

# SLAM Problem Statement

- Inputs:
  - No external coordinate reference
  - Time sequence of proprioceptive and exteroceptive measurements\* made as robot moves through an initially unknown environment
- Outputs:
  - A *map*\* of the environment
  - A robot *pose estimate* associated with each measurement, in the coordinate system in which the map is defined

\*Not yet fully defined

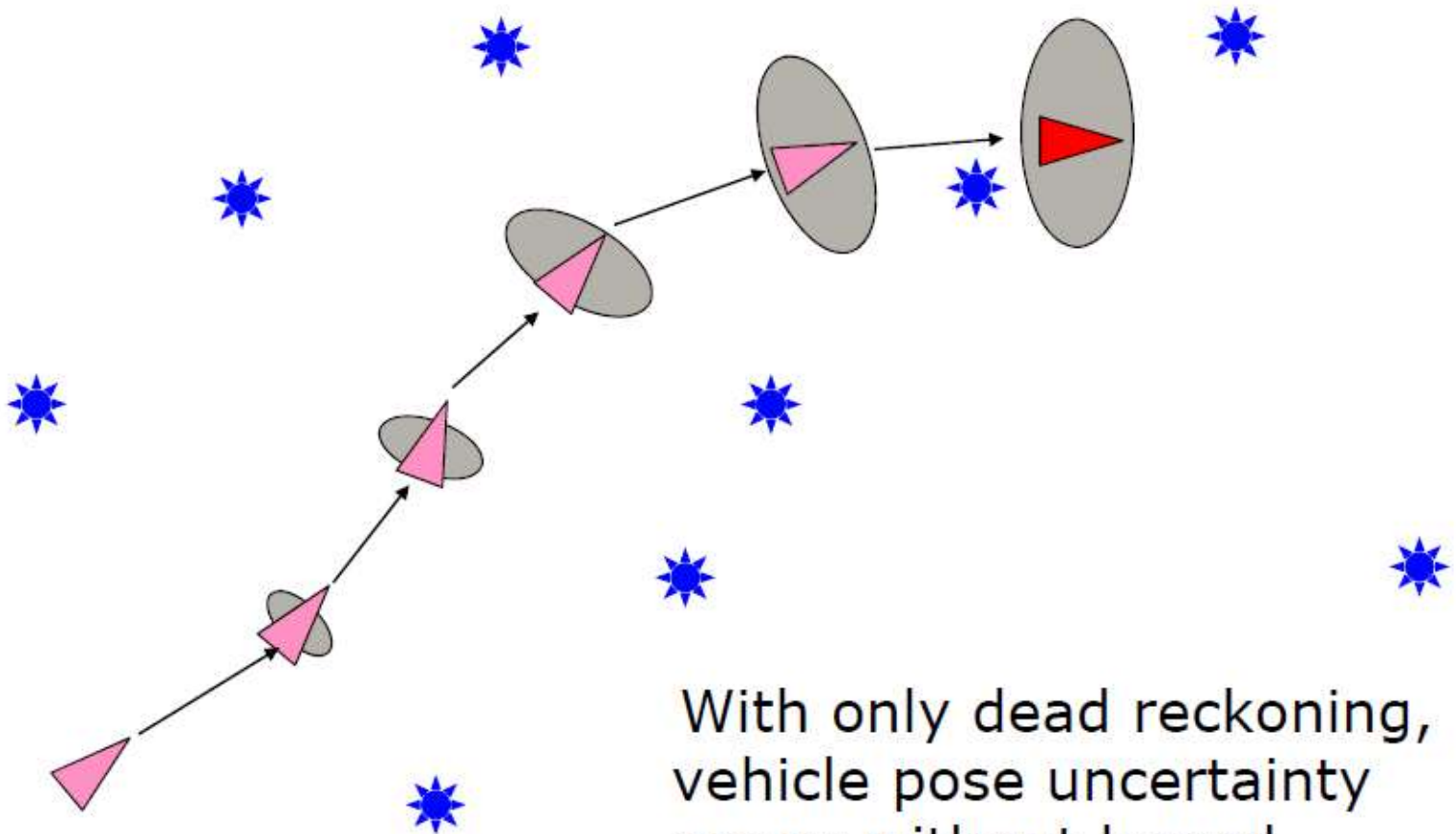
# SLAM Problem -- Incremental

- State/Output:
  - Map of env't observed "so far"
  - Robot pose estimate w.r.t. map
- Action/Input:
  - Move to a new position/orientation
  - Acquire additional observation(s)
- Update State:
  - Re-estimate the robot's pose
  - Revise the map appropriately

# SLAM Aspects

- What is a measurement?
  - What is a map?
  - How are map, pose coupled?
  - How should robot move?
  - What is hard about SLAM?
- 
- But first: some intuition

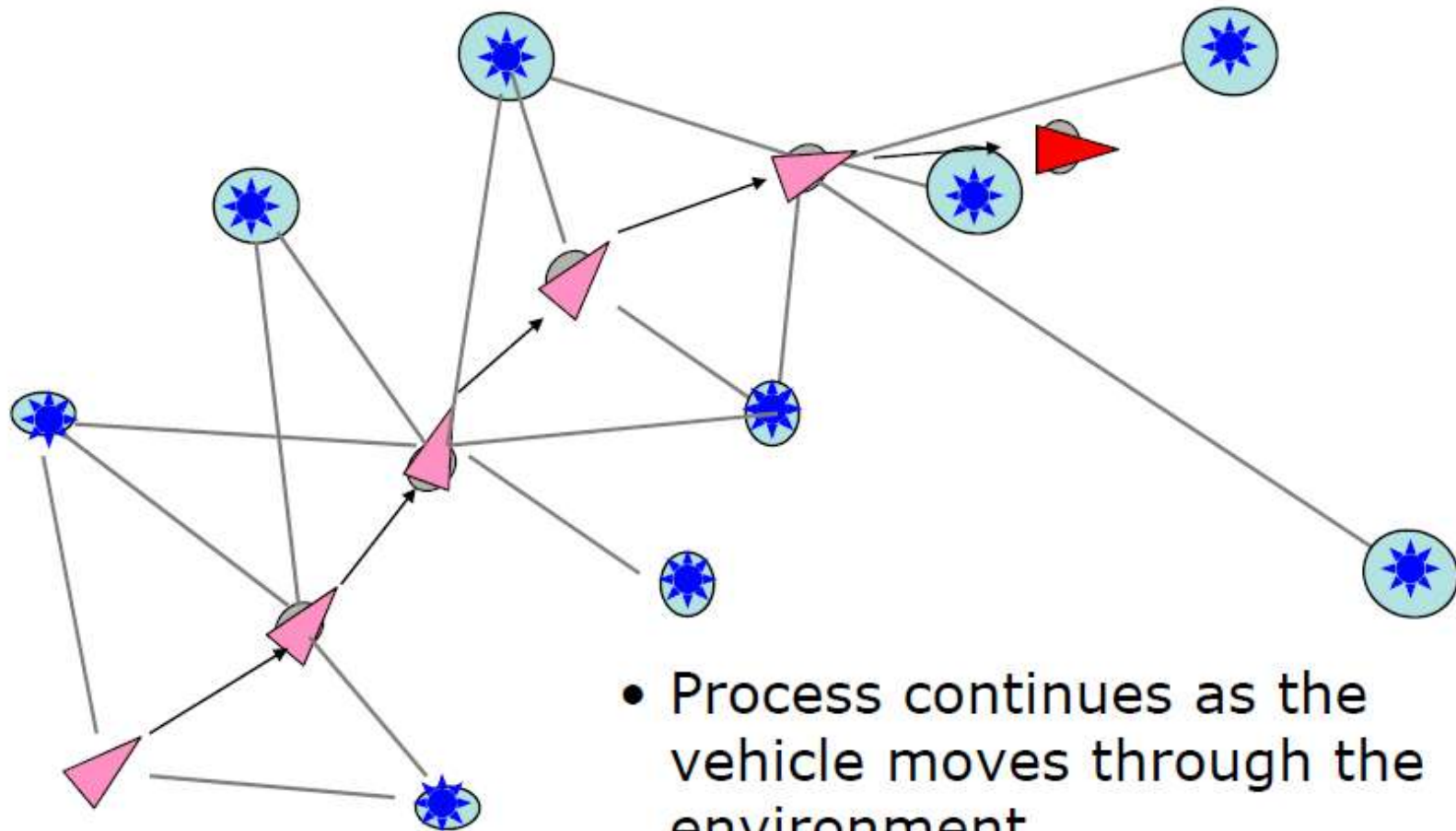
# Illustration of SLAM without Landmarks



With only dead reckoning,  
vehicle pose uncertainty  
grows without bound



# Illustration of SLAM with Landmarks



- Process continues as the vehicle moves through the environment

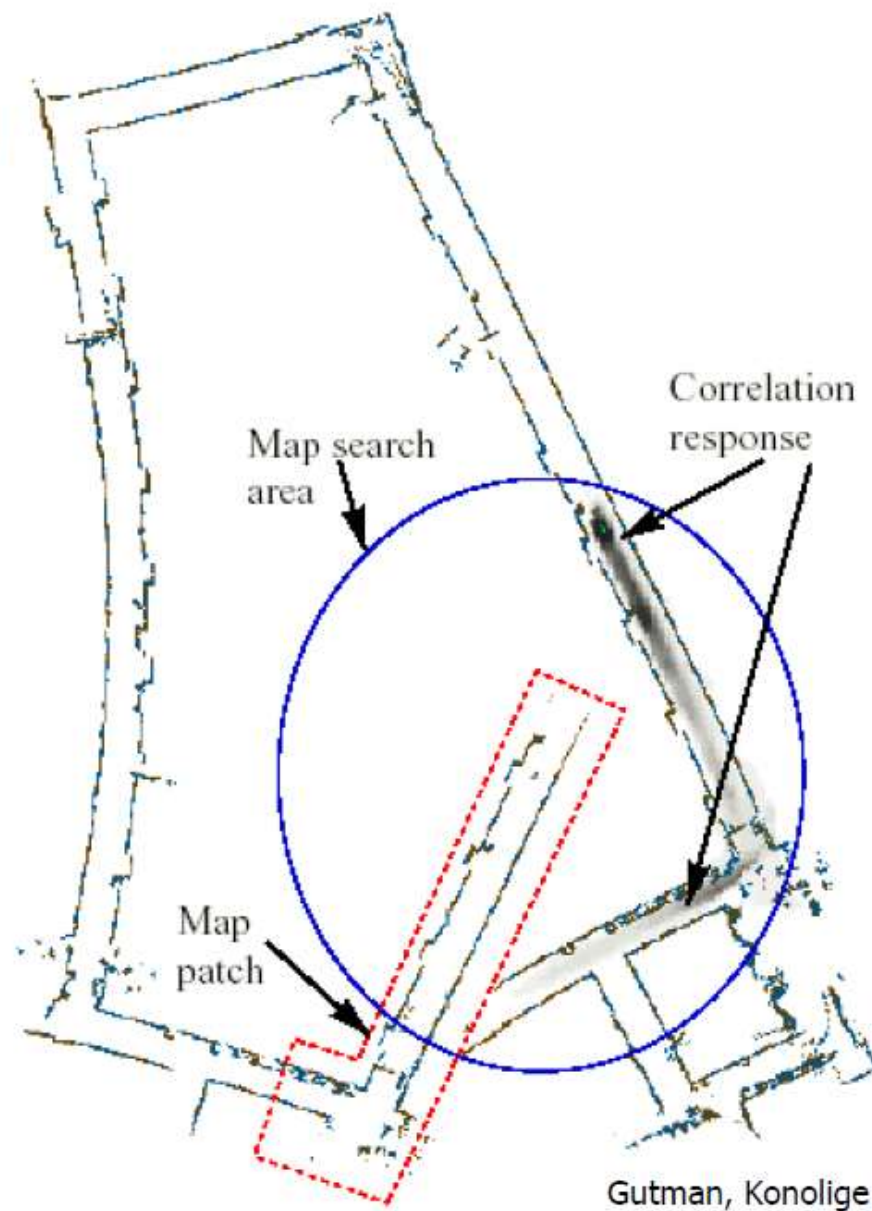
# Why is SLAM Hard?

- “Grand challenge”-level robotics problem
  - Autonomous, persistent, collaborative robots mapping multi-scale, generic environments
- Map-making = learning
  - Difficult even for humans
  - Even skilled humans make mapping mistakes
- Scaling issues
  - Space: Large extent (combinatorial growth)
  - Time: Persistent autonomous operation
- “Chicken and Egg” nature of problem
  - If robot had a map, localization would be easier
  - If robot could localize, mapping would be easier
  - ... But robot has neither; starts from blank slate
  - Must also execute an *exploration strategy*
- **Uncertainty** at every level of problem

# Uncertainty in Robotic Mapping



Scale: Uncertainty:	Continuous	Discrete
Local	Sensor noise	Data association
Global	Navigation drift	Loop closing

# Loop Closing

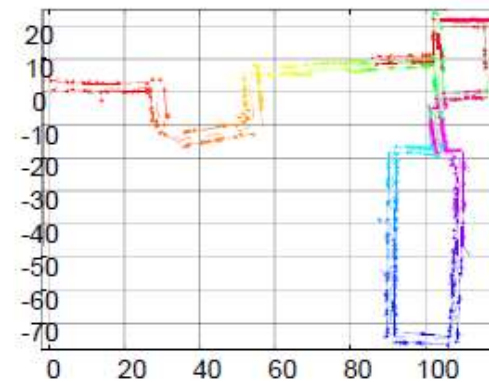
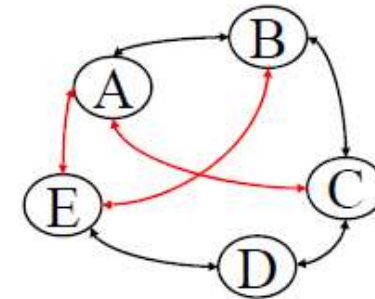
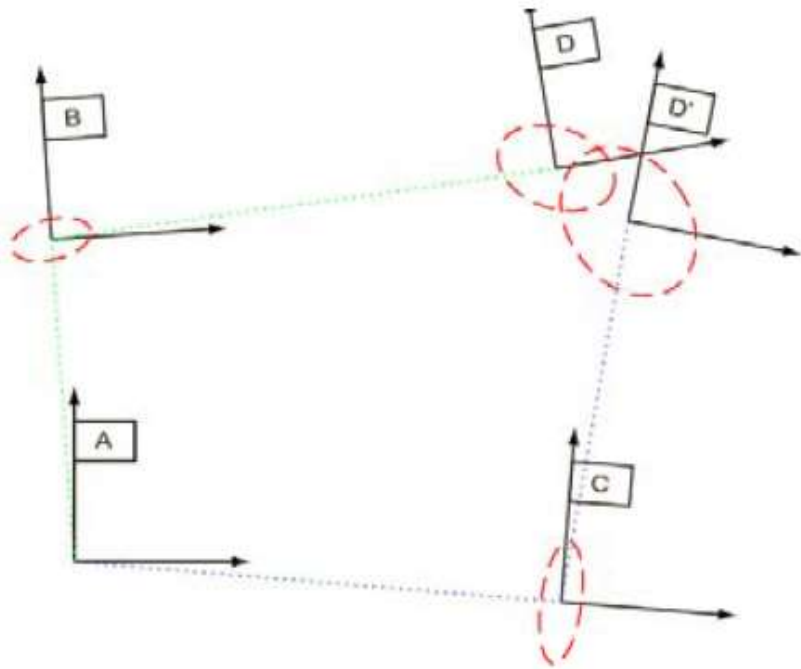




# What is a map?

- Collection of *features* with some *relationship* to one another
- What is a *feature*?  **Uncertainty**
  - Occupancy grid cell
  - Line segment
  - Surface patch
- What is a feature *relationship*? 
  - Rigid-body transform (metrical mapping)
  - Topological path (chain of co-visibility)
  - Hybrid representation (geom. + topo.)

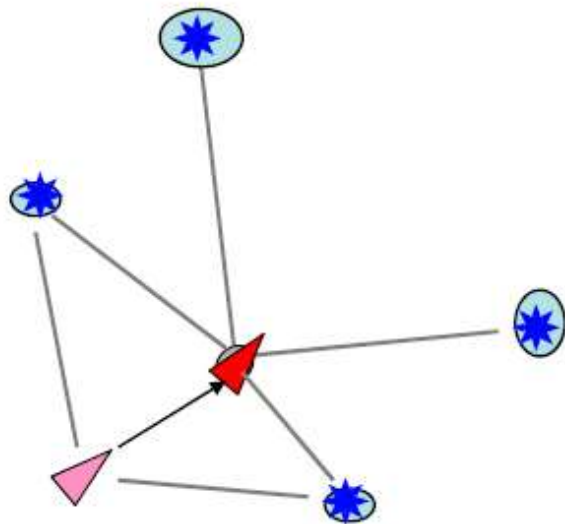
# Atlas hybrid maps (Bosse et al.)



- Features: point, line, patch clouds
- Geometry: rigid frames, submaps
- Topology: map adjacencies
- Hybrid: uncertain map-to-map transformations

# What is *pose* w.r.t. a map?

- Pose estimate that is (maximally) *consistent* with the estimated features observed from vicinity
- Consistency can be evaluated locally, semi-locally, or globally



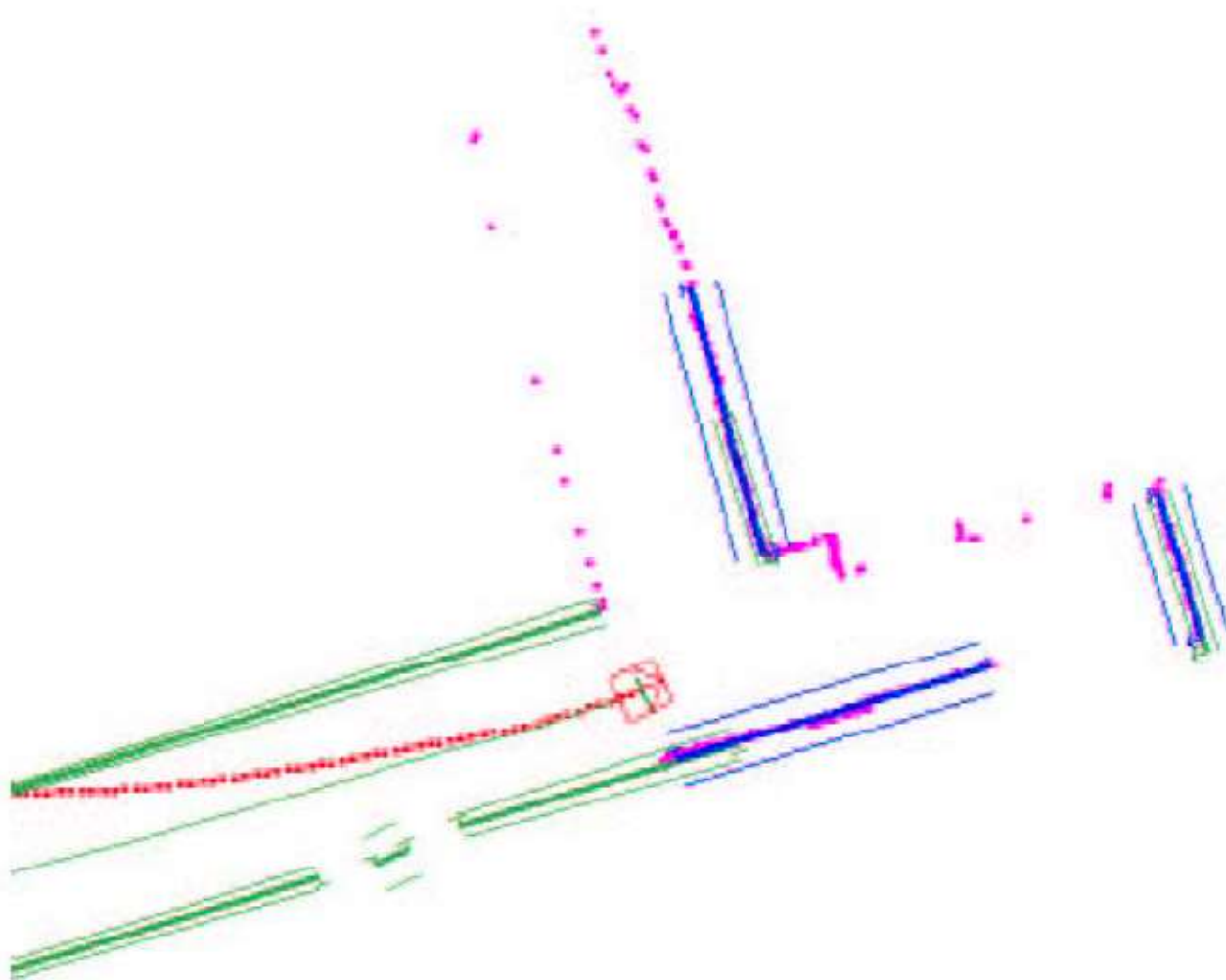
- Note tension between estimation *precision* and solution *consistency*

# Example

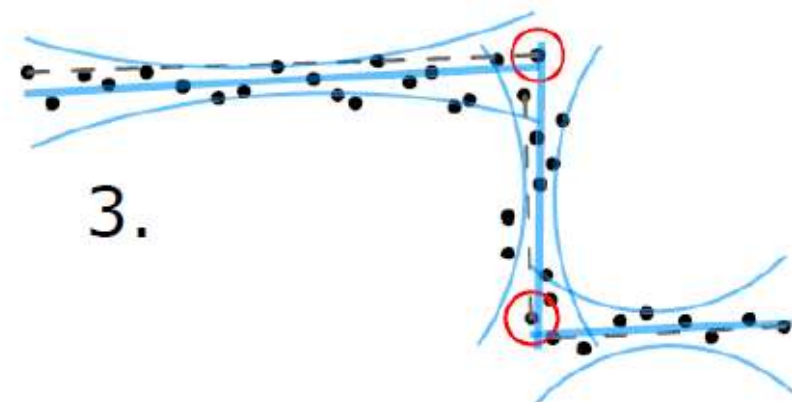
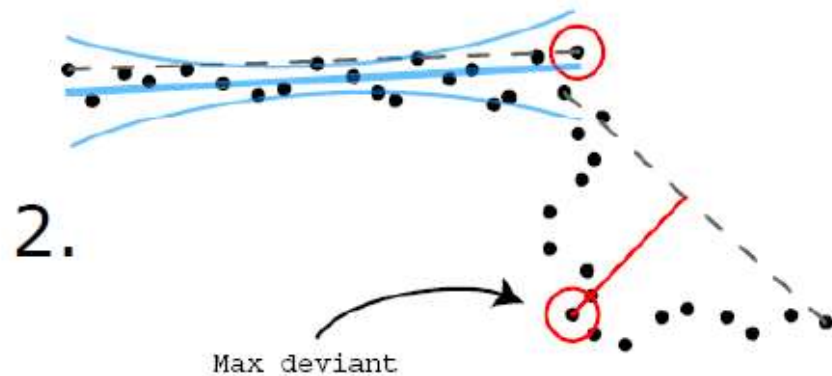
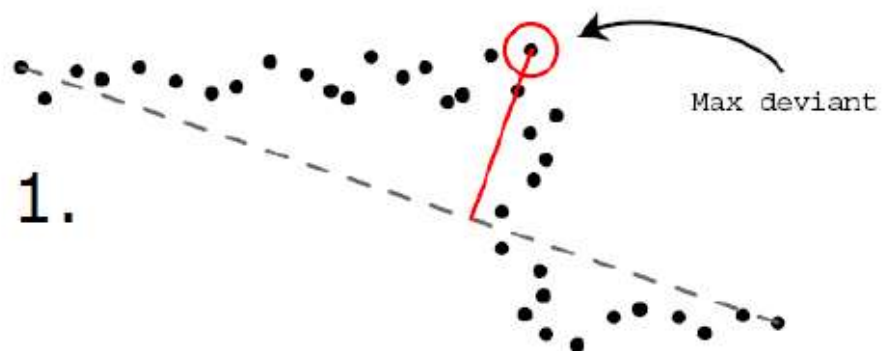
- SLAM with laser scanning
- Observations
- Local mapping
  - Iterated closest point
- Loop closing
  - Scan matching
  - Deferred validation
  - Search strategies



# Observations

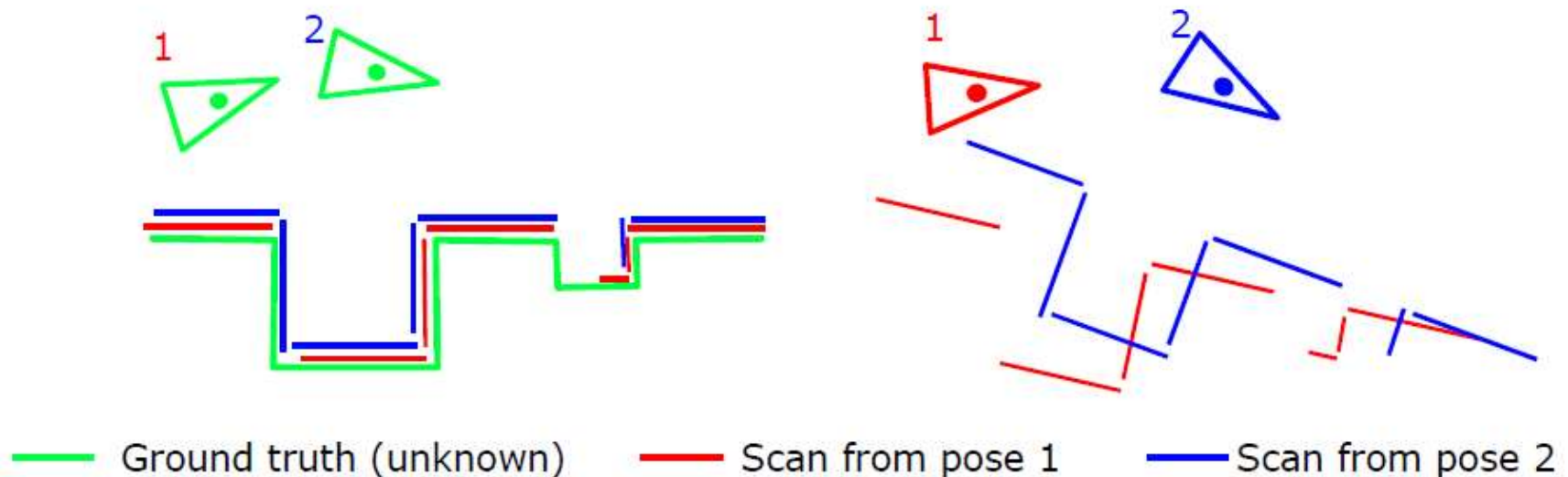


# Observations



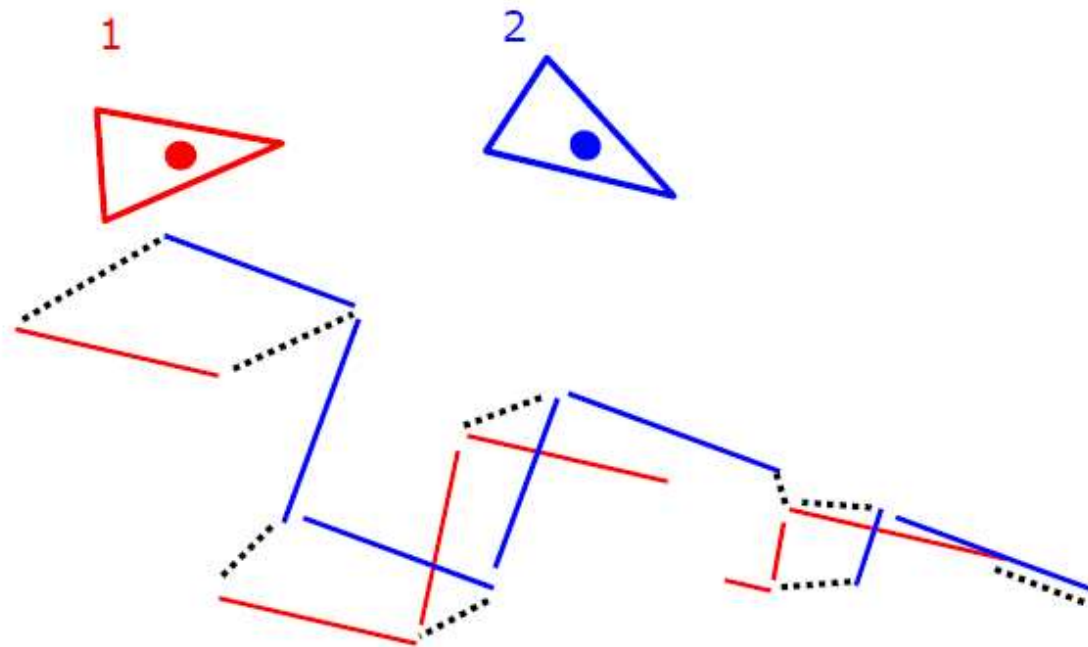
# Scan Matching

- Robot scans, moves, scans again
- Short-term odometry/IMU error causes misregistration of scans
- *Scan matching* is the process of bringing scan data into alignment



# Iterated Closest Point

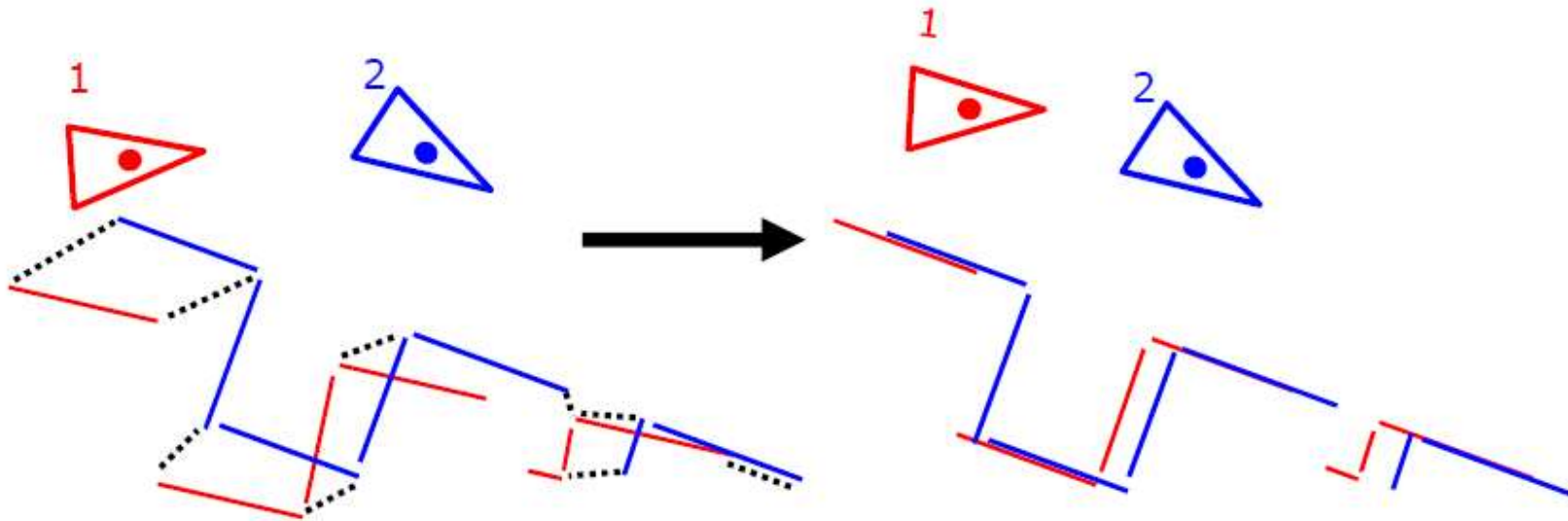
- For each point in scan 1
  - Find closest point in scan 2





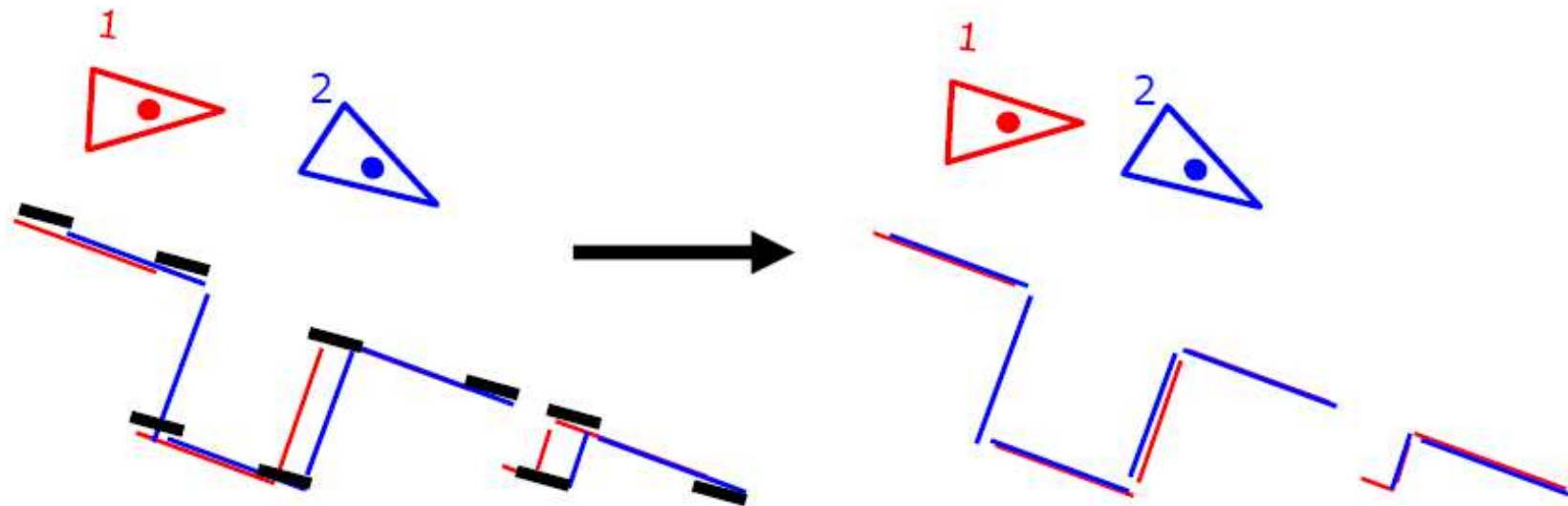
# Iterated Closest Point

- Find the transformation that best aligns the matching sets of points



# Iterated Closest Point

- ... Repeat until convergence



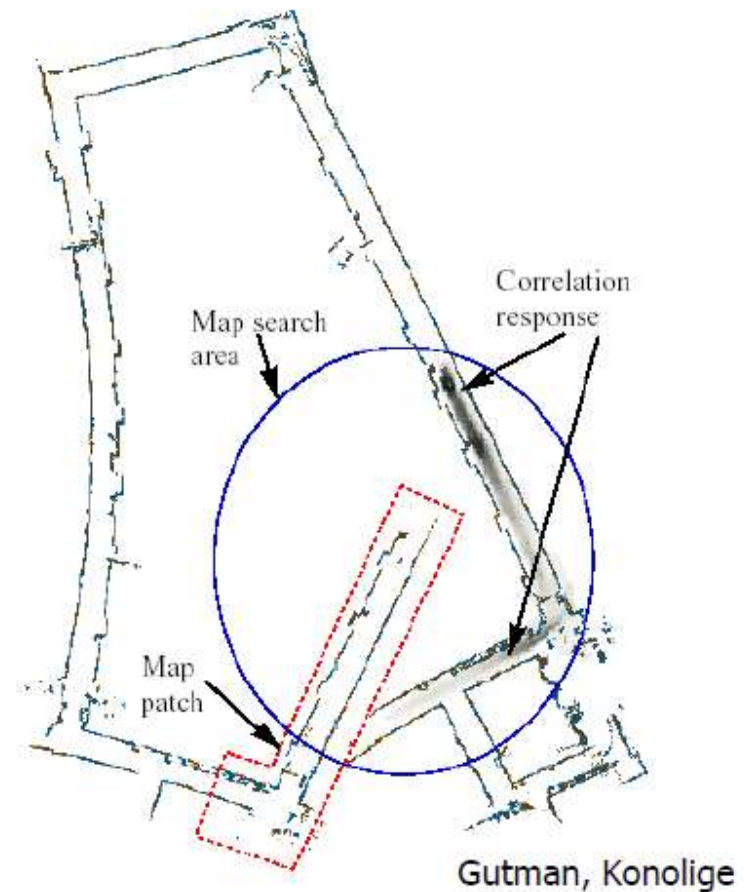
- Can ICP across scans, across a scan and a (sub)map, or even across submaps!

# Limitations / failure modes

- Computational cost (two scans of size  $n$ )
  - Naively,  $O(n^2)$  plus cost of alignment step
- False minima
  - If ICP starts far from true alignment
  - If scans exhibit repeated local structure
- Bias
  - Anisotropic point sampling
  - Differing sensor fields of view (occlusion)
- Lots of research on improved ICP methods (see, e.g., Rusinkiewicz)

# Loop Closing

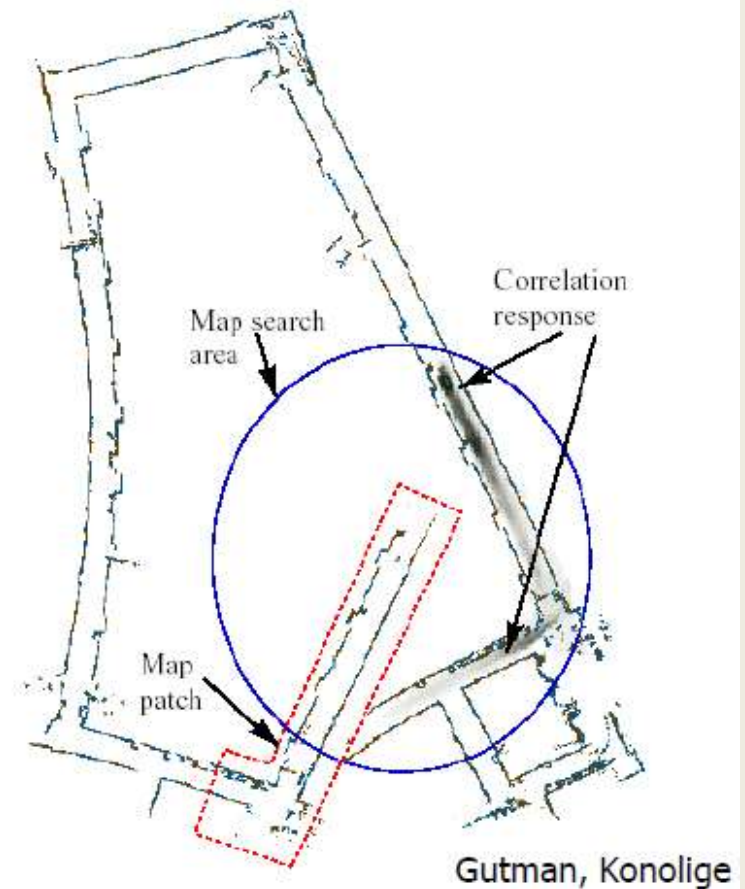
- ICP solves small-scale, short-duration SLAM fairly well
- But now, consider:
  - Large scale
  - High uncertainty





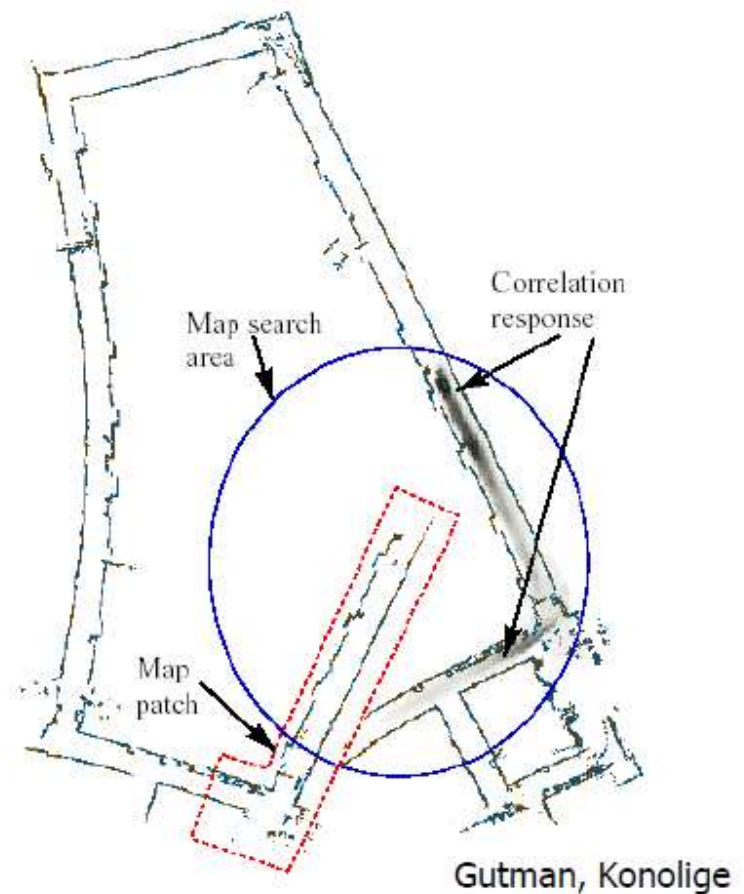
# Loop Closing

- Naive ICP ruled out:
  - Too CPU-intensive
- Assume we have a pose *uncertainty bound*
- This limits portion of existing map that must be searched
- Still have to face the problem of matching two partial scans that are far from aligned



# Scan Matching Strategies

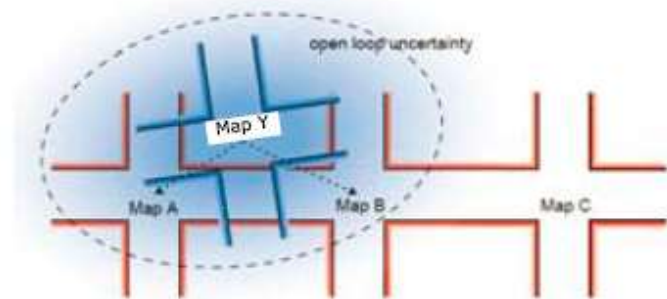
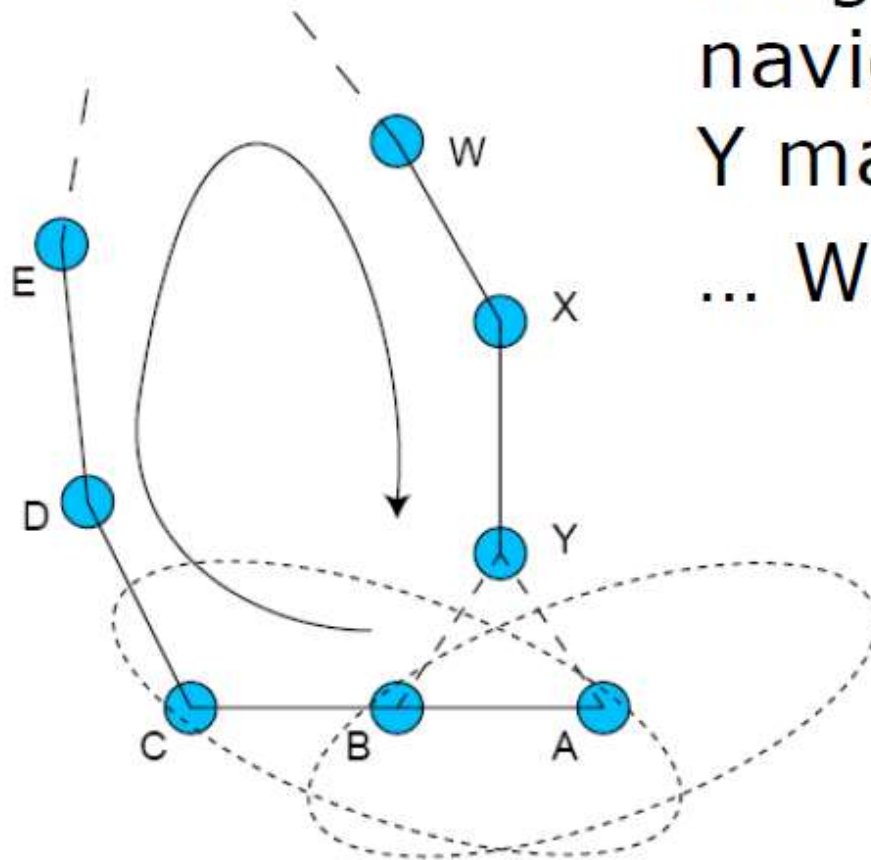
- Exhaustive search
  - Discretize robot poses
  - Find implied alignments
  - Assign score to each
  - Choose highest score
  - Pros, Cons?
- Randomized search
  - Choose minimal sufficient match, at random
  - Align and score
  - Choose highest score
  - RANSAC (1981)
  - Pros, Cons?



# Loop Closing Ambiguity

- Consider SLAM state after ABC ... XY

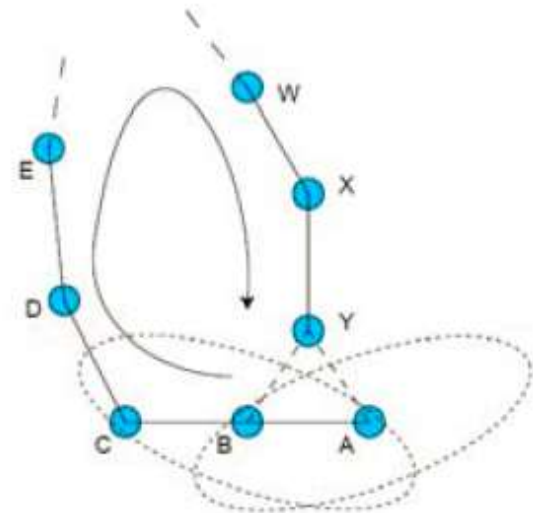
Large open-loop  
navigation uncertainty  
Y matches *both* A & B  
... What to do?





# Loop Closing Choices

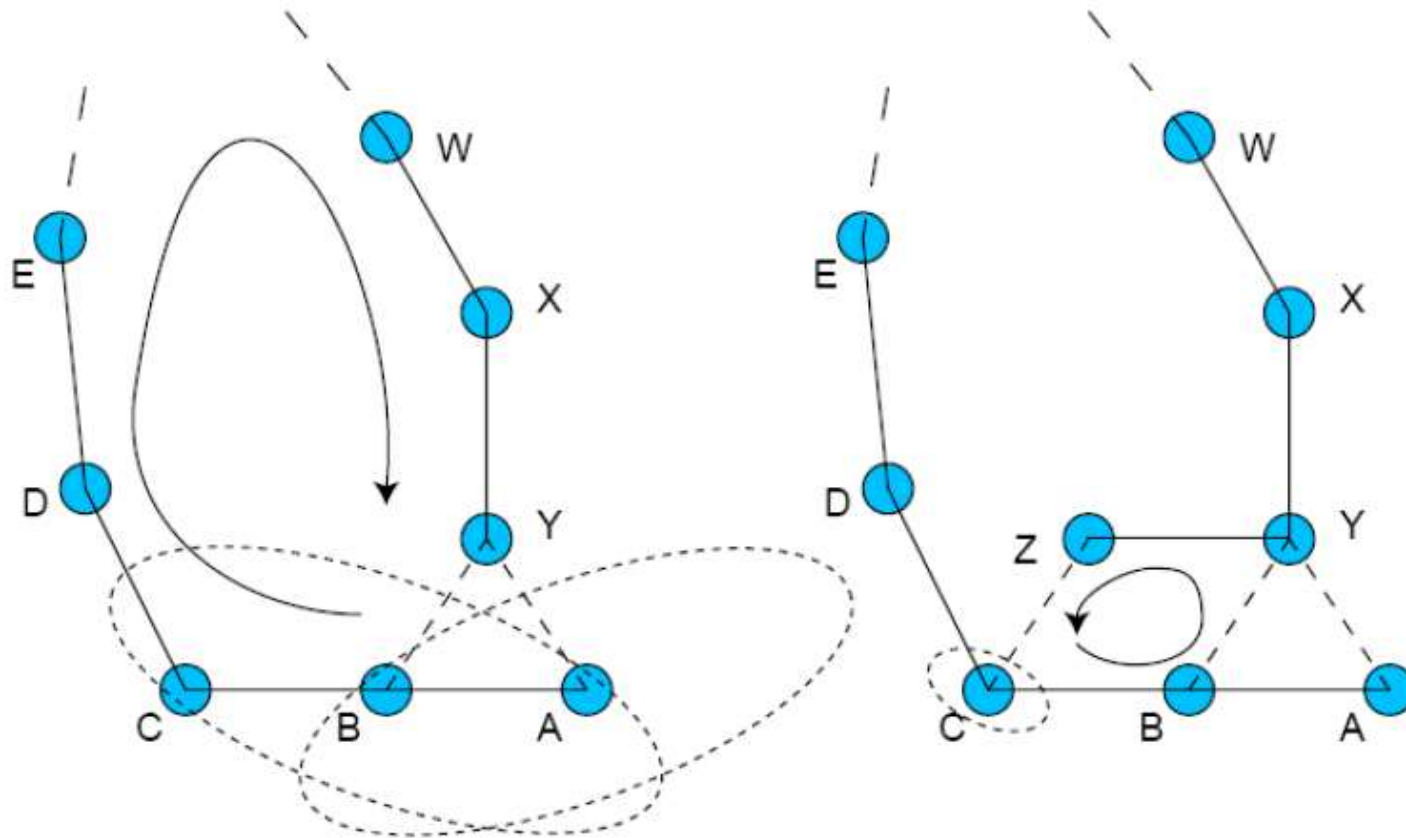
- Choose neither match
  - Pros, cons?
- Choose one match
  - Pros, cons?
- Choose both matches
  - Pros, cons?





# Deferred Loop Validation

- Continue SLAM until Z matches C
- Examine graph for *~identity cycle*



# Occupancy Grid Map (OGM)

- Maps the environment as a grid of cells
  - Cell sizes typically range from 5 to 50 cm
- Each cell holds a probability value that the cell is occupied in the range  $[0,100]$
- Unknown is indicated by -1
  - Usually unknown areas are areas that the robot sensors cannot detect (beyond obstacles)

# Occupancy Grid Map



White pixels represent free cells  
Black pixels represent occupied cells  
Gray pixels are in unknown state

# Occupancy Grid Maps

- Pros:
  - Simple representation
  - Speed
- Cons:
  - Not accurate - if an object falls inside a portion of a grid cell, the whole cell is marked occupied
  - Wasted space



# Maps in ROS

- Map files are stored as images, with a variety of common formats being supported (such as PNG, JPG, and PGM)
- Although color images can be used, they are converted to grayscale images before being interpreted by ROS
- Associated with each map is a YAML file that holds additional information about the map

# Map YAML File

```
image: map.pgm  
resolution: 0.050000  
origin: [-100.000000, -100.000000, 0.000000]  
negate: 0  
occupied_thresh: 0.65  
free_thresh: 0.196
```

**resolution:** Resolution of the map, meters / pixel

**origin:** 2D pose of the lower-left pixel as (x, y, yaw)

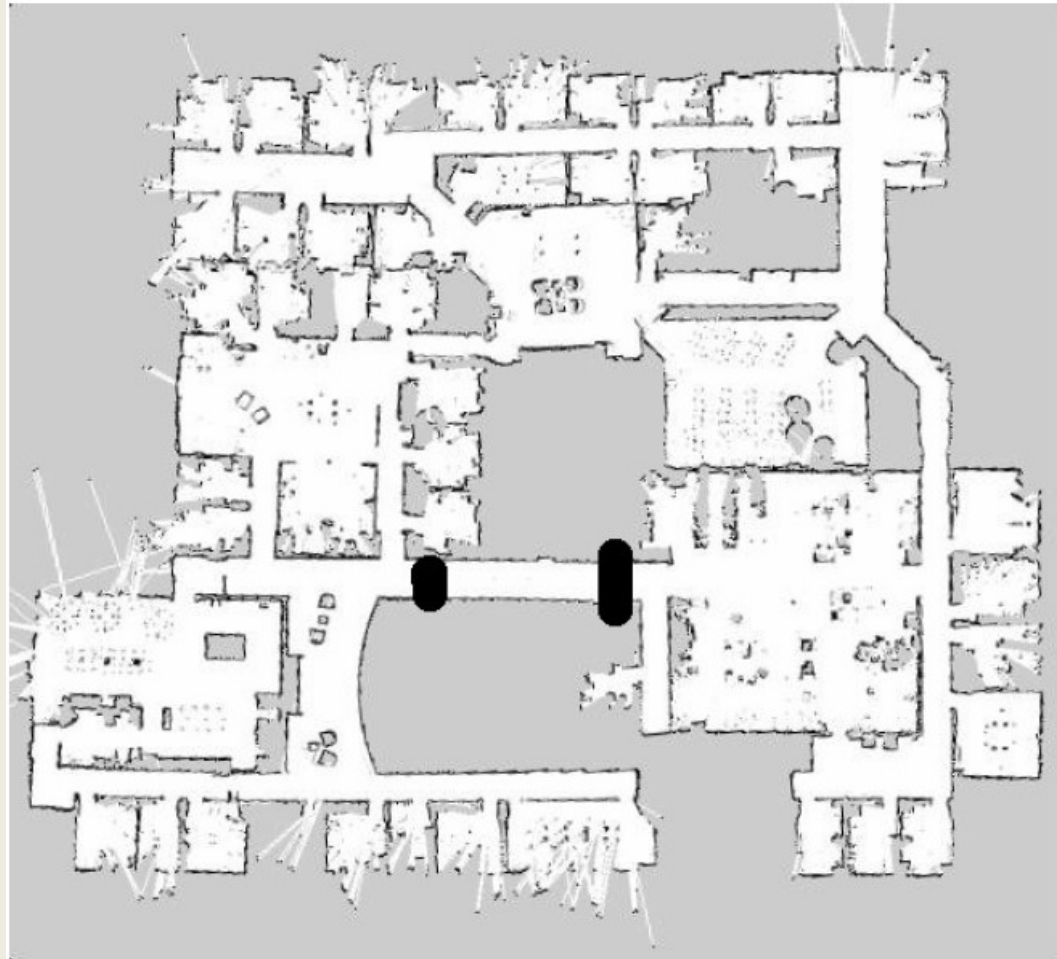
**occupied\_thresh:** Pixels with occupancy probability greater than this threshold are considered completely occupied

**free\_thresh:** Pixels with occupancy probability less than this threshold are considered completely free

# Editing Map Files

- Since maps are represented as image files, you can edit them in your favorite image editor
- This allows you to tidy up any maps that you create from sensor data, removing things that shouldn't be there, or adding in fake obstacles to influence path planning
- For example, you can stop the robot from planning paths through certain areas of the map by drawing a line across a corridor you don't want the robot to drive through

# Editing Map Files





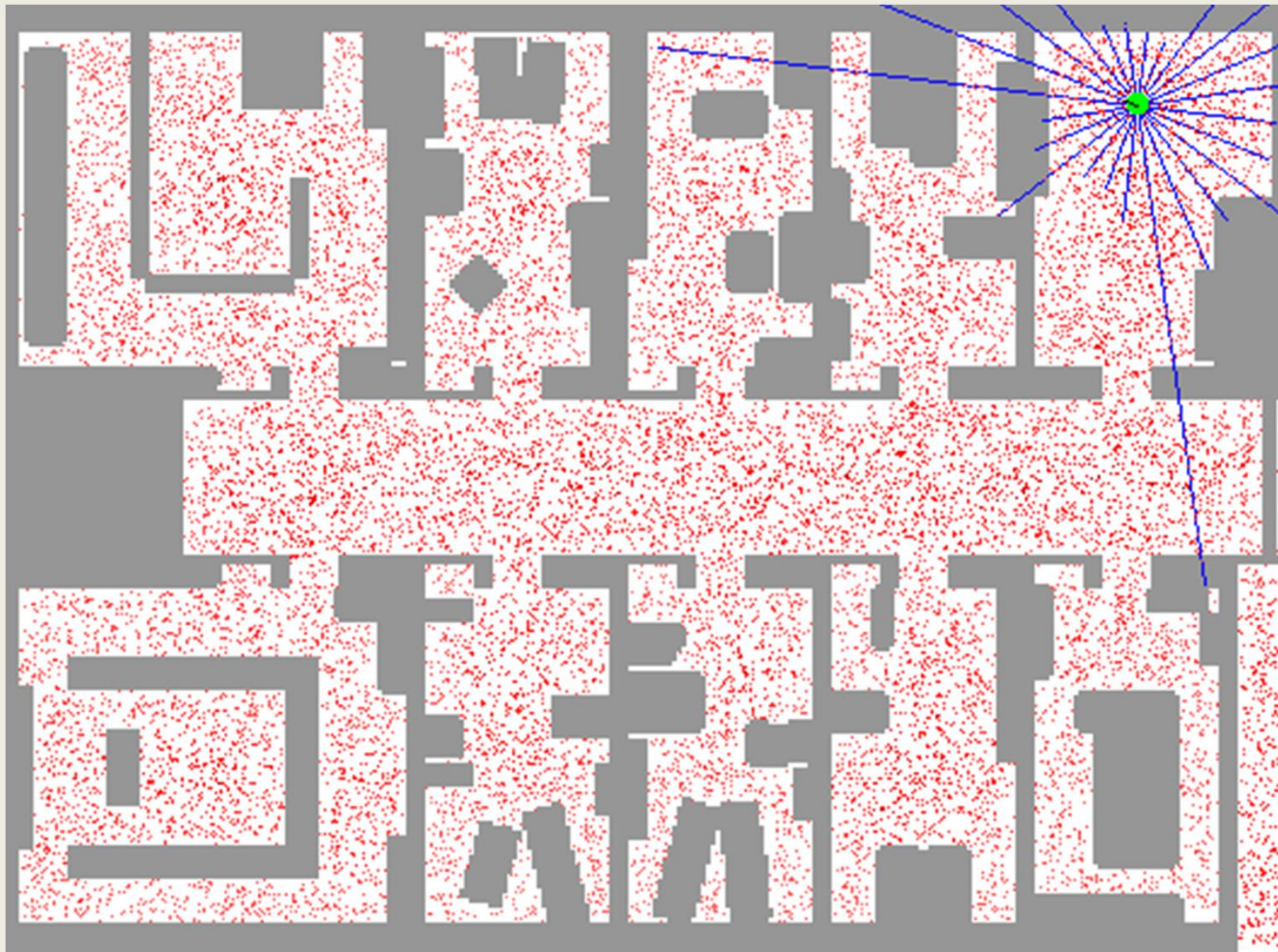
# SLAM

- **Simultaneous localization and mapping (SLAM)** is a technique used by robots to build up a map within an unknown environment while at the same time keeping track of their current location
- A chicken or egg problem: An unbiased map is needed for localization while an accurate pose estimate is needed to build that map

# Particle Filter – FastSLAM

- Represent probability distribution as a set of discrete particles which occupy the state space
- Main steps of the algorithm:
  - Start with a random distribution of particles
  - Compare particle's prediction of measurements with actual measurements
  - Assign each particle a weight depending on how well its estimate of the state agrees with the measurements
  - Randomly draw particles from previous distribution based on weights creating a new distribution
- **Efficient:** scales logarithmically with the number of landmarks in the map

# Particle Filter



# gmapping

- <http://wiki.ros.org/gmapping>
- The gmapping package provides laser-based SLAM as a ROS node called **slam\_gmapping**
- Uses the FastSLAM algorithm
- It takes the laser scans and the odometry and builds a 2D occupancy grid map
- It updates the map state while the robot moves
- [ROS with gmapping video](#)



# Gmapping vs RSLAM

- [Video](#)

# RSLAM - Ayasofya

- [Alt Kat](#)
- [Bahçe](#)
- [Üst Kat](#)

# Dinamik Ortamda Navigasyon+SLAM

- 5. Türkiye Gençlik Zirvesi