Navigation: Mapping

RSS Lecture

W March 11, 2009

Prof. Teller

Text: Siegwart and Nourbakhsh Ch. 5, 6

Dudek and Jenkin Ch. 8

Lecture Overview

- What are maps?
- Why are they important?
- Map types & design alternatives
- Fusing observations
- Example mapping robots

What are maps?

- Collection of elements or features at some scale of interest, and a representation of the geometric and/or topological relationships among them
- Also *semantic information (metadata)*
  - Segmentation, place/object naming, function, etc.
- We will focus on geometry and topology

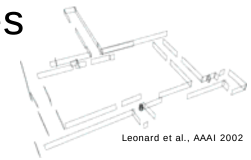
  But *semantics* are critical to real-world applications!

Why maps? From where?

- Essential for a wide variety of human, robotic activities (localization, planning)
- Maps are highly labor-intensive to create:
  - Exploration (global coverage)
  - Measurement (local coverage)
  - Validity (correctness, error bounds)
  - Currency (freshness)
  - As-planned vs. as-built building models
  - Not to mention metadata/semantics …
- Map creation is an ideal robotics task!
  - Achieving robust, sustained, large-area autonomous mapping capability has been an “open” (i.e., unsolved) problem for decades

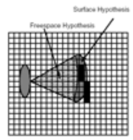
Some robot map types

- Continuous / “vector” format
  - Points, linear or curved segments, surface patches

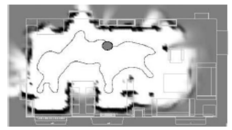


Leonard et al., AAAI 2002

- Discrete / “raster” format
  - Occupancy grids

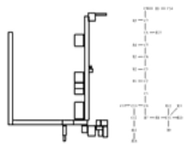


Konolige

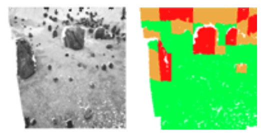


Chatila, SSS 2004

- Metrical / Topological
- Global / Local



Metrical / Topological



Chatila, SSS 2004

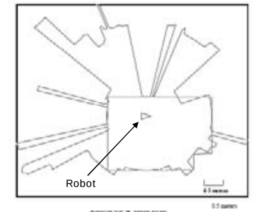
Local, Metrical, Qualitative

- Hybrid

Common range sensors

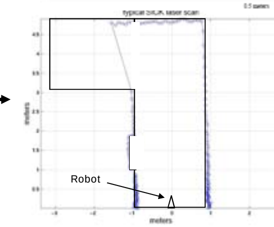
Polaroid sonar ring
12 range returns, one per 30 degrees, at ~4 Hz

→ (+ servoed rotation)



SICK laser scanner
180 range returns, one per degree, at 5-75 Hz

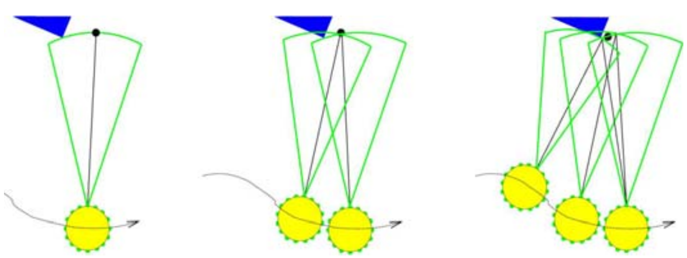
→



Other possibilities: Stereo/monocular vision; Robot body (e.g. stall, bump sensing)

Fusing multiple returns

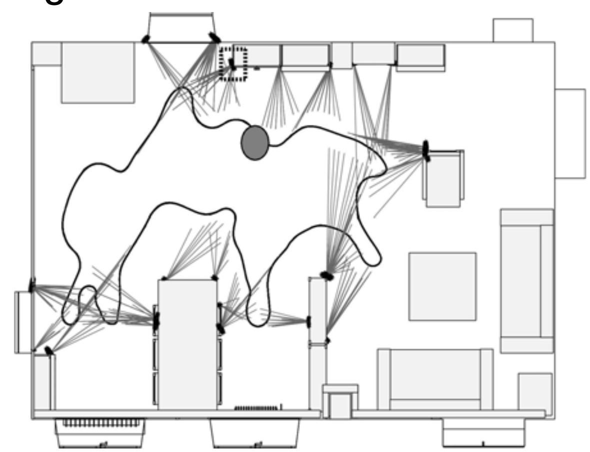
- Crucial assumption: pose estimation (e.g., odometry, dead reckoning) is accurate over short times and distances



Wijk 2001

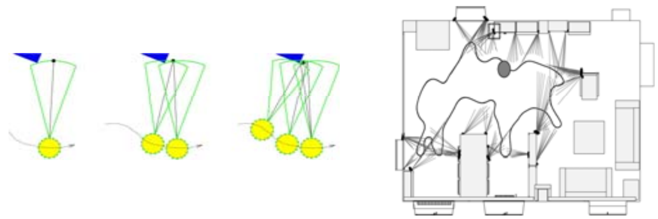
- Can then localize features using conventional triangulation (sonar beam width complicates things)

Fusing data with motion



Wijk 2001

Local vs. global data fusion



- Crucial assumption: that robot can solve strong localization (global pose estimation) throughout
- This is a very difficult problem without a map! (It's even difficult *with* a map or partial map.)
- SLAM: Simultaneous Localization and Mapping
- For now, we assume localization (SLAM covered later in RSS)

Representation considerations

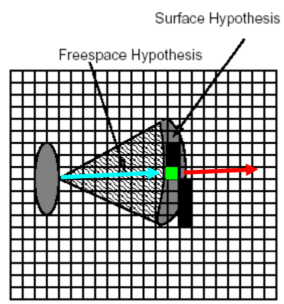
- We want our robot to be able to plan and execute high-level motions amongst obstacles

- What do we want from our map?
  - Consistent global, or locally metrical, coordinate system
  - Identification and localization of substantial *features,* e.g., obstacles that may hinder or damage the robot
  - All of this should be well-defined and computationally accessible (data model, data structure, API)
  - Scalability (reasonable search, access times as exploration continues, and map gets really large)

- … Is that all we need/want from a map?

Alternative 1: Discretize

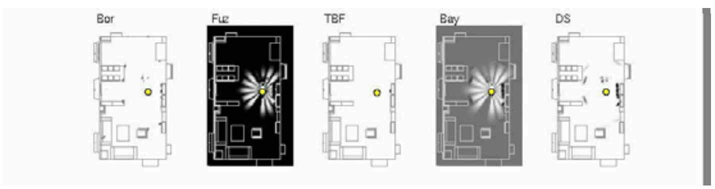
- *Occupancy grid* of *cells*
  - Regular subdivision of region
  - Models free & occupied space
- Cells accumulate *evidence* of presence of obstacle surface
- Grid is updated on-line with recent measurements
- Range return from obstacle implies three grid intervals:
  - From robot to obstacle (FS)
  - At (quantized) obstacle depth
  - Beyond obstacle (from robot's point of view)



Konolige

Many occupancy grid methods

- Example: sonar data, varying update rules
  - White: free-space; black: obstacle; grey: unknown



Wijk 2001

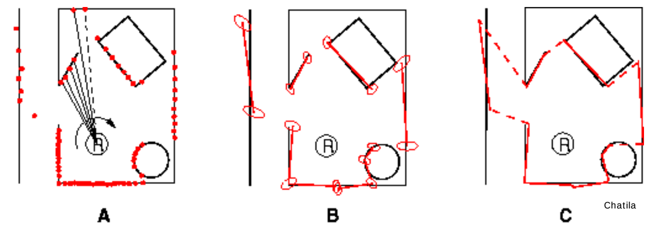
Bor: Histogramic (Borenstein 1991); accumulates hits
Fuz: Fuzzy (Zadeh 1973; Ribo and Pinz 1999); with weights
TBF: Triangulation-Based Fusion (Wijk 2000); local triangulation
Bay: Bayesian (Elfes 1988); probabilistic occupancy/emptiness
DS: Dempster-Shafer (Shafer 1976; Pagac 1996); with "ignorance"

Pitfalls of occupancy grids

- Quantization error
  - Cells too large: not faithful to environment or robot task
  - Cells too small: too numerous (expensive) to process efficiently
  - Task-dependent: grid size can be at once too small and too large!
- Blurring
  - Caused by pose estimation error, sensor uncertainty, grid quantization

Alternative 2: Line Features

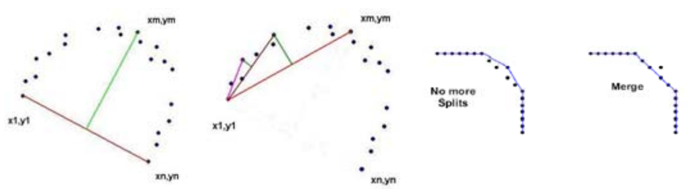
- Piecewise linear approximation of sequence of point features (i.e., ranges)



- How are individual ranges, point features grouped into useable line segments?
- How to counteract noise inherent in data?

Split, Merge, Fit algorithm

- Used for *ordered sets* of laser or sonar returns
- Takes two thresholds: split distance, merge angle
- Split phase:
  - Recursively split until (max) distance criterion is met
- Merge phase:
  - Merge adjacent segments until (min) angle criterion is met
- Fit phase (perhaps with outlier classification):
  - Fit line segments to resulting (noisy) point sequences



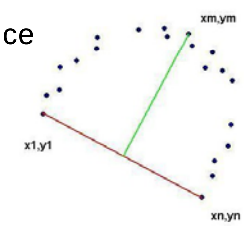
Split phase

- Point list: $P = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$
- Split into two subsets:

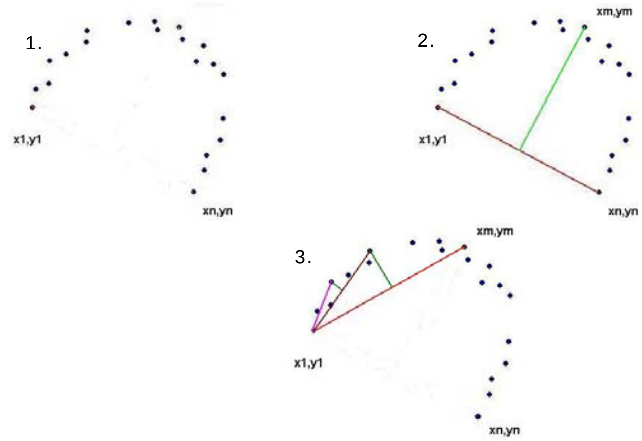
$$P' = \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$$

$$P'' = \{(x_m, y_m), (x_{m+1}, y_{m+1}), \ldots, (x_n, y_n)\}$$

- (x_m, y_m): point of max distance to line $L = \{(x_1, y_1), (x_n, y_n)\}$

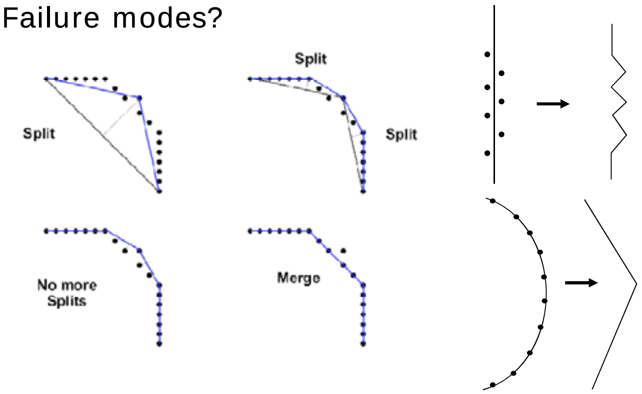


Splitting is recursive



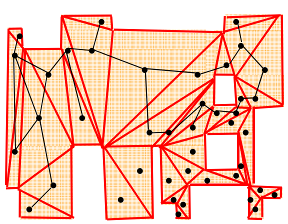
Segment merging phase

- Merge adjacent segments if nearly collinear
- Failure modes?



Storing extracted features

- Store as linear list
  - Advantage: very simple. Drawbacks: ?
- Or, store in *proximity data structure*
  - E.g., constrained Delaunay triangulation



- CDT has many nice properties:
  - Linear size; logarithmic search; temporal coherence; maximum minimum angle; dual to Voronoi diagram; etc.

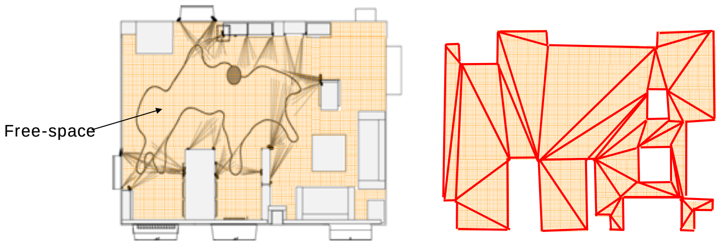
Alternative 3: Free-space Map

- Robot spends its time well away from obstacles

- Call this area "free-space," *i.e., the region through which the robot can expect to be free to move*
- The *complement* of the *union* of all *obstacles*

Free-space complexity

- It's empty… but that doesn't mean it is easy to represent! What is *descriptive complexity* of FS?



- Free-space is *more complex* than obstacle union n
  - 2D simple polygon (no holes):
  - 2D segments:
  - 3D polyhedron:

Task dependence

- Representation depends on *task*
  - Which sensors are available?
  - Type(s) of output models desired?
  - Scale/extent of region to be mapped?
  - Coarse-grained or fine-grained?
  - Low or high spatial dynamic range?

Mapping summary

- Maps are critical to many tasks
- Assumed localization for now
- Saw several map representations, data fusion algorithms
- Considered scaling requirements

Line fitting

- Input: n unordered points $(x_i, y_i), i = 1..n$
- Output: Best-fit line $x\cos\alpha + y\sin\alpha - d = 0$ where:

$$\bar{x} = \frac{1}{n}\sum_i x_i \quad \bar{y} = \frac{1}{n}\sum_i y_i \quad \text{(means)}$$

$$\mu_{xx} = \sum_i (x_i - \bar{x})^2$$

(variances)

$$\mu_{yy} = \sum_i (y_i - \bar{y})^2$$

$$\mu_{xy} = \sum_i (x_i - \bar{x})(y_i - \bar{y}) \quad \text{(covariance)}$$

$$\tan\alpha = \frac{2\mu_{xy}}{\mu_{xx} - \mu_{yy} - ((\mu_{xx} - \mu_{yy})^2 + 4\mu_{xy}^2)^{1/2}} \quad \text{(major axis of best-fit ellipse)}$$