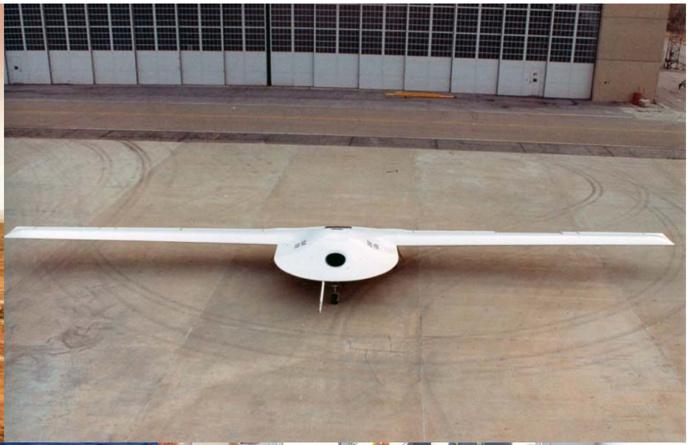
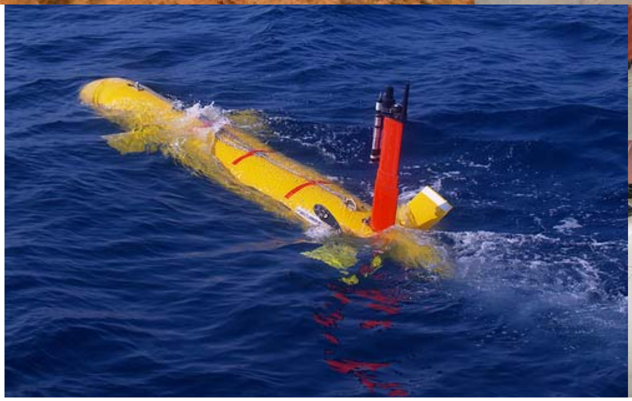
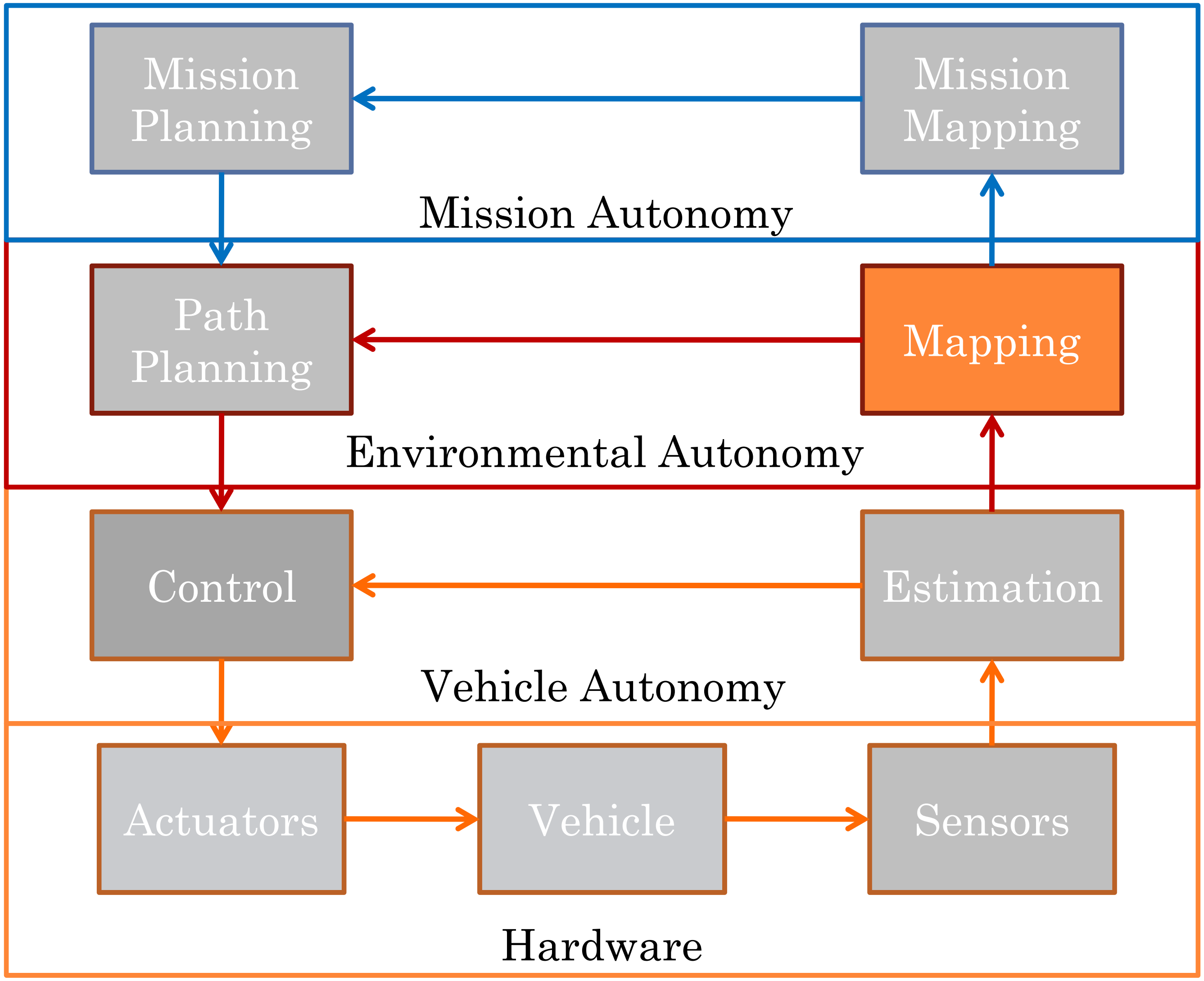




ME 597: Autonomous Mobile Robotics
Section 8 – Mapping II

Prof. Steven Waslander

Components



Mission Planning
Mission Mapping
Mission Autonomy
Path Planning
Mapping
Environmental Autonomy
Control
Estimation
Vehicle Autonomy
Actuators
Vehicle
Sensors
Hardware

OUTLINE

- Localization
  - EKF
  - Particle

- Mapping
  - Occupancy Grid based

- Simultaneous Localization and Mapping
  - EKF SLAM
  - Particle based FastSLAM
  - Occupancy Grid SLAM
  - Iterated Closest Point Scan Matching
  - Pose Graph Optimization

Simultaneous Localization and Mapping

- SLAM
  - Given
    - Motion model
    - Measurement model
    - Uniquely identifiable static features
    - Vehicle inputs, u_t
    - Measurements to some features, y_t
  - Find
    - Vehicle state, x_t^r
    - Feature locations, m^i
  - Relative calculation, coordinate system determined upon initialization
  - Significantly larger estimation problem than straight localization

SLAM

- SLAM Types
  - Online SLAM
    - Estimates the current state and the map given all information to date

$$p(x_t^r, m \mid y_{1:t}, u_{1:t})$$

    - Most useful for a moving vehicle that needs to estimate its state relative to its environment in real time
    - Usually run online
  - Full SLAM
    - Estimates the entire state history and the map given all information

$$p(x_{1:t}^r, m \mid y_{1:t}, u_{1:t})$$

    - Most useful for creating maps from sensor data after the fact
    - Usually run in batch mode

SLAM

- The four main SLAM Algorithms in Thrun
  - EKF/UKF SLAM (Thrun et al. Chap 10)
    - Extension of EKF localization to online SLAM problem
    - Very commonly used, especially for improving vehicle state estimation when static features are available
  - GraphSLAM (Thrun et al. Chap 11)
    - Solves the full SLAM problem by storing data as a set of constraints between variables
    - Can create maps based on 1000s of features, not possible with EKF due to matrix inversion limitations
    - Many variations, all boil down to a nonlinear optimization that needs to be fast to be useful

SLAM

- The four main SLAM Algorithms in Thrun
  - Sparse Extended Information Filter SLAM (Thrun et al. Chap 12)
    - Approximate application of Extended Information Filter to SLAM problem
    - Can create a sparse (nearly diagonal) information matrix, which also enables tracking many features, constant time updates
  - FastSLAM (Thrun et al. Chap 13)
    - Solves the online SLAM problem simultaneously by combining particles and EKFs
      - Rao-Blackwellized particle filters
    - Can track multiple correspondences with different particles
    - Shows robustness to incorrect correspondence
    - Most active area of research, large scale mapping

SLAM

- Our focus is the online SLAM problem
  - EKF SLAM
    - Quick SLAM solution, great for improving vehicle state estimation from information about the environment
    - Not too robust to incorrect feature correspondence
      - Be sure to pick features wisely
  - FastSLAM
    - A more robust approach, particularly with respect to feature correspondence
    - Computationally more expensive, especially with higher dimension vehicle state
  - Occupancy Grid SLAM
    - FastSLAM with mapping by each pixel
- But, I'll introduce GraphSLAM too
    - Predominant area of research over the last decade
    - Super-impressive results

SLAM

- A brittle problem, regardless of algorithm
  - Attempting to estimate nT + fM states using MT, 2MT, 3MT measurements, depending on sensor
    - T is the number of time steps
    - M is the number of features
    - n is the number of vehicle state variables
    - f is the number of map feature variables
  - Direct sensing of vehicle states can significantly improve estimation
    - GPS, odometry information very effective at reducing uncertainty
    - Use what you can

EKF SLAM

- Variables
  - Full state
    - Vehicle states
    - Feature locations
    - Signatures
      - Not included here

$$x_t = \begin{bmatrix} x_t^r \\ m \end{bmatrix} = \begin{bmatrix} x_t^r \\ m_x^1 \\ m_y^1 \\ \vdots \\ m_x^M \\ m_y^M \end{bmatrix}$$

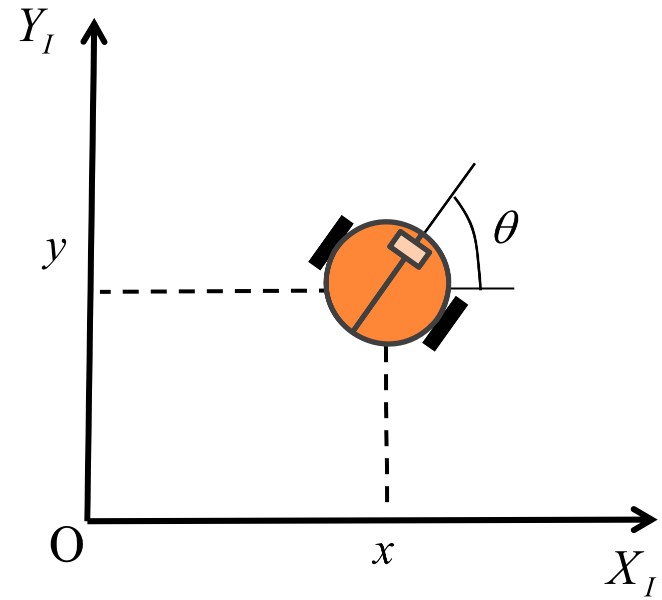
  - Belief: Full state mean and covariance
    - Components for vehicle state and map state

$$\mu_t = \begin{bmatrix} \mu_t^r \\ \mu_t^m \end{bmatrix} \quad \begin{matrix} \text{Robot} \\ \text{Map} \end{matrix} \qquad \Sigma_t = \begin{bmatrix} \Sigma_t^{rr} & \Sigma_t^{rm} \\ \Sigma_t^{mr} & \Sigma_t^{mm} \end{bmatrix}$$

EKF SLAM

- Once again, investigate with a specific vehicle and measurement model

$$\begin{bmatrix} x_1^r \\ x_2^r \\ x_3^r \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \qquad \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} v \\ \omega \end{bmatrix}$$

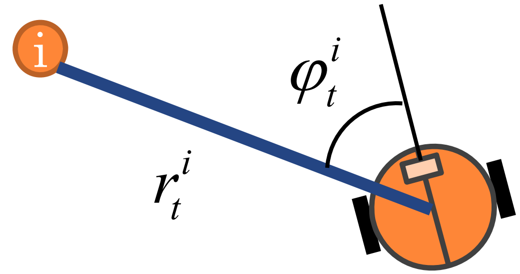


- Motion model for robot only
  - Feature are static, no motion

$$\begin{bmatrix} x_{1,t}^r \\ x_{2,t}^r \\ x_{3,t}^r \end{bmatrix} = g(x_{t-1}^r, u_t, \varepsilon_t) = \begin{bmatrix} x_{1,t-1}^r + u_{1,t} \cos x_{3,t-1}^r dt \\ x_{2,t-1}^r + u_{1,t} \sin x_{3,t-1}^r dt \\ x_{3,t-1}^r + u_{2,t} dt \end{bmatrix} + \varepsilon_t$$

$$\varepsilon_t \sim N(0, R_t)$$

EKF SLAM



- Measurement Model
  - Relative range and/or bearing to numerous features m^i in field of view
  - Define $\delta x_t^i = m_x^i - x_{1,t}$ $\delta y_t^i = m_y^i - x_{2,t}$

$$r_t^i = \sqrt{\left(\delta x_t^i\right)^2 + \left(\delta y_t^i\right)^2}$$

  - Then

$$\begin{bmatrix} y_{1,t}^i \\ y_{2,t}^i \end{bmatrix} = h^i(x_t, \delta_t) = \begin{bmatrix} \varphi_t^i \\ r_t^i \end{bmatrix} = \begin{bmatrix} \tan^{-1}\left(\frac{\delta y_t^i}{\delta x_t^i}\right) - x_{3,t}^r \\ \sqrt{\left(\delta x_t^i\right)^2 + \left(\delta y_t^i\right)^2} \end{bmatrix} + \delta_t$$

Bearing

Range

  - Noise

$$\delta_t \sim N(0, Q_t)$$

EKF SLAM

- Vehicle Prior
  - In localization or mapping, coordinate system was clearly defined
    - Localization relative to fixed map
    - Mapping relative to known vehicle motion
  - In pure SLAM, neither is known, so coordinate system is arbitrary choice
    - Assume vehicle starts at origin with zero heading
    - Know this with absolute certainty

$$x_0^r = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T \qquad \Sigma_0^{rr} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

EKF SLAM

- Map Prior
  - No clue where any of the features are
    - Theoretically, we could say

$$x_0^m = \begin{bmatrix} 0 & 0 & \dots & 0 & 0 \end{bmatrix}^T \qquad \Sigma_0^{mm} = \begin{bmatrix} \infty & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \infty \end{bmatrix}$$

    - In practice, not very useful
      - Linearization with all features assumed to be at the origin performs very poorly
      - Inversion with infinite diagonal numerically difficult

EKF SLAM

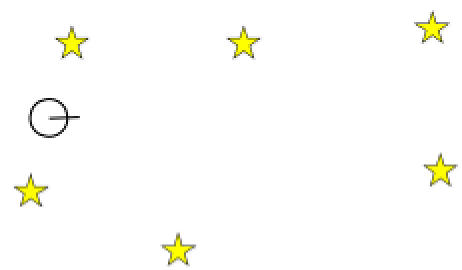
- Map Prior
  - Preferred method
    - Initialize each feature location based on first set of measurements
      - Measurements must uniquely define feature position
      - Bearing and range + vehicle state required

$$\mu_t^i = \begin{bmatrix} x_{1,t}^r + y_{2,t}^i \cos\left(y_{1,t}^i + x_{3,t}^r\right) \\ x_{2,t}^r + y_{2,t}^i \sin\left(y_{1,t}^i + x_{3,t}^r\right) \end{bmatrix}$$

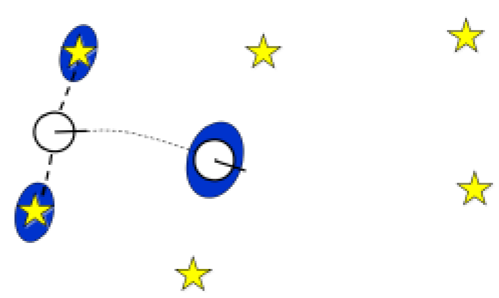
      - Can define covariance based on measurement noise and vehicle state uncertainty, or predefine explicitly
    - If initial measurements are insufficient, can accumulate multiple measurements before initialization
      - Bearing only SLAM (for vision data)

EKF SLAM

- A sketch
  - A vehicle and a set of features, perfect knowledge of vehicle location initially

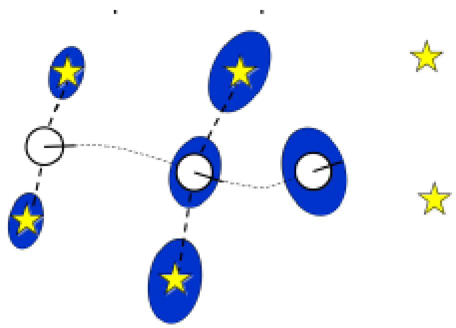


  - The vehicle measures the location of two features and moves one time step forward
    - Measurement and motion uncertainty



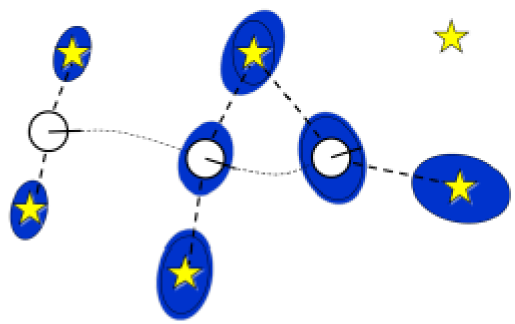
EKF SLAM

- A sketch
  - At the next time step, two new features are observed with more uncertainty
    - Combination of vehicle and measurement uncertainty
    - Motion uncertainty continues to grow



EKF SLAM

- A sketch
  - The next set of measurements includes a feature that has already been observed
    - The vehicle uncertainty can be reduced
    - The additional features are not as uncertain



  - The result: as old features are discarded and new features added, uncertainty grows

EKF SLAM

- EKF SLAM Algorithm
  - Prediction step
    - Only vehicle states and covariance change
    - Map states and covariance are unaffected
    - Quick 3X3 update

$$G_t = \frac{\partial}{\partial x_{t-1}^r} g(x_{t-1}^r, u_t)\Big|_{x_{t-1}^r = \mu_{t-1}^r}$$

$$\bar{\mu}_t^r = g(\mu_{t-1}^r, u_t)$$

$$\bar{\Sigma}_t^{rr} = G_t \Sigma_{t-1}^{rr} G_t^T + R_t$$

EKF SLAM

- Linearization of Motion Model, as before

$$\begin{bmatrix} x^r_{1,t} \\ x^r_{2,t} \\ x^r_{3,t} \end{bmatrix} = g(x^r_{t-1}, u_t) = \begin{bmatrix} x^r_{1,t-1} + u_{1,t} \cos x^r_{3,t-1} dt \\ x^r_{2,t-1} + u_{1,t} \sin x^r_{3,t-1} dt \\ x^r_{3,t-1} + u_{2,t} dt \end{bmatrix}$$

$$\Downarrow$$

$$G_t = \frac{\partial}{\partial x^r_{t-1}} g(x^r_{t-1}, u_t) = \begin{bmatrix} 1 & 0 & -u_{1,t} \sin x^r_{3,t-1} dt \\ 0 & 1 & u_{1,t} \cos x^r_{3,t-1} dt \\ 0 & 0 & 1 \end{bmatrix}$$

EKF SLAM

- EKF SLAM Algorithm
  - Measurement Update, for feature *i*
    - Since each measurement pair depends on one feature, independence means updates can be performed one feature at a time

$$H_t^i = \frac{\partial}{\partial x_t} h^i(x_t) \Big|_{x_t = \bar{\mu}_t}$$

$$K_t^i = \bar{\Sigma}_t \left(H_t^i\right)^T (H_t^i \bar{\Sigma}_t \left(H_t^i\right)^T + Q_t)^{-1}$$

$$\mu_t = \bar{\mu}_t + K_t^i (y_t - h(\bar{\mu}_t))$$

$$\Sigma_t = (I - K_t H_t^i) \bar{\Sigma}_t$$

EKF SLAM

- Linearization of Measurement Model

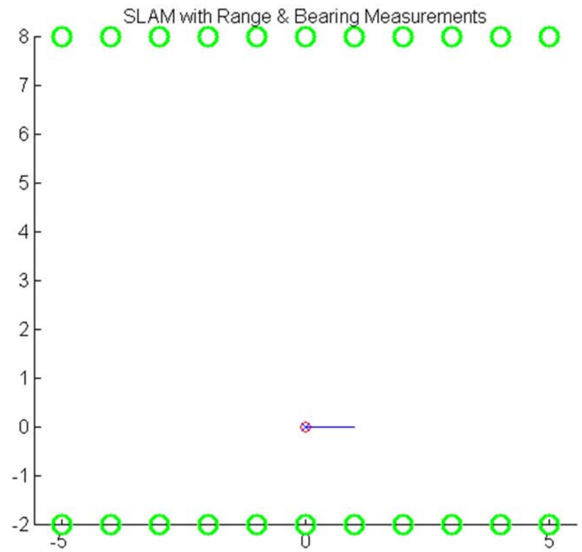
$$\begin{bmatrix} y^i_{1,t} \\ y^i_{2,t} \end{bmatrix} = h^i(x_t) = \begin{bmatrix} \tan^{-1}\left(\dfrac{dy^i_t}{dx^i_t}\right) - x^r_{3,t} \\ \sqrt{\left(dx^i_t\right)^2 + \left(dy^i_t\right)^2} \end{bmatrix}$$

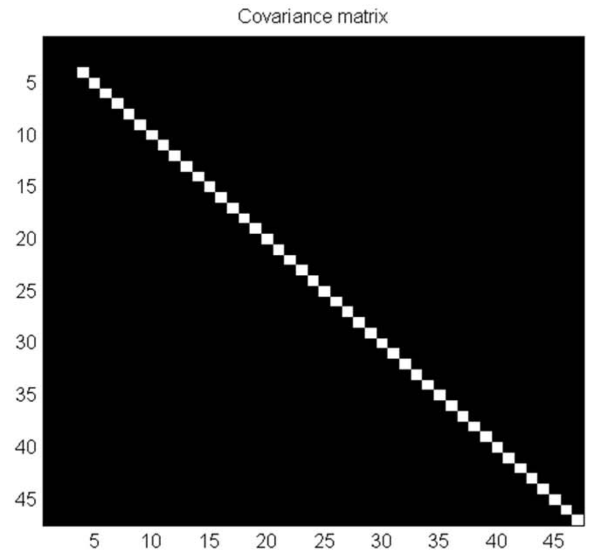
$$H^i_t = \frac{\partial}{\partial x_t} h^i(x_t) = \begin{bmatrix} \dfrac{dy^i_t}{r^2} & \dfrac{-dx^i_t}{r^2} & -1 & 0 \cdots 0 & \dfrac{-dy^i_t}{r^2} & \dfrac{dx^i_t}{r^2} & 0 \cdots 0 \\ \dfrac{-dx^i_t}{r} & \dfrac{-dy^i_t}{r} & 0 & 0 \cdots 0 & \dfrac{dx^i_t}{r} & \dfrac{dy^i_t}{r} & 0 \cdots 0 \end{bmatrix}$$

- Derivatives w.r.t. m^i in appropriate columns

EKF SLAM

- Example
  - 22 features in two lines
  - Same circular motion as for localization example
  - Field of view similar to camera
    - +/- 45 degrees
    - 5 m range

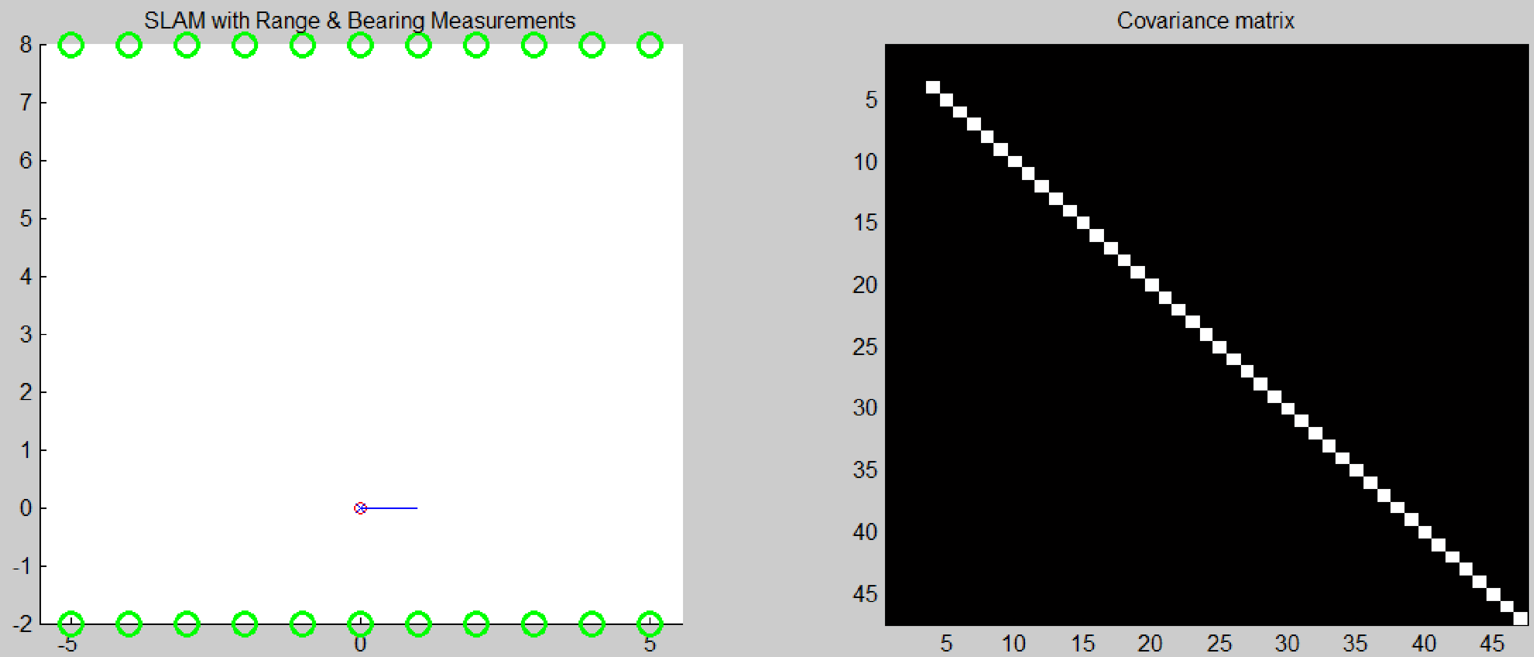




EKF SLAM

True state -o-
Belief -x-
Measurement ____
Features O

- Example



EKF SLAM

- Discussion
  - Vehicle state error correlates feature estimates
    - If vehicle state known exactly (mapping) features could be estimated independently
    - Knowing more about one feature improves estimates about entire map
  - Covariance matrix divided in 3X3 structure
    - Vehicle state and two sets of features
    - Each row of features strongly connected
    - Rows weakly connected by uncertain multiple time step motion
  - Growth in state uncertainty without loop closure
    - When first feature is re-observed, all estimates improve
    - Correction information carried in covariance matrix

EKF SLAM

- Wrong correspondence can be catastrophic
  - Linearization about wrong point can cause deterioration of estimate, divergence of covariance

- Strategies
  - Provisional Feature list
    - Features on the list are tracked identically to other features
    - Not used to update vehicle state or vehicle/map covariance
    - Once trace of covariance drops below threshold, incorporate feature into map

  - Feature selection
    - Features are selected so as to avoid correspondence issues
      - Spatially distributed
      - Distinct signatures

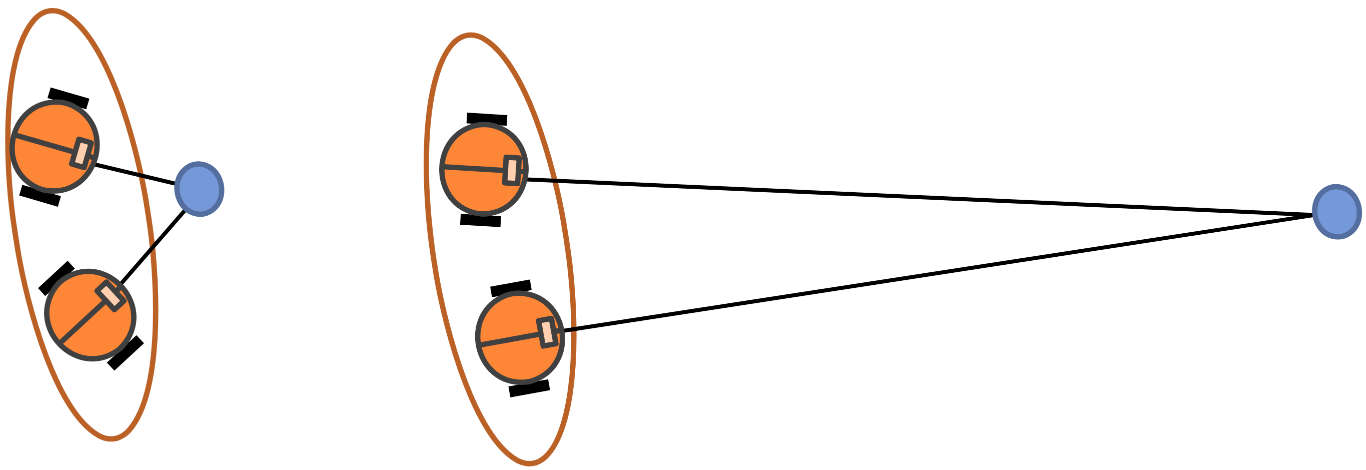
  - Feature Tracking and windowed correspondence
    - Features can be expected to move in a consistent way from frame to frame, so only a subset of features need be considered for matches

Outline

- Localization
  - EKF
  - Particle
- Mapping
  - Occupancy Grid based
- Simultaneous Localization and Mapping
  - EKF SLAM
  - Particle based FastSLAM
  - Occupancy Grid SLAM
  - Iterated Closest Point Scan Matching
  - Pose Graph Optimization

FastSLAM

- Divergence Issue with EKF primarily due to linearization about incorrect estimate
  - Fails when linearization is a poor approximation
    - Features at close range accentuate issue



- Particle filters avoid this linearization

FASTSLAM

- Recall Particle Filter Algorithm
  1. For each particle in S_{t-1}
     1. Propagate sample forward using motion model (sampling)

        $$\bar{x}_t^{r[i]} \sim p(x_t^r \mid x_{t-1}^{r[i]}, u_t)$$

     2. Calculate weight (importance)

        $$w_t^{[i]} = p(y_t \mid \bar{x}_t^{r[i]})$$

     3. Store in interim particle set

        $$S'_t = S'_t + \{s_t^{[i]}\}$$

  2. Normalize weights
  3. For $j = 1$ to I
     1. Draw index i with probability $\propto w_t^{[i]}$ (resampling)
        1. Add to final particle set

           $$S_t = S_t + \{s_t^{[i]}\}$$

FastSLAM

- Direct Particle Filter approach
  - Applied to example SLAM problem, state is too large to capture distributions with particles
    - Exponential growth in number of particles needed per dimension of the problem
  - SLAM problem has significant structure
    - Map features do not move
    - Measurements depend on only the vehicle state and one feature
  - Need a way to avoid issues of EKF and particle filters

FastSLAM

- Rao-Blackwellized Particle Filter
  - The vehicle state will be estimated with particles
  - Each feature will be estimated with an independent EKF
  - Each particle has the vehicle state and a bank of EKFs, one for each feature in the map

Particle	Robot State	Features		
1	$x^r_{t,1}$	$\mu^1_{t,1}, \Sigma^1_{t,1}$	$\cdots$	$\mu^M_{t,1}, \Sigma^M_{t,1}$
2	$x^r_{t,2}$	$\mu^1_{t,2}, \Sigma^1_{t,2}$		$\mu^M_{t,2}, \Sigma^M_{t,2}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
I	$x^r_{t,I}$	$\mu^1_{t,I}, \Sigma^1_{t,I}$	$\cdots$	$\mu^M_{t,I}, \Sigma^M_{t,I}$

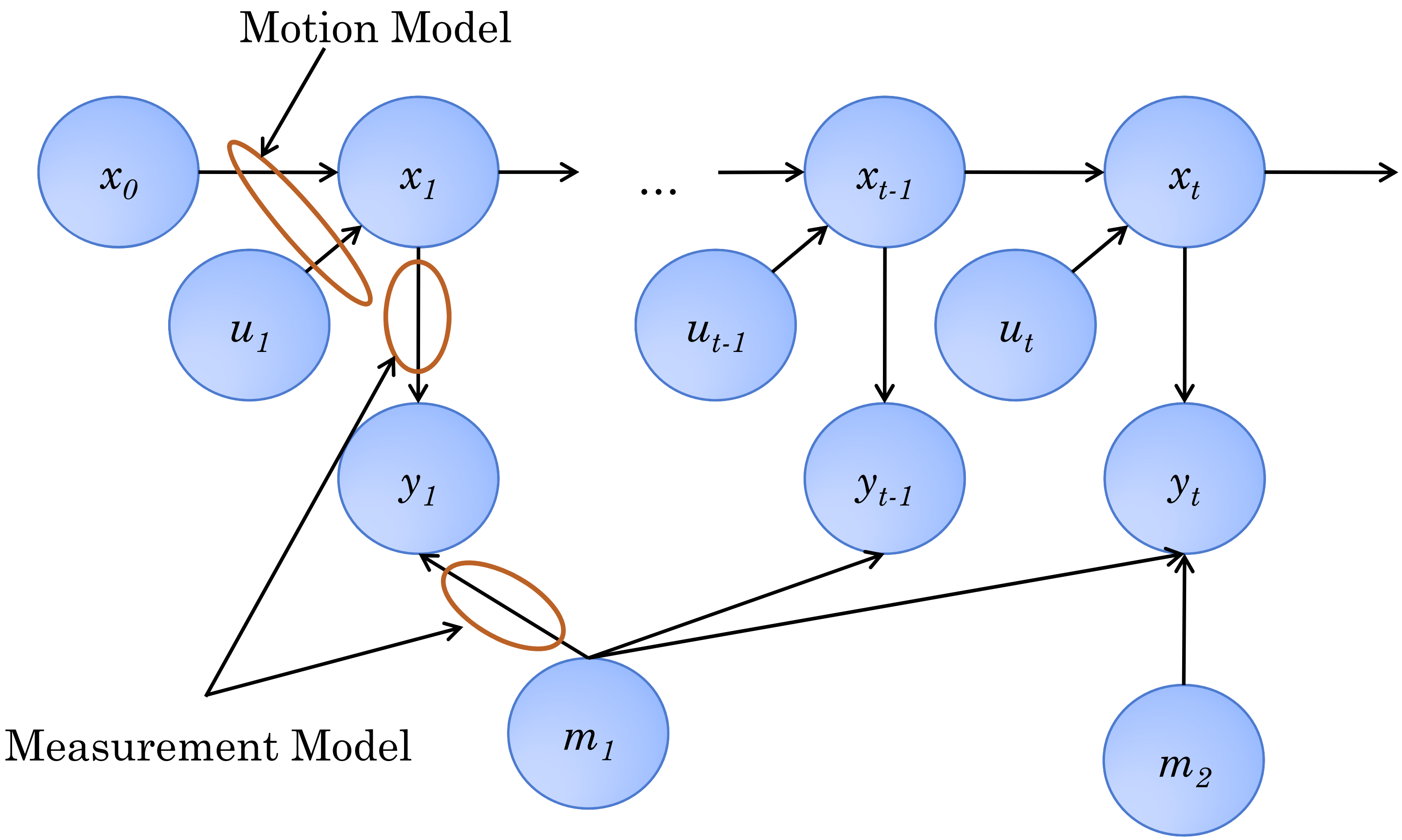
FastSLAM

- Key Insight
  - If vehicle state is known exactly, feature locations can be estimated independently
  - In a particle filter, each particle represents an exact belief about the state
  - Representing vehicle state belief with particles allows independent estimation of features for each particle
    - M+1 separate independent beliefs

$$p(x_{1:t} \mid y_{1:t}, u_{1:t}) = p(x^r_{1:t} \mid y_{1:t}, u_{1:t}) \prod_{i=1}^{M} p(m^i \mid y_{1:t}, u_{1:t})$$

FASTSLAM

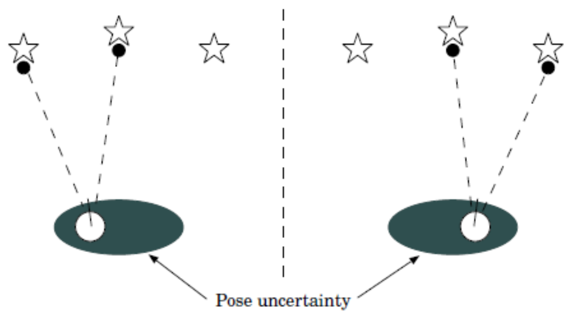
- Hidden Markov Model



FASTSLAM

- Feature Correspondence
  - Can also be incorporated, each particle need not use the same correspondence decisions
  - Avoids issue with EKF
  - Larger estimation problem, more particles needed

EKF

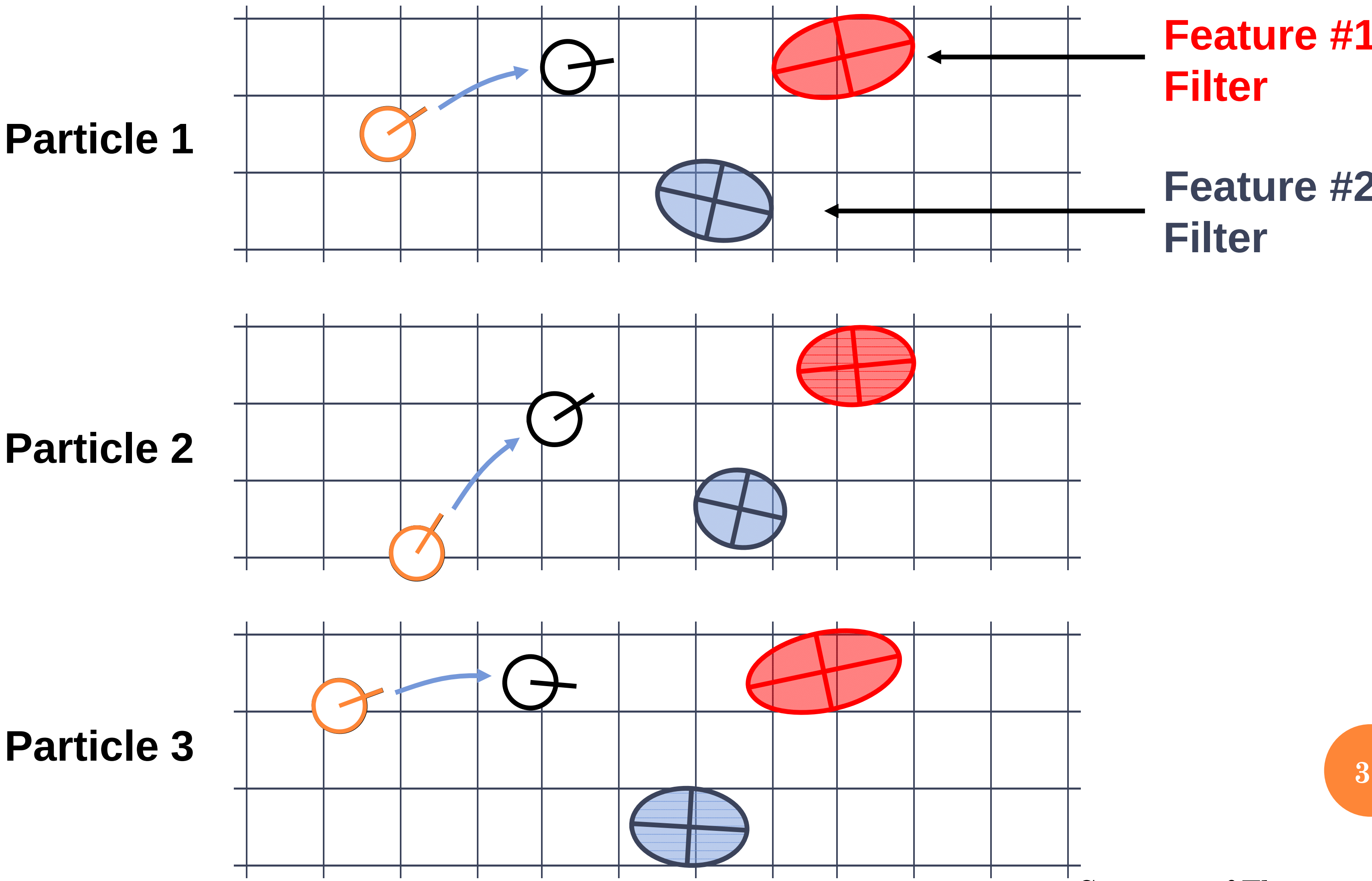


leads to divergence

Particle

consistent

FastSLAM – Prediction Step



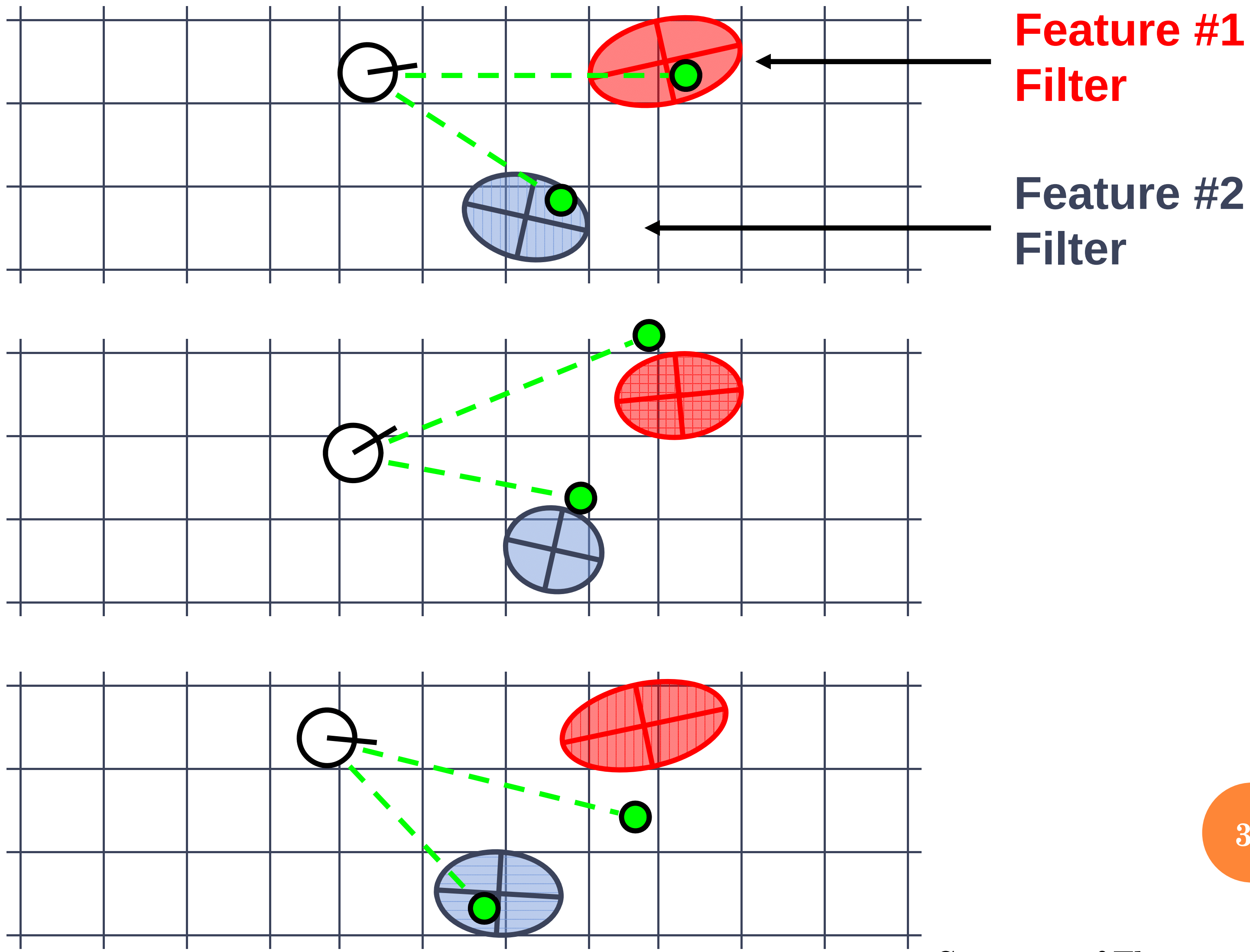
Courtesy of Thrun et al.

FastSLAM – Measurement Update

Particle 1

Particle 2

Particle 3



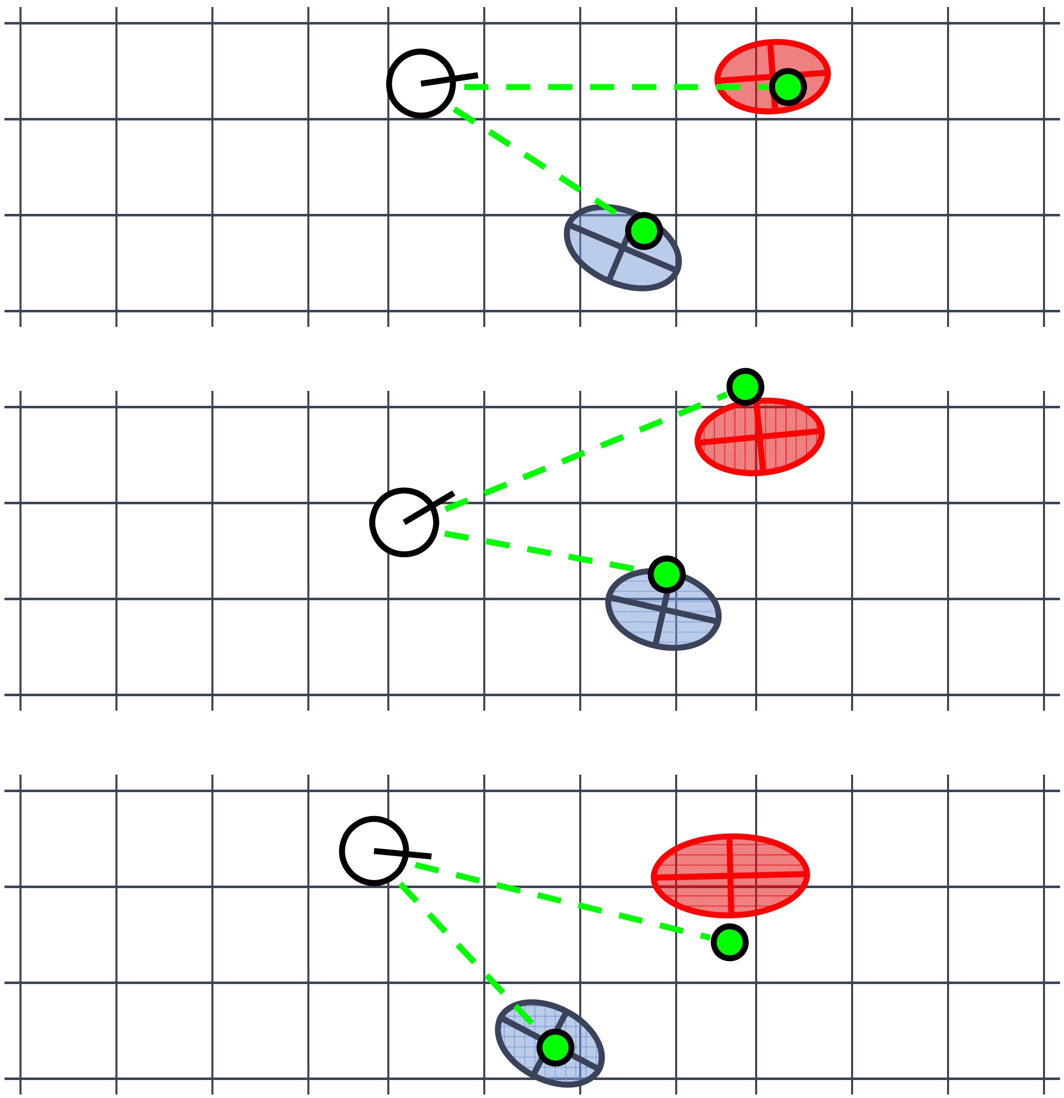
Courtesy of Thrun et al.

FastSLAM – Sensor Update

Particle 1 — Weight = 0.8

Particle 2 — Weight = 0.4

Particle 3 — Weight = 0.1



Courtesy of Thrun et al.

FastSLAM

- Prediction Step
  - Like Particle filter localization, propagate each particle through motion model with disturbance sample

$$x^r_{t,j} \sim p(x^r_t \mid x^r_{t-1,j}, u_t)$$

  - O(I), linear in the number of particles

FastSLAM

- Measurement Update
  - For each particle
    - Initialize EKF for each newly observed feature

$$\mu_{t,j}^{i} = h^{i^{-1}}\left(y_t, x_{t,j}^{r}\right)$$

$$H_{t,j}^{i} = \frac{\partial}{\partial m^{i}} h^{i}(x_t)\Big|_{x_t=[x_{t,j}^{r}\ \ \mu_{t-1,j}^{i}]}$$

$$\Sigma_{t,j}^{i} = H_{t,j}^{i^{-1}} Q_t \left(H_{t,j}^{i^{-1}}\right)^{T}$$

$$w_j = p_0$$

FastSLAM

- Measurement Update
  - For each particle
    - Update individual EKF for each previously observed feature

$$H_{t,j}^{i} = \frac{\partial}{\partial m^{i}} h^{i}(x_t) \Big|_{x_t = [x_{t,j}^{r} \; \mu_{t-1,j}^{i}]}$$

$$K_{t,j}^{i} = \Sigma_{t-1,j}^{i} \left(H_{t,j}^{i}\right)^{T} \left(H_{t,j}^{i} \Sigma_{t-1,j}^{i} \left(H_{t,j}^{i}\right)^{T} + Q_{t}^{i}\right)^{-1}$$

$$\mu_{t,j}^{i} = \mu_{t-1,j}^{i} + K_{t,j}^{i}(y_{t}^{i} - h(\mu_{t-1,j}^{i}))$$

$$\Sigma_{t,j}^{i} = (I - K_{t,j}^{i} H_{t,j}^{i}) \Sigma_{t-1,j}^{i}$$

FastSLAM

- Measurement Update
  - Importance sampling
    - Particle Weights are probability of measurement given particle state

$$w_j = p(y_t \mid x_{t,j})$$

    - Found by linearizing about particle state

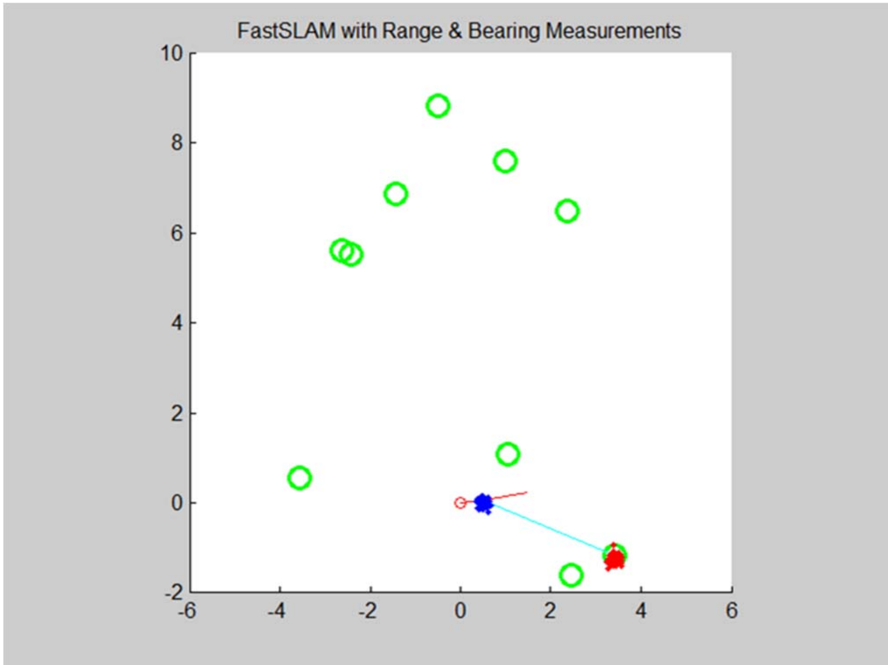
$$p(y_t \mid x_{t,j}) = \eta \, |2\pi Q_{t,j}|^{-1/2} \, e^{\left(-\frac{1}{2}\left(y_t - h(\mu_{t,j})\right)^T Q_{t,j}\left(y_t - h(\mu_{t,j})\right)\right)}$$

$$Q_{t,j} = H_{t,j}\Sigma^i_{t,j}H^T_{t,j} + Q^i_t$$

    - Resampling as before
      - Draw I samples from existing particles based on measurement model weights

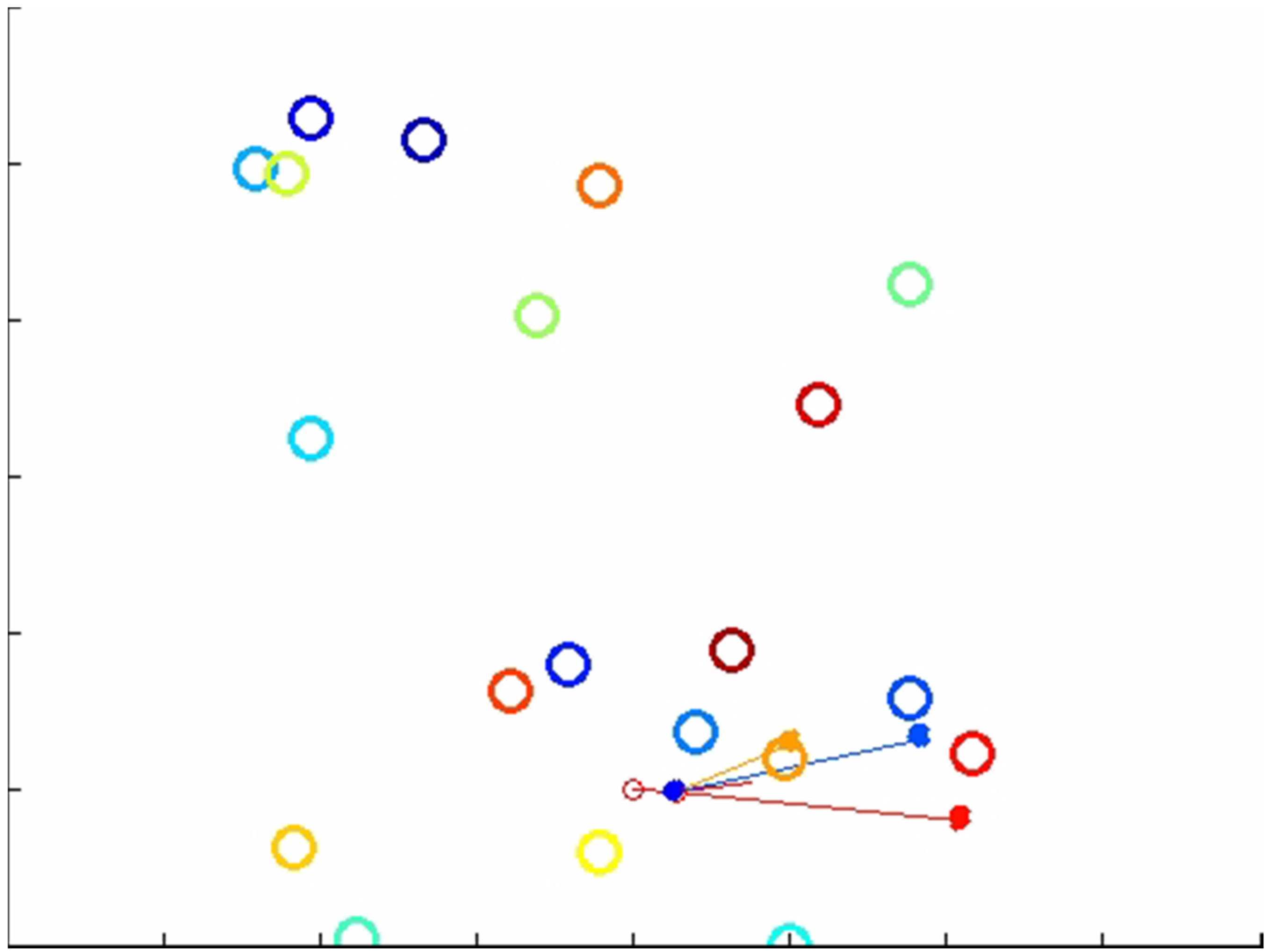
FASTSLAM

- Example
  - Two wheeled robot motion, going in a circle
  - Range and bearing measurements to features in view
    - 5 m range, 50 deg FOV
  - 100 particles, all means displayed



FastSLAM

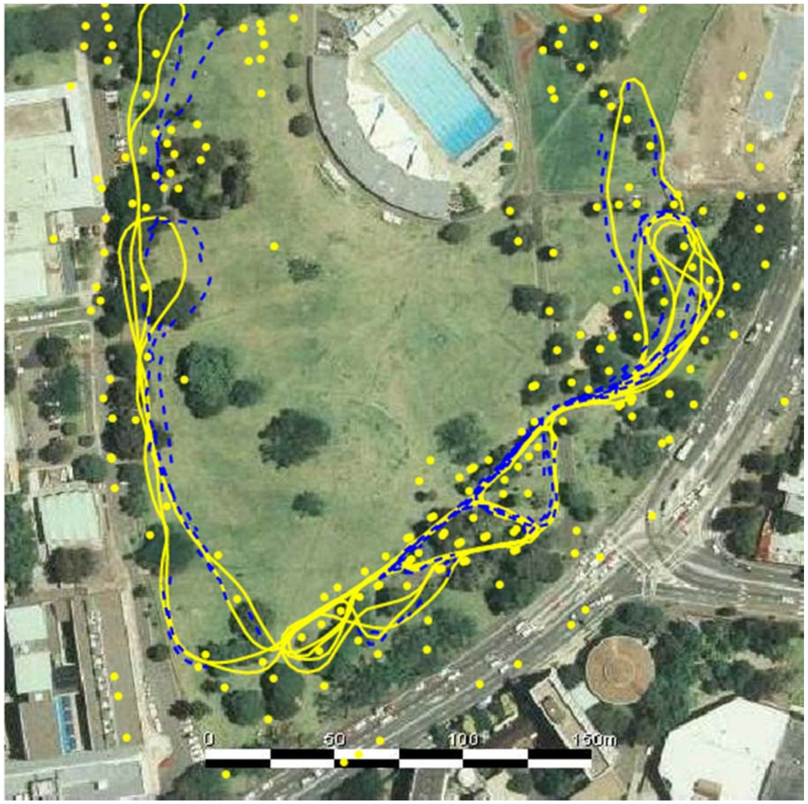
- Example



FastSLAM

- Victoria Park
  - 4 km traverse
  - < 5 m RMS position error
  - 100 particles

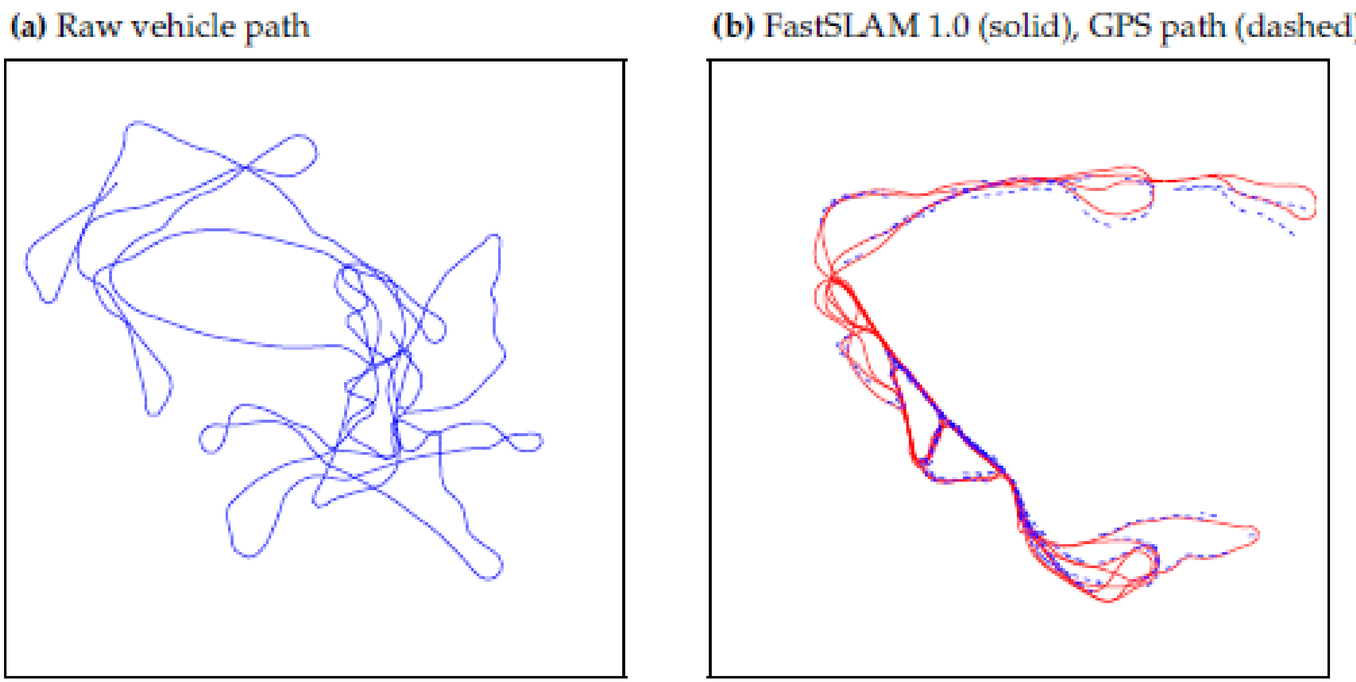
Blue = GPS
Yellow = FastSLAM



Dataset courtesy of University of Sydney
Results courtesy of Thrun et al.

FASTSLAM

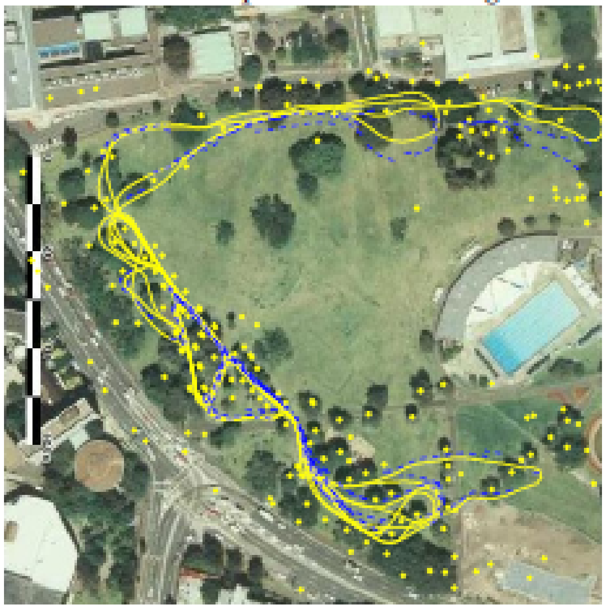
- Results from Victoria Park data set
  - Raw odometry vs FastSLAM with GPS ground truth



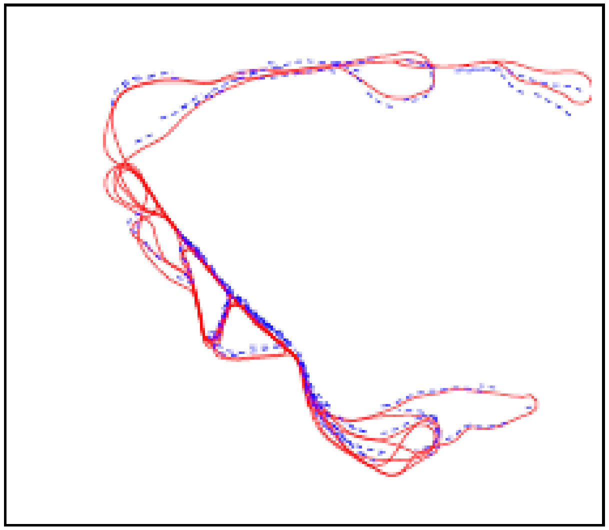
FastSLAM

- Results from Victoria Park data set
  - FastSLAM with GPS ground truth on satellite image
  - Ignoring odometry data, still successful

(c) Paths and map with aerial image



(d) Estimated path without odometry



FastSLAM

- FastSLAM 2.0
  - Improves motion update sampling to include measurement information
    - Useful when motion is relatively uncertain compared to measurements
    - Results in a better proposal distribution, which means less likely to encounter particle deprivation
      - Target distribution is closer to proposal
      - More particles present good estimates of the true state
      - More particles are weighted highly meaning more make it through resampling
    - Allows us to improve accuracy of estimation and/or reduce the number of particles needed
    - Useful for occupancy grid SLAM

OUTLINE

- Localization
  - EKF
  - Particle
- Mapping
  - Occupancy Grid based
- Simultaneous Localization and Mapping
  - EKF SLAM
  - Particle based FastSLAM
  - Occupancy Grid SLAM
  - Iterated Closest Point Scan Matching
  - Pose Graph Optimization

Occupancy Grid SLAM

- Example
  - Bruceton Research Mine
  - Results courtesy of Dirk Haehnel
  - Laser data collected while driving through underground mine

Occupancy Grid SLAM

- Occupancy grid based FastSLAM
  - Starting from the same belief representation as the FastSLAM algorithm
    - Instead of treating each feature individually, we think of the map as an occupancy grid problem

$$p(x_{1:t}^r, m \mid y_{1:t}, u_{1:t}) = p(x_{1:t}^r \mid y_{1:t}, u_{1:t})\, p(m \mid x_{1:t}^r, y_{1:t})$$

Particle filter prediction and measurement update

$$w_t^{[i]} = \eta \frac{bel(x_t)}{\overline{bel}(x_t)}$$

Occupancy grid mapping

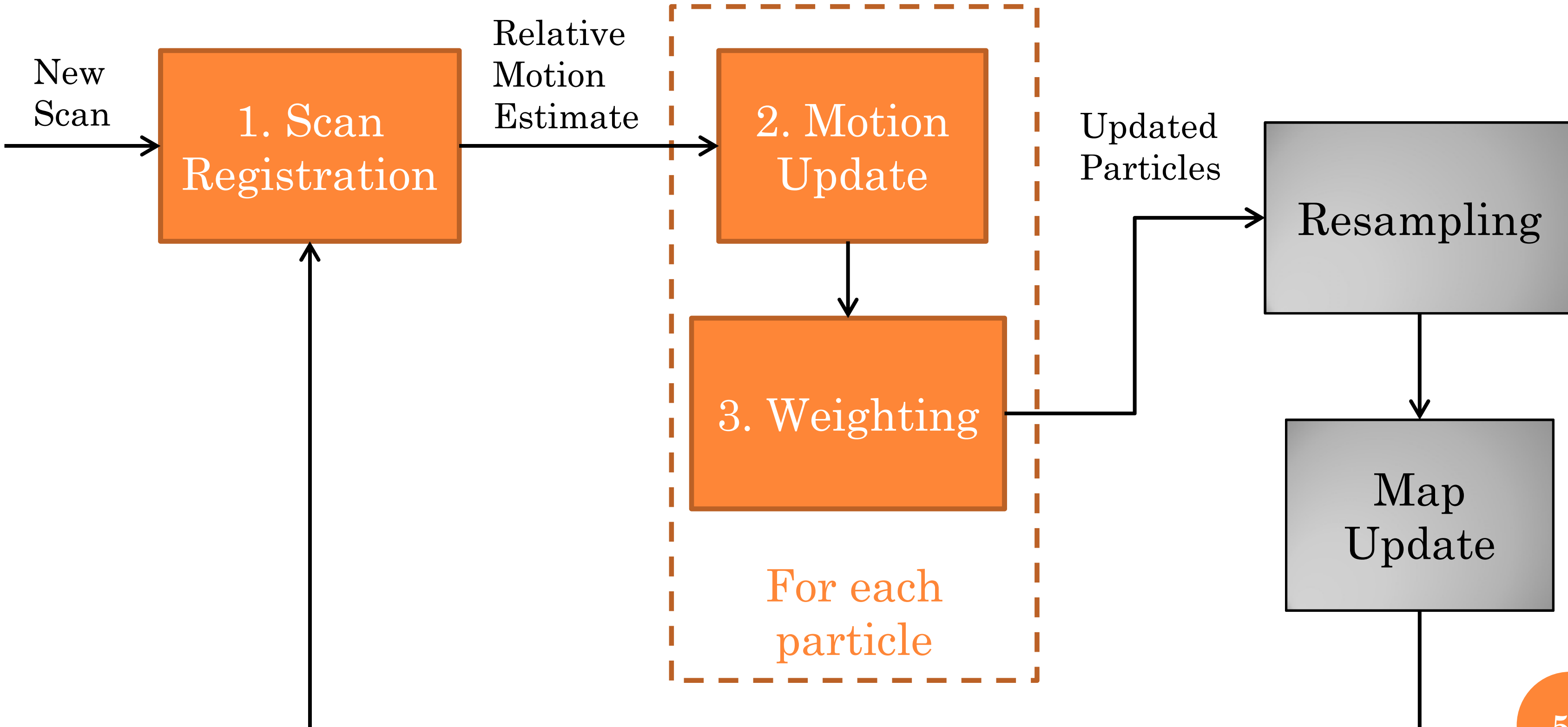
$$l_{t,i} = \text{logit}(p(m^i \mid y_t)) + l_{t-1,i} - l_{0,i}$$

Occupancy Grid SLAM

- Occupancy grid based FastSLAM: gmapping!
  - Creates complete map of the environment within each particle
  - Each cell becomes a feature with a probability of being occupied
  - Motion predictions can be improved by employing scan registration techniques
  - Weights are determined using measurement model, resampling as before
  - Occupancy probabilities are updated through inverse measurement model

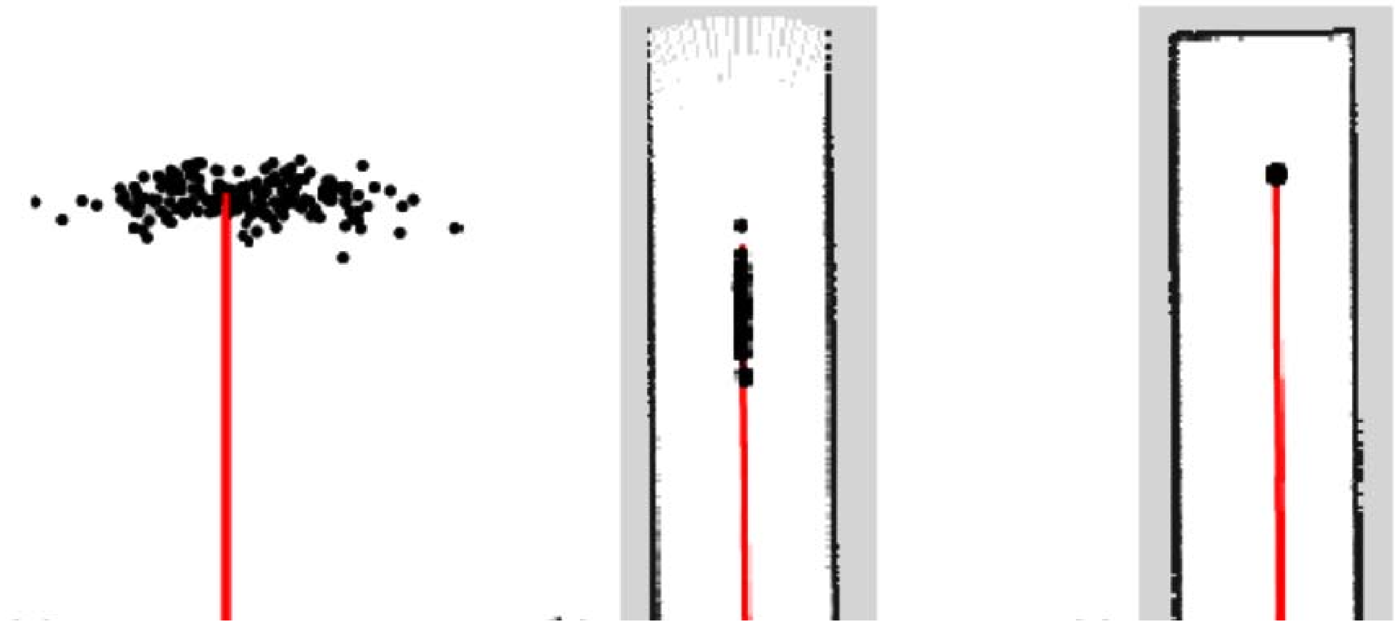
Occupancy Grid SLAM

- Occupancy grid based FastSLAM
  - Three new elements



Occupancy Grid SLAM

- Improved prediction step using scan registration
  - Disturbance distribution is dependent on scan and map



Motion model

Measurement models

Occupancy Grid SLAM

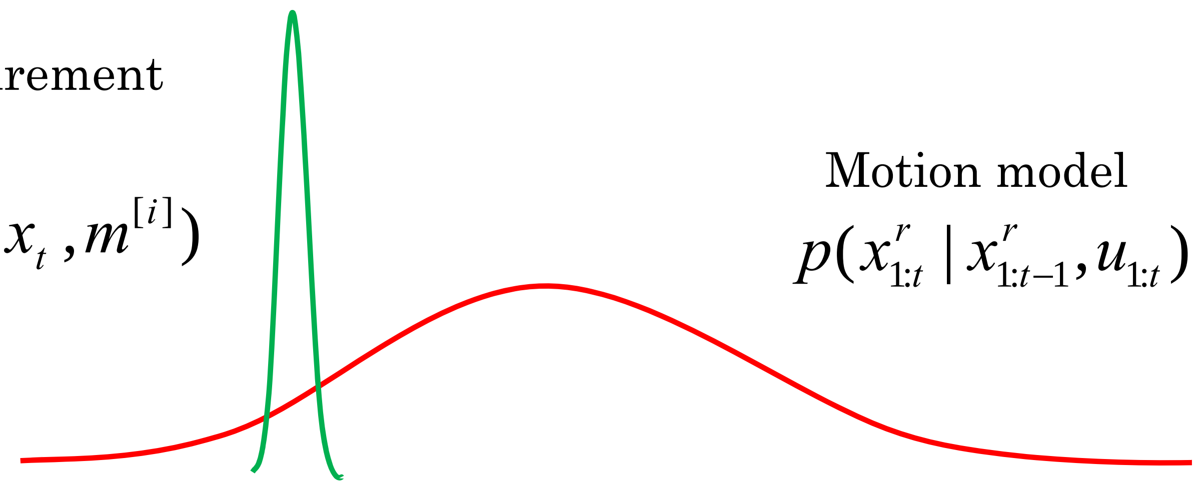
- Prediction step using scan registration
  - The idea is to include the current measurement information when updating particle location, but before incorporating it in the map
    - Apply measurement to robot pose only, save map update for later
    - Measurement model far more precise than motion model

Measurement model

$$p(y_t \mid x_t, m^{[i]})$$

Motion model

$$p(x^r_{1:t} \mid x^r_{1:t-1}, u_{1:t})$$

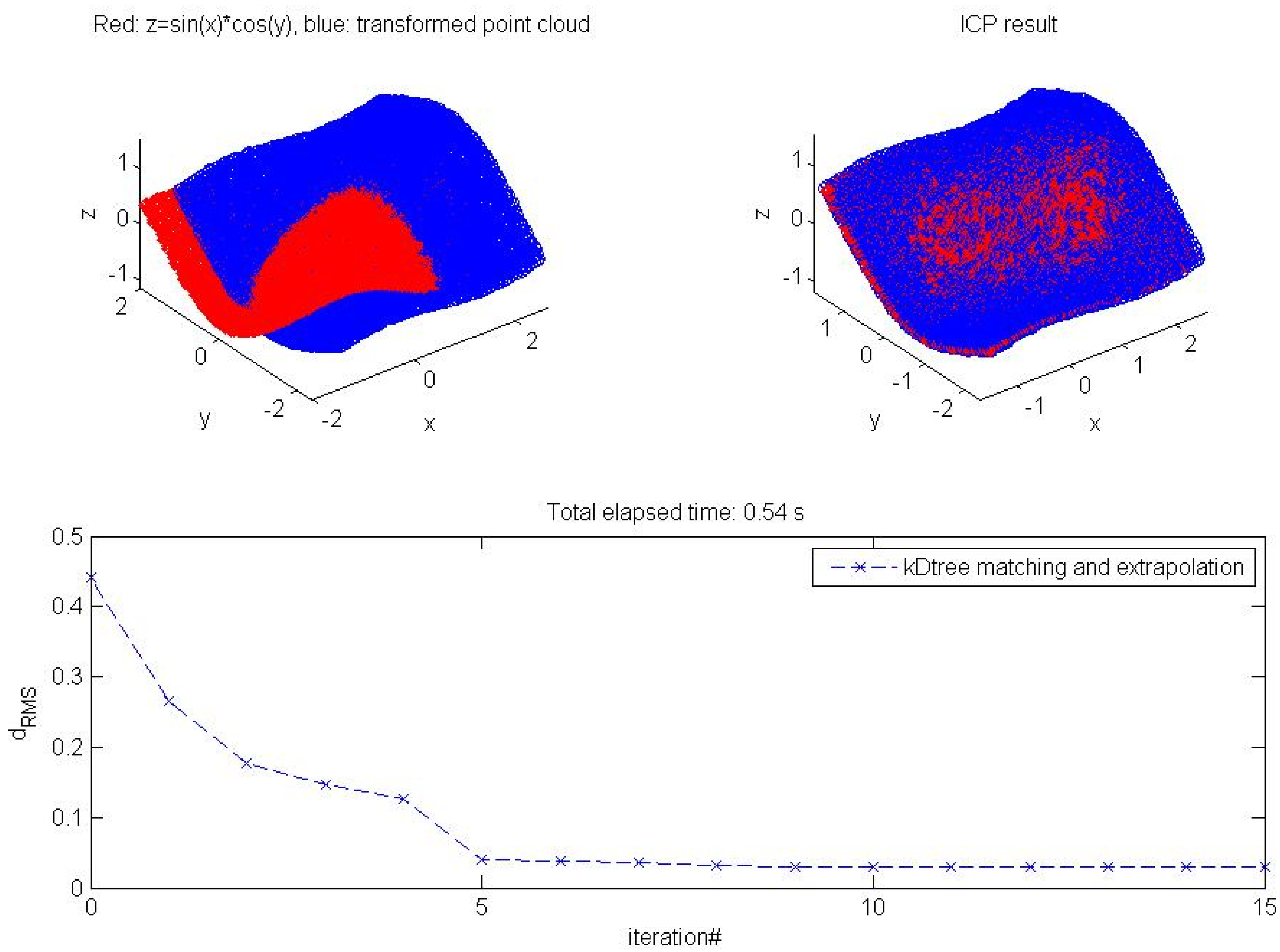


Occupancy Grid SLAM

- Prediction step using scan registration
- Use scan registration to define transformation
  - Iterative Closest Point: Given two laser scans, optimize the transformation between them by corresponding the closest points and minimizing the mean squared error.
  - Variants/Improvements
    - Generalized Iterative Closest Point : match normals
    - Normal Distribution Transform : convert to grid of Gaussians
    - Feature correspondence: only match feature points

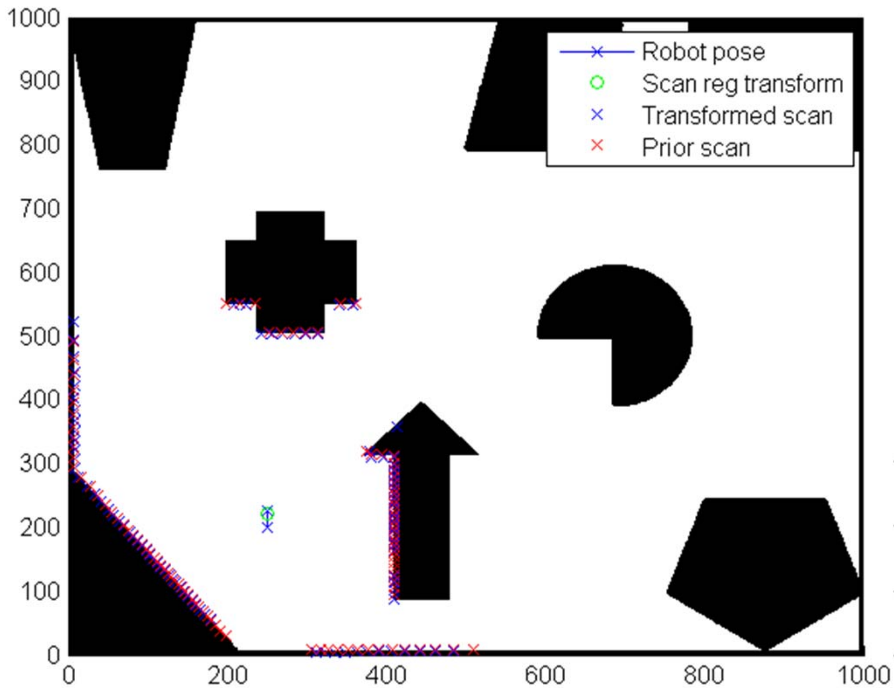
Occupancy Grid SLAM

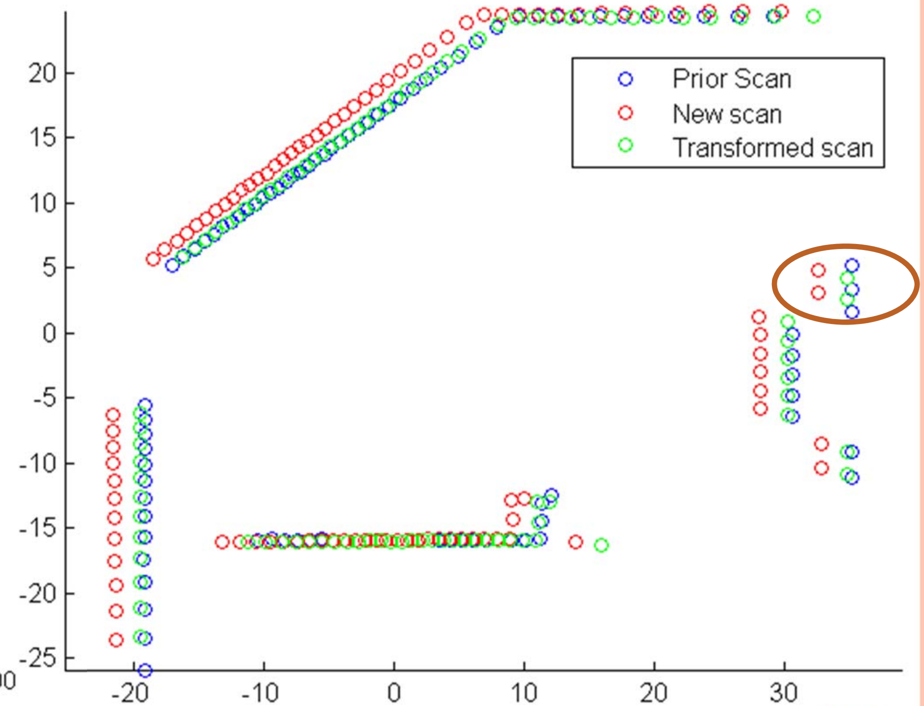
- Scan Registration Example – ICP matlab code



OCCUPANCY GRID SLAM

- Scan Registration Example – ICP on laser data





Occupancy Grid SLAM

- Prediction step using scan registration
  - Result of scan registration

$$T_t^* = \left\{ R_t^*, t_t^* \right\}$$

    - Rotation and translation needed to match new scan to previous scan or current map.
    - Provides a transformation to apply to particle robot state

$$x_t^{r[i]} = R_t^* x_{t-1}^{r[i]} + t_t^*$$

    - Must also derive disturbance distribution from which to sample a disturbance to apply to each particle

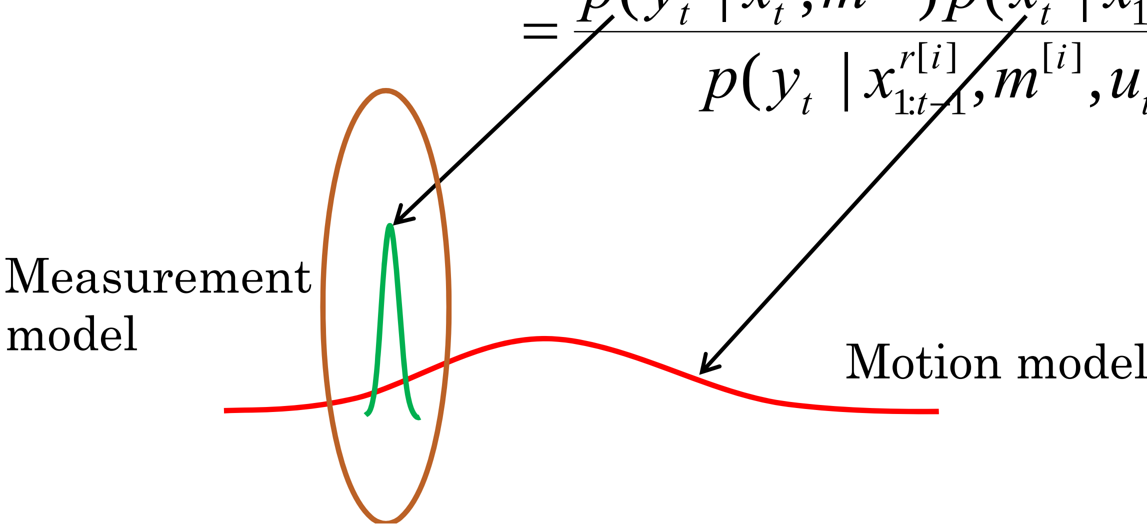
Occupancy Grid SLAM

- Prediction step using scan registration
  - In order to combine measurement model and motion model, need to evaluate both over region around scan registration estimate
  - Applying the Markov assumption

$$p(x_t^r \mid x_{1:t-1}^{r[i]}, m^{[i]}, u_{1:t}, y_{1:t}) = p(x_t^r \mid x_{1:t-1}^{r[i]}, m^{[i]}, u_t, y_t)$$

  - And Bayes Theorem + Markov

$$= \frac{p(y_t \mid x_t^r, m^{[i]})\, p(x_t^r \mid x_{1:t-1}^{r[i]}, u_t)}{p(y_t \mid x_{1:t-1}^{r[i]}, m^{[i]}, u_t)}$$

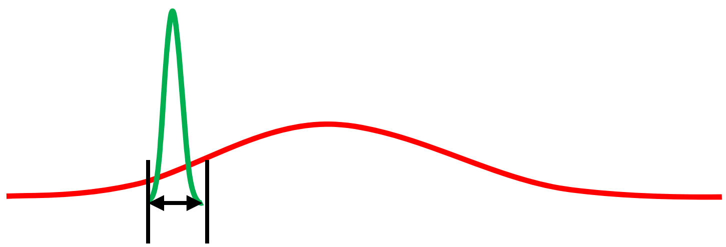


Occupancy Grid SLAM

- Prediction step using scan registration
  - Create samples around scan point, and propagate through motion and measurement models using Gaussian approximation

$$\mu^{[i]} = \frac{1}{\eta}\sum_{j=1}^{L} x_j p(y_t \mid x_t^r, m^{[i]}) p(x_t^r \mid x_{1:t-1}^{r[i]}, u_t)$$

$$\Sigma^{[i]} = \frac{1}{\eta}\sum_{j=1}^{L} \left(x_j - \mu_j\right)\left(x_j - \mu_j\right)^T p(y_t \mid x_t^r, m^{[i]}) p(x_t^r \mid x_{1:t-1}^{r[i]}, u_t)$$



Occupancy Grid SLAM

- Prediction step using scan registration
  - Measurement model
    - Given scan registration result, compute for each particle

$$p(y_t \mid x_t^{r[i]}, m^{[i]})$$

    - Done by multiplying probabilities in each cell traversed by scan
      - Let y_t^{jk} be the measurement {occupied or not occupied} for each cell j along the ray defined by a measurement k

$$p(y_t^{jk} = 1 \mid m_j) = p(m_j)$$

$$p(y_t^{jk} = 0 \mid m_j) = 1 - p(m_j)$$

      - Then the likelihood of a measurement given the map is

$$p(y_t \mid x_t^{r[i]}, m^{[i]}) = \prod_{k=1}^{K} \prod_{j=1}^{J} p(y_t^{jk} \mid m_j^{[i]})$$

Occupancy Grid SLAM

- Weighting
  - Importance sampling
    - Particle Weights can also be computed quickly through the following update equation

$$w_{t,j} = \sum_{k=1}^{K} p(y_t \mid x^r_{t,j}, m_{t-1,j})\, p(x^r_{t,j} \mid x^r_{t-1,j}, u_t)\, w_{t-1,j}$$

    - Derived from definitions of

$$w_t^{[i]} = \eta \frac{bel(x_t)}{\overline{bel}(x_t)} = \frac{p(x_t^r \mid x_{1:t-1}^{r[i]}, m^{[i]}, u_t, y_t)}{\pi(x_t^r \mid x_{1:t-1}^{r[i]}, m^{[i]}, u_t, y_t)}$$

      - Where π is the improved proposal distribution discussed above

Occupancy Grid SLAM

- Resampling
  - Most dangerous step of Particle filter update
    - Can lose good particles, lead to deprivation
  - Only perform resampling updates when necessary
    - Adaptive resampling based on threshold

$$N_{eff} = \frac{1}{\sum_{i=1}^{I} \left(w_t^{[i]} \right)^2} < \frac{I}{2}$$

- Reaches a maximum when all particles are equally weighted
- Becomes smaller as some particles are more heavily weighted than others

Occupancy Grid SLAM

- Map Update
  - Since each particle has a known position, standard mapping update applies

$$l_{t,i} = \text{logit}(p(m^i \mid y_t)) + l_{t-1,i} - l_{0,i}$$

    - The log odd ratio at *t* is the sum of the ratio at *t-1* + the inverse measurement ratio – the initial belief

  - Once again relies on inverse measurement model

$$p(m^i \mid y_t, x_t^{r[i]})$$

  - Can be delayed to after resampling to reduce number of updates required

Occupancy Grid SLAM

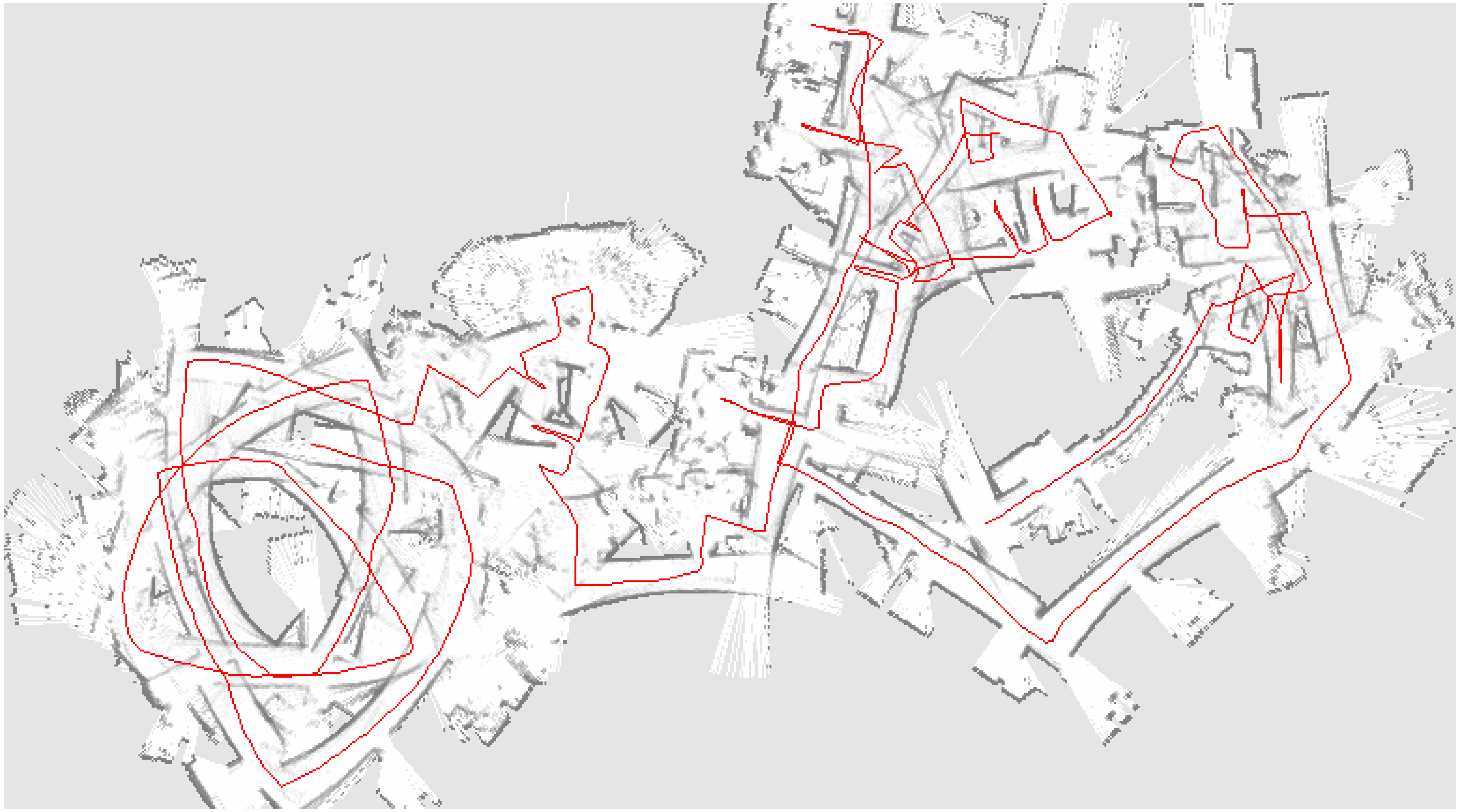
- Example Results for gmapping
  - Intel Research Lab
    - 28 m X 28 m, 2D SICK Lidar
    - Only 15 particles needed for maximum accuracy
    - Can be run in real time
  - MIT Kilian Court
    - The infinite corridor, 250m X 215m
    - 60-80 particles used
    - Nested loops

Occupancy Grid SLAM

- Intel Results – Map using only integrated wheel odometry (Dirk Haehnel)

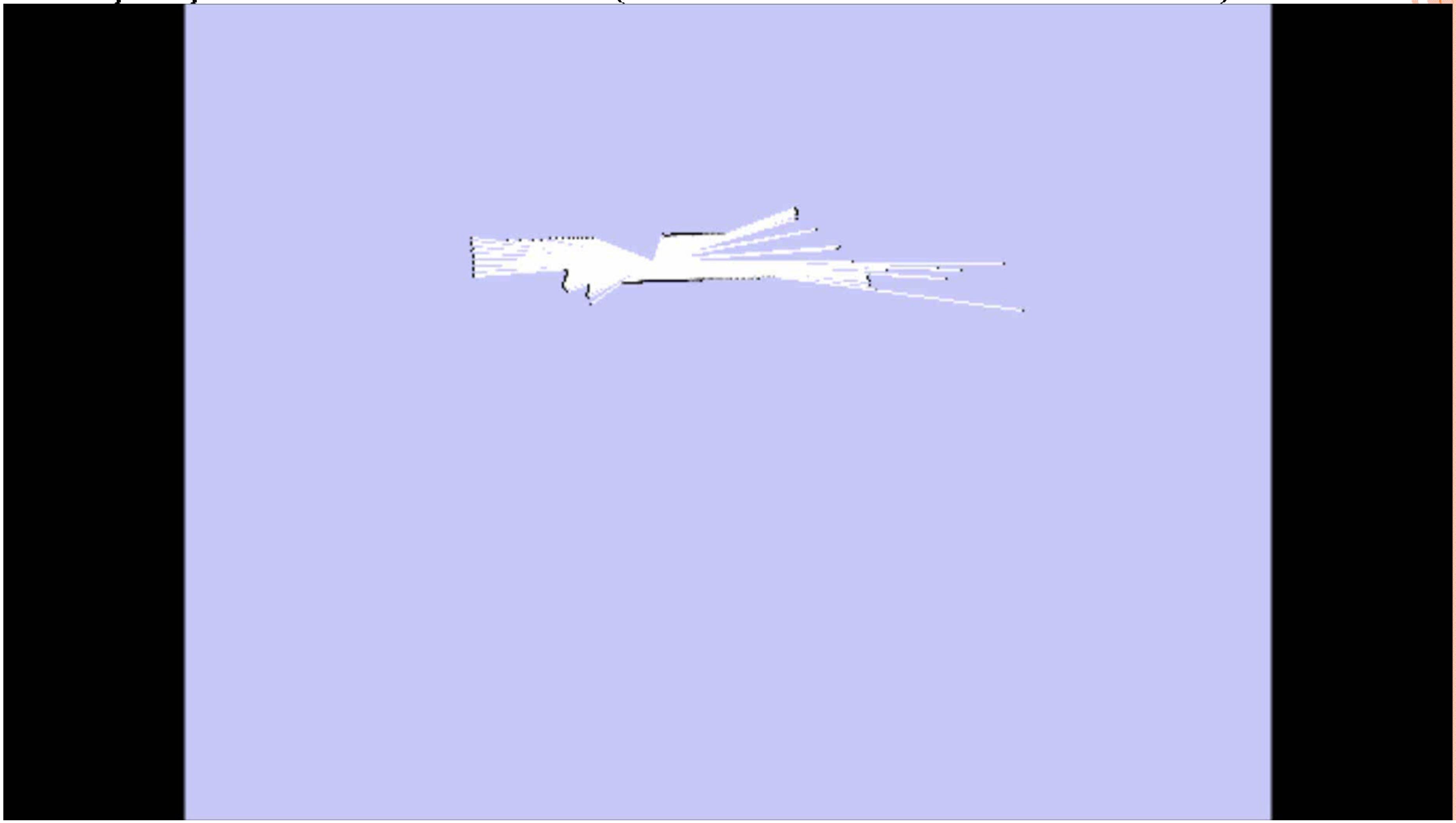
Occupancy Grid SLAM

- Results of Occupancy Grid SLAM with standard motion model



OCCUPANCY GRID SLAM

- Results of Occupancy Grid SLAM with improved proposal distribution (motion and measurement)

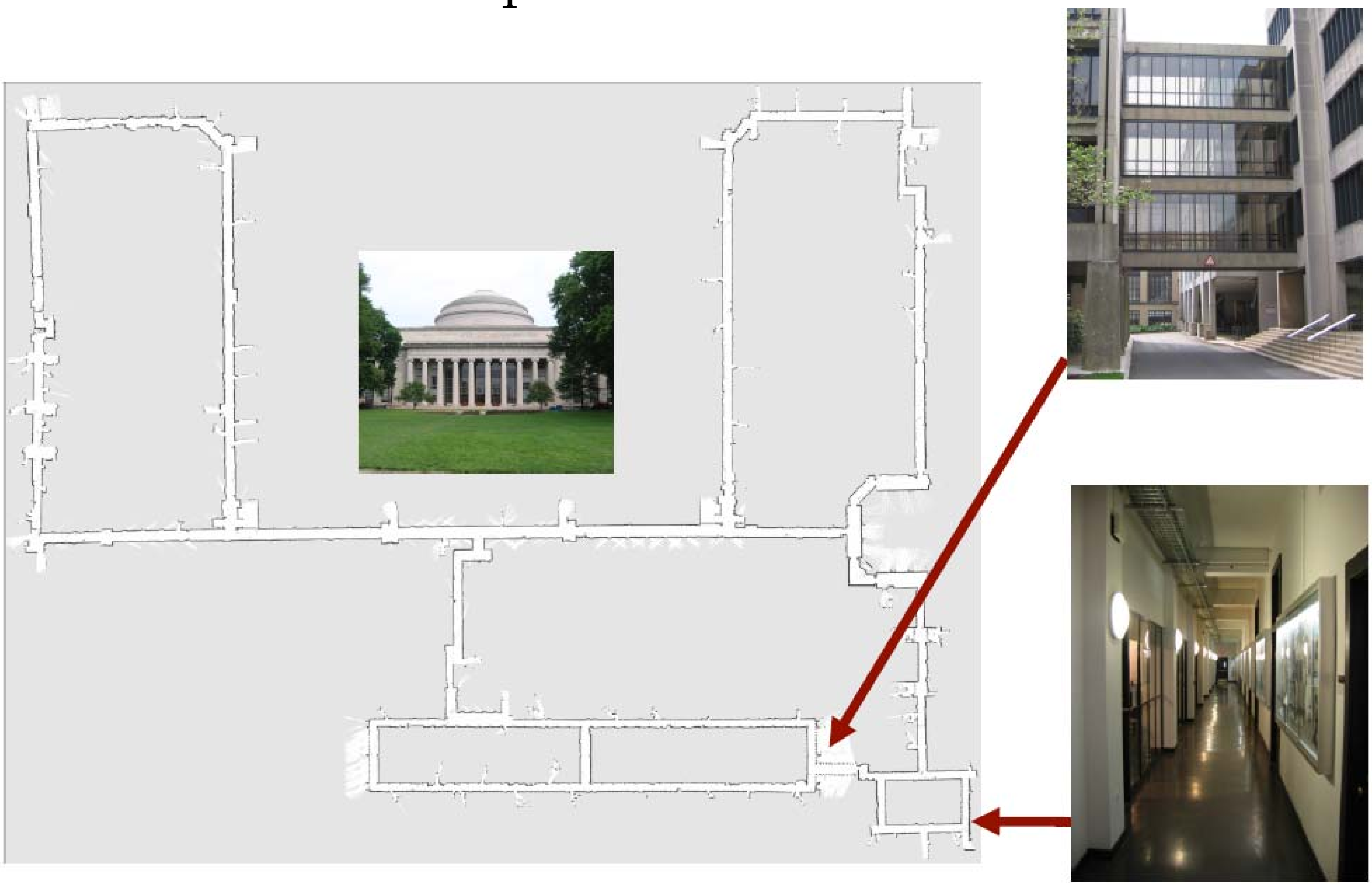


Occupancy Grid SLAM

- Results of Occupancy Grid SLAM with improved proposal distribution

Occupancy Grid SLAM

- MIT Results – 80 particles

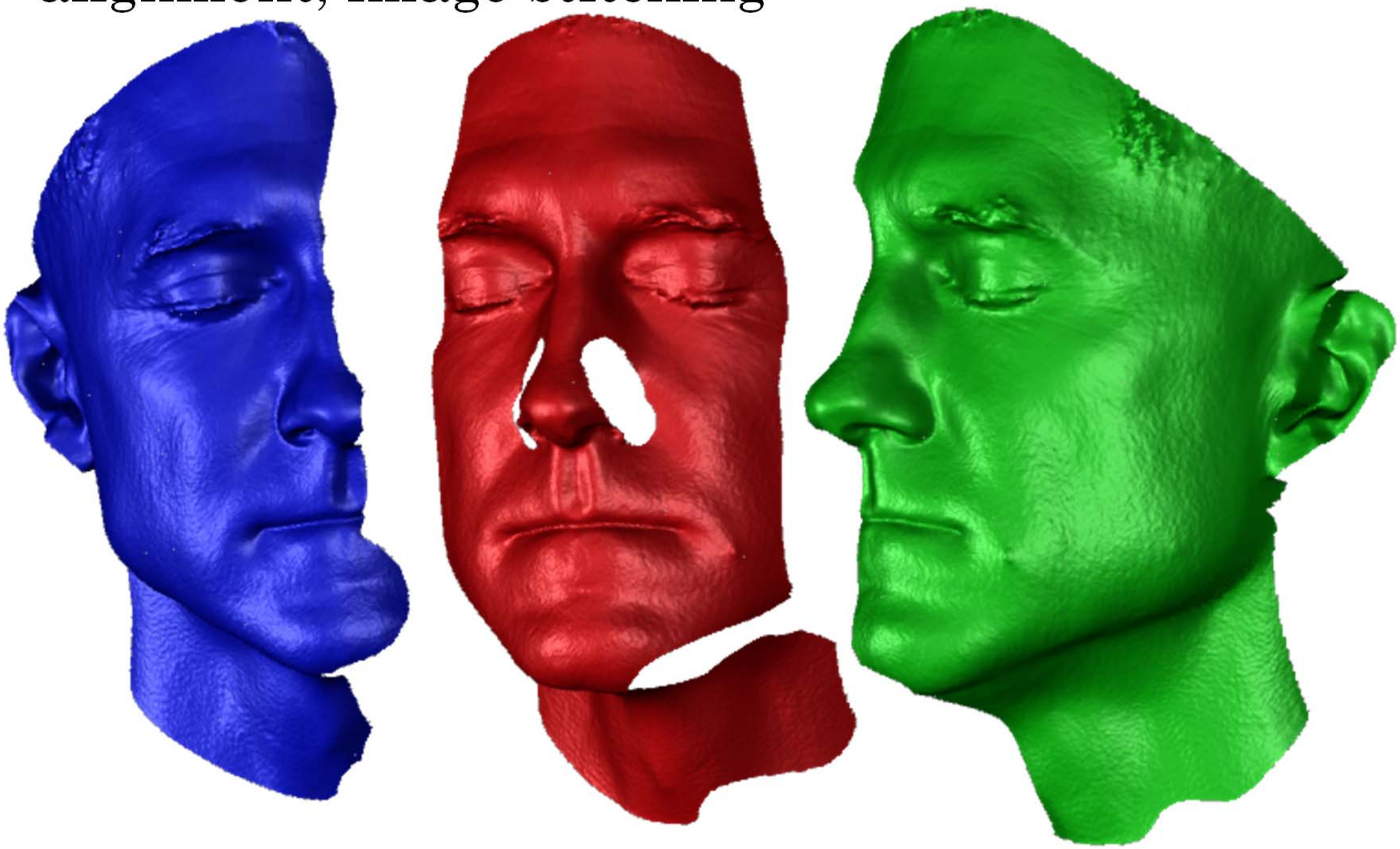


Outline

- Localization
  - EKF
  - Particle
- Mapping
  - Occupancy Grid based
- Simultaneous Localization and Mapping
  - EKF SLAM
  - Particle based FastSLAM
  - Occupancy Grid SLAM
  - Iterated Closest Point Scan Matching
  - Pose Graph Optimization

Scan Registration

- Widely used for 3D modeling, robotics, map alignment, image stitching



Matt Chiang, Jay Busch @ USC Graphics Lab

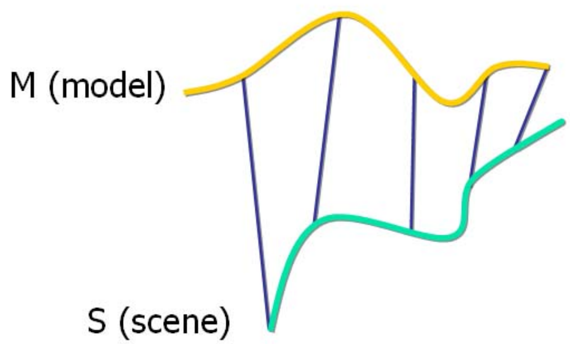
Scan Registration

- Widely used for 3D modeling, robotics, map alignment, image stitching

Matt Chiang, Jay Busch @ USC Graphics Lab

Iterative Closest Point Algorithm

- Let M be a model (reference) point set.
- Let S be a scene (target) point set.



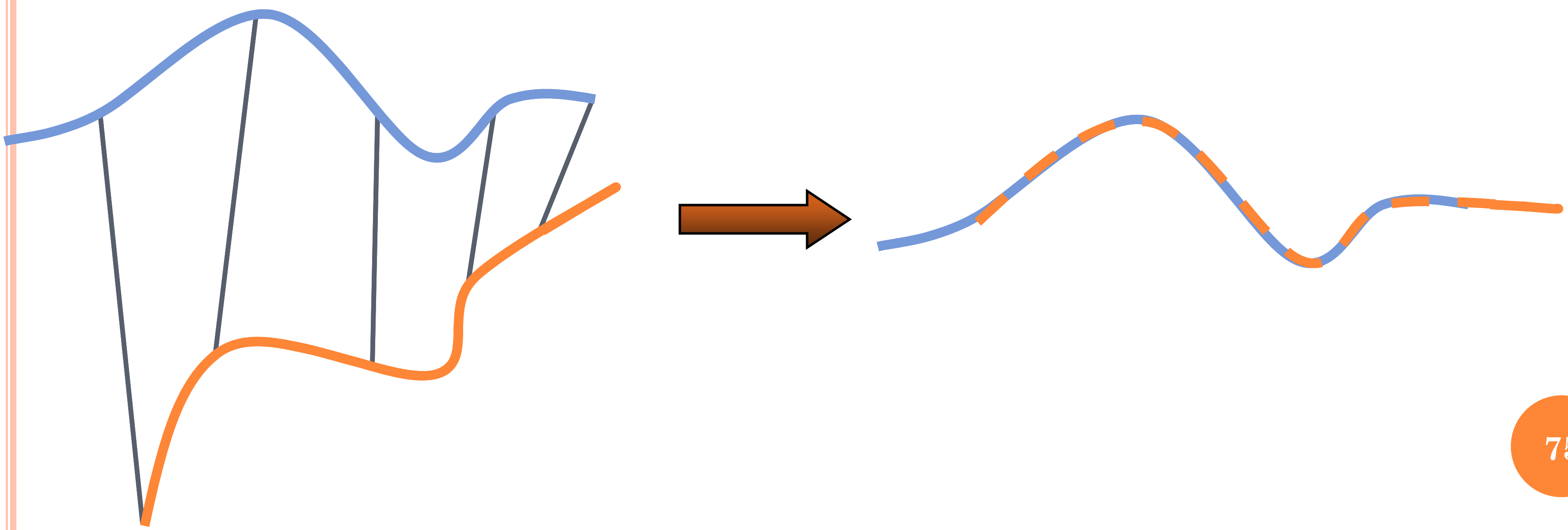
- We assume for now that:
  - $N_M = N_S$.
  - Each point S_i correspond to a point in M_i .

Iterative Closest Point Algorithm

- The transformation between the two scans is represented as a rotation and a translation

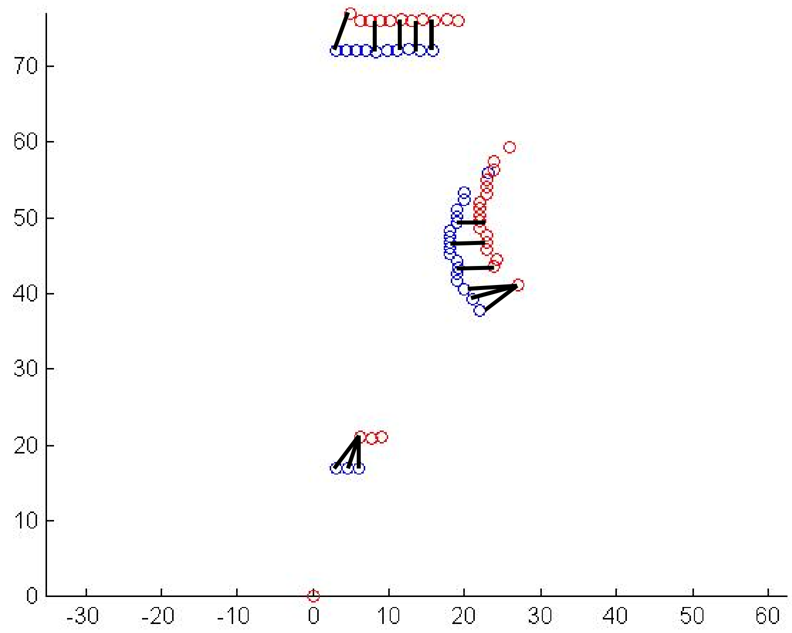
$$T_s^m = \left\{ R_s^m, t_s^m \right\} \qquad m_i = R_s^m s_i + t_s^m$$

- If correct correspondences are known, can find relative rotation/translation that minimizes error



Iterative Closest Point Algorithm

- Given two scans and an initial transformation:
  - Transform scene point set into model frame
  - Find nearest neighbour correspondences
  - Sum quadratic distance error between points
  - Calculate descent direction and improve transformation



ITERATIVE CLOSEST POINT ALGORITHM

- The unconstrained optimization cost function is

$$\min \quad f(R,T) = \frac{1}{N_S}\sum_{i=1}^{N_S}\left\| m_i - R(s_i) - t \right\|^2$$

$$\min \quad f(q) = \frac{1}{N_S}\sum_{i=1}^{N_S}\left\| m_i - R(q_R)s_i - q_T \right\|^2$$

  - Where the optimization variables are parameters that define the rotation and translation
    - Euler angles, quaternions etc.

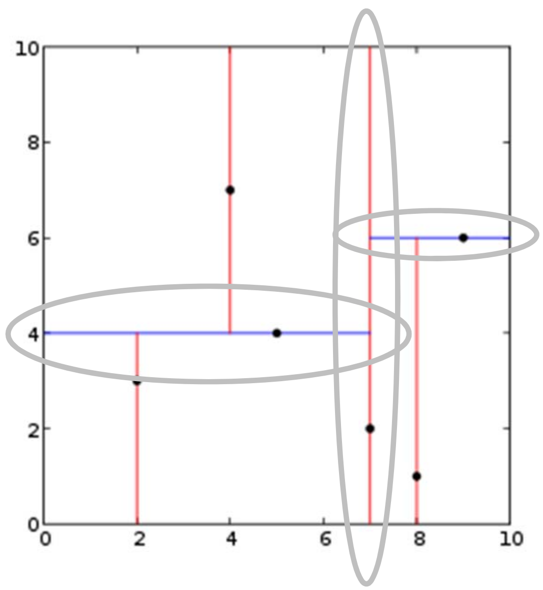
$$q = \begin{bmatrix} q^r & q^t \end{bmatrix} \in \mathbb{R}^6$$

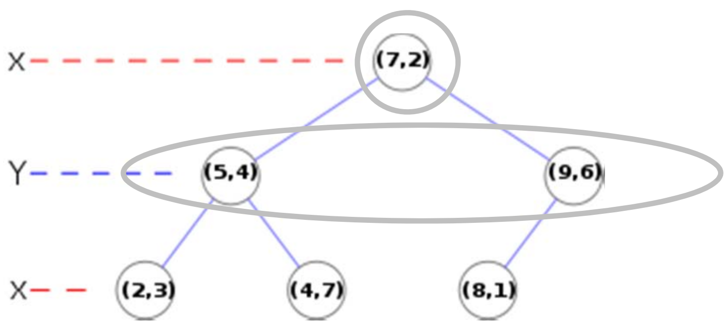
Iterative Closest Point Algorithm

- Most expensive part to compute is nearest neighbour
  - Brute force is O(n^2)
    - Must be repeated each optimization iteration
  - KD-tree is most widely used improvement
    - K-dimensional tree
    - Construction time: O($kn\log(n)$)
    - Space: O(n)
    - Search time: O($\log(n)$)

Iterative Closest Point Algorithm

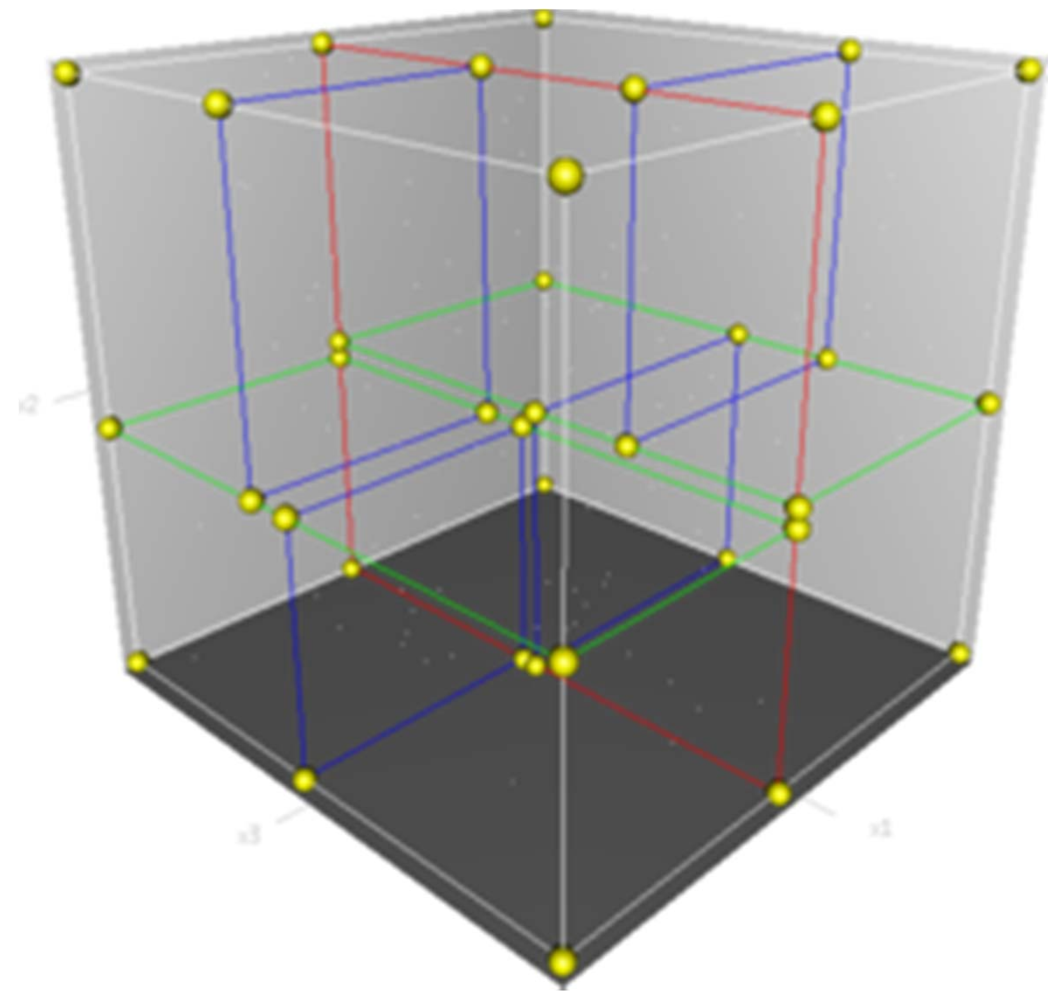
- 2D-Tree construction
  - Median slicing
    - Select axis, find median, divide points around median
    - Repeat for each subsection





ITERATIVE CLOSEST POINT ALGORITHM

- 3D-Tree



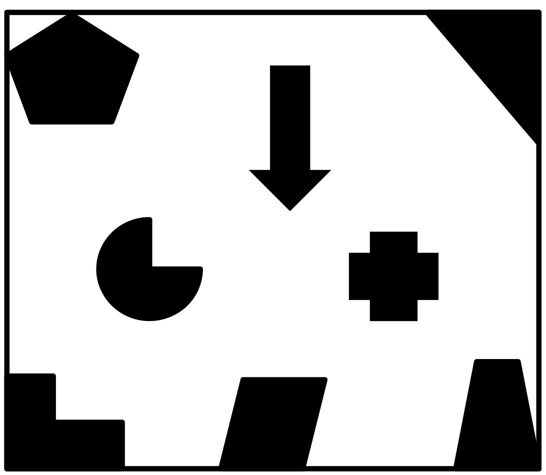
Iterative Closest Point Algorithm

- Can also perform insertion
  - Not needed for ICP

- Nearest neighbour lookup
  - Given a point p
    - Start at root node, proceed left or right down tree, selecting the side that contains the point
    - Once a point is found (leaf of the tree), set as the current best (upper bound on closest distance)
    - Backtrack and check other branches that are not eliminated by branch and bound until nearest neighbour is guaranteed

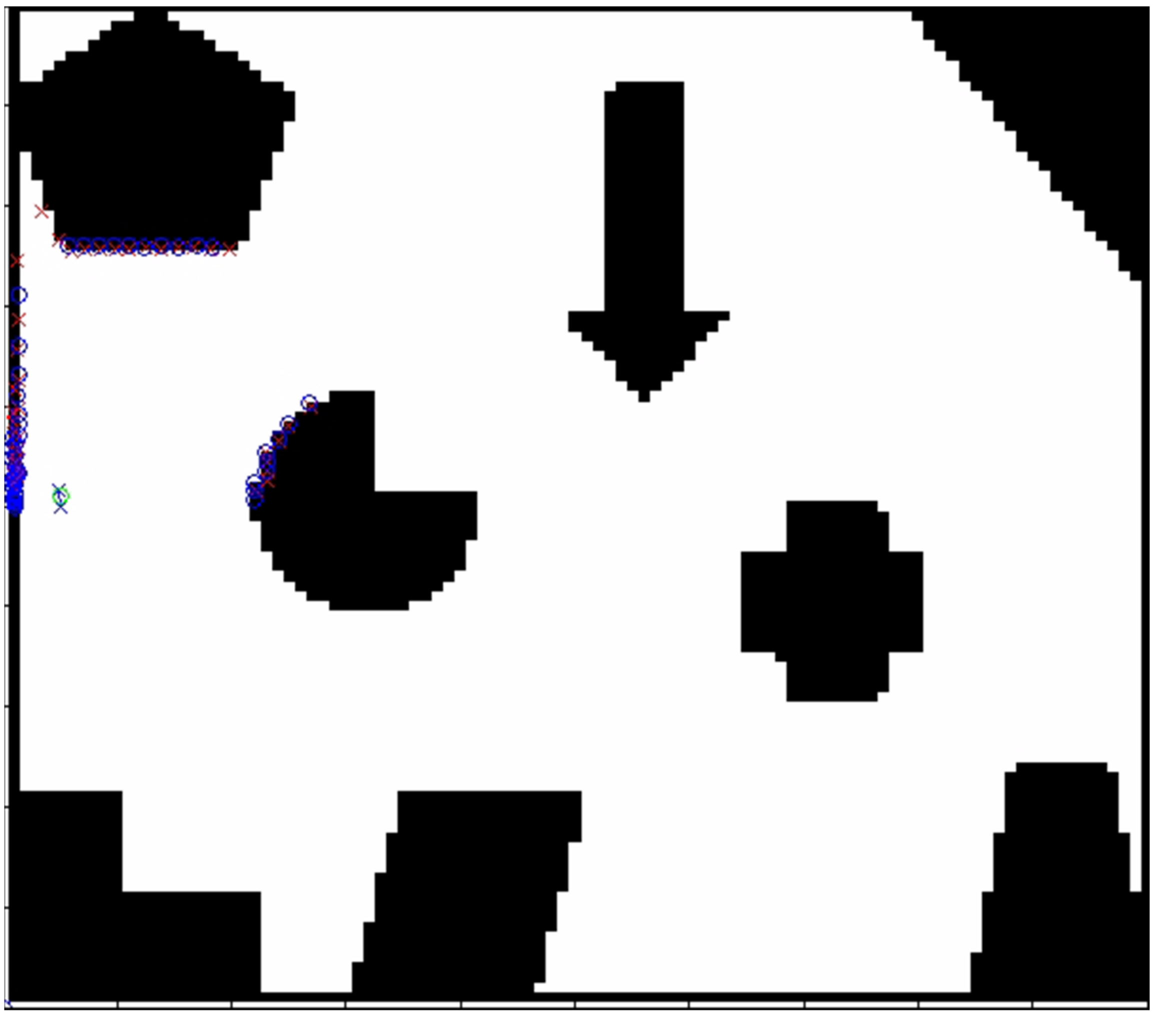
ITERATIVE CLOSEST POINT ALGORITHM

- Matlab Example
  - Uses ICP code from Jakob Wilm and Martin Kjer, Technical University of Denmark, 2012
  - Working on an interesting map
  - Robot drives in a big circle, quantum tunnels through obstacles
  - Scan registration shown relative to true robot pose at t-1.



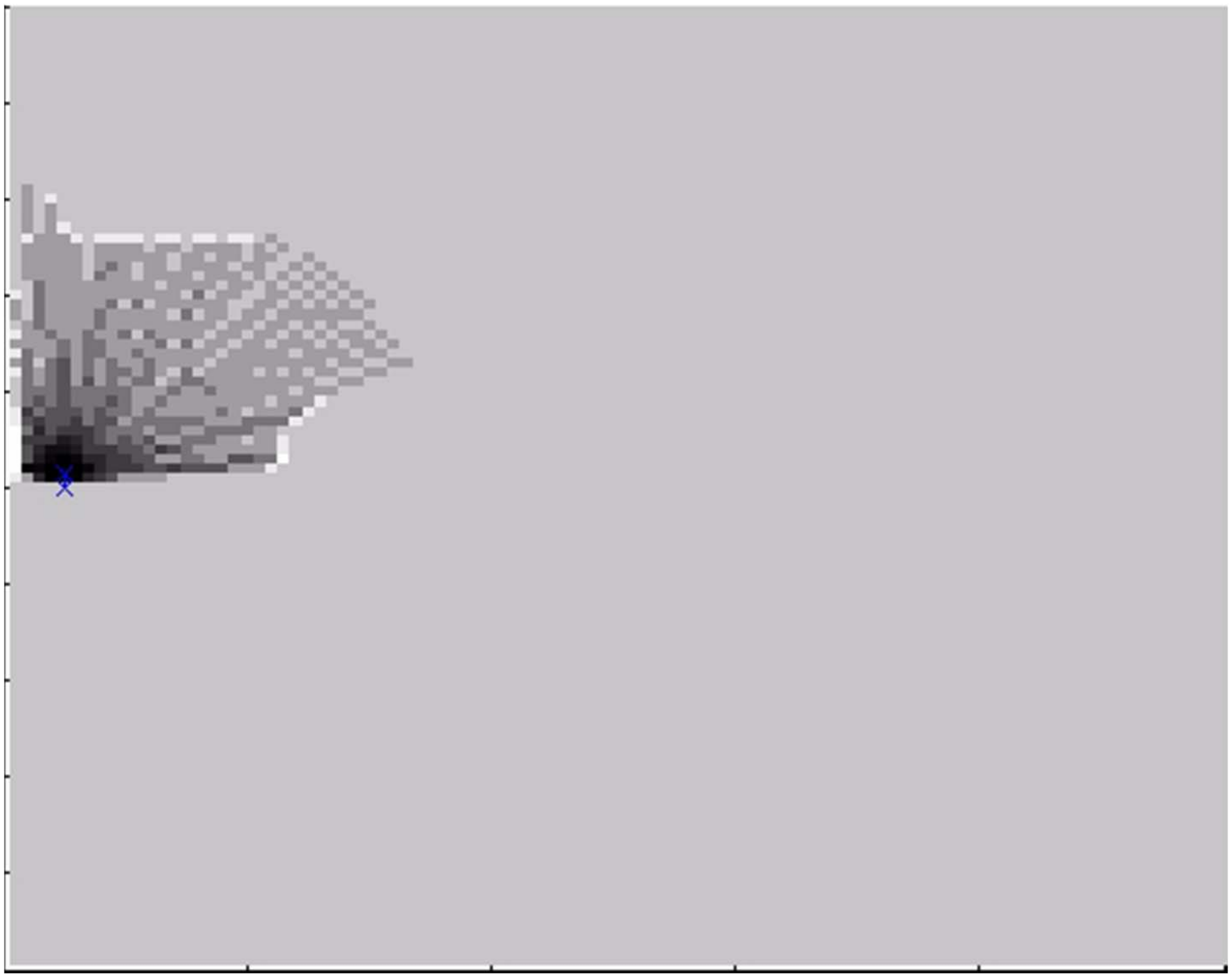
Iterative Closest Point Algorithm

- Scan registration



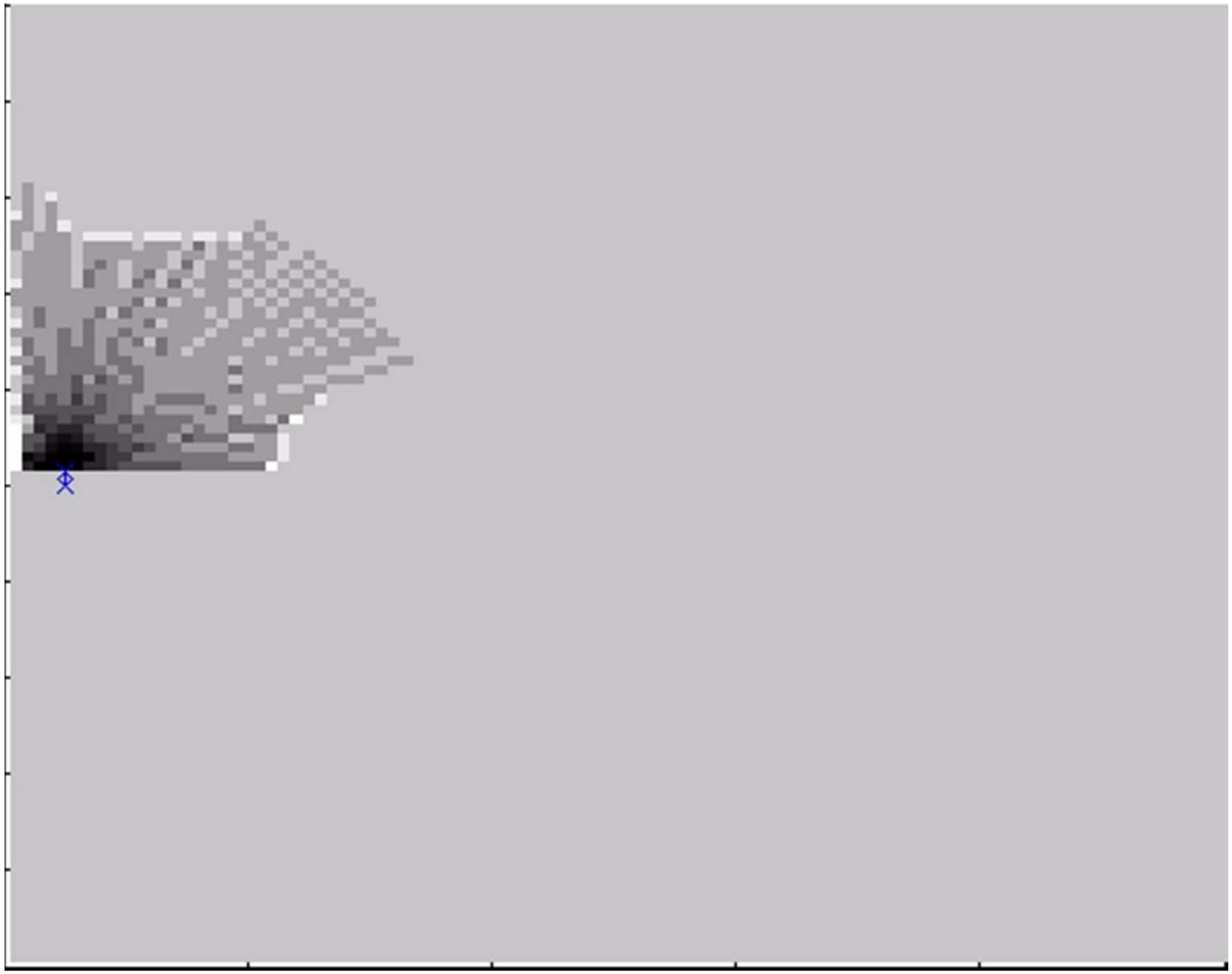
Iterative Closest Point Algorithm

- Resulting Map with scan matching only



Iterative Closest Point Algorithm

- Resulting Map with motion model only



Iterative Closest Point Algorithm

- Updated 2D code for 2014
  - Based on code from Ajmal Saeed Mian at CMU in 2005
    - Simpler, easier to modify
    - Uses singular value decomposition to identify transformation steps
    - More detailed map, more scan points, more accurate registrations
  - Accurate enough to simply accumulate registrations
    - Slowly growing error, with bias
  - Added easy collision avoidance
    - Turn right if something is directly in front of robot

Iterative Closest Point Algorithm

- Updated 2D ICP code for 2014