Probabilistic Robotics Course

Localization with Kalman Filters
Bearing only
[Example Application]

Omar Salem
salem@diag.uniroma1.it

Department of Computer, Control and Management Engineering
Sapienza University of Rome

EKF: recap

- Estimate the current state distribution from
  - Previous state distribution
  - Sequence of observations $z_{0:t}$
  - Sequence of controls $u_{0:t-1}$
  - Transition model
  - Observation model

$$
\begin{aligned}
\mu_{t|t-1} &= \boxed{\mathbf{f}(\boxed{\mu_{t-1|t-1}}, \mathbf{u}_{t-1})} \\
\mathbf{\Sigma}_{t|t-1} &= \mathbf{A}_t \boxed{\mathbf{\Sigma}_{t-1|t-1}} \mathbf{A}_t^T + \mathbf{B}_t \mathbf{\Sigma}_u \mathbf{B}_t^T
\end{aligned}
$$

$$
\begin{aligned}
\mu_z &= \boxed{\mathbf{h}(\mu_{t|t-1})} \\
\boxed{\mu_{t|t}} &= \mu_{t|t-1} + \mathbf{K}_t \left(\mathbf{z}_t - \mu_z\right) \\
\boxed{\mathbf{\Sigma}_{t|t}} &= \left(\mathbf{I} - \mathbf{K}_t \mathbf{C}_t\right) \mathbf{\Sigma}_{t|t-1}
\end{aligned}
$$

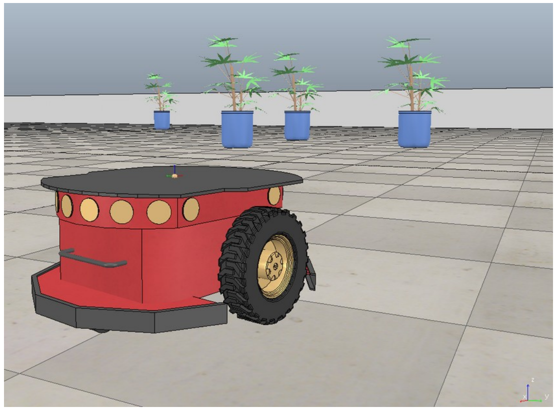
Outline

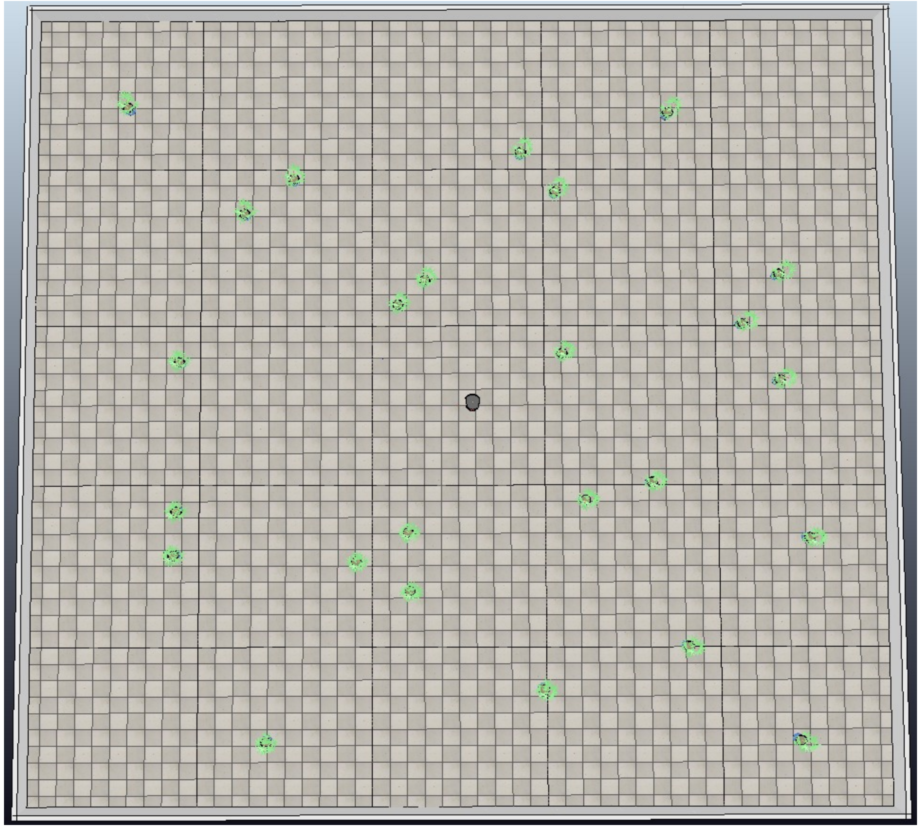
- Scenario
- Controls
- Observations
- Jacobians
- Implementation

Scenario

Orazio moves on a 2D plane

- It is controlled by translational and rotational velocities
- Senses a set of uniquely distinguishable landmarks through a "2D bearing only landmark sensors"
- The location of the landmarks in the world is known

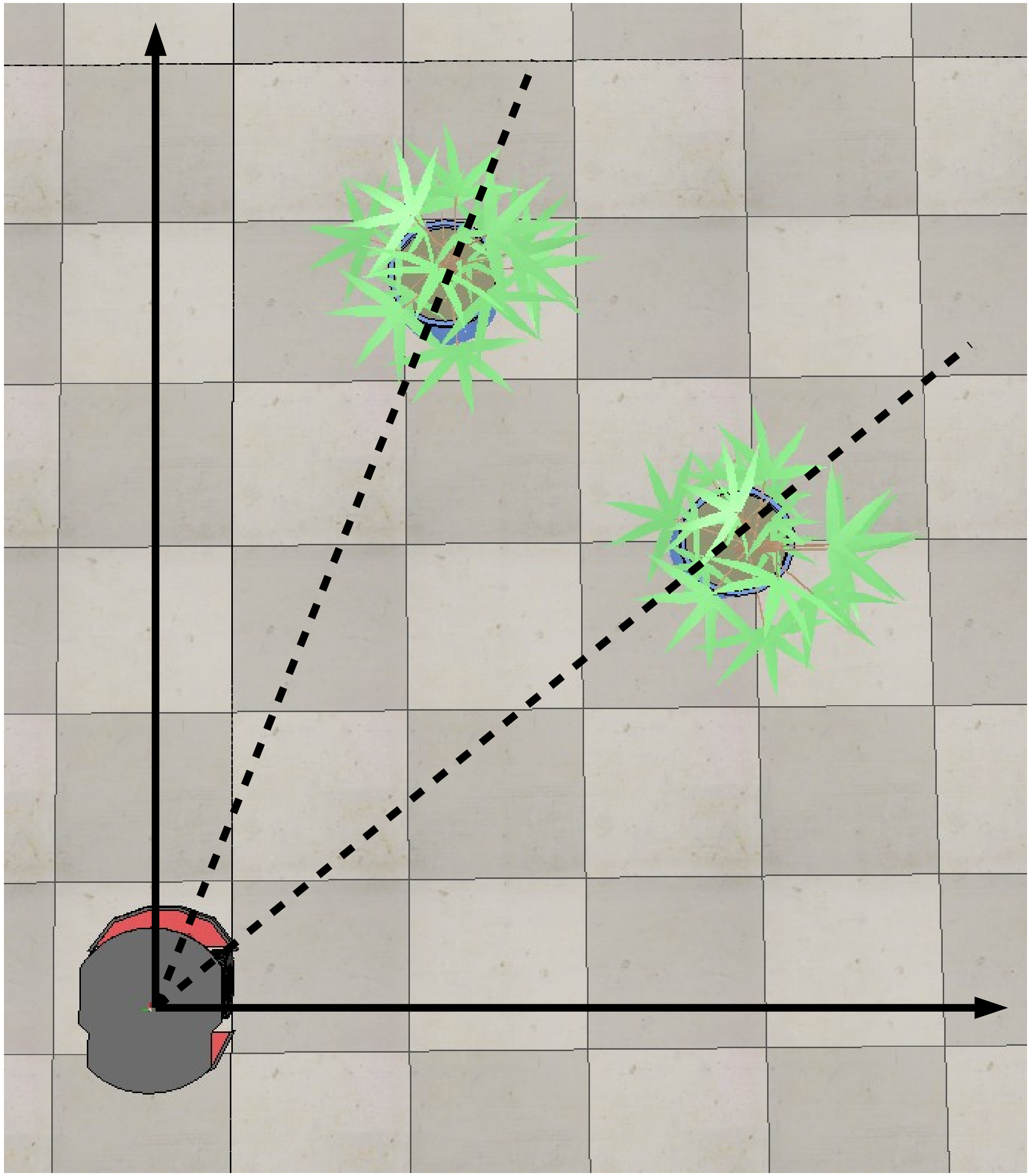




Bearing only sensor

Orazio moves on a 2D plane

- It is controlled by translational and rotational velocities
- Senses a set of uniquely distinguishable landmarks through a “2D bearing only landmark sensors”
- The location of the landmarks in the world is known



Approaching the problem

We want to develop a EKF based algorithm to track the position of Orazio as it moves

The inputs of our algorithms will be

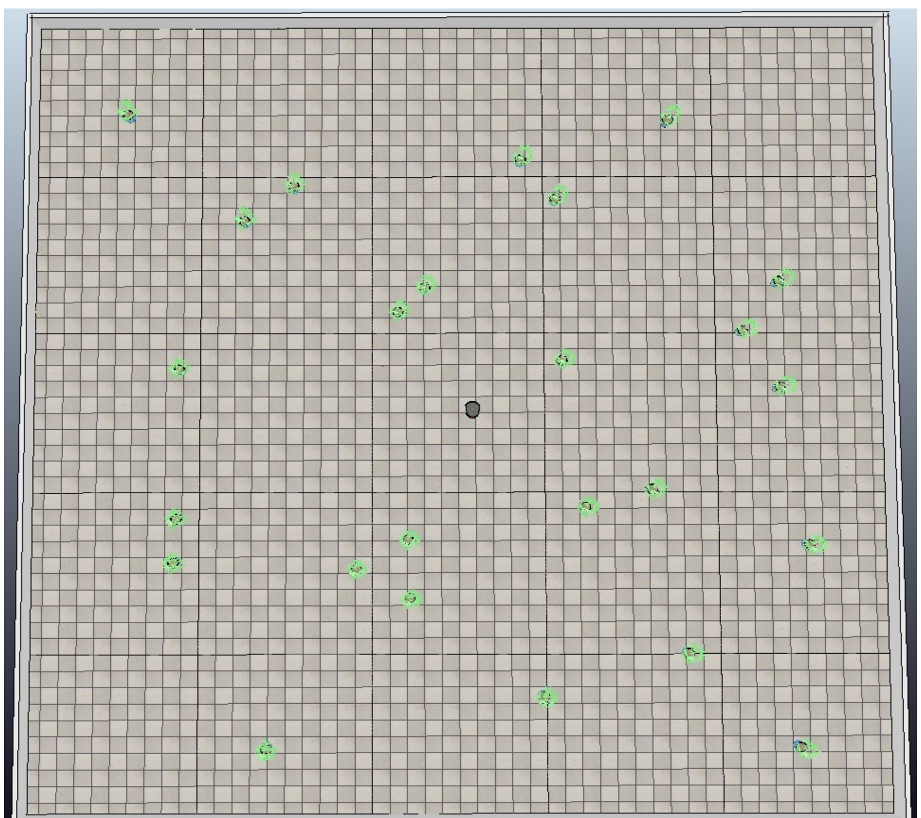
- velocity measurements
- landmark measurements

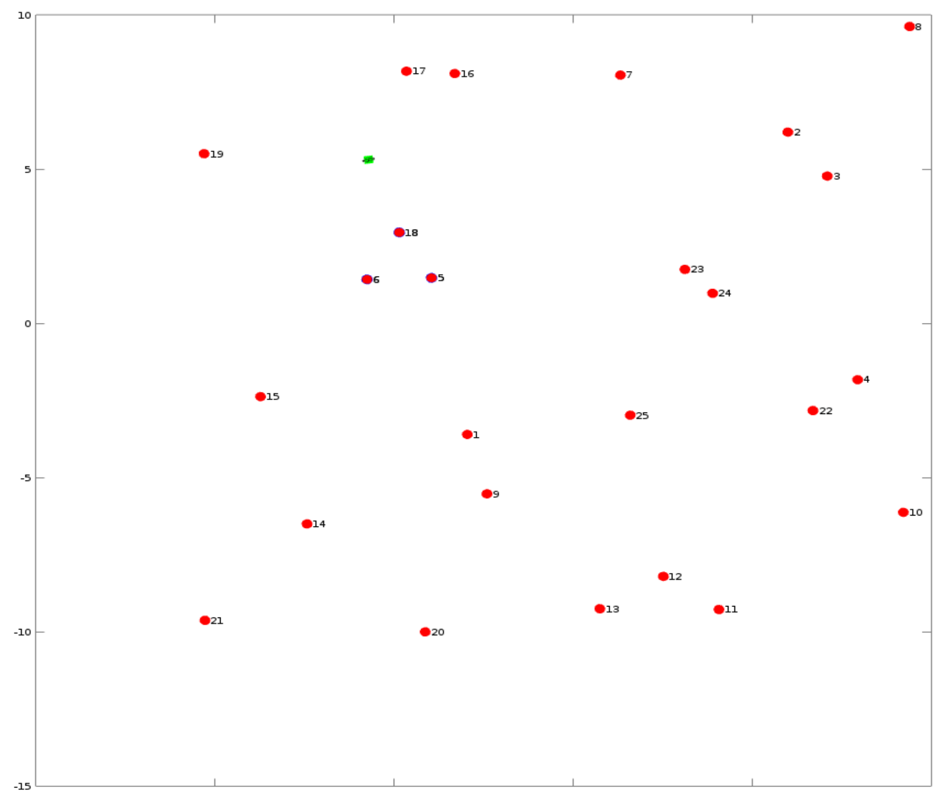
The prior knowledge about the map is represented by the location of each landmark in the world

Prior

The map is represented as a set of landmark coordinates

$$\mathbf{l}^{[i]} = \begin{pmatrix} x^{[i]} \\ y^{[i]} \end{pmatrix} \in \Re^2$$





Domains

Define

- state space

$$\mathbf{X}_t = [\mathbf{R}_t | \mathbf{t}_t] \in SE(2)$$

Instead of considering rotational and translational velocities, we consider the integrated motion in the interval as input

This leads to a lighter notation

- space of controls (inputs)

$$\mathbf{u}_t = \begin{pmatrix} \Delta_t v_t \\ \Delta_t \omega_t \end{pmatrix} = \begin{pmatrix} u_t^1 \\ u_t^2 \end{pmatrix} \in \Re^2$$

- space of observations (measurements)

$$z_t \in SO(2)$$

Domains

Find a Euclidean parameterization of non-Euclidean spaces

poses are not Euclidean, we map them to 3D vectors

- state space

$$\mathbf{X}_t = [\mathbf{R}_t | \mathbf{t}_t] \in SE(2) \Rightarrow \mathbf{x}_t = \begin{pmatrix} x_t \\ y_t \\ \theta_t \end{pmatrix} \in \Re^3$$

- space of controls (inputs)

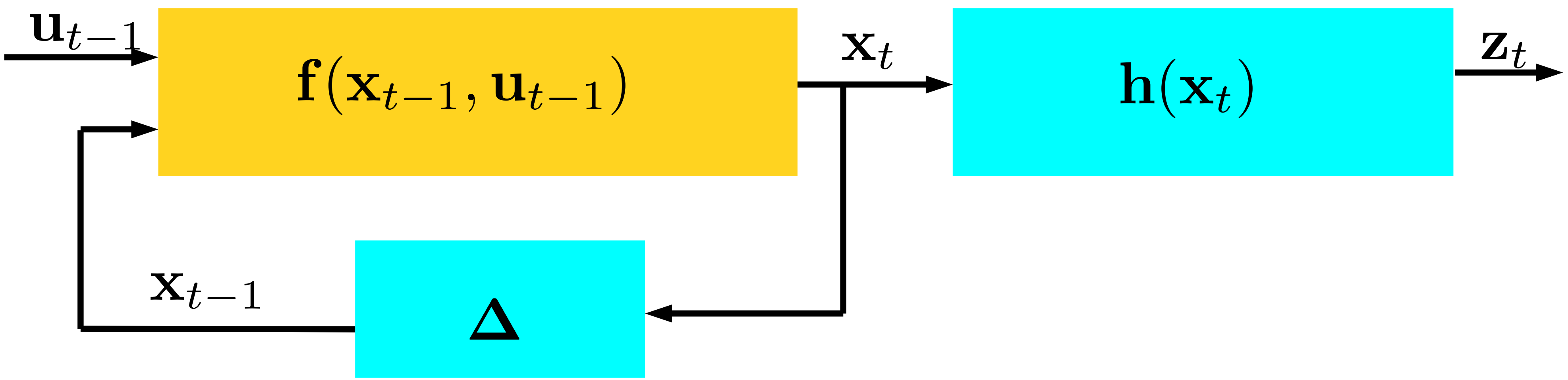
$$\mathbf{u}_t = \begin{pmatrix} u_t^1 \\ u_t^2 \end{pmatrix} \in \Re^2$$

measurement and control, in this problem are easily mapped to an Euclidean Domain

- space of observations (measurements)

$$z_t \in SO(2) \Rightarrow \Re$$

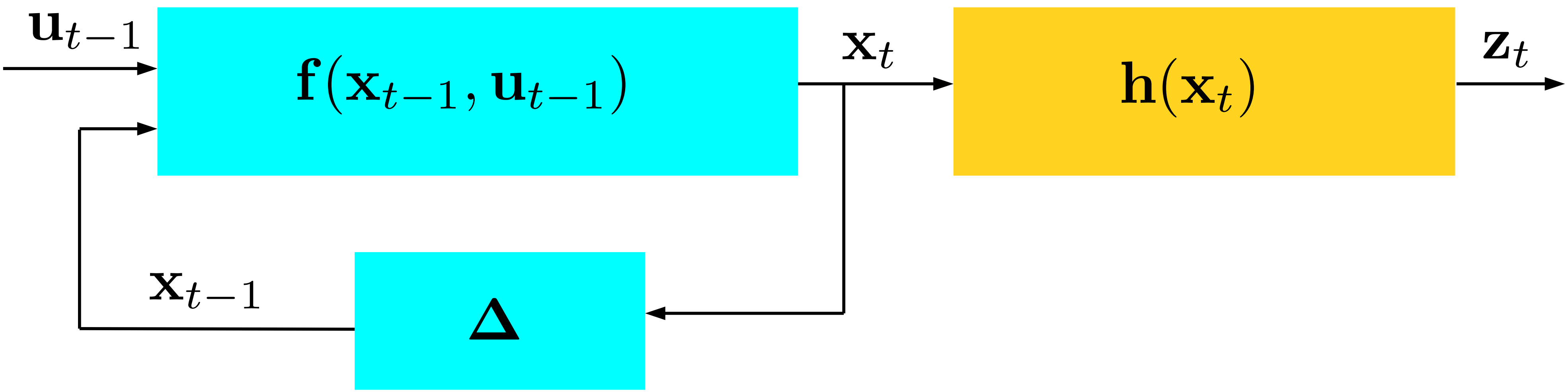
Transition Function



- Consider constant velocity in interval $[t_{t-1}, t_t]$
- State x_t is obtained by Euler integration

$$\mathbf{x}_t = \mathbf{f}(\mathbf{x}_{t-1}, \mathbf{u}_{t-1}) = \left(\begin{array}{c} x_{t-1} + u_{t-1}^1 \cdot \cos(\theta_{t-1}) \\ y_{t-1} + u_{t-1}^1 \cdot \sin(\theta_{t-1}) \\ \theta_{t-1} + u_{t-1}^2 \end{array} \right)$$

Measurement Function



We have [i] measurement functions, one per landmark

$$
\begin{aligned}
z_t^{[i]} &= \mathbf{h}^{[i]}(\mathbf{x}_t) \\
&= \operatorname{atan}(\hat{y}_t^{[i]} / \hat{x}_t^{[i]})
\end{aligned}
$$

relative bearing of the Ith landmark w.r.t the robot at time t

$$
\hat{\mathbf{p}}_t^{[i]} = \begin{pmatrix} \hat{x}_t^{[i]} \\ \hat{y}_t^{[i]} \end{pmatrix} = \mathbf{R}_t^T(\mathbf{l}^{[i]} - \mathbf{t}_t) = \begin{pmatrix} \cos\theta_t(x^{[i]} - x_t) + \sin\theta_t(y^{[i]} - y_t) \\ -\sin\theta_t(x^{[i]} - x_t) + \cos\theta_t(y^{[i]} - y_t) \end{pmatrix}
$$

$$
\mathbf{R}_t = \begin{pmatrix} \cos\theta_t & -\sin\theta_t \\ \sin\theta_t & \cos\theta_t \end{pmatrix}
$$

rotation matrix of theta

Measurement Function

At each point in time, our robot will sense only a subset of K landmarks in the map

The measurement is thus consisting of a stack of measurements

$$\mathbf{z}_t = \begin{pmatrix} z^{[i_1]} \\ z^{[i_2]} \\ \dots \\ z^{[i_K]} \end{pmatrix} = \mathbf{h}(\mathbf{x}_t) = \begin{pmatrix} \mathbf{h}^{[i_1]}(\mathbf{x}_t) \\ \mathbf{h}^{[i_2]}(\mathbf{x}_t) \\ \dots \\ \mathbf{h}^{[i_K]}(\mathbf{x}_t) \end{pmatrix}$$

index of the landmark generating the measurement

Control Noise

We assume the velocity measurements are effected by a Gaussian noise resulting from the sum of two aspects

- a constant noise
- a velocity dependent term whose standard deviation grows with the speed
- translational and rotational noise are assumed independent

$$\mathbf{n}_{u,t} \sim \mathcal{N}\left(\mathbf{n}_{u,t}; \mathbf{0}, \begin{pmatrix} (u_t^{[1]})^2 + \sigma_v^2 & 0 \\ 0 & (u_t^{[2]})^2 + \sigma_\omega^2 \end{pmatrix}\right)$$

Measurement Noise

We assume it is zero mean, constant

$$\mathbf{n}_z \sim \mathcal{N}\left(\mathbf{n}_z; \mathbf{0}, \left(\quad \sigma_z^2 \quad \right)\right)$$

Jacobians!

At each time step our system will need to compute the derivatives of transition and measurement functions

$$f(x,u) = \begin{pmatrix} x_{t-1} + u^1_{t-1}\cos(\theta_{t-1}) \\ y_{t-1} + u^1_{t-1}\sin(\theta_{t-1}) \\ \theta_{t-1} + u^2_{t-1} \end{pmatrix}$$

$$\mathbf{A}_t = \frac{\partial \mathbf{f}(\cdot)}{\partial \mathbf{x}} = \begin{pmatrix} 1 & 0 & -u^{[1]}_{t-1}\sin(\theta_{t-1}) \\ 0 & 1 & u^{[1]}_{t-1}\cos(\theta_{t-1}) \\ 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{B}_t = \frac{\partial \mathbf{f}(\cdot)}{\partial \mathbf{u}} = \begin{pmatrix} \cos(\theta_{t-1}) & 0 \\ \sin(\theta_{t-1}) & 0 \\ 0 & 1 \end{pmatrix}$$

Jacobians (cont)

We will have *K* measurement functions, one for each landmark

$$\mathbf{h}^{[i]}(\mathbf{x}_t) \quad = \quad \mathrm{atan}(\hat{y}_t^{[i]}/\hat{x}_t^{[i]})$$

$$\hat{\mathbf{p}}_t^{[i]} = \begin{pmatrix} \hat{x}_t^{[i]} \\ \hat{y}_t^{[i]} \end{pmatrix} = \mathbf{R}_t^T \underbrace{(\mathbf{l}^{[i]} - \mathbf{t}_t)}_{\boldsymbol{\Delta}\mathbf{t}^{[i]}} = \begin{pmatrix} \cos\theta_t(x^{[i]} - x_t) + \sin\theta_t(y^{[i]} - y_t) \\ -\sin\theta_t(x^{[i]} - x_t) + \cos\theta_t(y^{[i]} - y_t) \end{pmatrix}$$

Use the multivariate chain rule.

Multivariate Chain Rule

Let $\mathbf{f}(\mathbf{y})$ and $\mathbf{g}(\mathbf{x})$ two multivariate functions of compatible dimensions

$$\left.\frac{\partial \mathbf{f}(\mathbf{g}(\mathbf{x}))}{\partial \mathbf{x}}\right|_{\mathbf{x}=\breve{\mathbf{x}}} = \left.\frac{\partial \mathbf{f}(\mathbf{u}))}{\partial \mathbf{u}}\right|_{\mathbf{u}=\mathbf{g}(\breve{\mathbf{x}})} \left.\frac{\partial \mathbf{g}(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\breve{\mathbf{x}}}$$

$$\left.\frac{\partial \mathbf{f}(\mathbf{g}(\mathbf{x}))}{\partial \mathbf{x}}\right|_{\mathbf{x}=\breve{\mathbf{x}}} = \breve{\mathbf{J}}_f \breve{\mathbf{J}}_g$$

Exactly as in case of scalar functions.

Jacobians (cont)

We will have *n* measurement functions, one for each landmark

$$\mathbf{h}^{[i]}(\mathbf{x}_t) \quad = \quad \operatorname{atan}(\hat{y}_t^{[i]} / \hat{x}_t^{[i]})$$

$$\hat{\mathbf{p}}_t^{[i]} = \begin{pmatrix} \hat{x}_t^{[i]} \\ \hat{y}_t^{[i]} \end{pmatrix} = \mathbf{R}_t^T \underbrace{(\mathbf{l}^{[i]} - \mathbf{t}_t)}_{\mathbf{\Delta t}^{[i]}} = \begin{pmatrix} \cos\theta_t(x^{[i]} - x_t) + \sin\theta_t(y^{[i]} - y_t) \\ -\sin\theta_t(x^{[i]} - x_t) + \cos\theta_t(y^{[i]} - y_t) \end{pmatrix}$$

using multivariate chain rule

$$\frac{\partial \mathbf{h}^{[i]}(\cdot)}{\partial \mathbf{x}} = \left.\frac{\partial \operatorname{atan}(\hat{\mathbf{p}}_t^{[i]})}{\partial \hat{\mathbf{p}}_t^{[i]}}\right|_{\hat{\mathbf{p}}_t^{[i]} = \mathbf{g}(\breve{\mathbf{x}})} \left.\frac{\partial \mathbf{g}_i(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x} = \breve{\mathbf{x}}_i}$$

Already computed in point based localization

$$\frac{\partial \operatorname{atan}(\hat{\mathbf{p}}_t^{[i]})}{\partial \hat{\mathbf{p}}_t^{[i]}} = \frac{1}{1 + (\hat{y}_t^{[i]} / \hat{x}_t^{[i]})^2} \cdot \begin{pmatrix} -\frac{\hat{y}_t^{[i]}}{\hat{x}_t^{[i]2}} & \frac{1}{\hat{x}_t^{[i]}} \end{pmatrix}$$

1x2

$$\frac{\partial \mathbf{h}^{[i]}(\cdot)}{\partial \mathbf{x}} = \mathbf{C}_t^{[i]} = \frac{1}{\hat{x}_t^{[i]2} + \hat{y}_t^{[i]2}} \cdot \begin{pmatrix} -\hat{y}_t^{[i]} & \hat{x}_t^{[i]} \end{pmatrix} \cdot \begin{pmatrix} -\mathbf{R}_t^T & \frac{\partial \mathbf{R}_t^T}{\partial \theta_t} \mathbf{\Delta t}^{[i]} \end{pmatrix}$$

Jacobians (cont)

The total Jacobian of the measurement will be the stack of the individual measurement functions

$$\mathbf{C}_t = \frac{\partial \mathbf{h}}{\partial \mathbf{x}} = \begin{pmatrix} \frac{\partial \mathbf{h}^{[i_1]}}{\partial \mathbf{x}} \\ \frac{\partial \mathbf{h}^{[i_2]}}{\partial \mathbf{x}} \\ \vdots \\ \frac{\partial \mathbf{h}^{[i_K]}}{\partial \mathbf{x}} \end{pmatrix} = \begin{pmatrix} \mathbf{C}_t^{[i_1]} \\ \mathbf{C}_t^{[i_2]} \\ \vdots \\ \mathbf{C}_t^{[i_K]} \end{pmatrix}$$

Hands on!

g2o Wrapper

Load your Vrep acquired dataset

```
[land, pose, transition, obs] = loadG2o('my_dataset.g2o');
```

It returns 4 Struct-Array(Landmark, Poses, Transitions, Observations), *i.e.* :

```
land =

  1x25 struct array containing the fields:

    id
    x_pose
    y_pose
```

```
pose =

  1x137 struct array containing the fields:

    id
    x
    y
    theta
```

```
transition =

  1x136 struct array containing the
fields:

    id_from
    id_to
    v
```

```
obs =

  1x136 struct array containing the fields:

    pose_id
    observation
```

EKF Localization – Bearing Only

```
% load your own dataset dataset
[landmarks, poses, transitions, observations] = loadG2o('dataset.g2o');
mu = rand(3,1)*20-10; % init mean
mu(3) = normalizeAngle(mu(3));

sigma = eye(3)*0.001; % init covariance

%simulation cycle
for i=1:length(transitions)
    % predict with transitions
    [mu, sigma] = ekf_prediction(mu, sigma, transitions(i));
    % correct with observations
    [mu, sigma] = ekf_correction(mu, sigma, landmarks, observations(i));

    plot_state(landmarks, mu, sigma, observations(i));
endfor
```

EKF Localization – Bearing Only

```
% load your own dataset dataset
[landmarks, poses, transitions, observations] = loadG2o('dataset.g2o');
mu = rand(3,1)*20-10; % init mean
mu(3) = normalizeAngle(mu(3));

sigma = eye(3)*0.001; % init covariance

%simulation cycle
for i=1:length(transitions)
    % predict with transitions TODO
    [mu, sigma] = ekf_prediction(mu, sigma, transitions(i));
    % correct with observations TODO
    [mu, sigma] = ekf_correction(mu, sigma, landmarks, observations(i));

    plot_state(landmarks, mu, sigma, observations(i));
endfor
```