Introduction to SLAM Simultaneous Localization And Mapping

Paul Robertson

Cognitive Robotics

Wed Feb 9th, 2005

Outline

- Introduction
- Localization
- SLAM
- Kalman Filter
  - Example
- Large SLAM – Scaling to large maps

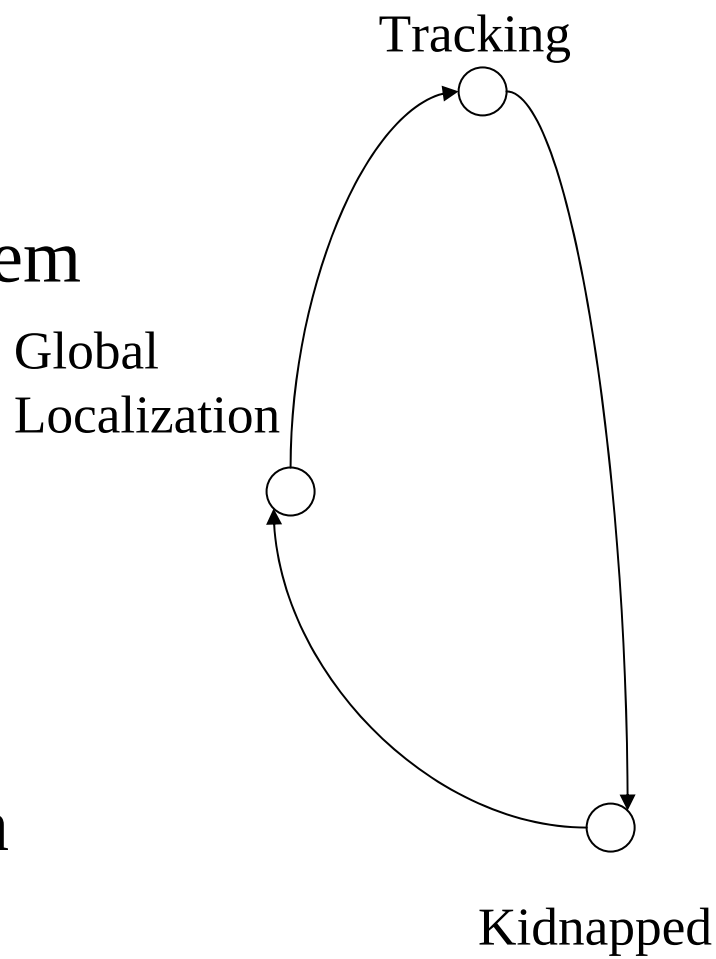
Introduction

- (Localization) Robot needs to estimate its location with respects to objects in its environment (Map provided).
- (Mapping) Robot need to map the positions of objects that it encounters in its environment (Robot position known)
- (SLAM) Robot simultaneously maps objects that it encounters and determines its position (as well as the position of the objects) using noisy sensors.

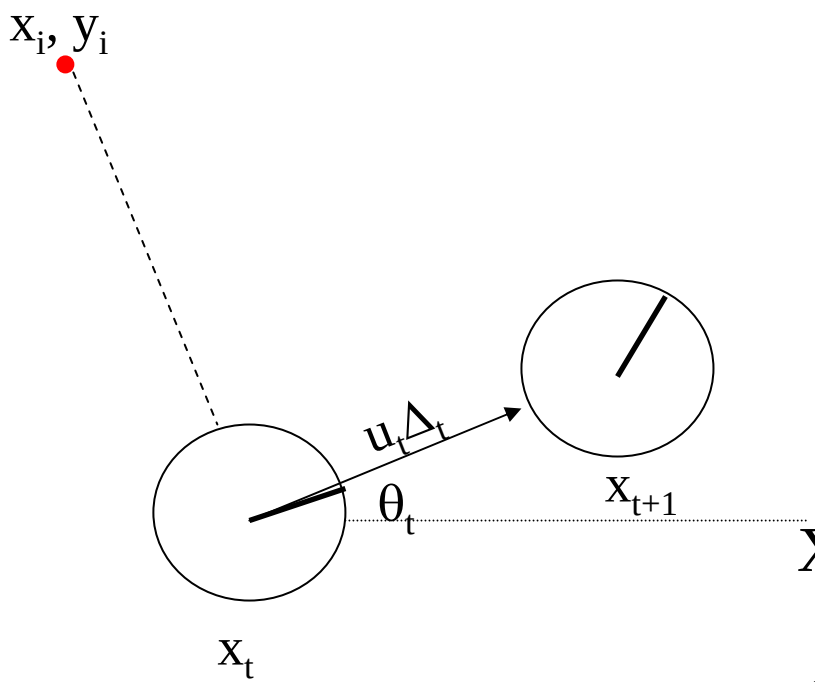
Show Movie

Localization

- Tracking
  - Bounded uncertainty
  - Can flip into kidnapping problem
- Global Localization
  - Initially huge uncertainties
  - Degenerates to tracking
- Kidnapping Problem
  - Unexpected global localization



Representing Robots

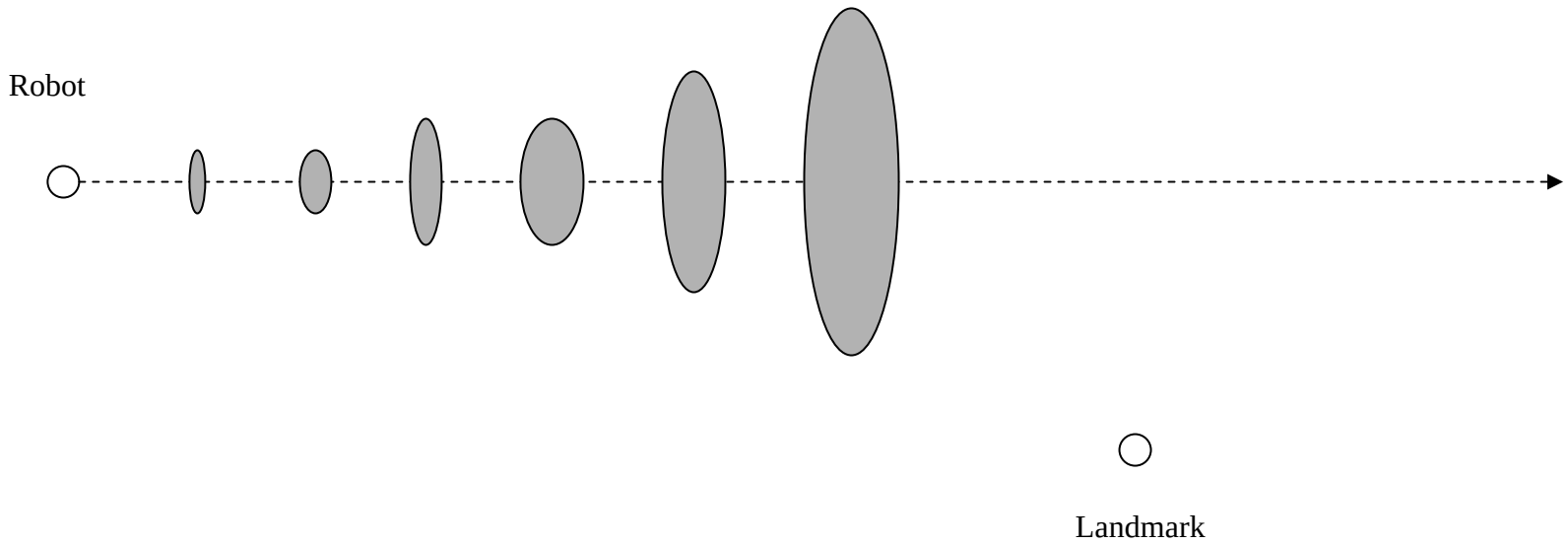


Position of robot is represented as a triple consisting of its x_t, y_t components and its heading θ_t:

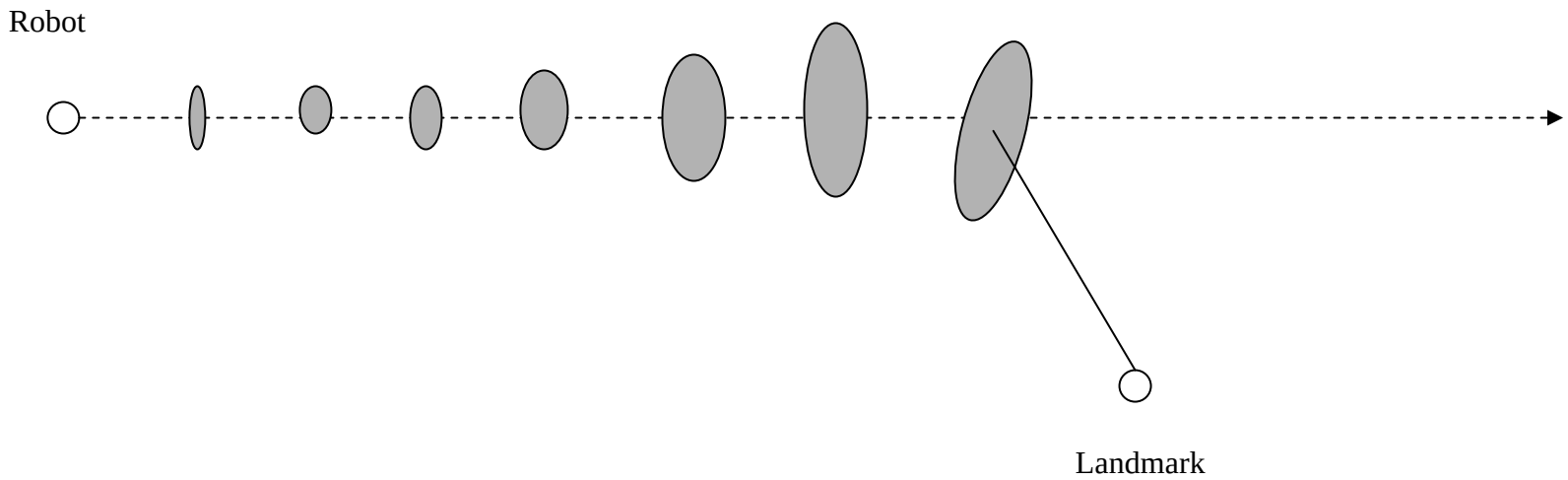
$$X_t=(x_t, y_t, \theta_t)^T$$

$$\widehat{X}_{t+1}=X_t + (u_t\Delta_t \cos\theta_t, u_t\Delta_t \sin\theta_t, u_t\Delta_t)^T$$

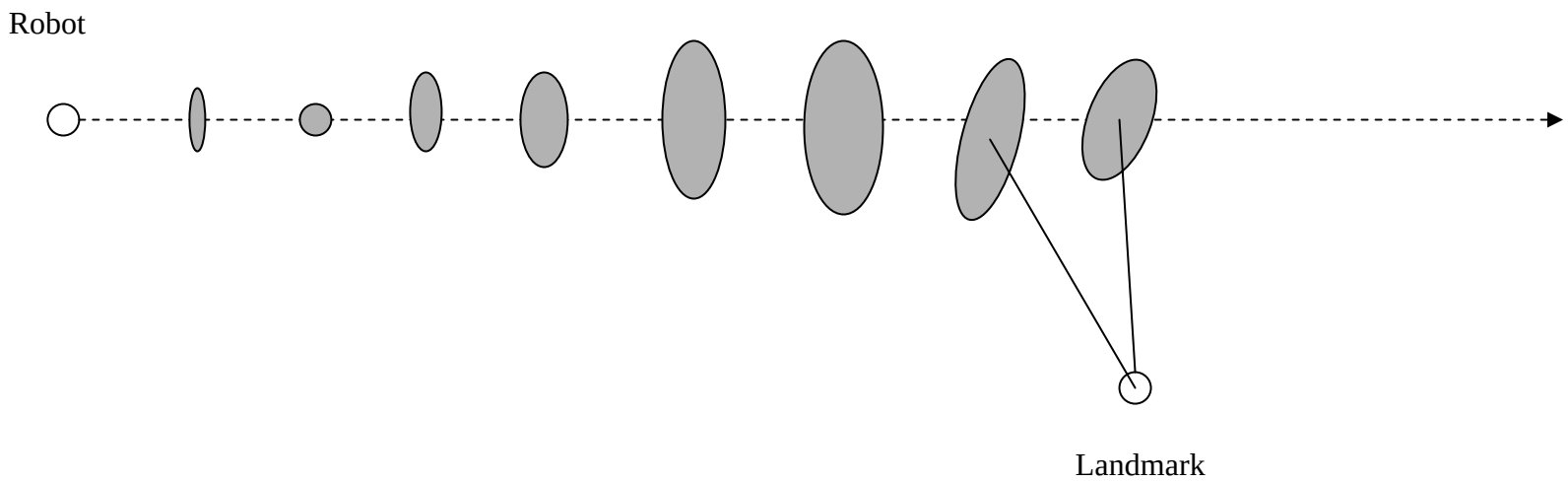
Kalman Filter Localization



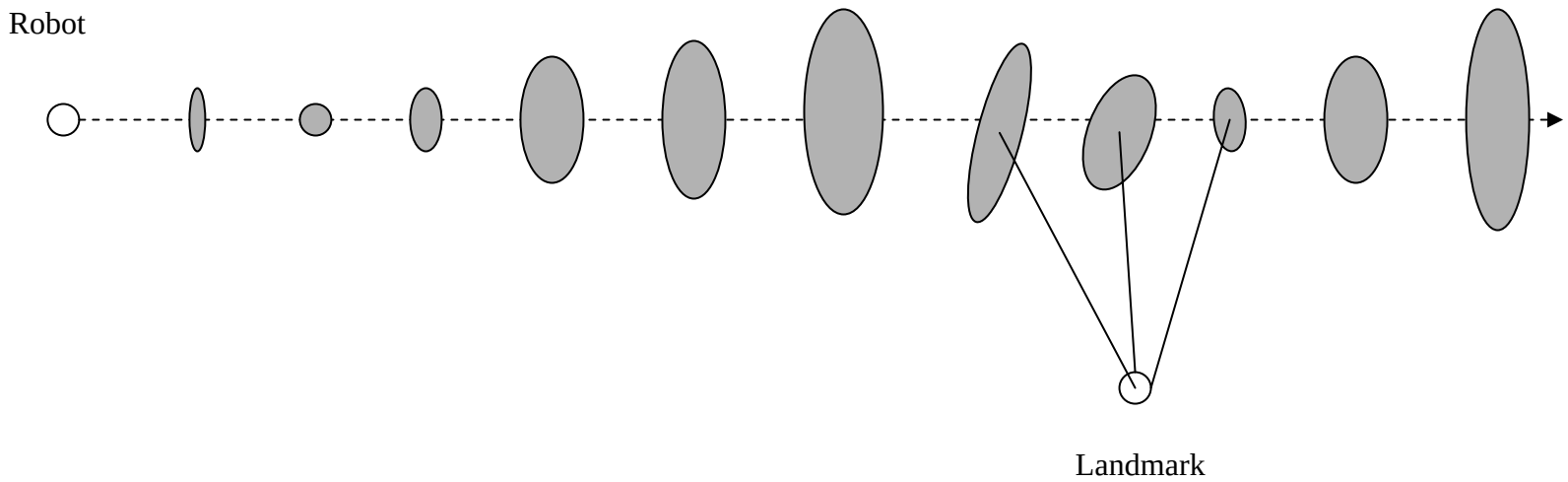
Kalman Filter Localization



Kalman Filter Localization



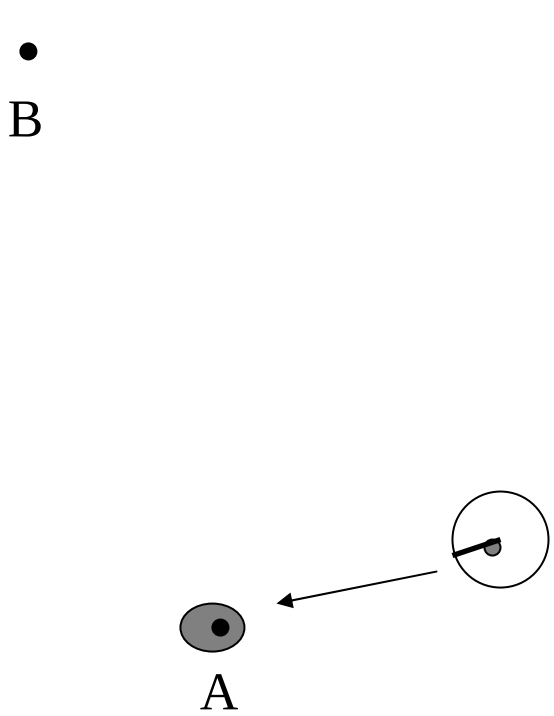
Kalman Filter Localization



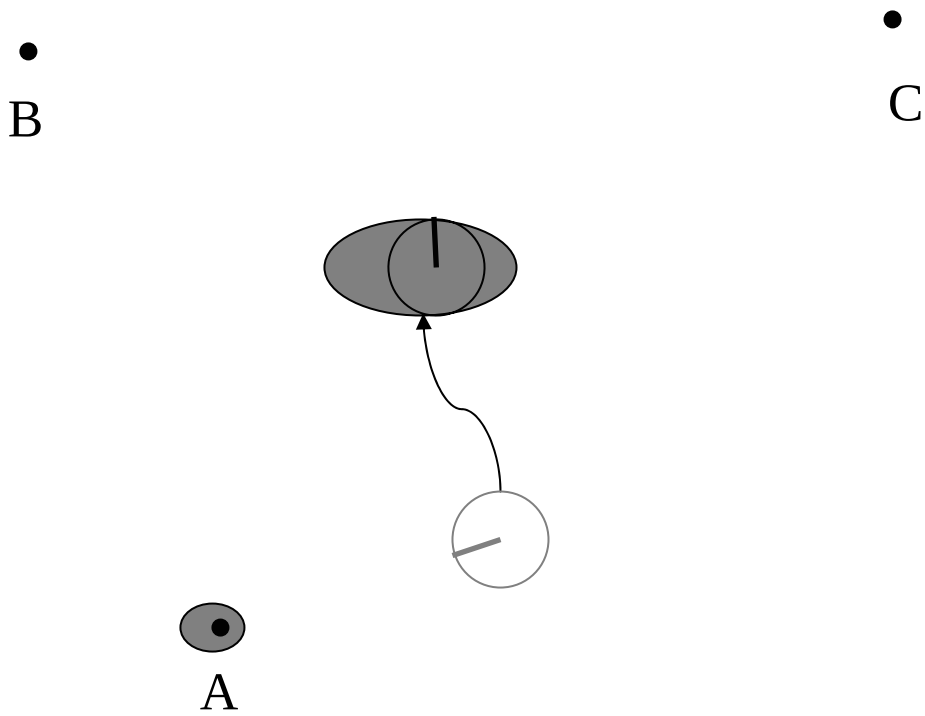
Basic SLAM

- Localize using a Kalman Filter (EKF)
- Consider all landmarks as well as the robot position as part of the posterior.
- Use a single state vector to store estimates of robot position and feature positions.
- Closing the loop allows estimates to be improved by correctly propagating all the coupling between estimates which arise in map building.

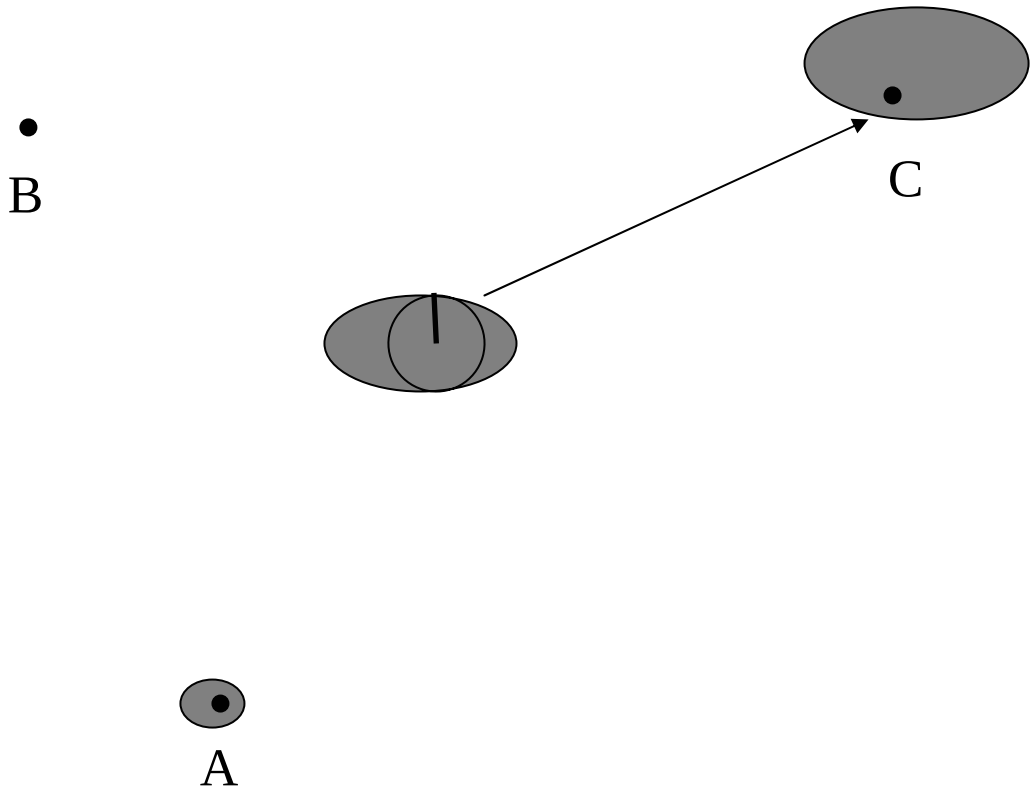
Initialize Feature A



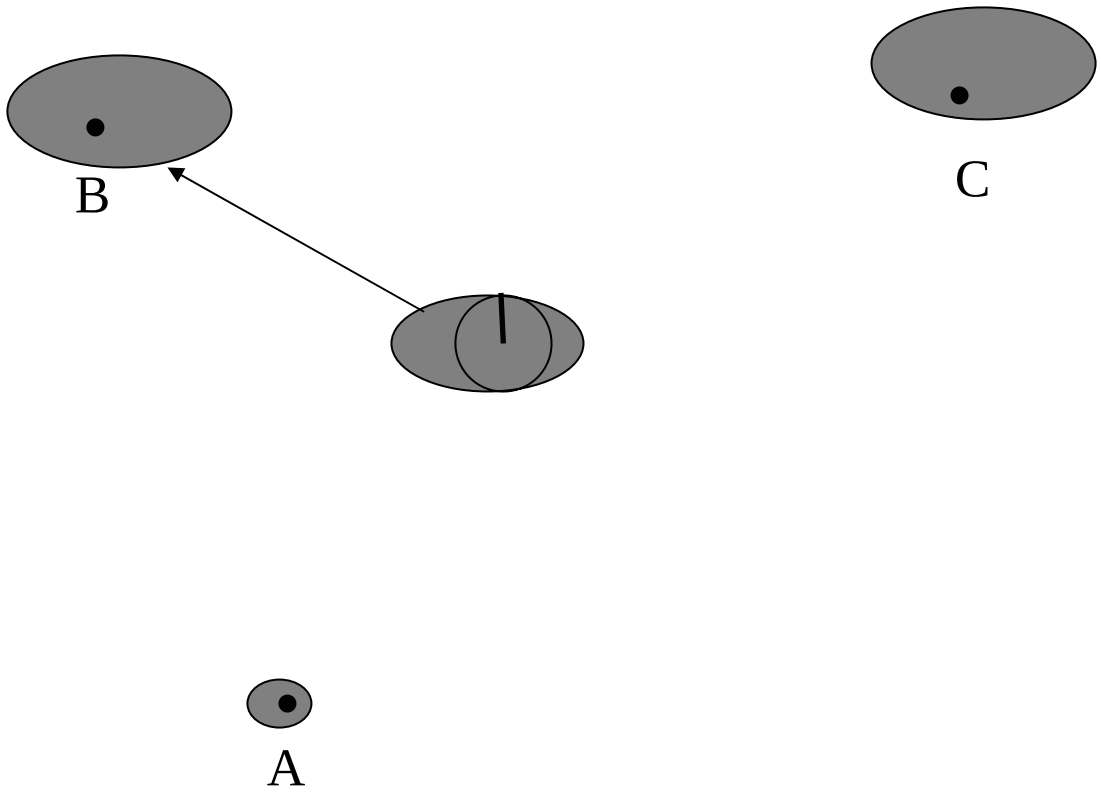
Drive Forward



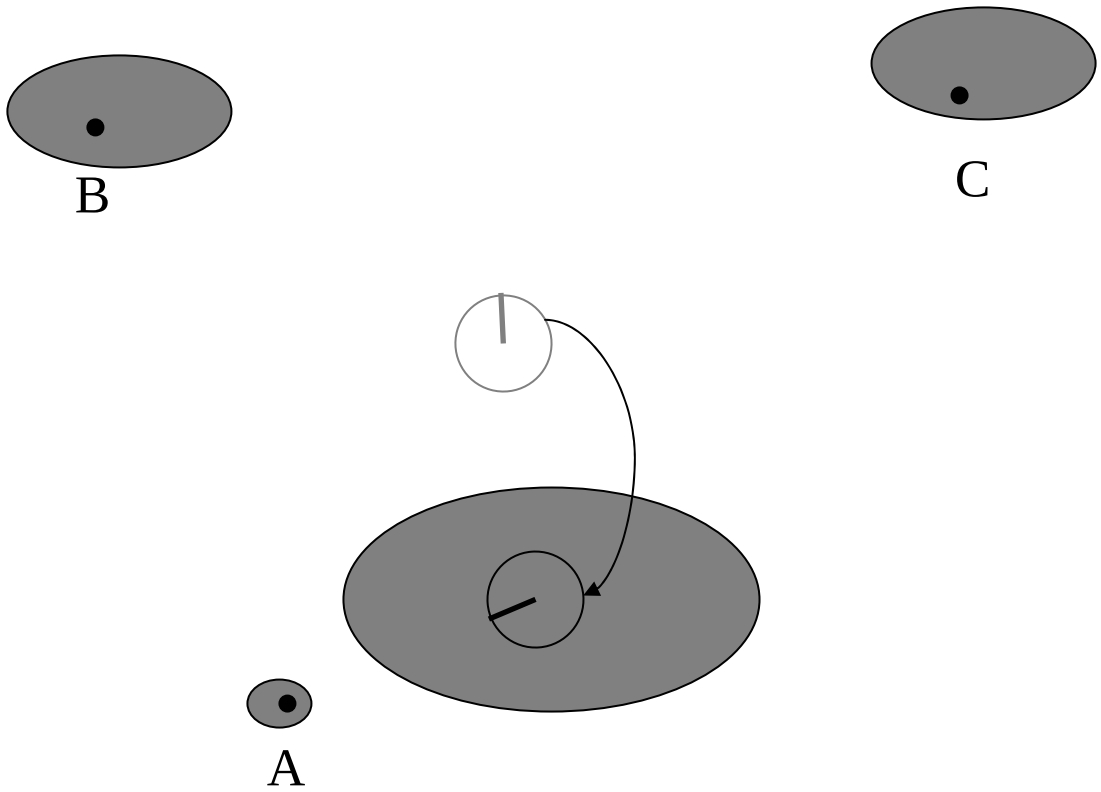
Initialize C



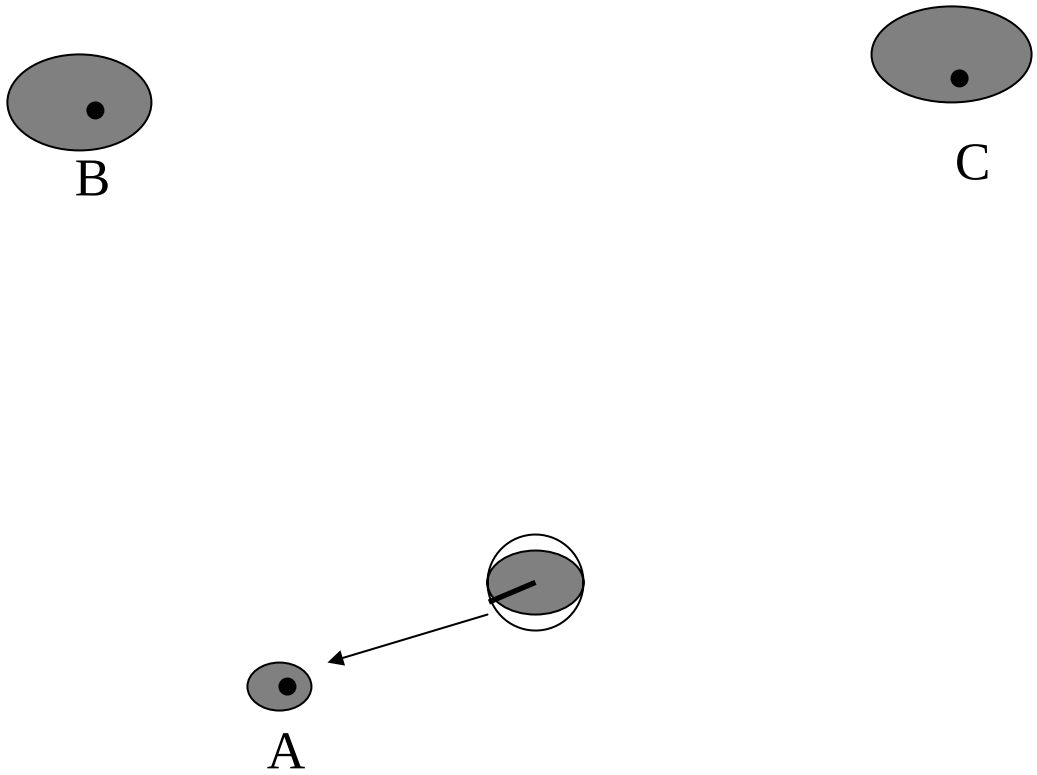
Initialize B



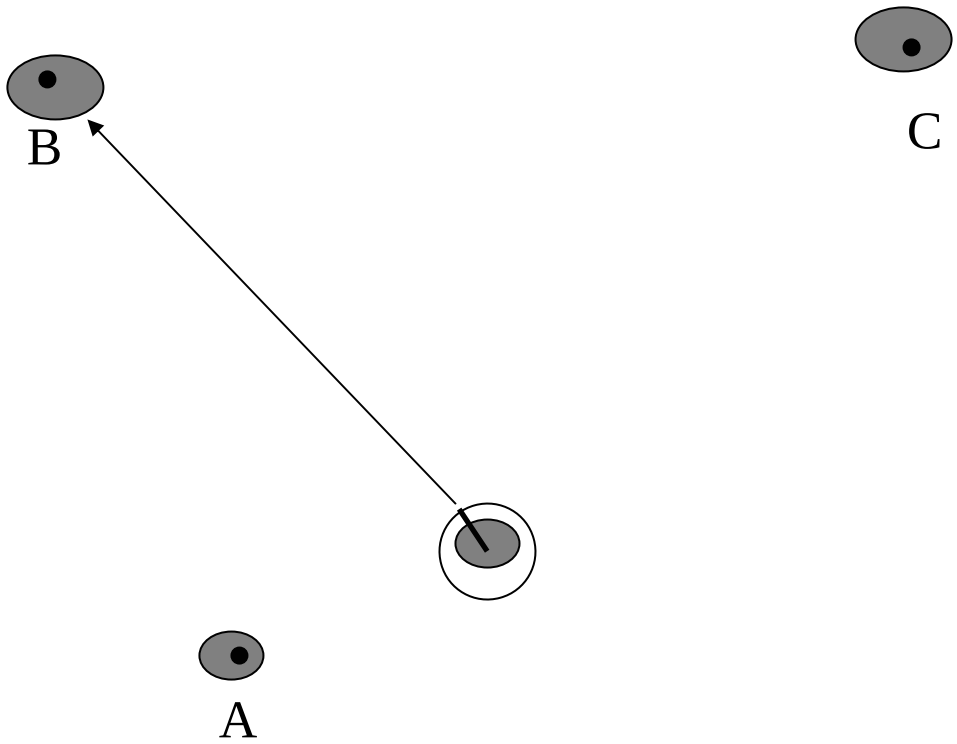
Drive Back



Re-measure A



Re-measure B



Basic EKF framework for SLAM

$$\hat{x} = \begin{pmatrix} \hat{x}_v \\ \hat{y}_1 \\ \hat{y}_2 \\ . \\ . \\ . \end{pmatrix} \quad , \qquad P = \begin{pmatrix} P_{xx} & P_{xy1} & P_{xy2} & \cdots \\ P_{y1x} & P_{y1y1} & P_{y1y2} & \cdots \\ P_{y2x} & P_{y2y1} & P_{y2y2} & \cdots \\ . & . & . & \\ . & . & . & \\ . & . & . & \end{pmatrix}$$

System State Vector

Covariance Matrix (Square, Symmetric)

$\hat{x}$ and P grow as features are added to the map!

SLAM steps…

1. Define robot initial position as the root of the world coordinate space – or start with some pre-existing features in the map with high uncertainty of the robot position.
2. Prediction: When the robot moves, motion model provides new estimates of its new position and also the uncertainty of its location – positional uncertainty always increases.
3. Measurement: (a) Add new features to map (b) re-measure previously added features.
4. Repeat steps 2 and 3 as appropriate.

The Kalman Filter

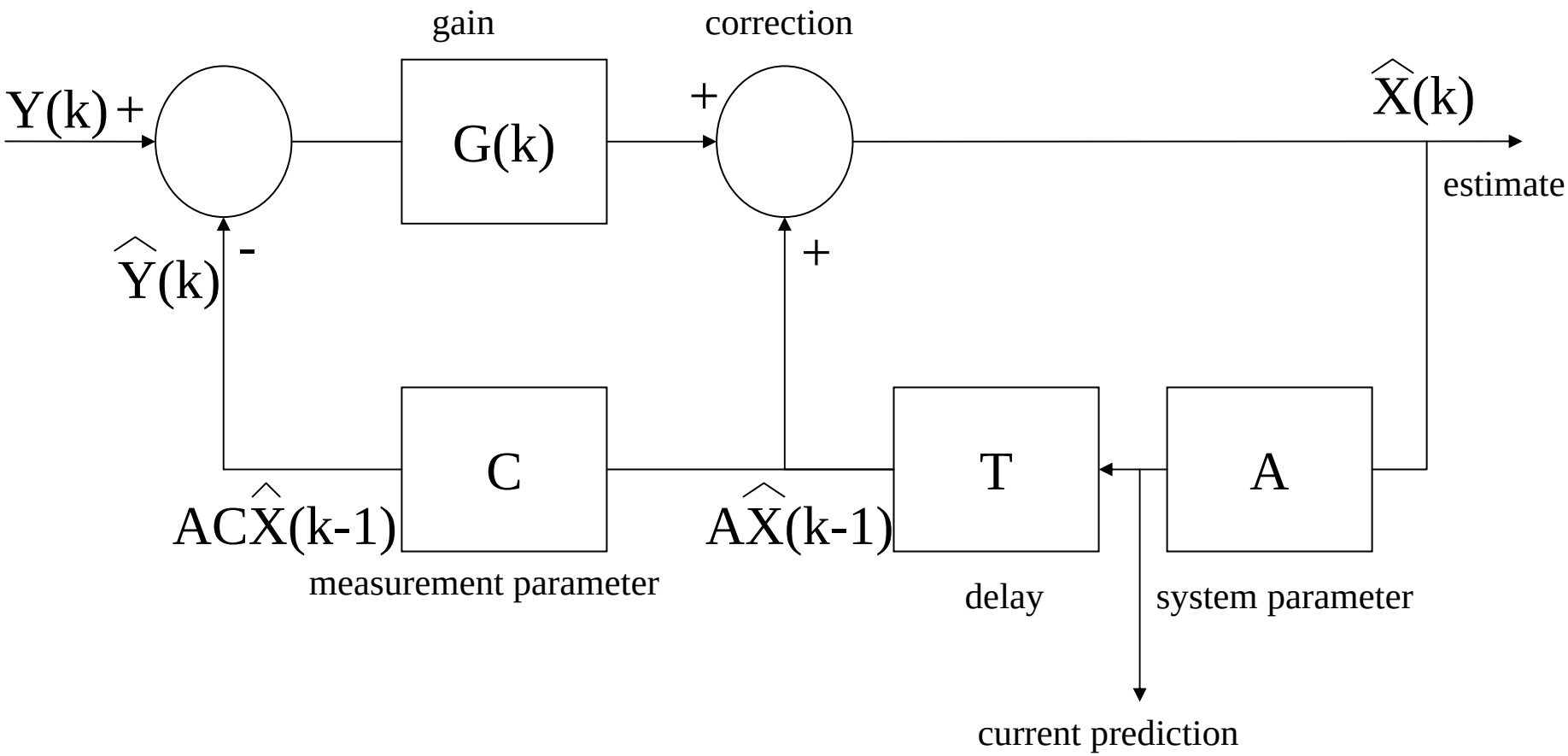
Features:

1. Gaussian Noise
2. Gaussian State Model
3. Continuous States
4. Not all state variables need to be observable.
5. Only requires memory of the previous estimate.
6. Update is quadratic on the number of state variables.

Updating state estimate with a new observation

1. Predict current state from the previous state estimate and the time step.
2. Estimate the observation from the prediction.
3. Computer the difference between the predicted observation and the actual observation.
4. Update the estimate of the current state

Vector Kalman Filter



$$G(k) = AP(k|k-1)C^T[CP(k|k-1)C^T+R(k)]^{-1} \quad \text{(Predictor gain)}$$

$$P(k+1|k) = [A-G(k)C]P(k|k-1)A^T+Q(k) \quad \text{(Prediction mean square error)}$$

```
Matrix newMeasurement (Matrix measurement, double ts)
    {
        Matrix matP1=null;   // variances of components of prediction
        Matrix cvecE=null;   // residual error
        Matrix cvecY=null;   // estimated input
        double timeDelta=ts-pts;  // Establish system matrix for time duration since the last observation.
        matA=matAorig.add(matStime.scalarMultiply(timeDelta));
        matAt=matA.transpose();
        // Predict the new state at the given time step
        cvecX=matA.mul (cvecX);
        // Measurement prediction- estimate of input
        cvecY=matC.mul (cvecX);
        cvecE=measurement.subtract(cvecY);
        // Calculate Kalman gain matK
        matP1=matA.mul (matP).mul (matAt).add(matQ);
        matK=matP1.mul (matCt).mul (matC.mul (matP1).mul (matCt).add(matR).invert());
        // update the error covariance matrix
        matP=matP1.subtract(matK.mul (matC).mul (matP1));
        cvecX=cvecX.add(matK.mul (cvecE));                    // Correct
        pts=ts;
        return matC.mul (cvecX); // return estimated values
    }
```

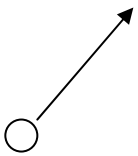
Simple Example

Robot with estimated position (P_x, P_y) and estimated velocity (V_x, V_y). Four state variables $(P_x, P_y, V_x, V_y)^T$

Relationships between the state variables:

$P_x \mathrel{+}= tV_x$

$P_y \mathrel{+}= tV_y$



Observations: (P_x, P_y)

Position variance (σp) = 100.0

Velocity variance (σv) = 0.1

System Equation

$$\underbrace{\begin{pmatrix} P_x(k+1) \\ P_y(k+1) \\ V_x(k+1) \\ V_y(k+1) \end{pmatrix}}_{X(k+1)} = \underbrace{\begin{pmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} P_x(k) \\ P_y(k) \\ V_x(k) \\ V_y(k) \end{pmatrix}}_{X(k)} + \underbrace{\begin{pmatrix} 0 \\ 0 \\ \sigma_v \\ \sigma_v \end{pmatrix}}_{w(k)}$$

Observations

$$\underbrace{\begin{bmatrix} P_x(k) \\ P_y(k) \end{bmatrix}}_{Y(k)} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}}_{C} \underbrace{\begin{bmatrix} P_x(k) \\ P_y(k) \\ V_x(k) \\ V_y(k) \end{bmatrix}}_{X(k)} + \underbrace{\begin{bmatrix} \sigma_p \\ \sigma_p \end{bmatrix}}_{v(k)}$$

Observation Matrix $Y(k)$ $= \begin{bmatrix} P_x \\ P_y \end{bmatrix}$

Measurement Covariance Matrix $= \begin{bmatrix} \sigma_p & 0 \\ 0 & \sigma_p \end{bmatrix}$

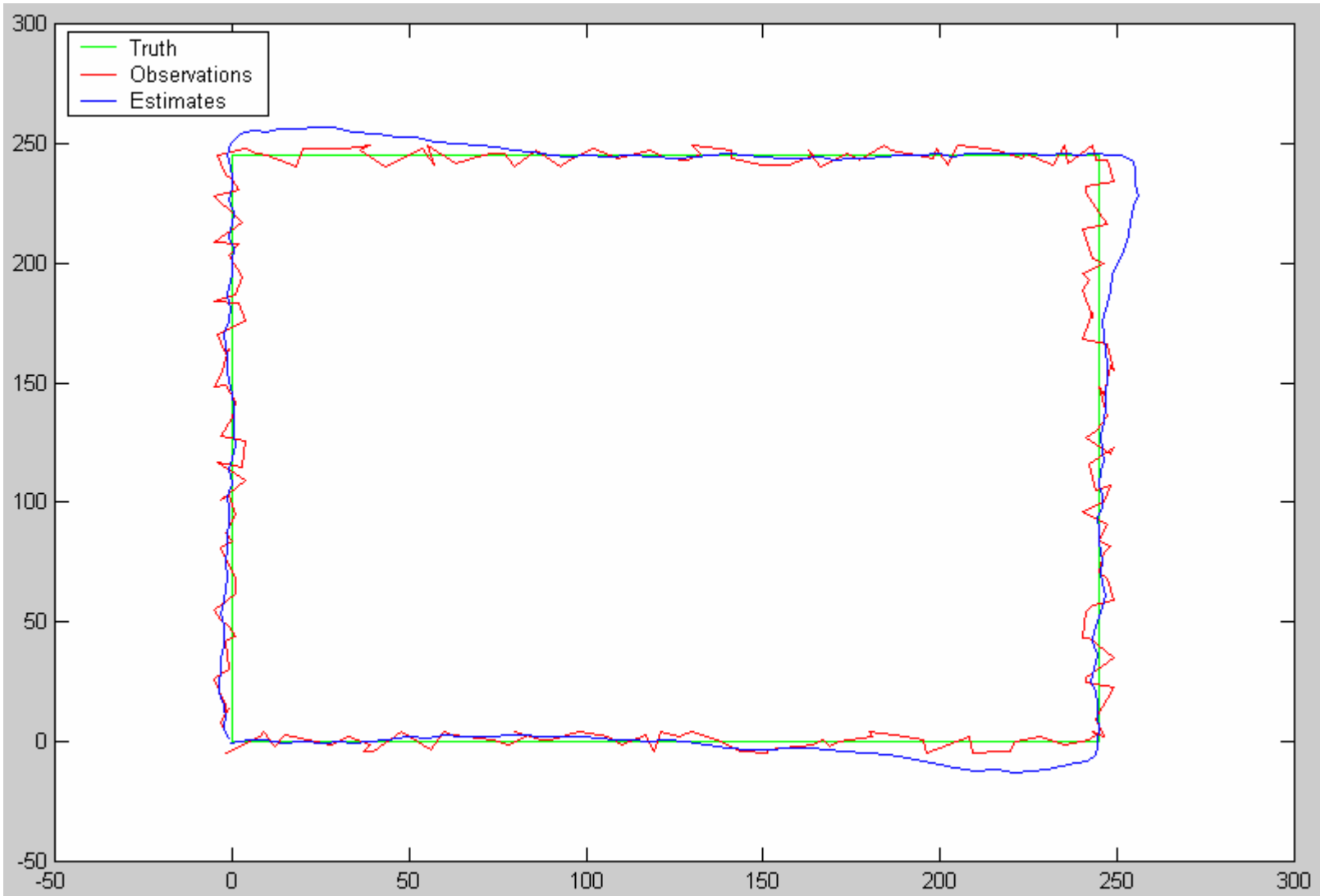
System Noise

System Noise Covariance Matrix $Q(k) = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_v & 0 \\ 0 & 0 & 0 & \sigma_v \end{pmatrix}$

Prediction Matrices

Initial Prediction Covariance Matrix (P) = $\begin{pmatrix} \sigma_p & 0 & \sigma_p & 0 \\ 0 & \sigma_p & 0 & \sigma_p \\ 0 & \sigma_p & \sigma_p\sigma_v & 0 \\ 0 & \sigma_p & 0 & \sigma_p\sigma_v \end{pmatrix}$

State Equations (A) = $\begin{pmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$


Truth
Observations
Estimates
300
250
200
150
100
50
0
-50
-50
0
50
100
150
200
250
300

Run Kalman Filter Demo Program

Problems with the Kalman Filter

1. Uni-modal distribution (Gaussian) often problematic.
2. Can be expensive with large number of state variables.
3. Assumes ‘linear transition model’ – system equations must be specifiable as a multiplication of the state equation. The Extended Kalman Filter (EKF) attempts to overcome this problem.

Additional Reading List for Kalman Filter

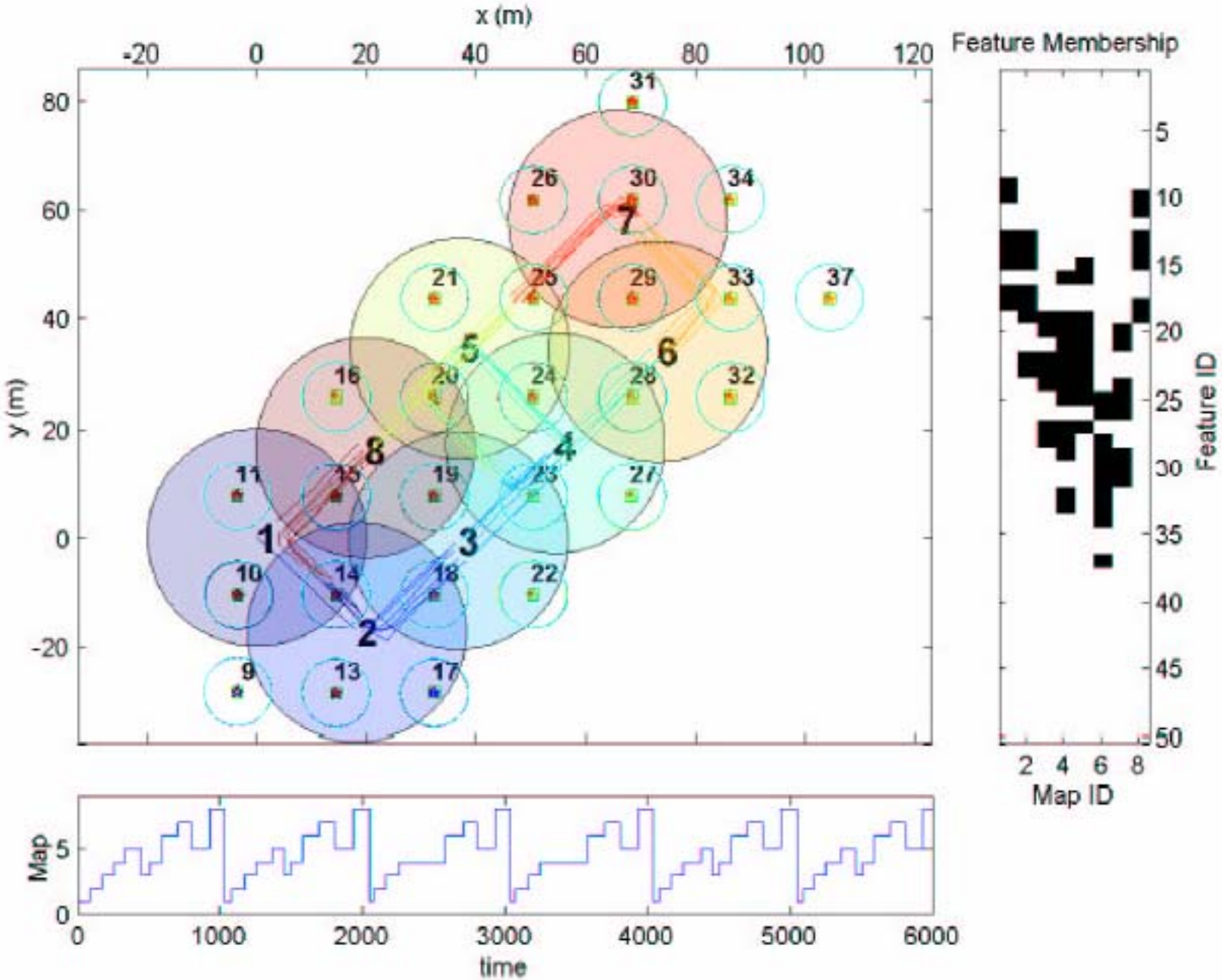
Russell&Norvig ‘Artificial Intelligence a modern approach’ second edition 15.4 (pp551-559).

Large SLAM

Basic SLAM is quadratic on the number of features and the number of features can be very large. Intuitively we want the cost of an additional piece of information to be constant.

Lets look at one approach that addresses this issue by dividing the map up into overlapping sub maps.

Leonard&Newman ‘Consistent, Convergent, and Constant-Time SLAM’ IJCAI’03


x (m)
-20
0
20
40
60
80
100
120
y (m)
80
60
40
20
0
-20
1
2
3
4
5
6
7
8
9
10
11
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
37
Map
5
0
0
1000
2000
3000
4000
5000
6000
time
Feature Membership
Feature ID
5
10
15
20
25
30
35
40
45
50
2
4
6
8
Map ID

Location Vector

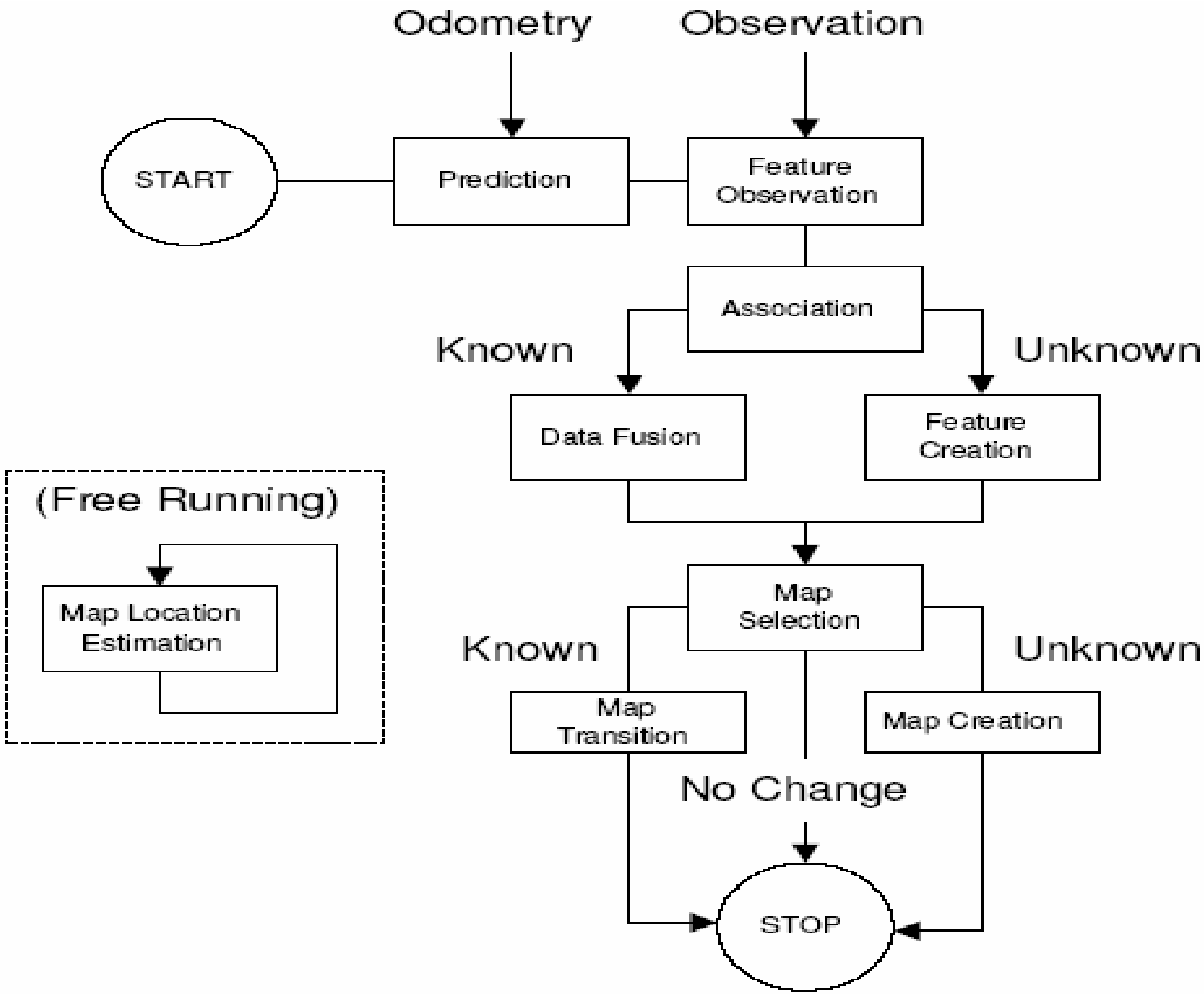
- The mapped space is divided up into overlapping sub maps with shared features in the overlapping sub maps.

$T(i, j) = (x, y, \theta)^T$ (Location vector)

An entity is a location + a unique ID.

A Map is a collection of entities described with respect to a local coordinate frame

Each map has a root entity i and a map location vector T(G,m)


Odometry
Observation
START
Prediction
Feature Observation
Association
Known
Unknown
Data Fusion
Feature Creation
(Free Running)
Map Location Estimation
Map Selection
Known
Unknown
Map Transition
Map Creation
No Change
STOP

Complete Algorithm Description on Blackboard