ME 598: Introduction to Robotics

Lecture 10: Localization, Path Planning, & Navigation

Stevens Institute of Technology
Dr. Mishah U. Salman
Fall 2013

Date:
By:

Slides adapted from Dr. David J. Cappelleri,
© R. Siegwart, ETH Zurich – ASL, http://www.mobilerobots.ethz.ch/

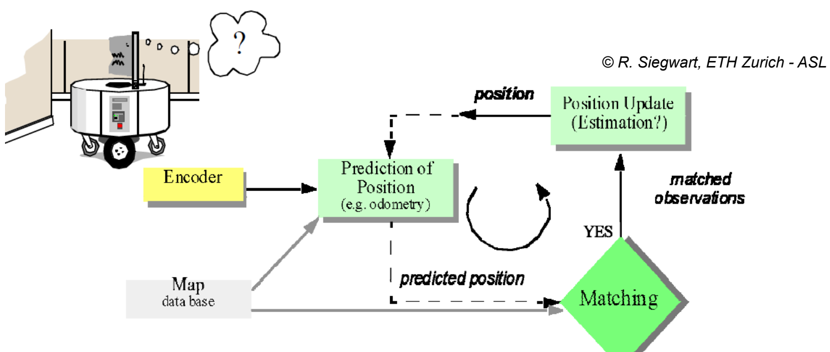
STEVENS INSTITUTE of TECHNOLOGY THE INNOVATION UNIVERSITY

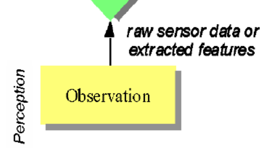
©2011 Stevens Institute of Technology

Localization, Path Planning, & Navigation

Localization, Path Planning, & Navigation: Localization- Where am I?

© R. Siegwart, ETH Zurich - ASL





- *Odometry, Dead Reckoning*
- *Localization based on external sensors, beacons or landmarks*
- *Probabilistic Map Based Localization*

Localization, Path Planning, & Navigation: Localization- Challenges

- Knowing the absolute position (e.g. GPS) is not sufficient
- Localization in human-scale in relation with environment
- Planning in the *Cognition* step requires more than only position as input
- Perception and motion plays an important role
  - Sensor noise
  - Sensor aliasing
  - Effector noise
  - Odometric position estimation

© R. Siegwart, ETH Zurich - ASL

Localization, Path Planning, & Navigation: Sensor Noise

- Sensor noise is mainly influenced by environment
  e.g. surface, illumination …
- or by the measurement principle itself
  e.g. interference between ultrasonic sensors
- Sensor noise drastically reduces the useful information of sensor readings. The solution is:
  - to take multiple readings into account
  - employ temporal and/or multi-sensor fusion

© R. Siegwart, ETH Zurich - ASL

Localization, Path Planning, & Navigation: Sensor Aliasing

- In robots, non-uniqueness of sensors readings is the norm
- Even with multiple sensors, there is a many-to-one mapping from environmental states to robot's perceptual inputs
- Therefore the amount of information perceived by the sensors is generally insufficient to identify the robot's position from a single reading
  - Robot's localization is usually based on a series of readings
  - Sufficient information is recovered by the robot over time

© R. Siegwart, ETH Zurich - ASL

Localization, Path Planning, & Navigation: Effector Noise- Odometry, Deduced Reckoning

- Odometry and dead reckoning:
  Position update is based on proprioceptive sensors
  - Odometry: wheel sensors only
  - Dead reckoning: also heading sensors
- The movement of the robot, sensed with wheel encoders and/or heading sensors is integrated to the position.
  - Pros: Straight forward, easy
  - Cons: Errors are integrated -> unbound
- Using additional heading sensors (e.g. gyroscope) might help to reduce the cumulated errors, but the main problems remain the same.

© R. Siegwart, ETH Zurich - ASL

Localization, Path Planning, & Navigation: Odometry- Error Sources

deterministic (systematic) ⟷ non-deterministic (non-systematic)

- deterministic errors can be eliminated by proper calibration of the system.
- non-deterministic errors have to be described by error models and will always lead to uncertain position estimate.

- Major Error Sources:
  - Limited resolution during integration (time increments, measurement resolution)
  - Misalignment of the wheels (deterministic)
  - Unequal wheel diameter (deterministic)
  - Variation in the contact point of the wheel
  - Unequal floor contact (slipping, not planar …)

© R. Siegwart, ETH Zurich - ASL

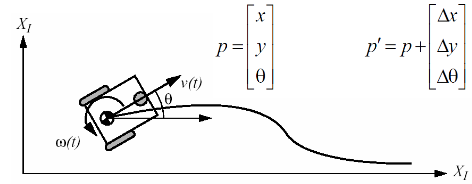
Localization, Path Planning, & Navigation: Odometry- Classification of Integration Errors

- Range error: integrated path length (distance) of the robots movement
  - sum of the wheel movements
- Turn error: similar to range error, but for turns
  - difference of the wheel motions
- Drift error: difference in the error of the wheels leads to an error in the robots angular orientation
- **Over long periods of time, turn and drift errors** far outweigh range errors!
  - Consider moving forward on a straight line along the x axis. The error in the y-position introduced by a move of d meters will have a component of dsinDq, which can be quite large as the angular error Dq grows.

© R. Siegwart, ETH Zurich - ASL

Localization, Path Planning, & Navigation: Odometry- Differential Drive Robot

- Kinematics



$$p=\begin{bmatrix}x\\y\\\theta\end{bmatrix}\qquad p'=p+\begin{bmatrix}\Delta x\\\Delta y\\\Delta\theta\end{bmatrix}$$

$\Delta s_r;\Delta s_l$ = travled distances for right and left wheel

b = distance between two wheels on robot

$$\Delta x = \Delta s\cos(\theta+\Delta\theta/2)$$

$$\Delta y = \Delta s\sin(\theta+\Delta\theta/2)$$

$$\Delta\theta = \frac{\Delta s_r-\Delta s_l}{b}$$

$$\Delta s = \frac{\Delta s_r+\Delta s_l}{2}$$

$$p' = f(x,y,\theta,\Delta s_r,\Delta s_l) = \begin{bmatrix}x\\y\\\theta\end{bmatrix}+\begin{bmatrix}\frac{\Delta s_r+\Delta s_l}{2}\cos\left(\theta+\frac{\Delta s_r-\Delta s_l}{2b}\right)\\ \frac{\Delta s_r+\Delta s_l}{2}\sin\left(\theta+\frac{\Delta s_r-\Delta s_l}{2b}\right)\\ \frac{\Delta s_r-\Delta s_l}{b}\end{bmatrix}$$

© R. Siegwart, ETH Zurich - ASL

Localization, Path Planning, & Navigation: Odometry- Differential Drive Robot

- Error model
  - Assumptions:
    - Errors of individual wheels are independent
    - Variance of wheel errors are proportional to absolute value of traveled distance

$$p' = \begin{bmatrix}x\\y\\\theta\end{bmatrix}+\begin{bmatrix}\frac{\Delta s_r+\Delta s_l}{2}\cos\left(\theta+\frac{\Delta s_r-\Delta s_l}{2b}\right)\\ \frac{\Delta s_r+\Delta s_l}{2}\sin\left(\theta+\frac{\Delta s_r-\Delta s_l}{2b}\right)\\ \frac{\Delta s_r-\Delta s_l}{b}\end{bmatrix}$$

(Sections 4.2 and 5.2.4)

$$\Sigma_\Delta = covar(\Delta s_r,\Delta s_l) = \begin{bmatrix}k_r|\Delta s_r| & 0\\ 0 & k_l|\Delta s_l|\end{bmatrix}$$

Known initial conditions

$$\Sigma_{p'} = \nabla_p f\cdot\Sigma_p\cdot\nabla_p f^T+\nabla_{\Delta_{rl}} f\cdot\Sigma_\Delta\cdot\nabla_{\Delta_{rl}} f^T$$

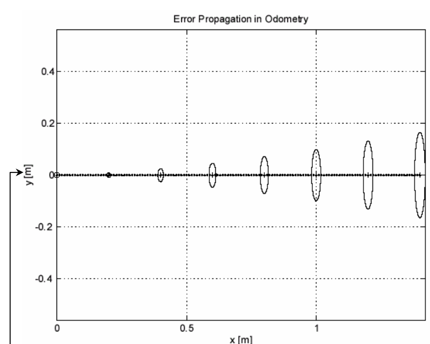
$$F_p=\nabla_p f=\nabla_p(f^T)=\begin{bmatrix}\frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} & \frac{\partial f}{\partial \theta}\end{bmatrix}=\begin{bmatrix}1 & 0 & -\Delta s\sin(\theta+\Delta\theta/2)\\0 & 1 & \Delta s\cos(\theta+\Delta\theta/2)\\0 & 0 & 1\end{bmatrix}$$

$$F_{\Delta_{rl}}=\begin{bmatrix}\frac{1}{2}\cos\left(\theta+\frac{\Delta\theta}{2}\right)-\frac{\Delta s}{2b}\sin\left(\theta+\frac{\Delta\theta}{2}\right) & \frac{1}{2}\cos\left(\theta+\frac{\Delta\theta}{2}\right)+\frac{\Delta s}{2b}\sin\left(\theta+\frac{\Delta\theta}{2}\right)\\ \frac{1}{2}\sin\left(\theta+\frac{\Delta\theta}{2}\right)+\frac{\Delta s}{2b}\cos\left(\theta+\frac{\Delta\theta}{2}\right) & \frac{1}{2}\sin\left(\theta+\frac{\Delta\theta}{2}\right)-\frac{\Delta s}{2b}\cos\left(\theta+\frac{\Delta\theta}{2}\right)\\ \frac{1}{b} & -\frac{1}{b}\end{bmatrix}$$

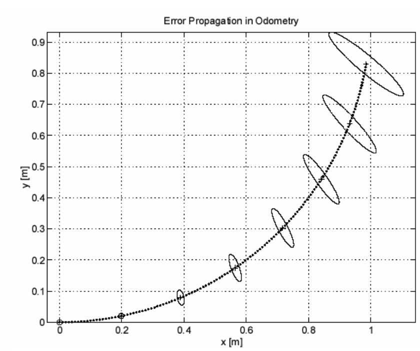
Localization, Path Planning, & Navigation: Odometry- Growth of Pose Uncertainty

© R. Siegwart, ETH Zurich - ASL

Straight Line Movement



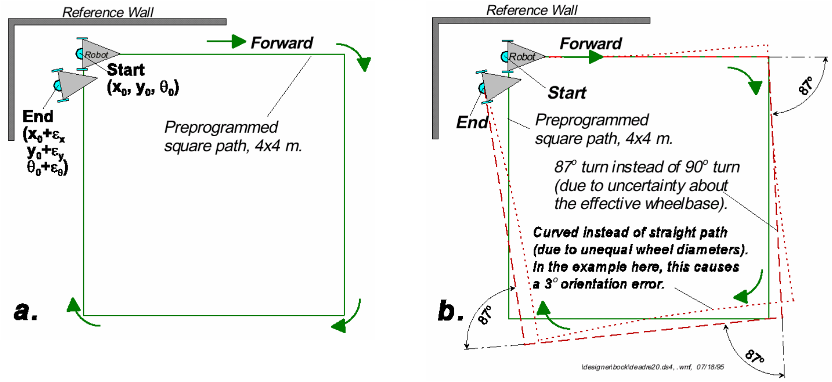
Movement on a Circle



- Errors perpendicular to the direction of movement grow much more quickly
- Error ellipses do not remain perpendicular to the direction of movement

Localization, Path Planning, & Navigation: Odometry- Calibration of Errors

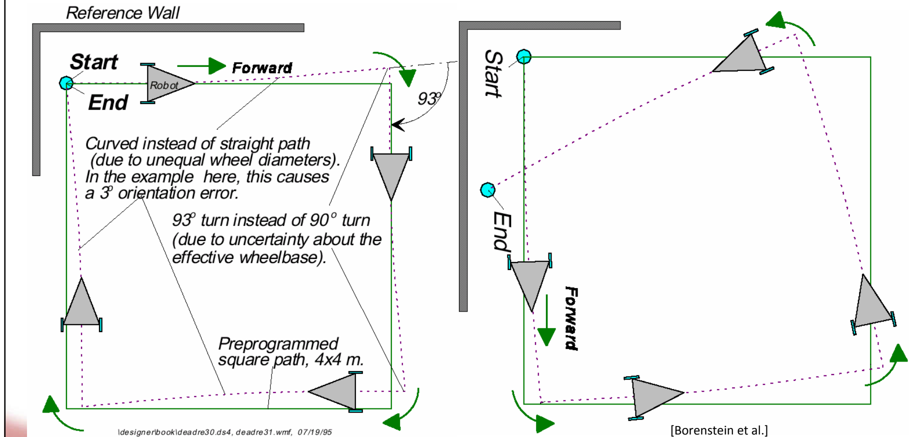
- Unidirectional square path experiment



[Borenstein et al.]

Localization, Path Planning, & Navigation: Odometry- Calibration of Errors

- Bi-directional square path experiment



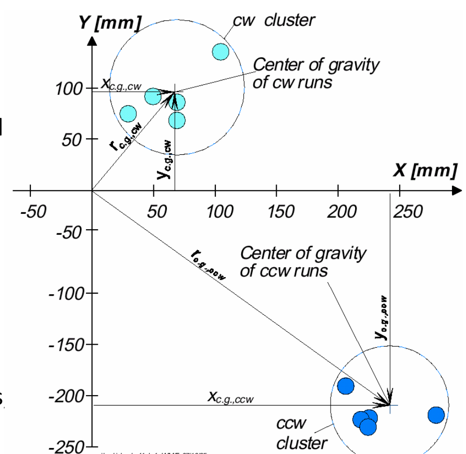
[Borenstein et al.]

Localization, Path Planning, & Navigation: Odometry- Calibration of Errors

- Deterministic errors (systematic)
  - From wheels diameters, wheel base, misalignment, encoder errors, etc.

$$E_{systematic} = \max(r_{c.g.,cw}; r_{c.g.,ccw})$$

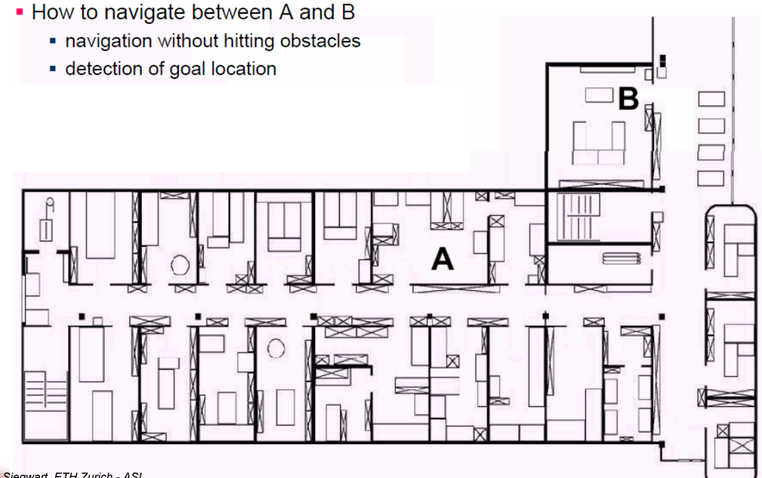
- Non-deterministic errors (non-systematic)
  - From travel over uneven floors objects, wheel slippage, etc.



[Borenstein et al.]

Localization, Path Planning, & Navigation: To Localize or Not?

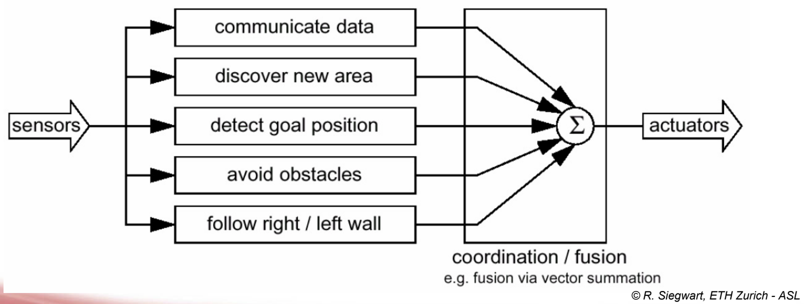
- How to navigate between A and B
  - navigation without hitting obstacles
  - detection of goal location



© R. Siegwart, ETH Zurich - ASL

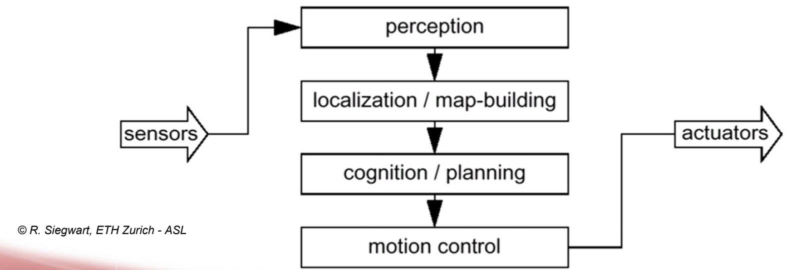
Localization, Path Planning, & Navigation: Behavior (Sensor) Based Navigation

- Procedural solution to navigation problem
  - Simple and Quick implementation (+)
  - Doesn't translate/scale well to other environments (-)
  - Underlying procedures can be complicated (-)
  - Running multiple behaviors at once requires fine tuning (-)



Localization, Path Planning, & Navigation: Model (Map) Based Navigation

- Robot explicitly attempts to localize by collecting sensor data and updates belief about position wrt a map
  - Requires more upfront effort (-)
  - Architecture can be leveraged to map and navigate a variety of environments (+)
  - Behavior only as good as map (-)



Localization, Path Planning, & Navigation: Probabilistic, Map-Based Localization

- Consider a mobile robot moving in a known environment.
- As it start to move, say from a precisely known location, it might keep track of its location using odometry.
- However, after a certain movement the robot will get very uncertain about its position.

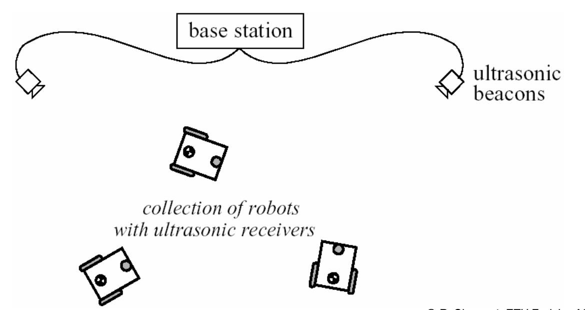
➔ update using an observation of its environment.

- observation leads also to an estimate of the robots position which can than be fused with the odometric estimation to get the best possible update of the robots actual position.

© R. Siegwart, ETH Zurich - ASL

Localization, Path Planning, & Navigation: Positioning Beacon Systems- Triangulation

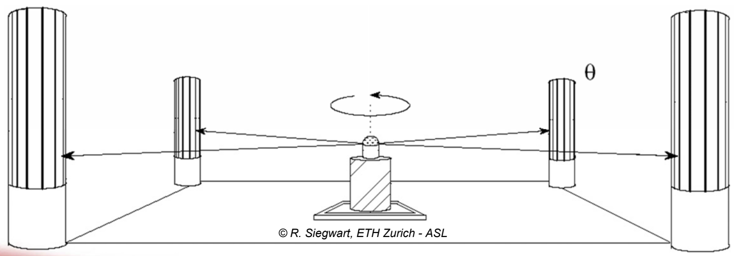
- Robot knows positions of beacons in global reference frame
- Localizes own position in frame through triangulation, i.e. geometry



© R. Siegwart, ETH Zurich - ASL

Localization, Path Planning, & Navigation: Positioning Beacon Systems- Triangulation

- Industrial setting example:
  - Beacons are retroreflective markers that reflect energy back to robot
  - Known positions for optical retroreflectors
  - Need 3 beacons in sight to determine position
    - High reliability
    - Costly setup, only works in that particular environment



© R. Siegwart, ETH Zurich - ASL

Localization, Path Planning, & Navigation: SLAM: Simultaneous Localization and Mapping

- Goal:
  - Start robot from an arbitrary initial point
  - Autonomous exploration of environment with on-board sensors
  - Acquire knowledge about environment
  - Interpret the scene and build an appropriate map
  - Localize itself relative to this map

Localization, Path Planning, & Navigation: Competencies for Navigation

- Cognition / Reasoning :
  - is the ability to decide ***what actions are required*** to achieve a ***certain goal*** in a ***given situation (belief state)***.
  - decisions ranging from ***what path to take*** to what ***information on the environment to use***.
- Today's industrial robots can operate without any cognition (reasoning) because their environment is static and very structured.
- In mobile robotics, cognition and reasoning is primarily of geometric nature, such as picking safe path or determining where to go next.
  - already been largely explored in literature for cases in which complete information about the current situation and the environment exists (e.g. sales man problem).

© R. Siegwart, ETH Zurich - ASL

Localization, Path Planning, & Navigation: Competencies for Navigation

- However, in mobile robotics the knowledge of about the environment and situation is usually only partially known and is uncertain.
  - makes the task much more difficult
  - requires multiple tasks running in parallel, some for planning (global), some to guarantee "survival of the robot".
- Robot control can usually be decomposed in various behaviors or functions
  - e.g. wall following, localization, path generation or obstacle avoidance.
- In chapter 6 we are concerned with path planning and navigation

© R. Siegwart, ETH Zurich - ASL

Localization, Path Planning, & Navigation: Path Planning

- The problem: find a path in the physical space from the initial position to the goal position avoiding all collisions with the obstacles
- We can generally distinguish between
  - (*global*) path planning and
  - (*local*) obstacle avoidance.

© R. Siegwart, ETH Zurich - ASL

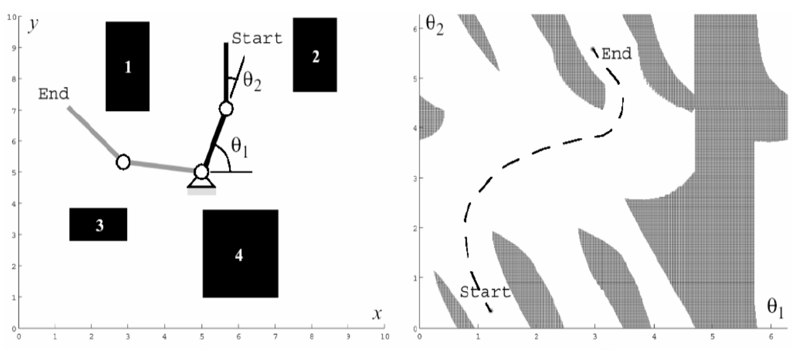
Localization, Path Planning, & Navigation: Global Path Planning

- Assumption: there exists a good enough map of the environment for navigation.
  - Topological or metric or a mixture between both.
- First step:
  - Representation of the environment by a road-map (graph), cells or a potential field. The resulting discrete locations or cells allow then to use standard planning algorithms.
- Examples that we will see:
  - Visibility Graph
  - Voronoi Diagram
  - Cell Decomposition -> Connectivity Graph
  - Potential Field

© R. Siegwart, ETH Zurich - ASL

Localization, Path Planning, & Navigation: Path Planning- Configuration Space

- State or configuration q can be described with k values q_i



Work Space

Configuration Space: the dimension of this space is equal to the Degrees of Freedom (DoF) of the robot

© R. Siegwart, ETH Zurich - ASL

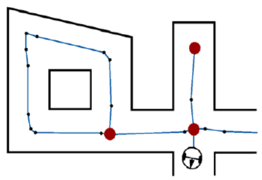
Localization, Path Planning, & Navigation: Configuration Space- Mobile Robot

- Mobile robots operating on a flat ground have 3 DoF: (x, y, θ)
- For simplification, mobile roboticists assume that the robot is a point. In this way the configuration space is reduced to 2D (x,y)
- Because we have reduced each robot to a point, we have to inflate each obstacle by the size of the robot radius to compensate.

© R. Siegwart, ETH Zurich - ASL

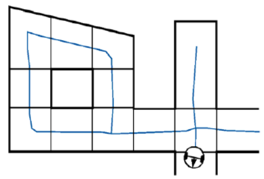
Localization, Path Planning, & Navigation: Path Planning Overview

1. Road Map, Graph construction
   - Identify a set of routes within the free space



- Where to put the nodes?
- Topology-based:
  - at distinctive locations
- Metric-based:
  - where features disappear or get visible

2. Cell decomposition
   - Discriminate between free and occupied cells

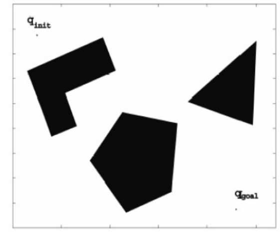


- Where to put the cell boundaries?
- Topology- and metric-based:
  - where features disappear or get visible

3. Potential Field
   - Imposing a mathematical function over the space

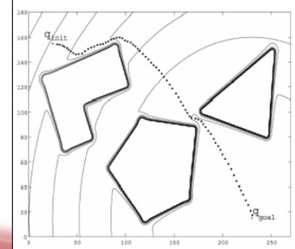
© R. Siegwart, ETH Zurich - ASL

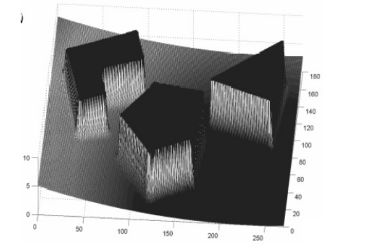
Localization, Path Planning, & Navigation: Potential Field Path Planning



- Robot is treated as a *point under the influence* of an artificial potential field.
  - Generated robot movement is similar to ball rolling down the hill
  - Goal generates attractive force
  - Obstacle are repulsive forces

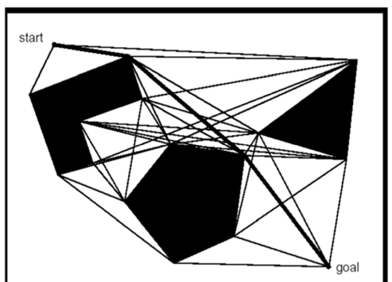
© R. Siegwart, ETH Zurich - ASL





Localization, Path Planning, & Navigation: Road-Map Path Planning- Visibility Graph

- Nodes of graph:
  - initial and goal positions
  - vertices of obstacles
- Road map:
  - All nodes visible from each other connected by straight-line segments to define map

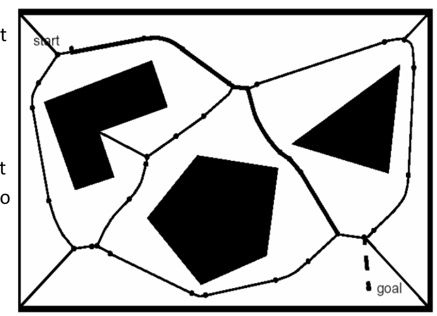


- Pros
  - It is easy to find the shortest path from the start to the goal positions
  - Implementation simple when obstacles are polygons
- Cons
  - Number of edges and nodes increases with the number of polygons
  - Thus it can be inefficient in densely populated environments
  - The solution path found by the visibility graph tend to take the robot asclose as possible to obstacles: the common solution is to grow obstacles by more than robot's radius

© R. Siegwart, ETH Zurich - ASL

Localization, Path Planning, & Navigation: Road-Map Path Planning- Voronoi Diagram

- Lines constructed from points that are equidistant from two or more obstacles
- Maximizes distance between robot and obstacles
- Initial and goal states mapped to diagram by drawing line to edge along which its distance to the boundary of the obstacle increases the fastest
- Direction of movement selected so the distance to the boundaries increases fastest
- Easy to execute: maximize sensor readings
- Works for map-building: move on Voronoi edges



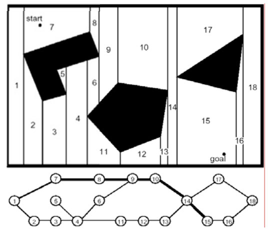
© R. Siegwart, ETH Zurich - ASL

- Pros
  - Using range sensors like laser or sonar, a robot can navigate along the Voronoi diagram using simple control rules
- Cons
  - Because the Voronoi diagram tends to keep the robot as far as possible from obstacles, any short range sensor will be in danger of failing
- Peculiarities
  - when obstacles are polygons, the Voronoi map consists of straight and parabolic segments

Localization, Path Planning, & Navigation: Road-Map Path Planning- Cell Decomposition

- Divide space into simple, connected regions called cells
- Determine which open sells are adjacent and construct a connectivity graph
- Find cells in which the initial and goal configuration (state) lie and search for a path in the connectivity graph to join them.
- From the sequence of cells found with an appropriate search algorithm, compute a path within each cell.
  - e.g. passing through the midpoints of cell boundaries or by sequence of wall following movements.
- Possible cell decompositions:
  - Exact cell decomposition
  - Approximate cell decomposition:
    - Fixed cell decomposition
    - Adaptive cell decomposition

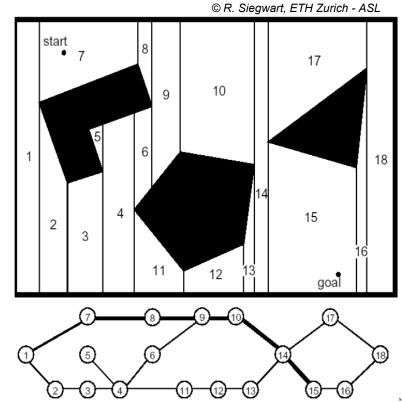
© R. Siegwart, ETH Zurich - ASL



Localization, Path Planning, & Navigation: Exact Cell Decomposition

© R. Siegwart, ETH Zurich - ASL

- Boundary of cells based on critical geometry
- Cells are either completely free or completely occupied
- Robot position in free cell does not matter
- Robot ability to traverse from free cell to adjacent free cell matters
- # of cells and planning computation efficiency depends on density and complexity of obstacles in environment (-)
- In large sparse environments, very small # of cells and efficient (+)

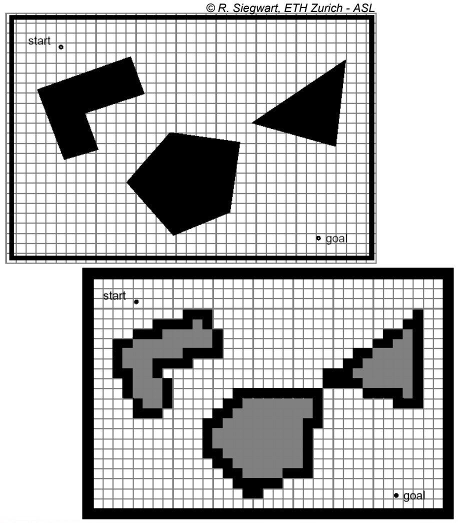


Connectivity Graph

Localization, Path Planning, & Navigation: Approximate Cell Decomposition- Grids

© R. Siegwart, ETH Zurich - ASL

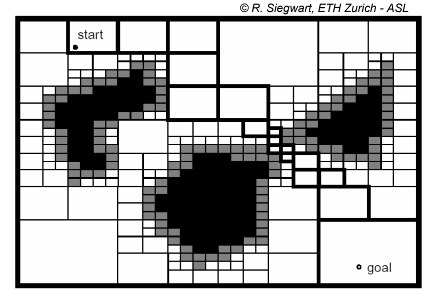
- Fixed grid-sized decomposition
- Cell size not dependant on particular objects in environment
- Cell is either free or obstacle-filled
- Low computational complexity for path planning (+)
- Fundamental cost is memory
  - Even sparse environment must be represented in its entirety (-)
- Narrow passageways can be lost (-)



Localization, Path Planning, & Navigation: Adaptive Cell Decomposition

- Free space externally bounded by rectangle and internally bounded by 3 polygons
- Recursively decompose rectangle into 4 smaller rectangles
- At each resolution, only cells whose interiors lie entirely in free space are used to construct connectivity graph
- Adapts to complexity of environment

© R. Siegwart, ETH Zurich - ASL

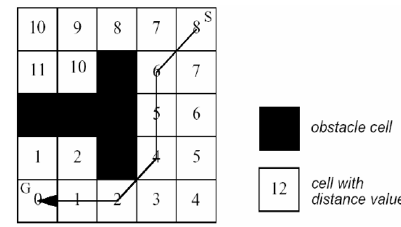


Localization, Path Planning, & Navigation: Path/Graph Search Strategies

- Wavefront Expansion
- Breadth-First Search
- Depth-First Search
- A*

Localization, Path Planning, & Navigation: Path/Graph Search Strategies

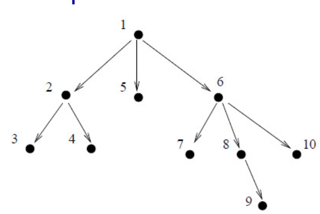
- Wavefront Expansion (grassfire)
  - Starting from goal position, mark each cell its distance to the to the goal cell
  - Continue until start position is reached
    - Estimate of robots distance to goal
  - Planner:
    - Links together cells that are adjacent and always closer to the goal = path



© R. Siegwart, ETH Zurich - ASL

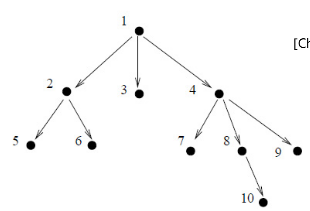
Localization, Path Planning, & Navigation: Depth-First Search vs. Breadth-First Search

Depth-first search



[One branch at a time]

Breath-first search



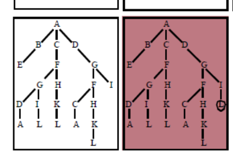
[All branches at same time]

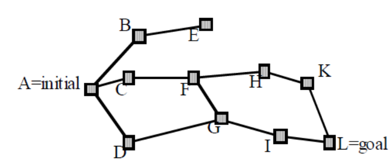
[Choset et al.]

- Numbers on each node reflect the order in which nodes are expanded in the search

Localization, Path Planning, & Navigation: Depth-First Search

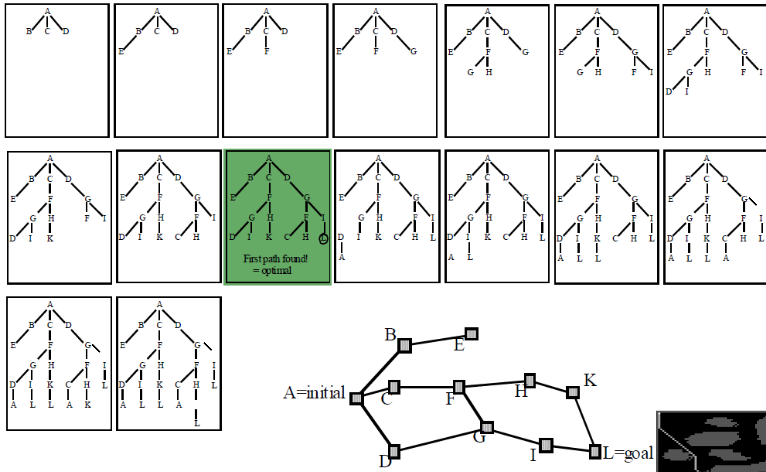
[Choset et al.]

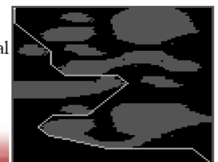




Localization, Path Planning, & Navigation: Breadth-First Search

[Choset et al.]





Localization, Path Planning, & Navigation: Search Algorithms

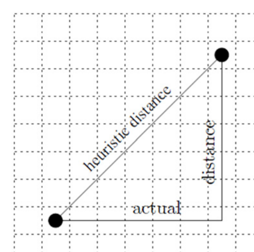
- Depth-first: fastest solution to find a path
- Breadth-first: shortest path to start node in terms of link lengths
- Wavefront: shortest path with respect to Manhattan distance (graph with edge lengths = 1)
- Shortest-path length may not always be the only metric want to optimize
  - Energy, time, traversability, safety, etc.
- Minimize the # of nodes to be visited to locate the goal node subject to path optimality criteria
  - Optimality: measures path
  - Efficiency: measures the search (# of nodes visited to determine path)

Localization, Path Planning, & Navigation: Search Algorithms

- Define a *heuristic*: an expected but not necessarily actual, cost to the goal node
- Example:
  - Search may choose explore next node that has shortest Euclidean distance to goal bc/ node has highest possibility (based on local info) of getting closest to goal
  - No guarantee that node will lead to (globally) shortest path in the graph to goal
  - Good guess, based on information that is available

Localization, Path Planning, & Navigation: A* Algorithm

- Searches a graph efficiently with respect to a chosen heuristic
  - "Good" heuristic, efficient search
  - "Bad" heuristic, path will be found, inefficient search, suboptimal path
  - "Optimistic" heuristic will return an optimal path
    - Heuristic always returns a value less than or equal to the cost of the shortest path from the current node to the goal node

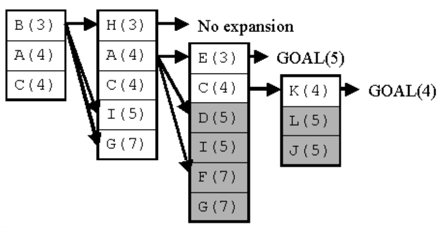


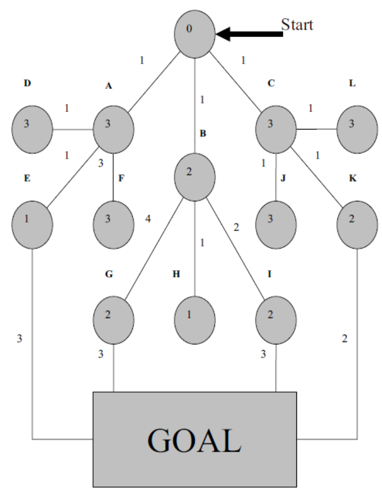
[Choset et al.]

Localization, Path Planning, & Navigation: A* Search Example

[Choset et al.]

- Nodes: A→ I
- Heuristic values inside node icon
- Edge costs represented by #'s adjacent to edges
- Start node = 0 (highest priority)





Localization, Path Planning, & Navigation: A* Algorithm

[Choset et al.]

Input: A graph
Output: A path between start and goal nodes

repeat
  Pick n_{best} from O such that $f(n_{best}) \le f(n), \forall n \in O$.
  Remove n_{best} from O and add to C.
  If $n_{best} = q_{goal}$, EXIT.
  Expand n_{best}: for all $x \in \text{Star}(n_{best})$ that are not in C.
  if $x \notin O$ **then**
    add x to O.
  else if $g(n_{best}) + c(n_{best}, x) < g(x)$ **then**
    update x's backpointer to point to n_{best}
  end if
until O is empty

- O = open set: priority queue
- C = closed set: all processed nodes
- $\text{Star}(n)$ represents the set of nodes which are adjacent to n.
- $c(n_1, n_2)$ is the length of edge connecting n_1 and n_2.
- $g(n)$ is the total length of a backpointer path from n to q_{start}.
- $h(n)$ is the heuristic cost function, which returns the estimated cost of shortest path from n to q_{goal}.
- $f(n) = g(n) + h(n)$ is the estimated cost of shortest path from q_{start} to q_{goal} via n.

Localization, Path Planning, & Navigation: A* Special Cases

- Greedy Search: f(n) = h(n)
  - Search is only considering what it "believes" is the best path to the goal from the current node
- Dijkstra's Algorithm: f(n) = g(n)
  - Planner is not using any heuristic information
  - It grows a path that is shortest from the start until it encounters the goal

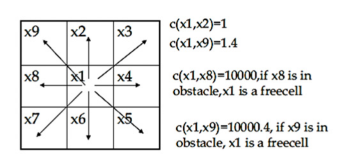
Localization, Path Planning, & Navigation: A* on a Grid

- Heuristic values (h) are set
- Backpointers (b) and priorities (f) are not

r/c	1	2	3	4	5	6	7
6	h =6 f = b=()			h =3 f = b=()	h =2 f = b=()	h =1 f = b=()	h =0 f = b=() Goal
5	h =6.4 f = b=()			h =3.4 f = b=()	h =2.4 f = b=()	h =1.4 f = b=()	h =1 f = b=()
4	h =6.8 f = b=()			h =3.8 f = b=()	h =2.8 f = b=()	h =2.4 f = b=()	h =2 f = b=()
3	h =7.2 f = b=()			h =4.2 f = b=()	h =3.8 f = b=()	h =3.4 f = b=()	h =3 f = b=()
2	h =7.6 f = b=()	h =6.6 f = b=()	h =5.6 f = b=()		h =4.8 f = b=()	h =4.4 f = b=()	h =4 f = b=()
1	h =8.0 f = b=()	h =7.0 f = b=() Start	h =6.6 f = b=()	h=6.2 f= b=()	h =5.8 f = b=()	h =5.4 f = b=()	h =5 f = b=()

[Choset et al.]

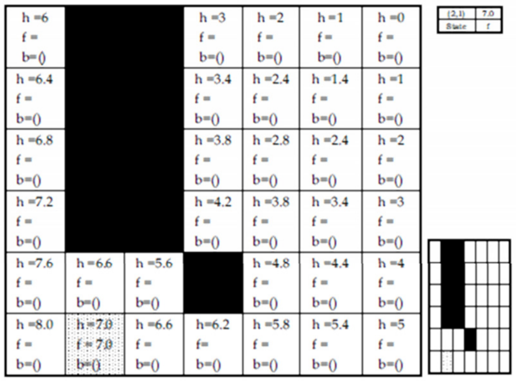
Heuristic:
Horizontal/Vertical Step: length = 1
Diagonal Step: length = 1.4 → optimistic ($< \sqrt{2}$)
Edge (step) Cost:
Step from free space to obstacle pixel = 1000
Step from free space to free space = 1



[8 point connectivity]

Localization, Path Planning, & Navigation: A* on a Grid

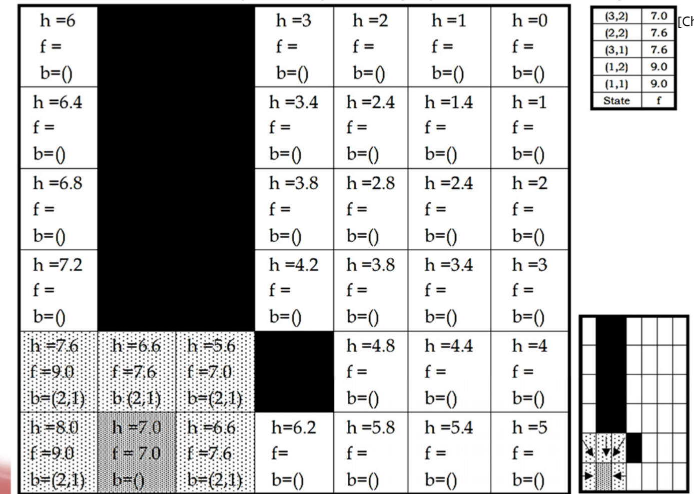
- Start node is put on the priority queue, with f = h:



[Choset et al.]

Localization, Path Planning, & Navigation: A* on a Grid

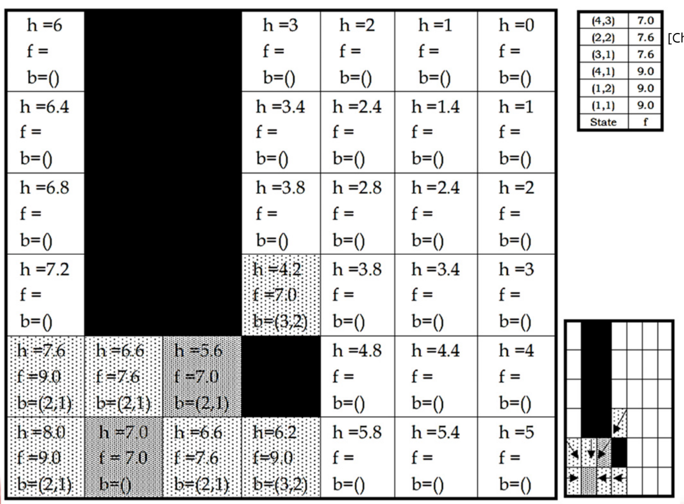
- Expand the start node, update priority queue, set backpointers:



[Choset et al.]

Localization, Path Planning, & Navigation: A* on a Grid

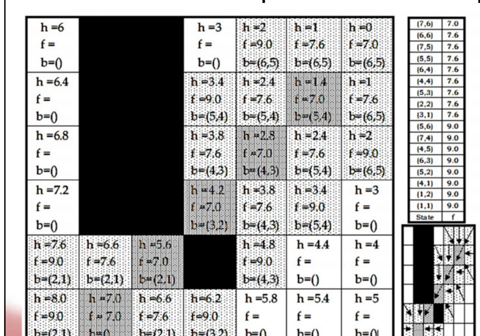
- Expand cell with highest priority next (lowest f)

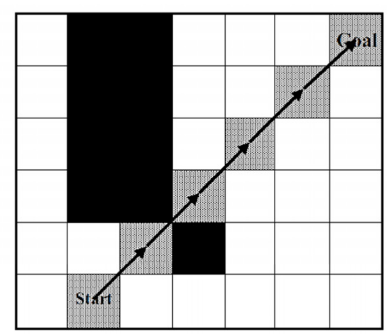


[Choset et al.]

Localization, Path Planning, & Navigation: A* on a Grid

- Continue until goal state gets expanded
- Since priority value of goal cell is lower than the priorities of all other cells in queue, the path is optimal, and A* terminates
- Trace the backpointers to find optimal path from start to goal



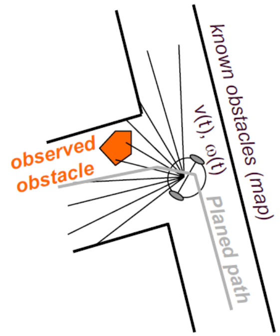


[Choset et al.]

Localization, Path Planning, & Navigation: Obstacle Avoidance
(Local Path Planning)

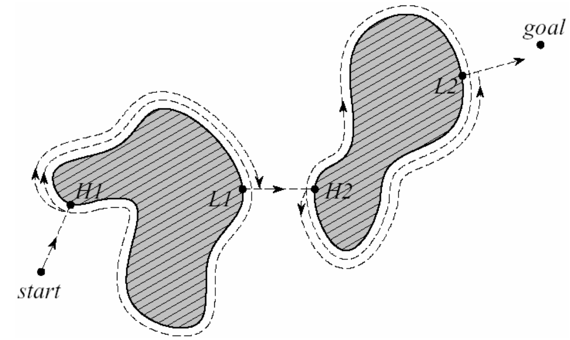
- The goal of the obstacle avoidance algorithms is to avoid collisions with obstacles
- It is usually based on local map
- Often implemented as a more or less independent task
- However, efficient obstacle avoidance should be optimal with respect to
  - the overall goal
  - the actual speed and kinematics of the robot
  - the on boards sensors
  - the actual and future risk of collision

© R. Siegwart, ETH Zurich - ASL



Localization, Path Planning, & Navigation: Obstacle Avoidance- Bug1 Algorithm

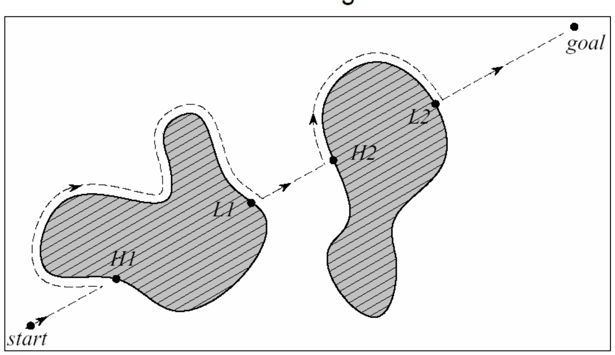
- Following along the obstacle to avoid it
- Each encountered obstacle is once fully circled before it is left at the point closest to the goal

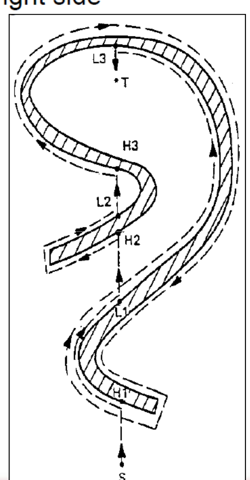


© R. Siegwart, ETH Zurich - ASL

Localization, Path Planning, & Navigation: Obstacle Avoidance- Bug2 Algorithm

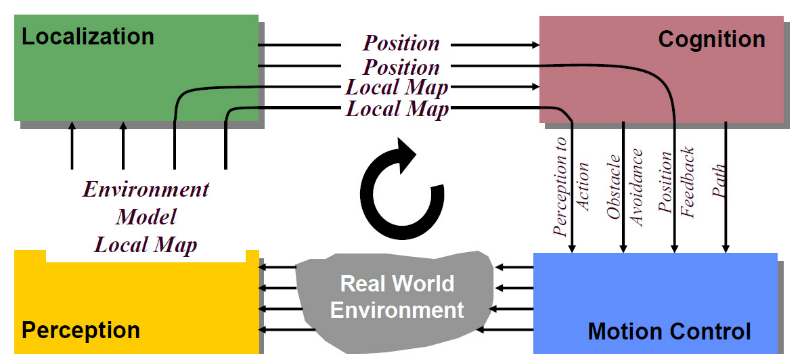
- Following the obstacle always on the left or right side
- Leaving the obstacle if the direct connection between start and goal is crossed





© R. Siegwart, ETH Zurich - ASL

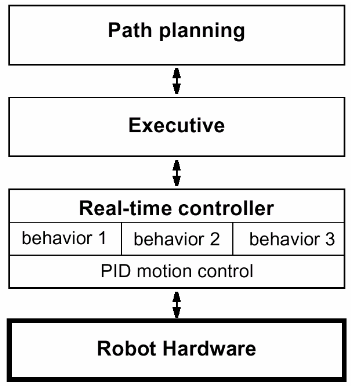
Localization, Path Planning, & Navigation: Mobile Robot General Architecture



© R. Siegwart, ETH Zurich - ASL

Localization, Path Planning, & Navigation: Tiered Navigation Architecture

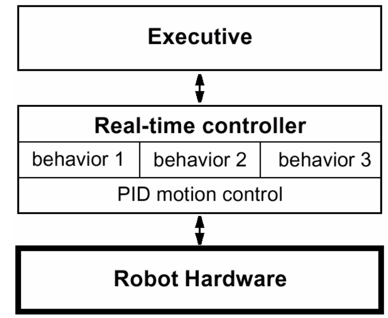
- Path Planning
  - Strategic level decision making
  - Uses global information (in non-real-time) to identify sequence of local actions for robot
- Real-time controller
  - Requires high-band width and tight sensor-effector loops
  - Includes lower level behaviors that may switch or run in parallel
- Executive
  - Responsible for mediating interface between planning and execution
  - Manages the activation of behaviors, failure recognition, and re-initiating planner



© R. Siegwart, ETH Zurich - ASL

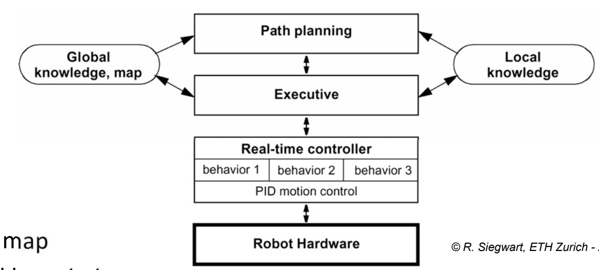
Localization, Path Planning, & Navigation: Two-Tiered Architecture for Off-line Planning

- Executive must contain a priori all relevant schemes for traveling to desired destinations
- Not useful as general solution to navigation
- Good for static route-based applications
  - Factory or warehouse settings
  - Number of discrete goal positions small enough that executive can cache paths required to reach each goal rather than generic map which a planner could search for solution paths
- Good for extreme reliability demands
  - Can't afford a bad plan, compute it off-line ahead of time
  - Example: contingency flight plans for space shuttle in advance of shuttle flights



© R. Siegwart, ETH Zurich - ASL

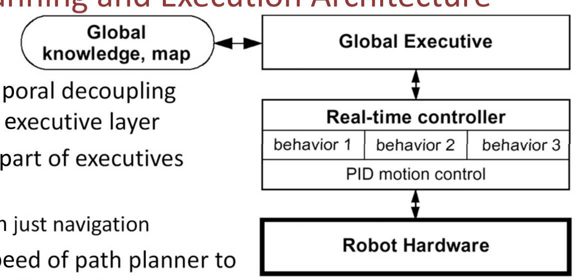
Localization, Path Planning, & Navigation: Three-Tiered Episodic Planning Architecture



© R. Siegwart, ETH Zurich - ASL

- Strategic, global map
- Short-term, local knowledge
- Executive decides when to trigger planner based on local information
  - Path blockage, failure, etc.
- Executive will then update global knowledge base accordingly

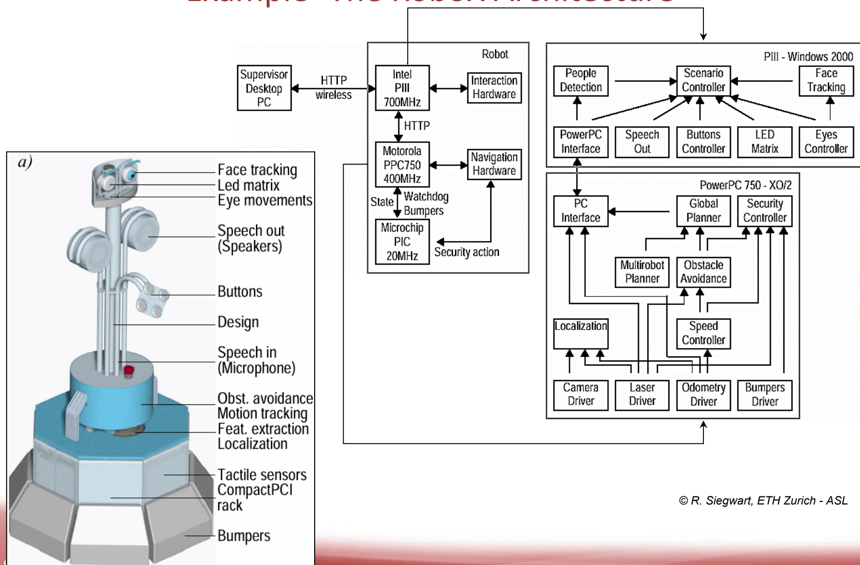
Localization, Path Planning, & Navigation: Integrated Planning and Execution Architecture



© R. Siegwart, ETH Zurich - ASL

- All integrated, no temporal decoupling between planner and executive layer
- Planning is one small part of executives cycle of activities
  - More functions than just navigation
- Requires execution speed of path planner to run within basic control loop of executive
  - Very computationally challenging
  - Example:
    - large off-road vehicle traveling over partially know terrains at high speeds
    - Local and global representations are the same
  - Not possible in complex environments with current processor speeds

Localization, Path Planning, & Navigation: Example- The RoboX Architecture



© R. Siegwart, ETH Zurich - ASL

Localization, Path Planning, & Navigation: Extra References

- J. Borenstein, H. Everett, L. Feng, *Where am I? Sensors and Methods for Mobile Robot Positioning*. Ann Arbor, University of Michigan, 1996. Available at http://www-personal.umich.edu/~johannb/shared/pos96rep.pdf

- H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavarki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementation,* MIT Press, Boston, 2005 http://www.cs.cmu.edu/~biorobotics/book/