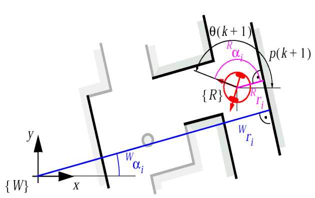
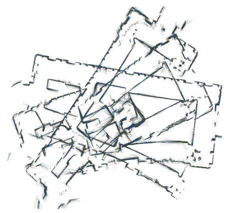
Probabilistic Localization 2 Mapping





Some slides adapted from Cyrill Stachniss, with many thanks
Check out his (instructional) YouTube videos!

Today's Class: Navigation

2

- **Localization:** estimating the robot'sa location in an environment
- **Mapping:** creating a map of the environment
- **SLAM:** simultaneously building a map and locating the robot in it
- So far we've focused on localization

Today's Class

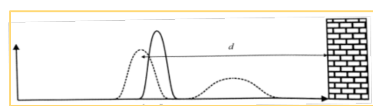
3

- Probabilistic Localization
  - Core ideas
  - The basics of Markov localization, Kalman filters (a little)
- Mapping
- SLAM
- Upcoming office hours
  - Today, 12-1
  - Tomorrow, 10-11
  - Unfortunately, I have to run after this

Probabilistic Localization: Idea

4

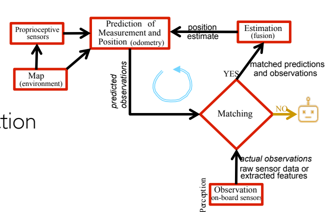
- State estimate (belief in current position) represented as a probability distribution
- Ex: robot moving in a straight line
- Over time, position becomes less certain
  - Distribution gets "wider"
- When the environment (distance from wall) is observed, belief state gets less uncertain



State Estimation

5

- Probabilistic localization:
  1. Predict state (location) from internal sensors
  2. Observe environment with external sensors
  3. Fuse prediction and observation
  4. Estimate current state
- Approaches:
  1. Markov Localization
  2. Kalman Filter Localization



Kinds of Localization Problems

6

- Position tracking:
  - Known start state
  - Relatively high precision (low error)
  - Usually normal distribution
- Global localization:
  - Start state can be anywhere/unknown
  - Usually initially uniform
- "Kidnapped robot" problem:
  - Robot "gets moved" to another location (includes losing track of position)
  - Key question: does it know this happened?

Markov vs. Kalman Filter Localization

8

Markov Localization

- Localization starting from any unknown position
  - ⇒ Recovers from ambiguous situations
- Update probability of all positions in whole state space at each time
- Uses discretized map representation
  - Cell size matters!
  - Much determined by computational feasibility

Kalman Filter Localization

- Tracks robot as it navigates
- Inherently precise, efficient
- Can work in continuous maps
- Can't recover from ambiguous situations
  - If uncertainty becomes too large, Kalman filter will fail and the position is definitively lost
  - For example, colliding with something

Markov Localization: Idea

9

- Markov localization: explicit, discrete representation for probability of each position in the state space
- Usually represents environment as:
  - Grid or topological graph
  - ⇒ …with finite number of possible states (positions)
- During *each* update, the probability for *each* state (element) of the entire space is updated
  - This can get expensive!

Markov Localization: Probability

10

- Prior probability: probability distribution describing likelihood of random variable x having some value before observations
  - $p(x)$ *before* observation y
  - $p(l)$ = probability of robot being in a particular place l
- Posterior probability: belief that x is some value after observing y
  - $p(l \mid i)$ = probability of robot being in l, given observation i
- Divided by some normalization constant $p(y)$
  - Which is constant, so we assign it a label and forget about it

Markov Localization (2)

11

- P(A): Probability that A is true

  $P(r_t = l)$: probability that robot r is at position l at time t
- Estimate probability robot is at each position given
  - Actions and
  - Sensor readings
- P(A | B): Conditional probability of A, given B

  $P(r_t = l \mid i_t)$: probability robot is at position l given sensor input i_t

Markov Localization (3)

14

- Law of total probability:

$$p(x)\sum_y p(x \mid y)p(y) \quad \text{discrete}$$

$$p(x)\int_y p(x \mid y)p(y)\,dy \quad \text{continuous}$$

- Bayes rule:

$$p(A|B) = \frac{p(B|A)p(A)}{p(B)}$$

Markov Localization (4)

15

- Bayes rule: $p(A|B) = \frac{p(B|A)p(A)}{p(B)}$
- Map from belief state + sensor input to new belief state:

$$p(l|i) = \frac{p(i|l)p(l)}{p(i)}$$

- $p(l)$: belief state before perceptual update process
- $p(i \mid l)$: probability of getting measurement i at pos. l
  - Consult map, identify probability of sensor i reading for every possible position
- $p(i)$: normalization factor (so sum over all l in L = 1)

Markov Localization (5)

16

- Map from belief state + sensor input to new belief state:

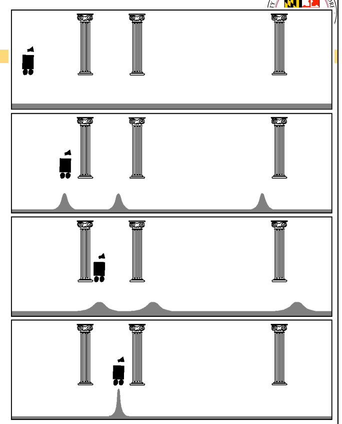
$$p(l_t|o_t) = \int p(l_t|l'_{t-1}, o_t)p(l'_{t-1})dl'_{t-1}$$

- Summing over all possible ways in which the robot may have reached *l*.
- Markov assumption: Update depends only on previous state and most recent actions and percepts.
  - Not usually a true assumption, but usually close enough

Beliefs/Time

17

- Improving belief state by moving
  - Given knowledge of the environment
- Each external measurement ***i*** updates probability distribution
- Example: →



Markov Localization in Grid

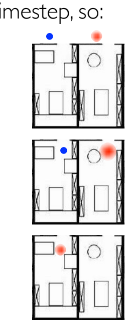
18

1. Start: No knowledge, so use uniform probability distribution
2. Robot perceives a pillar: Equal probability of being at 1, 2 or 3
3. Robot moves: Action model gives estimate of new probability distribution based on previous estimate + motion.
4. Robot perceives second pillar: Probability of being at pillar 2 becomes dominant

Markov Localization: Pros

33

- Considering all possible locations at each timestep, so:
- Doesn't need known start point
- Resistant to "kidnapping" / errors fixable
  - Example:
  - If you're here, but estimate you're here…
  - …sensor evidence accumulates until…
  - …this is the new best estimate
- Handles position tracking, global localization, and kidnapping



Markov Localization: Cons

34

- Considering all possible locations at each timestep, so:
- Computationally expensive
  - Practical result: discretized maps only
- Always evaluating some probability of being in every wrong place
- Minimizes use of observation history

less precise

- Can "jump" to the wrong place, too

Kalman Filter Localization

35

- Markov localization can represent any probability function over robot's location
  - This is **general** and **powerful**
  - This is **imprecise**
- Do we really need a completely arbitrary probability function for position?
  - Can we use our sensors better?
  - Can we use our prior knowledge better?
- Kalman filters use *all* information to estimate position

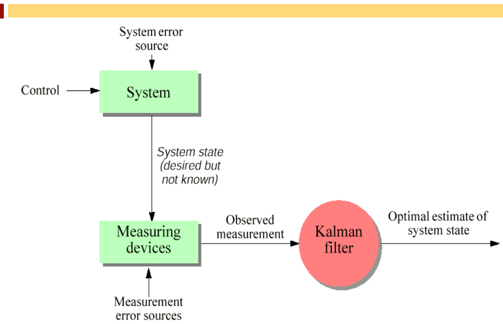
Kalman Filters

36

- Inputs to system:
  - Control signal
  - System error model (uncertainty in dynamics)
    - That is, motion error
- All sensors produce measurements
  - With some sensor error
- Kalman filter fuses sensor measurements with system knowledge

Kalman Filter Localization

37



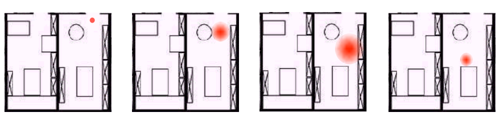
Reminder…

38

- Mobile robot moving in known environment
  1. Starts from known location
  2. Keeps track (maintains a belief) using odometry
  3. Over time, uncertainty about position grows
  - → Observe environment for new estimate
  4. Fuse estimate with odometric estimation to give…
  5. Best possible belief update

Prediction (or action) update

Perception (or measurement, or correction) update



Kalman Filter Localization: Idea

39

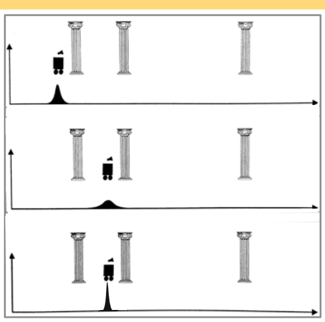
- Kalman filters are a special case of Markov localization
- Instead of arbitrary probability function describing state, use a Gaussian function
- Very efficient to update (only mean and variance change)
- But initial belief must also be a Gaussian
- Start state must be known
  - With *some* precision
- Can't recover if it gets lost

position tracking only (no global localization, no kidnapping)

Beliefs/Time

40

1. Start: Fairly precise Gaussian for x
2. Robot moves: Action model gives estimate of new probability distribution
   1. Variance increases
   2. Gaussian becomes "wider"
3. Robot perceives a pillar: New probability distribution based on previous estimate + motion (action model)



The Core Idea*

41

- Treat localization as a sensor fusion problem
- Robot's sensory input (observations) treated as a set of features that relate to objects in the environment
- Kalman filter fuses distance estimate from each feature to an object in the map
- Let's look at how that works for the localization cycle

act → estimate → observe → update estimate → act →…

** We will not break down the theory behind Kalman filters in class; SNS pg. 325–342 do so, with examples.*

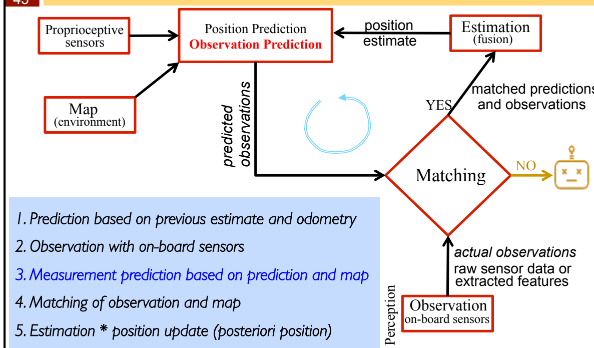
KF Localization Cycle

42

1. Position estimation: estimate position from odometry
2. Measurement prediction: predict what features the robot expects to see at estimated position
   - Given where I (think I) am, what should I see?
3. Observation step: Collect percepts & extract features
   - Lines/doors/sensor values/… } map representation elements
4. Matching: Find best match between observed and expected features (for each feature)
5. Estimation step: fuse matches to get updated belief

Map-Based Localization

43



Kalman Filter Localization

44

- Position estimate as probability distribution
- Gaussian distribution
  - Only considering estimates "around" a single belief
    - Single hypothesis belief state
  - Update step = update mean and variance of initial Gaussian
  - Need approximate starting position
- Precise, efficient, works in continuous environments
- Use sensor fusion to combine estimated observations with actual observations

A Few More Ideas

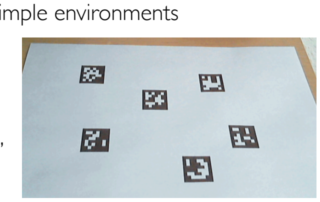
63

- Other kinds of probabilistic localization exist
  - Best source: Probabilistic robotics. MIT press, 2005. Sebastian Thrun, Wolfram Burgard, and Dieter Fox.
- Most other approaches instrument the environment
  - Markers or beacons
- Feasible in, e.g., some factories, the CSEE building
- Infeasible if the robot moves around a lot

Artificial Landmarks

64

- Goal: globally unique navigation
  - Any possible sensor reading uniquely identifies position
- Feasible with certain simple environments
  - Markers can give a lot of information
  - ArUco markers are unique, give orientation
- Less workable in "real" environments

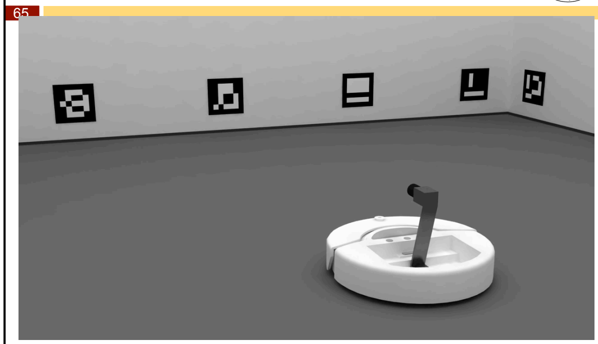


ArUco markers on surface

github.com/ziox/leonardo/wiki/Relazione-(originale)
docs.opencv.org/3.4.0/d5/dae/tutorial_aruco_detection.html

Using ArUco Markers

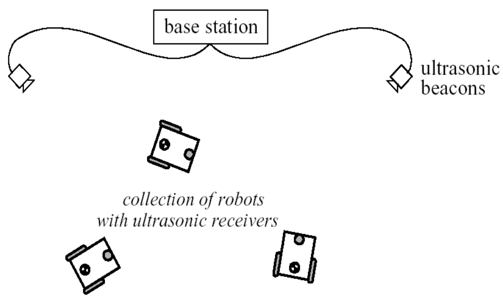
65



www.youtube.com/watch?v=VsIMl8O_F1w
www.youtube.com/watch?v=Q1HIJEjW_j0

Beacon Systems: Triangulation

68



Active Beacons: Triangulation

69

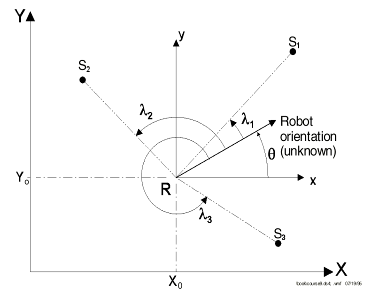
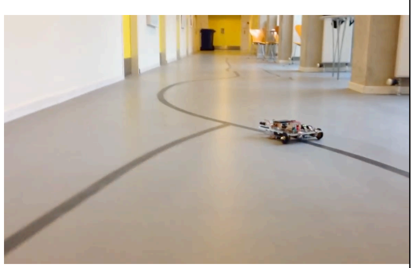


Figure 6.1: The basic triangulation problem: a rotating sensor head measures the three angles λ_1, λ_2, and λ_3 between the vehicle's longitudinal axes and the three sources S_1, S_2, and S_3.

Route-Based Localization

70

- Annotate exact route
- Not very consistent with autonomy
- Works for, e.g., factories

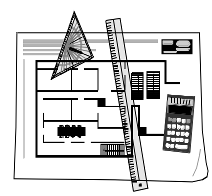


www.youtube.com/watch?v=Cf-V-giXiRw

How to Get a Map?

76

- By hand?
  - Slow, expensive and imprecise
- Automatically: Mapping
  - The robot learns its environment
- Can cope with dynamically changing environment



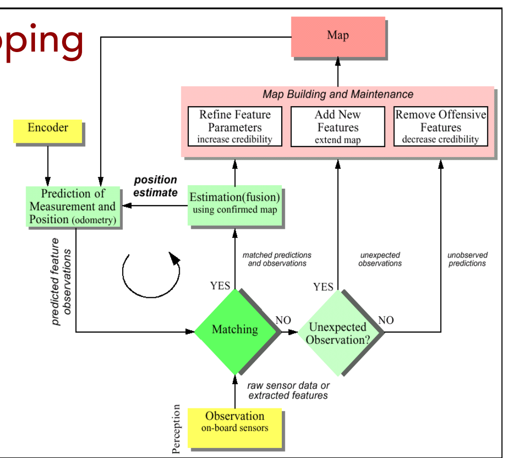
- Map is built from sensors that will be used to navigate

Mapping

77

- Basic requirements:
- A way to incorporate sensed information into world model
- Sufficient information for navigation tasks
  - Estimating position, path planning, obstacle avoidance, …
- Correctness
- Predictability
  - Most environments are a mixture of predictable and unpredictable features

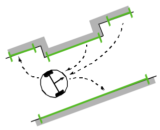
Mapping



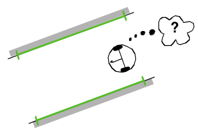
Challenges: Maintenance

84

- Map Maintenance: keeping track of changes in the environment



example: disappearing cupboard

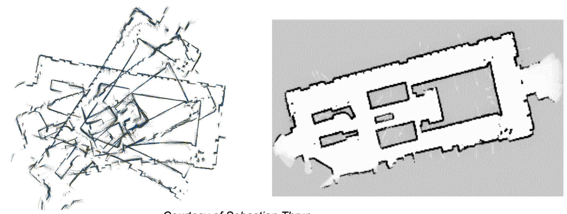


- Often uses a *measure of belief* of each feature
- If localizing: did the world change, or are you lost?

Challenges: Cyclic Environments

85

- Small local error accumulates
- This is usually irrelevant for navigation
- However, when closing loops, global error does matter

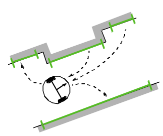


Courtesy of Sebastian Thrun

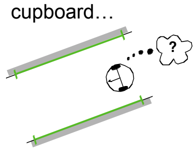
Dynamic Environments

86

- Dynamic changes require continuous (re-)mapping
- High-level features help if you can get them
  - Humans are more dynamic map features than walls
  - Can also learn that certain things are dynamic over time



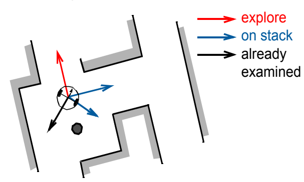
navigate around cupboard…



Exploration & Graph Construction

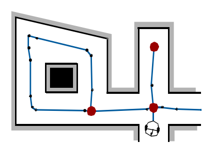
87

1. Exploration



- provides correct topology
- must recognize already visited location
- backtracking for unexplored openings

2. Graph Construction



Where to put the nodes?

- Topology-based: at distinctive locations
- Metric-based: where features disappear or become visible

SLAM

88

- Simultaneous Localization and Mapping
  - A map is needed for localizing a robot
  - A pose estimate is needed to build a map
- Hard to decouple these problems!
  - It can be done, with certain assumptions.
- Several approaches exist
  - Extended Kalman filter (EKF): similar to Kalman filter, but state vector (being estimated) includes position of map features
  - Graph-based SLAM
  - Particle filter-based SLAM

Applications

90

- SLAM is central to a range of indoor, outdoor, in-air and underwater applications for both manned and autonomous vehicles.

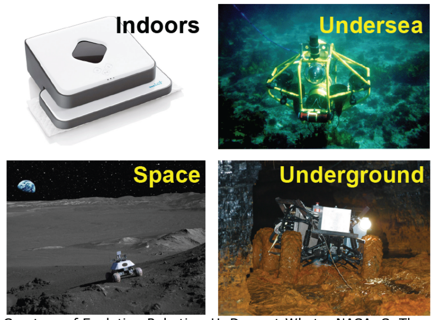
Examples:

- At home: vacuum cleaner, lawn mower
- Air: surveillance with unmanned air vehicles
- Underwater: reef monitoring
- Underground: exploration of mines
- Space: terrain mapping for localization

Cyrill Stachniss @ Freiburg

Applications Everywhere…

91



Courtesy of Evolution Robotics, H. Durrant-Whyte, NASA, S. Thrun

Cyrill Stachniss @ Freiburg

Full vs. Online SLAM

92

- Full SLAM estimates the entire path

$$p(x_{0:T}, m \mid z_{1:T}, u_{1:T})$$

- Online SLAM only wants to find the robot's most recent location (pose)

$$p(x_t, m \mid z_{1:t}, u_{1:t})$$

Cyrill Stachniss @ Freiburg

SLAM Definitions

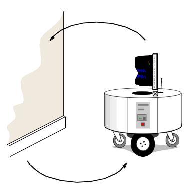
93

Given

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

Wanted

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



Cyrill Stachniss @ Freiburg

Summary

103

- Mapping: modeling the environment
- Localization: estimating the robot's pose
- SLAM: simultaneously doing the above
- Full SLAM vs online SLAM
- Everything is probabilistic!