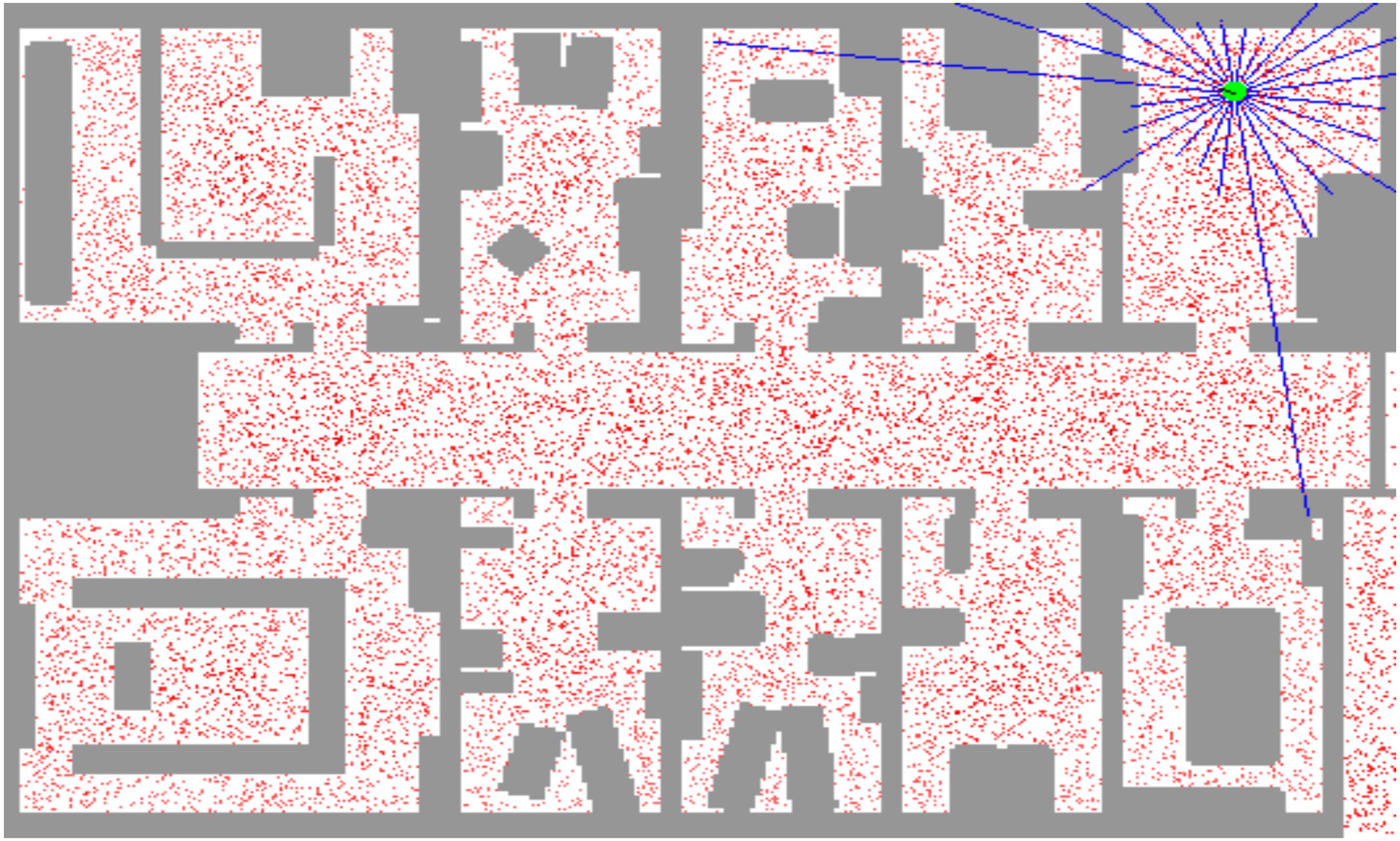
Lecture 17

Simultaneous Localization and Mapping

CS 3630

Many slides from Prof. Harish Ravichandar

Particle Filter Localization (using sonar)



We’ve seen how a robot can localize itself with respect to a known map.

But what if there’s no map??

http://www.cs.washington.edu/ai/Mobile_Robotics//mcl/animations/global-floor-start.gif

The central question for SLAM

Is it possible for a mobile robot to be placed at an **unknown location** in an **unknown environment** and for the robot to incrementally build a consistent map of this environment while simultaneously determining its location within this map?

In other words…

Given:

- The robot's controls $u_{1:T} = \{u_1, u_2, u_3 \ldots, u_T\}$
- The robot's sensing observations $z_{1:T} = \{z_1, z_2, z_3 \ldots, z_T\}$

Compute:

- Map of the environment m
- Path of the robot $x_{0:T} = \{x_0, x_1, x_2 \ldots, x_T\}$

A chicken and egg problem!

- If the robot had a map, localization would be easier.
- If the robot could localize, mapping would be easier.
- … But the robot has neither; it starts from a blank slate.
- Furthermore, the robot must also execute an exploration strategy.

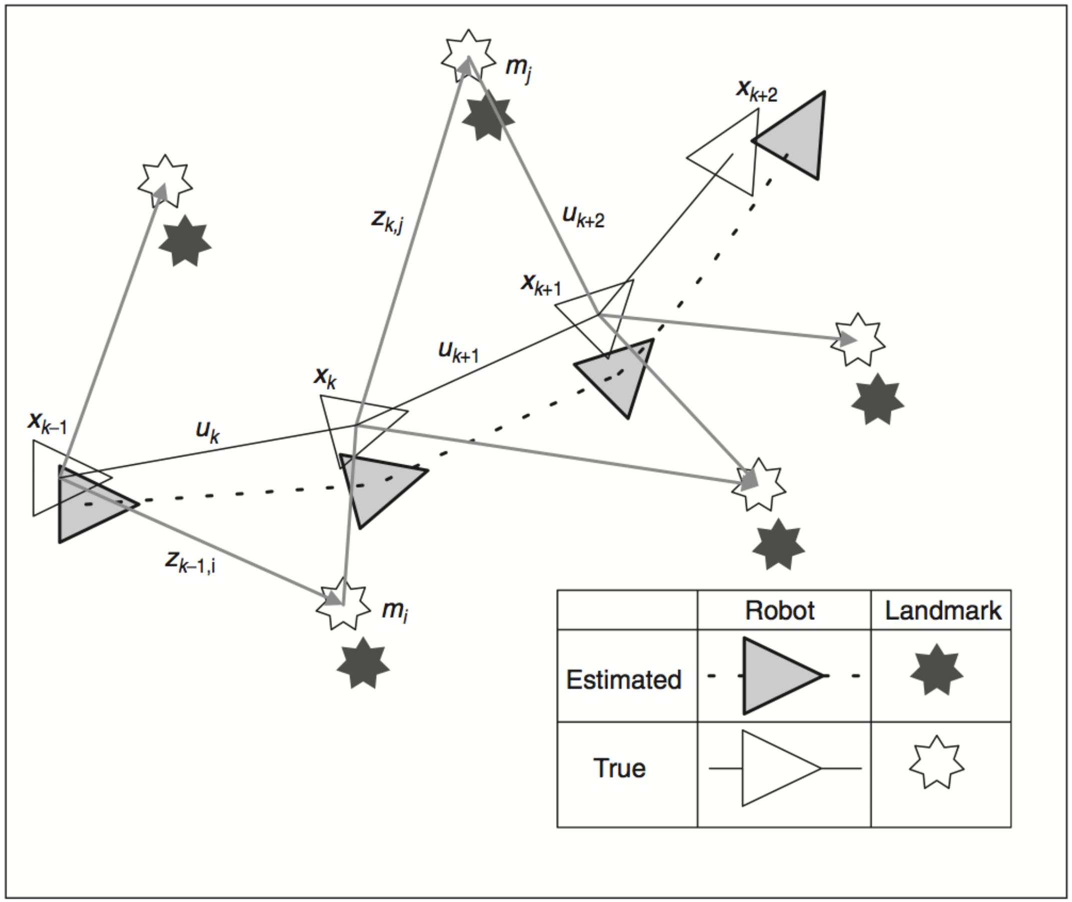
The basic algorithm

1. The robot moves
   - increases the uncertainty on robot pose
   - need a mathematical model for the motion (*motion model)*

2. The robot discovers interesting features in the environment called *landmarks*
   - need a mathematical model to determine the position of the landmarks from sensor data (*inverse observation model)*
   - need to model uncertainty in the location of landmarks

3. The robot observes landmarks that had been previously mapped
   - uses them to correct both self localization and the localization of all landmarks in space
   - uncertainties for both decrease
   - need a model to predict the measurement from predicted landmark location and robot localization (*direct observation model*)

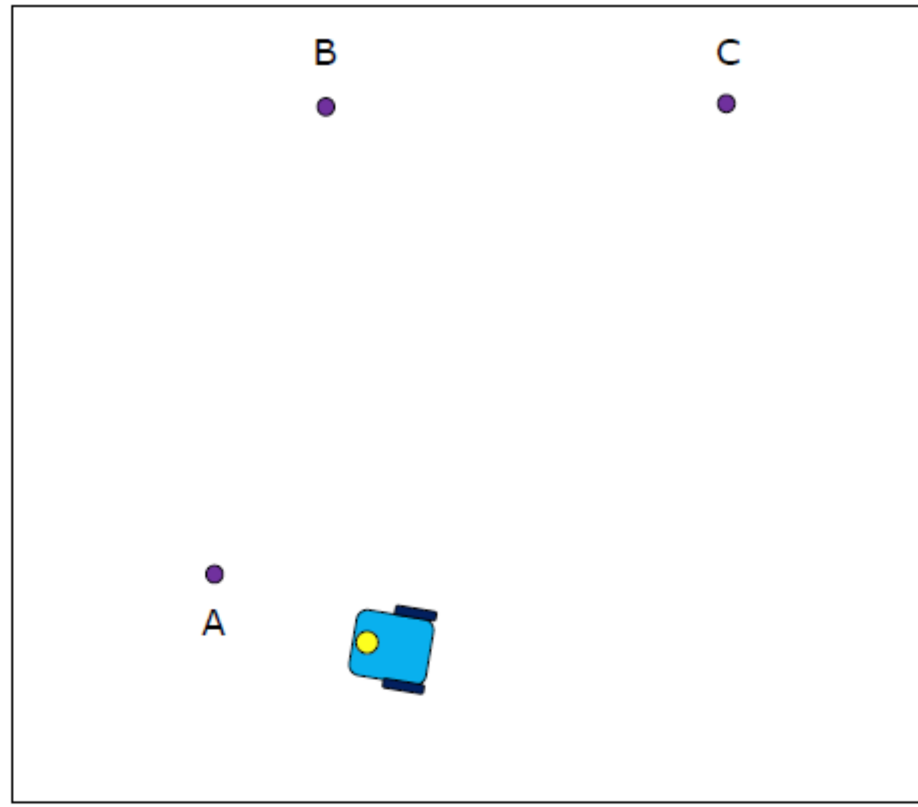
In pictures….



m_j
x_{k+2}
$z_{k,j}$
u_{k+2}
x_{k+1}
u_{k+1}
x_k
x_{k-1}
u_k
$z_{k-1,i}$
m_i
Robot
Landmark
Estimated
True

SLAM Example

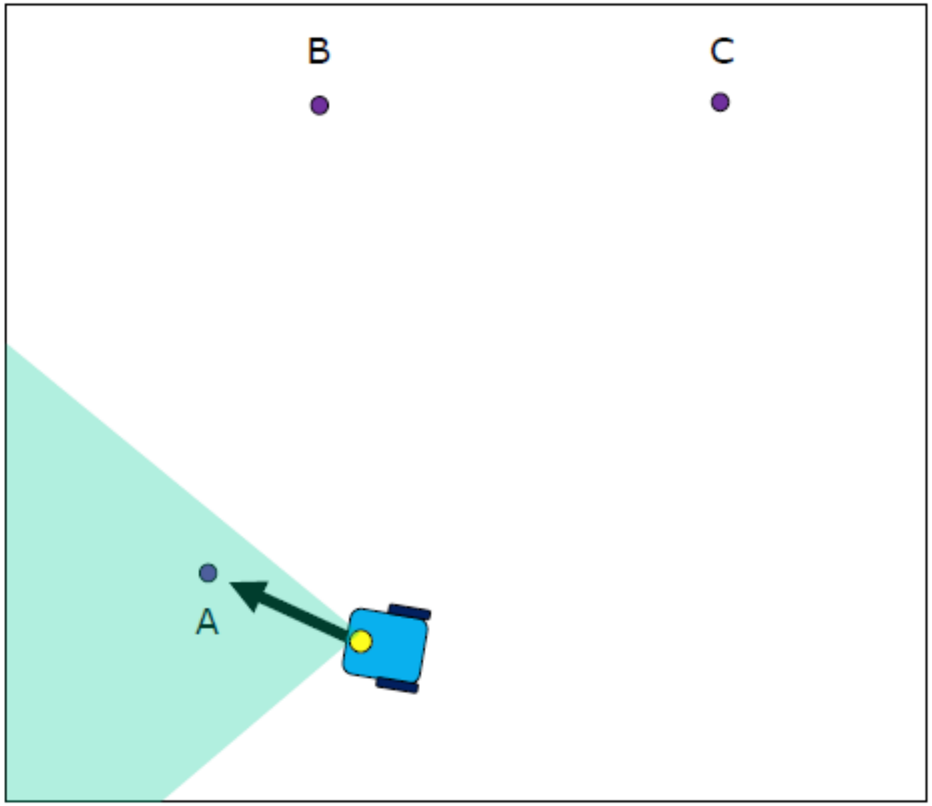
- Assumption: Robot's uncertainty at starting position is zero



Start: robot has zero uncertainty

SLAM Example

On every frame:
- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations



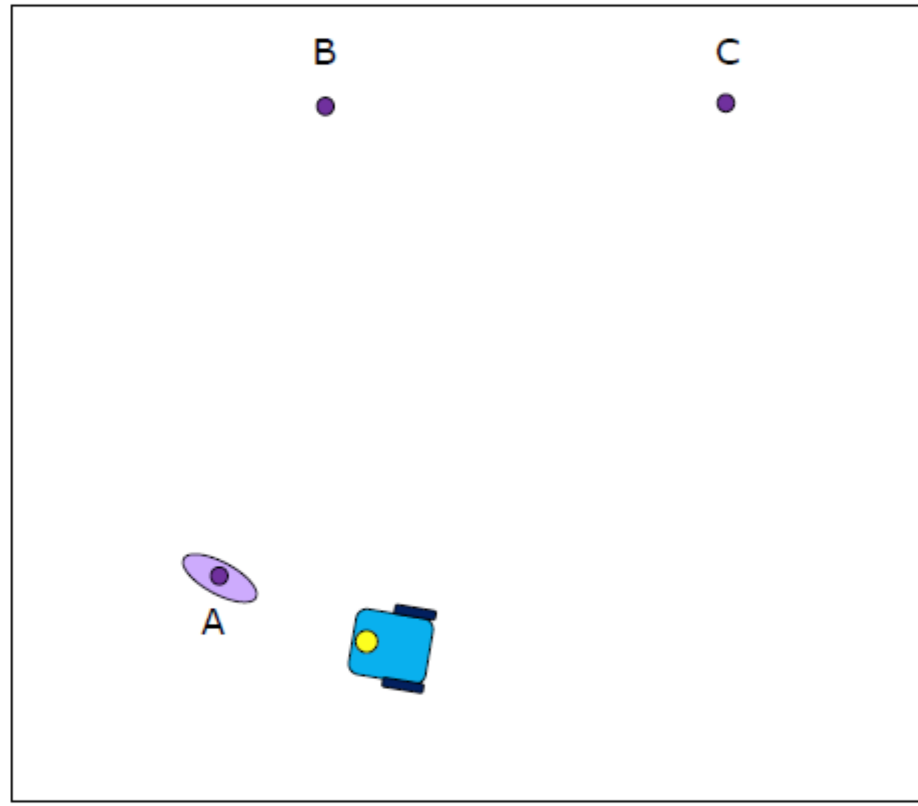
First measurement of feature A

SLAM Example

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

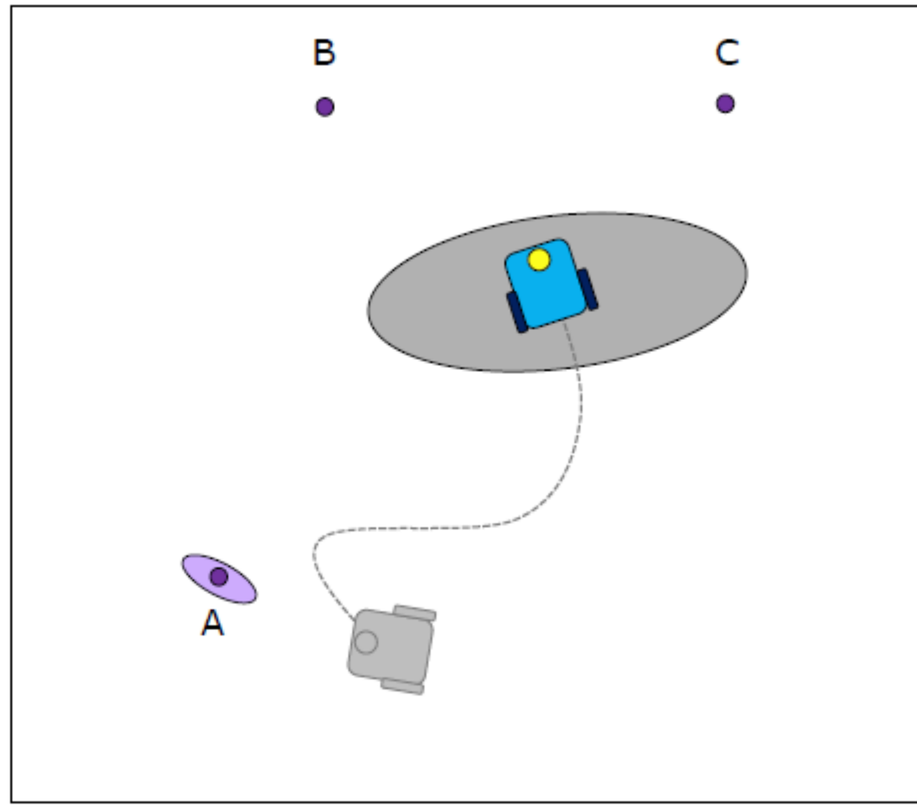
On every frame:

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



SLAM Example

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

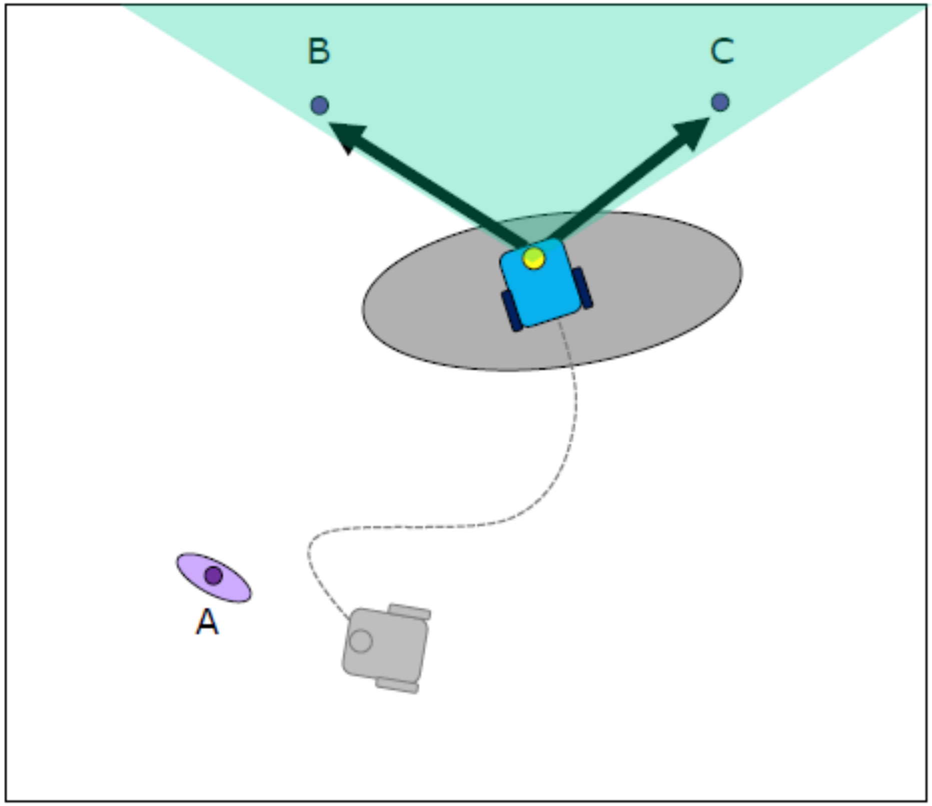


Robot moves forwards: uncertainty grows

On every frame:
- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations

SLAM Example

- Robot observes two new features.



Robot makes first measurements of B & C

On every frame:
- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations

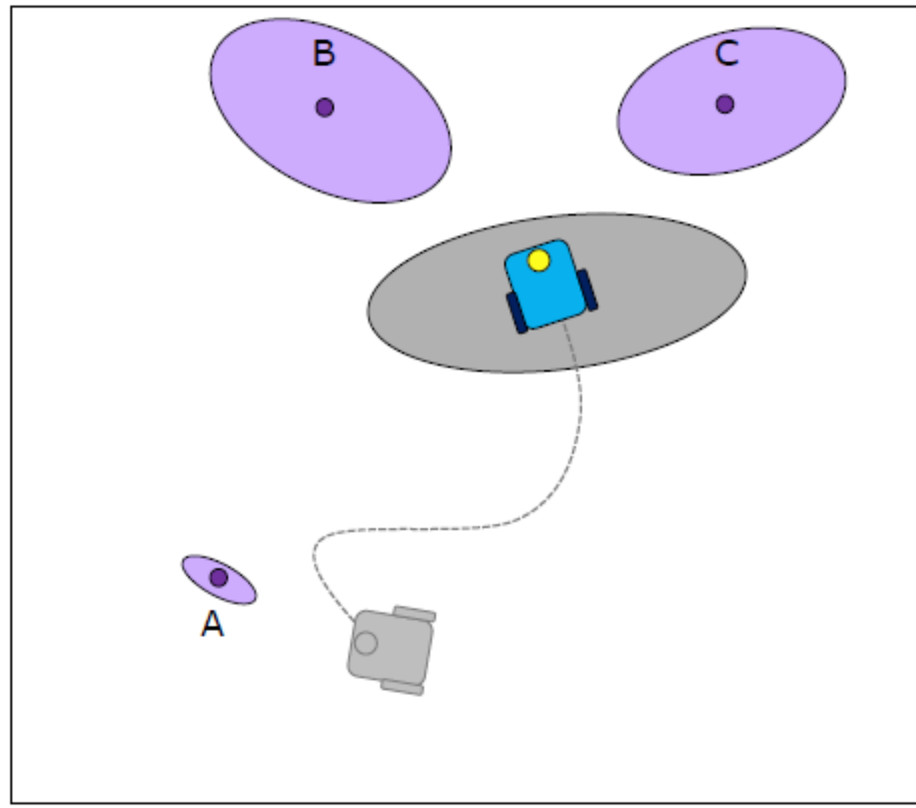
SLAM Example

- 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:

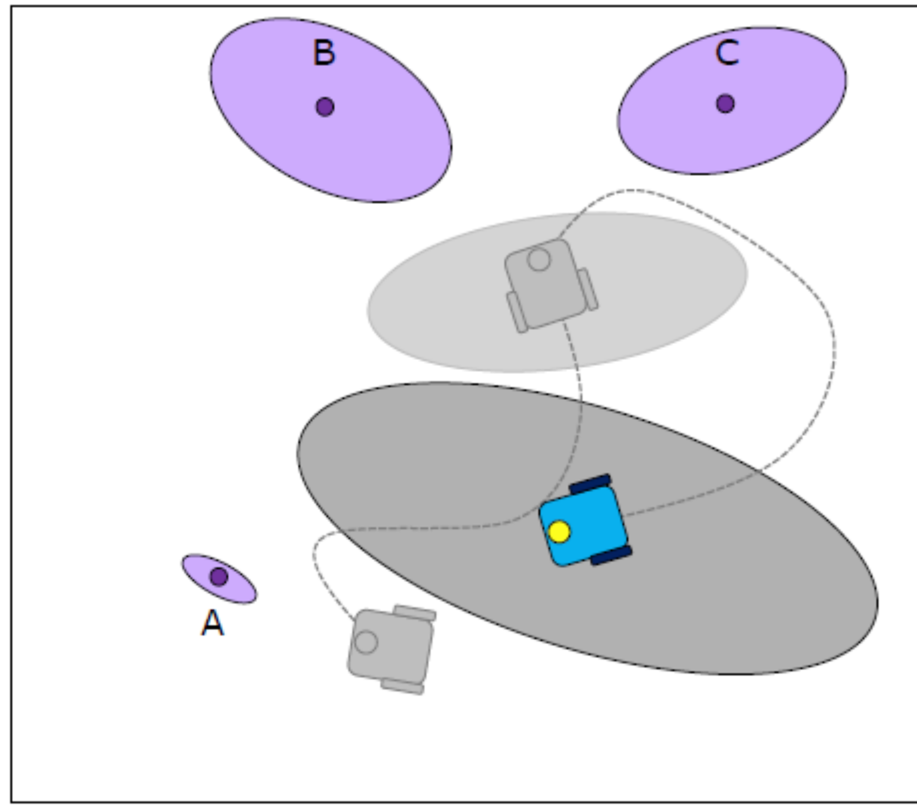
- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations



Robot makes first measurements of B & C

SLAM Example

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



Robot moves again: uncertainty grows more

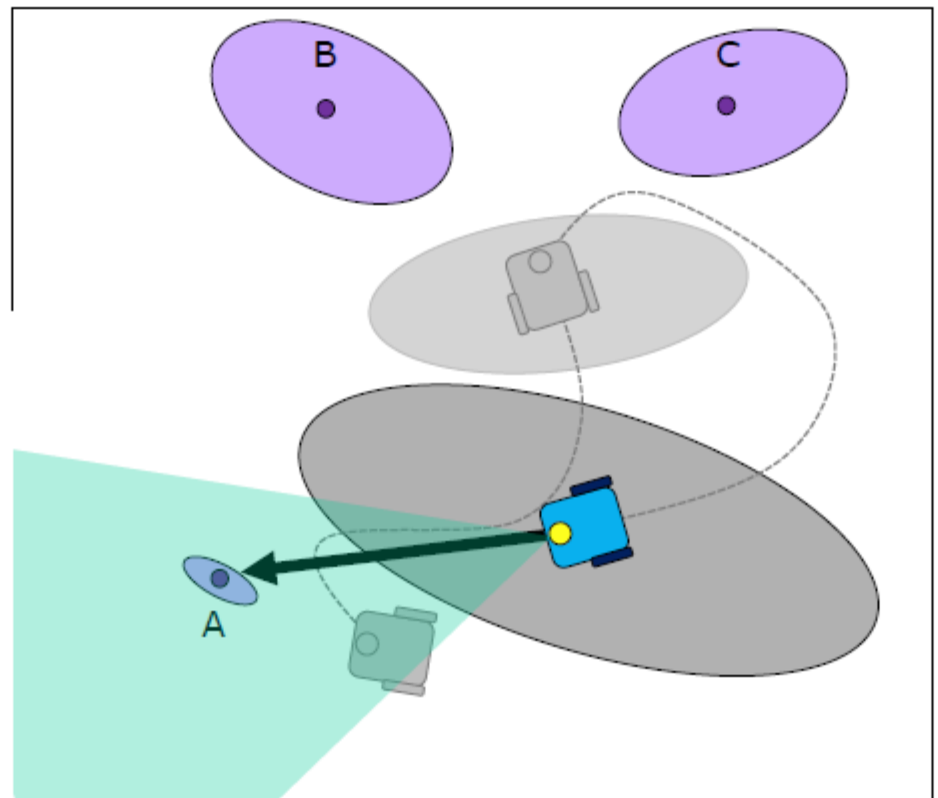
On every frame:

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

SLAM Example

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

On every frame:
- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations



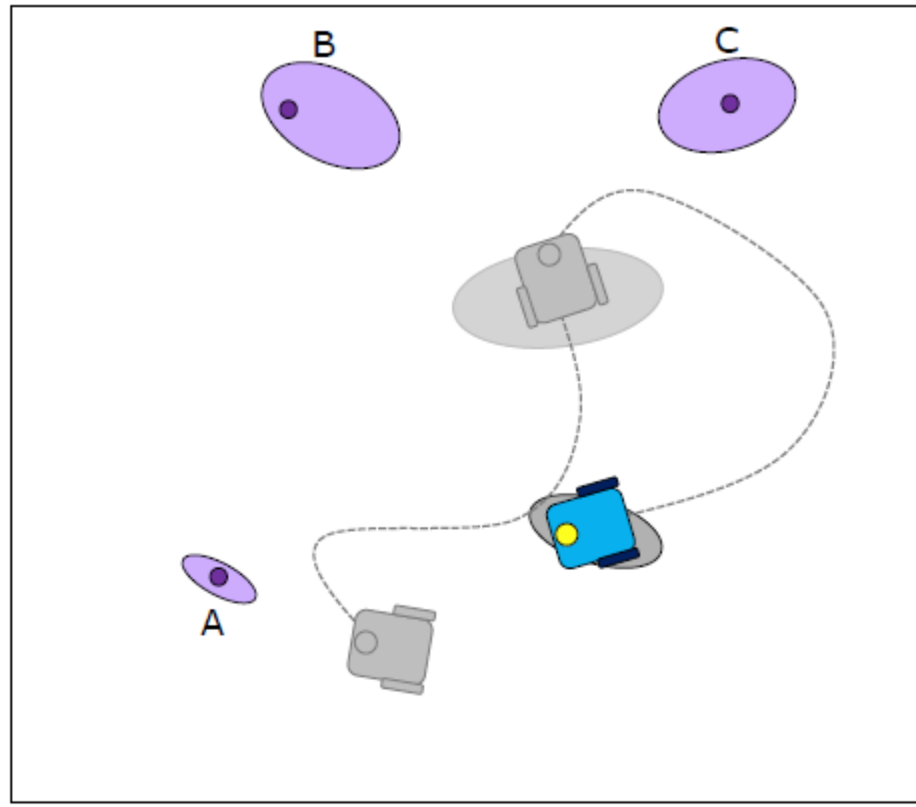
Robot re-measures A: "loop closure"

SLAM Example

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

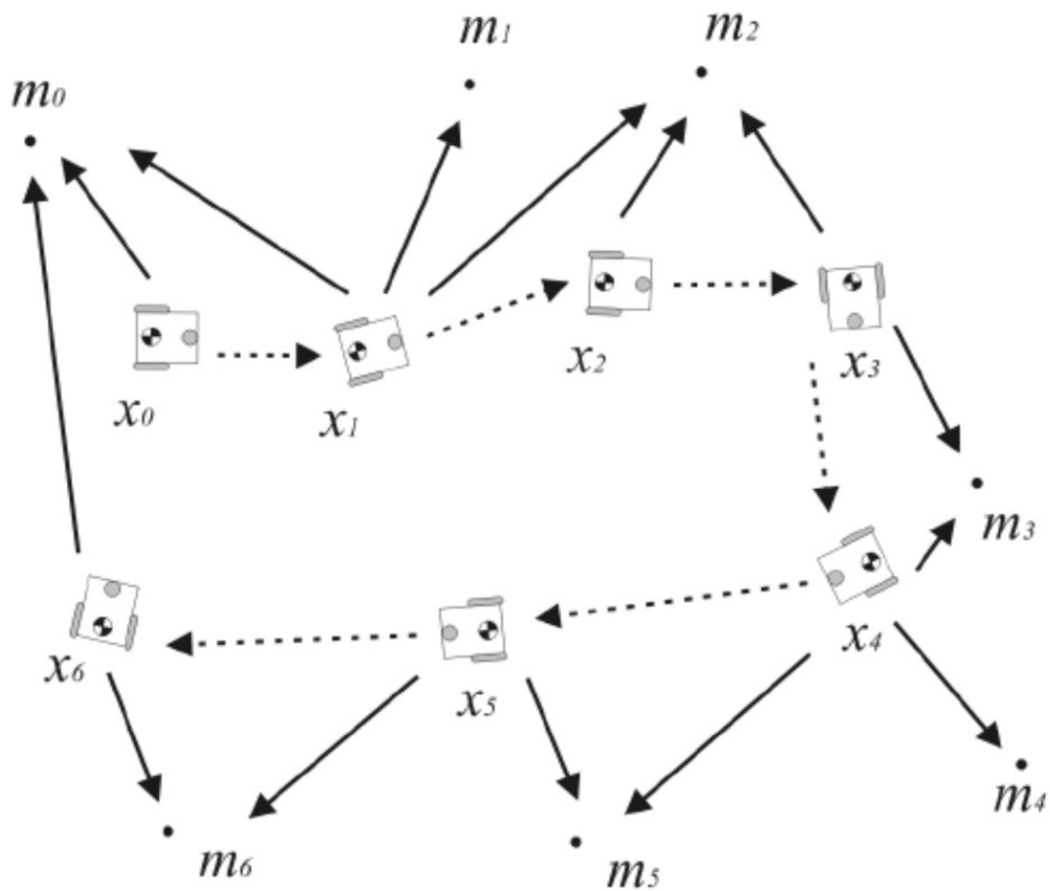
On every frame:

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



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

SLAM



SLAM In Action

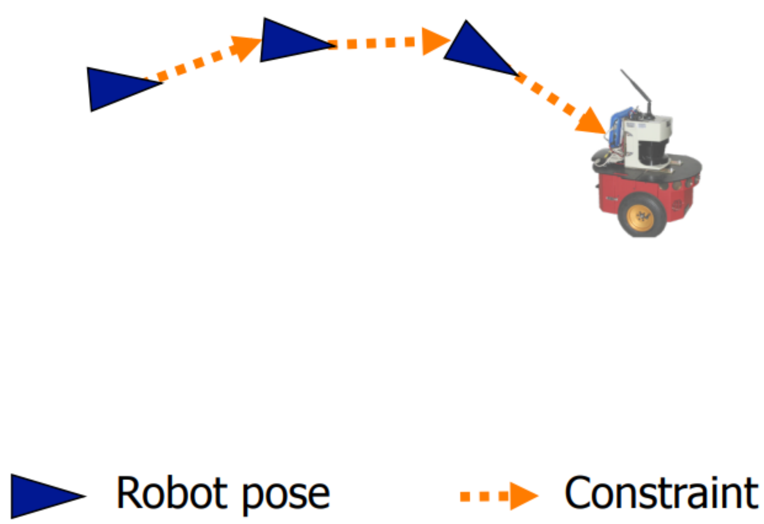
http://www.cs.washington.edu/ai/Mobile_Robotics/projects/mapping/

SLAM Formalisms

- Some of the most influential approaches to SLAM:
  - GraphSLAM (Factor Graphs) --- We'll see this in more detail next time
  - Extended Kalman Filter SLAM (EKF SLAM)
  - Particle Filter SLAM (FAST SLAM)
  - … many variants

Graph SLAM

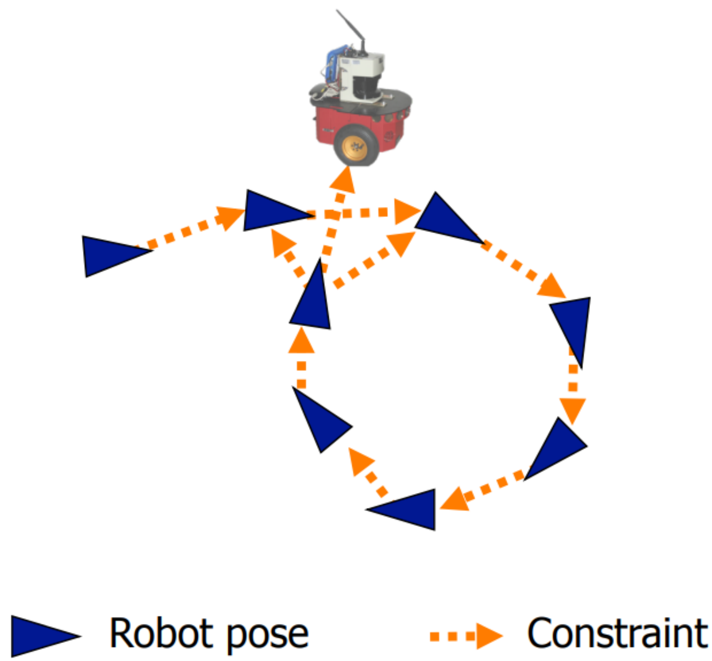
- Constraints connect the poses of the robot while it is moving
- Constraints are inherently uncertain



Source: Wolfram Burgard, Cyrill Stachniss, Maren Bennewitz, Diego Tipaldi, Luciano Spinello

Graph SLAM

- Observing previously seen areas generates constraints between non-successive poses
- Constraints are inherently uncertain



Source: Wolfram Burgard, Cyrill Stachniss, Maren Bennewitz, Diego Tipaldi, Luciano Spinello

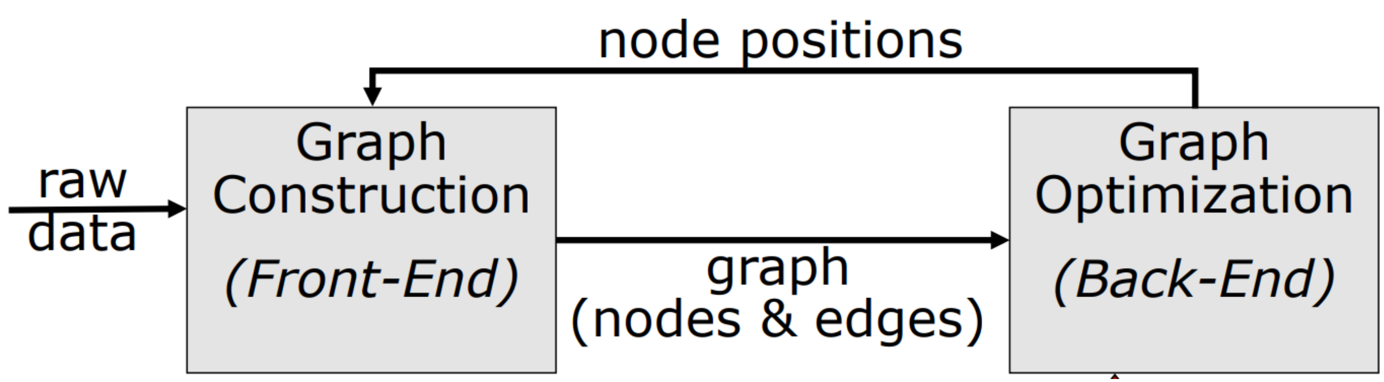
Basic idea behind Graph SLAM

- 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
- **The method**: Build the graph and find a node configuration that minimizes the error introduced by the constraints

Source: Wolfram Burgard, Cyrill Stachniss, Maren Bennewitz, Diego Tipaldi, Luciano Spinello

Graph SLAM: a system view

- Interplay of front-end and back-end
- A consistent map helps to determine new constraints by reducing the search space
- Optimize the graph by minimizing squared errors (i.e., least squares)

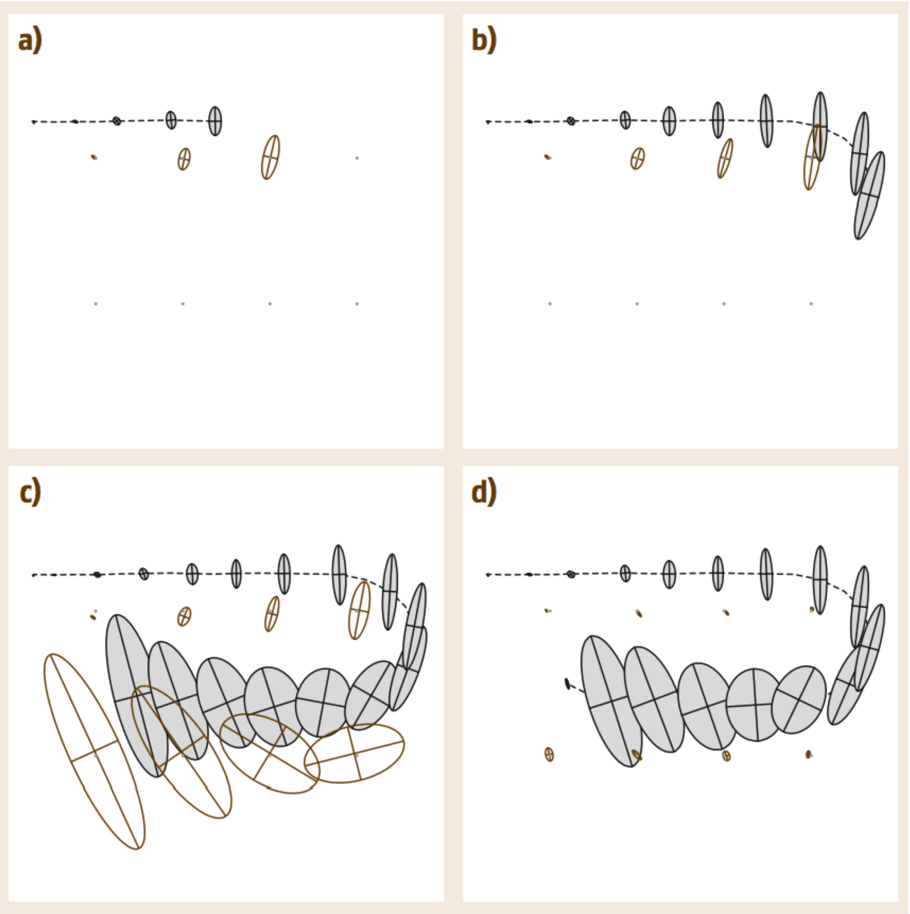


Source: Wolfram Burgard, Cyrill Stachniss, Maren Bennewitz, Diego Tipaldi, Luciano Spinello

EKF SLAM

- Extended Kalman Filter
  - Non-linear version of the Kalman Filter
- EKF SLAM
  - The “map” is a large vector that combines the robot’s position and landmark states, modeled by a Gaussian
  - Motion and sensing update the vector, adding new dimensions when new landmarks are found

EKF-SLAM



EKF SLAM state representation

State representation for n landmarks

$$x_t = (\underbrace{x, y, \theta}_{\text{robot's pose}}, \underbrace{m_{1,x}, m_{1,y}}_{\text{landmark 1}}, \ldots, \underbrace{m_{n,x}, m_{n,y}}_{\text{landmark n}})^T$$

EKF SLAM state representation

- Map with *n* landmarks results in a (3+2*n*)-dimensional Gaussian
- Belief is represented by

$$
\underbrace{\begin{pmatrix} x \\ y \\ \theta \\ m_{1,x} \\ m_{1,y} \\ \vdots \\ m_{n,x} \\ m_{n,y} \end{pmatrix}}_{\mu}
\quad
\underbrace{\begin{pmatrix}
\sigma_{xx} & \sigma_{xy} & \sigma_{x\theta} & \sigma_{xm_{1,x}} & \sigma_{xm_{1,y}} & \cdots & \sigma_{xm_{n,x}} & \sigma_{xm_{n,y}} \\
\sigma_{yx} & \sigma_{yy} & \sigma_{y\theta} & \sigma_{ym_{1,x}} & \sigma_{ym_{1,y}} & \cdots & \sigma_{m_{n,x}} & \sigma_{m_{n,y}} \\
\sigma_{\theta x} & \sigma_{\theta y} & \sigma_{\theta\theta} & \sigma_{\theta m_{1,x}} & \sigma_{\theta m_{1,y}} & \cdots & \sigma_{\theta m_{n,x}} & \sigma_{\theta m_{n,y}} \\
\sigma_{m_{1,x}x} & \sigma_{m_{1,x}y} & \sigma_{\theta} & \sigma_{m_{1,x}m_{1,x}} & \sigma_{m_{1,x}m_{1,y}} & \cdots & \sigma_{m_{1,x}m_{n,x}} & \sigma_{m_{1,x}m_{n,y}} \\
\sigma_{m_{1,y}x} & \sigma_{m_{1,y}y} & \sigma_{\theta} & \sigma_{m_{1,y}m_{1,x}} & \sigma_{m_{1,y}m_{1,y}} & \cdots & \sigma_{m_{1,y}m_{n,x}} & \sigma_{m_{1,y}m_{n,y}} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
\sigma_{m_{n,x}x} & \sigma_{m_{n,x}y} & \sigma_{\theta} & \sigma_{m_{n,x}m_{1,x}} & \sigma_{m_{n,x}m_{1,y}} & \cdots & \sigma_{m_{n,x}m_{n,x}} & \sigma_{m_{n,x}m_{n,y}} \\
\sigma_{m_{n,y}x} & \sigma_{m_{n,y}y} & \sigma_{\theta} & \sigma_{m_{n,y}m_{1,x}} & \sigma_{m_{n,y}m_{1,y}} & \cdots & \sigma_{m_{n,y}m_{n,x}} & \sigma_{m_{n,y}m_{n,y}}
\end{pmatrix}}_{\Sigma}
$$

EKF SLAM state representation

- More compactly

$$\underbrace{\begin{pmatrix} x_R \\ m_1 \\ \vdots \\ m_n \end{pmatrix}}_{\mu} \quad \underbrace{\begin{pmatrix} \Sigma_{x_R x_R} & \Sigma_{x_R m_1} & \cdots & \Sigma_{x_R m_n} \\ \Sigma_{m_1 x_R} & \Sigma_{m_1 m_1} & \cdots & \Sigma_{m_1 m_n} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_{m_n x_R} & \Sigma_{m_n m_1} & \cdots & \Sigma_{m_n m_n} \end{pmatrix}}_{\Sigma}$$

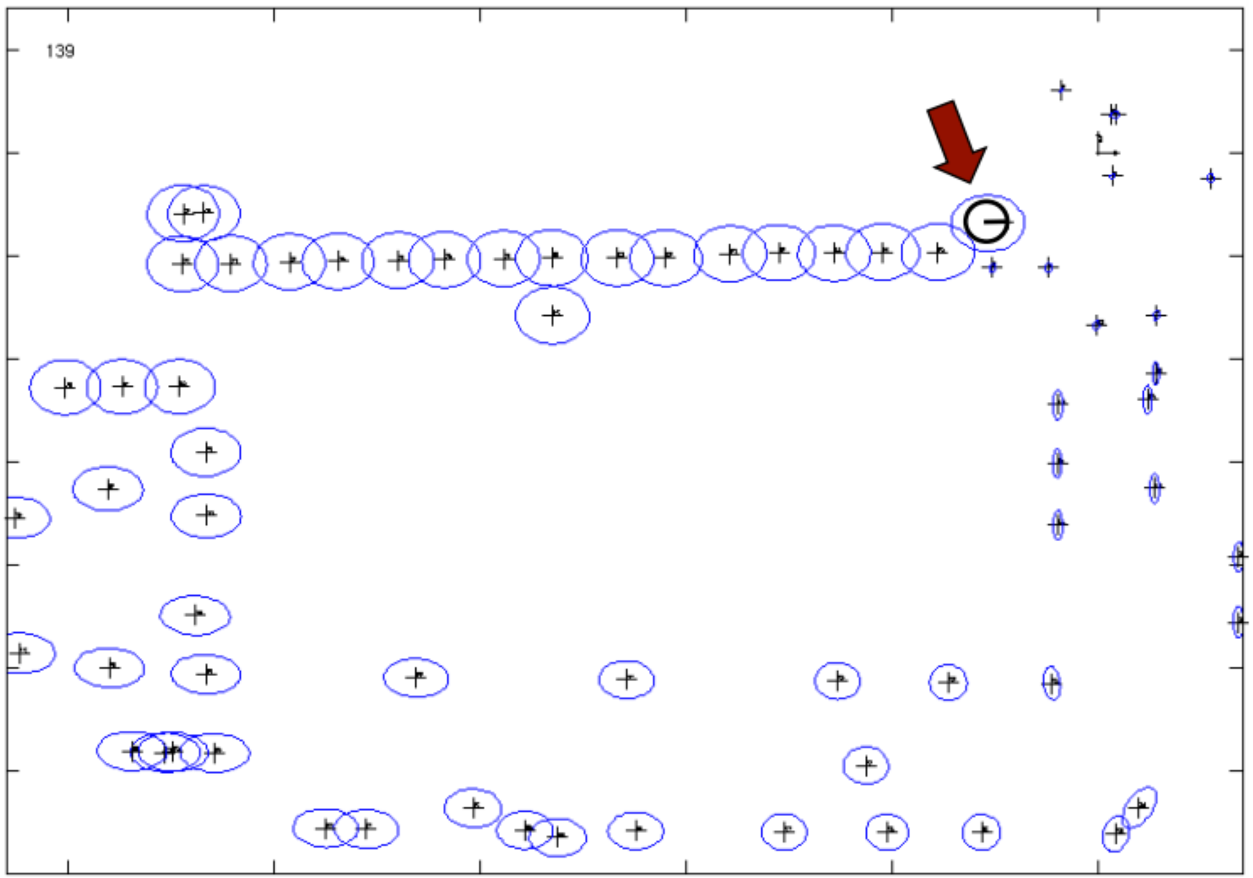
Loop closure

- **Recognizing an already mapped area**

- Data association with
  - high ambiguity
  - possible environment symmetries

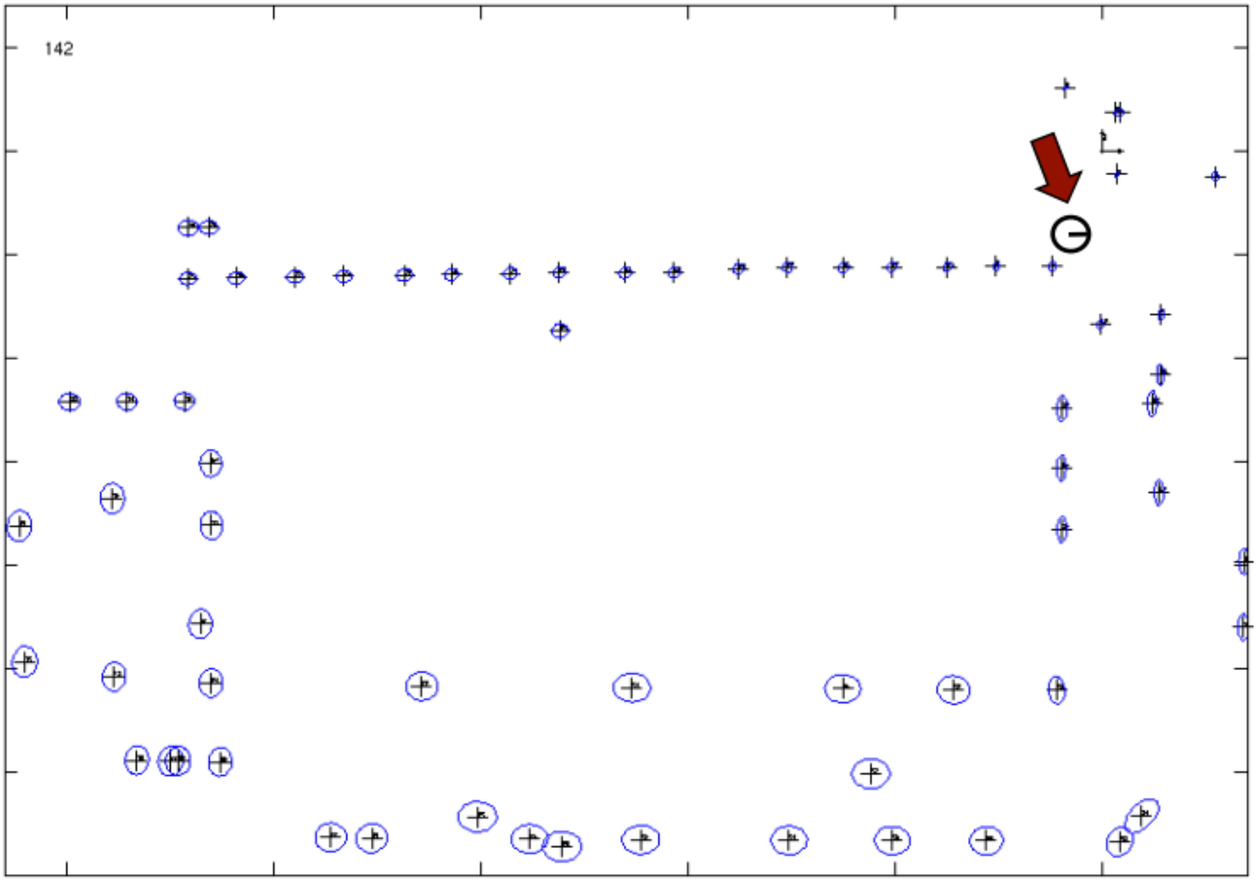
- Uncertainties collapse after a loop closure (whether the closure was correct or not!)

Example: before loop closure



Courtesy of K. Arras

Example: after loop closure



Courtesy of K. Arras

Loop Closure

- Loop closing reduces the uncertainty in robot and landmark estimates
- This can be exploited when exploring an environment for the sake of better (e.g. more accurate) maps
- Wrong loop closures lead to filter divergence!

SLAM simulation

SLAM simulation by Sjoerd de Jong under supervision of Gert Kootstra. The Kalman functionality is partly based on the EKF SLAM simulation of Tim Bailey. *www-personal.acfr.usyd.edu.au/tbailey*

Department of Artificial Intelligence
University of Groningen

FastSLAM

- Use particle filter to estimate the state of the robot
- Use Extended Kalman Filter (EKF) to estimate the state of the landmarks
- Each particle keeps track of its own estimate of the map (i.e., the landmark positions)
- Each particle does its own data association (i.e., matching landmarks between image frames)
- If a particle is lost during resampling, its entire map estimate is also lost.

Algorithm FastSLAM 1.0(z_t, u_t, S_{t-1}):

for $m = 1$ to M do // loop over all particles

retrieve $\left\langle s_{t-1}^{[m]}, N_{t-1}^{[m]}, \left\langle \mu_{1,t-1}^{[m]}, \Sigma_{1,t-1}^{[m]}, i_1^{[m]} \right\rangle, \ldots, \left\langle \mu_{N_{t-1}^{[m]},t-1}^{[m]}, \Sigma_{N_{t-1}^{[m]},t-1}^{[m]}, i_{N_{t-1}^{[m]}}^{[m]} \right\rangle \right\rangle$ from S_{t-1}

$s_t^{[m]} \sim p(s_t \mid s_{t-1}^{[m]}, u_t)$ // sample new pose

for $n = 1$ to $N_{t-1}^{[m]}$ do // calculate measurement likelihoods

$\hat{z}_n = g(\mu_{n,t-1}^{[m]}, s_t^{[m]})$ // measurement prediction

$G_n = g'(s_t^{[m]}, \mu_{n,t-1}^{[m]})$ // calculate Jacobian

$Q_n = G_n^T \Sigma_{n,t-1}^{[m]} G_n + R_t$ // measurement covariance

$w_n = |2\pi Q_n|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(z_t - \hat{z}_n)^T Q_n^{-1} (z_t - \hat{z}_n)\right\}$ // likelihood of correspondence

endfor

$w_{N_{t-1}^{[m]}+1} = p_0$ // importance factor of new landmark

$\hat{n} = \underset{n=1,\ldots,N_{t-1}^{[m]}+1}{\operatorname{argmax}} w_n$ // max likelihood correspondence

$N_t^{[m]} = \max\{N_{t-1}^{[m]}, \hat{n}\}$ // new number of features in map

for $n = 0$ to $N_t^{[m]}$ do // update Kalman fi lters

if $n = N_{t-1}^{[m]} + 1$ then // is new feature?

$\mu_{n,t}^{[m]} = g^{-1}(z_t, s_t^{[m]})$ // initialize mean

$\Sigma_{n,t}^{[m]} = G_{\hat{n}}^{-1} R_t (G_{\hat{n}}^{-1})^T$ // initialize covariance

$i_{n,t}^{[m]} = 1$ // initialize counter

else if $n = \hat{n}$ then // is observed feature?

$K = \Sigma_{n,t-1}^{[m]} G_{\hat{n}} Q_{\hat{n}}^{-1}$ // calculate Kalman gain

$\mu_{n,t}^{[m]} = \mu_{n,t-1}^{[m]} + K(z_t - \hat{z}_{\hat{n}})^T$ // update mean

$\Sigma_{n,t}^{[m]} = (I - K G_{\hat{n}}^T)\Sigma_{n,t-1}^{[m]}$ // update covariance

$i_{n,t}^{[m]} = i_{n,t-1}^{[m]} + 1$ // increment counter

else // all other features

$\mu_{n,t}^{[m]} = \mu_{n,t-1}^{[m]}$ // copy old mean

$\Sigma_{n,t}^{[m]} = \Sigma_{n,t-1}^{[m]}$ // copy old covariance

if $\mu_{n,t-1}^{[m]}$ outside perceptual range of $s_t^{[m]}$ then // should feature have been observed?

$i_{n,t}^{[m]} = i_{n,t-1}^{[m]}$ // no, do not change

else

$i_{n,t}^{[m]} = i_{n,t-1}^{[m]} - 1$ // yes, decrement counter

if $i_{n,t-1}^{[m]} < 0$ then discard feature n endif // discard dubious features

endif

endif

endfor

add $\left\langle s_t^{[m]}, N_t^{[m]}, \left\langle \mu_{1,t}^{[m]}, \Sigma_{1,t}^{[m]}, i_1^{[m]} \right\rangle, \ldots, \left\langle \mu_{N_t^{[m]},t}^{[m]}, \Sigma_{N_t^{[m]},t}^{[m]}, i_{N_t^{[m]}}^{[m]} \right\rangle \right\rangle$ to S_{aux}

endfor

$S_t = \emptyset$ // construct new particle set

for $m' = 1$ to M do // resample M particles

draw random index m with probability $\propto w_t^{[m]}$ // resample

add $\left\langle s_t^{[m]}, N_t^{[m]}, \left\langle \mu_{1,t}^{[m]}, \Sigma_{1,t}^{[m]}, i_1^{[m]} \right\rangle, \ldots, \left\langle \mu_{N_t^{[m]},t}^{[m]}, \Sigma_{N_t^{[m]},t}^{[m]}, i_{N_t^{[m]}}^{[m]} \right\rangle \right\rangle$ to S_t

end for

return S_t

end algorithm

FastSLAM: An Efficient Solution to the Simultaneous Localization And Mapping Problem with Unknown Data Association

Sebastian Thrun, Michael Montemerlo, Daphne Koller, Ben Wegbreit, Juan Nieto, and Eduardo Nebot

May 2004, Journal of Machine Learning Research 4(3)

Algorithm FastSLAM 1.0(z_t, u_t, S_{t-1}):

for $m = 1$ to M do // loop over all particles

retrieve $\left\langle s_{t-1}^{[m]}, N_{t-1}^{[m]}, \left\langle \mu_{1,t-1}^{[m]}, \Sigma_{1,t-1}^{[m]}, i_1^{[m]} \right\rangle, \ldots, \left\langle \mu_{N_{t-1}^{[m]},t-1}^{[m]}, \Sigma_{N_{t-1}^{[m]},t-1}^{[m]}, i_{N_{t-1}^{[m]}}^{[m]} \right\rangle \right\rangle$ from S_{t-1}

$s_t^{[m]} \sim p(s_t \mid s_{t-1}^{[m]}, u_t)$ // sample new pose

for $n = 1$ to $N_{t-1}^{[m]}$ do // calculate measurement likelihoods

$\hat{z}_n = g(\mu_{n,t-1}^{[m]}, s_t^{[m]})$ // measurement prediction

$G_n = g'(s_t^{[m]}, \mu_{n,t-1}^{[m]})$ // calculate Jacobian

$Q_n = G_n^T \Sigma_{n,t-1}^{[m]} G_n + R_t$ // measurement covariance

$w_n = |2\pi Q_n|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(z_t - \hat{z}_n)^T Q_n^{-1}(z_t - \hat{z}_n)\right\}$ // likelihood of correspondence

endfor

$w_{N_{t-1}^{[m]}+1} = p_0$ // importance factor of new landmark

$\hat{n} = \underset{n=1,\ldots,N_{t-1}^{[m]}+1}{\operatorname{argmax}} \; w_n$ // max likelihood correspondence

$N_t^{[m]} = \max\{N_{t-1}^{[m]}, \hat{n}\}$ // new number of features in map

for $n = 0$ to $N_t^{[m]}$ do // update Kalman fi lters

if $n = N_{t-1}^{[m]} + 1$ then // is new feature?

$\mu_{n,t}^{[m]} = g^{-1}(z_t, s_t^{[m]})$

$\Sigma_{n,t}^{[m]} = G_{\hat{n}}^{-1} R_t (G_{\hat{n}}^{-1})^T$

$i_{n,t}^{[m]} = 1$

else if $n = \hat{n}$ then

$K = \Sigma_{n,t-1}^{[m]} G_{\hat{n}} Q_{\hat{n}}^{-1}$

$\mu_{n,t}^{[m]} = \mu_{n,t-1}^{[m]} + K(z_t -$

$\Sigma_{n,t}^{[m]} = (I - K G_{\hat{n}}^T)\Sigma_{n,t}^{[m]}$

$i_{n,t}^{[m]} = i_{n,t-1}^{[m]} + 1$

else

$\mu_{n,t}^{[m]} = \mu_{n,t-1}^{[m]}$

$\Sigma_{n,t}^{[m]} = \Sigma_{n,t-1}^{[m]}$

if $\mu_{n,t-1}^{[m]}$ outside percept

$i_{n,t}^{[m]} = i_{n,t-1}^{[m]}$

else

$i_{n,t}^{[m]} = i_{n,t-1}^{[m]} - 1$ // yes, decrement counter

if $i_{n,t-1}^{[m]} < 0$ then discard feature n endif // discard dubious features

endif

endif

endfor

add $\left\langle s_t^{[m]}, N_t^{[m]}, \left\langle \mu_{1,t}^{[m]}, \Sigma_{1,t}^{[m]}, i_1^{[m]} \right\rangle, \ldots, \left\langle \mu_{N_t^{[m]},t}^{[m]}, \Sigma_{N_t^{[m]},t}^{[m]}, i_{N_t^{[m]}}^{[m]} \right\rangle \right\rangle$ to S_{aux}

endfor

$S_t = \emptyset$ // construct new particle set

for $m' = 1$ to M do // resample M particles

draw random index m with probability $\propto w_t^{[m]}$ // resample

add $\left\langle s_t^{[m]}, N_t^{[m]}, \left\langle \mu_{1,t}^{[m]}, \Sigma_{1,t}^{[m]}, i_1^{[m]} \right\rangle, \ldots, \left\langle \mu_{N_t^{[m]},t}^{[m]}, \Sigma_{N_t^{[m]},t}^{[m]}, i_{N_t^{[m]}}^{[m]} \right\rangle \right\rangle$ to S_t

end for

return S_t

end algorithm

1. Execute for every particle
2. Retrieve previous info for this particle
3. Particle filter:
   1. Sample pose
   2. Update measurement likelihoods
   3. Compute particle weights
4. Resample

Algorithm FastSLAM 1.0(z_t, u_t, S_{t-1}):

for $m = 1$ to M do // loop over all particles

retrieve $\left\langle s_{t-1}^{[m]}, N_{t-1}^{[m]}, \left\langle \mu_{1,t-1}^{[m]}, \Sigma_{1,t-1}^{[m]}, i_1^{[m]} \right\rangle, \ldots, \left\langle \mu_{N_{t-1}^{[m]},t-1}^{[m]}, \Sigma_{N_{t-1}^{[m]},t-1}^{[m]}, i_{N_{t-1}^{[m]}}^{[m]} \right\rangle \right\rangle$ from S_{t-1}

$s_t^{[m]} \sim p(s_t \mid s_{t-1}^{[m]}, u_t)$ // sample new pose

for $n = 1$ to $N_{t-1}^{[m]}$ do // calculate measurement likelihoods

$\hat{z}_n = g(\mu_{n,t-1}^{[m]}, s_t^{[m]})$ // measurement prediction

$G_n = g'(s_t^{[m]}, \mu_{n,t-1}^{[m]})$ // calculate Jacobian

$Q_n = G_n^T \Sigma_{n,t-1}^{[m]} G_n + R_t$ // measurement covariance

$w_n = |2\pi Q_n|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(z_t - \hat{z}_n)^T Q_n^{-1}(z_t - \hat{z}_n)\right\}$ // likelihood of correspondence

endfor

$w_{N_{t-1}^{[m]}+1} = p_0$ // importance factor of new landmark

$\hat{n} = \underset{n=1,\ldots,N_{t-1}^{[m]}+1}{\operatorname{argmax}} \; w_n$ // max likelihood correspondence

$N_t^{[m]} = \max\{N_{t-1}^{[m]}, \hat{n}\}$ // new number of features in map

for $n = 0$ to $N_t^{[m]}$ do // update Kalman fi lters

if $n = N_{t-1}^{[m]} + 1$ then // is new feature?

$\mu_{n,t}^{[m]} = g^{-1}(z_t, s_t^{[m]})$ // initialize mean

$\Sigma_{n,t}^{[m]} = G_{\hat{n}}^{-1} R_t (G_{\hat{n}}^{-1})^T$ // initialize covariance

$i_{n,t}^{[m]} = 1$ // initialize counter

else if $n = \hat{n}$ then // is observed feature?

$K = \Sigma_{n,t-1}^{[m]} G_{\hat{n}} Q_{\hat{n}}^{-1}$ // calculate Kalman gain

$\mu_{n,t}^{[m]} = \mu_{n,t-1}^{[m]} + K(z_t - \hat{z}_{\hat{n}})^T$ // update mean

$\Sigma_{n,t}^{[m]} = (I - K G_{\hat{n}}^T)\Sigma_{n,t-1}^{[m]}$ // update covariance

$i_{n,t}^{[m]} = i_{n,t-1}^{[m]} + 1$ // increment counter

else // all other features

$\mu_{n,t}^{[m]} = \mu_{n,t-1}^{[m]}$ // copy old mean

$\Sigma_{n,t}^{[m]} = \Sigma_{n,t-1}^{[m]}$ // copy old covariance

if $\mu_{n,t-1}^{[m]}$ outside perceptual range of $s_t^{[m]}$ then // should feature have been observed?

$i_{n,t}^{[m]} = i_{n,t-1}^{[m]}$ // no, do not change

else

$i_{n,t}^{[m]} = i_{n,t-1}^{[m]} - 1$ // yes, decrement counter

if $i_{n,t-1}^{[m]} < 0$ then discard feature n endif // discard dubious features

endif

endif

endfor

add $\left\langle s_t^{[m]}, N_t^{[m]}, \left\langle \mu_{1,t}^{[m]}, \Sigma_{1,t}^{[m]}, i_1^{[m]} \right\rangle, \ldots, \left\langle \mu_{N_t^{[m]},t}^{[m]}, \Sigma_{N_t^{[m]},t}^{[m]}, i_{N_t^{[m]}}^{[m]} \right\rangle \right\rangle$ to S_{aux}

endfor

$S_t = \emptyset$ // construct new particle set

for $m' = 1$ to M do // resample M particles

draw random index m with probability $\propto w_t^{[m]}$ // resample

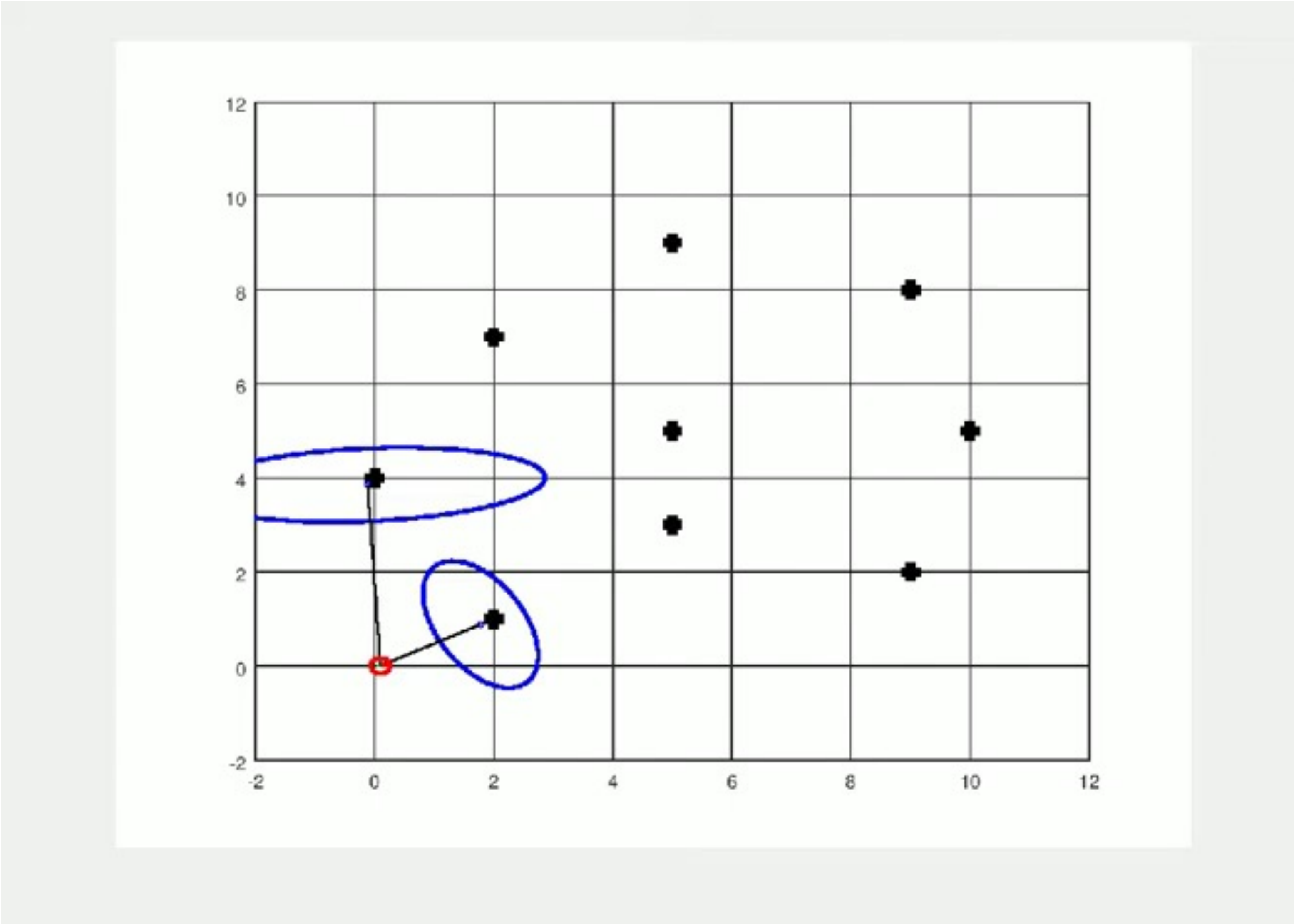
add $\left\langle s_t^{[m]}, N_t^{[m]}, \left\langle \mu_{1,t}^{[m]}, \Sigma_{1,t}^{[m]}, i_1^{[m]} \right\rangle, \ldots, \left\langle \mu_{N_t^{[m]},t}^{[m]}, \Sigma_{N_t^{[m]},t}^{[m]}, i_{N_t^{[m]}}^{[m]} \right\rangle \right\rangle$ to S_t

end for

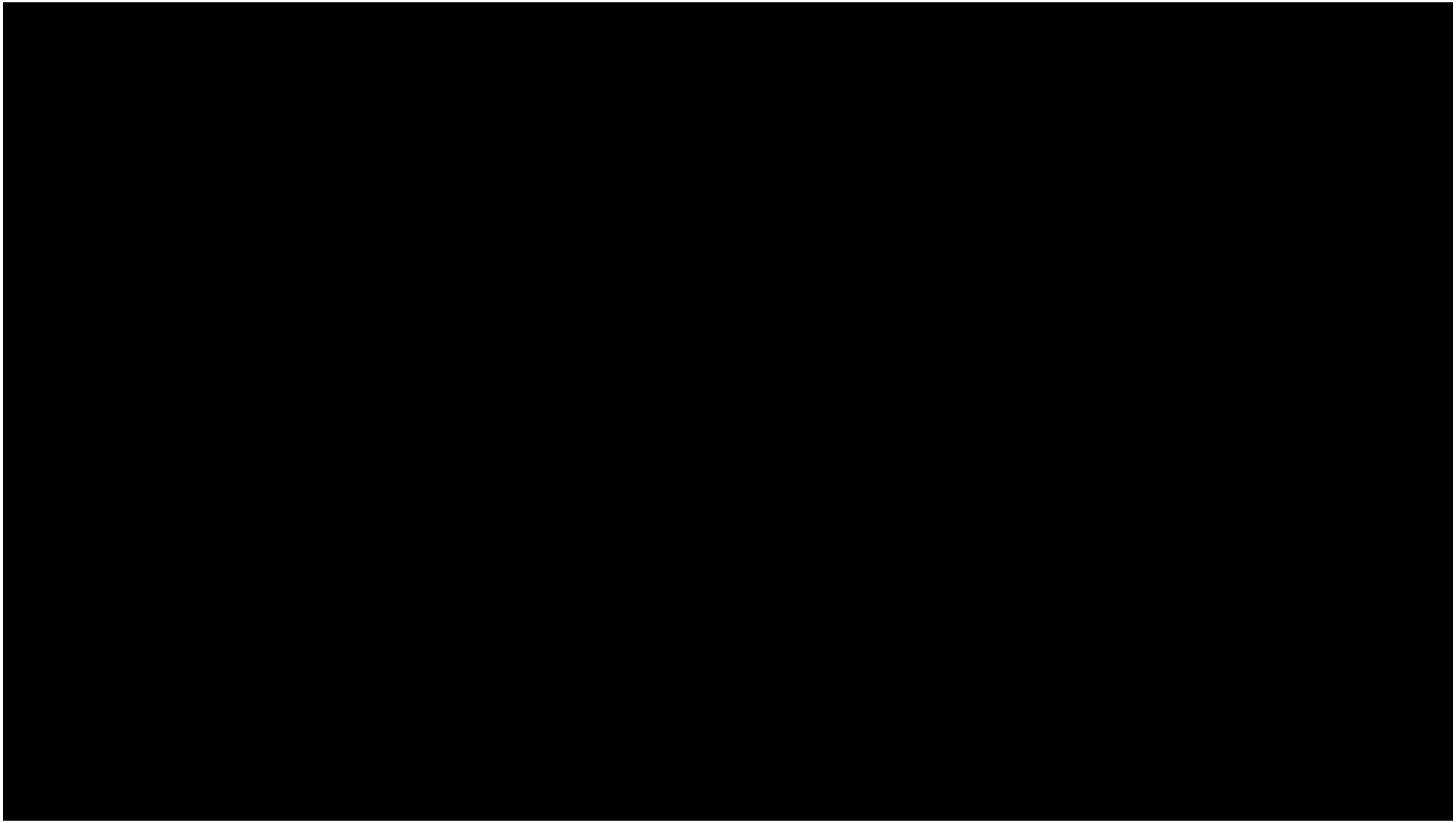
return S_t

end algorithm

Update the estimates for each particle using the EKF



https://www.youtube.com/watch?v=viMDOaV9bK8

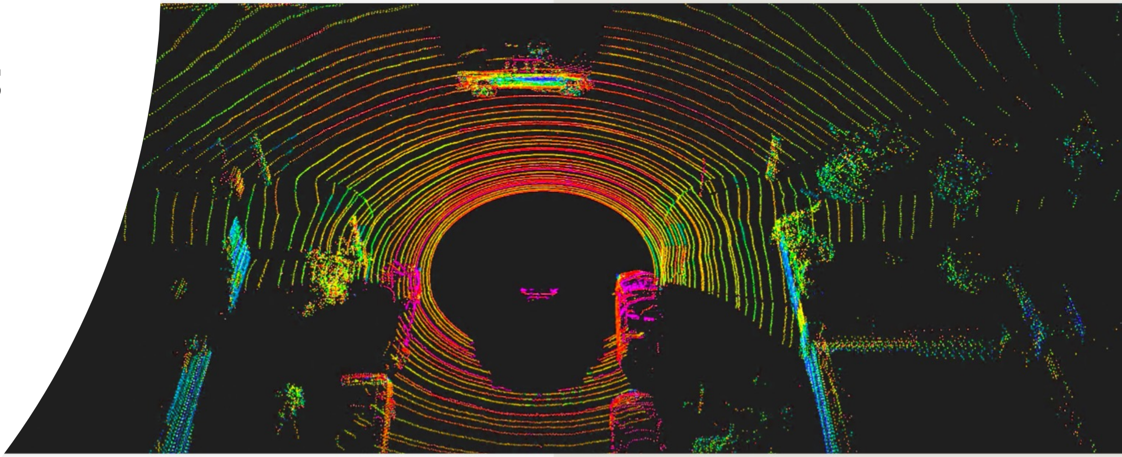


https://www.youtube.com/watch?v=kuxI8two5GA

[Smart Mobility Research Team@AIST]()

LIDAR

- Superpowers:
  - 360 Visibility
  - Accurate depth!
- Almost all AV prototypes have them (not all 360)



[Images and exposition take from excellent Voyage Blog post](https://news.voyage.auto/an-introduction-to-lidar-the-key-self-driving-car-sensor-a7e405590cf)

https://news.voyage.auto/an-introduction-to-lidar-the-key-self-driving-car-sensor-a7e405590cf

Localization with LIDAR

- ICP = Iterated Closest Points:
- Call current scan S, map M
- Predict pose from motion model: use other sensors if available
- Iterate:
  - For every point s: find closest m
  - Re-estimate pose
- In practice:
  - outlier rejection to account for moving objects, unmodeled structures, parked cars etc…

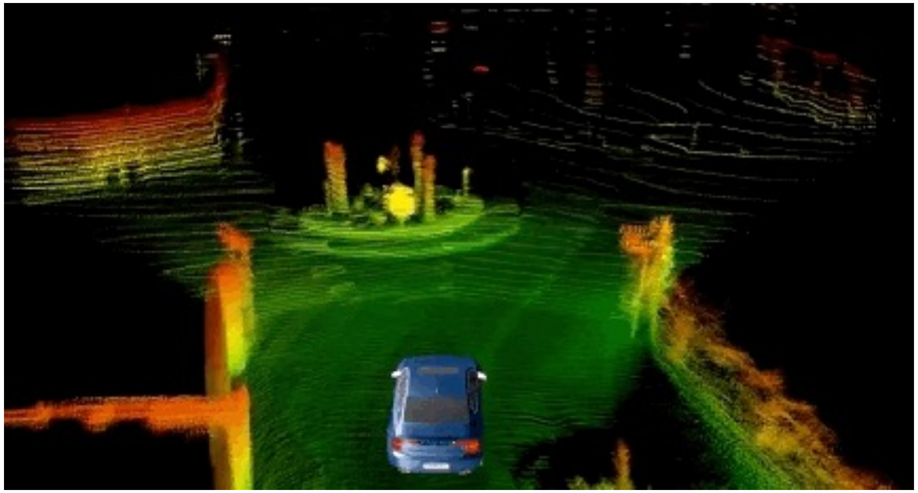
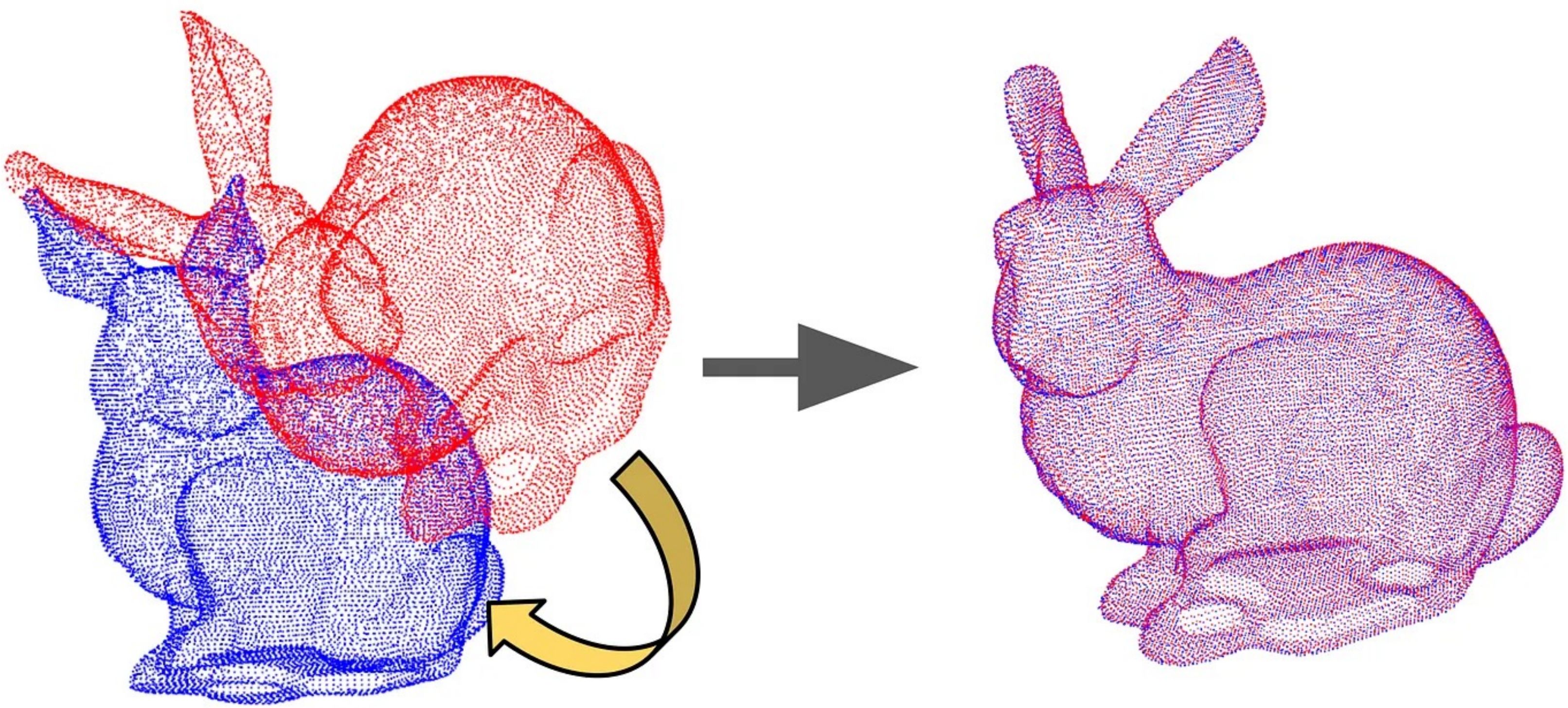


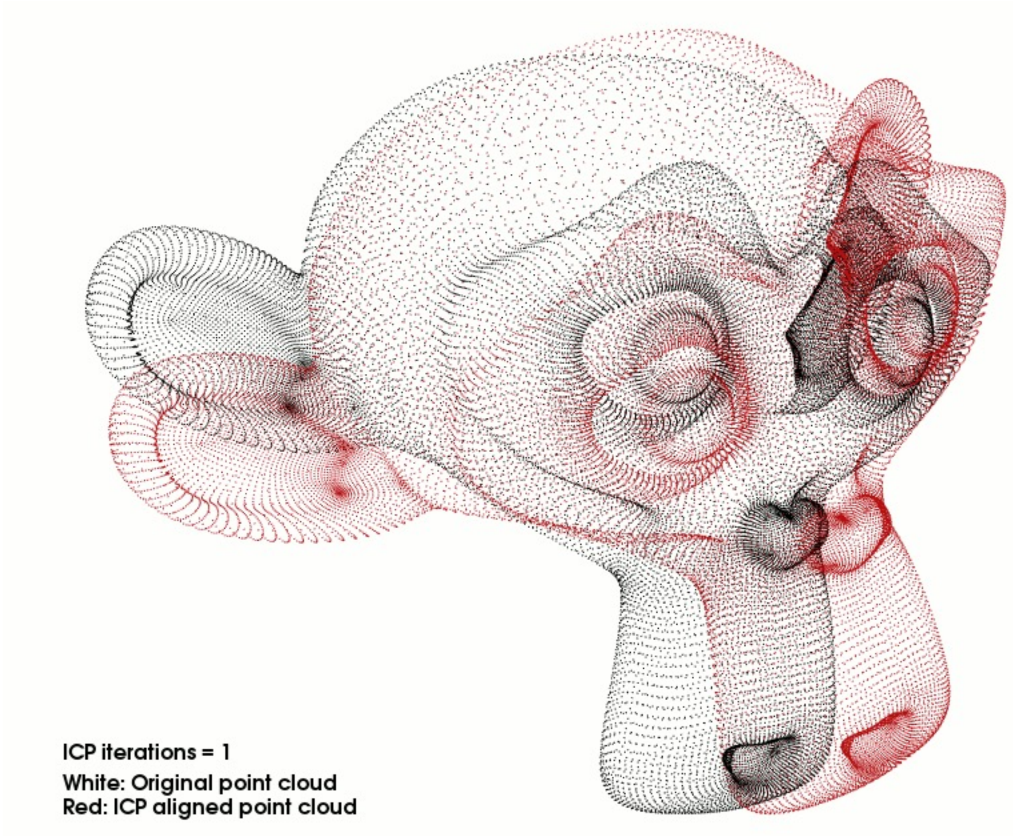
Image Credits: Innoviz

Aligning two point clouds using Iterative Closest Point Algorithm



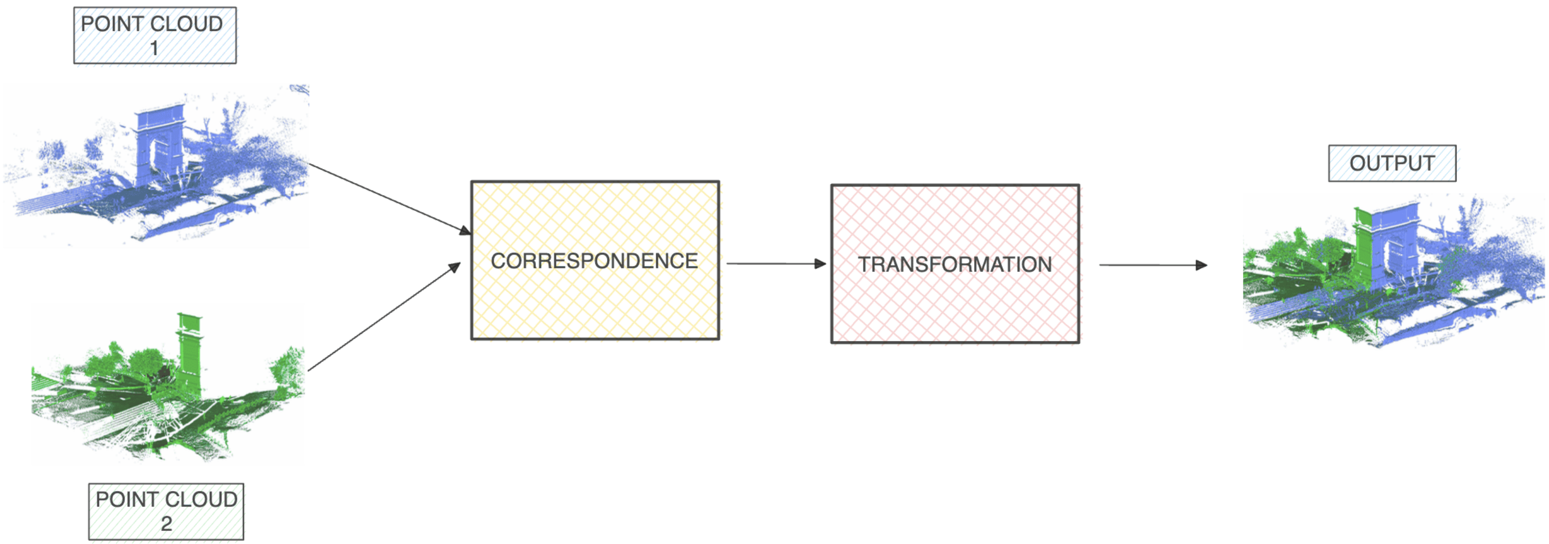
Source: Biorobotics Lab at Carnegie Mellon University.

Aligning two point clouds using Iterative Closest Point Algorithm



Source: Point Cloud Library Documentation

Two key elements: Correspondence and Transformation



More on this next time!