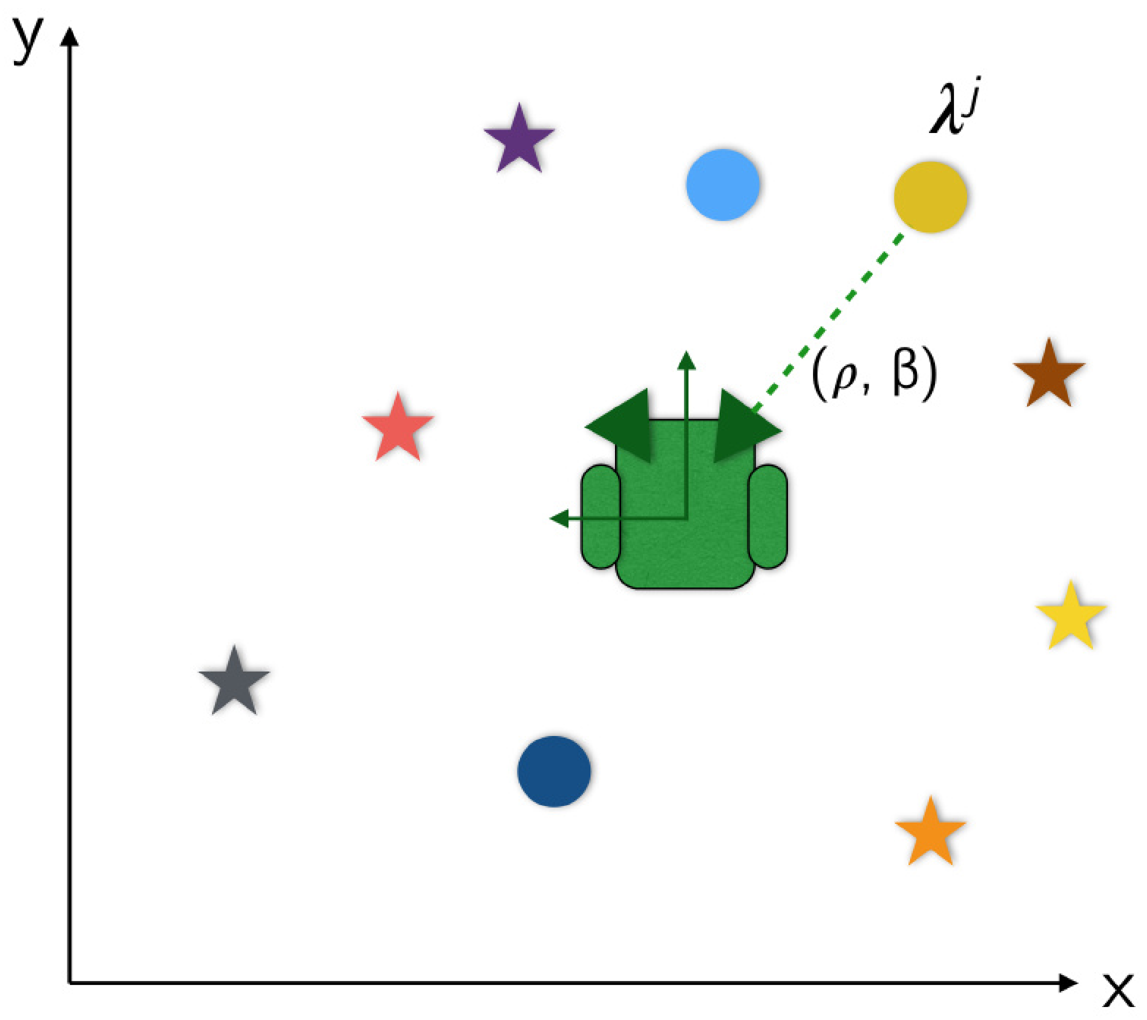
16-311-Q Introduction to Robotics Fall'17

# Lecture 21: EKF for Map Building

Instructor:
Gianni A. Di Caro

## SO FAR: WE HAVE A MAP, THE ROBOT DOES MOVE

- Scenario: The robot does move, external observations of landmarks are made, and a map is given in input with the coordinates of the landmarks (*Prediction problem*)



Landmark Map

| Id | x | y |
|---|---|---|
| 1 | 2.2 | 1.5 |
| 2 | 0.5 | 4.0 |
| ... | ... | ... |

## LINEARIZATION WAS REQUIRED→EKF

- The state-observation equations: the state vector $\boldsymbol{\xi}$ corresponds to the 2D pose of the robot; the observations of the landmarks are made using a range finder sensor that returns range $\rho_i$, bearing $\beta_i$ and identity $i$ of the observed landmark; the identity information is used to retrieve the position $(\lambda_x^i, \lambda_y^i)$ of the landmark from the map:

$$\boldsymbol{\xi}_{k+1} = \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} + \begin{bmatrix} (\Delta S_k + \nu_k^s)\cos(\theta_k + \frac{\Delta\theta_k}{2} + \nu_k^\theta) \\ (\Delta S_k + \nu_k^s)\sin(\theta_k + \frac{\Delta\theta_k}{2} + \nu_k^\theta) \\ \Delta\theta_k + \nu_k^\theta \end{bmatrix} = \boldsymbol{f}_k(\boldsymbol{\xi}_k, \boldsymbol{\nu}_k; \Delta S_k, \Delta\theta_k)$$

$$z_{k+1} = \begin{bmatrix} \sqrt{(\lambda_{kx}^i - x_k)^2 + (\lambda_{ky}^i - y_k)^2} \\ \arctan\Big((\lambda_{ky}^i - y_k)/(\lambda_{kx}^i - x_k)\Big) - \theta_k \end{bmatrix} + \begin{bmatrix} w_k^\rho \\ w_k^\beta \end{bmatrix} = \boldsymbol{h}_k(\boldsymbol{\xi}_k, \boldsymbol{w}_k; \boldsymbol{\lambda}_k^i)$$

**Non linear equations → 1st Taylor series for linearization → EKF**

## THE EKF EQUATIONS

- The EKF equations:

At every time step $k$:

$$\hat{\boldsymbol{\xi}}_{k+1|k} = \boldsymbol{f}_k(\hat{\boldsymbol{\xi}}_{k|k}, \mathbf{0}; \Delta S_k, \Delta\theta_k)$$

$$\boldsymbol{P}_{k+1|k} = \boldsymbol{F}_{k\xi} \boldsymbol{P}_k \boldsymbol{F}_{k\xi}^T + \boldsymbol{F}_{k\nu} \boldsymbol{V}_k \boldsymbol{F}_{k\nu}^T$$

At every time step $k+1$ when a *landmark* is observed

$$\hat{\boldsymbol{\xi}}_{k+1} = \hat{\boldsymbol{\xi}}_{k+1|k} + \boldsymbol{G}_{k+1}(z_{k+1} - \boldsymbol{h}_k(\hat{\boldsymbol{\xi}}_{k+1|k}, \mathbf{0}; \boldsymbol{\lambda}^i))$$

$$\boldsymbol{P}_{k+1} = \boldsymbol{P}_{k+1|k} - \boldsymbol{G}_{k+1} \boldsymbol{H}_{k\xi} \boldsymbol{P}_{k+1|k}$$

$$\boldsymbol{G}_{k+1} = \boldsymbol{P}_{k+1|k} \boldsymbol{H}_{k\xi}^T \boldsymbol{S}_{k+1}^{-1}$$

$$\boldsymbol{S}_{k+1} = \boldsymbol{H}_{k\xi} \boldsymbol{P}_{k+1|k} \boldsymbol{H}_{k\xi}^T + \boldsymbol{H}_{kw} \boldsymbol{W}_{k+1} \boldsymbol{H}_{kw}^T$$

- The Jacobians $\boldsymbol{F}_{k\xi}$ and $\boldsymbol{F}_{k\nu}$ of $\boldsymbol{f}_k()$, that have to be evaluated in $(\boldsymbol{\xi}_k = \hat{\boldsymbol{\xi}}_{k|k}, \boldsymbol{\nu}_k = 0)$, for the **linearization of the motion dynamics**:

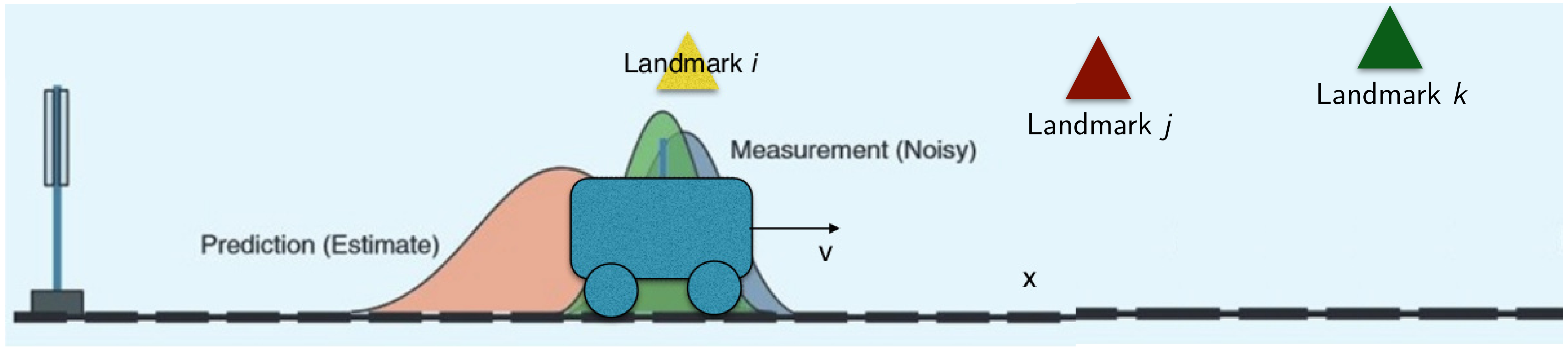
$$\boldsymbol{F}_{k\xi} = \begin{bmatrix} 1 & 0 & -\Delta S_k \sin(\theta_k + \frac{\Delta\theta_k}{2}) \\ 0 & 1 & \Delta S_k \cos(\theta_k + \frac{\Delta\theta_k}{2}) \\ 0 & 0 & 1 \end{bmatrix}_{\hat{\xi}_{k|k}, \mathbf{0}} \qquad \boldsymbol{F}_{k\nu} = \begin{bmatrix} \cos(\theta_k + \frac{\Delta\theta_k}{2}) & -\Delta S_k \sin(\theta_k + \frac{\Delta\theta_k}{2}) \\ \sin(\theta_k + \frac{\Delta\theta_k}{2}) & \Delta S_k \cos(\theta_k + \frac{\Delta\theta_k}{2}) \\ 0 & 1 \end{bmatrix}_{\hat{\xi}_{k|k}, \mathbf{0}}$$

- The Jacobians $\boldsymbol{H}_{k\xi}$ and $\boldsymbol{H}_{kw}$ of $\boldsymbol{h}_k()$, that have to be evaluated in $(\boldsymbol{\xi}_k = \hat{\boldsymbol{\xi}}_{k+1|k}, \boldsymbol{\nu}_k = 0)$, for the **linearization of the observation model** (the $\boldsymbol{\lambda}_k^i$ are parameters):

$$\boldsymbol{H}_{k\xi} = \begin{bmatrix} -\frac{\lambda_{kx}^i - x_k}{r_k^i} & -\frac{\lambda_{ky}^i - y_k}{r_k^i} & 0 \\ \frac{\lambda_{ky}^i - y_k}{(r_k^i)^2} & -\frac{\lambda_{kx}^i - x_k}{(r_k^i)^2} & -1 \end{bmatrix}_{\hat{\xi}_{k+1|k}, \mathbf{0}} \qquad \boldsymbol{H}_{kw} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad r_k^i = \sqrt{(\lambda_{kx}^i - x_k)^2 + (\lambda_{ky}^i - y_k)^2}$$

# STATE ESTIMATION FOR A ROBOT MOVING ON A TRACK

- **Scenario:** The robot does move, but its motion is constrained on a *rectilinear track* (e.g., an automatic-driving train) $\rightarrow$ Motion happens along one single dimension, $x$



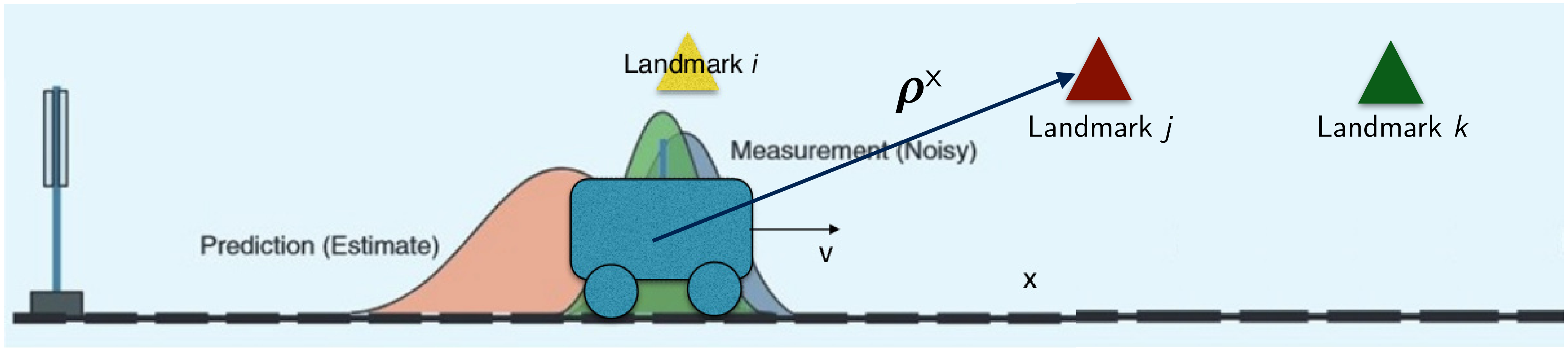
- The robot issues velocity control actions, $u(t)$, making the robot always moving in one direction. Robot's velocity between control inputs is constant.
- Slipping/friction effects make the relation between velocity controls and traveled distance (robot's position) *noisy*.
- Observable landmarks are present and can be used to correct position prediction when observed (Prediction problem)

## STATE ESTIMATION FOR A ROBOT MOVING ON A TRACK

- **State vector:** pair (position - velocity) $\rightarrow \boldsymbol{\xi} = [\, x \;\; v \,]^T$
- *Velocity inputs* are given at discrete time intervals $\Delta T$ (i.e., the time between step $k$ and step $k+1$ is $\Delta T$ seconds
- Landmarks' observations are measures returning:
  - The relative distance $\boldsymbol{\rho}^x$ of the landmark from the train along the track:

$$\boldsymbol{z}_{k+1} = \boldsymbol{\rho}^x$$

  - The identity $i$ of the observed landmark, whose 1D position coordinate $\lambda^i{}_x$ can be retrieved from the map given as input
  - The bearing is not needed in this case given that the robot is constrained on moving along the track



## STATE ESTIMATION FOR A ROBOT MOVING ON A TRACK

- **State dynamics**

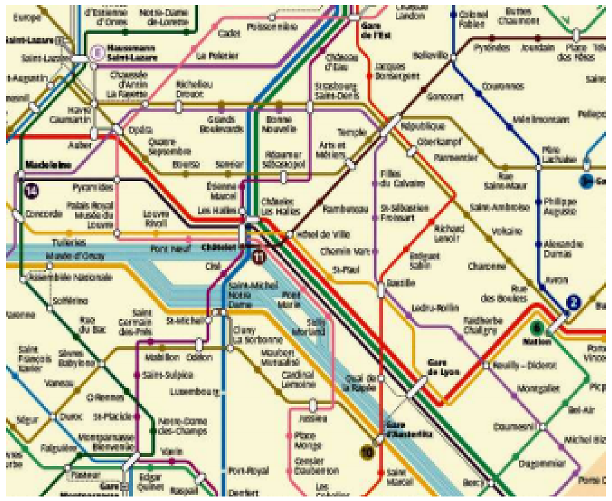
$$\boldsymbol{\xi}_{k+1} = \begin{bmatrix} x_{k+1} \\ v_{k+1} \end{bmatrix} = \begin{bmatrix} x_k + v_k \Delta T \\ u_k \end{bmatrix} + \begin{bmatrix} \nu_k^x \\ \nu_k^v \end{bmatrix} = \begin{bmatrix} 1 & \Delta T \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_k \\ v_k \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u_k + \begin{bmatrix} \nu_k^x \\ \nu_k^v \end{bmatrix} = \boldsymbol{A}\boldsymbol{\xi}_k + \boldsymbol{B} u_k + \boldsymbol{\nu}_k$$

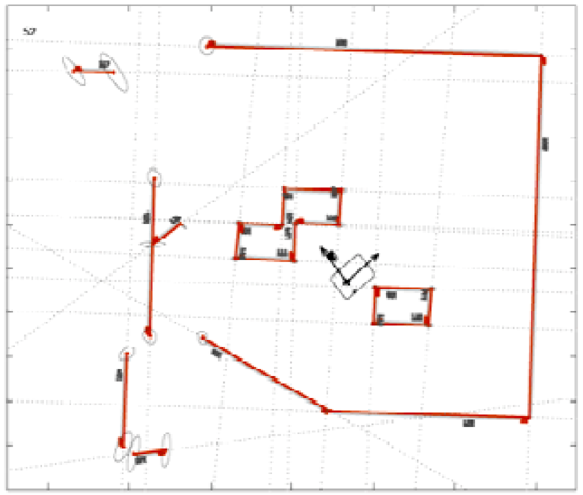
- **Observation prediction equation**

$$z_{k+1} = \left[\rho^x\right] = \left[\sqrt{(\lambda_x^i - x_k)^2}\right] + w_k^x = \boldsymbol{h}_k(x_k, w_k^x; \lambda_x^i) \equiv \boldsymbol{h}_k(\boldsymbol{\xi}_k, \boldsymbol{w}_k; \lambda_x^i)$$

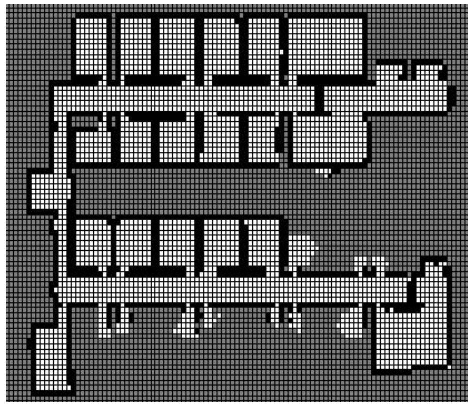
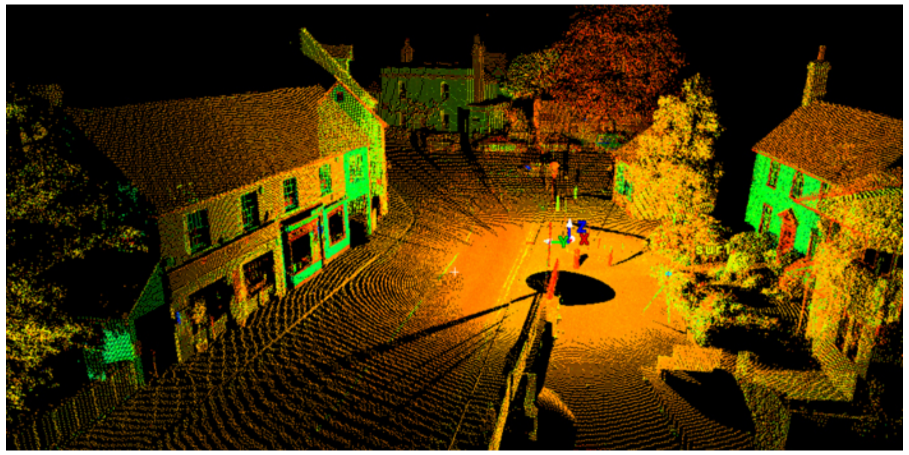
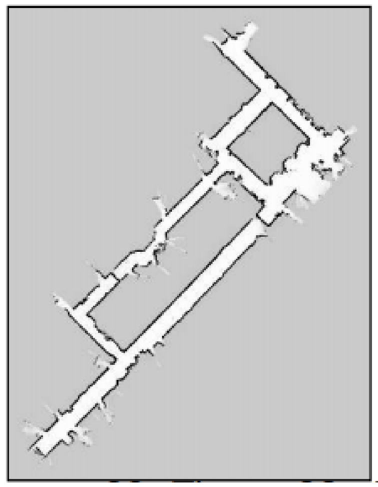
## SO FAR, WE HAVE A MAP…

- Metric and/or topological representations of the environment

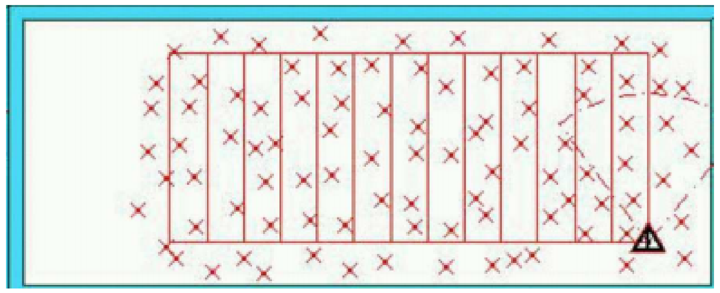
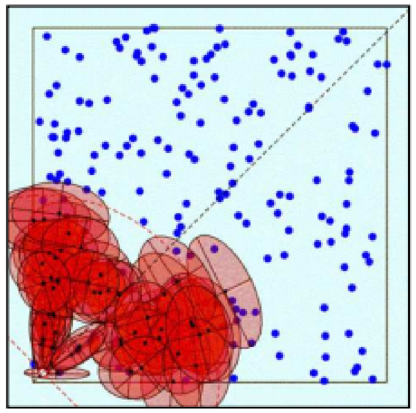




- Grid-based, 2D-3D scan

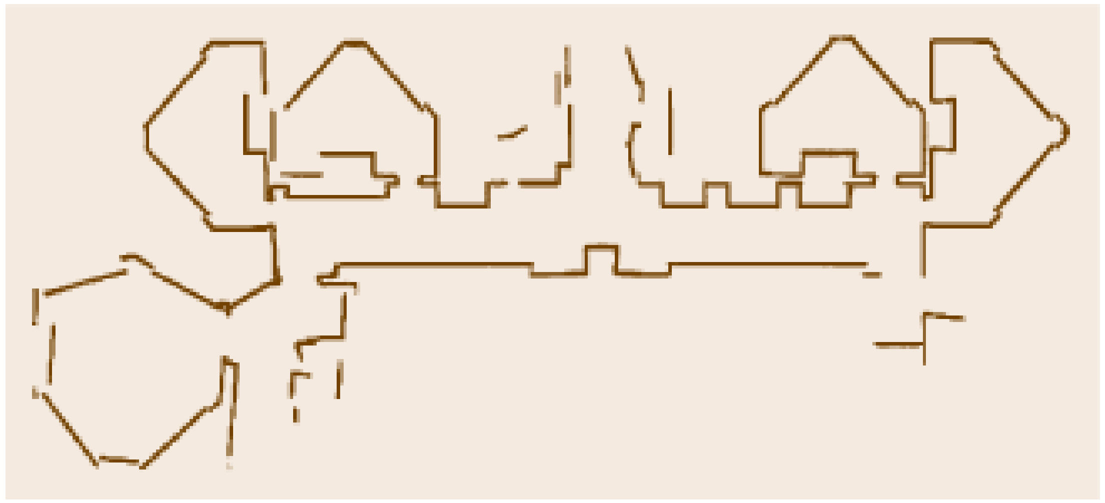




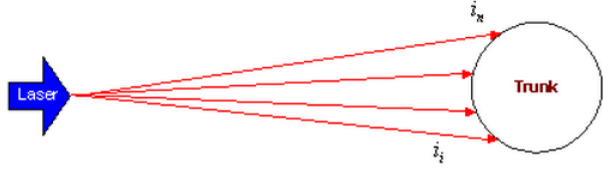
- Landmark-based

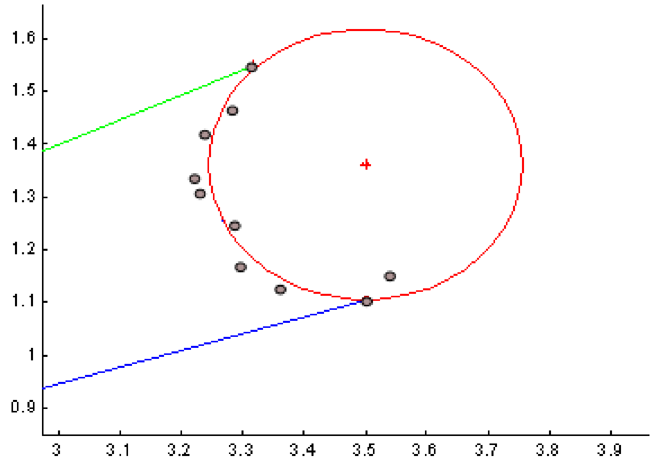



# TYPES OF MAPS: FEATURES

- An occupancy grid can, in principle, be based on raw sensor measurements (e.g., from a range sensor). An alternative approach is to extract features from the stream of raw measurements. This amounts to a reduction in complexity, but requires a **feature extractor**
- For instance, for *range sensors*, it is common to extract **geometric features** such as lines, corners, or arcs, that can correspond respectively to walls, intersections, or trees.







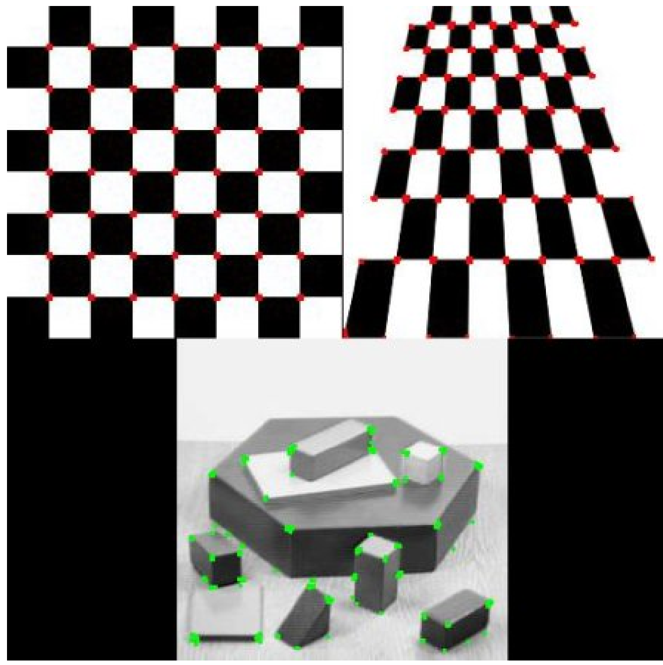
- Extracted features might correspond to **distinct objects in the physical world**, such as door posts, window stills, tree trunks, or corners of buildings $\rightarrow$ In robotics, it is common to call those physical objects landmarks or beacons (if they are explicitly used to guide navigation towards a desired destination).

## TYPES OF MAPS: FEATURES

- For *vision-based sensors*, a number of techniques have been developed to automatically extract a large number of features from images. Popular approaches include SIFT and SURF.

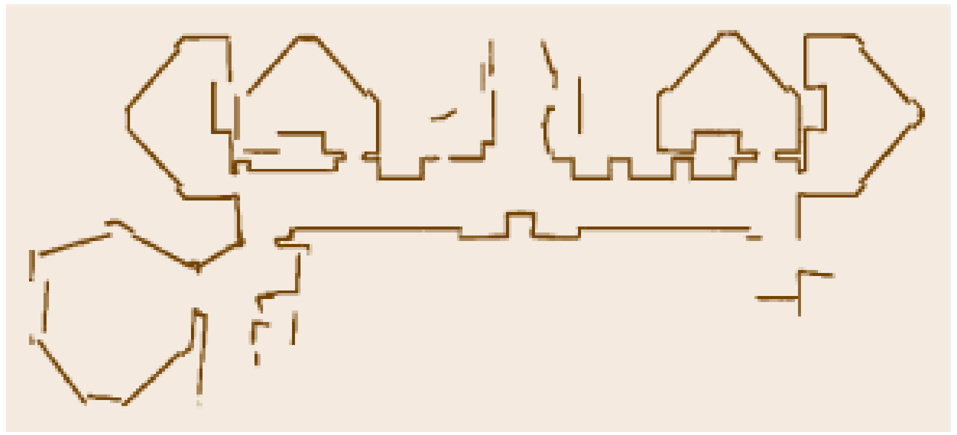
SIFT

SURF


Harris

# LINE FEATURE MAPS

- Line models, compact and often obtainable with closed forms

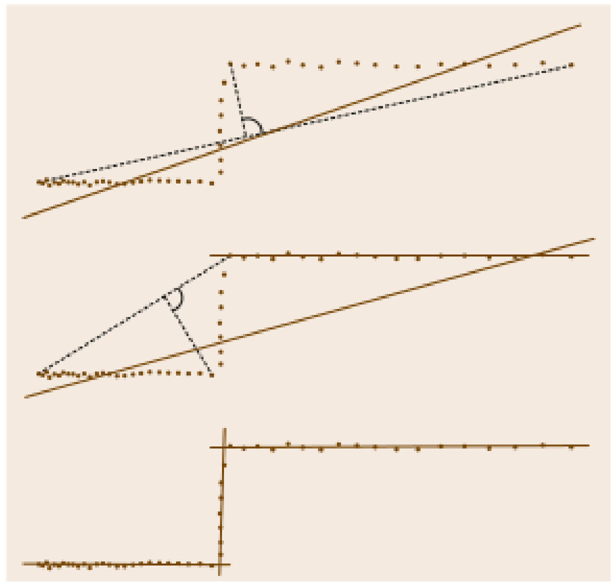


- If $n$ data points are returned from the sensor as Cartesian coordinates $(x_i, y_i)$, the line that minimizes the squared distances from all points can be calculated in closed form by solving

$$\begin{aligned} \tan 2\phi &= \frac{-2\sum_i(\bar{x}-x_i)(\bar{y}-y_i)}{\sum_i\left[(\bar{y}-y_i)^2-(\bar{x}-x_i)^2\right]} \\ r &= \bar{x}\cos\phi + \bar{y}\sin\phi \end{aligned}$$

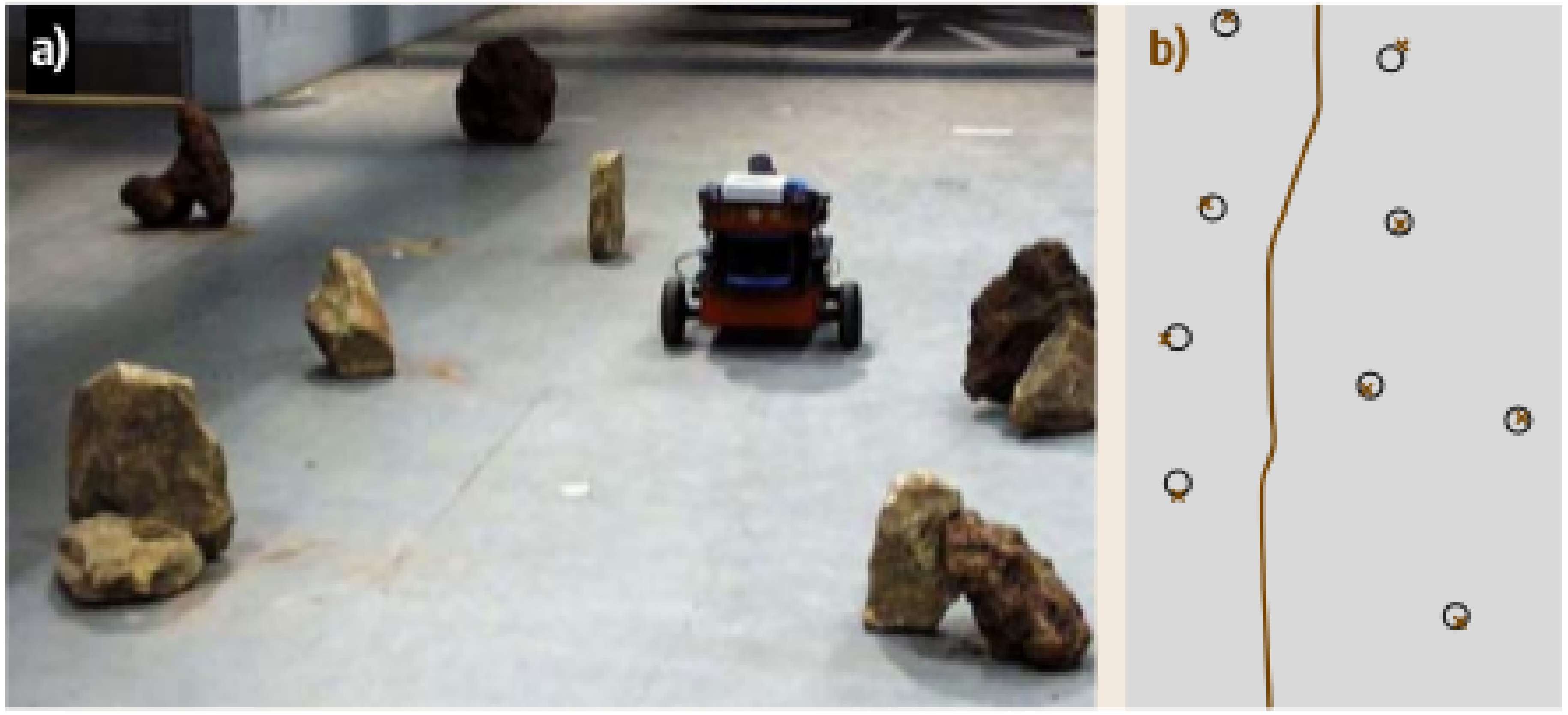
where $\bar{x} = (\sum_i x_i)/n$, $\bar{y} = (\sum_i y_i)/n$, $r$ is the normal distance of the line from the origin, and $\phi$ is the angle of the normal

- When the data points are generated from multiple linear structures no closed form exists $\rightarrow$ *Split-and-merge* algorithm that recursively subdivides the point set into subsets that can be more accurately approximated by a line



## LANDMARKS

- Landmarks can be naturally present in the considered environment (e.g., doors in indoors) or can be placed ad hoc, precisely to favor robot navigation (e.g., the use of RFID or LED beacons)
- A landmark in the map is described by its measured features, its estimated location, and by a *signature* (e.g., a distinctive color), that can be thought as its **identity** (we have mostly assumed that the signature is read from the data, as it could be for a radio beacon)
- A map can be populated by a relatively high number of point landmarks (e.g., 1,000), but this is usually much smaller than the number of grid cells in an occupancy map (*sparse* vs. *dense* mapping)



## WHAT IF NO MAP IS AVAILABLE?

What if the map is *not* given?

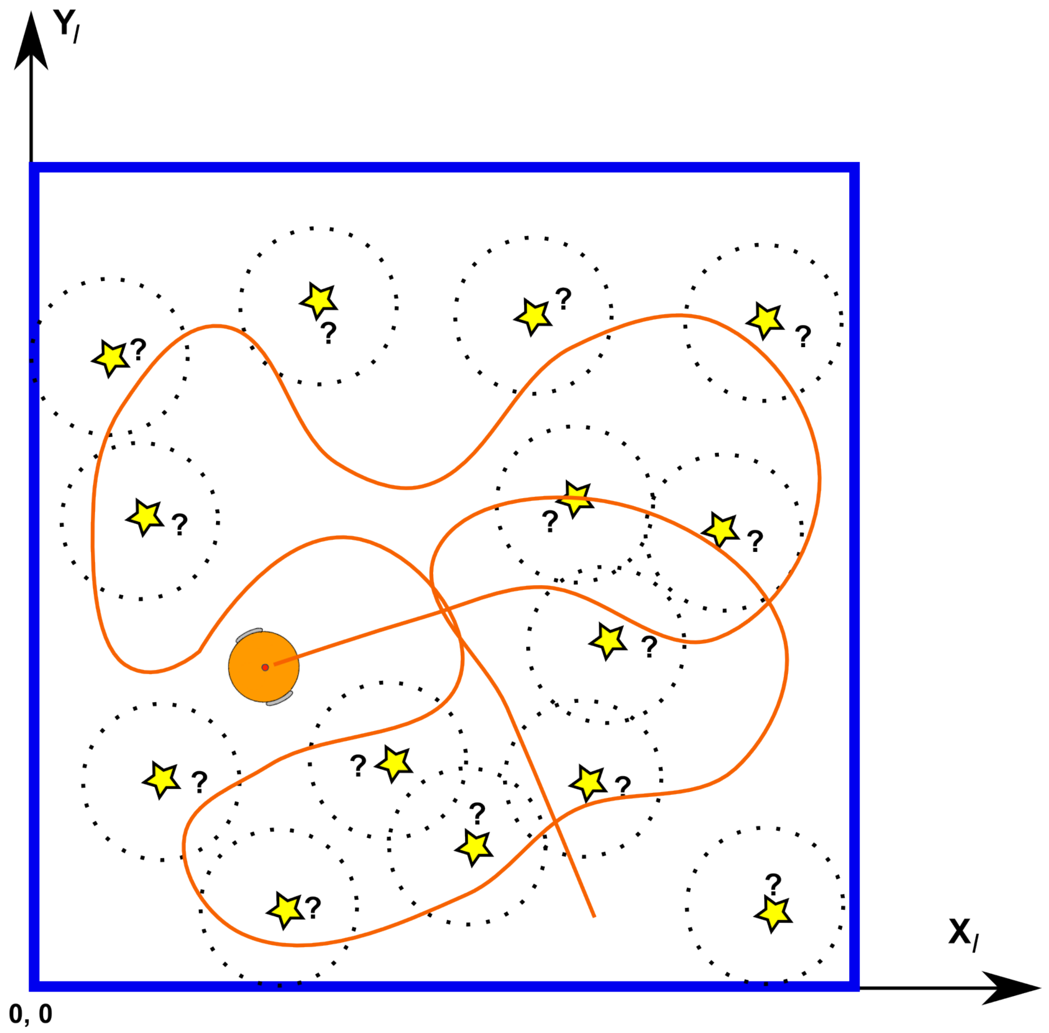
There are $M$ landmarks in the environment but the robot does know neither the **number** $M$ nor the **position** of the landmarks

$\Downarrow$

Use the EKF (or, more generally, an estimator) to **create** the map:

the robot moves around and makes landmark observations

$\rightarrow$ from the observations the position of the landmarks is recursively estimated



## ROAD MAP TO SLAM

The (final) goal is to build a map while at the same time performing pose estimation:
**Simultaneous Localization and Mapping (SLAM)**

Road map:

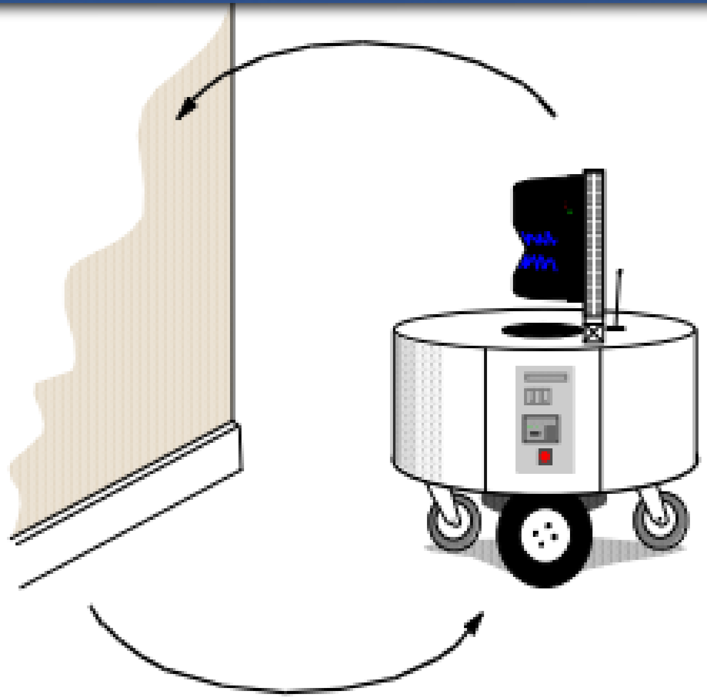
1. Previous results: We know how to make localization estimation in the presence of a map using KF/EKF

2. Let's now first learn how to make a map assuming perfect pose knowledge for the mobile robot, meaning that $\boldsymbol{\xi} = \begin{bmatrix} x_k & y_k & \theta_k \end{bmatrix}^T$ is known exactly any step $k$:

$$\widehat{\boldsymbol{\xi}}_k \equiv \boldsymbol{\xi}_k, \quad P_k = \mathbf{0}$$

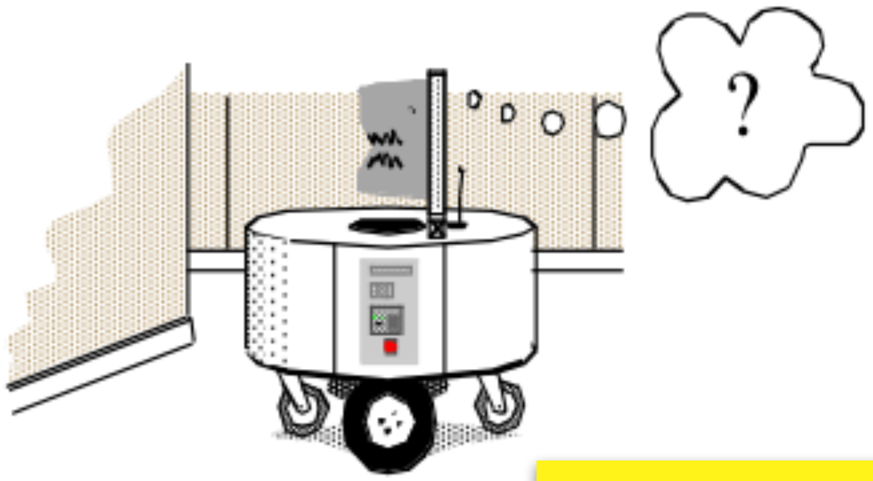
3. Finally, combine the results from 1. and 2. to deal with the general case in which **both the map of the environment and the pose of the robot are unknown** $\rightarrow$ SLAM

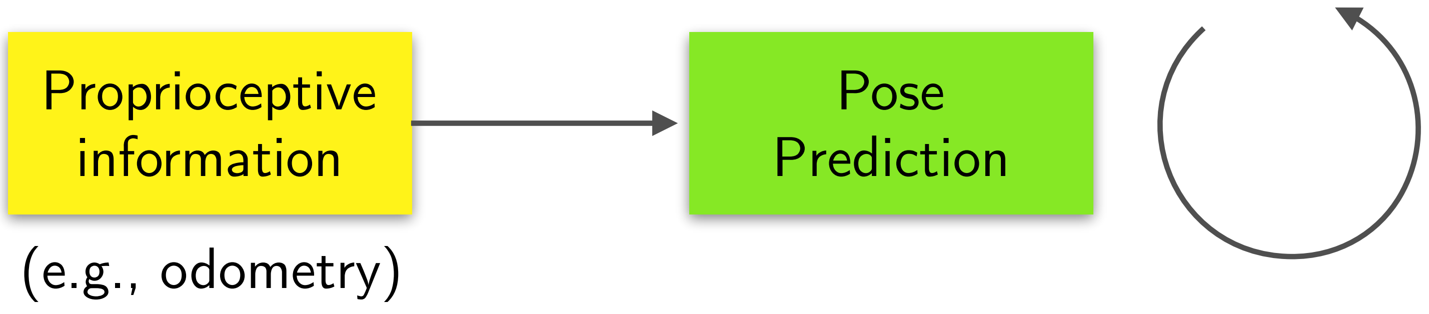
## SIMULTANEOUS LOCALIZATION AND MAPPING



- SLAM is a chicken-or-egg problem:
  - A map is needed for localizing a robot
  - A good pose estimate is needed to build a map
- It's a **fundamental** but **hard** problem, necessary to achieve robot autonomy
- Applications examples are:
  1. Indoor: vacuum cleaner, hospital logistics
  2. Air: surveillance, forest monitoring
  3. Underwater: sea-life and coastal monitoring
  4. Underground: mine exploration and mapping
  5. Space: terrain mapping for localization

## LOCALIZATION AND MAPPING SCENARIOS (1)

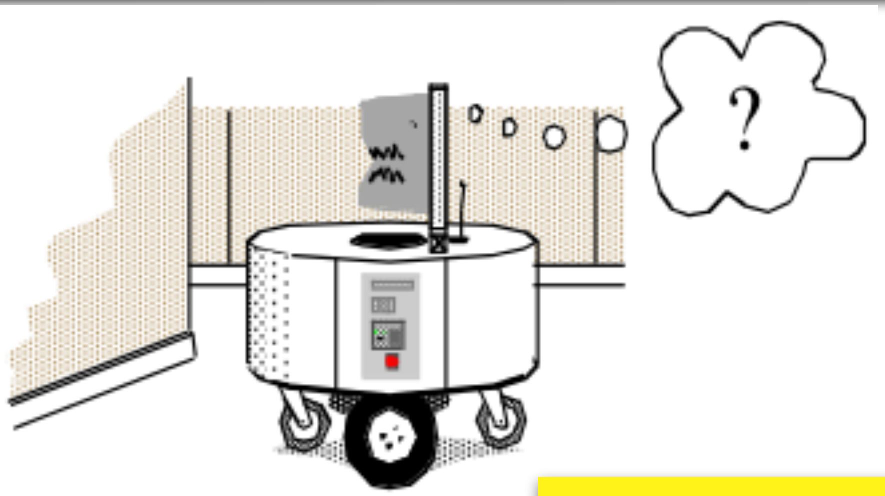


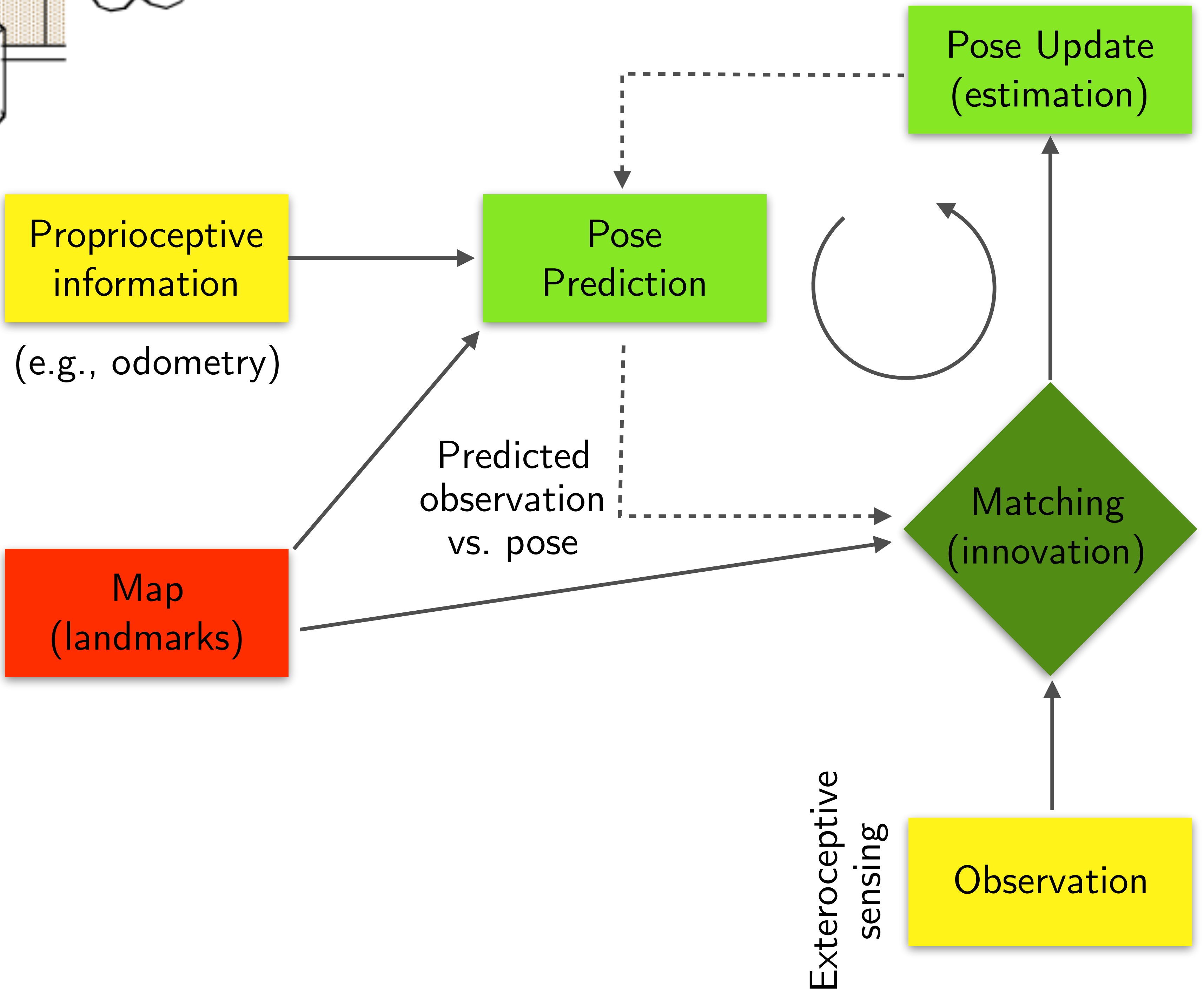


$$\mathit{State} = \mathit{Robot\ pose}$$

- ***Robot Localization scenario***
- Odometry measures for issued velocity controls
- Pose prediction errors grow unbounded
- EKF: linearization of the motion process equations

# LOCALIZATION AND MAPPING SCENARIOS (2)

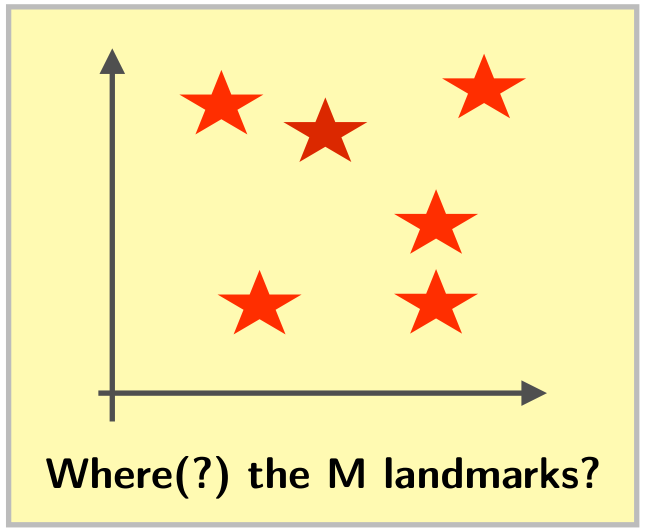


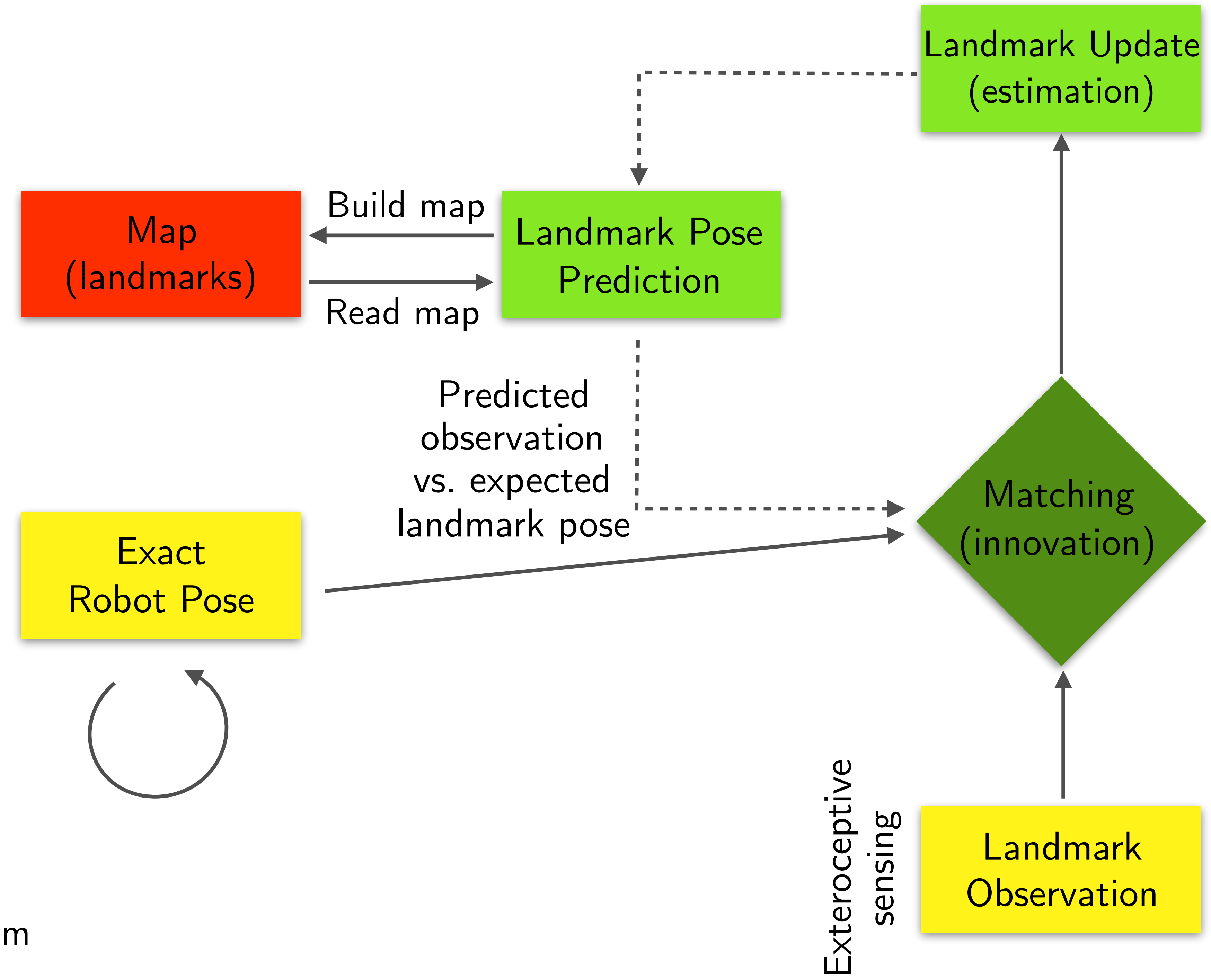


- ***Robot Localization scenario***
- Odometry measures for issued velocity controls
- Landmark map in input
- Noisy landmark observations by range and bearing sensors
- Pose prediction errors depend on observations
- EKF: linearization of the motion and observation equations

*State = Robot pose*

## LOCALIZATION AND MAPPING SCENARIOS (3)

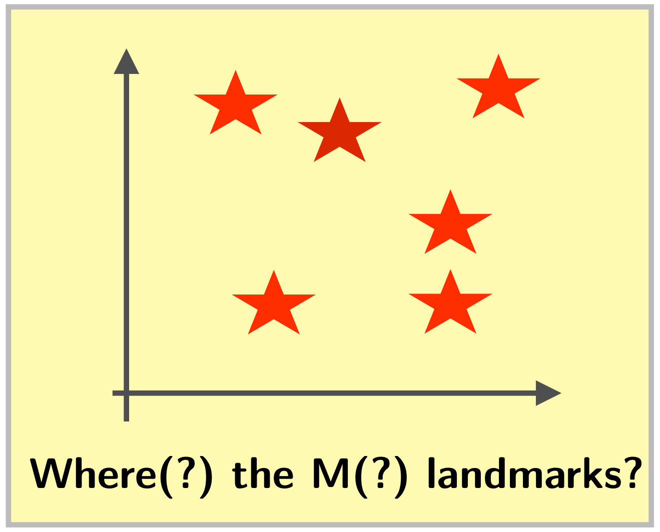


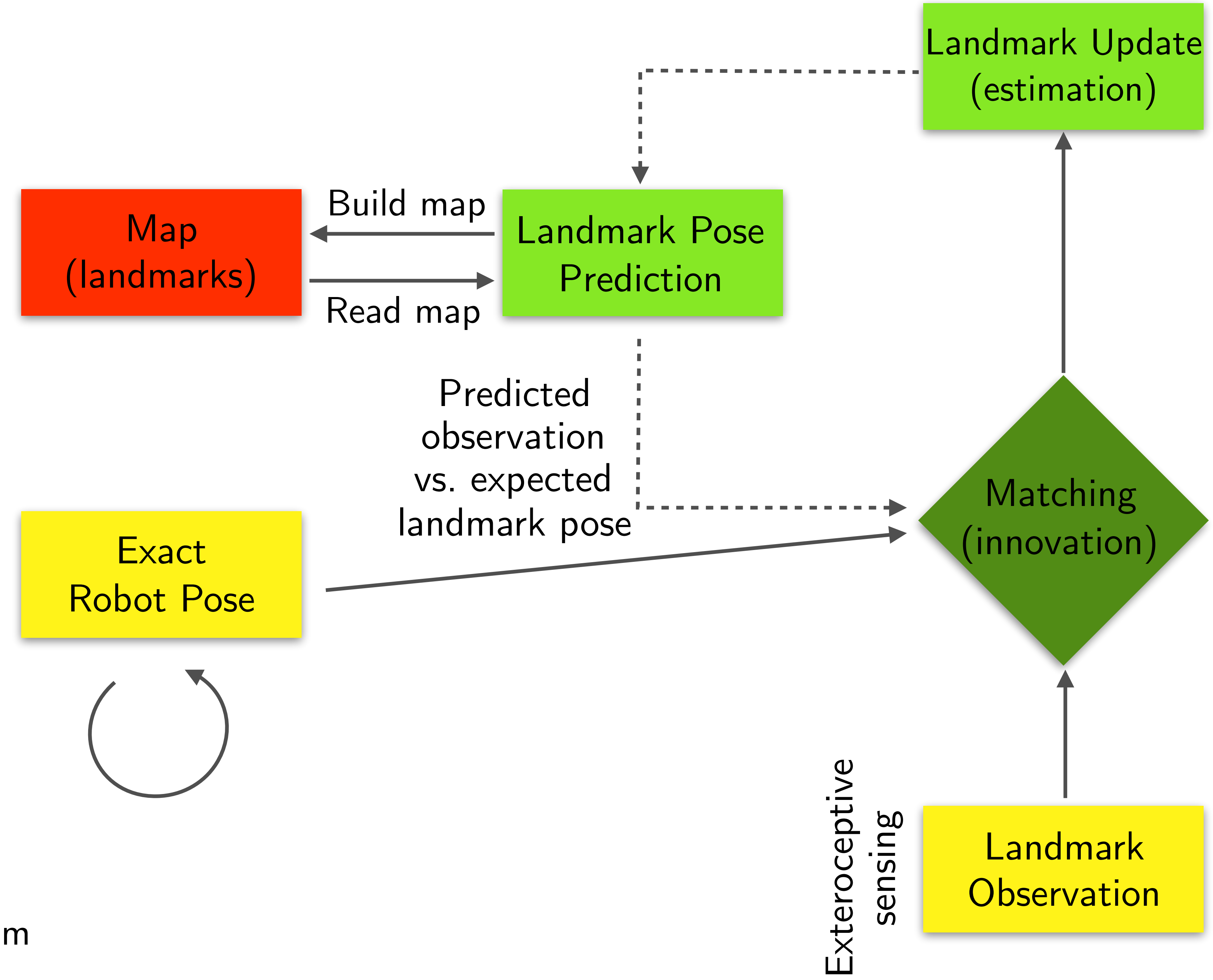


- ***Map-building scenario***
- Robot pose is known exactly
- No landmark map in input, but known number of landmarks
- Landmark pose prediction errors depend on noisy observations from range and bearing sensors
- EKF: no motion error, linearization of the landmark observation equations

*State = Coordinates of map landmarks*

# LOCALIZATION AND MAPPING SCENARIOS (4)

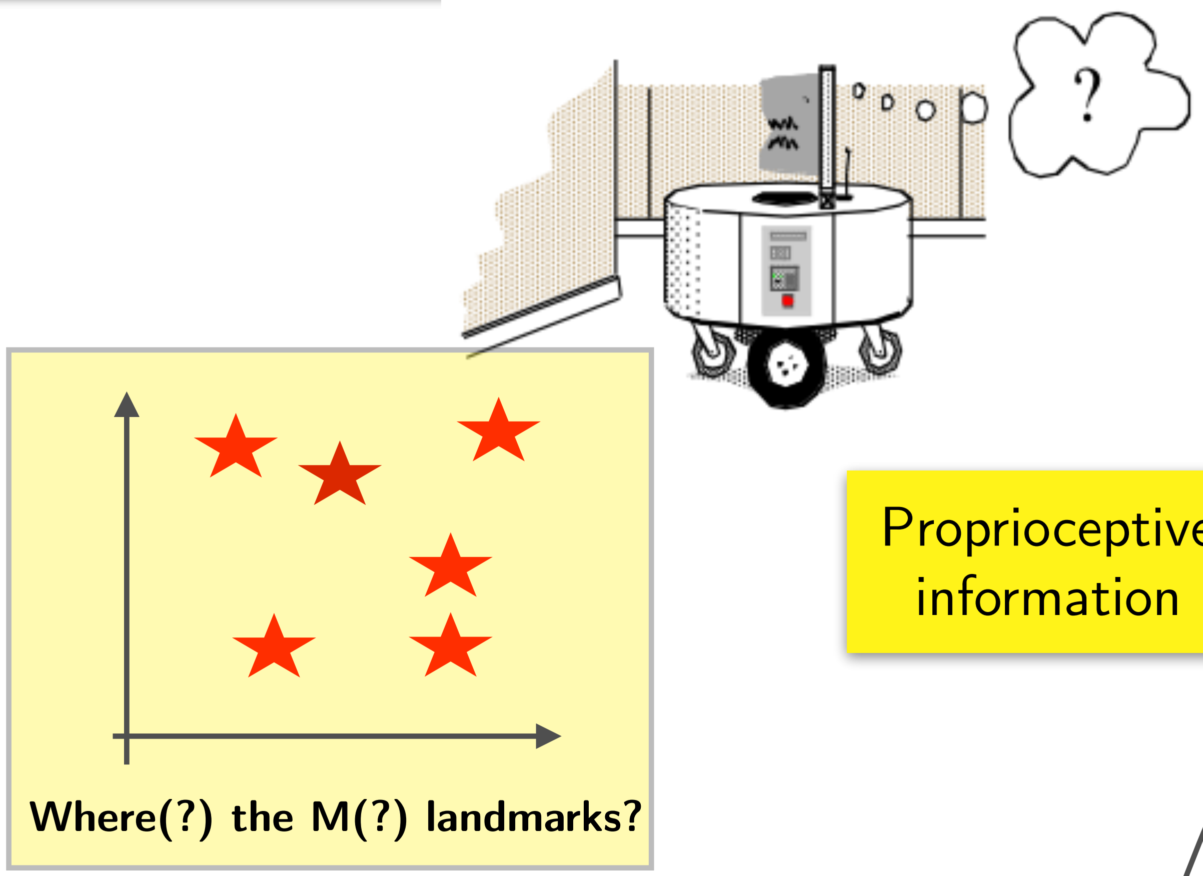




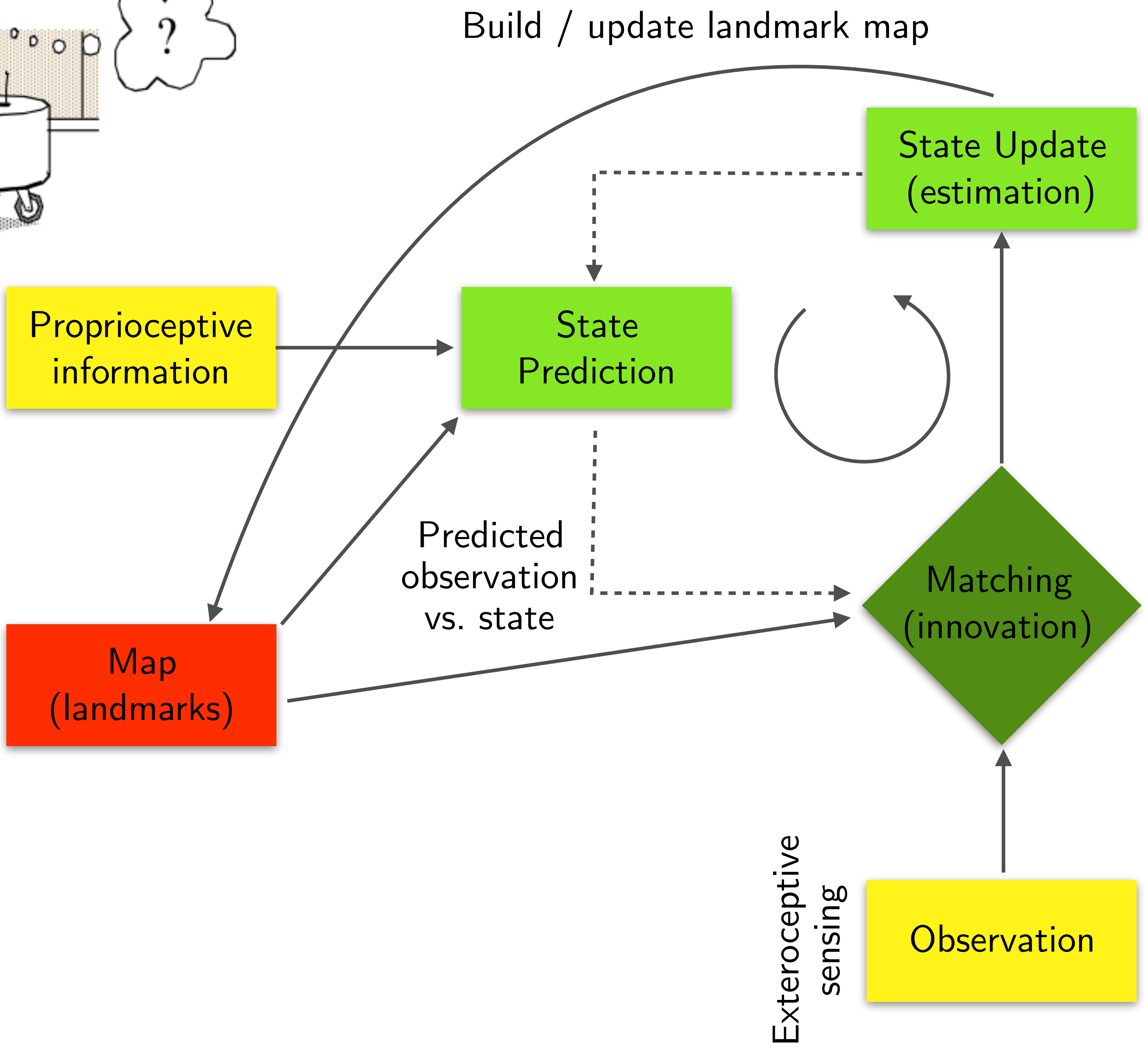
- ***Map-building scenario***
- Robot pose is known exactly
- No landmark map in input, unknown number of landmarks
- Landmark pose prediction errors depend on noisy observations from range and bearing sensors
- EKF: no motion error, linearization of the landmark observation equations

*State (variable size) = Coordinates of map landmarks*

# LOCALIZATION AND MAPPING SCENARIOS (5)



- ***Mapping & Localization scenario***
- Estimation of robot pose
- No landmark map in input
- Noisy landmark observations by range and bearing sensors
- Pose prediction errors depend on landmark and odometry measures
- EKF: linearization of the motion and observation equations


Build / update landmark map
State Update (estimation)
Proprioceptive information
State Prediction
Predicted observation vs. state
Matching (innovation)
Map (landmarks)
Exteroceptive sensing
Observation

*State (variable size) = Robot pose + Coordinates of map landmarks*

## ESTIMATING LANDMARK POSITIONS

- **Scenario:** The robot needs to move (which mobility model?) in the environment in order to observe landmarks using its sensors (affected by noise) and recursively adjust the estimated position of the observed landmarks, in order to eventually build a (usable) landmark map
- **Assumption** (for the time being)**:**
  - While moving, the pose of the robot in the environment is assumed to be precisely known $\rightarrow$ robot's coordinates $[\,x_k \;\; y_k \;\; \theta_k]$ are *parameters*
  - The number, $M$, of the landmarks to map is known

▶ The **state vector $\boldsymbol{\xi}$** corresponds to the unknown locations of the $M$ landmarks that are *known* to be in the environment (as a first step $M$ is known . . . ):

$$\boldsymbol{\xi} = \left[\lambda_x^1 \;\; \lambda_y^1 \;\; \lambda_x^2 \;\; \lambda_y^2 \;\; \ldots \;\; \lambda_x^M \;\; \lambda_y^M\right]^T$$

$\rightarrow$ $\boldsymbol{\xi}$ has (max) dimensions: $2M \times 1$, $\quad \boldsymbol{P}$ has (max) dimensions: $2M \times 2M$

**Goal**: Recursively estimate the state $\boldsymbol{\xi} \rightarrow$ Build the landmark map with good accuracy $\rightarrow$ Output good estimates of landmark positions

## STATE VECTOR AND COVARIANCE MATRIX

$$
\boldsymbol{\xi} = \begin{bmatrix} \lambda_x^1 \\ \lambda_y^1 \\ \lambda_x^2 \\ \lambda_y^2 \\ \dots \\ \dots \\ \lambda_x^M \\ \lambda_y^M \end{bmatrix}
\qquad
\boldsymbol{P}_{2M\times 2M} = \begin{bmatrix}
\sigma_{\lambda_x^1\lambda_x^1} & \sigma_{\lambda_x^1\lambda_y^1} & \sigma_{\lambda_x^1\lambda_x^2} & \sigma_{\lambda_x^1\lambda_y^2} & \dots & \sigma_{\lambda_x^1\lambda_x^M} & \sigma_{\lambda_x^1\lambda_y^M} \\
\sigma_{\lambda_y^1\lambda_x^1} & \sigma_{\lambda_y^1\lambda_y^1} & \sigma_{\lambda_y^1\lambda_x^2} & \sigma_{\lambda_y^1\lambda_y^2} & \dots & \sigma_{\lambda_y^1\lambda_x^M} & \sigma_{\lambda_y^1\lambda_y^M} \\
\sigma_{\lambda_x^2\lambda_x^1} & \sigma_{\lambda_x^2\lambda_y^1} & \sigma_{\lambda_x^2\lambda_x^2} & \sigma_{\lambda_x^2\lambda_y^2} & \dots & \sigma_{\lambda_x^2\lambda_x^M} & \sigma_{\lambda_x^2\lambda_y^M} \\
\sigma_{\lambda_y^2\lambda_x^1} & \sigma_{\lambda_y^2\lambda_y^1} & \sigma_{\lambda_y^2\lambda_x^2} & \sigma_{\lambda_y^2\lambda_y^2} & \dots & \sigma_{\lambda_y^2\lambda_x^M} & \sigma_{\lambda_y^2\lambda_y^M} \\
\dots & \dots & \dots & \dots & \dots & \dots & \dots \\
\sigma_{\lambda_x^M\lambda_x^1} & \sigma_{\lambda_x^M\lambda_y^1} & \sigma_{\lambda_x^M\lambda_x^2} & \sigma_{\lambda_x^M\lambda_y^2} & \dots & \sigma_{\lambda_x^M\lambda_x^M} & \sigma_{\lambda_x^M\lambda_y^M} \\
\sigma_{\lambda_y^M\lambda_x^1} & \sigma_{\lambda_y^M\lambda_y^1} & \sigma_{\lambda_y^M\lambda_x^2} & \sigma_{\lambda_y^M\lambda_y^2} & \dots & \sigma_{\lambda_y^M\lambda_x^M} & \sigma_{\lambda_y^M\lambda_y^M}
\end{bmatrix}
$$

In a more compact way, grouping individual covariance $2 \times 2$ sub-matrices:

$$
\Sigma_{\boldsymbol{\lambda}^i\boldsymbol{\lambda}^k} = \begin{bmatrix} \sigma_{\lambda_x^i\lambda_x^k} & \sigma_{\lambda_x^i\lambda_y^k} \\ \sigma_{\lambda_y^i\lambda_x^k} & \sigma_{\lambda_y^i\lambda_y^k} \end{bmatrix}
$$

$$
\boldsymbol{\xi} = \begin{bmatrix} \boldsymbol{\lambda}^1 \\ \boldsymbol{\lambda}^2 \\ \dots \\ \boldsymbol{\lambda}^M \end{bmatrix}
\qquad
\boldsymbol{P} = \begin{bmatrix}
\Sigma_{\boldsymbol{\lambda}^1\boldsymbol{\lambda}^1} & \Sigma_{\boldsymbol{\lambda}^1\boldsymbol{\lambda}^2} & \dots & \Sigma_{\boldsymbol{\lambda}^1\boldsymbol{\lambda}^M} \\
\Sigma_{\boldsymbol{\lambda}^2\boldsymbol{\lambda}^1} & \Sigma_{\boldsymbol{\lambda}^2\boldsymbol{\lambda}^2} & \dots & \Sigma_{\boldsymbol{\lambda}^2\boldsymbol{\lambda}^M} \\
\dots & \dots & \dots & \dots \\
\Sigma_{\boldsymbol{\lambda}^M\boldsymbol{\lambda}^1} & \Sigma_{\boldsymbol{\lambda}^M\boldsymbol{\lambda}^2} & \dots & \Sigma_{\boldsymbol{\lambda}^M\boldsymbol{\lambda}^M}
\end{bmatrix}
$$

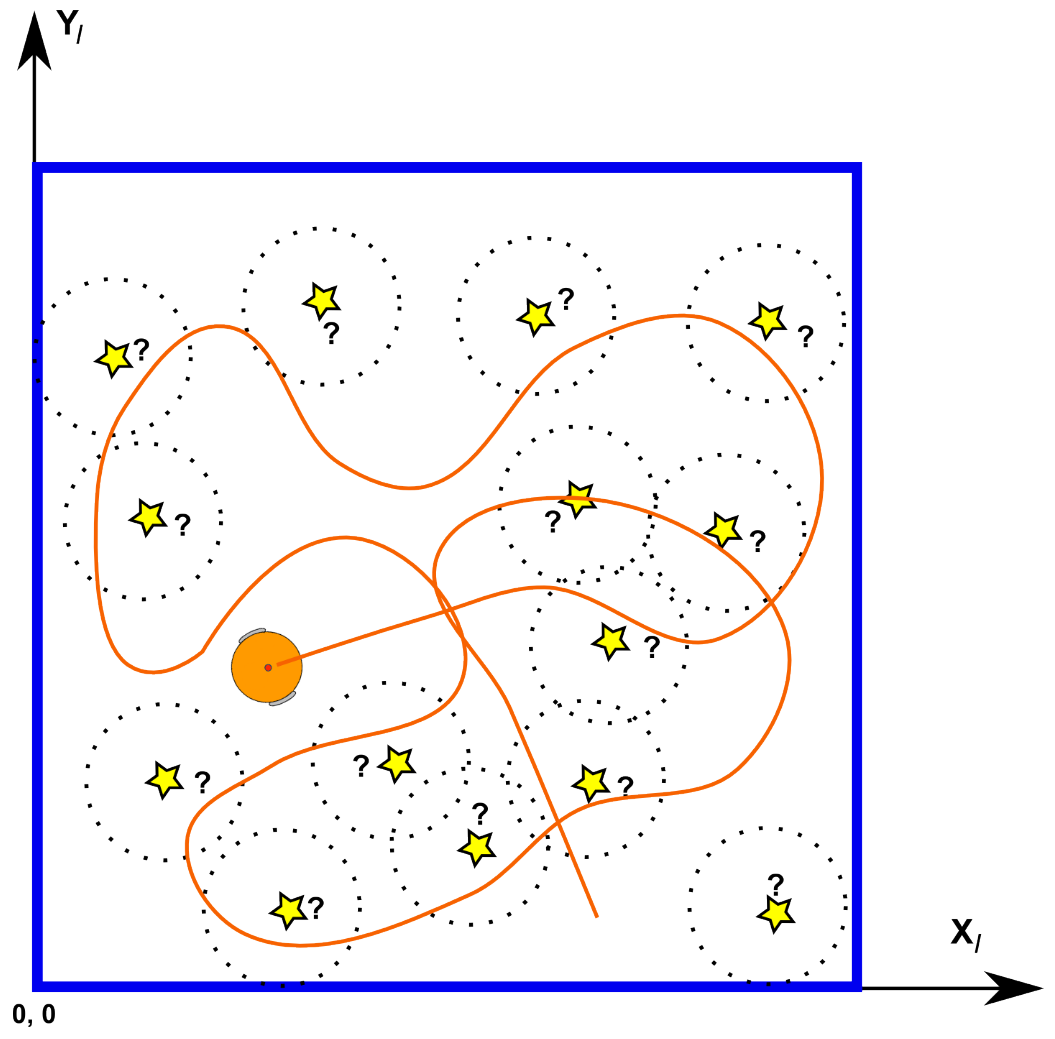
## SYSTEM PROCESS EQUATIONS

- Process equations (motion dynamics): **landmarks do not move**, therefore system's dynamics is the same as when the KF is used to iteratively refine the estimate of a measured quantity/event which is stationary and there is no process error:

$$\boldsymbol{\xi}_{k+1} = \boldsymbol{\xi}_k, \quad V = 0$$

$\rightarrow$ the prediction part of the filter equations is:

$$\hat{\boldsymbol{\xi}}_{k+1|k} = \hat{\boldsymbol{\xi}}_{k|k}, \quad P_{k+1|k} = P_{k|k}$$



## OBSERVATION PROCESS EQUATIONS

Observation model (measurements): as before, the robot uses its on-board sensors to measure the **relative range** $\rho^i$ and **bearing** $\beta^i$, with respect to landmark with identity $i$ when this falls in its sensing range:

$$z_{k+1} = \left[\rho^i \;\; \beta^i\right]^T.$$

$z_{k+1}$ is (possibly) corrupted by **white Gaussian noise** $\boldsymbol{w}_k$. The observation equation is as before, but now the state variables are the $\boldsymbol{\lambda}$s, while robot's pose $\left[x_k \;\; y_k \;\; \theta_k\right]^T$ is a parameter vector:

$$z_{k+1} = \boldsymbol{\ell}(\boldsymbol{\lambda}_k, \boldsymbol{w}_k; x_k, y_k, \theta_k) = \begin{bmatrix} \sqrt{(\lambda^i_{kx} - x_k)^2 + (\lambda^i_{ky} - y_k)^2} \\ \arctan\left((\lambda^i_{ky} - y_k)/(\lambda^i_{kx} - x_k)\right) - \theta_k \end{bmatrix} + \begin{bmatrix} w^{\rho}_k \\ w^{\beta}_k \end{bmatrix}$$

- As before, measurement noises in range and bearing are assumed uncorrelated and Gaussian:

$$\boldsymbol{w} = \begin{bmatrix} \boldsymbol{w}_\rho \\ \boldsymbol{w}_\beta \end{bmatrix}^T \sim N(0, \boldsymbol{W}), \quad \boldsymbol{W} = \begin{bmatrix} \sigma^2_\rho & 0 \\ 0 & \sigma^2_\beta \end{bmatrix}$$

Non linear equations: Linearization is required for an EKF

# JACOBIANS FOR LINEARIZATION

- The Jacobian of the non-linear function $\boldsymbol{\ell}_k$ is computed at the mean of the Gaussian measurement noise ($\boldsymbol{w} = \mathbf{0}$) and at the current state estimate $\hat{\boldsymbol{\xi}}_{k+1|k}$ (which corresponds to the estimated mean of the Gaussian distribution of the landmarks' positions):
- The vector function $\boldsymbol{\ell}$, with variables $\boldsymbol{\lambda}$s and $\boldsymbol{w}$, is written in its two components, $\boldsymbol{\ell}_k = \begin{bmatrix} h_{k\rho} & h_{k\beta} \end{bmatrix}^T$, where $\boldsymbol{\lambda}^i_k$ is the position estimate of the currently observed landmark:

$$\begin{aligned} h_{k\rho} &= \sqrt{(\lambda^i_{kx} - x_k)^2 + (\lambda^i_{ky} - y_k)^2} + w^\rho_k \\ h_{k\beta} &= \text{arctan}\left((\lambda^i_{ky} - y_k)/(\lambda^i_{kx} - x_k)\right) - \theta_k + w^\beta_k \end{aligned}$$

The Jacobian matrix of $\boldsymbol{\ell}_k$ is therefore:

$$\boldsymbol{L}_k(\boldsymbol{\lambda}^1, \ldots, \boldsymbol{\lambda}^M, w^\rho_k, w^\beta_k) = \begin{bmatrix} \frac{\partial h_{k\rho}}{\partial \lambda^1_{kx}} & \frac{\partial h_{k\rho}}{\partial \lambda^1_{ky}} & \frac{\partial h_{k\rho}}{\partial \lambda^2_{kx}} & \frac{\partial h_{k\rho}}{\partial \lambda^2_{ky}} & \cdots & \frac{\partial h_{k\rho}}{\partial \lambda^M_{kx}} & \frac{\partial h_{k\rho}}{\partial \lambda^M_{ky}} & \frac{\partial h_{k\rho}}{\partial w^\rho_k} & \frac{\partial h_{k\rho}}{\partial w^\beta_k} \\ \frac{\partial h_{k\beta}}{\partial \lambda^1_{kx}} & \frac{\partial h_{k\beta}}{\partial \lambda^1_{ky}} & \frac{\partial h_{k\beta}}{\partial \lambda^2_{kx}} & \frac{\partial h_{k\beta}}{\partial \lambda^2_{ky}} & \cdots & \frac{\partial h_{k\beta}}{\partial \lambda^M_{kx}} & \frac{\partial h_{k\beta}}{\partial \lambda^M_{ky}} & \frac{\partial h_{k\beta}}{\partial w^\rho_k} & \frac{\partial h_{k\beta}}{\partial w^\beta_k} \end{bmatrix} = \begin{bmatrix} \boldsymbol{L}_{k\xi} & \boldsymbol{L}_{kw} \end{bmatrix}$$

$$\boldsymbol{L}_{k\xi} = \begin{bmatrix} 0 & 0 & \ldots & \frac{\lambda^i_{kx} - x_k}{r^i_k} & \frac{\lambda^i_{ky} - y_k}{r^i_k} & \ldots & 0 & 0 \\ 0 & 0 & \ldots & -\frac{\lambda^i_{ky} - y_k}{(r^i_k)^2} & \frac{\lambda^i_{kx} - x_k}{(r^i_k)^2} & \ldots & 0 & 0 \end{bmatrix}_{\hat{\boldsymbol{\xi}}_{k+1|k}, \boldsymbol{w}=0} \qquad \boldsymbol{L}_{kw} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$r^i_k$ is the predicted distance of landmark $i$ from robot position: $r^i_k = \sqrt{(\lambda^i_{kx} - x_k)^2 + (\lambda^i_{ky} - y_k)^2}$

## FILTER EQUATIONS

**Prediction update**
(at every time step $k$)

$$\begin{cases} \hat{\boldsymbol{\xi}}_{k+1|k} = \hat{\boldsymbol{\xi}}_k & \text{(State prediction)} \\ \boldsymbol{P}_{k+1|k} = \boldsymbol{P}_k & \text{(Covariance prediction)} \end{cases}$$

**Measurement correction**
(every time a landmark $i$ is observed)

$$\begin{cases} \hat{\boldsymbol{\xi}}_{k+1} = \hat{\boldsymbol{\xi}}_{k+1|k} + \boldsymbol{G}_{k+1}(z_{k+1} - \boldsymbol{\ell}_k(\hat{\boldsymbol{\lambda}}_{k+1|k}, \boldsymbol{0}; x_k, y_k, \theta_k)) & \text{(State update)} \\ \boldsymbol{P}_{k+1} = \boldsymbol{P}_{k+1|k} - \boldsymbol{G}_{k+1}\boldsymbol{L}_{k\xi}\boldsymbol{P}_{k+1|k} & \text{(Covariance update)} \\ \boldsymbol{G}_{k+1} = \boldsymbol{P}_{k+1|k}\boldsymbol{L}_{k\xi}{}^T\boldsymbol{S}_{k+1}^{-1} & \text{(Kalman gain)} \\ \boldsymbol{S}_{k+1} = \boldsymbol{L}_{k\xi}\boldsymbol{P}_{k+1|k}\boldsymbol{L}_{k\xi}{}^T + \boldsymbol{L}_{kw}\boldsymbol{W}_{k+1}\boldsymbol{L}_{kw}{}^T & \end{cases}$$

## FILTER EQUATIONS

- The innovation term ($\boldsymbol{\epsilon}_{k+1}$ is a $2 \times 1$ matrix):

$$\boldsymbol{\epsilon}_{k+1} = \boldsymbol{z}_{k+1} - \boldsymbol{\ell}_k(\hat{\boldsymbol{\lambda}}_{k+1|k}, \mathbf{0}; \boldsymbol{\xi}_{R_k}) = \begin{bmatrix} \rho^i_{k+1} \\ \beta^i_{k+1} \end{bmatrix} - \begin{bmatrix} \sqrt{(\hat{\lambda}^i_{kx} - x_k)^2 + (\hat{\lambda}^i_{ky} - y_k)^2} \\ \arctan\left((\hat{\lambda}^i_{ky} - y_k)/(\hat{\lambda}^i_{kx} - x_k)\right) - \theta_k \end{bmatrix}$$

- The state update ($\boldsymbol{G}_{k+1}$ is a $2M \times 2$ matrix):

$$\hat{\boldsymbol{\xi}}_{k+1} = \begin{bmatrix} \hat{\lambda}^1_{kx} & \hat{\lambda}^1_{ky} & \dots & \dots & \hat{\lambda}^M_{kx} & \hat{\lambda}^M_{ky} \end{bmatrix}^T + \boldsymbol{G}_{k+1} \begin{bmatrix} \rho^i_{k+1} - \sqrt{(\hat{\lambda}^i_{kx} - x_k)^2 + (\hat{\lambda}^i_{ky} - y_k)^2} \\ \beta^i_{k+1} - \arctan\left((\hat{\lambda}^i_{ky} - y_k)/(\hat{\lambda}^i_{kx} - x_k)\right) - \theta_k \end{bmatrix}$$

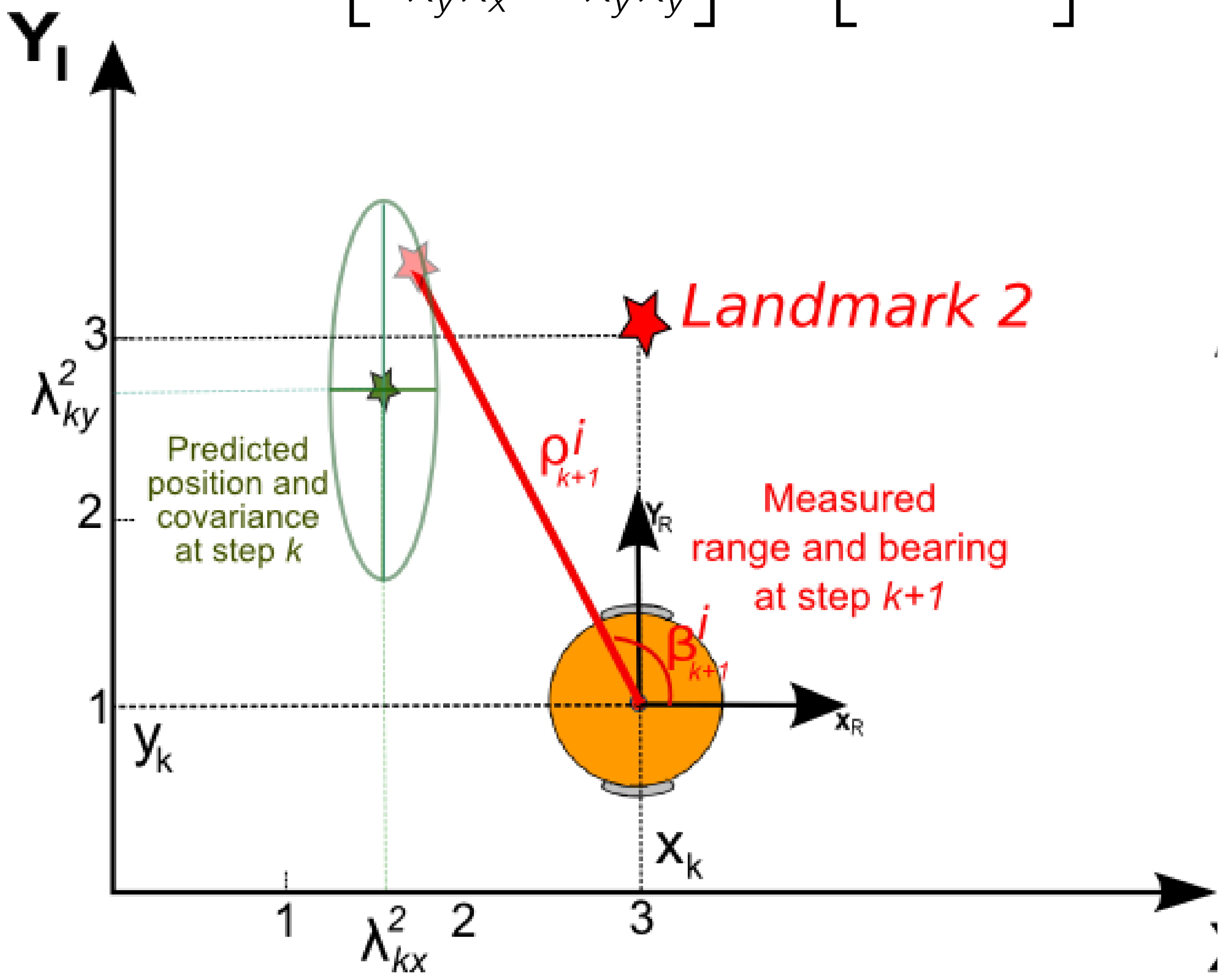
- The covariance matrix:

$$\boldsymbol{P}_{k+1} = \boldsymbol{P}_{k+1|k} - \boldsymbol{G}_{k+1} \begin{bmatrix} 0 & 0 & \dots & \dfrac{\hat{\lambda}^i_{kx} - x_k}{\hat{r}^i_k} & \dfrac{\hat{\lambda}^i_{ky} - y_k}{\hat{r}^i_k} & \dots 0 & 0 \\ 0 & 0 & \dots & -\dfrac{\hat{\lambda}^i_{ky} - y_k}{(\hat{r}^i_k)^2} & \dfrac{\hat{\lambda}^i_{kx} - x_k}{(\hat{r}^i_k)^2} & \dots 0 & 0 \end{bmatrix} \begin{bmatrix} \hat{\Sigma}_{\boldsymbol{\lambda}^1\boldsymbol{\lambda}^1} & \hat{\Sigma}_{\boldsymbol{\lambda}^1\boldsymbol{\lambda}^2} & \dots & \hat{\Sigma}_{\boldsymbol{\lambda}^1\boldsymbol{\lambda}^M} \\ \hat{\Sigma}_{\boldsymbol{\lambda}^2\boldsymbol{\lambda}^1} & \hat{\Sigma}_{\boldsymbol{\lambda}^2\boldsymbol{\lambda}^2} & \dots & \hat{\Sigma}_{\boldsymbol{\lambda}^2\boldsymbol{\lambda}^M} \\ \dots & \dots & \dots & \dots \\ \hat{\Sigma}_{\boldsymbol{\lambda}^M\boldsymbol{\lambda}^1} & \hat{\Sigma}_{\boldsymbol{\lambda}^M\boldsymbol{\lambda}^2} & \dots & \hat{\Sigma}_{\boldsymbol{\lambda}^M\boldsymbol{\lambda}^M} \end{bmatrix}_{\hat{\boldsymbol{\lambda}}_{k+1|k}}$$

## EXAMPLE

- At step $k+1$ the robot detects landmark 2 at a relative range of 2.5m and a relative angle of $130^o$, that is, $z_{k+1} = \begin{bmatrix}2.5 & 130\end{bmatrix}^T$;
- Robot's known pose is $\boldsymbol{\xi}_{k+1} = \begin{bmatrix}3 & 1 & 0\end{bmatrix}^T$
- The true (unknown) position of landmark 2 is $\boldsymbol{\lambda}^2 = \begin{bmatrix}3 & 3\end{bmatrix}^T$
- Based on current filter status, the predicted position of landmark 1 is: $\hat{\boldsymbol{\lambda}}^2_k = \begin{bmatrix}1.55 & 2.65\end{bmatrix}^T$
- The covariance sub-matrix quantifying the estimated error in landmark position at step $k$ is:

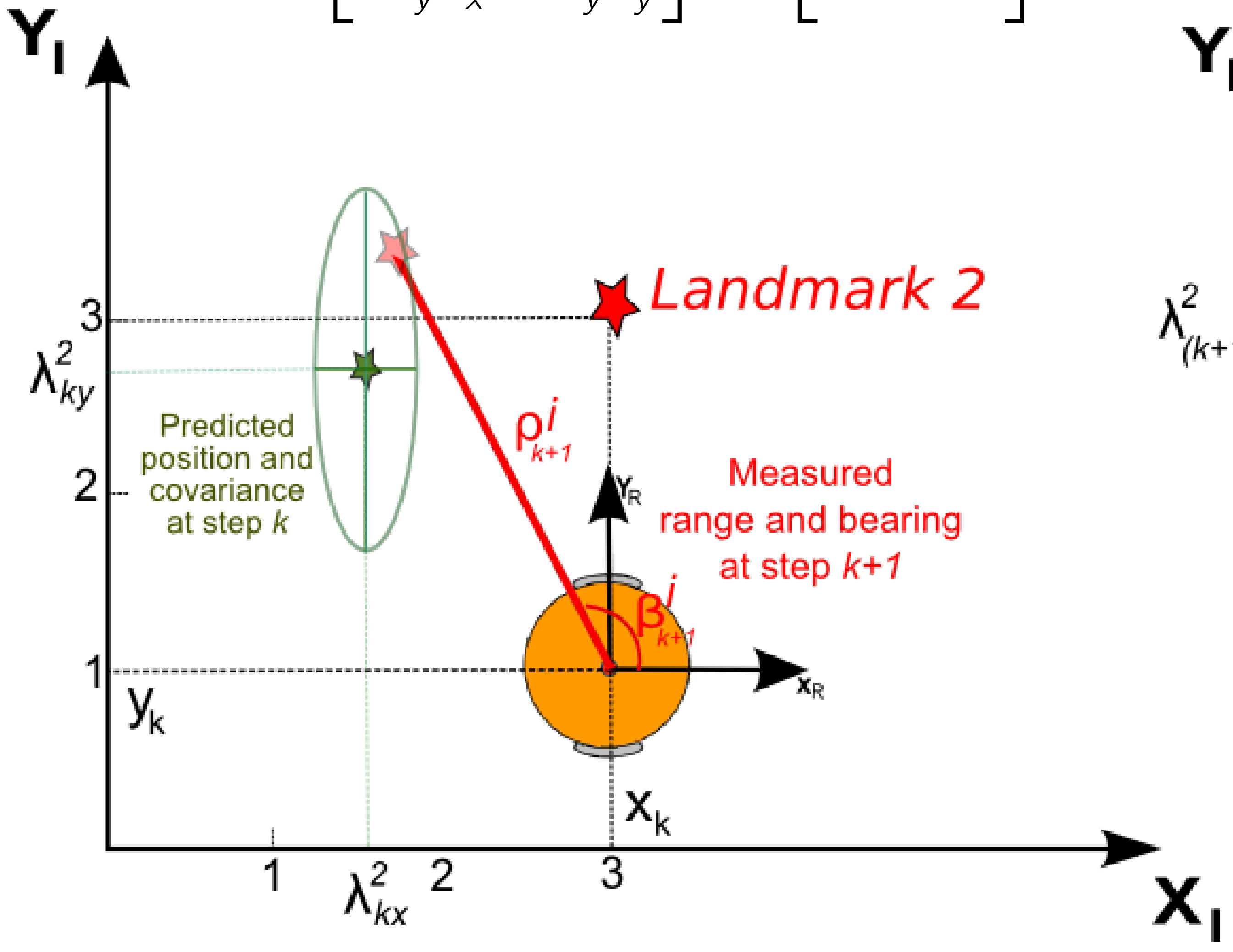
$$\Sigma_{\boldsymbol{\lambda}^2\boldsymbol{\lambda}^2} = \begin{bmatrix}\sigma_{\lambda^2_x\lambda^2_x} & \sigma_{\lambda^2_x\lambda^2_y} \\ \sigma_{\lambda^2_y\lambda^2_x} & \sigma_{\lambda^2_y\lambda^2_y}\end{bmatrix} = \begin{bmatrix}0.5 & 0 \\ 0 & 1.8\end{bmatrix}$$

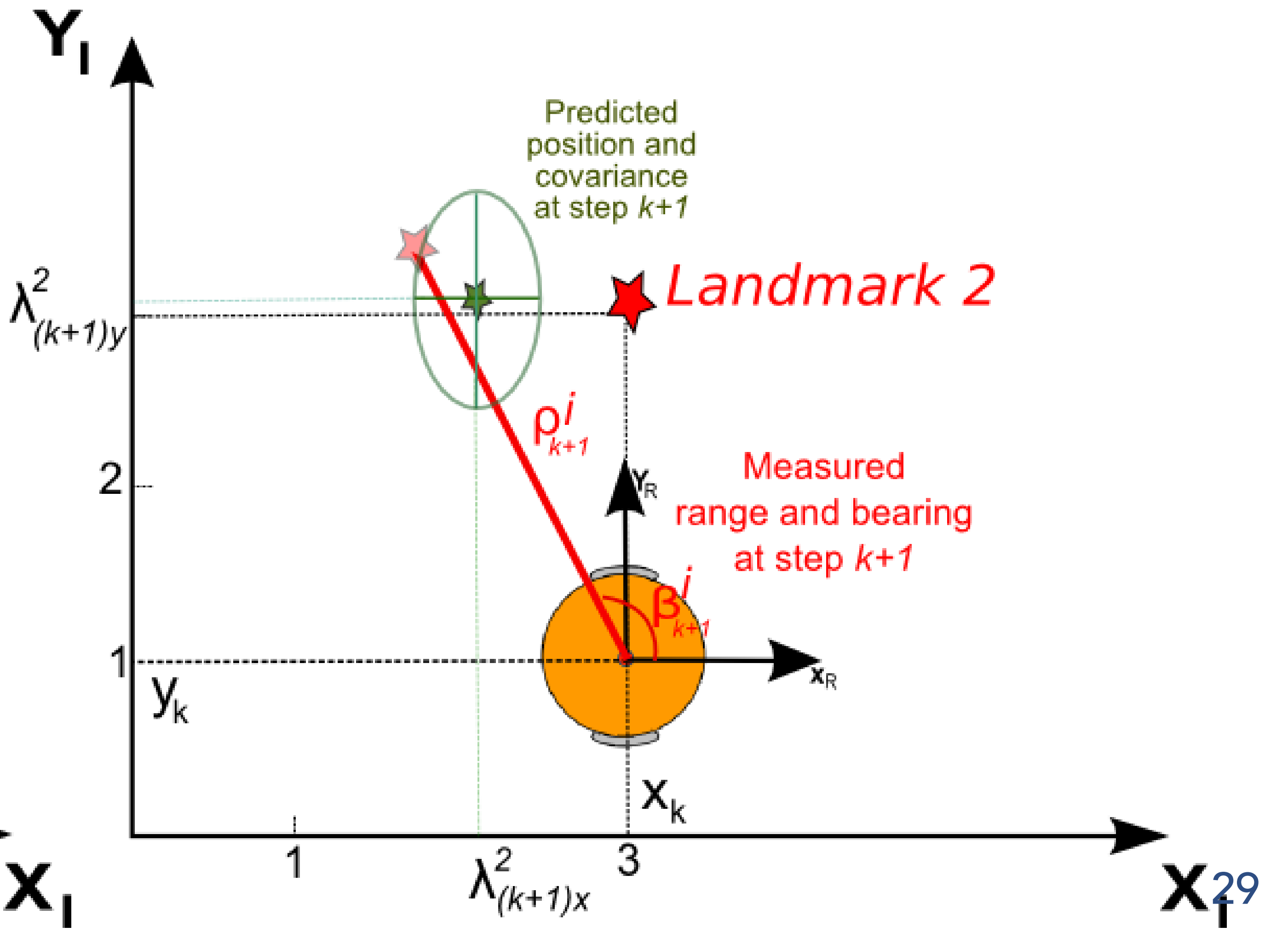


## EXAMPLE

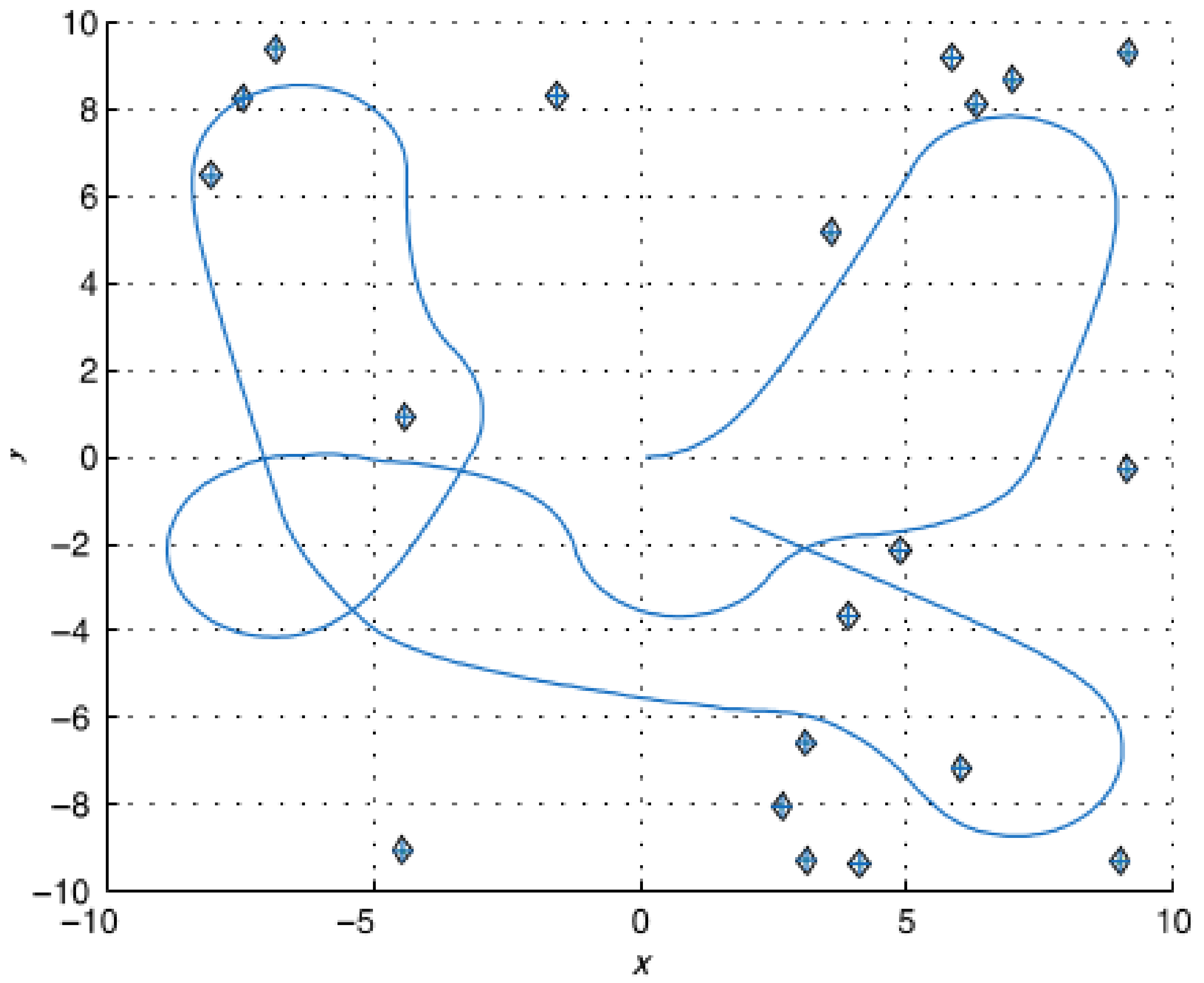
- At step $k+1$ the robot detects landmark 2 at a relative range of 2.5m and a relative angle of $130^o$, that is, $z_{k+1} = \begin{bmatrix}2.5 & 130\end{bmatrix}^T$;
- Robot's known pose is $\boldsymbol{\xi}_{k+1} = \begin{bmatrix}3 & 1 & 0\end{bmatrix}^T$
- The true (unknown) position of landmark 2 is $\boldsymbol{\lambda}^2 = \begin{bmatrix}3 & 3\end{bmatrix}^T$
- Based on current filter status, the predicted position of landmark 1 is: $\hat{\boldsymbol{\lambda}}^2_k = \begin{bmatrix}1.55 & 2.65\end{bmatrix}^T$
- The covariance sub-matrix quantifying the estimated error in landmark position at step $k$ is:

$$\Sigma_{\boldsymbol{\lambda}^2\boldsymbol{\lambda}^2} = \begin{bmatrix}\sigma_{\lambda^2_x\lambda^2_x} & \sigma_{\lambda^2_x\lambda^2_y} \\ \sigma_{\lambda^2_y\lambda^2_x} & \sigma_{\lambda^2_y\lambda^2_y}\end{bmatrix} = \begin{bmatrix}0.5 & 0 \\ 0 & 1.8\end{bmatrix}$$



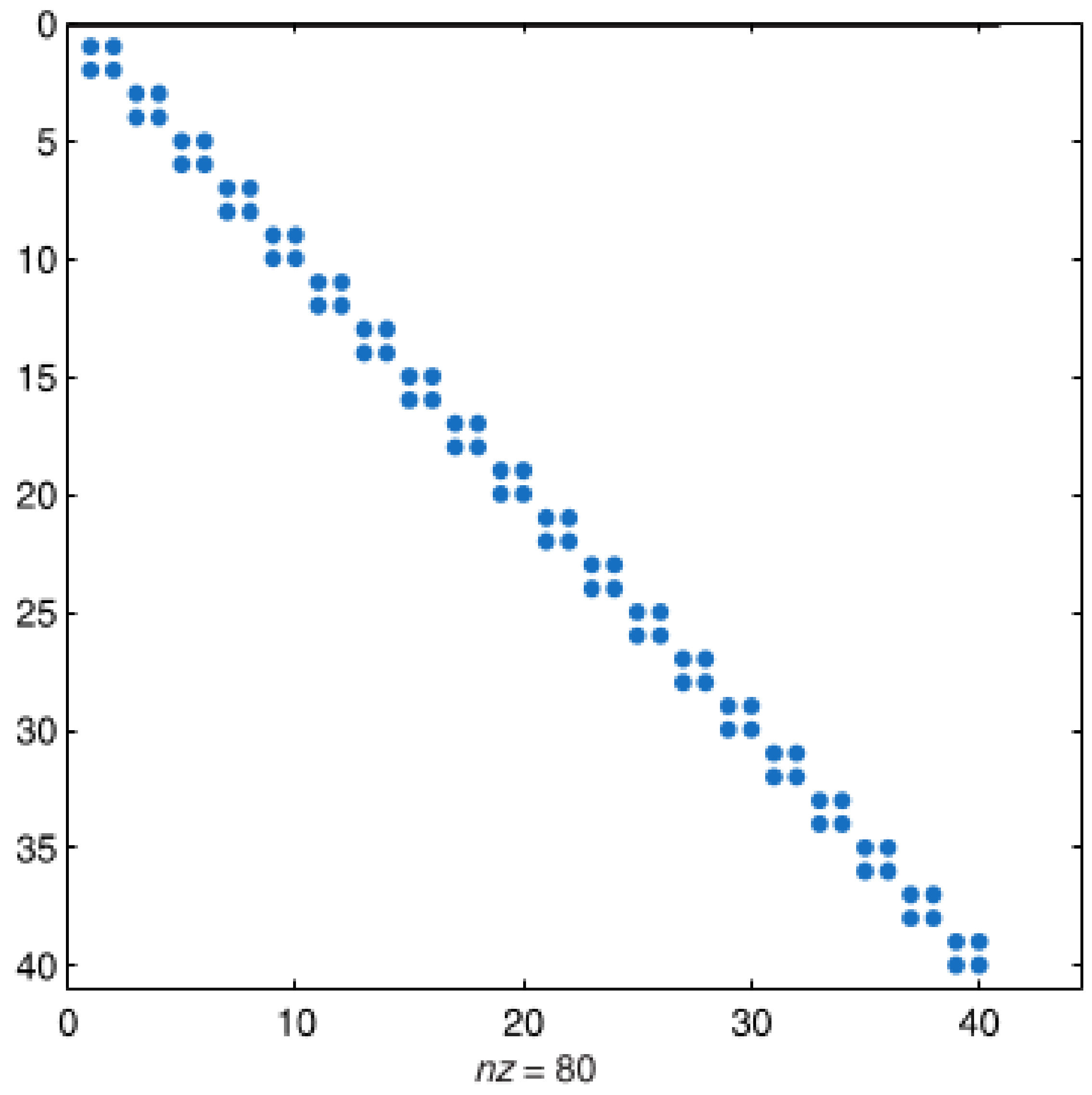


# SIMULATION RESULTS



- $n = 20$ landmarks are randomly deployed in a squared environment of $20 \times 20$ m$^2$
- $\sigma_\rho = 0.1$ m, $\sigma_\beta = 1^o$
- Total of 1000 steps (about 40–70 measures per landmark)
- Axes of the $5\sigma$ confidence ellipses are shown at each landmark point

# SIMULATION RESULTS



- The resulting covariance matrix (40× 40)
- Block diagonal structure: each set of 4 points represent the values of the covariance of the position of a map landmark: $\Sigma_{\boldsymbol{\lambda}^n \boldsymbol{\lambda}^n}$
- All the non-diagonal entries are zero: positions of any pair of landmarks $n$ and $j$ are uncorrelated, which is expected since observing landmark $n$ provides no new information about landmark $j \neq n$