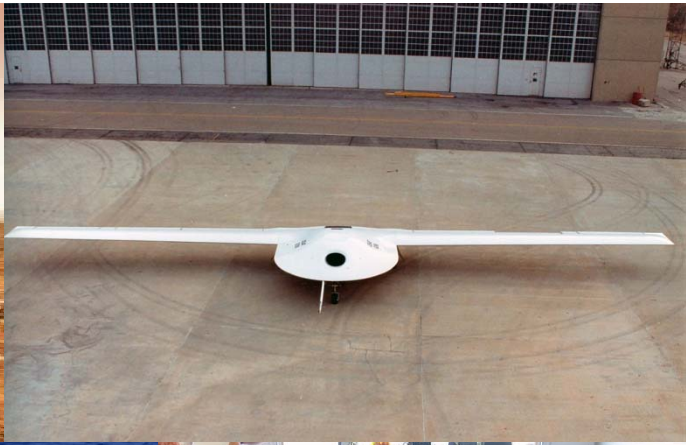
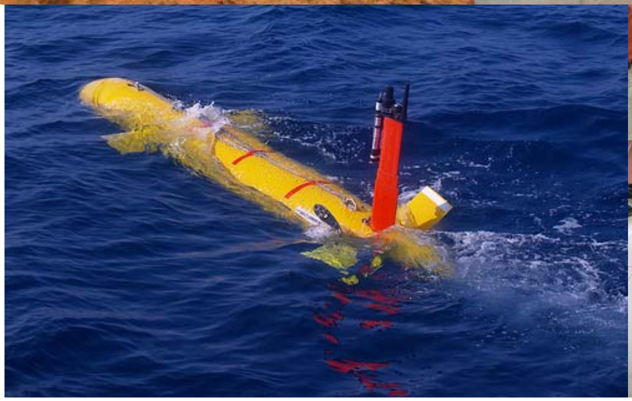
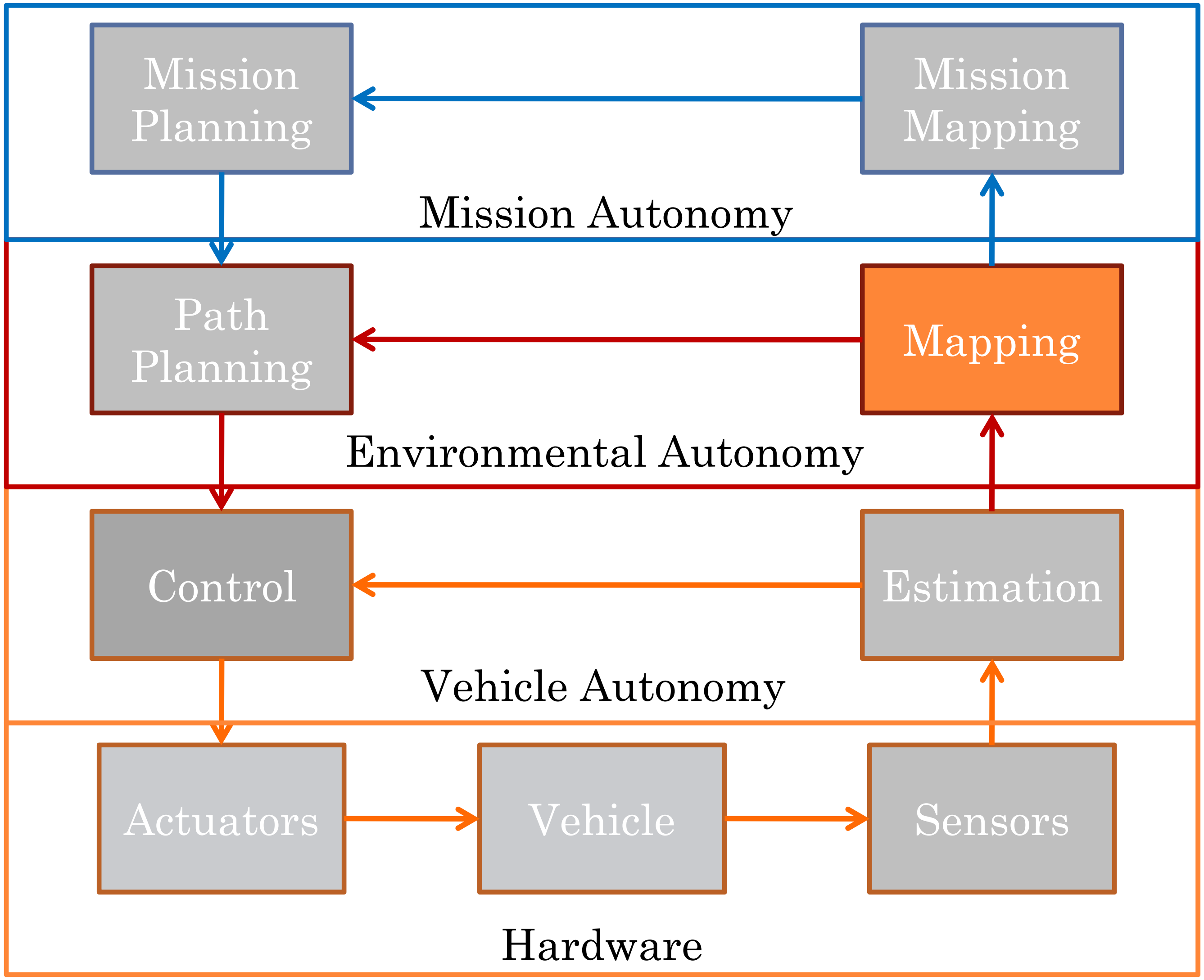




ME 597: Autonomous Mobile Robotics
Section 8 – Mapping I

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 – Determining position relative to known environment
  - EKF
  - Particle
- Mapping – Determining environment relative to known position
  - Feature based (not covered)
  - Occupancy grid based
- Simultaneous Localization and Mapping – unknown position and environment
  - EKF SLAM
  - Particle based FastSLAM
  - Occupancy Grid SLAM
  - Iterated Closest Point Scan Matching
  - Pose Graph Optimization

Localization and Mapping

- Map Types
  - Location based
    - Map is defined by occupancy of each location

$$m = \begin{bmatrix} m^1 & \cdots & m^N \\ \vdots & \ddots & \vdots \\ m^{M-N+1} & \cdots & m^M \end{bmatrix}$$

    - Can be probabilistic in formulation

$$m^i \in [0,1]$$

    - Scales poorly
      - Works well in two dimensions (planar position)

Localization and Mapping

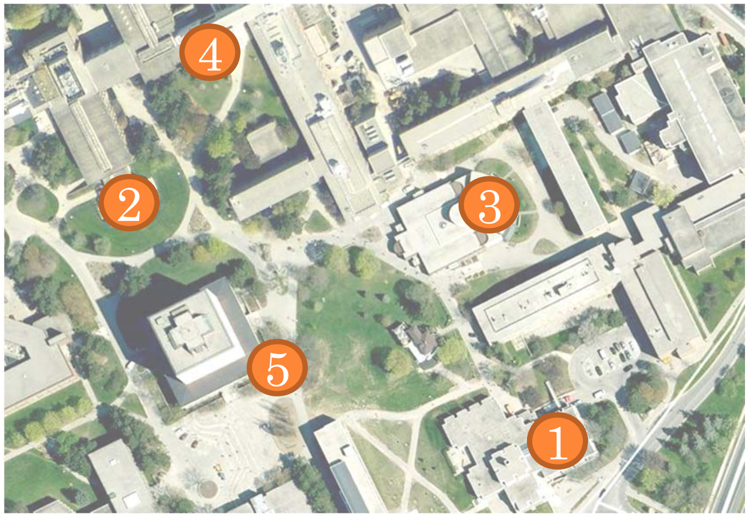
- Map Types
  - Feature Based
    - A feature is defined at a specific location, and may have a signature
    - The set of all features defines the map
    - Effective for localization
    - Scales well to larger dimensions
    - Hard to use for collision avoidance

$$m^i = \{x^i, y^i, s^i\}$$

$$m^i = \{r^i, \theta^i, s^i\}$$

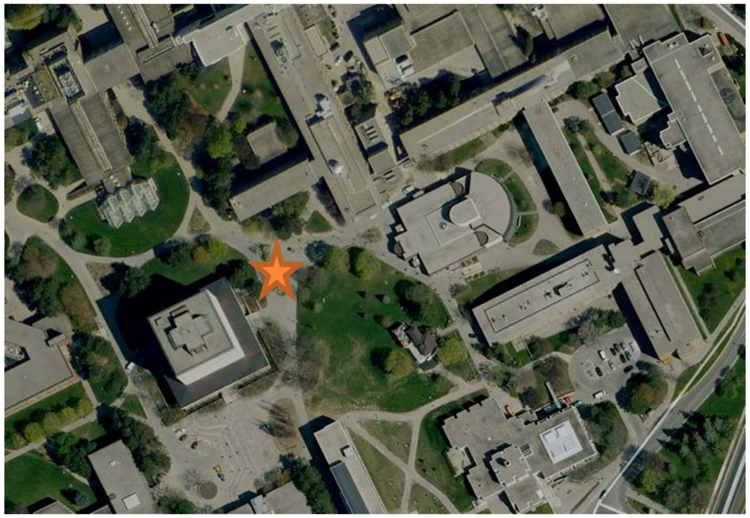
$$\vdots$$

$$m = \{m^1, \ldots, m^M\}$$



Localization

- Localization
  - Using sensor information to locate the vehicle in a known environment
  - Given:
    - Control inputs and motion model
    - Sensor measurements and measurement model relative to environment
    - Environment model
  - Find:
    - Vehicle position



Localization

- Localization Problems

  - Initial conditions
    - Local: Known initial position
      - Tracking position through motions with inputs and measurements
    - Global: Unknown initial position
      - Finding position and then continuing to track
    - Kidnapped: Incorrect initial position
      - Correcting incorrect prior beliefs to recover true position and motion

Localization

- Assumptions
  - Known static environment
    - No moving obstacles, or other vehicles that cannot be removed from sensor measurements
  - Passive Estimation
    - Control law does not seek to minimize estimation error
  - Single vehicle
    - Only one measurement location is available

- Each assumption can be addressed through more complex algorithms
  - Good starting points available in Thrun et al.

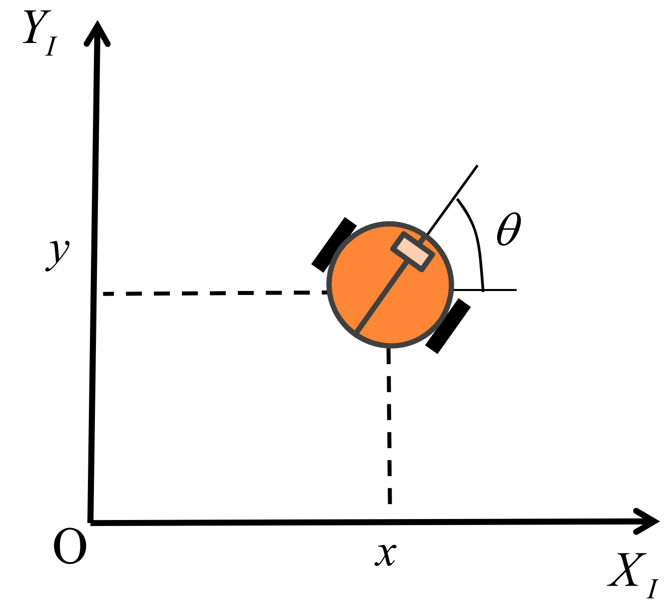
LOCALIZATION

- Feature-based localization
  - Most natural formulation of localization problem
    - Sensors measure bearing, range, relative position of features
    - Location based maps can be reduced to a set of measurable features
    - The more features tracked the better the solution
      - But the larger the matrix inverse at each timestep

Localization

- Example: Two-wheeled robot
  - Vehicle State, Inputs:

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



  - Motion Model

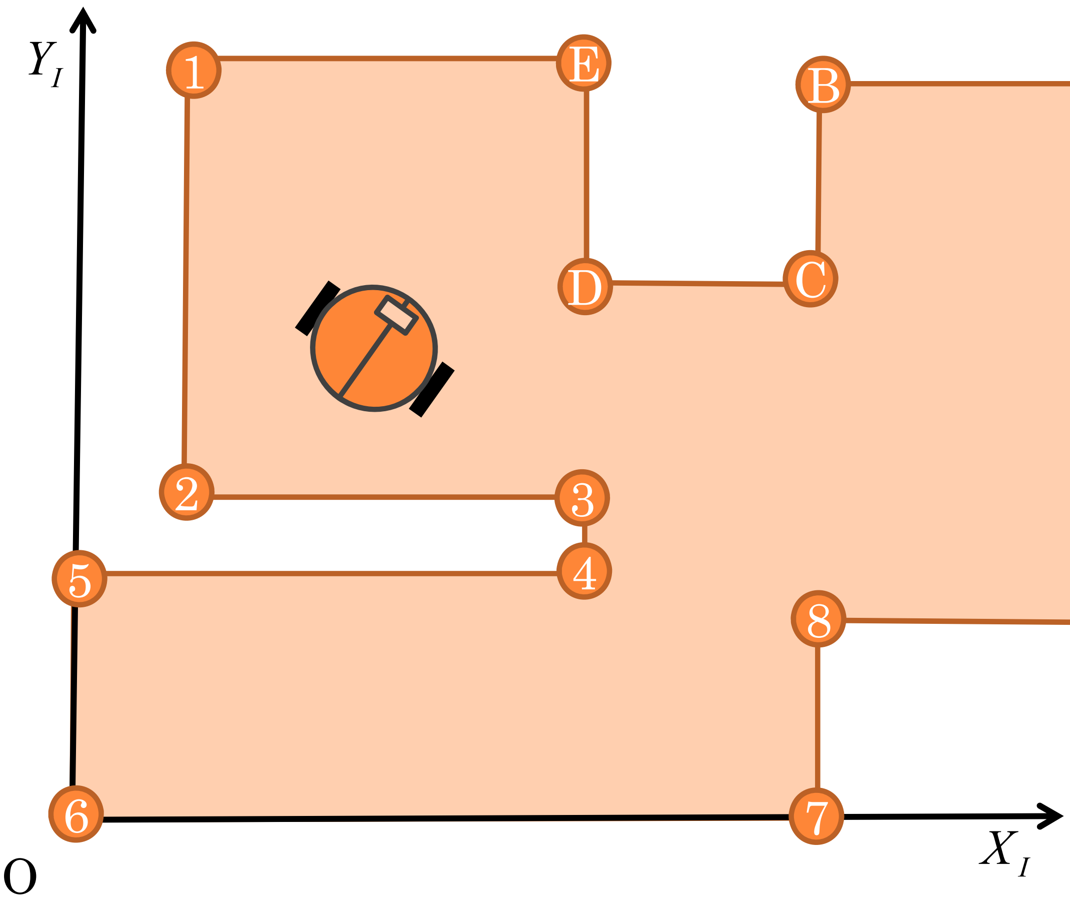
$$\begin{bmatrix} x_{1,t} \\ x_{2,t} \\ x_{3,t} \end{bmatrix} = g(x_{t-1}, u_t) = \begin{bmatrix} x_{1,t-1} + u_{1,t} \cos x_{3,t-1} dt \\ x_{2,t-1} + u_{1,t} \sin x_{3,t-1} dt \\ x_{3,t-1} + u_{2,t} dt \end{bmatrix}$$

LOCALIZATION

- Example: Feature Map

$$m = \{m^1, \ldots, m^M\}, \quad m^i = \{m_x^i, m_y^i\}$$

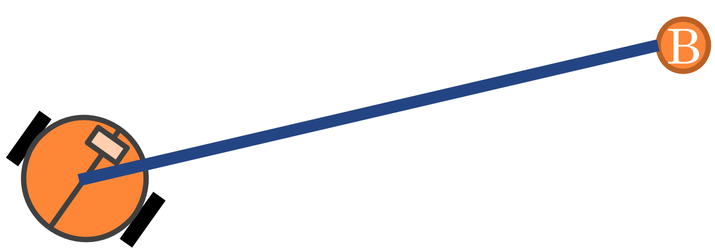
  - Assume all features are uniquely identifiable



Localization

- Example: Measurement Model
  - Relative range and/or bearing to closest feature m^i, regardless of heading
  - Assume measurement of closest feature only

$$\begin{bmatrix} y_{1,t} \\ y_{2,t} \end{bmatrix} = h(x_t) = \begin{bmatrix} \tan^{-1}\left(\dfrac{m_y^i - x_{2,t}}{m_x^i - x_{1,t}}\right) - x_{3,t} \\ \sqrt{\left(m_x^i - x_{1,t}\right)^2 + \left(m_y^i - x_{2,t}\right)^2} \end{bmatrix} \begin{matrix} \text{Bearing} \\ \\ \text{Range} \end{matrix}$$



Localization

- We'll try localization with two approaches
  - EKF (UKF) based localization
    - Fast computationally
    - Intuitive formulation
    - Most frequently implemented
    - Possibility for divergence if nonlinearities are severe
    - Additive Gaussian noise

$$\varepsilon_t \sim N(0, R_t) \qquad \delta_t \sim N(0, Q_t)$$

  - Particle filter based localization
    - Slightly cooler visualizations
    - More expensive computationally
    - More capable of handling extreme nonlinearities, constraints, discontinuities

Localization

- Recall Extended Kalman Filter Algorithm

1. Prediction Update

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

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

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

2. Measurement Update

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

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

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

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

Localization

- Linearization of Motion Model

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

$$\Downarrow$$

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

Localization

- Linearization of Measurement Model

$$\begin{bmatrix} y_{1,t} \\ y_{2,t} \end{bmatrix} = h(x_t) = \begin{bmatrix} \tan^{-1}\left(\dfrac{m_y^i - x_{2,t}}{m_x^i - x_{1,t}}\right) - x_{3,t} \\ \sqrt{\left(m_x^i - x_{1,t}\right)^2 + \left(m_y^i - x_{2,t}\right)^2} \end{bmatrix}$$

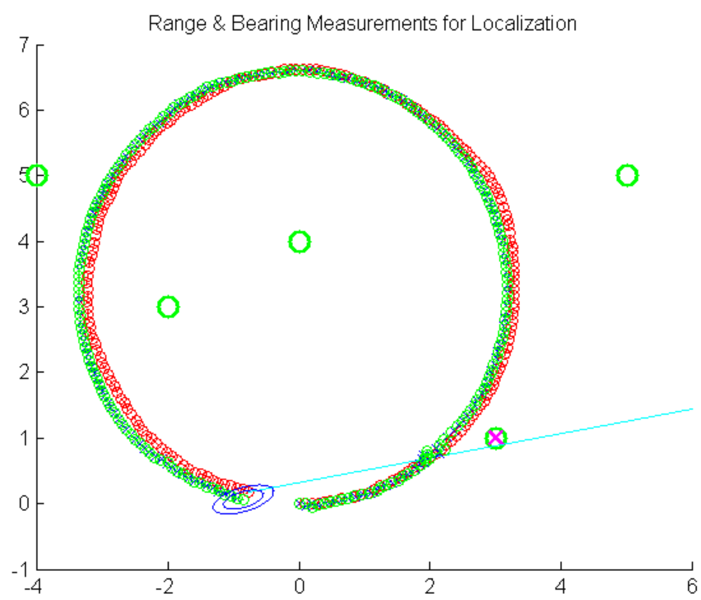
$$\Downarrow$$

$$\frac{\partial}{\partial x_t} h(x_t) = \begin{bmatrix} \dfrac{\left(m_y^i - x_{2,t}\right)}{q} & \dfrac{-\left(m_x^i - x_{1,t}\right)}{q} & -1 \\ -\dfrac{\left(m_x^i - x_{1,t}\right)}{\sqrt{q}} & -\dfrac{\left(m_y^i - x_{2,t}\right)}{\sqrt{q}} & 0 \end{bmatrix}$$

- where $q = \left(m_x^i - x_{1,t}\right)^2 + \left(m_y^i - x_{2,t}\right)^2$

EKF Localization - Simulation

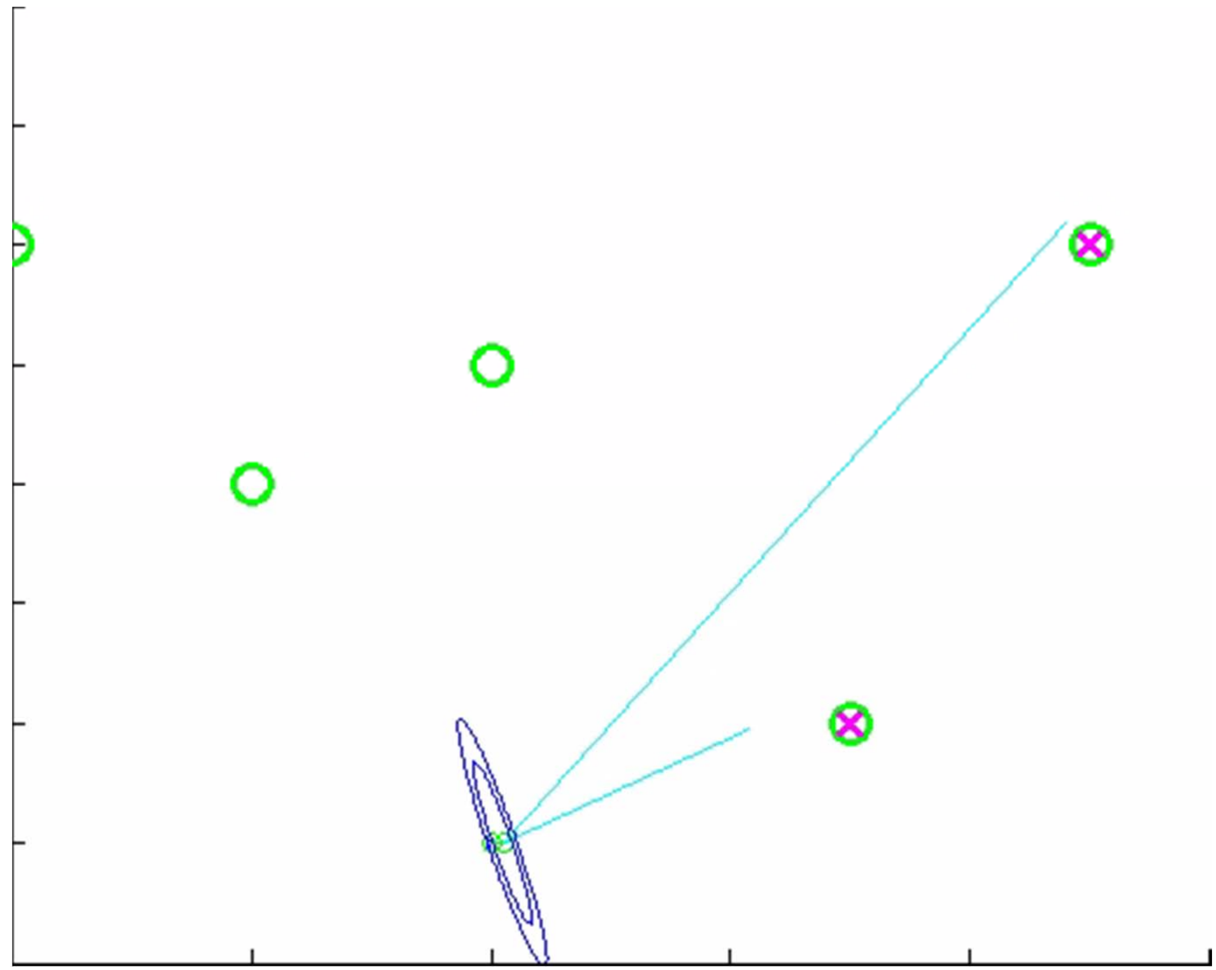
- Five features in a 2D world
- No confusion over which is which
  - Correct correspondence
- Two wheeled robot (x, y, θ)
- Measurement to feature of Range, bearing, both



LOCALIZATION

True state -o-
Belief -x-
Predicted Belief -o-
Measurement ___
Features O
Current Feature X

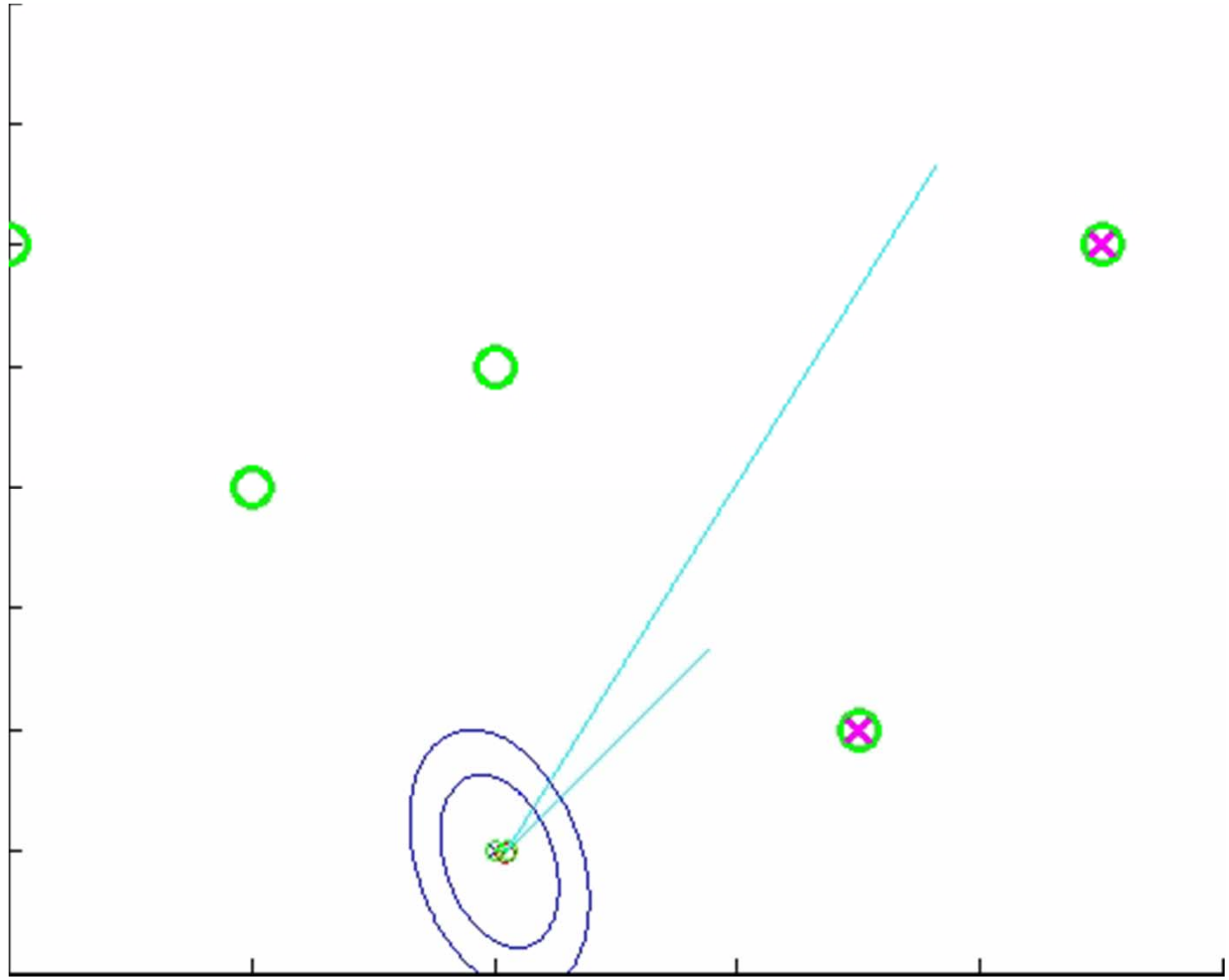
- Example
  - Both measurements, very low noise, correct prior



LOCALIZATION

True state -o-
Belief -x-
Predicted Belief -o-
Measurement ___
Features O
Current Feature X

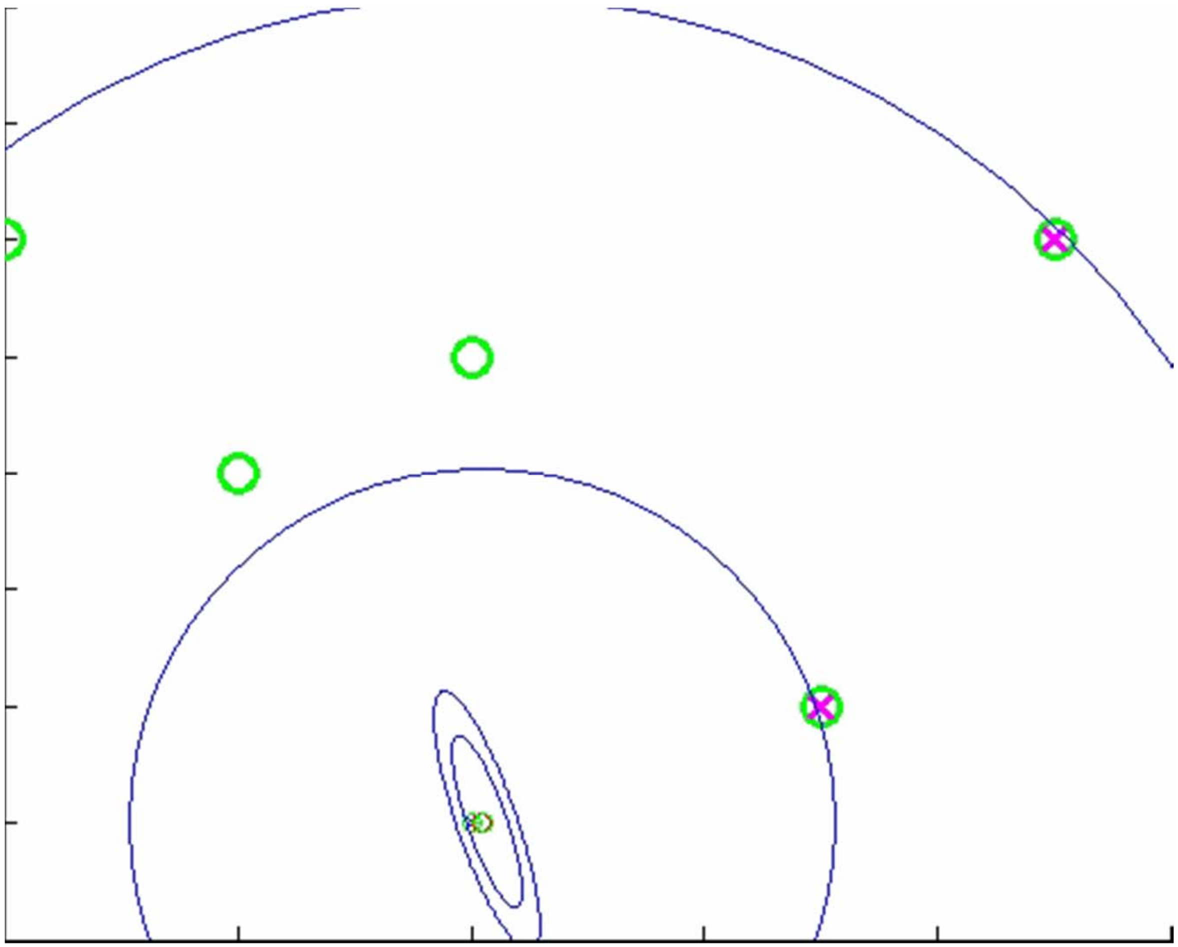
- Example with moderate noise
  - Both measurements noisy, correct prior, large disturbances



LOCALIZATION

True state -o-
Belief -x-
Predicted Belief -o-
Measurement ___
Features O
Current Feature X

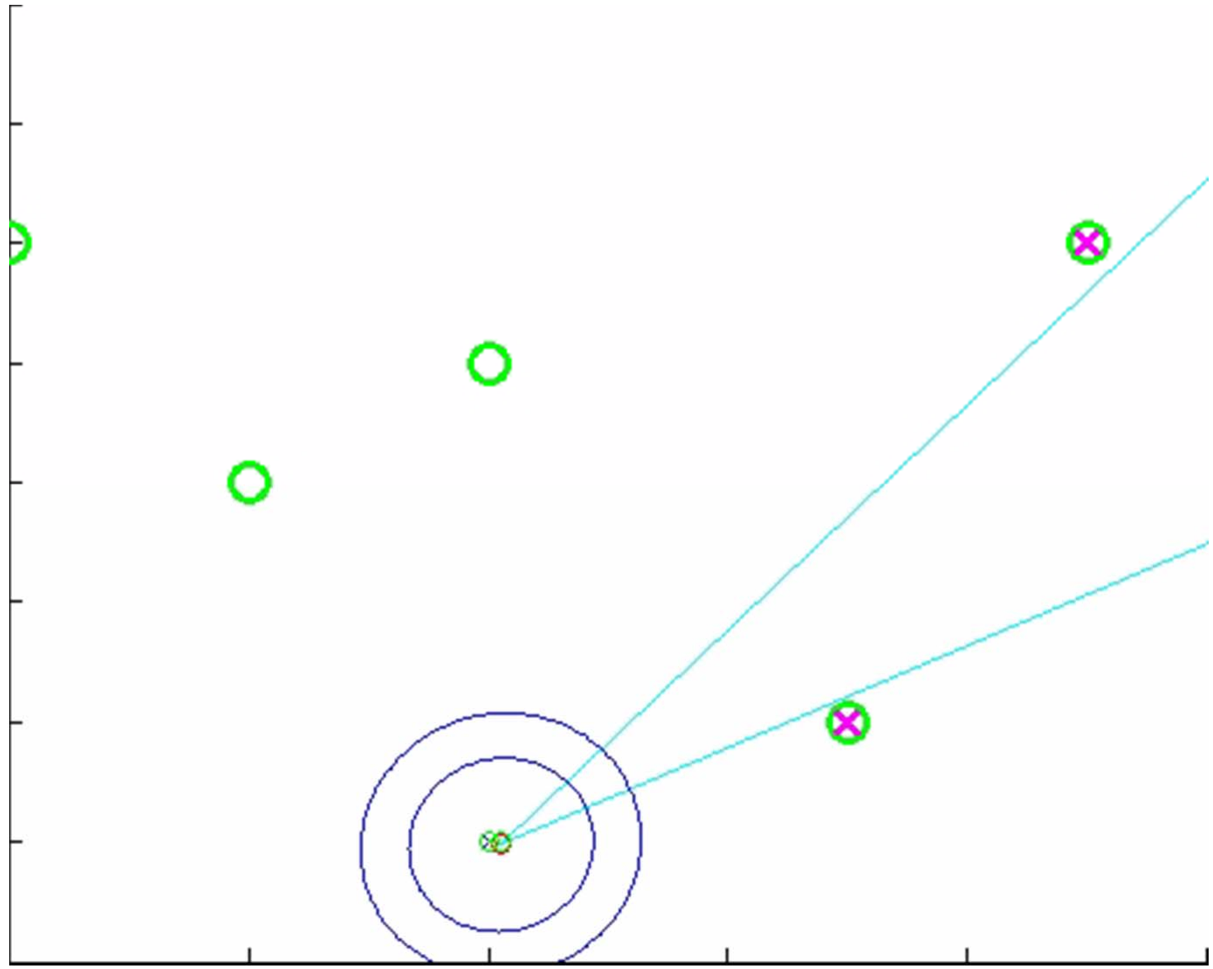
- Example with moderate noise
  - Range only, correct prior



LOCALIZATION

- Example with moderate noise
  - Bearing only, correct prior

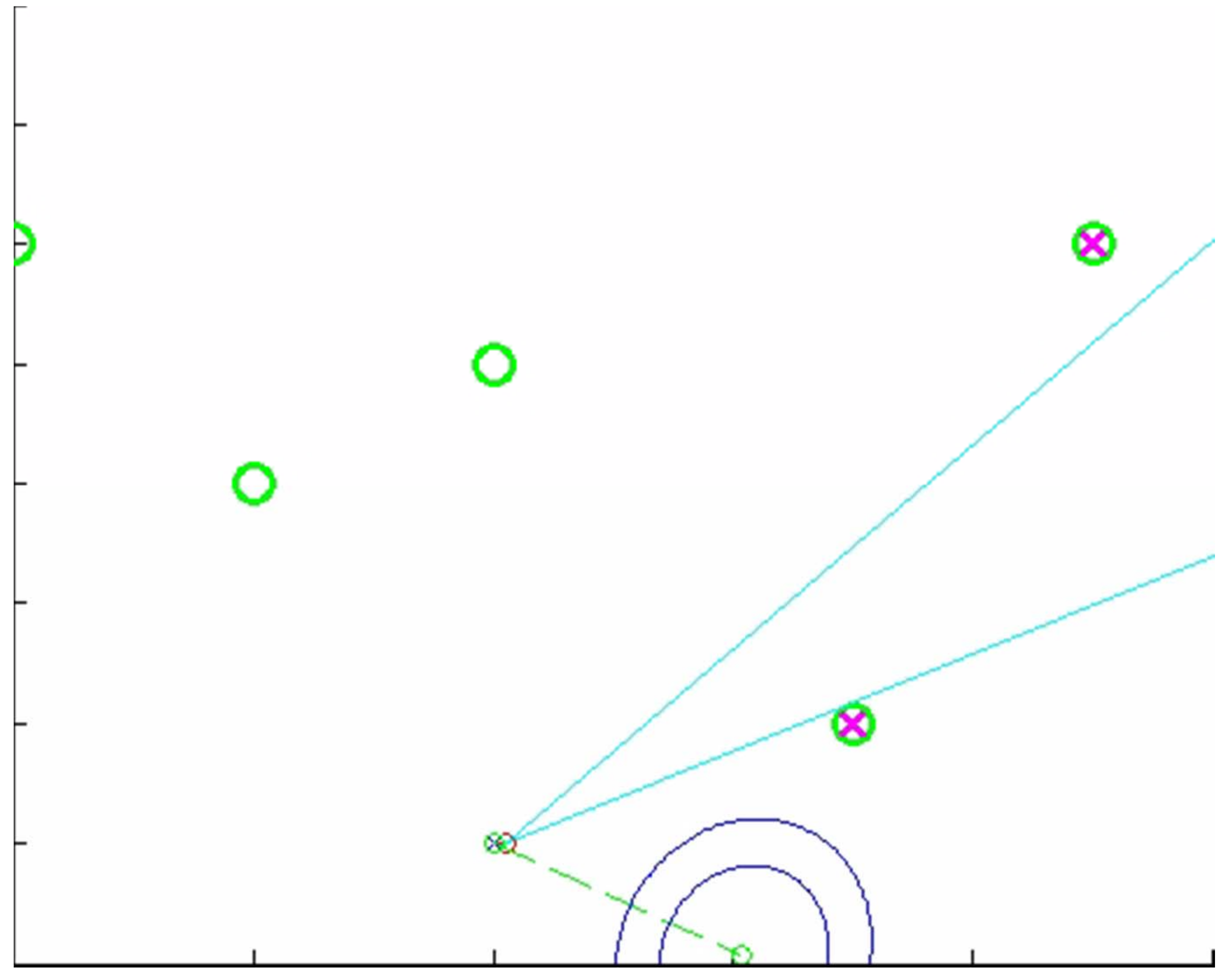
True state -o-
Belief -x-
Predicted Belief -o-
Measurement ___
Features O
Current Feature X



LOCALIZATION

True state -o-
Belief -x-
Predicted Belief -o-
Measurement ___
Features O
Current Feature X

- Example with moderate noise
  - Bearing only, incorrect prior of [2 -1 pi/4]



LOCALIZATION

- Particle Filter implementation
  - All the components are defined above
    - Same prior
    - Same motion model
    - Same measurement model
    - Standard particle filter implementation

Particle Filters

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

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

     2. Calculate weight (importance)

        $$w_t^{[i]} = p(y_t \mid x_t^{[i]})$$

     3. Store in interim particle set

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

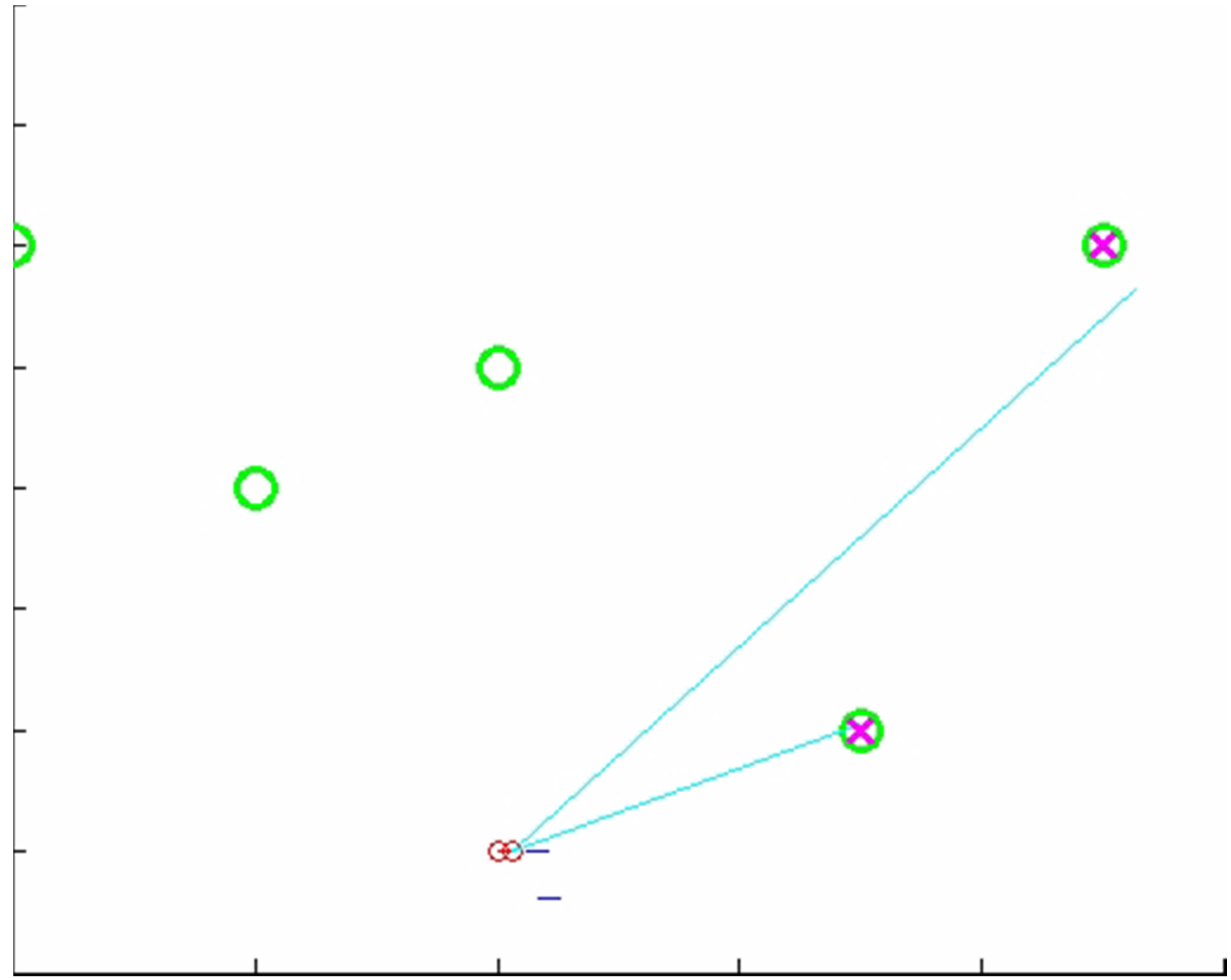
  2. For j = 1 to D
     1. Draw index i with probability $\propto w_t^{[i]}$ (resampling)
        1. Add to final particle set

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

Localization

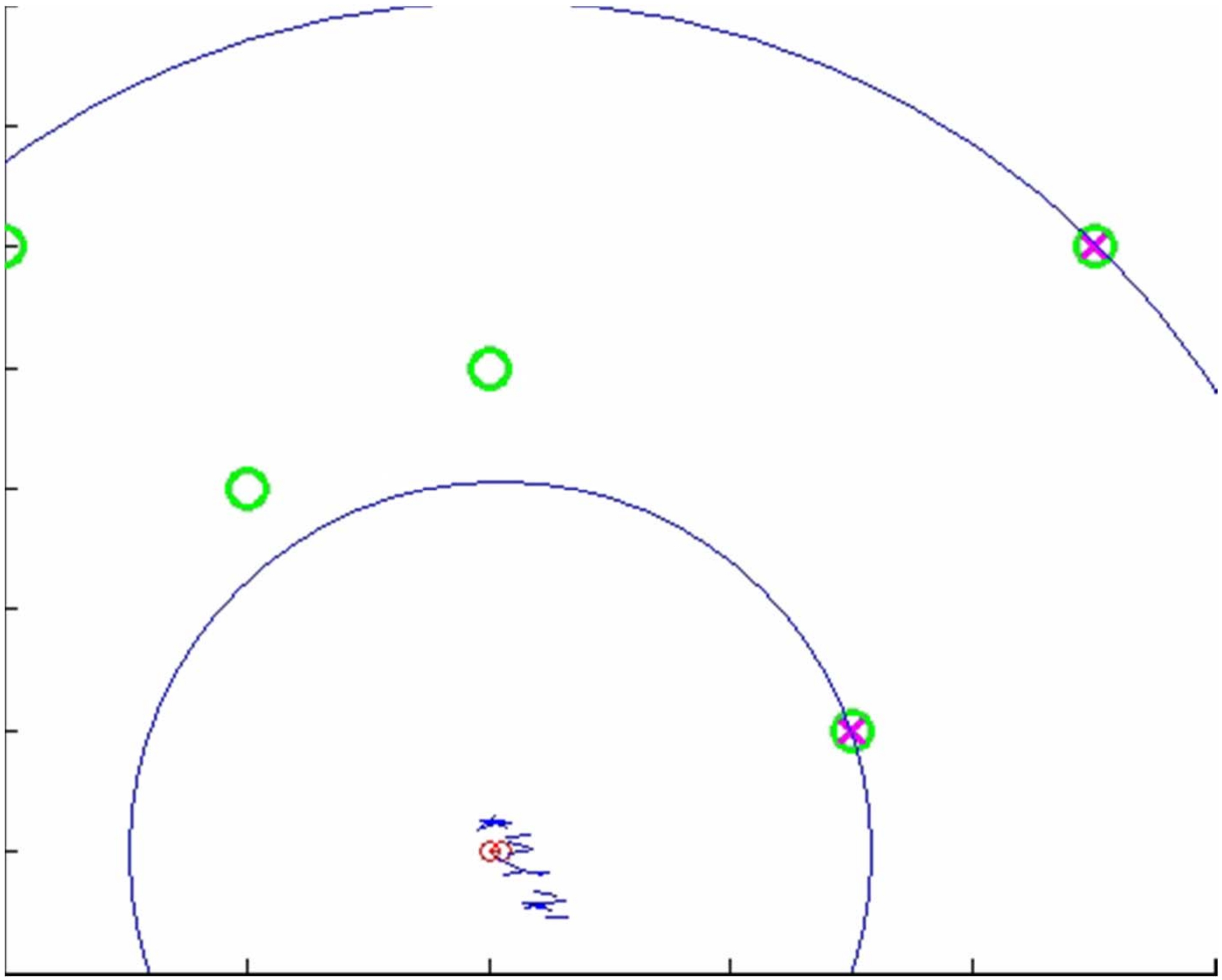
True state -o-
Particles .
Measurement ___
Features O
Current Feature X

- Particle Filter results
  - Range & bearing measurements with 500 particles



LOCALIZATION

- Particle Filter results
  - Range only with 500 particles

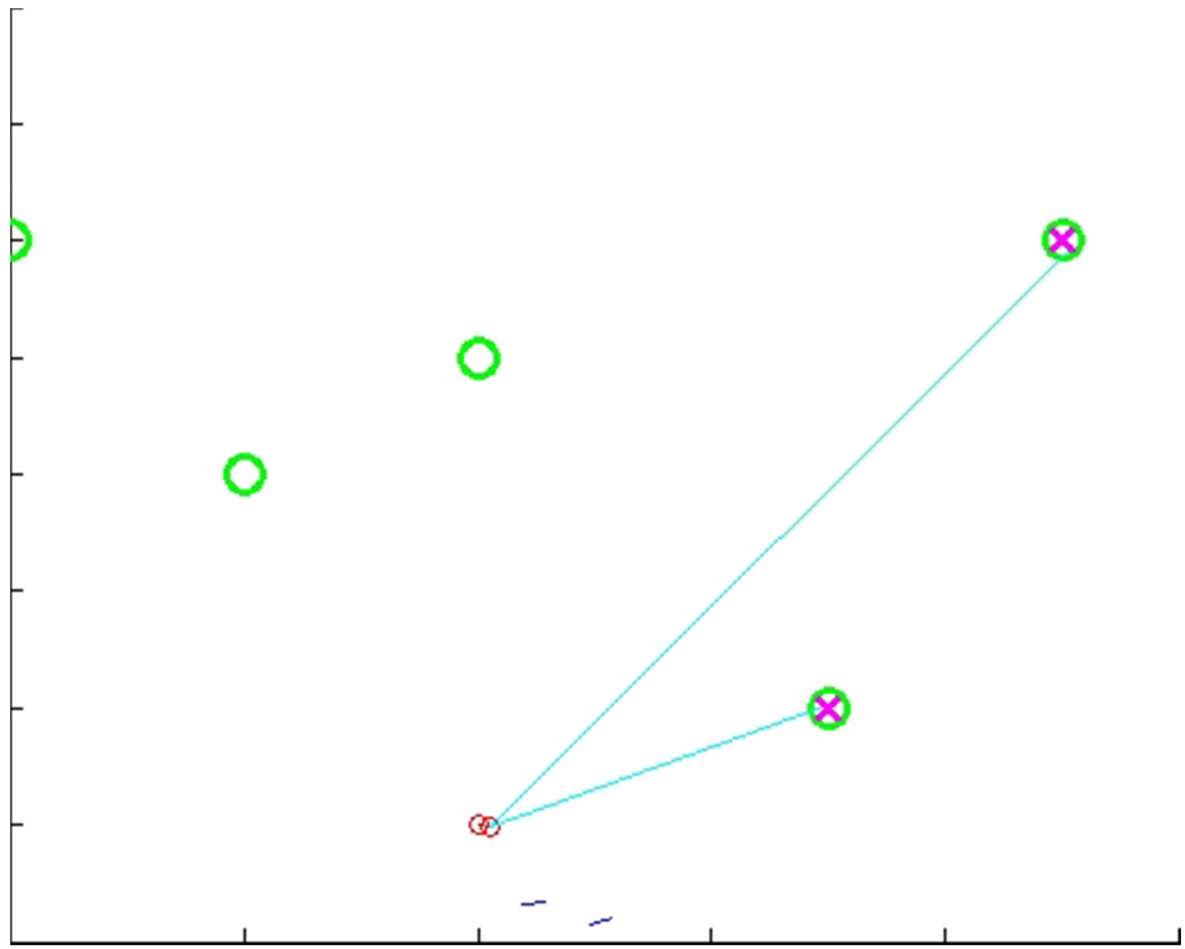


True state	-o-
Particles	.
Measurement	___
Features	O
Current Feature	X

LOCALIZATION

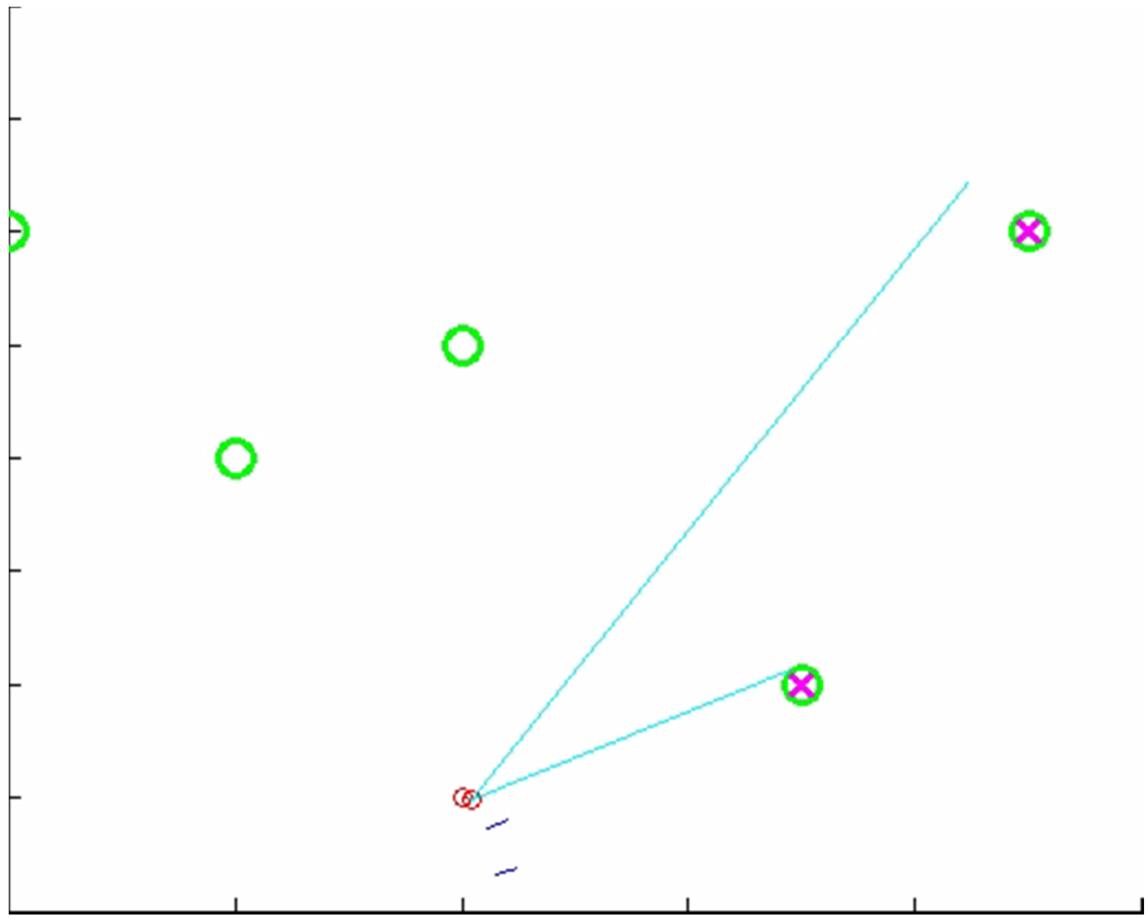
True state -o-
Particles .
Measurement ___
Features O
Current Feature X

- Particle Filter results
  - Range & bearing with 100 particles, poor prior



LOCALIZATION

- Particle Filter results
  - Range & bearing with 100 particles, poor prior, large disturbance



Localization

- Feature Based Localization
  - Unknown Correspondences
    - It may not be obvious from measurements which feature has been measured
    - A major issue with all real world implementations
      - Popularity of SIFT/SURF features arises from uniqueness of signature
      - Corners, edges, color blobs etc. not easy to distinguish
      - Maximum Likelihood correspondence
      - Augmented with geometric configuration of matches
      - Random Sample Consensus

Localization

- Unknown Correspondence
  - Maximum Likelihood Correspondence
    - Find the most likely feature a measurement corresponds to based on state and measurement info

$$c_t^* = \arg\max_{c_t} p(y_t \mid c_{1:t}, m, y_{1:t-1}, u_{1:t})$$

    - Works poorly if many features are equally likely
    - Integer optimization
      - Exponential complexity growth in the number of variables
      - Often avoided by doing correspondence for each measurement independently

$$c_{t,i}^* = \arg\max_{c_{t,i}} p(y_t \mid c_{1:t,i}, m, y_{1:t-1}, u_{1:t})$$

        - Suboptimal, could get multiple distinct measurements assigned to the same feature

Localization

- Random Sample Consensus (RANSAC)
  - While not out of time
    - Pick a small subset of measurement correspondences

$$y^k \subset y_t$$

    - Perform temporary measurement update with this subset

$$\mu^k = EKF(\bar{\mu}_t, y^k)$$

    - Find all features that agree with current estimate to within a fixed threshold (identify inlier set)

$$I^k = \left\{ y^k \,\middle|\, \left\| y^k - h\left(\mu^k\right) \right\| < \varepsilon \right\}$$

  - Select largest inlier set, reject all outliers

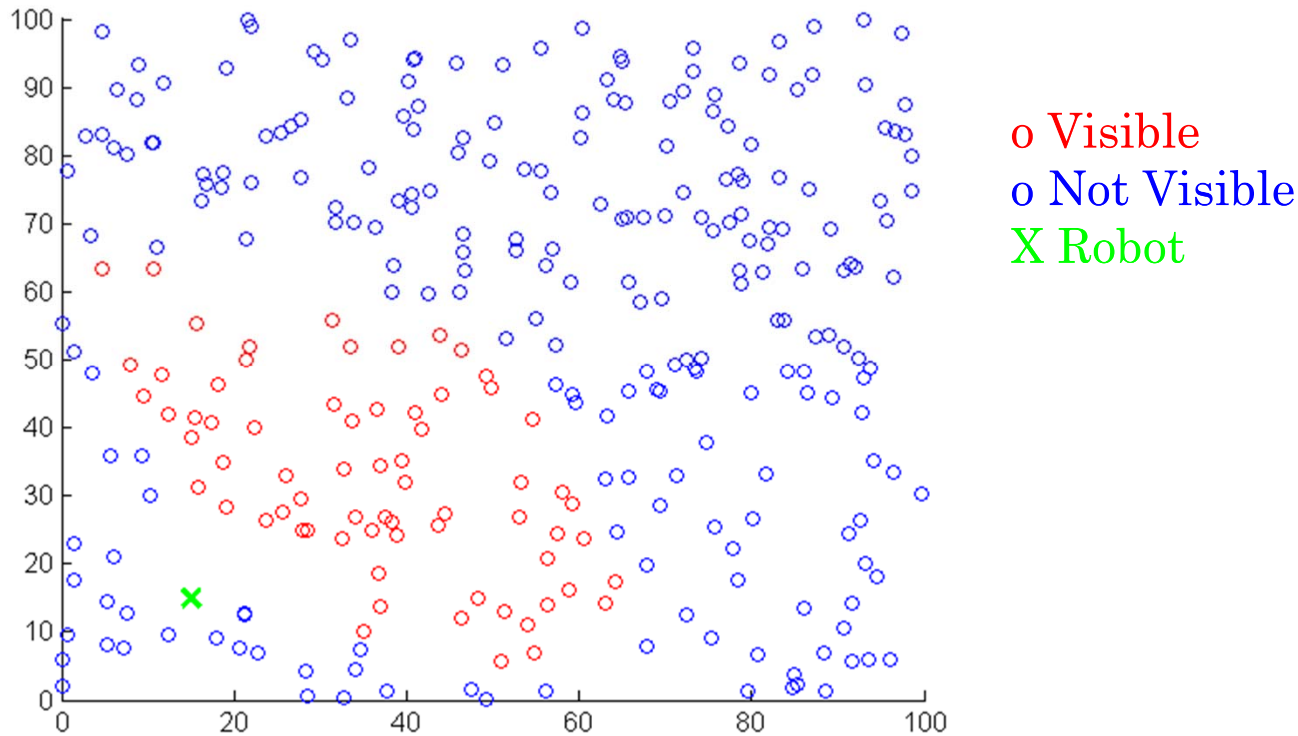
$$I^* = \max_k \left| I^k \right|$$

  - Recompute solution using the inlier set

$$\mu_t = EKF(\bar{\mu}_t, I^*)$$

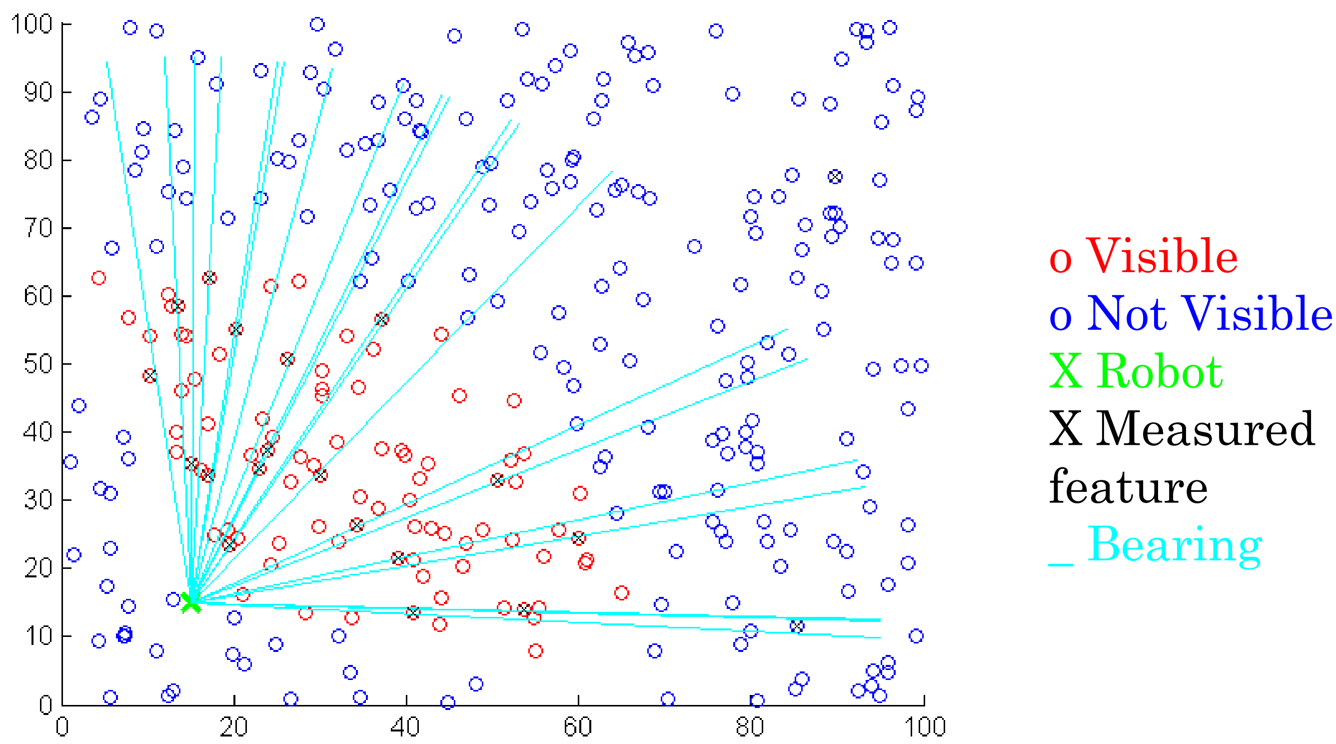
RANSAC Example

- Nonlinear least squares using bearing measurements in 2D
  - Known map of features
  - A subset fall in the field of view of the robot (50 m, 60 $^\circ$)



RANSAC Example

- Nonlinear least squares using bearing measurements in 2D
  - Measurements to features are bearings



RANSAC Example

- Nonlinear least squares using bearing measurements in 2D
  - Given an initial estimate of the pose of the robot and a measurement model,

$$\left[y_{i,t} \right] = h_i(x_t) = \left[\tan^{-1}\left(\frac{m_y^i - x_{2,t}}{m_x^i - x_{1,t}} \right) - x_{3,t} \right]$$

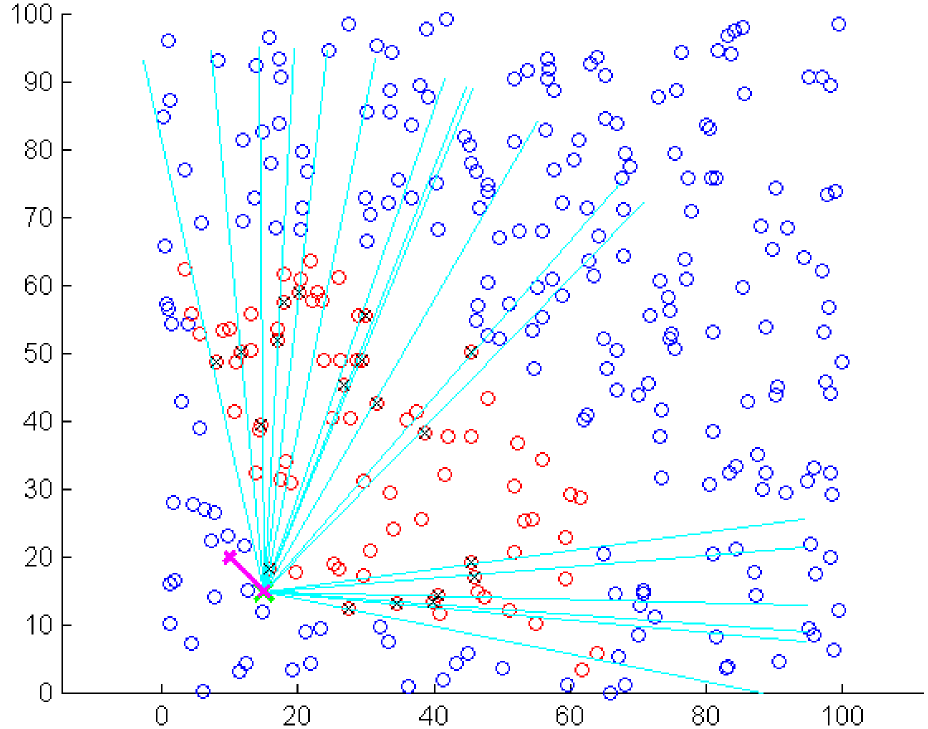
  - Solve nonlinear least squares problem (NLLS)

$$\mu(k+1) = \mu(k) + H^{\dagger}(y - h(\mu(k)))$$

    - Analogous to EKF, without motion update
    - At each step, find linear least squares solution, then relinearize and repeat until convergence

RANSAC Example

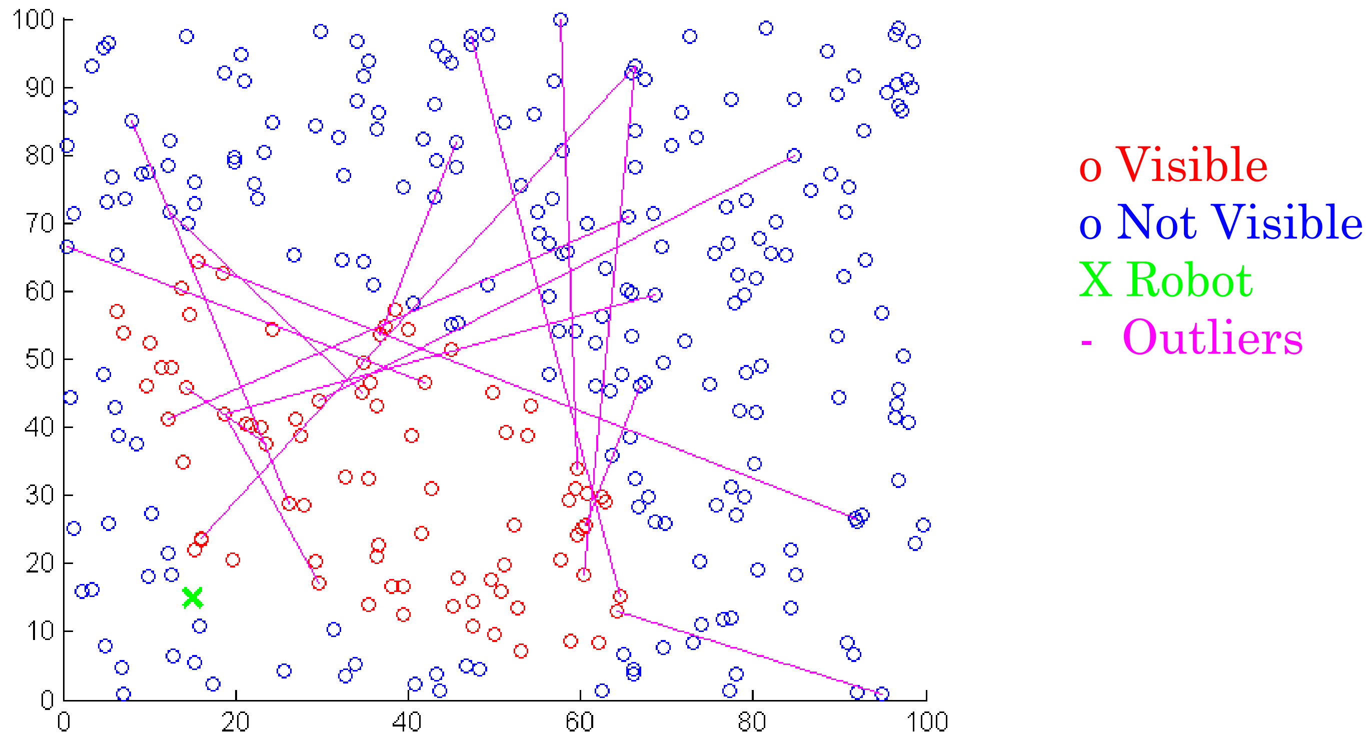
- Nonlinear least squares using bearing measurements in 2D
  - Prior x0 = [10 20 90]
  - Solution with 20 measurements, usually works.



o Visible
o Not Visible
X Robot
X Measured feature
_ Bearing
x- Solution

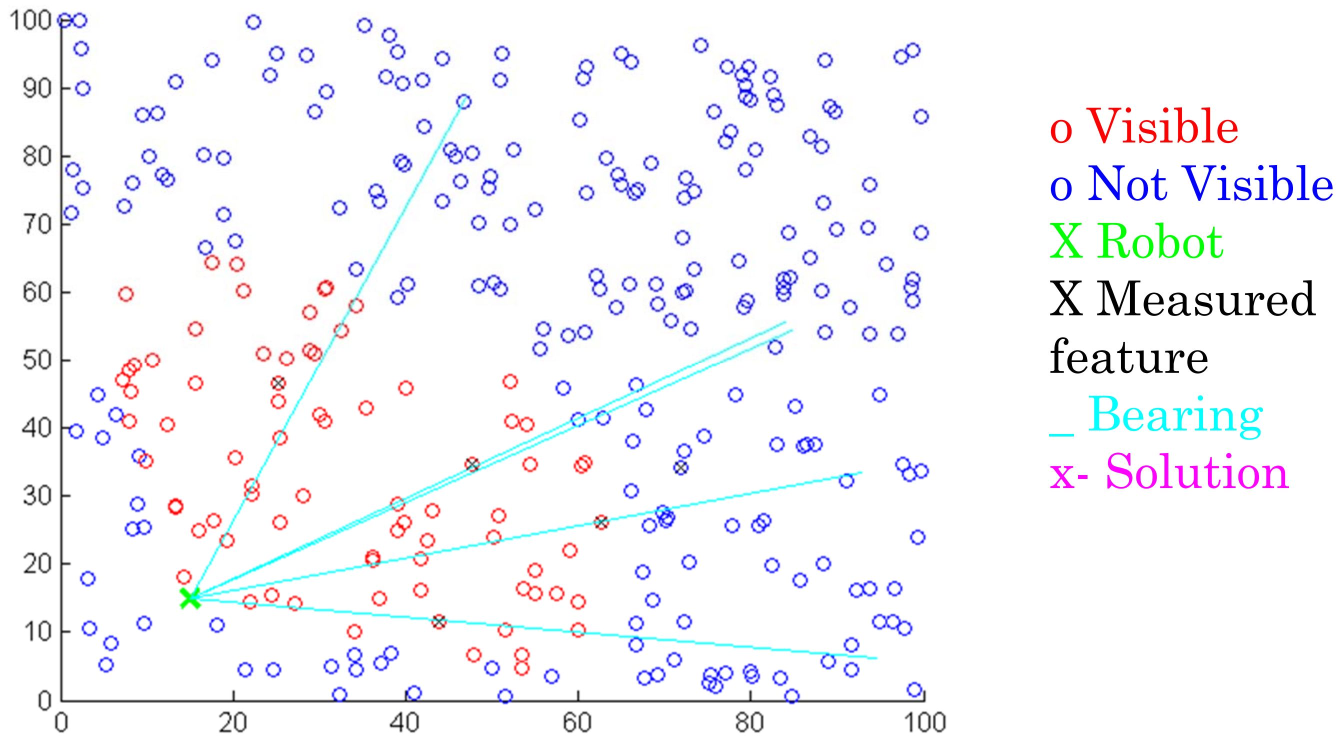
RANSAC Example

- Nonlinear least squares using bearing measurements in 2D
  - Add a certain percentage of outliers to the mix (e.g. 20%)
    - Measurements to the incorrect map feature



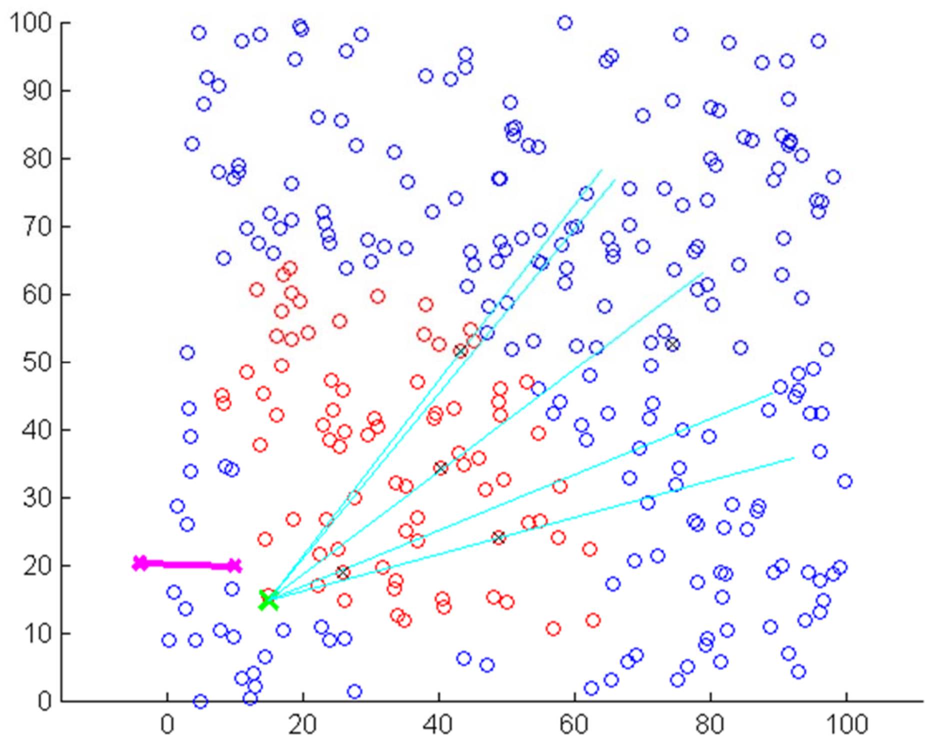
RANSAC EXAMPLE

- Nonlinear least squares using bearing measurements in 2D
  - Apply RANSAC to remove outliers and still get a good estimate (e.g. 100 iterations)
    - Pick small feature set (5 features) and solve NLLS



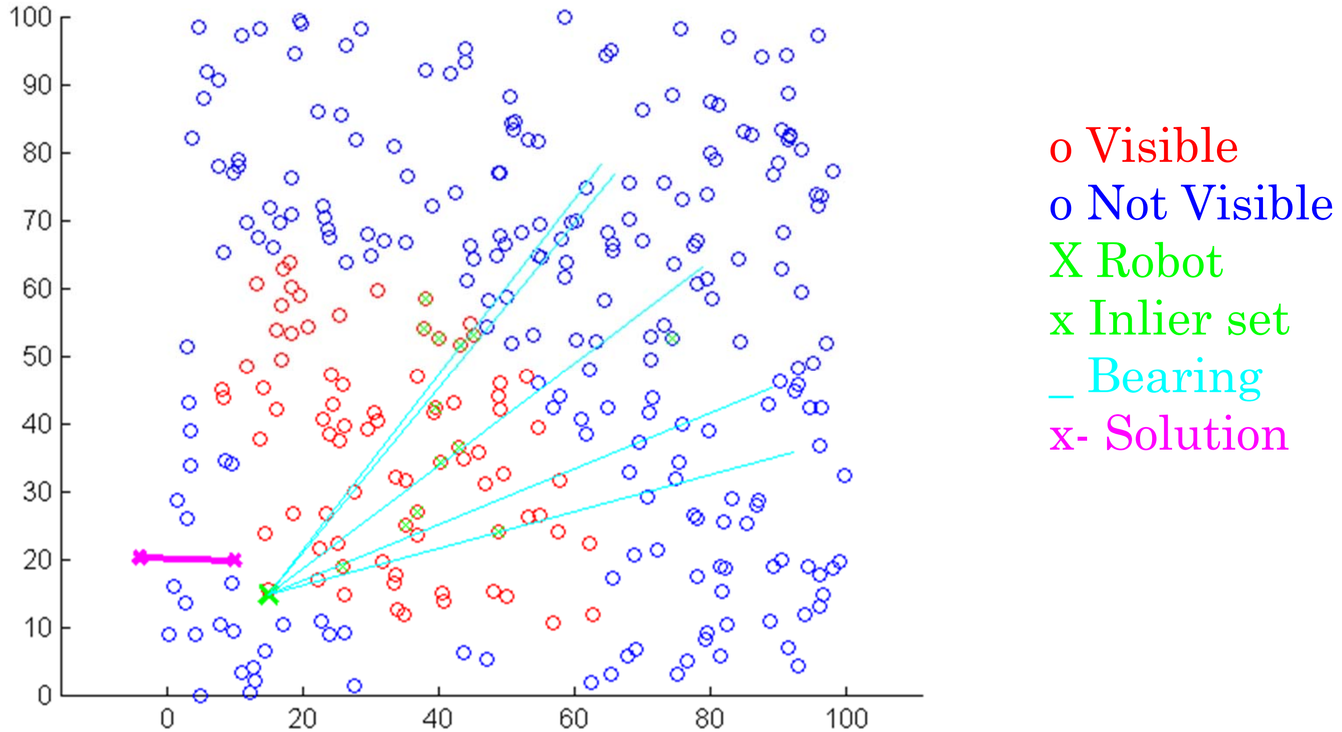
RANSAC EXAMPLE

- Nonlinear least squares using bearing measurements in 2D
  - Apply RANSAC to remove outliers and still get a good estimate (e.g. 100 iterations)
    - Pick small feature set (5 features) and solve NLLS



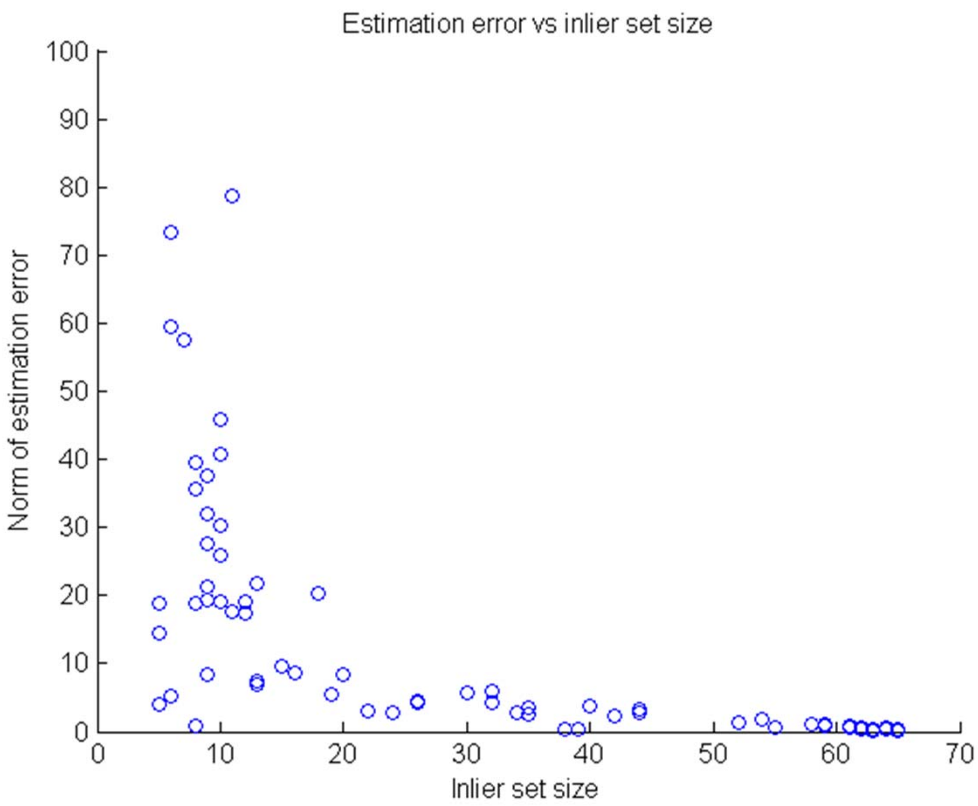
RANSAC Example

- Nonlinear least squares using bearing measurements in 2D
  - Find inlier set of all measurements that agree with current seed solution
    - Threshold on measurement error



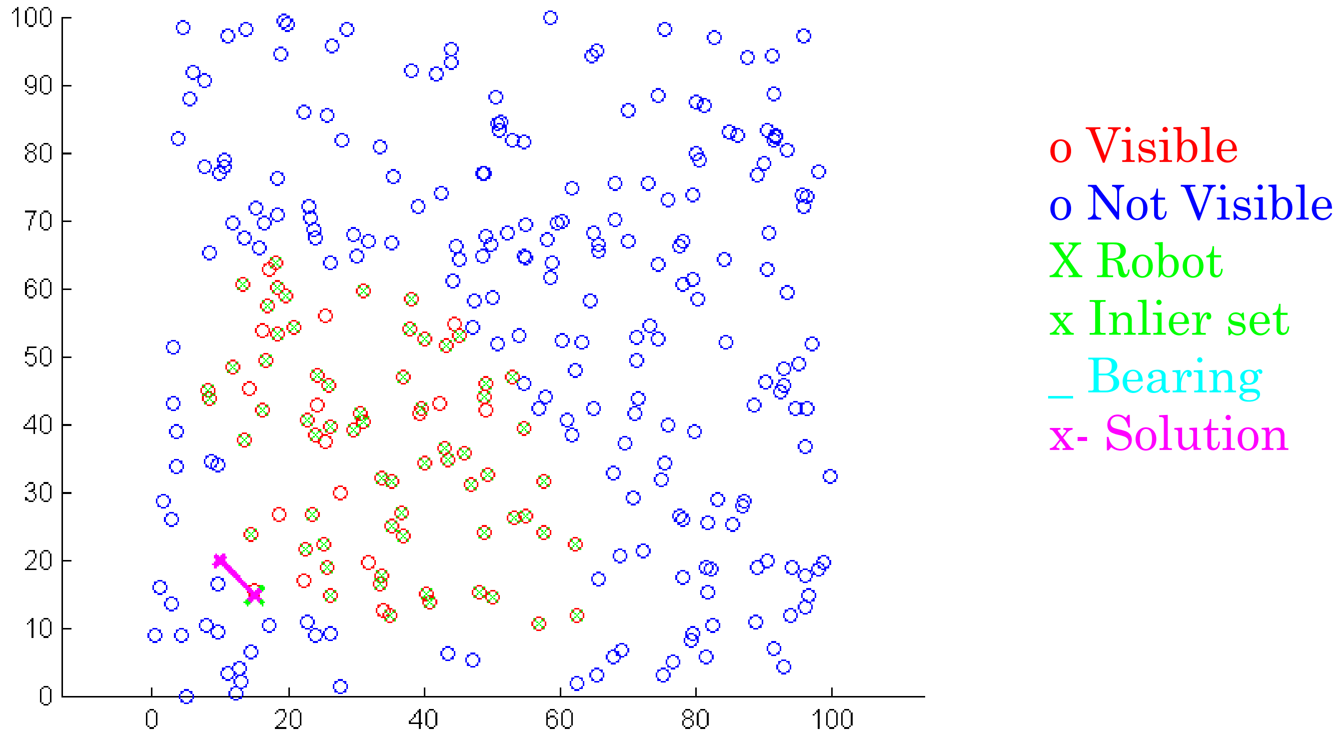
RANSAC Example

- Nonlinear least squares using bearing measurements in 2D
  - Repeat many times and save biggest inlier set



RANSAC EXAMPLE

- Nonlinear least squares using bearing measurements in 2D
  - Final solution looks quite good, and inlier set includes almost all measurements taken.
  - Not very expensive compared to finding features in the first place.



Mapping

- Mapping
  - Using sensor information from known vehicle locations to define a map of the environment
  - Given:
    - Vehicle location model
    - Sensor measurements and inverse measurement model
  - Find:
    - Environment map

MAPPING

- Occupancy Grid Mapping
  - Find probability at time t that each grid cell contains an obstacle

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

    - Subscript t moved to emphasize that features are static

- Assumptions
  - Static environment
  - Independence of cells
  - Known vehicle state at each time step
  - Sensor model is known

MAPPING

- Recall Discrete Bayes Filter Algorithm
  1. Prediction update (Discrete Total probability)

$$\overline{bel}(x_t) = \sum p(x_t \mid u_t, x_{t-1}) bel(x_{t-1})$$

  1. Measurement update (Bayes Theorem)

$$bel(x_t) = \eta p(y_t \mid x_t) \overline{bel}(x_t)$$

    - η is a normalizing constant that does not depend on the state (will become apparent in derivation)

MAPPING

- Bayes Filter with static states
  - Since the cell contents do not move, the motion model is trivial
    - The predicted belief is simply the belief from the previous time step

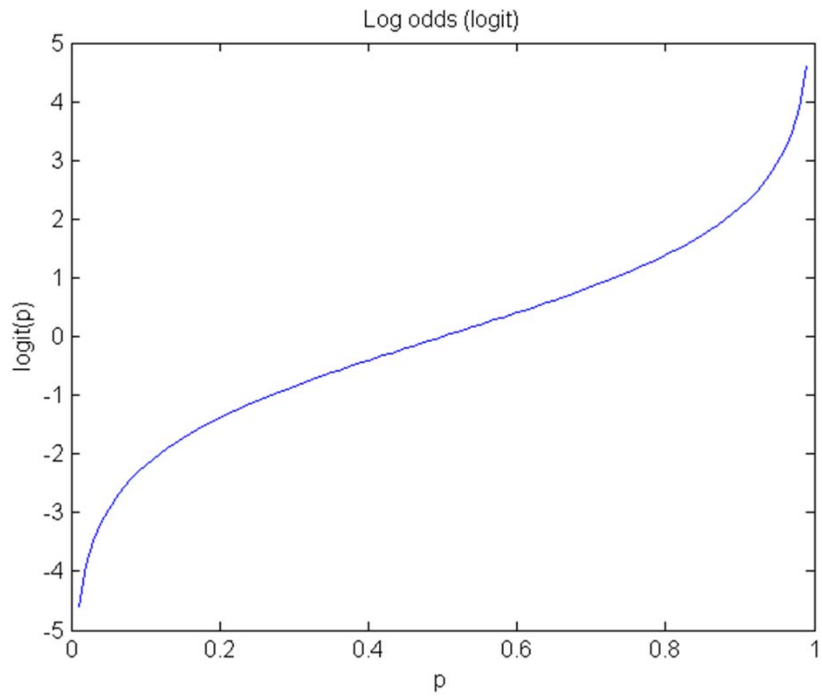
$$\overline{bel}_t(m) = bel_{t-1}(m)$$

    - The prediction step is no longer needed, so we update with each new measurement regardless of vehicle motion

$$bel_t(m) = \eta p(y_t \mid m) bel_{t-1}(m)$$

MAPPING

- Log Odds Ratio
  - Instead of tracking the probability, we track the log odds ratio for each cell



$$\text{logit}(p) = \log\left(\frac{p}{1-p}\right)$$

  - Referred to as logit function (logistic regression)

Mapping

- Log odds ratio
  - The big advantage is in dealing with low (and high) probability discrete states
    - Avoids issues with truncation in multiplicative combination of probabilities
  - As we'll see, the update rule involves addition only
  - Can always recover probability with

$$p = \frac{e^{\text{logit}(p)}}{1 + e^{\text{logit}(p)}}$$

MAPPING

- Bayesian log odds update derivation
  - For each cell, we have a measurement update (with the normalizer defined explicitly)

$$p(m^i \mid y_{1:t}) = \frac{p(y_t \mid y_{1:t-1}, m^i)\, p(m^i \mid y_{1:t-1})}{p(y_t \mid y_{1:t-1})}$$

  - We still trust in the Markov assumption

$$p(m^i \mid y_{1:t}) = \frac{p(y_t \mid m^i)\, p(m^i \mid y_{1:t-1})}{p(y_t \mid y_{1:t-1})}$$

Mapping

- Bayesian log odds update derivation
  - Let's apply Bayes rule to the measurement model

$$p(y_t \mid m^i) = \frac{p(m^i \mid y_t)\, p(y_t)}{p(m^i)}$$

  - Combining

$$p(m^i \mid y_{1:t}) = \frac{p(m^i \mid y_t)\, p(y_t)\, p(m^i \mid y_{1:t-1})}{p(m^i)\, p(y_t \mid y_{1:t-1})}$$

MAPPING

- Bayesian log odds update derivation
  - The same holds for the opposite event

$$p(\neg m^i \mid y_{1:t}) = 1 - p(m^i \mid y_{1:t}) = \frac{p(\neg m^i \mid y_t)\, p(y_t)\, p(\neg m^i \mid y_{1:t-1})}{p(\neg m^i)\, p(y_t \mid y_{1:t-1})}$$

  - Combining to get ratio

$$\frac{p(m^i \mid y_{1:t})}{p(\neg m^i \mid y_{1:t})} = \frac{\dfrac{p(m^i \mid y_t)\, p(y_t)\, p(m^i \mid y_{1:t-1})}{p(m^i)\, p(y_t \mid y_{1:t-1})}}{\dfrac{p(\neg m^i \mid y_t)\, p(y_t)\, p(\neg m^i \mid y_{1:t-1})}{p(\neg m^i)\, p(y_t \mid y_{1:t-1})}}$$

MAPPING

- Bayesian log odds update derivation
  - The ratio can now be simplified

$$\frac{p(m^i \mid y_{1:t})}{p(\neg m^i \mid y_{1:t})} = \frac{\dfrac{p(m^i \mid y_t)\, p(m^i \mid y_{1:t-1})}{p(m^i)}}{\dfrac{p(\neg m^i \mid y_t)\, p(\neg m^i \mid y_{1:t-1})}{p(\neg m^i)}}$$

  - And rewritten as

$$\frac{p(m^i \mid y_{1:t})}{p(\neg m^i \mid y_{1:t})} = \frac{p(m^i \mid y_t)\, p(\neg m^i)\, p(m^i \mid y_{1:t-1})}{p(\neg m^i \mid y_t)\, p(m^i)\, p(\neg m^i \mid y_{1:t-1})}$$

MAPPING

- Bayesian log odds update derivation
  - It is now possible to form the log odds ratio, expanding the negated terms

$$\frac{p(m^i \mid y_{1:t})}{1-p(m^i \mid y_{1:t})} = \frac{p(m^i \mid y_t)}{1-p(m^i \mid y_t)} \frac{1-p(m^i)}{p(m^i)} \frac{p(m^i \mid y_{1:t-1})}{1-p(m^i \mid y_{1:t-1})}$$

  - Finally, taking the log yields

$$\begin{aligned}\text{logit}(p(m^i \mid y_{1:t})) = {} & \text{logit}(p(m^i \mid y_t)) \\ & + \text{logit}(p(m^i \mid y_{1:t-1})) \\ & - \text{logit}(p(m^i))\end{aligned}$$

MAPPING

- Bayesian log odds update
  - A shorthand version of the update rule is

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

  - The log odd ratio at *t* is the sum of the ratio at *t-1* + the inverse measurement ratio – the initial belief
  - To get the inverse measurement ratio, we need an inverse measurement model
    - Probability of a state given a certain measurement occurs $$p(m^i \mid y_t)$$
    - Inverse conditional probability of the measurement models used to date $$p(y_t \mid m^i)$$

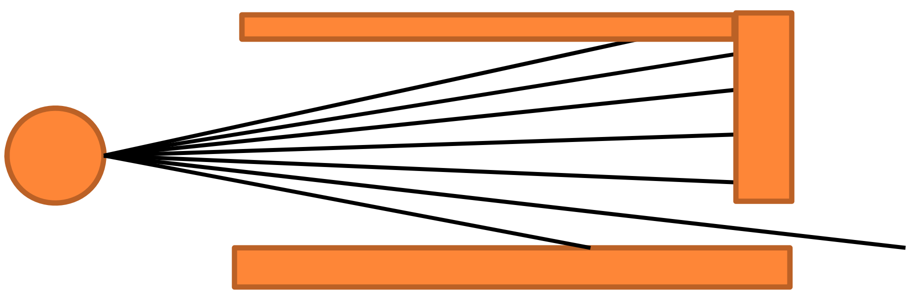
MAPPING

- Example: Laser Scanner
  - Returns a range to the closest objects at a set of bearings relative to the vehicle heading
    - Scanner bearings

$$\phi^s = \begin{bmatrix} -\phi^s_{\max} & \cdots & \phi^s_{\max} \end{bmatrix} \qquad \phi^s_j \in \phi^s$$

    - Scanner ranges

$$r^s = \begin{bmatrix} r^s_1 & \cdots & r^s_J \end{bmatrix} \qquad r^s_j \in \left[0, r^s_{\max}\right]$$

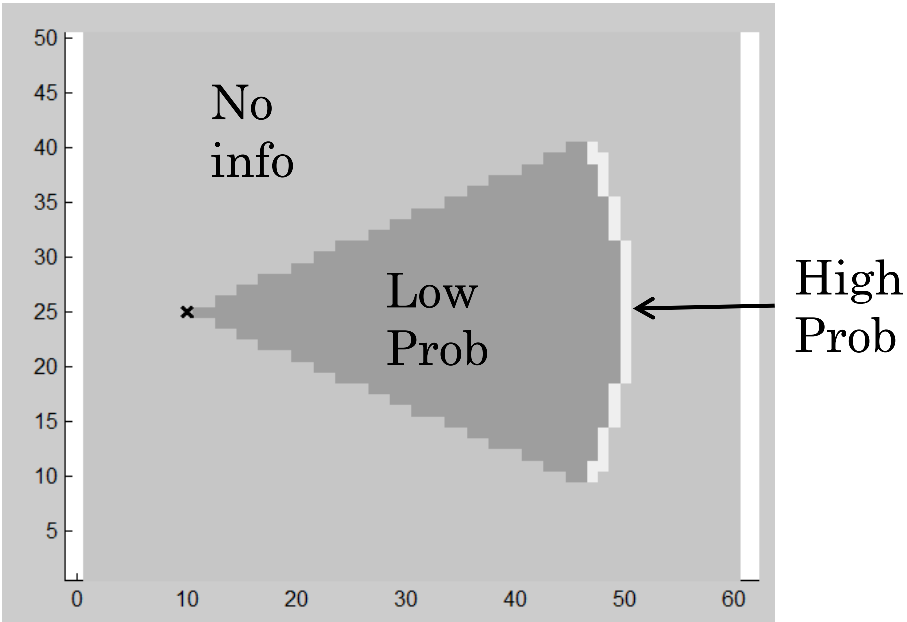


MAPPING

- Example: Laser Scanner
  - Inverse measurement model - easy
    - In 2D environment, three regions result

$$y_t = \begin{bmatrix} 40 \\ \vdots \\ 40 \end{bmatrix}$$

$$x_t = \begin{bmatrix} 10 \\ 25 \end{bmatrix}$$



- Simple and useful model, many improvements possible
  - See Thrun et al. Chap 6

MAPPING

- Example: Laser Scanner
  - Inverse measurement model - easy
    - Define relative range and bearing to each cell

$$r^i = \sqrt{\left(m_x^i - x_{1,t}\right)^2 + \left(m_y^i - x_{2,t}\right)^2}$$

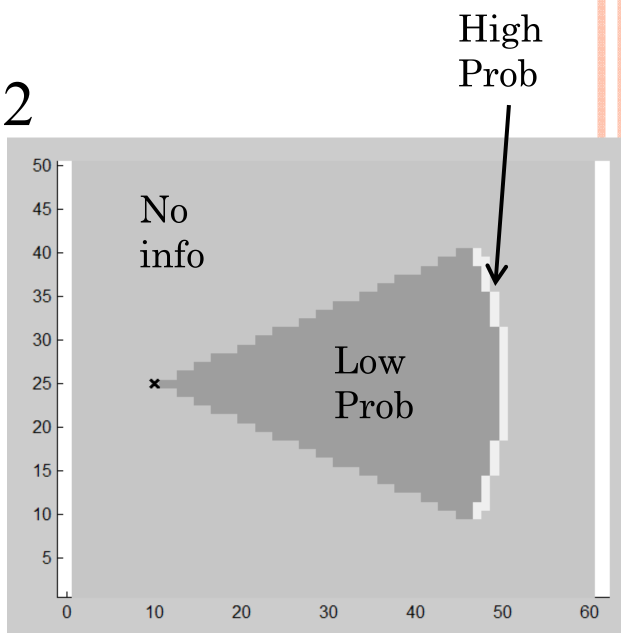
$$\phi^i = \tan^{-1}\left(\frac{m_y^i - x_{2,t}}{m_x^i - x_{1,t}}\right) - x_{3,t}$$

    - Find relevant range measurement for that cell
      - Closest bearing of a measurement

$$k = \arg\min\left(\left|\phi^i - \phi_j^s\right|\right)$$

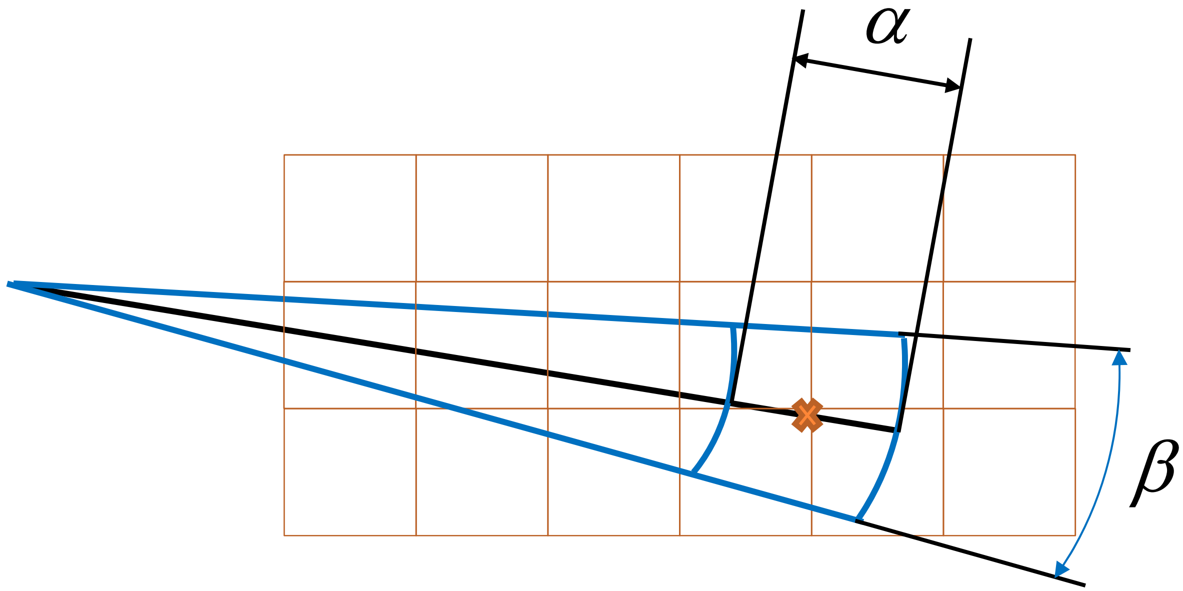
MAPPING

- Example: Laser Scanner
  - Inverse measurement model - easy
    - Identify each of the three regions and assign correct probability of object
    - if $r^i > \min(r^s_{\max}, r^s_k)$ or $|\phi^i - \phi^s_k| > \beta / 2$
      - then no info
    - else if $r^s_k < r^s_{\max}$ and $|r^i - r^s_k| < \alpha / 2$
      - then high probability of an object
    - else if $r^i < r^s_k$
      - then low probability of an object



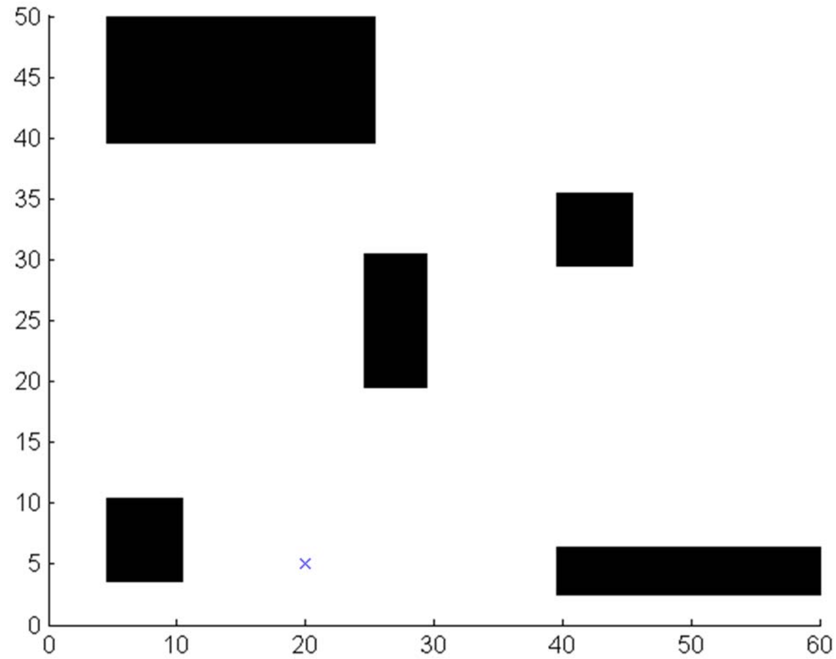
MAPPING

- Example: Laser Scanner
  - Inverse measurement model - easy
    - The parameters α and β define the extent of the region to be updated



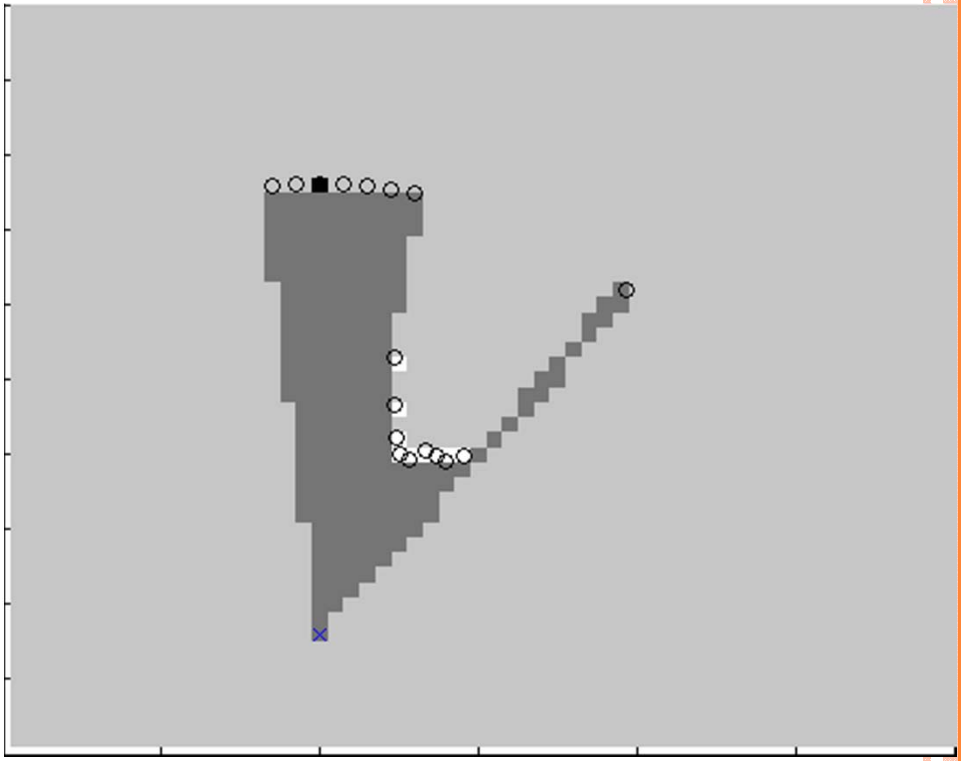
MAPPING

- Example
  - Simple motion
    - Move up until stuck
    - Turn right
    - Repeat
    - Rotate scanner at each timestep
  - Fixed map



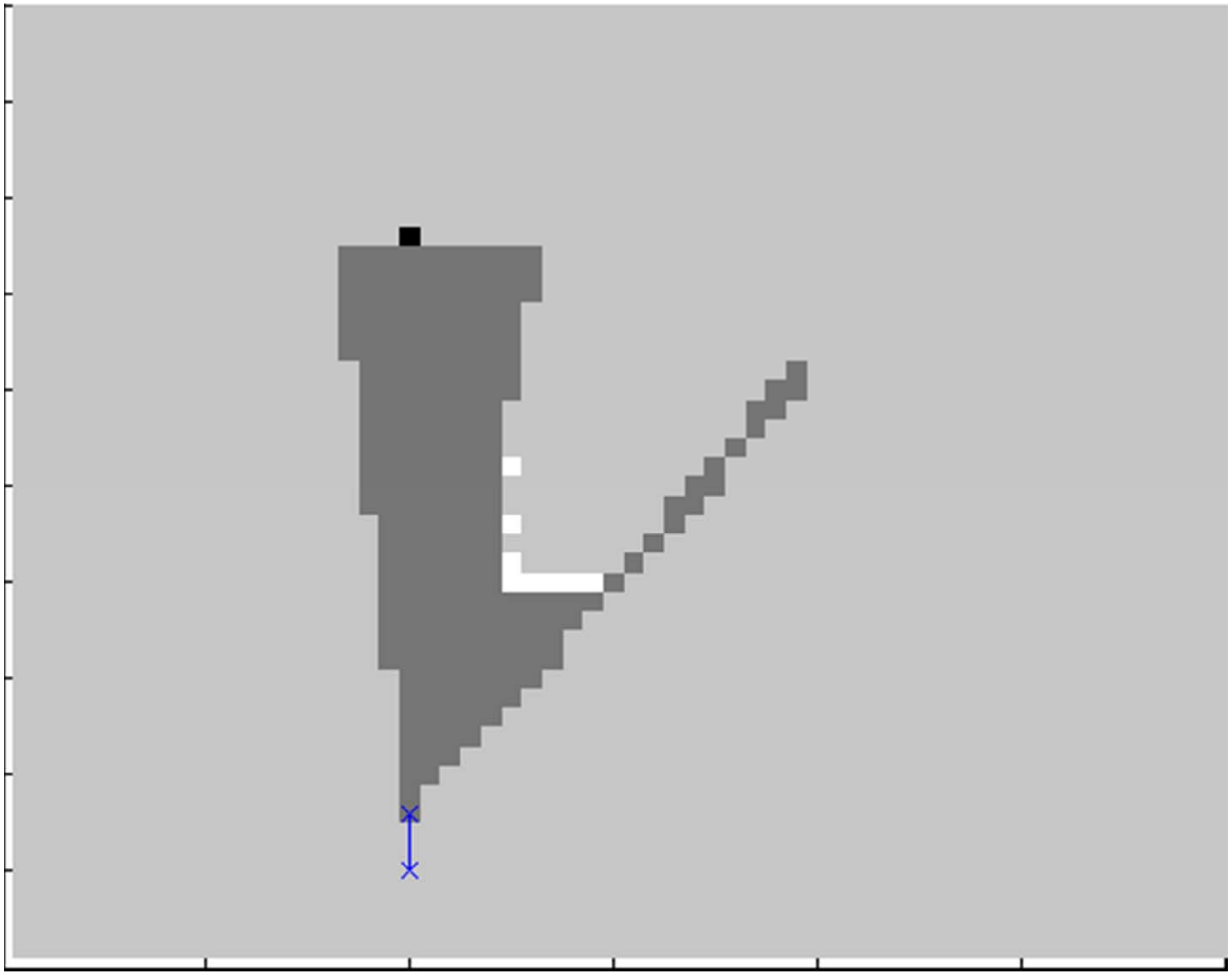
MAPPING

- Example
  - 17 Measurements
    - 46 degree FOV
    - 30 m max range
    - 1 set of measurements per time step
    - Probability of object at scan range: 0.6
    - Probability of no object in front: 0.4



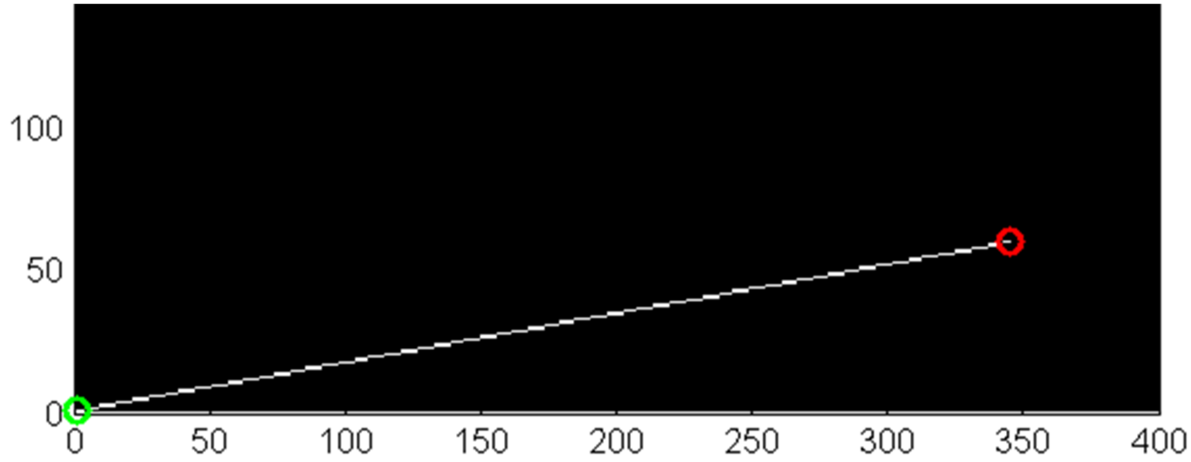
MAPPING

- Results
  - Map results



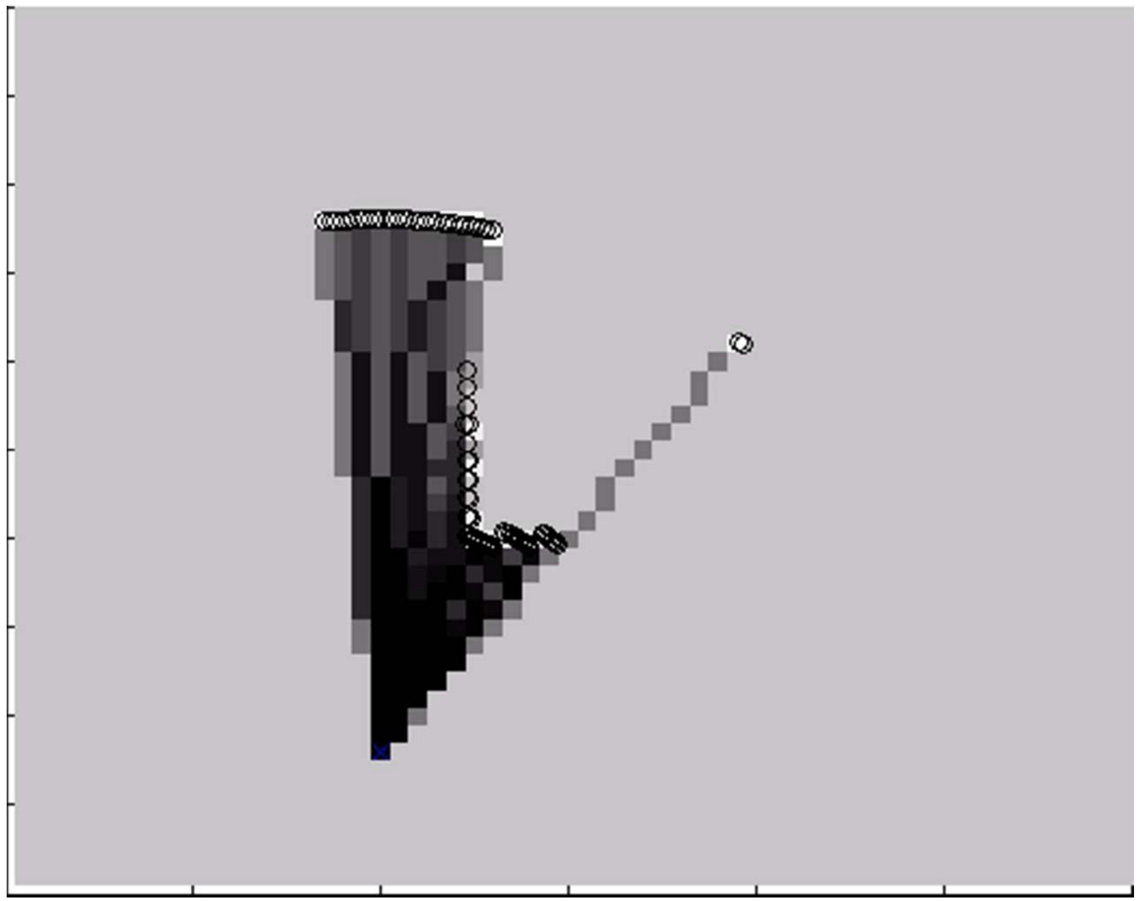
Mapping

- Inverse Measurement model - accurate
  - Instead of updating each cell once for a complete scan
  - Perform one update per range measurement
    - Raytracing using Bresenham's line algorithm
      - Bresenham at IBM in 1962
      - Used to draw lines for a plotter
      - Converted ray tracing into integer math update
      - Function provided in matlab library, details in extra slides



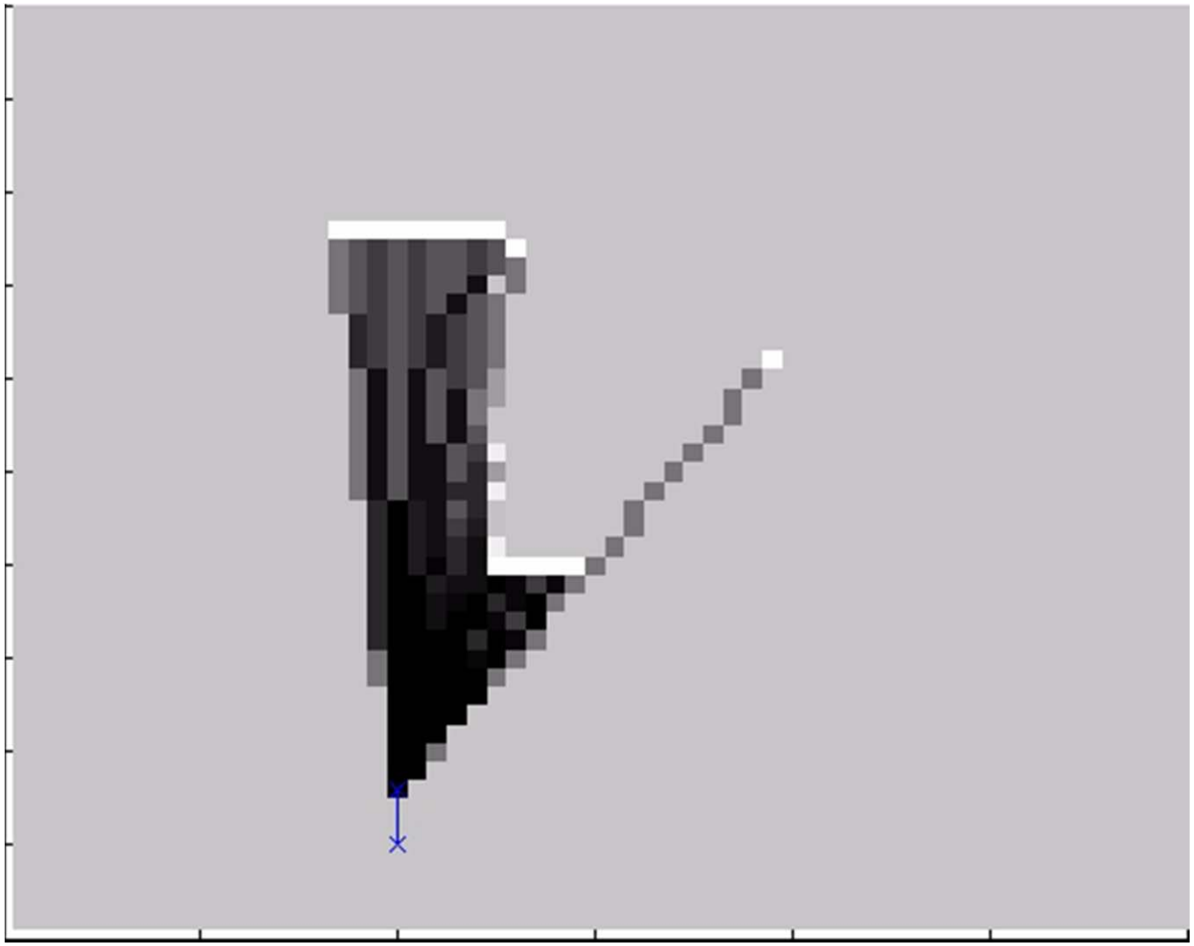
MAPPING

- Revisiting mapping with Bresenham's line algorithm
  - Inverse measurement model



MAPPING

- Revisiting mapping with Bresenham's line algorithm
  - Resulting map



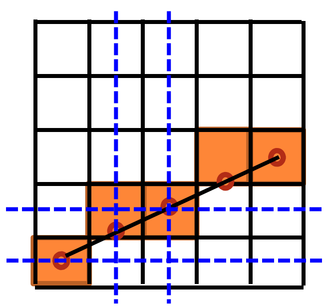
MAPPING

- Computation Issues
  - Grid Size
    - Calculation grows as resolution of grid increases
    - Topological approximations possible
  - Measurement model pre-caching
    - Model does depend on state, but does not change, so entire model can be pre-calculated
  - Sensor subsampling
    - Not all measurements need be applied, may be significant overlap in scans
  - Selective updating
    - Only update cells for which significant new information is available. (Do not update 3rd region).

Extra Slides

MAPPING

- Start with Simple Line Algorithm
  - Between 0-45 degrees, *x* increases faster than *y*
    - For all other ranges performed similarly, by switching *x* for *y* and flipping signs
  - Step one column at a time (move incrementally in *x*)
  - Decide if *y* should be incremented
- **Initialization: given (x0, y0, x1, y1)**
  - Slope = (y1 - y0) / (x1-x0)
  - error = 0
  - y = y0
- **Main loop: for** x **from** x0 **to** x1
  - plot(x, y)
  - error := error + slope*1
  - **if** error ≥ 0.5
    - y := y + 1
    - error := error - 1.0



MAPPING

- Simple line algorithm works well with a couple of exceptions
  - Floating point math, slower than necessary
  - Rounding error can lead to problems (addition of slope*1 at each step)
- Bresenham found a way to solve these problems by converting to integer math
  - Uses the following line definition

$$y = mx + b \;\Rightarrow\; y = \frac{\Delta y}{\Delta x}x + b \;\Rightarrow\; (\Delta x)y = (\Delta y)x + (\Delta x)b$$

$$f(x,y) = (\Delta y)x - (\Delta x)y + (\Delta x)b = 0$$

MAPPING

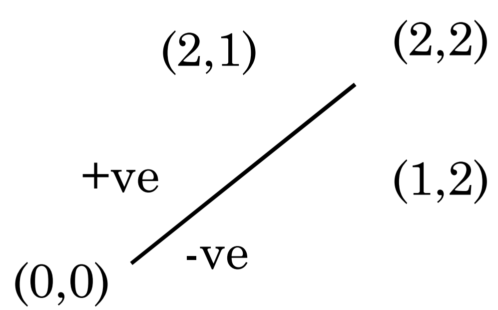
- Bresenham's line algorithm
  - Δx, Δy, b are all integers, as are x,y for any pixel location
    - Start and end pixels define delta Δx, Δy
    - Offset b at x = 0 is also in pixels
  - Given a line of this form

$$f(x,y) = (\Delta y)x - (\Delta x)y + (\Delta x)b = 0$$

  - Any point (x,y) not on the line has $f(x,y) \neq 0$
    - Above the line is positive
    - Below the line is negative

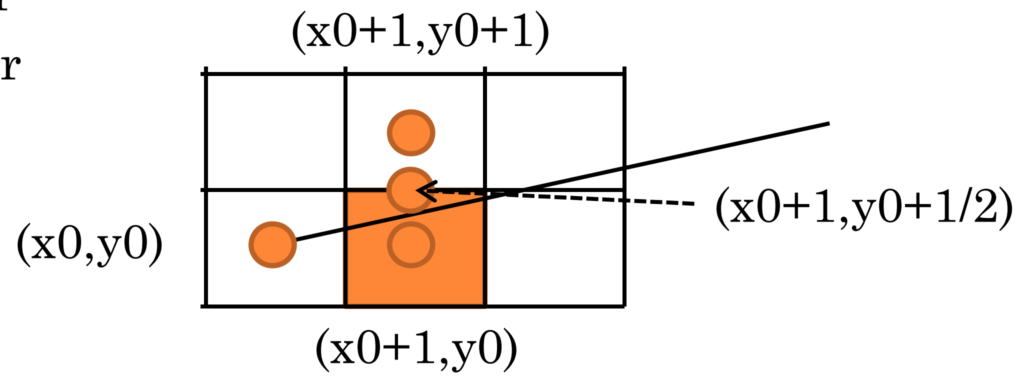
(2,1) $(2)2-(2)1+0=2$

(1,2) $(2)1-(2)2+0=-2$



Mapping

- Bresenham's line algorithm
  - Starting at (x0,y0), can now define two possible next pixels to add
    - (x0+1,y0+1) or (x0+1,y0)
    - Should select the one closer to the line
      - arg min (f(x0+1,y0), f((x0+1,y0+1))
  - To find out which, look at sign of line equation at f(x0+1,y0+1/2)
    - If > 0 pick lower
    - If < 0 pick upper



MAPPING

- Bresenham's line algorithm
  - Since we only care about the sign, can equally check the following line equation

$$2f(x0+1, y0+1/2) = 2(\Delta y)(x0+1) - 2(\Delta x)(y0+1/2) + 2(\Delta x)b = 0$$

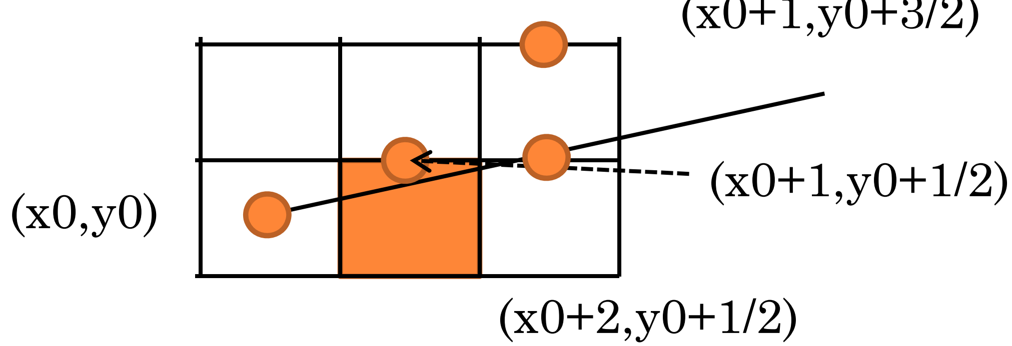
  - And better, we take the following difference

$$\begin{aligned} D &= 2f(x0+1, y0+1/2) - 2f(x0, y0) \\ &= 2(\Delta y)(x0+1) - 2(\Delta x)(y0+1/2) + 2(\Delta x)b - \\ &-2(\Delta y)(x0) - 2(\Delta x)(y0) + 2(\Delta x)b \\ &= 2(\Delta y) - (\Delta x) \end{aligned}$$

  - Since 2f(x0,y0) is on the line, it is equal to 0, so the sign of the difference D is all we need.

MAPPING

- Bresenham's line algorithm
  - This decision takes us one step forward along the line
  - To do the next step, we consider 2f(x0+2,y0+1/2) for this example, or 2f(x0+2,y0+3/2) if the line was steeper



  - Looking at the differences for those two points relative to the current midpoint value gives us an iterative update method for the difference value in the next column

MAPPING

- Bresenham's line algorithm
  - So we can pick the right piece to add to D, and make the next decision

$$
\begin{aligned}
D_{1/2} &= 2f(x0+2, y0+1/2) - 2f(x0+1, y0+1/2) \\
&= 2(\Delta y)(x0+2) - 2(\Delta x)(y0+1/2) + 2(\Delta x)b - \\
&\quad -2(\Delta y)(x0+1) - 2(\Delta x)(y0+1/2) + 2(\Delta x)b \\
&= 2(\Delta y)
\end{aligned}
$$

$$
\begin{aligned}
D_{3/2} &= 2f(x0+2, y0+3/2) - 2f(x0+1, y0+1/2) \\
&= 2(\Delta y)(x0+2) - 2(\Delta x)(y0+3/2) + 2(\Delta x)b - \\
&\quad -2(\Delta y)(x0+1) - 2(\Delta x)(y0+1/2) + 2(\Delta x)b \\
&= 2(\Delta y) - 2(\Delta x)
\end{aligned}
$$

MAPPING

- **Bresenham's line algorithm**
  - **function** line(x0, y0, x1, y1)
  - dx := abs(x1-x0)
  - dy := abs(y1-y0)
  - Inc1 = 2*dy
  - Inc2 = 2*dy-2*dx
  - D = 2*dy-dx
  - loop
    - *plot*(x0,y0)
    - **if** x0 = x1 **and** y0 = y1
      - return
    - x0 =x0+1;
    - **if** D < 0
      - D = D+Inc1
    - Else
      - D = D+Inc2
      - y0 = y0+1