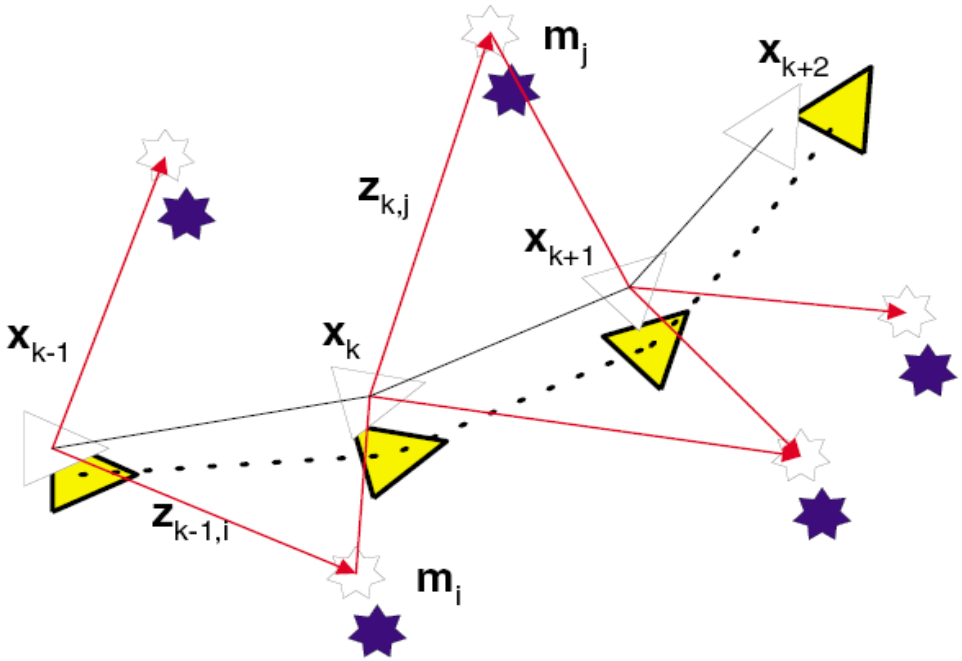
- Given X, Z
- Compute M



(image: Durrant-Whyte)

SLAM

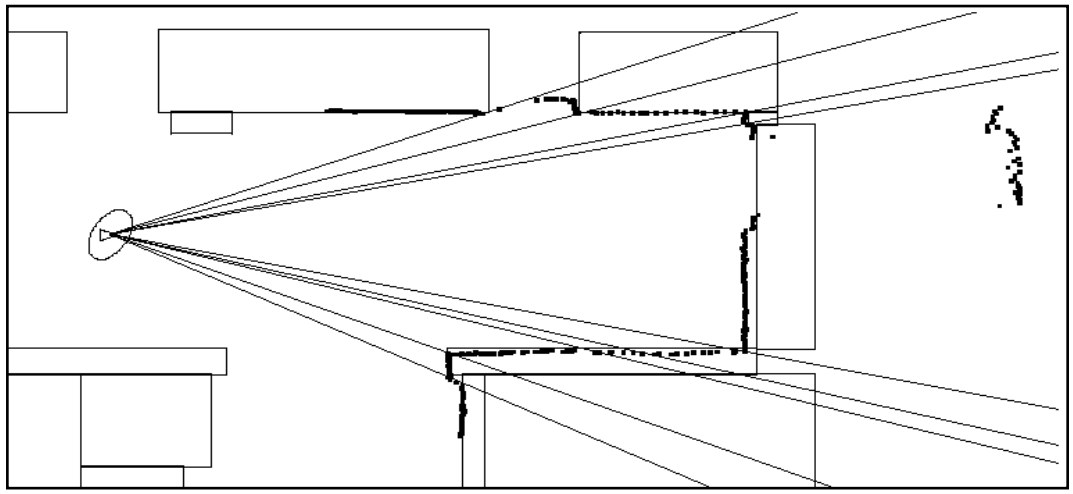
- Given U, Z
- Compute X, M



(image: Durrant-Whyte)

The need for SLAM

- In many applications the environment is unknown
- A priori maps usually are:
  - Costly to obtain
  - Inaccurate
  - Incomplete
  - Out of date



➔ SLAM

External sensors: Laser

TRC LightRanger

SICK LMS200
- 180º coverage
- 0.5º resolution
- 10 mm resolution
- up to 80 m
- up to 75 Hz
- indoor applications

180º

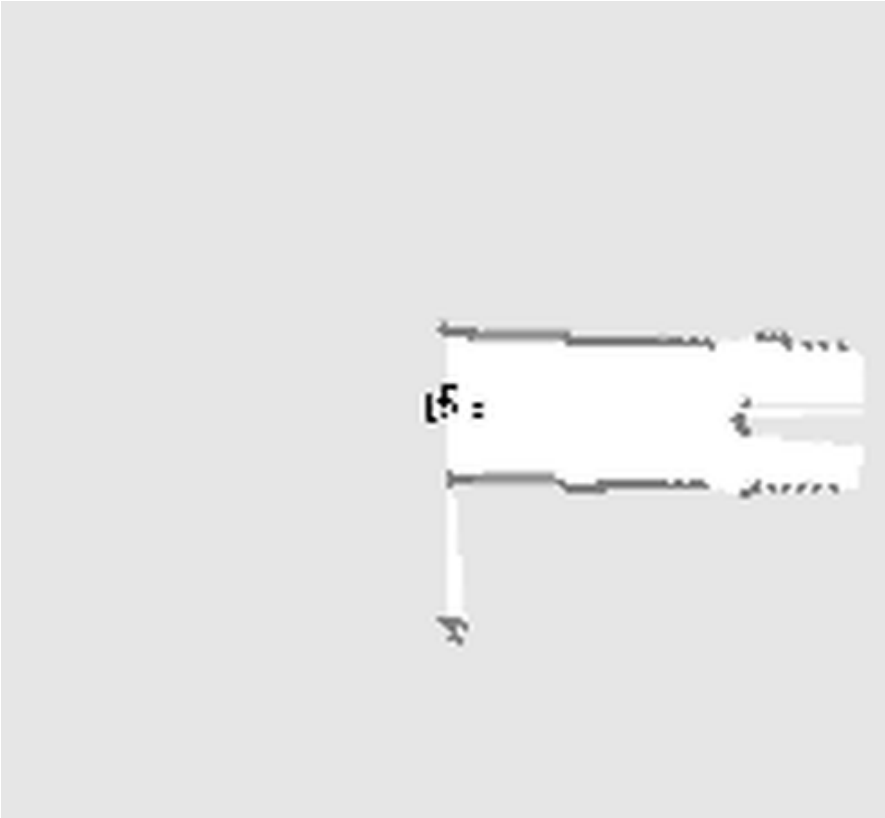
Raw data: $\{(\rho_1, \phi_1)^T, ..., (\rho_n, \phi_n)^T\}$

LMS291

PLS101/201

ROBOTICS
UNIVERSIDAD DE ZARAGOZA

Laser



- Obtain line segments from a laser scan:
  - Segmentation
  - Line estimation

Split and Merge

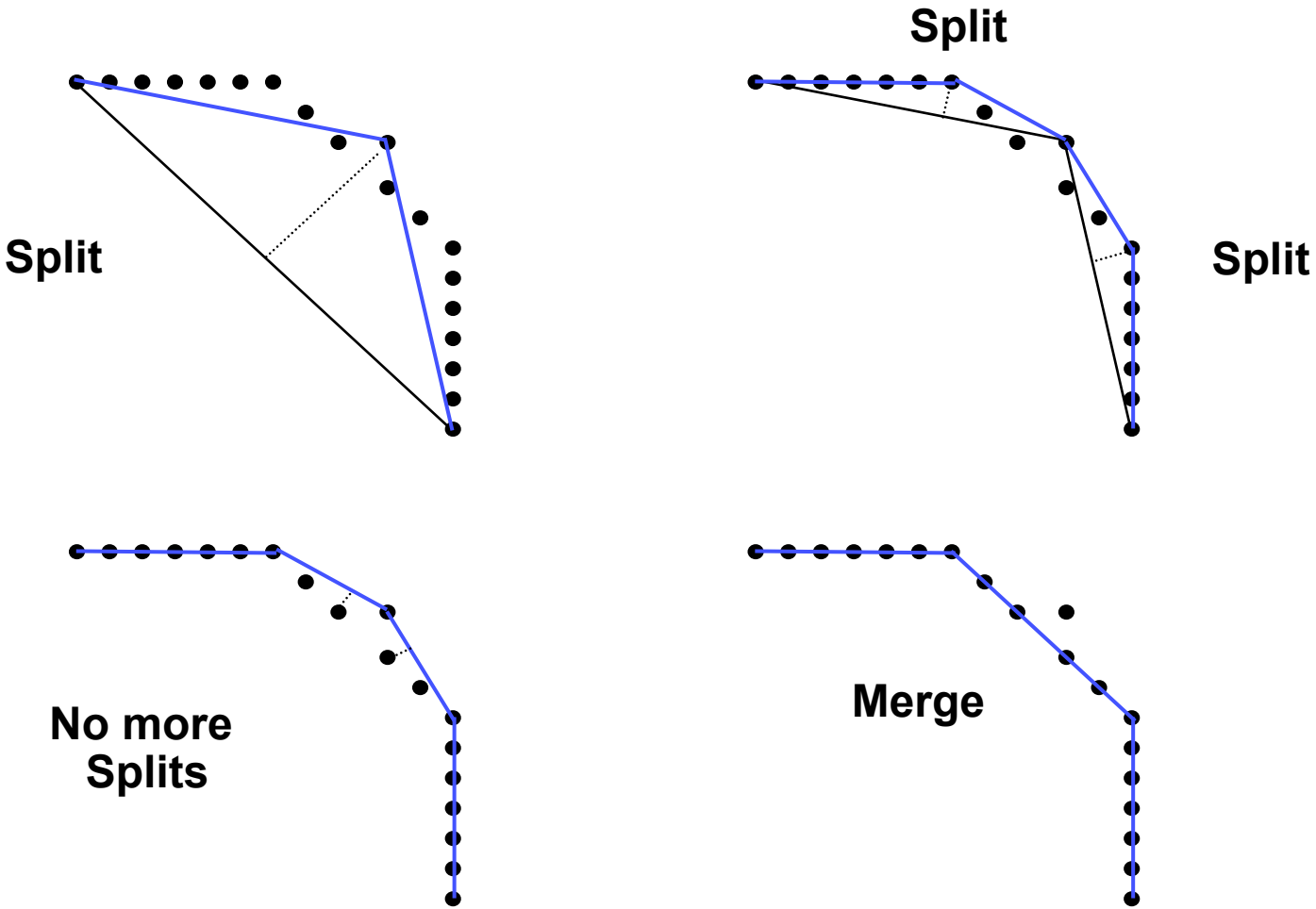
1. **Recursive Split:**
    1. Obtain the line passing by the two extreme points
    2. Obtain the point more distant to the line
    3. If distance > **error_max**, split and repeat with the left and right sub-scan

2. **Merge:**
    1. If two consecutive segments are close enough, obtain the common line and the more distant point
    2. If distance <= **error_max**, merge both segments

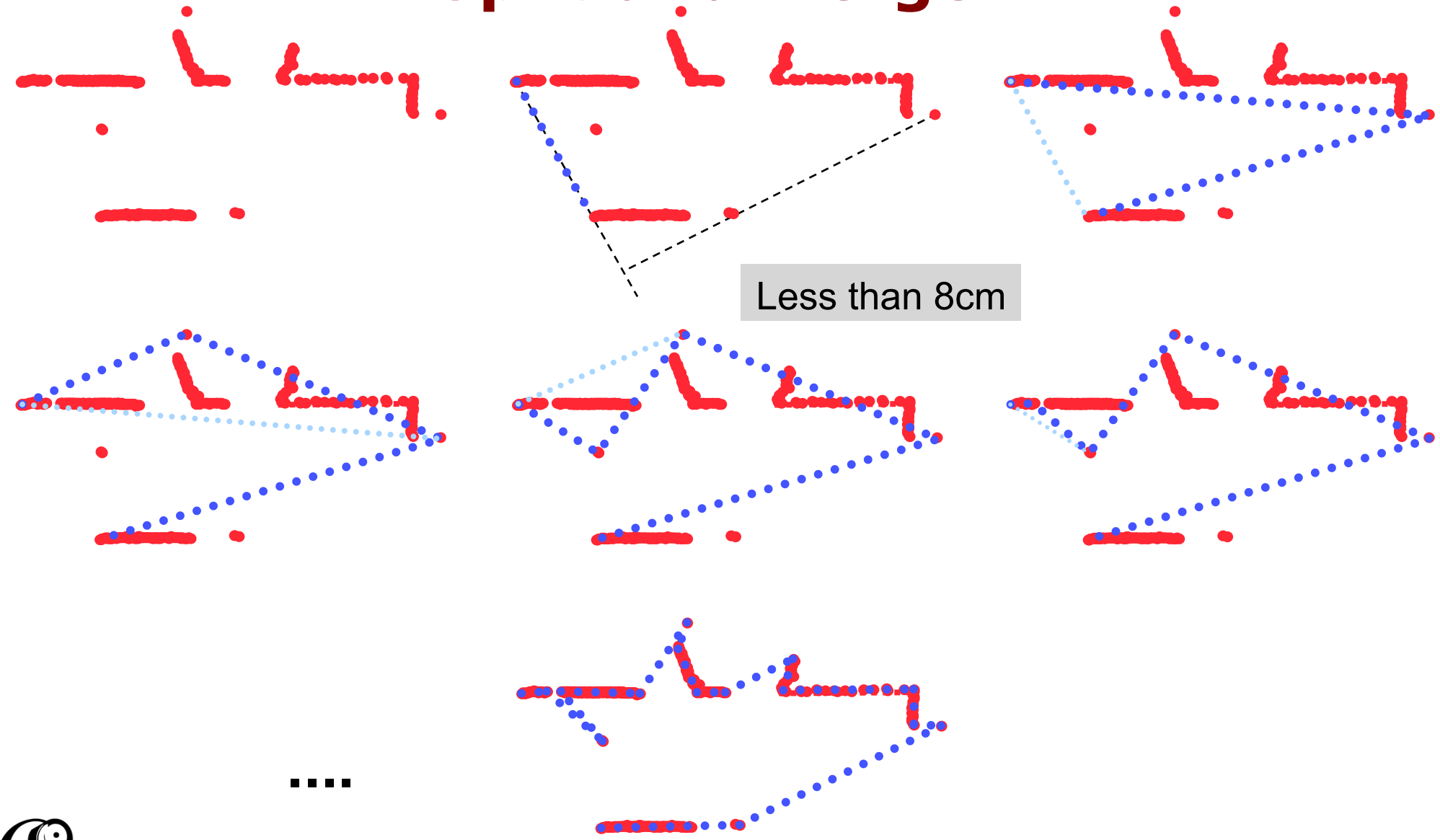
3. **Prune short** segments

4. **Estimate line equation**

Split and Merge

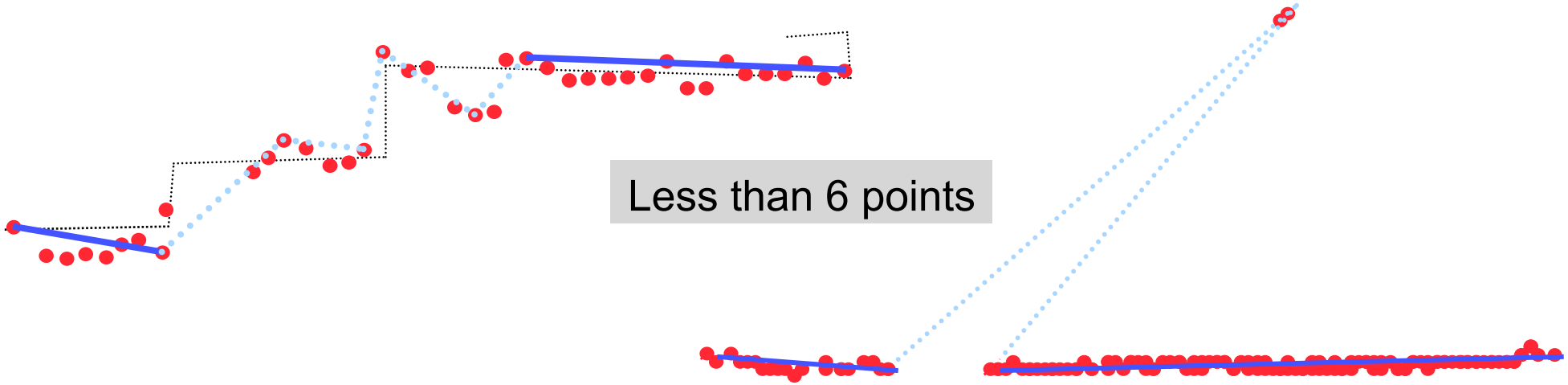


Split and Merge

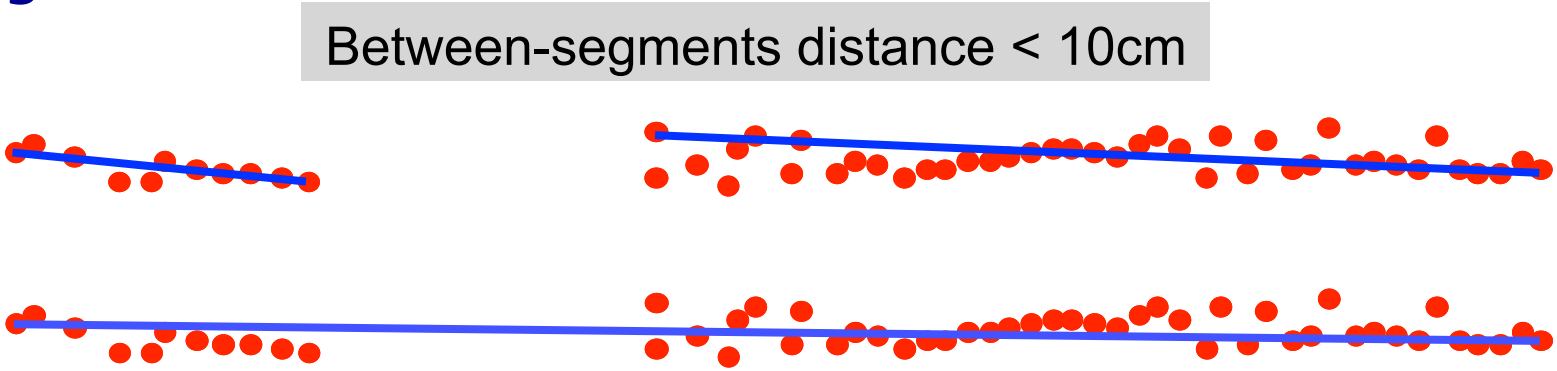


Split and Merge

- Elimination of small segments:

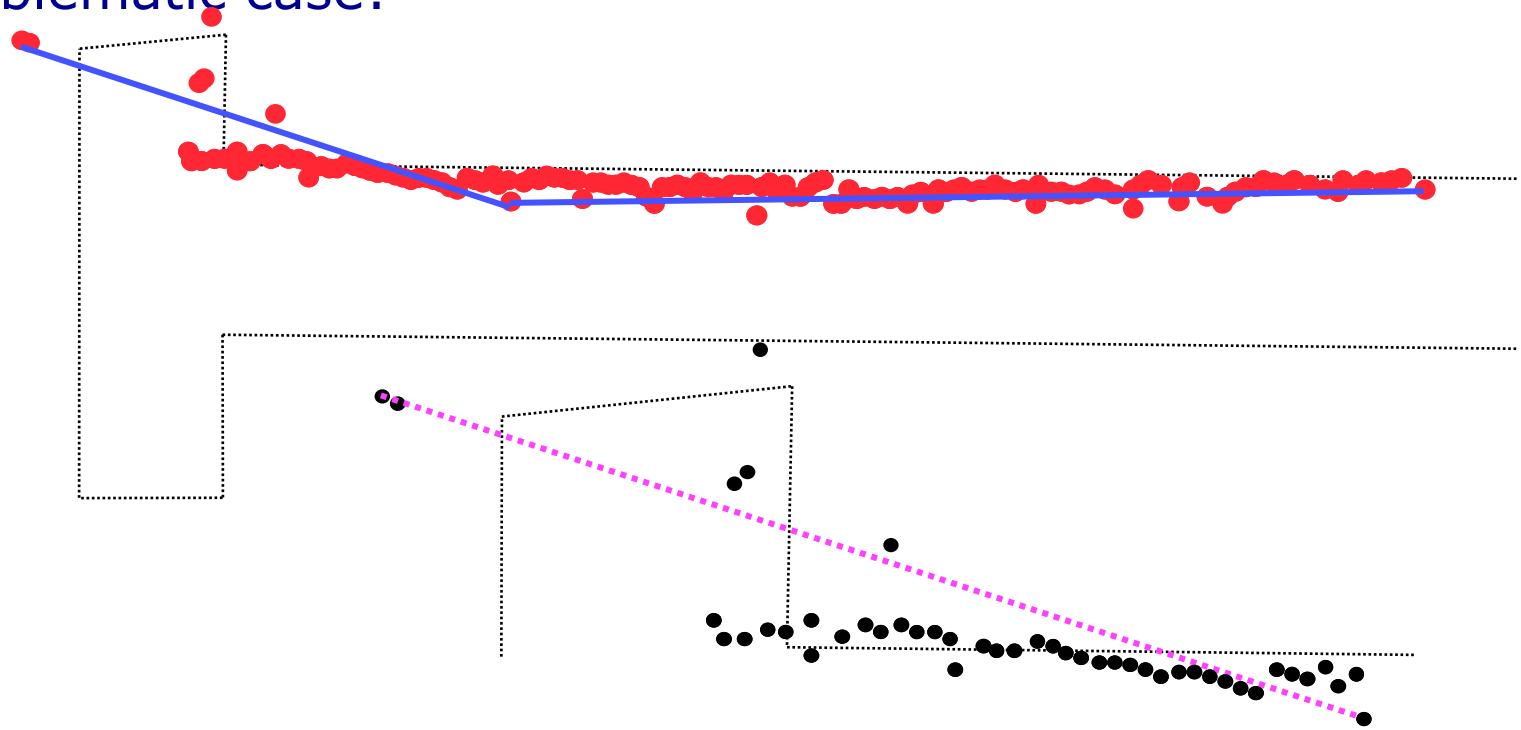


- Segment fusion:



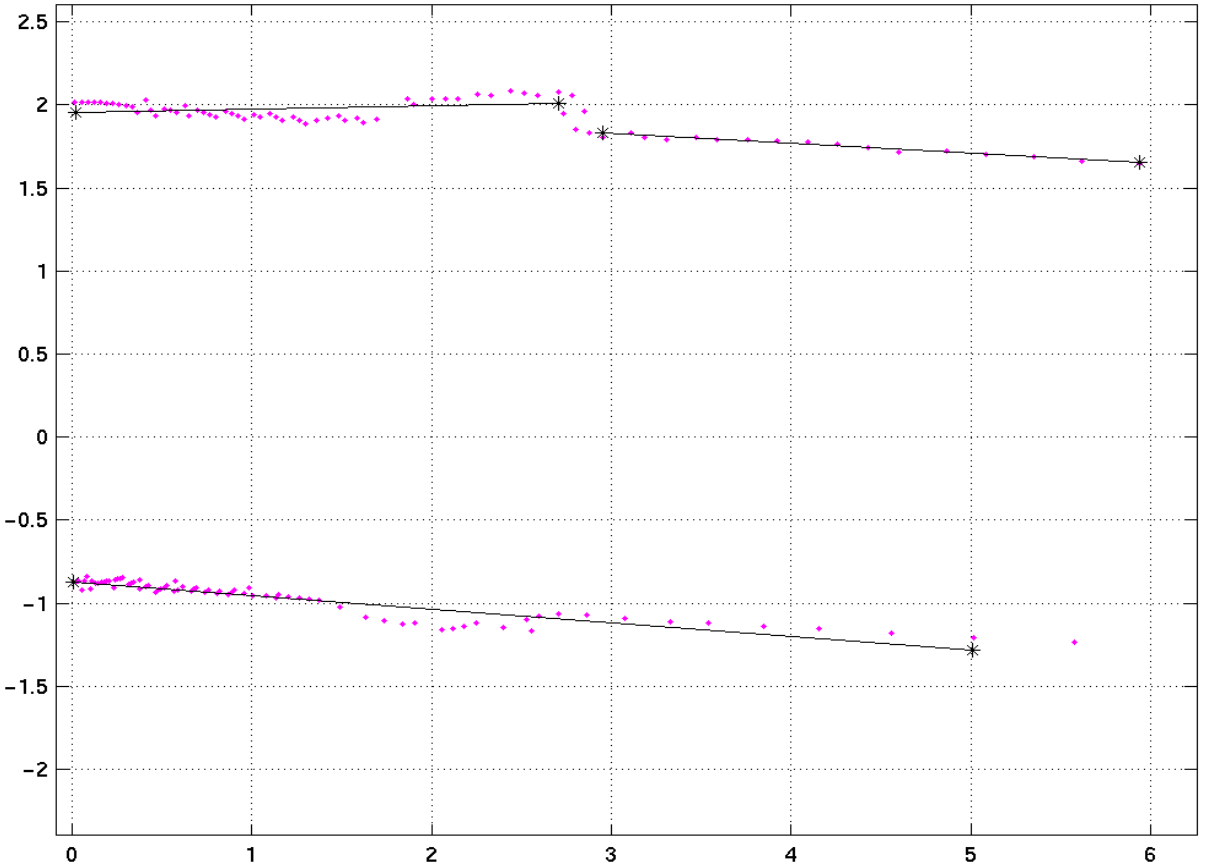
Split and Merge

- Problematic case:



- **Split and merge:** only the extreme points are used
- **Alternatives:**
  - Linear regression
  - RANSAC
  - Hough transform

Split and Merge



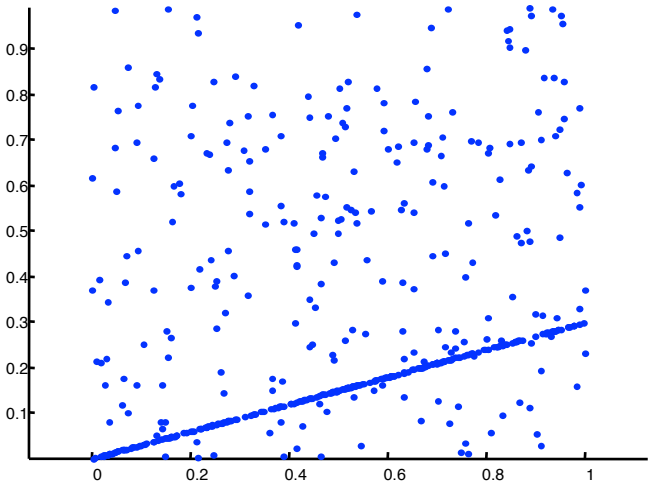
Not robust to complex and/or spurious data

ROBOTICS
UNIVERSIDAD DE ZARAGOZA

RANSAC

- Given a model that requires n data points to compute a solution and a set of data points P, with #(P) > n :
  - Randomly select a subset S1 of n data points and compute the model M1
  - Determine the **consensus** set S1* of points is P compatible with M1 (within some error tolerance)
  - If #(S1*) > th, use S1* to compute (maybe using least squares) a new model M1*
  - If #(S1*) < th, randomly select another subset S2 and repeat
  - If, after t trials there is no consensus set with th points, return with failure

RANSAC



z acceptable probability of failure

t tries ¿?

$$(1 - w^n)^t = z$$

$$t = \left\lceil \frac{\log z}{\log (1 - w^n)} \right\rceil$$

p no. of points

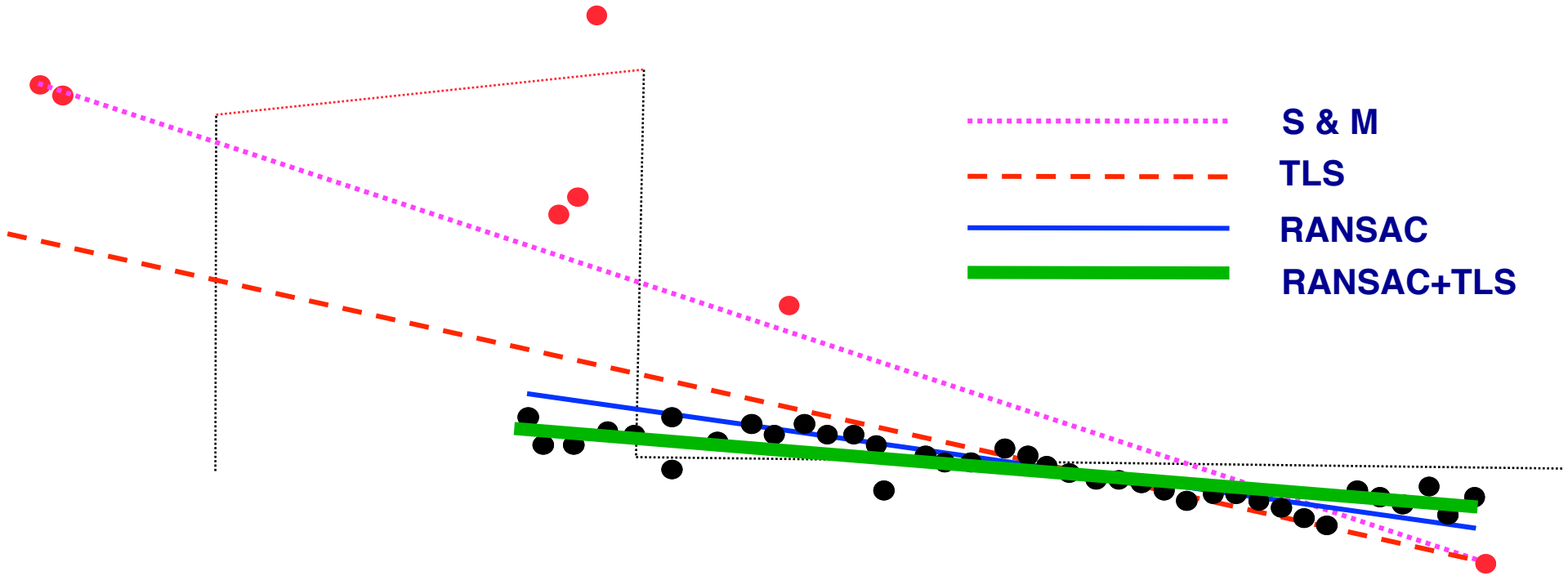
n points to build model

w probability that a point is good

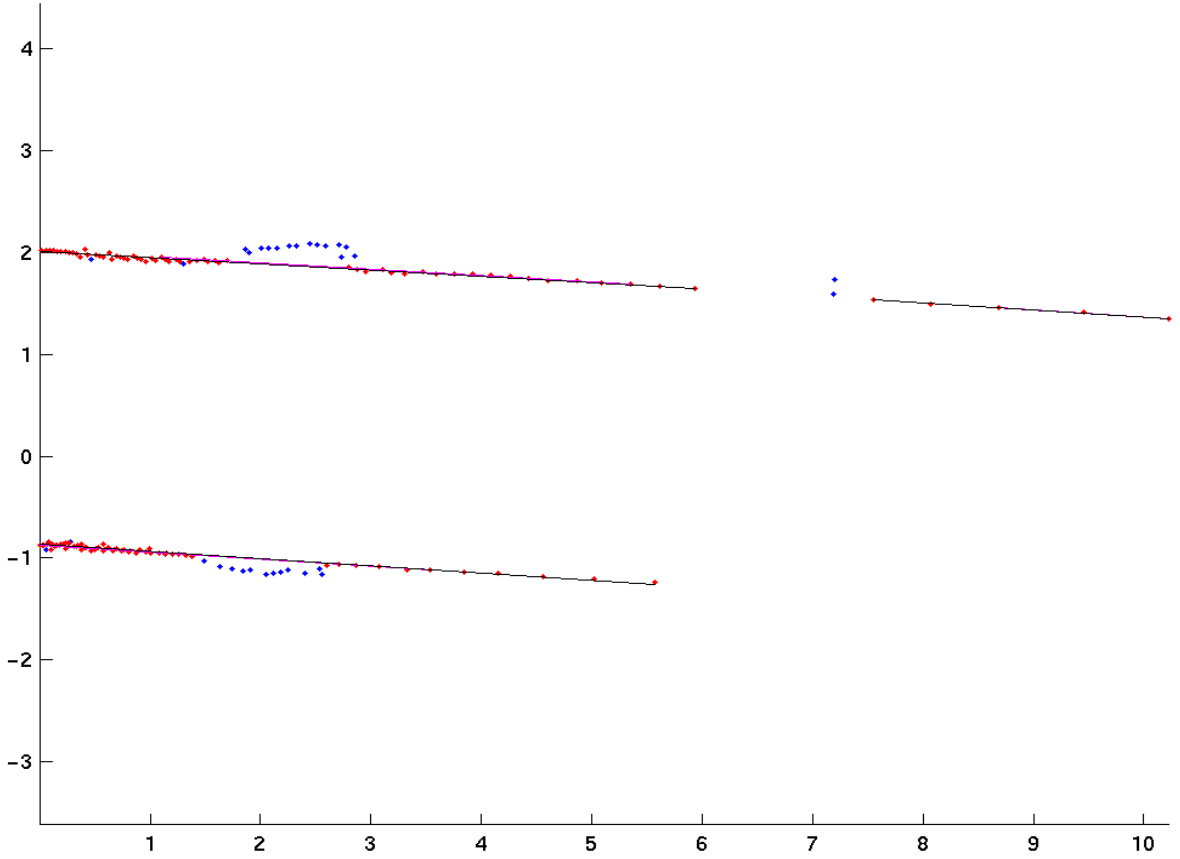
$O(p^n)$_possible_models

w	0,01
n	2
z	0,05
t	29956

RANSAC

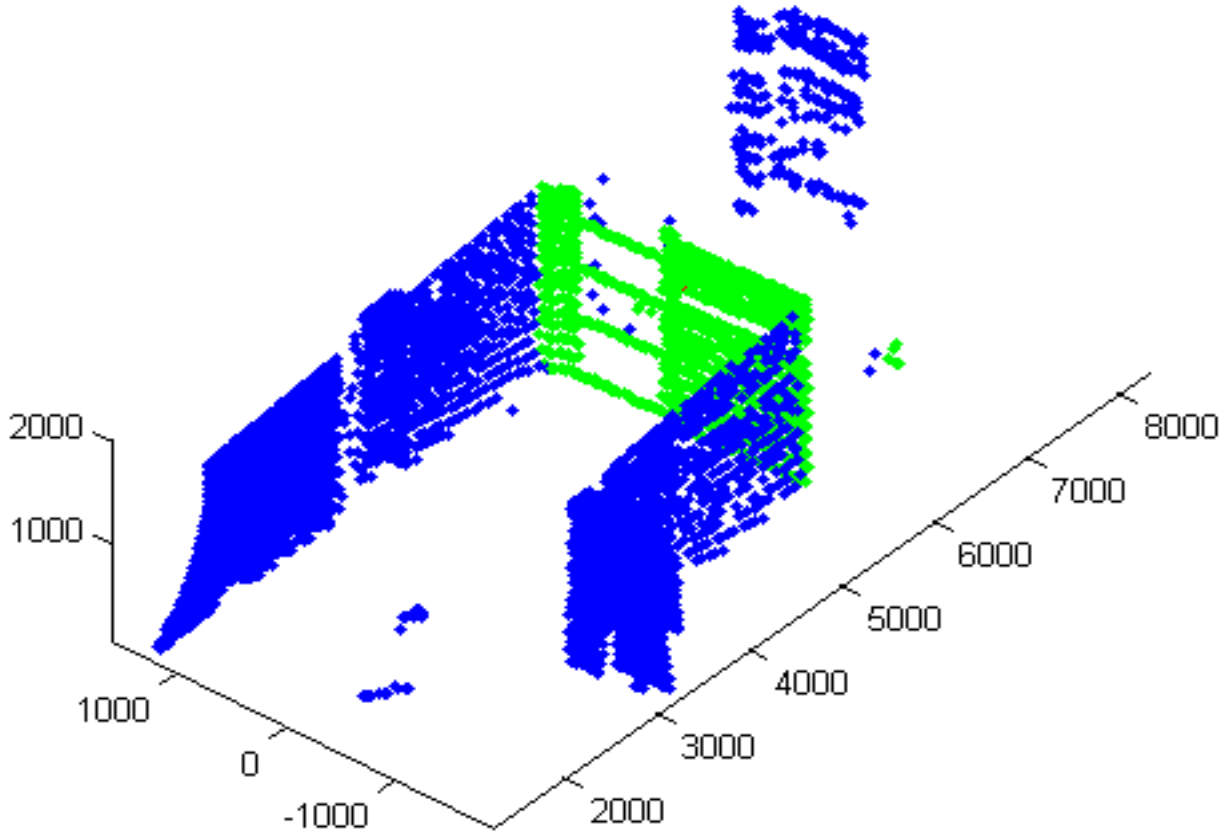


RANSAC



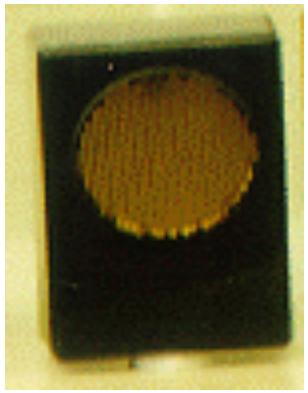
Robust statistics deal with spuriousness

RANSAC for 3D planes

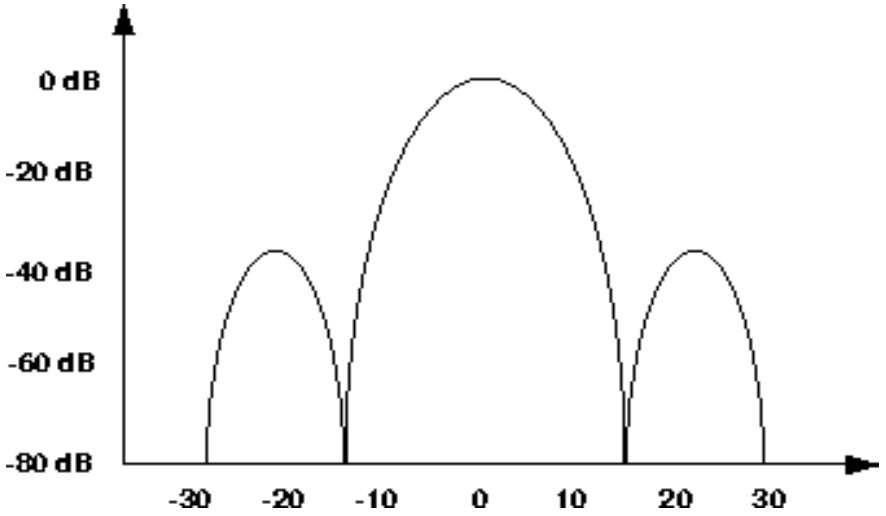


P.M. Newman, J.J. Leonard, J. Neira and J.D. Tardós: **Explore and Return: Experimental Validation of Real Time Concurrent Mapping and Localization**. IEEE Int. Conf. Robotics and Automation, May, 2002

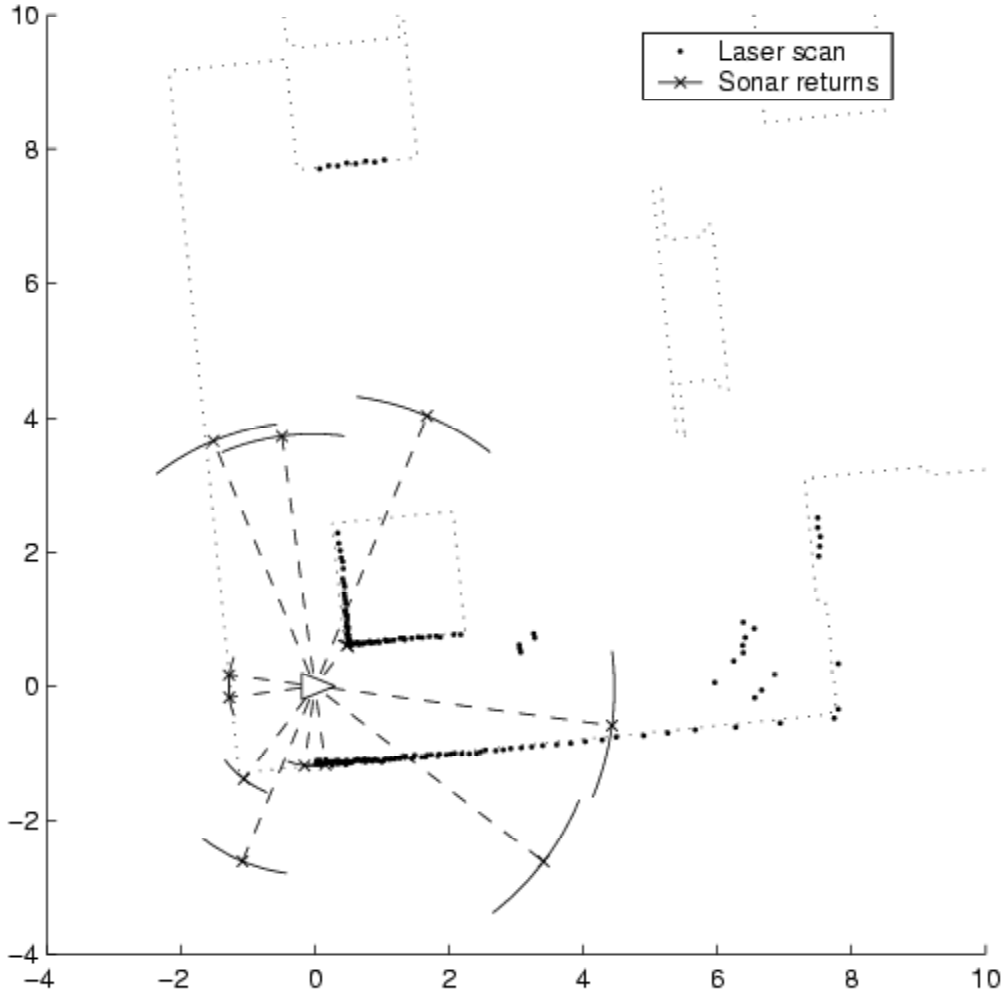
External sensors: Sonar



Polaroid US

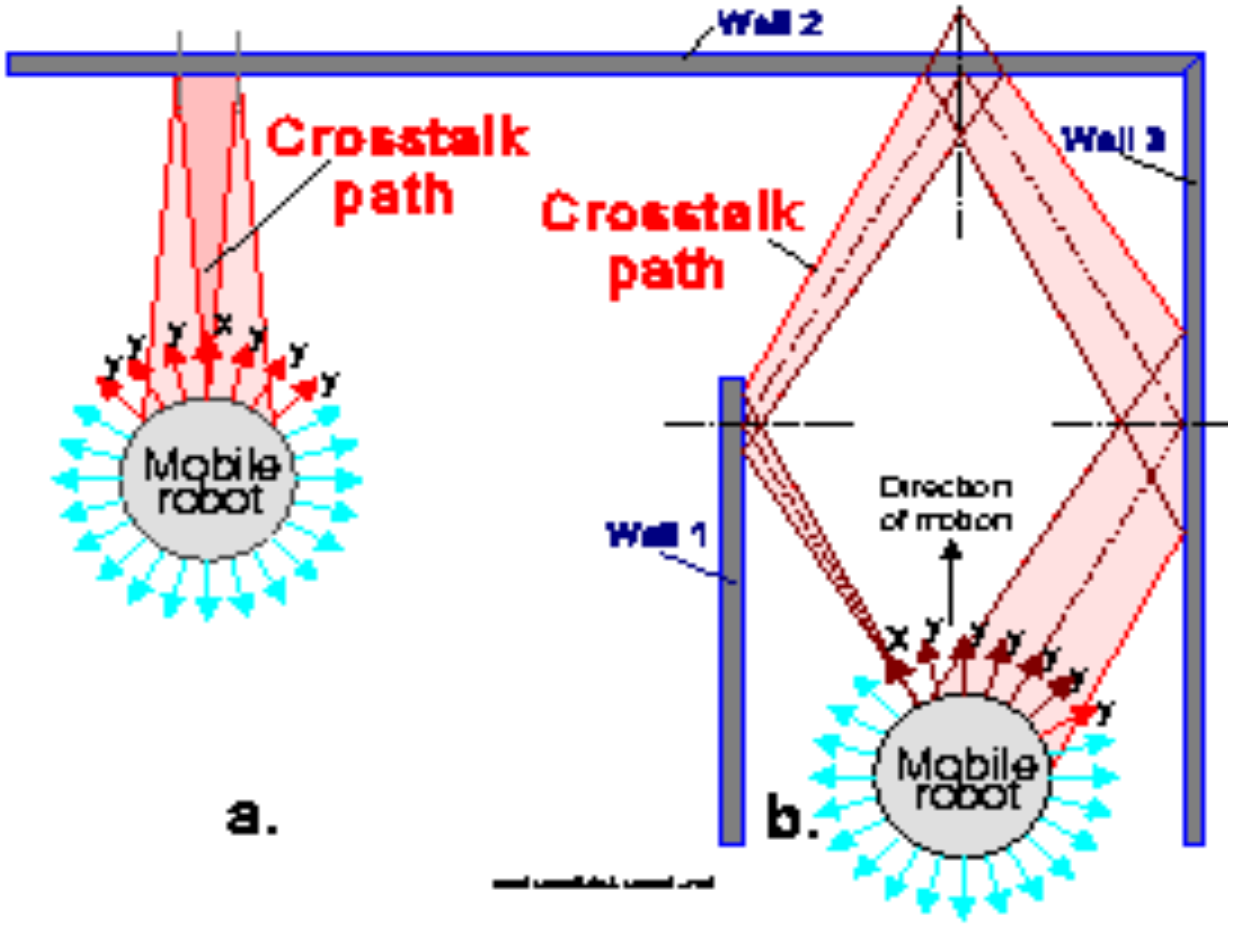


Frequency lobes

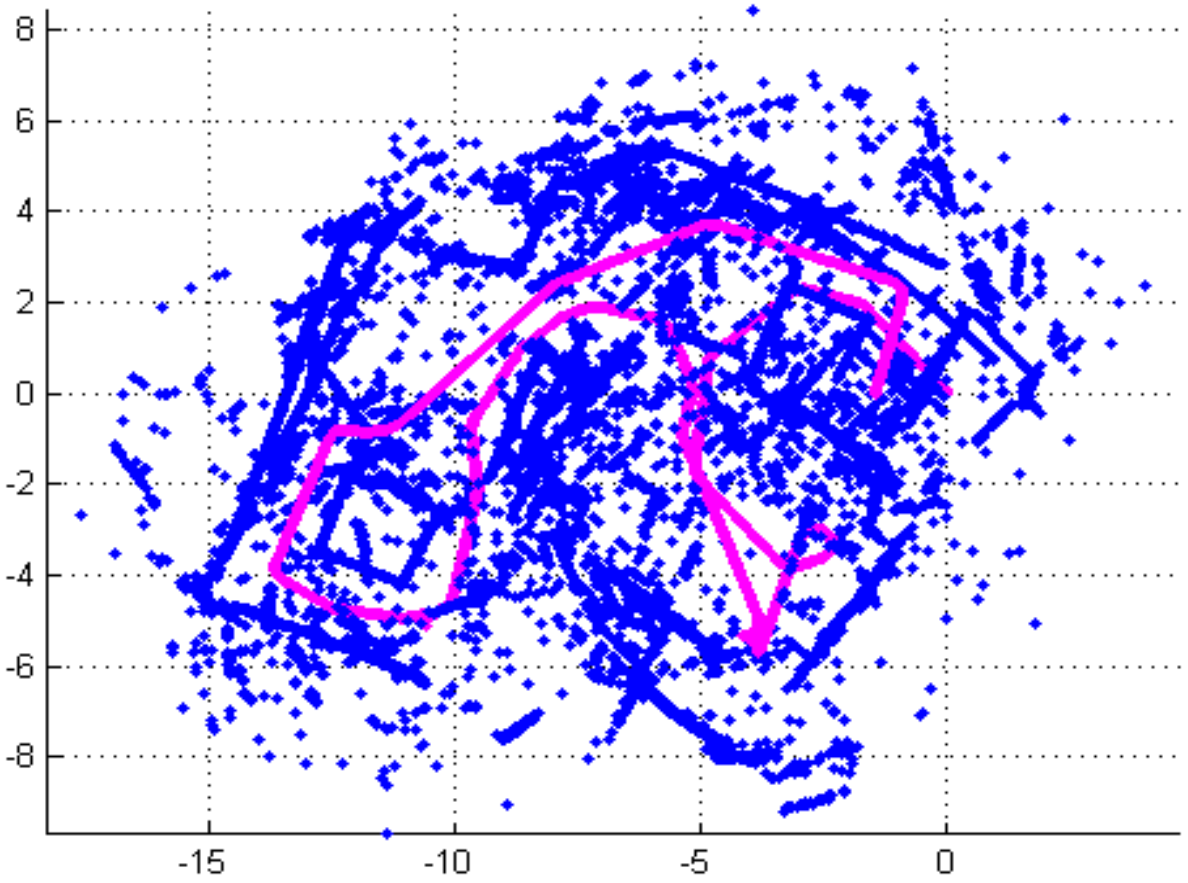


Very sparse and noisy data

Sonar

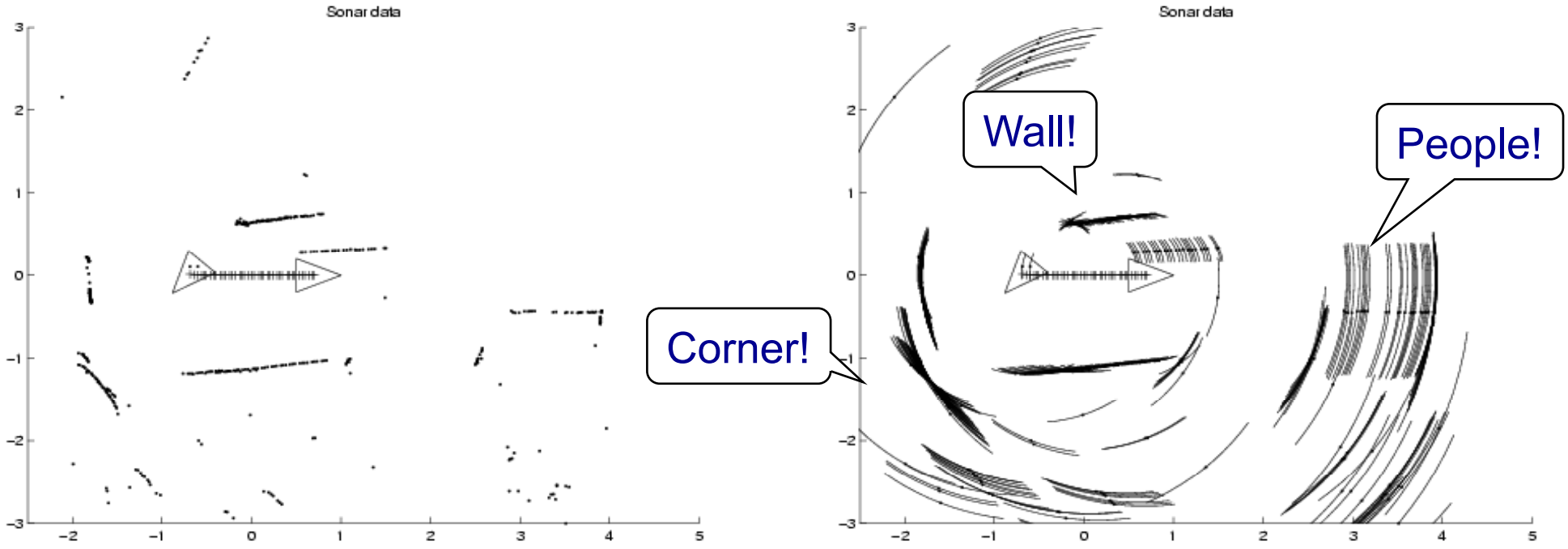

Wall 2
Crosstalk path
Crosstalk path
Wall 3
Mobile robot
Direction of motion
Wall 1
a.
b.
Mobile robot

Sonar



Very sparse and noisy data

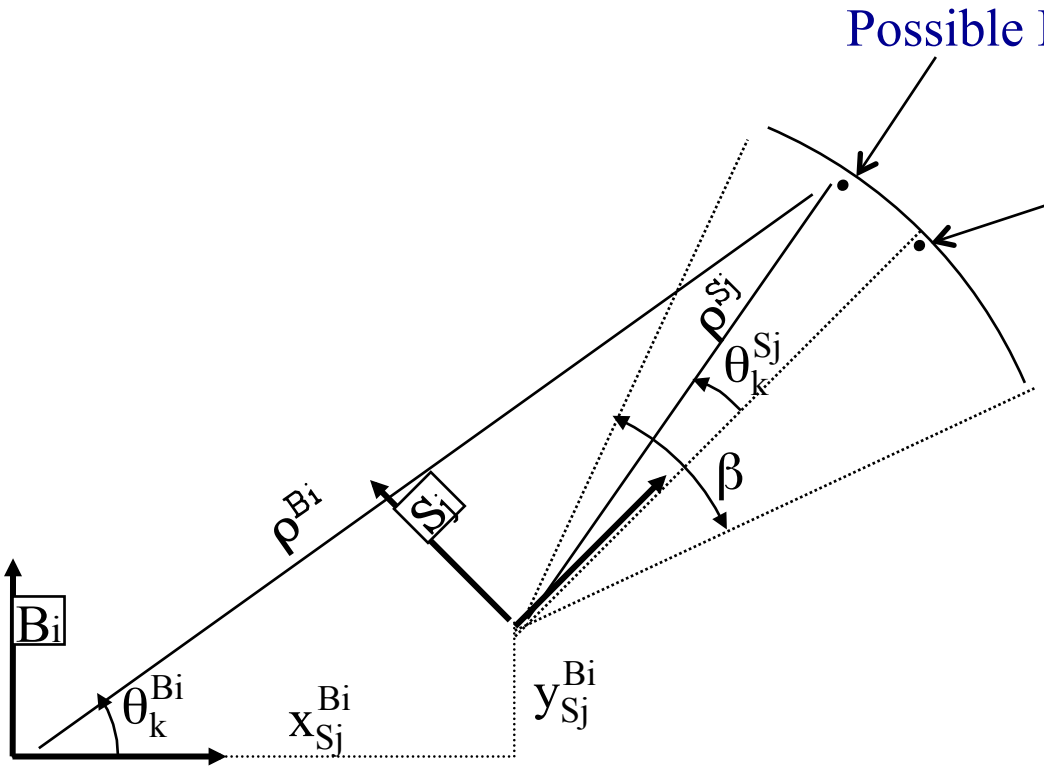
Move and build a local map



Exploit redundancy

Use a good sensor model

Sonar Model for Points

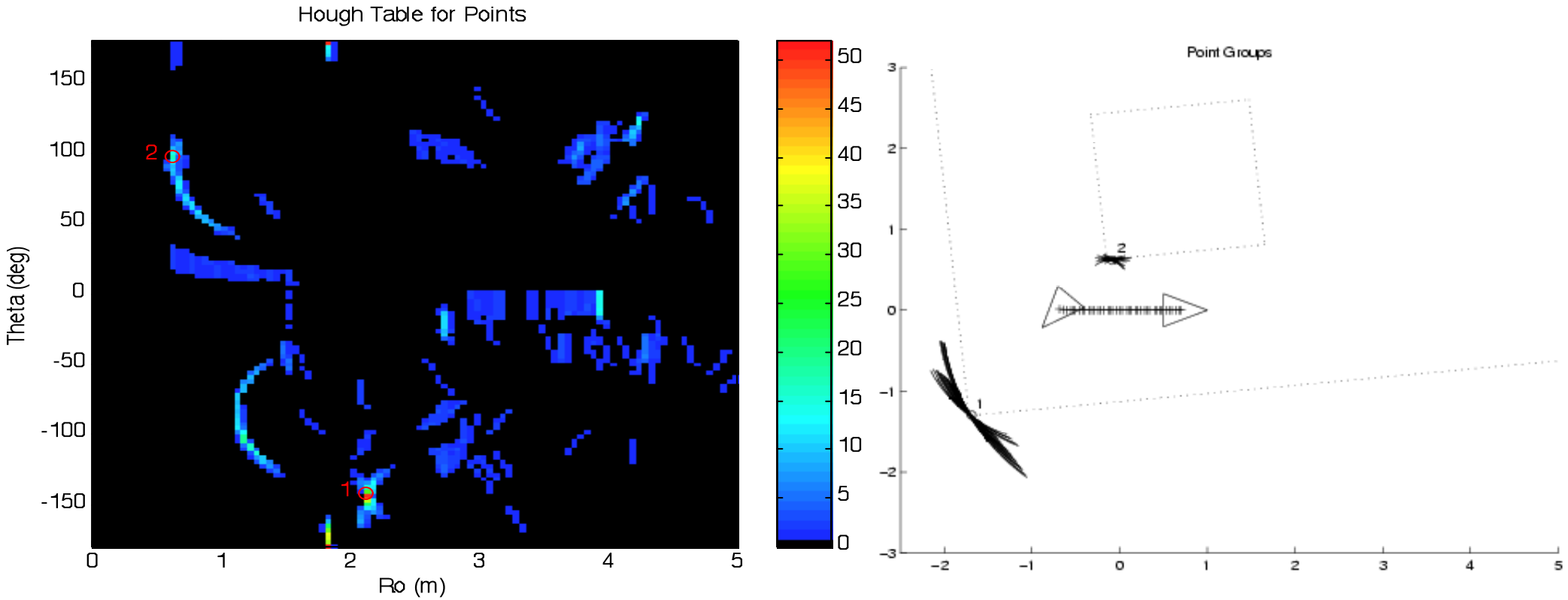


Hough Voting

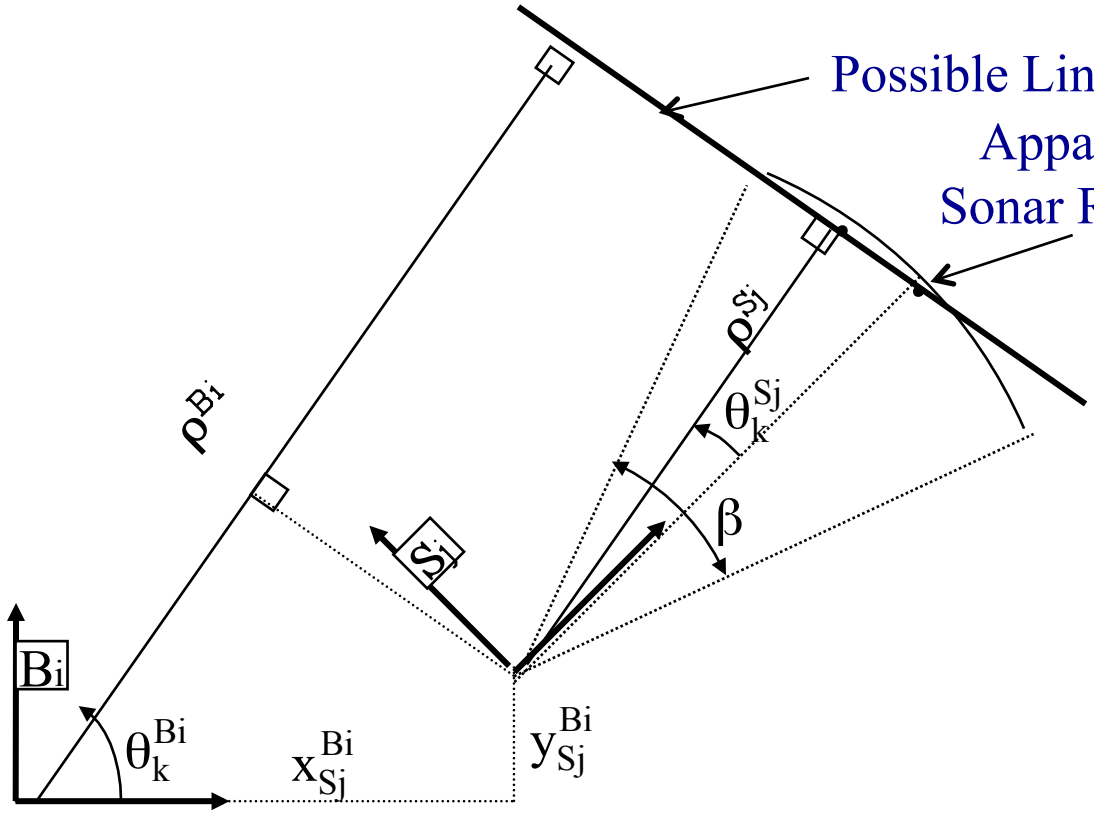
```
for i in 1..n_positions
    for j in 1..n_sensors
      Compute x^Bi_Sj
      for θ^Sj_k in -β/2..β/2 step δ
        Compute θ^Bj_k ρ^Bj_k
        Vote(θ^Bj_k, ρ^Bj_k)
      end
    end
end
```

Hough Transform: Corners

- Sonar returns **vote** for points
- Look for local maxima



Sonar Model for Lines

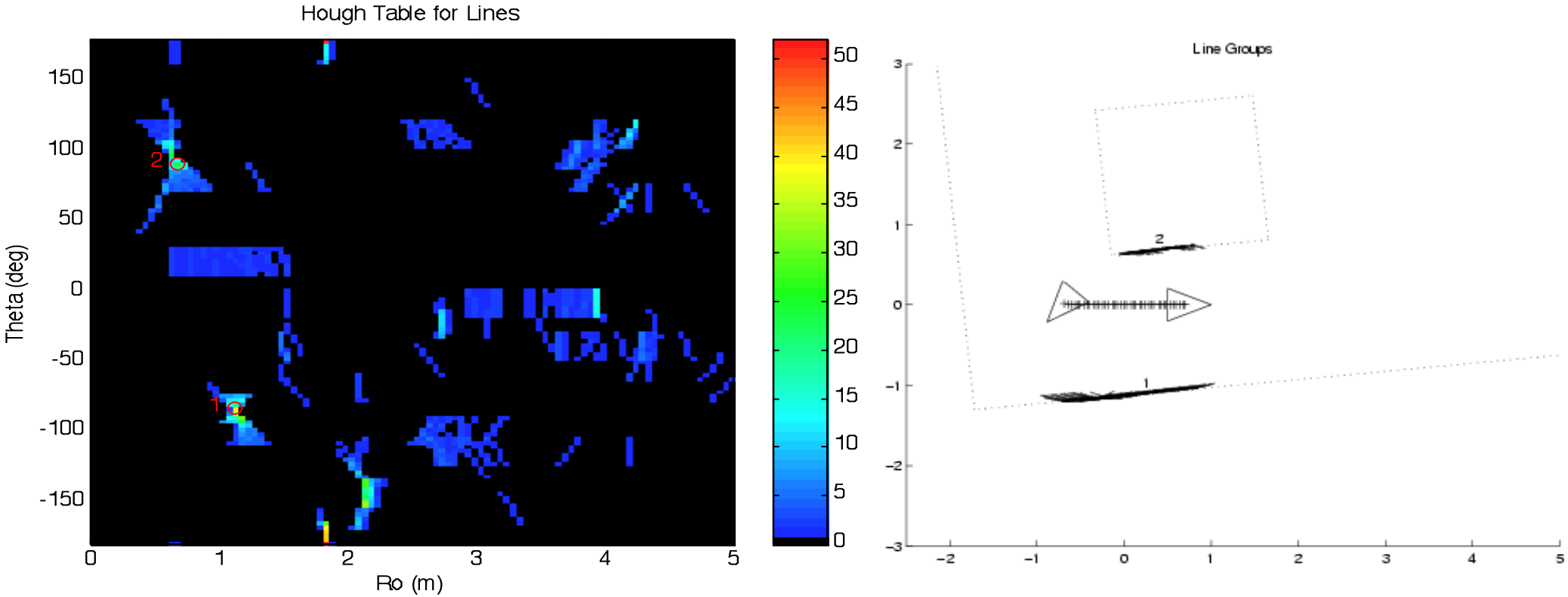


Hough Voting

```
for i in 1..n_positions
    for j in 1..n_sensors
      Compute x^Bi_Sj
      for θ^Sj_k in -β/2..β/2 step δ
        Compute θ^Bj_k ρ^Bj_k
        Vote(θ^Bj_k, ρ^Bj_k)
      end
    end
end
```

Hough Transform: Lines

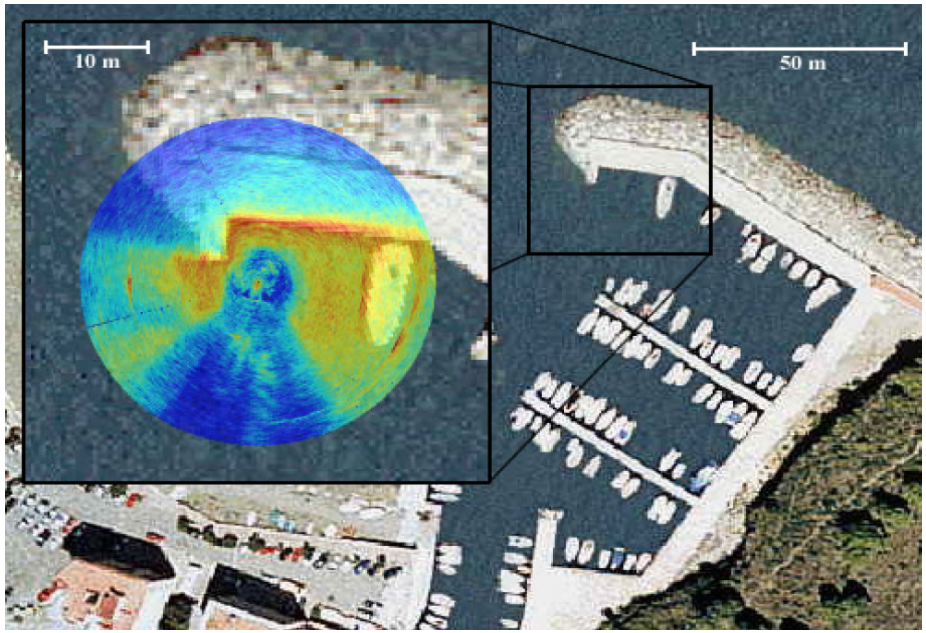
- Sonar returns **vote** for lines
- Look for local maxima

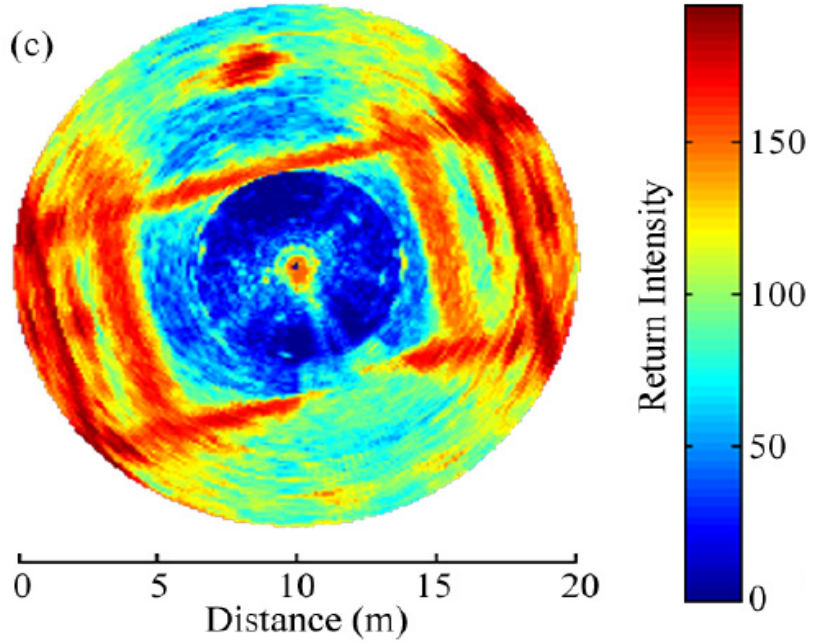


The Hough gives robust **local** data associations

Hough Transform

- Imaging sonar





Hough Transform

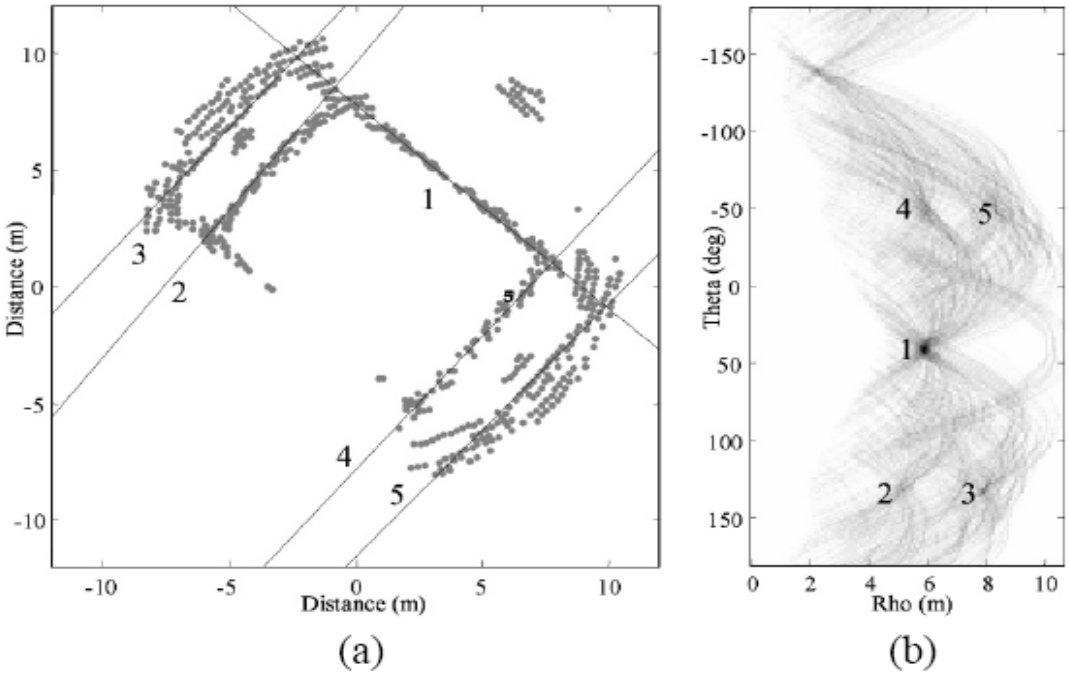
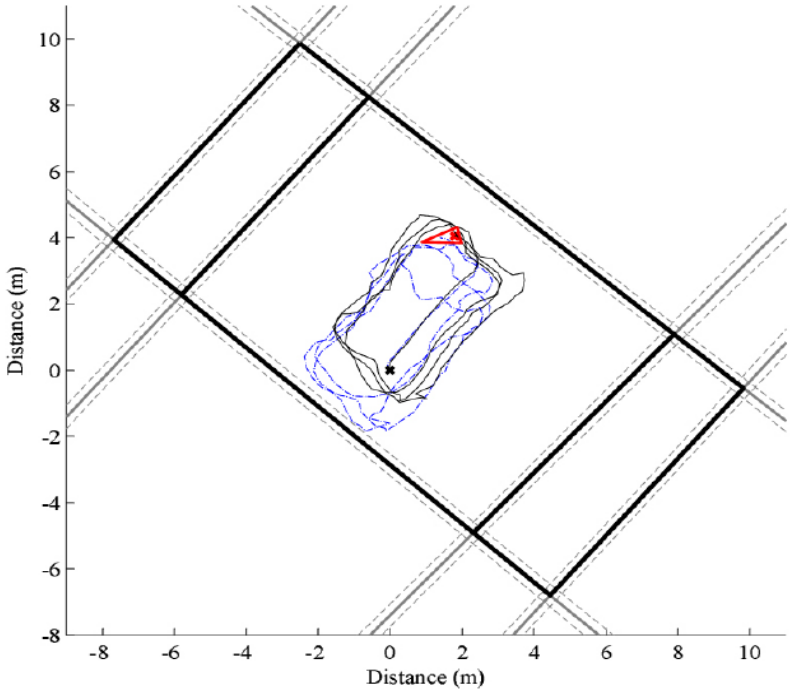


Fig. 3. Hough transform for line detection. (a) High echo-amplitude returns and the winning lines. (b)The obtained Hough voting space

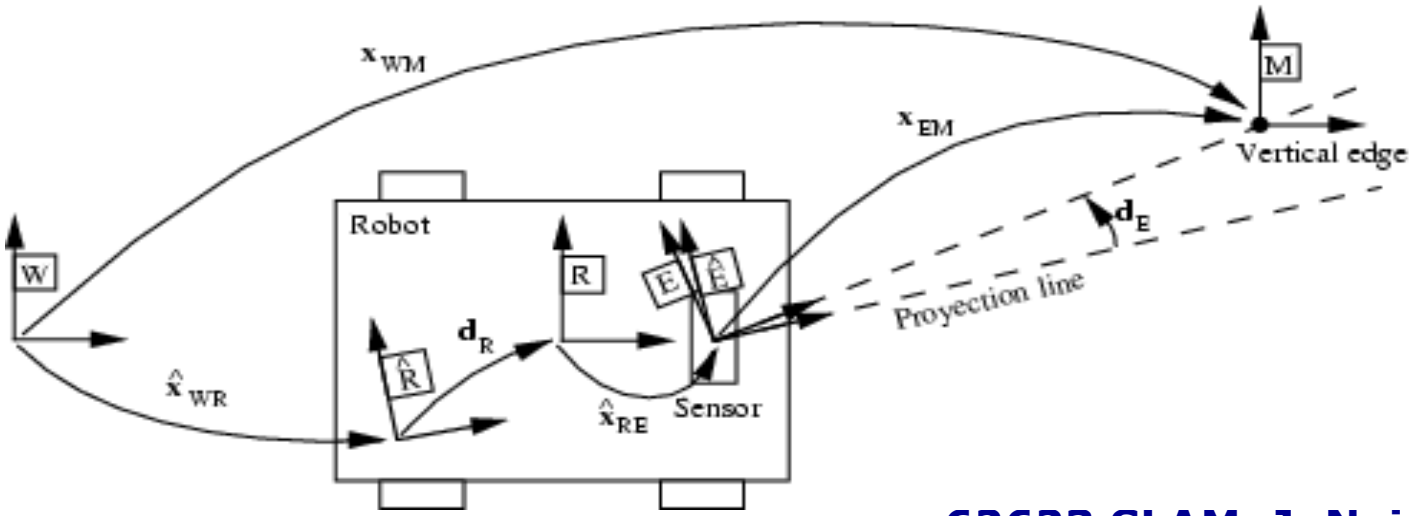


D. Ribas, P. Ridao, J. Neira, J.D. Tardós, **SLAM using an Imaging Sonar for Partially Structured Underwater Environments**, The 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems (to appear).

ROBOTICS
UNIVERSIDAD DE ZARAGOZA

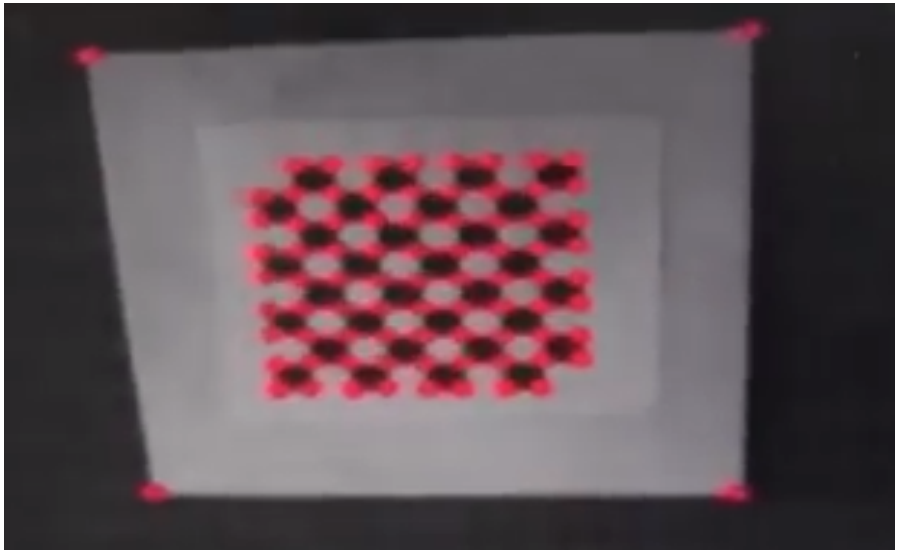
External Sensors: Monocular Vision

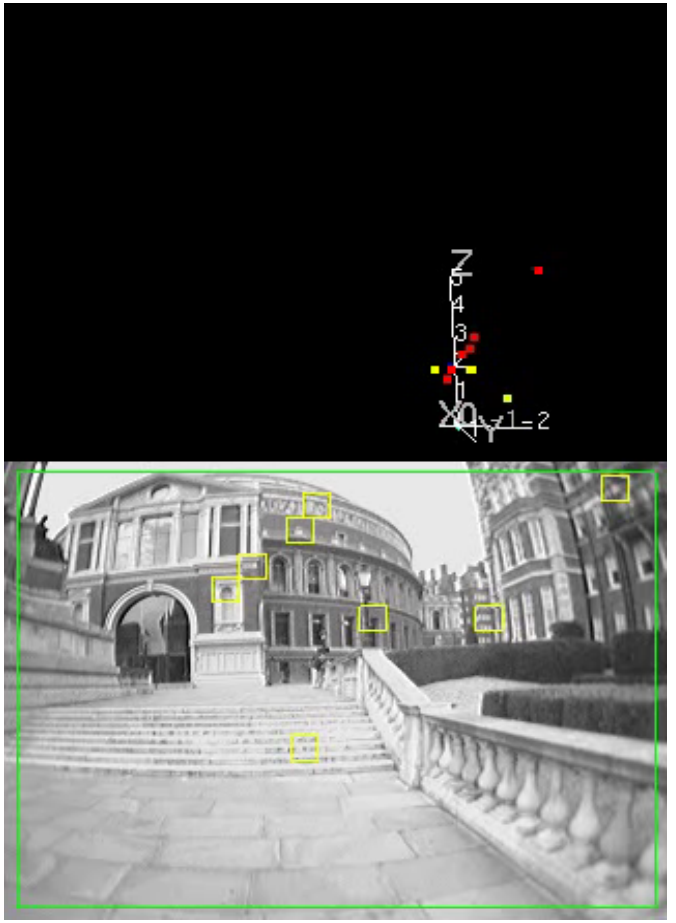
- Vertical edges (doors, corners, ...)
  - Hough transformt,
  - Canny….



External Sensors: Monocular Vision

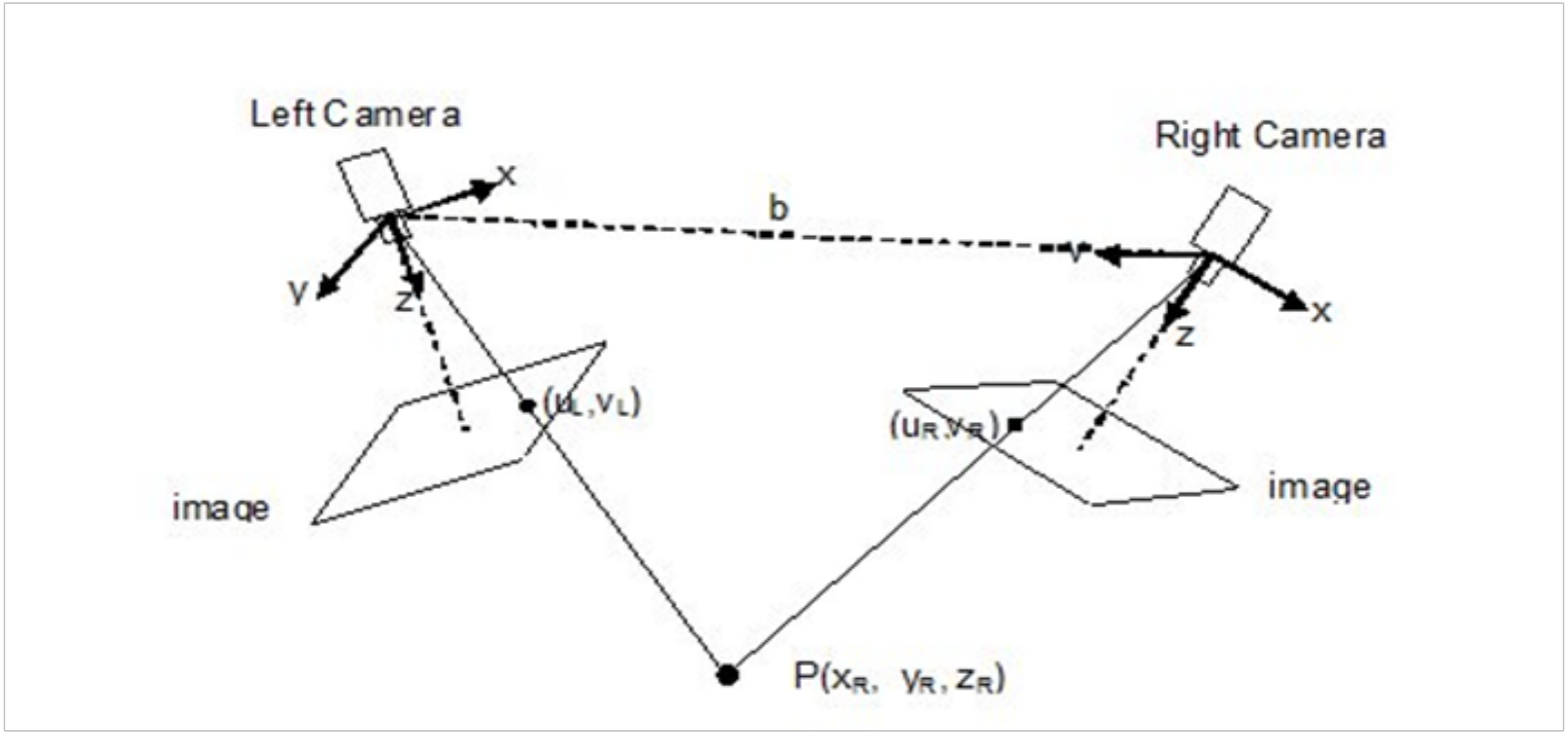
- Points of interest:
  - Harris corners
  - Shi-Tomasi
  - SIFT, SURF, FAST, …





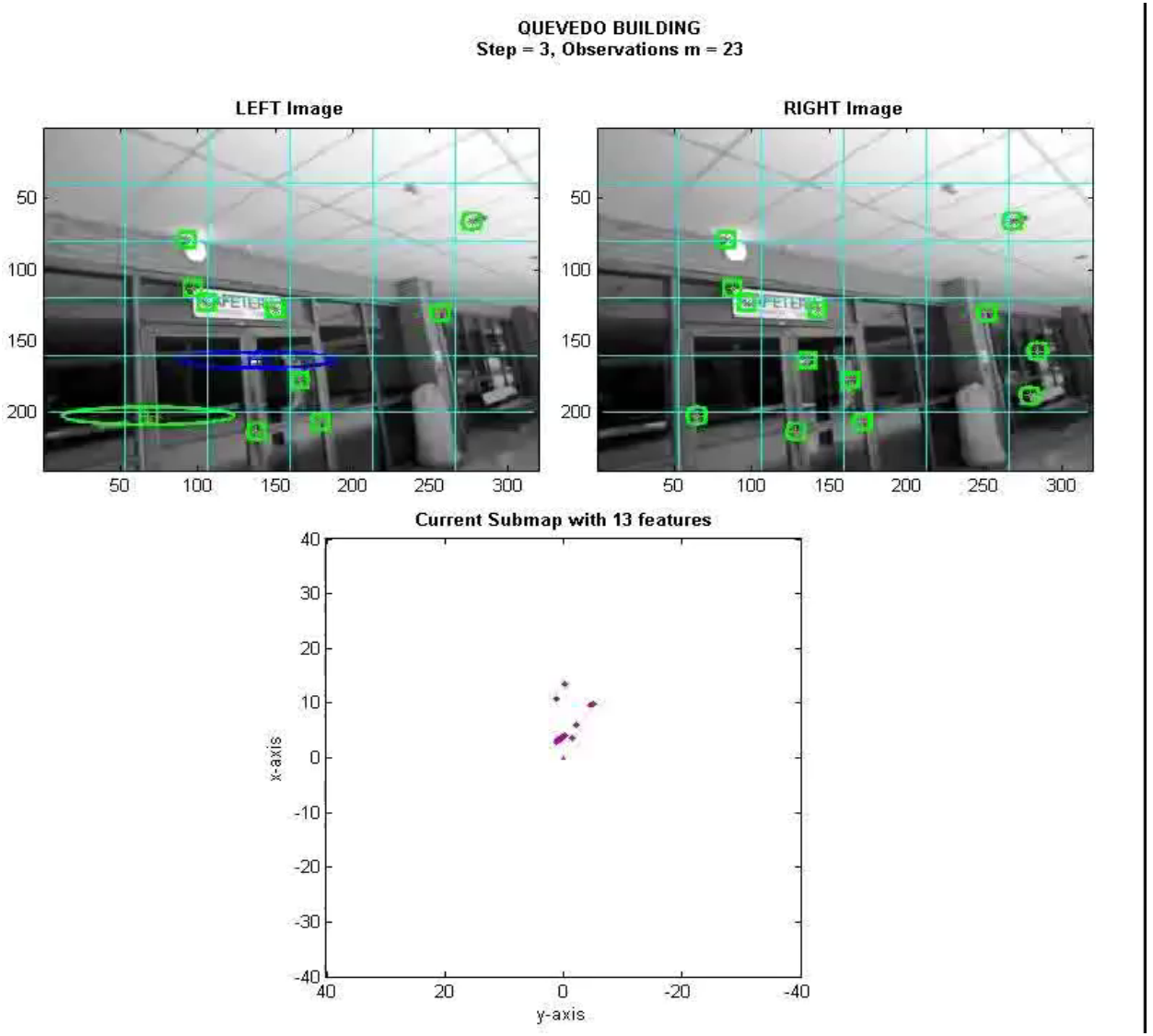
- The scale is unobservable

External Sensors: Stereo Vision

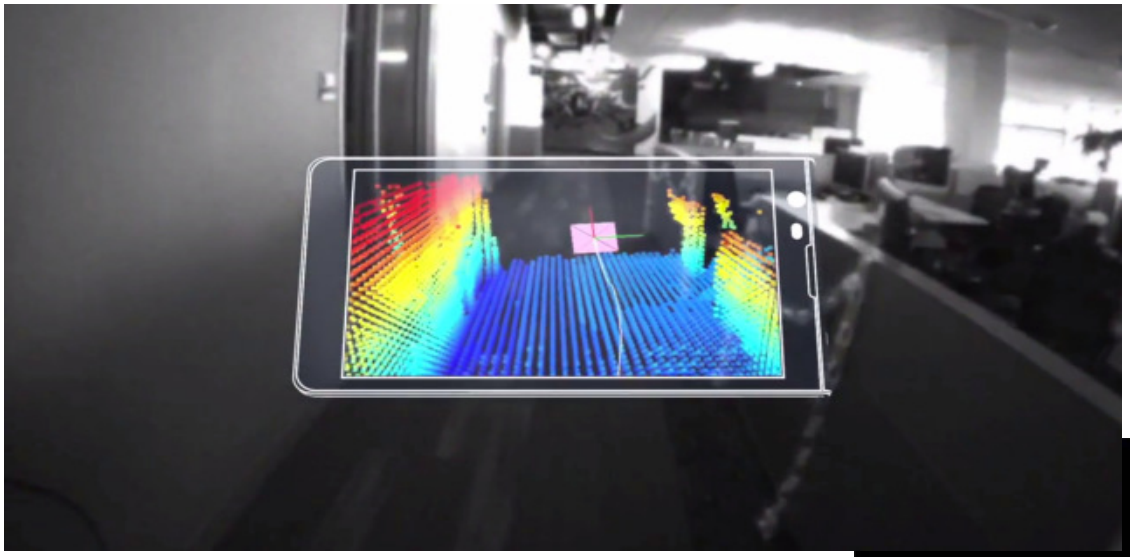


- Given the baseline b, the scale is observable

External Sensors: Stereo Vision



External Sensors: 3D Vision



Kinect-like sensors