



## Robotics - SLAM

Simone Ceriani

*[ceriani@elet.polimi.it](mailto:ceriani@elet.polimi.it)*

Dipartimento di Elettronica e Informazione  
Politecnico di Milano

14 June 2012

# Outline

1 Introduction

2 EKF-SLAM

3 Landmark Addition

4 Measurement & Update

5 SLAM example

6 Correspondences

7 Visual SLAM

8 Conclusion

# Outline

## 1 Introduction

## 2 EKF-SLAM

## 3 Landmark Addition

## 4 Measurement & Update

## 5 SLAM example

## 6 Correspondences

## 7 Visual SLAM

## 8 Conclusion









## LOCALIZATION - A SIMULATED EXAMPLE

- The position of the coloured box is known (i.e. the *map* is known)
- The robot sense and distinguish the *map* elements

6/73

## MAPPING - A SIMULATED EXAMPLE

Video mapping.flv

- The map is initially unknown (gray window)
- The map is incrementally builded by measurements

Video from <http://www.youtube.com/watch?v=ZfqLnZSAhZw>

## KNOWLEDGE OF THE POSITION

- ❖ Impossible in a lot of applications  
e.g.: exploration of buildings,  
underwater operations, ...
- ❖ Thus, localization is not applicable

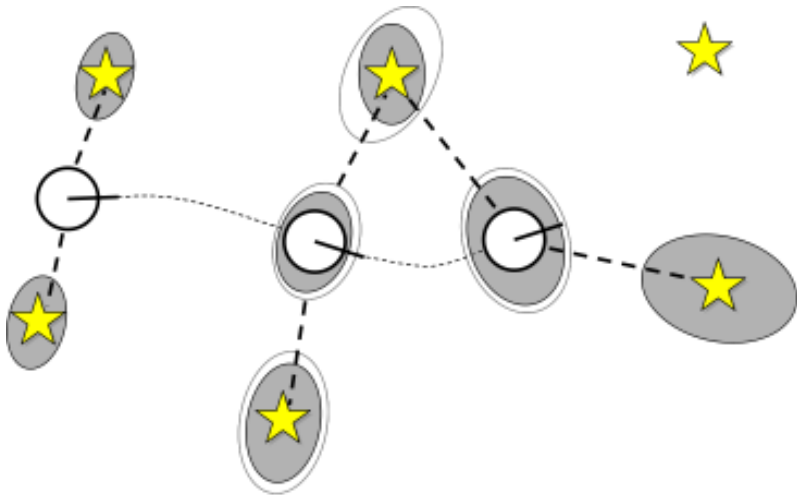
- Usually unavailable or noisy/uncertain  
e.g.: lack of GPS signal in indoor, ...
- Thus, mapping is not applicable

## SLAM - Simultaneous Localization And Mapping

- The *holy grail* of the robotics, but not solely mobile robotics!
- **Mapping requires localization  $\Leftrightarrow$  localization requires map**
- Answer to this question:

*"Is it possible for a mobile robot to be placed in an **unknown** location in an **unknown** environment and for the robot to incrementally build a consistent map of this environment while simultaneously determining its location within this map?"*

## SLAM

ROBOT PATH AND MAP ARE BOTH UNKNOWN

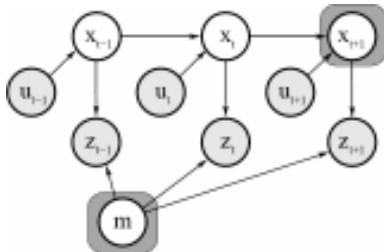
Robot path error correlates errors in the map

# On-line and Full SLAM

## ON-LINE SLAM

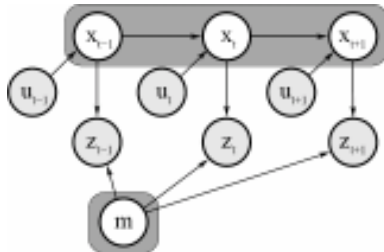
$$p(x_t, m | z_{1:t}, u_{1:t})$$

- $x_t$ : pose at time  $t$
- $m$ : map (static)
- $z_{1:t}$ : measurements
- $u_{1:t}$ : controls
- *On-line*: estimate the current pose
- Most online-SLAM are incremental, they discard  $z_{1:t-1}, u_{1:t-1}$



## FULL SLAM $p(x_{1:t}, m | z_{1:t}, u_{1:t})$

- $x_{1:t}$ : entire path or trajectory
- $m$ : map (static)
- $z_{1:t}$ : measurements
- $u_{1:t}$ : controls









## STATE

- $\mathbf{x}_t = [x, y, \theta]^T$  robot complete pose in world reference frame  $\mathbf{T}_{WR}^{(W)}$
- $m = [\mathbf{p}_{x_1}^{(W)}, \mathbf{p}_{y_1}^{(W)}, \mathbf{p}_{x_2}^{(W)}, \mathbf{p}_{y_2}^{(W)}, \dots, \mathbf{p}_{x_n}^{(W)}, \mathbf{p}_{y_n}^{(W)}]^T$  cartesian coordinates of points in world reference frame

$$\mathbf{X} = \begin{bmatrix} \mathbf{x} \\ \mathbf{p}_1^{(W)} \\ \mathbf{p}_2^{(W)} \\ \mathbf{p}_3^{(W)} \\ \vdots \\ \mathbf{p}_{x_n}^{(W)} \end{bmatrix} \quad \Sigma = \begin{bmatrix} \Sigma_{xx} & \Sigma_{\mathbf{x}\mathbf{p}_1}^{(W)} & \Sigma_{\mathbf{x}\mathbf{p}_2}^{(W)} & \Sigma_{\mathbf{x}\mathbf{p}_3}^{(W)} & \cdots & \Sigma_{\mathbf{x}\mathbf{p}_n}^{(W)} \\ \Sigma_{\mathbf{p}_1}^T & \Sigma_{\mathbf{p}_1}^{(W)} \mathbf{p}_1^{(W)} & \Sigma_{\mathbf{p}_1}^{(W)} \mathbf{p}_2^{(W)} & \Sigma_{\mathbf{p}_1}^{(W)} \mathbf{p}_3^{(W)} & \cdots & \Sigma_{\mathbf{p}_1}^{(W)} \mathbf{p}_n^{(W)} \\ \Sigma_{\mathbf{p}_2}^T & \Sigma_{\mathbf{p}_1}^{(W)} \mathbf{p}_2^{(W)} & \Sigma_{\mathbf{p}_2}^{(W)} \mathbf{p}_2^{(W)} & \Sigma_{\mathbf{p}_2}^{(W)} \mathbf{p}_3^{(W)} & \cdots & \Sigma_{\mathbf{p}_2}^{(W)} \mathbf{p}_n^{(W)} \\ \Sigma_{\mathbf{p}_3}^T & \Sigma_{\mathbf{p}_1}^{(W)} \mathbf{p}_3^{(W)} & \Sigma_{\mathbf{p}_2}^{(W)} \mathbf{p}_3^{(W)} & \Sigma_{\mathbf{p}_3}^{(W)} \mathbf{p}_3^{(W)} & \cdots & \Sigma_{\mathbf{p}_3}^{(W)} \mathbf{p}_n^{(W)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \Sigma_{\mathbf{p}_n}^T & \Sigma_{\mathbf{p}_1}^{(W)} \mathbf{p}_n^{(W)} & \Sigma_{\mathbf{p}_2}^{(W)} \mathbf{p}_n^{(W)} & \Sigma_{\mathbf{p}_3}^{(W)} \mathbf{p}_n^{(W)} & \cdots & \Sigma_{\mathbf{p}_n}^{(W)} \mathbf{p}_n^{(W)} \end{bmatrix}$$

$$\Sigma = \begin{bmatrix} \Sigma_{xx} & \Sigma_{xm} \\ \Sigma_{xm}^T & \Sigma_{mm} \end{bmatrix}$$

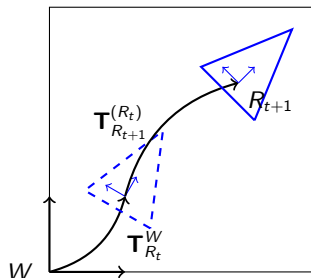
# Prediction Step - Robot motion

## STATE PREDICTION

$$[\mathbf{x}_t^T, m^T]^T = g(\mathbf{x}_{t-1}, \mathbf{u}_t, m, \epsilon)$$

$$\left\{ \begin{array}{lcl} x_{t+1} & = & \cos(\theta_t)\tilde{\Delta}x - \sin(\theta_t)\tilde{\Delta}y + x_t \\ y_{t+1} & = & \sin(\theta_t)\tilde{\Delta}x + \cos(\theta_t)\tilde{\Delta}y + y_t \\ \theta_{t+1} & = & \tilde{\theta}_t + \Delta\theta \\ \\ \mathbf{p}_{x_1}^{(W)}{}_{t+1} & = & \mathbf{p}_{x_1}^{(W)}{}_t \\ \mathbf{p}_{y_1}^{(W)}{}_{t+1} & = & \mathbf{p}_{y_1}^{(W)}{}_t \\ \dots & & \\ \mathbf{p}_{x_n}^{(W)}{}_{t+1} & = & \mathbf{p}_{x_n}^{(W)}{}_t \\ \mathbf{p}_{y_n}^{(W)}{}_{t+1} & = & \mathbf{p}_{y_n}^{(W)}{}_t \end{array} \right.$$

- Motion is the standard motion in 2D
- Map points are static,  
i.e., the prediction left them unchanged
- $\epsilon = [\epsilon_x, \epsilon_y, \epsilon_\theta] \sim \mathcal{N}(0, \Sigma_\epsilon)$   
i.e., noise is only on motion



# Prediction Step - Robot motion - Jacobians - 1

## STATE PREDICTION - JACOBIANS WRT STATE

$$[\mathbf{x}_t^T, m^T]^T = g(\mathbf{x}_{t-1}, \mathbf{u}_t, m, \epsilon)$$

$$\mathbf{G}_t = \left. \frac{\partial g(\mathbf{x}, \mathbf{u}, m, \epsilon)}{\partial \mathbf{X}} \right|_{\mathbf{x}=\boldsymbol{\mu}_{t-1}, \mathbf{u}=\mathbf{u}_t, m=\mu_m, \epsilon=0}$$

$$= \begin{bmatrix} \frac{\partial g_1(\mathbf{x}, \mathbf{u}, m, \epsilon)}{\partial x} & \frac{\partial g_1(\mathbf{x}, \mathbf{u}, m, \epsilon)}{\partial y} & \frac{\partial g_1(\mathbf{x}, \mathbf{u}, m, \epsilon)}{\partial \theta} & \frac{\partial g_1(\mathbf{x}, \mathbf{u}, m, \epsilon)}{\partial \mathbf{p}_{x_1}^{(W)}} & \frac{\partial g_1(\mathbf{x}, \mathbf{u}, m, \epsilon)}{\partial \mathbf{p}_{y_1}^{(W)}} & \dots & \frac{\partial g_1(\mathbf{x}, \mathbf{u}, m, \epsilon)}{\partial \mathbf{p}_{y_n}^{(W)}} \\ \frac{\partial g_2(\mathbf{x}, \mathbf{u}, m, \epsilon)}{\partial x} & \frac{\partial g_2(\mathbf{x}, \mathbf{u}, m, \epsilon)}{\partial y} & \frac{\partial g_2(\mathbf{x}, \mathbf{u}, m, \epsilon)}{\partial \theta} & \frac{\partial g_2(\mathbf{x}, \mathbf{u}, m, \epsilon)}{\partial \mathbf{p}_{x_1}^{(W)}} & \frac{\partial g_2(\mathbf{x}, \mathbf{u}, m, \epsilon)}{\partial \mathbf{p}_{y_1}^{(W)}} & \dots & \frac{\partial g_2(\mathbf{x}, \mathbf{u}, m, \epsilon)}{\partial \mathbf{p}_{y_n}^{(W)}} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \frac{\partial g_n(\mathbf{x}, \mathbf{u}, m, \epsilon)}{\partial x} & \frac{\partial g_n(\mathbf{x}, \mathbf{u}, m, \epsilon)}{\partial y} & \frac{\partial g_n(\mathbf{x}, \mathbf{u}, m, \epsilon)}{\partial \theta} & \frac{\partial g_n(\mathbf{x}, \mathbf{u}, m, \epsilon)}{\partial \mathbf{p}_{x_1}^{(W)}} & \frac{\partial g_n(\mathbf{x}, \mathbf{u}, m, \epsilon)}{\partial \mathbf{p}_{y_1}^{(W)}} & \dots & \frac{\partial g_n(\mathbf{x}, \mathbf{u}, m, \epsilon)}{\partial \mathbf{p}_{y_n}^{(W)}} \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & -\sin(\theta)\Delta x - \cos(\theta)\Delta y & 0 & 0 & 0 \\ 0 & 1 & \cos(\theta)\Delta x - \sin(\theta)\Delta y & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{G}_{motion_t} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \Rightarrow \begin{cases} \text{few} \neq 0 \\ \text{diagonal} = 1 \end{cases}$$

## Prediction Step - Robot motion - Jacobians - 2

### STATE PREDICTION - JACOBIANS WRT NOISE

$$\mathbf{N}_t = \left. \frac{\partial g(\mathbf{x}, \mathbf{u}, m, \epsilon)}{\partial \epsilon} \right|_{\mathbf{x}=\boldsymbol{\mu}_{t-1}, \mathbf{u}=\mathbf{u}_t, m=\mu_m, \epsilon=0} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ & \dots & \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} \mathbf{N}_{motion_t} \\ \mathbf{0} \end{bmatrix}$$

### PREDICTION STEP

$$\bar{\boldsymbol{\mu}}_t = g(\boldsymbol{\mu}_{t-1}, \mathbf{u}_t, 0)$$

$$\begin{aligned} \bar{\boldsymbol{\Sigma}}_t &= \mathbf{G}_t \boldsymbol{\Sigma}_{t-1} \mathbf{G}_t^T + \mathbf{N}_t \mathbf{R}_t \mathbf{N}_t^T = \\ &= \begin{bmatrix} \mathbf{G}_{motion_t} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{\Sigma}_{xx} & \boldsymbol{\Sigma}_{xm} \\ \boldsymbol{\Sigma}_{xm}^T & \boldsymbol{\Sigma}_{mm} \end{bmatrix} \begin{bmatrix} \mathbf{G}_{motion_t}^T & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} + \begin{bmatrix} \mathbf{N}_{motion_t} \\ \mathbf{0} \end{bmatrix} \mathbf{R}_t \begin{bmatrix} \mathbf{N}_{motion_t}^T & \mathbf{0} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{G}_{motion_t} \boldsymbol{\Sigma}_{xx} \mathbf{G}_{motion_t}^T + \mathbf{N}_{motion_t} \mathbf{R}_t \mathbf{N}_{motion_t}^T & \mathbf{G}_{motion_t} \boldsymbol{\Sigma}_{xm} \\ \boldsymbol{\Sigma}_{xm}^T \mathbf{G}_{motion_t}^T & \boldsymbol{\Sigma}_{mm} \end{bmatrix} \end{aligned}$$

$\Rightarrow$  Only top-left block and two band are changed, most remains unchanged  
this allow to speed up computation  $\Rightarrow O(n)$

## EKF STATE

-

# Outline

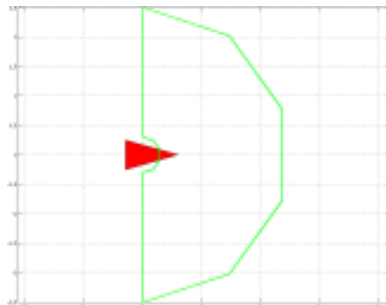
- 1 Introduction
- 2 EKF-SLAM
- 3 Landmark Addition**
- 4 Measurement & Update
- 5 SLAM example
- 6 Correspondences
- 7 Visual SLAM
- 8 Conclusion



# The sensor

## THE SENSOR

- Measure points in *polar coordinates*  
i.e.,  $\rho$ ,  $\theta$  values
- w.r.t. robot reference frame
- It recognize the ID of the landmark
  - i.e., Landmarks uniquely identifiable
  - Correspondences are known
  - No data association issues
- Physical limits:
  - Min and max distance
  - Min and max angle
  - Additive zero mean noise on measures  
both for distance and angle



# New Feature Addition - 1

## SENSOR MEASUREMENT

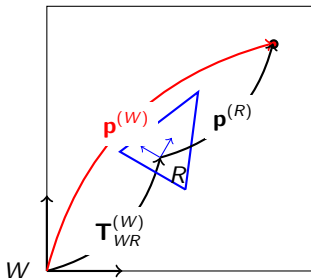
- Suppose the point  $i$  is perceived and is not currently in the map
- $\rho_i, \theta_i$ : the point in polar coordinate, perceived by the sensor
- $\mathbf{p}_i^{(R)} = [\rho_i \cos(\theta_i), \rho_i \sin(\theta_i)]$

## “INVERSE” MEASUREMENT

- $\mathbf{p}_i^{(W)} = \mathbf{T}_{WR}^{(W)} \mathbf{p}_i^{(R)}$

## CONSIDER NOISE

- $\eta = [\eta_\rho, \eta_\theta]^T \sim \mathcal{N}(\mathbf{0}, \Sigma_\eta)$
- $\tilde{\rho}_i = \rho_i + \eta_\rho$
- $\tilde{\theta}_i = \theta_i + \eta_\theta$
- $\tilde{\mathbf{p}}_i^{(R)} = [\tilde{\rho}_i \cos(\tilde{\theta}_i), \tilde{\rho}_i \sin(\tilde{\theta}_i)]$
- $\tilde{\mathbf{p}}_i^{(W)} = \mathbf{T}_{WR}^{(W)} \tilde{\mathbf{p}}_i^{(R)}$





$$\mathbf{F} = \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \dots & \mathbf{0} & \mathbf{0} \\ & & & \dots & & \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{I} & \mathbf{0} \\ \frac{\partial f_{n+1}(\cdot)}{\partial \mathbf{x}} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \end{bmatrix}$$

## New Feature Addition - 3

### JACOBIANS

$$\mathbf{F} = \frac{\partial f(\cdot)}{\partial \mathbf{x}} = \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \frac{\partial f_{n+1}(\cdot)}{\partial \mathbf{x}} & \mathbf{0} & \mathbf{0} \end{bmatrix}$$

$$\mathbf{N} = \frac{\partial f(\cdot)}{\partial \eta} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \frac{\partial f_{n+1}(\cdot)}{\partial \eta} \end{bmatrix}$$

matrices are sparse

### THE NEW STATE

- $\mu = f(\mu_{\mathbf{x}}, \mu_{\mathbf{m}}, [\rho_{new}, \theta_{new}], 0)$
- $\Sigma = \mathbf{F}\Sigma^*\mathbf{F}^T + \mathbf{N}\Sigma_{\eta}\mathbf{N}^T$
- $\Sigma^*$  is the covariance with the increased size
- Products are simple due to sparsity

### NOTES

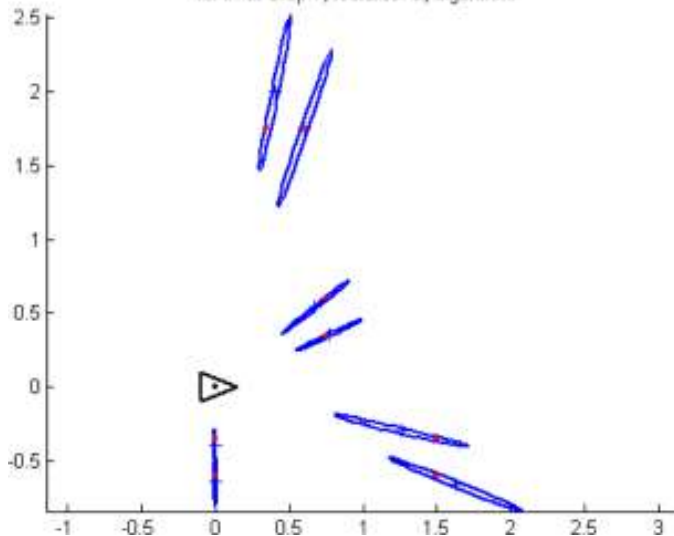
- A new feature is added to the state
- Measure uncertainty is taken into account (thanks to  $\eta$ )
- Robot position uncertainty is taken into account (thanks to  $\mathbf{T}_{WR}^{(W)}$ )



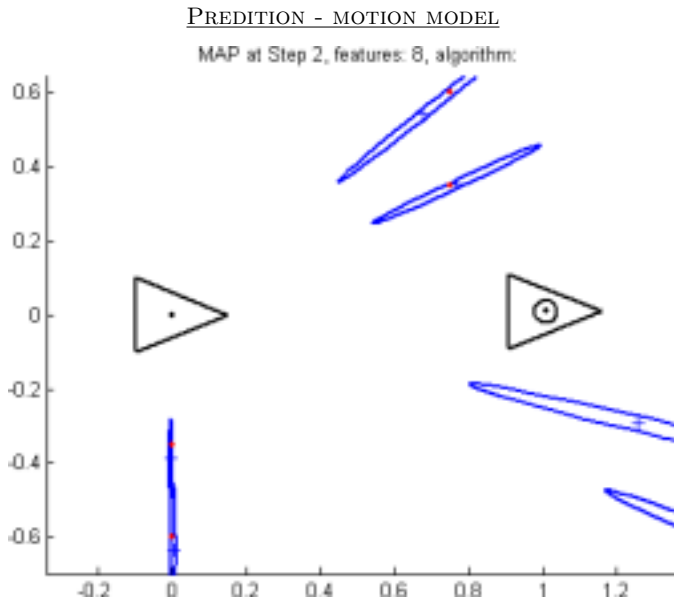
## Qualitative example - 2

### ADDITION TO THE STATE

MAP at Step 1, features: 8, algorithm:



## Qualitative example - 3



# Outline

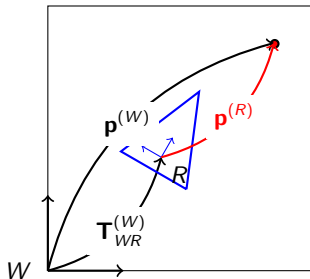
- 1 Introduction
- 2 EKF-SLAM
- 3 Landmark Addition
- 4 Measurement & Update**
- 5 SLAM example
- 6 Correspondences
- 7 Visual SLAM
- 8 Conclusion



# Measurement & Update Step - The equation

## MEASUREMENT

- Measure:  $h_i(\mathbf{x}, m, \delta)$ 
  - It express what we expect from the sensor
  - Given the estimate robot pose  $\mathbf{x} \rightarrow \mathbf{T}_{WR}^{(W)}$
  - Given a single estimated map point  $\mathbf{p}_i^{(W)}$  that is in the EKF state too!
  - i.e.,  $\mathbf{p}_i^{(R)}$  in polar coordinates wrt



## MEASUREMENT

- $\mathbf{p}_i^{(R)} = (\mathbf{T}_{WR}^{(W)})^{-1} \mathbf{p}_i^{(W)}$
- $\rho_i = \sqrt{\mathbf{p}_{i_x}^{(R)^2} + \mathbf{p}_{i_y}^{(R)^2}}$
- $\theta_i = \text{atan2}(\mathbf{p}_{i_y}^{(R)}, \mathbf{p}_{i_x}^{(R)})$

## MEASUREMENT WITH NOISE

- $h_i(\mathbf{x}, m, \delta_i) = \begin{cases} \tilde{\rho}_i = \sqrt{\mathbf{p}_{i_x}^{(R)^2} + \mathbf{p}_{i_y}^{(R)^2}} + \delta_{\rho_i} \\ \tilde{\theta}_i = \text{atan2}(\mathbf{p}_{i_y}^{(R)}, \mathbf{p}_{i_x}^{(R)}) + \delta_{\theta_i} \end{cases}$
- $\delta_i = [\delta_{\rho_i}, \delta_{\theta_i}]^T \sim \mathcal{N}(0, \mathbf{Q}_i)$

## Measurement & Update Step - Jacobians

### MEASUREMENT EQUATION

$$h_i(\mathbf{x}, m, \delta_i) = \begin{cases} \sqrt{\mathbf{p}_{i_x}^{(R)^2} + \mathbf{p}_{i_y}^{(R)^2}} + \delta_{\rho_i} \\ \text{atan2}(\mathbf{p}_{i_y}^{(R)}, \mathbf{p}_{i_x}^{(R)}) + \delta_{\theta_i} \end{cases}$$

$$\mathbf{p}_i^{(R)} = (\mathbf{T}_{WR}^{(W)})^{-1} \mathbf{p}_i^{(W)}$$

### EKF JACOBIANS

•  $\mathbf{H}_i = \frac{\partial h_i(\mathbf{x}, m, \delta_i)}{\partial \mathbf{x}} \Big|_{\mathbf{x}=\bar{\boldsymbol{\mu}}_{t-1}, \mathbf{p}=\mathbf{p}_i^{(W)}, \delta_i=0}$   
derivate of the measurement function  
w.r.t. state variables

•  $\mathbf{M}_i = \frac{\partial h_i(\mathbf{x}, m, \delta_i)}{\partial \delta_i} \Big|_{\mathbf{x}=\bar{\boldsymbol{\mu}}_{t-1}, \mathbf{p}=\mathbf{p}_i^{(W)}, \delta_i=0}$   
derivate of the measurement function  
w.r.t. noise variables

### JACOBIANS

$$\mathbf{H}_i = \begin{bmatrix} \frac{\partial h_i(\mathbf{X}, m, \delta)}{\partial \mathbf{x}} & \frac{\partial h_i(\mathbf{x}, m, \delta)}{\partial \mathbf{p}_1^{(W)}} & \frac{\partial h_i(\mathbf{x}, m, \delta)}{\partial \mathbf{p}_2^{(W)}} & \dots & \frac{\partial h_i(\mathbf{x}, m, \delta)}{\partial \mathbf{p}_i^{(W)}} & \dots & \frac{\partial h_i(\mathbf{x}, m, \delta)}{\partial \mathbf{p}_n^{(W)}} \end{bmatrix}$$

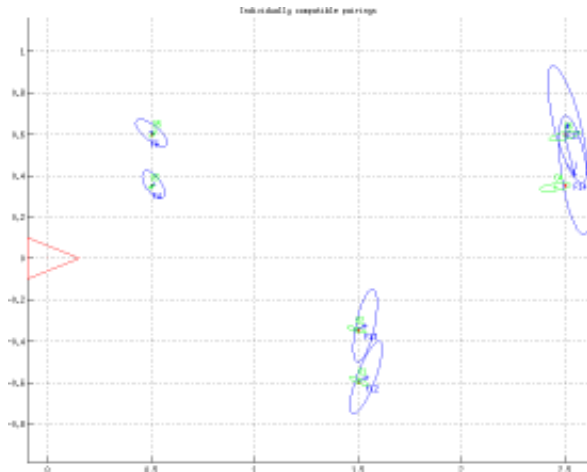
$$= \begin{bmatrix} \frac{\partial h_i(\mathbf{X}, m, \delta)}{\partial \mathbf{x}} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \frac{\partial h_i(\mathbf{x}, m, \delta)}{\partial \mathbf{p}_i^{(W)}} & \mathbf{0} & \dots & \mathbf{0} \end{bmatrix}$$

$$\mathbf{M}_i = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Very sparse, useful to speed up calculation.



## MEASUREMENT IN ROBOT FRAME



- Ellipses: given by covariance  

$$\mathbf{S}_t = \mathbf{H}_t \bar{\Sigma}_t \mathbf{H}_t^T + \mathbf{M}_t \mathbf{Q}_t \mathbf{M}_t^T$$
- Innovation:  $\mathbf{z}_i - h_i(\cdot)$

- $h_i(\cdot), z_i(\cdot), \mathbf{H}_i, \mathbf{M}_i(\cdot), \mathbf{Q}_i(\cdot)$   
feasible measurements and Jacobians
- How to update?

$$\bullet \quad h(\mathbf{x}, \mathbf{m}, \delta) = \begin{cases} h_1(\mathbf{x}, \mathbf{p}_1^{(W)}, \delta_1) \\ h_2(\mathbf{x}, \mathbf{p}_2^{(W)}, \delta_2) \\ \dots \\ h_m(\mathbf{x}, \mathbf{p}_m^{(W)}, \delta_m) \end{cases}$$

$$\bullet \quad \delta = [\delta_1^T \quad \delta_2^T \quad \dots \quad \delta_m^T]^T$$

$$\mathbf{H} = \begin{bmatrix} \frac{\partial h_1(\mathbf{x}, \mathbf{p}_1^{(W)}, \delta_1)}{\partial \mathbf{x}} \\ \frac{\partial h_2(\mathbf{x}, \mathbf{p}_2^{(W)}, \delta_2)}{\partial \mathbf{x}} \\ \vdots \\ \frac{\partial h_m(\mathbf{x}, \mathbf{p}_m^{(W)}, \delta_m)}{\partial \mathbf{x}} \end{bmatrix} \quad \mathbf{M} = \begin{bmatrix} \frac{\partial h_1(\mathbf{x}, \mathbf{p}_1^{(W)}, \delta_1)}{\partial \delta_1} \\ \frac{\partial h_2(\mathbf{x}, \mathbf{p}_2^{(W)}, \delta_2)}{\partial \delta_2} \\ \vdots \\ \frac{\partial h_m(\mathbf{x}, \mathbf{p}_m^{(W)}, \delta_m)}{\partial \delta_m} \end{bmatrix}$$

$$\mathbf{h} = [h_1^T \quad h_2^T \quad \dots \quad h_m^T]^T$$

$$\mathbf{H} = [\mathbf{H}_1^T \quad \mathbf{H}_2^T \quad \cdots \quad \mathbf{H}_m^T]^T$$

$$\mathbf{z} = \begin{bmatrix} \mathbf{z}_1^T & \mathbf{z}_2^T & \cdots & \mathbf{z}_m^T \end{bmatrix}^T$$

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_1 & 0 & \cdots & 0 \\ 0 & \mathbf{M}_2 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & \mathbf{M}_{m-1} & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & 0 & \mathbf{M}_m \end{bmatrix}$$

$$\mathbf{Q} = \begin{bmatrix} \mathbf{Q}_1 & 0 & \cdots & 0 \\ 0 & \mathbf{Q}_2 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & \cdots & 0 \\ \cdots & \cdots & \mathbf{Q}_{m-1} & \cdots \\ 0 & \cdots & 0 & \mathbf{Q}_m \end{bmatrix}$$

## Measurement & Update Step - Unique Update - 2

Notice that

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_1 \\ \mathbf{H}_2 \\ \vdots \\ \mathbf{H}_i \\ \vdots \\ \mathbf{H}_m \end{bmatrix} = \begin{bmatrix} \frac{\partial h_1(\mathbf{X}, m, \delta)}{\partial \mathbf{x}} & \frac{\partial h_1(\mathbf{X}, m, \delta)}{\partial \mathbf{p}_1^{(W)}} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \frac{\partial h_2(\mathbf{X}, m, \delta)}{\partial \mathbf{x}} & \mathbf{0} & \frac{\partial h_2(\mathbf{X}, m, \delta)}{\partial \mathbf{p}_2^{(W)}} & \mathbf{0} & \dots & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial h_i(\mathbf{X}, m, \delta)}{\partial \mathbf{x}} & \mathbf{0} & \dots & \mathbf{0} & \frac{\partial h_i(\mathbf{X}, m, \delta)}{\partial \mathbf{p}_i^{(W)}} & \mathbf{0} \dots \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial h_m(\mathbf{X}, m, \delta)}{\partial \mathbf{x}} & \mathbf{0} & \dots & \dots & \mathbf{0} \dots & \frac{\partial h_m(\mathbf{X}, m, \delta)}{\partial \mathbf{p}_n^{(W)}} \end{bmatrix}$$

is very sparse, it has two non zero blocks for each row

This is very useful for real time implementations

$$\mathbf{x}_0^B = \mathbf{0}; \mathbf{P}_0^B = \mathbf{0} \text{ \{Map initialization\}}$$
$$[z_0, R_0] = \text{get\_measurements}$$
$$[\mathbf{x}_0^B, \mathbf{P}_0^B] = \text{add\_new\_features}(\mathbf{x}_0^B, \mathbf{P}_0^B, \mathbf{z}_0, \mathbf{R}_0)$$
for  $k = 1$  to steps do
$$[\mathbf{x}_{R_k}^{R_{k-1}}, \mathbf{Q}_k] = \text{get\_odometry}$$
$$[\mathbf{x}_{k|k-1}^B, \mathbf{P}_{k|k-1}^B] = \text{EKF\_prediction}(\mathbf{x}_{k-1}^B, \mathbf{P}_{k-1}^B, \mathbf{x}_{Rk-1}^{Rk-1}, \mathbf{Q}_k)$$
$$[z_k, \mathbf{R}_k] = \text{get\_measurements}$$
$$\mathcal{H}_k = \text{data\_association}(\mathbf{x}_{k|k-1}^B, \mathbf{P}_{k|k-1}^B, \mathbf{z}_k, \mathbf{R}_k)$$
$$[\mathbf{x}_k^B, \mathbf{P}_k^B] = \text{EKF\_update}(\mathbf{x}_{k|k-1}^B, \mathbf{P}_{k|k-1}^B, \mathbf{z}_k, \mathbf{R}_k, \mathcal{H}_k)$$
$$[\mathbf{x}_k^B, \mathbf{P}_k^B] = \text{add\_new\_features}(\mathbf{x}_k^B, \mathbf{P}_k^B, \mathbf{z}_k, \mathbf{R}_k, \mathcal{H}_k)$$

end for

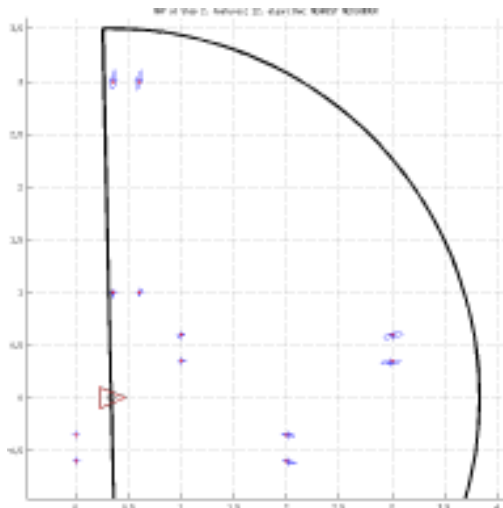
# Outline

- 1 Introduction
- 2 EKF-SLAM
- 3 Landmark Addition
- 4 Measurement & Update
- 5 SLAM example**
- 6 Correspondences
- 7 Visual SLAM
- 8 Conclusion



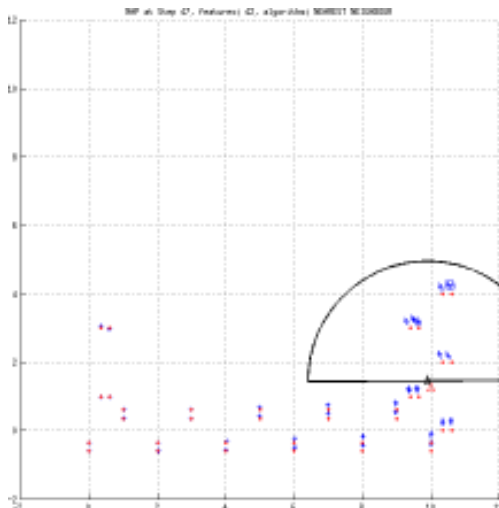
# Some iterations - 1

## SECOND STEP



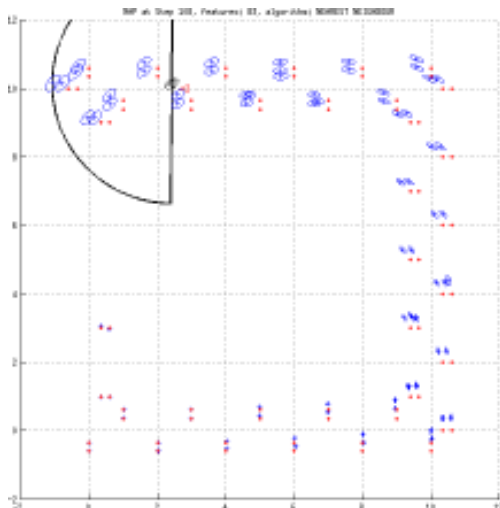
# Some iterations - 2

47<sup>th</sup> STEP



# Some iterations - 3

108<sup>th</sup> STEP





# Some iterations - 4

137<sup>th</sup> STEP

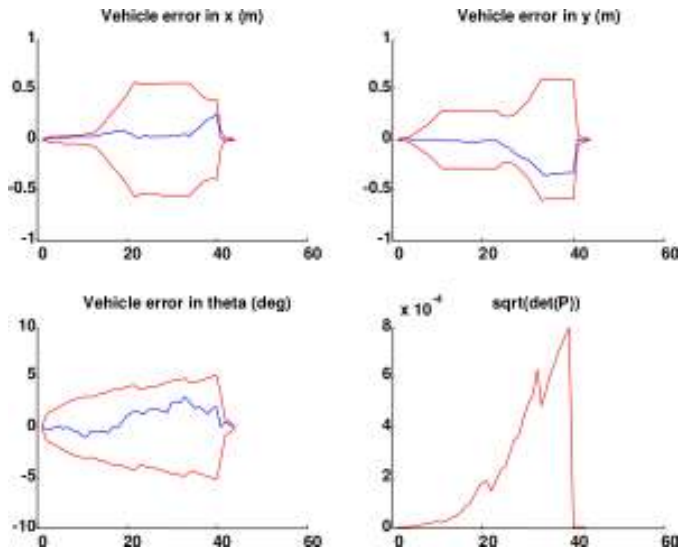






# The role of *loop closure* - 2

## UNCERTAINTY ON ROBOT POSE



# Outline

- 1 Introduction
- 2 EKF-SLAM
- 3 Landmark Addition
- 4 Measurement & Update
- 5 SLAM example
- 6 Correspondences**
- 7 Visual SLAM
- 8 Conclusion



# Correspondences

## CORRESPONDENCES

- Correspondences are known  $\rightarrow$  this is uncommon in real environments
- If correspondences are unknown we have to perform the *data association*

## DATA ASSOCIATION

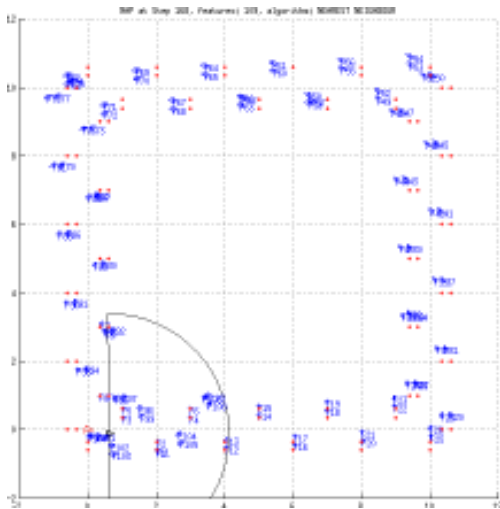
- Given a set of measurements  $\{\mathbf{z}_i\}, i = 1 : m$
- Given a set of *measurements prediction*  $\{\mathbf{h}_j\}, j = 1 : w$
- We have to select correspondences  $c_{ij}$
- Or to add measurements as new landmarks

## MAHALANOBIS DISTANCE NEAREST NEIGHBOURS APPROACH

- 1  $k = 1$
- 2 Select  $w$  such that  $\mathbf{z}_w$  closest to  $\mathbf{h}_k$  in  $D^2(\mathbf{z}_w, \mathbf{h}_k)$
- 3 Remove  $\mathbf{z}_w$  from  $\{\mathbf{z}_i\}$
- 4 Repeat from 2
- 5 Incompatible measures are added as new landmarks

## Data association errors

## WRONG ASSOCIATIONS $\Rightarrow$ BAD RESULTS



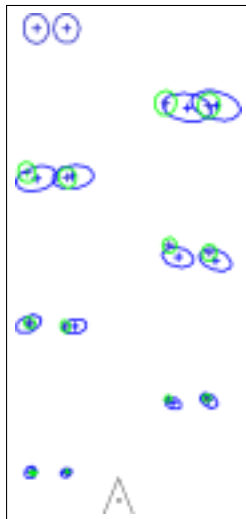
[Matlab: RUN1]



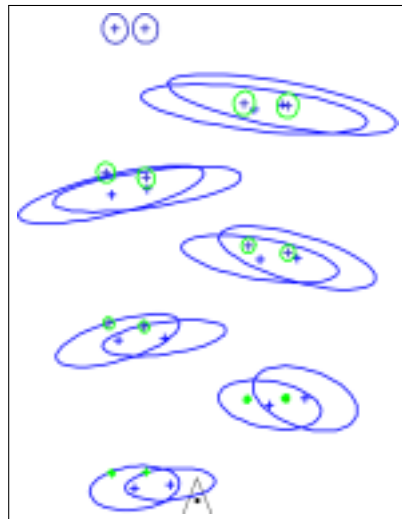


## When data association is difficult - 2

LOW ODOMETRY ERROR



HIGH ODOMETRY ERROR





# Nearest Neighbour Data association pitfall

## MAHALANOBIS DISTANCE

- Evaluate *Individual Compatibility*

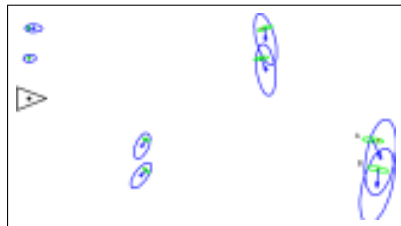


- This could result in wrong associations



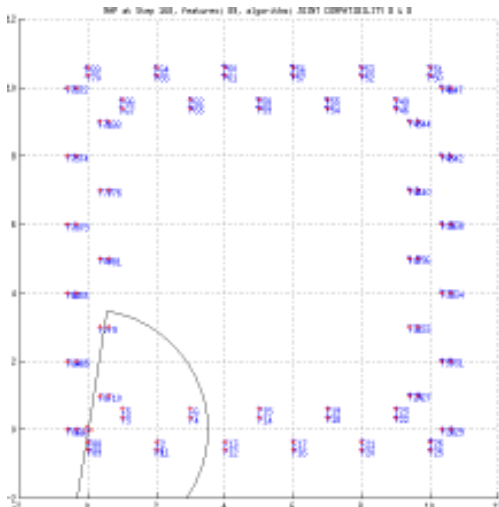
## JOINT COMPATIBILITY

- Evaluate Mahalanobis distance on a subset of associations
- To reduce computational complexity use Branch & Bound technique
- This performs better than Individual Compatibility



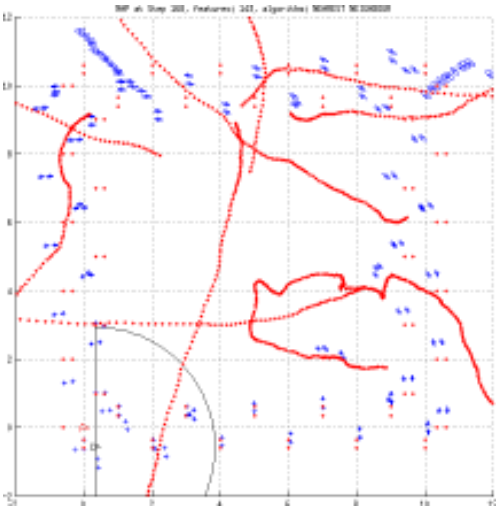
# Joint Compatibility Branch And Bound (JCBB)

## USING JCBB DATA ASSOCIATION



# Non-static environment - 1

## PEOPLE WALKING IN THE CLOISTER

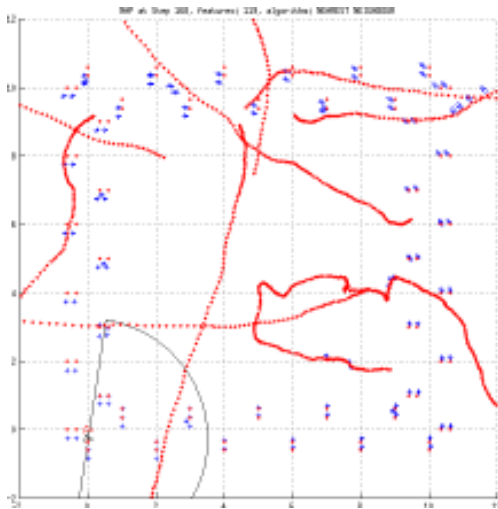


● Using Nearest Neighbour

[Matlab: RUN3]

# Non-static environment - 2

## PEOPLE WALKING IN THE CLOISTER

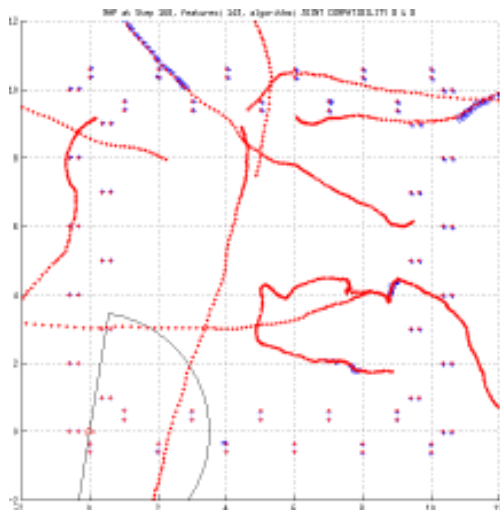


- Using Nearest Neighbour
- Delete landmarks that have a measurement prediction but are not matched for a while

[Matlab: RUN4]

# Non-static environment - 3

## PEOPLE WALKING IN THE CLOISTER

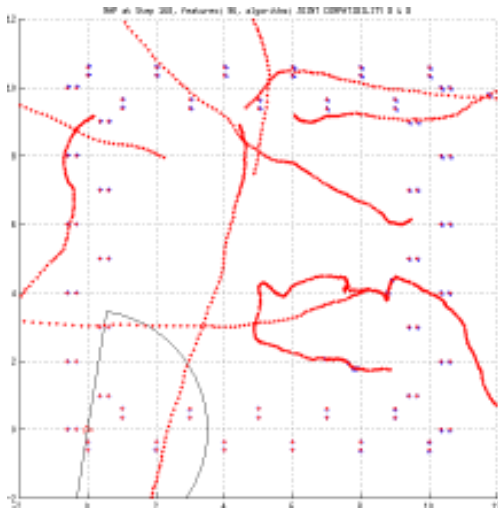


● Using JCBB

[Matlab: RUN5]

## Non-static environment - 4

### PEOPLE WALKING IN THE CLOISTER



- Using JCBM
- Delete landmarks that have a measurement prediction but are not matched for a while

[Matlab: RUN6]



# A note on motion model

## MOTION MODEL

- We have always used odometry as input
- This controls the robot motion in the prediction step
- Absolutely necessary? **NO!**

## STEADY STATE MOTION MODEL

- $x_{t+1} = x_t + \eta_x$
- $y_{t+1} = y_t + \eta_y$
- $\theta_{t+1} = \theta_t + \eta_\theta$
- The noise “code” the (unknown) motion

## CONSTANT VELOCITY MOTION MODEL

- $x_{t+1} = x_t + v_t \cos(\theta_t) \Delta t$
- $y_{t+1} = y_t + v_t \sin(\theta_t) \Delta t$
- $\theta_{t+1} = \theta_t + w_t \Delta t$
- $v_{t+1} = v_t + \eta_v$
- $w_{t+1} = w_t + \eta_w$
- Suppose speed is constant in  $\Delta t$
- The noise “code” the (unknown) speed change
- Measurements change position and speed thanks to correlations



# Visual SLAM

## VISUAL SLAM PROPERTIES

- Rely only on camera(s)  
solution with one camera easily  
extends on multi camera systems
- Extensible with measures  
motion, GPS position, ...
- Smart and cheap
- Challenging  
lack of depth with one camera
- Could be solved in Real Time



## EKF-BASED SLAM

- The most consolidated methodology
- Use an Extended Kalman Filter as *engine*
- State vector (multivariate gaussian variable):
  - robot pose
  - map points
- *Predict* robot motion
- *Observe* features in image

### PRO:

- Could run in Real-Time on standard PC
- Well known approach
- Scalability to large scale  
through sub-mapping techniques

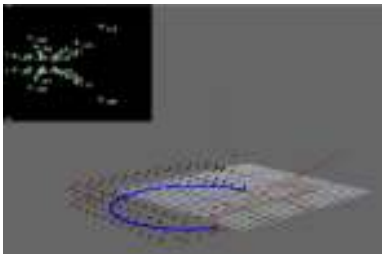
### CONS:

- Needs a specific *parametrization* of points
- Suffer of approximation

# Landmarks & Features

## LANDMARKS

- Elements of the map
- They code a 3D point  
notice: we consider 3D environment



## FEATURES

- The *measurable quantity* of a landmark
- *Good features to track in image*





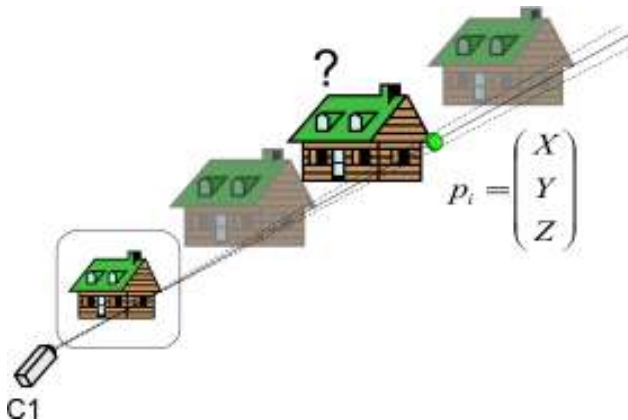
Algorithm that extracts image locations which are easily found in other images of the same scene (repeatability)  $\Rightarrow$  **Corner detector**

Algorithm used to convert a region around a detected keypoint into a more compact and stable (invariant) form that can be successfully matched against other descriptors (saliency)  $\Rightarrow$  **Patch around the corner**

An algorithm that efficiently searches for likely matching candidates in other images even when large amount of motion or appearance change has occurred

Similar to the previous one but more suitable when images are taken from nearby viewpoints or in rapid succession  $\Rightarrow$  **Template matching with patches**

# Monocular SLAM key problem - 1

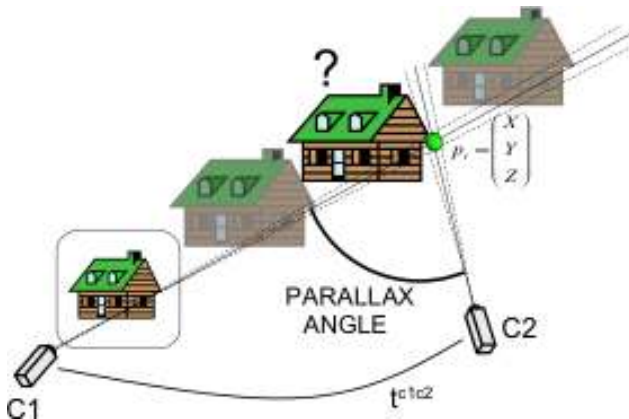


Camera is a bearing-only sensor

Depth is unknown from a single image

Depth can be estimated with triangulation after camera motion

## Monocular SLAM key problem - 2



Depth can be estimated with triangulation after camera motion

Parallax angle cover a key role



# Monocular SLAM key problem - 3

## FEATURE DEPTH

- Unknown at initialization
- Uniform distribution from 0 to  $\infty$

## SOLUTION 1: DELAYED INITIALIZATION

For each feature

- Use a set of 3D hypothesis on view ray
- Choose the right depth hypothesis
- Add it to the filter

## SOLUTION 2: UNDELAYED INITIALIZATION

For each feature

- Add one  $n$ -dimensional element that code
  - The viewing ray
  - The unknown depth
- following a specific *Parametrization*



# Real Time Monocular SLAM

## REAL TIME MONOCULAR SLAM - SINCE 2003

videos/monoRT.flv

Video from <http://www.youtube.com/watch?v=mimAWVm-0qA>

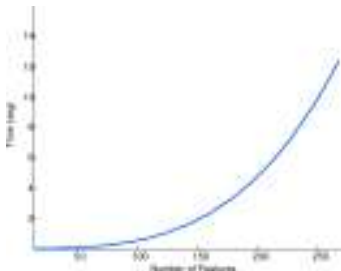
Davison "Real-time Simultaneous Localization And Mapping with a Single Camera", 2003



# Large Scale SLAM Issues

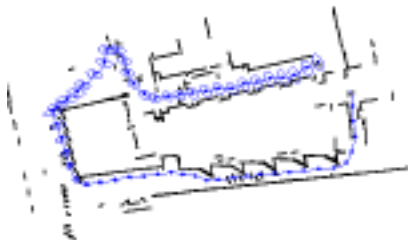
## COMPUTATIONAL COST

Grows with # features



## CONSISTENCY

Due to linearizations of EKF



## SOME SOLUTION

- Conditional Independent Submapping SLAM
- Explicit Loop Detection & Loop closure recovery methods

Pinies, Tardos "Large Scale SLAM Building Conditionally Independent Local Maps: Application to Monocular Vision", 2008

Pinies, Paz, Tardos "CI-Graph: An efficient approach for Large Scale SLAM", 2009

**The map is represented by points location**

**Theoretically  
reconstruction is up to a  
single scale factor**

**Practically there is a scale drift**

The map is represented by points location

**The stereo vision  
eliminate the scale  
factor ambiguity**

## Example in a Real Environment - Trinocular Vision

**The path is estimated without any external information, using a constant velocity motion model**

The map is represented by points location

videos/tri.flv





## Example in a Real Environment - Omnidirectional Camera - 2

**The path is estimated without any external information, using a constant velocity motion model**

The map is not shown in this case

videos/omni.flv

# Outline

1 Introduction

2 EKF-SLAM

3 Landmark Addition

4 Measurement & Update

5 SLAM example

6 Correspondences

7 Visual SLAM

8 Conclusion



# Only EKF-SLAM?

## NOT ONLY EKF-SLAM

- Particle Filters → FastSLAM & FastSLAM 2.0
- Extended Information Filter
- Parallel Tracking and Mapping (PTAM)
- Junction tree filters
- Incremental Smoothing and Mapping (iSAM)
- Local Sparse Bundle Adjustment
- ...

## PTAM EXAMPLE

videos/ptam.webm

from  
[http : //www.youtube.com/watch?v = Y9HMn6bd - v8](http://www.youtube.com/watch?v=Y9HMn6bd-v8)

# Only EKF-SLAM?

## Laser Range Scanner based SLAM

2D SLAM

3D SLAM

videos/slam2dlaser.webm

from [http : // www.youtube.com/watch?v = flfNOXHxBKY](http://www.youtube.com/watch?v=flfNOXHxBKY)

videos/slam3dlaser.webm

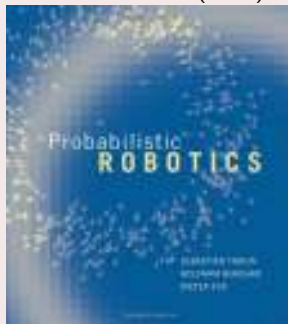
from [http : // www.youtube.com/watch?v = QQeJ1xd5OU](http://www.youtube.com/watch?v=QQeJ1xd5OU)

other sensors: Microsoft Kinect, etc...

## Reference

*(Intelligent Robotics and Autonomous Agents series)*

The MIT Press (2005)



Chapters 2, 3, 4, 7, 10.