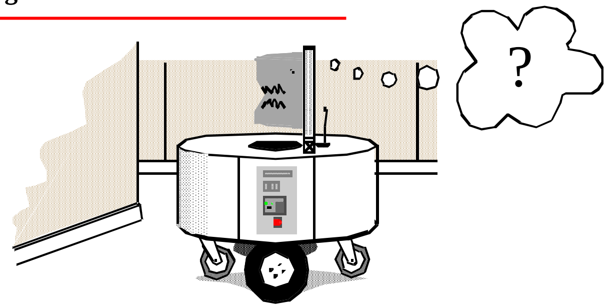
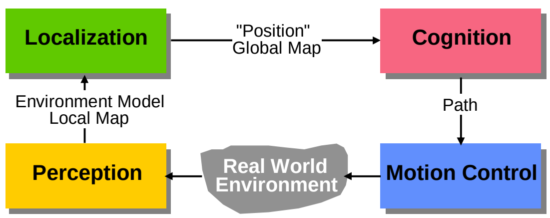
Planning and Navigation

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 exists (e.g. sales man problem).*

Competencies for Navigation II

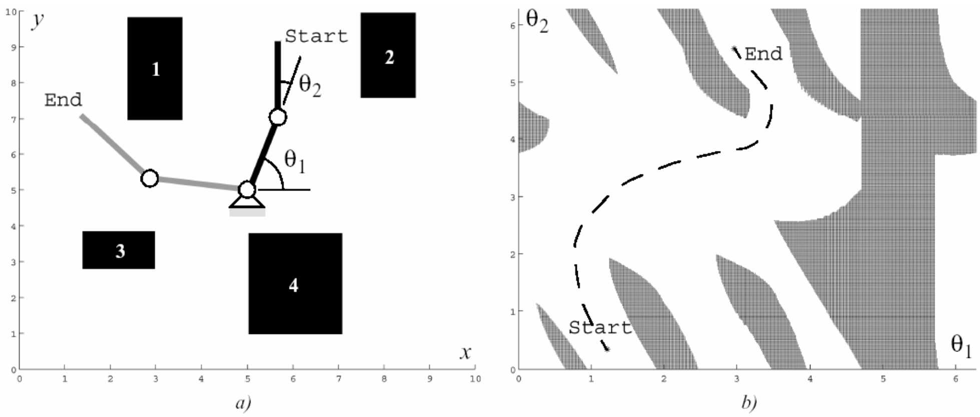
- However, in mobile robotics the knowledge of about the environment and situation is usually only partially known and is uncertain.
  - *makes the task much more difficult*
  - *requires multiple tasks running in parallel, some for planning (global), some to guarantee "survival of the robot".*
- Robot 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

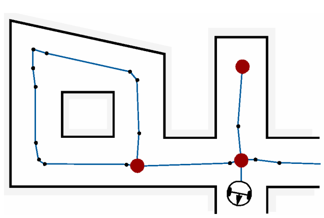
- State or configuration q can be described with k values q_i



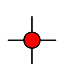
- What is the configuration space of a mobile robot?

Path Planning Overview

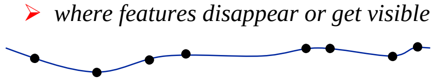
1. Road Map, Graph construction
   - *Identify a set of routes within the free space*



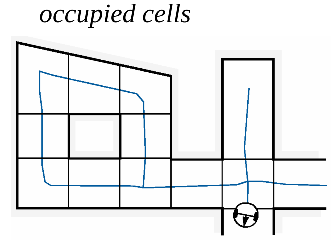
- Where to put the nodes?
- Topology-based:
  - *at distinctive locations*



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



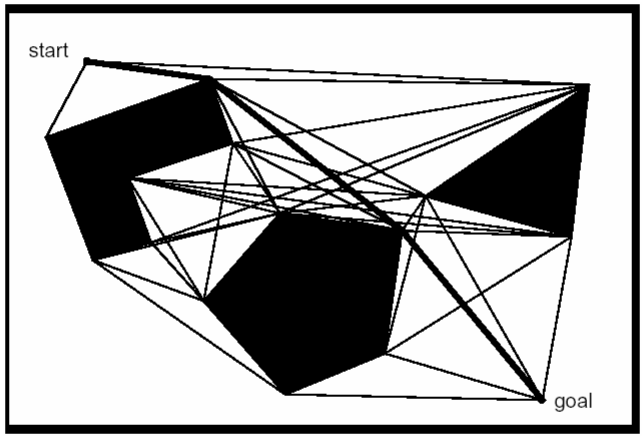
2. Cell decomposition
   - *Discriminate between free and occupied cells*



- 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*

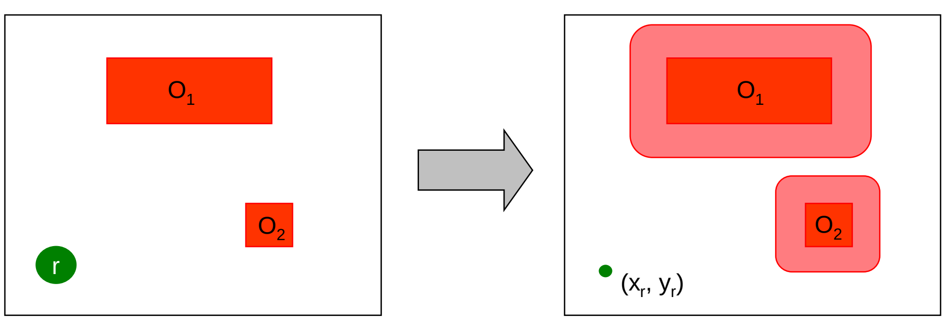
Road-Map Path Planning: **Visibility Graph**



- Shortest path length
- Grow obstacles to avoid collisions

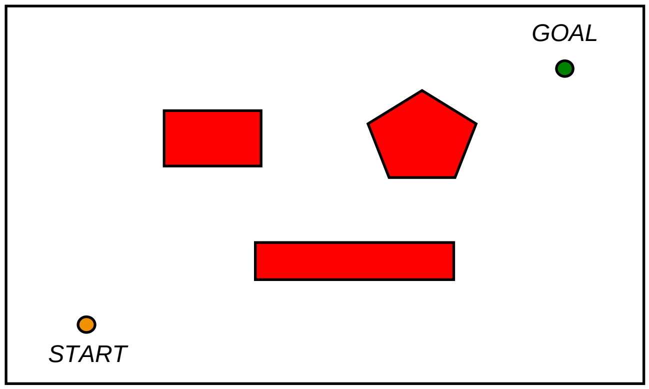
From Workspace to Configuration Space

- In order to model robots as *points*, the standard practice is to “grow” the obstacles by convolving them with the robot’s dimensions



Workspace (W) Configuration Space (C)

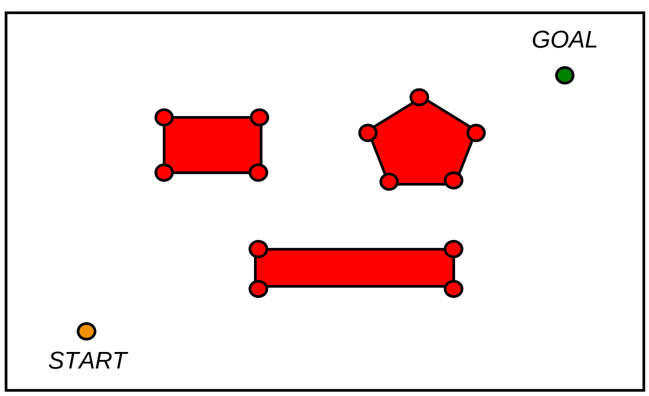
The Visibility Graph Method (1)



1) Model obstacles as polygons

Autonomous Mobile Robots, Chapter 6

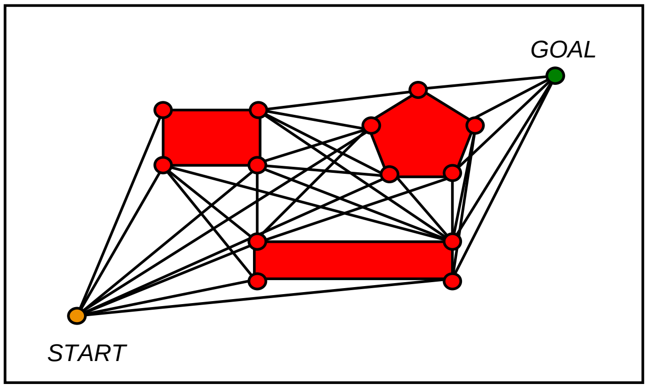
The Visibility Graph Method (2)



2) Construct a graph G(V,E). All polygon vertices are added to V, as are the start and goal positions.

Autonomous Mobile Robots, Chapter 6

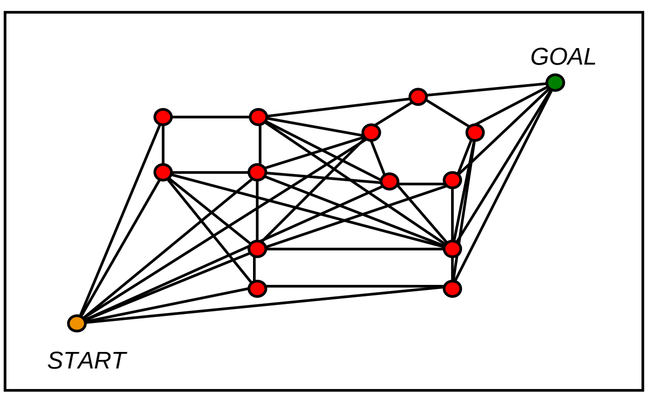
The Visibility Graph Method (3)



3) All vertices that are visible to one another are connected with an edge. These edges are added to E.

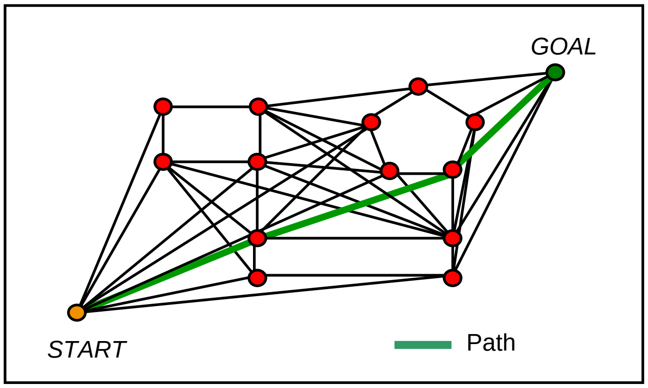
Autonomous Mobile Robots, Chapter 6

The Visibility Graph Method (4)



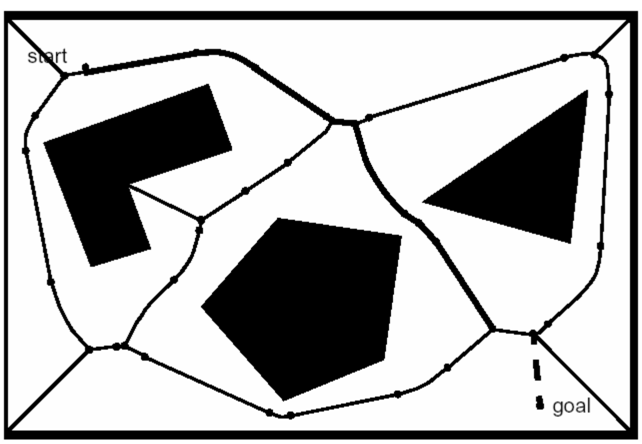
4) Polygon edges are also added to E. Then we only need to find the shortest path from the start vertex to the goal vertex in G. How can we find this???

The Visibility Graph Method (5)



We can find the shortest path using Dijkstra's Algorithm!!!

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



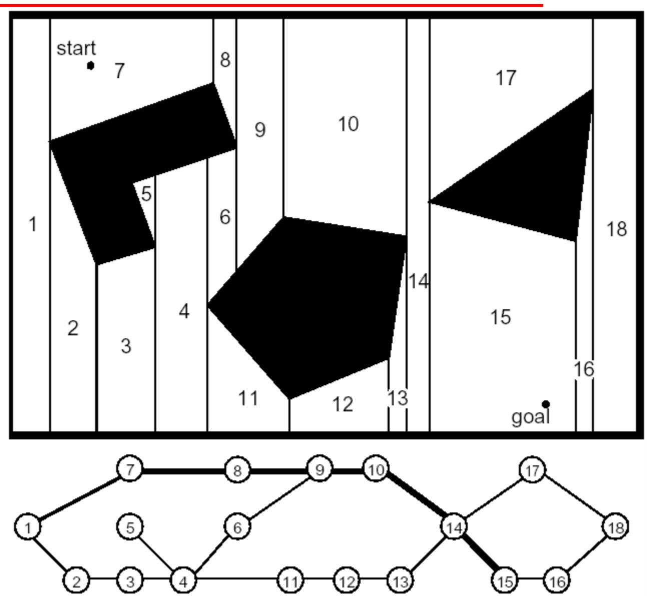
- Easy executable: Maximize the sensor readings
- Works also for map-building: Move on the Voronoi edges

Road-Map Path Planning: **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.*

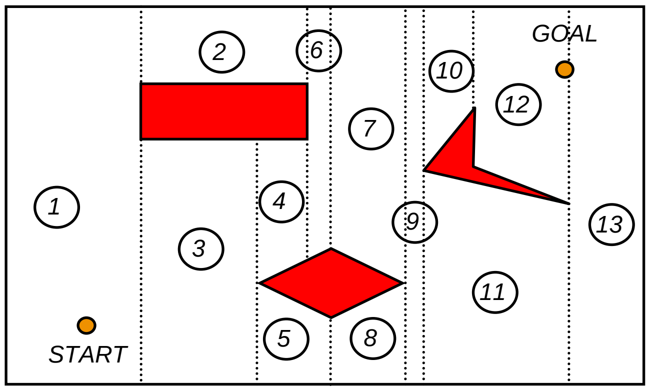
Autonomous Mobile Robots, Chapter 6
6.2.1

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



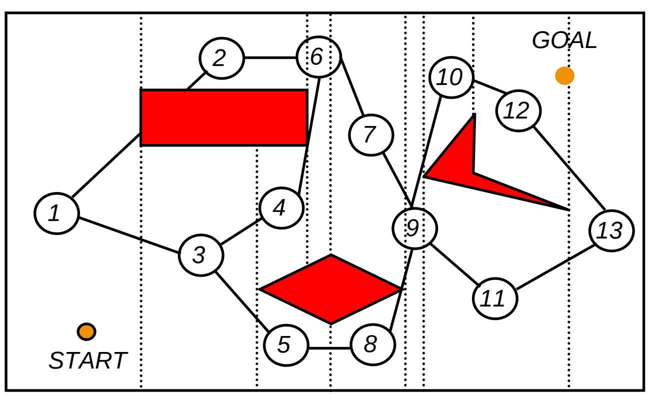
© R. Siegwart, I. Nourbakhsh

Exact Cell Decomposition Method (1)



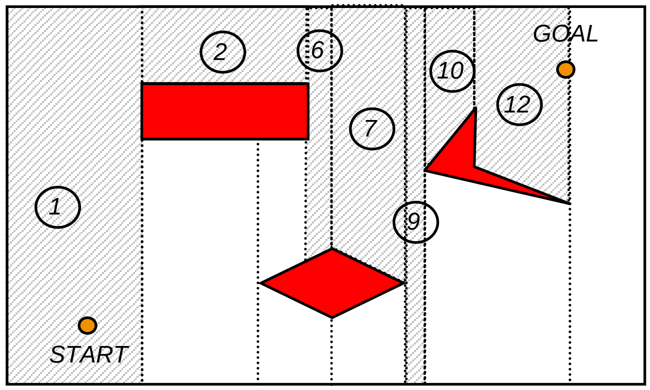
1) Decompose Region Into Cells

Exact Cell Decomposition Method (2)



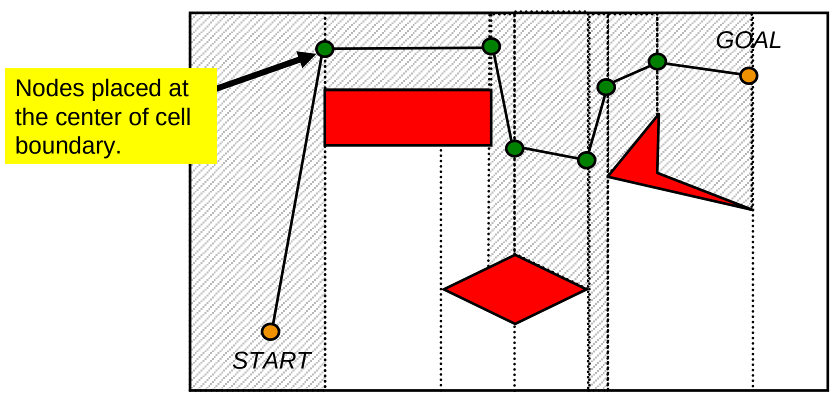
2) Construct Adjacency Graph

Exact Cell Decomposition Method (3)



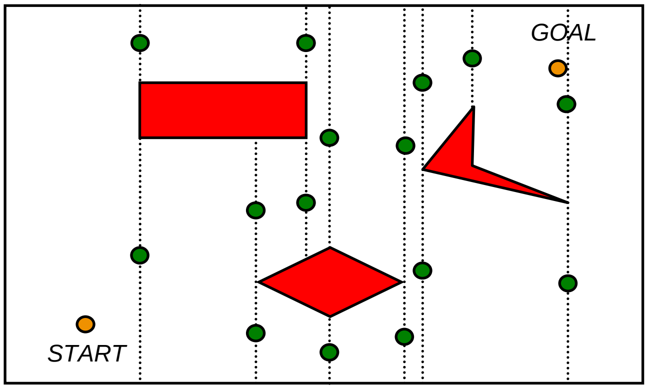
3) Construct Channel from shortest *cell* *path*
(via Depth-First-Search)

Exact Cell Decomposition Method (4)

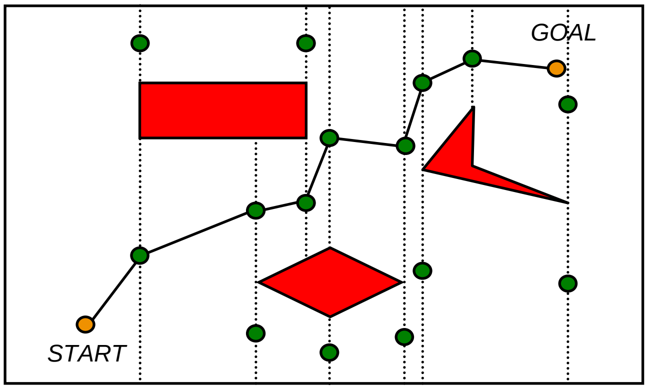


*4) Construct Motion Path **P** from channel cell borders*

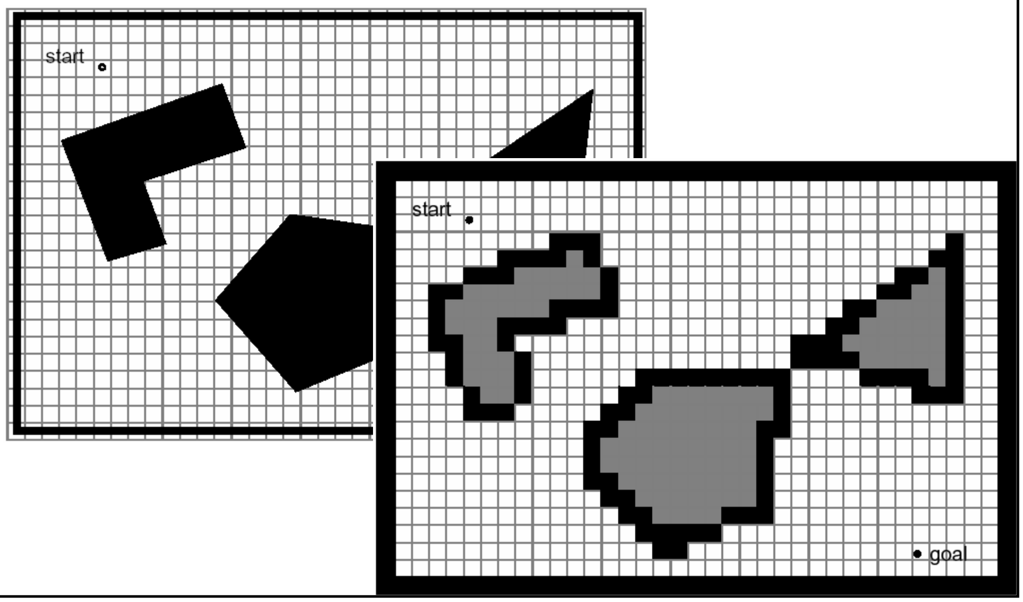
Exact Cell Decomposition with Euclidean Metric (1)



Exact Cell Decomposition with Euclidean Metric (2)



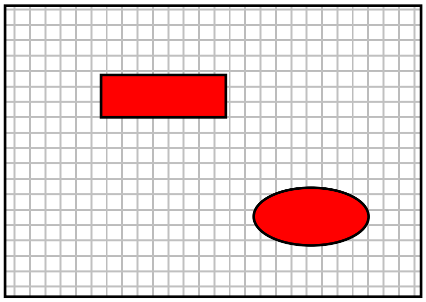
Road-Map Path Planning: **Approximate Cell Decomposition**



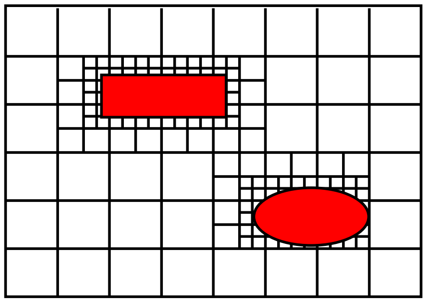
Approximate Cell Decomposition (1)

1. Perform cell tessellation of configuration space ***C***
   - *Uniform or quadtree*
2. Generate the cell graph *G(V,E)*
   - *Each cell* $v \in C_{free} \subseteq C$ *corresponds to a vertex in* **V**
   - *Two vertices* $v_i, v_j \in \mathbf{V}$ *are connected by an edge* e_{ij} *if they are adjacent (8-connected for exact)*
   - *Edges are weighted by Euclidean distance*
3. Find the shortest path from v_{init} to v_{goal}

Cell Decomposition (1)

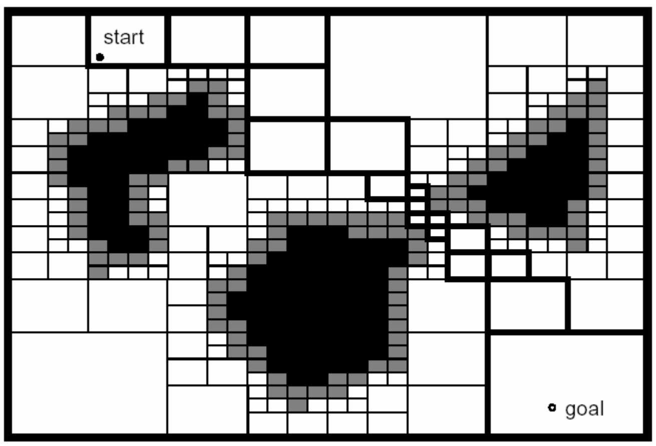


Uniform



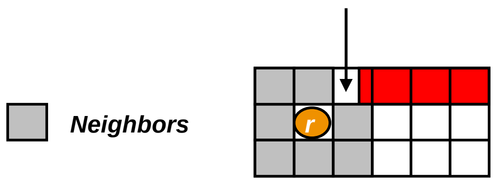
Quadtree

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



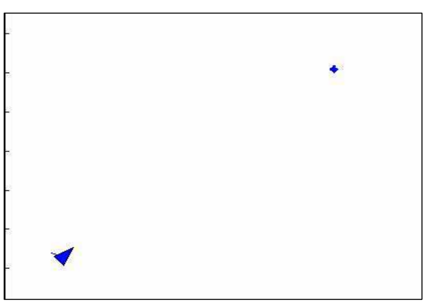
Cell Decomposition (2)

1. Continuity of trajectory a function of resolution
2. Computational complexity increases dramatically with resolution
3. Inexactness. Is this cell an obstacle or in C_{free}?

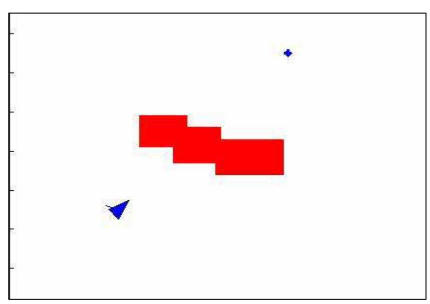


Autonomous Mobile Robots, Chapter 6

Cell Decomposition Simulations (1)

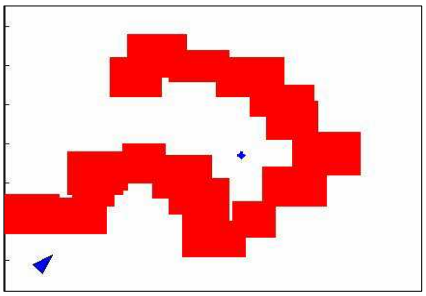


No Obstacles

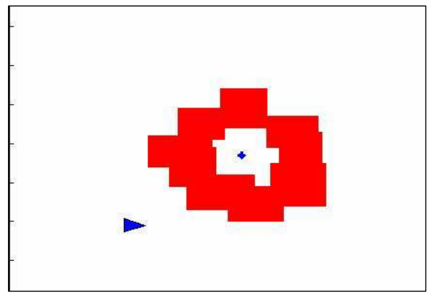


Single Obstacle

Cell Decomposition Simulations (2)



Multiple Obstacles



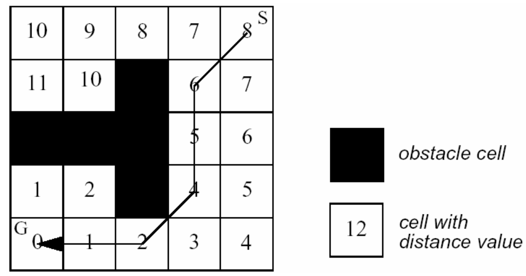
No Path

Approximate Cell Summary

- PROS
  - *Applicable to general obstacle geometries*
  - *Provides shorter paths than exact decomposition*
- CONS
  - *Performance a function of discretization resolution (δ)*
    - *Inefficiencies*
    - *Lost paths*
    - *Undetected collisions*
  - *Number of graph vertices $|V|$ grows with δ^2 and Dijkstra's runs in $O(|E| \lg |V|)$ (an A* implementation in $O(V^2)$)*

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

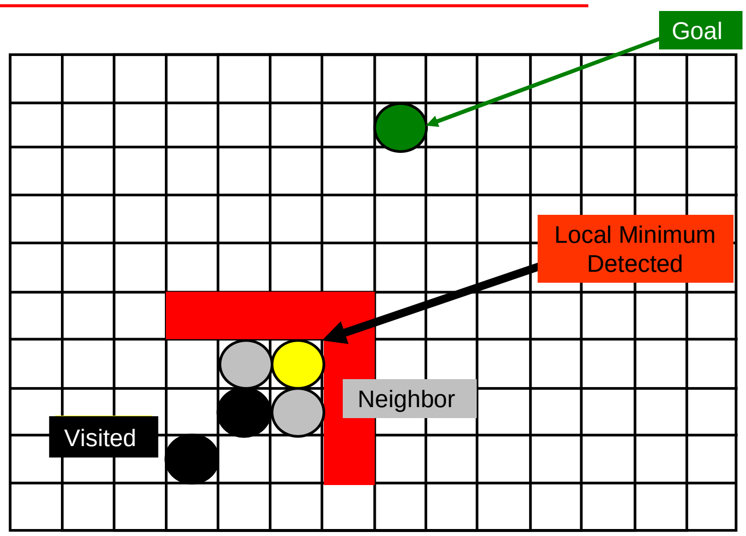
- Wavefront Expansion NF1 (see also later)
- Breadth-First Search
- Depth-First Search
- Greedy search and A*



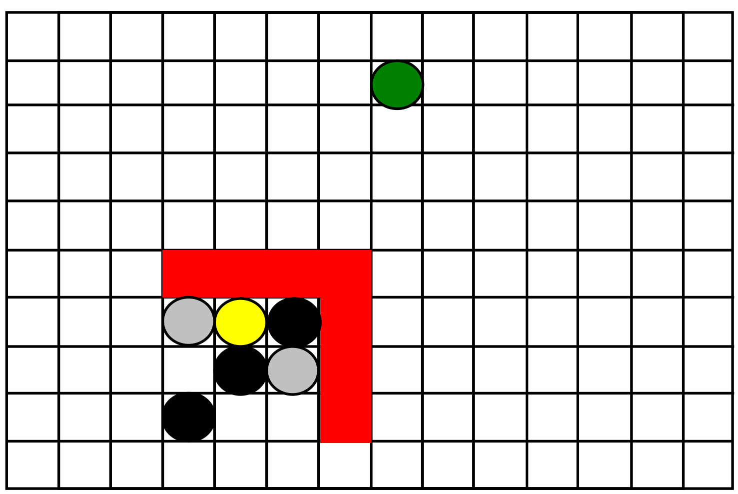
Best First Search

1. Workspace discretized into cells
2. Insert (x_{init},y_{init}) into list OPEN
3. Find all 4-way neighbors to (x_{init},y_{init}) that have *not been previously visited* and insert into OPEN
4. Sort neighbors by minimum potential
5. Form paths from neighbors to (x_{init},y_{init})
6. Delete (x_{init},y_{init}) from OPEN
7. (x_{init},y_{init}) = minPotential(OPEN)
8. GOTO 2 until (x_{init},y_{init})=goal (SUCCESS) or OPEN empty (FAILURE)

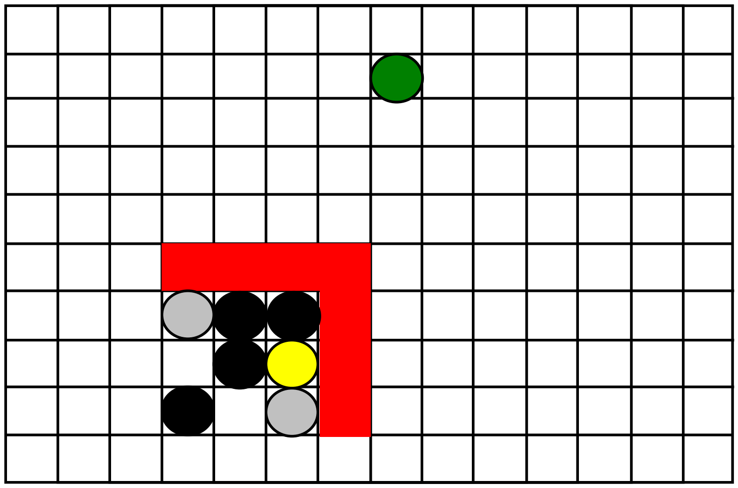
Best First Search Example (1)



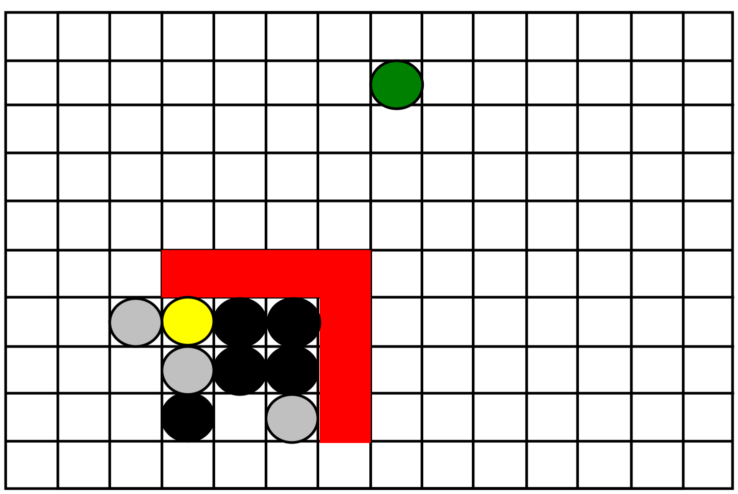
Best First Search Example (2)



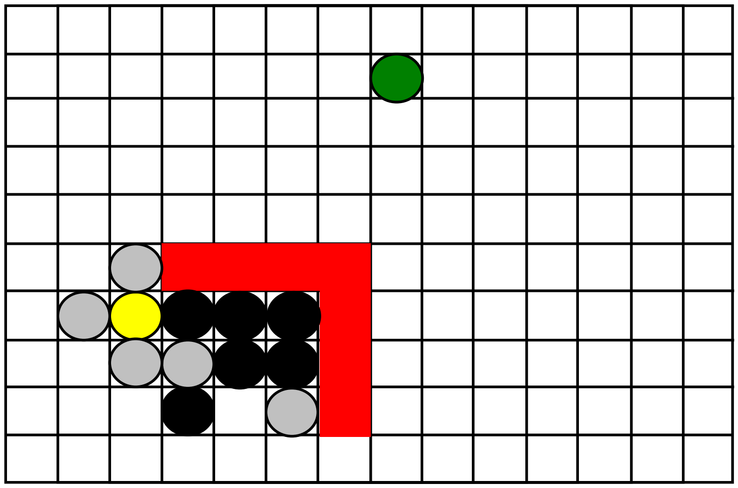
Best First Search Example (3)



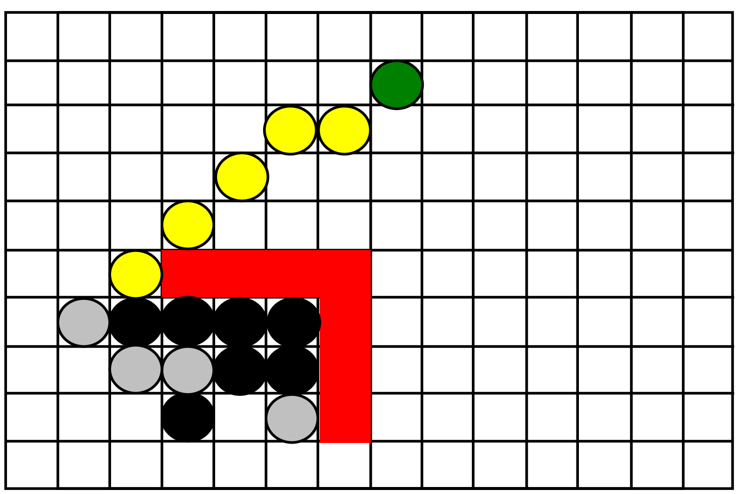
Best First Search Example (4)



Best First Search Example (5)



Best First Search Example (6)



Wavefront Propagation (1)

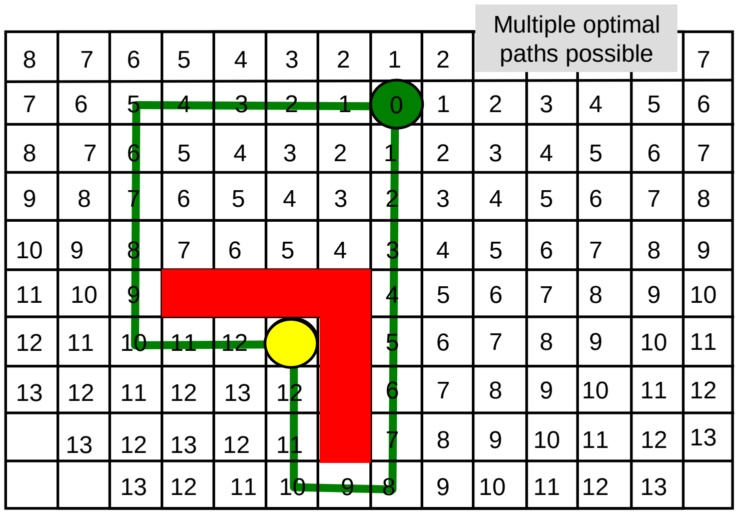
- Example of a discrete navigation function *E(x,y)*
- “Dynamic Programming” type approach
- Typically uses L_1 distance (aka Manhattan Distance) from the goal as a metric for function values

WeightNeighbors(Q, C, E)
$\vec{x}_w = \text{Head}(Q);$ // Removes head
Find all 4-way neighbors $N_w \in C$ to $\vec{x}_w$
$\forall \; x \in N_w$ if $E(x) \neq \infty$
  $E(x) = E(x_w) + 1;$
  Insert(x,Q)
end

Wavefront Propagation (2)

Wavefront Propagation$(\vec{x}_{goal}, \vec{x}_{robot}, C)$
 Discretize C into cells C_d;
 $E(x) = \infty \; \forall \; x \in C_d;$
 $E(\vec{x}_{goal}) = 0;$
 Insert$(\vec{x}_{goal}, Q);$
 while notEmpty(Q)
  WeightNeighbors(Q, C_d, E)
 end
 "Gradient" Descent from $\vec{x}_{robot}$ to $\vec{x}_{goal}$

Wavefront Propagation Example



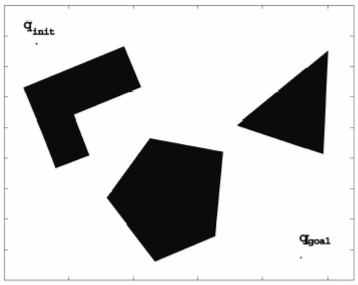
Autonomous Mobile Robots, Chapter 6

Potential Field Approaches

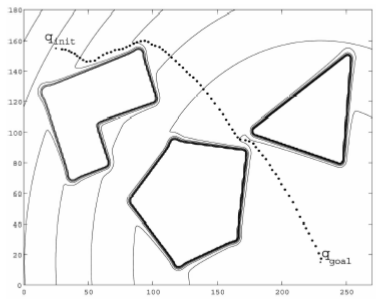
- Potential fields can live in continuous space
  - *No cell decomposition issues*
- The field is modeled by a *potential function* ***E(x,y)*** over our configuration space ***C***
- Local method
  - *Implicit trajectory generation*
  - *Prior knowledge of obstacle positions not required*
- The bad: Weaker performance guarantees

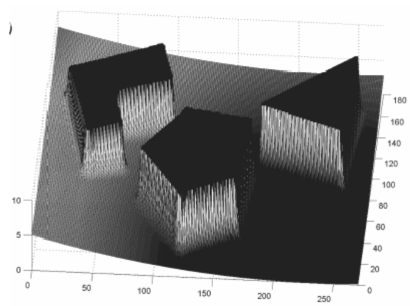
© JR Spletzer

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*





. Siegwart, I. Nourbakhsh

Autonomous Mobile Robots, Chapter 6

Flashback: What is the Gradient?

- In 2D, the gradient of a function *f* is defined as

$$\nabla f = \frac{\partial f}{\partial x}\hat{x} + \frac{\partial f}{\partial y}\hat{y}$$

- The gradient points in the direction where the derivative has the largest value (the greatest rate of increase in the value of *f*)
- The *gradient descent* optimization algorithm searches in the *opposite* direction of the gradient to find the *minimum* of a function
- Potential field methods employ a similar approach

Potential Fields for Motion Planning Using Gradient Descent

GradientDescent($x_{init}, x_{final}, -\nabla E$)

  while $x_{init} \neq x_{final}$

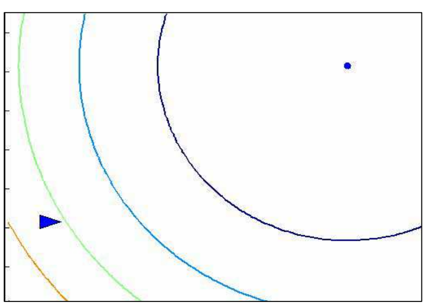
    $x_{init} = x_{init} - \delta \frac{\nabla E}{\|\nabla E\|}$; // Move δ in direction $-\nabla E$

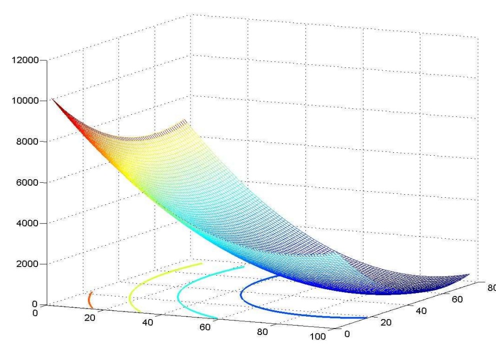
  end

Note that in practice, we will stop within some tolerance (like δ) of the final position to account for positional uncertainties, etc.

Also note that the step size δ must be small enough to ensure that we do not collide with an obstacle or overshoot our goal position.

Generating the Potential Field
A Parabolic Well for Attracting to Goal





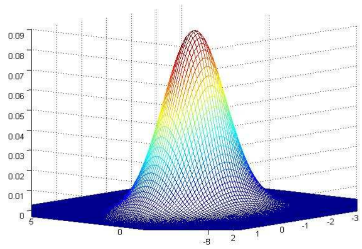
$$E(x) = \frac{1}{2}(x - x_{goal})^T (x - x_{goal})$$

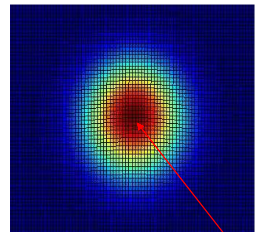
$$\nabla E(x) = x - x_{goal}$$

$$x = \begin{bmatrix} x_w \\ y_w \end{bmatrix}$$

NOTE: x is a vector corresponding to a position in the workspace

Gaussian Obstacle Potential Function





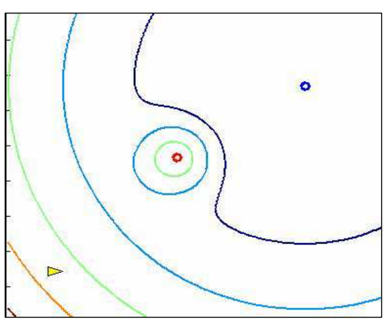
$$f(x)=\frac{1}{2\pi\sqrt{\sigma_x^2\sigma_y^2}}e^{-\left(\frac{(x-\mu_x)^2}{2\sigma_x^2}+\frac{(y-\mu_y)^2}{2\sigma_y^2}\right)}$$

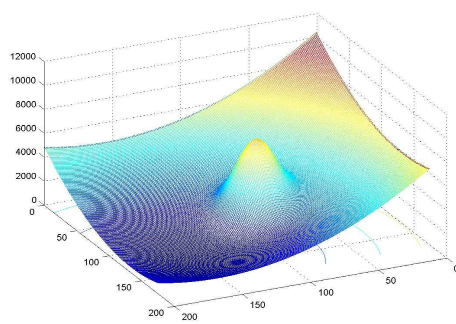
Gaussian centered at the obstacle coordinates

For a symmetric 2D Gaussians

$$f(x)=\frac{1}{2\pi\sigma^2}e^{-\frac{1}{2\sigma^2}\|\vec{x}\|^2}=\beta e^{-\frac{\gamma}{2}\|\vec{x}\|^2}$$

Parabolic Well for Goal
Exponential Source for Obstacle

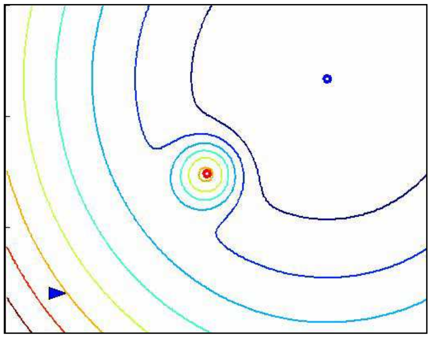


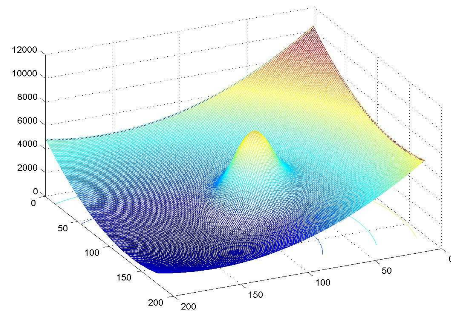


$$E(\vec{x})=\frac{1}{2}(\vec{x}-\vec{x}_{goal})^T(\vec{x}-\vec{x}_{goal})+\beta e^{\frac{-\gamma}{2}\|\vec{x}-\vec{x}_{obs}\|^2}$$

$$\nabla E(\vec{x})=\vec{x}-\vec{x}_{goal}-\gamma(\vec{x}-\vec{x}_{obs})\beta e^{\frac{-\gamma}{2}\|\vec{x}-\vec{x}_{obs}\|^2}$$

Parabolic Well /Exponential Source Unstable Equilibrium Example

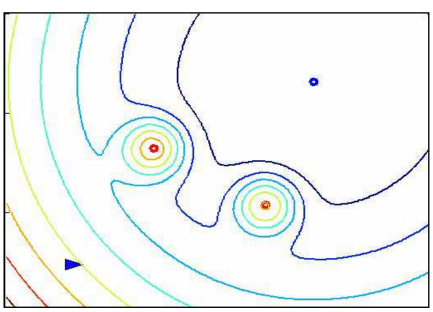


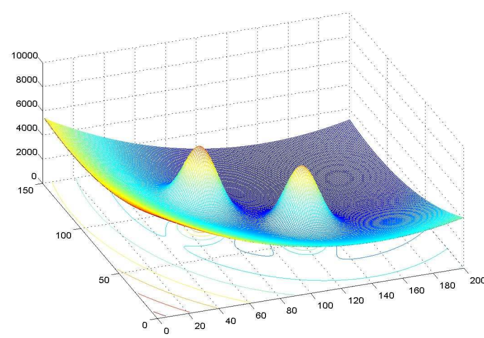


$$E(\vec{x}) = \frac{1}{2}(\vec{x}-\vec{x}_{goal})^T(\vec{x}-\vec{x}_{goal}) + \beta e^{\frac{-\gamma}{2}\left\|\vec{x}-\vec{x}_{obs}\right\|^2}$$

$$\nabla E(\vec{x}) = \vec{x}-\vec{x}_{goal} - \gamma(\vec{x}-\vec{x}_{obs})\beta e^{\frac{-\gamma}{2}\left\|\vec{x}-\vec{x}_{obs}\right\|^2}$$

Parabolic Well Goal & Two Exponential Source Obstacles (1)

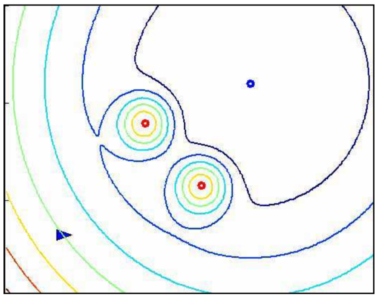


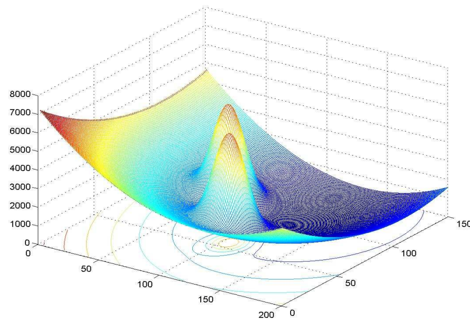


$$E(\vec{x}) = \frac{1}{2}(\vec{x}-\vec{x}_{goal})^T(\vec{x}-\vec{x}_{goal}) + \sum_{i=1}^{n}\beta_i e^{\frac{-\gamma_i}{2}\left\|\vec{x}-\vec{x}_{obs(i)}\right\|^2}$$

$$\nabla E(\vec{x}) = \vec{x}-\vec{x}_{goal} - \sum_{i=1}^{n}\gamma_i(\vec{x}-\vec{x}_{obs(i)})\beta_i e^{\frac{-\gamma_i}{2}\left\|\vec{x}-\vec{x}_{obs(i)}\right\|^2}$$

Parabolic Well Goal & Two Exponential Source Obstacles (2)

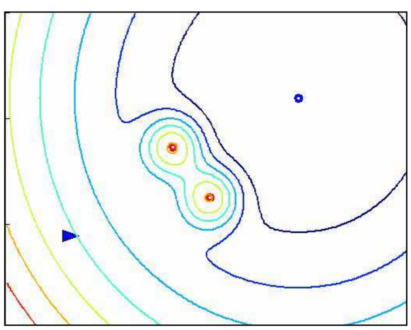


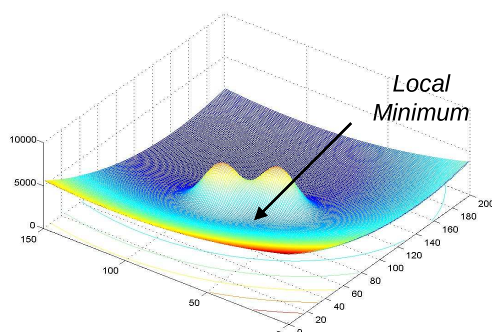


$$E(\vec{x}) = \frac{1}{2}(\vec{x}-\vec{x}_{goal})^T(\vec{x}-\vec{x}_{goal}) + \sum_{i=1}^{n} \beta_i e^{\frac{-\gamma_i}{2}\left\|\vec{x}-\vec{x}_{obs(i)}\right\|^2}$$

$$\nabla E(\vec{x}) = \vec{x}-\vec{x}_{goal} - \sum_{i=1}^{n} \gamma_i(\vec{x}-\vec{x}_{obs(i)})\beta_i e^{\frac{-\gamma_i}{2}\left\|\vec{x}-\vec{x}_{obs(i)}\right\|^2}$$

Parabolic Well Goal & Two Exponential Source Obstacles (3)



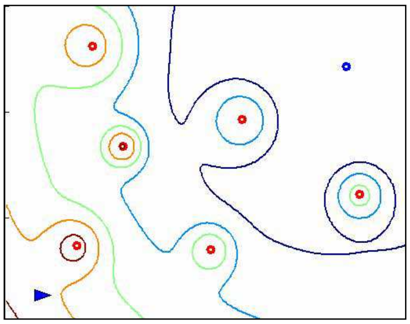


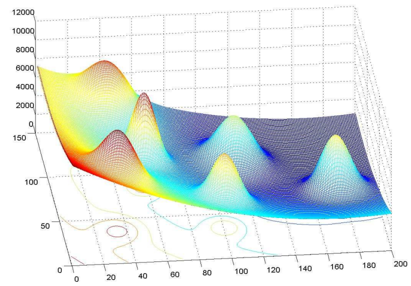
$$E(\vec{x}) = \frac{1}{2}(\vec{x}-\vec{x}_{goal})^T(\vec{x}-\vec{x}_{goal}) + \sum_{i=1}^{n} \beta_i e^{\frac{-\gamma_i}{2}\left\|\vec{x}-\vec{x}_{obs(i)}\right\|^2}$$

$$\nabla E(\vec{x}) = \vec{x}-\vec{x}_{goal} - \sum_{i=1}^{n} \gamma_i(\vec{x}-\vec{x}_{obs(i)})\beta_i e^{\frac{-\gamma_i}{2}\left\|\vec{x}-\vec{x}_{obs(i)}\right\|^2}$$

Autonomous Mobile Robots, Chapter 6

Parabolic Well Goal & Multiple Exponential Source Obstacles



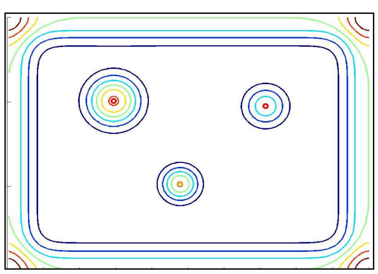


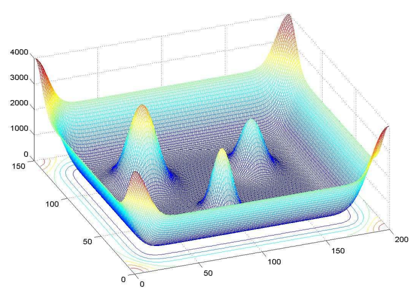
$$E(\vec{x}) = \frac{1}{2}(\vec{x}-\vec{x}_{goal})^T(\vec{x}-\vec{x}_{goal}) + \sum_{i=1}^{n} \beta_i e^{\frac{-\gamma_i}{2}\left\|\vec{x}-\vec{x}_{obs(i)}\right\|^2}$$

$$\nabla E(\vec{x}) = \vec{x}-\vec{x}_{goal} - \sum_{i=1}^{n} \gamma_i(\vec{x}-\vec{x}_{obs(i)})\beta_i e^{\frac{-\gamma_i}{2}\left\|\vec{x}-\vec{x}_{obs(i)}\right\|^2}$$

Autonomous Mobile Robots, Chapter 6

Modeling Walls in a Closed Workspace



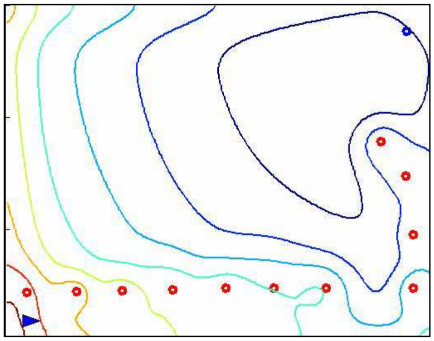


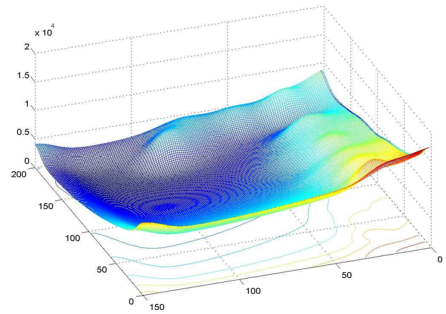
NOTATION WARNING: (x & y are scalars on RHS)

$$E_{wall}(\vec{x}) = \alpha e^{\frac{-\gamma}{2}(y-YMAX)^2} + \alpha e^{\frac{-\gamma}{2}y^2} + \alpha e^{\frac{-\gamma}{2}(x-XMAX)^2} + \alpha e^{\frac{-\gamma}{2}x^2}$$

$$\nabla E(\vec{x}) = -\gamma\alpha\left[(x-XMAX)e^{\frac{-\gamma}{2}(x-XMAX)^2} + xe^{\frac{-\gamma}{2}x^2}\right]i - \gamma\alpha\left[(y-YMAX)e^{\frac{-\gamma}{2}(y-YMAX)^2} + ye^{\frac{-\gamma}{2}y^2}\right]j$$

A Summary Example for Potential Fields
Goal, Obstacles, Walls





Issues with
Reactive Approaches Presented

- Obstacles are not points
  - *Model as points*
  - *Bound with ellipses, polygons (one or more obstacles)*
- Local minima proliferate with multiple obstacles, and failure to achieve goal often results
- Our choice of potential functions for the goal, obstacles, etc., was somewhat arbitrary. There are many others (linear, trapezoidal, cones, etc.)
- Is there a smarter choice of potential functions that eliminates the local minima?
- No ☹ (Koditschek 1987)

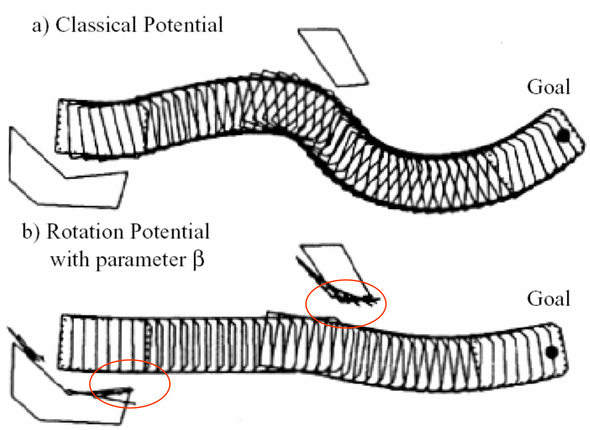
What to do when a Local Minimum is Detected?

- Combine with global/discrete approaches
  - *Best first search*
  - *Wavefront propagation*
  - *Voronoi*

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*



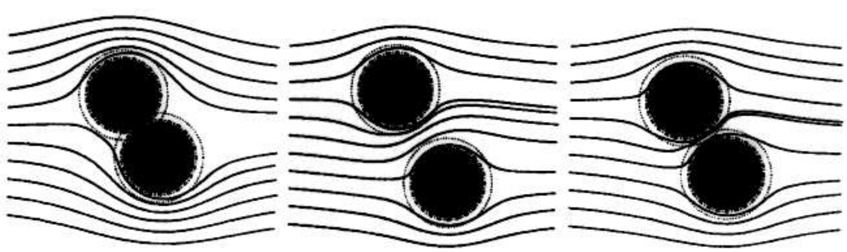
Potential Field Path Planning: **Potential Field using a Dyn. Model**

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**

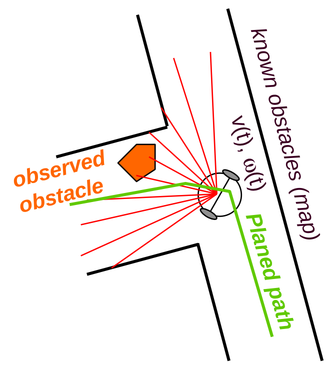
- Hydrodynamics analogy
  - *robot is moving similar to a fluid particle following its stream*
- Ensures that there are no local minima



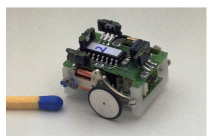
- Note:
  - *Complicated, only simulation shown*

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*

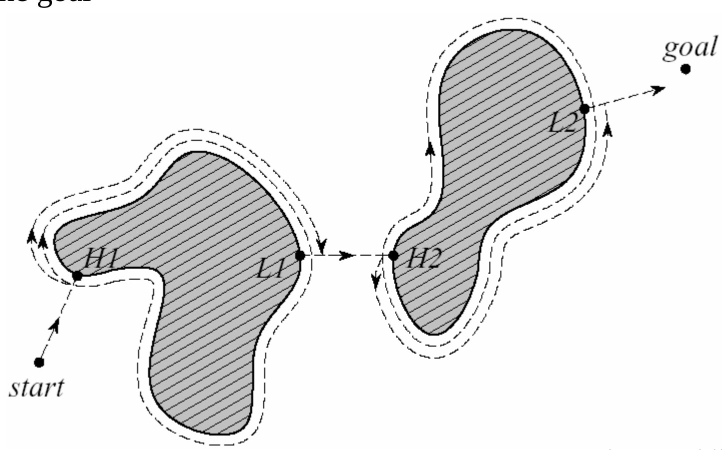


- Example: Alice



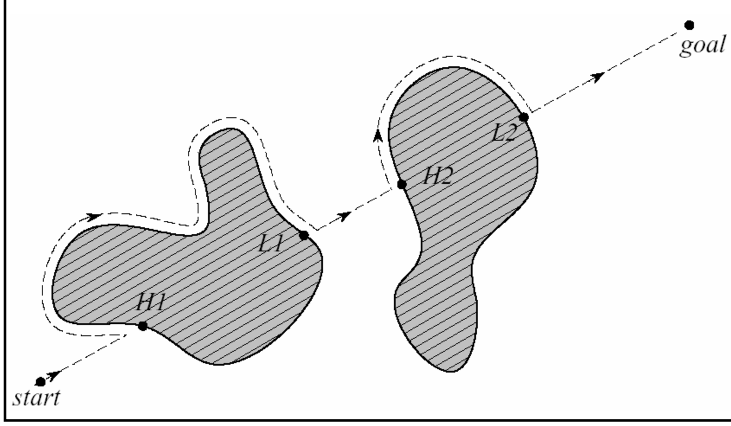
Obstacle Avoidance: **Bug1**

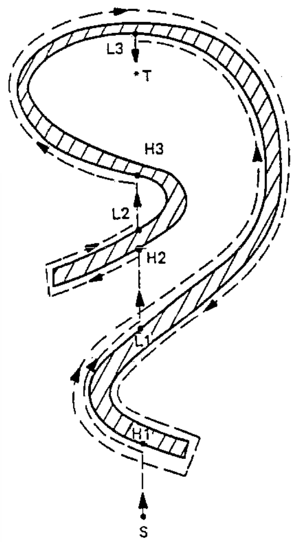
- Following along the obstacle to avoid it
- Each encountered obstacle is once fully circled before it is left at the point closest to the goal



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*



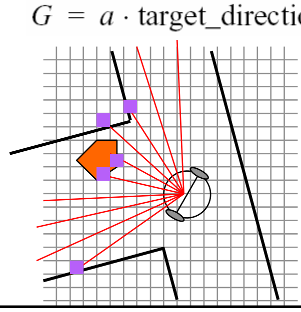


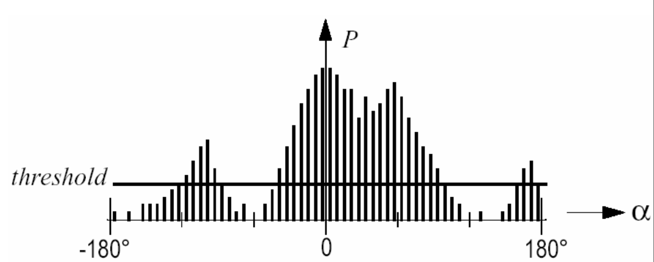
Obstacle Avoidance: **Vector Field Histogram (VFH)**

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*

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

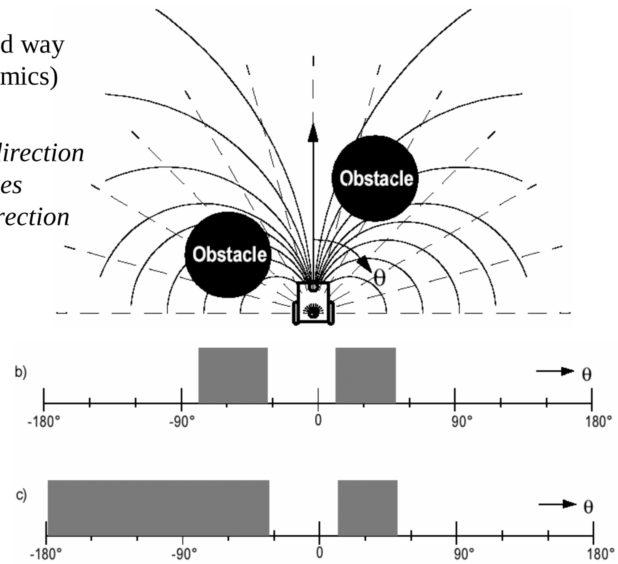




Obstacle Avoidance: **Vector Field Histogram + (VFH+)**

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**

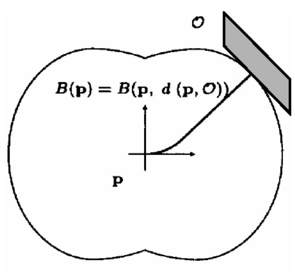
Borenstein et al.

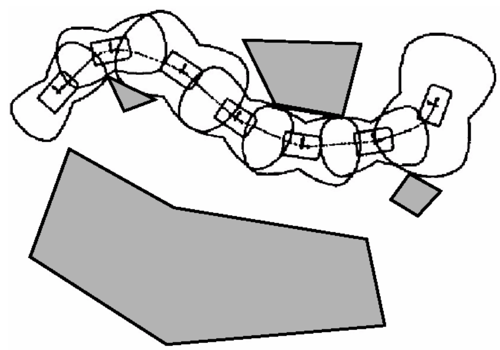
- Notes:
  - *Limitation if narrow areas (e.g. doors) have to be passed*
  - *Local minimum might not be avoided*
  - *Reaching of the goal can not be guaranteed*
  - *Dynamics of the robot not really considered*

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*

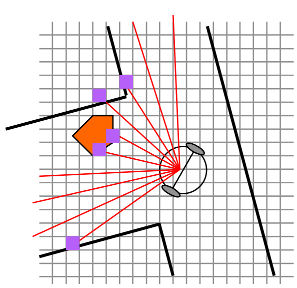


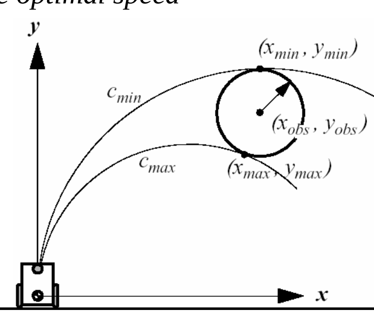


Obstacle Avoidance: Basic **Curvature Velocity Methods** (CVM)

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 ($c = \omega / v$)*
  - *Acceleration constraints:*
  - *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*

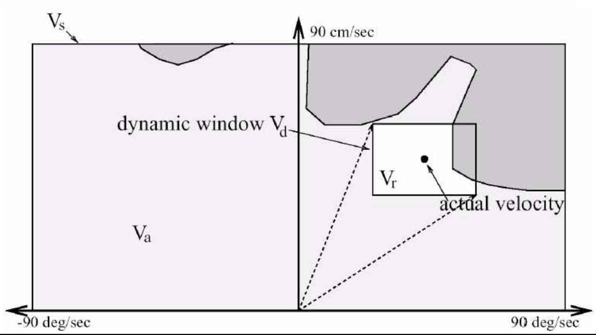
- Note:
  - *Better performance to pass narrow areas (e.g. doors)*
  - *Problem with local minima persists*

Obstacle Avoidance: Dynamic Window Approach

Fox and Burgard, Brock and Khatib

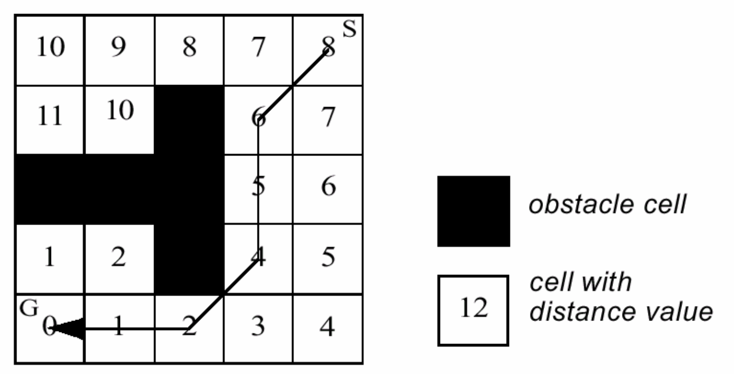
- The kinematics of the robot is considered by searching a well chosen velocity space
  - *velocity space -> some sort of configuration space*
  - *robot is assumed to move on arcs*
  - *ensures that the robot comes to stop before hitting an obstacle*
  - *objective function is chosen to select the optimal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

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



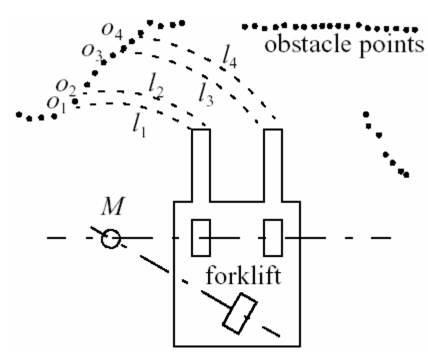
Obstacle Avoidance: Global **Dynamic Window Approach**

- 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*

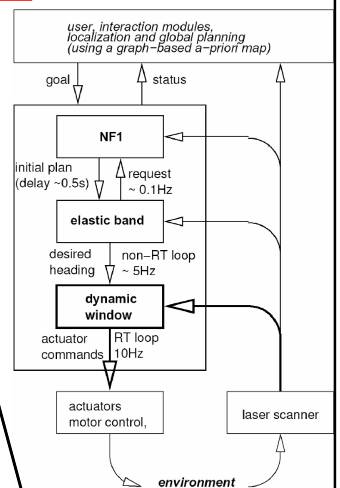


Obstacle Avoidance: **The EPFL-ASL approach**

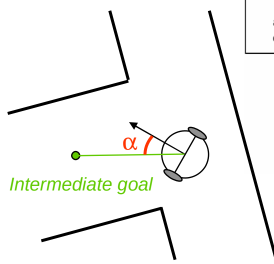
- Dynamic window approach with global path planing
  - *Global path generated in advance*
  - *Path adapted if obstacles are encountered*
  - *dynamic window considering also the shape of the robot*
  - *real-time because only max speed is calculated*
- Selection (Objective) Function:

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

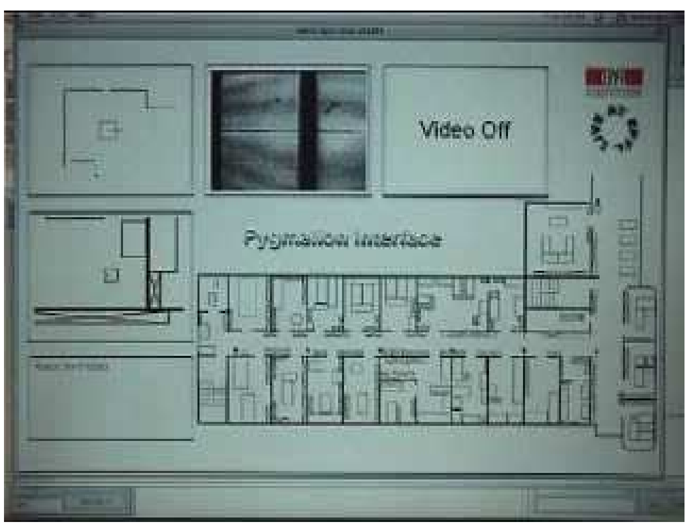
  - $speed = v / v_{max}$
  - $dist = L / L_{max}$
  - $goal\_heading = 1 - (\alpha - \omega T) / \pi$



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



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



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]
model fidelity	shape	point	point	point	C-space	C-space	circle	circle	simplistic
	kinematics				exact		basic	basic	
	dynamics						simplistic	simplistic	
view		local	local	local	local	global	essentially local	local	local
other requisites	local map	local tangent graph					histogram grid	histogram grid	histogram grid
	global map				polygonal	polygonal			
	path planner				required	required			
sensors		range	tactile	tactile			sonars	sonars	range
tested robots					various	various	nonholonomic (GuideCane)	nonholonomic (GuideCane)	synchro-drive (hexagonal)
performance	cycle time						6 ... 242 ms	6 ms	27 ms
	architecture						66 MHz, 486 PC	66 MHz, 486 PC	20 MHz, 386 AT
remarks		efficient in many cases, robust	inefficient, robust	very inefficient, robust			fewer local minima	local minima	local minima, oscillating trajectories

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	

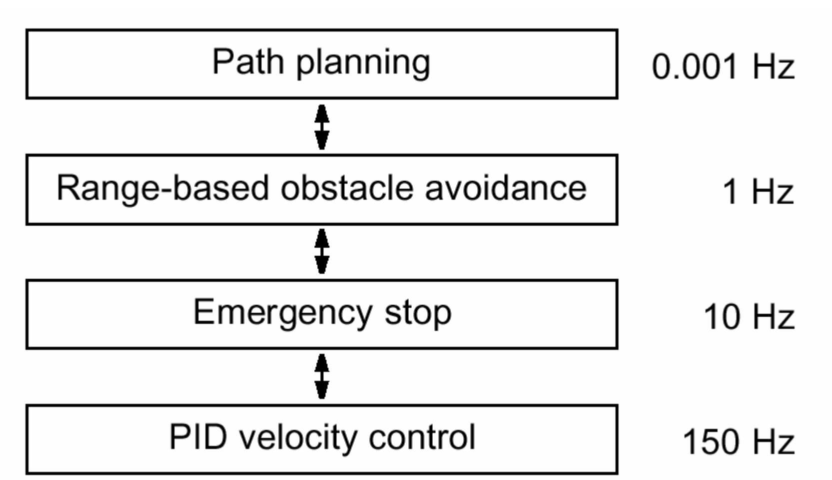
Autonomous Mobile Robots, Chapter 6
6.2.2

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

Autonomous Mobile Robots, Chapter 6
6.3.3

4-level temporal decomposition



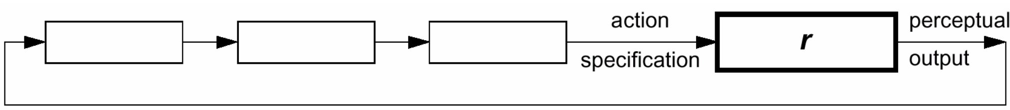
© R. Siegwart, I. Nourbakhsh

Autonomous Mobile Robots, Chapter 6

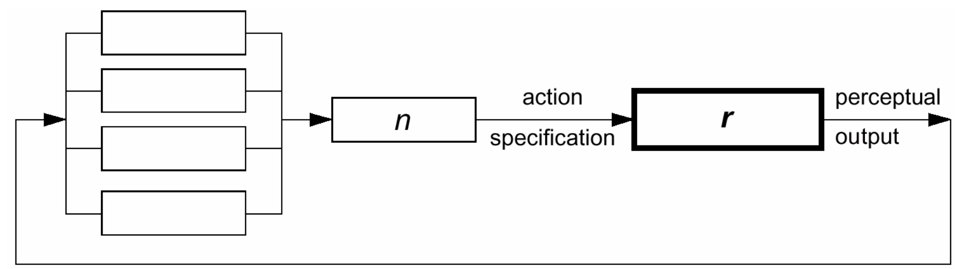
6.3.3

Control decomposition

- Pure serial decomposition



- Pure parallel decomposition

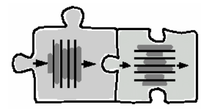


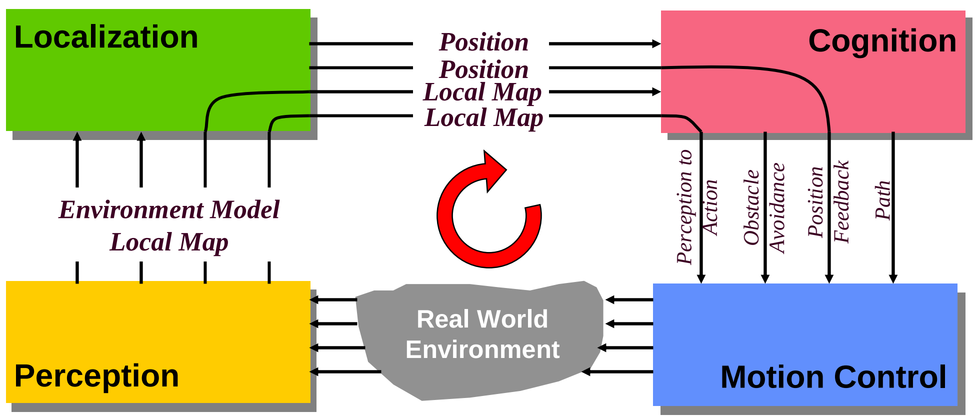
Autonomous Mobile Robots, Chapter 6

6.3.4

Sample Environment

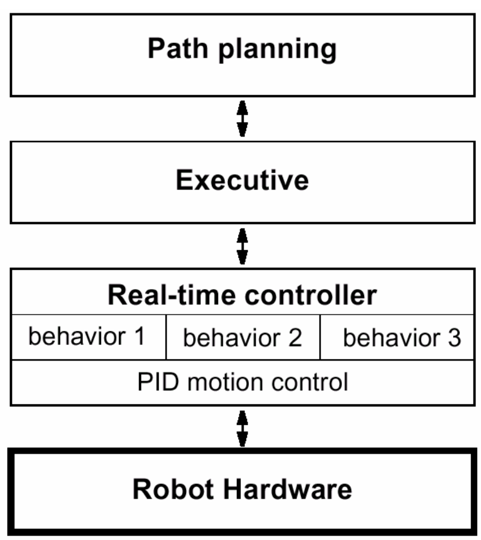
Our basic architectural example



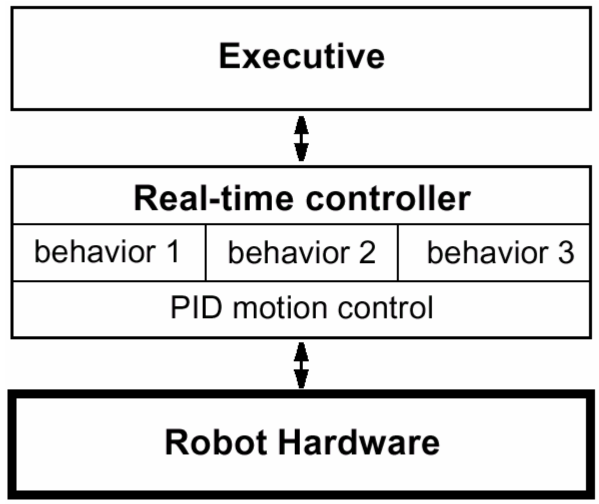


General Tiered Architecture

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

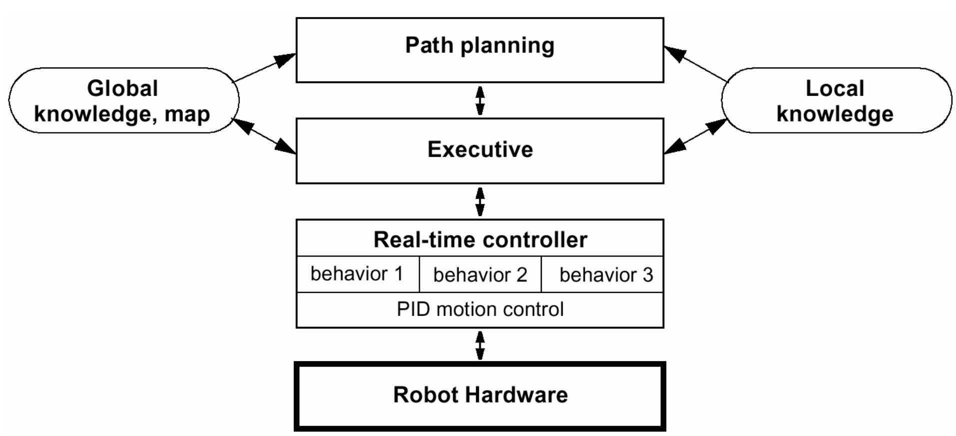


A Tow-Tiered Architecture for Off-Line Planning



Autonomous Mobile Robots, Chapter 6
6.3.4

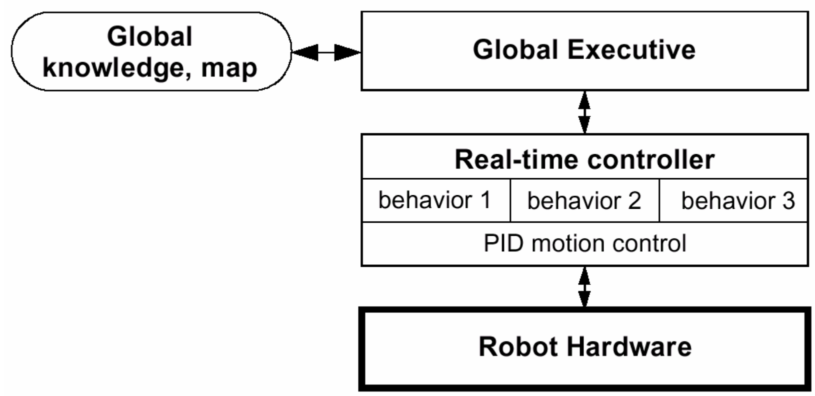
A Three-Tiered Episodic Planning Architecture.



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

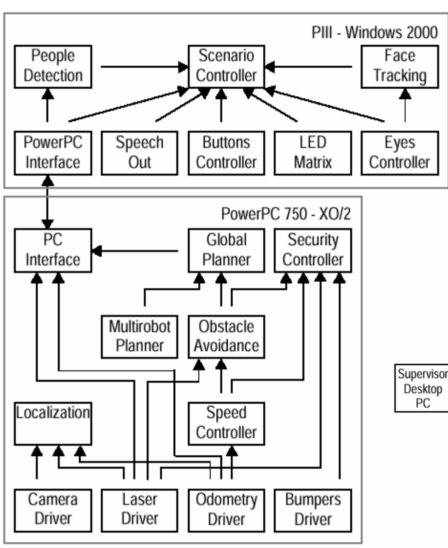
© R. Siegwart, I. Nourbakhsh

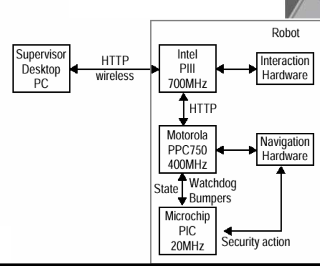
An integrated planning and execution architecture

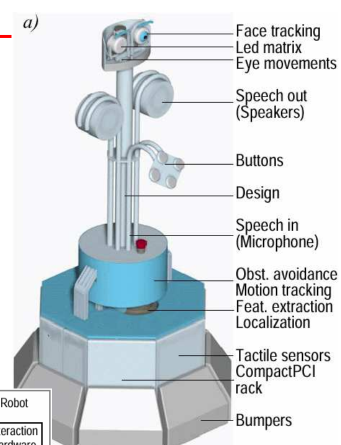


- All integrated, no temporal between planner and executive layer

Example: The RoboX Architecture







Example: RoboX @ EXPO.02