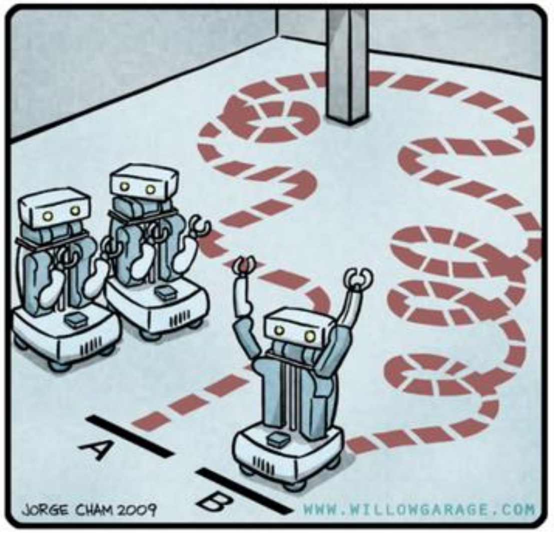
POLITECNICO DI MILANO

Robot Motion Control

Matteo Matteucci – matteo.matteucci@polimi.it

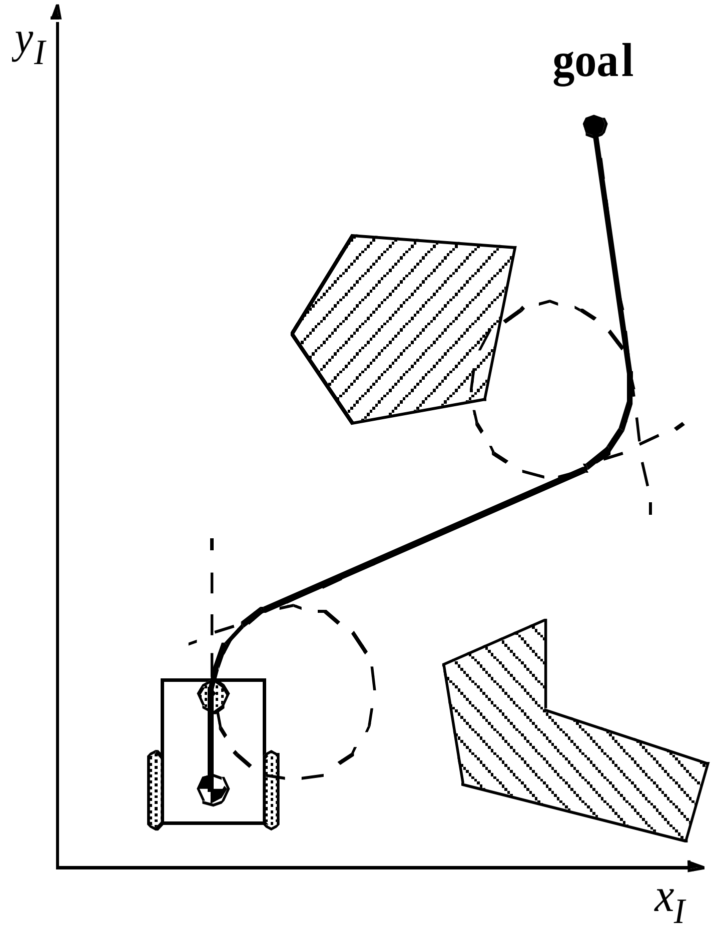


"HIS PATH-PLANNING MAY BE SUB-OPTIMAL, BUT IT'S GOT FLAIR."

Open loop control

A mobile robot is meant to move from one place to another

- Pre-compute a smooth trajectory based on motion segments (e.g., line and circle segments) from start to goal
- Execute the planned trajectory along the way till the goal



Disadvantages:

- Not an easy task to pre-compute a feasible trajectory
- Limitations and constraints of the robots velocities and accelerations
- Does not handle dynamical changes in the environment
- Resulting trajectories are usually not smooth

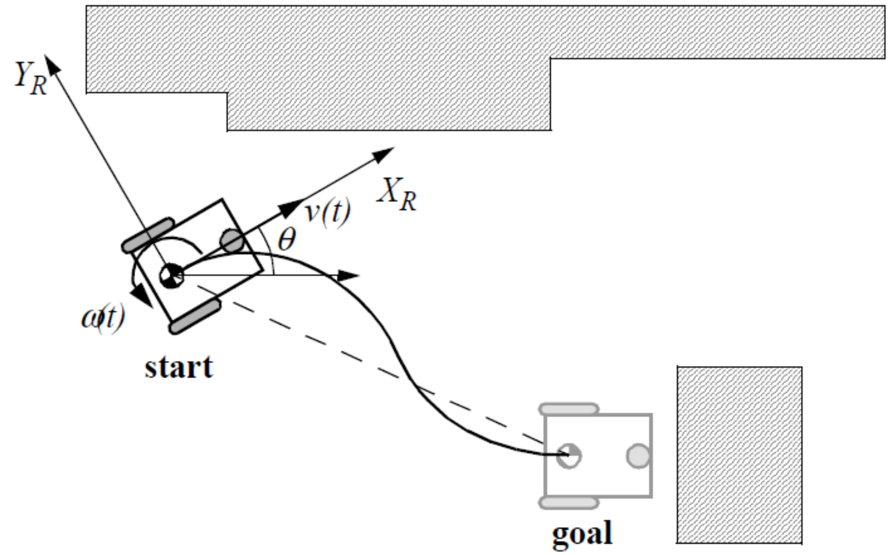
Matteo Matteucci – matteo.matteucci@polimi.it

Feedback control (diff drive example)

With feedback control the trajectory is recomputed and adapted online

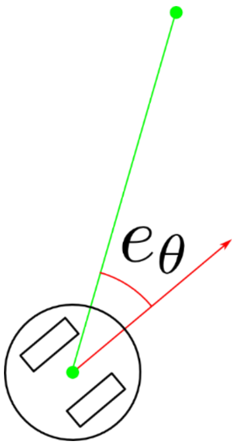
We can design a simple control schema for path following:

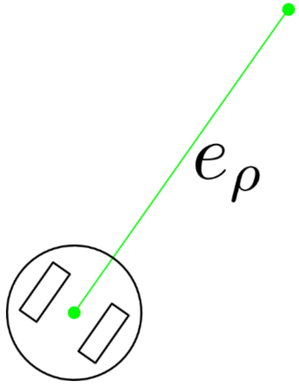
- First we close a speed control loop on the wheels
- Then divide the problem in:
  - Control of the orientation
  - Control of the distance



Control orientation acting on angular velocity

Control distance acting on linear velocity





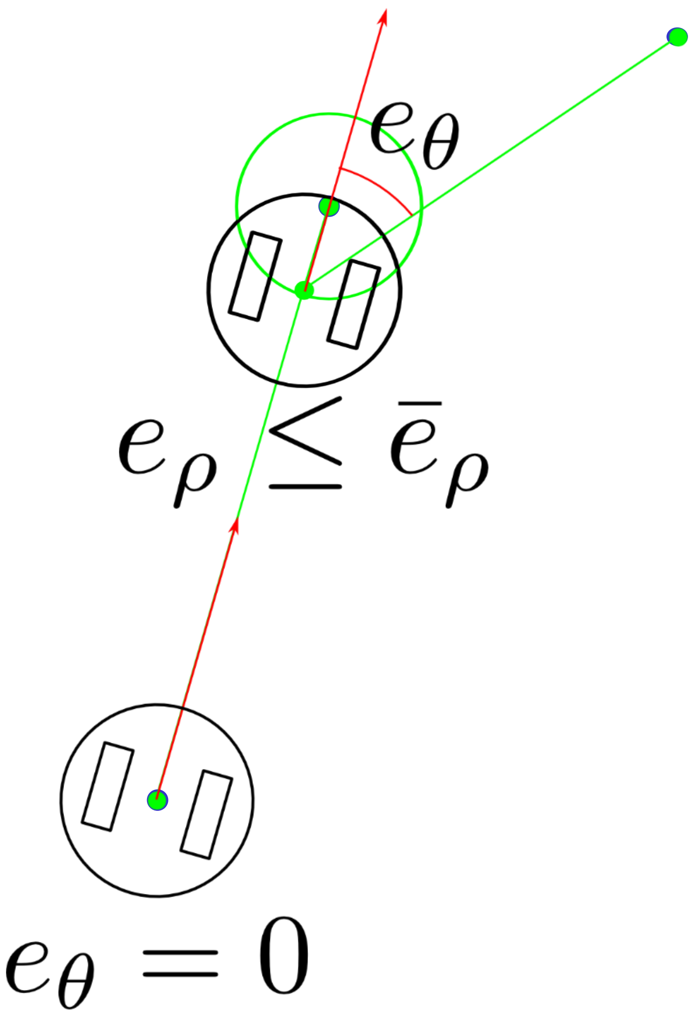
Matteo Matteucci – matteo.matteucci@polimi.it

Feedback control (diff drive example)

With feedback control the trajectory is recomputed and adapted online

We can design a simple control schema for path following:

- First we close a speed control loop on the wheels
- Then divide the problem in:
  - Control of the orientation
  - Control of the distance



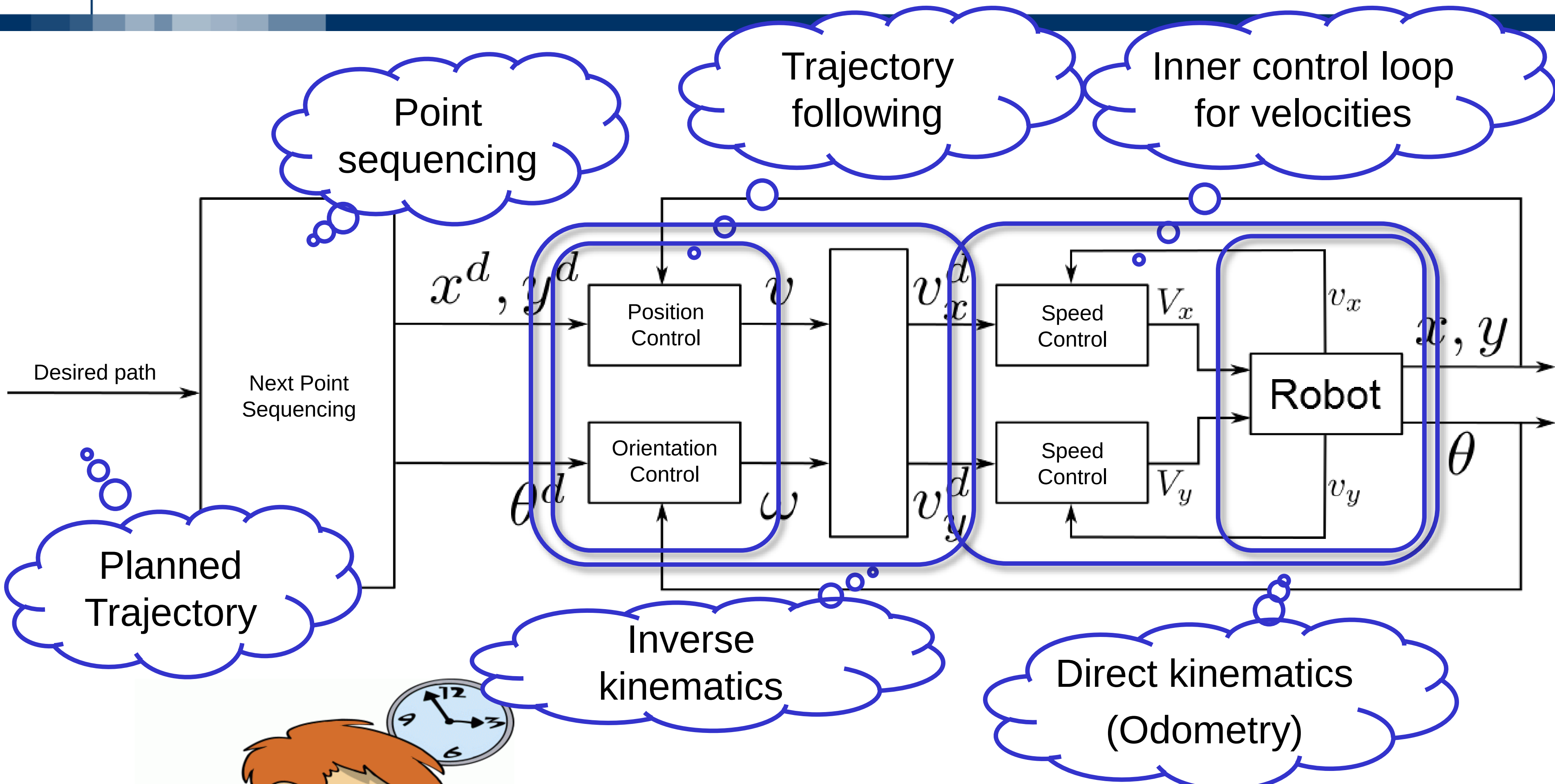
Control orientation acting on angular velocity

Control distance acting on linear velocity

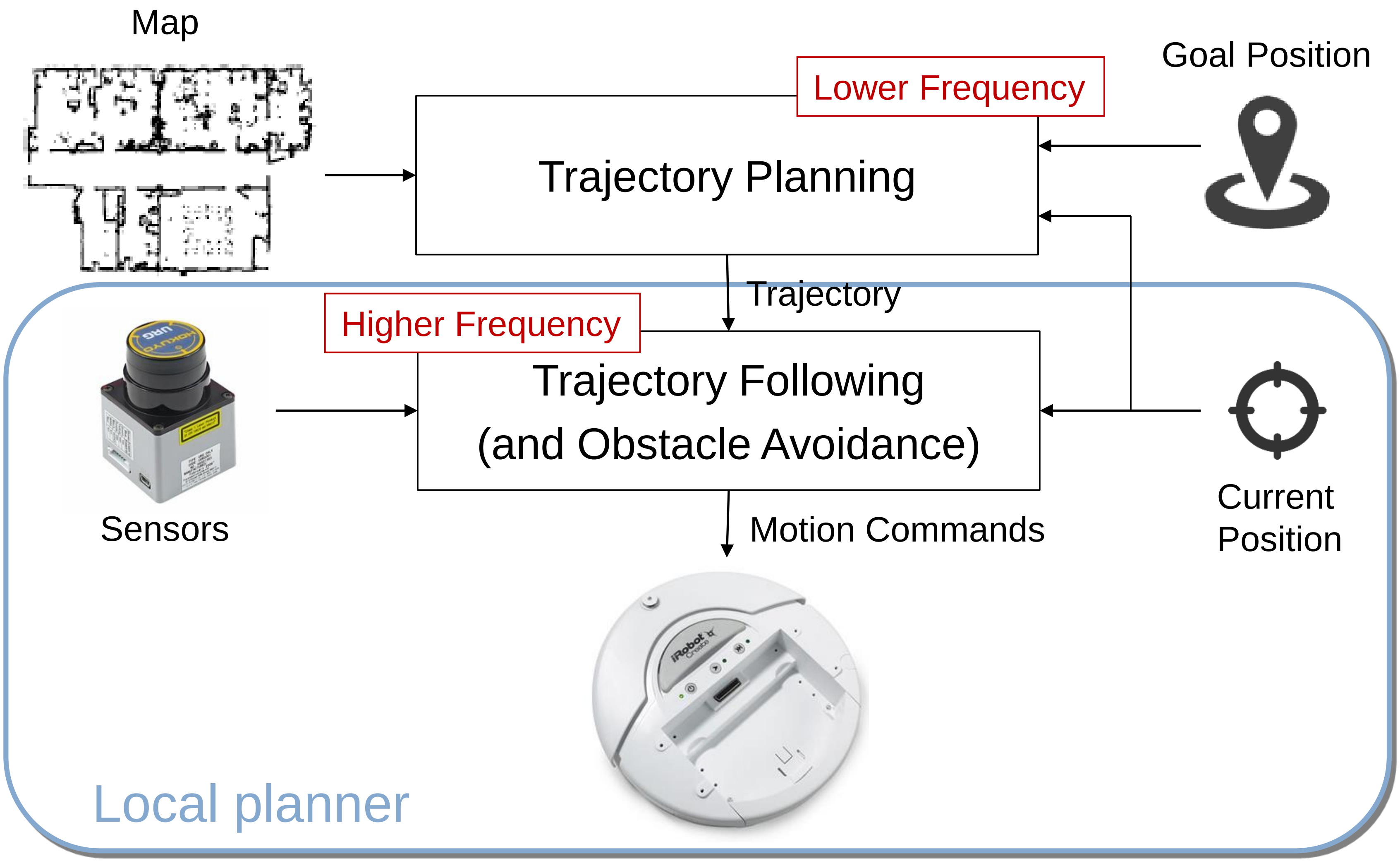
A simple logic handles the next point

Matteo Matteucci – matteo.matteucci@polimi.it

Feedback control (diff drive example)



What about obstacles?
Who does provide the trajectory?
Do you trust direct kinematics?



Matteo Matteucci – matteo.matteucci@polimi.it

Obstacle Avoidance (Local Path Planning)

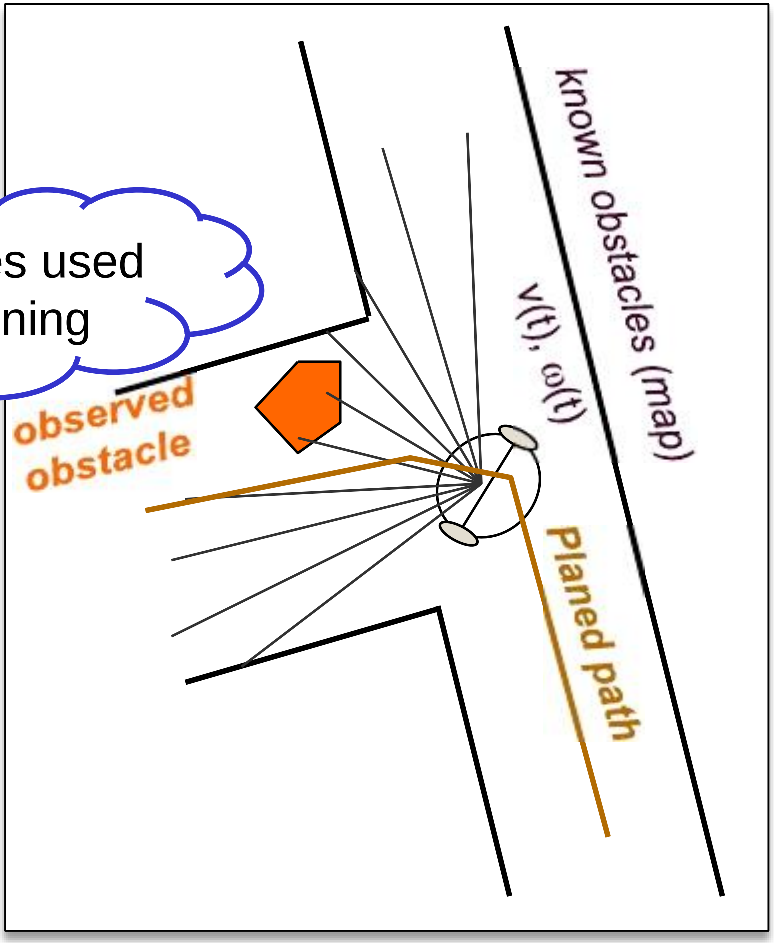
Obstacle avoidance should:

- Follow the planned path
- Avoid unexpected obstacle,
  i.e., those that were not in the map

Several proposed methods in the l

- Potential field methods
  [Borenstein, 1989]
- Vector field histogram
  [Borenstein, 1991, 1998, 2000]
- Nearness diagram
  [Minguez & Montano, 2000]
- Curvature-Velocity [Simmons, 1996]
- Dynamic Window Approach
  [Fox, Burgard, Thrun, 1997]
- …

Sometimes used
for planning



Matteo Matteucci – matteo.matteucci@polimi.it

The Simplest One …

"Bugs" have little if any knowledge …

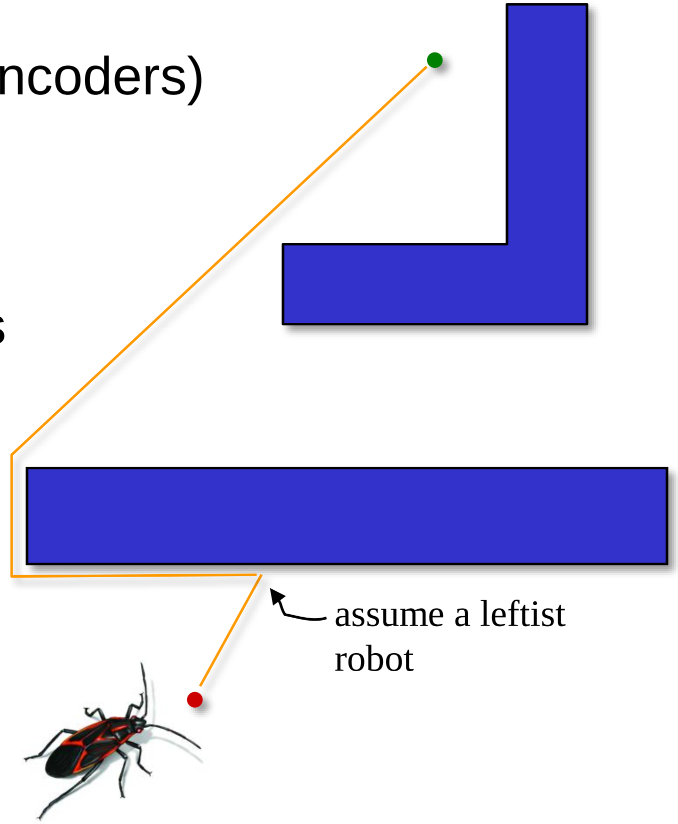
- known direction to the goal
- only local sensing (walls/obstacles + encoders)

… and their world is reasonable!

- finite obstacles in any finite range
- a line intersects an obstacle finite times

Switch between two basic behaviors

1. head toward goal
2. follow obstacles until you can head toward the goal again

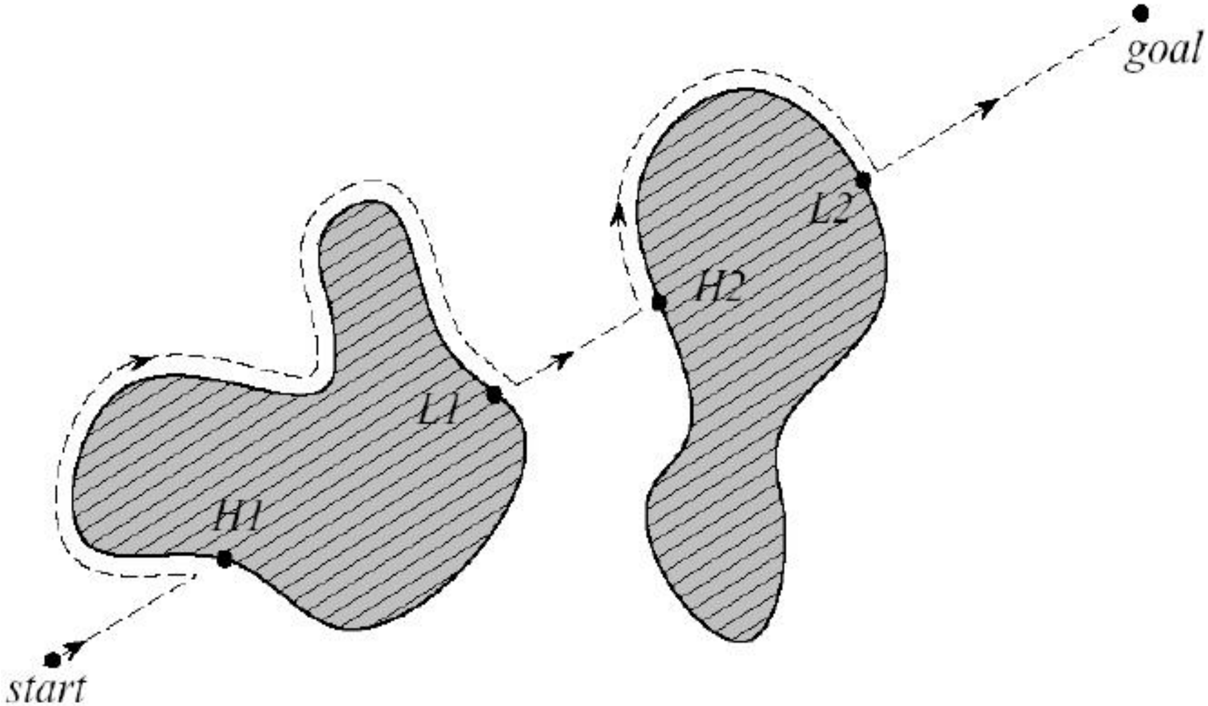


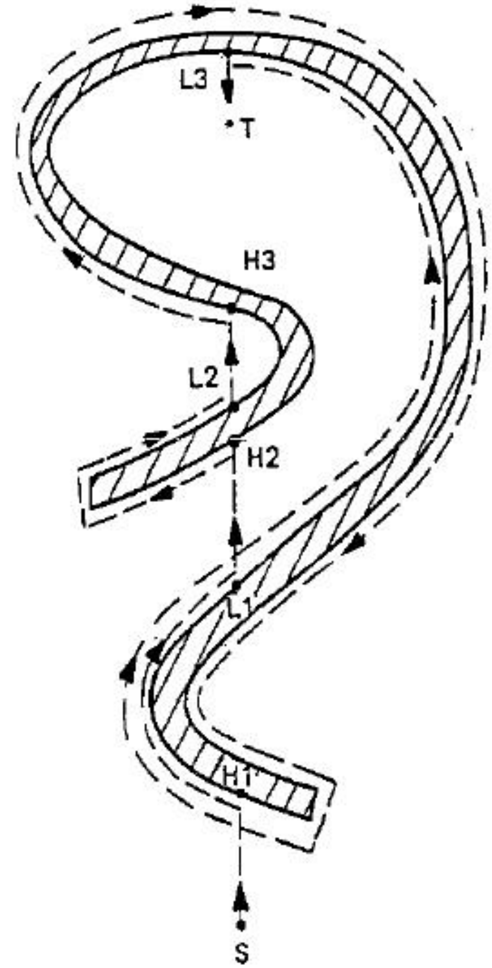
Matteo Matteucci – matteo.matteucci@polimi.it

Bugs and Features …

Each obstacle is fully circled before it is left at the point closest to the goals

- Advantages
  - No global map required
  - Completeness guaranteed
- Disadvantages
  - Solution are often highly suboptimal





Matteo Matteucci – matteo.matteucci@polimi.it

Vector Field Histograms (VHF) *[Borenstein et al. 1991]*

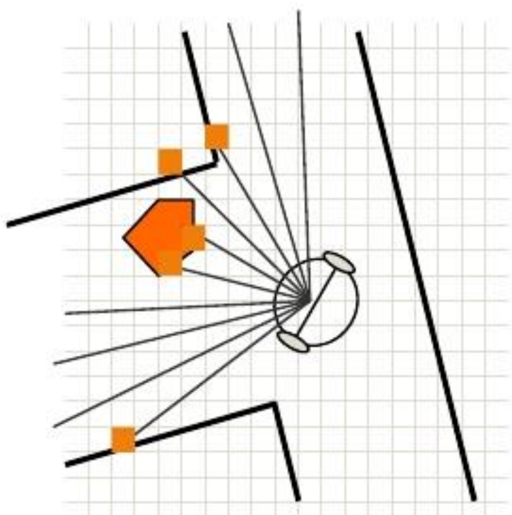
Use a local map of the environment and evaluate the angle to drive towards

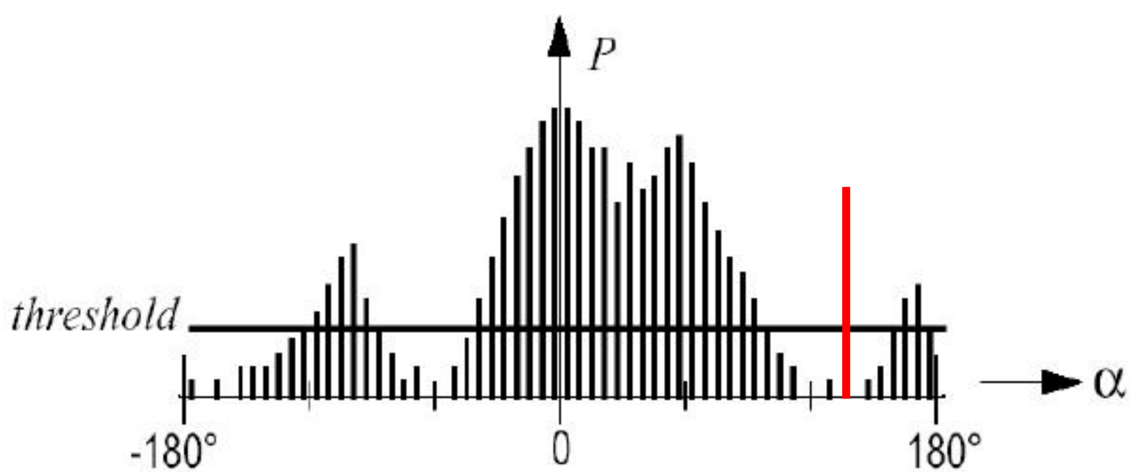
- Environment represented in a grid (2 DOF) with
- The steering direction is computed in two steps:
  - all openings for the robot to pass are found
  - the one with lowest cost function *G* is selected

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

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

Borenstein et al.

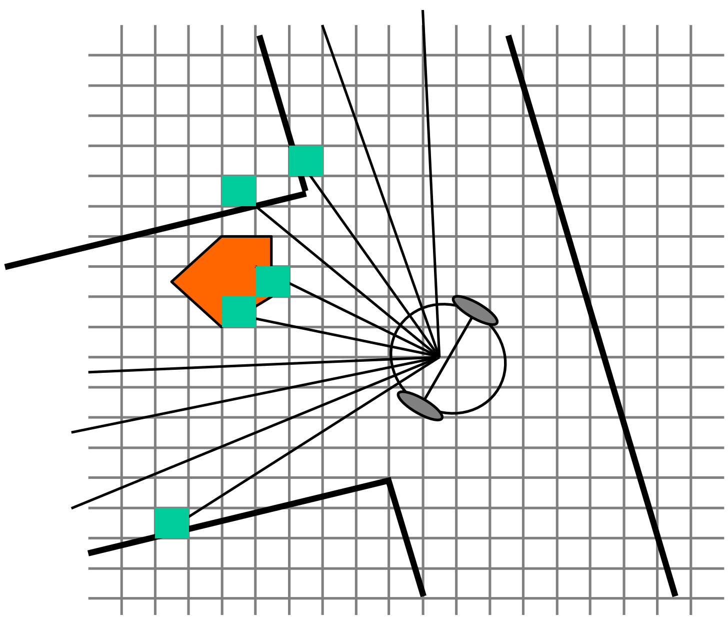


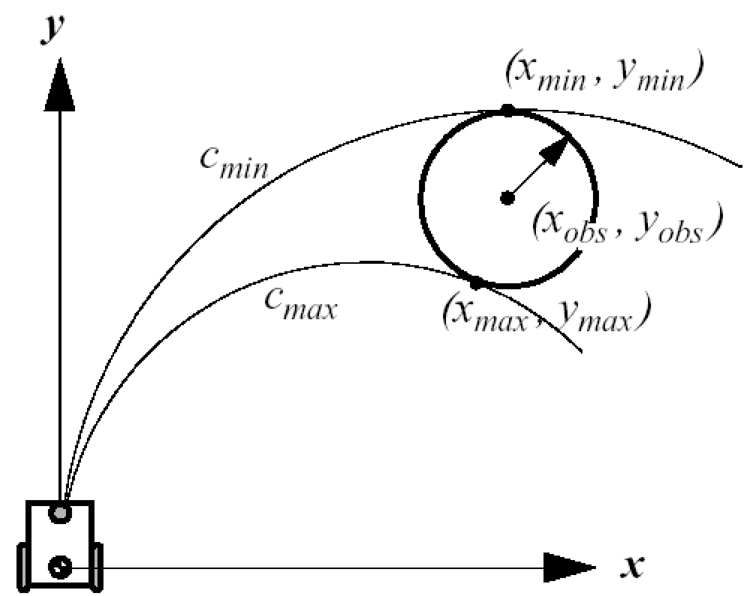


Curvature Velocity Methods (CVM) *[Simmons et al. 1996]*

CVMs add physical constraints from the robot and the environment on (v, w)

- Assumption that robot is traveling on arcs (c= w / v) with acceleration constraints
- Obstacles are transformed in velocity space
- An objective function to select the optimal speed



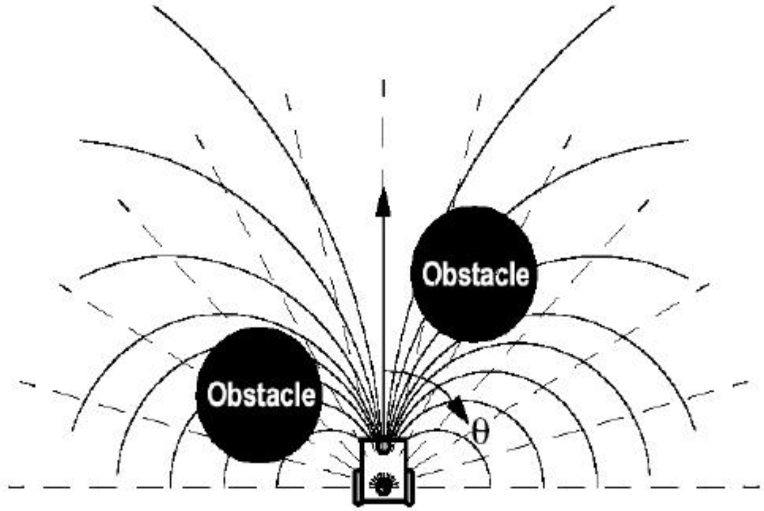


Simmons et al.

Vector Field Histogram+ (VFH+) *[Borenstein et al. 1998]*

VHF+ accounts also in a very simplified way for vehicle kinematics

- robot moving on arcs or straight lines
- obstacles blocking a given direction also blocks all the trajectories (arcs) going through this direction like in an Ackerman vehicle
- obstacles are enlarged so that all kinematically blocked trajectories are properly taken into account



Borenstein et al.

However VHF+ as VHF suffers

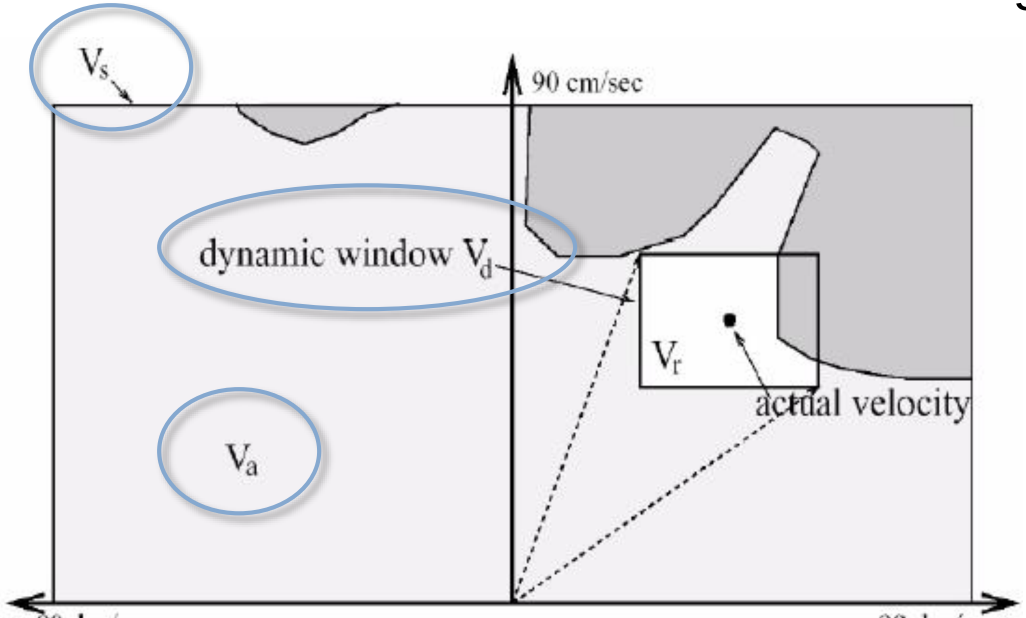
- Limitation if narrow areas (e.g. doors) have to be passed
- Local minima might not be avoided
- Reaching of the goal can not be guaranteed
- Dynamics of the robot not really considered

Matteo Matteucci – matteo.matteucci@polimi.it

Dynamic Window Approach (DWA) *[Fox et al. 1997]*

The kinematics of the robot are considered via local search in velocity space:

- Consider only circular trajectories determined by pairs $V_s=(v,\omega)$ of translational and rotational speeds
- A pair $V_a=(v, \omega)$ is considered admissible, if the robot is able to stop before it reaches the closest obstacle on the corresponding curvature.
- A dynamic window restricts the reachable velocities V_d to those that can be reached within a short time given limited robot accelerations



Fox et al.

$$V_d = \begin{cases} v \in [v - a_{tr} \cdot t, v + a_{tr} \cdot t] \\ \omega \in [\omega - a_{rot} \cdot t, \omega + a_{rot} \cdot t] \end{cases}$$

DWA Search Space

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

Matteo Matteucci – matteo.matteucci@polimi.it

How to choose (v,ω)?

Steering commands are chosen maximizing a heuristic navigation function:

- Minimize the travel time by "driving fast in the right direction"
- Planning restricted to V_r space [Fox, Burgard, Thrun '97]

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

Alignment with target direction

Distance to closest obstacle intersecting with curvature

Forward velocity of the robot

- Global approach [Brock & Khatib 99] in <x,y>-space uses

Forward robot velocity

Follows global path

$$NF = \alpha \cdot vel + \beta \cdot nf + \gamma \Delta nf + \delta goal$$

Cost to reach the goal

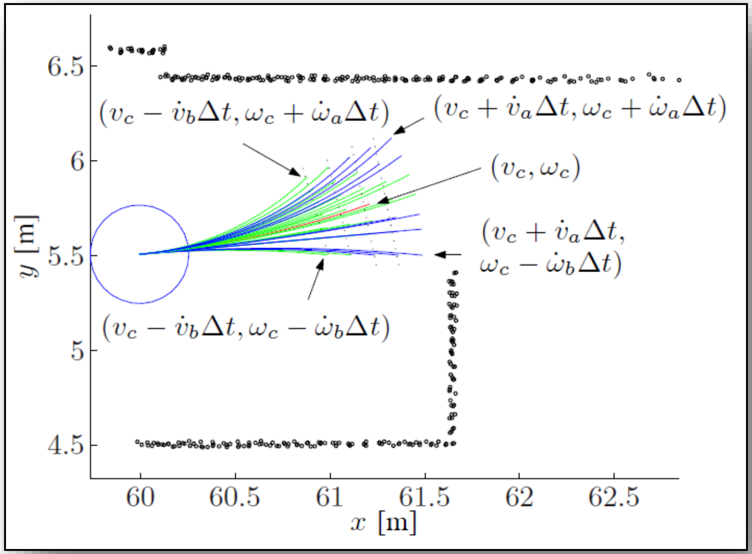
Goal nearness

Matteo Matteucci – matteo.matteucci@polimi.it

DWA Algorithm (as implemented in ROS movebase)

The basic idea of the Dynamic Window Approach (DWA) algorithm follows …

1. Discretely sample robot control space
2. For each sampled velocity, perform forward simulation from current state to predict what would happen if applied for some (short) time.
3. Evaluate (score) each trajectory resulting from the forward simulation
4. Discard illegal trajectories, i.e., those that collide with obstacles, and pick the highest-scoring trajectory

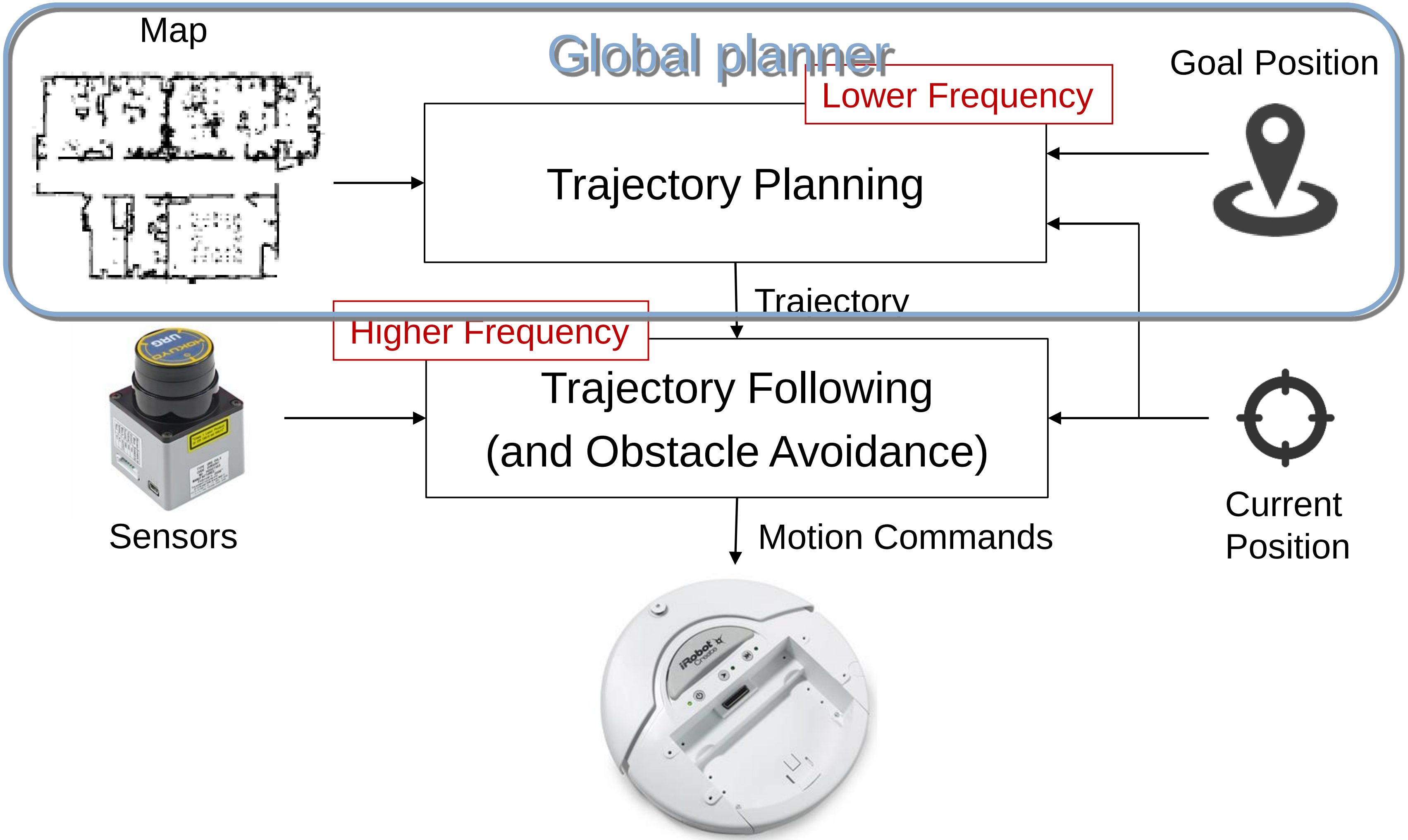


What about non circular kinematics?

Clothoid: $S(x) = \int_0^x \sin(t^2)\,dt, \quad C(x) = \int_0^x \cos(t^2)\,dt.$

Matteo Matteucci – matteo.matteucci@polimi.it

A Two Layered Approach



Matteo Matteucci – matteo.matteucci@polimi.it