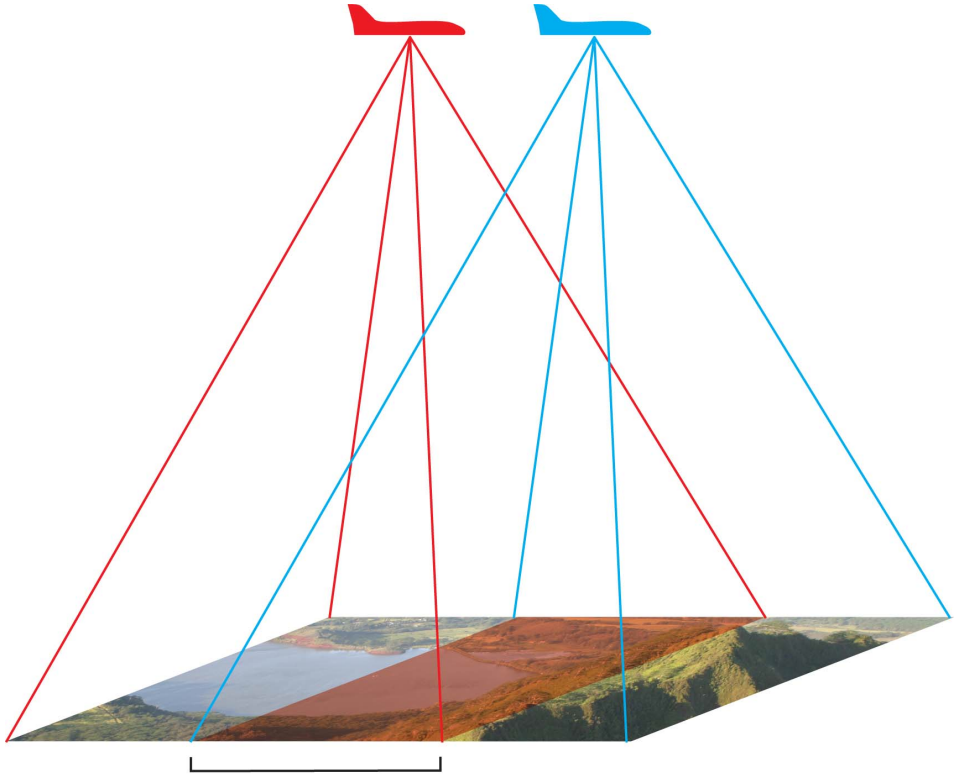
SLAM Tutorial

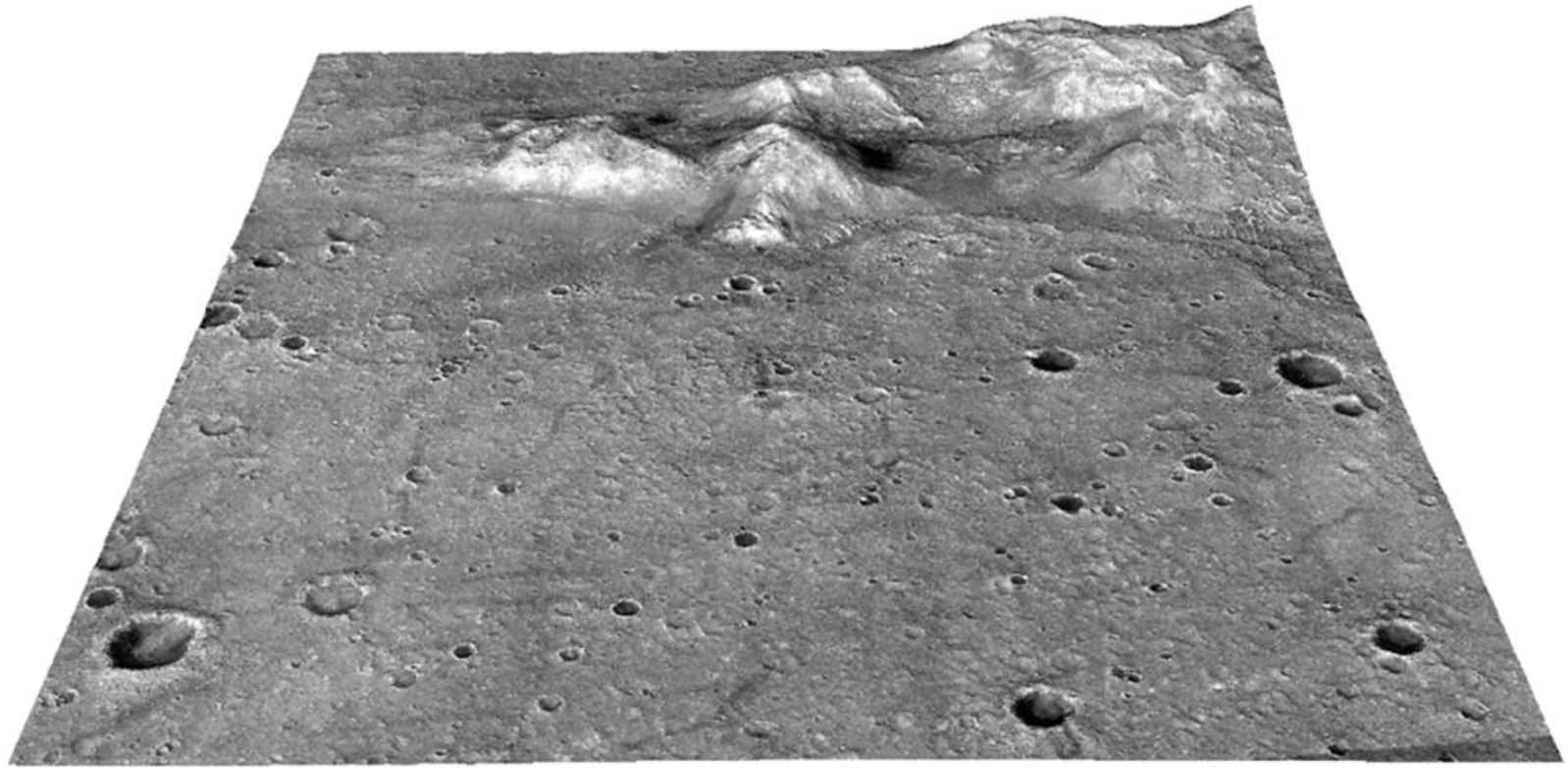
Slides by Marios Xanthidis,
Cyril Stachniss, Peter Allen
Paul Furgale, Margarita Chli,
Marco Hutter, Martin Rufli,
Davide Scaramuzza, Roland Siegwart

SLAM – A short history of photogrammetry



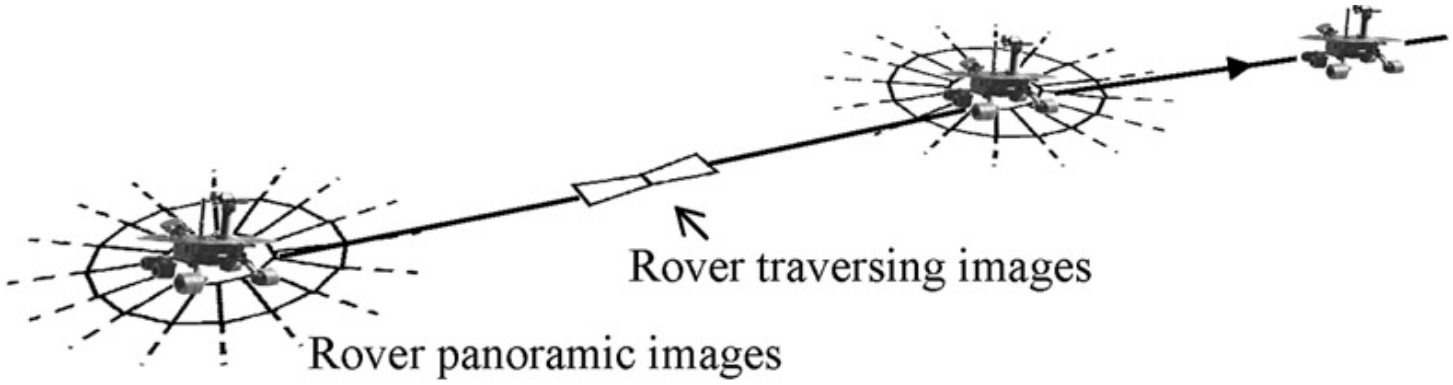
Surveying for Mapping---part 1, http://www.icsm.gov.au/mapping/surveying1.html

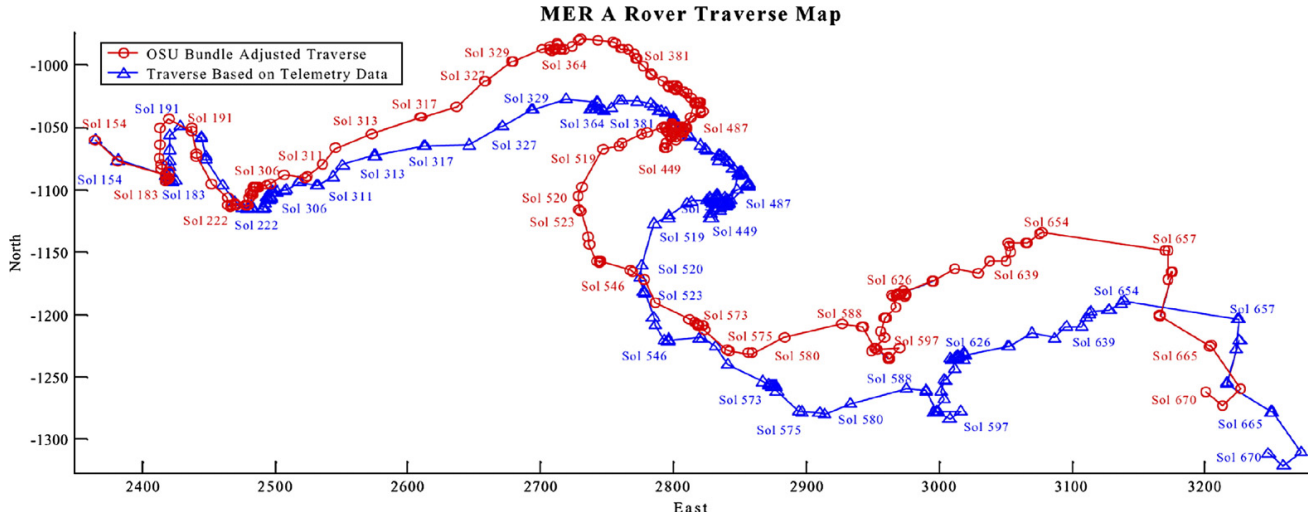
SLAM – A short history of photogrammetry



R. Li, J. Hwangbo, Y. Chen, and K. Di. Rigorous photogrammetric processing of hirise stereo imagery for mars topographic mapping. Geoscience and Remote Sensing, IEEE Transactions on, (99):1–15, 2008.

SLAM – A short history of photogrammetry





K. Di, F. Xu, J. Wang, S. Agarwal, E. Brodyagina, R. Li, and L. Matthies. Photogrammetric processing of rover imagery of the 2003 mars exploration rover mission. ISPRS Journal of Photogrammetry and Remote Sensing, 63(2):181–201, 2008.

What is the SLAM problem?

- The problem could described in the following question:

  "If we leave a robot in an unknown location in an unknown environment can the robot make a satisfactory map while simultaneously being able to find its pose in that map?"
- The solution to this problem **was** the "Holy Grail" of the field of mobile robotics.

The SLAM Problem (Difficulties)

- The bad news:
  - Pretty difficult problem because it combines the difficulties of both the localization and mapping problem without the essential assumptions of the known map or the known pose. Classic chicken or egg problem. Data Association problem also key.
- The good news:
  - The problem is considered solved but there are still some issues on having more general SLAM solutions and creating better maps.

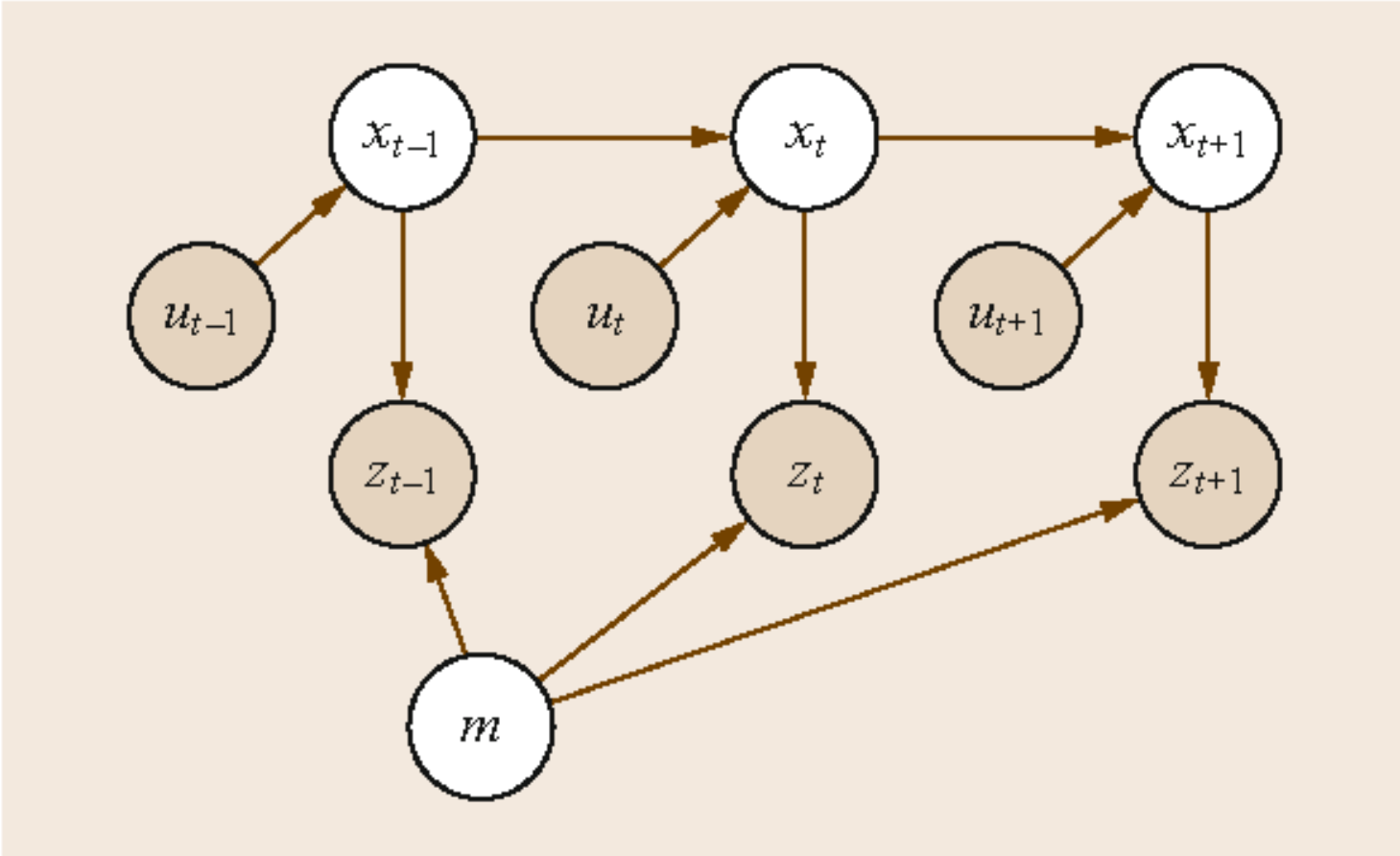
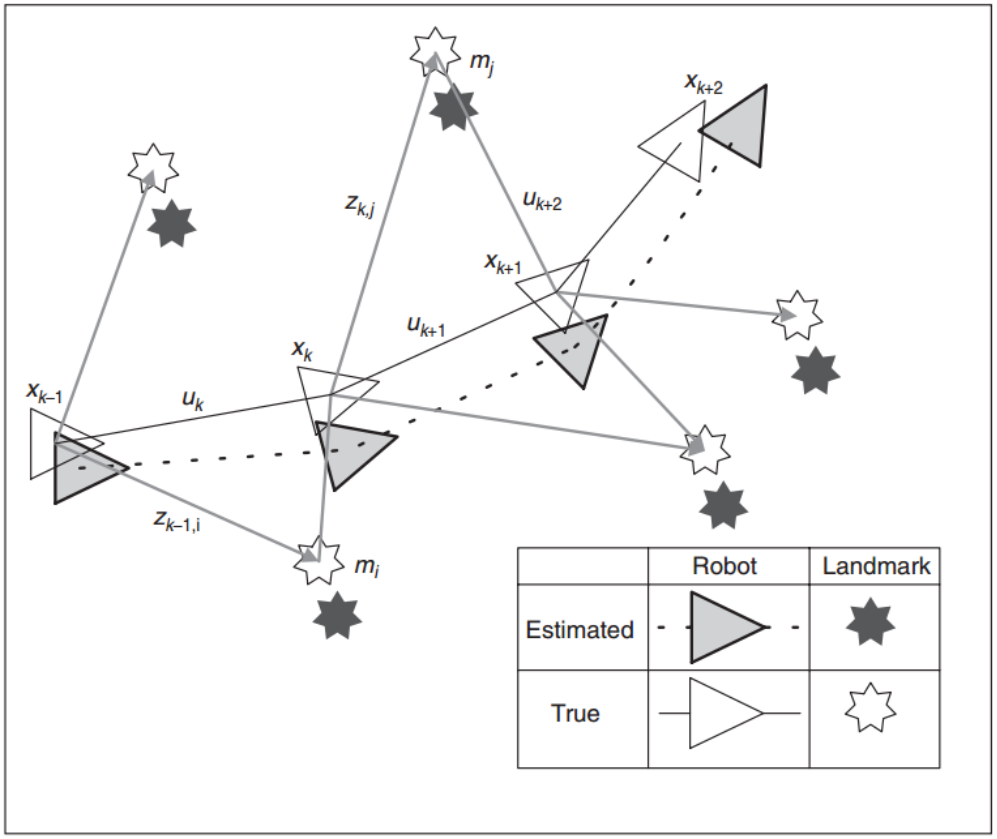


Fig. 37.1 Graphical model of the SLAM problem. *Arcs* indicate causal relationships, and *shaded nodes* are directly observable to the robot. In SLAM, the robot seeks to recover the unobservable variables

The SLAM Problem- Preliminaries(1)

- $\mathbf{X_k}$: the state vector describing the location and orientation of the vehicle at time k.
- $\mathbf{u_k}$: the control vector applied the time k-1.
- $\mathbf{m_i}$: a vector describing the location of the i^{th} landmark. The landmarks are motionless.
- $\mathbf{z_{ik}}$: an observation taken from the vehicle of the location of the i^{th} landmark at time k.



The SLAM Problem- Preliminaries(2)

- Also the following sets are defined:
  - $X_{0:k} = \{x_0, x_1, \cdots, x_k\}$ Position estimates
  - $U_{0:k} = \{u_1, u_2, \cdots, u_k\}$ Motion update controls
  - $Z_{0:k} = \{z_1, z_2, \cdots, z_k\}$ Sensor measurements
  - $m = \{m_1, m_2, \cdots, m_n\}$ Landmark locations

Full SLAM Problem

- We need to compute for all times k the probabilistic distribution of the joint posterior density of the landmarks and the state:

  $\mathbf{P(x_k, m | Z_{0:k}, U_{0:k}, x_0)}$ Why do we need the x_0 element?
- In order to compute that probability we need to compute previously:
  - $P(z_k | x_k, m)$ (measurement model)
  - $P(x_k | x_{k-1}, u_k)$ (motion model)
- Online SLAM just calculates the most recent state based upon the previous states

Probabilistic SLAM - Updates

- Time update (prediction):

$$P(x_k, m|Z_{0:k-1}, U_{0:k}, x_0) = \int P(x_k|x_{k-1}, u_k) \times P(x_{k-1}, m|Z_{0:k-1}, U_{0:k-1}, x_0) dx_{k-1}$$

- Measurement update (correction):

$$P(x_k, m|Z_{0:k-1}, U_{0:k}, x_0) = \frac{P(z_k|x_k, m)P(x_k, m|Z_{0:k-1}, U_{0:k}, x_0)}{P(z_k|Z_{0:k-1}, U_{0:k})}$$

- We can solve the localization problem with the assumption that we know the map:

$$P(x_k|Z_{0:k}, U_{0:k}, m)$$

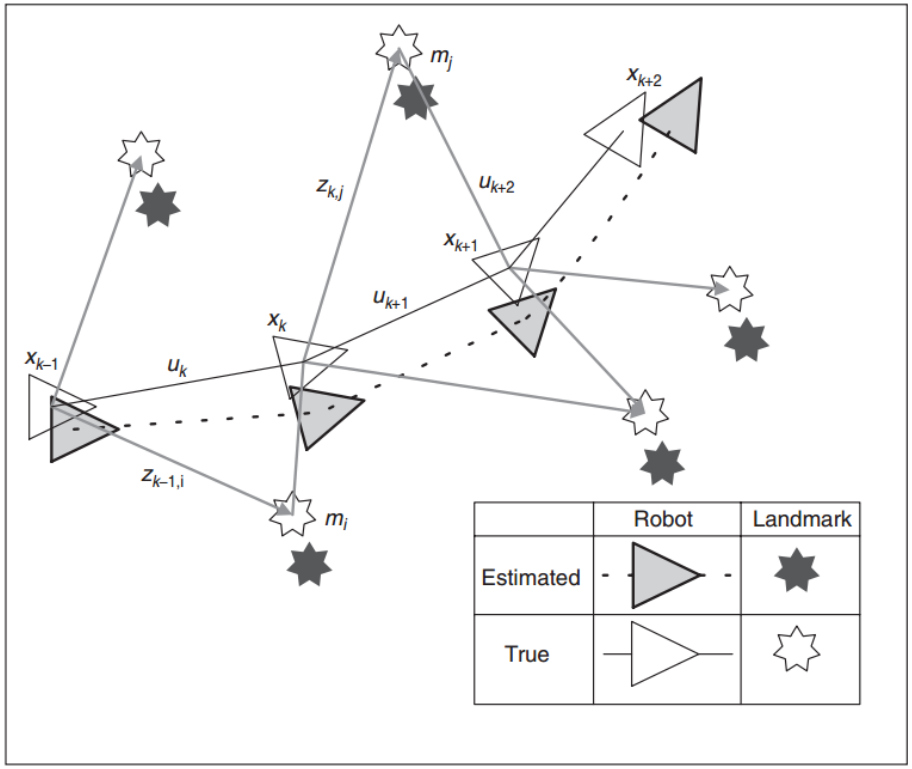
- And the mapping problem with the assumption we know the location:

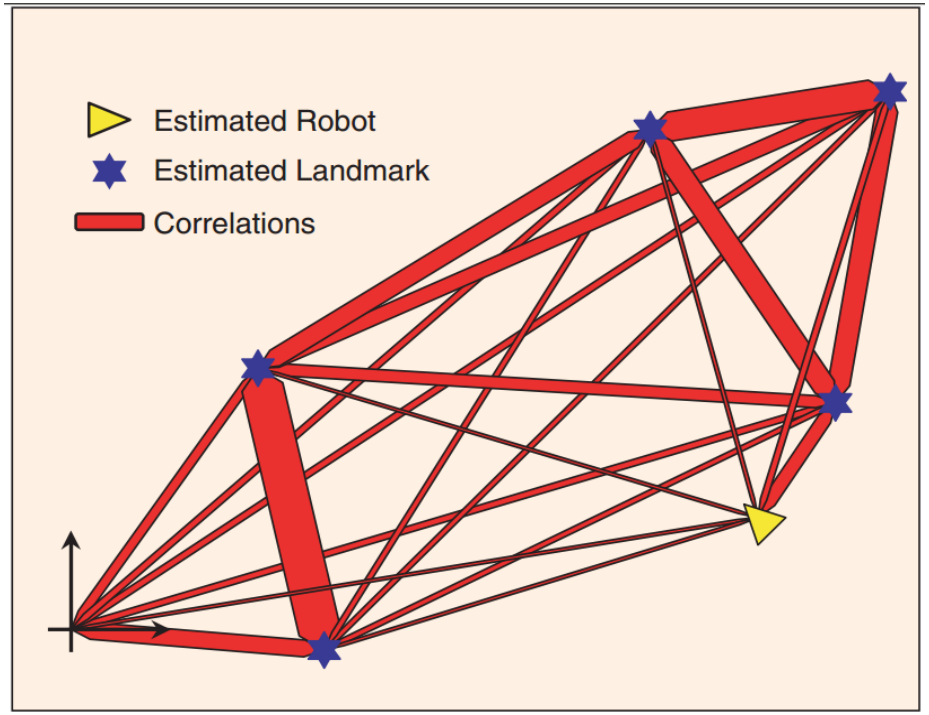
$$P(m|X_{0:k}, Z_{0:k}, U_{0:k})$$

Structure of Probabilistic SLAM(1)

- The landmark locations estimates are highly correlated. We may know with high accuracy the relation between the landmarks even if the absolute location is uncertain!
- The correlations are increased for every observations. Also the estimates for the relative location for every landmark are improved monotonically as more observations are made!

Structure of Probabilistic SLAM(2)





Structure of Probabilistic SLAM(3)

- The observation made by the robot regarding the relative location of the landmarks can be considered nearly independent, because the relative location of the landmarks is independent from the robot's coordinate frame.
- The observation made by the robot regarding the absolute location of the landmarks is more uncertain because the absolute location of each landmark is strongly related to the robots coordinate frame.
- Because of the correlations of the landmarks we can update the location of landmarks even we cannot observe. So the correlations are increased for every observation we make.
- Thus, the robot 's accuracy on building the relative map of the environment increased for more observations.

Solutions to SLAM Problem

- The goal is to find an appropriate representation for the observation and motion problem.
- Three different methods:
  - Graph Slam
  - EKF-SLAM: Using the Extended Kalman Filter.
  - Using particle filters :
    - Rao-Blackwellized particle filter (FastSLAM)

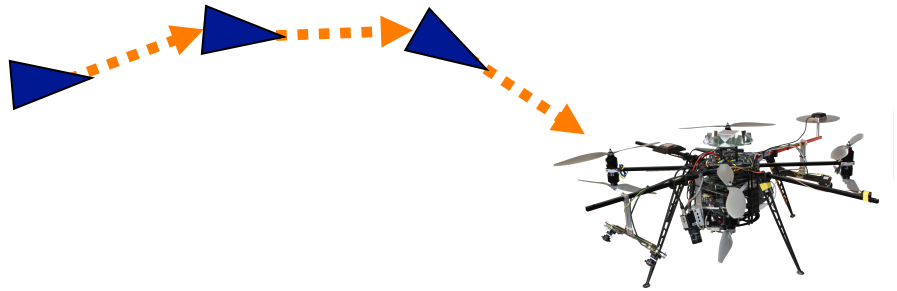
Graph-Based SLAM ??

SLAM = simultaneous localization and mapping

graph = representation of a set of objects where pairs of objects are connected by links encoding relations between the objects

Graph-Based SLAM

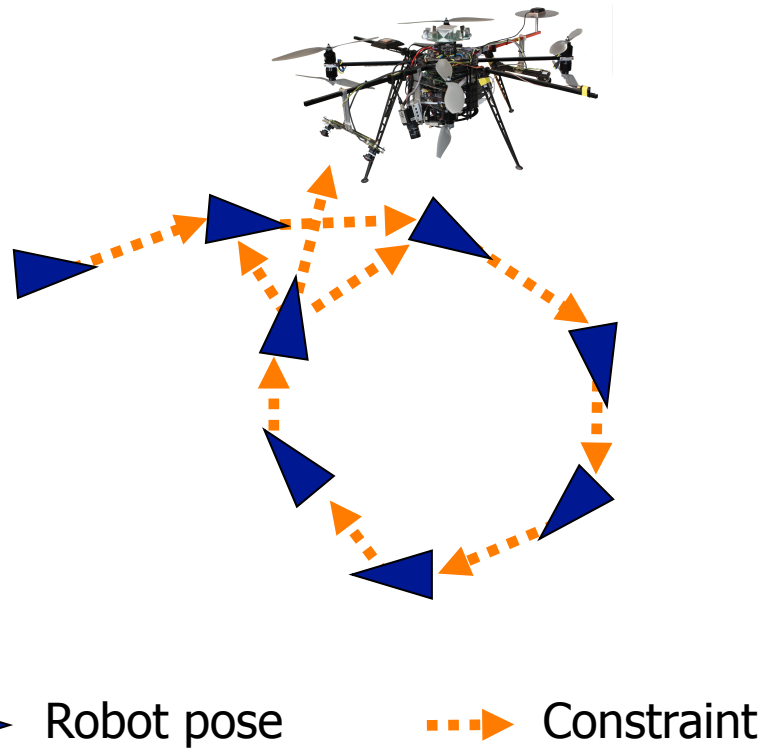
- Nodes represent poses or locations
- Constraints connect the poses of the robot while it is moving
- Constraints are inherently uncertain



Robot pose Constraint

Graph-Based SLAM

- Observing previously seen areas generates constraints between non-successive poses



Idea of Graph-Based SLAM

- Use a **graph** to represent the problem
- Every **node** in the graph corresponds to a pose of the robot during mapping
- Every **edge** between two nodes corresponds to a spatial constraint between them
- **Graph-Based SLAM:** Build the graph and find a node configuration that minimize the error introduced by the constraints

Graph-SLAM and Least Squares

- The nodes represent the **state**
- Given a state, we can compute what we **expect** to perceive
- We have **real observations** relating the nodes with each other

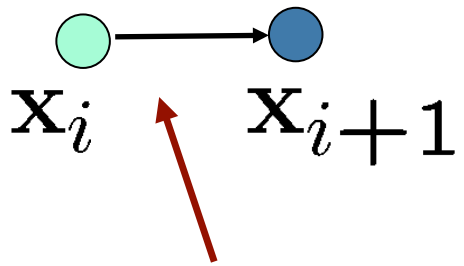
Graph-SLAM and Least Squares

- The nodes represent the **state**
- Given a state, we can compute what we **expect** to perceive

Find a configuration of the nodes so that the real and predicted observations are as similar as possible

Create an Edge If… (1)

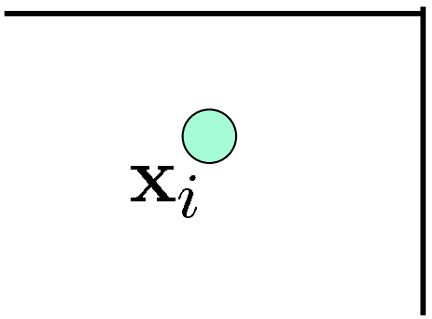
- …the robot moves from $\mathbf{x}_i$ to $\mathbf{x}_{i+1}$
- Edge corresponds to odometry



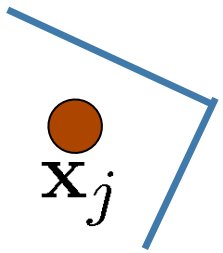
The edge represents the **odometry** measurement

Create an Edge If… (2)

- …the robot observes the same part of the environment from $\mathbf{x}_i$ and from $\mathbf{x}_j$



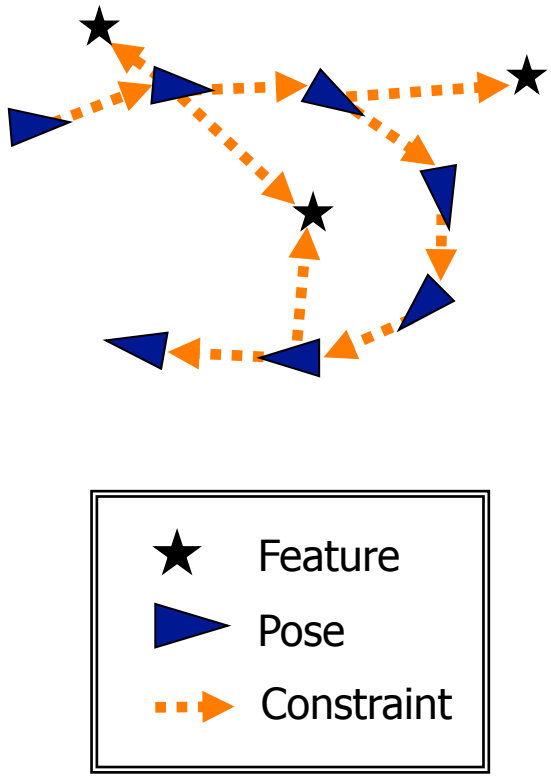
Measurement from $\mathbf{x}_i$



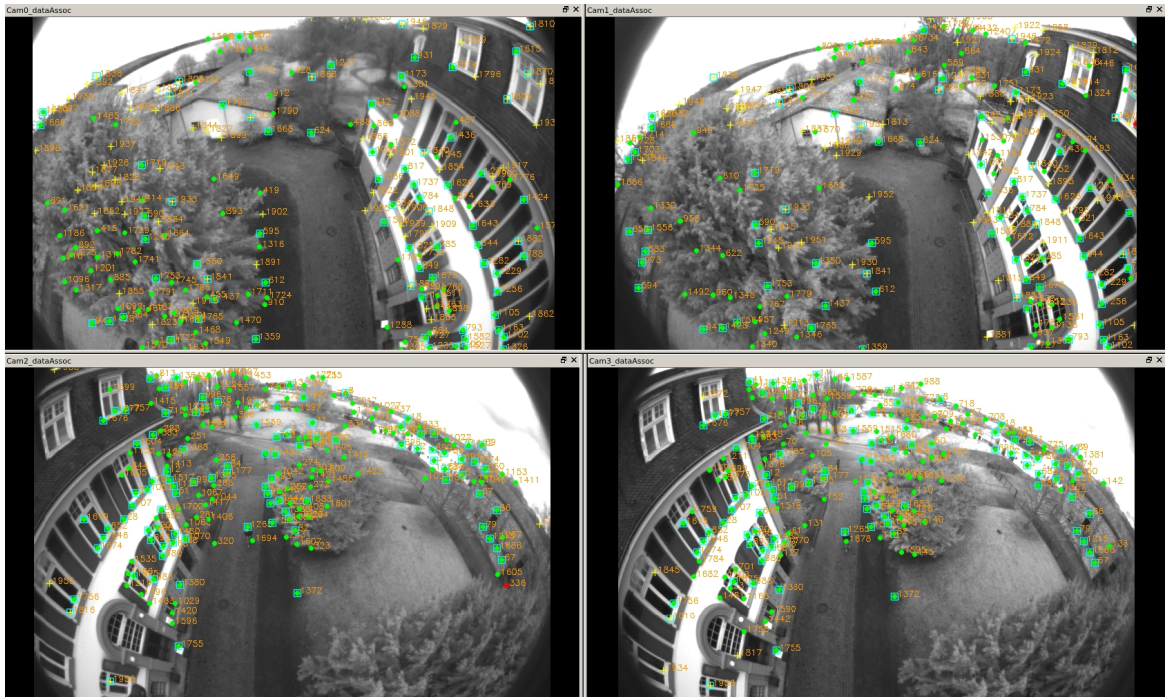
Measurement from $\mathbf{x}_j$

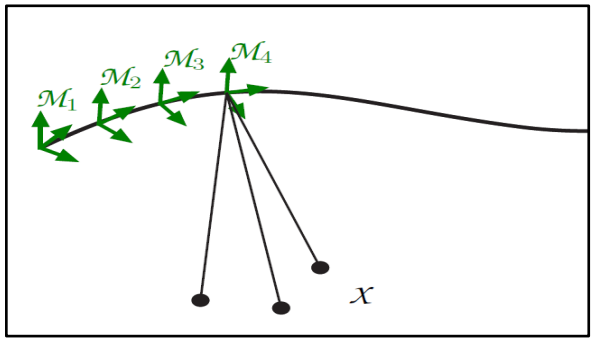
The Graph with Landmarks

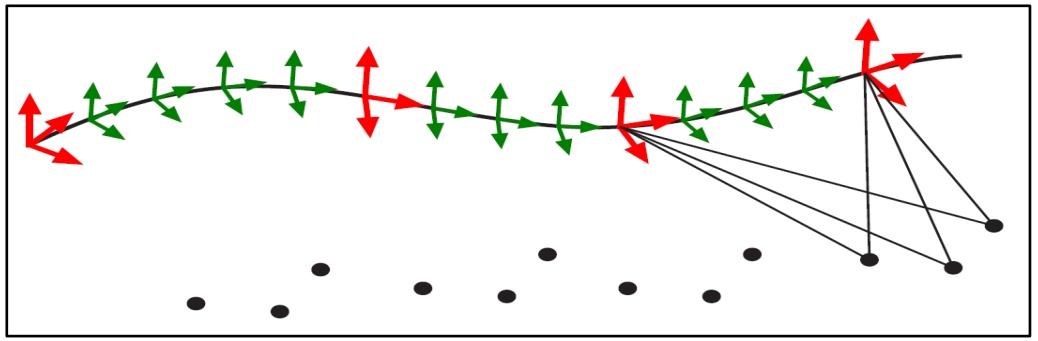
- **Nodes** can represent:
  - Robot poses
  - Landmark locations
- **Edges** can represent:
  - Landmark observations
  - Odometry measurements
- The minimization optimizes the **landmark locations and robot poses**



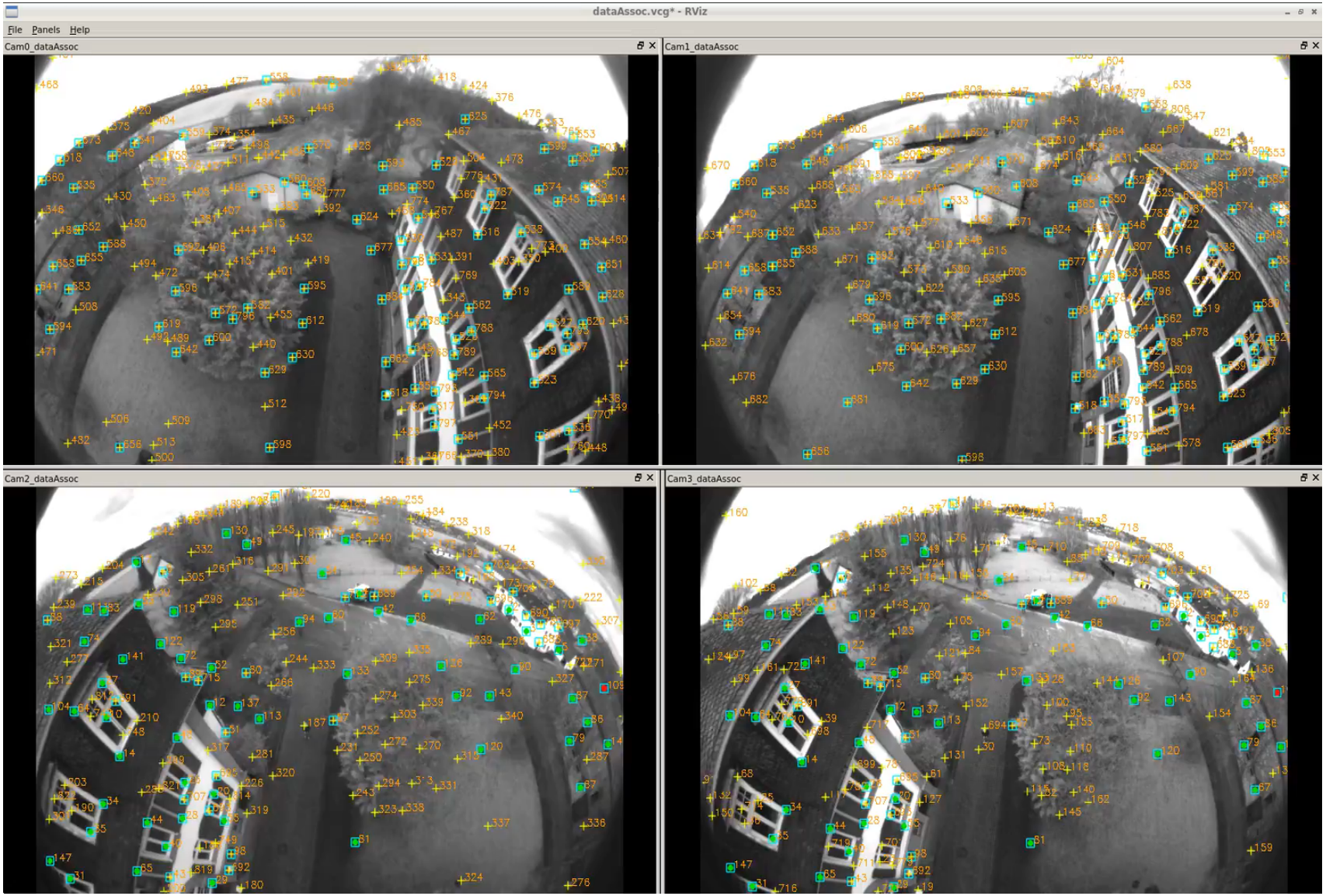
UAV Example


Cam0_dataAssoc
Cam1_dataAssoc
Cam2_dataAssoc
Cam3_dataAssoc


$\mathcal{M}_1$
$\mathcal{M}_2$
$\mathcal{M}_3$
$\mathcal{M}_4$
x

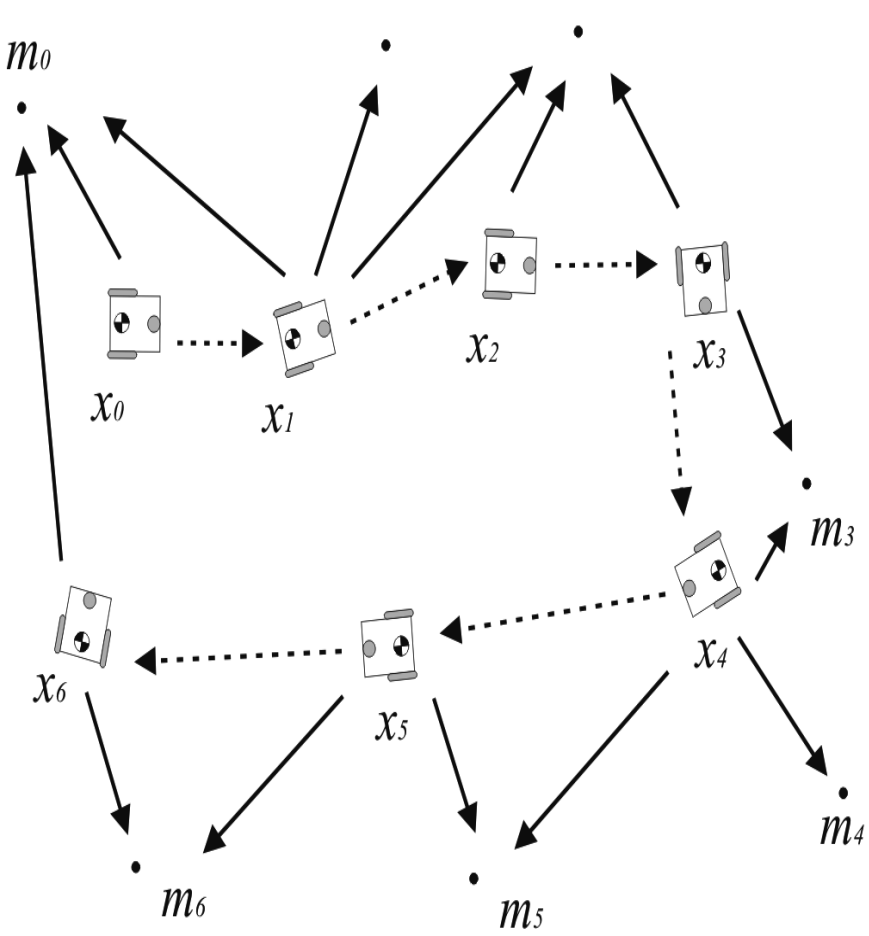


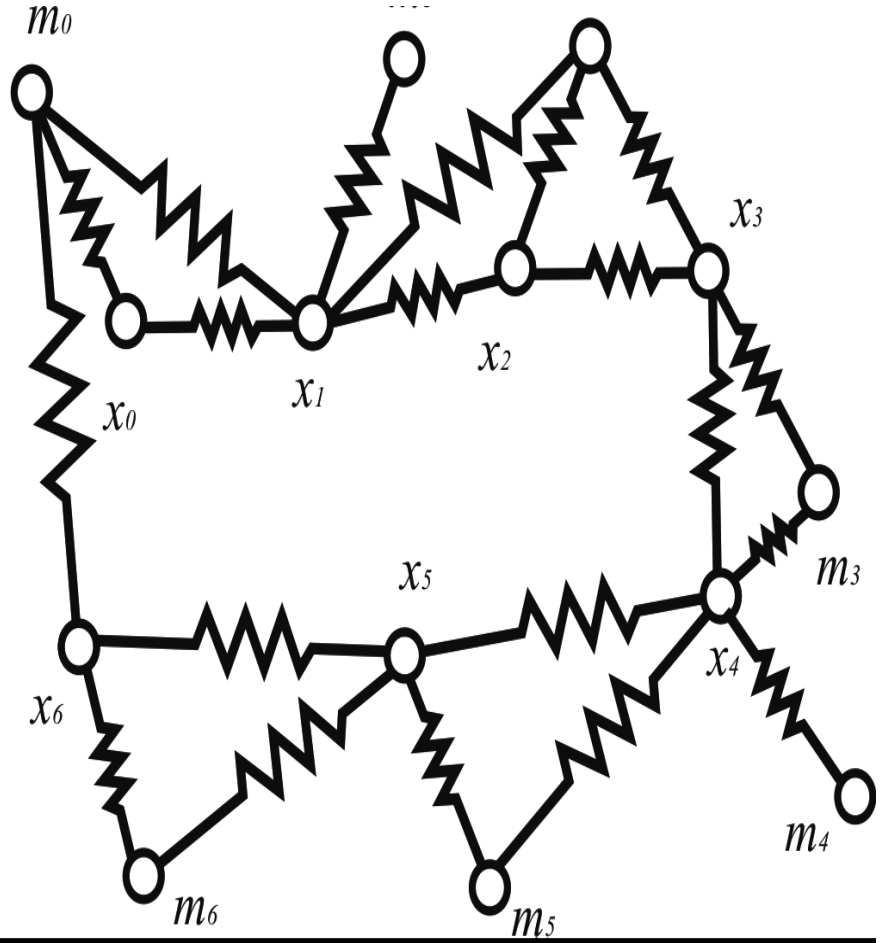
UAV Example


dataAssoc.vcg* - RViz
File Panels Help
Cam0_dataAssoc
Cam1_dataAssoc
Cam2_dataAssoc
Cam3_dataAssoc

Graph Slam

- Treat constraints (motion and measurement) as “soft” elastic springs
- Want to minimize energy in the springs





Linear 1-D Graph Slam

- Solution: List all constraints in a linear system
- Absolute constraint: $X(0) = Q$ - starting position
- Movement Constraints: $X(t) = X(t-1) + Dx(t)$
- Measurement Constraints: Use distance measure to landmark:

  Landmark $L(k) = X(t) + N$

- Each constraint can be an estimate based upon a probability distribution

Linear 1-D Graph Slam

- For exact solution, we need 1 constraint per unknown
- Example: $X(0) = -3$; $X(1) = X(0) + 5$; $X(2) = X(1) + 3$
- Each constraint is a linear equation in the unknowns
- Solution:

$$\begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} X(0) \\ X(1) \\ X(2) \end{bmatrix} = \begin{bmatrix} -3 \\ 5 \\ 3 \end{bmatrix}$$

$$A * X = B$$

$$X = A^{-1} * B$$

$$X = [-3 \quad 2 \quad 5]$$

$$A^{-1} = \begin{matrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{matrix}$$

Linear 1-D Graph Slam

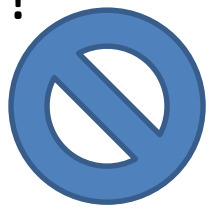
- Now add landmark constraints, linking position with map:
- At X(0) see L(0) at distance 10; At X(1) see L(0) at distance 5; at X(2) see L(0) at distance 2
- Solution:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ L(0) \end{bmatrix} = \begin{bmatrix} -3 \\ -5 \\ -3 \\ -10 \\ 5 \\ 2 \end{bmatrix}$$

$A * X = B$

Cannot Invert A !

$X = A^{-1} * B$



X = [-3 2 5 7]

Linear 1-D Graph Slam

For overdetermined system, compute PseudoInverse Matrix:

$A * X = B$

$A^T * A * X = A^T * B$; A^T = transpose of A

$X = (A^T * A)^{-1} * A^T * B$; $A^T A$ guaranteed to be square

$(A^T * A)^{-1} * A^T$ is PseudoInverse

Solution:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ L(0) \end{bmatrix} = \begin{bmatrix} -3 \\ -5 \\ -3 \\ -10 \\ 5 \\ 2 \end{bmatrix}$$

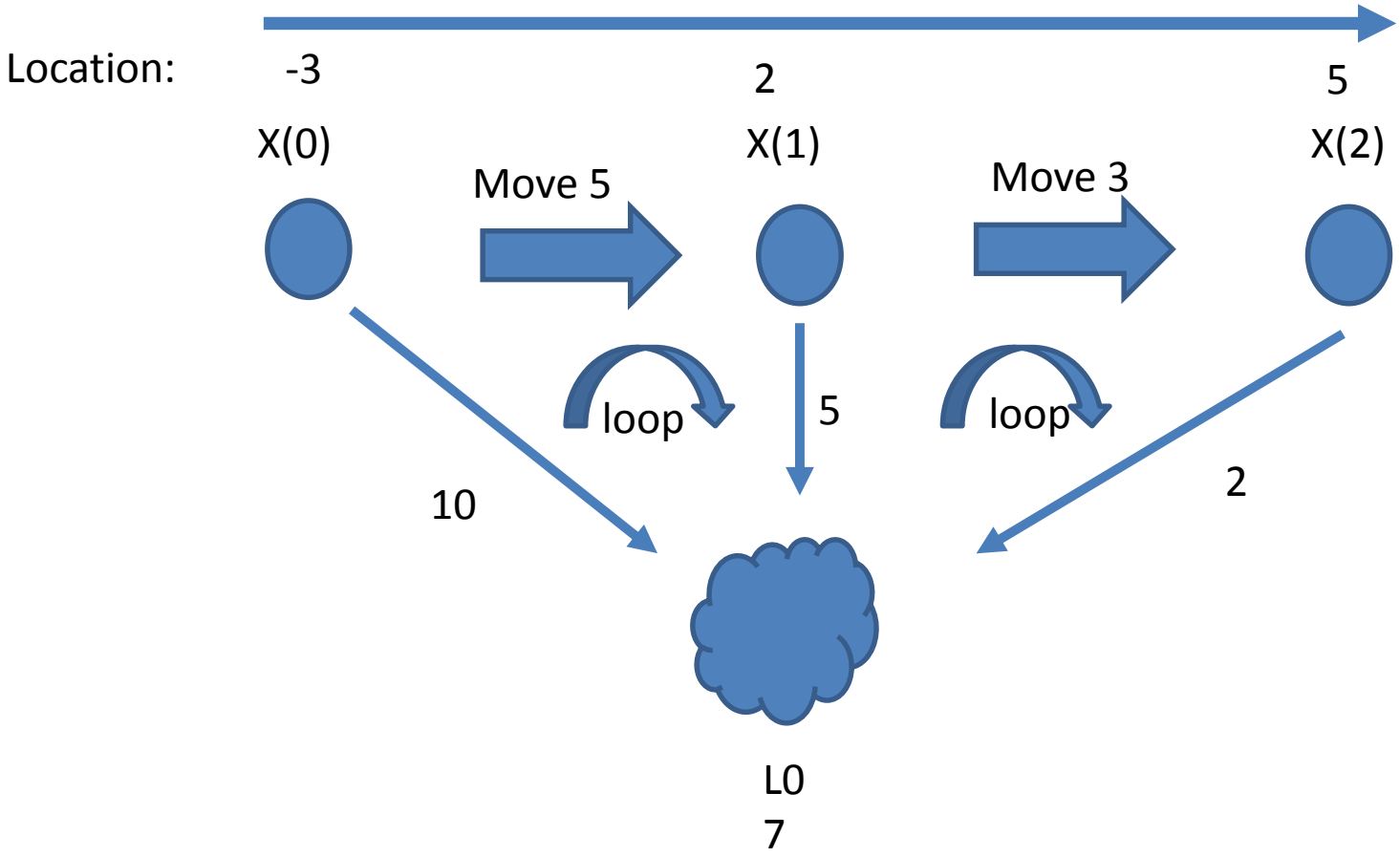
X = [-3 2 5 7]

Matlab Code: Least Squares

```
A=[1 0 0 0;-1 1 0 0; 0 -1 1 0; 1 0 0  -1;0 1 0 -1; 0 0 1 -1]
B=[-3; 5; 3; -10; -5; -2]

ATA = transpose(A)*A
ATB = transpose(A)*B
X= inv(ATA)*ATB;
disp('Solution:')
disp(X)
```

X(0) = -3; X(1) = 2; X(2)=5; L(0) = 7

```
A =
     1     0     0     0
    -1     1     0     0
     0    -1     1     0
     1     0     0    -1
     0     1     0    -1
     0     0     1    -1
B =
    -3
     5
     3
   -10
    -5
    -2
ATA =
     3    -1     0    -1
    -1     3    -1    -1
     0    -1     2    -1
    -1    -1    -1     3
ATB =
   -18
    -3
     1
    17
Solution:
    -3
     2
     5
     7
```



Consistent Solution: If L0 is at 7, each loop is consistent

Matlab Code: Inconsistent Measurements?

Include an inconsistent measurement, find BEST solution

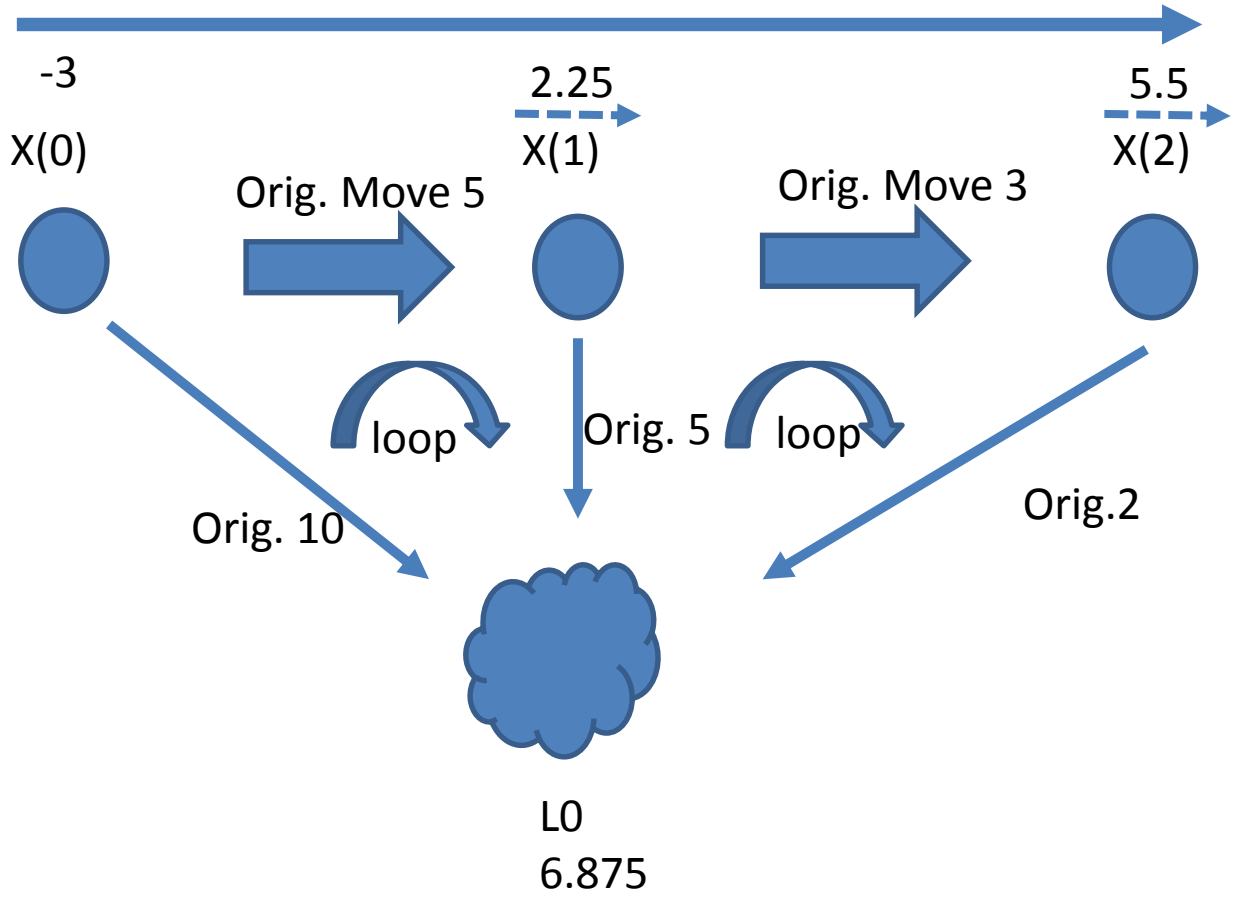
```
A=[1 0 0 0;-1 1 0 0; 0 -1 1 0; 1 0 0  -1;0 1 0 -1; 0 0 1 -1]
B=[-3; 5; 3; -10; -5; -1]
```

Inconsistent measurement
Was X(2) +2 =L(0)
Now (X2) +1 =L(0)

```
ATA = transpose(A)*A
ATB = transpose(A)*B
X= inv(ATA)*ATB;
disp('Solution:')
disp(X)
```

Least squares shifts the robot locations and landmark Location to reduce the overall error. Notice the locations and the landmark are adjusted! However X(0) Remains unchanged as it has an absolute constraint (X(0) = fixed location)

```
A =
     1     0     0     0
    -1     1     0     0
     0    -1     1     0
     1     0     0    -1
     0     1     0    -1
     0     0     1    -1
B =
    -3
     5
     3
   -10
    -5
    -1
ATA =
     3    -1     0    -1
    -1     3    -1    -1
     0    -1     2    -1
    -1    -1    -1     3
ATB =
   -18
    -3
     1
    16
Solution:
    -3
     2.125
     5.5
     6.875
```



Inconsistent Measurement Solution: Adjust L0, X(1), X(2) to minimize error in measurements and movement. Now L0 = X(2) + 1.375, closer to measured value = 1. Note ALL relative values are shifted to reduce global error.

Adding Confidence Measures

- Linear Least Squares allows us to include a weighting of each linear constraint.
- If we know something about how confident a measure is, we can include that in the computation
- We weight each constraint by a diagonal matrix where the weights are 1/variance for each constraint.
- Highly confident constraints have low variance; 1/variance is large weight.
- Unconfident constraints have high variance; 1/variance is small weight.
- Matrix Formulation for weighted Least Squares (A*X=B):

$$X = (A^T * W * A)^{-1} * A^T * W * B$$

Weights amplify/attenuate measurements based on measurements variance

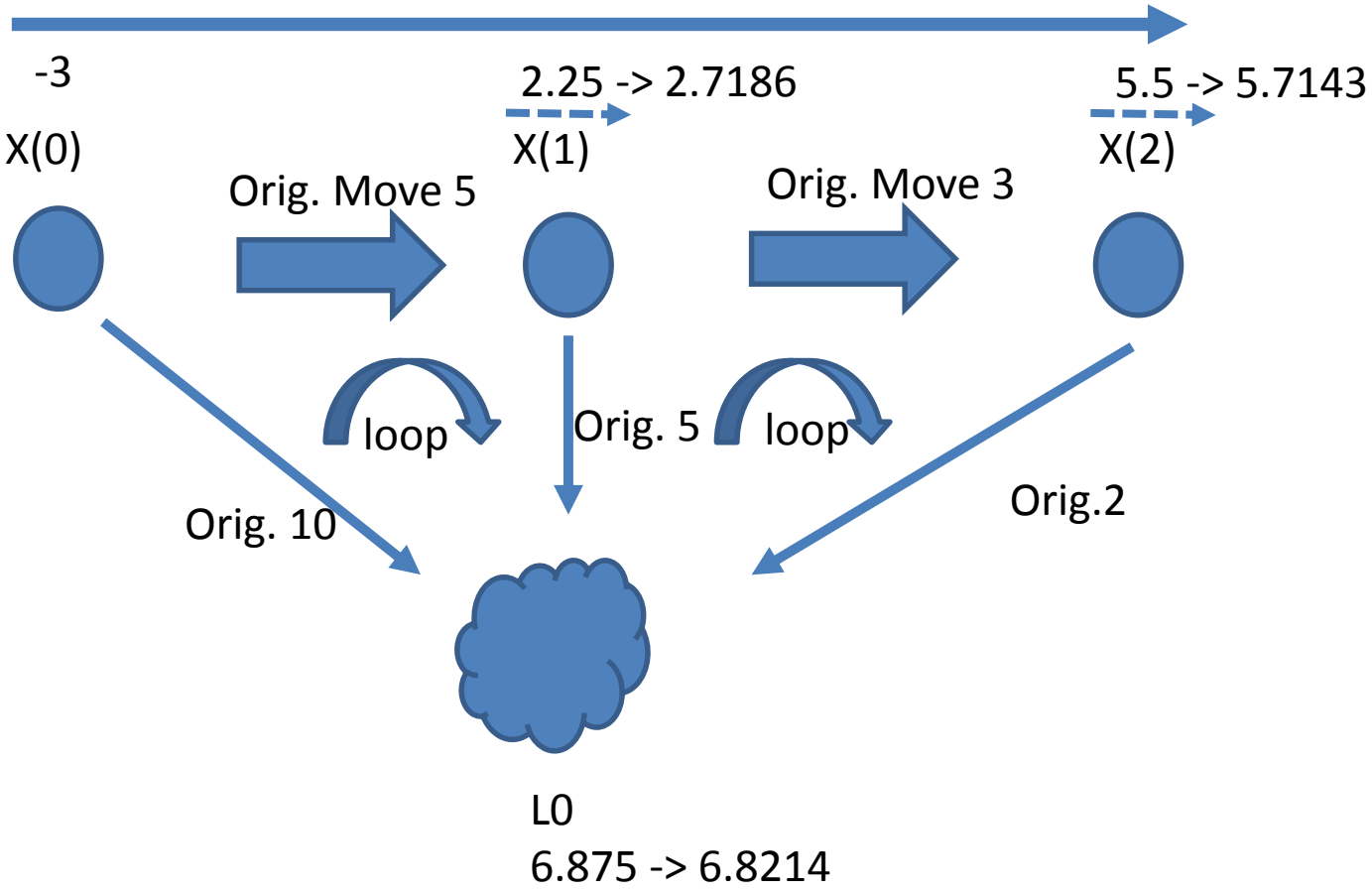
Matlab Code: Confidence Weights

Include a weight on a measurement, scaled by 1/variance

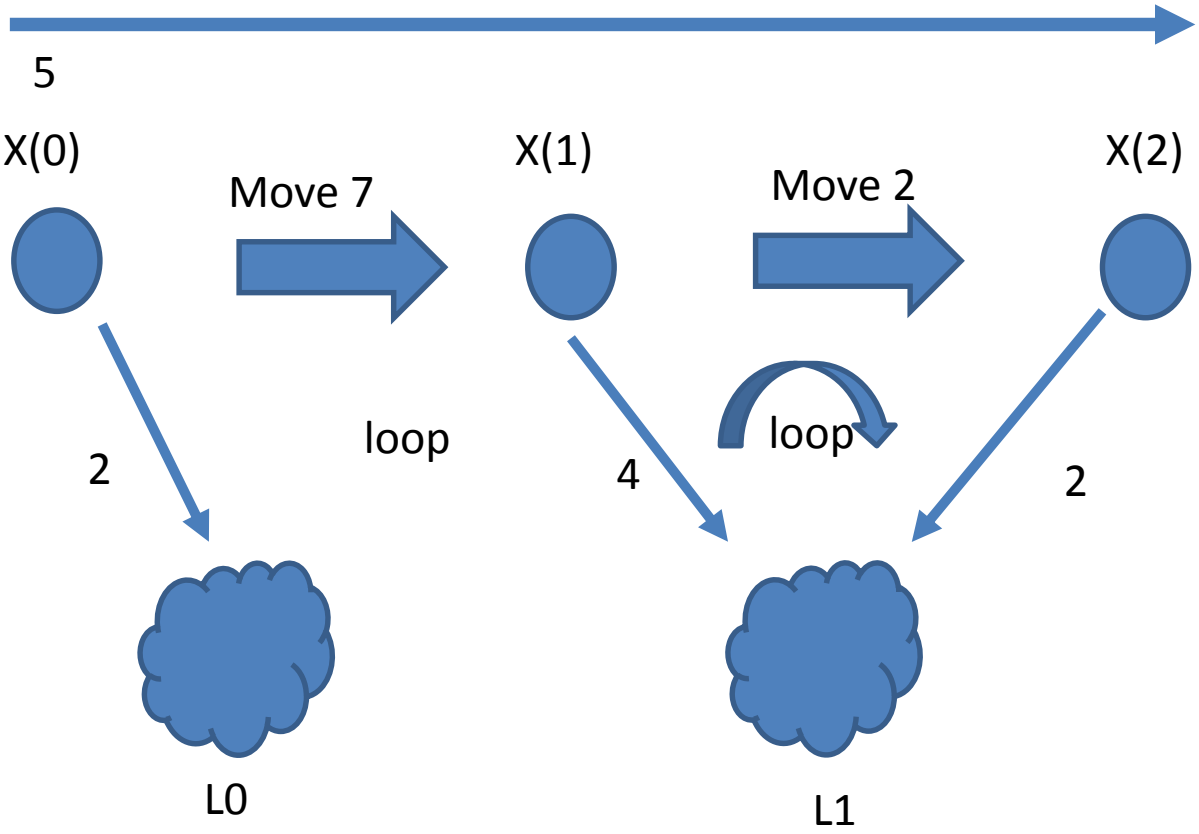
```
% x0= -3, x1= x0+5; x2=x1+3; xo sees L0 at 10; x1 sees L0 at
5, x2 sees L0 at 1
% we also have MUCH higher confidence (small variance) in
% X(2)  measurement of L0
% use weight matrix where diagonals are 1/sigma**2 –
% variance in each measurement.  Use variance of 0.2 (1/.2
=5)

A=[1 0 0 0;-1 1 0 0; 0 -1 1 0; 1 0 0  -1;0 1 0 -1; 0 0 1 -1]
B=[-3; 5; 3; -10; -5; -1]
W=[1 0 0 0 0 0 ; 0 1 0 0 0 0; 0 0 1 0 0 0; 0 0 0 1 0 0; 0 0 0 0 1
0; 0 0 0 0 0 5]

ATWA = transpose(A)*W*A
ATWB = transpose(A)*W*B
X= inv(ATWA)*ATWB;
disp('Solution:')
disp(X)
```

Note: Notice that L0 location from X(2) is now closer to the Measured value of 1 due to confidence weights

```
A =
     1     0     0     0
    -1     1     0     0
     0    -1     1     0
     1     0     0    -1
     0     1     0    -1
     0     0     1    -1
B =
    -3
     5
     3
   -10
    -5
    -1
W =
     1     0     0     0     0     0
     0     1     0     0     0     0
     0     0     1     0     0     0
     0     0     0     1     0     0
     0     0     0     0     1     0
     0     0     0     0     0     5
ATWA =
     3    -1     0    -1
    -1     3    -1    -1
     0    -1     6    -5
    -1    -1    -5     7
ATWB =
   -18
    -3
    -2
    20
Solution:
   -3.0000
    2.1786
    5.7143
    6.8214
```



Confidence Measures Solution: Adjust L0, X(1), X(2) to minimize error in measurements and movement, given GREATER confidence in measurement of L0 from X(2). Now L0 = X(2) + 1.107, even closer to measured value = 1. Note ALL relative values are shifted to reduce global error.



Example: Multiple landmarks: Can incorporate multiple landmarks – each measurement is a constraint. Int his example, X(0)=5, X(1)=X(0)+7, X(2)=X(1)+2; X(0) sees L0 at 2, X(1) sees L1 at 4 and X(2) sees L1 at 2

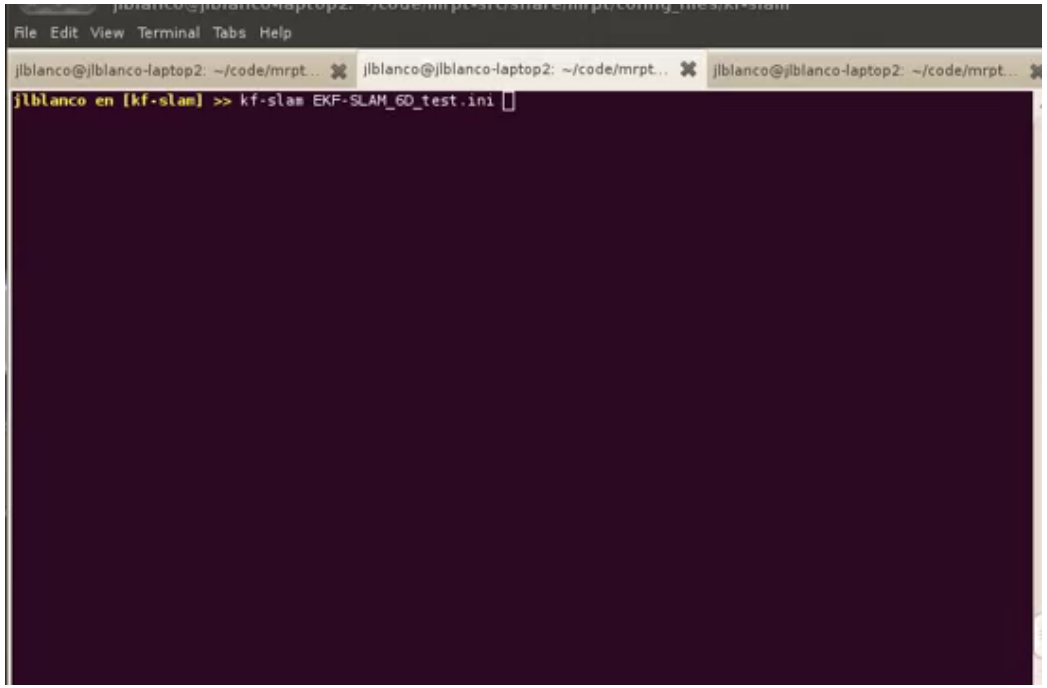
Solution:
X(0)= 5.0000
X(1)= 12.0000
X(2)= 14.0000
L0= 7.0000
L1= 16.0000

If X(1) sees L0 at 4 and X(2) sees L1 at 1

Solution:
X(0)= 5.0000
X(1)= 12.0000
X(2)= 14.3000
L0= 6.0000
L1= 15.6667

SLAM: EKF Slam

- Solution 1: Gaussian Filtering (EKF, UKF)
  - Track a Gaussian belief of the state/landmarks
  - Assume all noise is Gaussian
  - Follow the well-known “predict/correct” approach
- Pros
  - Runs online
  - Well understood
  - Works well for low-uncertainty problems
- Cons
  - Works poorly for high-uncertainty problems
  - Unimodal estimate
  - States must be well approximated by a Gaussian

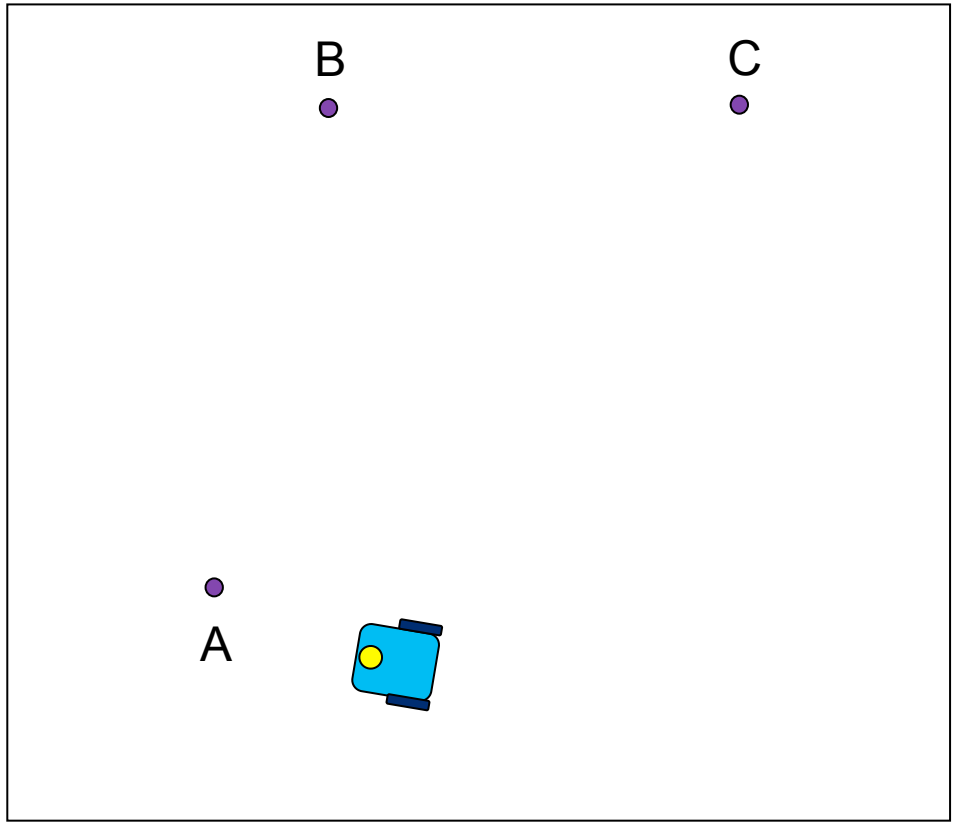


$$\text{Goal: estimate } p\left(\mathbf{x}_{0:K}, \mathbf{m} \middle| \mathbf{u}_{1:K}, \mathbf{z}_{1:N}\right)$$

SLAM | how to do SLAM: Gaussian Filter

- Use internal representations for
  - the positions of landmarks (: map)
  - the camera parameters

- Assumption:
  Robot's uncertainty at starting position is zero

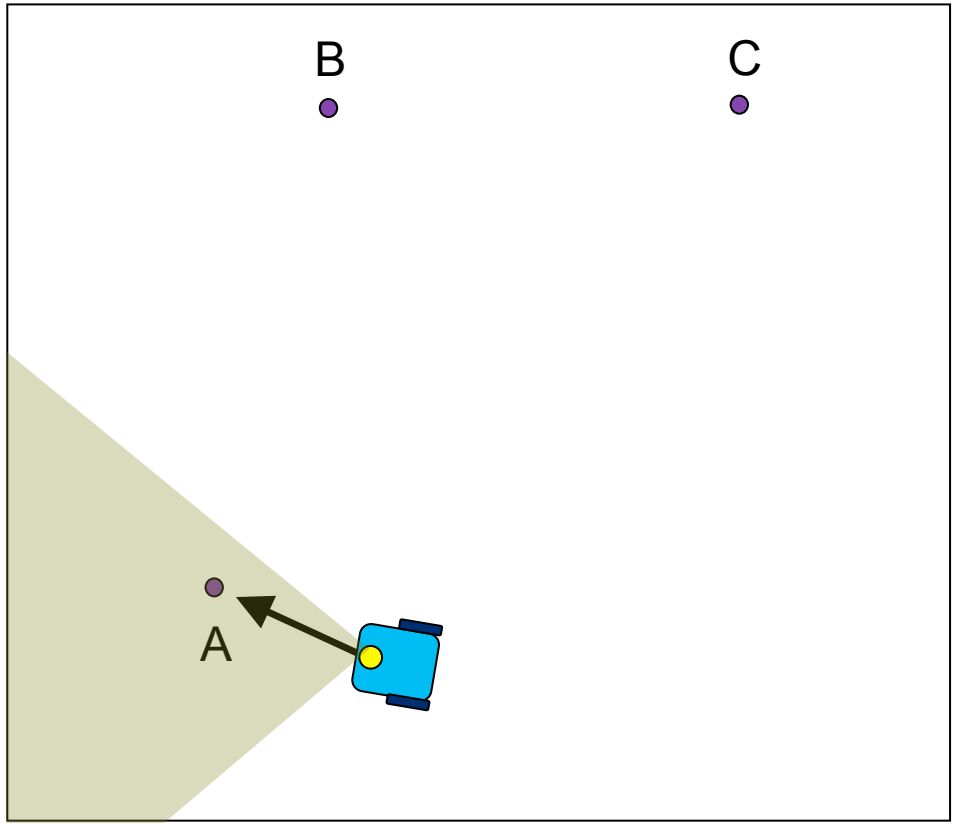


Start: robot has zero uncertainty

SLAM | how to do SLAM: Gaussian Filter

On every frame:

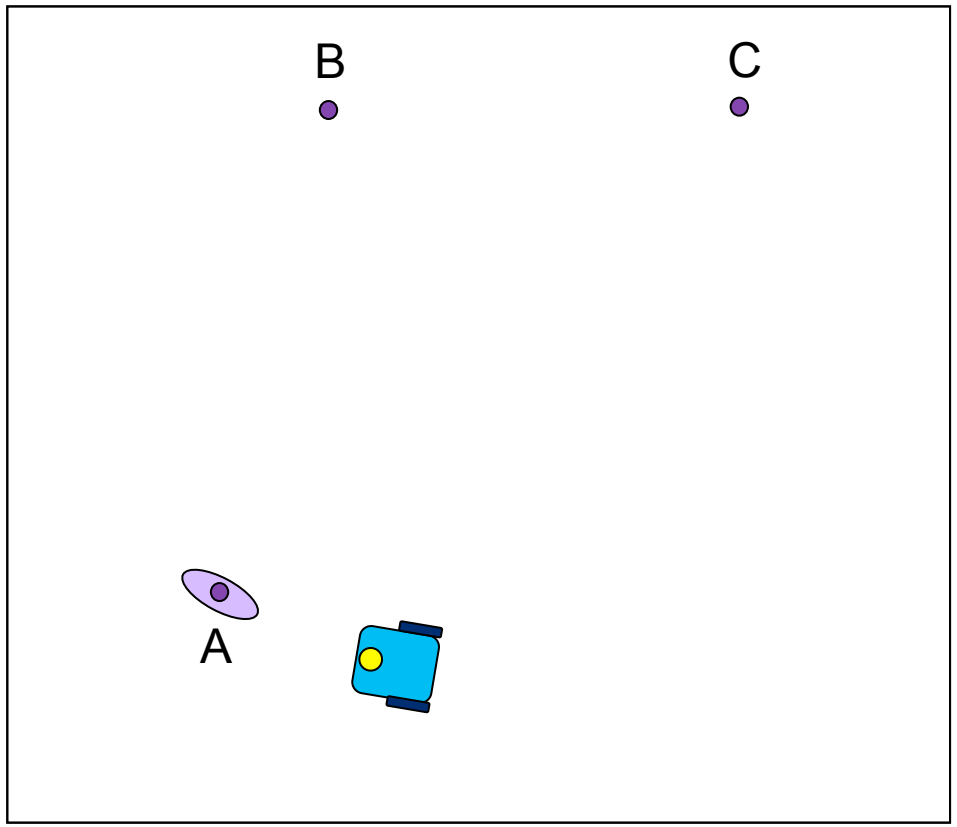
- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations



First measurement of feature A

SLAM | how to do SLAM: Gaussian Filter

- The robot observes a feature which is mapped with an uncertainty related to the **measurement model**



On every frame:

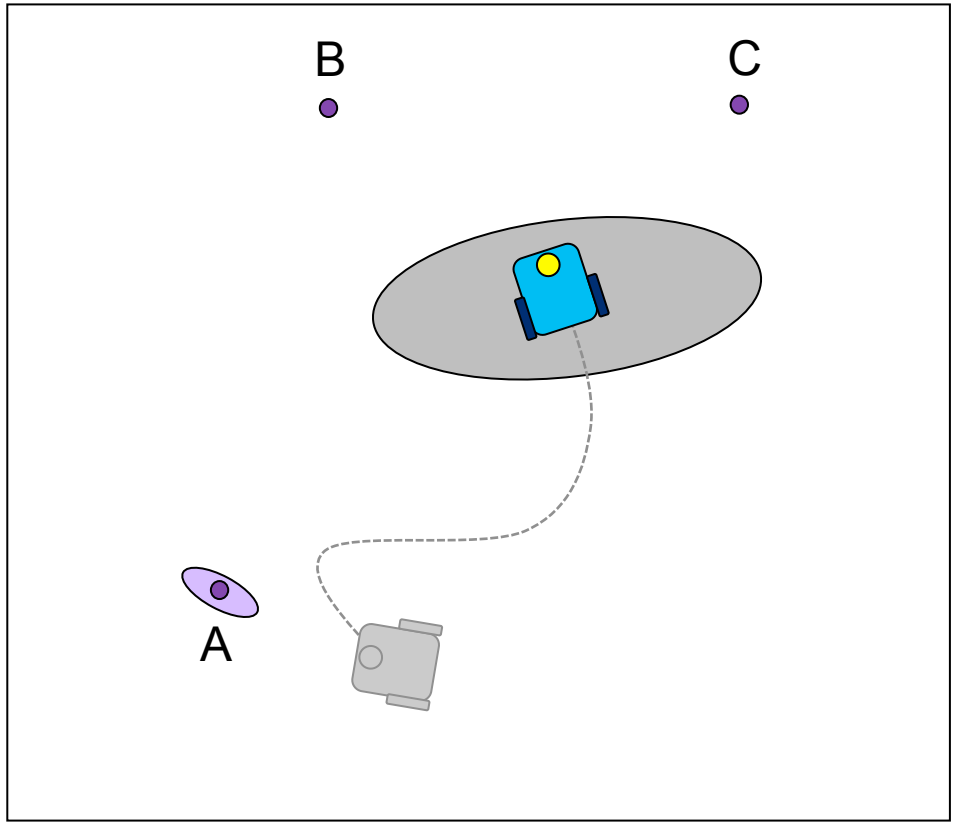
- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations

SLAM | how to do SLAM: Gaussian Filter

- As the robot moves, its pose uncertainty increases, obeying the robot's **motion model**.

On every frame:

- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations



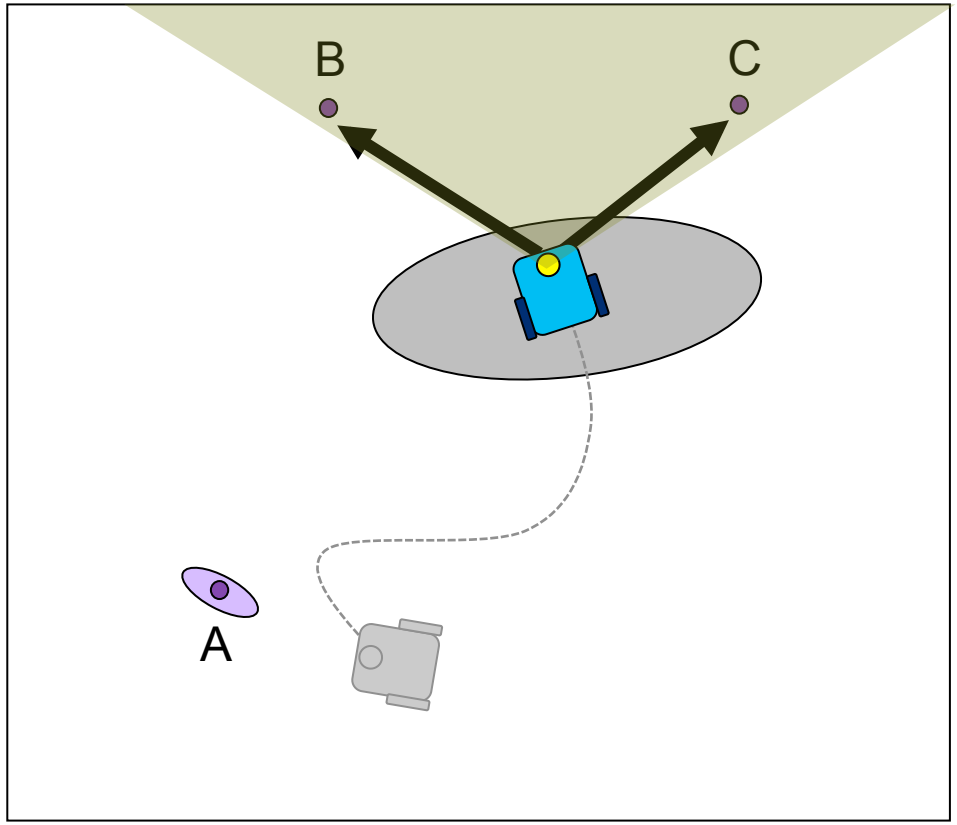
Robot moves forwards: uncertainty grows

SLAM | how to do SLAM: Gaussian Filter

- Robot observes two new features.

On every frame:

- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations



Robot makes first measurements of B & C

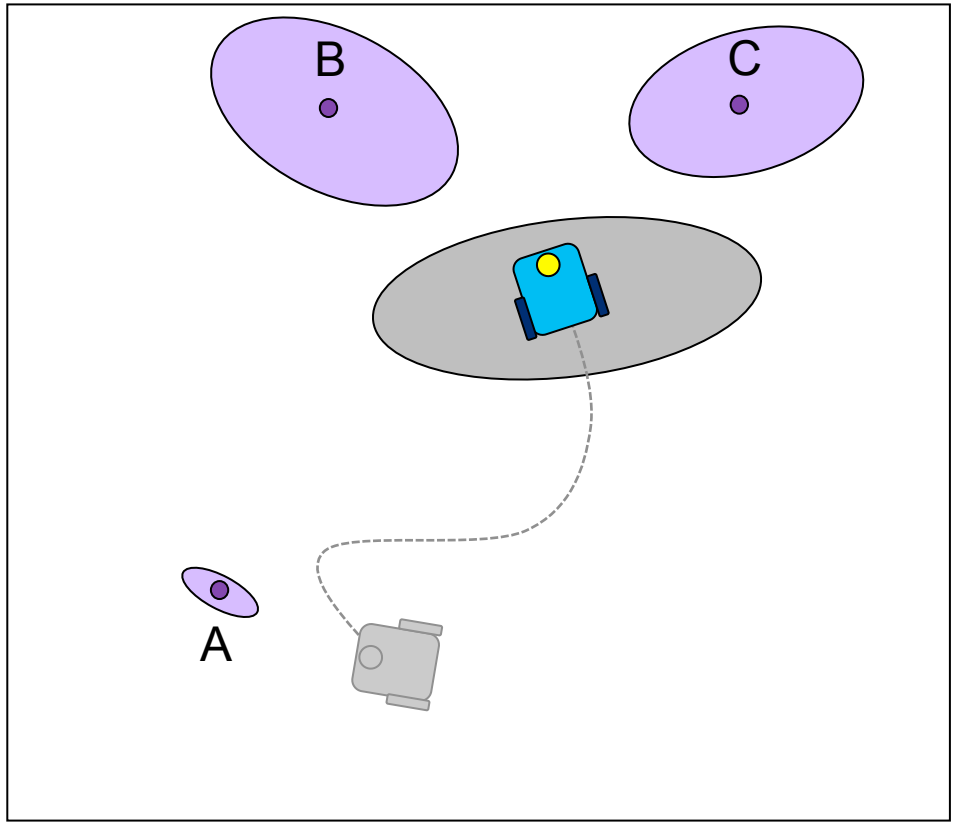
SLAM | how to do SLAM: Gaussian Filter

- Their position uncertainty results from the **combination** of the measurement error with the robot pose uncertainty.

⇨ map becomes **correlated** with the robot pose estimate.

On every frame:

- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations



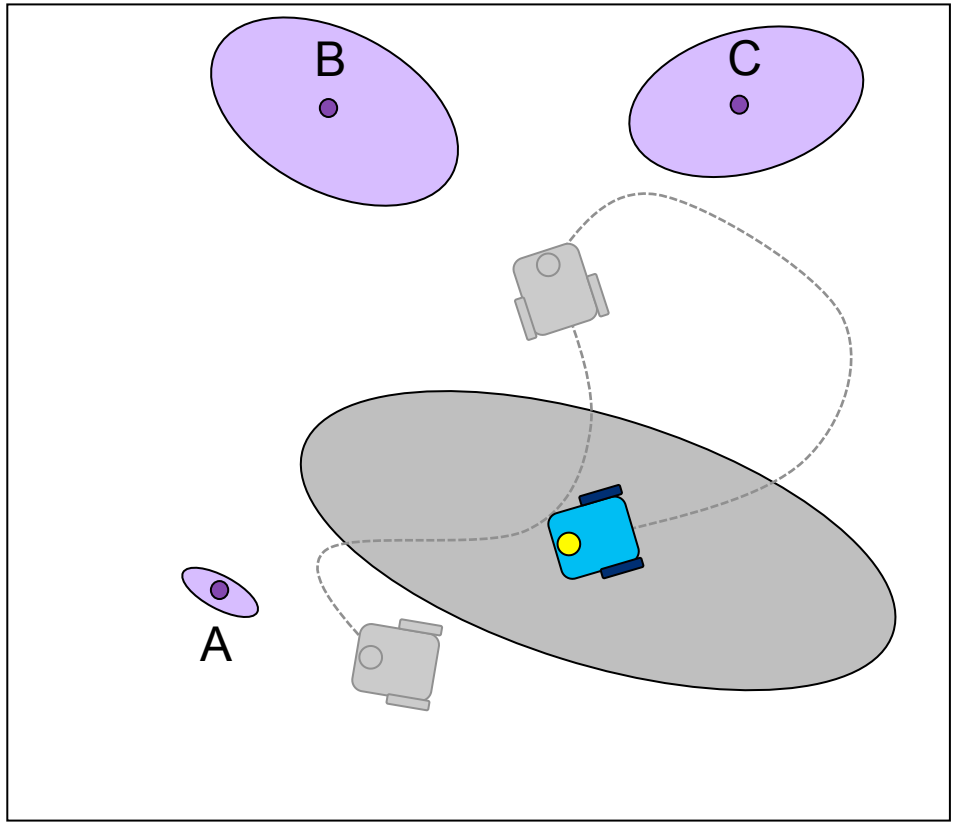
Robot makes first measurements of B & C

SLAM | how to do SLAM: Gaussian Filter

- Robot moves again and its uncertainty increases (motion model)

On every frame:

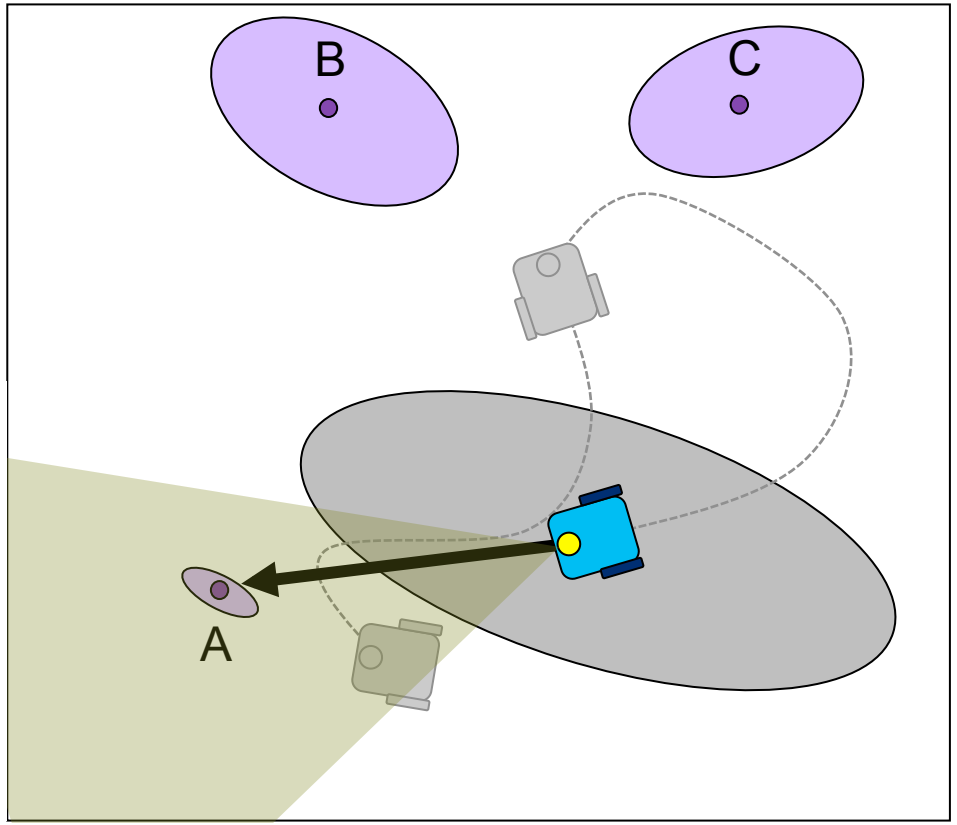
- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations



Robot moves again: uncertainty grows more

SLAM | how to do SLAM: Gaussian Filter

- Robot re-observes an old feature
  ⇨ **Loop closure** detection



On every frame:

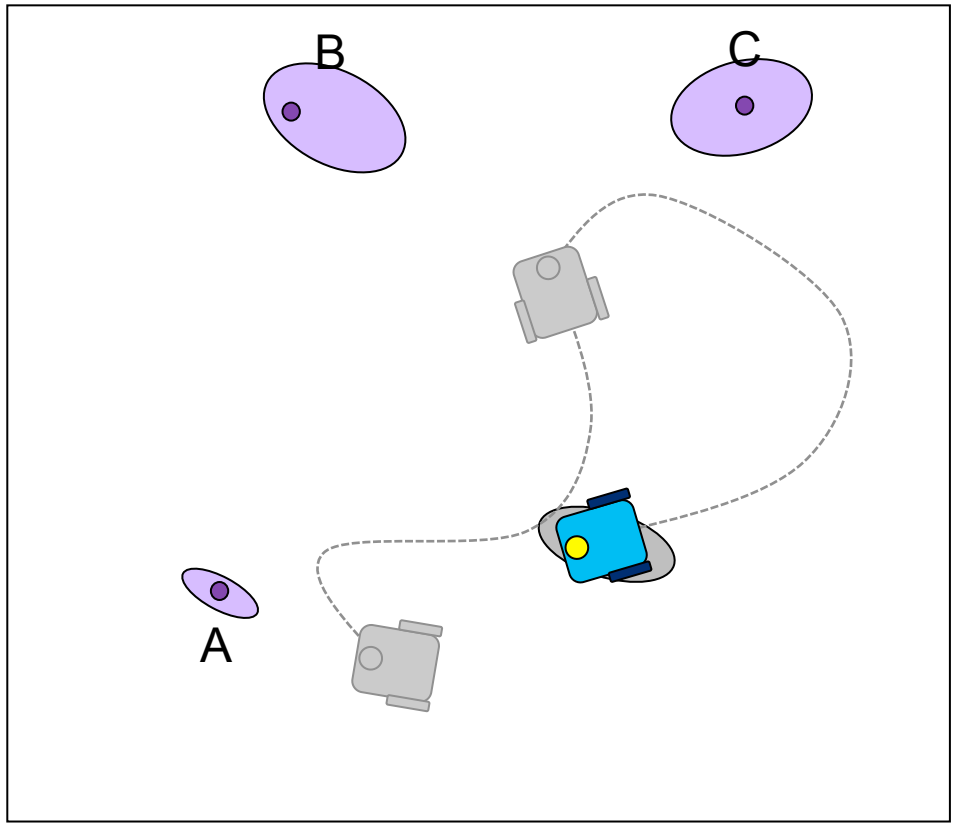
- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations

SLAM | how to do SLAM: Gaussian Filter

- Robot updates its position: the resulting **pose** estimate becomes **correlated** with the feature **location estimates**.
- Robot's uncertainty **shrinks** and so does the uncertainty in the rest of the map

On every frame:

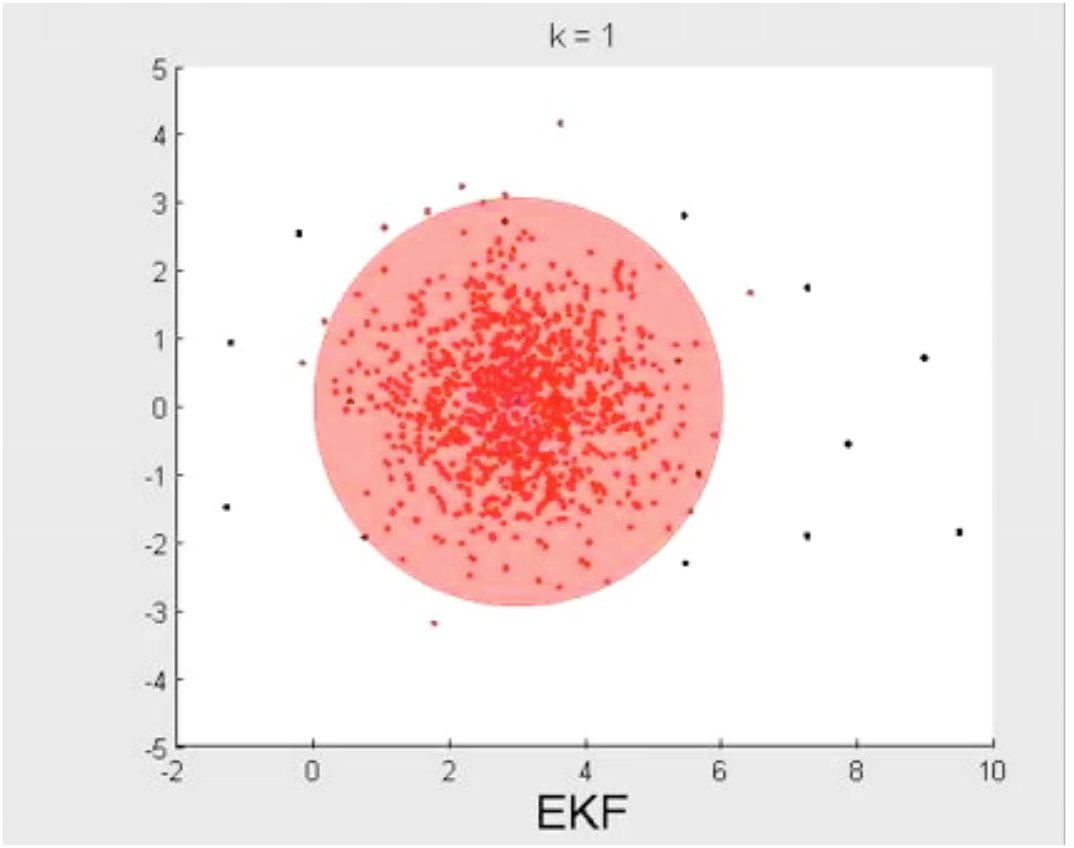
- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations



Robot re-measures A: "loop closure" uncertainty shrinks

SLAM: The three main solutions

- Solution 2: Particle Filtering
  - Represent our belief by a series of samples
  - Follow the well-known “predict/correct” approach
- Pros
  - Noise densities can be from any distribution
  - Works for multi-modal distribtuions
  - Easy to implement
- Cons
  - Does not scale to high-dimensional problems
  - Requires many particles to have good convergence

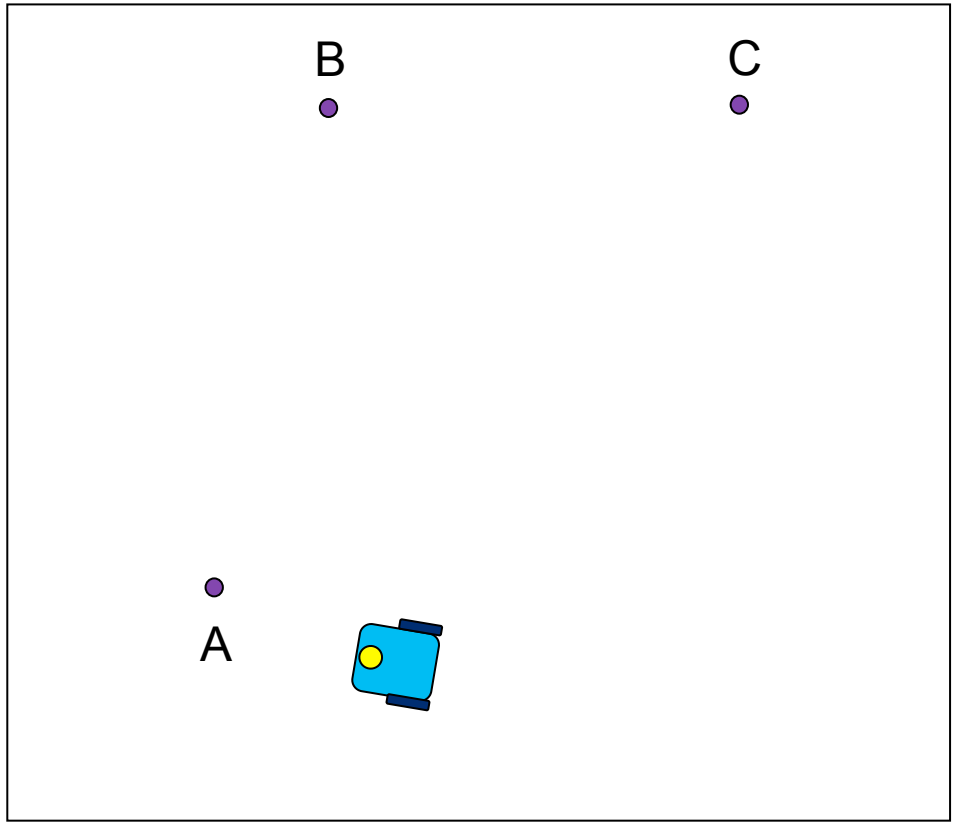


$$\text{Goal: estimate } p\left(\mathbf{x}_{0:K}, \mathbf{m} | \mathbf{u}_{1:K}, \mathbf{z}_{1:N}\right)$$

SLAM | how to do SLAM: Particle Filter

- Use internal representations for
  - the positions of landmarks (: map)
  - the camera parameters

- Assumption:
  Robot's uncertainty at starting position is zero

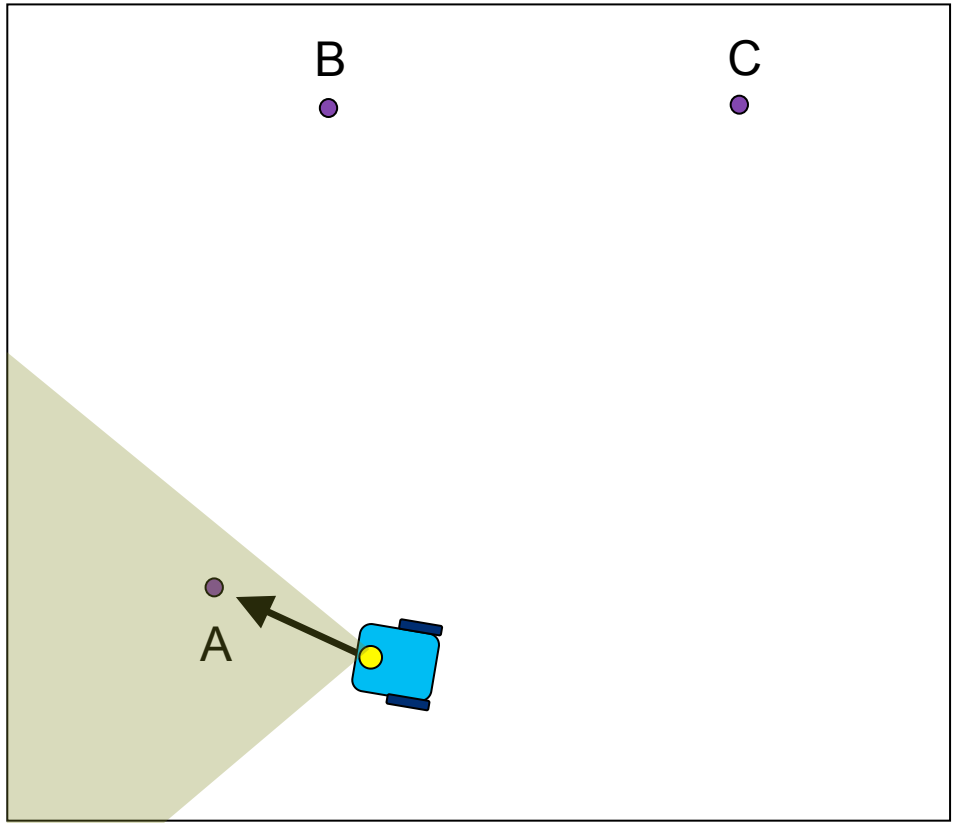


Start: robot has zero uncertainty

SLAM | how to do SLAM: Particle Filter

On every frame:

- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations



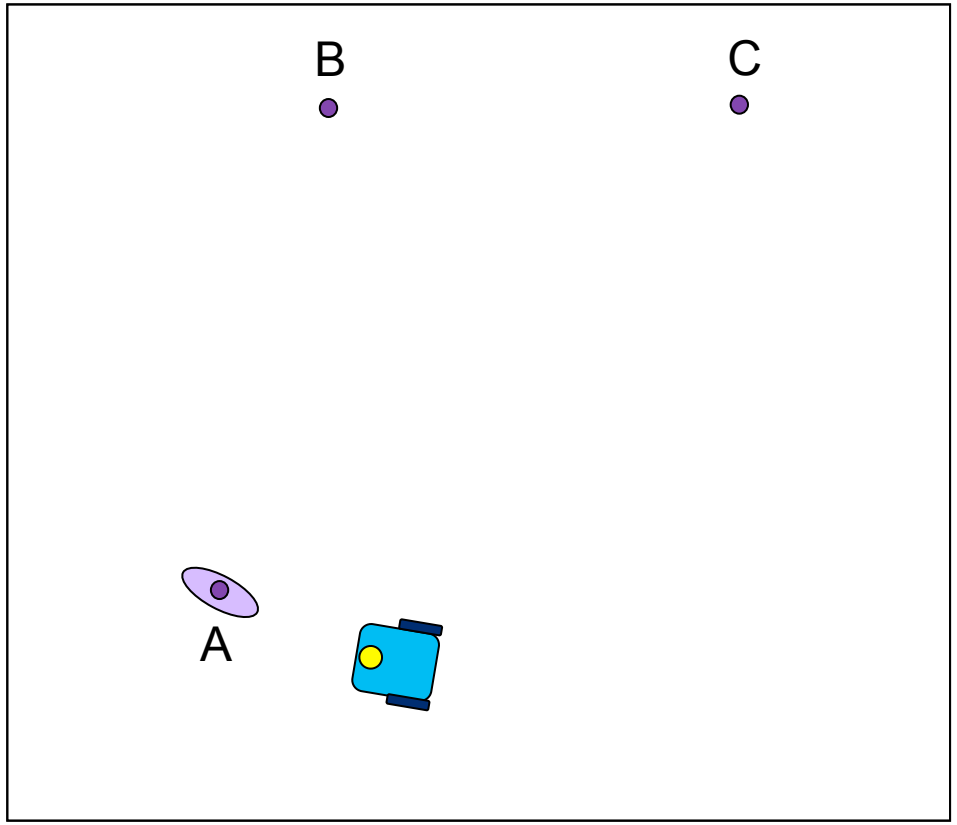
First measurement of feature A

SLAM | how to do SLAM: Particle Filter

- The robot observes a feature which is mapped with an uncertainty related to the **measurement model**

On every frame:

- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations

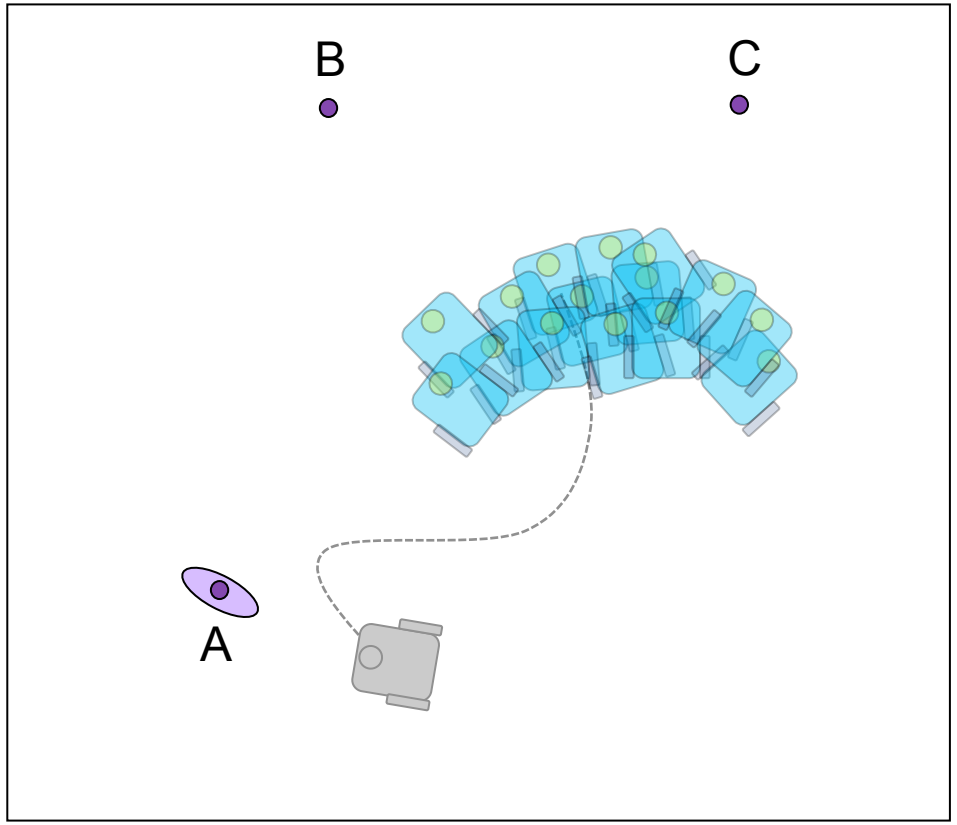


SLAM | how to do SLAM: Particle Filter

- As the robot moves, its pose uncertainty increases, obeying the robot's **motion model**.

On every frame:

- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations



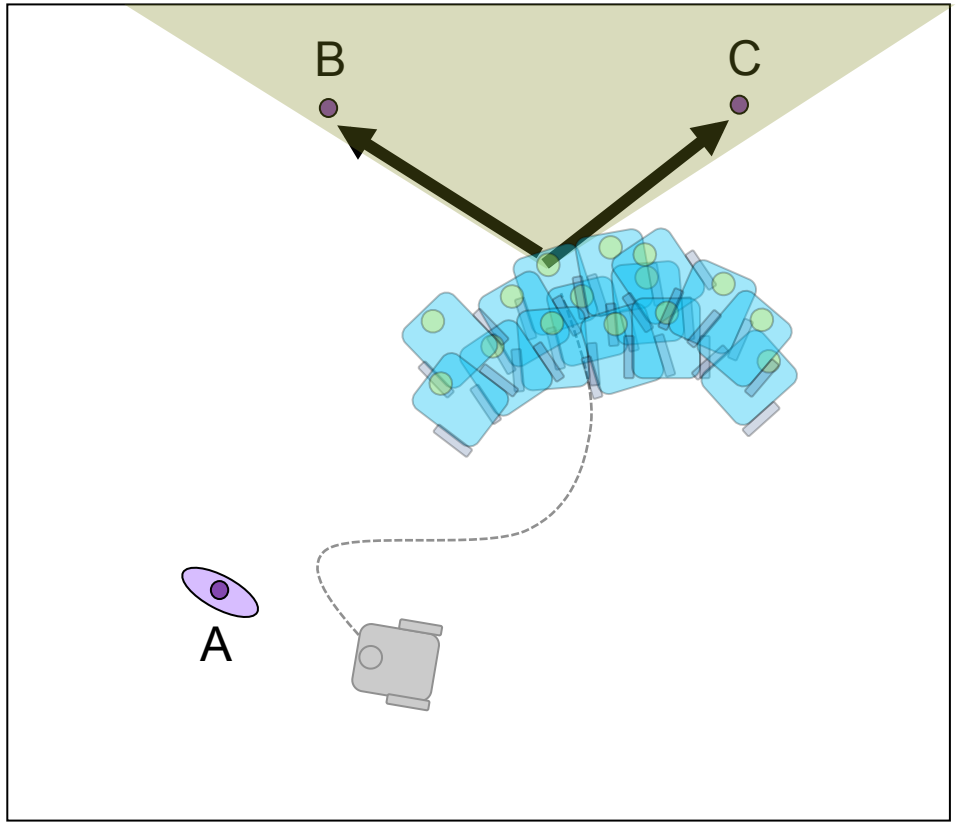
Robot moves forwards: uncertainty grows

SLAM | how to do SLAM: Particle Filter

- Robot observes two new features.

On every frame:

- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations



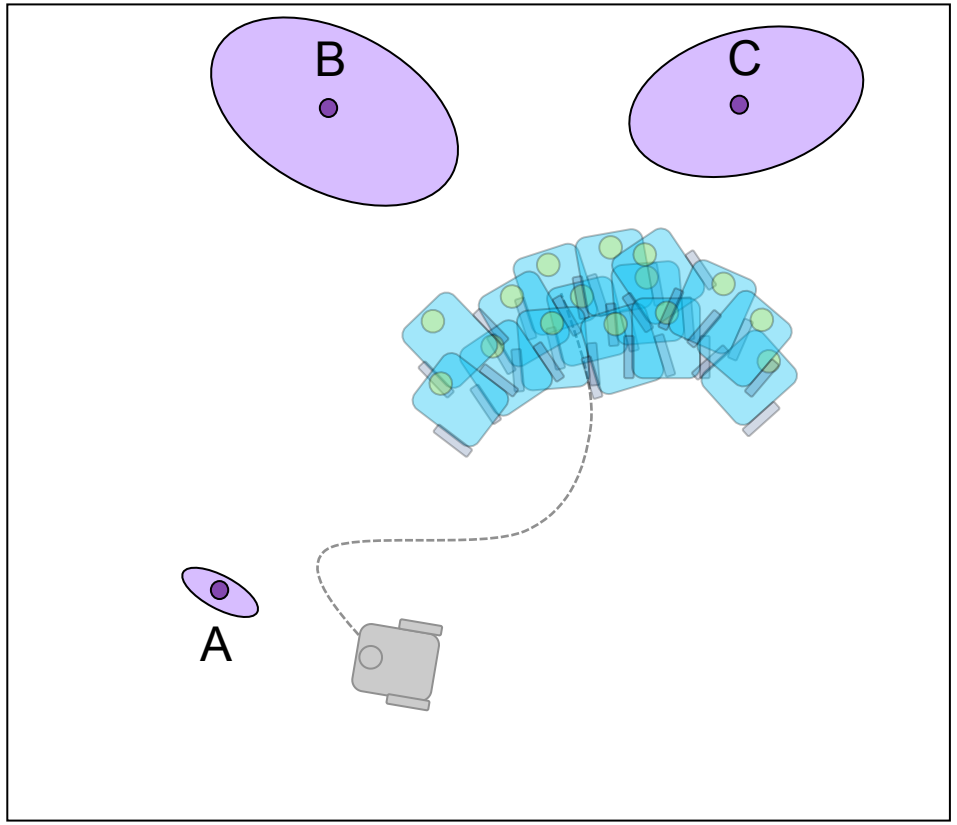
Robot makes first measurements of B & C

SLAM | how to do SLAM: Particle Filter

- Their position uncertainty is encoded for each particle individually

On every frame:

- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations



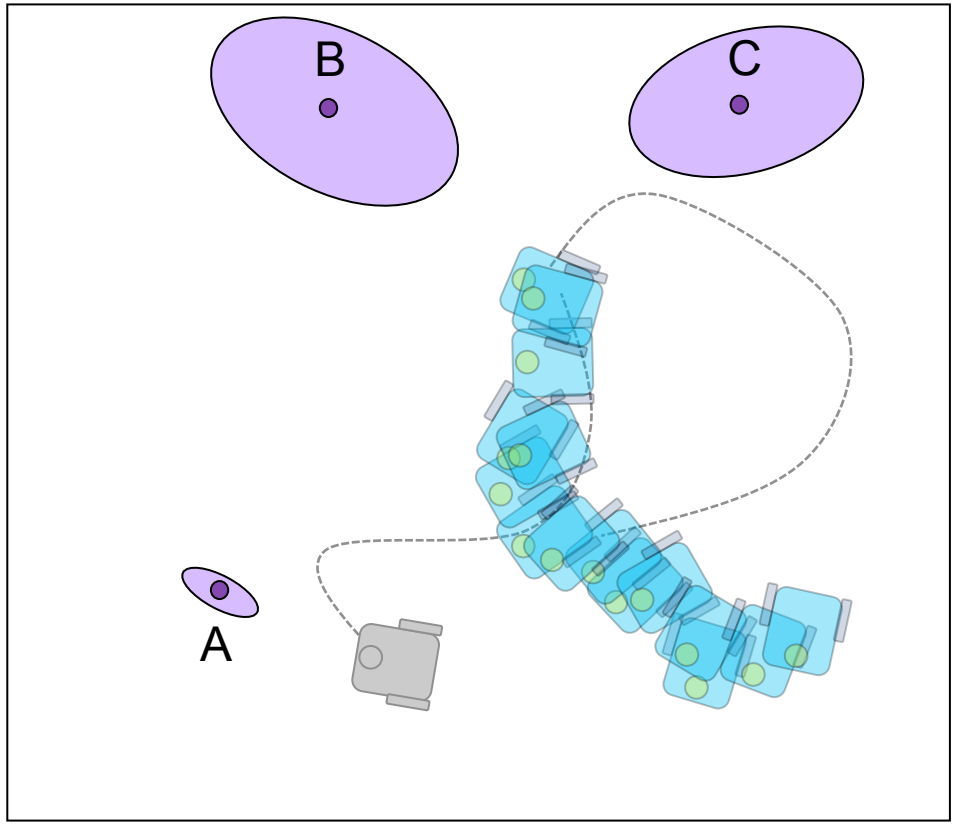
Robot makes first measurements of B & C

SLAM | how to do SLAM: Particle Filter

- Robot moves again and its uncertainty increases (motion model)

On every frame:

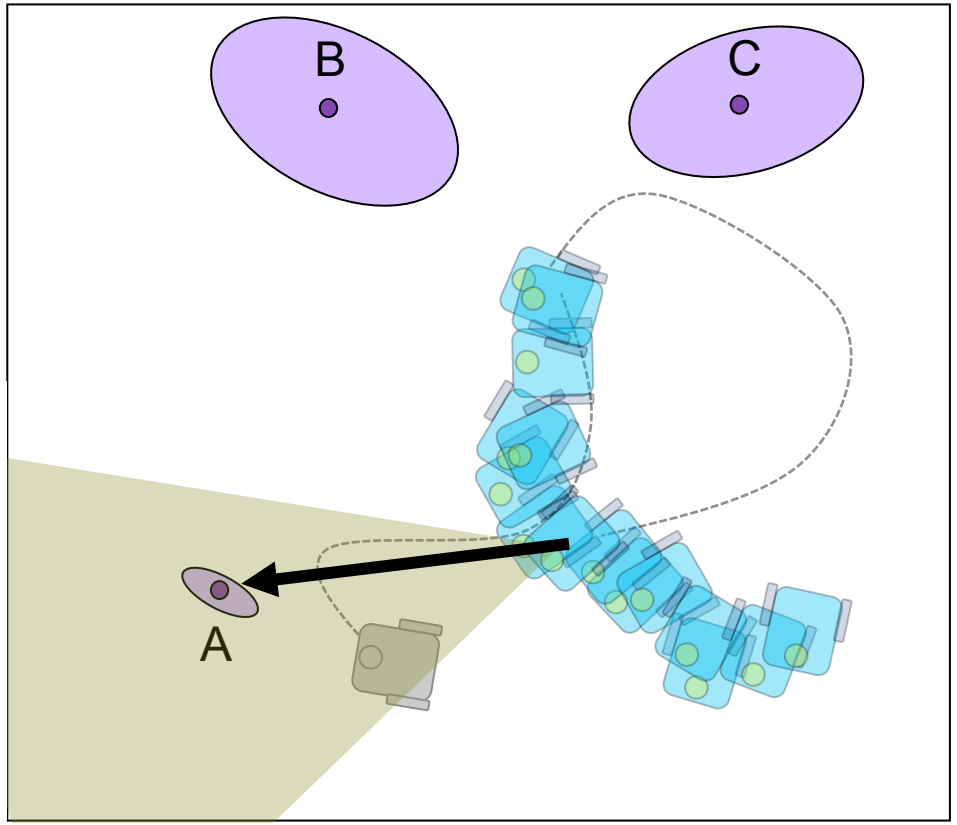
- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations



Robot moves again: uncertainty grows more

SLAM | how to do SLAM: Particle Filter

- Robot moves again and its uncertainty increases (motion model)



On every frame:

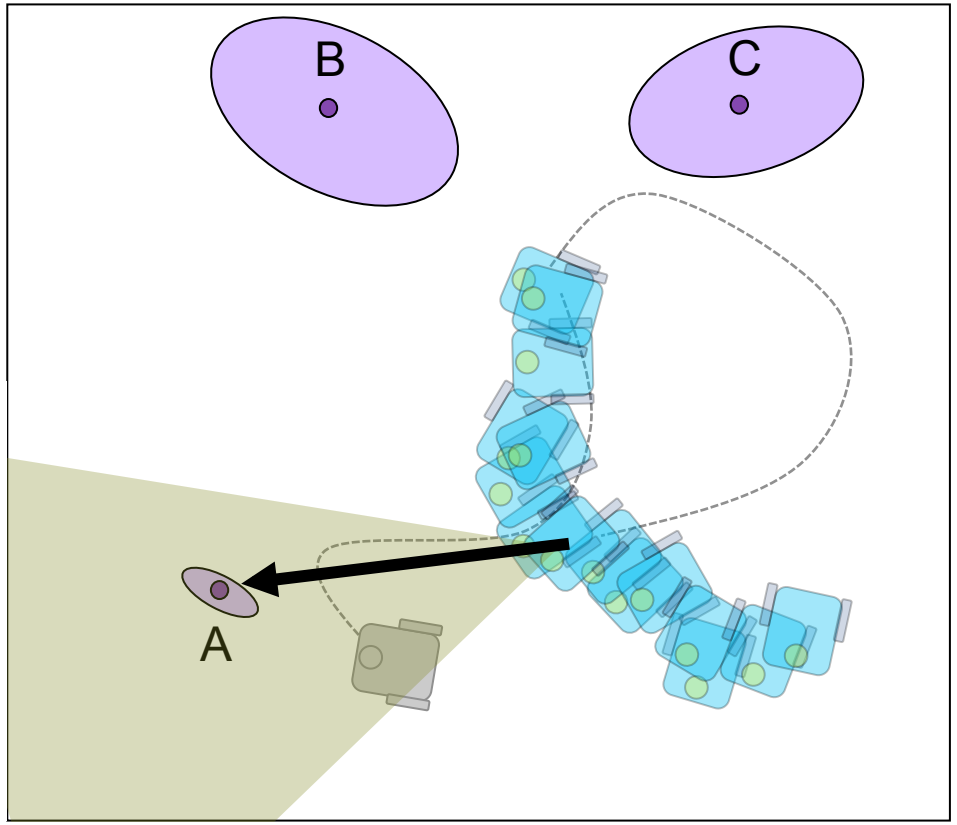
- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations

SLAM | how to do SLAM: Particle Filter

- Robot re-observes an old feature
  ⇨ **Loop closure** detection

On every frame:

- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations

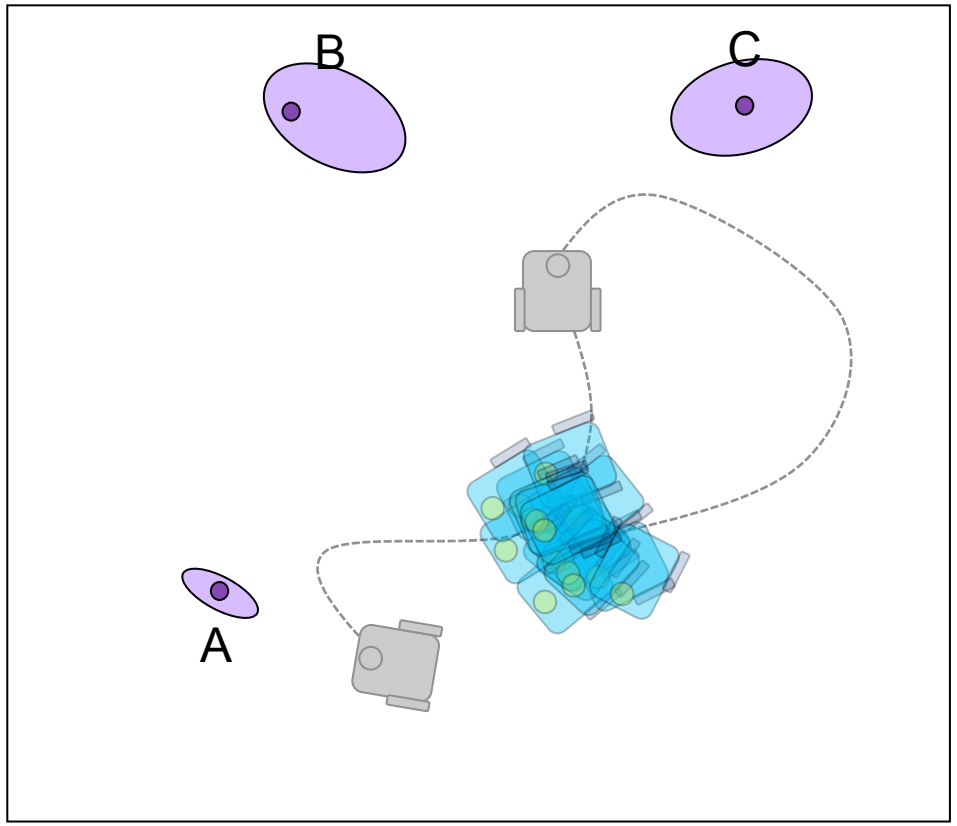


SLAM | how to do SLAM: Particle Filter

- After the measurement, particles are resampled according to the likelihood of the measurements.

On every frame:

- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations



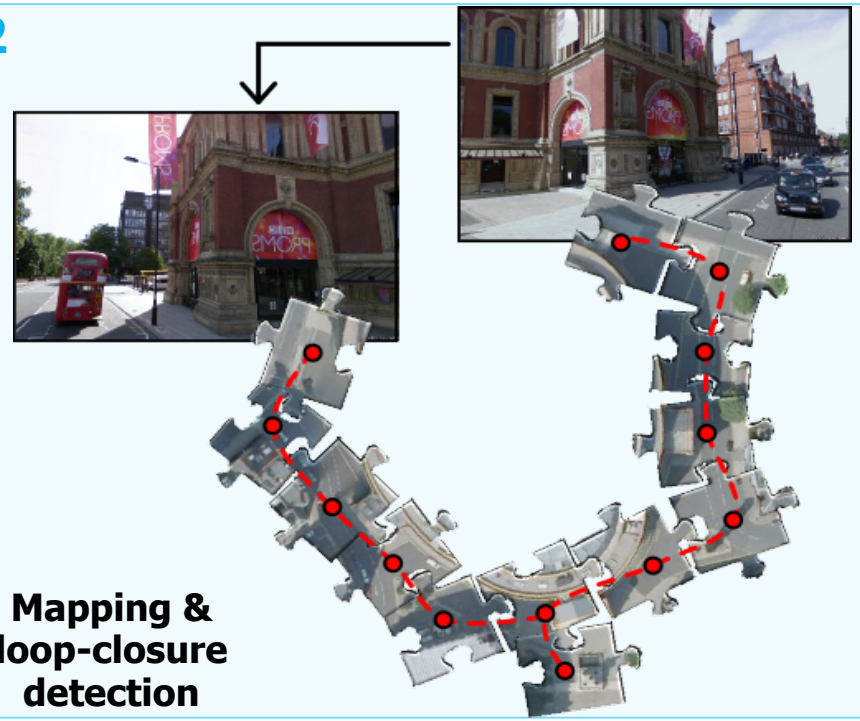
Robot re-measures A: “loop closure” uncertainty shrinks

SLAM Challenges | components for scalable SLAM

1

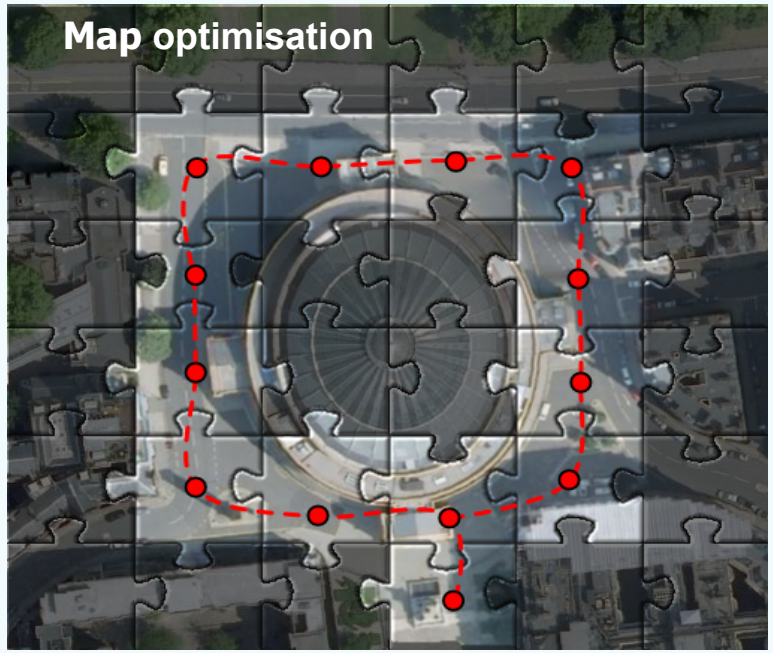
Robust local motion estimation

2

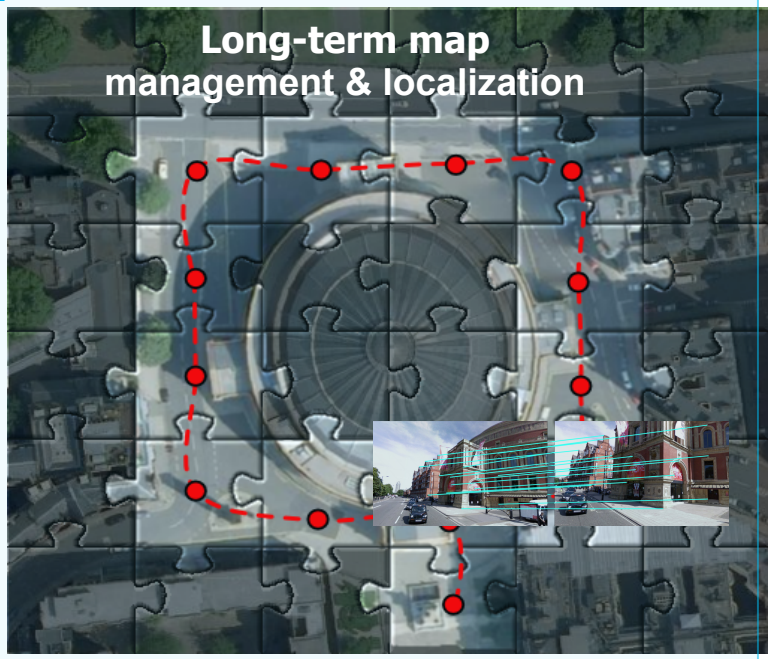


Mapping & loop-closure detection

3



4



[Chli,2009, PhD thesis]