BEST2015 — Autonomous Mobile Robots
Lecture 4: Mobile Robot Planning and Navigation

Renaud Ronsse
renaud.ronsse@uclouvain.be

École polytechnique de Louvain, UCLouvain

July 2015

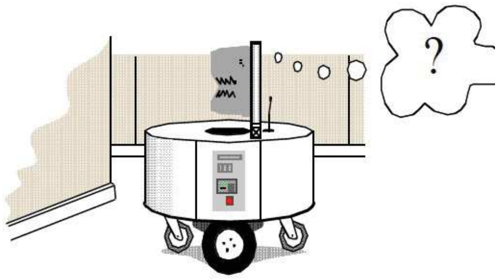
DARPA urban challenge 2007

http://en.wikipedia.org/wiki/DARPA_Grand_Challenge_(2007)

Key Concepts in Autonomous Mobile Robotics

The three key questions in Mobile Robotics:

1. Where am I?
2. Where am I going?
3. How do I get there?

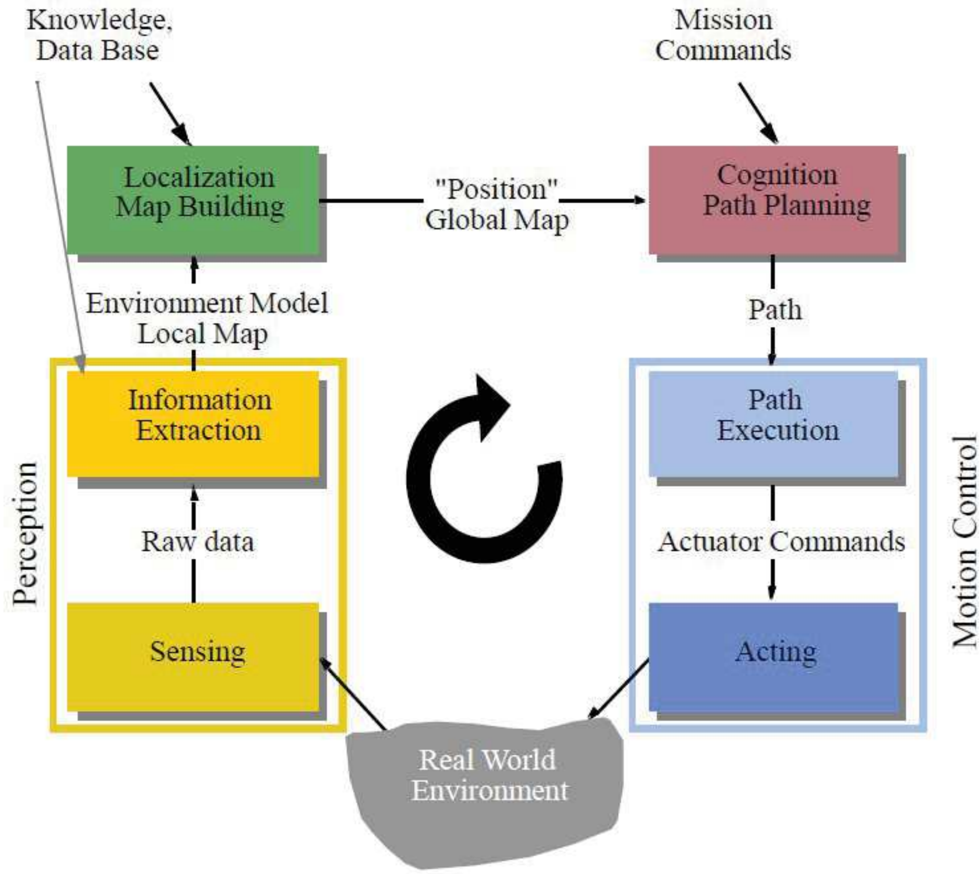


To answer these questions the robot has to:

- have a model of the environment (given or autonomously built);
- perceive and analyze the environment;
- find its position/situation within the environment;
- plan and execute the movement.

This lecture is about this last point, i.e. trajectory planning and navigation in mobile robotics.

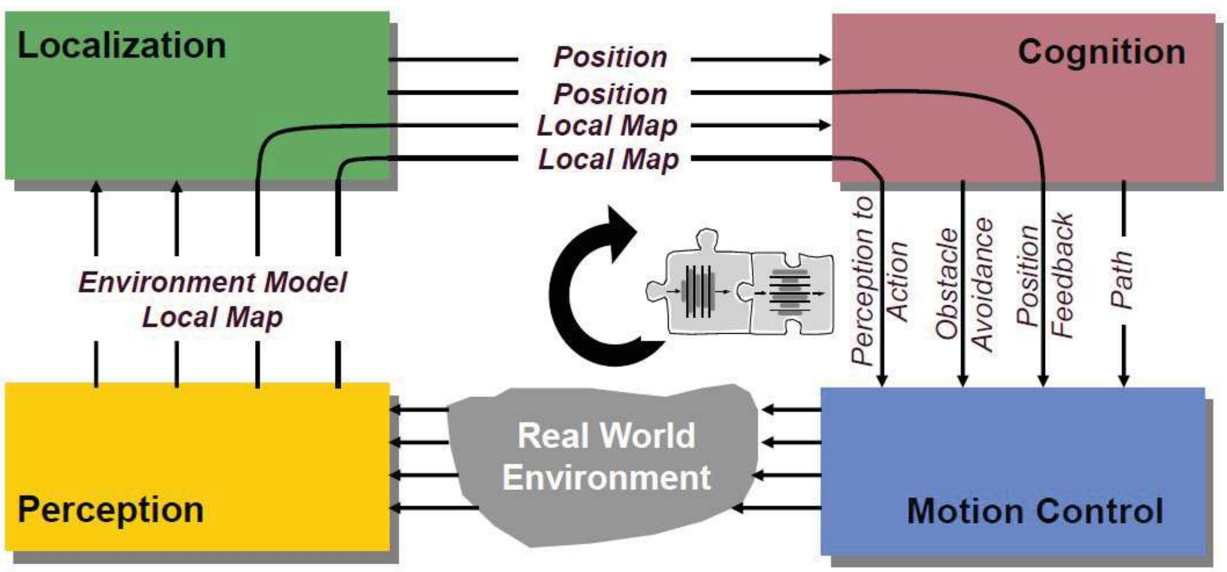
General Control Scheme for Mobile Robot Systems



ⓒ Siegwart et al., 2011, Fig. 1.15, p. 10

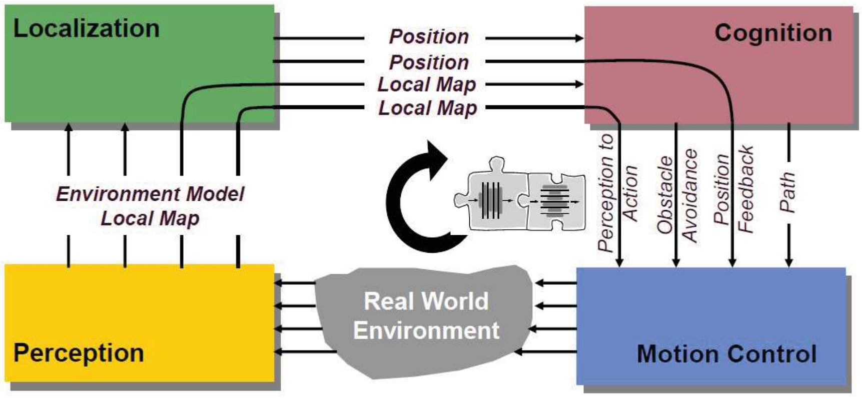
Control Architectures/Strategies

Control Loop: dynamically changing; no compact model available; and many sources of uncertainties!



ⓒ Siegwart et al., 2011, Fig. 6.27, p. 417

Case studies

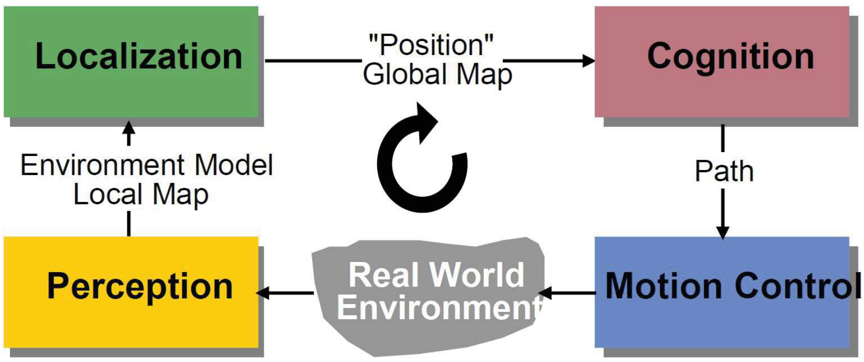


© Siegwart et al., 2011, Fig. 6.27, p. 417

Some bypasses are possible:

- Obstacle avoidance requires little input from the localization module and consists of fast decisions at the cognition level followed by execution in motion control.
- PID position feedback loops bypass all high-level processing, tying the perception of encoder values directly to lowest-level PID control loops in motion control.

Planning and Navigation



© Siegwart et al., 2011

Cognition: purposeful decision making and execution to achieve the highest-order goal.

For a mobile robot: navigation encompasses the ability of the robot to act based on its knowledge and sensor values so as to reach its goal positions as efficiently and as reliably as possible.

Two key and complementary competences:

- path planning;
- (changing) obstacle avoidance.

Competences for Navigation: Planning and Reacting

Strong complementarity: navigation requires to reach the goal position while reacting to unforeseen events (e.g., obstacles).

- A plan q is nothing more than one or more trajectories from b_i to b_g (goal), consistent with the current map of the environment M_i.
- Reacting will modulate robot behavior locally (correction of the planned-upon trajectory) or will require changes to the robot's strategic plans

Limit: the planner incorporates every new piece of information in real time, i.e. merged planning and reacting: integrated planning and execution.

Competences for Navigation: Planning and Reacting

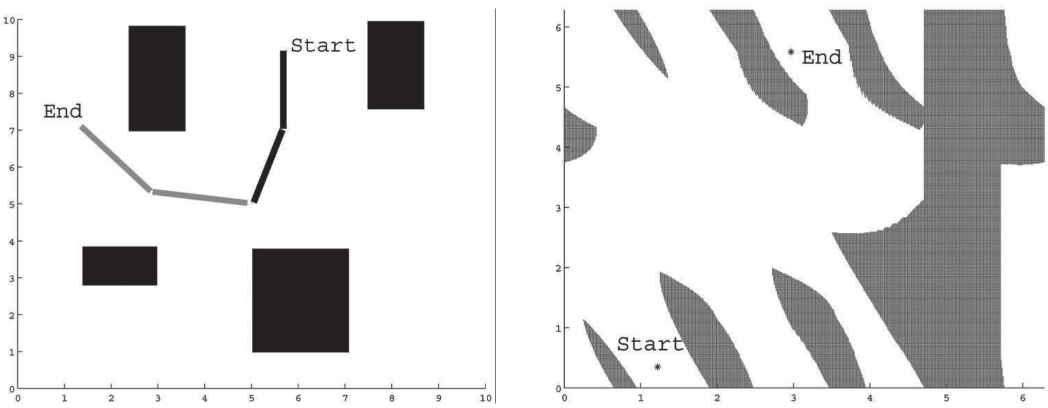
Completeness: The robot system is complete if and only if, for all possible problems (i.e., initial belief states, maps, and goals), when there exists a trajectory to the goal belief state b_g, the system will achieve the goal belief state b_g.

When a system is incomplete, there is at least one example problem for which, although there is a solution, the system fails to generate a solution. Often, completeness is sacrificed for computational complexity at the level of representation or reasoning.

Path Planning

In general, simpler problem for mobile robots than for multiple dof industrial robots.

Configuration space C: set of points p whose coordinates are $q_1, \ldots, q_k$.



© Siegwart et al., 2011, Fig. 6.1, p. 372

Path planning: find a trajectory in $F = C - O$ (where O is the configuration space obstacle) connecting the start configuration to the end configuration.

Path Planning

In mobile robotics a simplifying assumption is to assume the robot to be holonomic, i.e. $C = \mathbb{R}^2 \times \mathbb{S}\ (x, y, \theta)$.
A further simplification is to assume the robot to be point, i.e. $C = \mathbb{R}^2\ (x, y)$. In this case, the obstacle must be inflated by the size of the robot radius!

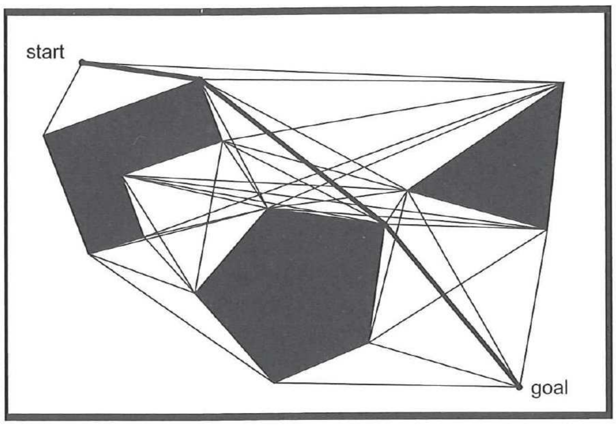
Advantage: configuration space C = physical space...

Two general approaches for path planning:

1. Graph search: a connectivity graph in free space is first constructed (offline) and then searched.
2. Potential field planning: a mathematical function is imposed directly on the free space. The gradient of this function can then be followed to the goal.

Graph construction – example 1: Visibility graph

Edges joining all pairs of vertices that can see each other (including both initial and goal positions).
Path planning: find a (shortest) path from the initial position to the goal position along the defined roads.



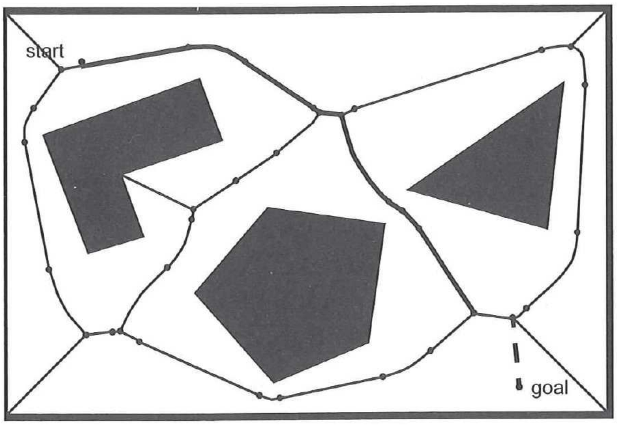
© Siegwart et al., 2011, Fig. 6.2, p. 374

Caveats:

1. number of edges and nodes increases with number of obstacle polygons;
2. take the robot as close as possible to obstacles (safety).

Graph construction – example 2: Voronoi diagram

Maximize the distance between the robot and obstacles in the map.
Set of edges formed by points equidistant to two obstacles. If O are polygons, the Voronoi diagram consists of straight line and parabolic segments only.
Weak if limited range localization sensors.
Simple control: maximize local minima in sensor readings.

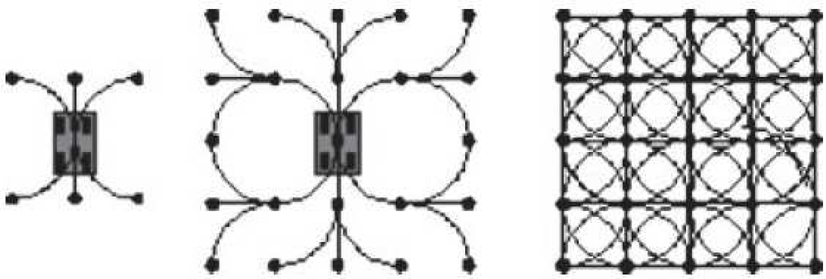


© Siegwart et al., 2011, Fig. 6.3, p. 376

Visibility graph and Voronoi diagram are complete algorithms.

Graph construction – example 3: Lattice graph

Constructing a base set of edges and then repeating it over the whole configuration space.
Lattice graphs are typically precomputed for a given robotic platform and stored in memory. They thus belong to the class of approximate (i.e. not complete) decomposition methods.



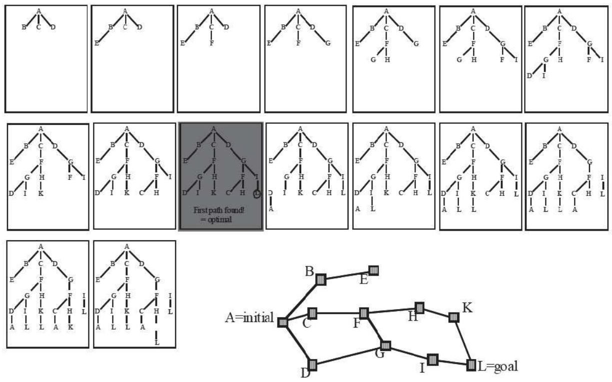
© Pivtoraiko et al., *Journal of Fields Robotics* 2009

Caveat: memory.

Graph search

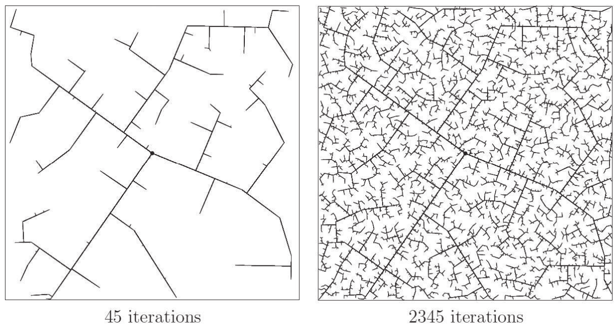
Search the graph to find the best path.

- Deterministic: exhaustive search of all possible solutions.



© Siegwart et al., 2011, Fig. 6.6, p. 381

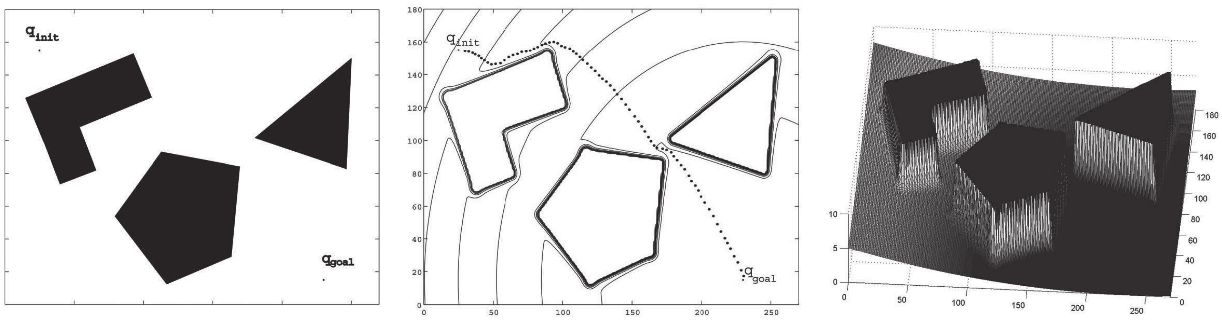
- Random, e.g. by growing an online graph (high-dimensional).



© Siegwart et al., 2011, Fig. 6.11, p. 387

Potential Field Path Planning

Create a field, or gradient, across the robot's map that directs the robot to the goal position from multiple prior positions: the robot is a point under the influence of an artificial potential field $U(q)$.



© Siegwart et al., 2011, Fig. 6.12, p. 388

Also a control law for the robot: can always determine its next required action based on the field.

- attracted toward the goal;
- repulsed by the obstacles known in advance;
- if new obstacles appear during motion, the potential field is updated.

Interesting source: Leng-Feng Lee MS thesis (Youtube)

Implementation of Potential Field Path Planning

If the robot is a point, the resulting potential field is only 2D (x, y):

$$F(q) = -\nabla U(q) = -\begin{bmatrix} \frac{\partial U}{\partial x} \\ \\ \frac{\partial U}{\partial y} \end{bmatrix}$$

where $\nabla U(q)$ denotes the gradient vector of U at position q.
Sum of attractive and repulsive potentials:
$U(q) = U_{att}(q) + U_{rep}(q)$.

$$F(q) = F_{att}(q) + F_{rep}(q) = -\nabla U_{att}(q) - \nabla U_{rep}(q)$$

Attractive potential

For example, parabolic function:

$$U_{att}(q) = \frac{1}{2} k_{att} \rho_{goal}^2(q)$$

where k_{att} is a positive scaling factor and $\rho_{goal}(q)$ denotes the Euclidean distance $\|q - q_{goal}\|$.

$$\begin{aligned} F_{att}(q) &= -\nabla U_{att}(q) \\ &= -k_{att}\, \rho_{goal}(q)\, \nabla \rho_{goal}(q) \\ &= -k_{att}(q - q_{goal}) \end{aligned}$$

Repulsive potential

Generate a force away from all known obstacles:

- very strong when the robot is close to the object;
- no influence when the robot is far from the object.

Example:

$$U_{rep}(q) = \begin{cases} \frac{1}{2}k_{rep}\left(\frac{1}{\rho(q)} - \frac{1}{\rho_0}\right)^2 & \text{if } \rho(q) \leq \rho_0 \\ 0 & \text{if } \rho(q) > \rho_0 \end{cases}$$

where k_{rep} is a scaling factor, $\rho(q)$ denotes the minimal distance from q to the object and ρ_0 is the distance of influence of the object.

$$\begin{aligned} F_{rep}(q) &= -\nabla U_{rep}(q) \\ &= \begin{cases} k_{rep}\left(\frac{1}{\rho(q)} - \frac{1}{\rho_0}\right)\frac{1}{\rho^2(q)}\frac{q-q_{obstacle}}{\rho(q)} & \text{if } \rho(q) \leq \rho_0 \\ 0 & \text{if } \rho(q) > \rho_0 \end{cases} \end{aligned}$$

Control: set the robot's velocity vector proportional to the force field (ball down a hill).

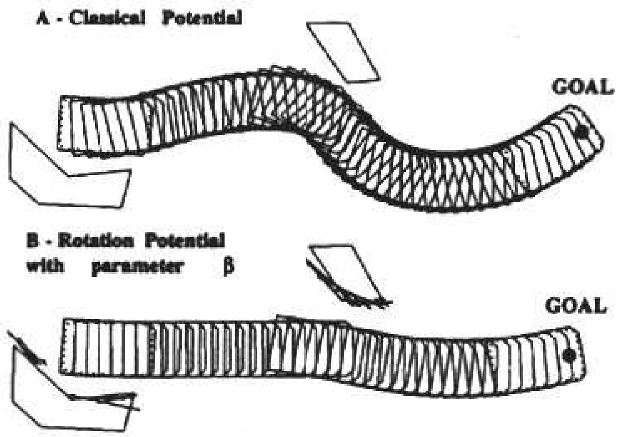
Limitations

- local minima (not a complete algorithm);
- if an object is concave: several minimal distances $\rho(q)$ could exist, resulting in oscillations between the two closest points.

Extensions

- Rotation potential field: the repulsive force is a function of the distance from the obstacle and the orientation of the robot relative to the obstacle. Gain factor reducing $F_{rep}(q)$ when an obstacle is parallel to the robot's direction of travel.
- Task potential fiel: filters out obstacles that should not affect the near-term potential based on robot velocity.

Give rise to smoother trajectories through space



© Siegwart et al., 2011, Fig. 6.13, p. 391

Extensions

Harmonic potential fields: no local minimum! Define $U(q)$ as the solution of the Laplace equation

$$\nabla^2 U(q) \equiv 0, q \in \Omega$$

where Ω represents the workspace the robot operates in. Boundary conditions: $U(q_{goal}) = 0$ and

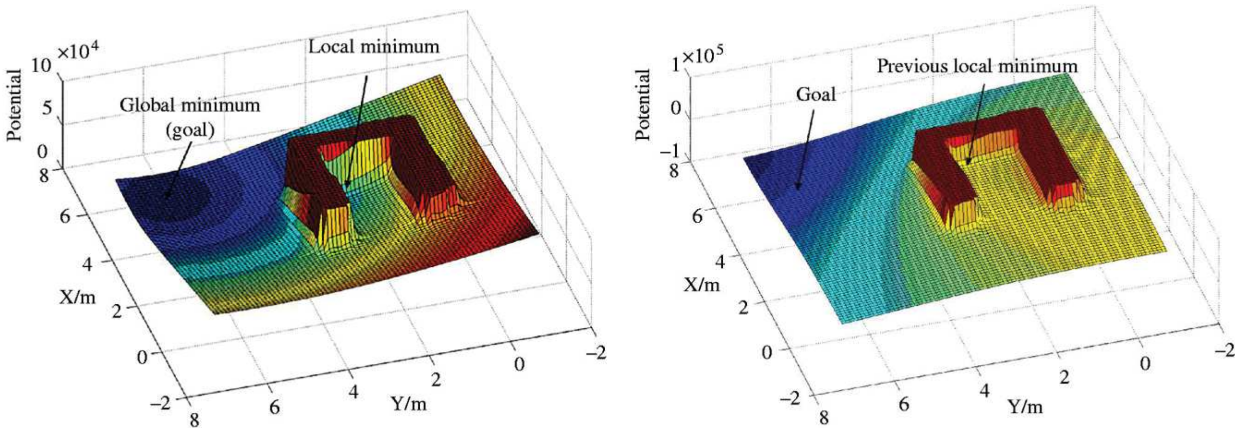
$$U(q) = f(q), q \in \Lambda$$

where Λ denotes the obstacles boundaries (Dirichlet condition). If $f(q) = const$, the robot follows a path perpendicular to the boundaries. Alternative (von Neumann):

$$\frac{\partial U(q)}{\partial q} = g(q), q \in \Lambda$$

If $g(q) = 0$, the robot follows a path parallel to the boundaries.

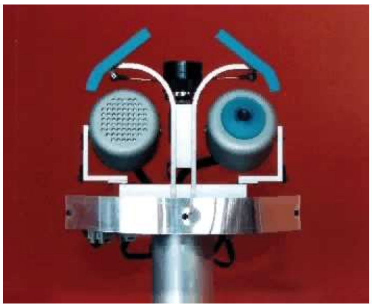
Extensions



© Zhang et al., *Industrial Robot: An International Journal* 2010

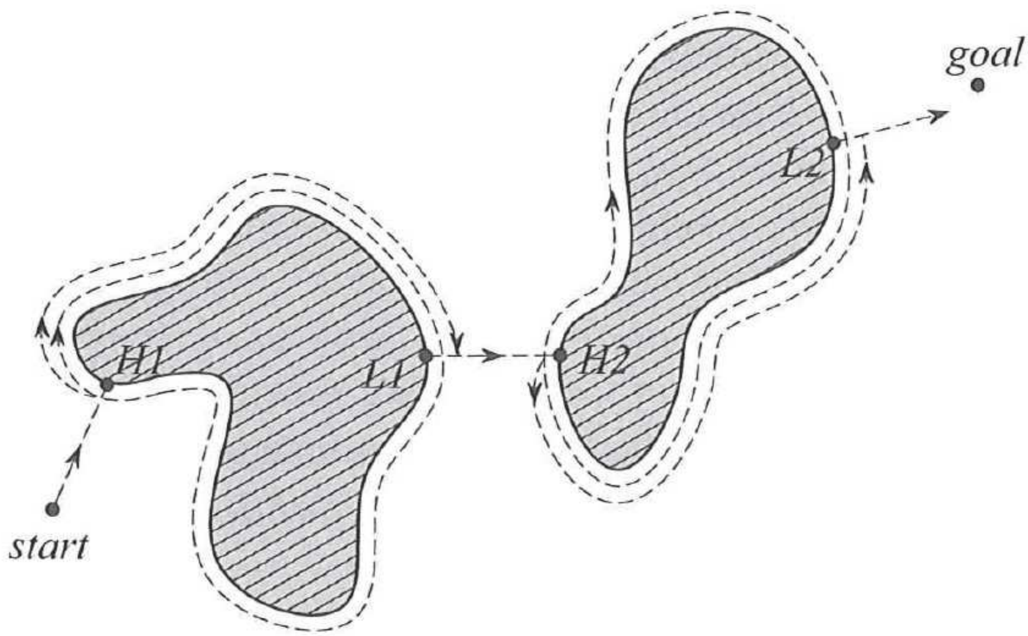
Obstacle Avoidance – Statement

The robot must be able to modify its path in real time based on sensor values.
Watch movies:

- 
- 

Implementation #1: Bug algorithm

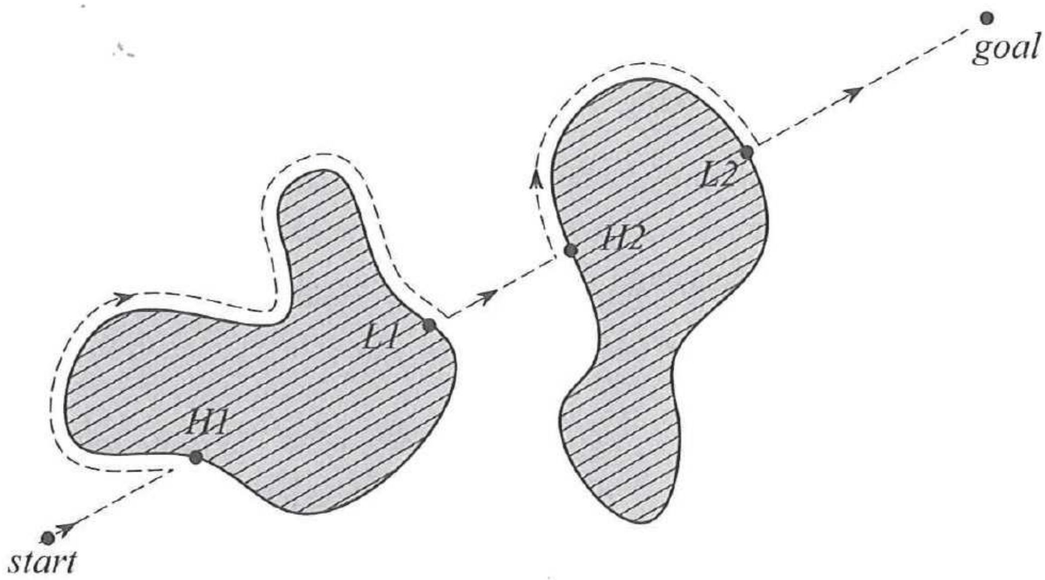
Simplest one could imagine: follow the contour of each obstacle, then departs from the point with the shortest distance toward the goal.



© Siegwart et al., 2011, Fig. 6.14, p. 394

Implementation #1: Bug algorithm

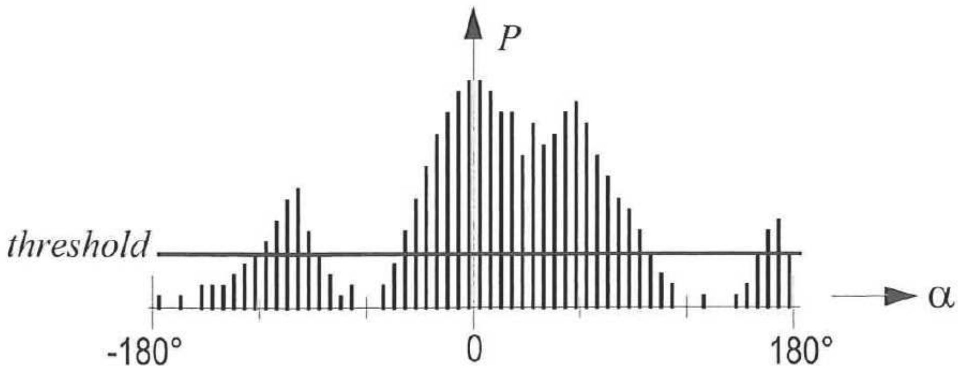
Bug2: the robot departs immediately when it is able to move towards the goal.



© Siegwart et al., 2011, Fig. 6.15, p. 394

Implementation #2: Vector field histogram

VFH creates a local map of the environment around the robot, and generates a polar histogram:



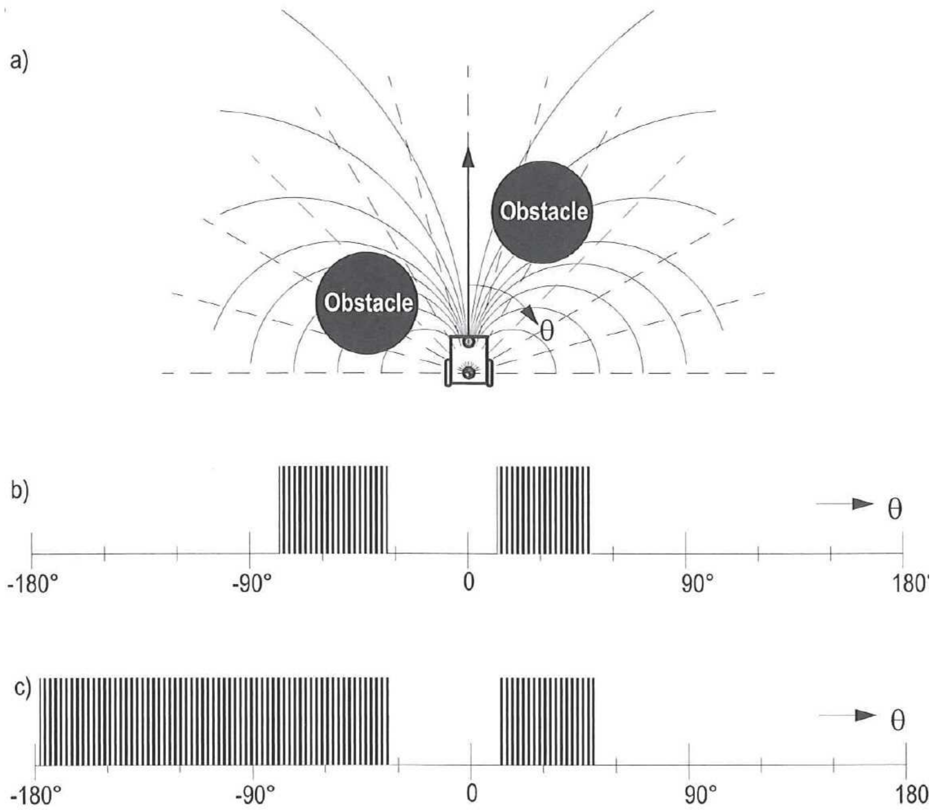
© Siegwart et al., 2011, Fig. 6.16, p. 398

Among the possible openings, one is chosen according to:

- the alignment of the robot path with the goal;
- the difference between the new direction and the current wheel orientation;
- the difference between the previously selected direction and the new direction.

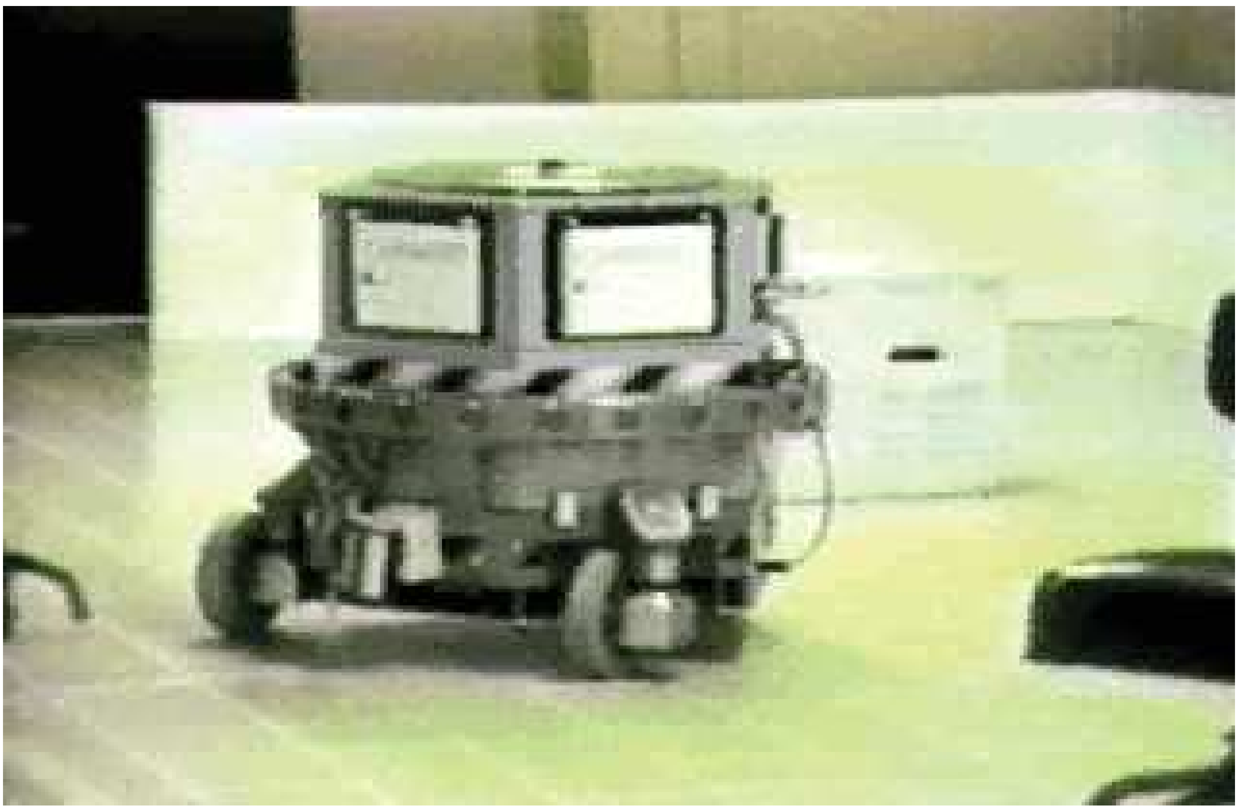
Implementation #2: Vector field histogram

VFH+: simplified model of the robot's possible trajectories



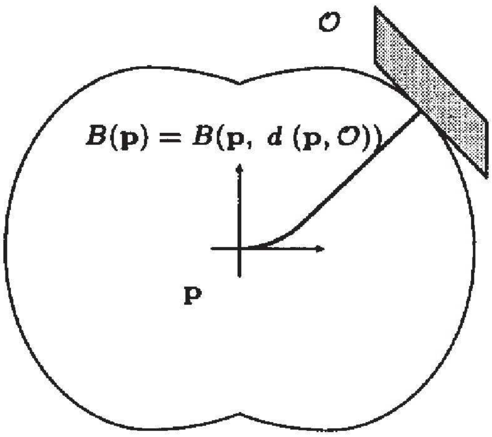
© Siegwart et al., 2011, Fig. 6.17, p. 399

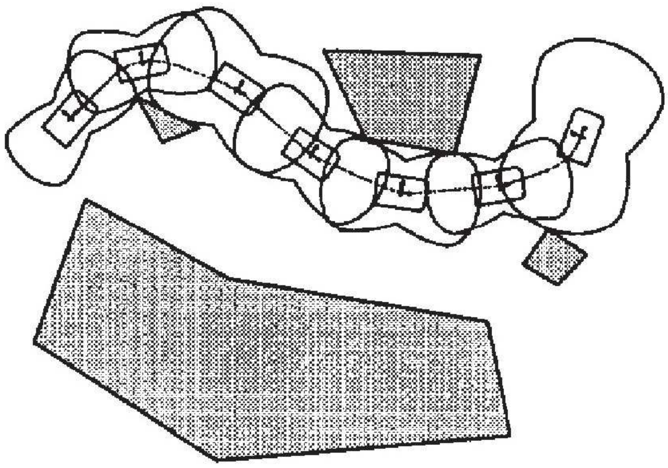
Implementation #2: Vector field histogram



Implementation #3: The bubble band technique

Bubble: maximum local subset of the free space which can be traveled without collision.
Idea: connect a series of bubble to make a global path. Real time changes are governed by "internal forces" which tend to minimize the "tension" between the adjacent bubbles.



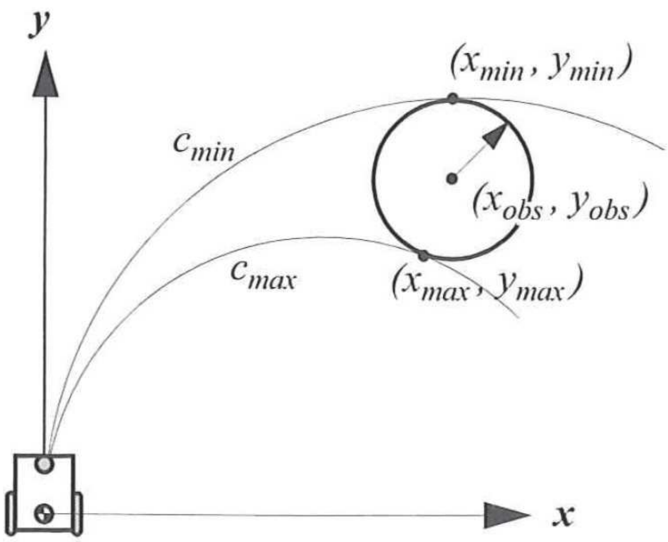


© Siegwart et al., 2011, Figs. 6.18 and 6.19, p. 400

Implementation #4: Curvature velocity techniques

CVM takes the actual kinematic constraints and even some dynamic constraints into account. Works in the velocity space.

- $-v_{max} < v < v_{max}$ and $-\omega_{max} < \omega < \omega_{max}$;
- obstacles block certain v and ω due to their position.



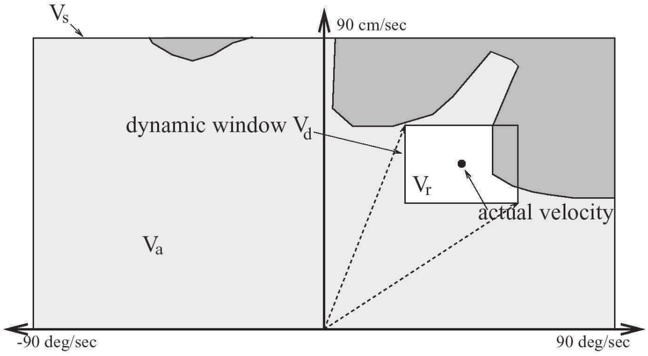
© Siegwart et al., 2011, Fig. 6.20, p. 402

New velocity is chosen from an objective function.

Implementation #5: Dynamic window approaches

Simple but very effective dynamic model.
Local dynamic window approach: the velocity space is all possible sets of tuples (v, ω) where v is the velocity and ω is the angular velocity: only circular arcs.

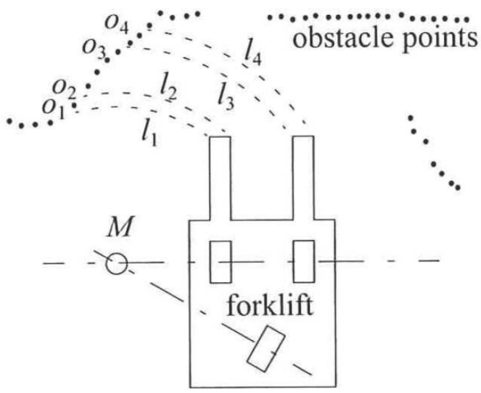
- Selection of a dynamic window of all (v, ω) that can be reached taking the acceleration capabilities into account.
- Reducing this window by keeping only the (v, ω) such that the vehicle can stop before hitting an obstacle.
- Motion direction is chosen from an objective function: fast forward motion, maintenance of large distances to obstacles, and alignment to the goal heading.



© Siegwart et al., 2011, Fig. 6.21, p. 403

Implementation #6: The Schlegel approach

Considers the dynamics as well as the actual shape of the robot.
Again, assumes that the robot moves in trajectories built up by circular arcs.
l_i: distance to collision between a single obstacle point i and the robot.



© Siegwart et al., 2011, Fig. 6.22, p. 405

Search space window V_s: all the possible speeds of the left and right wheels (dynamic constraints).
Objective function: best speed and direction by trading off goal direction, speed, and distance until collision.

Other implementations

Nearest diagram : similar to VFH but with more precise geometric, kinematic, and dynamic constraints.

Gradient method formulates a grid global path planning and allows generating continuous interpolations of the gradient direction in the grid.

Adding dynamic constraints : transforms obstacles into distances that depend on the breaking constraints.

Some more : fuzzy and neurofuzzy, neural network, Liapunov functions approaches, . . .

Overview

Vector Field Histogram (VFH)			Bug				
VFH* [322]	VFH+ [176, 323]	VFH [77]	Tangent Bug [161]	Bug2 [198, 199]	Bug1 [198, 199]	method	
circle	circle	simplistic	point	point	point	shape	model fidelity
basic	basic					kinematics	model fidelity
simplistic	simplistic					dynamics	model fidelity
essentially local	local	local	local	local	local	view	
histogram grid	histogram grid	histogram grid	local tangent graph			local map	other requisites
						global map	other requisites
						path planner	other requisites
sonars	sonars	range	range	tactile	tactile	sensors	
nonholonomic (GuideCane)	nonholonomic (GuideCane)	synchro-drive (hexagonal)				tested robots	
6 ... 242 ms	6 ms	27 ms				cycle time	performance
66 MHz, 486 PC	66 MHz, 486 PC	20 MHz, 386 AT				architecture	performance
fewer local minima	local minima	local minima, oscillating trajectories	efficient in many cases, robust	inefficient, robust	very inefficient, robust	remarks	

Overview

Dynamic window		Curvature velocity		Bubble band		method	
Global dynamic window [81]	Dynamic window approach [130]	Lane curvature method [168]	Curvature velocity method [291]	Bubble band [165]	Elastic band [166]		
circle	circle	circle	circle	C-space	C-space	shape	model fidelity
(holonomic)	exact	exact	exact	exact		kinematics	
basic	basic	basic	basic			dynamics	
global	local	local	local	local	global	view	
	obstacle line field	histogram grid	histogram grid			local map	other requisites
C-space grid				polygonal	polygonal	global map	
NF1				required	required	path planner	
180° FOV SCK laser scanner	24 sonars ring, 56 infrared ring, stereo camera	24 sonars ring, 30° FOV laser	24 sonars ring, 30° FOV laser			sensors	
holonomic (circular)	synchro-drive (circular)	synchro-drive (circular)	synchro-drive (circular)	various	various	tested robots	
6.7 ms	250 ms	125 ms	125 ms			cycle time	performance
450 MHz, PC	486 PC	200 MHz, Pentium	66 MHz, 486 PC			architecture	
turning into corridors	local minima	local minima	local minima, turning into corridors			remarks	

© Siegwart et al., 2011, Tab. 6.1, p. 408

Overview

Other				method	
Gradient method [171]	Global nearness diagram [225]	Nearness diagram [222, 223]	Schlegel [280]		
circle	circle (but general formulation)	circle (but general formulation)	polygon	shape	model fidelity
exact	(holonomic)	(holonomic)	exact	kinematics	
basic			basic	dynamics	
global	global	local	global	view	
	grid			local map	other requisites
local perceptual space	NF1		grid	global map	
fused			wavefront	path planner	
180° FOV distance sensor	180° FOV SCK laser scanner	180° FOV SCK laser scanner	360° FOV laser scanner	sensors	
nonholonomic (approx. circle)	holonomic (circular)	holonomic (circular)	synchrodrive (circular), tricycle (forklift)	tested robots	
100 ms (core algorithm: 10 ms)				cycle time	performance
266 MHz, Pentium				architecture	
		local minima	allows shape change	remarks	

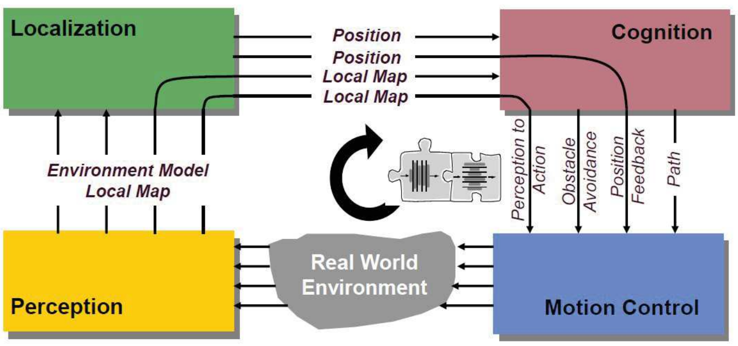
© Siegwart et al., 2011, Tab. 6.1, p. 409

Navigation Architectures

How do we combine path planning, obstacle avoidance, localization, and perceptual interpretation into one complete robot system for a real-world application?

Well-designed navigation architecture offers a number of concrete advantages:

- Modularity for code reuse and sharing;
- control localization;
- control decomposition.



Ⓒ Siegwart et al., 2011, Fig. 6.27, p. 417

Modularity for code reuse and sharing

Standard practice in software engineering: software modularity. Moreover, in the course of a single project the mobile robot hardware or its physical environmental characteristics can change dramatically. Examples:

- sick laser rangefinder $\leftrightarrow$ ultrasonic rangefinders;
- retain the obstacle avoidance module intact, even if the particular ranging sensor suite changes;
- design a new path-planning representation without changing the other modules;
- the nonholonomic obstacle avoidance module does not change when the robot's kinematic structure changes from a tricycle chassis to a differential-drive chassis.

Control localization

Multiple types of control functionalities: obstacle avoidance, path planning, path execution, . . .

Localizing each functionality to a specific unit in the architecture $\rightarrow$ individual testing and principled strategy for control composition.

Examples: collision avoidance, high-level planning and task decision making, . . .

Exhaustive tests in simulation (even without a direct connection to the physical robot).

Localization of control can enable a specific learning algorithm to be applied to just one aspect of a mobile robot's overall control system.

Techniques for decomposition

Decompositions identify axes along which we can justify discrimination of robot software into distinct modules.

- Temporal decomposition: real-time and non real-time demands on operations.
- Control decomposition: identifies how the various control outputs (within the mobile robot architecture) combine to yield the physical actions.

Temporal decomposition

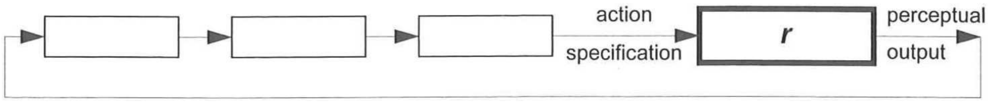
© Siegwart et al., 2011, Fig. 6.23, p. 411

- lowest level: guaranteed fast cycle time (e.g. 40 Hz);
- quasi real-time: 0.1 second response time, with allowable worst-case individual cycle times;
- tactical layer: decision making affecting the immediate actions (temporal constraints);
- strategic and offline: decisions affecting the long-term behavior.

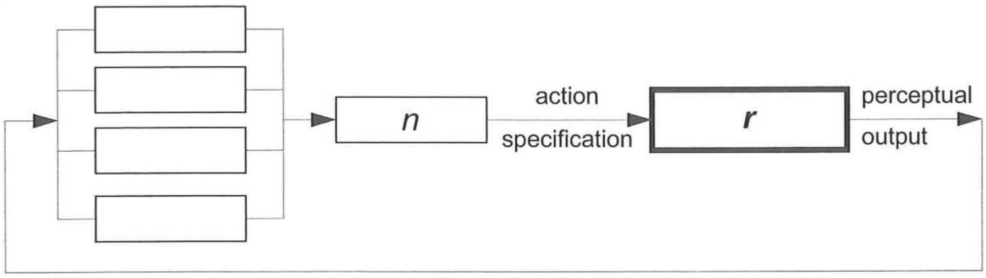
Control decomposition

Identifies the way in which each module's output contributes to the overall robot control outputs.

- Perfectly linear, or sequential pathway: predictability and verifiability



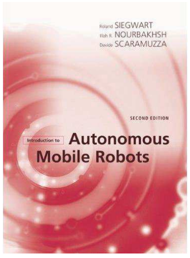
- Perfectly parallel pathway: contains a combination step n



Summary

- Planning and execution of the movement are autonomous, cognitive, decisions made by a mobile robot to decide how to reach its goal.
- It requires two key and complementary competences: path planning and (real-time) obstacle avoidance.
- Two general approaches exist for path planning: graph search and potential field planning.
- General potential field planning is prone to get trapped into local minima. Possible solution: harmonic potential fields.
- Obstacle avoidance can be implemented in many different ways, if possible accounting for geometric, kinematic, and dynamic constraints.
- Good navigation architecture requires software modularity: code reuse and sharing, control localization, and control decomposition.

References



Autonomous Mobile Robots (2nd Edition)
Siegwart et al.; The MIT Press, 2011
`http://www.mobilerobots.ethz.ch/`
Chapter 6