DD2426 – Robotics and Autonomous Systems
Lecture 8: Planning and Navigation

Patric Jensfelt

Kungliga Tekniska Högskolan
patric@kth.se

April 22,2007

Navigation and Path Planning

- Where am I?
- Where do I need to go?
- How do I get there?

Navigation competencies

- Reasoning and cognition
  - the ability to decide on what actions are required to achieve a particular goal given the present situation
  - decision ranging from what path to take to what information in the environment to use
  - Can be on many different levels of abstraction
- Most of today's industrial robots operate without (or with very little) cognition (reasoning) as the environment is assumed to be static and highly structured
- In mobile robotics (service applications) the environment is non-static and in many cases unstructured.
  - Planning of a path to arrive at a goal
  - Determine the most robust/safe course of actions

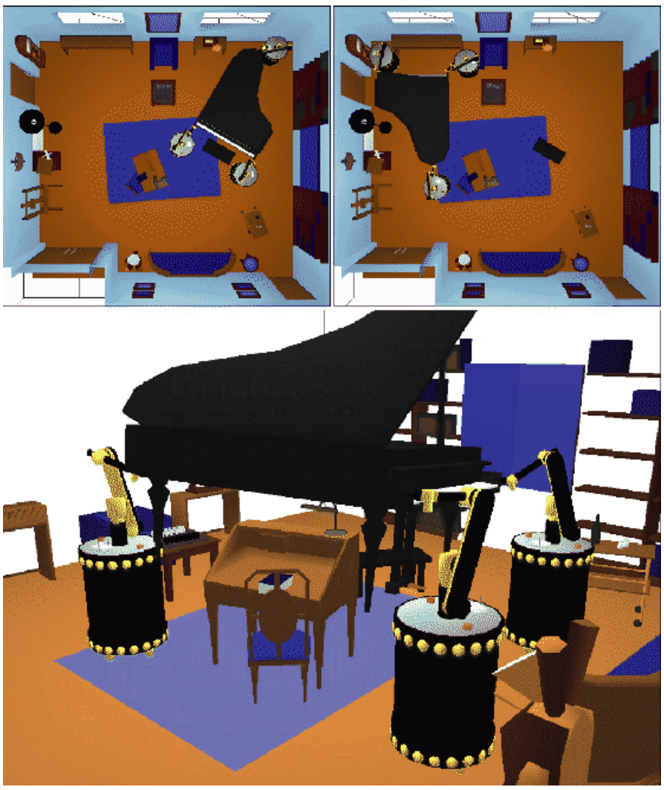
Navigation competencies

- Challenges
  - Knowledge is partial at best
  - The execution if often association with uncertainty
- Control is often decomposed into high level function termed behaviors or function
  - wall following, localization, exploration, door traversal, ...
- Two planning issues
  - Global path planning (resaoning)
  - Local obstacle handling (reactive)

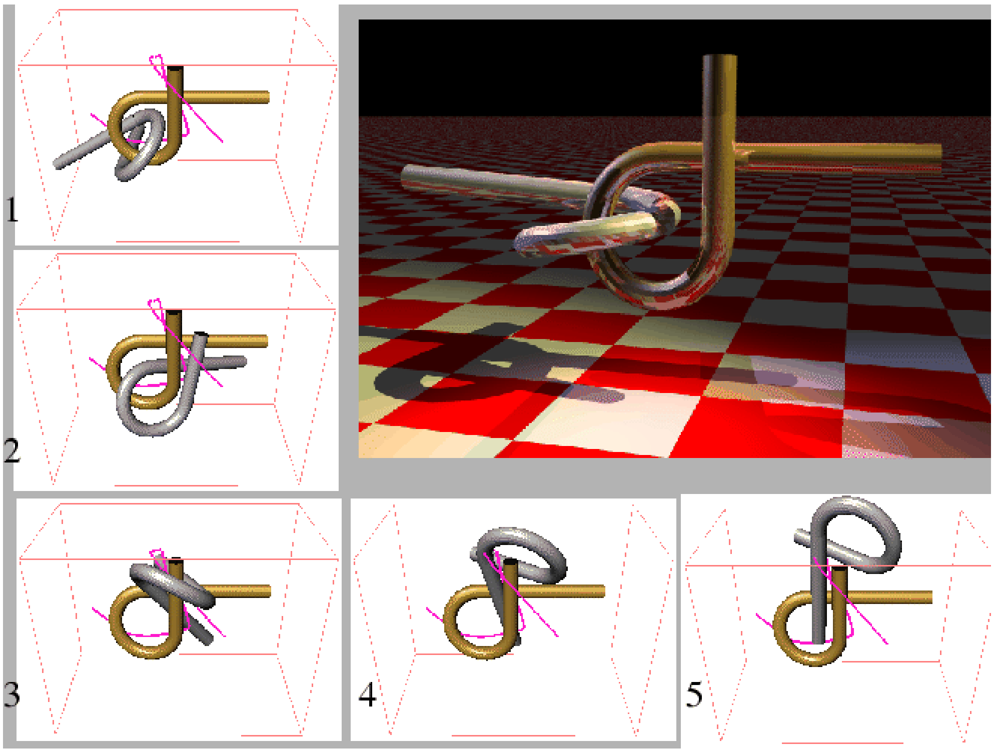
Global Path Planning

- Typical assumption: there is an adequate model/map of the environment for navigation
  - the model could be topological, metric or a mixture
- Steps:
  - Generate a representation of the map for planning
  - Populate the map with a “distance metric”
  - Perform search for the presentation configuration to the goal configuration

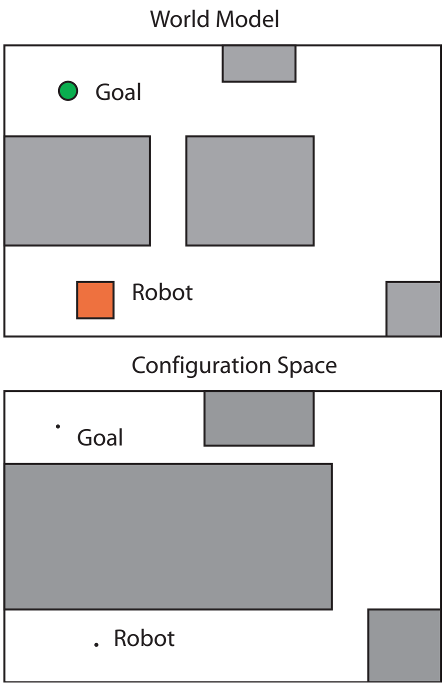
Classic research path planning examples



Classic research path planning examples

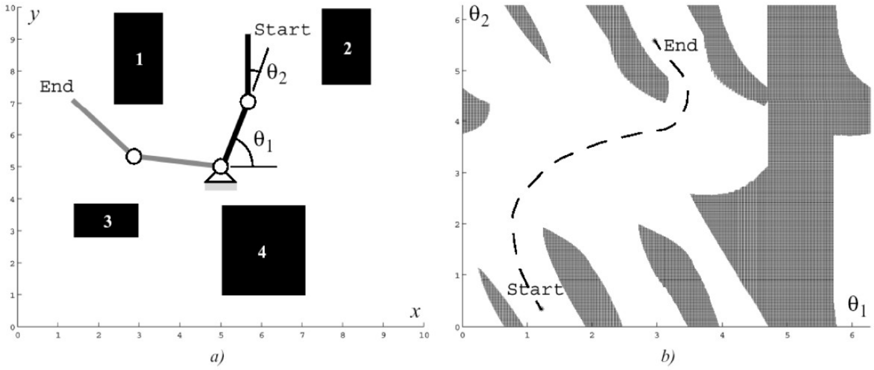


Configuration space (C-space)



- Each pose of the robot is represented as point in the configuration space
- Joint angles often used for manipulators, Cartesian for mobile robots
- The dimensionality of the C space is #DOF
- Robot treated as a point

C space



- Example for a simple manipulator
- Path planning by finding connection between start and goal pose
- The space can be very sparsely populated
- For more info have a look at
  - http://msl.cs.uiuc.edu/planning
  - http://www.laas.fr/˜ jpl/book.html

Path planning with mobile robot

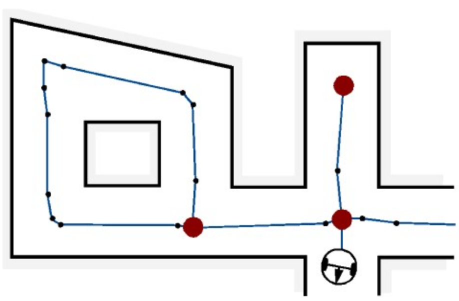
- Most robots are non-holonomic
- However, most path planning methods assume holonomic
- Treats the robot as a point in a 2D C-space (x,y)
- Obstacles are expanded to account for the robot size

Common planning representations

- Configuration space
- Road maps
  - Visibility graph
  - Voronoi diagram
- Cell decomposition
- Potential field

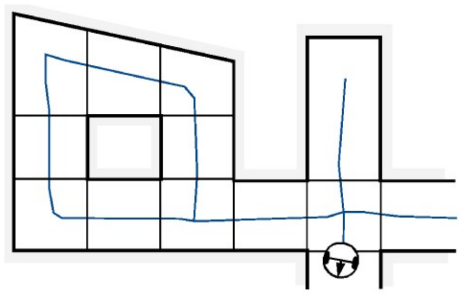
Road-map vs cell decomposition

- Road-map/graph definition



- Selection of nodes?

- Cell decomposition

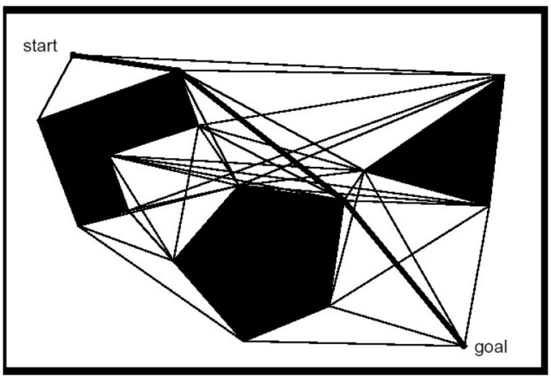


- Divide free and occupied cells
- Where to put boundaries?

Road map path planning

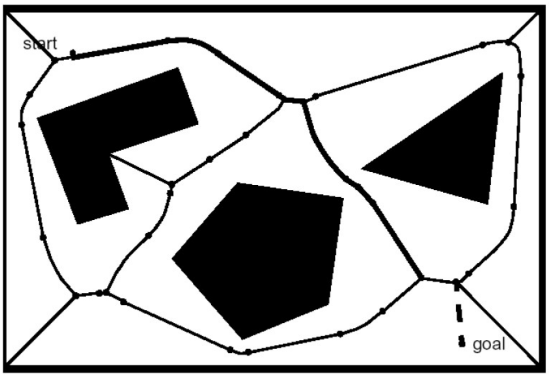
- Build a network of road segments in the free space
- Search for a sequence of roads that connect start and goal
- Want to minimize the number of road segments

Visibility graph



- Connect "visible" vertices of the space
- Perform graph search to generate a path
- Shortest path, but tries to stay close to obstacles
- Little margin for errors in motion

Voronoi Diagram

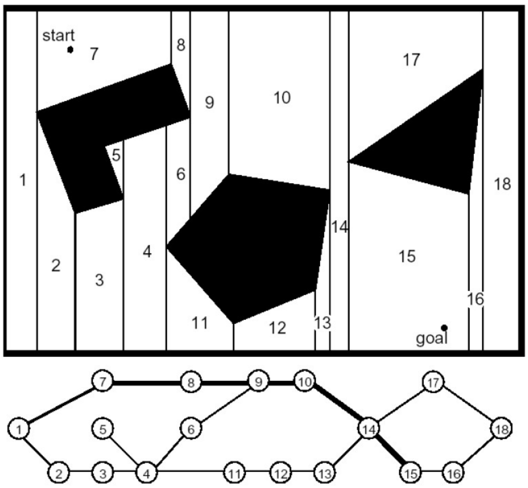


- Path of maximum distance from the obstacles
- Search for shortest path along edges
- Cons: Conservative paths (max distance), unstable to small changes
- Check demo at http://www.cs.cornell.edu/Info/People/chew/Delaunay.html

Cell Decomposition

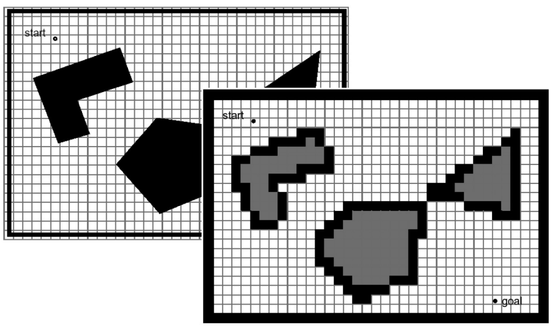
- Divide the free space (F) into simple connected regions called cells
- Generate a connectivity graph with connected open regions
- Locate “goal” and “start” cells
- Search for a path through the graph
- Generate a motion strategy through the cells

Exact cell decomposition



- Any straight line path in one cell is guaranteed not to collide with obstacle if decomposition is exact
- Result could be complex if the world is complex
- Good for sparse environments
- Idea: The position in the cell does not matter, only the ability to travel to another free cell

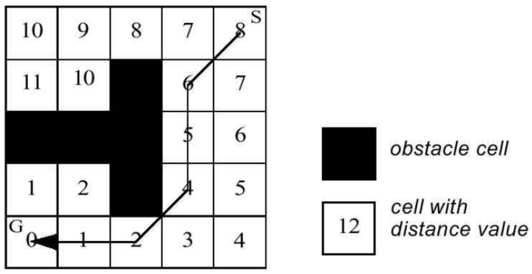
Approximate cell decomposition



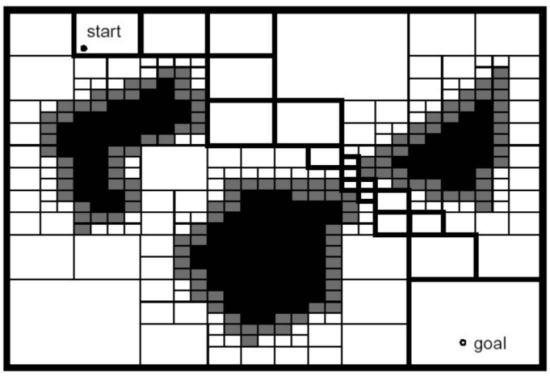
- Occupancy grids are examples of an approximate cell decomposition (see lecture 5 and 7).
- Easy to implement
- Very frequently used

Search strategies

- Wavefront expansion NF1 (see fig)
- Breadth-first search
- Depth-first search
- Greedy search
- A*
- ...



Adaptive cell decomposition



- Effecient representation of space
- Use for example quad-trees
- Well suited for space work spaces

Potential field path planning

- Think of the robot as a particle in a potential field $U(x)$
- The goal serves as an attractor
- Obstacles gives repulsive forces
- With a differential potential field the force is given by

$$F(x) = -\nabla U(x)$$

- Basic idea: Robot is attracted towards the goal and repulsed by the obstacles
- One of the first wide spread approaches to mobile robot obsacle avoidance
- Use superposition principle to get resulting field
  - Generate potential field from goal
  - Generate potential field from each obstacle
  - Sum the potential fields

Potential field details

- The superposition

$$U(x) = U_{goal}(x) + \sum U_{obst_i}(x)$$

- Example attractor

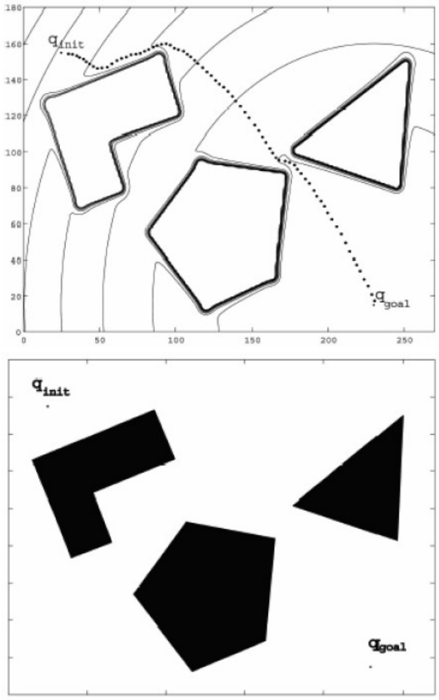
$$U_{goal} = k_{goal}\rho_{goal}(x)$$

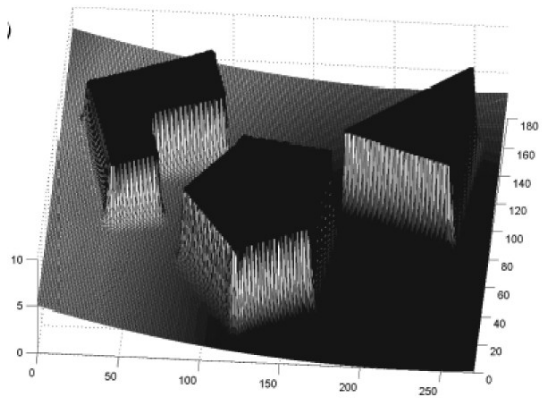
- Example repulsive force

$$U_{obst}(x) = \begin{cases} \frac{1}{\rho(x)} - \frac{1}{\rho_0(x)} & \rho(x) < \rho_0 \\ 0 & \rho(x) \geq \rho_0 \end{cases}$$

- In an implementation you need to use $\frac{1}{\rho(x)+\epsilon}$ where $\epsilon \ll 0$ avoids singularities when the distance $\rho(x)$ to an obstacle becomes 0.

Potential field example





Robot rolls like a ball towards the goal

Potential field characteristics

- Pros:
  - Easy to implement
- Cons:
  - Often leads to oscillative motion
  - Parameter tuning problems
  - Local minimas

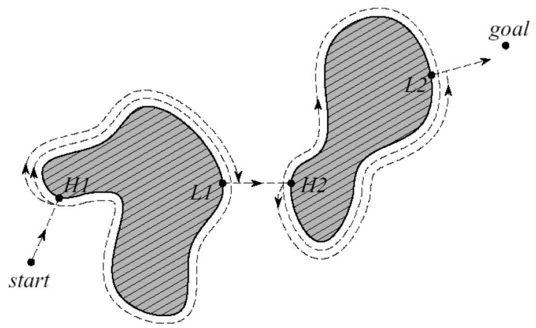
Obstacle avoidance

- In general the environment is not fully modelled
- The environment might be dynamically changing
- $\Rightarrow$ Need to be able modify the plan online
- Many algoritms from re-planning (of above algs) to reactive strategies

Obstacle avoidance

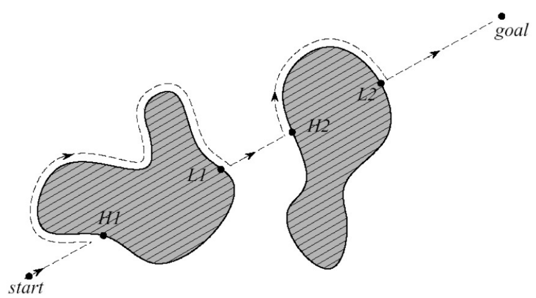
- Obstacle avoidance relies on
  - Information about the goal position
  - Local context information (where am I?)
  - Recent sensory information (a local map)

Bug algorithm 1



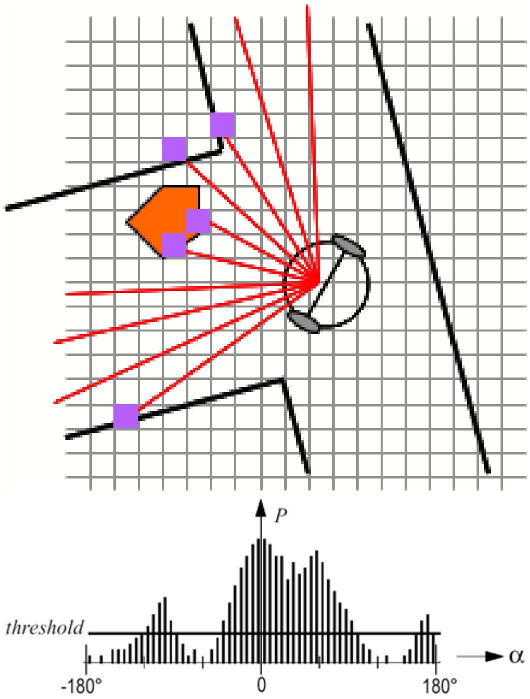
- The simplest possible stretegy
- Hit (H) and Leave (L)
- Do a full tour when hitting obstacle, leave at point with minimum distance to the goal
- Inefficient but does the job

Bug algorithm 2



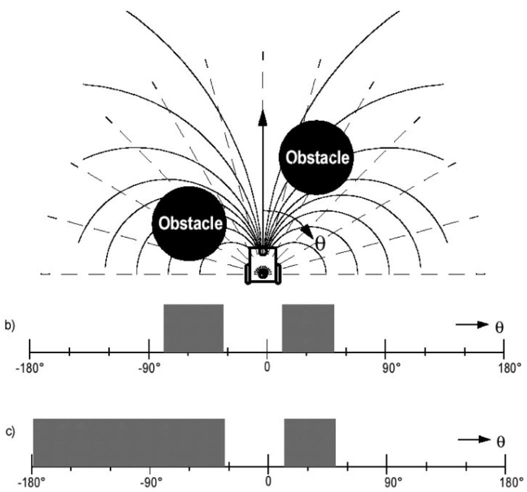
- Do a travesal but exit on line to target
- More effecient for open spaces
- Could be trapped in maze structures

Vector field histogram (VFH)



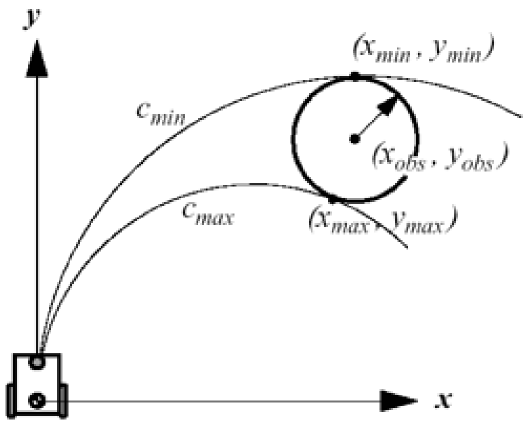
- Calculate "obstacle forces" similar to potential field
- Generate polar histogram by adding forces in sectors
- Threshold to get binary diagram
- Can find passable regions
- Select best direction

VFH cont'd



- Need to take kinematics into account
- Extensions account for dynamics (VFH+)
- Only feasible trajectories considered
- Not allowed directions are masked out in the histogram (c)

Curvature based methods



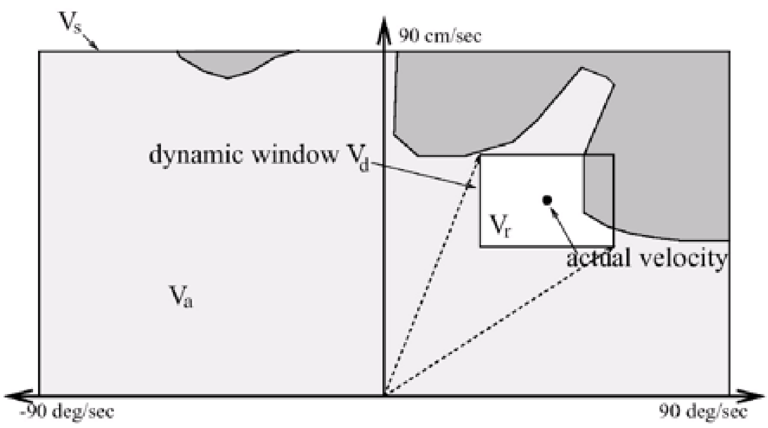
- Assume robot moves on arcs (e.g. true for differential drive robot)
- Curvature $c = \frac{\omega}{v}$
- Include kinematic and dynamic constraints

$$-v_{max} < v < v_{max}$$
$$-\omega_{max} < \omega < \omega_{max}$$

- Divide curvature space into passable regions
- Select best option to reach goal

Dynamic window approach (DWA)

- Consider velocity space (v, ω)
- Search among velocities in "reachable" window around current velocity given dynamic constraints
- Optimize over heading, speed and distance to obstacles to ensure fast and safe motion

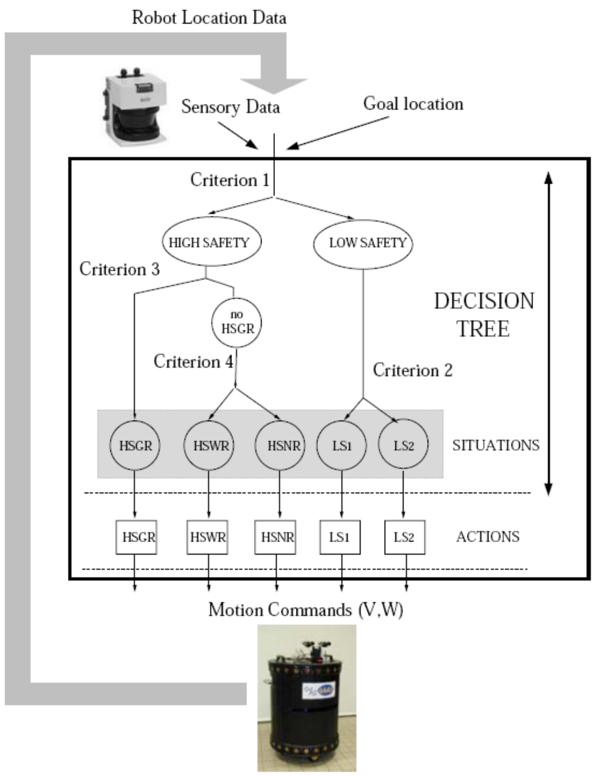


Example of Dynamic Window Approach

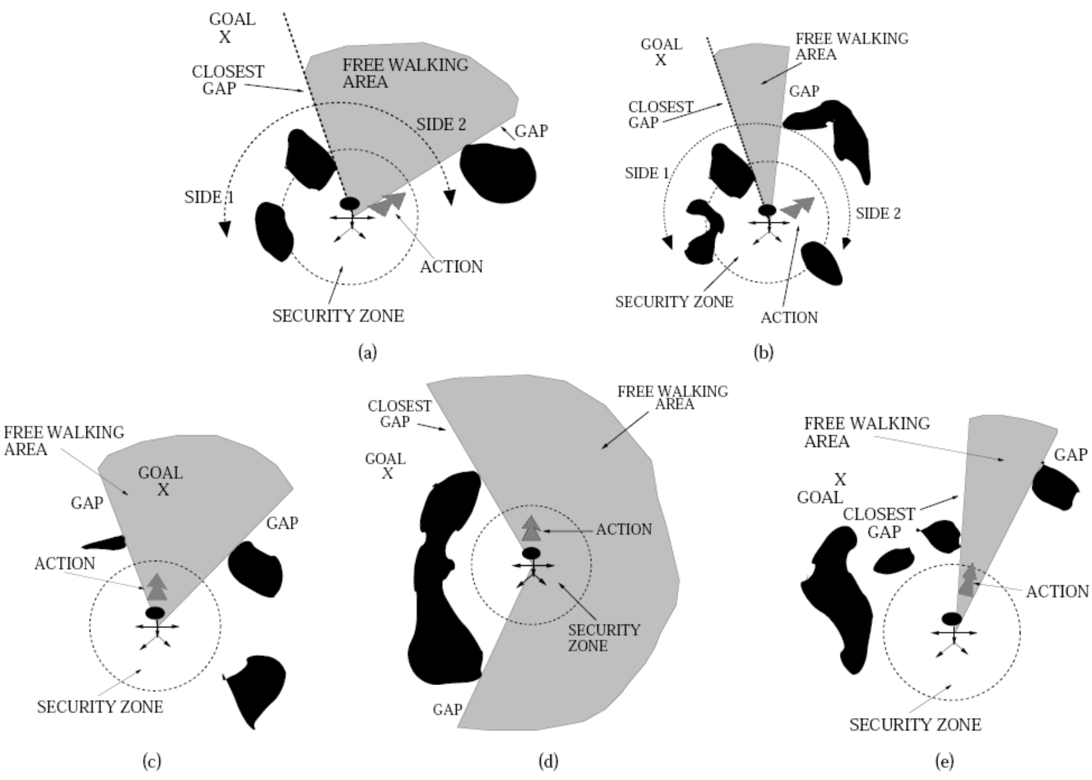
Nearness diagram (ND)

- Enumerate general obstacle situations
- Find good solution for each situation
- Switch between them

Nearness diagram



Nearness diagram

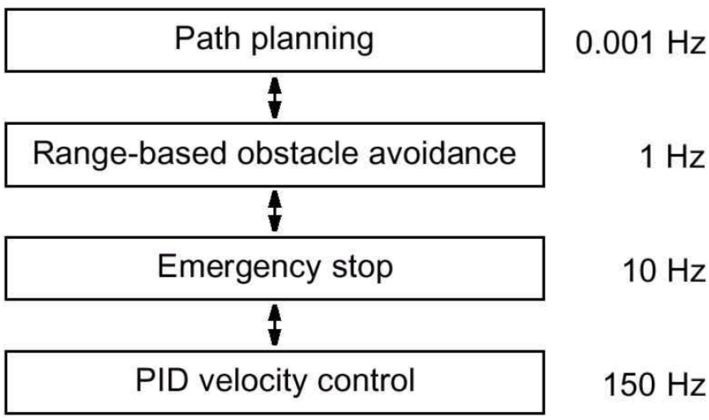


Combining local and global methods

- Neither global nor local methods are enough
- Global methods fail to address local variations, dynamics, etc
- Local methods get trapped in local minima
- $\Rightarrow$ combine the two
- Most of the above methods come in a "Global X" version such as the Global ND, Global DWA

Frequency range

- Large span in frequencies for navigation



- The robot architecture must support this large variation

Obstacle avoidance methods 1

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	

Obstacle avoidance methods 2

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	

Obstacle avoidance methods 3

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

Lectures are over...

but there is still stuff to do

- build your robots
- have fun at the competition (15/5)
- exam (22/5)
- write your reports