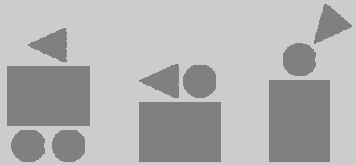
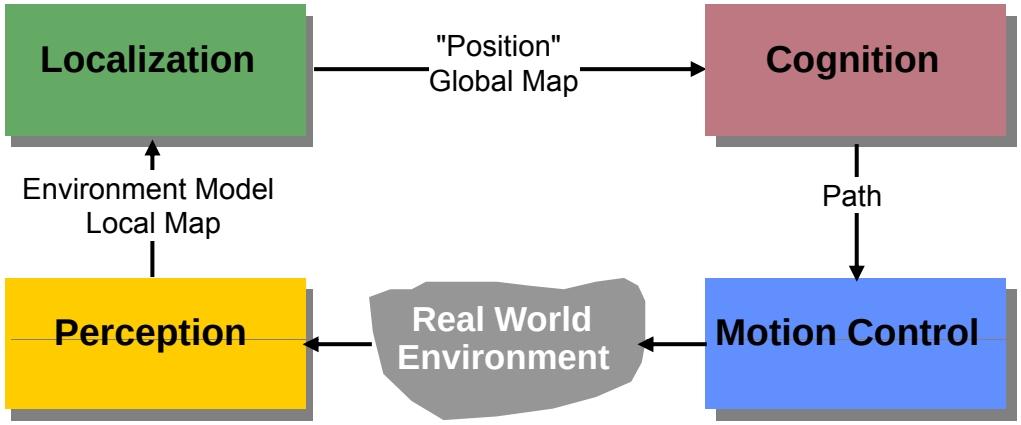
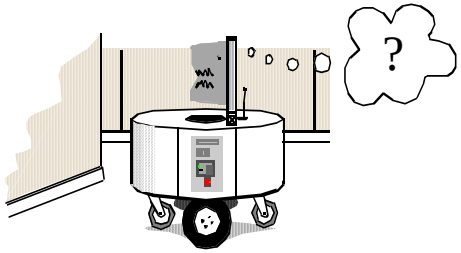




Planning and Navigation



Where am I?

Where am I going?

How do I get there?

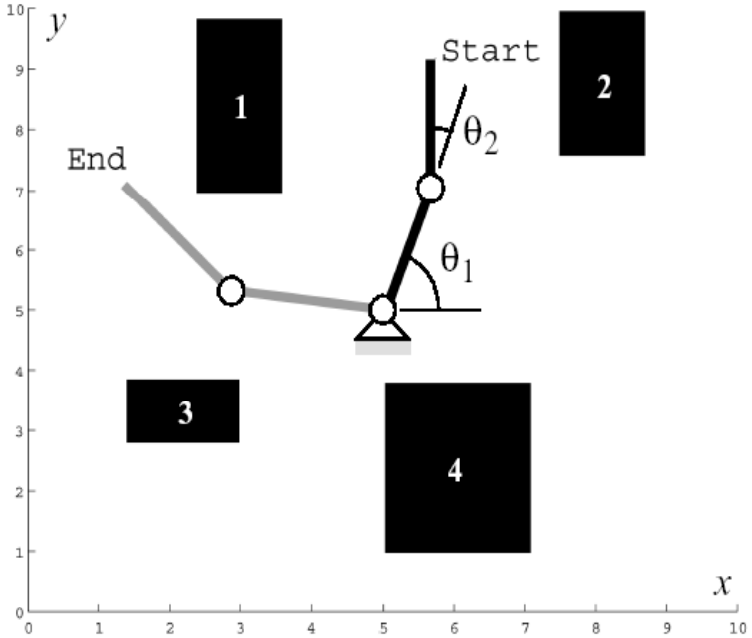
6.2 Path Planning

- The problem: find a path in the physical space from the initial position to the goal position avoiding all collisions with the obstacles

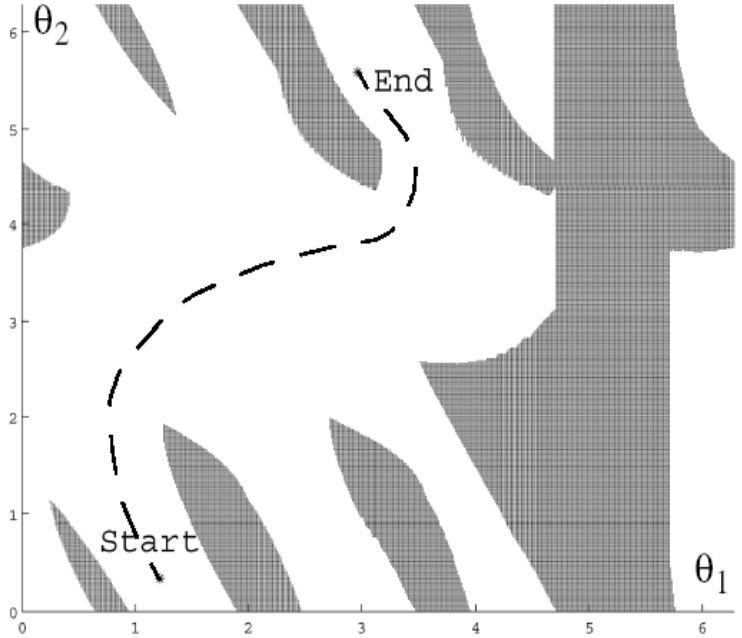
- We can generally distinguish between
  - (*global*) path planning and
  - (*local*) path planning: obstacle avoidance -> safe navigation!

6.3 Path Planning: Configuration Space

- Path planning = find a path to go from Start to End in the configuration space
- State or configuration q can be described with k values q_i



Work Space

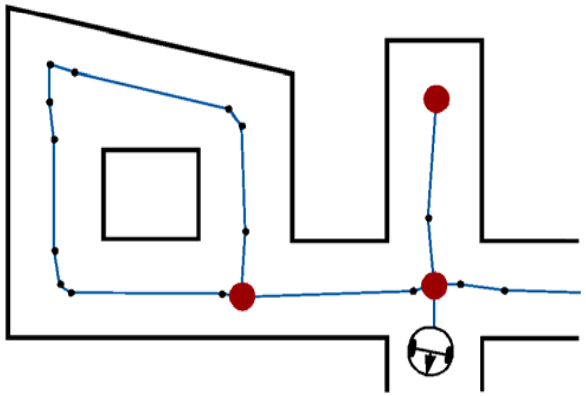


Configuration Space:
the dimension of this space is equal to the Degrees of Freedom (DoF) of the robot

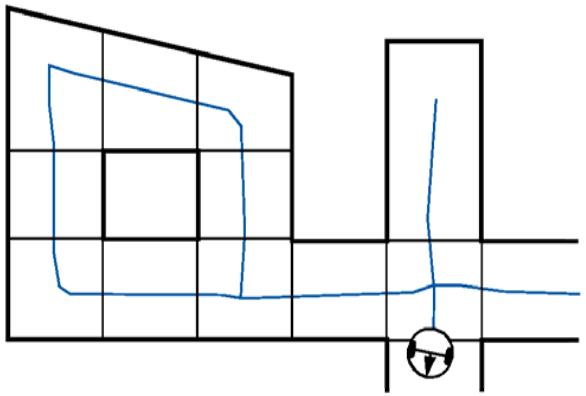
- What is the configuration space of a mobile robot?

6
4 Global Path Planing: outline

- Assumption: there exists a good enough map of the environment for navigation.
  - Topological or metric or a mixture between both.



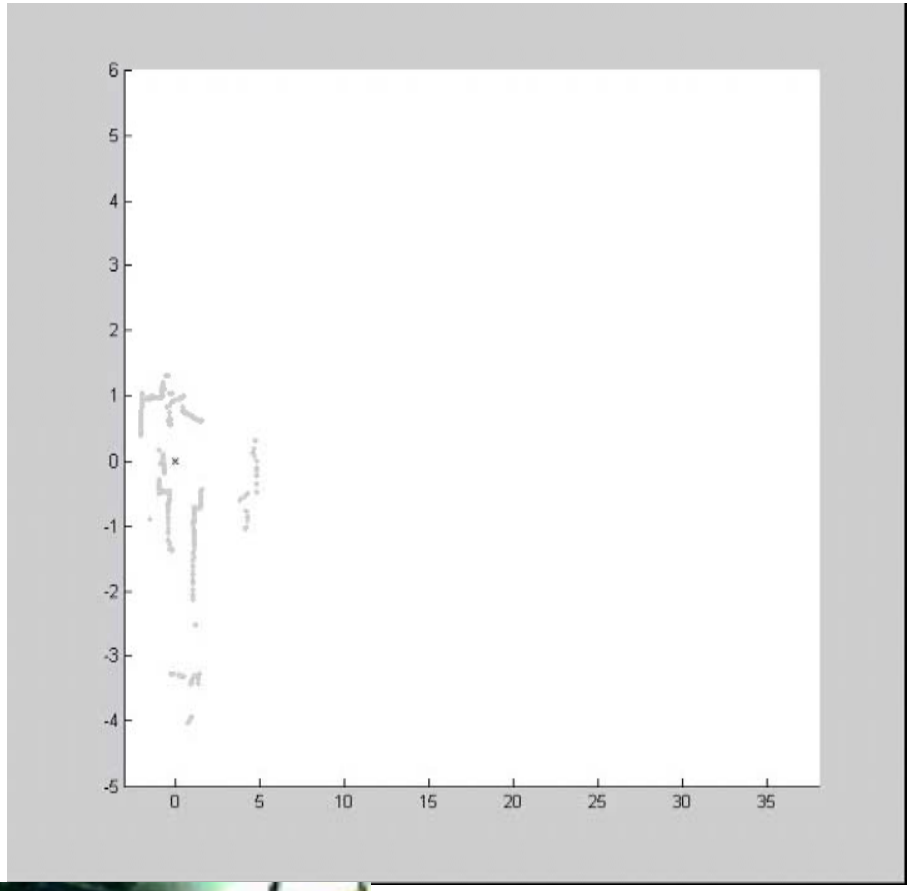
- First step:
  - Representation of the environment by a
    - road-map (graph),
    - cells
    - or a potential field.



- What we will see in this lecture:
  - Visibility Graph
  - Voronoi Diagram
  - Cell Decomposition -> Connectivity Graph
  - Potential Field

6.5 Example of Graph Construction using Fingerprints

- Using Fingerprints



6 Topological Fingerprint Map (Places)

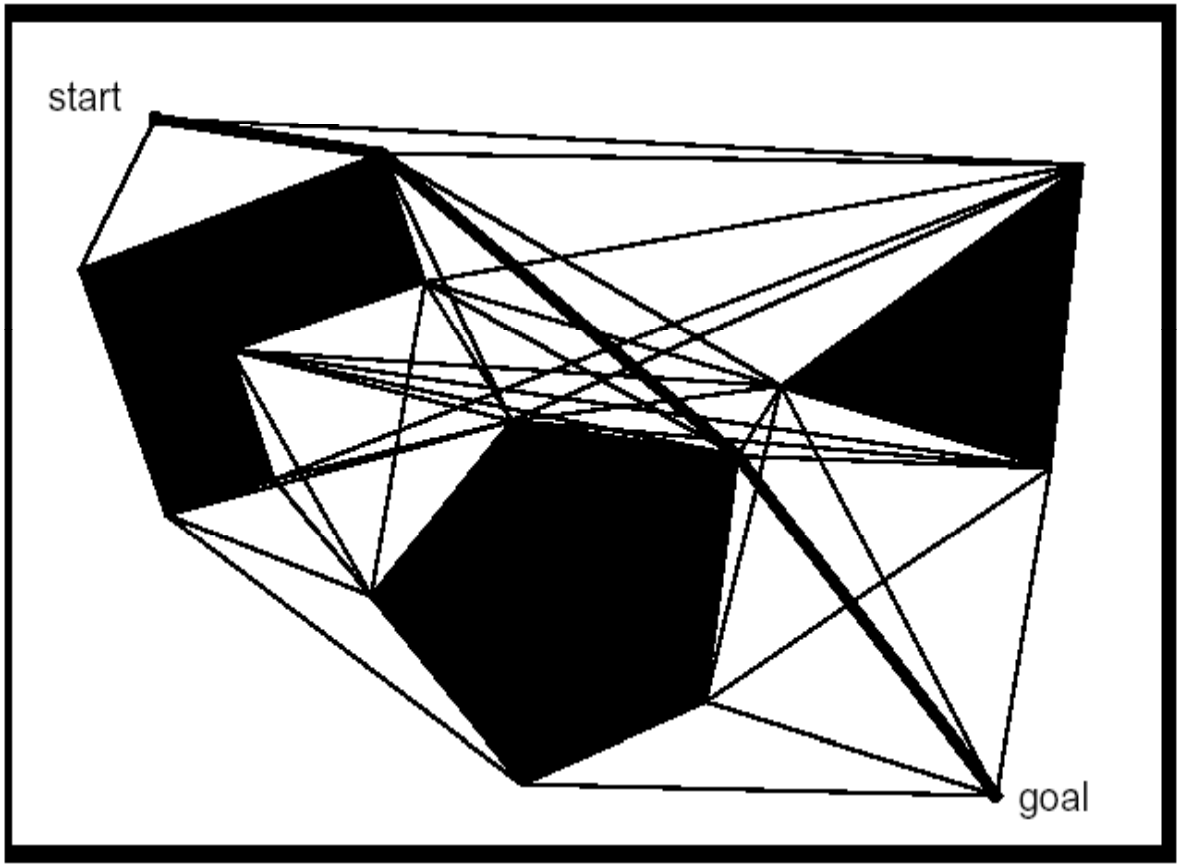
Configuration Space for a Mobile Robot

- Mobile robots operating on a flat ground have 3 DoF: (x, y, θ)

- For simplification, in path planning mobile roboticists often assume that the robot is
  - holonomic
  - and that it 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.

Road-Map Path Planning: Visibility Graph (1/2)



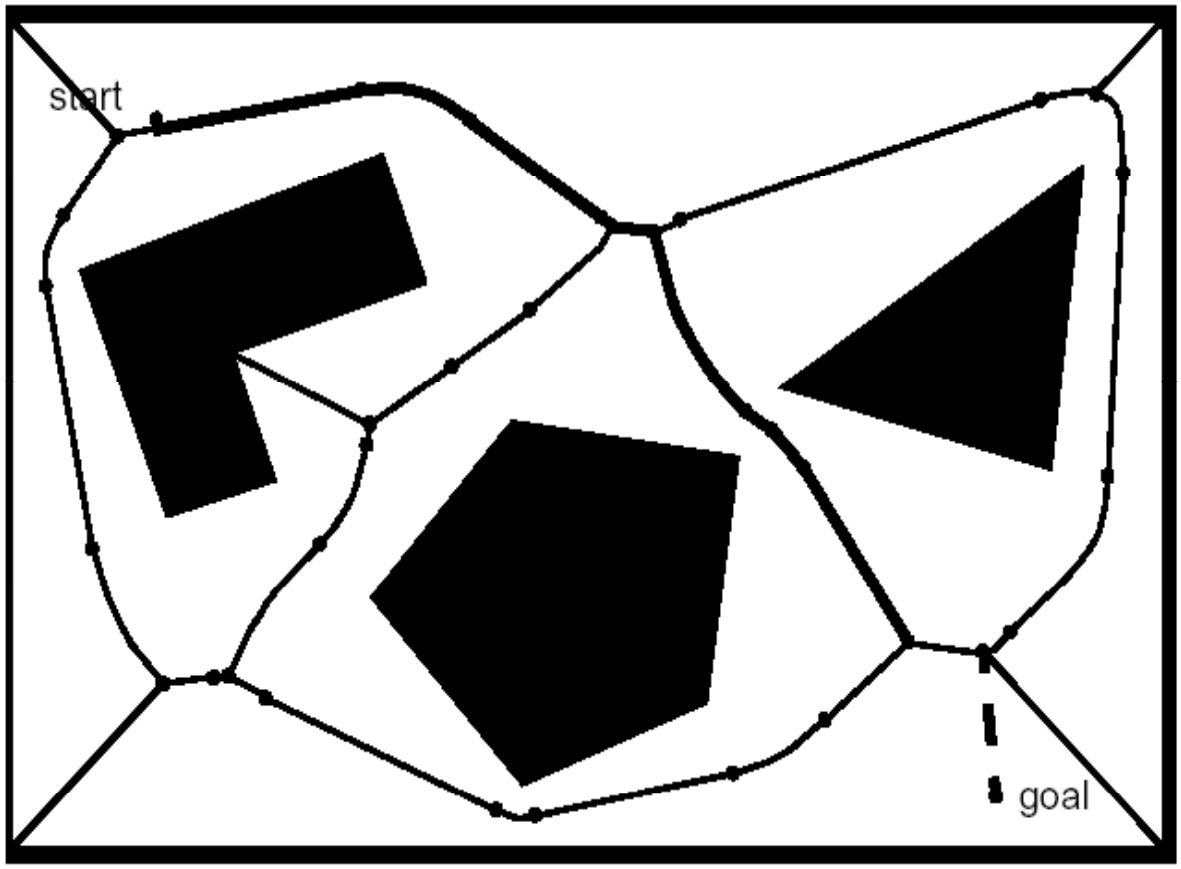
- Particularly suitable for polygon-like obstacles
- Shortest path length
- Grow obstacles to avoid collisions

Road-Map Path Planning: Visibility Graph (2/2)

- Pros
  - The found path is optimal because it is the shortest length path
  - Implementation simple when obstacles are polygons

- Cons
  - The solution path found by the visibility graph tend to take the robot as close as possible to the obstacles: the common solution is to grow obstacles by more than robot's radius
  - Number of edges and nodes increases with the number of polygons
  - Thus it can be inefficient in densely populated environments

Road-Map Path Planning: Voronoi Diagram (1/2)



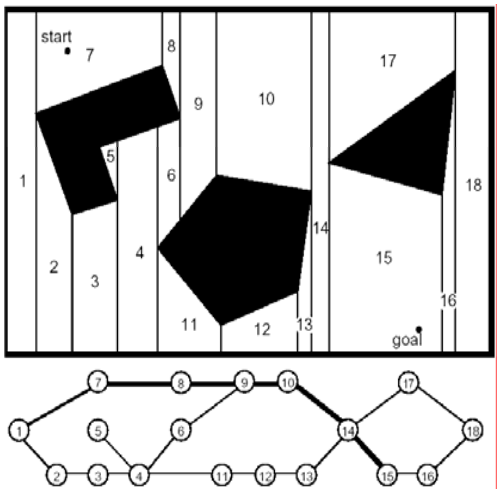
- In contrast to the Visibility Graph approach tends to maximize the distance between robot and obstacles
- Easy executable: Maximize the sensor readings
- Works also for map-building: Move on the Voronoi edges

Road-Map Path Planning: Voronoi Diagram (2/2)

- 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

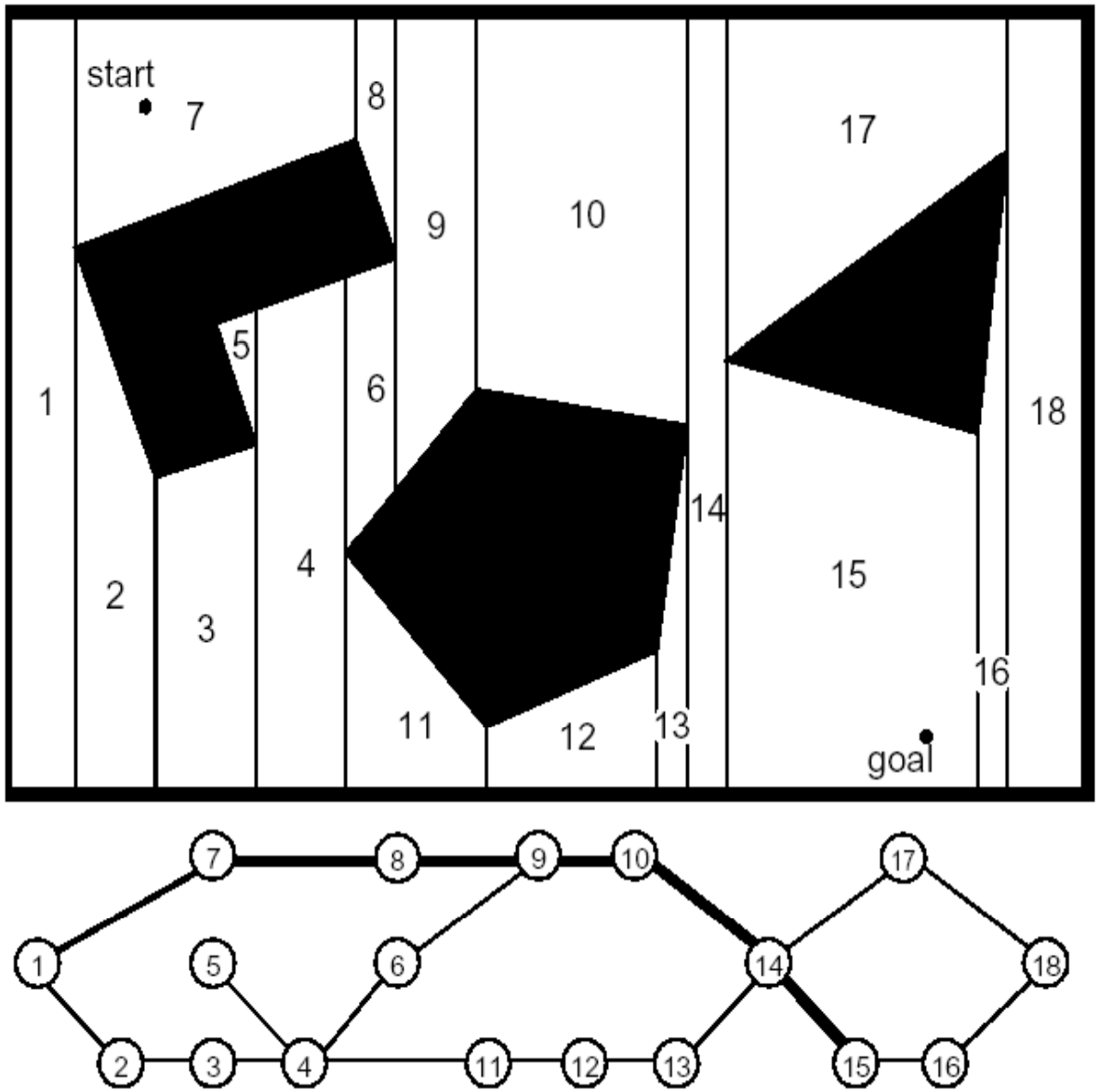
Road-Map Path Planning: Cell Decomposition (1/4)

- 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

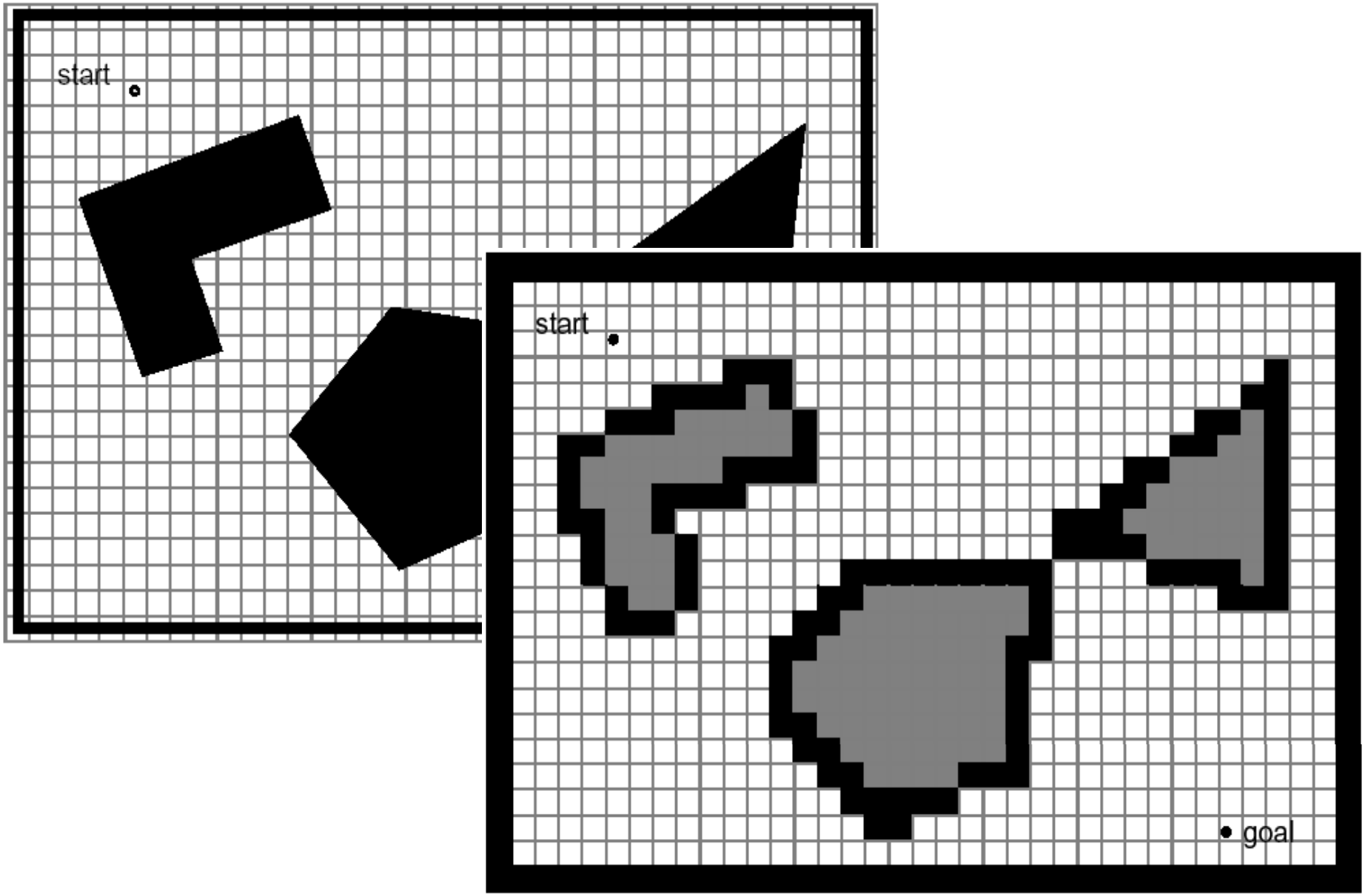


Example of exact cell decomposition

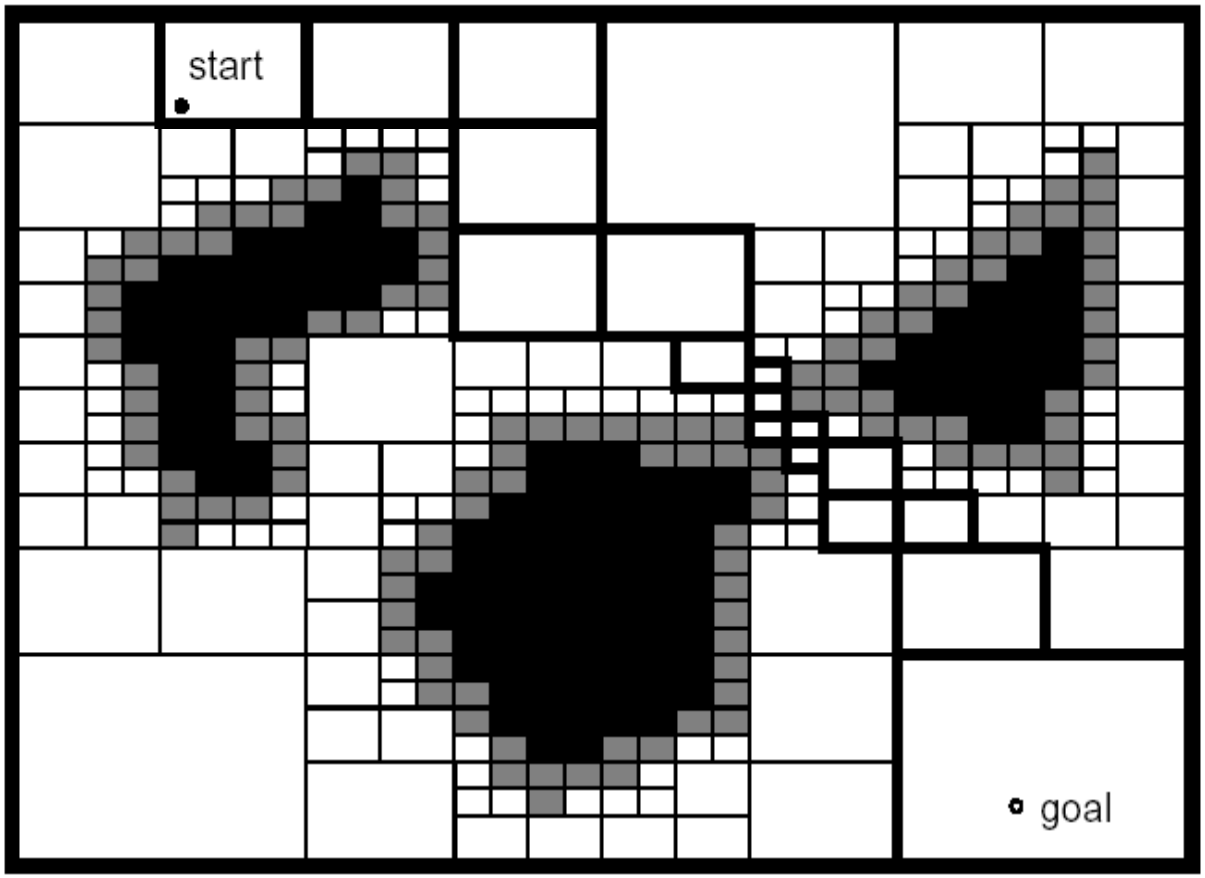
Road-Map Path Planning: Exact Cell Decomposition (2/4)



Road-Map Path Planning: Fixed Cell Decomposition (3/4)



Road-Map Path Planning: Adaptive Cell Decomposition (4/4)



Road-Map Path Planning: Path / Graph Search Strategies

- Wavefront Expansion NF1 (see also later)
- Breadth-First Search
- Depth-First Search
- Greedy search and A*

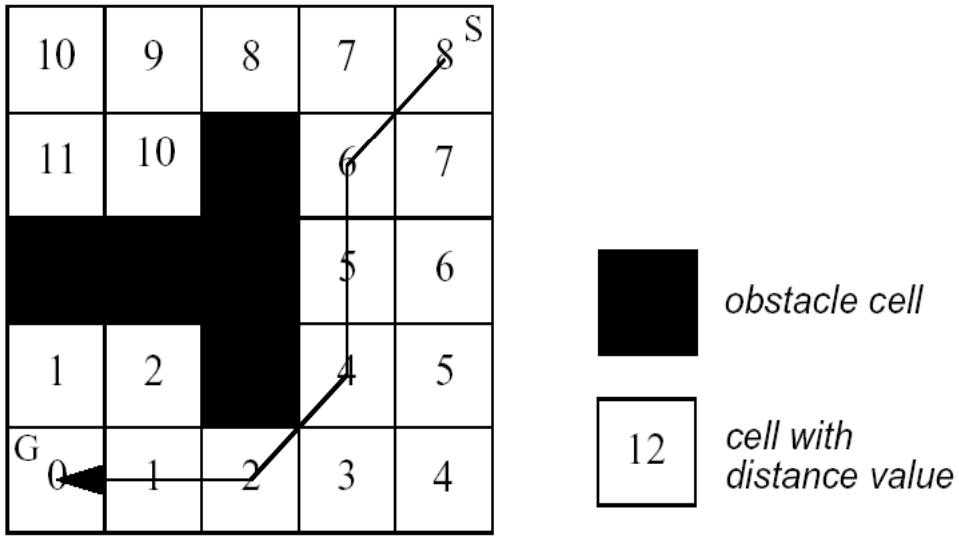
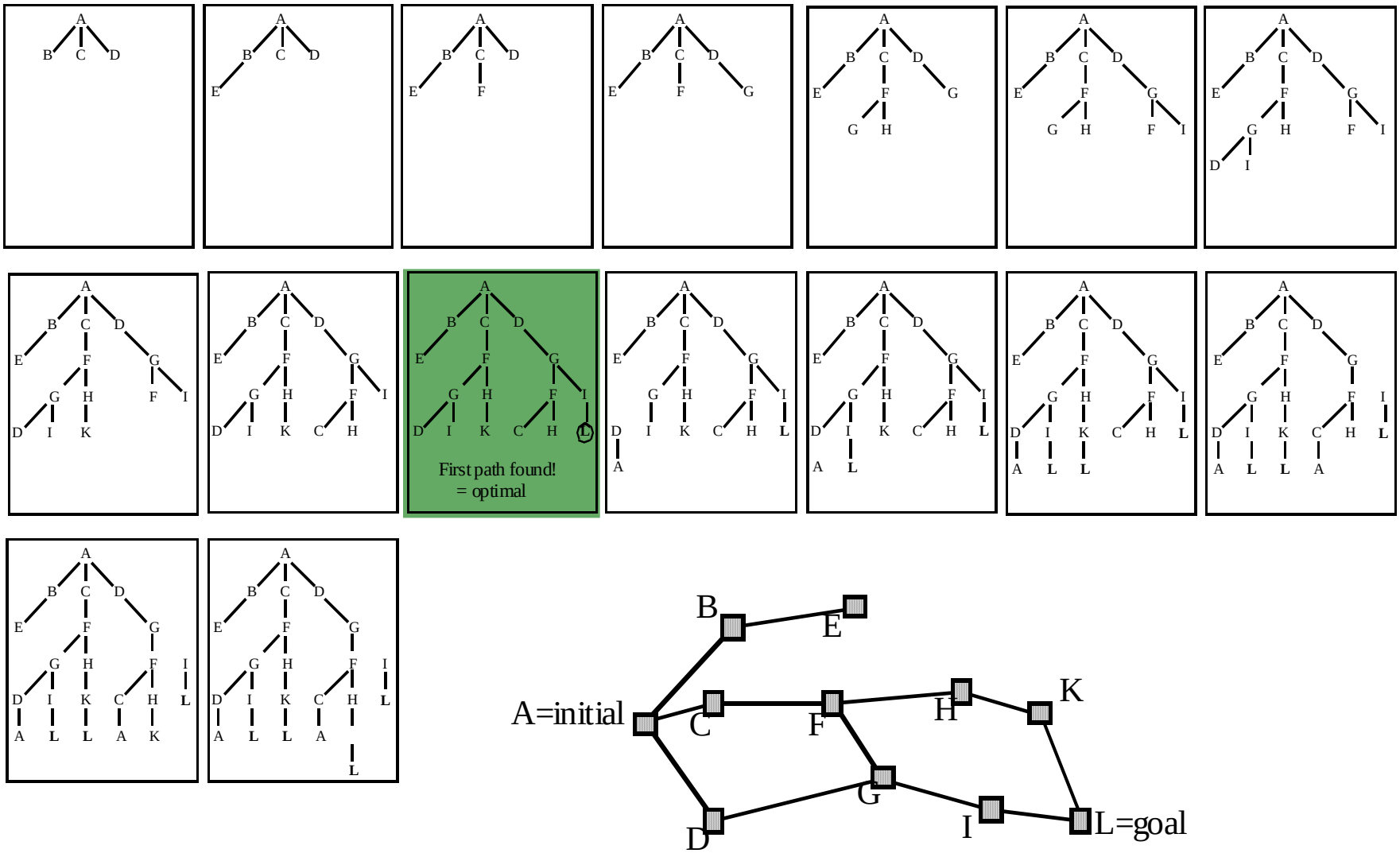


Fig. 1: NF1: put in each cell its L-distance from the goal position (used also in local path planning)

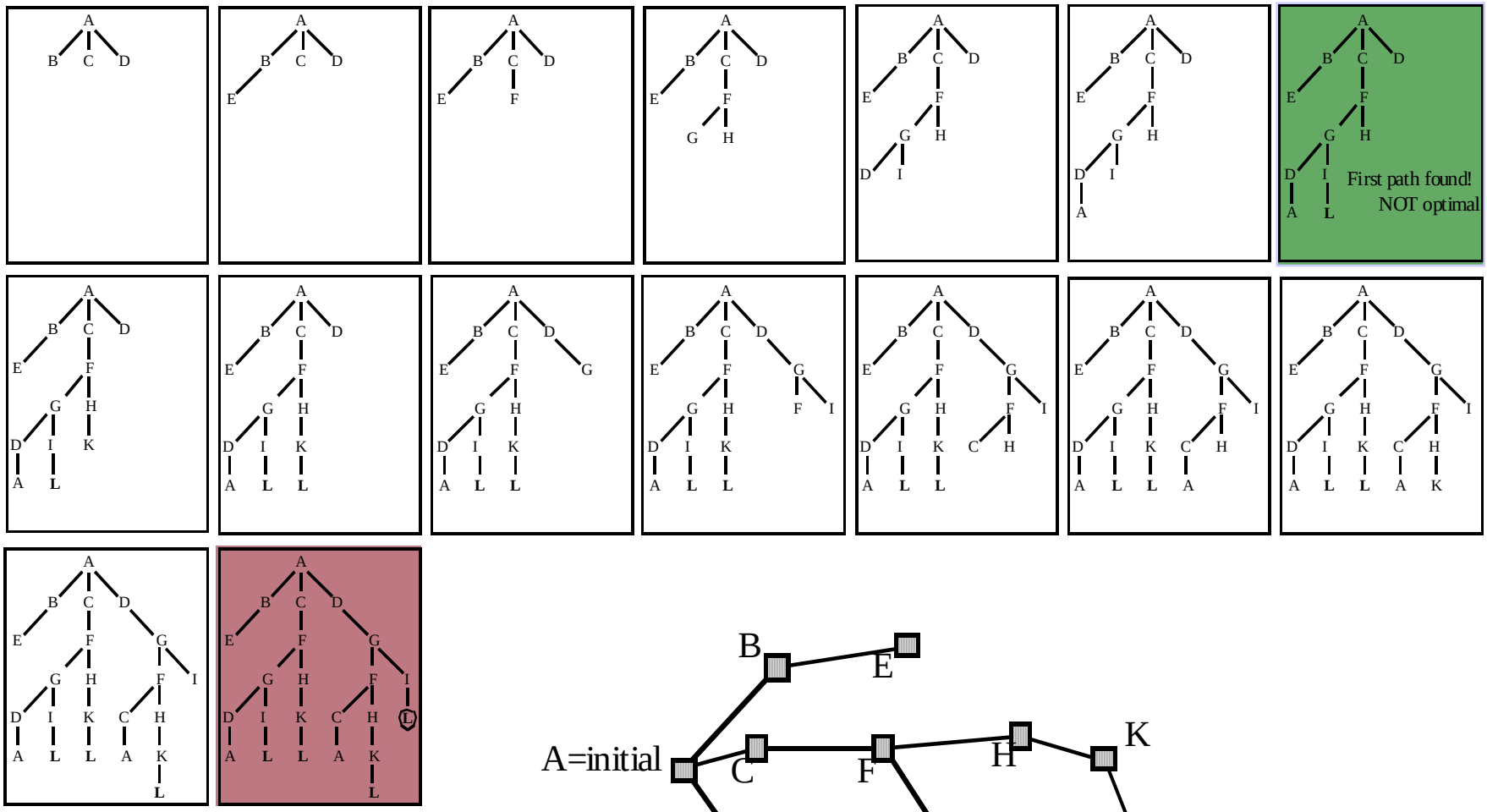
6
17

Breadth-First Search

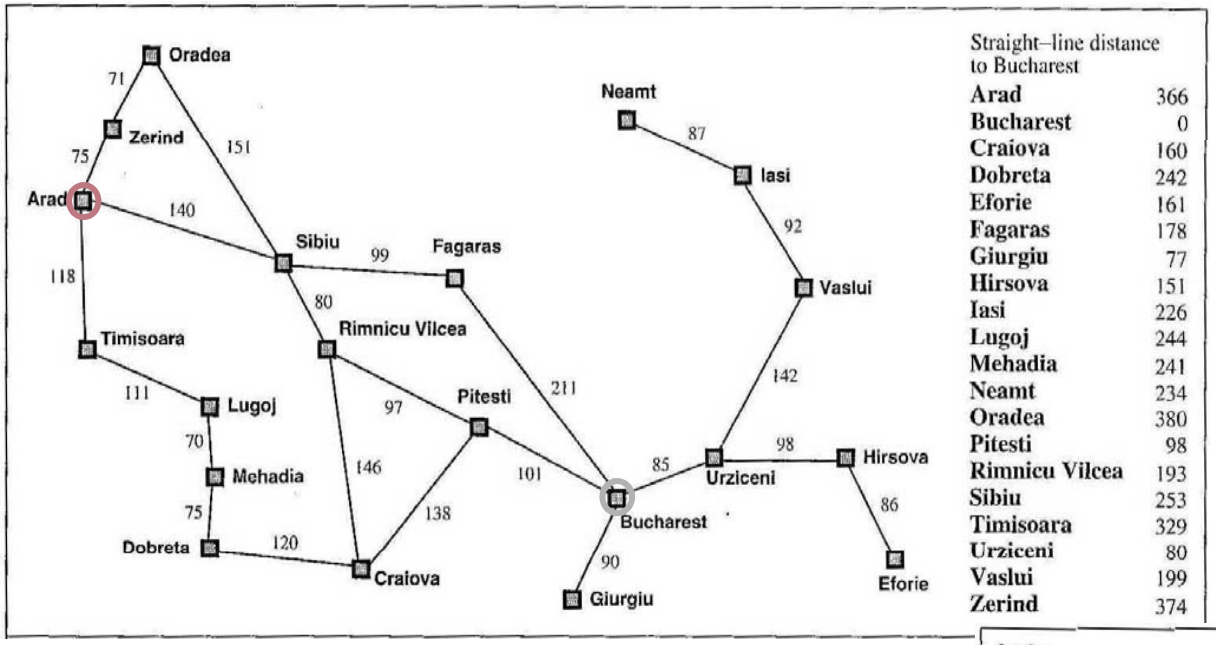


6
18

Depth-First Search

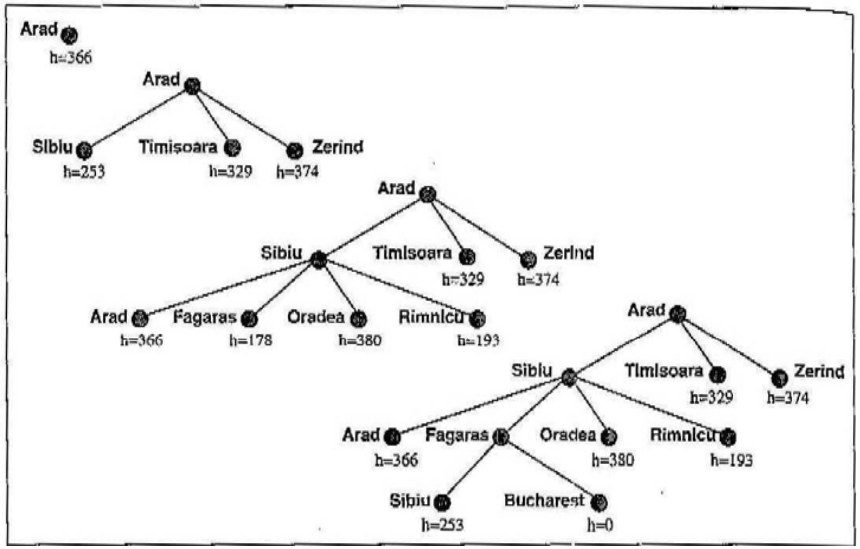


Greedy Search: A*



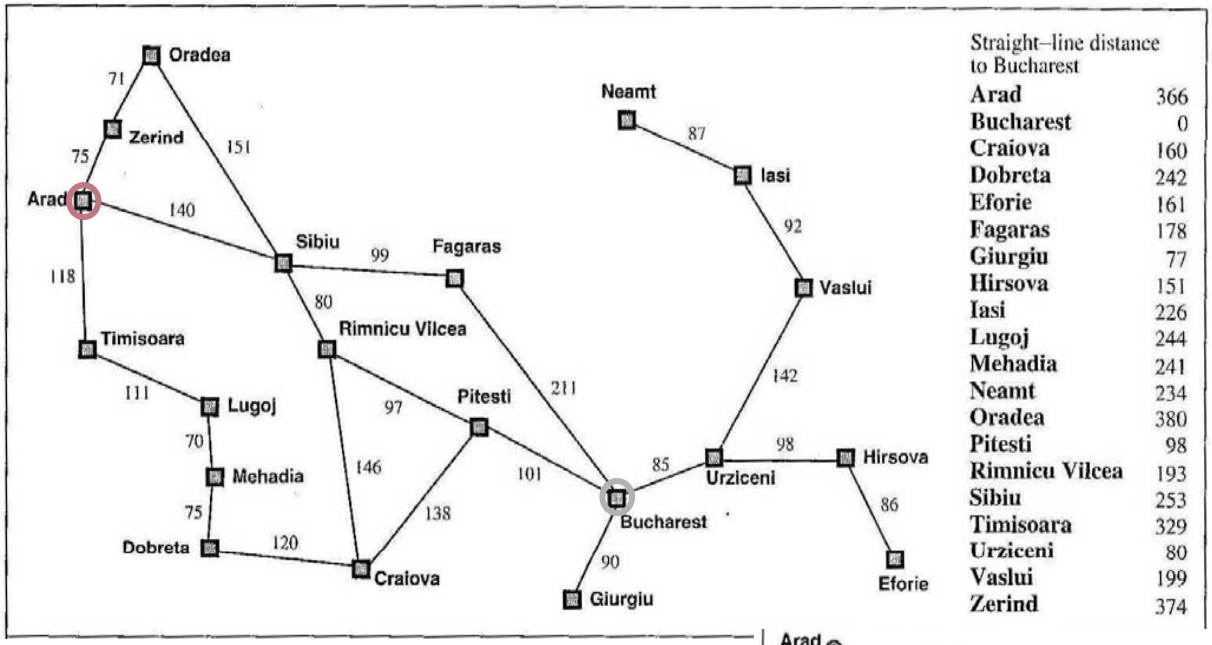
Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

- cost estimate of the cheapest path from state at node n to the goal



6
20

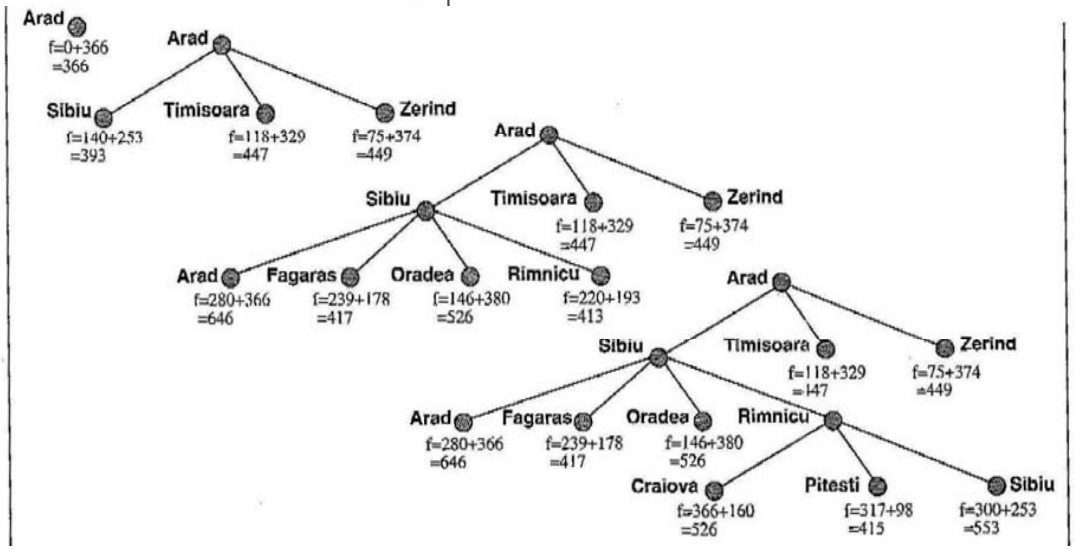
Greedy Search: A*



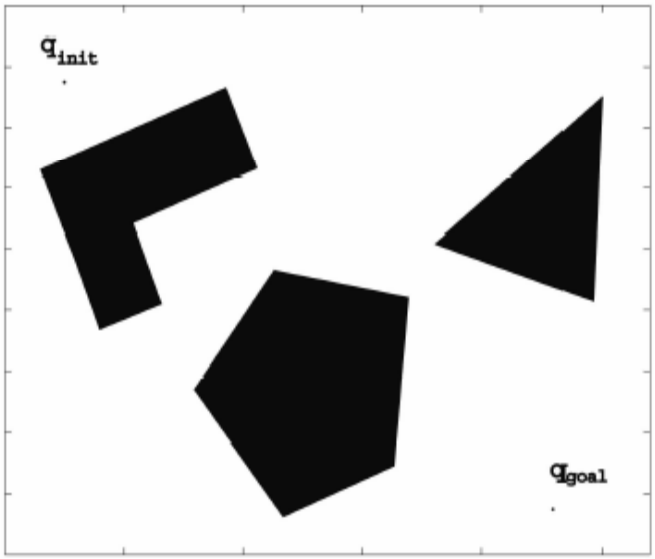
Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

$f(n) = g(n) + h(n)$

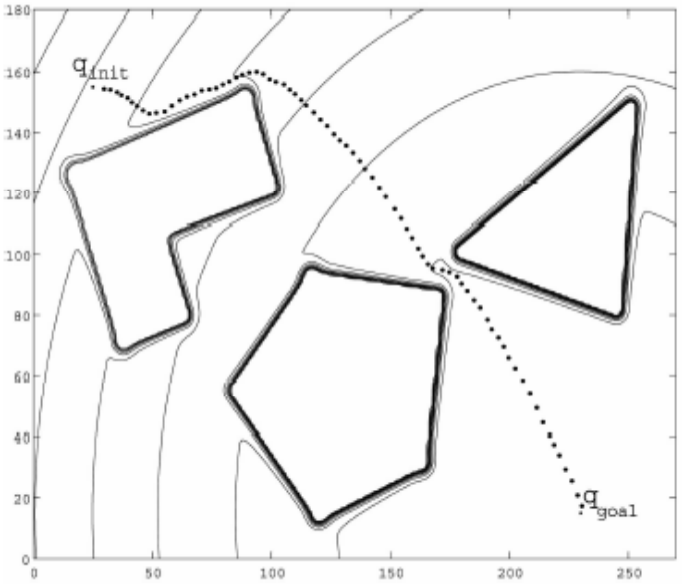
- heuristic cost function f(n) is equal to the sum of the path cost g(n) to get from the start to node n and the straight-line distance h(n) from node n to the goal.

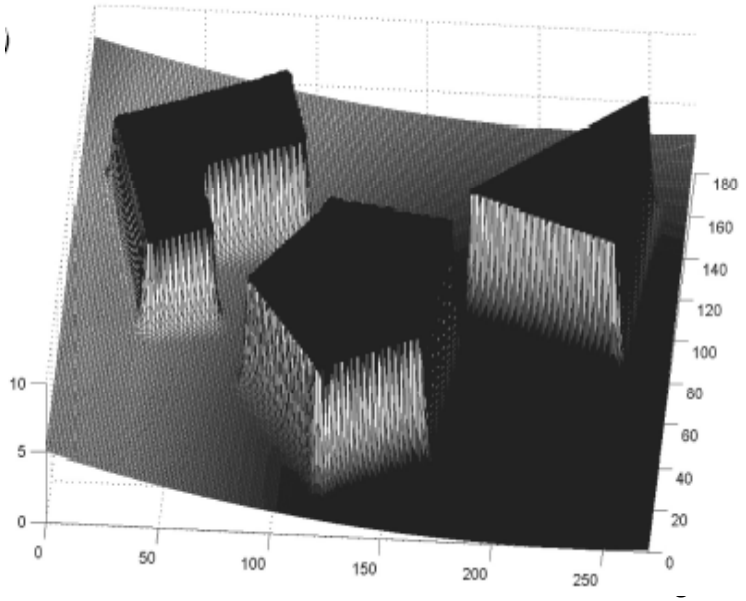


Potential Field Path Planning



- 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
  - robot is assumed to be a point mass

Potential Field Path Planning: Attractive Potential Field

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

$$\begin{aligned} U_{att}(q) &= \frac{1}{2} k_{att} \cdot \rho_{goal}^2(q) \\ &= \frac{1}{2} k_{att} \cdot (q - q_{goal})^2 \end{aligned}$$

- Attracting force converges linearly towards 0 (goal)

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

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

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

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

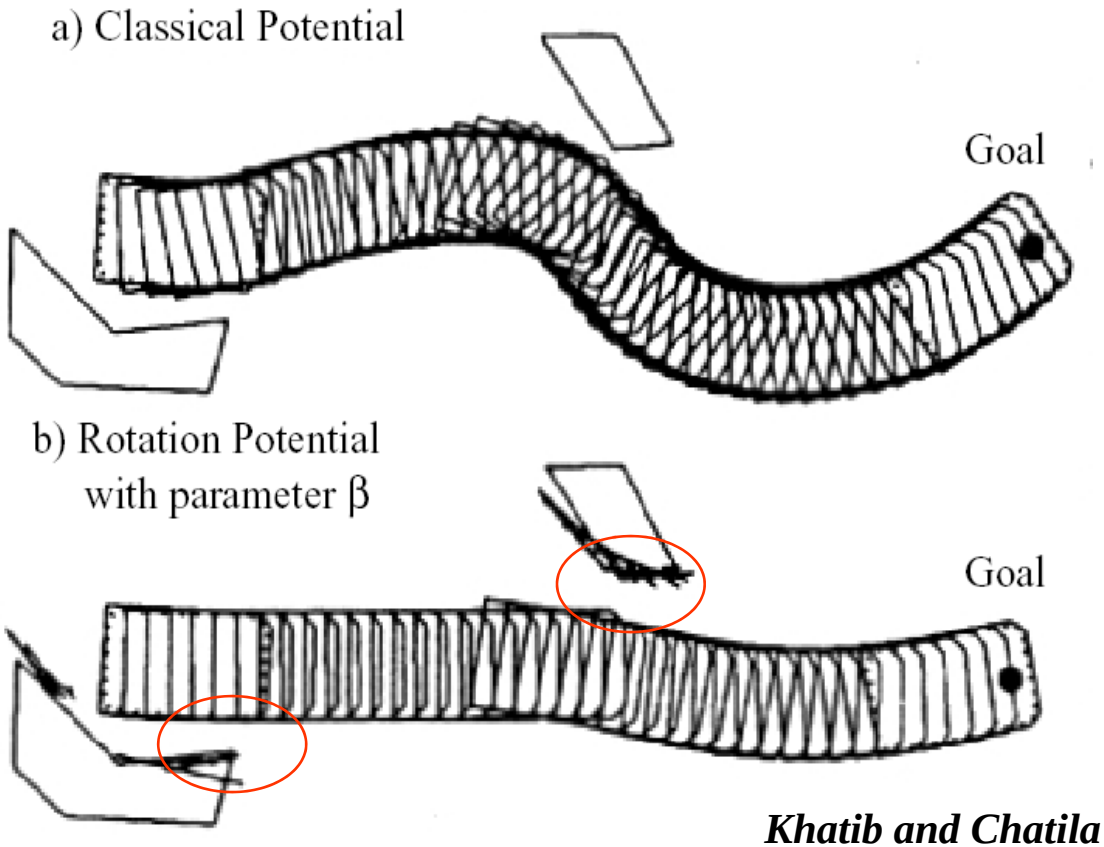
$$F_{rep}(q) = -\nabla U_{rep}(q) = \begin{cases} k_{rep} \left(\frac{1}{\rho(q)} - \frac{1}{\rho_0} \right) \frac{1}{\rho^2(q)} \frac{q - q_{obst.}}{\rho(q)} & \text{if } \rho(q) \leq \rho_0 \\ 0 & \text{if } \rho(q) \geq \rho_0 \end{cases}$$

Potential Field Path Planning:

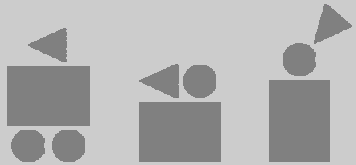
- PROS:
  - Very efficient and fast to compute
- CONS
  - Local minima problem exists
  - problem is getting more complex if the robot is not considered as a point mass
  - If objects are convex there exists situations where several minimal distances exist $\rightarrow$ can result in oscillations

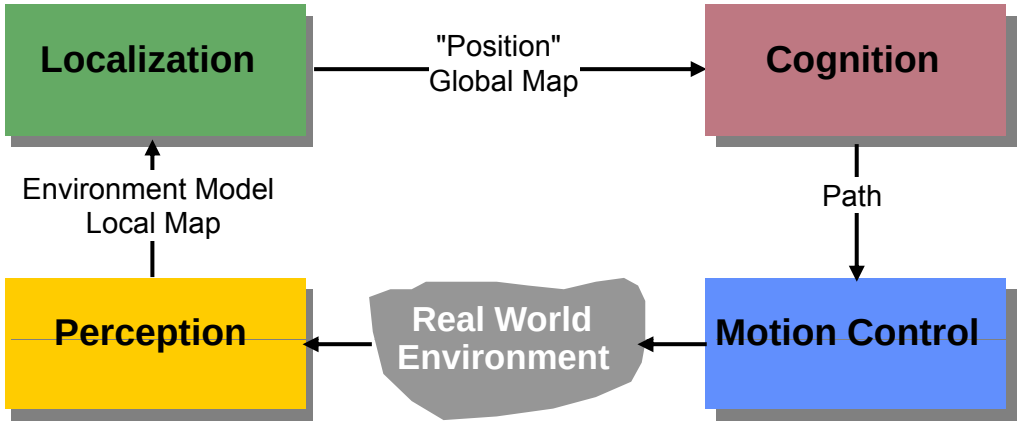
Potential Field Path Planning: Extended Potential Field Method

- Additionally a *rotation potential field* and a *task potential field* in introduced
- Rotation potential field
  - force is also a function of robots orientation relative to the obstacles. This is done using a gain factor that reduces the repulsive force when obstacles are parallel to robot's direction of travel
- Task potential field
  - Filters out the obstacles that should not influence the robots movements, i.e. only the obstacles in the sector in front of the robot are considered

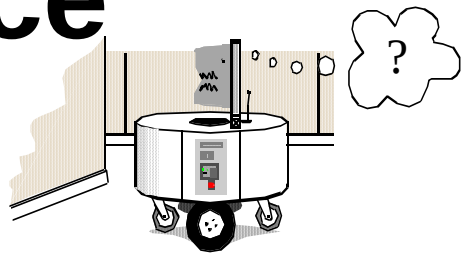


Khatib and Chatila





Planning and Navigation II: Obstacle Avoidance



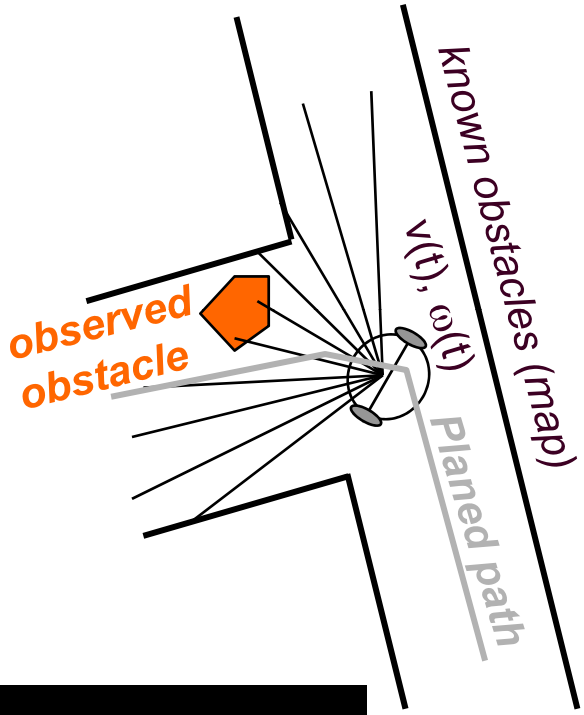
Where am I?

Where am I going?

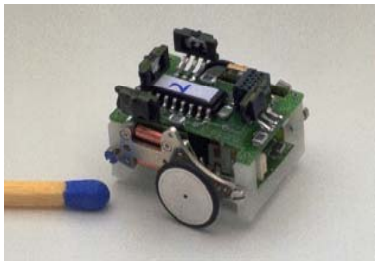
How do I get there?

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

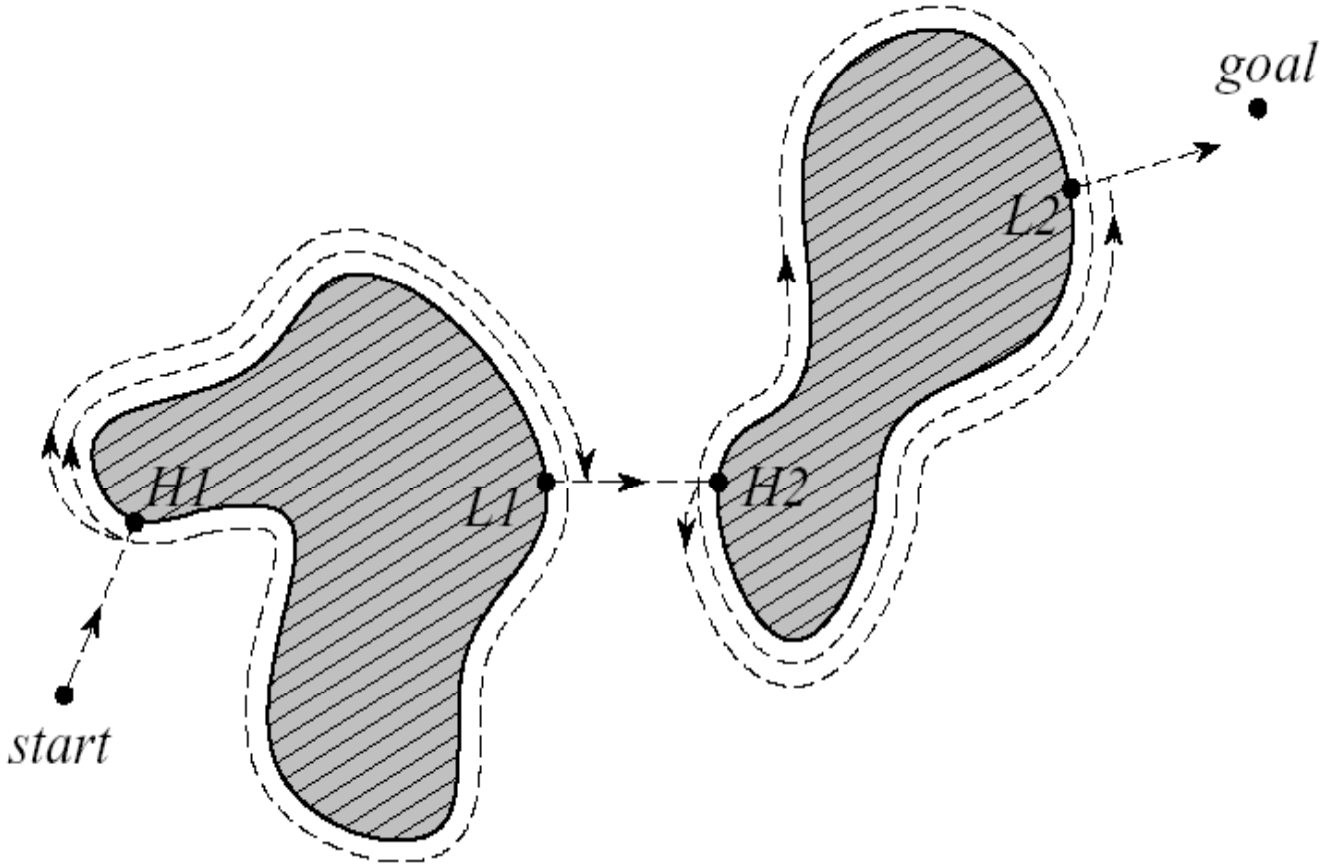


- Example: Alice



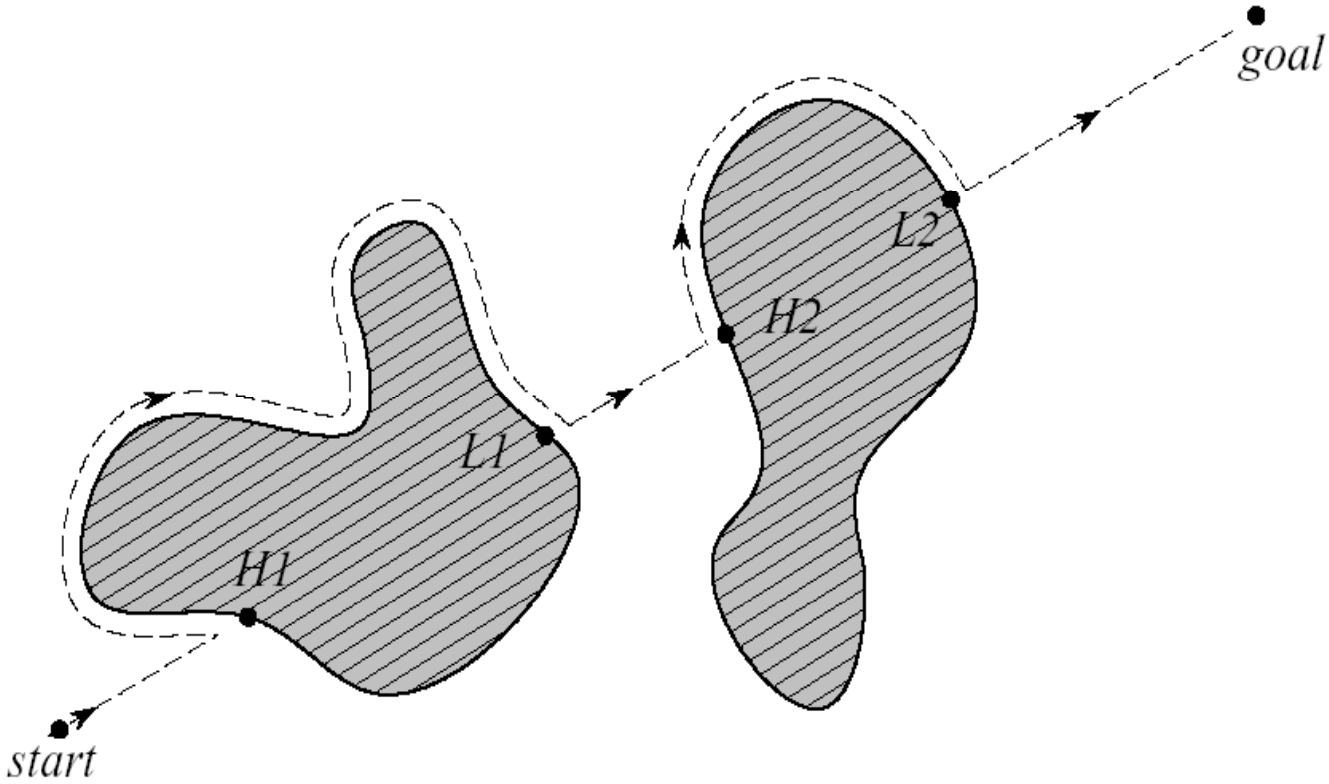
Obstacle Avoidance: Bug1

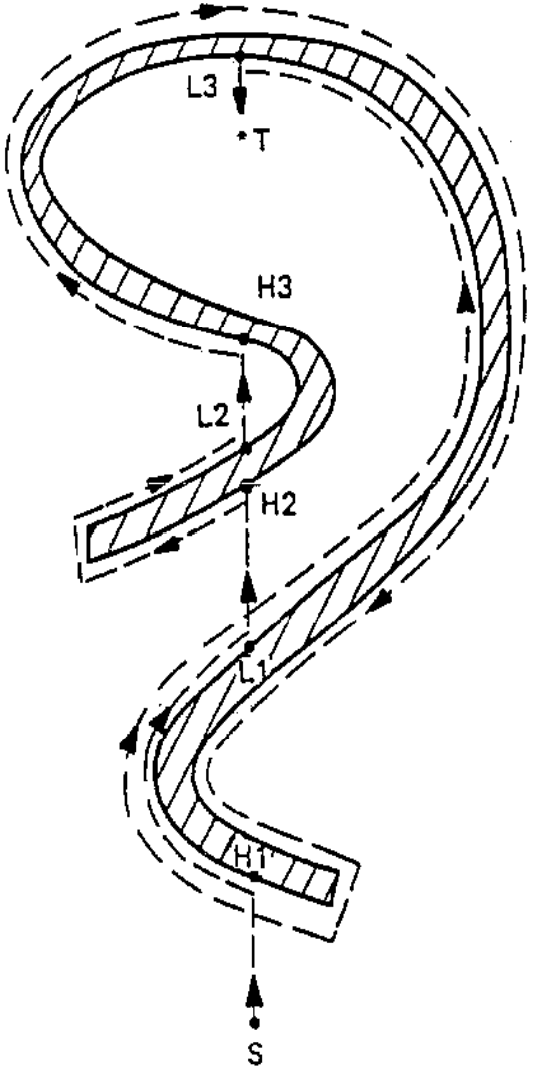
- Following along the obstacle to avoid it
- Each encountered obstacle is once fully circled before it is left at the point closest to the goal



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





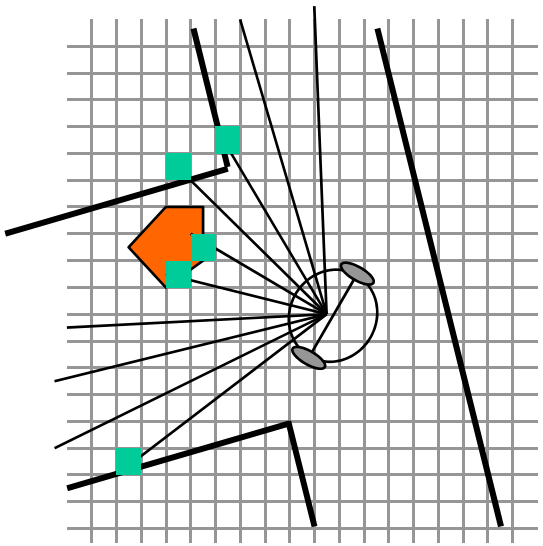
Obstacle Avoidance: Vector Field Histogram (VFH)

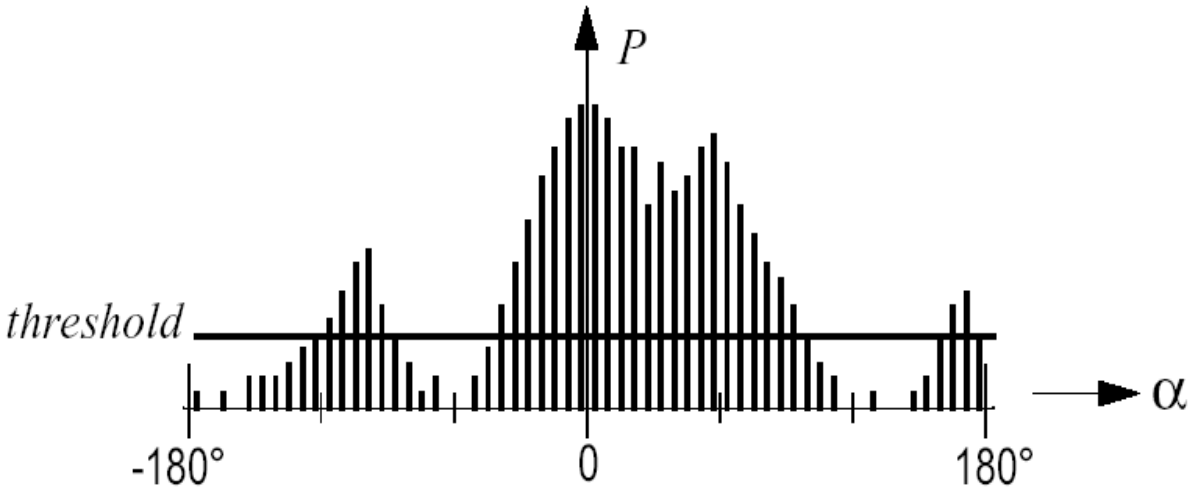
Borenstein et al.

- 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
  - The steering direction is computed in two steps:
    - all openings for the robot to pass are found
    - the one with lowest *cost function G* is selected

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

target_direction = alignment of the robot path with the goal
wheel_orientation = difference between the new direction and the currrent wheel orientation
previous_direction = difference between the previously selected direction and the new direction

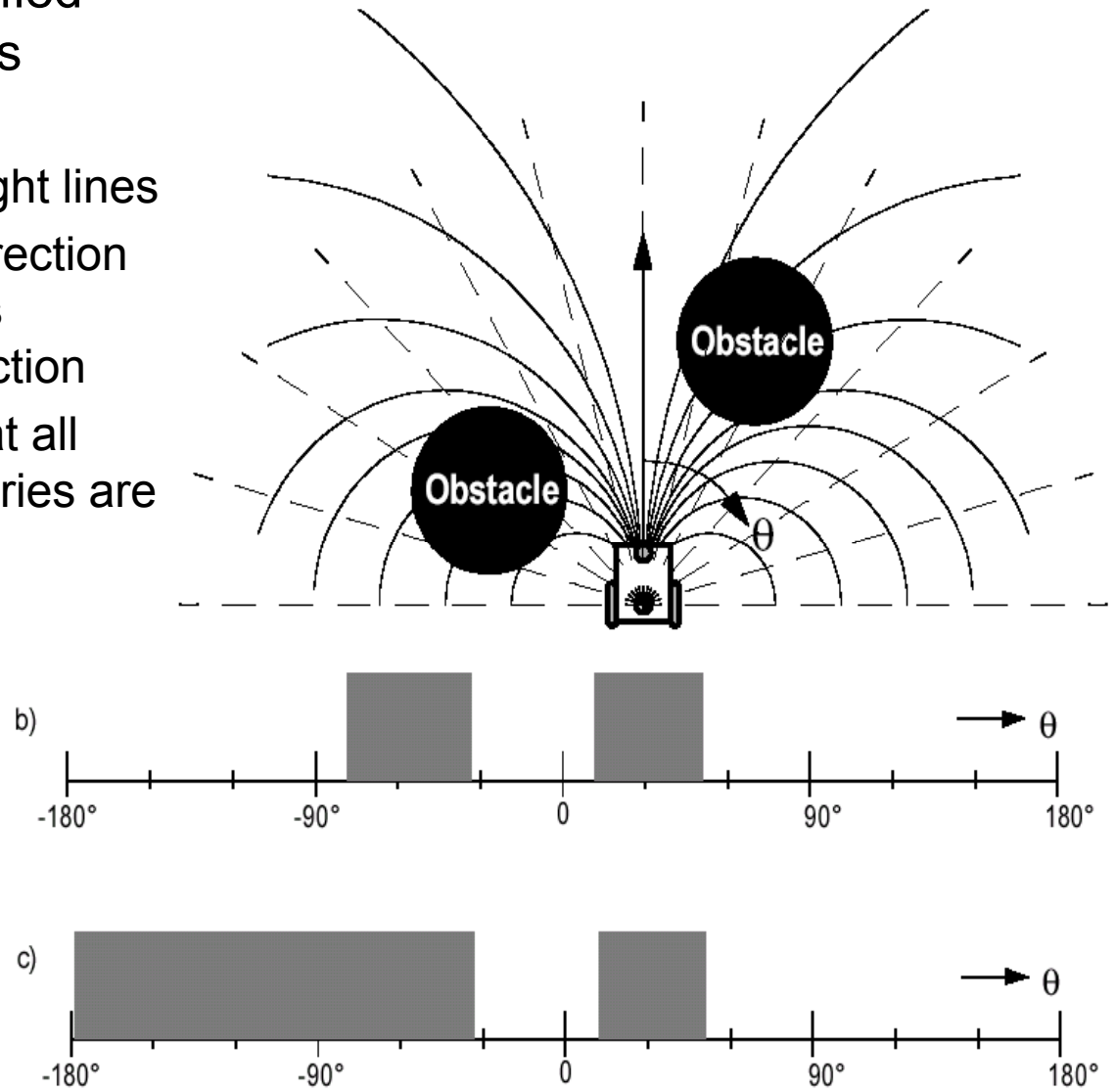




Obstacle Avoidance: Vector Field Histogram + (VFH+)

- Accounts also in a very simplified way for the moving trajectories (dynamics)
  - robot moving on arcs or straight lines
  - obstacles blocking a given direction also blocks all the trajectories (arcs) going through this direction
  - obstacles are enlarged so that all kinematically blocked trajectories are properly taken into account

Borenstein et al.



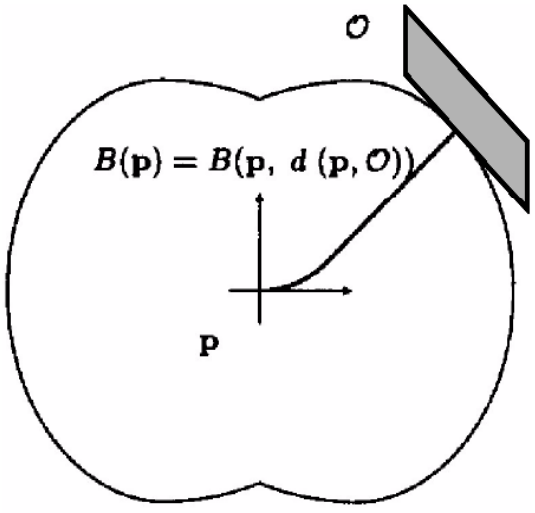
Obstacle Avoidance: Video VFH

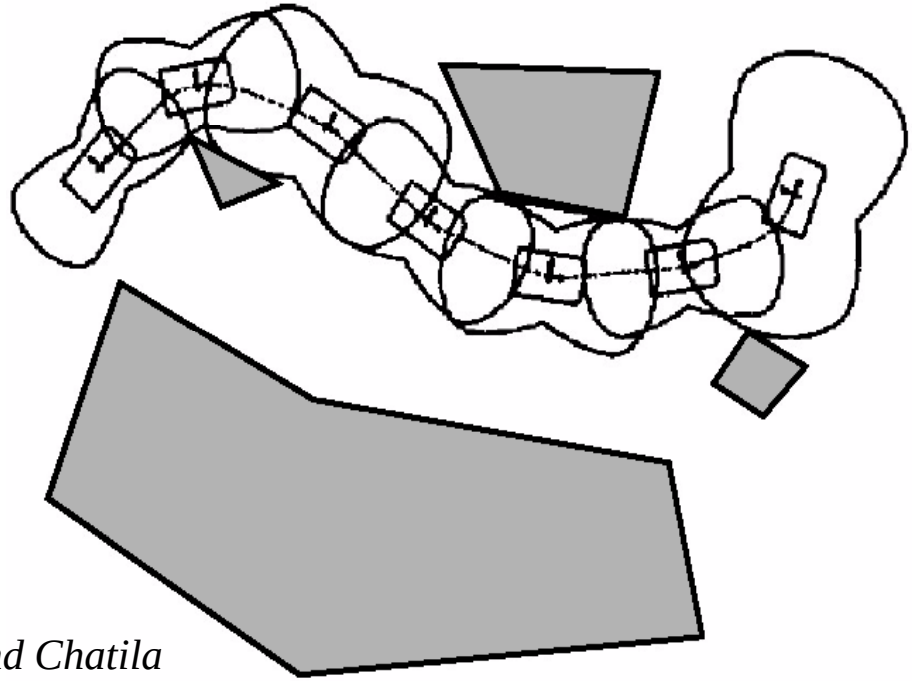
- Notes:
  - Limitation if narrow areas (e.g. doors) have to be passed
  - Local minimum might not be avoided
  - Reaching of the goal can not be guaranteed
  - Dynamics of the robot not really considered

Borenstein et al.

Obstacle Avoidance: The Bubble Band Concept

- 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

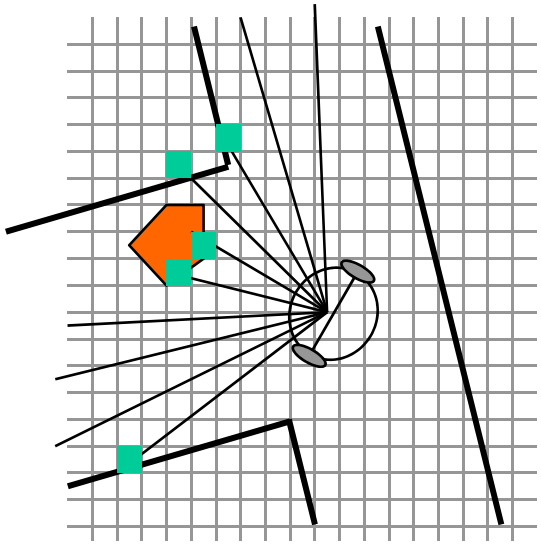


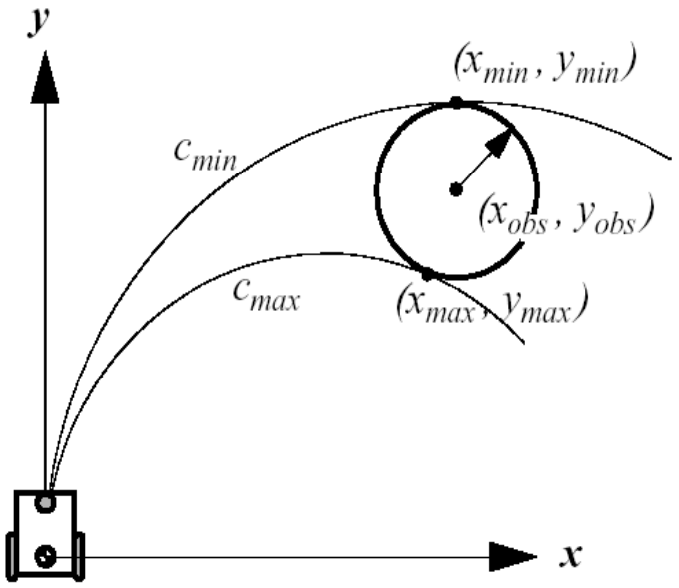


Khatib and Chatila

Obstacle Avoidance: Basic Curvature Velocity Methods (CVM)

- Adding physical constraints from the robot and the environment on the velocity space (v, w) of the robot *Simmons et al.*
  - Assumption that robot is traveling on arcs (c= w / v)
  - Acceleration constraints: -Vmin < v < Vmax, Wmin < w < Wmax
  - Obstacle constraints: Obstacles are transformed in velocity space
  - 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

Dynamic Window Approach

- A brief outline [Fox and Burgard, Brock and Khatib]
  - Search space
    - Circular trajectories
    - Admissible velocities
    - Dynamic window

Obstacle Avoidance: Dynamic Window Approach

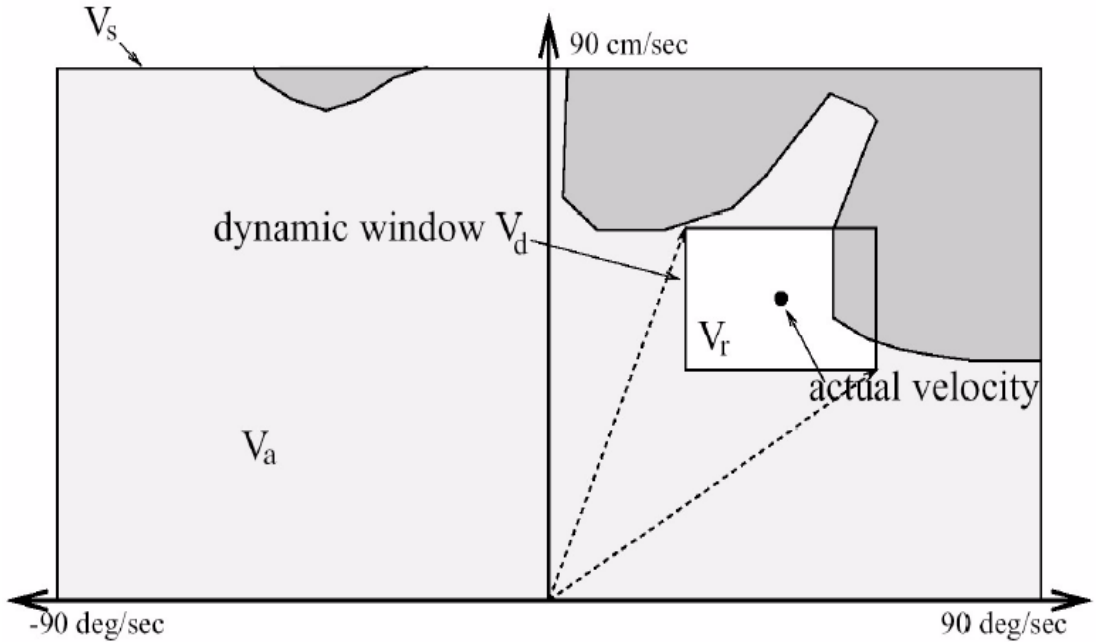
- The **kinematics** of the robot is considered by searching a well chosen velocity space:
  - Circular trajectories : The dynamic window approach considers only circular trajectories uniquely determined by pairs (v,□) of translational and rotational velocities.
  - Admissible velocities : A pair (v,□) is considered admissible, if the robot is able to stop before it reaches the closest obstacle on the corresponding curvature. (b:breakage)
  - Dynamic window : The dynamic window restricts the admissible velocities to those that can be reached within a short time interval given the limited accelerations of the robot

$$V_d = \left\{ (v,\omega) \middle| v \in \left[v_a - \dot{v} \cdot t, v_a + \dot{v} \cdot t \right] \wedge \omega \in \left[\omega_a - \dot{\omega} \cdot t, \omega_a + \dot{\omega} \cdot t \right] \right\}$$

Obstacle Avoidance: Dynamic Window Approach

- Resulting search space
  - The area *Vr* is defined as the intersection of the restricted areas, namely,

$$V_r = V_s \cap V_a \cap V_d$$



Dynamic Window Approach

- Maximizing the objective function
  - In order to incorporate the criteria target heading, and velocity, the maximum of the objective function, $G(v,\omega)$, is computed over V_r.

$$G(v,\omega) = \sigma\left(\alpha \cdot heading(v,\omega) + \beta \cdot dist(v,\omega) + \gamma \cdot velocity(v,\omega)\right)$$

heading = measure of progress toward the goal
dist = Distance to the closest obstacle in trajectory
velocity = Forward velocity of the robot, incoraging fast movements

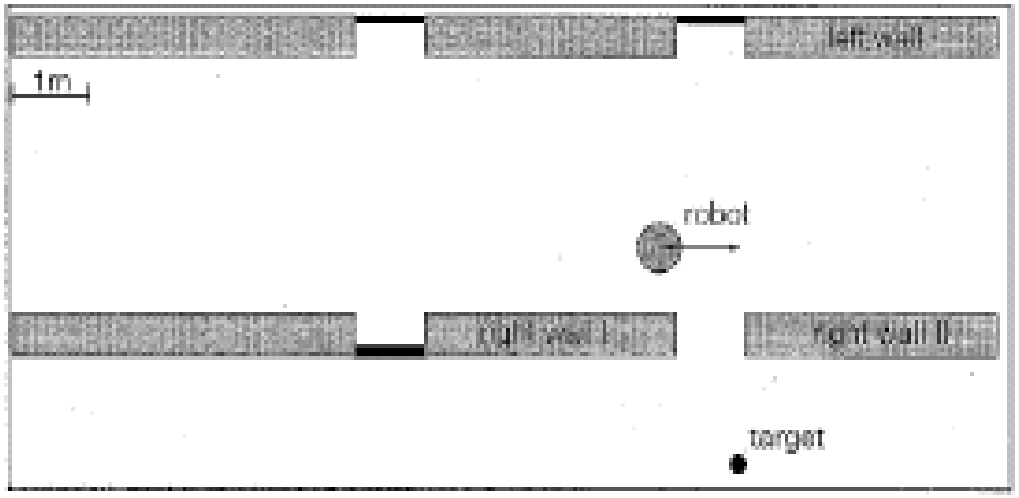


Figure 2. Example Situation.

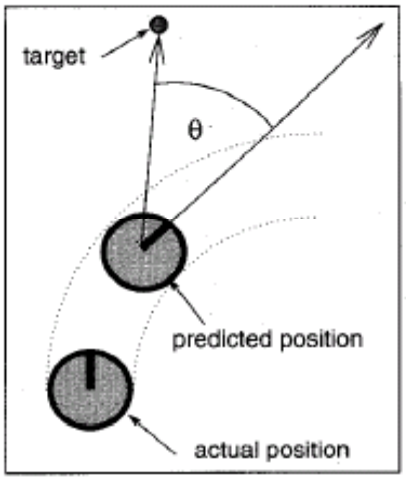
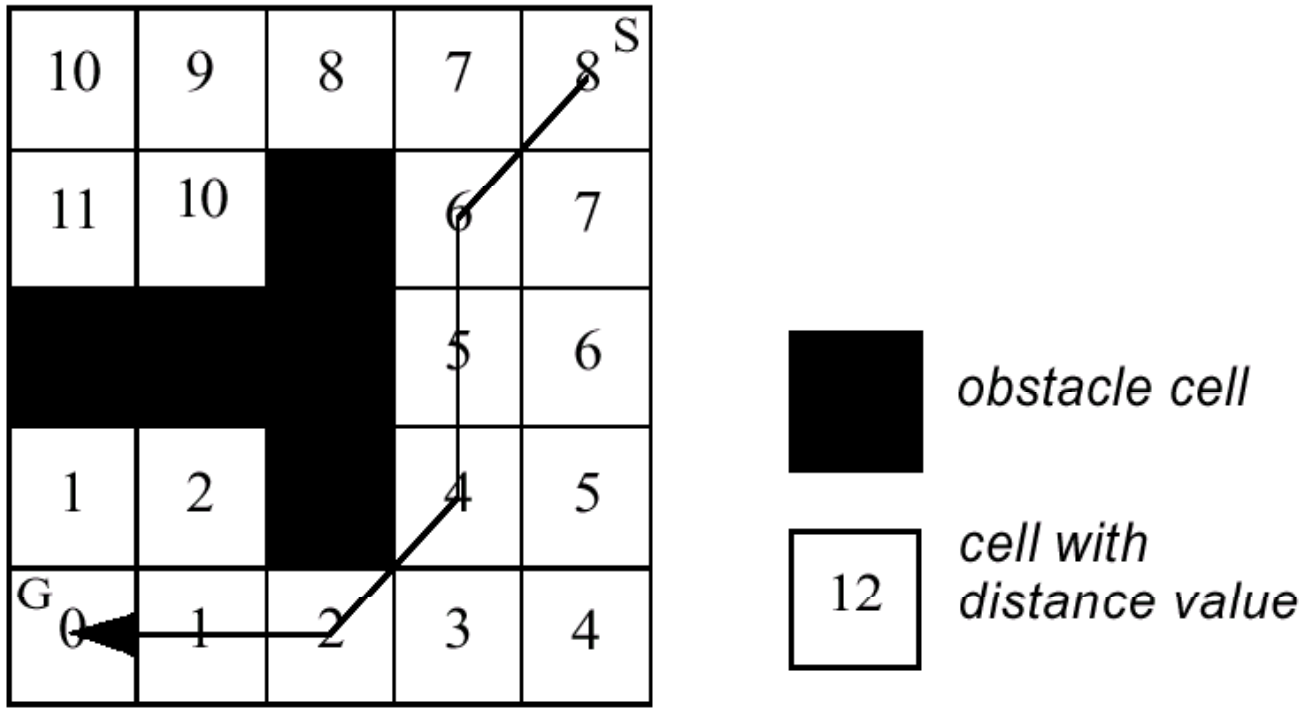


Figure 6. Angle θ to the Target.

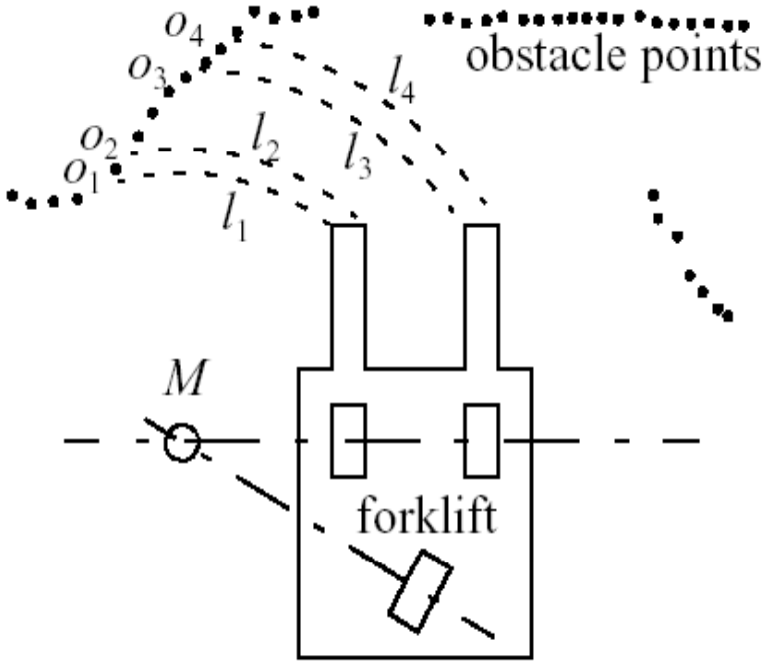
Obstacle Avoidance: Global Dynamic Window Approach

- 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 approach
  - Cartesian grid and motion of circular arcs
  - takes into account the shape of the robot
  - calculate the distance *li* to collision between a single obstacle point and the robot
  - NF1 planner
  - real time performance achieved by use of pre-calculated table

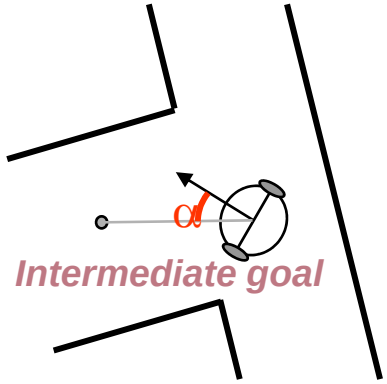


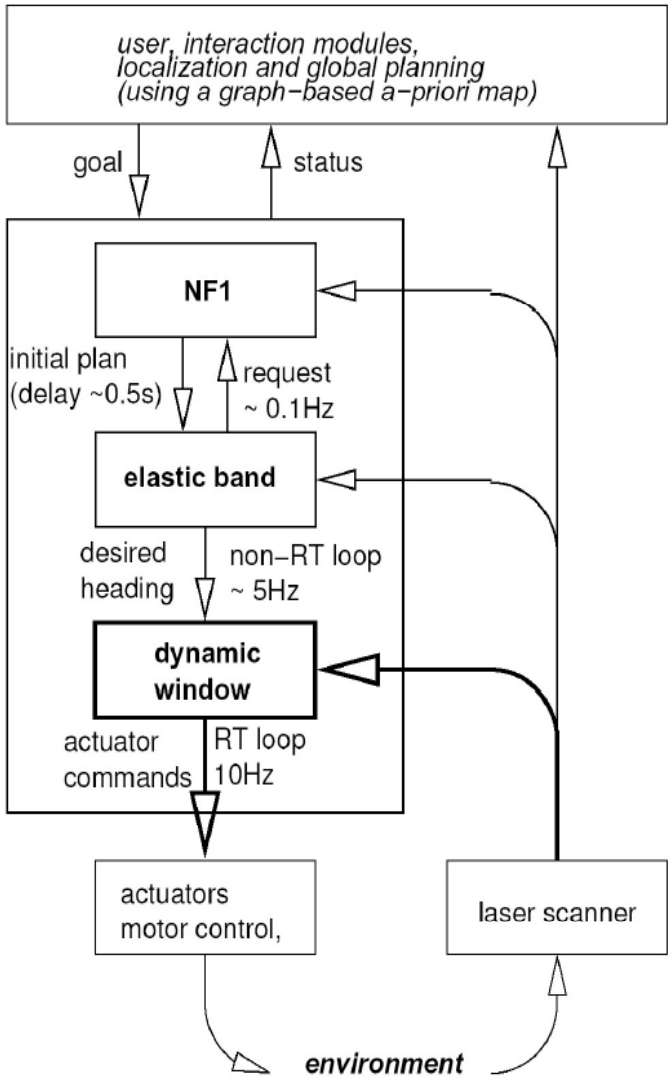
Path Planning and Obstacle Avoidance – ETH-ASL Approach

- 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:

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

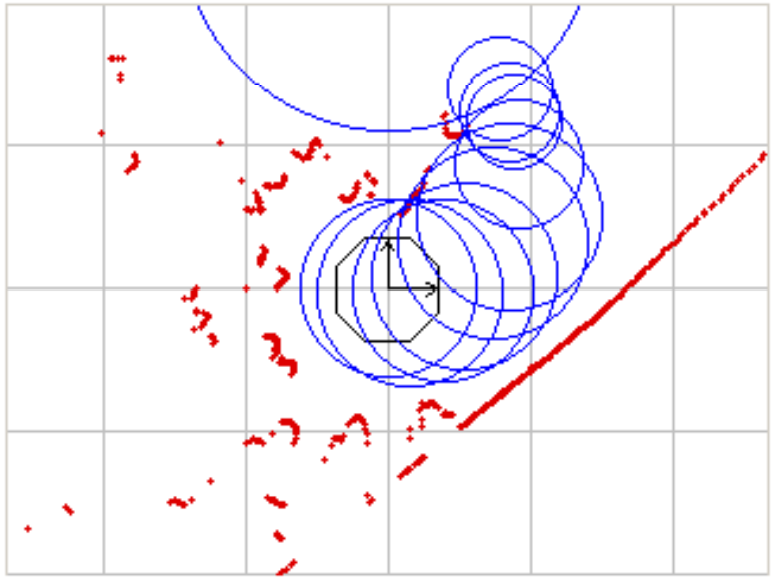
  - speed = v / vmax
  - dist = L / Lmax
  - goal_heading = 1- (a - wT) / p



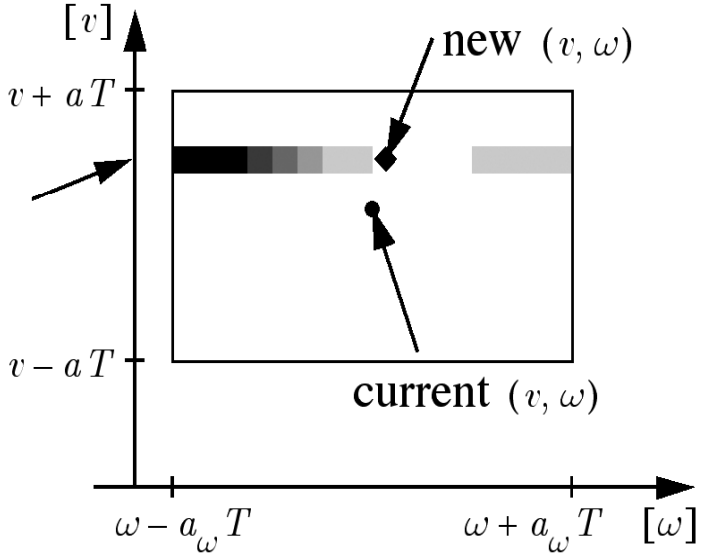


Path Planning and Obstacle Avoidance – ETH-ASL Approach

- A combination of:

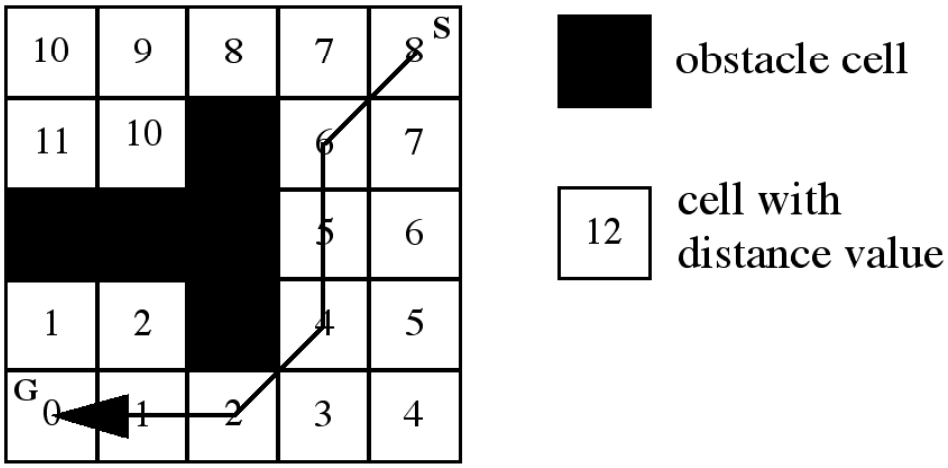


Bubble Band (for smooth trajectories)

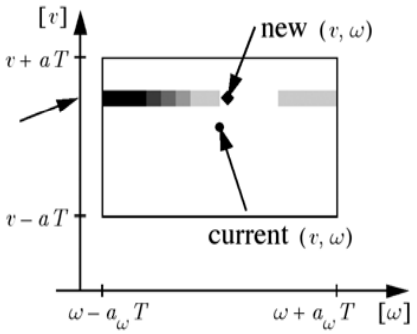


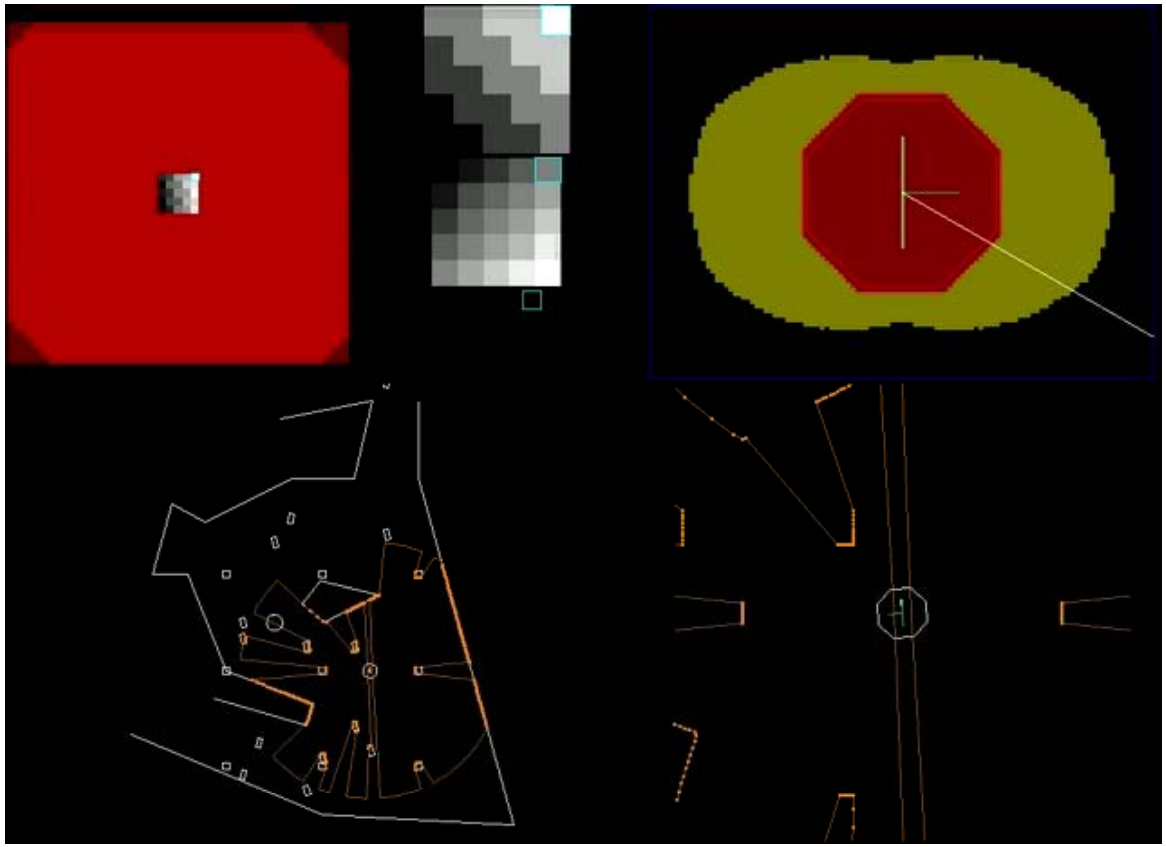
Dynamic Window (to avoid collisions)

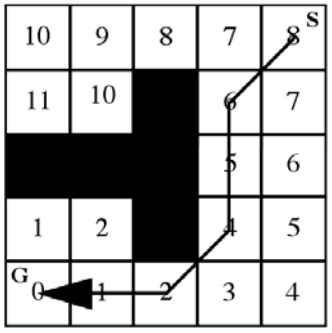
NF1 (local planning)



Path Planning and Obstacle Avoidance – ETH-ASL Approach

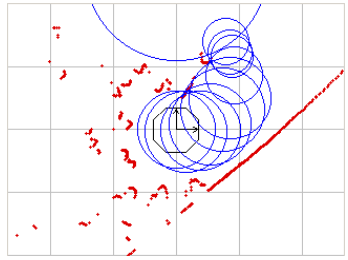






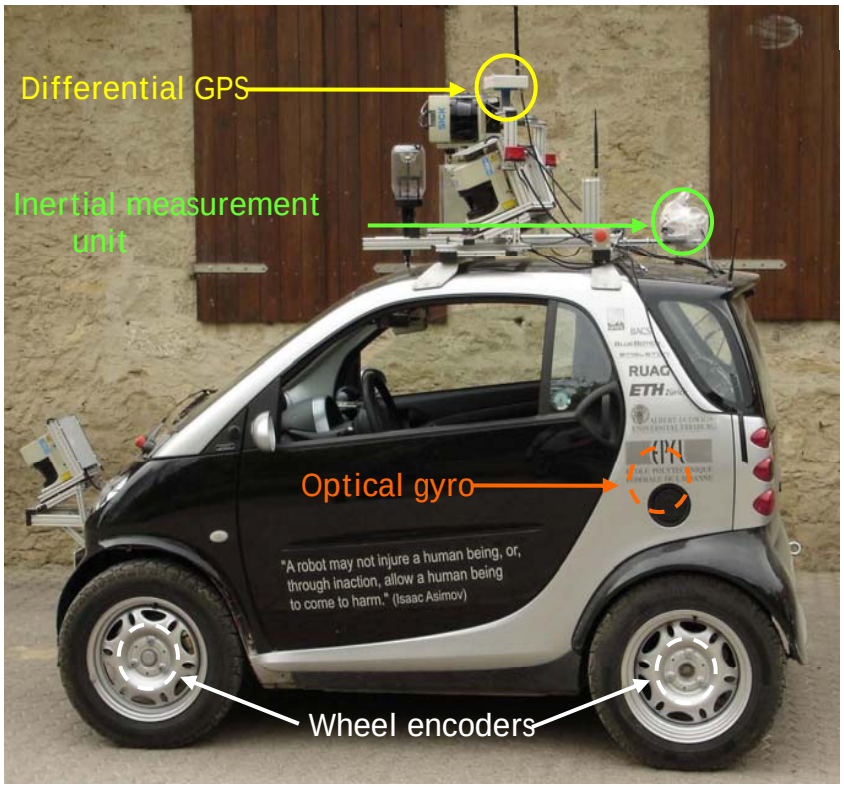
obstacle cell

12 cell with distance value



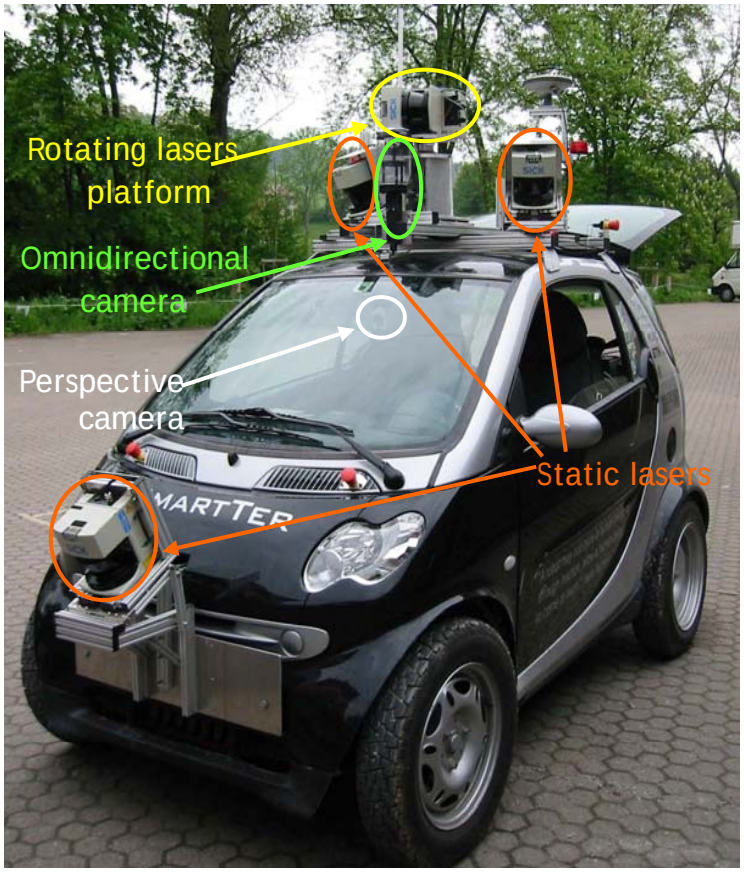
Localization – Position Estimation

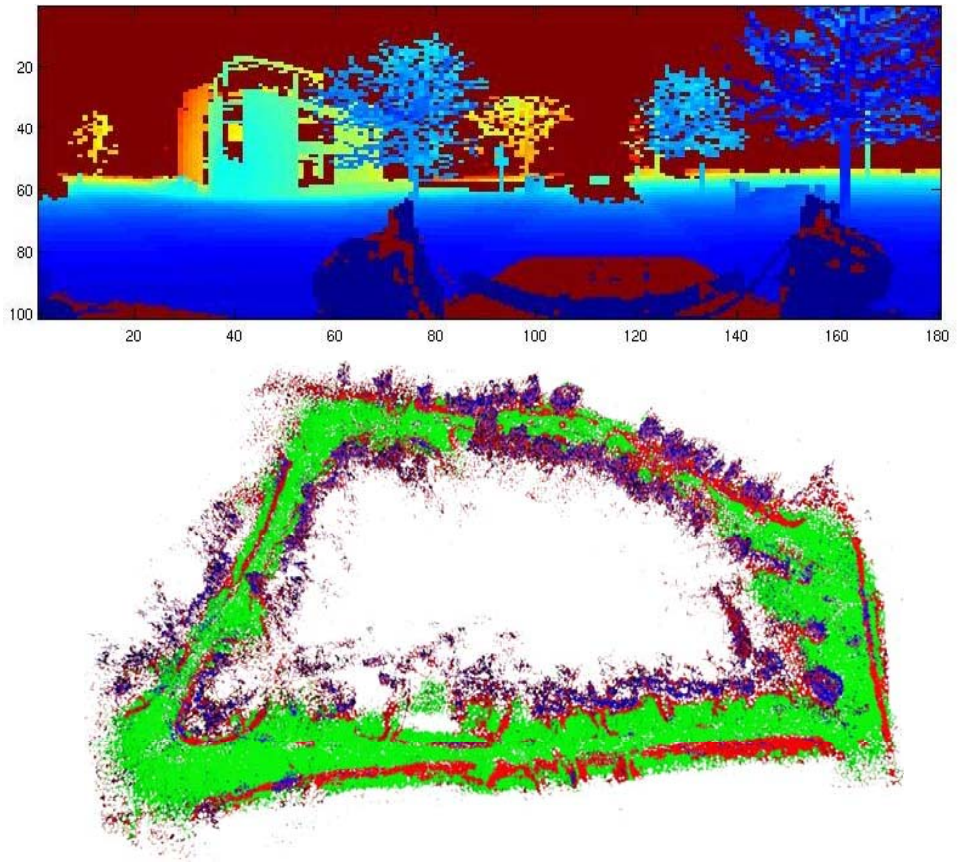
- 6 DOF position estimation based on information filter sensor fusion



Mapping – Environment Representation

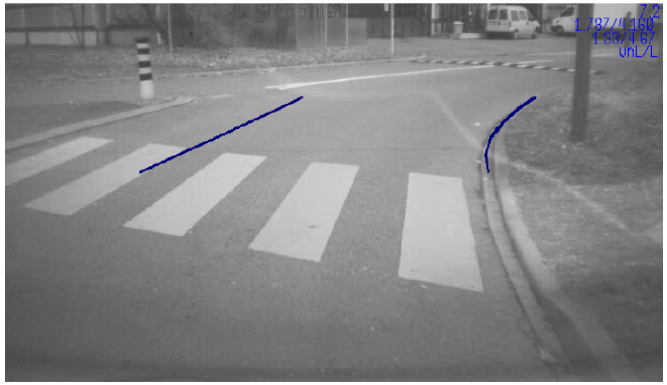
- 3D environment mapping based on laser and vision sensor fusion
- 2D traversability map for navigation in free space

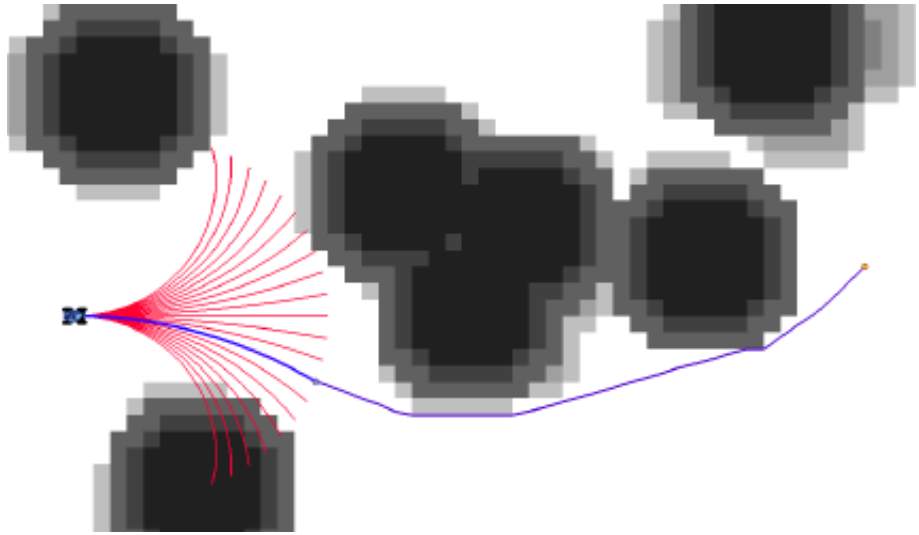


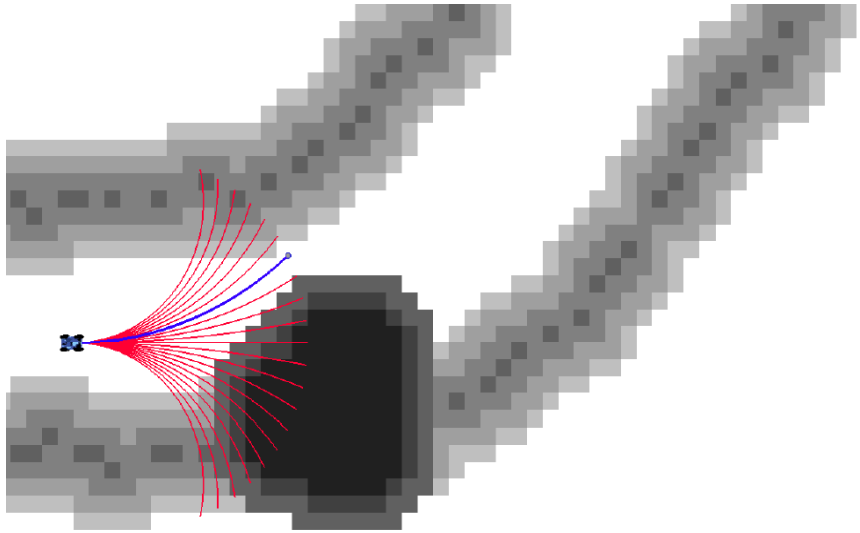


Planning in Mixed Environments

- Autonomous navigation based on extended D* and traversability maps

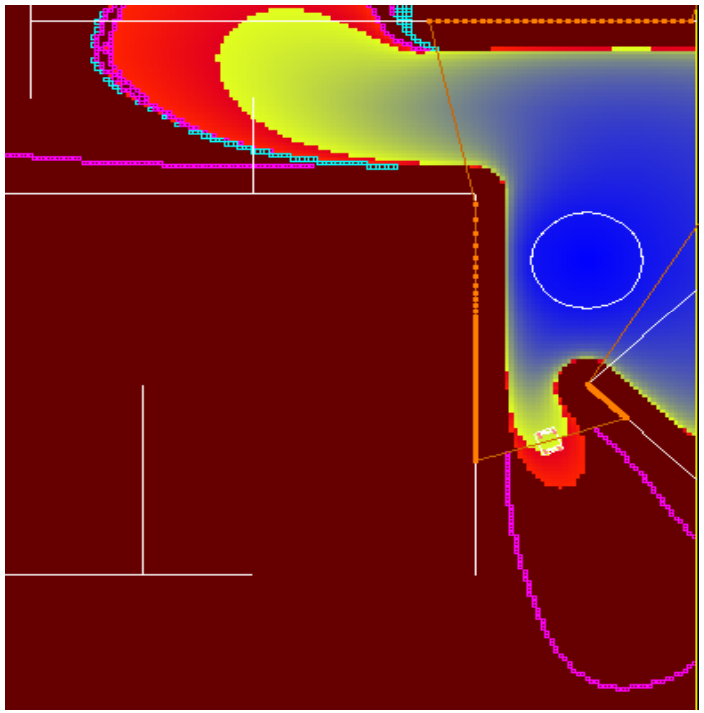




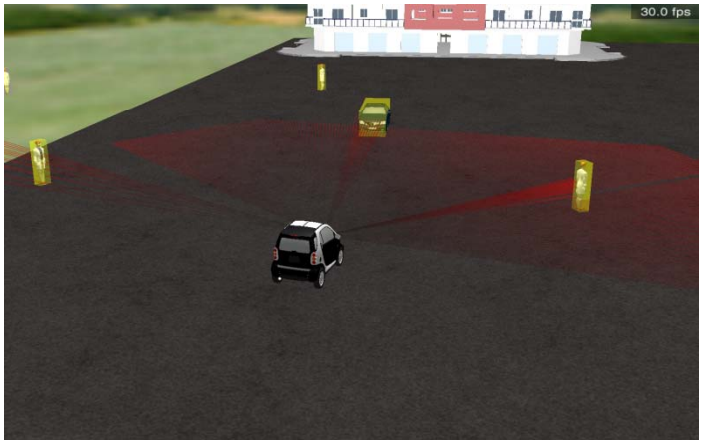


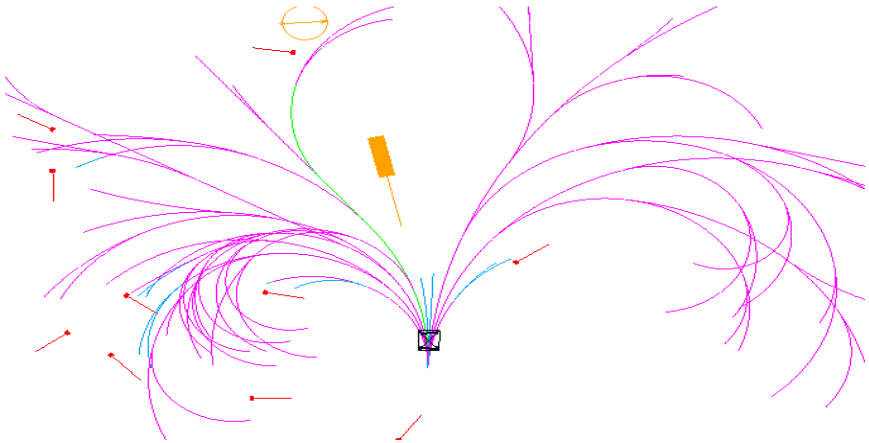
Navigation in Static and Dynamic Environments

- Global planning (E* or FD*) with traversability map
- Optimal trajectory based on global trav. cost and gradient of navigation function



- Estimation of object trajectories for multi-step collision check in dynamic environments

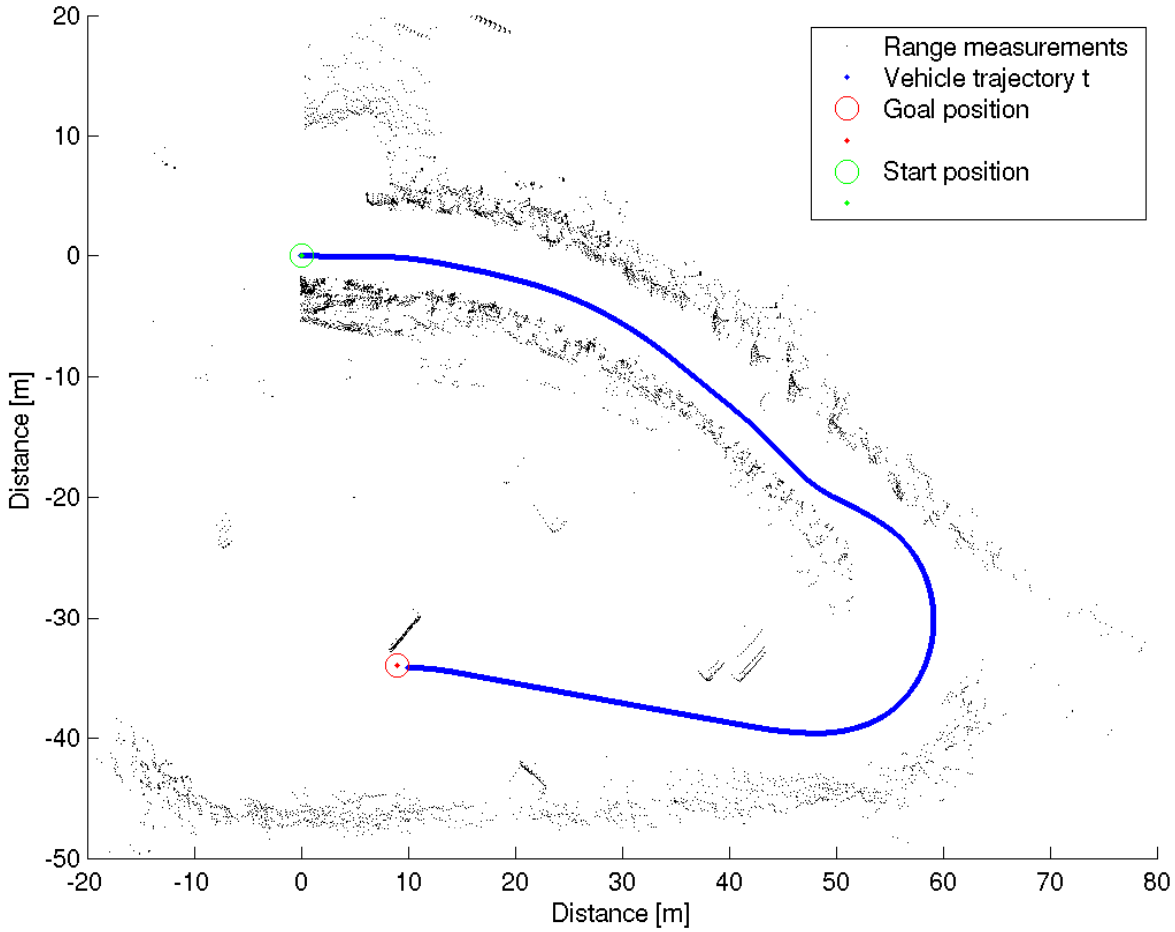




Navigation in Dynamic Environments

Results

- Localization based on odometry and IMU



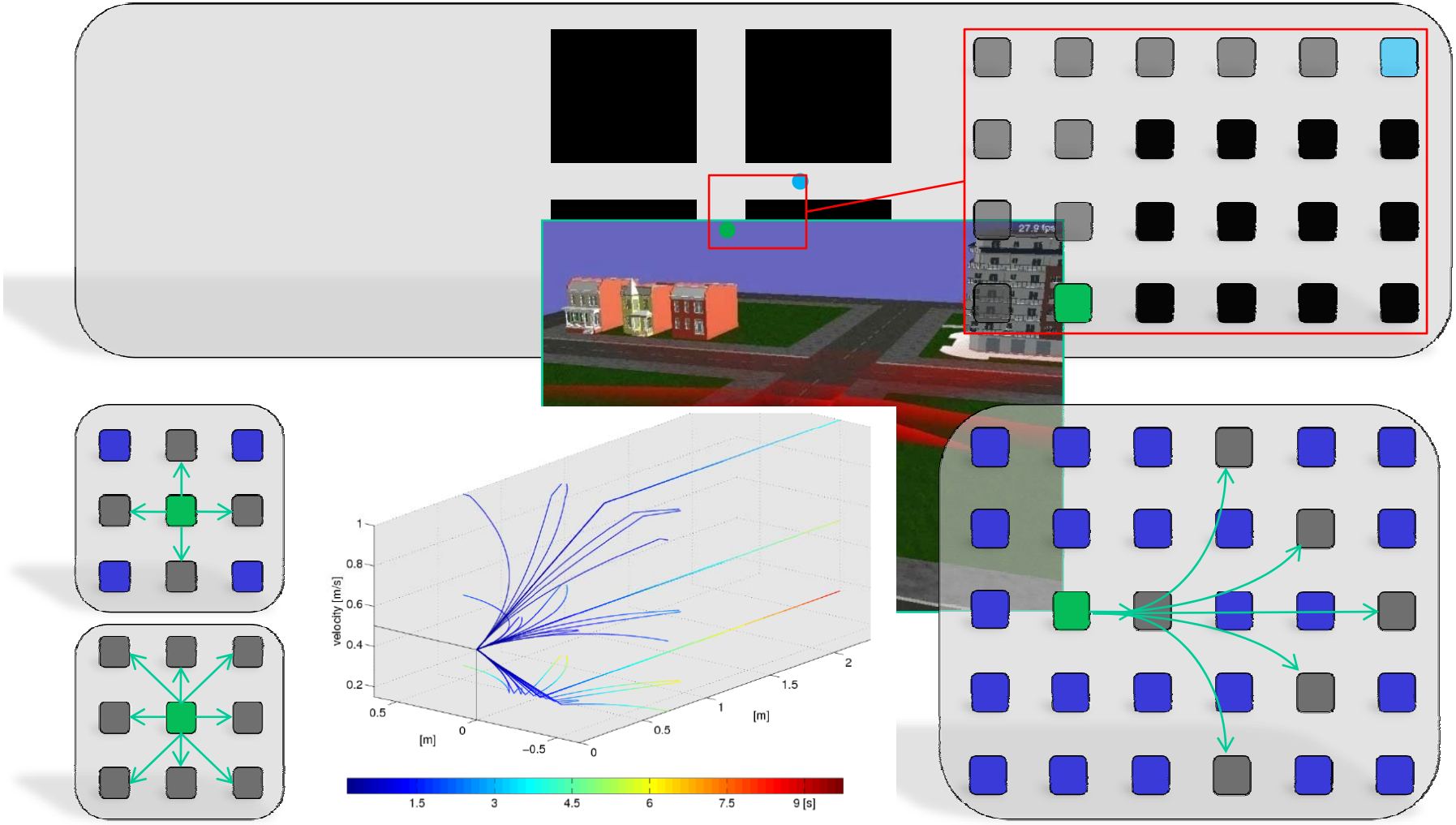
Way Forward: Planning Prerequisites

Martin Rufli

- Direct Creation of *Feasible Trajectories*
  - Design of a n-dimensional State Lattice
- Guarantees on *Solution Optimality*
  - Deterministic search
- Global *and* Local Planning
  - Unified Framework: Design of a Single Planner
- *Low Exectution Time* (i.e. <0.1[s])
  - Planner becomes (MP-) Controller

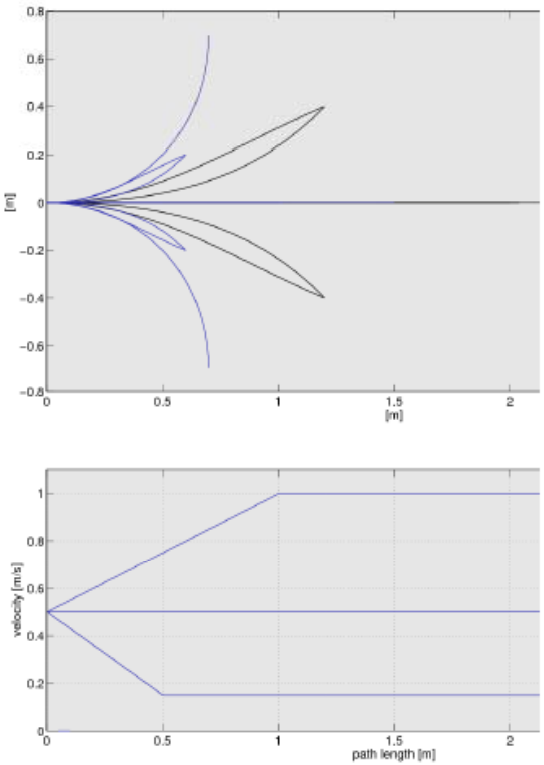
Graph Search on a State Lattice

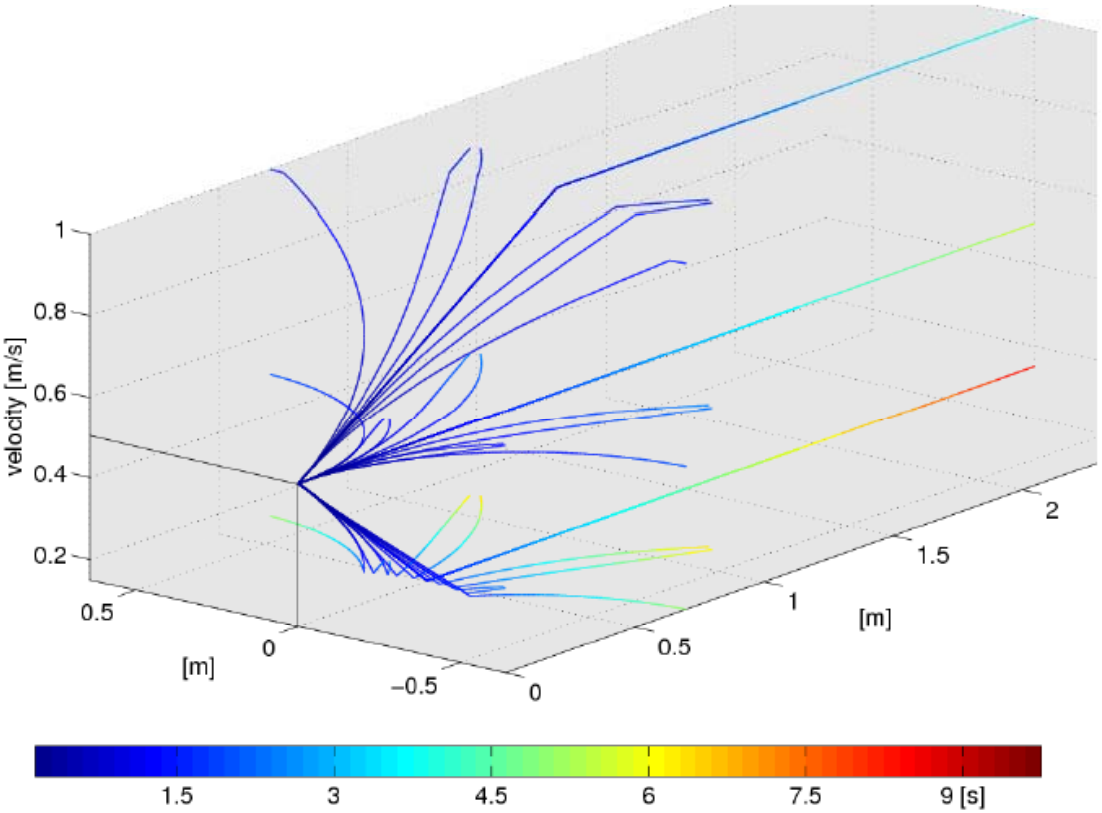
Martin Rufli



4D State Lattice

Martin Rufli





Segway RMP in Narrow Environments

Martin Rufli

- Planning in 4D: (x, y, heading, velocity)

Obstacle Avoidance: Other approaches

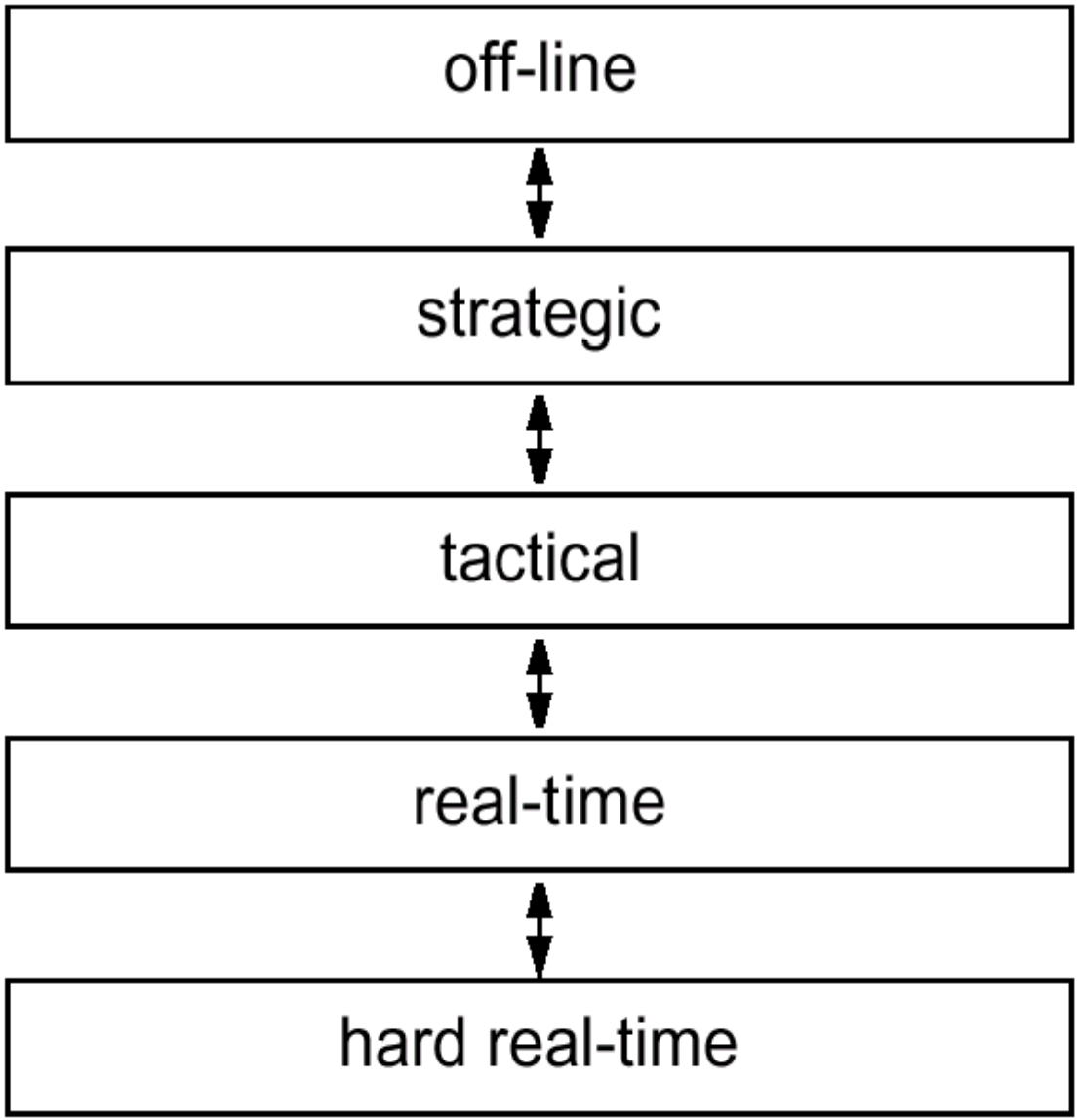
- Behavior based
  - difficult to introduce a precise task
  - reachability of goal not provable
- Fuzzy, Neuro-Fuzzy
  - learning required
  - difficult to generalize

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	

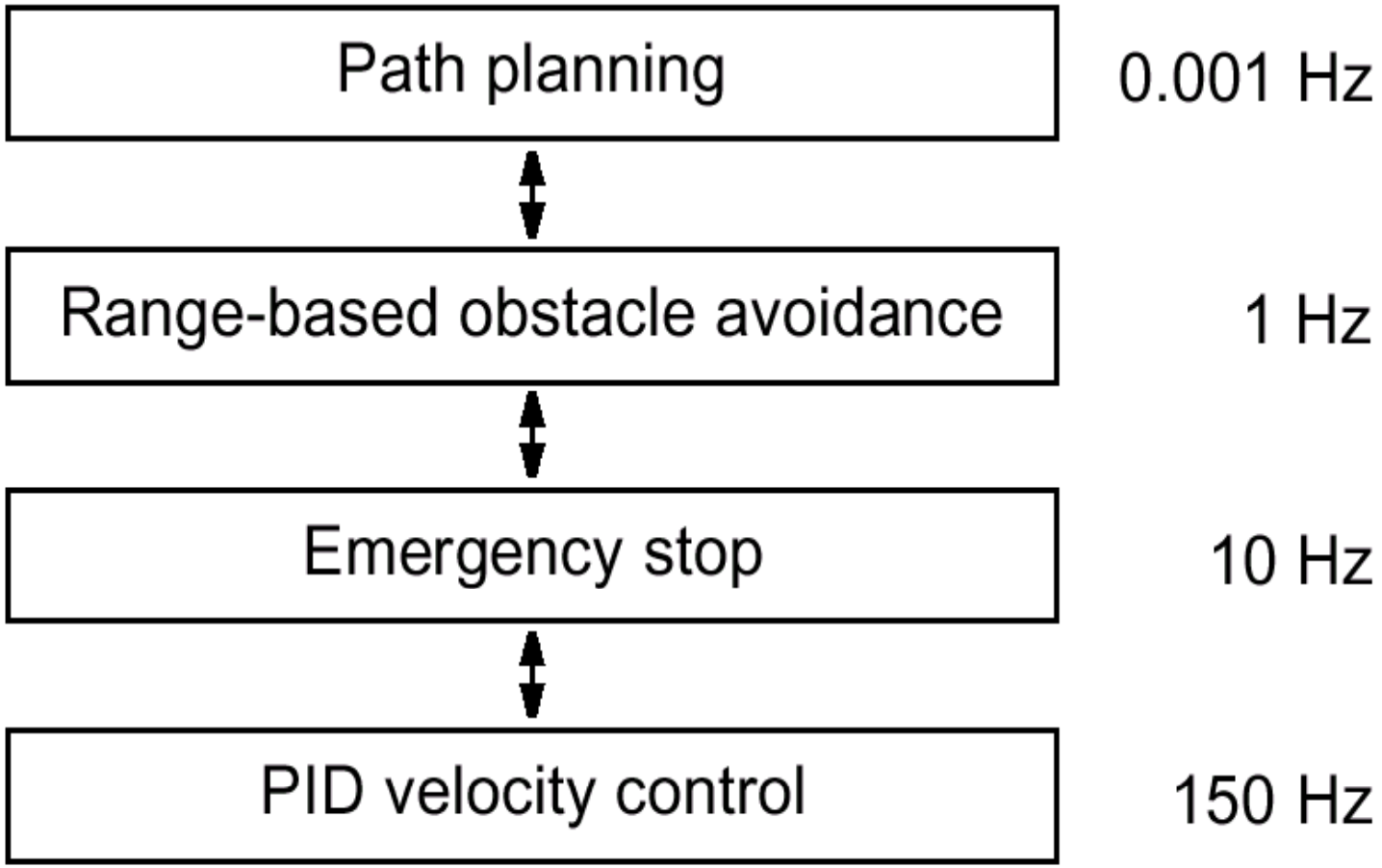
Dynamic window		Curvature velocity		method	
Global dynamic window [44]	Dynamic window approach [69]	Lane curvature method [87]	Curvature velocity method [135]		
circle	circle	circle	circle	shape	model fidelity
(holonomic)	exact	exact	exact	kinematics	
basic	basic	basic	basic	dynamics	
global	local	local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

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	
basic			basic	basic	dynamics	
global	global	local	local	global	view	
	grid		grid		local map	other requisites
local perceptual space	NF1			grid	global map	
fused			graph (topological), NF1	wavefront	path planner	
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	
		local minima	turning into corridors	allows shape change	remarks	

Generic temporal decomposition



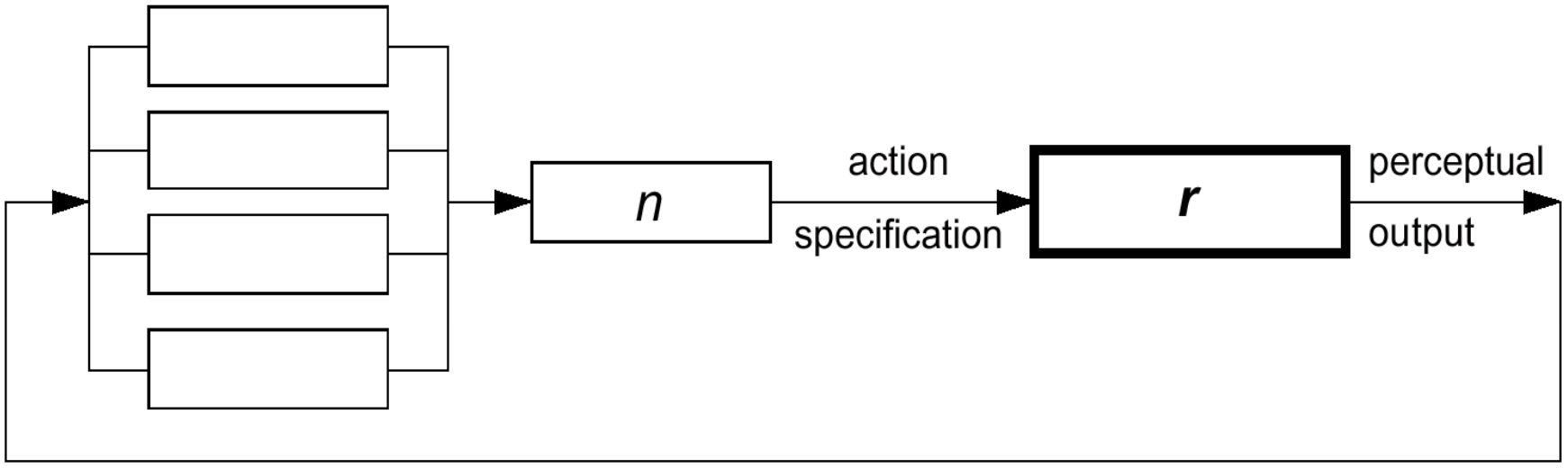
4-level temporal decomposition



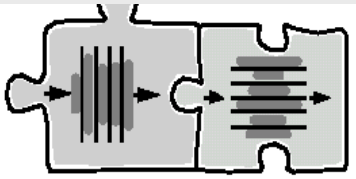
Control decomposition

- Pure serial decomposition

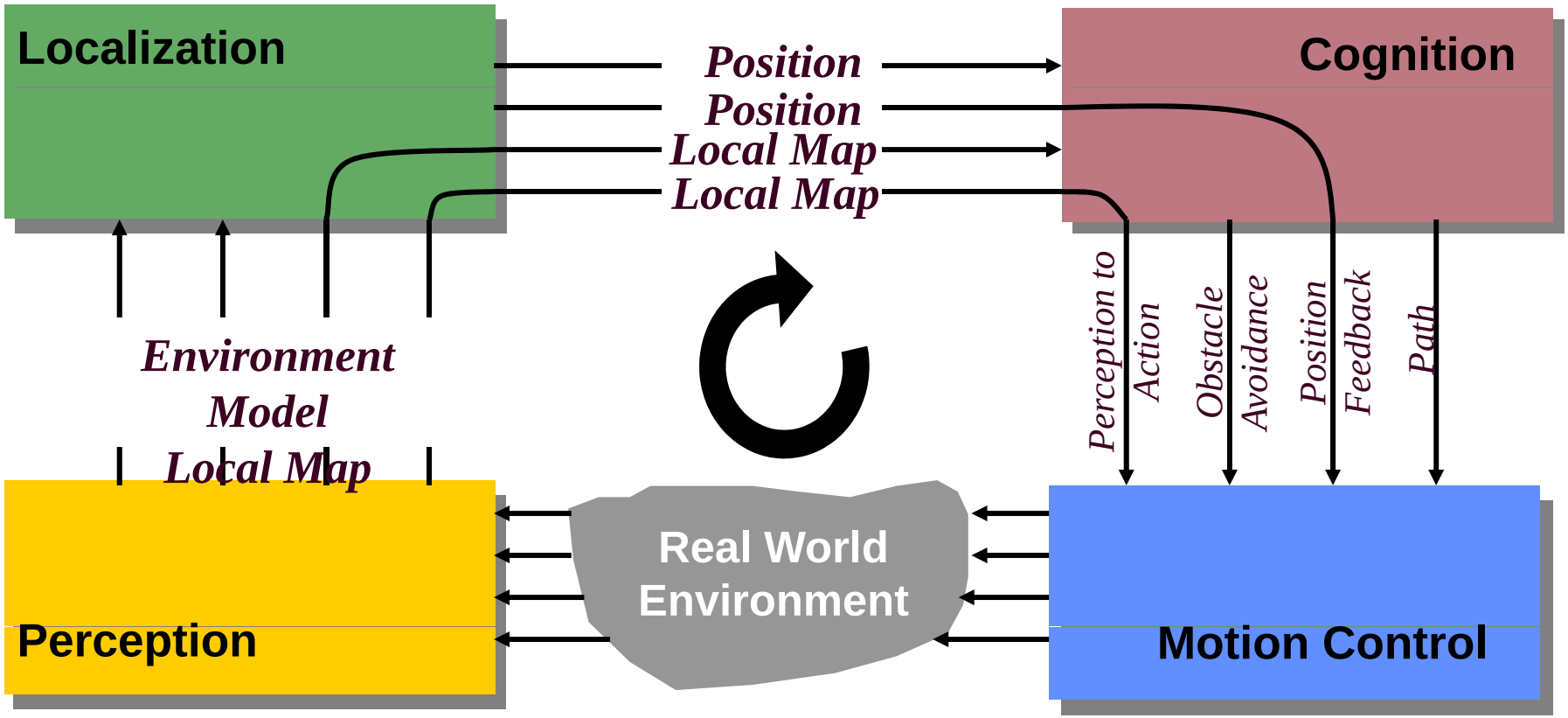
- Pure parallel decomposition



Our basic architectural example

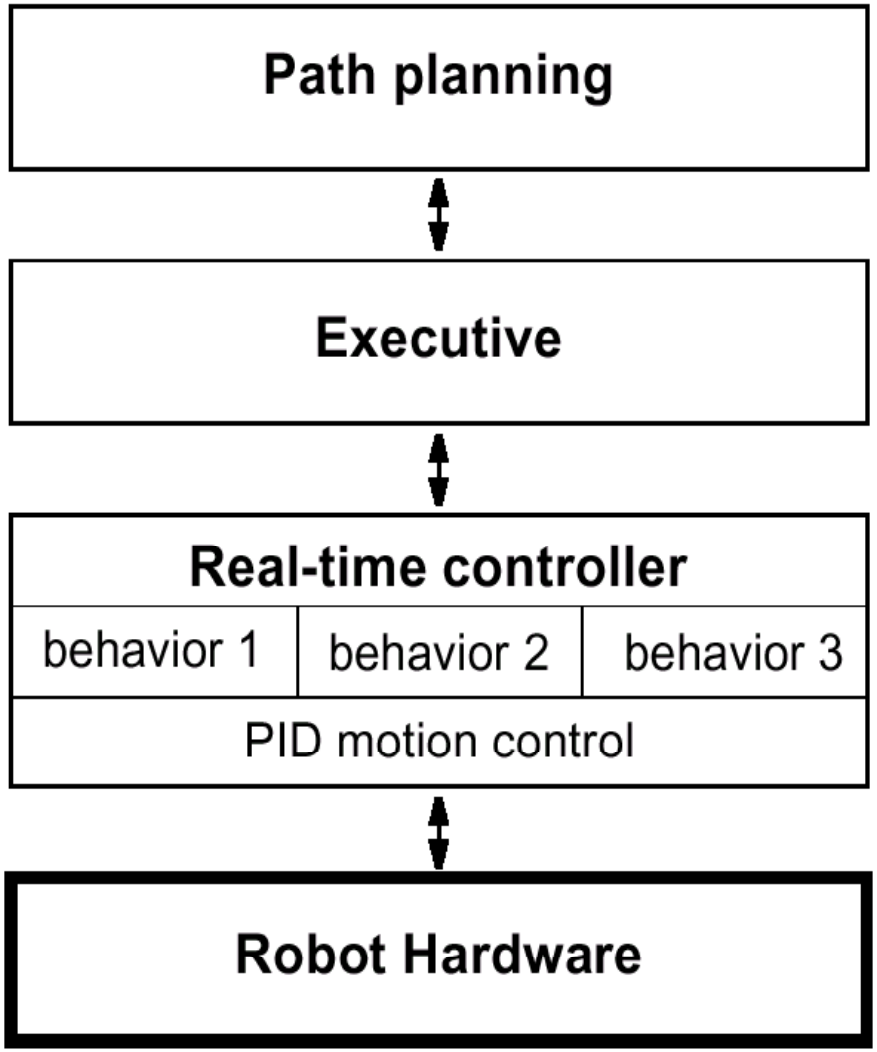


Mixed Approach

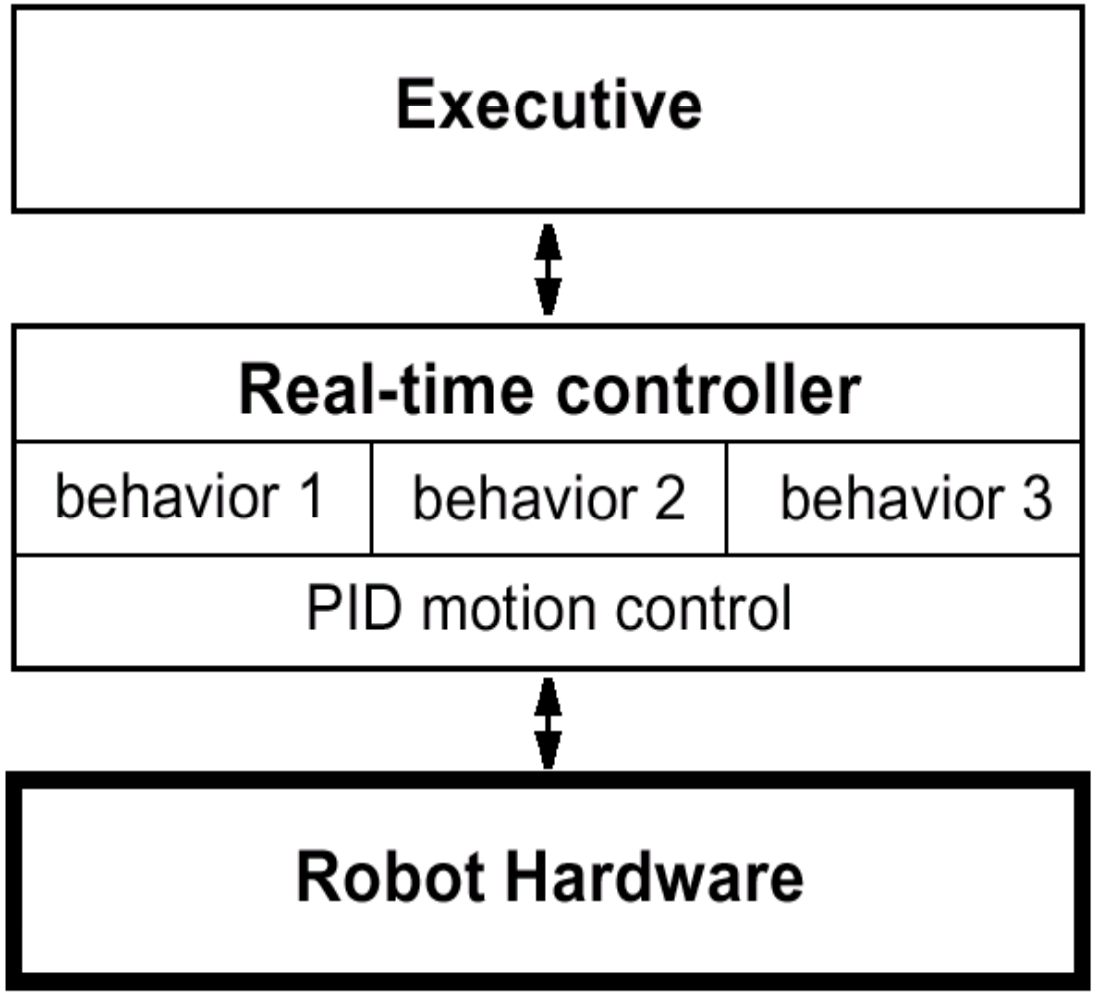


General Tiered Architecture

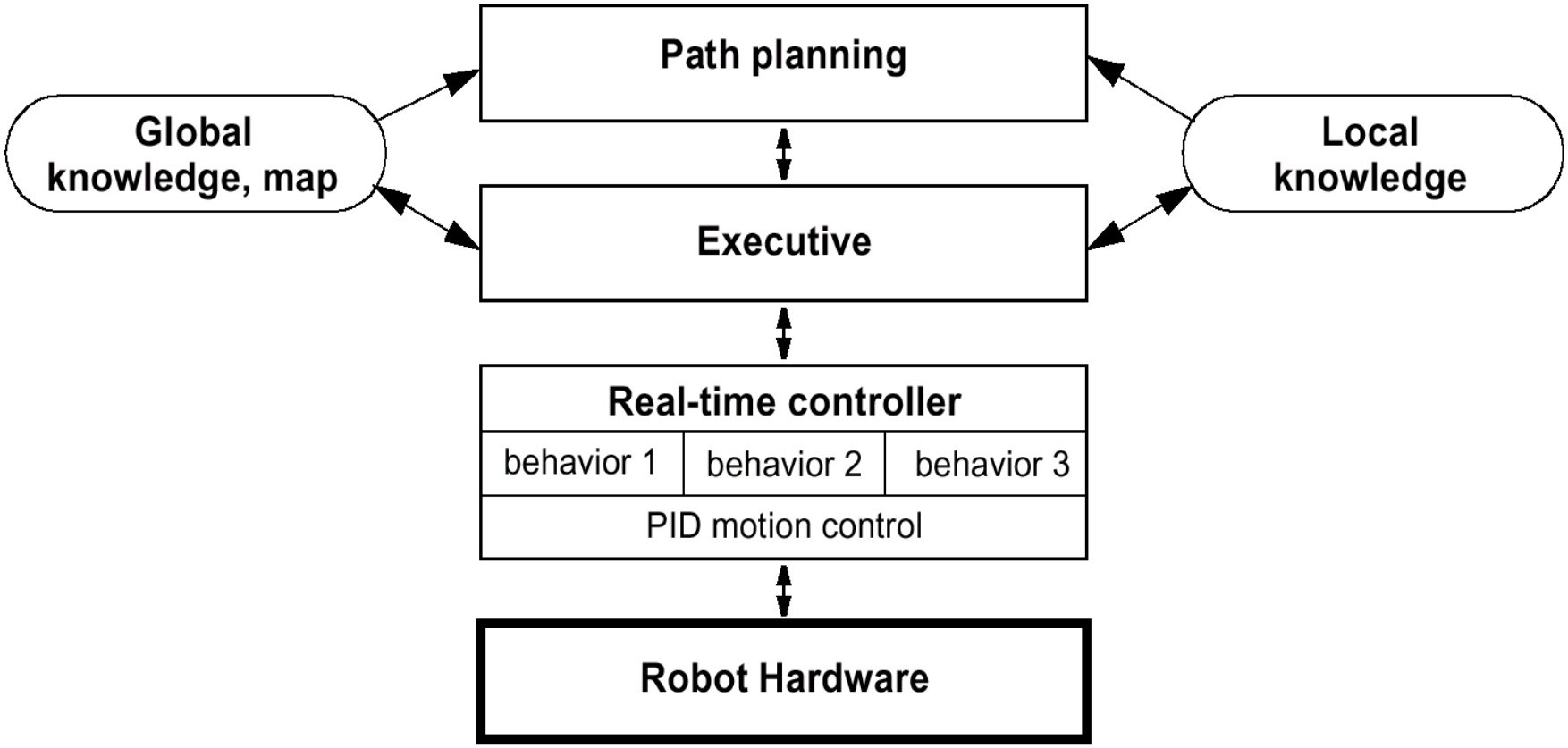
- Executive Layer
  - activation of behaviors
  - failure recognition
  - re-initiating the planner



A Tow-Tiered Architecture for Off-Line Planning

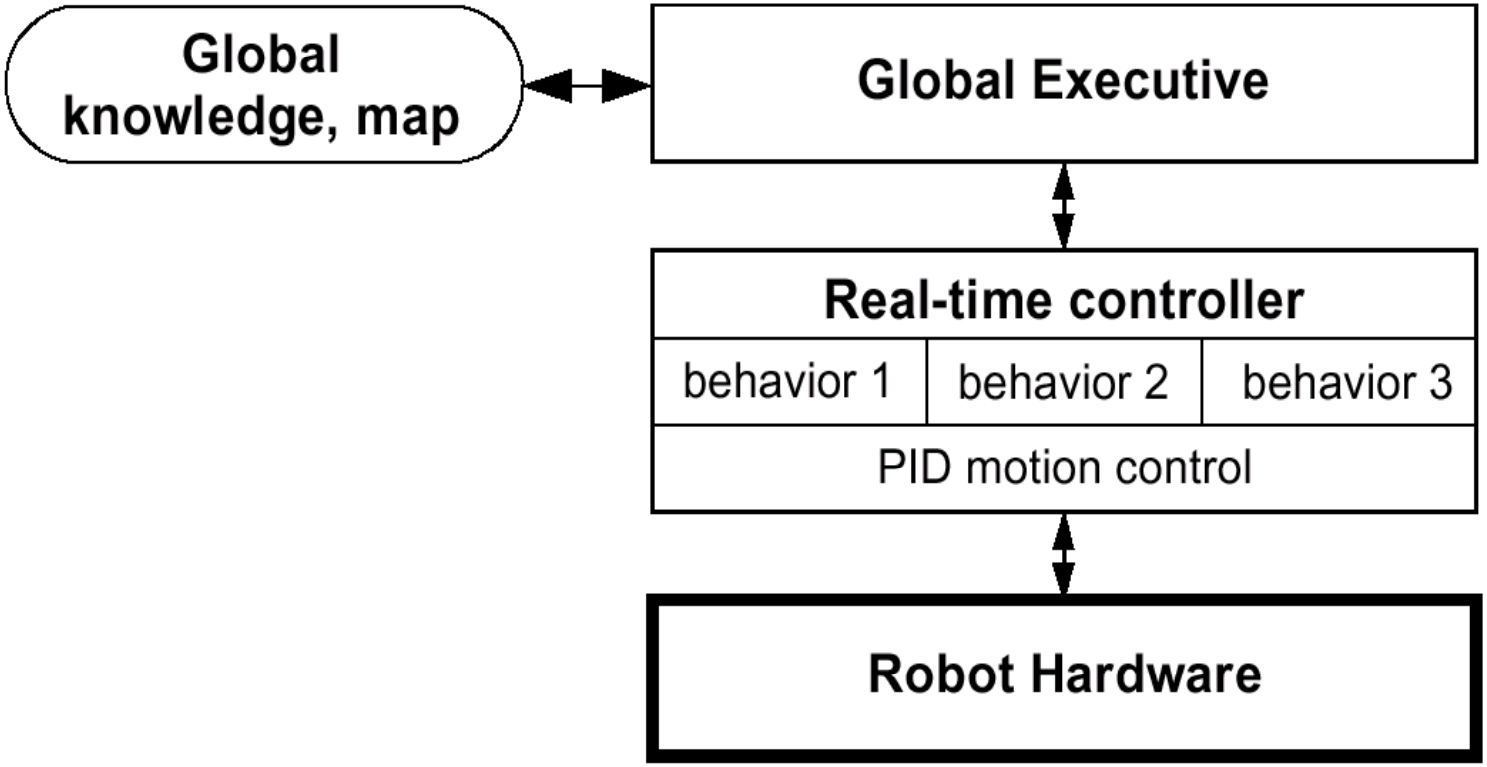


A Three-Tiered Episodic Planning Architecture.



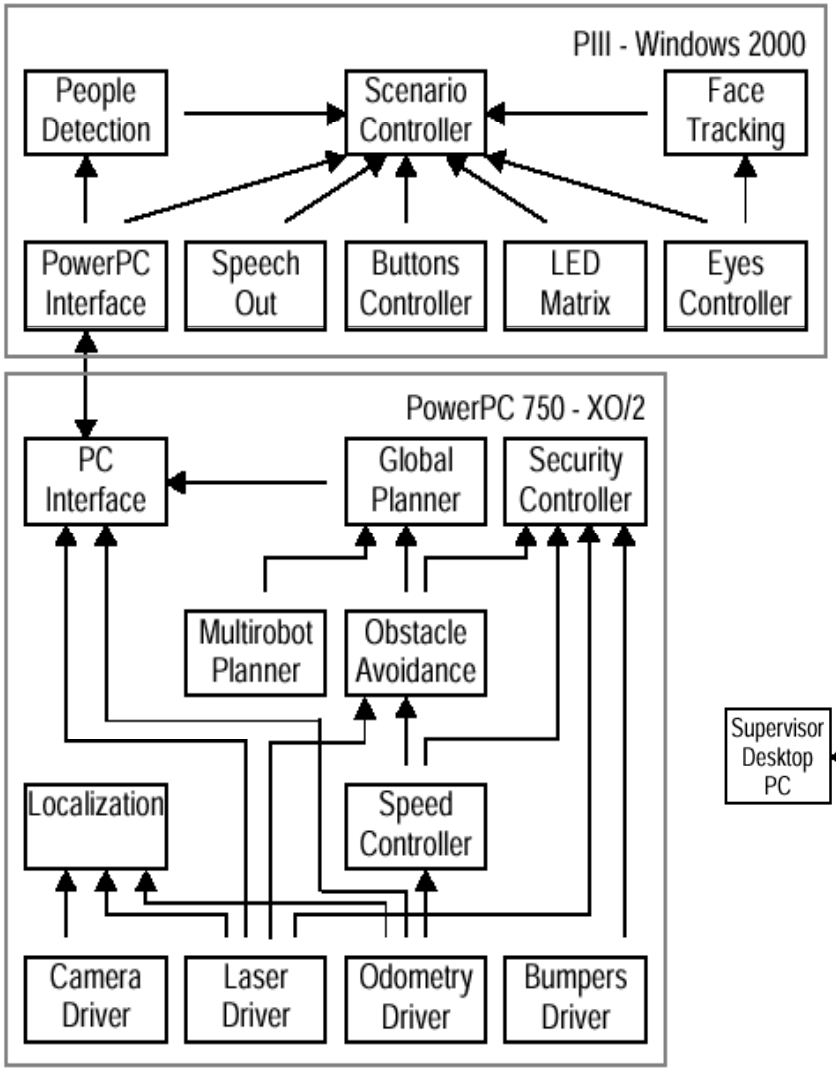
- Planner is triggered when needed: e.g. blockage, failure

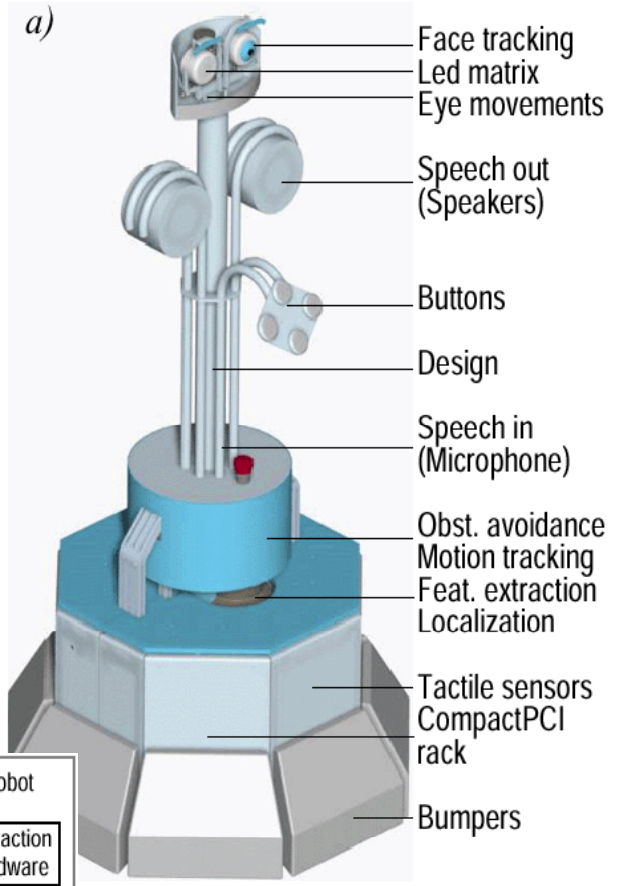
An integrated planning and execution architecture

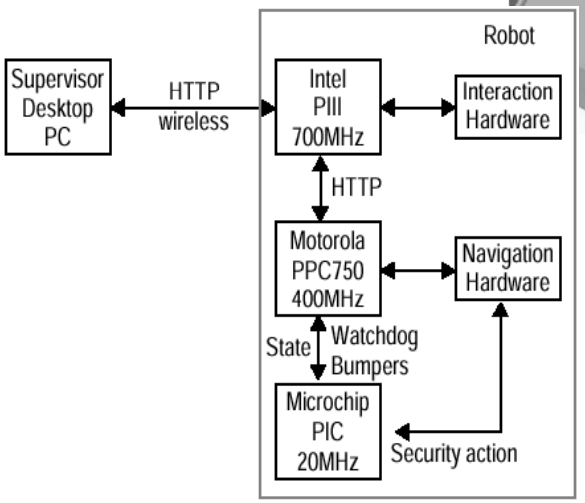


- All integrated, no temporal decoupling between planner and executive layer

Example: The RoboX Architecture







Example: RoboX @ EXPO.02