SLAM

Simone Ceriani

simone.ceriani@unimib.it
ceriani@ira.disco.unimib.it

Informatics and Robotics for Automation Laboratory
Dipartimento di Informatica Sistemistica e Comunicazione
Università degli Studi di Milano Bicocca

8 February 2013

Outline

Outline

SLAM - Introduction

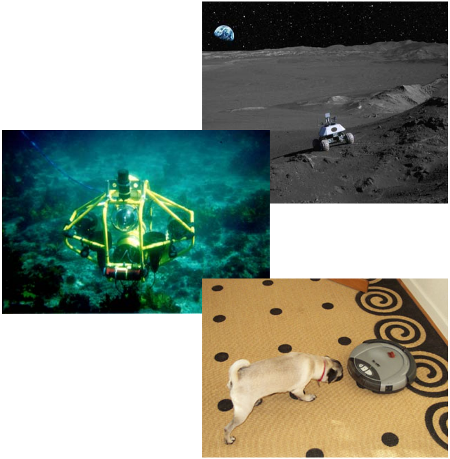
Autonomous robots need

- To localize itself
  e.g., GPS give us the coordinate
- To build the environment map
  e.g., know the plant of a building

$\rightarrow$ without any prior information
$\rightarrow$ in real time

What is SLAM?

- **S**imultaneous
- **L**ocalization
- **A**nd
- **M**apping



Let's start from . . .

Two common problems

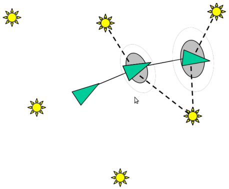
In mobile robotics, but not solely mobile robotics!

Localization

- Given a *map* of the environment
- Given sensor measurements

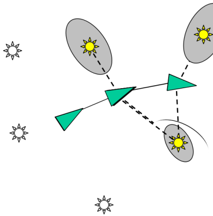
  e.g. images from cameras,

  laser range finder scans, . . .
- **Estimate the robot position**



Mapping

- Given the *robot* position
- Given sensor *measurements*
- **Build the map of the environment**



Localization

LOCALIZATION - A SIMULATED EXAMPLE

Video localization.flv

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

Video from http://www.youtube.com/watch?v=MELYZ5r5V1c

Mapping

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

What happens in the real world?

Knowledge of the map

- Impossible in a lot of applications

  e.g.: exploration of buildings, underwater operations, . . .
- Thus, localization is not applicable

Knowledge of the position

- Usually unavailable or noisy/uncertain

  e.g.: lack of GPS signal in indoor, . . .
- Thus, mapping is not applicable

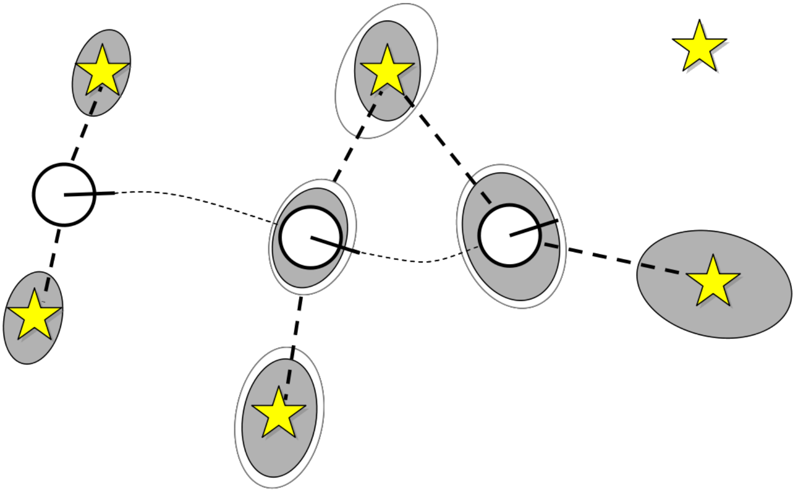
SLAM - **S**imultaneous **L**ocalization **A**nd **M**apping

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

  "Is it possible for a mobile robot to be placed in an **unknown** *location in an* **unknow** *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

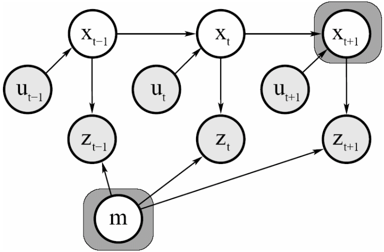
On-line and Full SLAM

On-line SLAM

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

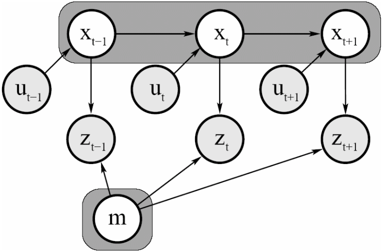
- x_t: pose at time t
- m: map (static)
- $z_{1:t}$: measurements
- $u_{1:t}$: controls
- *On-line*: estimate the current pose
- Most online-SLAM are incremental,

  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



Outline

1. Introduction
2. **EKF-SLAM Algorithm**
3. SLAM example
4. Correspondences
5. Visual SLAM
6. Conclusion

EKF-SLAM Introduction

EKF-SLAM

- Use a EKF as engine for the solution of SLAM problem
- The earliest and perhaps most influential SLAM algorithm
- Map is static and *feature based*, i.e., composed of points, $m = \{\mathbf{p}_i^{(W)}\}$
- For computational reason number of points small, e.g. < 1000
- All noises are assumed to be Gaussian
- Needs of relatively small uncertainty to reduce linearization effects

EKF-SLAM STATE

- $[\mathbf{x}_t, m]$: both pose and map are in the state
- Motion model: only the robot moves, features are static
- Map is initially empty
- Map grows when new landmarks are perceived
- Measures are sensor readings of map landmarks
- Updates refine current robot pose and map structure simultaneously

EKF State Details

State

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

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

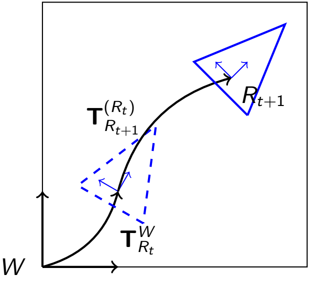
$$\Sigma = \begin{bmatrix} \Sigma_{\mathbf{xx}} & \Sigma_{\mathbf{xm}} \\ \Sigma^T_{\mathbf{xm}} & \Sigma_{\mathbf{mm}} \end{bmatrix}$$

Prediction Step - Robot motion

State prediction

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

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



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

Prediction Step - Robot motion - Jacobians - 1

State prediction - Jacobians wrt state

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

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

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

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

Prediction Step - Robot motion - Jacobians - 2

State prediction - Jacobians wrt noise

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

Prediction Step - Robot motion - Jacobians - 2

State prediction - Jacobians wrt noise

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

Prediction step

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

$$\overline{\Sigma}_t = \mathbf{G}_t \Sigma_{t-1} \mathbf{G}_t^T + \mathbf{N}_t \mathbf{R}_t \mathbf{N}_t^T =$$

$$= \begin{bmatrix} \mathbf{G}_{motion_t} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \Sigma_{\mathbf{xx}} & \Sigma_{\mathbf{xm}} \\ \Sigma_{\mathbf{xm}}^T & \Sigma_{\mathbf{mm}} \end{bmatrix} \begin{bmatrix} \mathbf{G}_{motion_t}^T & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} + \begin{bmatrix} \mathbf{N}_{motion_t} \\ \mathbf{0} \end{bmatrix} \mathbf{R}_t \begin{bmatrix} \mathbf{N}_{motion_t}^T & \mathbf{0} \end{bmatrix}$$

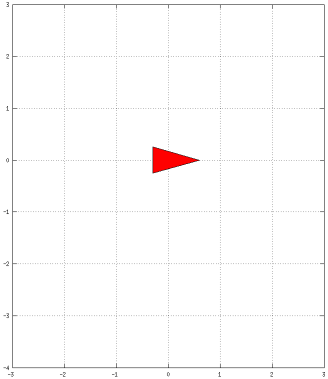
$$= \begin{bmatrix} \mathbf{G}_{motion_t} \Sigma_{\mathbf{xx}} \mathbf{G}_{motion_t}^T + \mathbf{N}_{motion_t} \mathbf{R}_t \mathbf{N}_{motion_t}^T & \mathbf{G}_{motion_t} \Sigma_{\mathbf{xm}} \\ \Sigma_{\mathbf{xm}}^T \mathbf{G}_{motion_t}^T & \Sigma_{\mathbf{mm}} \end{bmatrix}$$

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

Initial state

EKF STATE

- $\mu_0 = [0, 0, 0]$
- $\Sigma_0 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$
- Robot is in the origin
- No uncertainty on its initial position and orientation
- Trajectory and map are reconstructed up to a rototranslation
- The map is empty at initial step



The sensor

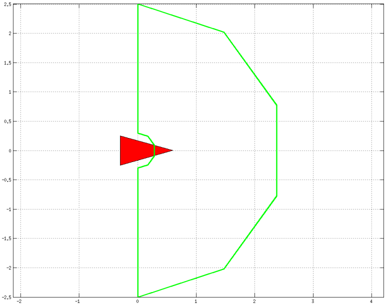
The Sensor

- Measure points in *polar coordinates*

  i.e., ρ, θ 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



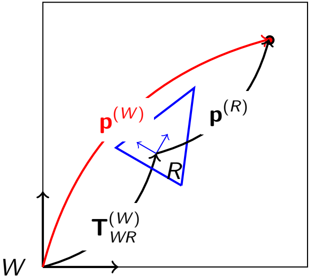
New Feature Addition - 1

Sensor measurement

- Suppose the point i is perceived and is not currently in the map
- ρ_i, θ_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

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

New Feature Addition - 2

Current state

$$\mathbf{X} = \begin{bmatrix} \mathbf{x} & \mathbf{p}_1^{(W)} & \cdots & \mathbf{p}_n^{(W)} & \end{bmatrix}$$

Increase the state dimension

$$\mathbf{X} = \begin{bmatrix} \mathbf{x} & \mathbf{p}_1^{(W)} & \cdots & \mathbf{p}_n^{(W)} & \mathbf{p}_{new}^{(W)} \end{bmatrix}$$

Assign the proper values

State modification:

$$\mathbf{X} = f(\mathbf{x}, \mathbf{m}, [\rho_{new}, \theta_{new}], \eta) =$$

$$= \begin{cases} \mathbf{x} & = & \mathbf{x} \\ \mathbf{p}_1^{(W)} & = & \mathbf{p}_1^{(W)} \\ \mathbf{p}_2^{(W)} & = & \mathbf{p}_2^{(W)} \\ & \cdots & \\ \mathbf{p}_n^{(W)} & = & \mathbf{p}_n^{(W)} \\ \mathbf{p}_{new}^{(W)} & = & \mathbf{T}_{WR}^{(W)} \, \tilde{\mathbf{p}}_{new}^{(R)} \end{cases}$$

New Feature Addition - 2

Current state

$$\mathbf{X} = \begin{bmatrix} \mathbf{x} & \mathbf{p}_1^{(W)} & \cdots & \mathbf{p}_n^{(W)} & \end{bmatrix}$$

Increase the state dimension

$$\mathbf{X} = \begin{bmatrix} \mathbf{x} & \mathbf{p}_1^{(W)} & \cdots & \mathbf{p}_n^{(W)} & \mathbf{p}_{new}^{(W)} \end{bmatrix}$$

Assign the proper values

State modification:

$$\mathbf{X} = f(\mathbf{x}, \mathbf{m}, [\rho_{new}, \theta_{new}], \eta) = \\ = \begin{cases} \mathbf{x} & = & \mathbf{x} \\ \mathbf{p}_1^{(W)} & = & \mathbf{p}_1^{(W)} \\ \mathbf{p}_2^{(W)} & = & \mathbf{p}_2^{(W)} \\ & \cdots & \\ \mathbf{p}_n^{(W)} & = & \mathbf{p}_n^{(W)} \\ \mathbf{p}_{new}^{(W)} & = & \mathbf{T}_{WR}^{(W)} \, \tilde{\mathbf{p}}_{new}^{(R)} \end{cases}$$

Jacobians

$$\mathbf{F} = \frac{\partial f(\cdot)}{\partial \mathbf{X}} =$$

$$\begin{bmatrix} \frac{\partial f_x(\cdot)}{\partial \mathbf{x}} & \frac{\partial f_x(\cdot)}{\partial \mathbf{p}_1^{(W)}} & \frac{\partial f_x(\cdot)}{\partial \mathbf{p}_2^{(W)}} & \cdots & \frac{\partial f_x(\cdot)}{\partial \mathbf{p}_n^{(W)}} & \frac{\partial f_x(\cdot)}{\partial \mathbf{p}_{new}^{(W)}} \\ \frac{\partial f_1(\cdot)}{\partial \mathbf{x}} & \frac{\partial f_1(\cdot)}{\partial \mathbf{p}_1^{(W)}} & \frac{\partial f_1(\cdot)}{\partial \mathbf{p}_2^{(W)}} & \cdots & \frac{\partial f_1(\cdot)}{\partial \mathbf{p}_n^{(W)}} & \frac{\partial f_1(\cdot)}{\partial \mathbf{p}_{new}^{(W)}} \\ \frac{\partial f_2(\cdot)}{\partial \mathbf{x}} & \frac{\partial f_2(\cdot)}{\partial \mathbf{p}_1^{(W)}} & \frac{\partial f_2(\cdot)}{\partial \mathbf{p}_2^{(W)}} & \cdots & \frac{\partial f_2(\cdot)}{\partial \mathbf{p}_n^{(W)}} & \frac{\partial f_2(\cdot)}{\partial \mathbf{p}_{new}^{(W)}} \\ & & & \cdots & & \\ \frac{\partial f_n(\cdot)}{\partial \mathbf{x}} & \frac{\partial f_n(\cdot)}{\partial \mathbf{p}_1^{(W)}} & \frac{\partial f_n(\cdot)}{\partial \mathbf{p}_2^{(W)}} & \cdots & \frac{\partial f_n(\cdot)}{\partial \mathbf{p}_n^{(W)}} & \frac{\partial f_n(\cdot)}{\partial \mathbf{p}_{new}^{(W)}} \\ \frac{\partial f_{n+1}(\cdot)}{\partial \mathbf{x}} & \frac{\partial f_{n+1}(\cdot)}{\partial \mathbf{p}_1^{(W)}} & \frac{\partial f_{n+1}(\cdot)}{\partial \mathbf{p}_2^{(W)}} & \cdots & \frac{\partial f_{n+1}(\cdot)}{\partial \mathbf{p}_n^{(W)}} & \frac{\partial f_{n+1}(\cdot)}{\partial \mathbf{p}_{new}^{(W)}} \end{bmatrix}$$

New Feature Addition - 2

Current state

$$\mathbf{X} = \begin{bmatrix} \mathbf{x} & \mathbf{p}_1^{(W)} & \cdots & \mathbf{p}_n^{(W)} & \end{bmatrix}$$

Increase the state dimension

$$\mathbf{X} = \begin{bmatrix} \mathbf{x} & \mathbf{p}_1^{(W)} & \cdots & \mathbf{p}_n^{(W)} & \mathbf{p}_{new}^{(W)} \end{bmatrix}$$

Assign the proper values

State modification:

$$\mathbf{X} = f(\mathbf{x}, \mathbf{m}, [\rho_{new}, \theta_{new}], \eta) = \\ = \begin{cases} \mathbf{x} & = & \mathbf{x} \\ \mathbf{p}_1^{(W)} & = & \mathbf{p}_1^{(W)} \\ \mathbf{p}_2^{(W)} & = & \mathbf{p}_2^{(W)} \\ & \cdots & \\ \mathbf{p}_n^{(W)} & = & \mathbf{p}_n^{(W)} \\ \mathbf{p}_{new}^{(W)} & = & \mathbf{T}_{WR}^{(W)} \, \tilde{\mathbf{p}}_{new}^{(R)} \end{cases}$$

Jacobians

$$\mathbf{F} = \frac{\partial f(\cdot)}{\partial \mathbf{X}} =$$

$$\begin{bmatrix} \frac{\partial f_x(\cdot)}{\partial \mathbf{x}} & \frac{\partial f_x(\cdot)}{\partial \mathbf{p}_1^{(W)}} & \frac{\partial f_x(\cdot)}{\partial \mathbf{p}_2^{(W)}} & \cdots & \frac{\partial f_x(\cdot)}{\partial \mathbf{p}_n^{(W)}} & \frac{\partial f_x(\cdot)}{\partial \mathbf{p}_{new}^{(W)}} \\ \frac{\partial f_1(\cdot)}{\partial \mathbf{x}} & \frac{\partial f_1(\cdot)}{\partial \mathbf{p}_1^{(W)}} & \frac{\partial f_1(\cdot)}{\partial \mathbf{p}_2^{(W)}} & \cdots & \frac{\partial f_1(\cdot)}{\partial \mathbf{p}_n^{(W)}} & \frac{\partial f_1(\cdot)}{\partial \mathbf{p}_{new}^{(W)}} \\ \frac{\partial f_2(\cdot)}{\partial \mathbf{x}} & \frac{\partial f_2(\cdot)}{\partial \mathbf{p}_1^{(W)}} & \frac{\partial f_2(\cdot)}{\partial \mathbf{p}_2^{(W)}} & \cdots & \frac{\partial f_2(\cdot)}{\partial \mathbf{p}_n^{(W)}} & \frac{\partial f_2(\cdot)}{\partial \mathbf{p}_{new}^{(W)}} \\ & & & \cdots & & \\ \frac{\partial f_n(\cdot)}{\partial \mathbf{x}} & \frac{\partial f_n(\cdot)}{\partial \mathbf{p}_1^{(W)}} & \frac{\partial f_n(\cdot)}{\partial \mathbf{p}_2^{(W)}} & \cdots & \frac{\partial f_n(\cdot)}{\partial \mathbf{p}_n^{(W)}} & \frac{\partial f_n(\cdot)}{\partial \mathbf{p}_{new}^{(W)}} \\ \frac{\partial f_{n+1}(\cdot)}{\partial \mathbf{x}} & \frac{\partial f_{n+1}(\cdot)}{\partial \mathbf{p}_1^{(W)}} & \frac{\partial f_{n+1}(\cdot)}{\partial \mathbf{p}_2^{(W)}} & \cdots & \frac{\partial f_{n+1}(\cdot)}{\partial \mathbf{p}_n^{(W)}} & \frac{\partial f_{n+1}(\cdot)}{\partial \mathbf{p}_{new}^{(W)}} \end{bmatrix}$$

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

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

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

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

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

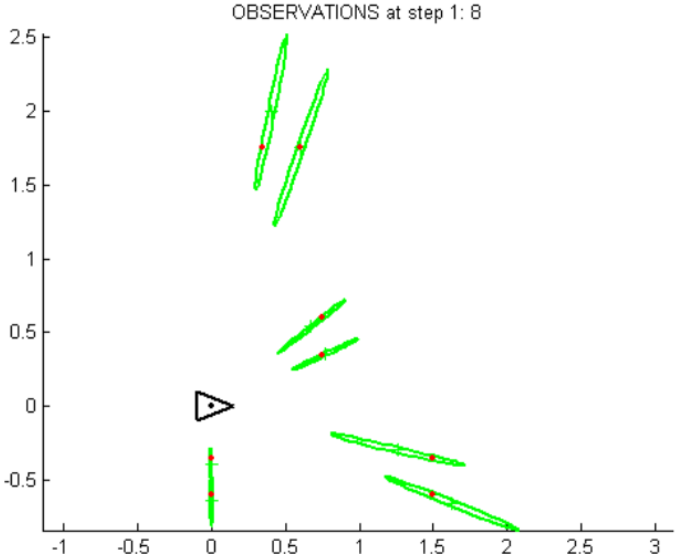
- $\mu = f(\mu_{\mathbf{x}}, \mu_{\mathbf{m}}, [\rho_{new}, \theta_{new}], 0)$
- $\Sigma = \mathbf{F}\Sigma^* \mathbf{F}^T + \mathbf{N}\Sigma_\eta \mathbf{N}^T$
- Σ^* is the covariance with the increased size
- Products are simple due to sparsity

Notes

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

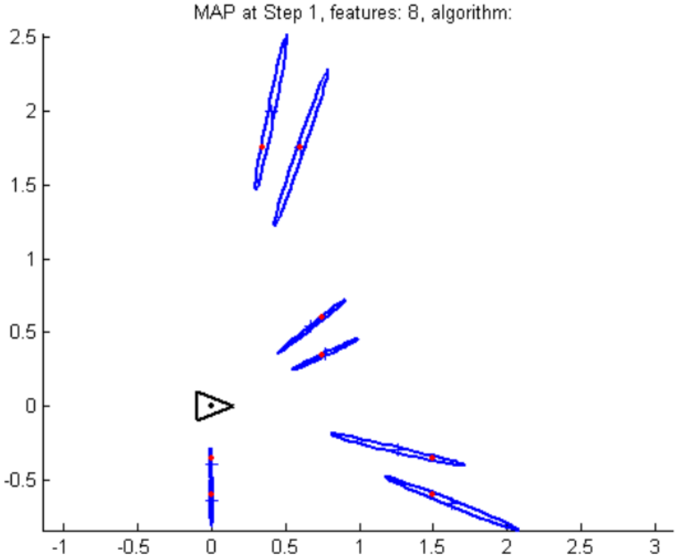
Qualitative example - 1

Sensor readings



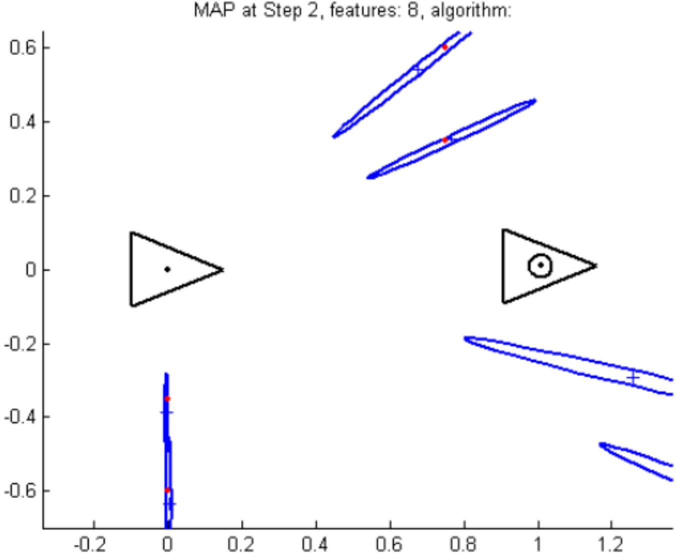
Qualitative example - 2

Addition to the state



Qualitative example - 3

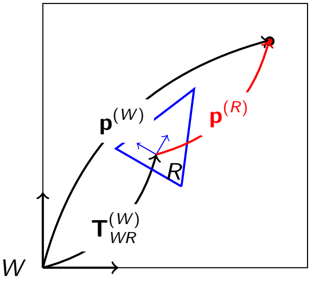
PREDITION - MOTION MODEL



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

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

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

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

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

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

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

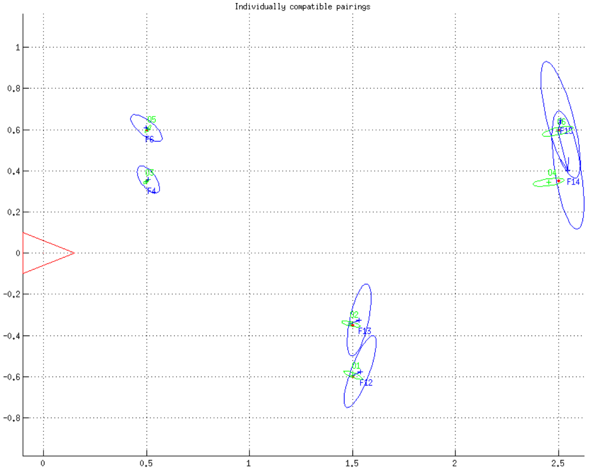
Jacobians

$$\begin{aligned} \mathbf{H}_i &= \begin{bmatrix} \frac{\partial h_i(\mathbf{X},m,\delta)}{\partial \mathbf{x}} & \frac{\partial h_i(\mathbf{x},m,\delta)}{\partial \mathbf{p}_1^{(W)}} & \frac{\partial h_i(\mathbf{x},m,\delta)}{\partial \mathbf{p}_2^{(W)}} & \cdots & \frac{\partial h_i(\mathbf{x},m,\delta)}{\partial \mathbf{p}_i^{(W)}} & \cdots & \frac{\partial h_i(\mathbf{x},m,\delta)}{\partial \mathbf{p}_n^{(W)}} \end{bmatrix} \\ &= \begin{bmatrix} \frac{\partial h_i(\mathbf{X},m,\delta)}{\partial \mathbf{x}} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \frac{\partial h_i(\mathbf{x},m,\delta)}{\partial \mathbf{p}_i^{(W)}} & \mathbf{0} & \cdots & \mathbf{0} \end{bmatrix} \\ \mathbf{M}_i &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \end{aligned}$$

Very sparse, useful to speed up calculation.

Measurement & Update Step - Measurement Details

Measurement in robot frame



- Blue: the *predicted measure*, $h_i(\cdot)$
- Red: the real map point in robot coordinates
- Green: the noisy sensor measurement $\mathbf{z}_i$
- Ellipses: given by covariance $\mathbf{S}_t = \mathbf{H}_t \overline{\Sigma}_t \mathbf{H}_t^T + \mathbf{M}_t \mathbf{Q}_t \mathbf{M}_t^T$
- Innovation: $\mathbf{z}_i - h_i(\cdot)$

Measurement & Update Step - Unique Update - 1

The measurements

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

The complete measurements

- $h(\mathbf{x}, \mathbf{m}, \delta) = \begin{cases} h_1(\mathbf{x}, \mathbf{p}_1^{(W)}, \delta_1) \\ h_2(\mathbf{x}, \mathbf{p}_2^{(W)}, \delta_2) \\ \cdots \\ h_m(\mathbf{x}, \mathbf{p}_m^{(W)}, \delta_m) \end{cases}$
- $\delta = \begin{bmatrix} \delta_1^T & \delta_2^T & \cdots & \delta_m^T \end{bmatrix}^T$

$$\mathbf{H} = \begin{bmatrix} \frac{\partial h_1(\mathbf{x}, \mathbf{p}_1^{(W)}, \delta_1)}{\partial \mathbf{x}} \\ \frac{\partial h_2(\mathbf{x}, \mathbf{p}_2^{(W)}, \delta_2)}{\partial \mathbf{x}} \\ \cdots \\ \frac{\partial h_m(\mathbf{x}, \mathbf{p}_m^{(W)}, \delta_m)}{\partial \mathbf{x}} \end{bmatrix} \quad \mathbf{M} = \begin{bmatrix} \frac{\partial h_1(\mathbf{x}, \mathbf{p}_1^{(W)}, \delta_1)}{\partial \delta_\mathbf{1}} \\ \frac{\partial h_2(\mathbf{x}, \mathbf{p}_2^{(W)}, \delta_2)}{\partial \delta_\mathbf{2}} \\ \cdots \\ \frac{\partial h_m(\mathbf{x}, \mathbf{p}_m^{(W)}, \delta_m)}{\partial \delta_\mathbf{m}} \end{bmatrix}$$

Measurement & Update Step - Unique Update - 1

The measurements

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

The complete measurements

- $$h(\mathbf{x}, \mathbf{m}, \delta) = \begin{cases} h_1(\mathbf{x}, \mathbf{p}_1^{(W)}, \delta_1) \\ h_2(\mathbf{x}, \mathbf{p}_2^{(W)}, \delta_2) \\ \cdots \\ h_m(\mathbf{x}, \mathbf{p}_m^{(W)}, \delta_m) \end{cases}$$
- $\delta = \begin{bmatrix} \delta_1^T & \delta_2^T & \cdots & \delta_m^T \end{bmatrix}^T$

The update

$$\mathbf{h} = \begin{bmatrix} h_1^T & h_2^T & \cdots & h_m^T \end{bmatrix}^T$$

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

$$\mathbf{z} = \begin{bmatrix} z_1^T & z_2^T & \cdots & z_m^T \end{bmatrix}^T$$

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

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

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

Measurement & Update Step - Unique Update - 2

Notice that

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

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

This is very useful for real time implementations

EKF-SLAM, the Algorithm

Algorithm 1 SLAM:

$\mathbf{x}_0^B = \mathbf{0}$; $\mathbf{P}_0^B = \mathbf{0}$ {*Map initialization*}

$[\mathbf{z}_0, \mathbf{R}_0] = \text{get\_measurements}$

$[\mathbf{x}_0^B, \mathbf{P}_0^B] = \text{add\_new\_features}(\mathbf{x}_0^B, \mathbf{P}_0^B, \mathbf{z}_0, \mathbf{R}_0)$

for $k = 1$ to steps **do**

$[\mathbf{x}_{R_k}^{R_{k-1}}, \mathbf{Q}_k] = \text{get\_odometry}$

$[\mathbf{x}_{k|k-1}^B, \mathbf{P}_{k|k-1}^B] = \text{EKF\_prediction}(\mathbf{x}_{k-1}^B, \mathbf{P}_{k-1}^B, \mathbf{x}_{R_k}^{R_{k-1}}, \mathbf{Q}_k)$

$[\mathbf{z}_k, \mathbf{R}_k] = \text{get\_measurements}$

$\mathcal{H}_k = \text{data\_association}(\mathbf{x}_{k|k-1}^B, \mathbf{P}_{k|k-1}^B, \mathbf{z}_k, \mathbf{R}_k)$

$[\mathbf{x}_k^B, \mathbf{P}_k^B] = \text{EKF\_update}(\mathbf{x}_{k|k-1}^B, \mathbf{P}_{k|k-1}^B, \mathbf{z}_k, \mathbf{R}_k, \mathcal{H}_k)$

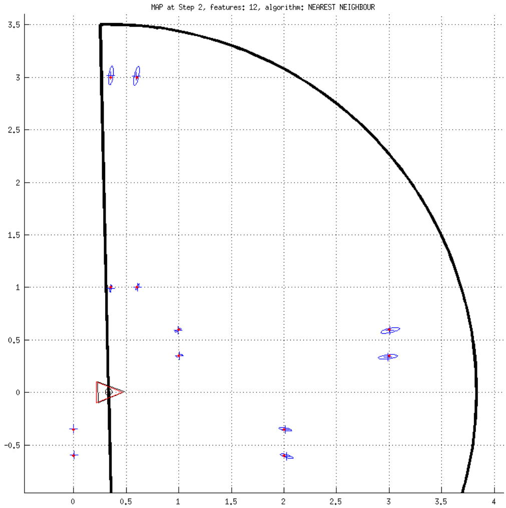
$[\mathbf{x}_k^B, \mathbf{P}_k^B] = \text{add\_new\_features}(\mathbf{x}_k^B, \mathbf{P}_k^B, \mathbf{z}_k, \mathbf{R}_k, \mathcal{H}_k)$

end for

Outline

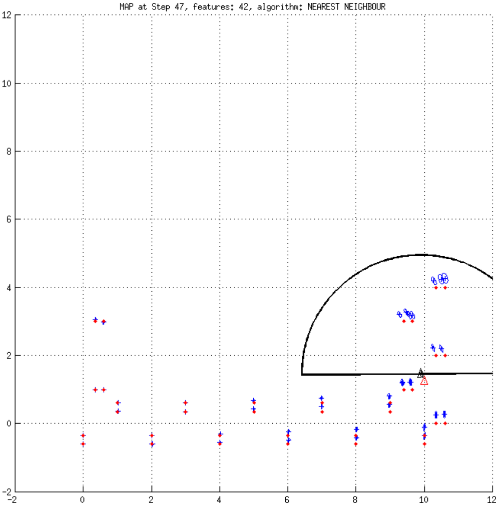
Some iterations - 1

Second step



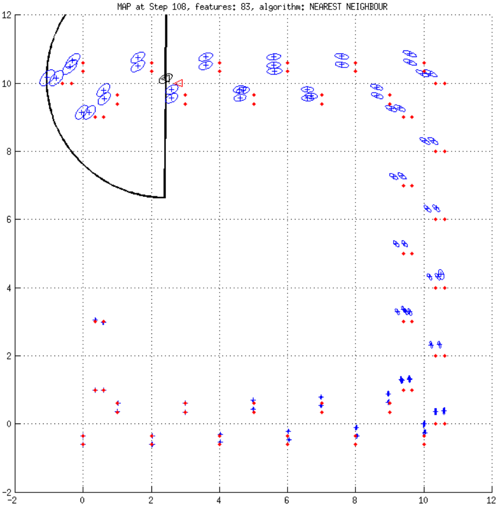
Some iterations - 2

47^{th} STEP



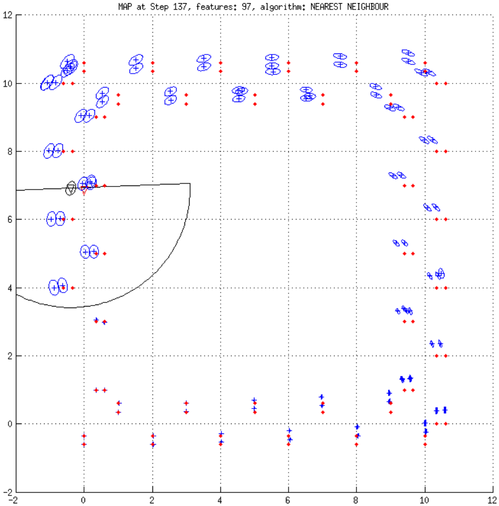
Some iterations - 3

<u>108^{th} STEP</u>



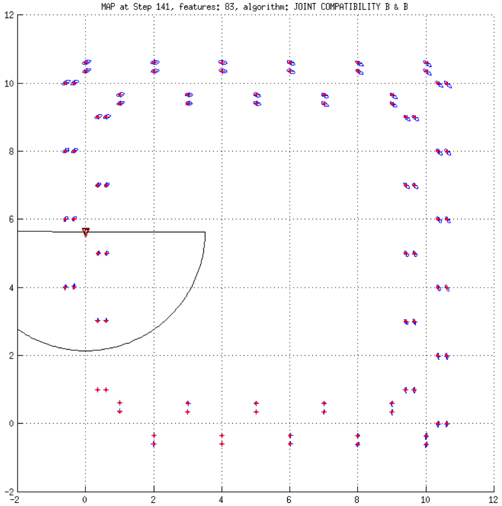
Some iterations - 4

137^{th} STEP

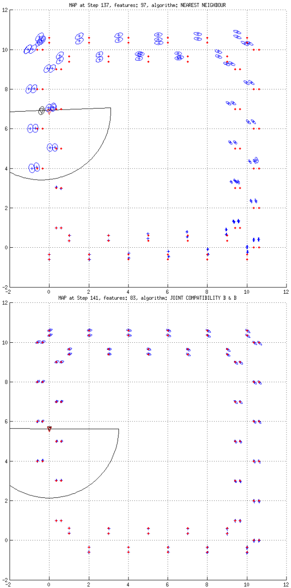


Some iterations - 5

141th step - After a Loop Closure



The role of *loop closure*



Uncertainty

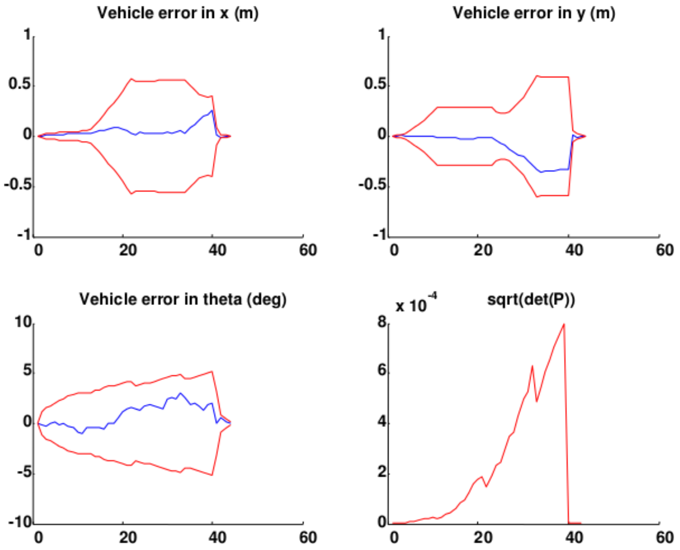
- Grows continuously also in SLAM
- The *loop closure* reduces uncertainties of
  - the current robot pose
  - the map landmark
- The *loop closure* propagates corrections

Loop closure

- A landmark i that is already in the map is perceived "after a while"
- Its uncertainty is lower than current, it gives a good information for localization

The role of *loop closure* - 2

UNCERTAINTY ON ROBOT POSE



Outline

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

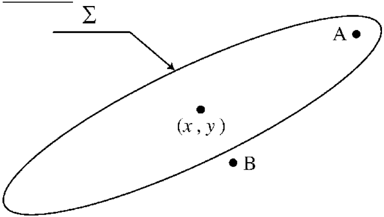
- Given a set of measurements $\{\mathbf{z}_i\}, i = 1 : m$
- Given a set of *measurements prediction* $\{\mathbf{h}_j\}, j = 1 : w$
- We have to select correspondences c_{ij}
- Or to add measurements as new landmarks

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

Mahalanobis Distance

Given



- Given A, B coordinates
- Distance to (x, y)
- Suppose to know covariance Σ
- i.e., $\sim \mathcal{N}(\mu = [x, y], \Sigma)$

Euclidean Distance

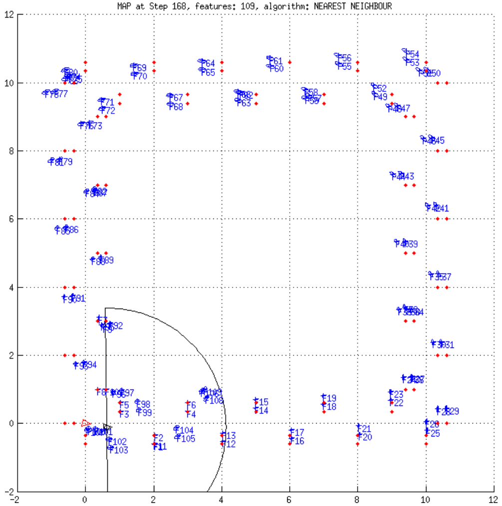
- A is closest to x, y
- B is far

Mahalanobis Distance

- $D^2 = (x - \mu)^T \Sigma^{-1} (x - \mu)$
- Squared distance weighted for the inverse of covariance
- $D^2(A) < D^2(B)$,
  A is inside the covariance ellipse
- It is a scaled and rotated distance
- Same probability $=$ same distance
- D^2 is distributed as a $\chi^2(n)$

Data association errors

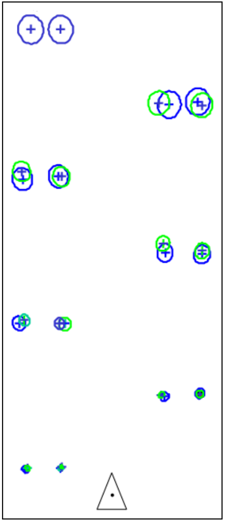
Wrong associations ⇒ bad results



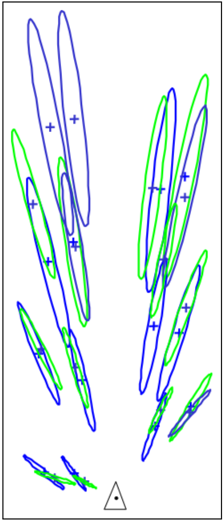
[Matlab: RUN1]

When data association is difficult - 1

Low sensor error

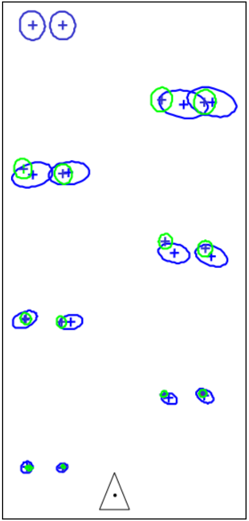


High sensor error

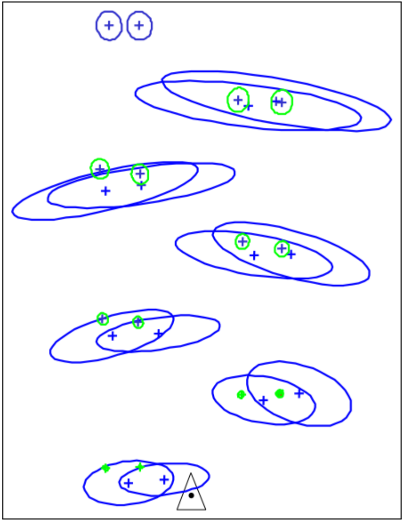


When data association is difficult - 2

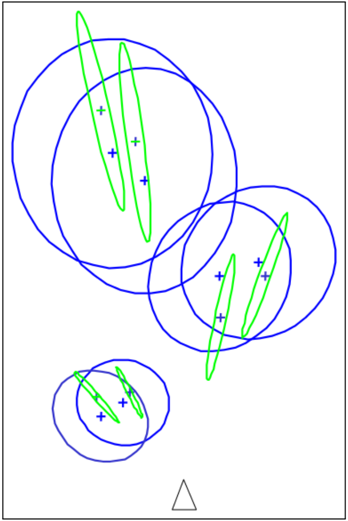
Low odometry error



High odometry error



When data association is difficult - 3



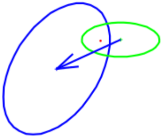
Low landmark density

High landmark density

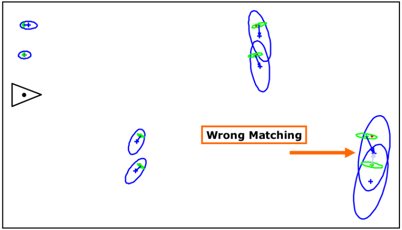
Nearest Neighbour Data association pitfall

Mahalanobis Distance

- Evaluate *Individual Compatibility*

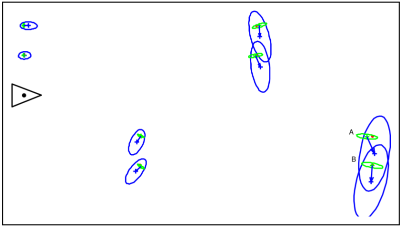


- This could result in wrong associations



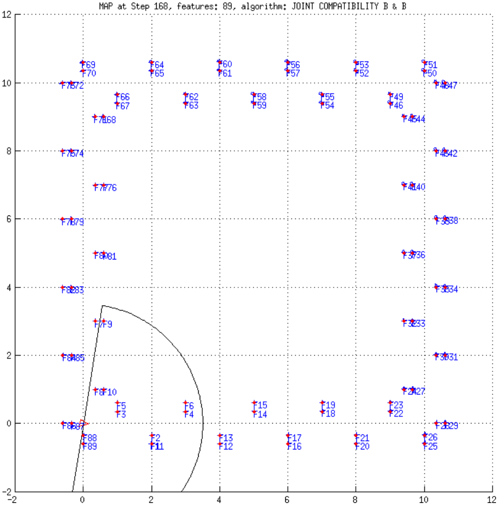
Joint Compatibility

- 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)

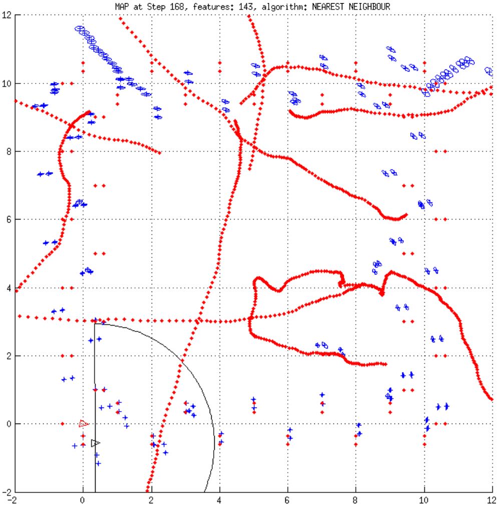
Using JCBB data association



[Matlab: RUN2]

Non-static environment - 1

People walking in the cloister

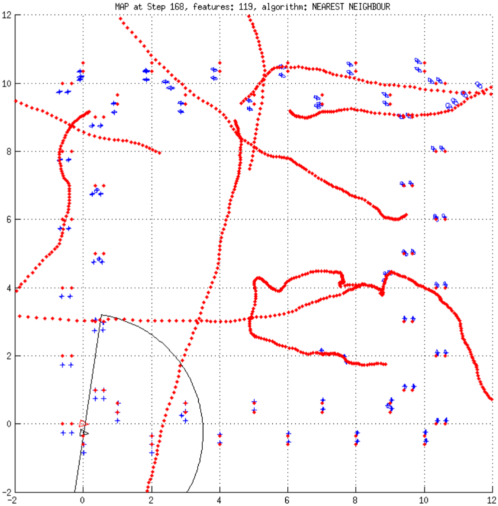


- Using Nearest Neighbour

[Matlab: RUN3]

Non-static environment - 2

People walking in the cloister

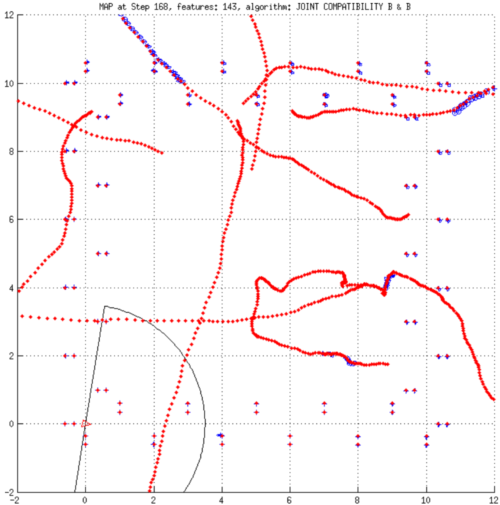


- Using Nearest Neighbour
- Delete landmarks that have a measurement prediction but are not matched for a while

[Matlab: RUN4]

Non-static environment - 3

People walking in the cloister

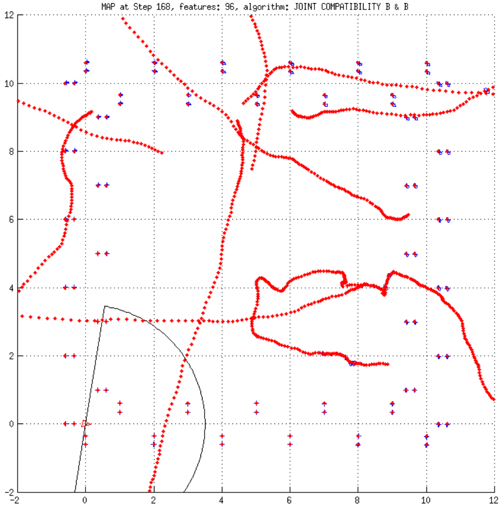


- Using JCBB

[Matlab: RUN5]

Non-static environment - 4

People walking in the cloister



- Using JCBB
- 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 Δt
- The noise "code" the (unknown) speed change
- Measurements change position and speed thanks to correlations

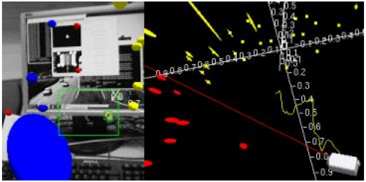
[Matlab: RUN7]

Outline

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



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

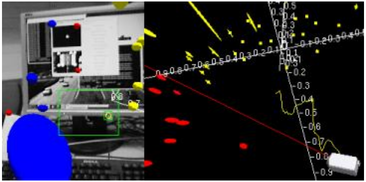
Visual SLAM

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

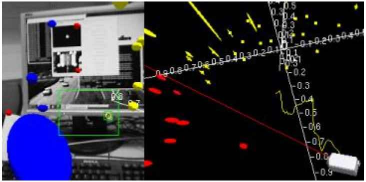


Davison *"Real-time Simultaneous Localization And Mapping with a Single Camera"*, 2003

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

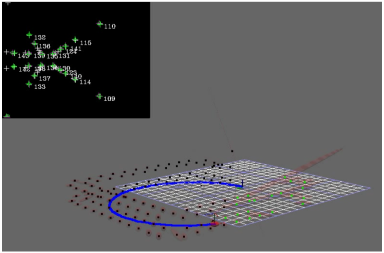
Davison *"Real-time Simultaneous Localization And Mapping with a Single Camera"*, 2003

Landmarks & Features

Landmarks

- Elements of the map
- They code a 3D point

  notice: we consider 3D environment

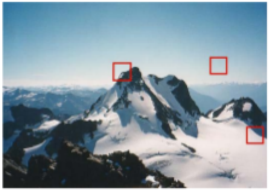


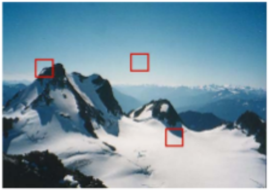
Features

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

Good Features to Track

Characteristics of good features





- Repeatability
  The same feature can be found in several images
  despite geometric and photometric transformations
- Saliency
  Each feature has a distinctive description
- Compactness and efficiency
  Many fewer features than image pixels
- Locality
  A feature occupies a relatively small area of the image
  robust to clutter and occlusion

Detector, descriptor, matching, tracking

DETECTOR:

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

DESCRIPTOR:

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) ⇒ **Patch around the corner**

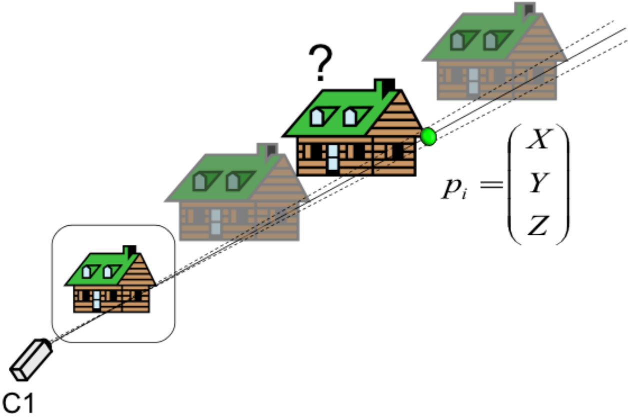
FEATURE MATCHING:

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

FEATURE TRACKING:

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

Monocular SLAM key problem - 1

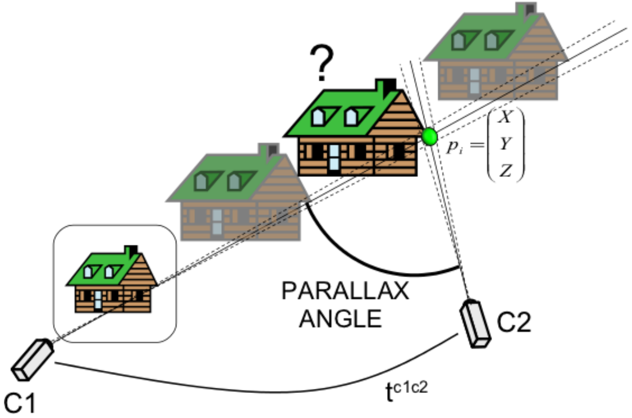


Camera is a bearing-only sensor

Depth is unknown from a single image

Depth can be estimated with triangulation after camera motion

Monocular SLAM key problem - 2



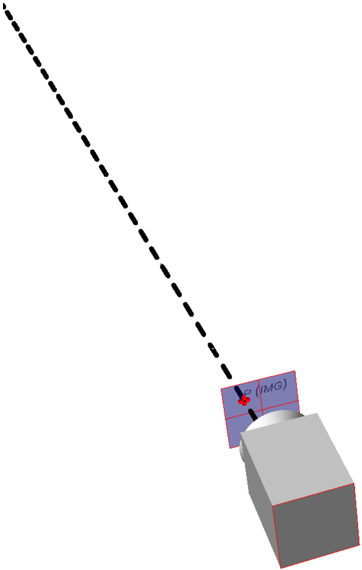
Depth can be estimated with triangulation after camera motion

Parallax angle cover a key role

Monocular SLAM key problem - 3

Feature depth

- Unknown at initialization
- Uniform distribution from 0 to ∞



Monocular SLAM key problem - 3

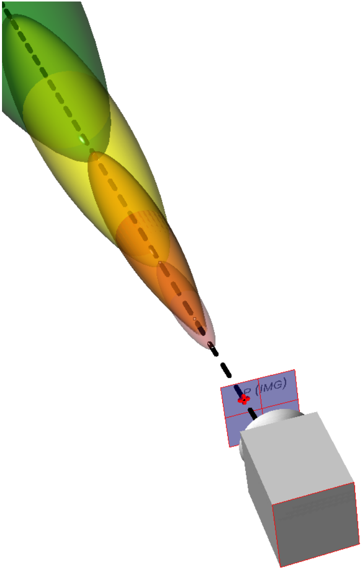
Feature depth

- Unknown at initialization
- Uniform distribution from 0 to ∞

Solution 1: Delayed initialization

For each feature

- Use a set of 3D hypotesis on view ray



Monocular SLAM key problem - 3

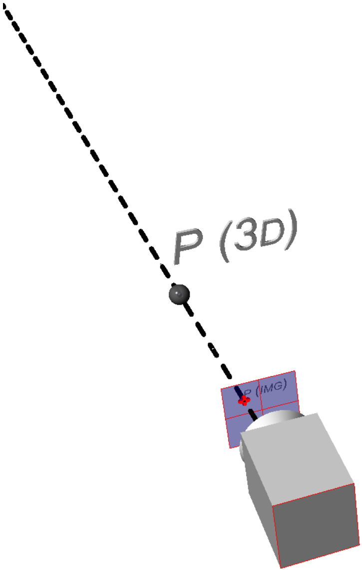
Feature depth

- Unknown at initialization
- Uniform distribution from 0 to ∞

Solution 1: Delayed initialization

For each feature

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



Monocular SLAM key problem - 3

Feature depth

- Unknown at initialization
- Uniform distribution from 0 to ∞

Solution 1: Delayed initialization

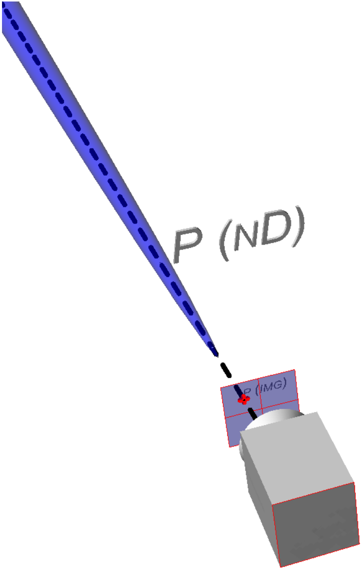
For each feature

- Use a set of 3D hypotesis 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

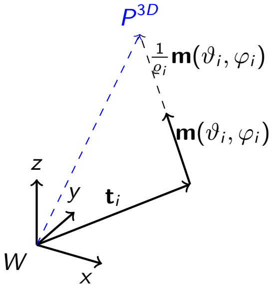
Davison *"Real-time Simultaneous Localization And Mapping with a Single Camera"*, 2003

Which parametrization?

UID
Unified Inverse Depth

$$\mathbf{y}_i^{UID} = \begin{bmatrix} \mathbf{t}_i^T & \vartheta_i & \varphi_i & \varrho_i \end{bmatrix}^T$$

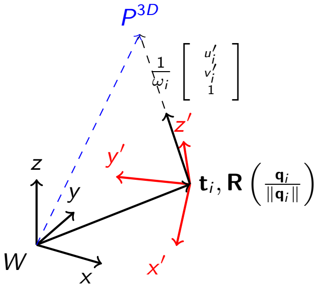
$$\mathbf{P}^{3D} = \mathbf{t}_i + \frac{1}{\varrho_i}\mathbf{m}(\vartheta_i, \varphi_i)$$



FHP
Framed Homogeneous Point

$$\mathbf{y}_i^{FHP} = \begin{bmatrix} \mathbf{t}_i^T & \mathbf{q}_i^T & u_i' & v_i' & \omega_i \end{bmatrix}^T$$

$$\mathbf{P}^{3D} = \mathbf{t}_i + \frac{1}{\omega_i}\,\mathbf{R}\left(\frac{\mathbf{q}_i}{\|\mathbf{q}_i\|}\right) \cdot \begin{bmatrix} u_i' & v_i' & 1 \end{bmatrix}^T$$



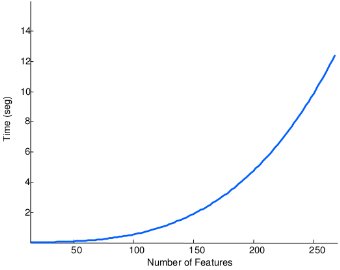
Ceriani et al. *"On Feature Parameterization for EKF-Based Monocular SLAM"*, 2010

Montiel, Civera, Davison *"Unified inverse depth parametrization for monocular slam"*, 2006

Large Scale SLAM Issues

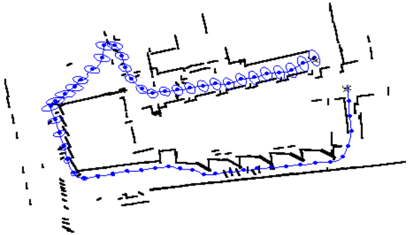
Computational cost

Grows with # features



Consistency

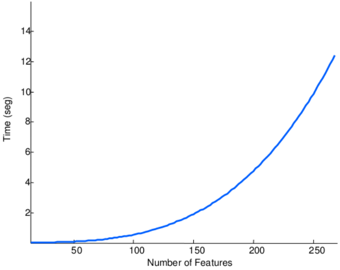
Due to linearizations of EKF



Large Scale SLAM Issues

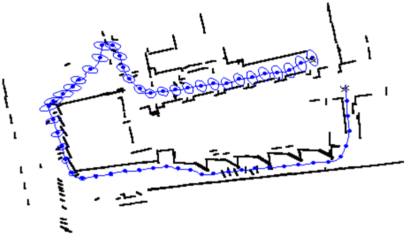
Computational cost

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

Example in a Real Environment - Monocular Vision

videos/mono.flv

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

The map is represented by points location

Theoretically reconstruction is up to a single scale factor

Practically there is a scale drift

Example in a Real Environment - Stereo Vision

videos/stereo.flv

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

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 - 1

- 360-degree field of view

1. Camera
2. Lower Mirror
3. Aperture
4. Glass Housing
5. Cover and Upper Mirror (hidden)

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 Algorithm
3. SLAM example
4. Correspondences
5. Visual SLAM
6. 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

Only EKF-SLAM?

Laser Range Scanner based SLAM

2D SLAM

3D SLAM

videos/slam2dlaser.webm

from $http://www.youtube.com/watch?v=flfNOXHxBKY$

videos/slam3dlaser.webm

from $http://www.youtube.com/watch?v=QQeJ1xd_sOU$

other sensors: Microsoft Kinect, etc...

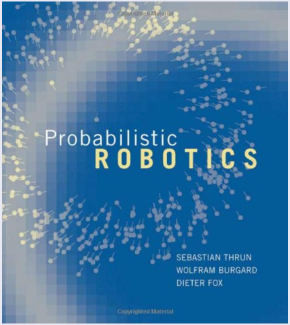
References

Reference

"Probabilistic Robotics"

(Intelligent Robotics and Autonomous Agents series)

The MIT Press (2005)



Chapters 2, 3, 7.