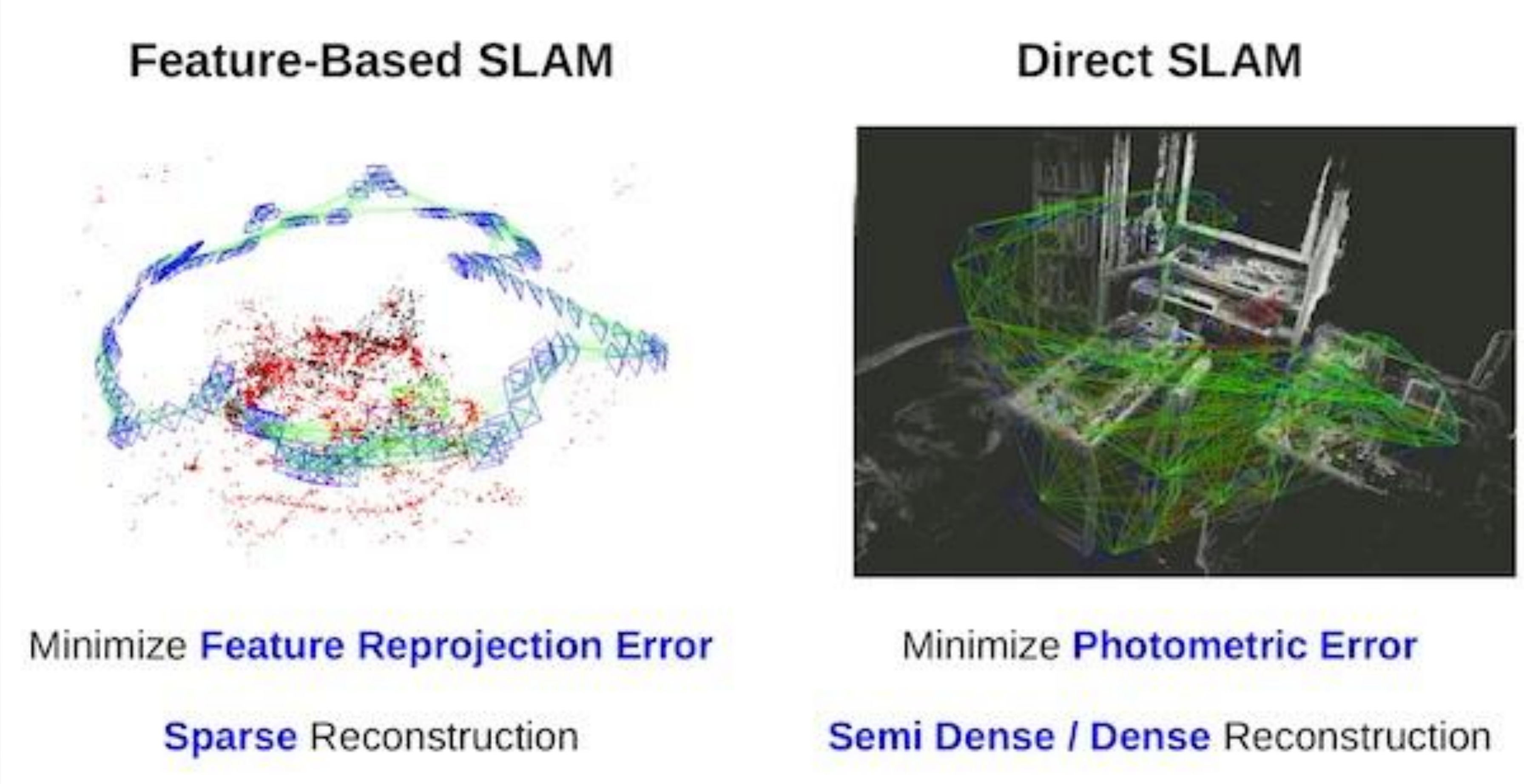
COMPUTER VISION SLAM

LECTURE 11 9.10.2019

Laura Ruotsalainen, Associate Professor
Department of Computer Science

VISUAL SLAM

- Feature based Visual SLAM
  - •Tracking features, e.g. SIFT, ORB
- Direct Visual SLAM
  - •Uses image pixels directly
- Pipeline
  - •Tracking (computing the motion and updating positions based on that) between consecutive images
  - •Mapping
  - •Global optimization (loop-closure and refining map based on that)
  - •Relocalization

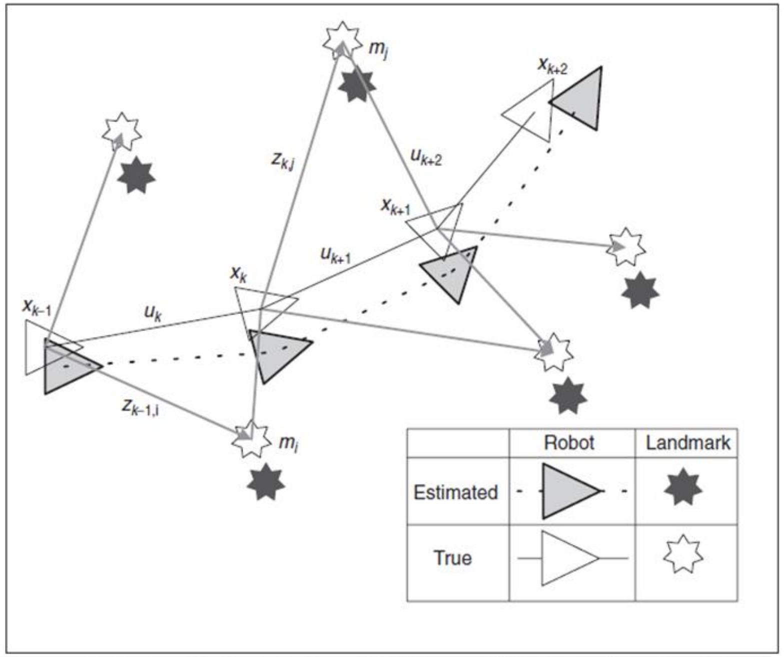


e.g. ORB-SLAM

e.g LSD-SLAM

Figure: Raúl Mur Artal

SLAM - PRINCIPLES



- The user does not know its location state vector x_k
- There is no map *M* of the area
- *m* is the location of i:th landmark,
  Z all landmark observations
- U control inputs
- In earlier developments the probabilistic form probability function $p(x_k, m| Z_{0:k}, U_{0:k}, x_0)$ was solved using EKF or Particle filtering

SLAM'S VARIANTS

- Feature based vs direct slam
  - Direct SLAM provides dense, more representative map
  - Feature-based methods are more invariant to viewpoint and illumination changes
  - Feature-based methods are less affected by dynamic objects
- Filtering vs keyframe-based
  - Monocular SLAMs are either filter based (e.g. Kalman filtering) or
  - Keyframe-based where they optimize the motion and map simultaneously

TEMPORAL STATE MODELS

Represent the 'world' as a set of random variables $\boldsymbol{X}$

$\boldsymbol{X} = \{\boldsymbol{x}, \boldsymbol{y}\}$ location on the ground plane

$\boldsymbol{X} = \{\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}\}$ position in the 3D world

$\boldsymbol{X} = \{\boldsymbol{x}, \dot{\boldsymbol{x}}\}$ position and velocity

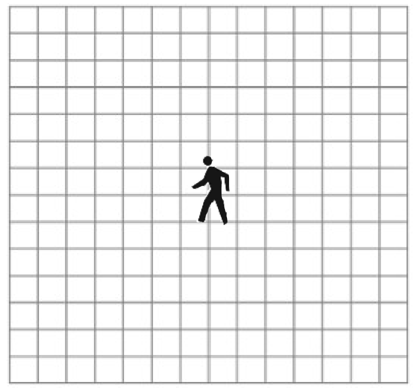
$\boldsymbol{X} = \{\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{f}_1, \ldots, \boldsymbol{f}_n\}$ position, velocity and location of landmarks

Slide credit: Kris Kitani

Object tracking (localization)

$$\boldsymbol{X} = \{\boldsymbol{x}, \boldsymbol{y}\}$$

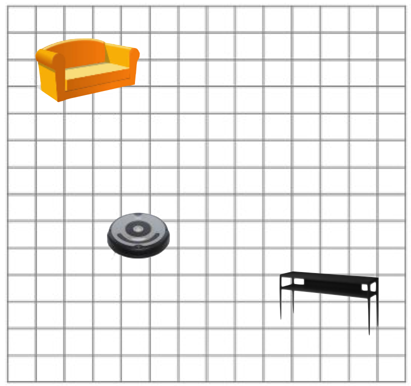
e.g., location on the ground plane



Object location and world landmarks (localization and mapping)

$$\boldsymbol{X} = \{\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{f}_1, \ldots, \boldsymbol{f}_n\}$$

e.g., position and velocity of robot
and location of landmarks



Slide credit: Kris Kitani

$$\mathbf{x}_t$$

The state of the world changes over time

Slide credit: Kris Kitani

$$\boldsymbol{X}_t$$

The state of the world changes over time

So we use a sequence of random variables:

$$\boldsymbol{X}_0, \boldsymbol{X}_1, \ldots, \boldsymbol{X}_t$$

Slide credit: Kris Kitani

$$\mathbf{X}_t$$

The state of the world changes over time

So we use a sequence of random variables:

$$\mathbf{X}_0, \mathbf{X}_1, \ldots, \mathbf{X}_t$$

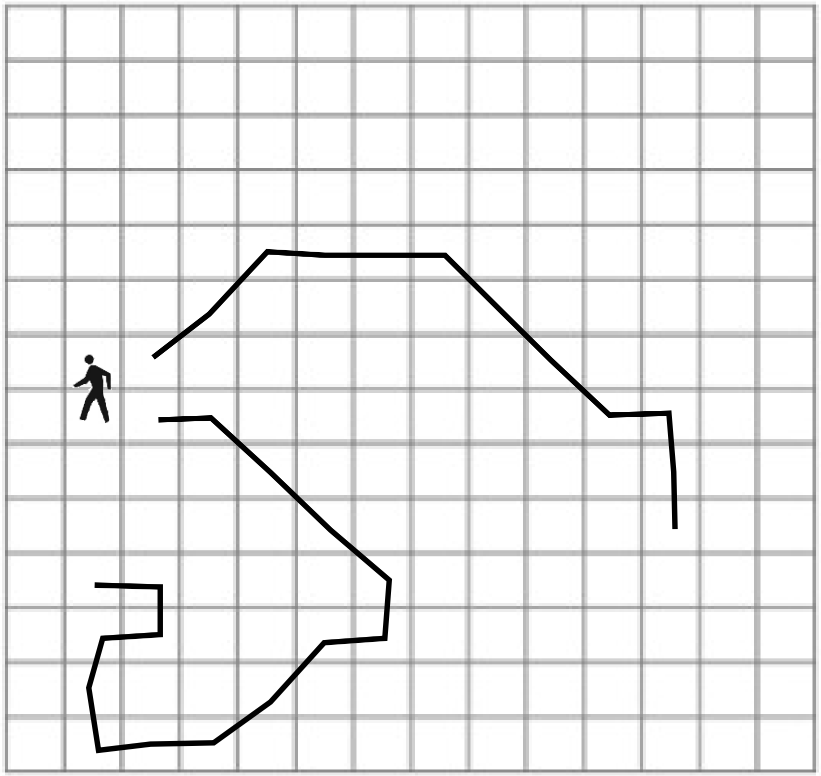
The state of the world is usually **uncertain** so we think in terms of a distribution

$$P(\mathbf{X}_0, \mathbf{X}_1, \ldots, \mathbf{X}_t)$$

How big is the space of this distribution?

Slide credit: Kris Kitani

If the state space is $\boldsymbol{X} = \{\boldsymbol{x}, \boldsymbol{y}\}$ the location on the ground plane



$$P(\boldsymbol{X}_0, \boldsymbol{X}_1, \ldots, \boldsymbol{X}_t)$$

is the probability over all possible trajectories through a room of length t+1

Slide credit: Kris Kitani

When we use a sensor (camera),
we don't have direct access to the state but noisy observations of the state

$$\boldsymbol{E}_t$$

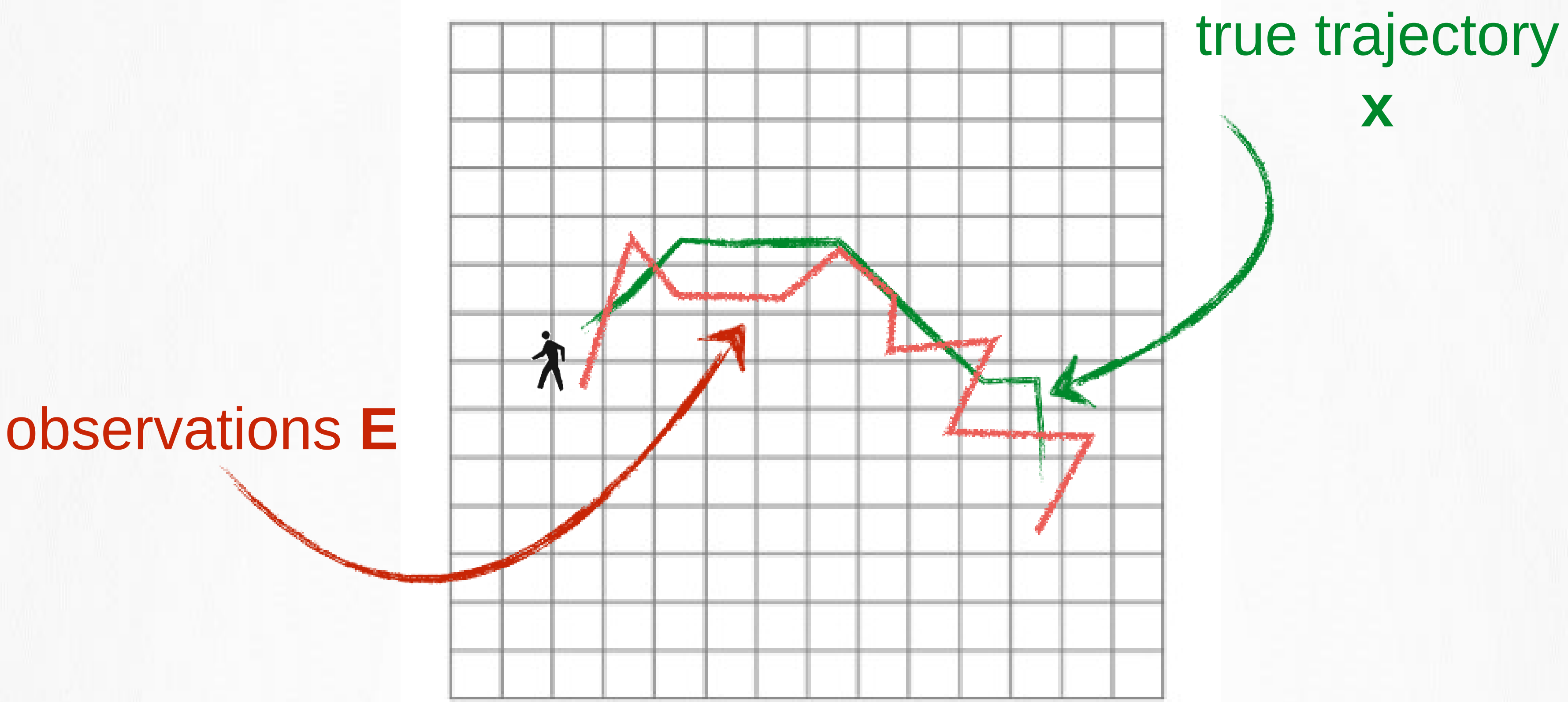
$$\boldsymbol{X}_0, \boldsymbol{X}_1, \ldots, \boldsymbol{X}_t, \boldsymbol{E}_1, \boldsymbol{E}_2, \ldots, \boldsymbol{E}_t$$

(**all** possible ways of observing **all** possible trajectories)

How big is the space of this distribution?

Slide credit: Kris Kitani

all possible ways of observing **all** possible trajectories of length t



HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

Slide credit: Kris Kitani

So we think of the world in terms of the distribution

$$P(\underbrace{\mathbf{X}_0, \mathbf{X}_1, \ldots, \mathbf{X}_t}_{\substack{\text{unobserved variables} \\ \text{(hidden state)}}}, \underbrace{\mathbf{E}_1, \mathbf{E}_2, \ldots, \mathbf{E}_t}_{\substack{\text{observed variables} \\ \text{(evidence)}}})$$

Slide credit: Kris Kitani

Reduction 1. Stationary process assumption:

'a process of change that is governed by laws that do not themselves change over time.'

$$P(\boldsymbol{E}_t|\boldsymbol{X}_t) = P_t(\boldsymbol{E}_t|\boldsymbol{X}_t)$$

the model doesn't change over time

Slide credit: Kris Kitani

Reduction 1. Stationary process assumption:

'a process of change that is governed by laws that do not themselves change over time.'

$$P(\boldsymbol{E}_t|\boldsymbol{X}_t) = P_t(\boldsymbol{E}_t|\boldsymbol{X}_t)$$

the model doesn't change over time

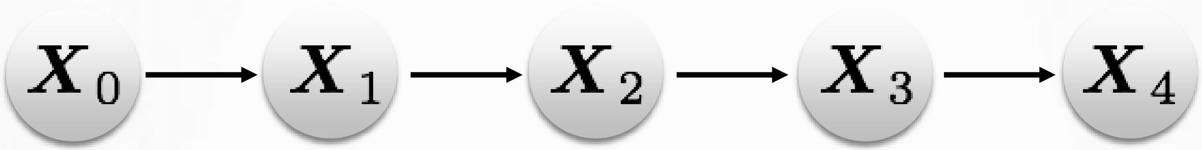
Only have to store **one** model.

Is this a reasonable assumption?

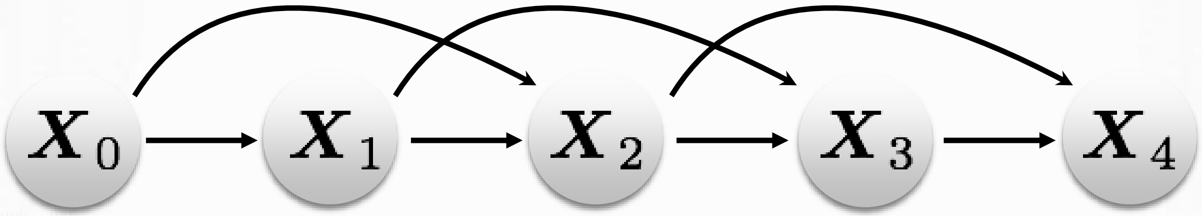
Reduction 2. Markov Assumption:

‘the current state only depends on a finite history of previous states.’

First-order Markov Model: $P(\boldsymbol{X}_t | \boldsymbol{X}_{t-1})$.



Second-order Markov Model: $P(\boldsymbol{X}_t | \boldsymbol{X}_{t-1}, \boldsymbol{X}_{t-2})$



(this relationship is called the **motion** model)

Reduction 2. Markov Assumption:

‘the current observation only depends on current state.’

The current observation is usually
most influenced by the current state

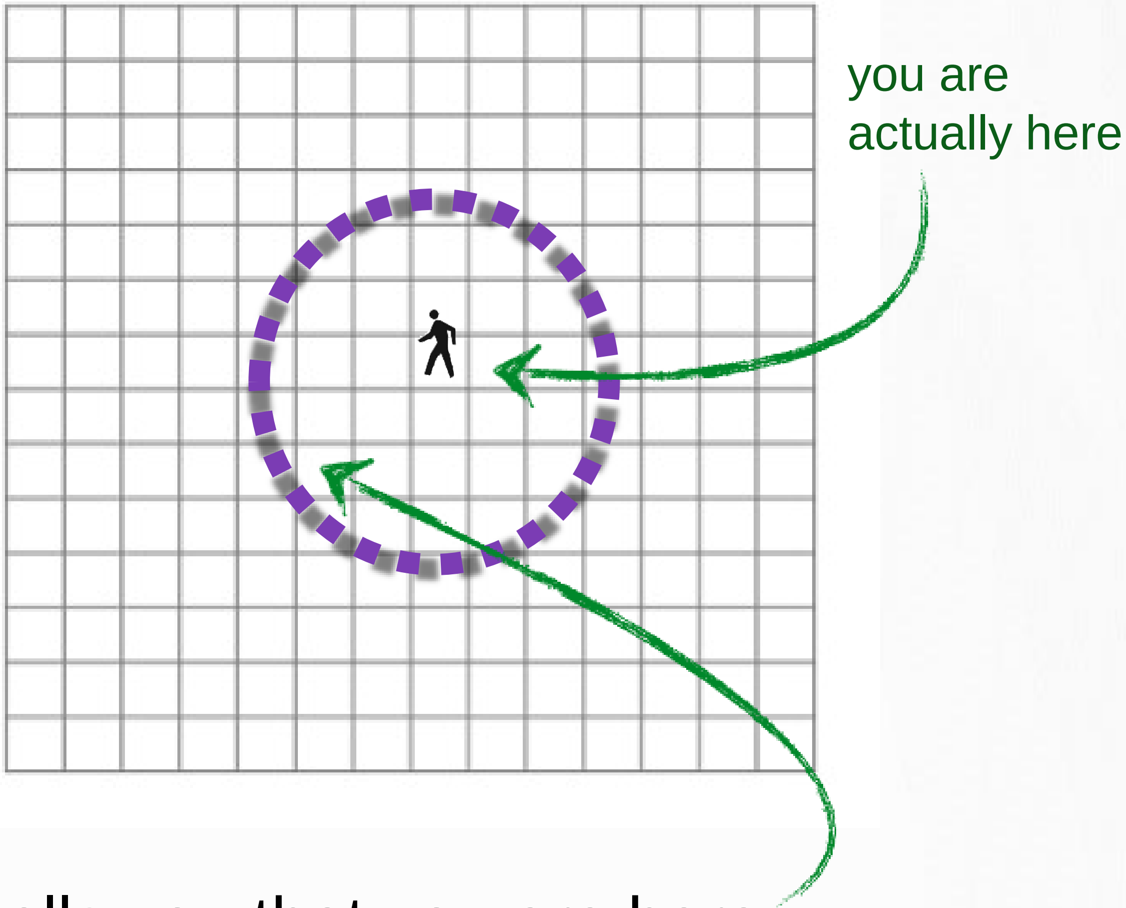
$$P(\boldsymbol{E}_t|\boldsymbol{X}_t)$$

(this relationship is called the **observation** model)

Can you think of an observation of a state?

Slide credit: Kris Kitani

For example, GPS is a noisy observation of location.



But GPS tells you that you are here

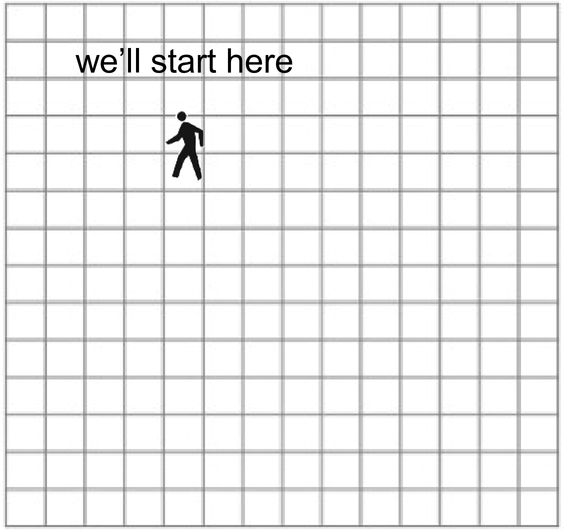
with probability $P(\boldsymbol{E}_t|\boldsymbol{X}_t)$

Slide credit: Kris Kitani

Reduction 3. Prior State Assumption:

‘we know where the process (probably) starts’

X_0



Joint Probability of a Temporal Sequence

$$P(\boldsymbol{X}_0)\prod_{t=1}^{T} P(\boldsymbol{X}_t|\boldsymbol{X}_{t-1})P(\boldsymbol{E}_t|\boldsymbol{X}_t)$$

state prior
prior

motion model
transition model

sensor model
observation model

Joint Distribution for a Dynamic Bayesian Network

specific instances of a DBN
covered in this class

Hidden Markov Model

(typically taught as discrete but not necessarily)

Kalman Filter

(Gaussian motion model, prior and observation model)

Filtering

$$P(\mathbf{X}_t | \mathbf{e}_{1:t})$$

Posterior probability over the **current** state, given all evidence up to present

Where am I now?

Slide credit: Kris Kitani

KALMAN FILTER 1/2

the Kalman Filter is “the best known filter, a simple and elegant algorithm, as an optimal recursive Bayesian estimator for a somewhat restricted class of linear Gaussian problems” (B. Ristic et al., Beyond the Kalman filter, particle filters for tracking applications, 2004)

Estimate the **state** $x \in \mathcal{R}^n$,

State transition model

process noise
p(ε)~N(0,Q)

$$\boldsymbol{x_k} = \boldsymbol{A x_{k-1}} + \boldsymbol{B u_{k-1}} + \boldsymbol{\varepsilon_{k-1}}$$

Optional control input

using **measurements** $z \in \mathcal{R}^m$

$$\boldsymbol{z_k} = \boldsymbol{H x_k} + \boldsymbol{\delta_k}$$

measurement noise
p(δ)~N(0,R)

Observation model

KALMAN FILTER 2/2

Feedback control:

time update
”predict”

measurement update
”correct”

P = estimate covariance
Q = process covariance

a priori estimate

Time update:

$$\hat{x}_k^- = A\hat{x}_{k-1}$$

$$P_k^- = AP_{k-1}A^T + Q$$

Measurement update:

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$$

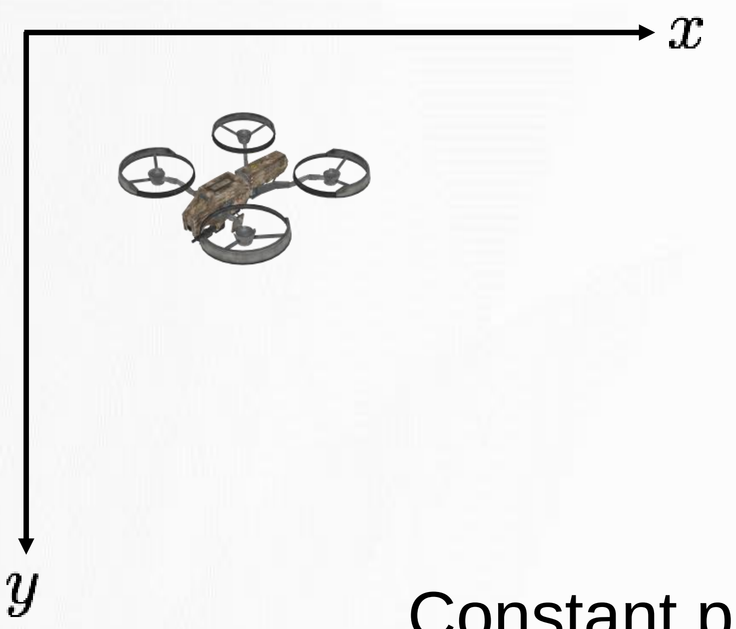
$$P_k = (I - K_k H)P_k^-$$

a posteriori estimate

R = measurement covariance

K = Kalman gain

2D EXAMPLE



state

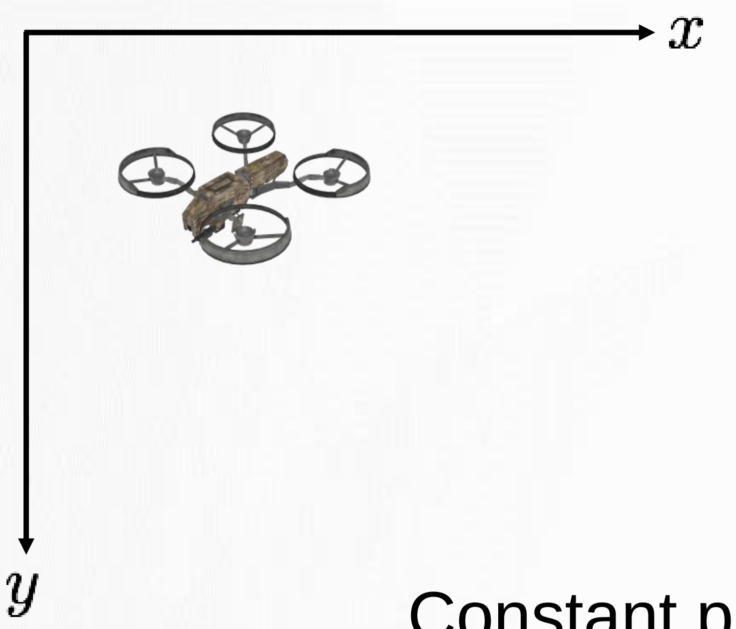
$$\boldsymbol{x} = \begin{bmatrix} x \\ y \end{bmatrix}$$

measurement

$$\boldsymbol{z} = \begin{bmatrix} x \\ y \end{bmatrix}$$

Constant position Motion Model

$$\boldsymbol{x}_t = A\boldsymbol{x}_{t-1} + B\boldsymbol{u}_t + \epsilon_t$$



state

$$\boldsymbol{x} = \begin{bmatrix} x \\ y \end{bmatrix}$$

measurement

$$\boldsymbol{z} = \begin{bmatrix} x \\ y \end{bmatrix}$$

Constant position Motion Model

$$\boldsymbol{x}_t = A\boldsymbol{x}_{t-1} + B\boldsymbol{u}_t + \epsilon_t$$

Constant position

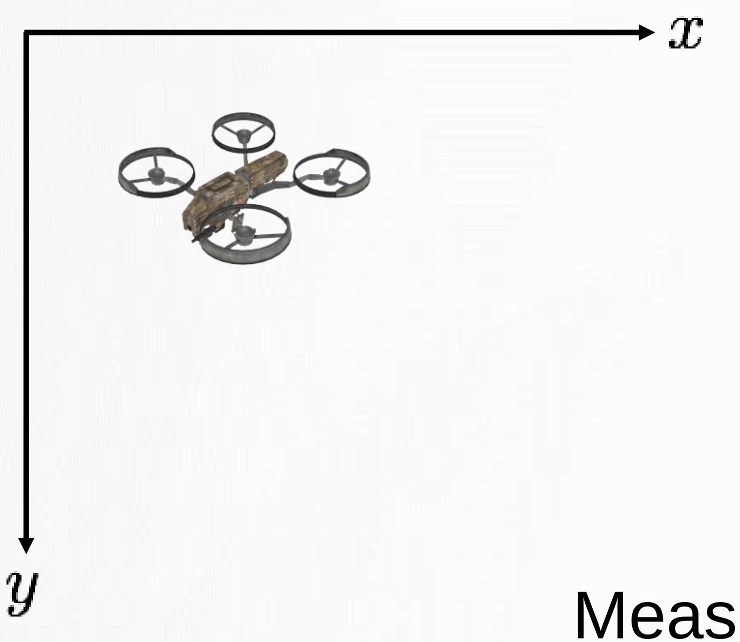
$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$B\boldsymbol{u} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

system noise

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

$$R = \begin{bmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_r^2 \end{bmatrix}$$



state

$$\boldsymbol{x} = \begin{bmatrix} x \\ y \end{bmatrix}$$

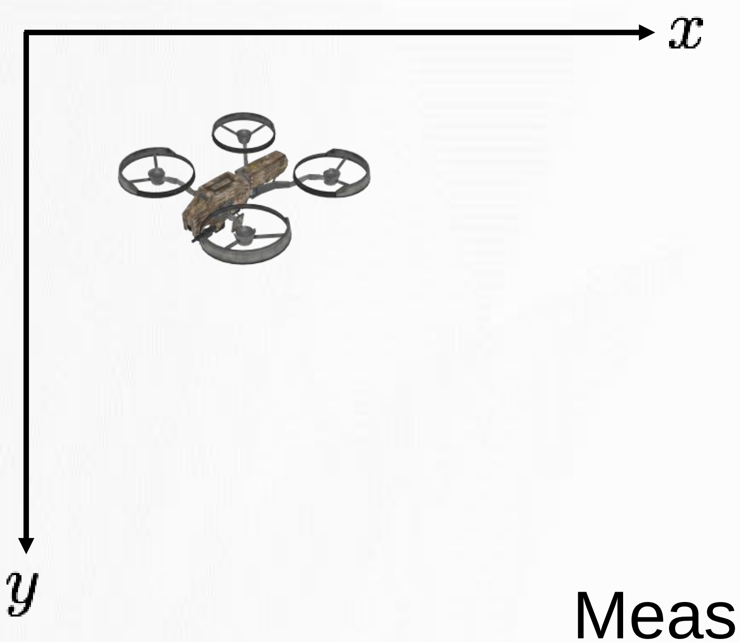
measurement

$$\boldsymbol{z} = \begin{bmatrix} x \\ y \end{bmatrix}$$

Measurement Model

$$\boldsymbol{z}_t = C_t \boldsymbol{x}_t + \delta_t$$

Slide credit: Kris Kitani



state

$$\boldsymbol{x} = \begin{bmatrix} x \\ y \end{bmatrix}$$

measurement

$$\boldsymbol{z} = \begin{bmatrix} x \\ y \end{bmatrix}$$

Measurement Model

$$\boldsymbol{z}_t = C_t \boldsymbol{x}_t + \delta_t$$

zero-mean measurement noise

$$C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \delta_t \sim \mathcal{N}(\mathbf{0}, Q) \qquad Q = \begin{bmatrix} \sigma_q^2 & 0 \\ 0 & \sigma_q^2 \end{bmatrix}$$

Slide credit: Kris Kitani

Algorithm for the 2D object tracking example

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

motion model

$$C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

observation model

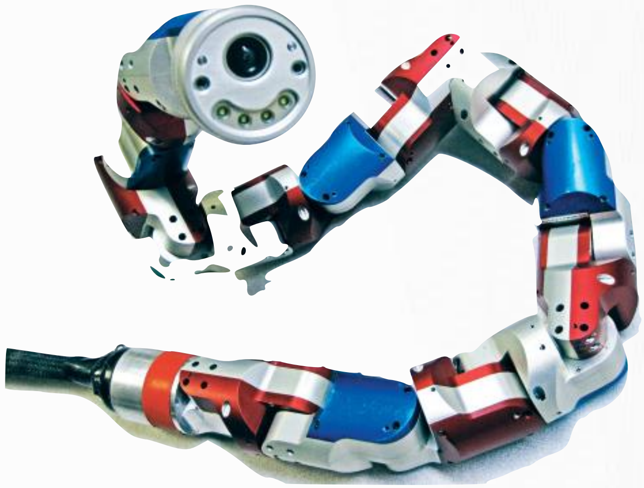
```
[x P] = KF_constPos(x,P,z)

  P = P + Q;

  K = P / (P + R);

  x   = x + K * (z - x);
  P   = (eye(size(K,1))-K) * P;
```


Slide credit: Kris Kitani

Motion model of the Kalman filter is linear

$$x_t = Ax_{t-1} + Bu_t + \epsilon_t$$

but motion is not always linear

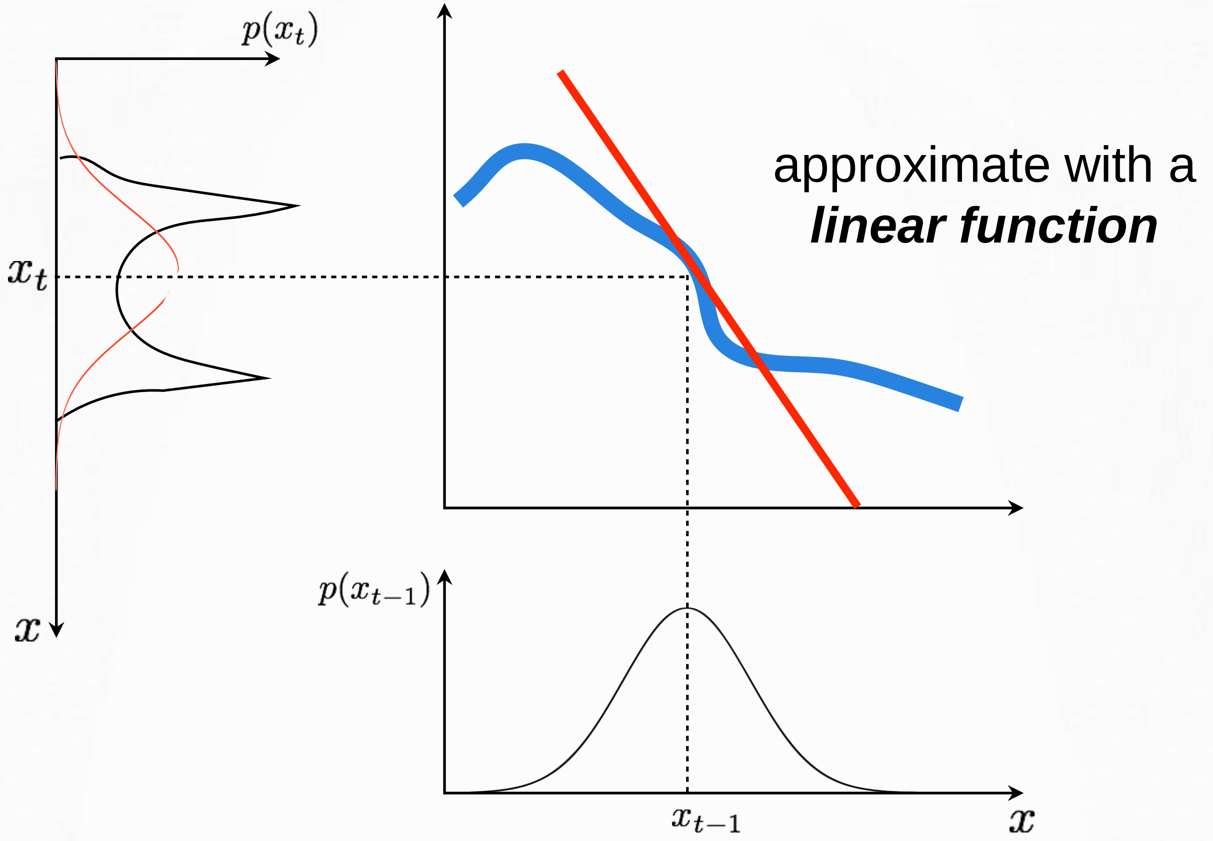


HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

Slide credit: Kris Kitani

How do you deal with non-linear models?

Output:
Gaussian



When does this trick work?

Input:
Gaussian

Slide credit: Kris Kitani

Extended Kalman Filter

- Does not assume linear Gaussian models
- Assumes Gaussian noise
- Uses local linear approximations of model to keep the efficiency of the KF framework

Kalman Filter

linear motion model

$$x_t = Ax_{t-1} + Bu_t + \epsilon_t$$

linear sensor model

$$z_t = C_t x_t + \delta_t$$

Extended Kalman Filter

non-linear motion model

$$x_t = g(x_{t-1}, u_t) + \epsilon_t$$

non-linear sensor model

$$z_t = H(x_t) + \delta_t$$

Slide credit: Kris Kitani

Motion model linearization

$$g(x_{t-1}, u_t) \approx g(\mu_{t-1}, u_t) + \frac{\partial g(\mu_{t-1}, u_t)}{\partial x_{t-1}}(x_{t-1} - \mu_{t-1})$$

Taylor series expansion

Slide credit: Kris Kitani

Motion model linearization

$$g(x_{t-1}, u_t) \approx g(\mu_{t-1}, u_t) + \frac{\partial g(\mu_{t-1}, u_t)}{\partial x_{t-1}} (x_{t-1} - \mu_{t-1})$$

$$\approx g(\mu_{t-1}, u_t) + G_t (x_{t-1} - \mu_{t-1})$$

What's this called?

Slide credit: Kris Kitani

Motion model linearization

$$
\begin{aligned}
g(x_{t-1}, u_t) &\approx g(\mu_{t-1}, u_t) + \frac{\partial g(\mu_{t-1}, u_t)}{\partial x_{t-1}} (x_{t-1} - \mu_{t-1}) \\
&\approx g(\mu_{t-1}, u_t) + \qquad G_t \qquad (x_{t-1} - \mu_{t-1})
\end{aligned}
$$

What's this called?

Jacobian Matrix

'the rate of change in x'
'slope of the function'

Slide credit: Kris Kitani

Motion model linearization

$$g(x_{t-1}, u_t) \approx g(\mu_{t-1}, u_t) + \frac{\partial g(\mu_{t-1}, u_t)}{\partial x_{t-1}}(x_{t-1} - \mu_{t-1})$$

$$\approx g(\mu_{t-1}, u_t) + \quad G_t \quad (x_{t-1} - \mu_{t-1})$$

Jacobian Matrix

'the rate of change in x'
'slope of the function'

Sensor model linearization

$$h(x_t) \approx h(\bar{\mu}_t) + \frac{\partial h(\bar{\mu}_t)}{\partial x_t}(x_{t-1} - \bar{\mu}_t)$$

$$\approx h(\bar{\mu}_t) + \quad H_t \quad (x_t - \bar{\mu}_t)$$

Slide credit: Kris Kitani

New EKF Algorithm

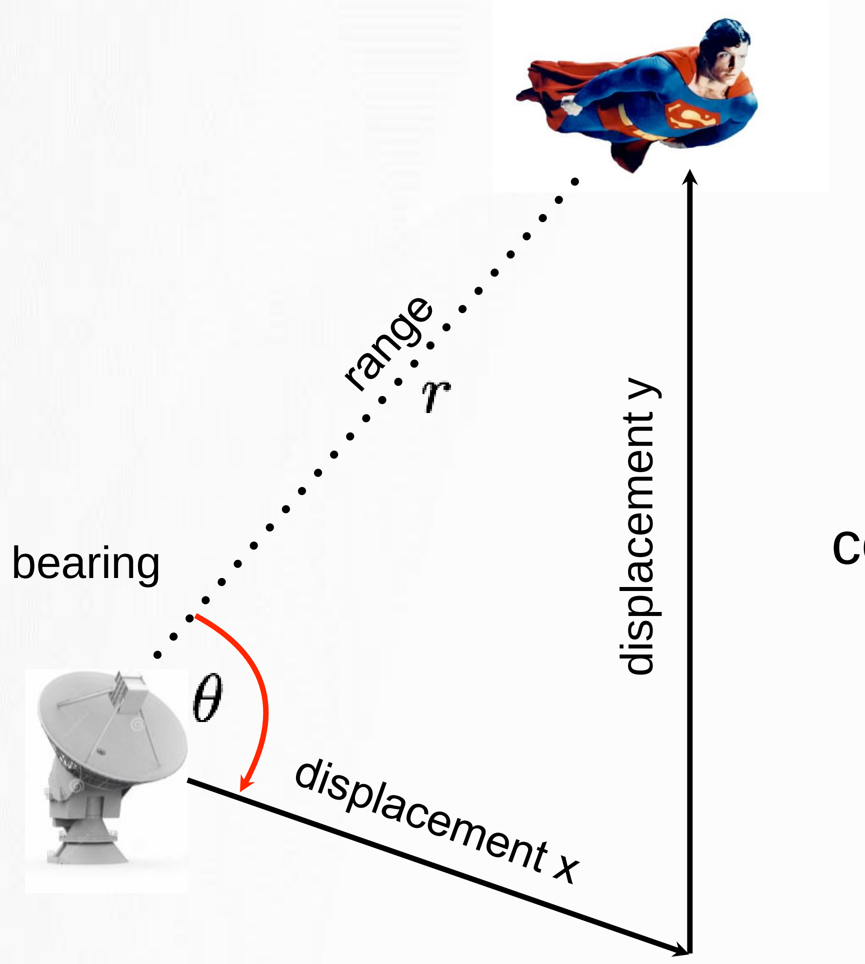
(pretty much the same)

Kalman Filter	Extended KF
$\bar{\mu}_t = A_t \mu_{t-1} + B u_t$	$\bar{\mu}_t = g(\mu_{t-1}, u_t)$
$\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^\top + R$	$\bar{\Sigma}_t = G_t \bar{\Sigma}_{t-1} G_t^\top + R$
$K_t = \bar{\Sigma}_t C_t^\top (C_t \bar{\Sigma}_t C_t^\top + Q_t)^{-1}$	$K_t = \bar{\Sigma}_t H_t^\top \left(H_t \bar{\Sigma}_t H^\top + Q\right)^{-1}$
$\mu_t = \bar{\mu}_t + K_t(z_t - C_t \bar{\mu}_t)$	$\mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t))$
$\Sigma_t = (I - K_t C_t)\bar{\Sigma}_t$	$\Sigma_t = (I - K_t H_t)\bar{\Sigma}_t$

Slide credit: Kris Kitani

2D EXAMPLE

Slide credit: Kris Kitani



state: position-velocity

$$\boldsymbol{x} = \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \end{bmatrix} \begin{array}{l} \text{position} \\ \text{velocity} \\ \text{position} \\ \text{velocity} \end{array}$$

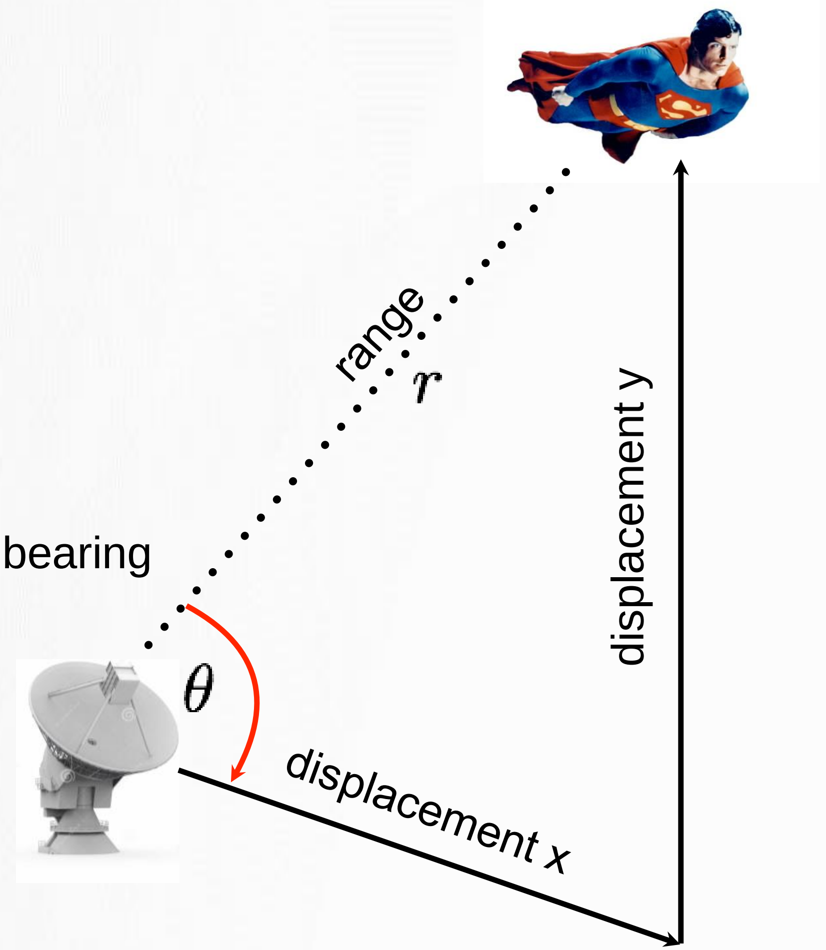
constant velocity motion model

$$A = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

with additive Gaussian noise

Slide credit: Kris Kitani

Motion model is linear but …



measurement: range-bearing

$$z = \begin{bmatrix} r \\ \theta \end{bmatrix}$$

$$= \begin{bmatrix} \sqrt{x^2 + y^2} \\ \tan^{-1}(y/x) \end{bmatrix}$$

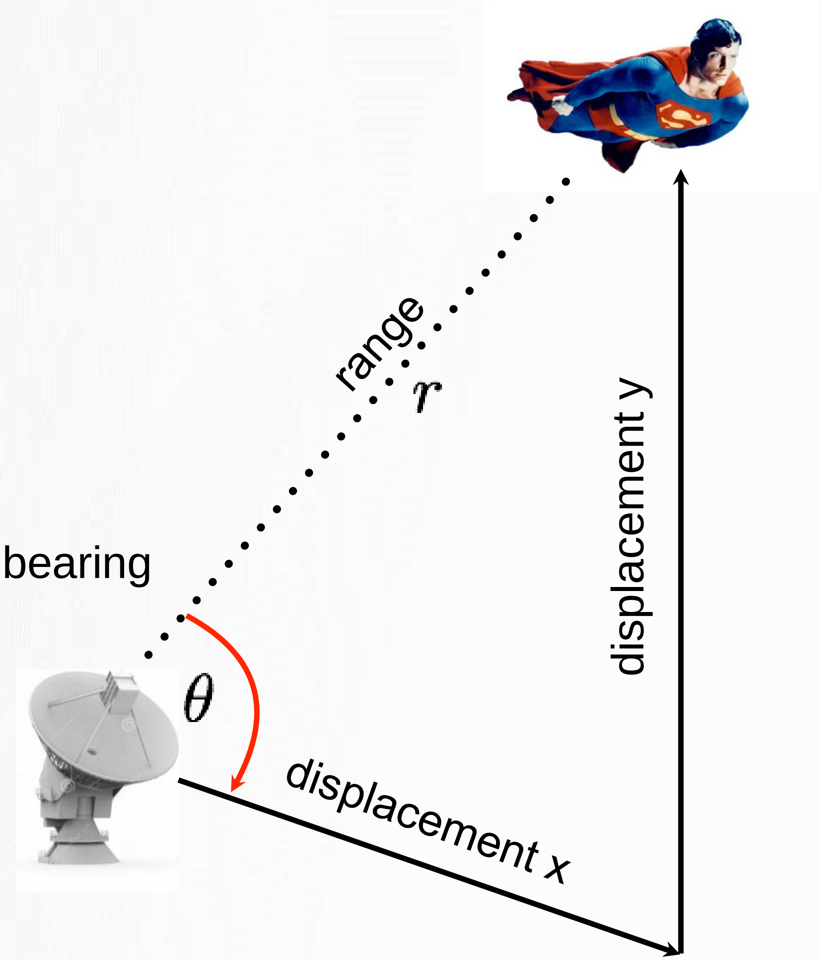
measurement model

Is the measurement model linear?

$$z = h(r, \theta)$$

with additive Gaussian noise

Slide credit: Kris Kitani



measurement: range-bearing

$$z = \begin{bmatrix} r \\ \theta \end{bmatrix}$$

$$= \begin{bmatrix} \sqrt{x^2 + y^2} \\ \tan^{-1}(y/x) \end{bmatrix}$$

measurement model

Is the measurement model linear?

$$z = h(r, \theta)$$

with additive Gaussian noise

non-linear!

What should we do?

Slide credit: Kris Kitani

linearize the observation/measurement model!

$$z = \begin{bmatrix} r \\ \theta \end{bmatrix}$$

$$= \begin{bmatrix} \sqrt{x^2 + y^2} \\ \tan^{-1}(y/x) \end{bmatrix}$$

$$H = \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{x}} = ?$$

What is the Jacobian?

$$H = \begin{bmatrix} \frac{\partial r}{\partial x} & \frac{\partial r}{\partial \dot{x}} & \frac{\partial r}{\partial y} & \frac{\partial r}{\partial \dot{y}} \\ \\ \frac{\partial \theta}{\partial x} & \frac{\partial \theta}{\partial \dot{x}} & \frac{\partial \theta}{\partial y} & \frac{\partial \theta}{\partial \dot{y}} \end{bmatrix} =$$

Slide credit: Kris Kitani

$$z = \begin{bmatrix} r \\ \theta \end{bmatrix}$$
$$= \begin{bmatrix} \sqrt{x^2 + y^2} \\ \tan^{-1}(y/x) \end{bmatrix}$$

$$H = \frac{\partial z}{\partial x} = ?$$

What is the Jacobian?

Jacobian used in the Taylor series expansion looks like …

$$H = \begin{bmatrix} \frac{\partial r}{\partial x} & \frac{\partial r}{\partial \dot{x}} & \frac{\partial r}{\partial y} & \frac{\partial r}{\partial \dot{y}} \\ \frac{\partial \theta}{\partial x} & \frac{\partial \theta}{\partial \dot{x}} & \frac{\partial \theta}{\partial y} & \frac{\partial \theta}{\partial \dot{y}} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) & 0 \\ -sin(\theta)/r & 0 & \cos(\theta)/r & 0 \end{bmatrix}$$

Slide credit: Kris Kitani

```
[x P] = EKF(x,P,z,dt)

  r   = sqrt (x(1)^2+x(3)^2);
  b   = atan2(x(3),x(1));
  y   = [r; b];

  H = [ cos(b)    0   sin(b)     0;
       -sin(b)/r 0   cos(b)/r   0];

  x = F*x;
  P = F*P*F' + Q;

  K = P*H'/(H*P*H' + R);

  x   = x + K*(z - y);
  P   = (eye(size(K,1))-K*H)*P;
```

Parameters:

```
Q = diag([0 .1 0 .1]);
R = diag([50^2 0.005^2]);
F = [ 1 dt 0 0;
      0 1 0 0;
      0 0 1 dt;
      0 0 0 1];
```

extra computation for the EKF measurement model Jacobian

Slide credit: Kris Kitani

Problems with EKFs

Taylor series expansion = poor approximation of non-linear functions
success of linearization depends on limited uncertainty and amount of local non-linearity

Computing partial derivatives is a pain

Drifts when linearization is a bad approximation

Cannot handle multi-modal (multi-hypothesis) distributions

Slide credit: Kris Kitani

Simultaneous Localization and Mapping

Given a **single camera** feed,
estimate the 3D **position of the camera** and
the 3D **positions of all landmark** points in the world

HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

Slide credit: Kris Kitani

What is the camera (robot) state?

What are the dimensions?

$$\mathbf{x}_c = \begin{bmatrix} \mathbf{r} \\ \mathbf{q} \\ \mathbf{v} \\ \omega \end{bmatrix}$$

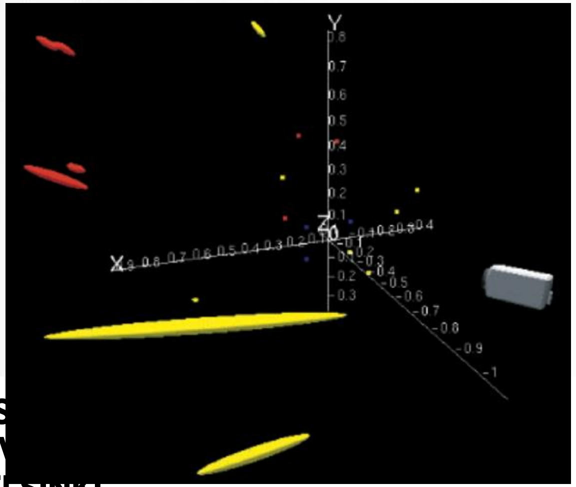
position

rotation (quaternion)

velocity

angular velocity

13 total



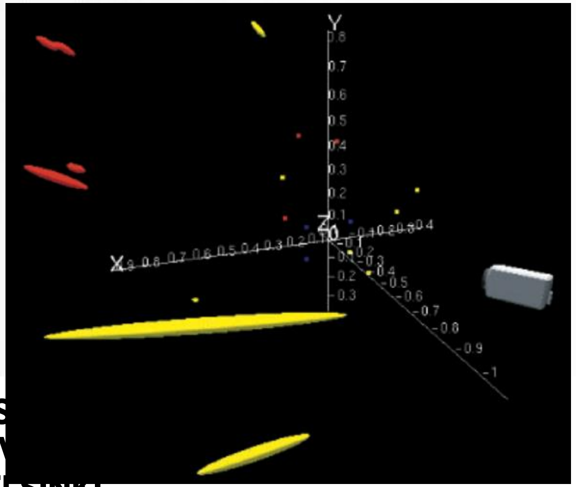
Slide credit: Kris Kitani

What is the camera (robot) state?

What are the dimensions?

$$\mathbf{x}_c = \begin{bmatrix} \mathbf{r} \\ \mathbf{q} \\ \mathbf{v} \\ \omega \end{bmatrix}$$

position	3
rotation (quaternion)	4
velocity	3
angular velocity	3
	13 total



Slide credit: Kris Kitani

What is the world (robot+environment) state?

What are the dimensions?

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_c \\ \mathbf{y}_1 \\ \mathbf{y}_2 \\ \vdots \\ \mathbf{y}_N \end{bmatrix}$$

state of the camera

location of feature 1

location of feature 2

location of feature N

Slide credit: Kris Kitani

What is the world (robot+environment) state?

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_c \\ \mathbf{y}_1 \\ \mathbf{y}_2 \\ \vdots \\ \mathbf{y}_N \end{bmatrix}$$

	What are the dimensions?
state of the camera	13
location of feature 1	3
location of feature 2	3
location of feature N	3
	13+3N total

Slide credit: Kris Kitani

Observations are…

detected visual features of landmark points.
(e.g., Harris corners)

Slide credit: Kris Kitani

What is the motion model? $P(\boldsymbol{x}_t | \boldsymbol{x}_{t-1})$

Landmarks:
constant position
(identity matrix)

Camera:
constant velocity
(not identity matrix and non-linear)

EKF!

What is the form of the belief? $P(\boldsymbol{x}_t | \boldsymbol{z}_{1:t-1})$

Slide credit: Kris Kitani

What is the motion model? $P(\boldsymbol{x}_t | \boldsymbol{x}_{t-1})$

Landmarks:
constant position
(identity matrix)

Camera:
constant velocity
(not identity matrix and non-linear)

What is the form of the belief? $P(\boldsymbol{x}_t | \boldsymbol{z}_{1:t-1})$

Gaussian!
(everything will be parametrized by a mean and variance)

Slide credit: Kris Kitani

Constant Velocity Motion Model

$$\mathbf{r}_t = \mathbf{r}_{t-1} + \mathbf{v}_{t-1}\Delta t$$ position

$$\mathbf{q}_t = \mathbf{q}_{t-1} \times [\mathbf{q}(\omega)\Delta t]$$ rotation (quaternion)

$$\mathbf{v}_t = \mathbf{v}_{t-1}$$ velocity

$$\omega_t = \omega_{t-1}$$ angular velocity

Slide credit: Kris Kitani

Gaussian noise uncertainty (only on velocity)

$$\mathbf{v}_t = \mathbf{v}_{t-1} + \mathbf{V}$$
$$\omega_t = \omega_{t-1} + \mathbf{\Omega}$$

$$\mathbf{V} \sim \mathcal{N}(\mathbf{0}, \begin{bmatrix} \sigma_v & 0 & 0 \\ 0 & \sigma_v & 0 \\ 0 & 0 & \sigma_v \end{bmatrix})$$

$$\mathbf{\Omega} \sim \mathcal{N}(\mathbf{0}, \begin{bmatrix} \sigma_w & 0 & 0 \\ 0 & \sigma_w & 0 \\ 0 & 0 & \sigma_w \end{bmatrix})$$

Slide credit: Kris Kitani

Prediction (mean of camera state):

$$P(\boldsymbol{x}_t|\boldsymbol{z}_{1:t-1}) = \int_{\boldsymbol{x}_{t-1}} P(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})P(\boldsymbol{x}_{t-1}|\boldsymbol{z}_{1:t-1})d\boldsymbol{x}_{t-1}$$

$$\mathbf{f}_t = \begin{bmatrix} \mathbf{r}_t \\ \mathbf{q}_t \\ \mathbf{v}_t \\ \omega_t \end{bmatrix} = \begin{bmatrix} \mathbf{r}_{t-1} + \mathbf{v}_{t-1}\Delta t \\ \mathbf{q}_{t-1} + \mathbf{q}(\omega)_{t-1}\Delta t \\ \mathbf{v}_{t-1} \\ \omega_{t-1} \end{bmatrix}$$

Slide credit: Kris Kitani

Prediction (covariance of camera state):

$$P(\boldsymbol{x}_t|\boldsymbol{z}_{1:t-1}) = \int_{\boldsymbol{x}_{t-1}} P(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})P(\boldsymbol{x}_{t-1}|\boldsymbol{z}_{1:t-1})d\boldsymbol{x}_{t-1}$$

$$\bar{\boldsymbol{\Sigma}}_{\mathbf{xx}} = \boxed{\frac{\partial \mathbf{f}_t}{\partial \mathbf{x}}} \boldsymbol{\Sigma}_{\mathbf{xx}} \frac{\partial \mathbf{f}_t}{\partial \mathbf{x}}^\top + \mathbf{Q}_t$$

new covariance

change around new state

old covariance

change around new state

system noise (process noise)

Bit of a pain to compute this term…

Slide credit: Kris Kitani

We just covered the **prediction** step for the camera state

$$P(\boldsymbol{x}_t|\boldsymbol{z}_{1:t-1}) = \int_{\boldsymbol{x}_{t-1}} P(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})P(\boldsymbol{x}_{t-1}|\boldsymbol{z}_{1:t-1})d\boldsymbol{x}_{t-1}$$

$$\mathbf{f}_t = \begin{bmatrix} \mathbf{r}_t \\ \mathbf{q}_t \\ \mathbf{v}_t \\ \omega_t \end{bmatrix} = \begin{bmatrix} \mathbf{r}_{t-1} + \mathbf{v}_{t-1} \\ \mathbf{q}_{t-1} + \mathbf{q}(\omega)_{t-1} \\ \mathbf{v}_{t-1} \\ \omega_{t-1} \end{bmatrix}$$

$$\bar{\boldsymbol{\Sigma}}_{\mathbf{xx}} = \frac{\partial \mathbf{f}_t}{\partial \mathbf{x}} \boldsymbol{\Sigma}_{\mathbf{xx}} \frac{\partial \mathbf{f}_t}{\partial \mathbf{x}}^\top + \mathbf{Q}_t$$

Now we need to do the **update** step!

General Filtering Equations

$$P(\boldsymbol{x}_t|\boldsymbol{z}_{1:t}) \propto P(\boldsymbol{z}_t|\boldsymbol{x}_t) \int_{\boldsymbol{x}_{t-1}} P(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) P(\boldsymbol{x}_{t-1}|\boldsymbol{z}_{1:t-1}) d\boldsymbol{x}_{t-1}$$

Prediction:

$$P(\boldsymbol{x}_t|\boldsymbol{z}_{1:t-1}) = \int_{\boldsymbol{x}_{t-1}} P(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) P(\boldsymbol{x}_{t-1}|\boldsymbol{z}_{1:t-1}) d\boldsymbol{x}_{t-1}$$

Update:

$$P(\boldsymbol{x}_t|\boldsymbol{z}_{1:t}) = P(\boldsymbol{z}_t|\boldsymbol{x}_t) P(\boldsymbol{x}_t|\boldsymbol{z}_{1:t-1})$$

Slide credit: Kris Kitani

Belief state State observation Predicted State

$$P(\boldsymbol{x}_t|\boldsymbol{z}_{1:t}) = P(\boldsymbol{z}_t|\boldsymbol{x}_t)P(\boldsymbol{x}_t|\boldsymbol{z}_{1:t-1})$$

What are the observations?

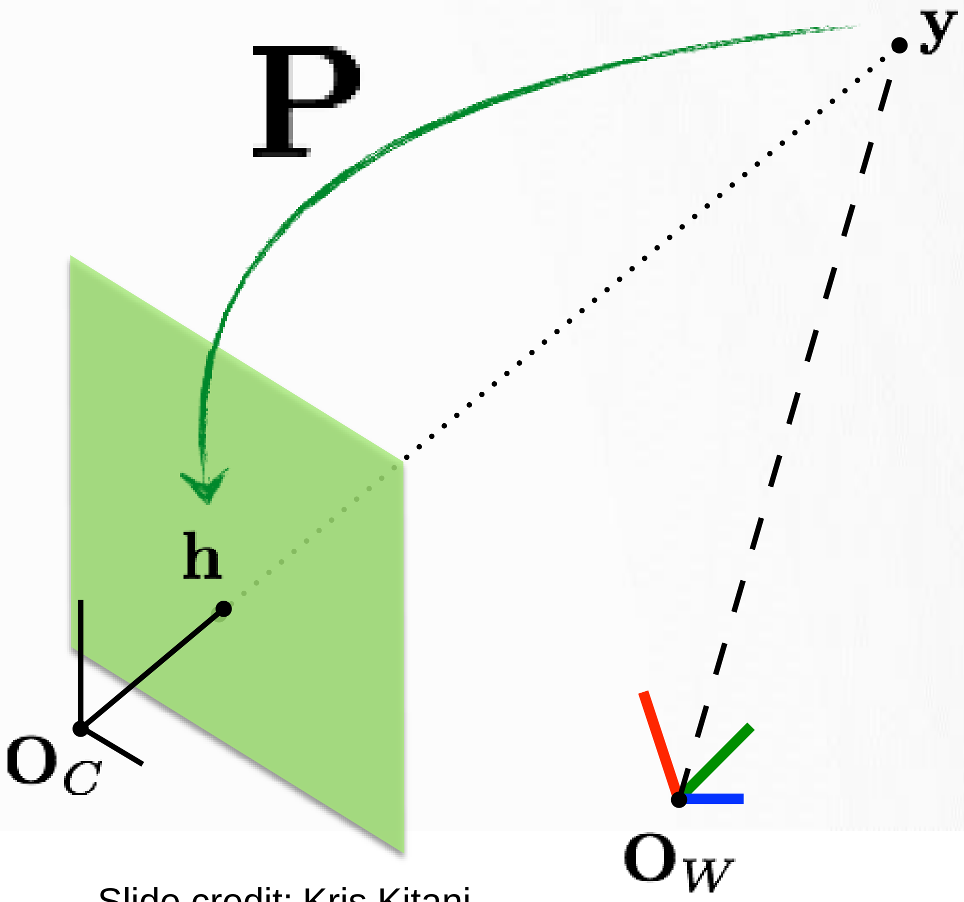
2D projections of 3D landmarks

Slide credit: Kris Kitani

Recall, the state includes the 3D location of landmarks

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_c \\ \mathbf{y}_1 \\ \mathbf{y}_2 \\ \vdots \\ \mathbf{y}_N \end{bmatrix}$$

What is the projection from 3D point to 2D image point?



Slide credit: Kris Kitani

Observation Model

$$P(z_t|x_t)$$

If you know the 3D location of a landmark, what is the 2D projection?

Non-linear observation model

$$\mathbf{h} \sim \mathbf{P}\ \mathbf{y}$$

2D Image Point

Camera matrix

3D World Point

$$\mathbf{P} = \mathbf{K}\left[\mathbf{R}|\mathbf{T}\right]$$

*What do we know about **P**?*

How do we make the observation model linear?

Slide credit: Kris Kitani

$$\underset{(2n \times 13)}{H} = \frac{\partial \mathbf{h}}{\partial \mathbf{x}}$$

n: number of visible points

I will spare you the pain of deriving the partial derivative…

$$P(\boldsymbol{x}_t|\boldsymbol{z}_{1:t}) = P(\boldsymbol{z}_t|\boldsymbol{x}_t)P(\boldsymbol{x}_t|\boldsymbol{z}_{1:t-1})$$

Update step (mean):

$$\underbrace{\mathbf{x}_t}_{\text{Updated state}} = \underbrace{\mathbf{x}_t}_{\text{Predicted state}} + \overbrace{\mathbf{K}_t}^{\text{Kalman gain}}(\underbrace{\mathbf{z}_t}_{\text{Matched 2D features}} - \underbrace{\mathbf{h}(\mathbf{y};\mathbf{x}_t)}_{\text{2D projection of 3D point}})$$

Update step (covariance):

$$\underbrace{\mathbf{\Sigma}_t}_{\text{Covariance (updated)}} = (\overbrace{\mathbf{I}}^{\text{Identity}} - \overbrace{\mathbf{K}_t}^{\text{Kalman gain}}\underbrace{\mathbf{H}_t}_{\text{Jacobian}})\underbrace{\mathbf{\Sigma}_t}_{\text{Covariance (predicted)}}$$

Slide credit: Kris Kitani

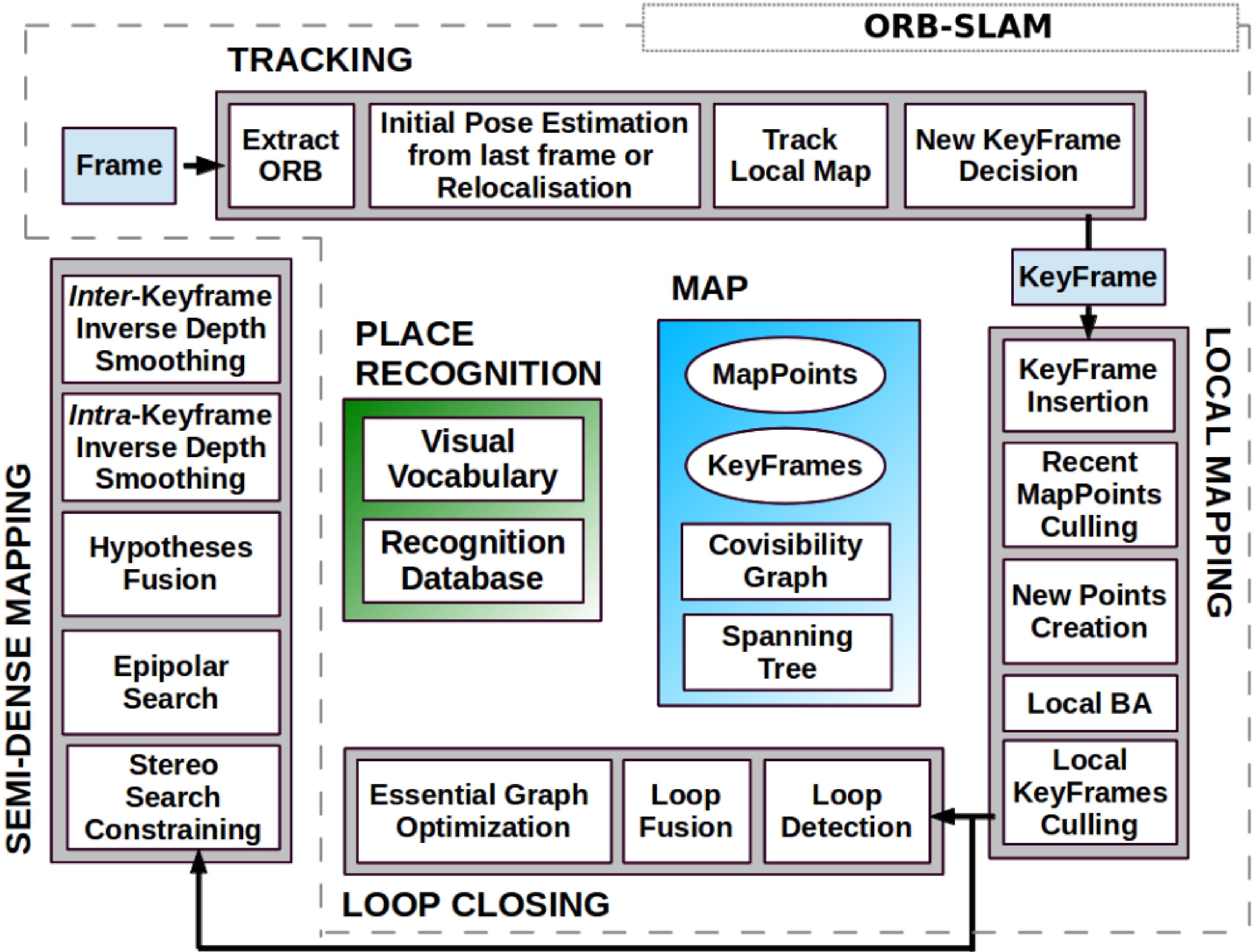
KEYFRAME- BASED SLAM

- Filterig is replaced with bundle adjustment
  - Keep every measurement that goes into the map
  - Keeping all previous poses in estimation is to ocomputationally heavy => using only subset of poses = **Key-frames**
  - Split map-making and camera pose tracking into two separate threads

ORB-SLAM

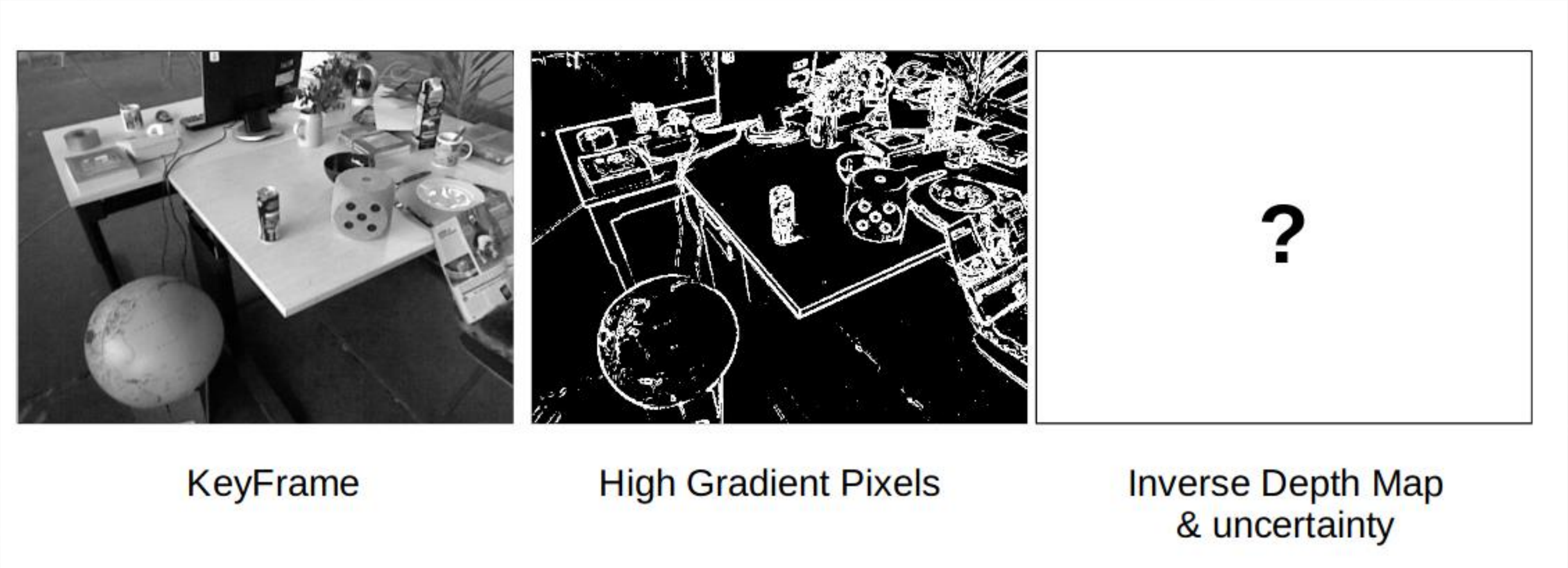
- State-of-the-art feature and keyframe based SLAM
- ORB name comes from the ORB features that are tracked
  - Similar to SIFT but much faster to be computed
- Semi-dense mapping

https://www.youtube.com/watch?v=HlBmq70LKrQ&feature=youtu.be



The Future of Real-Time SLAM (ICCV'15 Workshop). Raúl Mur Artal. University of Zaragoza

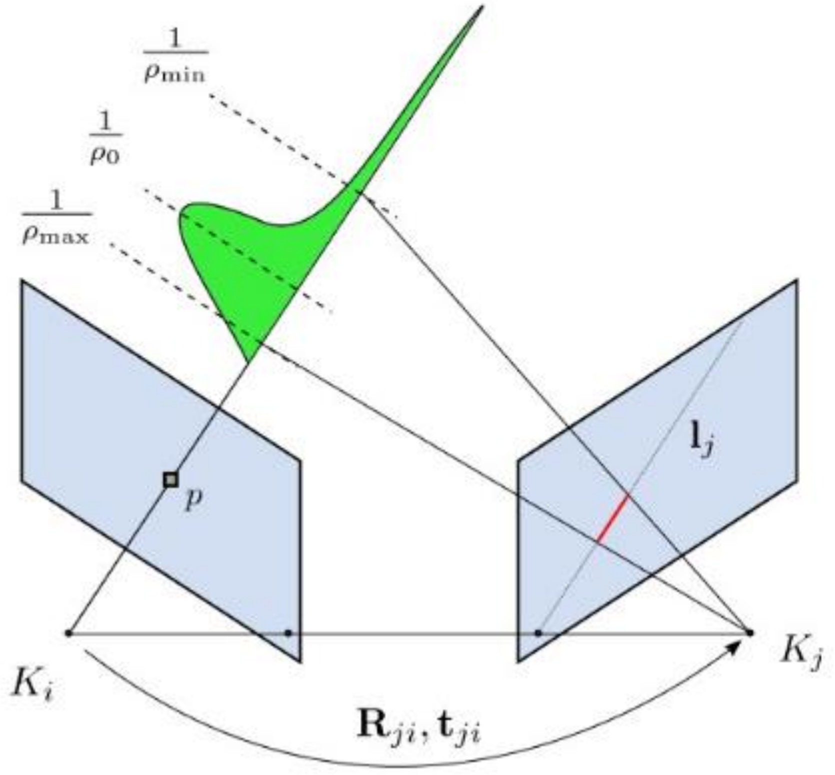
PROBABILISTIC SEMI-DENSE MAPPING

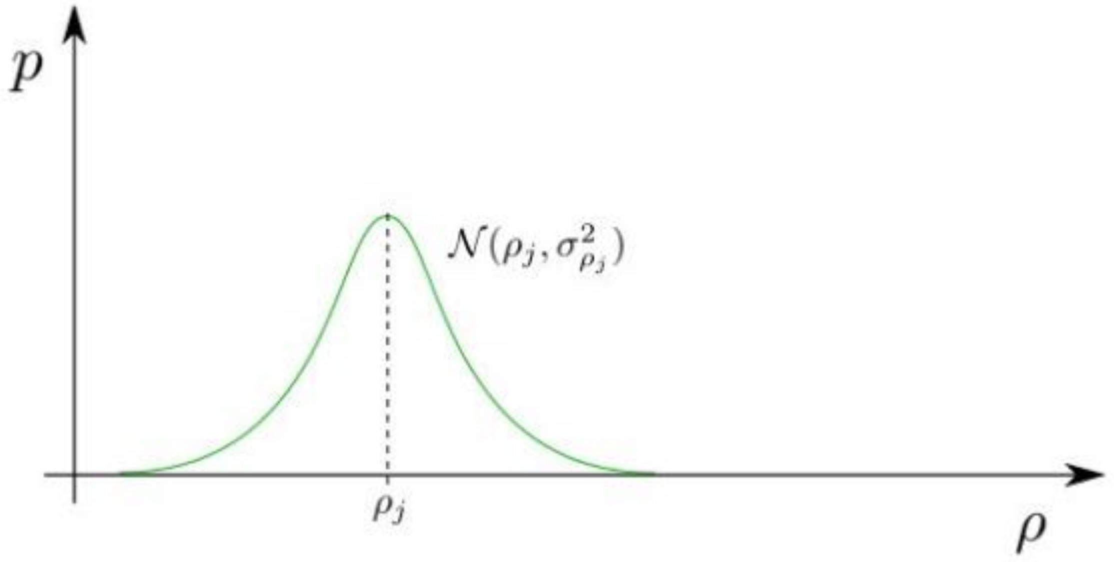


Compute each inverse depth map from scratch using neighbor keyframes

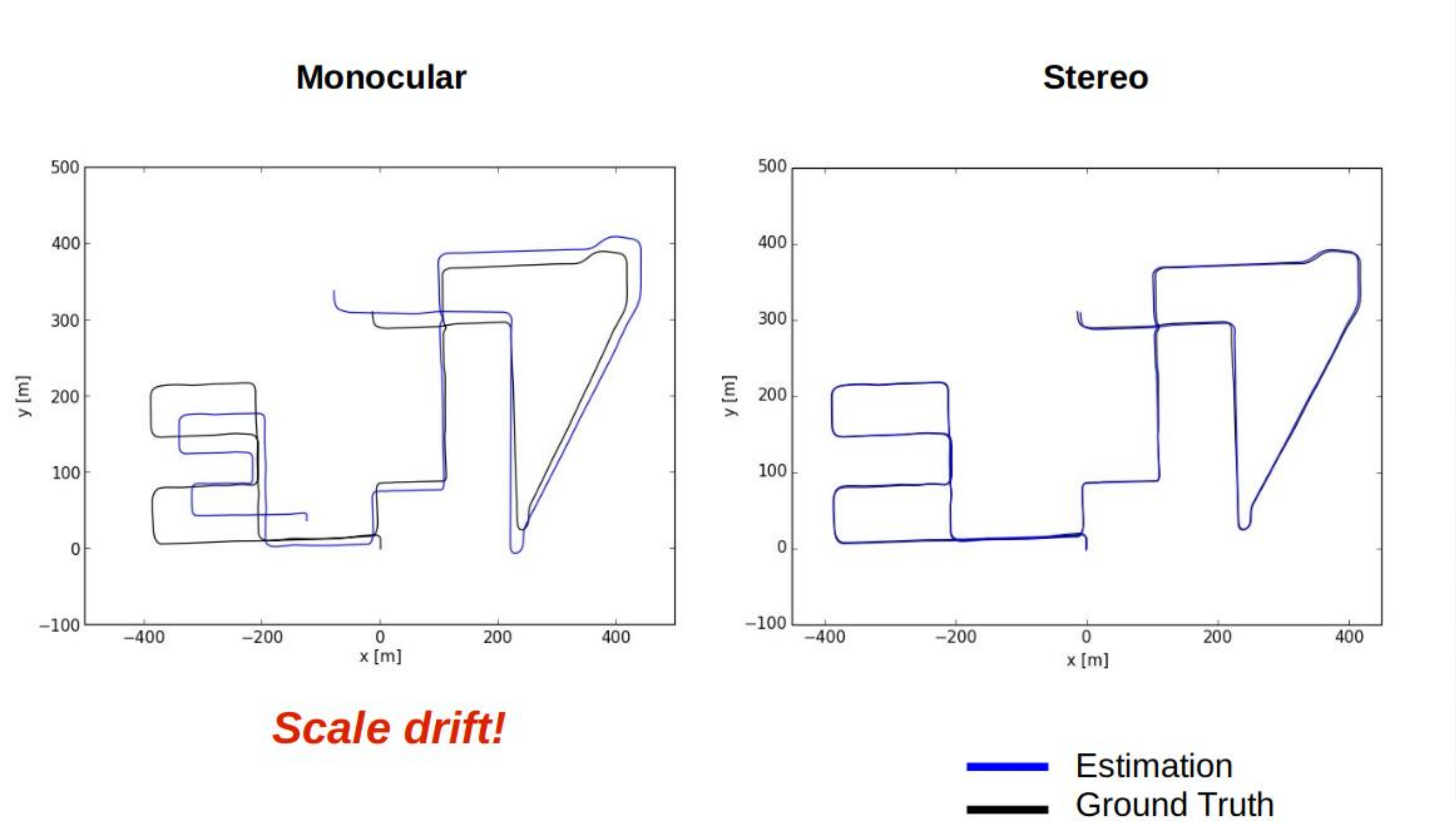
The Future of Real-Time SLAM (ICCV'15 Workshop). Raúl Mur Artal. University of Zaragoza

Per Pixel Operations





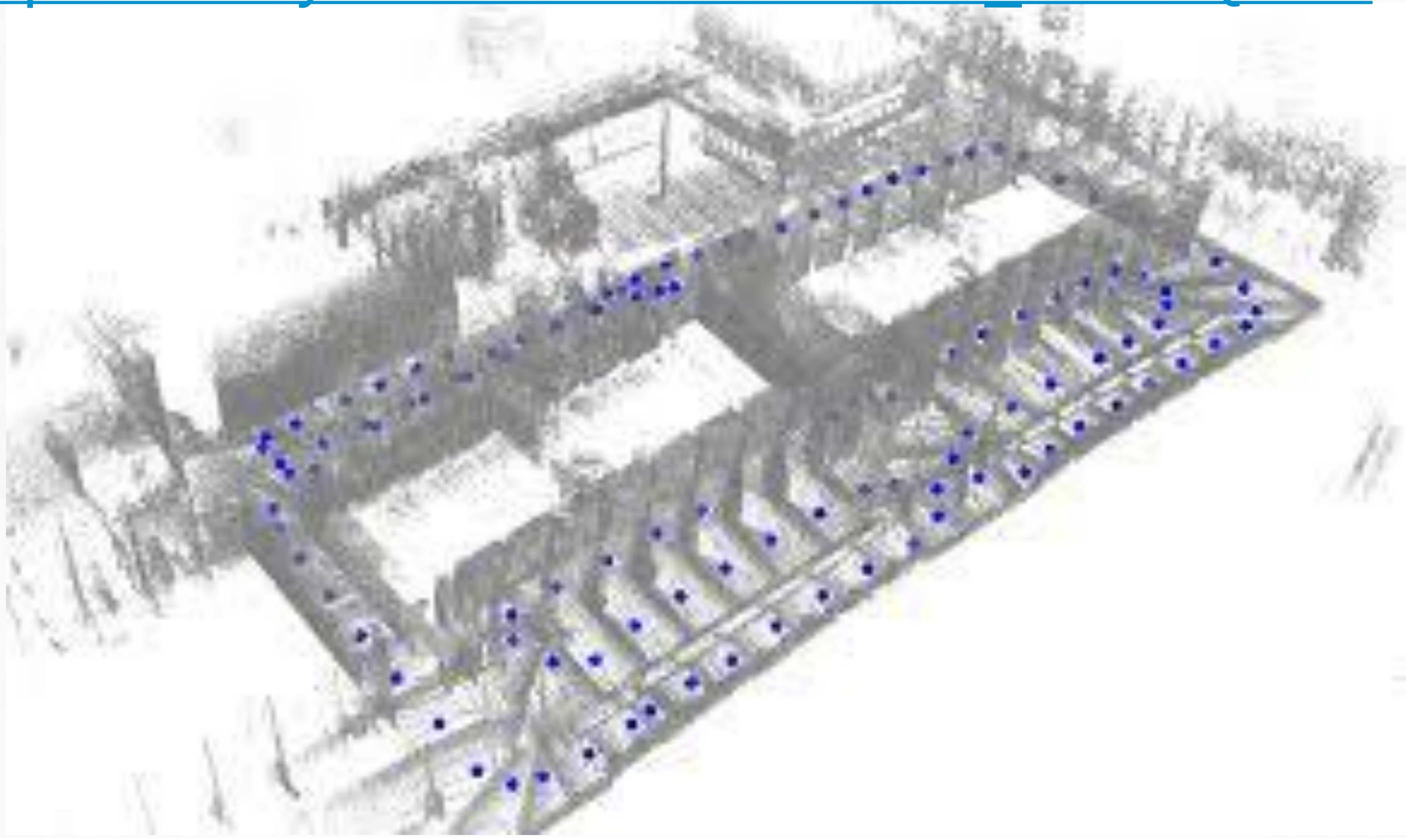
The Future of Real-Time SLAM (ICCV'15 Workshop). Raúl Mur Artal. University of Zaragoza



The Future of Real-Time SLAM (ICCV'15 Workshop). Raúl Mur Artal. University of Zaragoza

CNN-SLAM: REAL-TIME DENSE MONOCULAR SLAM WITH LEARNED DEPTH PREDICTION

https://www.youtube.com/watch?v=z_NJxbkQnBU




60˚ 10 1.2 N, 24˚ 57 18 E