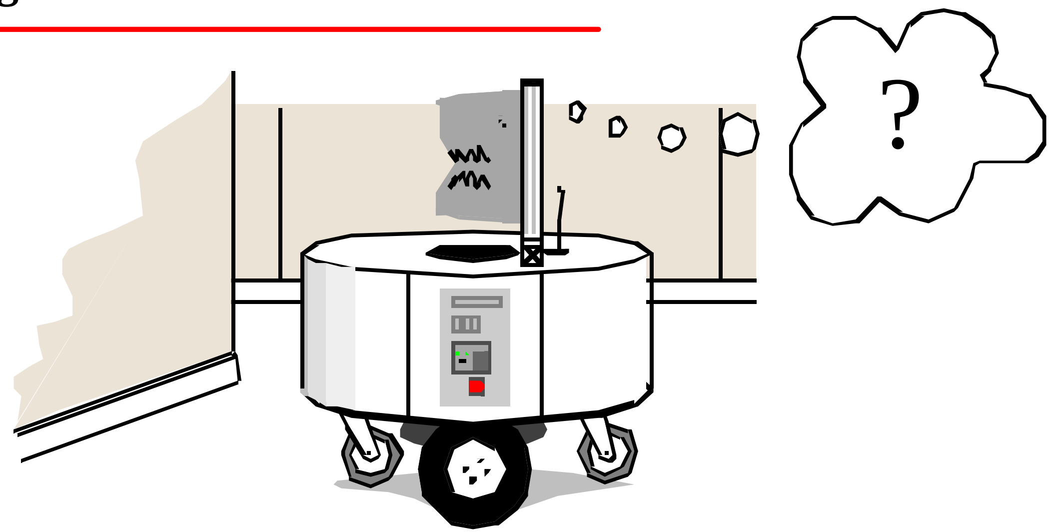
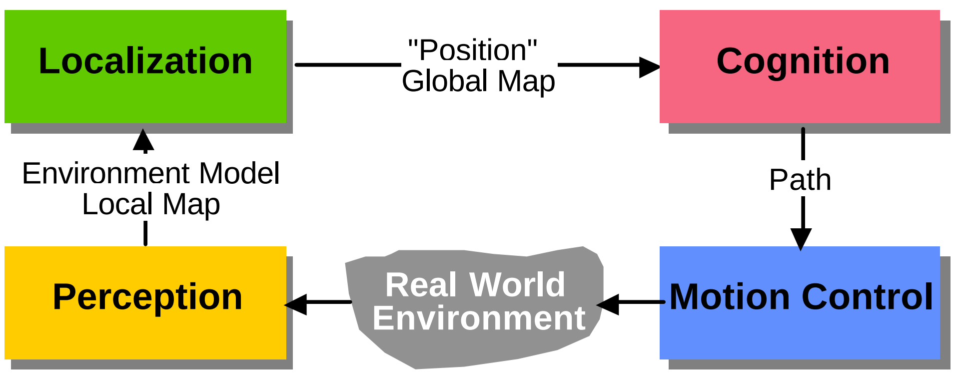
Planning and Navigation

Where am I going? How do I get there?





Competencies for Navigation I

- 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 exis ts (e.g. sales man problem).*

Competencies for Navigation II

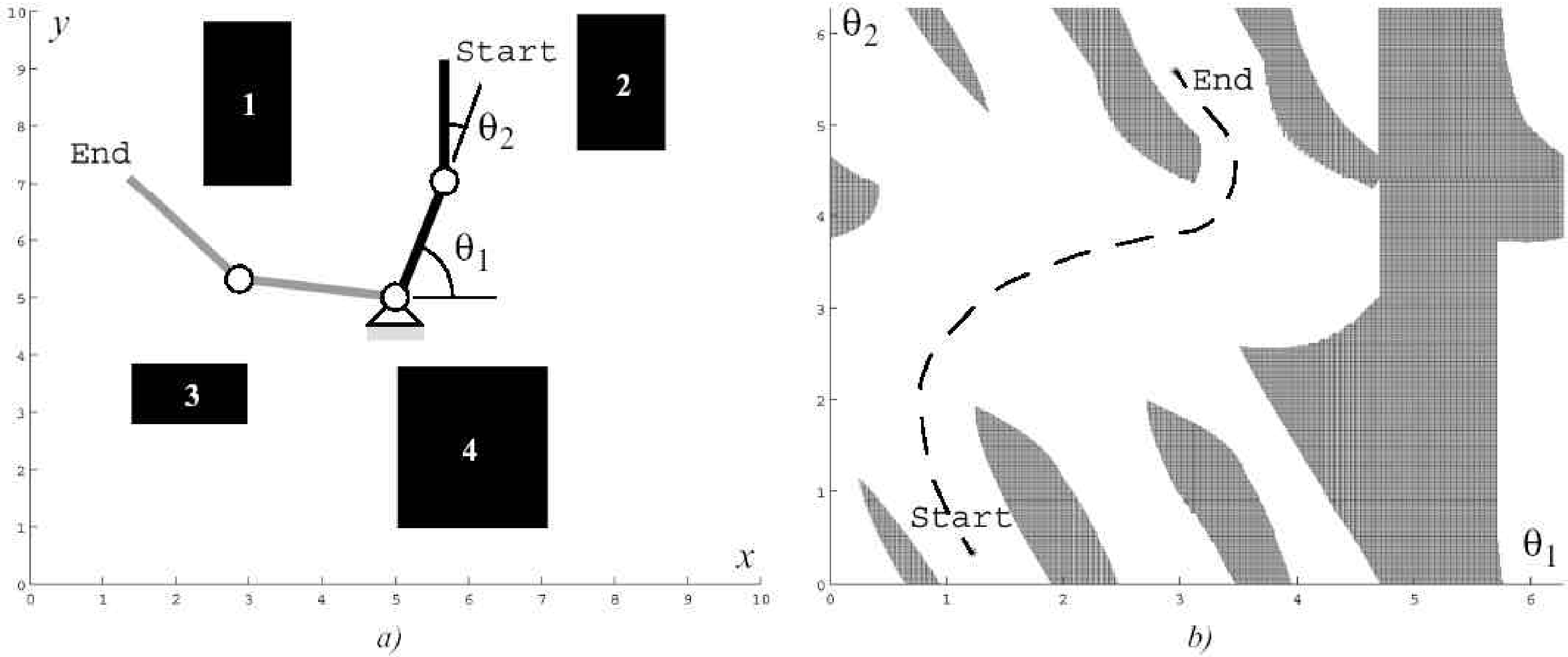
- However, in mobile robotics the knowledge of about the environment and situation is usually only partially known and is uncertain.
  - *makes the task much more difficult*
  - *requires multiple tasks running in parallel, some for planning (global), some to guarantee "survival of the robot".*
- Robot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

Global Path Planing

- 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:
  - *Visibility Graph*
  - *Voronoi Diagram*
  - *Cell Decomposition -> Connectivity Graph*
  - *Potential Field*

Path Planning: Configuration Space

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

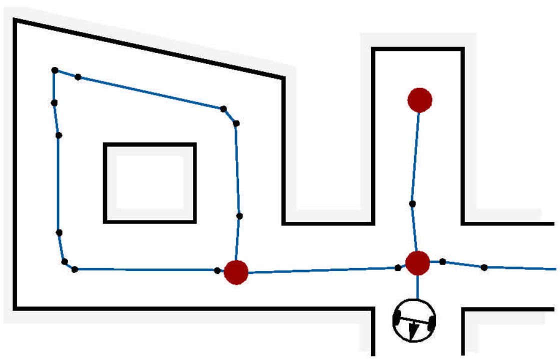


- What is the configuration space of a mobile robot?

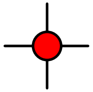
Path Planning Overview

1. Road Map, Graph construction

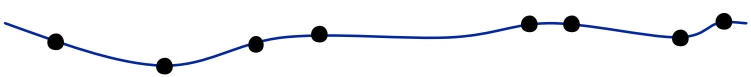
- *Identify a set of routes within the free space*



- Where to put the nodes?
- Topology-based:
  - *at distinctive locations*

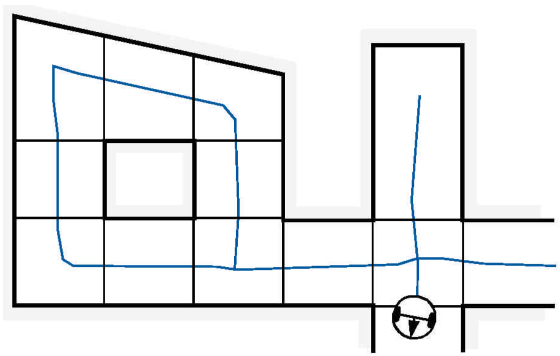


- Metric-based:
  - *where features disappear or get visible*



2. Cell decomposition

- *Discriminate between free and occupied cells*

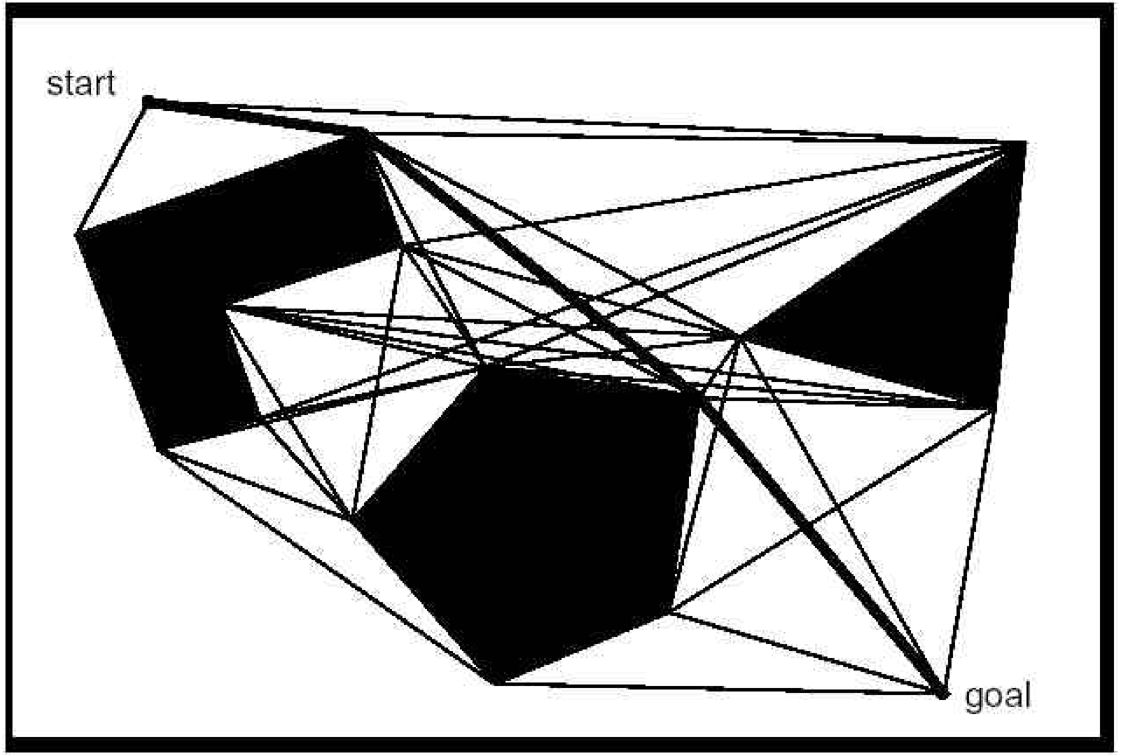


- Where to put the cell boundaries?
- Topology- and metric-based:
  - *where features disappear or get visible*

3. Potential Field

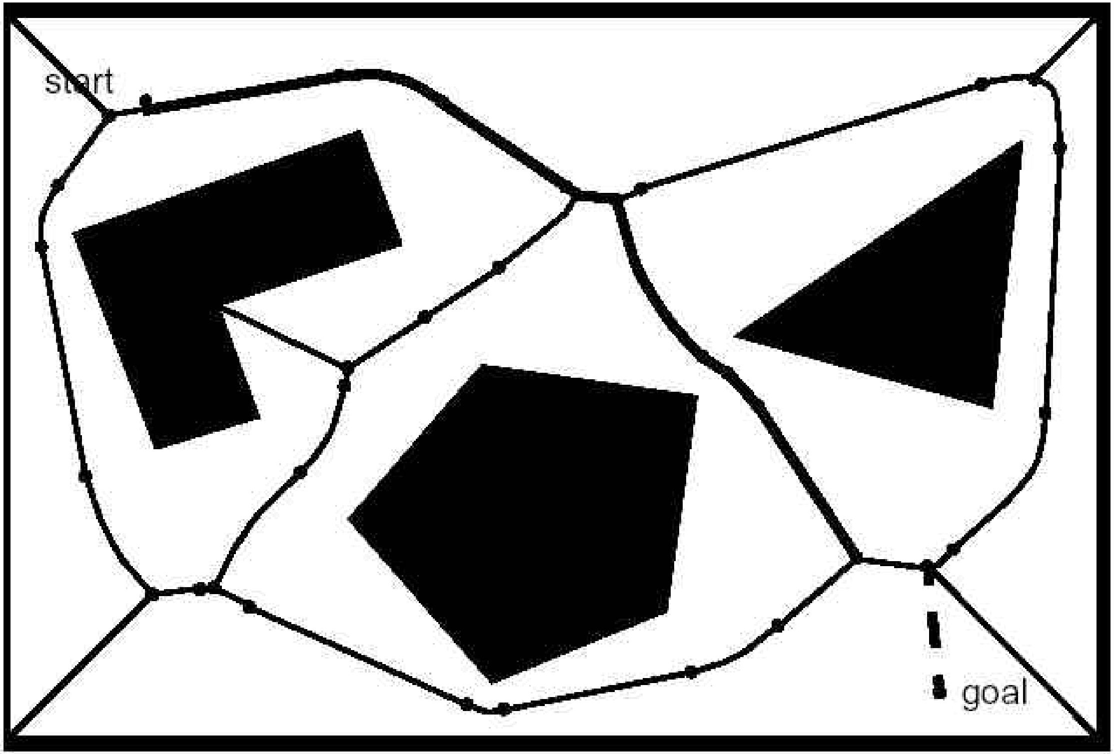
- *Imposing a mathematical function over the space*

Road-Map Path Planning: **Visibility Graph**



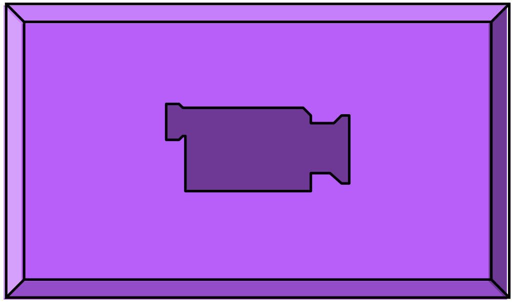
- Shortest path length
- Grow obstacles to avoid collisions

Road-Map Path Planning: **Voronoi Diagram**



- Easy executable: Maximize the sensor readings
- Works also for map-building: Move on the Voronoi edges

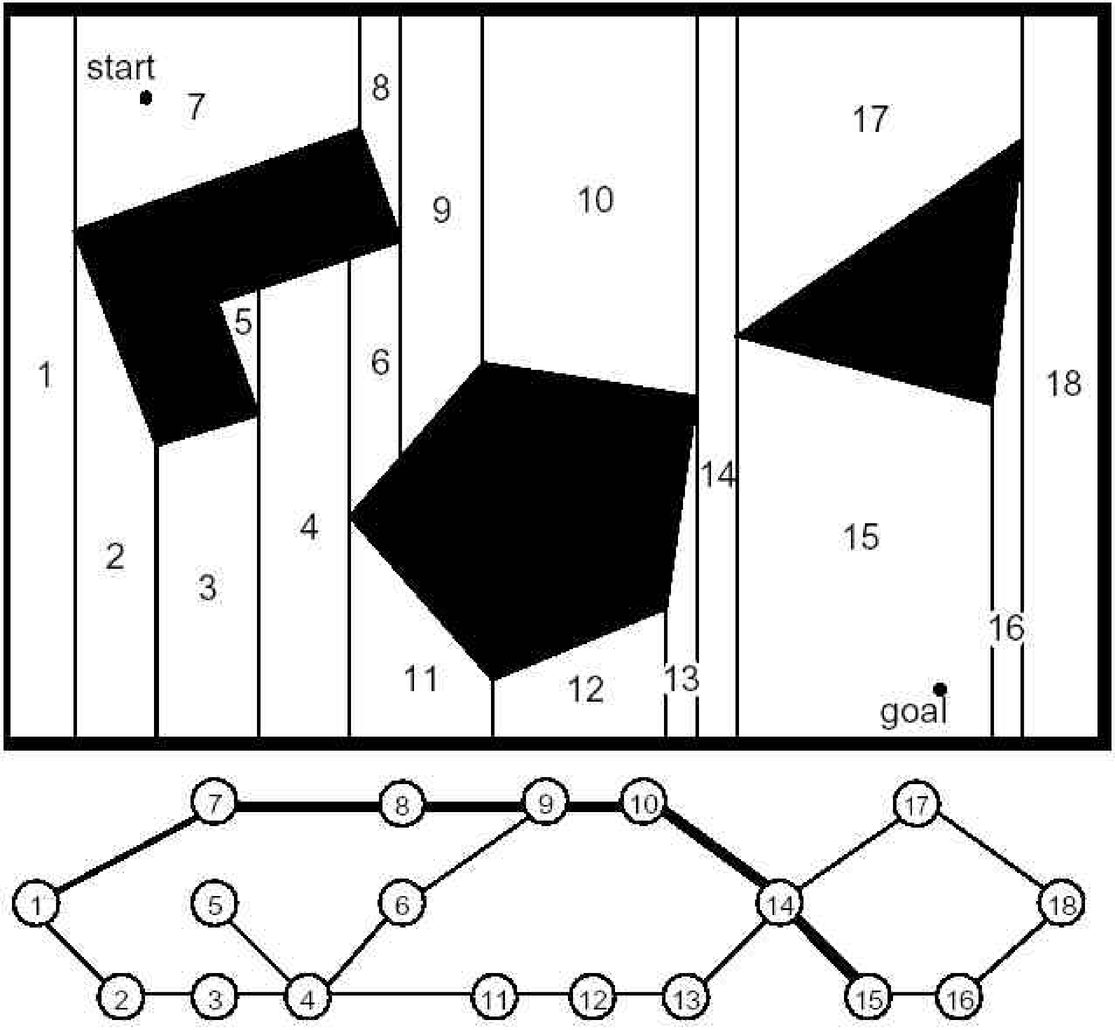
Road-Map Path Planning: **Voronoi, Sysquake Demo**



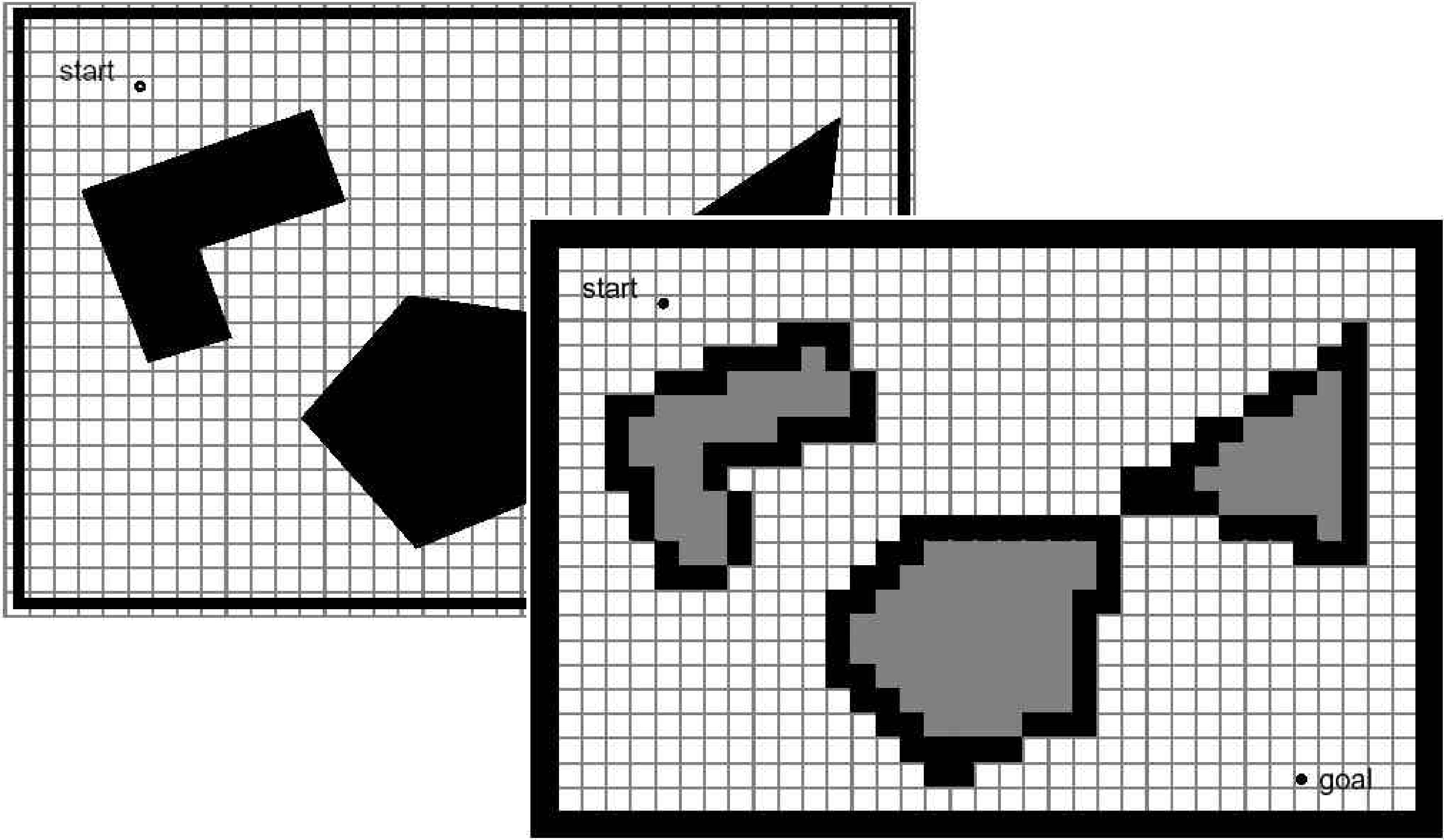
Road-Map Path Planning: **Cell Decomposition**

- 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.*

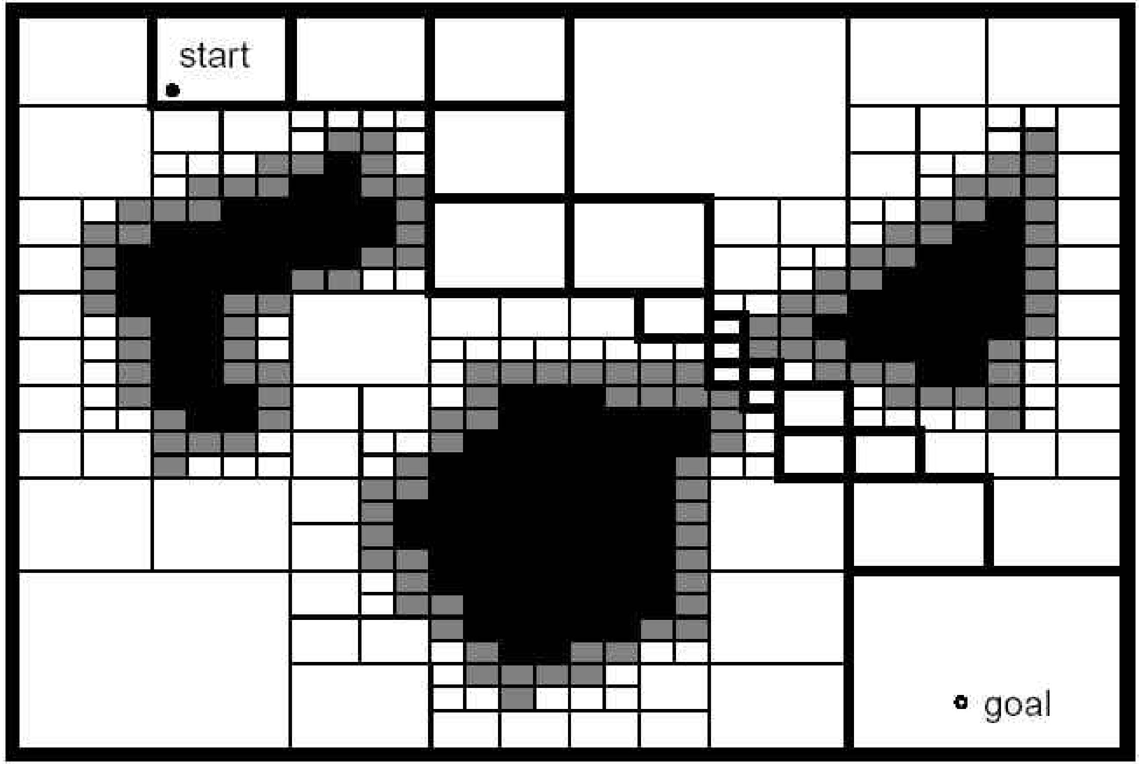
Road-Map Path Planning: **Exact Cell Decomposition**



Road-Map Path Planning: **Approximate Cell Decomposition**

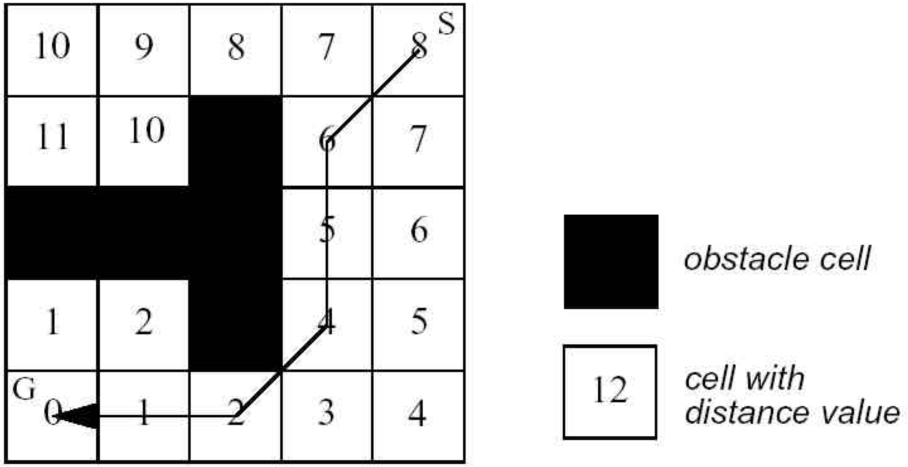


Road-Map Path Planning: **Adaptive Cell Decomposition**



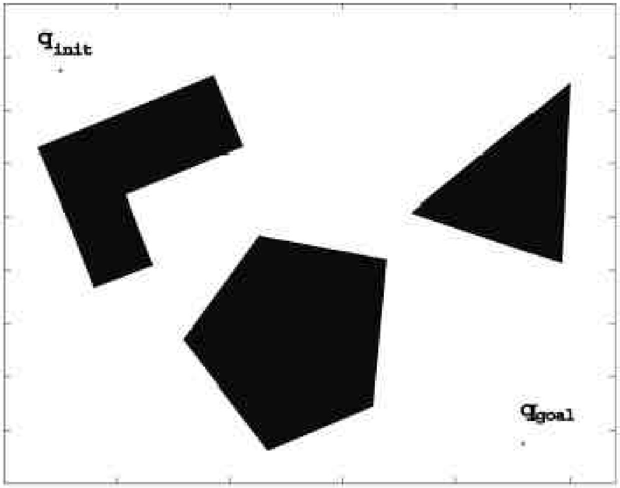
Road-Map Path Planning: **Path / Graph Search Strategies**

- Wavefront Expansion NF1 (see also later)

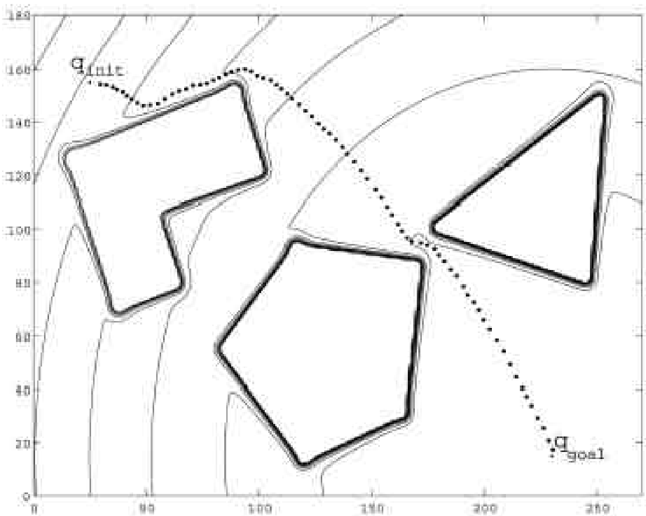


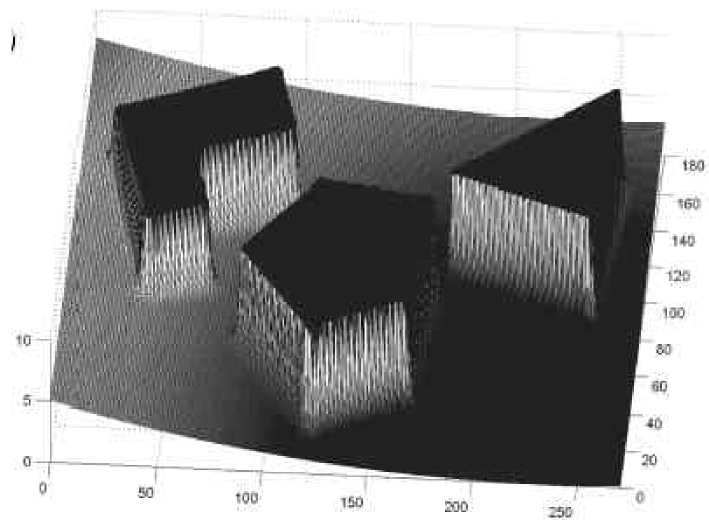
- Breadth-First Search
- Depth-First Search
- Greedy search and A*

Potential Field Path Planning



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





Potential Field Path Planning: **Potential Field Generation**

- Generation of potential field function *U(q)*
  - *attracting (goal) and repulsing (obstacle) fields*
  - *summing up the fields*
  - *functions must be differentiable*
- Generate artificial force field *F(q)*

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

- Set robot speed (v_x, v_y) proportional to the force $F(q)$ generated by the field
  - *the force field drives the robot to the goal*
  - *if robot is assumed to be a point mass*

Potential Field Path Planning: **Attractive Potential Field**

- Parabolic function representing the Euclidean distance $\|q - q_{goal}\|$ to the goal

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

- Attracting force converges linearly towards 0 (goal)

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

Potential Field Path Planning: **Repulsing Potential Field**

- Should generate a barrier around all the obstacle
  - *strong if close to the obstacle*
  - *not influence if fare from the obstacle*

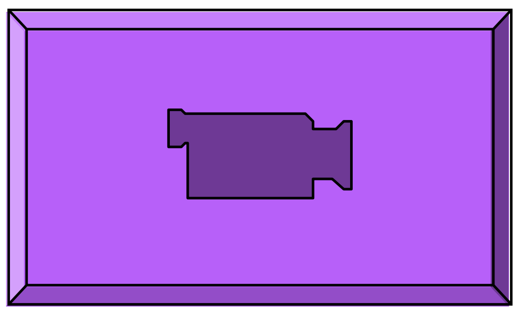
$$U_{rep}(q) = \begin{cases} \frac{1}{2}k_{rep}\left(\frac{1}{\rho(q)} - \frac{1}{\rho_0}\right)^2 & \text{if } \rho(q) \le \rho_0 \\ 0 & \text{if } \rho(q) \ge \rho_0 \end{cases}$$

  - $\rho(q)$ *: minimum distance to the object*
  - *Field is positive or zero and* tends to infinity *as q gets closer to the object*

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

Potential Field Path Planning: **Sysquake Demo**

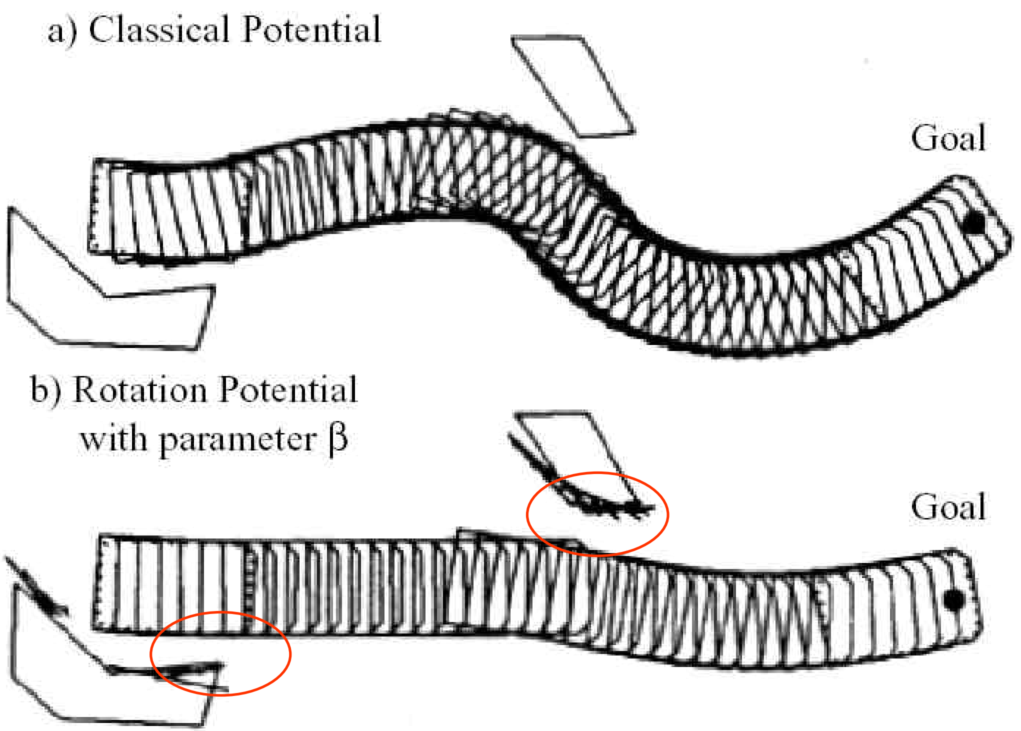
- Notes:
  - *Local minima problem exists*
  - *problem is getting more complex if the robot is not considered as a point mass*
  - *If objects are convex there exists situations where several minimal distances exist ® can result in oscillations*



Potential Field Path Planning: **Extended Potential Field Method**

Khatib and Chatila

- Additionally a *rotation potential field* and a *task potential field* in introduced
- Rotation potential field
  - *force is also a function of robots orientation to the obstacle*
- Task potential field
  - *Filters out the obstacles that should not influence the robots movements, i.e. only the obstacles in the sector Z in front of the robot are considered*



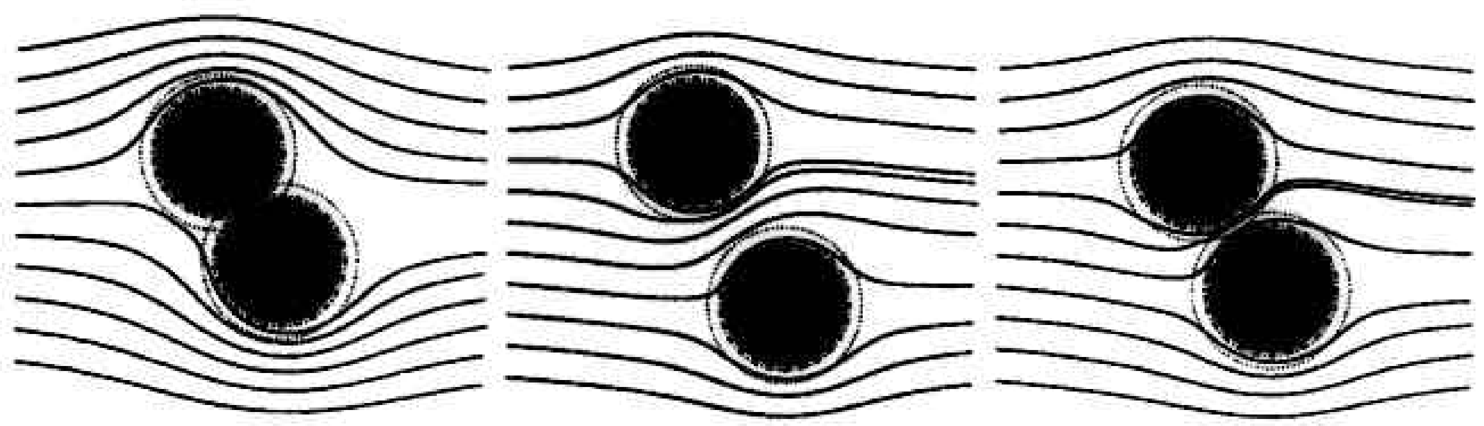
Potential Field Path Planning: **Potential Field using a Dyn. Model**

Monatana et at.

- Forces in the polar plane
  - *no time consuming transformations*
- Robot modeled thoroughly
  - *potential field forces directly acting on the model*
  - *filters the movement -> smooth*
- Local minima
  - *set a new goal point*

Potential Field Path Planning: **Using Harmonic Potentials**

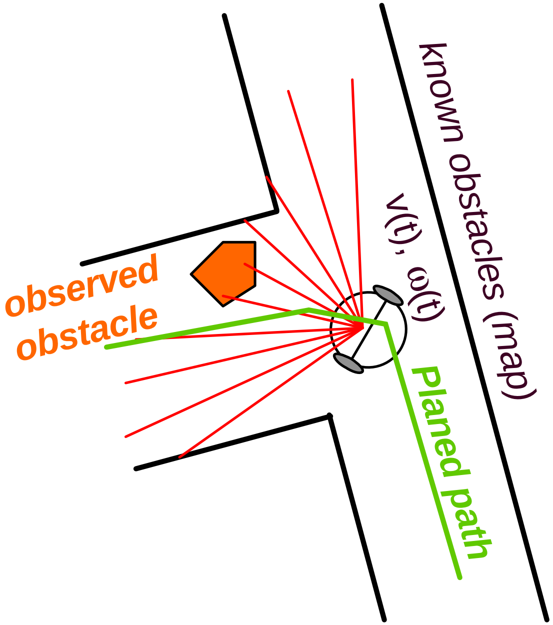
- Hydrodynamics analogy
  - *robot is moving similar to a fluid particle following its stream*
- Ensures that there are no local minima



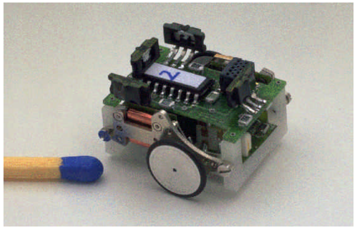
- Note:
  - *Complicated, only simulation shown*

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*

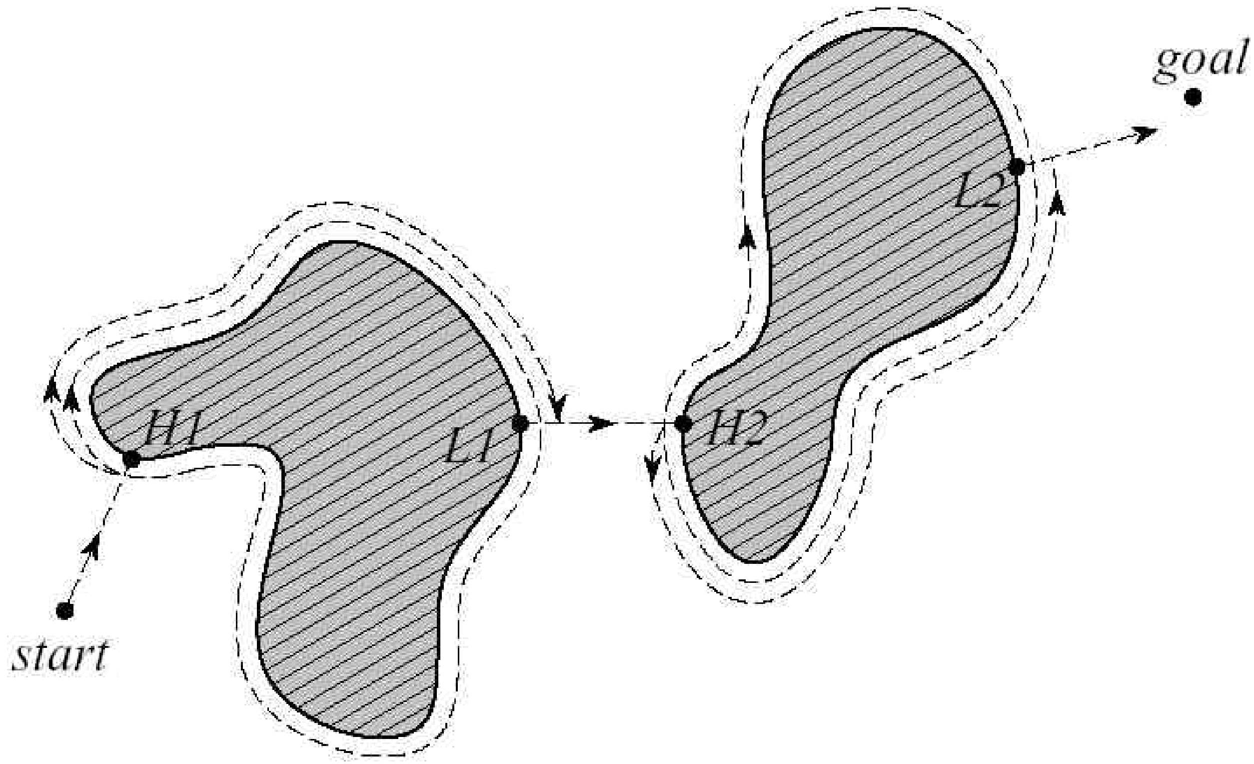


- Example: Alice



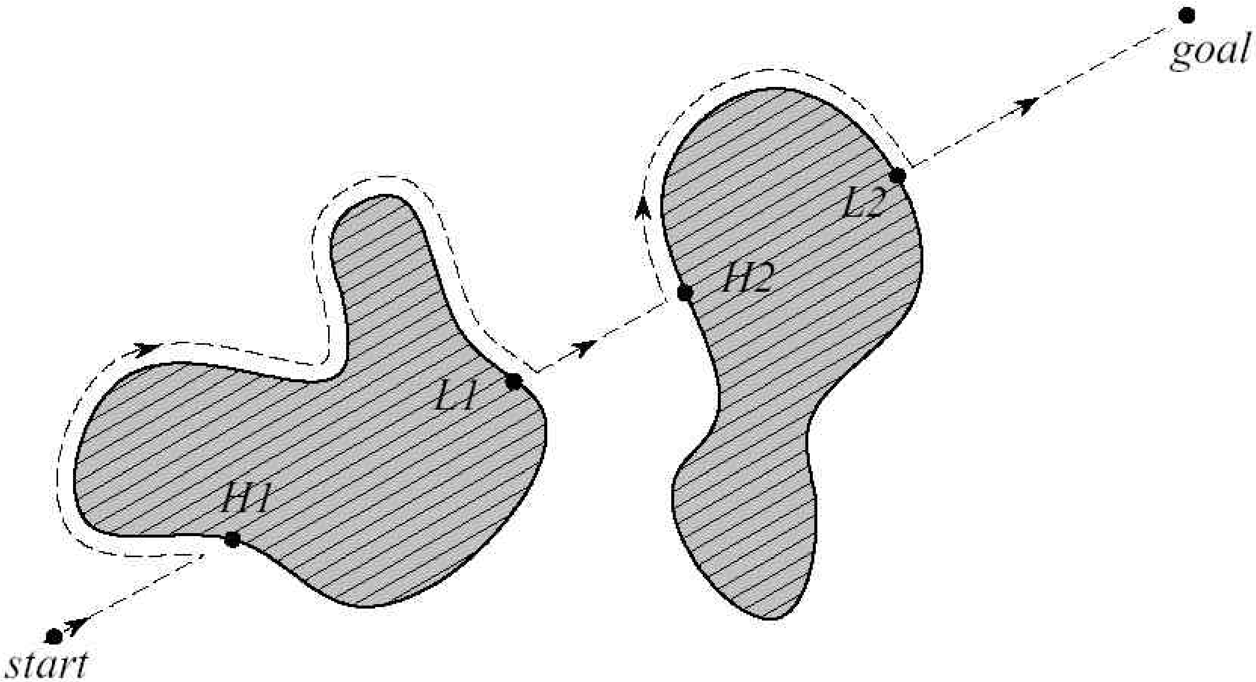
Obstacle Avoidance: **Bug1**

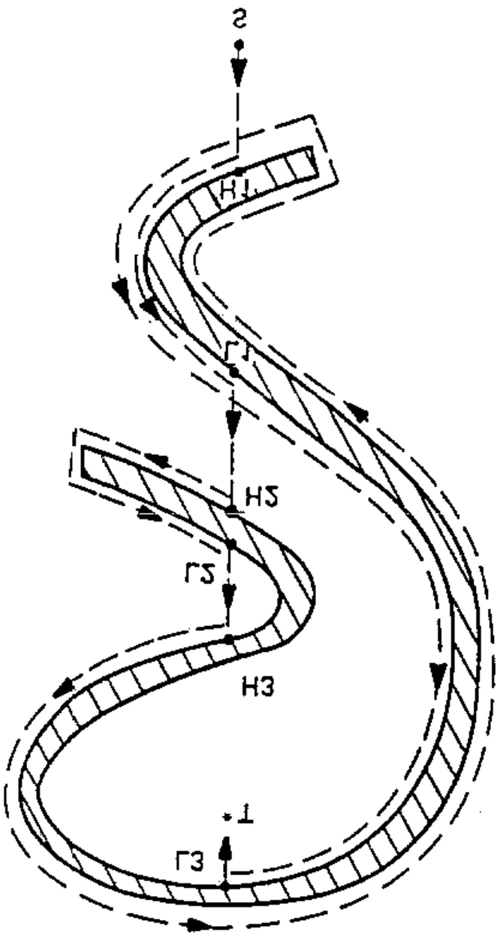
- Following along the obstacle to avoid it
- Each encountered obstacle is once fully circled before it is left at the point closest to the goal



Obstacle Avoidance: **Bug2**

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



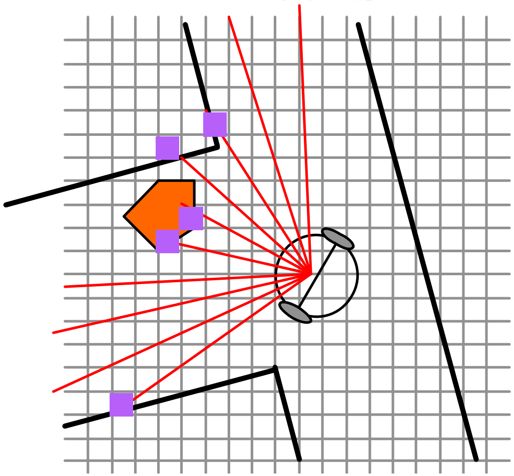


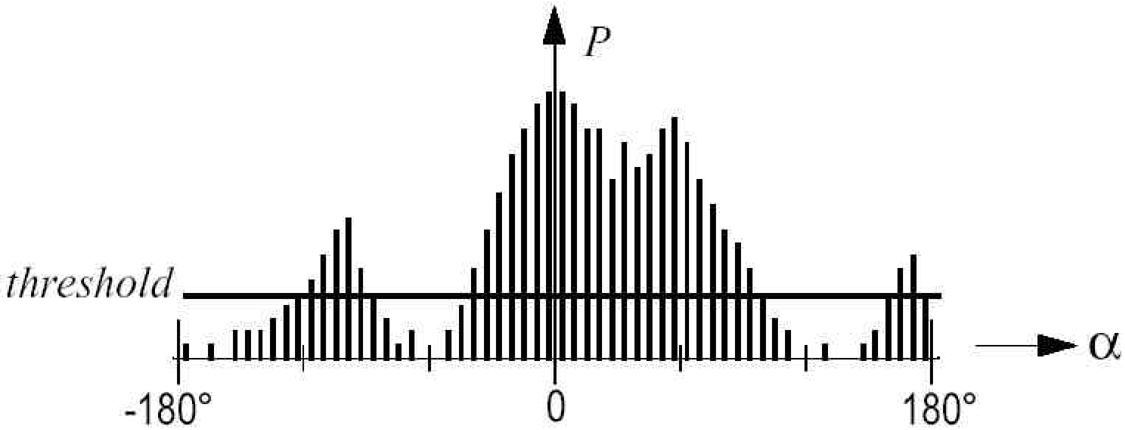
Obstacle Avoidance: **Vector Field Histogram (VFH)**

Borenstein et al.

- Environment represented in a grid (2 DOF)
  - *cell values equivalent to the probability that there is an obstacle*
- Reduction in different steps to a 1 DOF histogram
  - *calculation of steering direction*
  - *all openings for the robot to pass are found*
  - *the one with lowest cost function G is selected*

$$G = a \cdot \text{target\_direction} + b \cdot \text{wheel\_orientation} + c \cdot \text{previous\_direction}$$

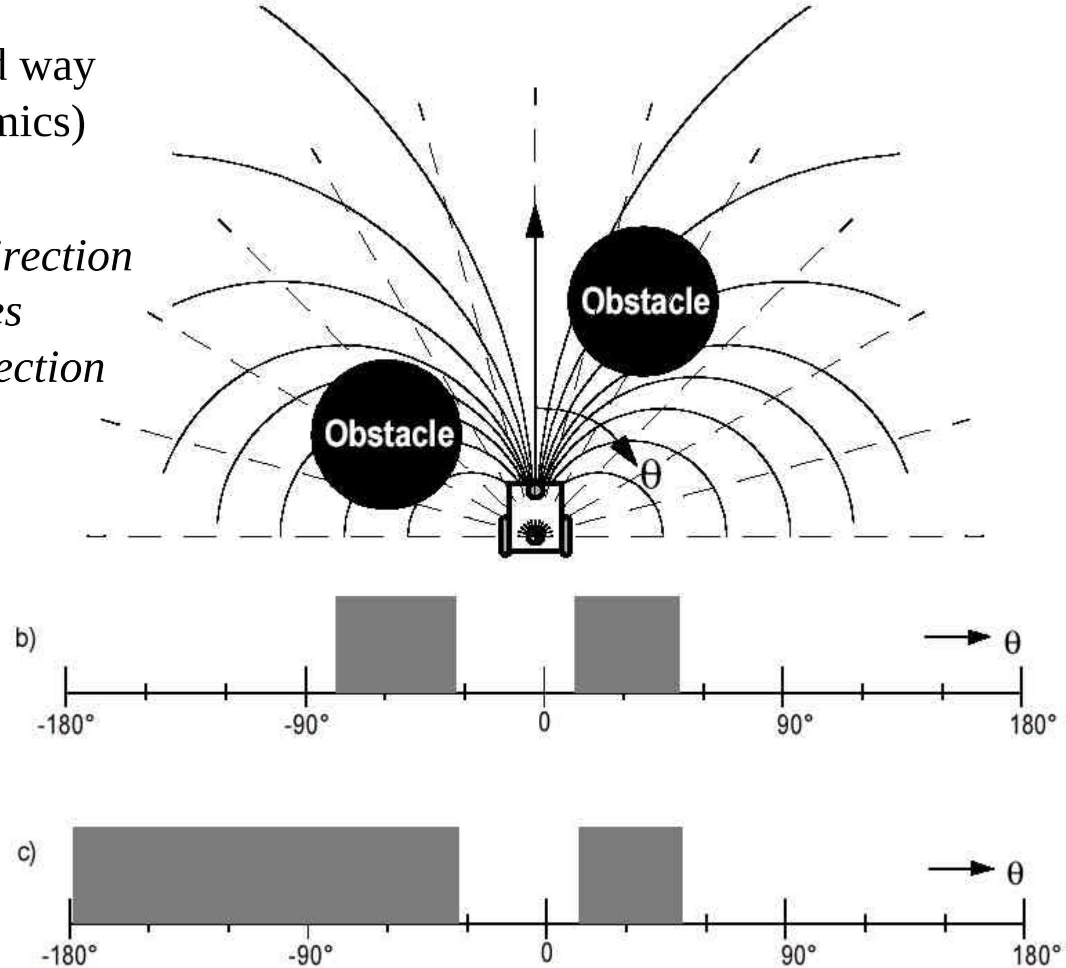




Obstacle Avoidance: **Vector Field Histogram + (VFH+)**

Borenstein et al.

- Accounts also in a very simplified way for the moving trajectories (dynamics)
  - *robot moving on arcs*
  - *obstacles blocking a given direction also blocks all the trajectories (arcs) going through this direction*



Obstacle Avoidance: **Video VFH**

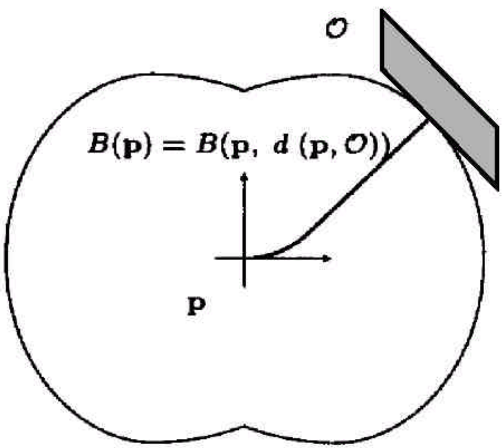
Borenstein et al.

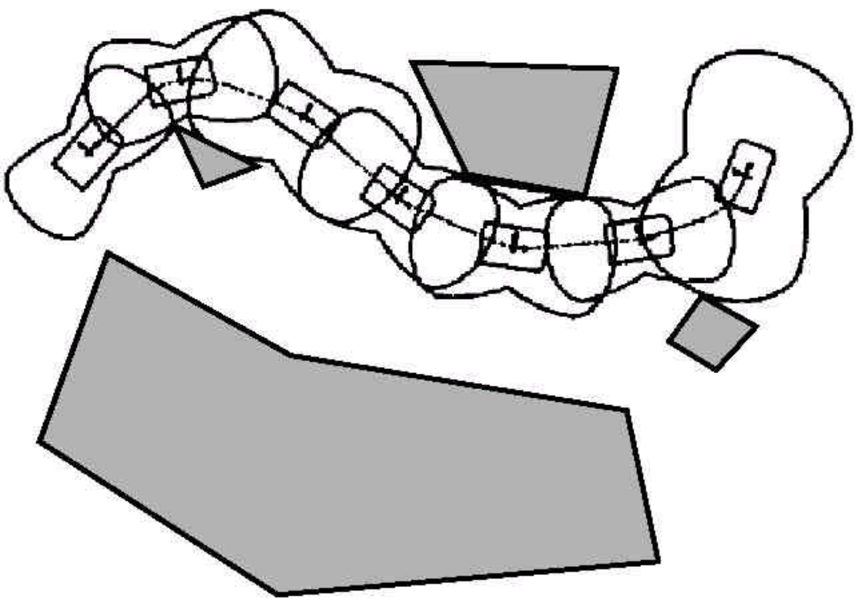
- Notes:
  - *Limitation if narrow areas (e.g. doors) have to be passed*
  - *Local minimum might not be avoided*
  - *Reaching of the goal can not be guaranteed*
  - *Dynamics of the robot not really considered*

Obstacle Avoidance: **The Bubble Band Concept**

Khatib and Chatila

- Bubble = Maximum free space which can be reached without any risk of collision
  - ➢ *generated using the distance to the object and a simplified model of the robot*
  - ➢ *bubbles are used to form a band of bubbles which connects the start point with the goal point*

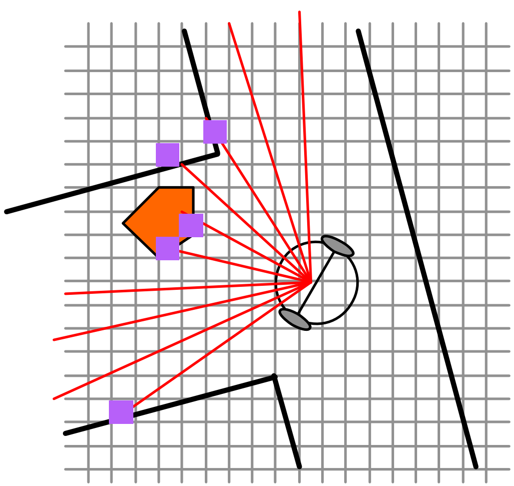


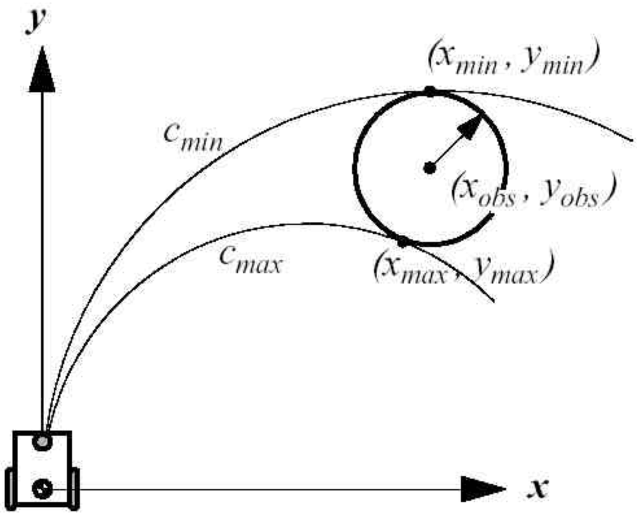


Obstacle Avoidance: Basic **Curvature Velocity Methods** (CVM)

Simmons et al.

- Adding *physical constraints* from the robot and the environment on the *velocity space* (v, ω) of the robot
  - *Assumption that robot is traveling on arcs (c= w / v)*
  - *Acceleration constraints:*
  - *Obstacle constraints: Obstacles are transformed in velocity spac e*
  - *Objective function to select the optimal speed*





Obstacle Avoidance: Lane Curvature Velocity Methods (CVM)

Simmons et al.

- Improvement of basic CVM
  - *Not only arcs are considered*
  - *lanes are calculated trading off lane length and width to the closest obstacles*
  - *Lane with best properties is chosen using an objective function*

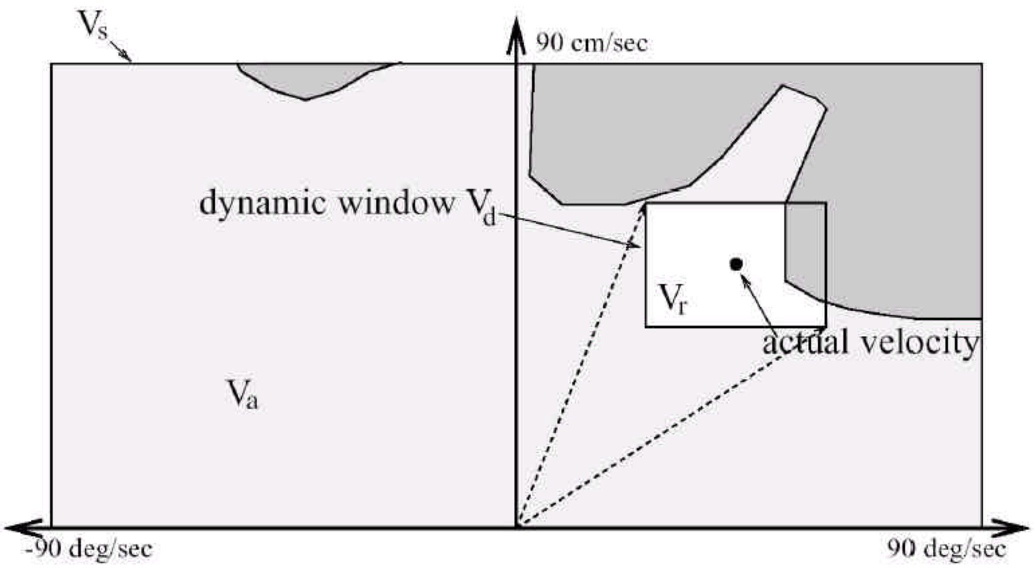
- Note:
  - *Better performance to pass narrow areas (e.g. doors)*
  - *Problem with local minima persists*

Obstacle Avoidance: **Dynamic Window Approach**

Fox and Burgard, Brock and Khatib

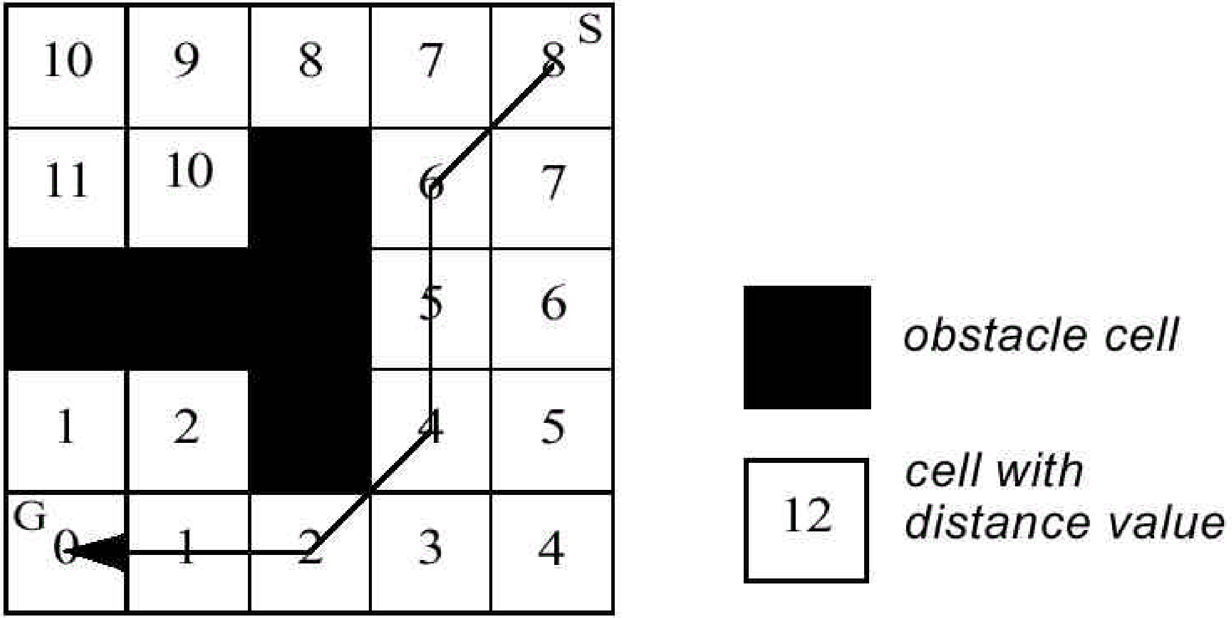
- The kinematics of the robot is considered by searching a well chosen velocity space
  - *velocity space -> some sort of configuration space*
  - *robot is assumed to move on arcs*
  - *ensures that the robot comes to stop before hitting an obstacle*
  - *objective function is chosen to select the optimal velocity*

$$O = a \cdot heading(v, \omega) + b \cdot velocity(v, \omega) + c \cdot dist(v, \omega)$$



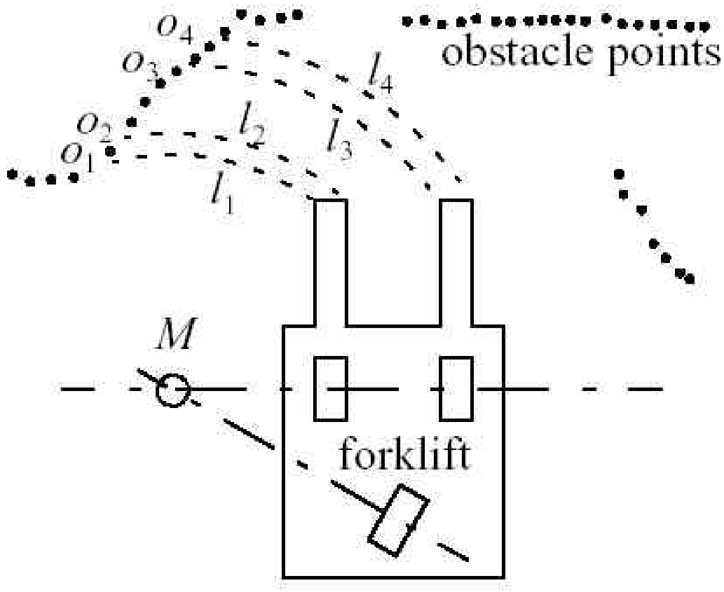
Obstacle Avoidance: Global Dynamic Window Approach

- Global approach:
  - *This is done by adding a minima-free function named NF1 (wave-propagation) to the objective function* O *presented above.*
  - *Occupancy grid is updated from range measurements*



Obstacle Avoidance: **The *Schlegel* Approach**

- Some sort of a variation of the dynamic window approch
  - *takes into account the shape of the robot*
  - *Cartesian grid and motion of circular arcs*
  - *NF1 planner*
  - *real time performance achieved by use of precalculated table*

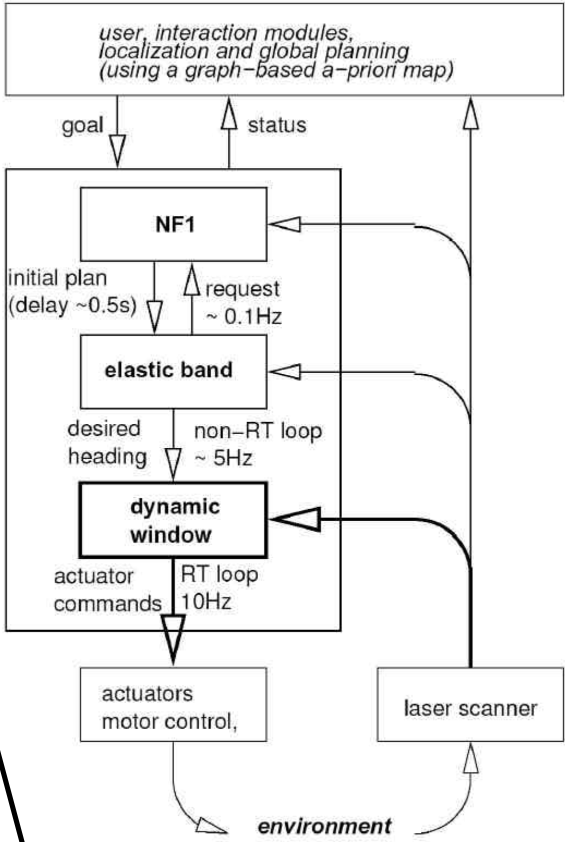


Obstacle Avoidance: **The EPFL-ASL approach**

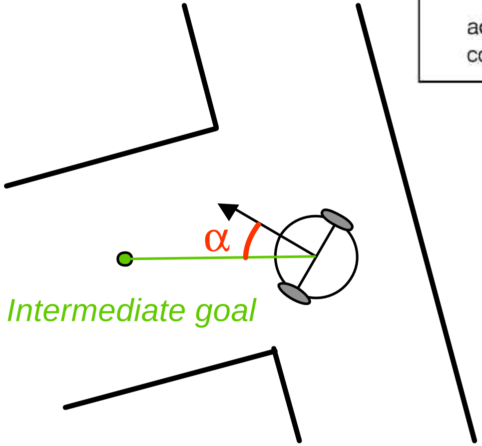
- Dynamic window approach with global path planing
  - *Global path generated in advance*
  - *Path adapted if obstacles are encountered*
  - *dynamic window considering also the shape of the robot*
  - *real-time because only max speed is calculated*
- Selection (Objective) Function:

$$Max(a \cdot speed + b \cdot dist + c \cdot goal\_heading)$$

  - $speed = v / v_{max}$
  - $dist = L / L_{max}$
  - $goal\_heading = 1 - (\alpha - \omega T) / \pi$



- Matlab-Demo
  - *start Matlab*
  - *cd demoJan (or cd E:\demo\demoJan)*
  - *demoX*



Obstacle Avoidance: **Other approaches**

- Behavior based
  - *difficult to introduce a precise task*
  - *reachability of goal not provable*

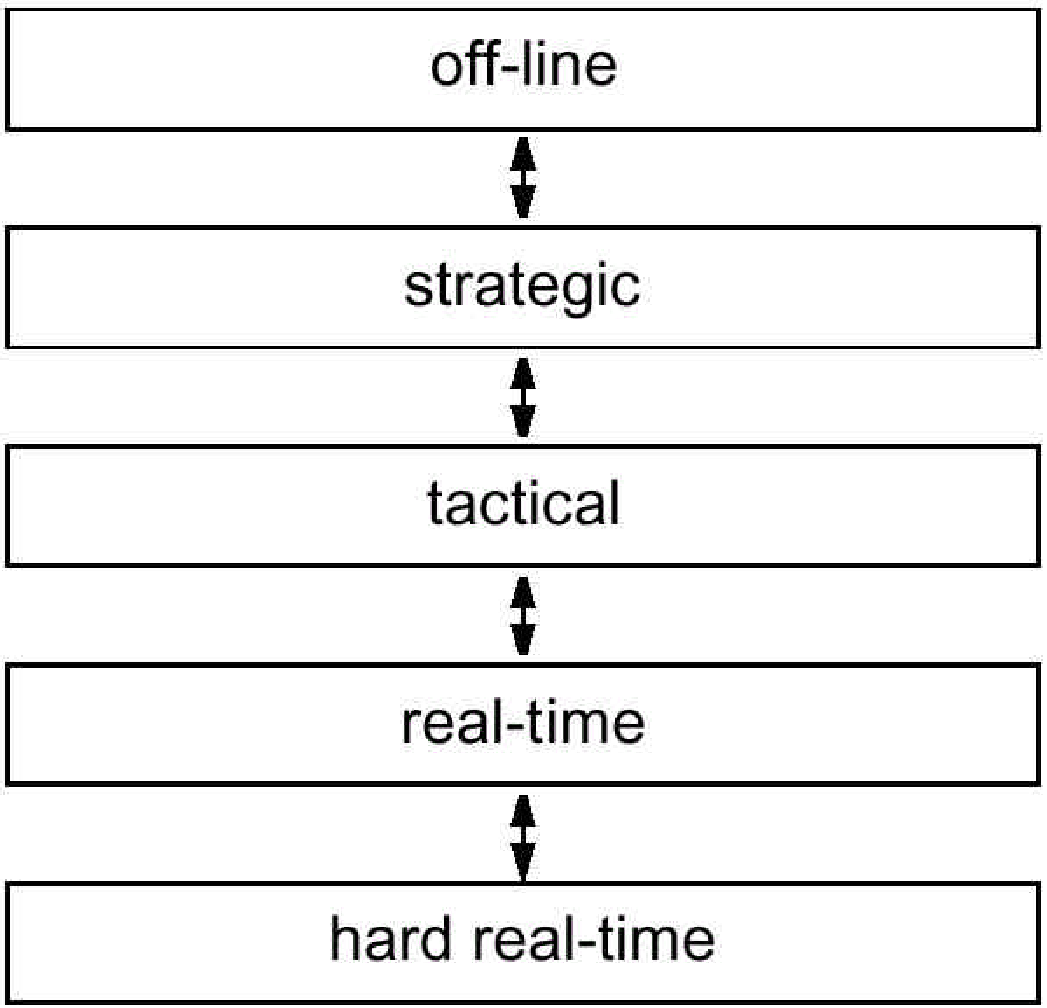
- Fuzzy, Neuro-Fuzzy
  - *learning required*
  - *difficult to generalize*

Bug			Bubble band		Vector Field Histogram (VFH)			method	
Tangent Bug [82]	Bug2 [101, 102]	Bug1 [101, 102]	Bubble band [85]	Elastic band [86]	VFH* [149]	VFH+ [92, 150]	VFH [43]		
point	point	point	C-space	C-space	circle	circle	simplistic	shape	model fidelity
			exact		basic	basic		kinematics	
					simplistic	simplistic		dynamics	
local	local	local	local	global	essentially local	local	local	view	
local tangent graph					histogram grid	histogram grid	histogram grid	local map	other requisites
			polygonal	polygonal				global map	
			required	required				path planner	
range	tactile	tactile			sonars	sonars	range	sensors	
			various	various	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	
efficient in many cases, robust	inefficient, robust	very inefficient, robust			fewer local minima	local minima	local minima, oscillating trajectories	remarks	

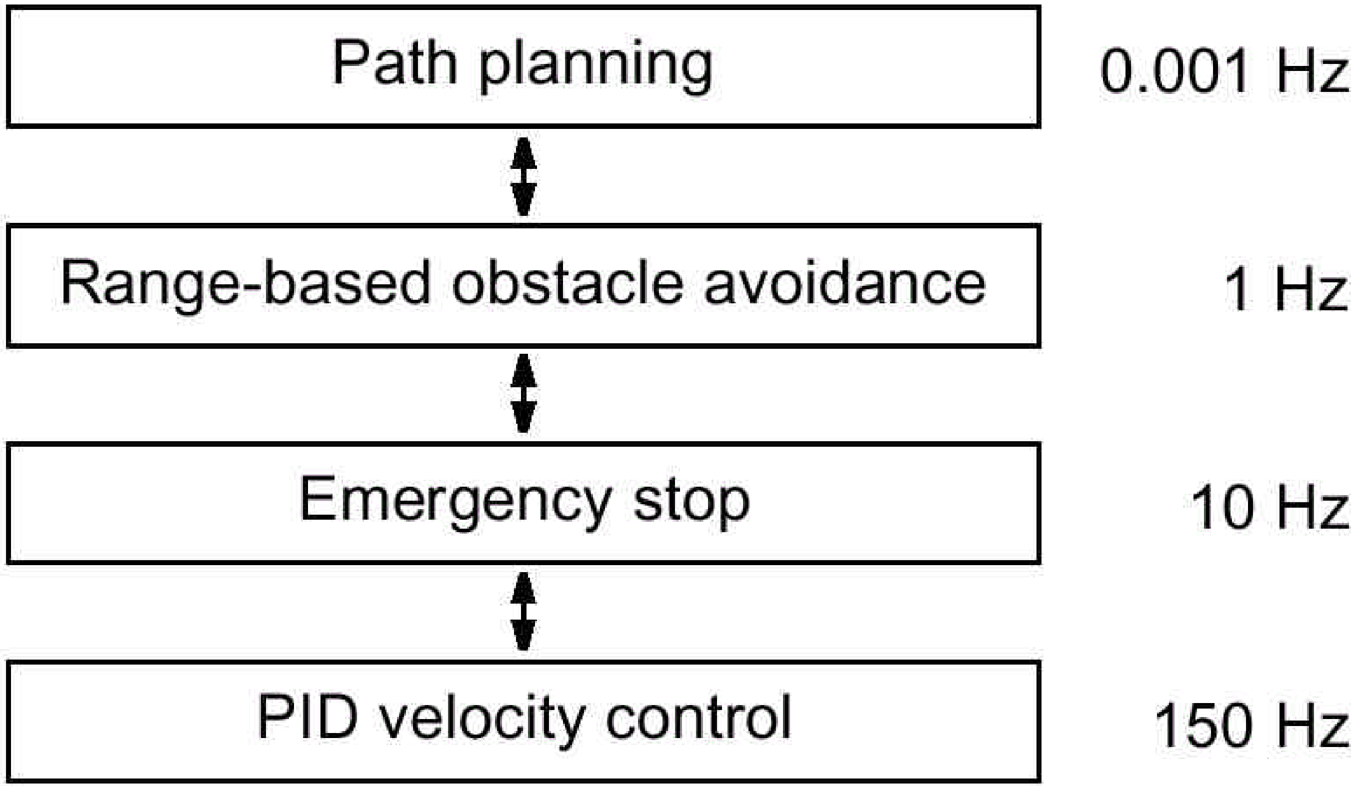
Dynamic window		Curvature velocity		method	
Global dynamic window [44]	Dynamic window approach [69]	Lane curvature method [87]	Curvature velocity method [135]		
circle	circle	circle	circle	shape	model fidelity
(holonomic)	exact	exact	exact	kinematics	model fidelity
basic	basic	basic	basic	dynamics	model fidelity
global	local	local	local	view	
	obstacle line field	histogram grid	histogram grid	local map	other requisites
C-space grid				global map	other requisites
NF1				path planner	other requisites
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)	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	performance
turning into corridors	local minima	local minima	local minima, turning into corridors	remarks	

Other					method	
Gradient method [89]	Global nearness diagram [110]	Nearness diagram [107, 108]	ASL approach [122]	Schlegel [128]		
circle	circle (but general formulation)	circle (but general formulation)	polygon	polygon	shape	model fidelity
exact	(holonomic)	(holonomic)	exact	exact	kinematics	model fidelity
basic			basic	basic	dynamics	model fidelity
global	global	local	local	global	view	
	grid		grid		local map	other requisites
local perceptual space	NF1			grid	global map	other requisites
fused			graph (topological), NF1	wavefront	path planner	other requisites
180° FOV distance sensor	180° FOV SCK laser scanner	180° FOV SCK laser scanner	2x 180° FOV SCK laser scanner	360° FOV laser scanner	sensors	
nonholonomic (approx. circle)	holonomic (circular)	holonomic (circular)	differential drive (octagonal, rectangular)	synchrodrive (circular), tricycle (forklift)	tested robots	
100 ms (core algorithm: 10 ms)			100 ms (core algorithm: 22 ms)		cycle time	performance
266 MHz, Pentium			380 MHz, G3		architecture	performance
		local minima	turning into corridors	allows shape change	remarks	

Generic temporal decomposition

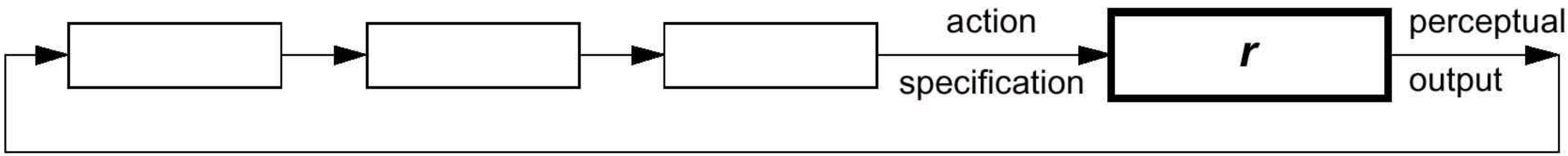


4-level temporal decomposition

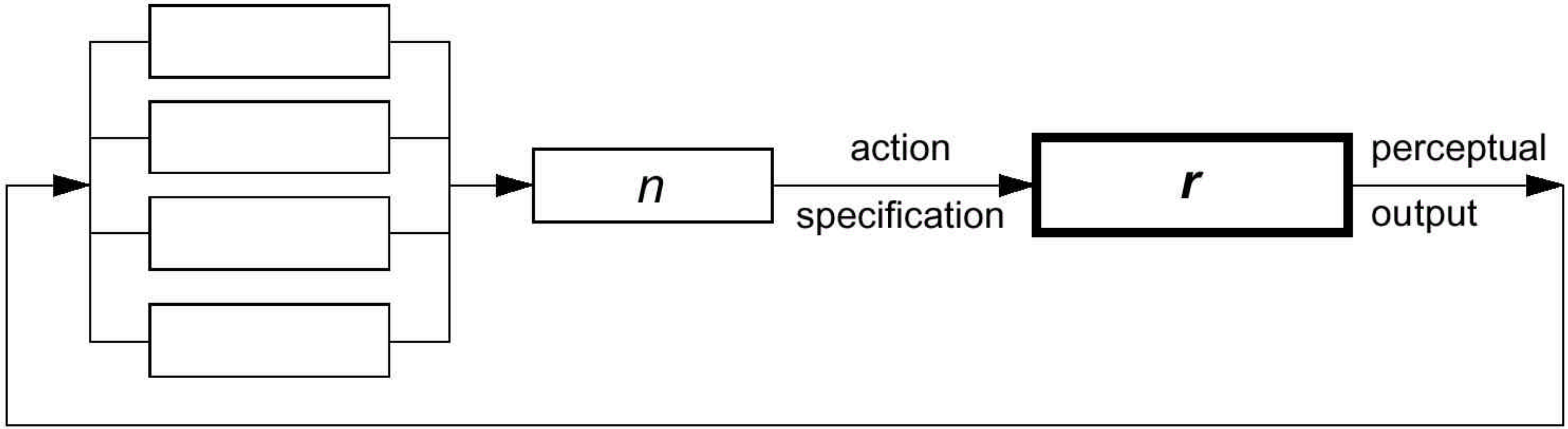


Control decomposition

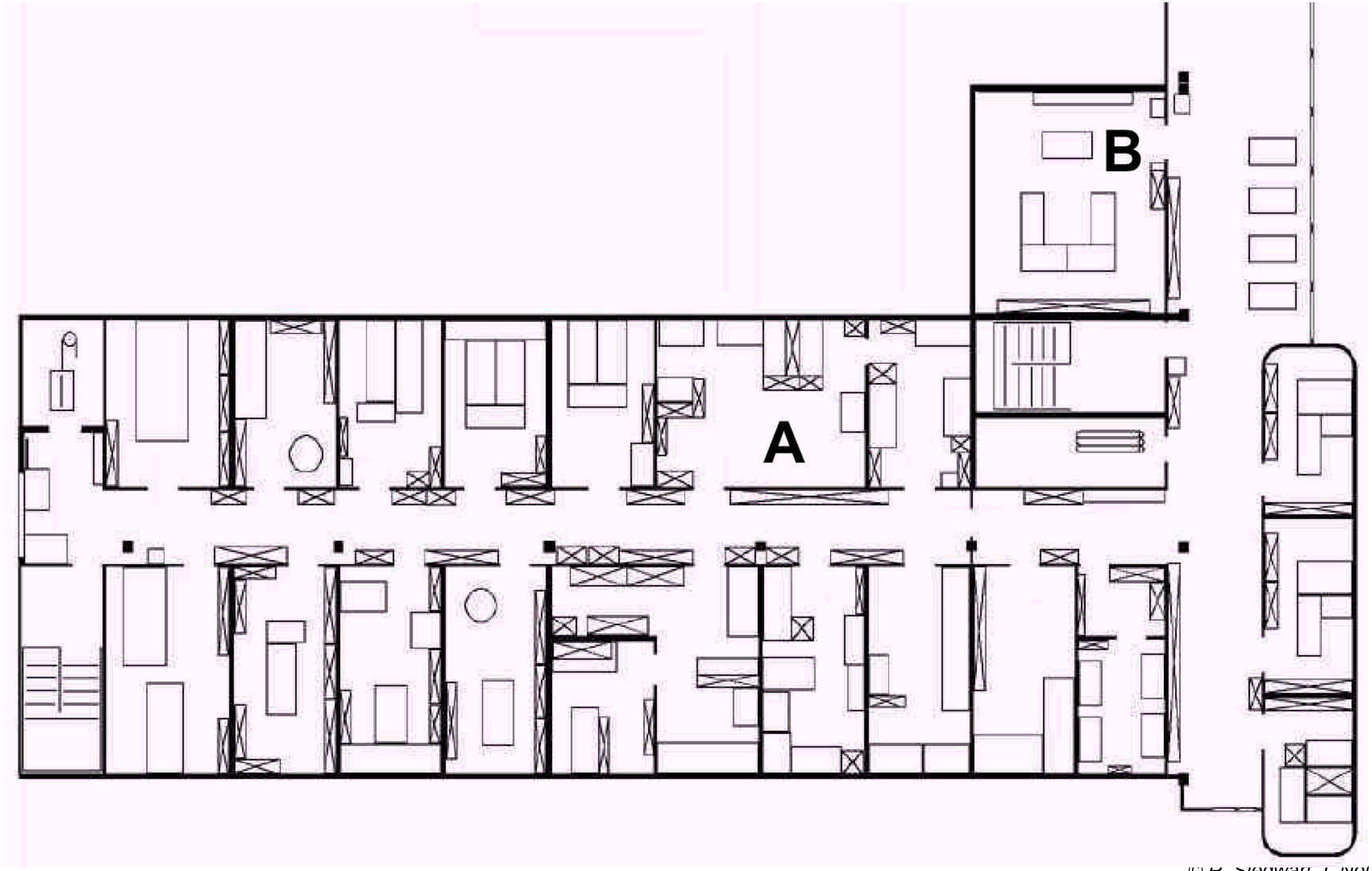
- Pure serial decomposition



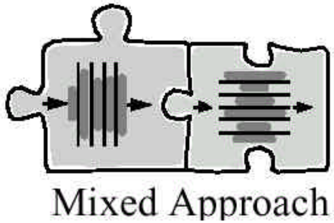
- Pure parallel decomposition

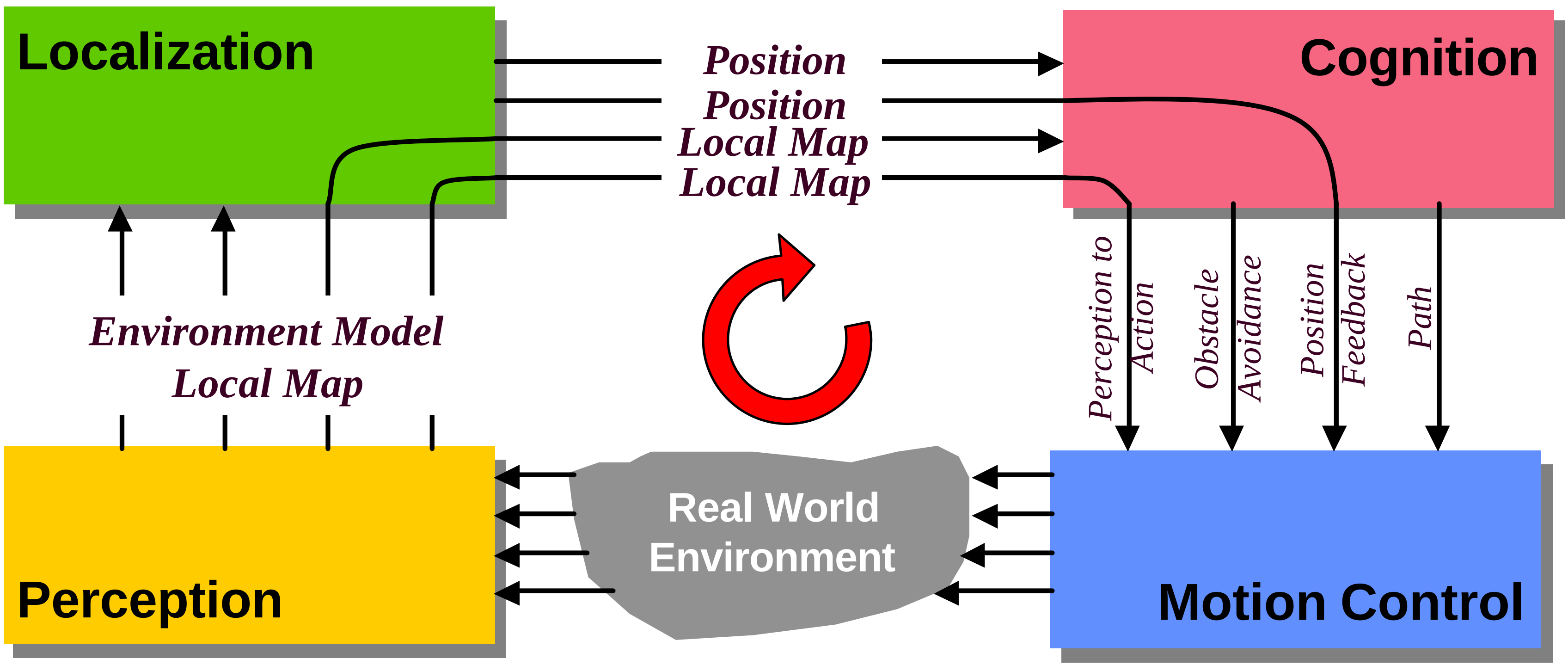


Sample Environment



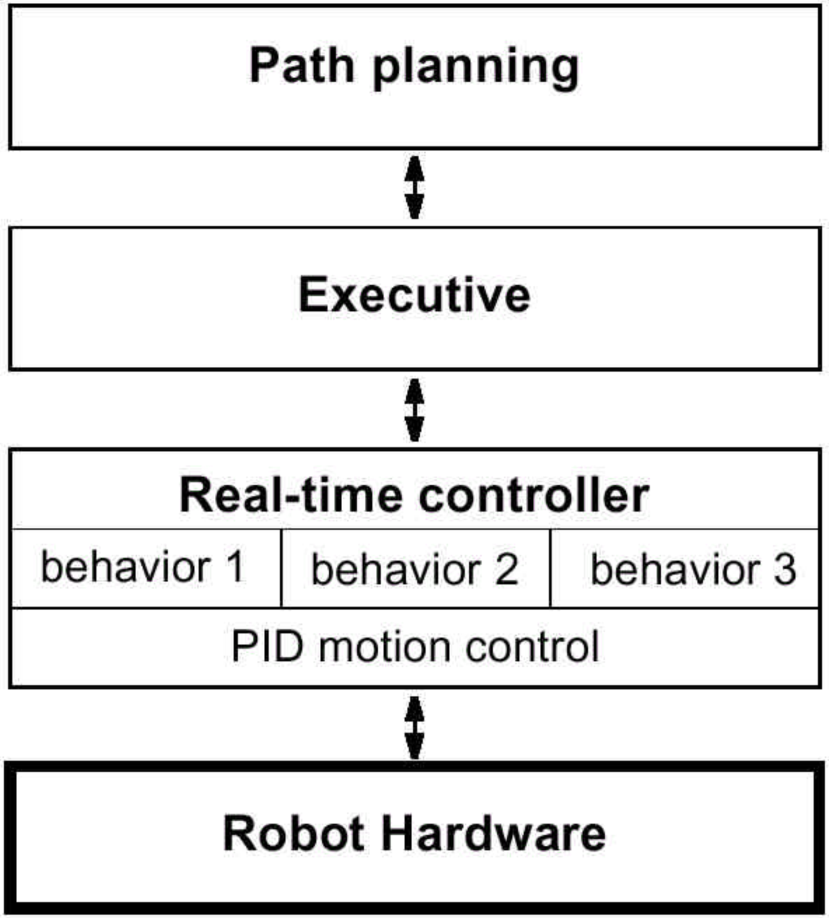
Our basic architectural example



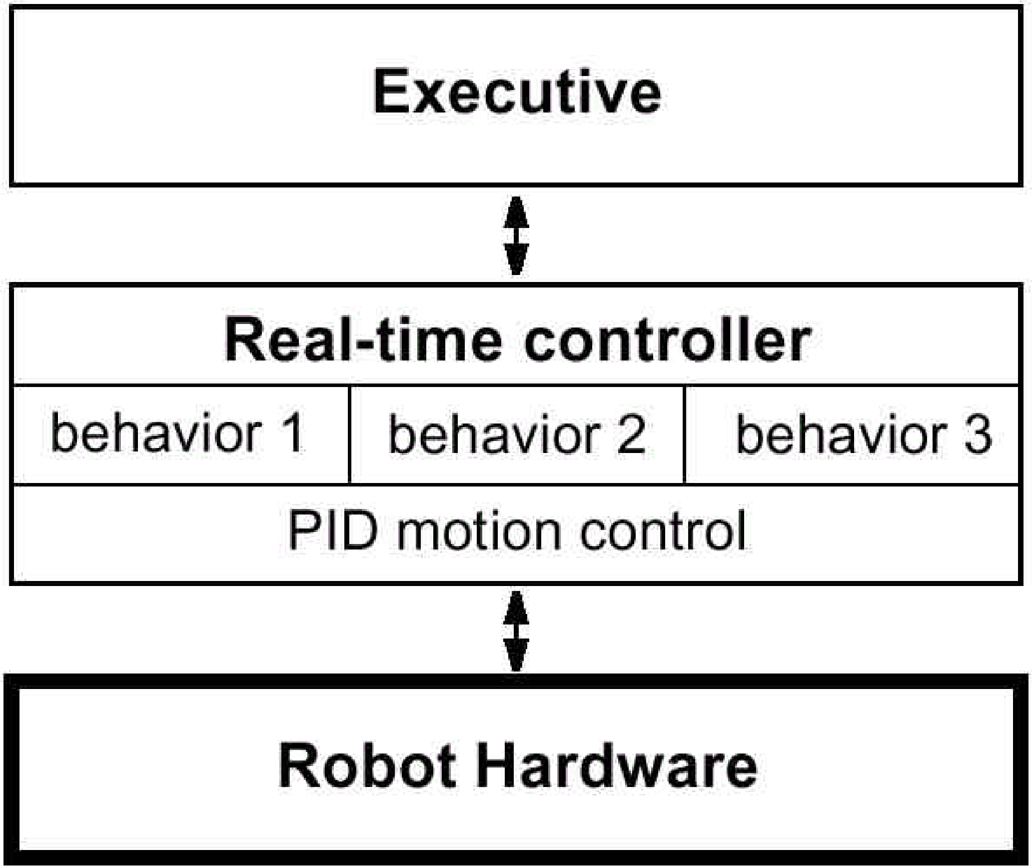


General Tiered Architecture

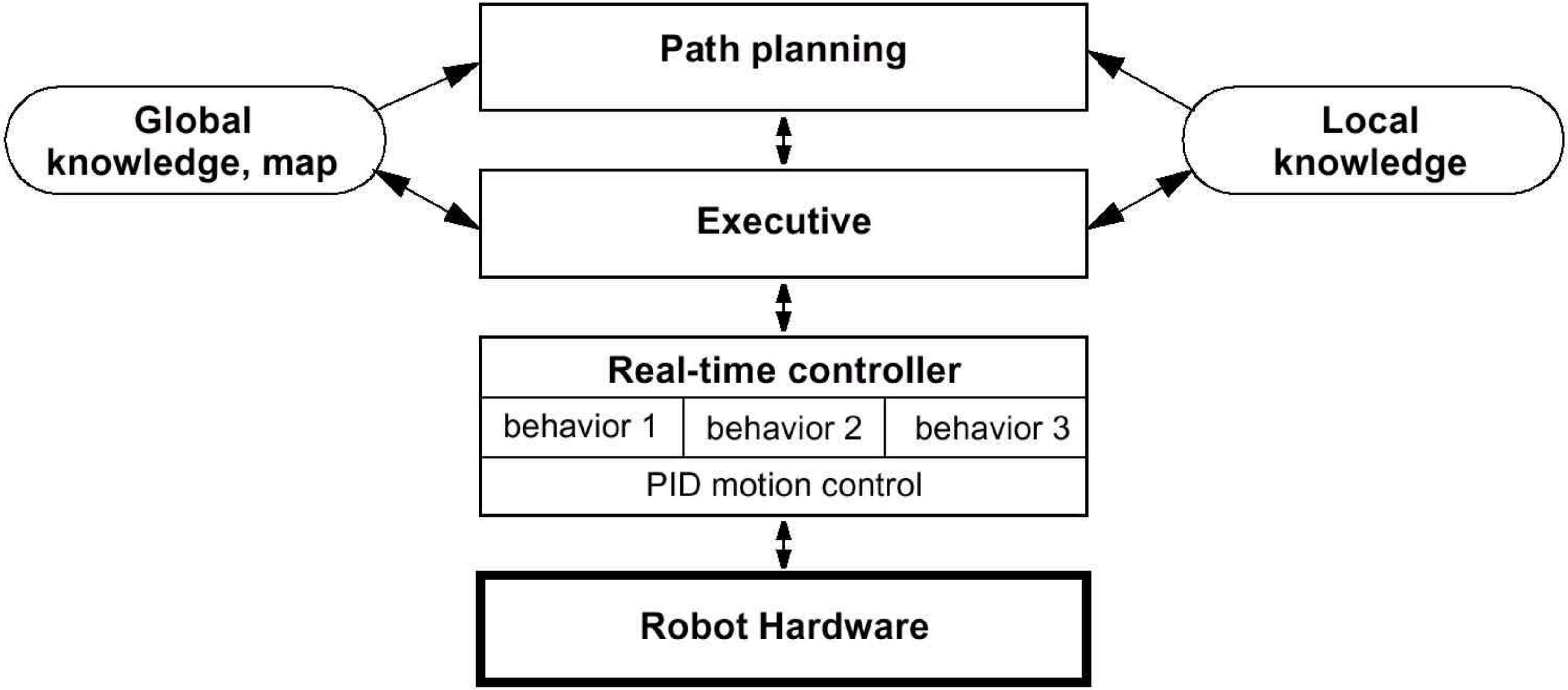
- Executive Layer
  - *activation of behaviors*
  - *failure recognition*
  - *re-initiating the planner*



A Tow-Tiered Architecture for Off-Line Planning

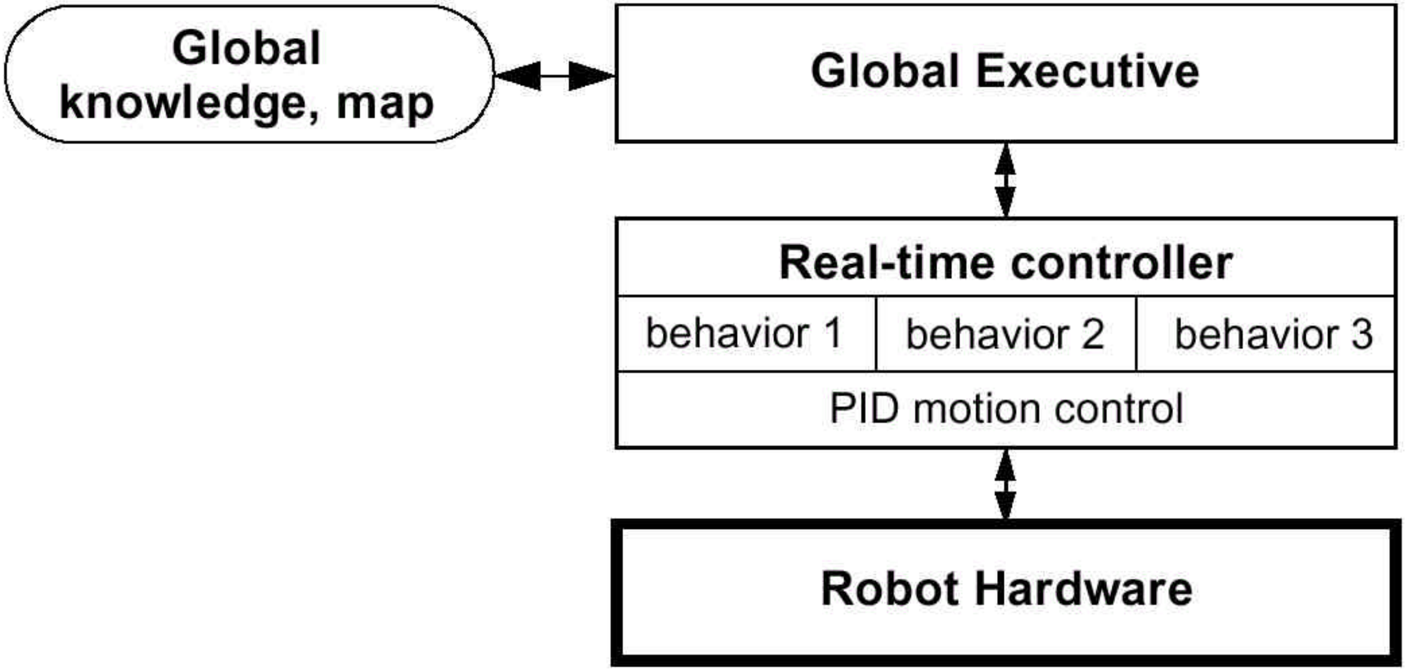


A Three-Tiered Episodic Planning Architecture.



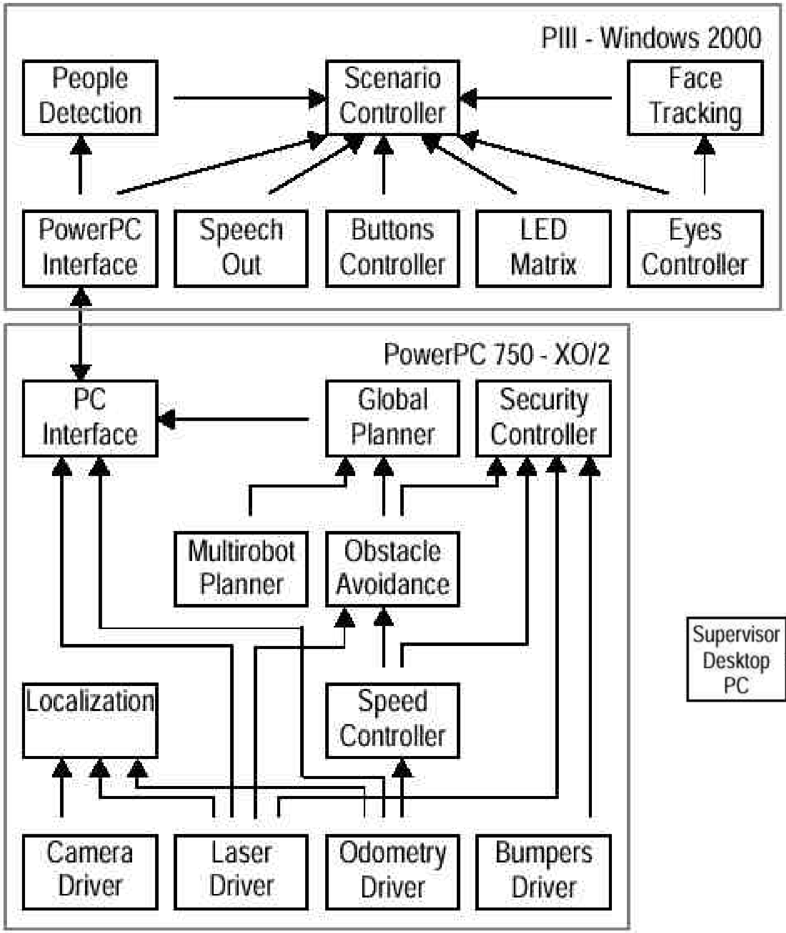
- Planner is triggered when needed: e.g. blockage, failure

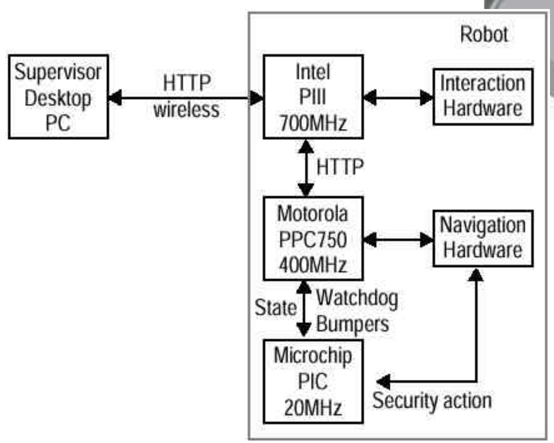
An integrated planning and execution architecture

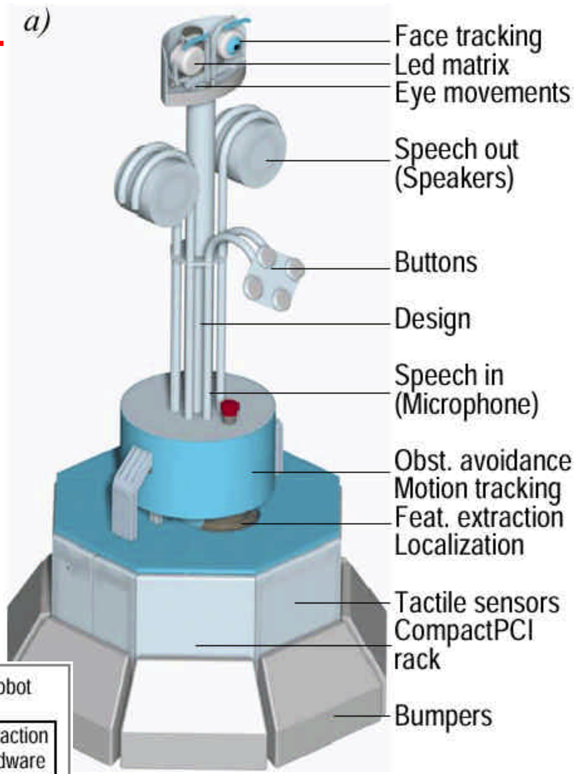


- All integrated, no temporal between planner and executive layer

Example: The RoboX Architecture







Example: RoboX @ EXPO.02