Simultaneous Localisation and Mapping

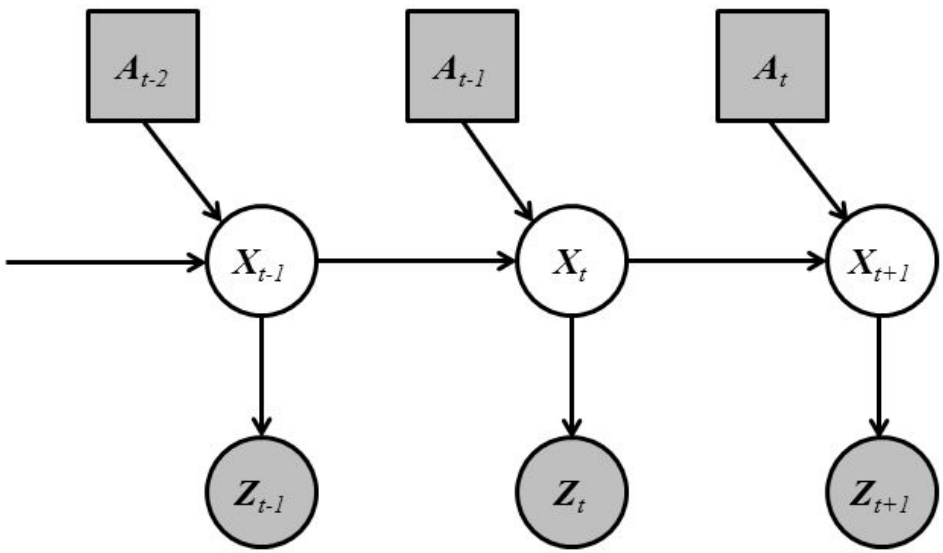
3007/7059 Artificial Intelligence

School of Computer Science
The University of Adelaide

Robot localisation – recap

Localisation can be cast as the problem of estimating the **current state** $\mathbf{X}_t$ of the robot, based on

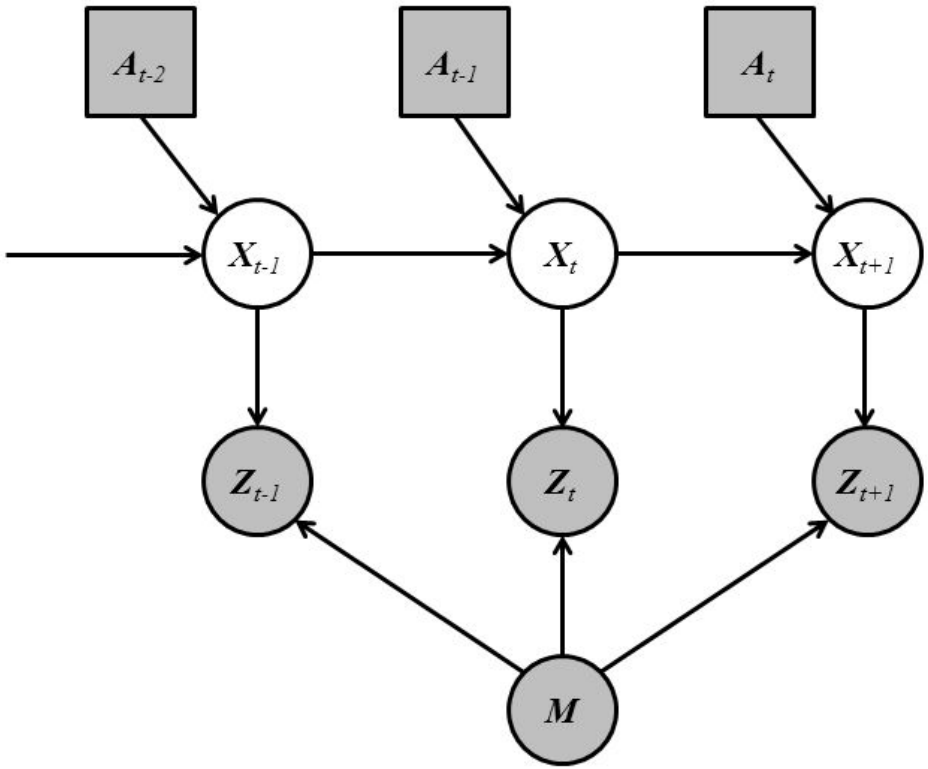
- the action $\mathbf{A}_{t-1}$ executed at the previous time step.
- and the observations of the environment $\mathbf{Z}_t$ obtained at the present time step.



Shaded boxes/circles mean the values of the associated variables are known. Unknown/latent variables are unshaded.

Robot localisation – recap (cont.)

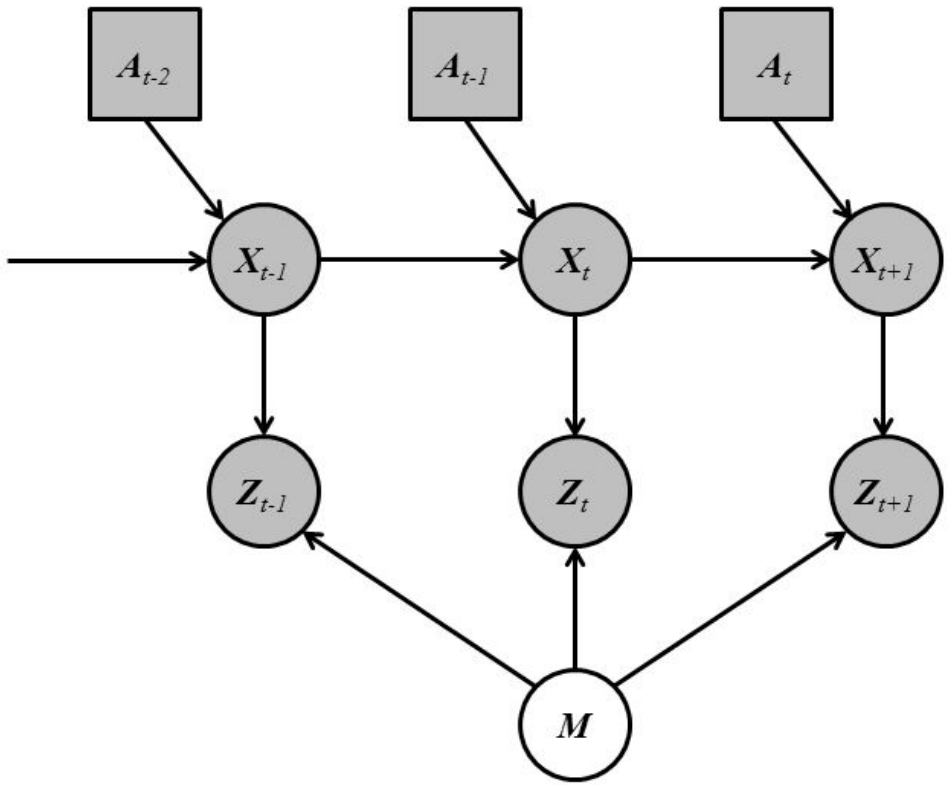
More specifically, we have the following, in which we know the actions $\mathbf{A}_t$, the observations $\mathbf{Z}_t$ and the map $\mathbf{M}$.



and we seek to estimate the robot location $\mathbf{X}_t$.

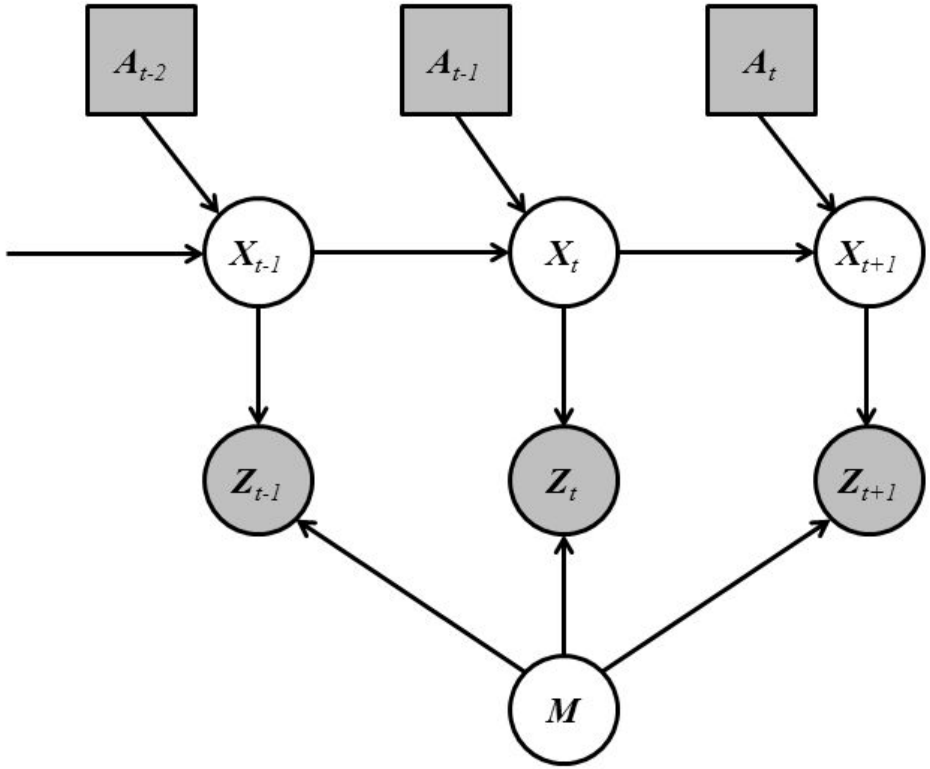
Robot mapping

A dual problem arises if we assume we know our own motion (i.e. $\mathbf{X}_t$) and seek to estimate a map from the measurements of the environment.



Simultaneous Localisation and Mapping

Suppose we want to estimate both:



This problem is known as SLAM or CML (Concurrent Mapping and Localisation).

Is it even possible? How can we do it?

Mathematical statement

We want to find:

$$P(\mathbf{X}_t, \mathbf{M} | \mathbf{Z}_{1:t}, \mathbf{A}_{1:t-1})$$

where $\mathbf{Z}_{1:t} = \{\mathbf{Z}_1, \ldots, \mathbf{Z}_t\}$ and $\mathbf{A}_{1:t-1} = \{\mathbf{A}_1, \ldots, \mathbf{A}_{t-1}\}$. The belief state is a **probability distribution** over the space of states.

Notice that the state space now includes the current robot location *and the map* (which is assumed to be static).

The SLAM filtering task is then **recursive update** of the belief state

$$P(\mathbf{X}_t, \mathbf{M} | \mathbf{Z}_{1:t}, \mathbf{A}_{1:t-1}) = \alpha P(\mathbf{Z}_t | \mathbf{X}_t, \mathbf{M}) \times \int P(\mathbf{X}_t, \mathbf{M} | \mathbf{X}_{t-1}, \mathbf{A}_{t-1}) P(\mathbf{X}_{t-1}, \mathbf{M} | \mathbf{Z}_{1:t-1}, \mathbf{A}_{1:t-2}) d\mathbf{X}_{t-1}$$

α is a normalisation constant.

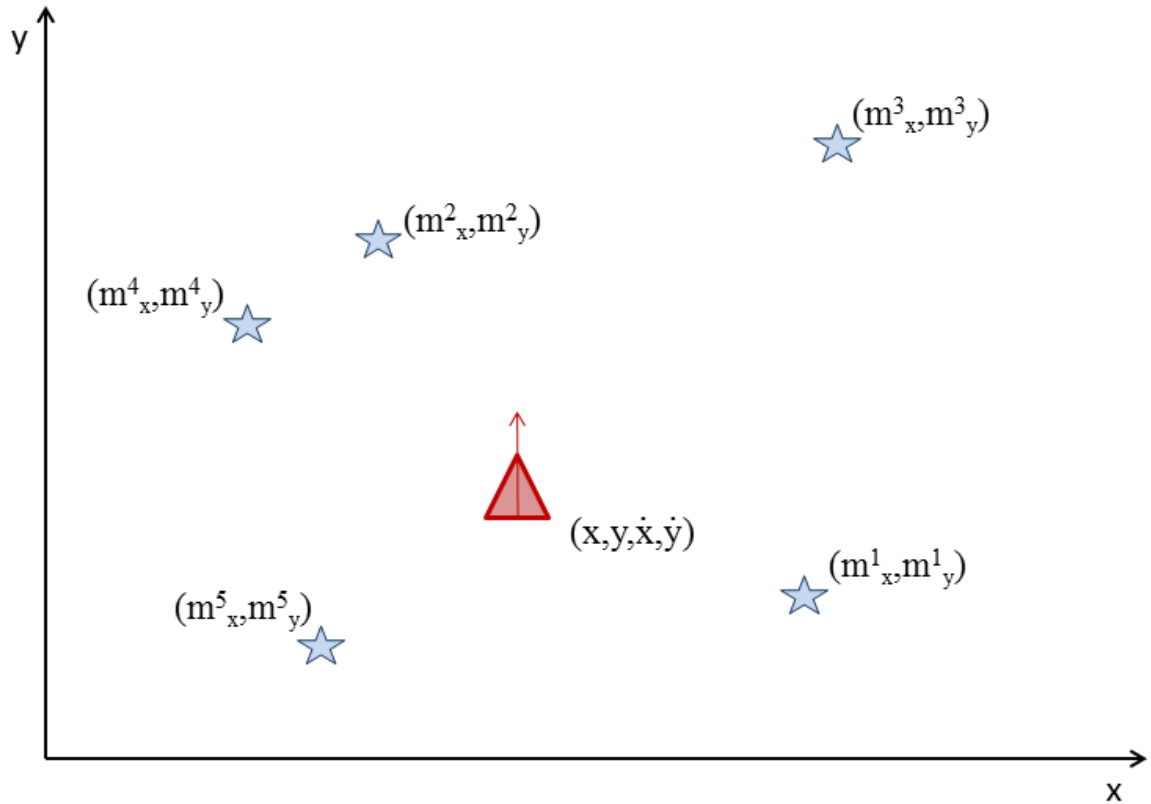
Kalman Filter update equations

Recall that when the belief are Gaussian distributions, a Bayesian update of the joint state is achieved via *linear* update equations known as the Kalman Filter (though we have not discussed their exact form).
These equations update the **mean** and **covariance** of the distribution representing the joint state $\mathbf{X}, \mathbf{M}$.

- The new state estimate is a weighted sum of a prediction and a measurement
- The weights are given by the inverse covariance or *information matrices* of the respective quantities
- The updated information matrix is a sum of the prediction and measurement information matrices
- The usual "textbook" Kalman Filter equations are written in terms of covaraince matrices, not information matrices. They are equivalent.

The map

Typically we would assume that the **map** comprises a set of 3D feature point locations $\mathbf{M}_i$ with each point represented either in 2D (x-y location on the plane) or 3D (x-y-z location in 3D space).



$$\mathbf{M}_i = \begin{pmatrix} m^i_x \\ m^i_y \end{pmatrix}$$

Kalman Filter SLAM

The original SLAM problem was solved using an Extended Kalman Filter (see Smith and Cheeseman, 1990).

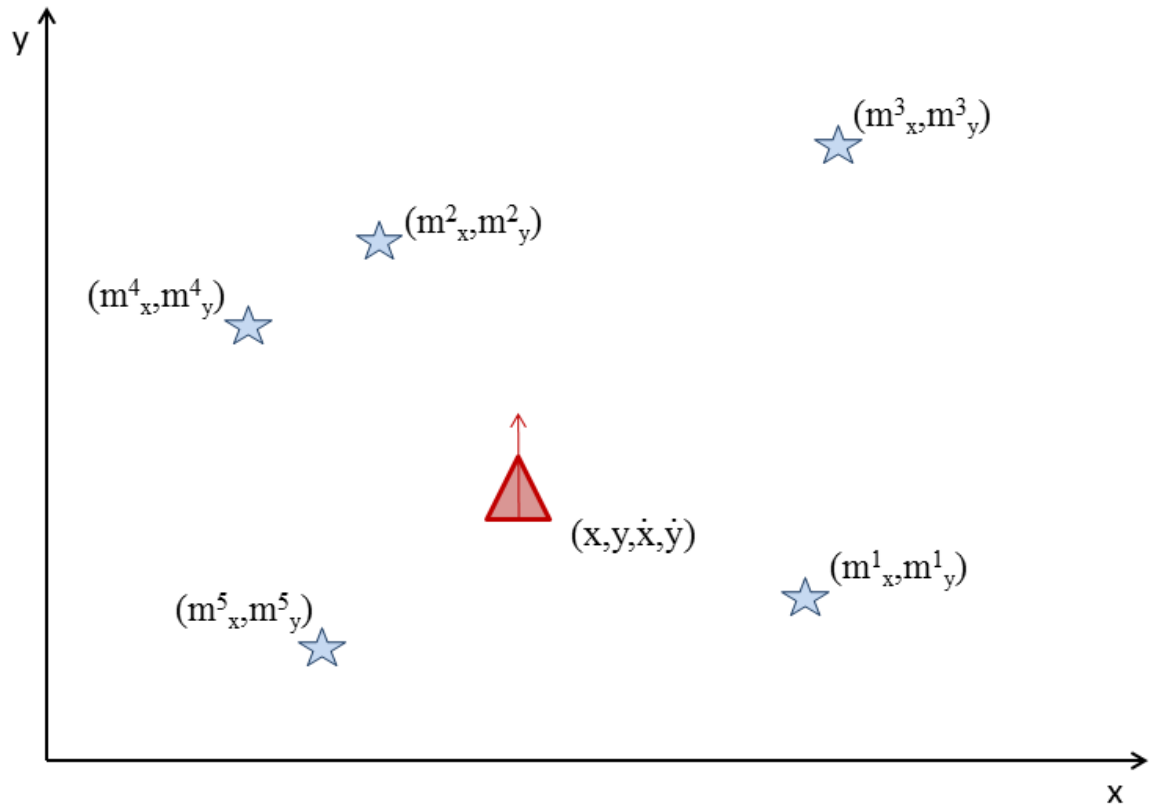
The key idea is to represent the *joint* state of robot position and the map feature locations, and the covariance of that joint state:

$$\mathbf{X} = \begin{bmatrix} \mathbf{X}_{\text{robot}} \\ \mathbf{m}^1 \\ \mathbf{m}^2 \\ \vdots \end{bmatrix} \quad , \quad \mathbf{P} = \begin{bmatrix} \mathbf{P}_{XX} & \mathbf{P}_{Xm^1} & \mathbf{P}_{Xm^2} & \dots \\ \mathbf{P}_{m^1X} & \mathbf{P}_{m^1m^1} & \mathbf{P}_{m^1m^2} & \dots \\ \mathbf{P}_{m^2X} & \mathbf{P}_{m^2m^1} & \mathbf{P}_{m^2m^2} & \dots \\ \vdots & \vdots & \vdots & \end{bmatrix}$$

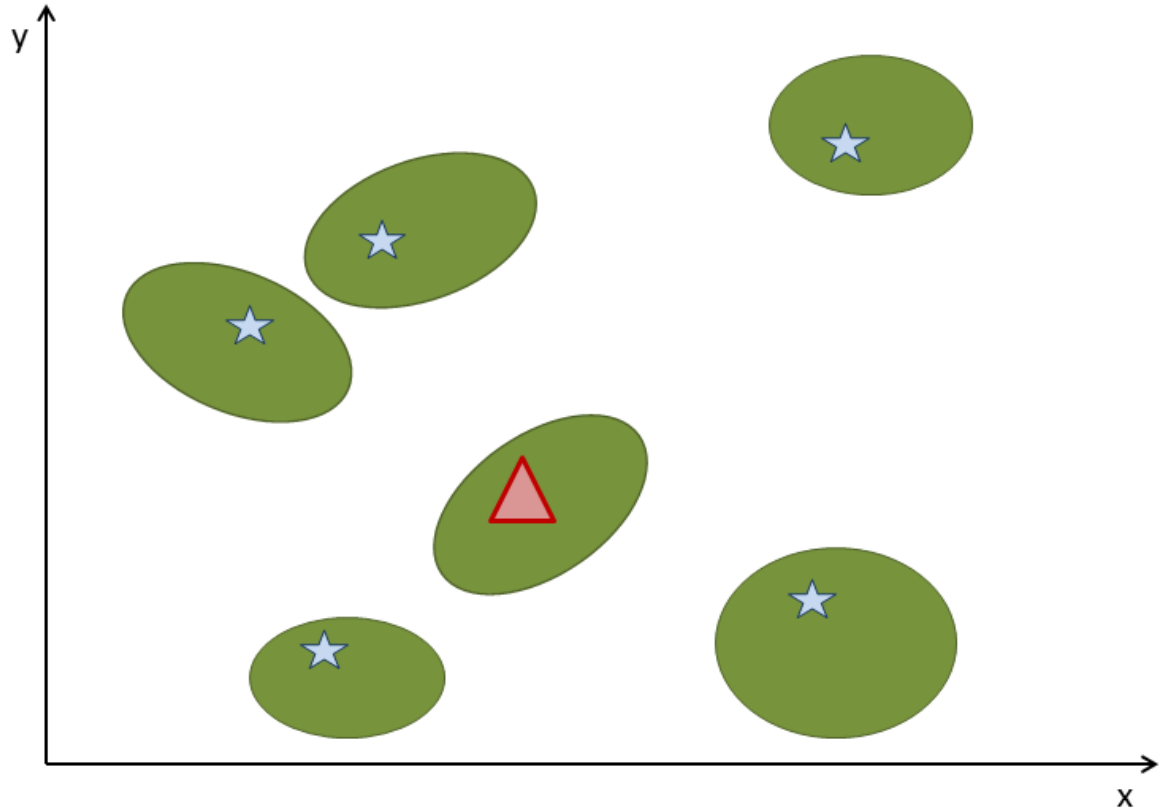
For a robot state of 3 degrees of freedom (translation in the plane and orientation) and feature locations in the plane, the state then has $3 + 2n$ entries, and the covariance is $3 + 2n \times 3 + 2n$, where n is the size of the map.

Simple example

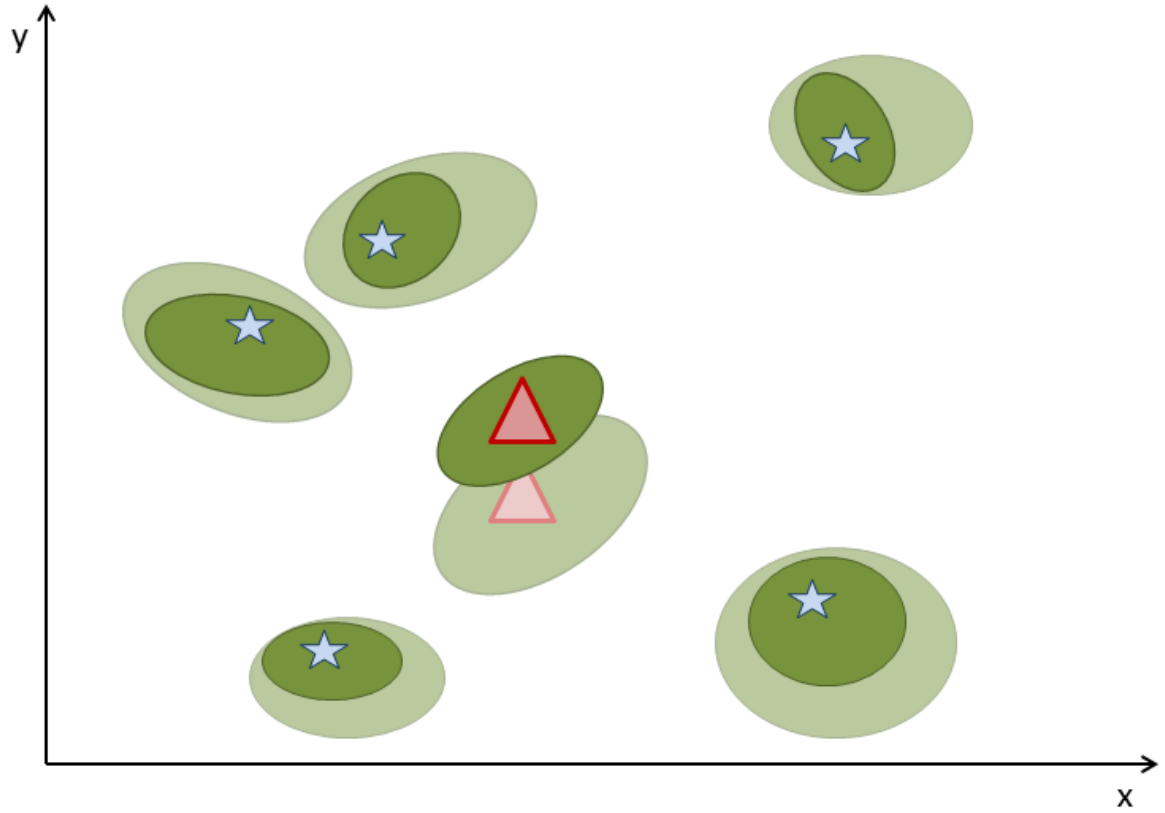
Robot state contains poistion in a plane, and the linear velocity in the plane.
Map comprises 2D point features
Robot measures the location of each feature relative to its *current* position, so the measurement is $\Delta x, \Delta y$.



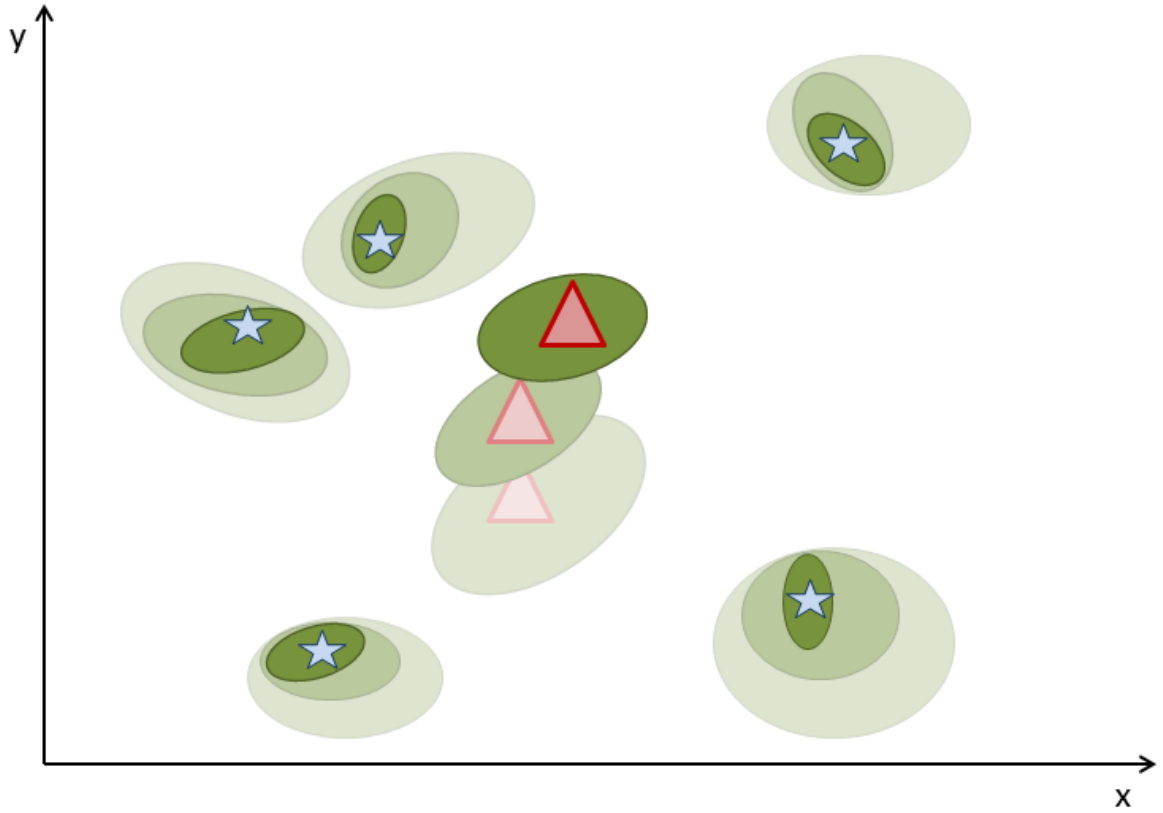
Simple example (cont.)



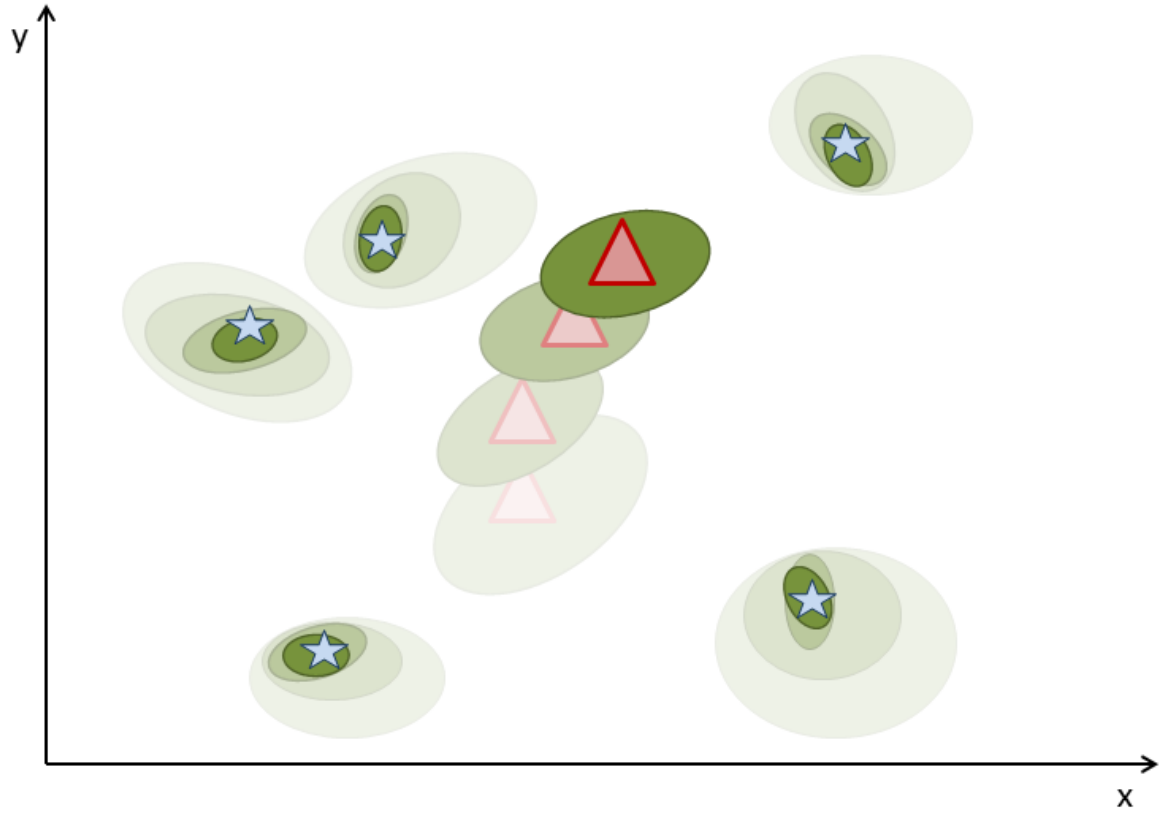
Simple example (cont.)



Simple example (cont.)

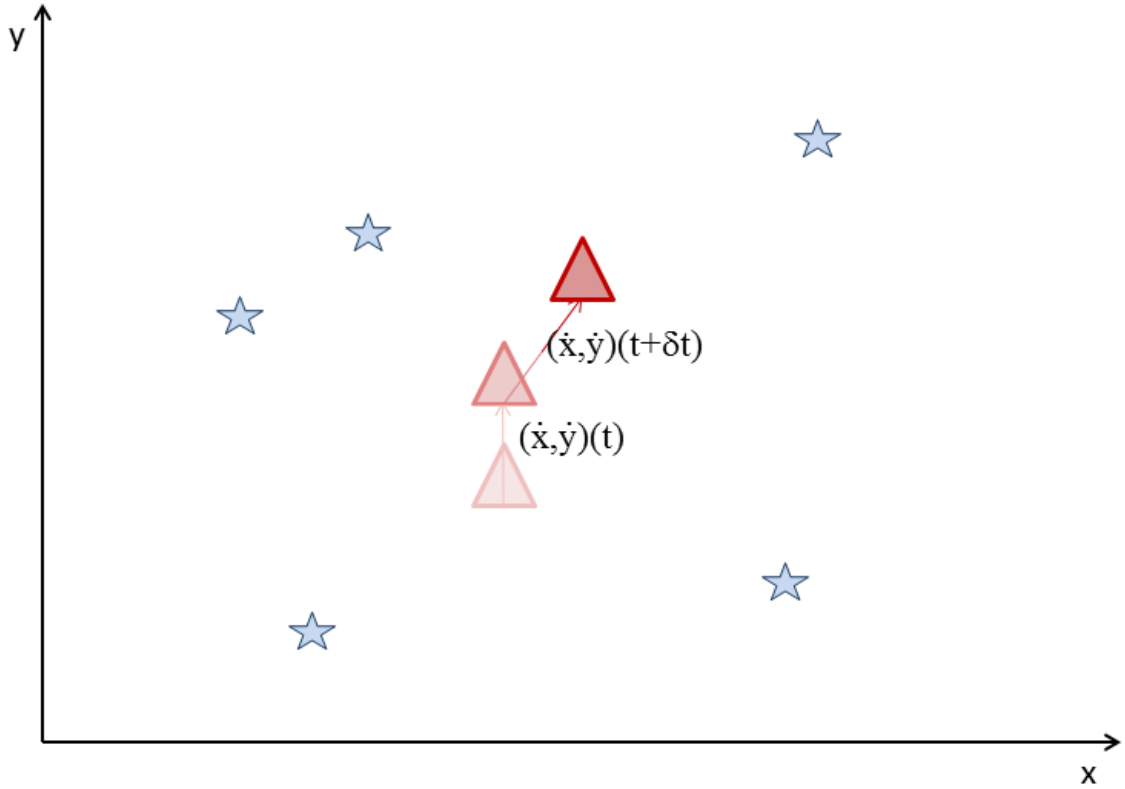


Simple example (cont.)



Simple example (cont.)

State transition:



$$\mathbf{X}_{t+1} = f(\mathbf{X}_t) + \text{unmodelled motion} = \mathbf{F}\mathbf{X}_t + \mathbf{w}$$

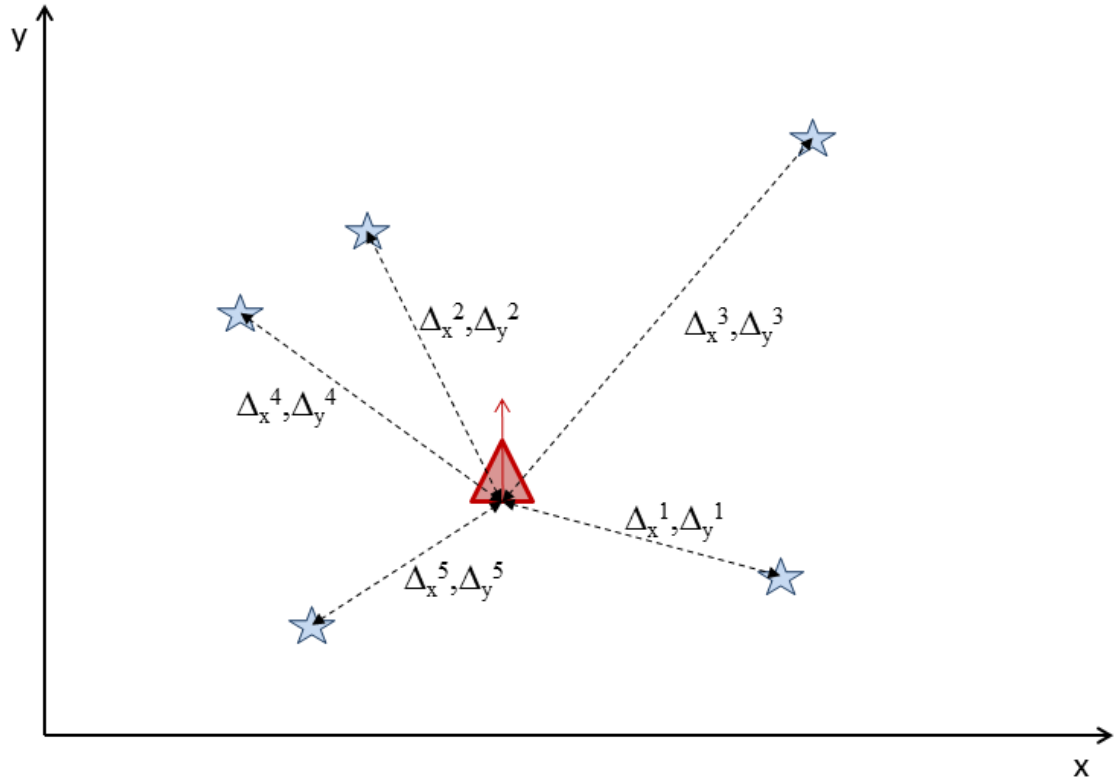
Simple example (cont.)

State transition:

$$\begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \\ m_x^1 \\ m_y^1 \\ \vdots \\ m_x^n \\ m_y^n \end{bmatrix} (t+\delta) = \begin{bmatrix} 1 & 0 & \delta t & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & 0 & \delta t & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & \dots & 0 & 0 \\ \vdots & & & & & & \vdots & & \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \\ m_x^1 \\ m_y^1 \\ \vdots \\ m_x^n \\ m_y^n \end{bmatrix} (t) + \begin{bmatrix} w_x \\ w_y \\ w_{\dot{x}} \\ w_{\dot{y}} \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

Simple example (cont.)

Measurement:



$$\mathbf{Z}_t = h(\mathbf{X}_t) + \text{noise} = \mathbf{H}\mathbf{X}_t + \mathbf{v}$$

Simple example (cont.)

Measurement:

$$\mathbf{Z}(t) = \begin{bmatrix} \Delta x^1 \\ \Delta y^1 \\ \vdots \\ \Delta x^n \\ \Delta y^n \end{bmatrix} (t) = \begin{bmatrix} -1 & 0 & 0 & 0 & 1 & 0 & \dots & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 & \dots & 0 & 0 \\ & \vdots & & & & & \vdots & & \\ -1 & 0 & 0 & 0 & 0 & 0 & \dots & 1 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & \dots & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \\ m_x^1 \\ m_y^1 \\ \vdots \\ m_x^n \\ m_y^n \end{bmatrix} (t)$$

Simple example (cont.)

Kalman filter cycle

Predict: use the state transition equation to get a new state estimate (and its covariance)

Measure: Collate (synchronous) measurements from the environment to form $\mathbf{Z}$, and compute the *innovation*, the difference between the actual values, and those predicted by the measurement equation, given the predicted state $\hat{\mathbf{X}}$.

$$\nu = \mathbf{Z} - h(\hat{\mathbf{X}})$$

Update: Fuse the prediction and innovation using the Kalman Filter update equations to obtain a new posterior mean and covariance.

Notes on EKF SLAM

The estimates of the feature locations become correlated through the uncertainty in the motion of the robot, so the off-diagonal submatrices of $\mathbf{P}$

$$\mathbf{P}_{m^1 m^2}$$

are non-zero. In other words, the state covariance matrix is non-sparse.

The KF (and EKF) are computationally dominated by an inversion of the state covariance matrix. Matrix inversion of a non-sparse matrix is in general $O(n^3)$.

In fact the particular structure of SLAM means that the inversion can be done in $O(n^2)$, so the cost of a SLAM update grows quadratically with the size of the map. This places an upper limit on the size of the map that can be handled in memory and in a computationally timely manner.

Notes on the Extended Kalman Filter

Most real-world Bayesian filtering problems do not have linear state transitions or measurement equations, as we used above, but these are assumptions in the Kalman Filter.

In particular, in *many* real-world SLAM problems:

- the state transition will be a non-linear function of the current state (see previous lecture)

$$\mathbf{X}_{t+1} = f(\mathbf{X}_t) + \mathbf{w}$$

and in *almost all* real-world SLAM problems:

- the measurement will be a non-linear fucntion of the feature locations and robot state

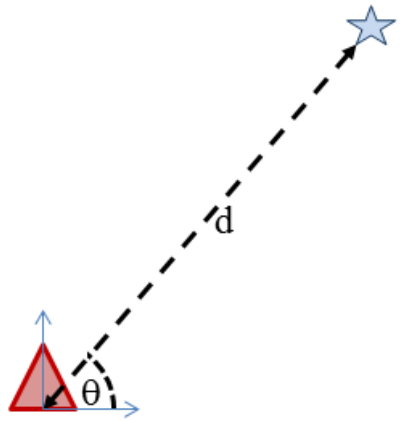
$$\mathbf{Z}_t = h(\mathbf{X}_t) + \mathbf{v}$$

Notes on the Extended Kalman Filter (cont.)

Example, range and bearing:

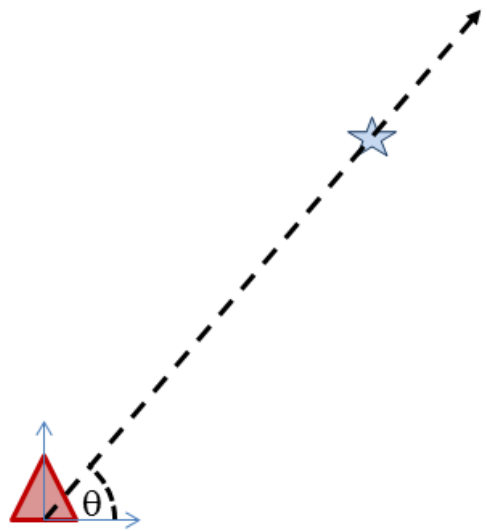
$$d = [(m_x - x)^2 + (m_y - y)^2]$$

$$\theta = \arctan \frac{m_y - y}{m_x - x}$$



Example, bearing-only:

$$\theta = \arctan \frac{m_y - y}{m_x - x}$$



Notes on the Extended Kalman Filter (cont.)

The Extended Kalman Filter addresses this by making a linear (first-order Taylor series) approximation of the non-linear state transition equation $f(.)$ and similarly for the measurement equation $h(.)$.

The Jacobians of $f(.)$ and $h(.)$

$$\nabla f = \left[\frac{\partial f}{\partial x_i}\right], \qquad \nabla h = \left[\frac{\partial h}{\partial x_i}\right]$$

replace transition and measurement matrices $\mathbf{F}$ and $\mathbf{H}$ in the covariance update equations.

The linearisation can lead to overconfidence in the estimates, and to divergence of the filter, so care is required.

Data association

Up to now we have assumed that we know *a priori* which measurement corresponds to which map feature.

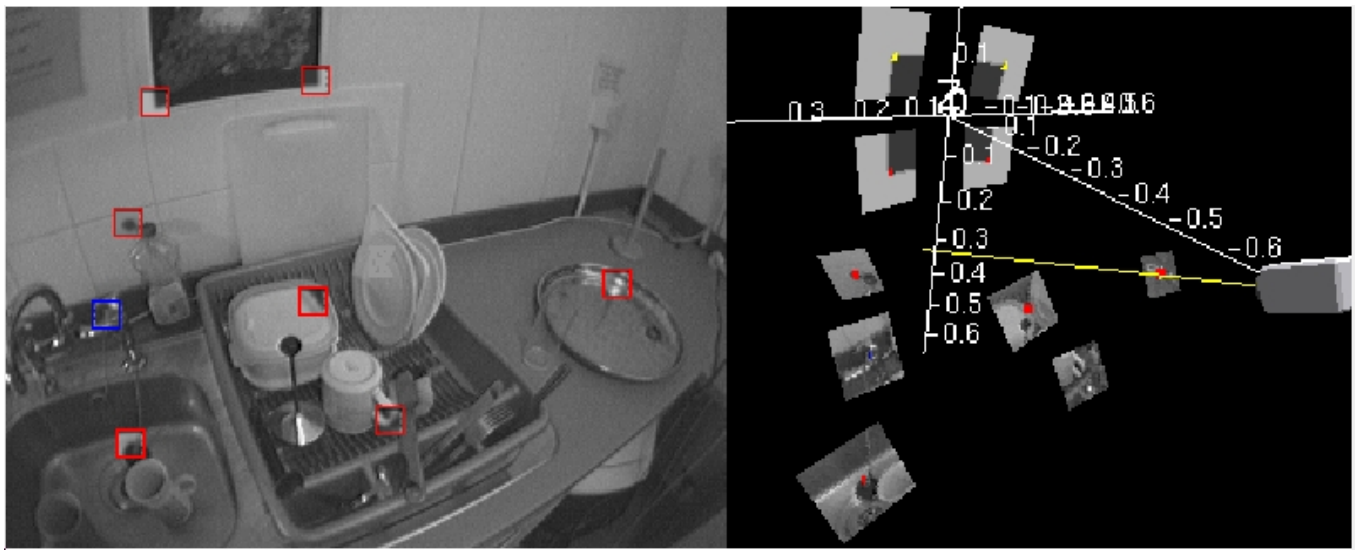
In fact it is rarely the case that we know this, and usually we must solve the **data association** problem.

Simplest solution is to adopt a greedy, nearest neighbour approach

Often the measurement will involve a search (eg find a feature in an image that is the projection of the world feature)
Getting the data association wrong violates a fundamental assumption of the filter and potentially leads to catastrophic results.

Visual SLAM

- Map comprises 3D locations of features and an intensity patch for each feature
- New features added automatically to the map during "idle cycles", by looking for salient regions [Davison, ICCV'03]



Visual SLAM (cont.)

Represent joint distribution over camera and feature positions using a single **multi-variate Gaussian**.

$$\hat{\mathbf{x}} = \begin{pmatrix} \hat{\mathbf{x}}_v \\ \hat{\mathbf{y}}_1 \\ \hat{\mathbf{y}}_2 \\ \vdots \end{pmatrix} \quad , \quad \mathrm{P} = \begin{bmatrix} \mathrm{P}_{xx} & \mathrm{P}_{xy_1} & \mathrm{P}_{xy_2} & \cdots \\ \mathrm{P}_{y_1x} & \mathrm{P}_{y_1y_1} & \mathrm{P}_{y_1y_2} & \cdots \\ \mathrm{P}_{y_2x} & \mathrm{P}_{y_2y_1} & \mathrm{P}_{y_2y_2} & \cdots \\ \vdots & \vdots & \vdots & \end{bmatrix}$$

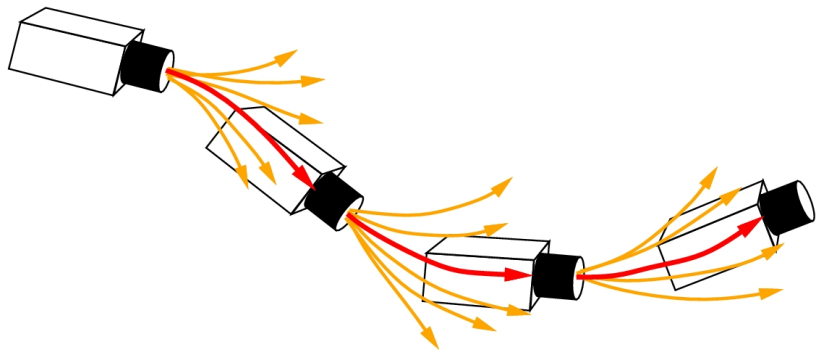
Use Kalman Filter:

predict $\rightarrow$ *measure* $\rightarrow$ *update*

framework to propagate uncertainty, and fuse measurement data

Visual SLAM (cont.)

- Prediction (modelling an agile camera) "Constant **velocity**, **angular velocity**" model $\Rightarrow$ bounded linear and angular acceleration

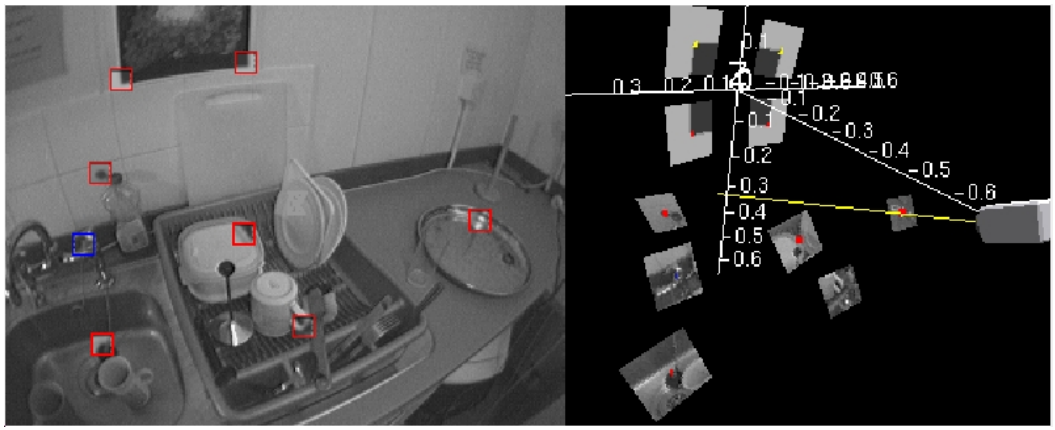


- Measurement involves projection from the 3D world into a 2D image plane and is a function of the pose of the camera, the location of the point, and the internal calibration parameters (focal length, etc) of the camera.

$$\mathbf{z} = h(\mathbf{X}) + \mathbf{v}$$

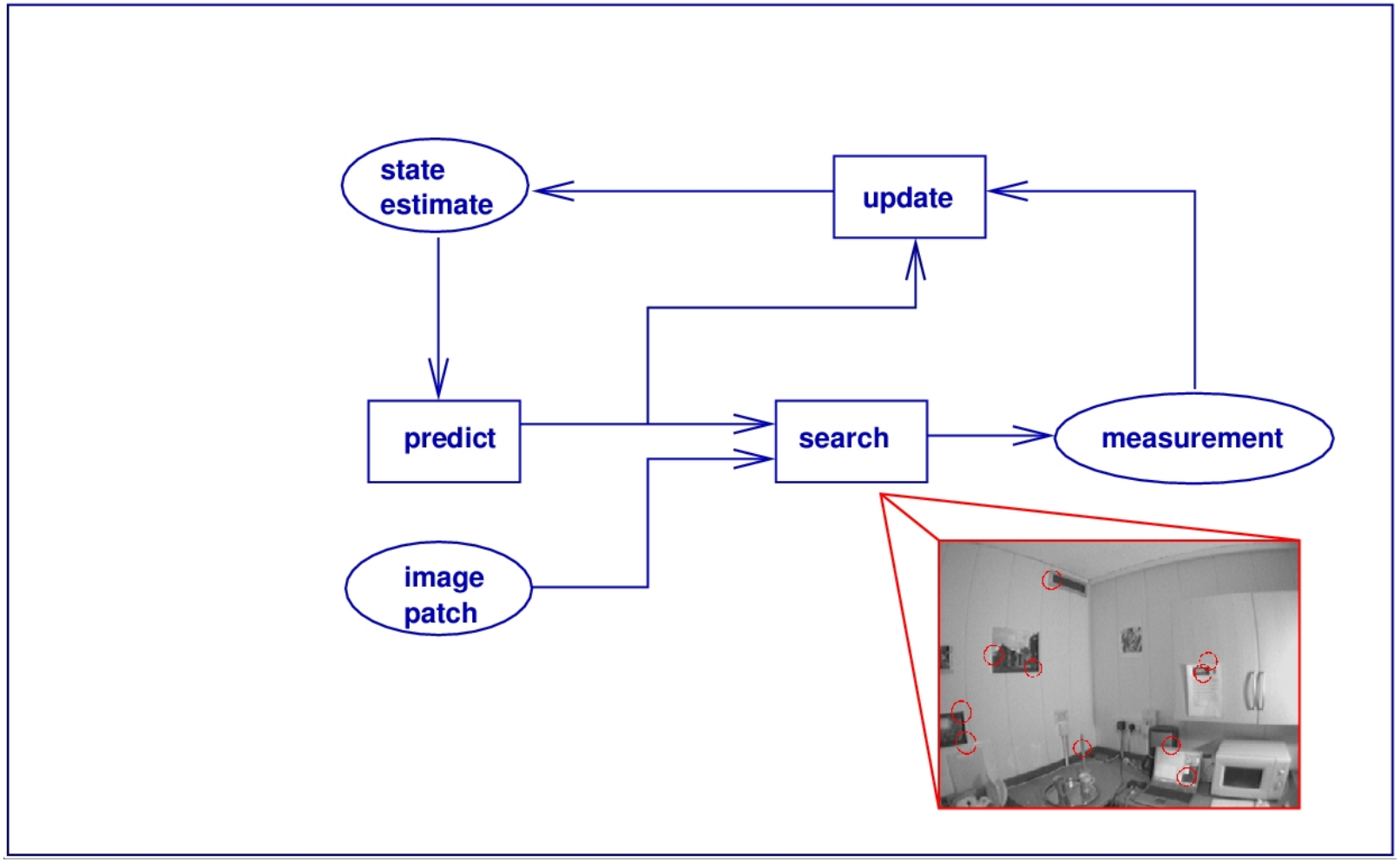
It is *non-linear*, so we use **Extended Kalman Filter**

Real-time, robust operation

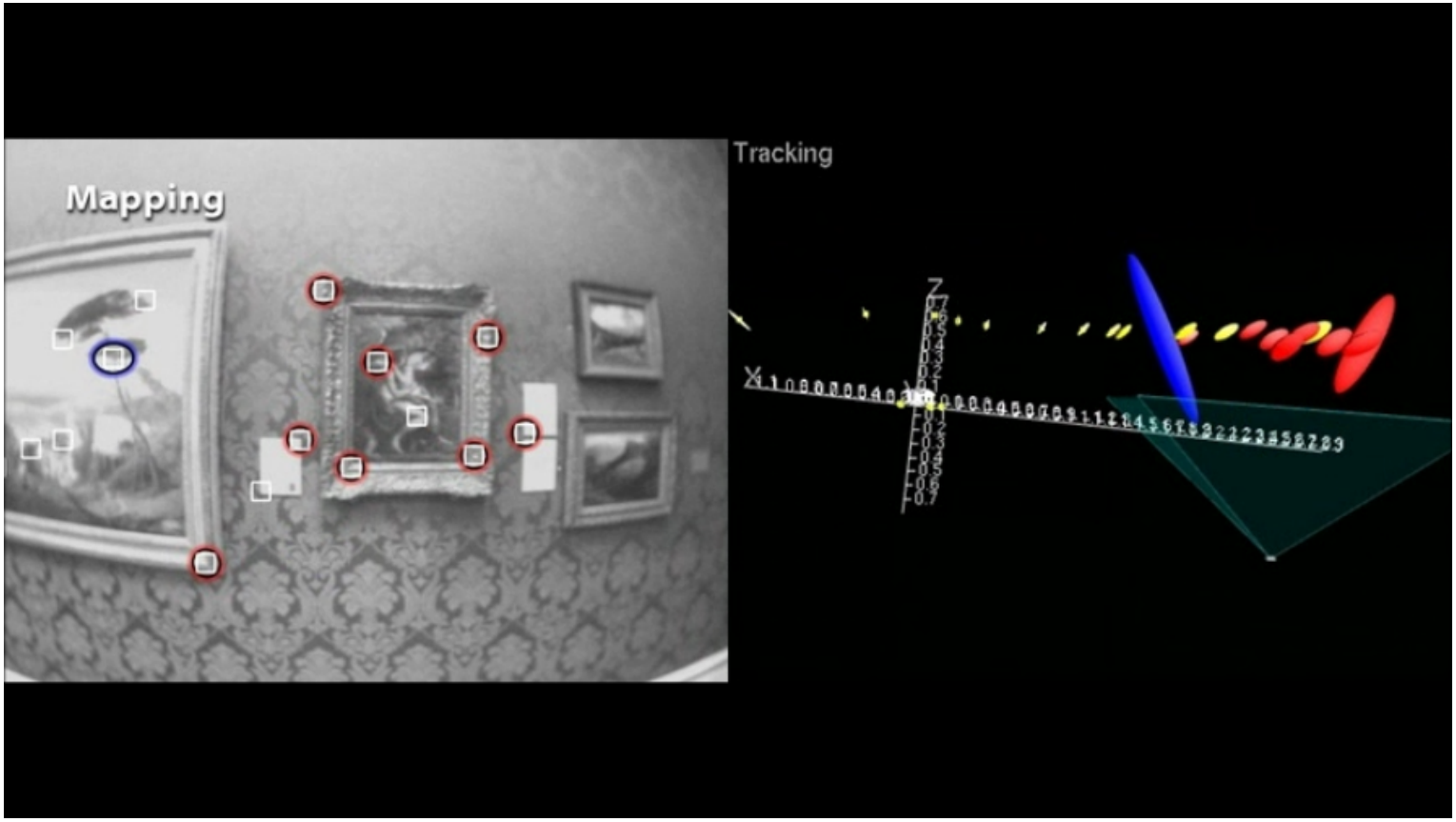


- Sparse, high quality map
  - **Salient feature patches** detected to serve as **visual landmarks**.
- Uncertainty-guided **active search** within an elliptical confidence region, using correlation to locate correspondence

Visual SLAM estimation cycle



Visual SLAM example



http://www.youtube.com/watch?v=uXqDA4do_s0&feature=player_embedded