# Control of Mobile Robots

Case studies

Prof. Luca Bascetta (luca.bascetta@polimi.it)

Politecnico di Milano – Dipartimento di Elettronica, Informazione e Bioingegneria

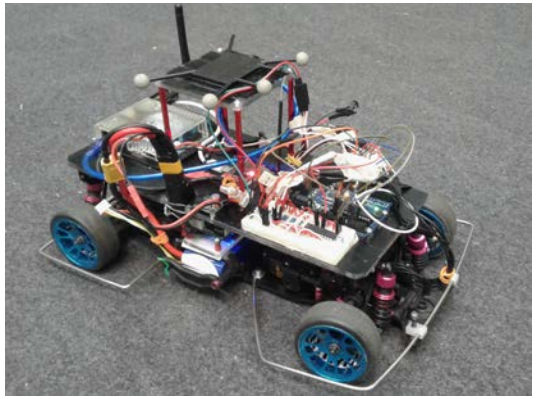
# Organization and motivations



This part aims at presenting three different case studies showing the importance of modelling, planning and control in real or realistic applications of mobile robotics.

The case studies are:

- autonomous navigation of a personal mobility device in human crowded scenarios
- autonomous driving at the limits of handling
- indoor mobile manipulation of an omnidirectional platform




The aim is to allow a personal mobility device to autonomously navigate in a complex environment crowded by walking humans (a street, an airport, a shopping mall, etc.), guaranteeing:

- obstacle avoidance
- comfort
- human acceptability (proxemics, social constraints, etc.)
- task completion

## *Experimental setup*



- an electric wheelchair Degonda Twist t4 2×2, with two independent driving wheels actuated by 350 W motors and three stabilizing caster wheels
- an on board computer, DS81L with Haswell Refresh Core i7 and 8 Gb RAM
- a dedicated board, allowing the computer to be interfaced to the wheelchair CAN bus
- two rotary encoders at the wheels
- two time-of-flight laser scanners Sick TiM 561, having an angular aperture of 270 deg, a range of 8 m, and a resolution of 0.33 deg

The first step to set up the navigation system is the selection of the model that is used:

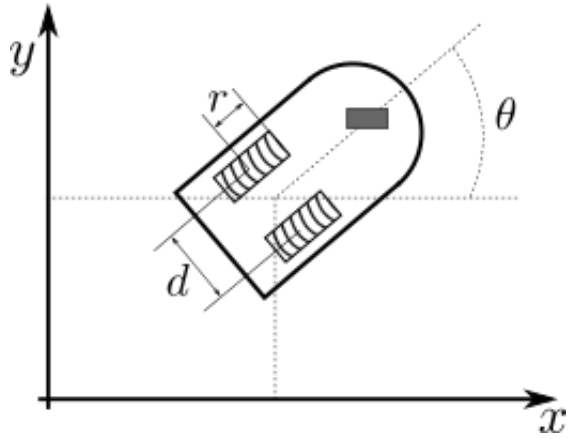
- to develop the control system
- to simulate the wheelchair for testing purposes

The wheelchair is a differential drive robot and can be represented by a kinematic or a dynamic model

$$\begin{cases} \dot{x} = \dfrac{\omega_R + \omega_L}{2} r \cos(\theta) \\ \dot{y} = \dfrac{\omega_R + \omega_L}{2} r \sin(\theta) \\ \dot{\theta} = \dfrac{\omega_R - \omega_L}{d} r \end{cases}$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}$$

$$\begin{bmatrix} \dot{v} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} 1/m & 0 \\ 0 & 1/I \end{bmatrix} \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix}$$



What are the advantages of using the dynamic model?

It allows to take into account:

- mass and inertia effects (e.g., people with different weight)
- tire-ground interaction effects (e.g., slippage and skidding)

And the disadvantages? …identifying unknown and time-varying parameters!

Let’s consider, however, that

- velocity and acceleration are relatively low for safety and comfort reasons
- low level motor velocity controllers guarantee a constant performance independently of the variation of some parameters (e.g., person weight)

A kinematic model is enough to represent the control relevant dynamics!

Assuming that we use the kinematic differential drive model

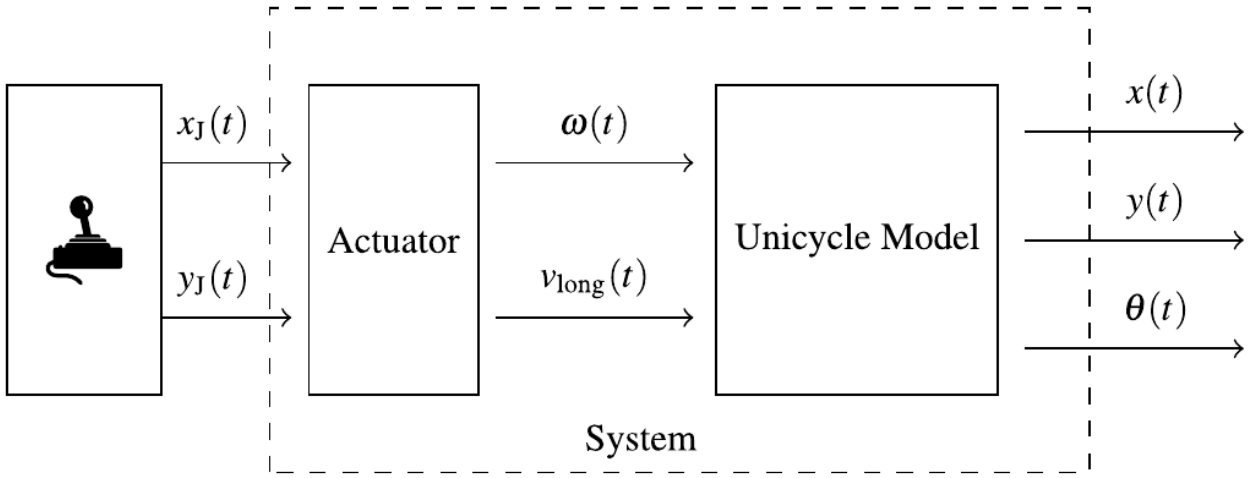
$$\begin{cases} \dot{x} = \dfrac{\omega_R + \omega_L}{2} r \cos(\theta) \\ \dot{y} = \dfrac{\omega_R + \omega_L}{2} r \sin(\theta) \\ \dot{\theta} = \dfrac{\omega_R - \omega_L}{d} r \end{cases}$$

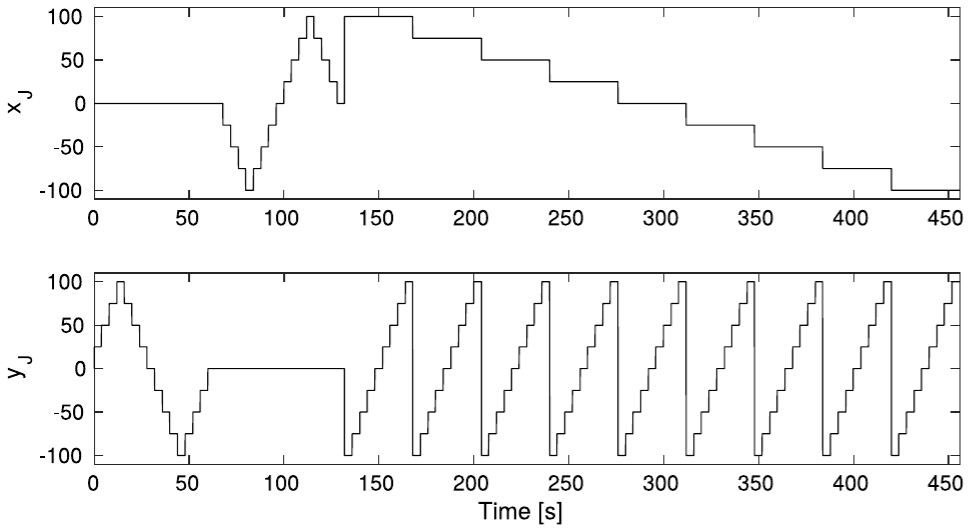
we need to identify a few parameters...

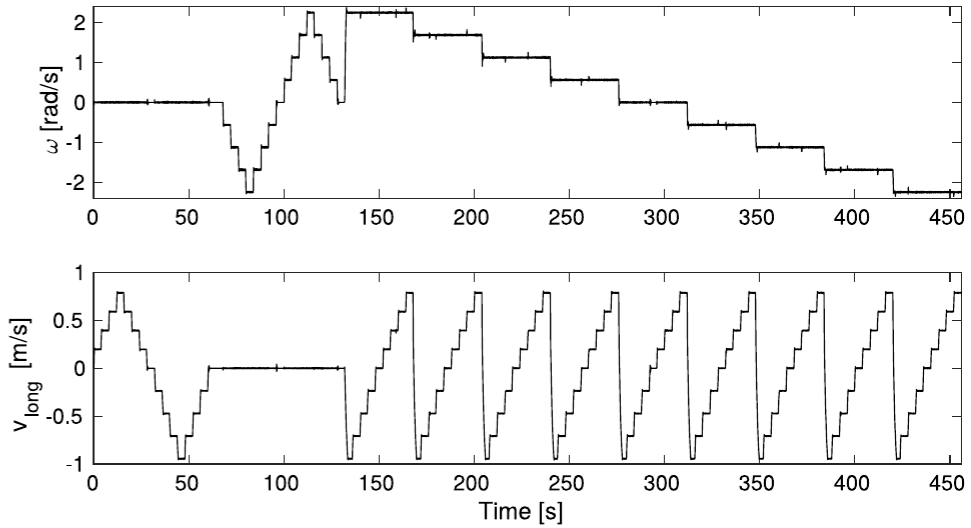
... $r$ and $d$, however, are simple parameters and the nominal values can be considered.

But the wheel velocities are not the actual input to the wheelchair, as they are not directly accessible. Instead, the $x$, $y$ position of the joystick has to be used… and the relation between joystick position and wheel velocities is unknown.

A static and dynamic relation between joystick position and unicycle linear and angular velocity can be estimated.



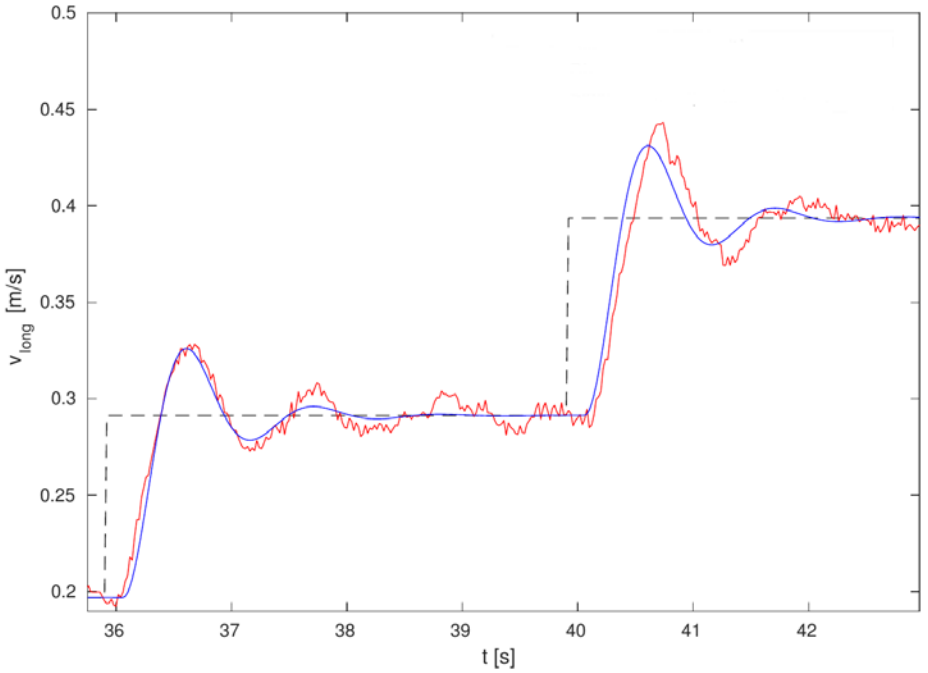



We can identify the following static model

$$\mu_x = 0.0225 \qquad \mu_y = \begin{cases} 0.007871 & \text{if} \quad v_{long} \geq 0 \\ 0.009444 & \text{if} \quad v_{long} < 0 \end{cases}$$

And using a sequence of step responses…




$$F(s) = \frac{\mu}{\frac{s^2}{\omega_n^2} + \frac{2\xi}{\omega_n}s + 1} e^{-s\tau_d}$$

… due to nonlinearities, however, the dynamic model turns out to be too sensitive to the step amplitude, we thus decided to use the static model.

The control problem can be addressed in two steps:

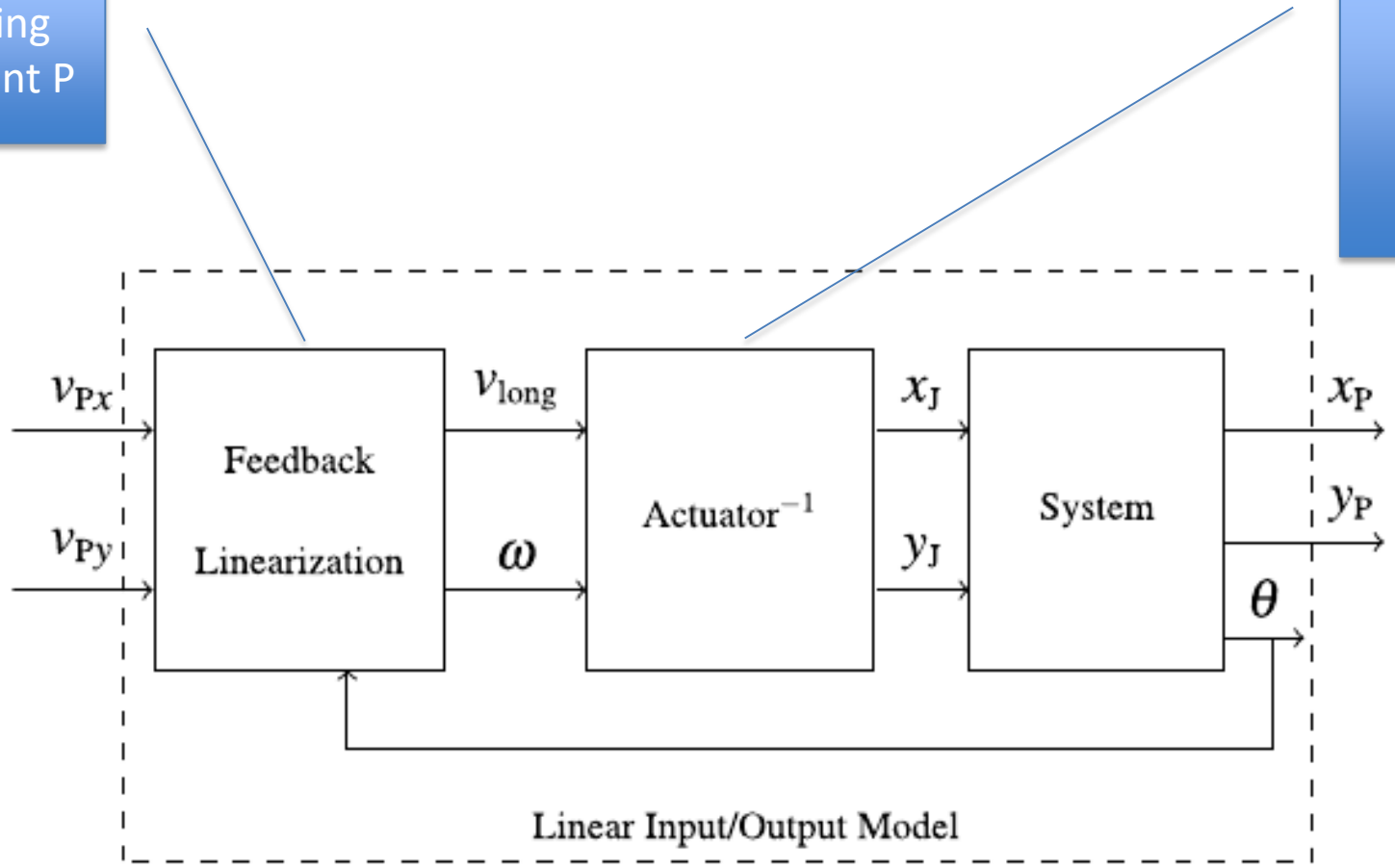
- closing an inner loop that linearizes the kinematic model and compensates for the actuator dynamics
- closing an outer loop that performs trajectory tracking and obstacle avoidance, and enforces human acceptability and comfort constraints

Let’s start from the inner loop…

The feedback linearizing law with respect to point P

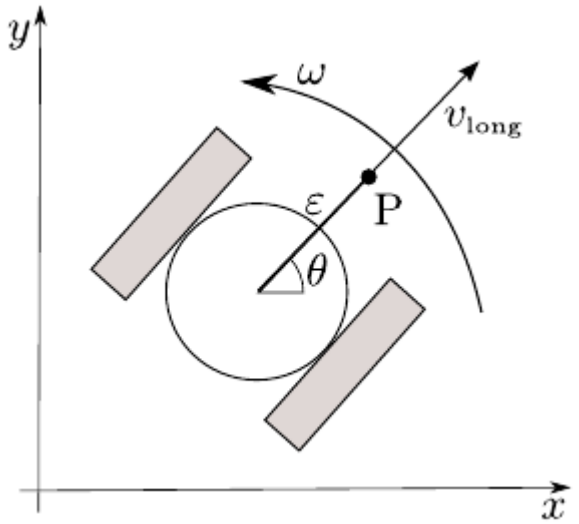
The inverse of the static actuator model:

$$x_J = \mu_x^{-1}\omega$$
$$y_J = \mu_y^{-1}v_{long}$$



The feedback linearizing law is based on the definition of a point $P$

$$x_P = x + \varepsilon \cos(\theta)$$
$$y_P = y + \varepsilon \sin(\theta)$$



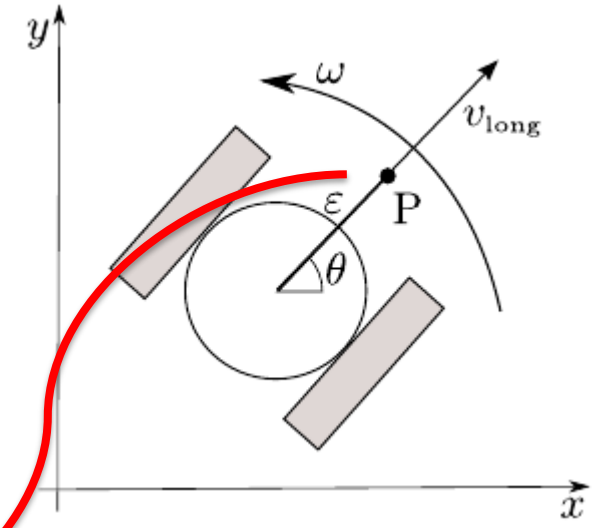

and a change of variables

$$v_{long} = v_{P_x} \cos(\theta) + v_{P_y} \sin(\theta)$$
$$\omega = \frac{v_{P_y} \cos(\theta) - v_{P_x} \sin(\theta)}{\varepsilon}$$

that makes the inner closed loop system equivalent to two independent integrators

$$\dot{x}_P = v_{P_x}$$
$$\dot{y}_P = v_{P_y}$$

We can set $\varepsilon = 0.5\ \mathrm{m}$, in this way point $P$ corresponds approximately to the wheelchair footboard.



What about the outer control loop?

The aims of this control loop are

- trajectory tracking
- obstacle avoidance
- human acceptability constraints
- comfort constraints

Considering that the model seen by the outer controller is linear and very simple, a viable solution to flexibly and modularly handle all those constraints is Model Predictive Control.

First of all, the discrete time model considered by the MPC is given by

$$x_P(k+1) = x_P(k) + \tau_s v_{P_x}(k)$$
$$y_P(k+1) = y_P(k) + \tau_s v_{P_y}(k)$$

and redefining the state and input vectors as

$$\xi(k) = \begin{bmatrix} x_P(k) \\ y_P(k) \end{bmatrix} \qquad \mathbf{u}(k) = \begin{bmatrix} v_{P_x}(k) \\ v_{P_y}(k) \end{bmatrix}$$

we obtain

$$\xi(k+1) = \mathbf{A}\xi(k) + \mathbf{B}\mathbf{u}(k)$$

Then, the cost function can be expressed as

$$J(k) = \sum_{i=0}^{N-1} \left( \left\| \xi(k+i) - \xi_{ref} \right\|_{\mathbf{Q}}^2 + \left\| \mathbf{u}(k+i) \right\|_{\mathbf{R}}^2 \right) + \left\| \xi(k+N) - \xi_{ref} \right\|_{\mathbf{P}}^2$$

The control problem is configured as a regulation problem.
This term penalizes the regulation error (Euclidean distance from the goal).

This term penalizes the control effort.

This term penalizes the regulation error at the end of the prediction horizon (stability).

MPC allows to easily introduce linear constraints on the state and control input.

Concerning actuation and comfort we can introduce constraints on the maximum velocity and maximum velocity variation.

Let's first consider the maximum velocity constraint

$$0 \leq \sqrt{v_{P_x}^2(k+i) + v_{P_y}^2(k+i)} \leq v_{max} \qquad i = 0, \ldots, N-2$$

as this constraint is nonlinear we need to linearize it. We can consider a first order truncated Taylor series around the predicted trajectory

$$0 \leq \frac{v_{P_x}(k+i|k-1)}{\bar{v}(k+i|k-1)} v_{P_x}(k+i) + \frac{v_{P_y}(k+i|k-1)}{\bar{v}(k+i|k-1)} v_{P_y}(k+i) \leq v_{max} \qquad i = 0, \ldots, N-2$$

where

$$\bar{v}(k+i|k-1) = \sqrt{v_{P_x}^2(k+i|k-1) + v_{P_y}^2(k+i|k-1)}$$

To take user comfort into account we can introduce a constraint on the velocity variation (acceleration)

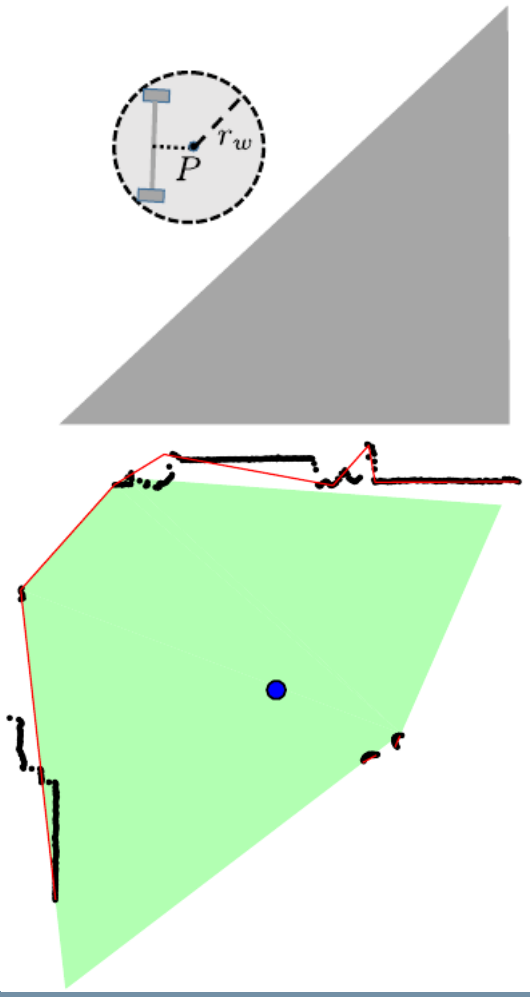
$$-\Delta v_{max} \leq v_{P_x}(k+i) - v_{P_x}(k+i-1) \leq \Delta v_{max}$$
$$-\Delta v_{max} \leq v_{P_y}(k+i) - v_{P_y}(k+i-1) \leq \Delta v_{max}$$

Further constraints have to be introduced in order to guarantee obstacle avoidance.

Obstacle avoidance is based on the idea of introducing a set of linear constraints that define the allowed (obstacle free) admissible convex state region.

The first step is obstacle detection based on laser scanner measurements. The algorithm is based on the following steps:

- point cloud roto-translation from the wheelchair to the global reference frame
- points are grouped into clusters according to their relative position to distinguish obstacles from spot openings
- clusters are simplified, by removing outliers and redundant points from the obstacle outline, and convexified
- a linear inequality constraint is selected for each object



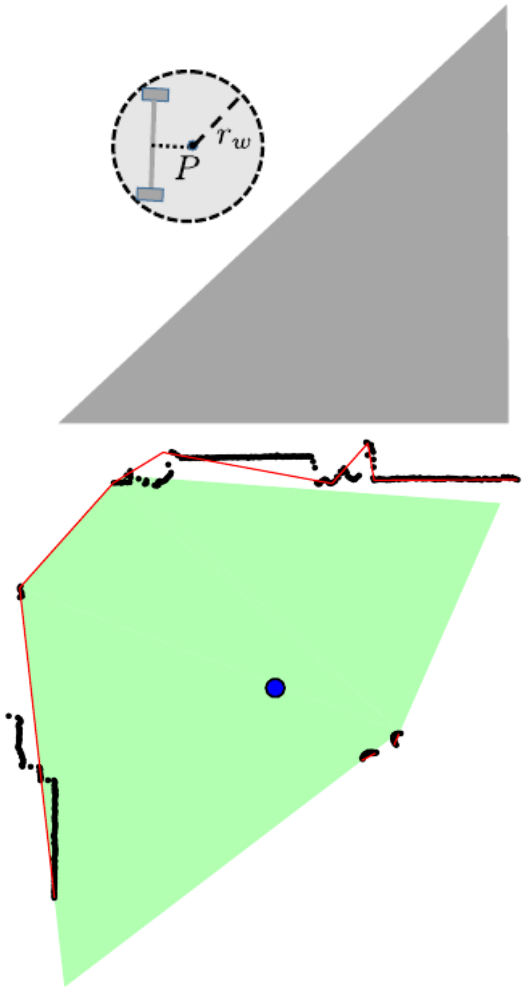
Once the allowed admissible convex state region has been determined, it can be represented as a set of linear constraints

$$h_1^j\, x_P(k+i) + h_2^j\, y_P(k+i) \leq l^j \qquad j = 1, \ldots, n_c$$

The resulting linear MPC problem can be efficiently solved in realtime.

The controller has been implemented in C++ under ROS as a local planner in the movebase navigation architecture.

It runs with a sampling time of 200 ms and a prediction horizon of 15 steps. Considering a maximum velocity of $0.55\ m/s$ and a maximum acceleration of $0.2\ m/s^2$ the prediction time is greater than the time required to stop the wheelchair travelling at the maximum speed.



As the goal of the MPC is to steer the wheelchair to a desired goal position, but it cannot control the final orientation, a further linear proportional controller

$$v_{long} = 0$$

$$\omega = K_\theta \left(\theta_d - \theta\right)$$

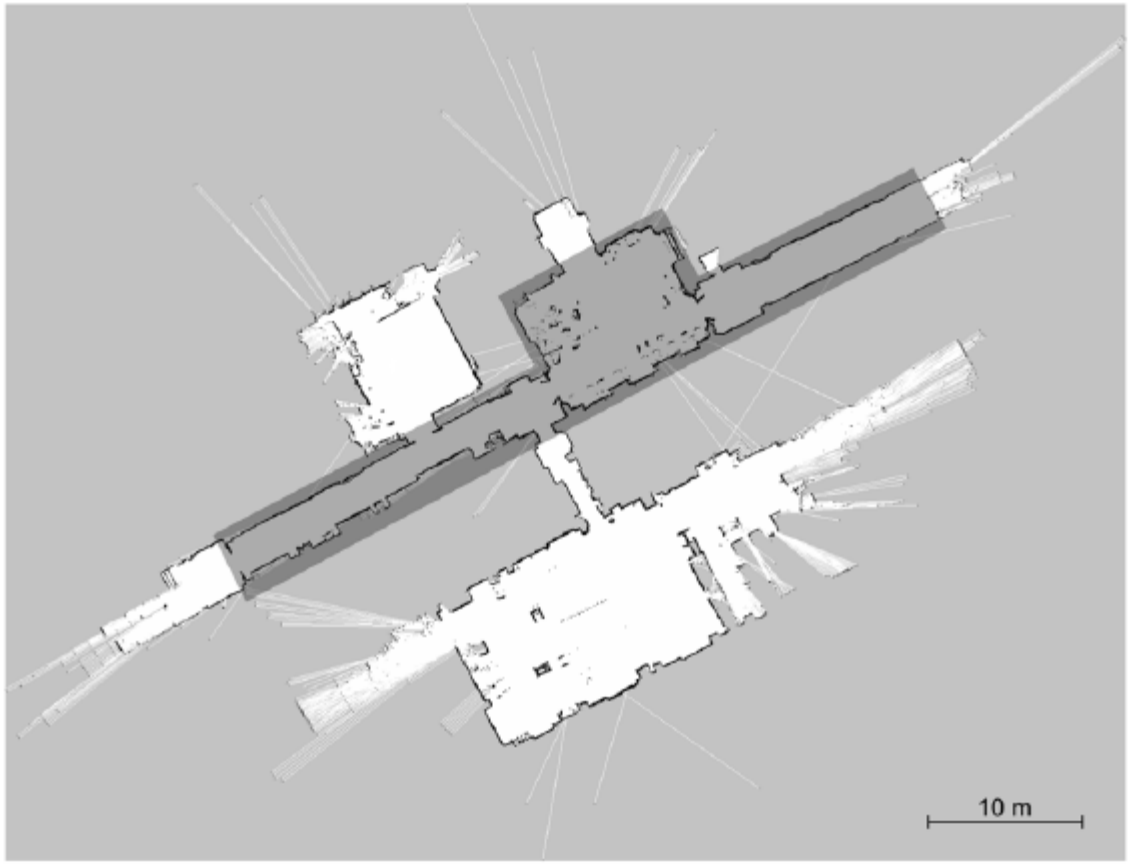
takes over when the final goal is achieved.

Let’s see some experimental results…

Tests have been performed in the ground floor of building 7.

This space has been first mapped using ROS standard mapping algorithms.

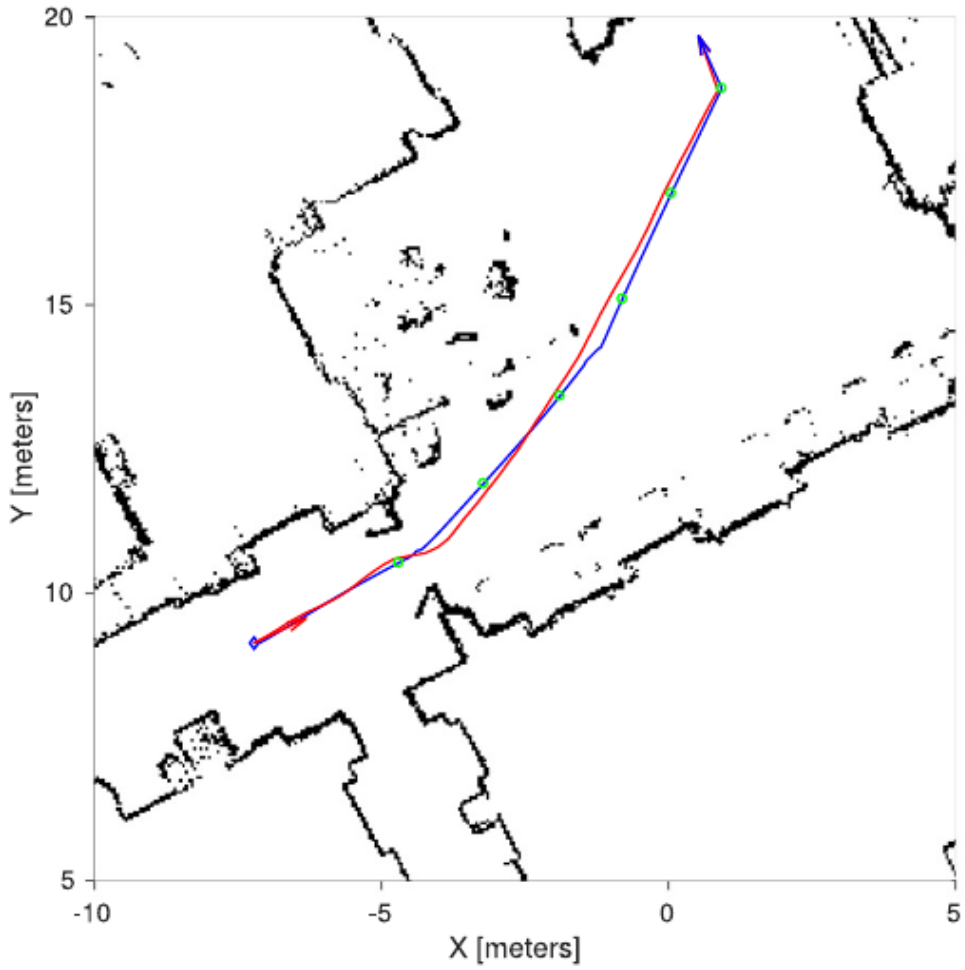
Experiments have been performed considering different starting/goal points, planning a desired trajectory with A* and sampling it approximately every one meter to generate the intermediate goals.



# Autonomous navigation of a personal mobility device in human crowded scenarios

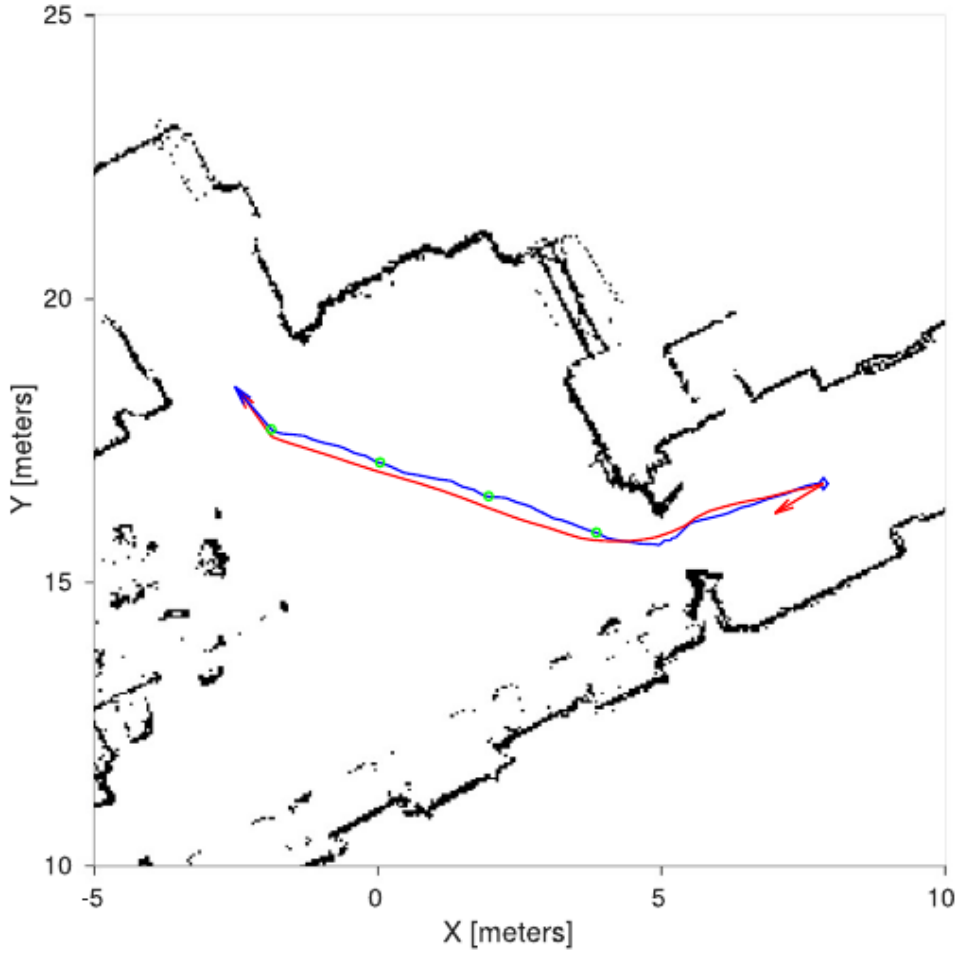
## *Experimental results*





## *Experimental results*

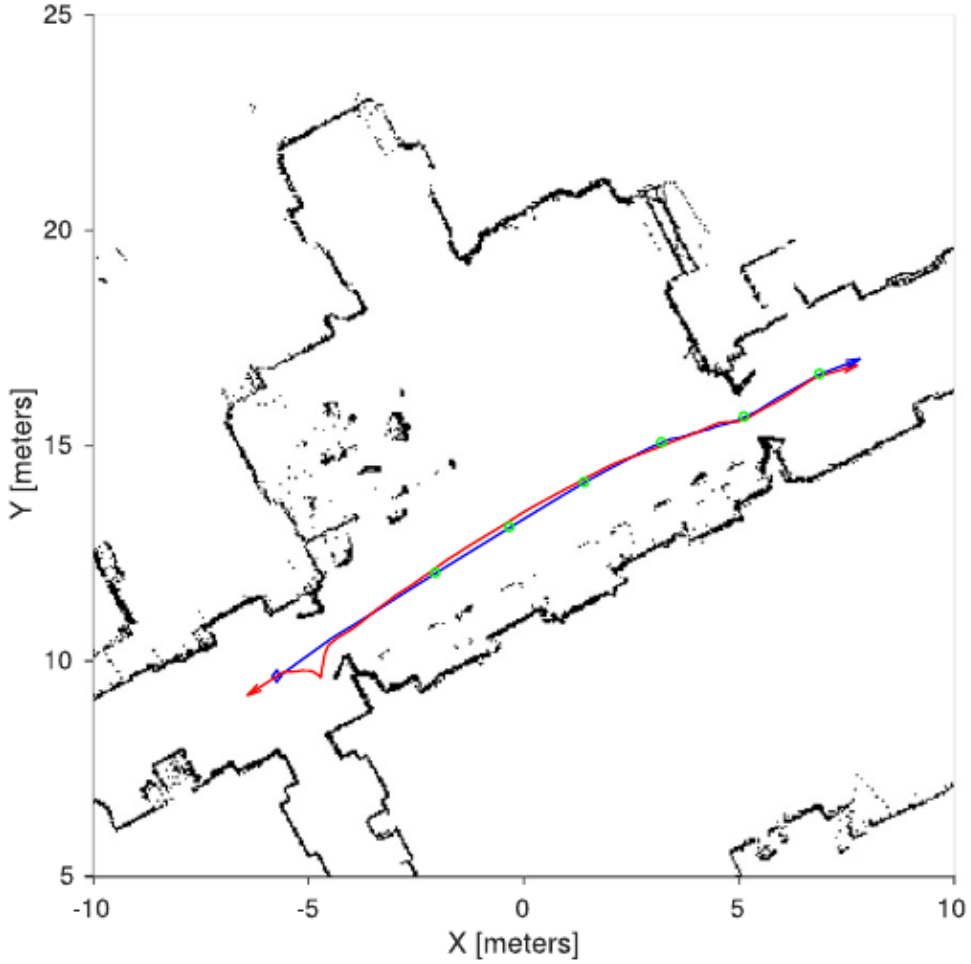




# Autonomous navigation of a personal mobility device in human crowded scenarios

## *Experimental results*

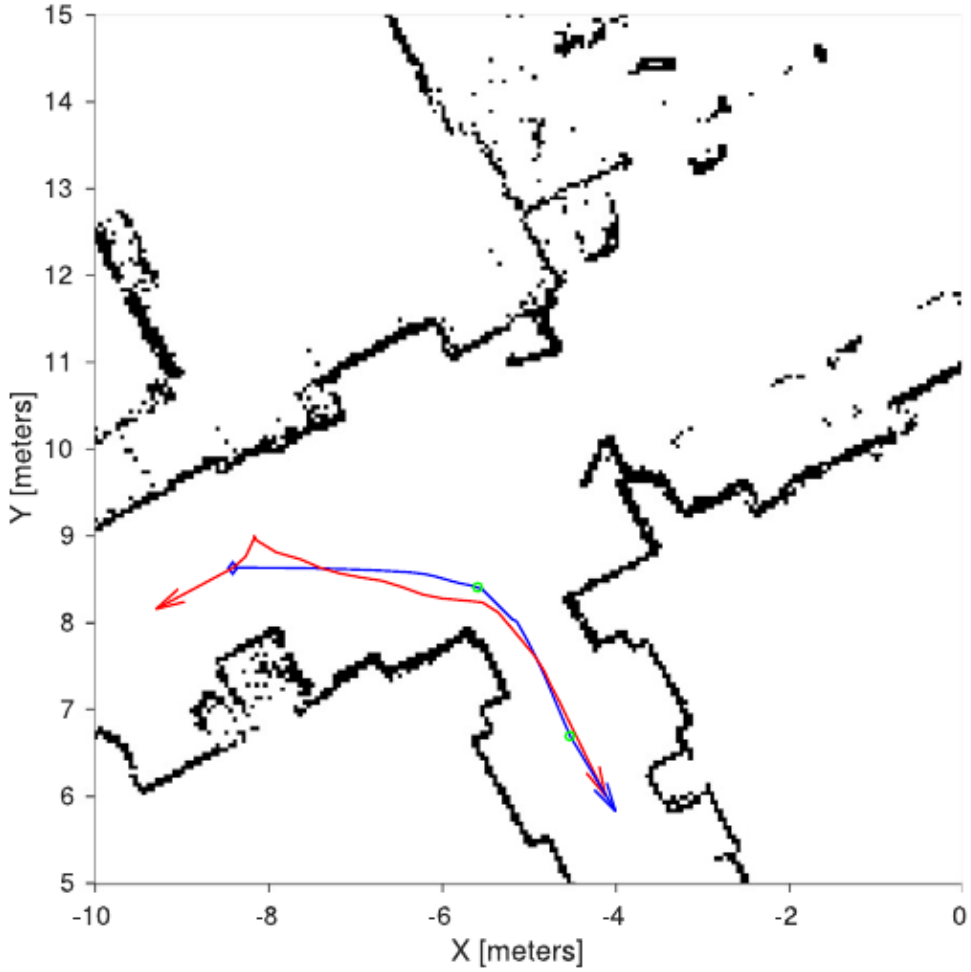




# Autonomous navigation of a personal mobility device in human crowded scenarios

## *Experimental results*





A few remarks to conclude the part on control.

In the actual version, the MPC controller does not consider:

- moving obstacles, including a prediction of their motion
- human acceptability constraints

Thanks to the modularity of this approach, new constraints can be easily added.

The planning algorithm used in the experiments is A*, and it does not take into account:

- moving obstacles
- vehicle kinematic/dynamic constraints
- human acceptability and social constraints

Two different approaches, partially including these requirements, can be developed using sampling-based motion planners.

A simple solution, based on RRT*, allows to consider the most crowded regions as zones the path should avoid.

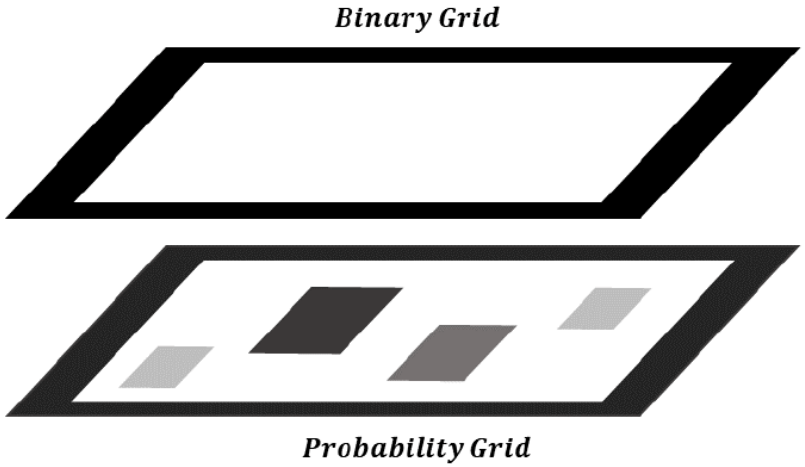
Assume that a grid is superimposed to the floor, and the environment is equipped with ceiling cameras that allows to periodically count the number of people in each cell.

A measure of the instantaneous density of people at time $k$ in the area $A$ framed by the camera can be defined as

$$\rho_k = \frac{n_k}{A}$$

and the probability that a cell of area $A_c$ is occupied is

$$P_{occ_k} = \rho_k A_c = \frac{n_k A_c}{A}$$



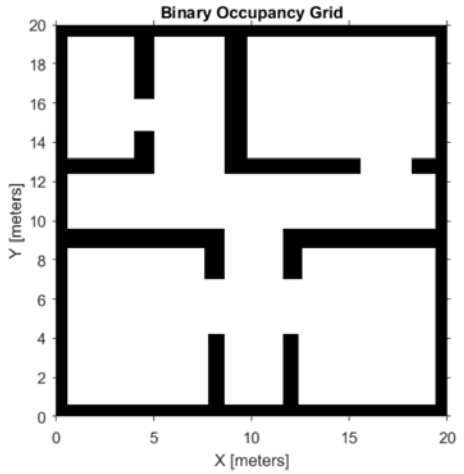
In order to guarantee that $0 \leq P_{occ_k} \leq 1$ we should modify the previous definition as

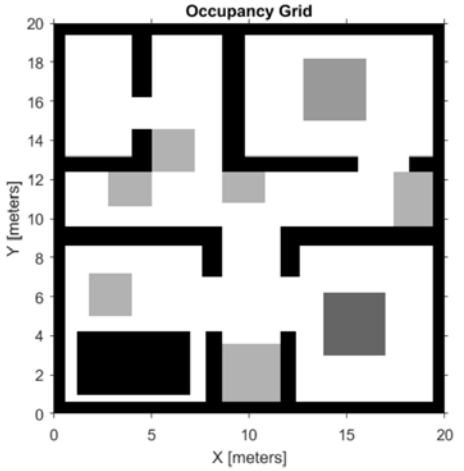
$$P_{occ_k} = \frac{\min\left(n_k A_c, A\right)}{A}$$

Finally, considering a time interval $[k, k+N]$ the probability map is constructed on the average probability of each cluster, defined as

$$\bar{P}_{occ} = \frac{1}{N} \sum_{k=0}^{N} P_{occ_k}$$

Based on this probability an additional cost can be defined associating to each edge a cost equal to the sum of the probabilities of the areas traversed by that edge.

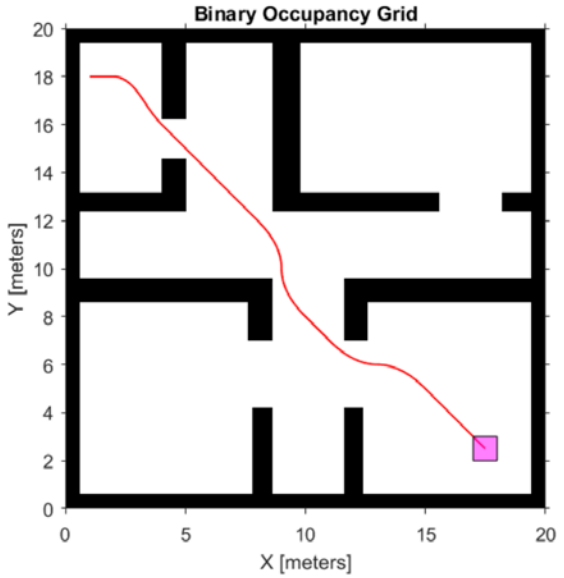


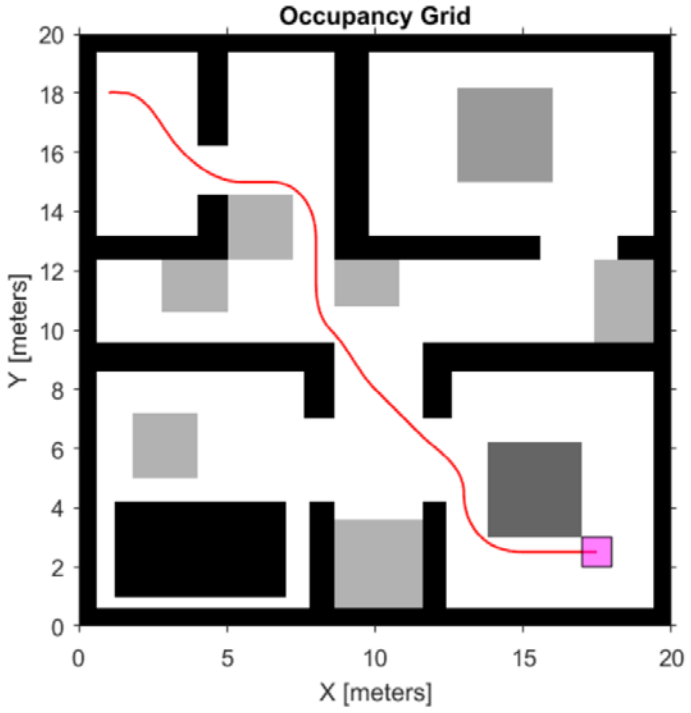



A kinodynamic motion planner like RRT*-MP can be modified considering this additional cost, in this way the planner can consider

- the motion model of the robot
- the density of people in the environment

This planner, however, is not able to completely solve the problem.

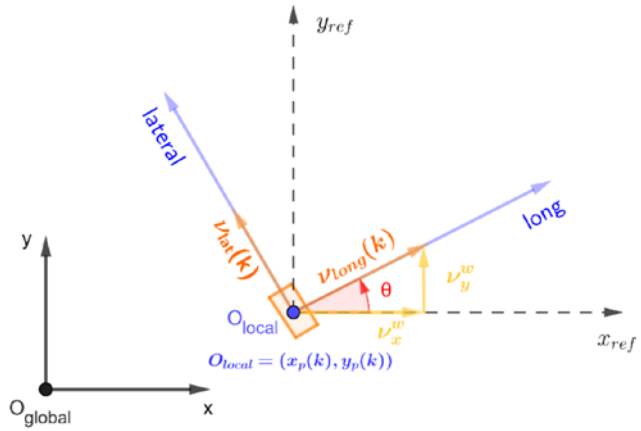


Considering the characteristics of the problem, a variant of RRT called RRTX, that is particularly suitable for real-time motion planning for environments with unpredictable obstacles, can be adopted.

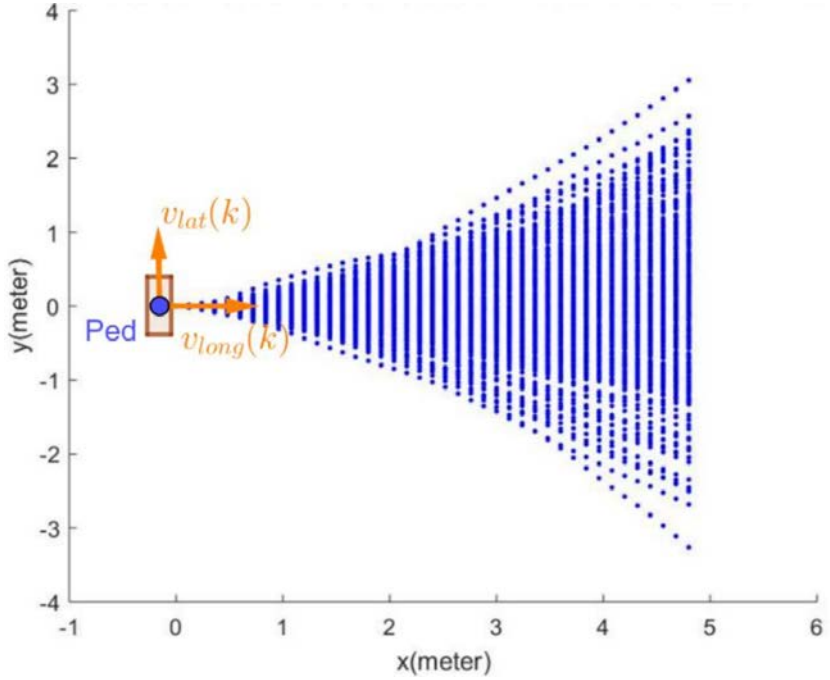
In order to use a planner that takes into account the actual and future motion of humans, we need to set up a human motion model. As this model has to be simple enough to predict future human location together with the probability of this estimate, we adopt a simplified linear model

$$
\begin{aligned}
p_{long}(k+1) &= p_{long}(k) + \tau v_{long}(k) \\
v_{long}(k+1) &= v_{long}(k) \\
p_{lat}(k+1) &= p_{lat}(k) + \tau v_{lat}(k) \\
v_{lat}(k+1) &= v_{lat}(k) + \tau \sigma_{lat}(k)
\end{aligned}
$$



We can assume:

- a constant longitudinal velocity of $1.2\ m/s$
- a standard deviation of the noise on the lateral velocity of $0.5\ m/s^2$



RRTX is an asymptotically optimal, sampling-based, motion re-planning algorithm.

It allows to re-plan “on-the-fly” changing the initial path according to the changes in the obstacles.

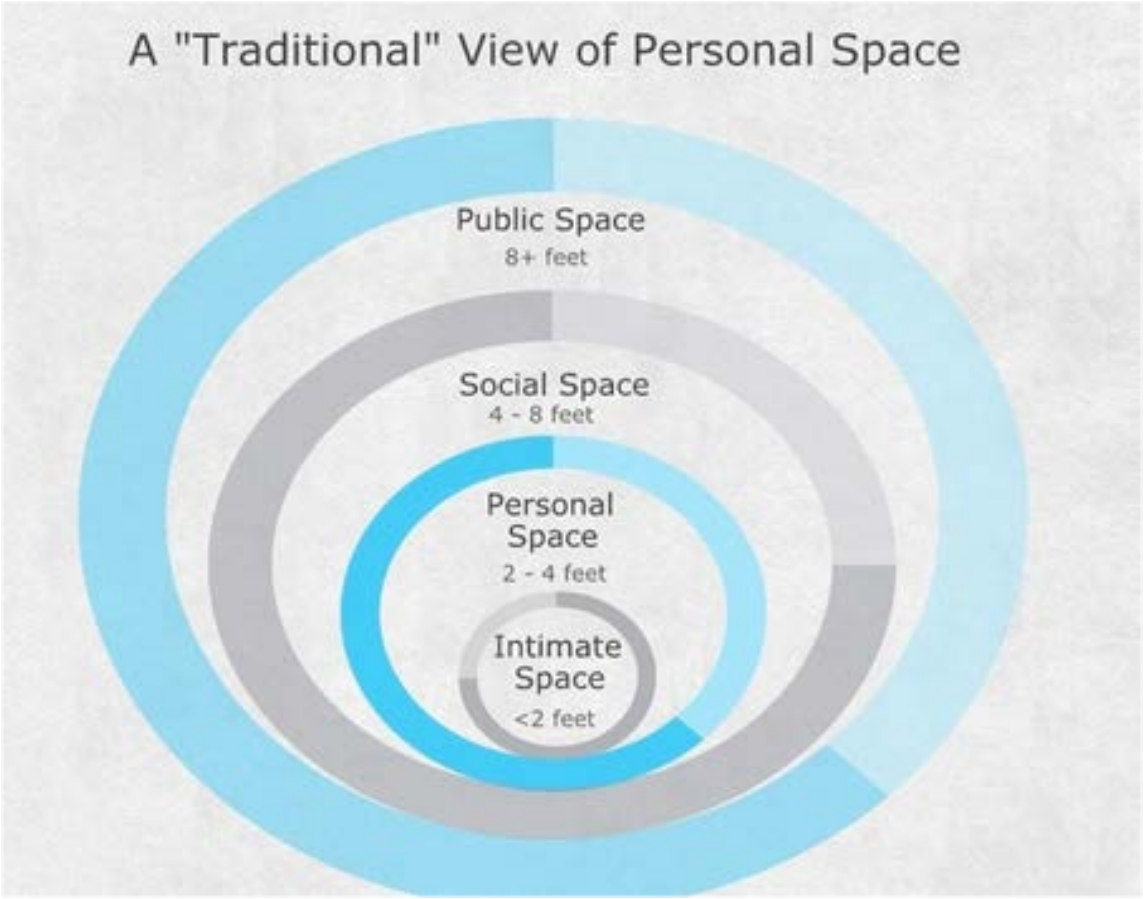
In the current version the algorithm does not support kinodynamic constraints.

1: $V \leftarrow \{v_{goal}\}$;
2: $v_{bot} \leftarrow v_{start}$;
3: **while** $v_{bot} \neq v_{goal}$ **do**
4: $r \leftarrow$ shrinkingBallRadius (|V|);
5: **if** *obstacles have changed* **then**
6: updateObstacles ();
7: **if** *robot is moving* **then**
8: $v_{bot} \leftarrow$ updateRobot($v_{bot}$);
9: $v \leftarrow$ randomNode($\mathcal{X}_{free}$);
10: $v_{nearest} \leftarrow$ nearest($v$);
11: **if** $d\ (v, v_{nearest}) > \delta$ **then**
12: $v \leftarrow$ saturate($v, v_{nearest}$);
13: **if** $v \notin X_{obs}$ **then**
14: extend($v, r$);
15: **if** $v \in V$ **then**
16: rewireNeighbours ($v$);
17: reduceInconsistency ();

To plan a path considering humans as moving obstacles and, possibly, enforcing social constraints, we have considered a 3D configuration space $(x, y, t)$.

The human model is used to predict the probability of each human to be at a certain position in the next seconds, and this probability is used as an additional cost associated to edges.

A further cost can be added in order to take proxemics into account.

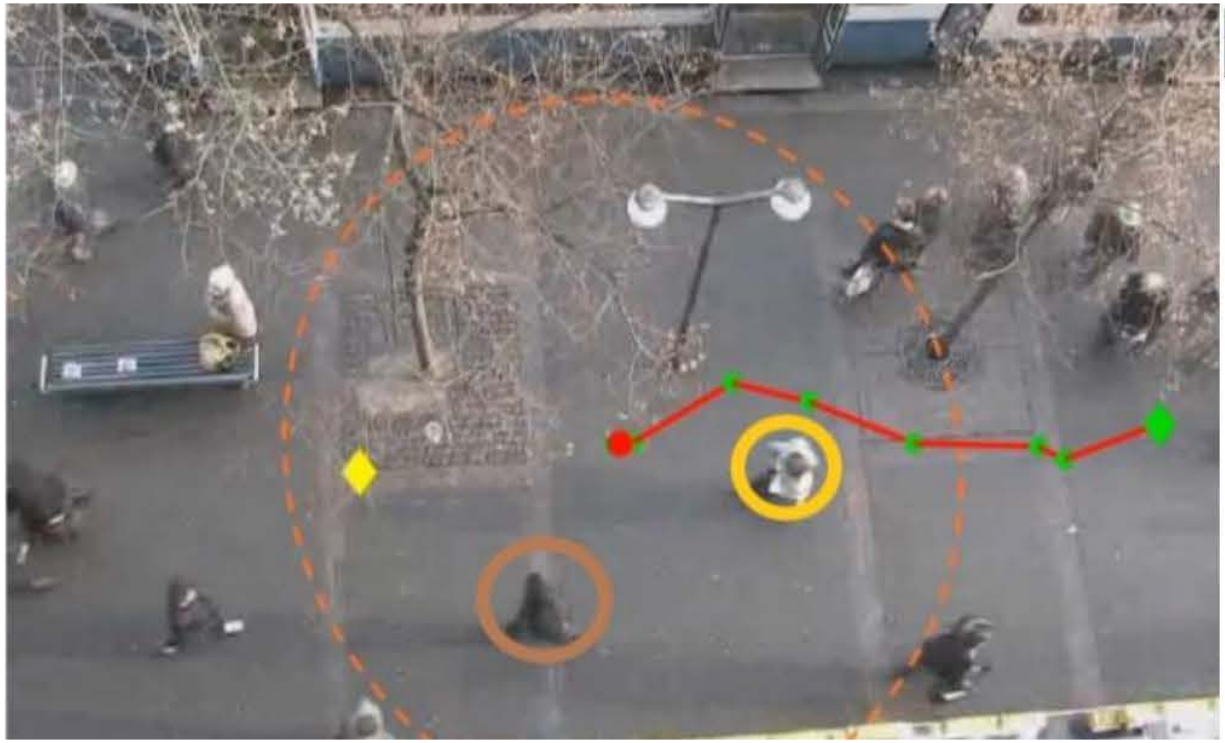


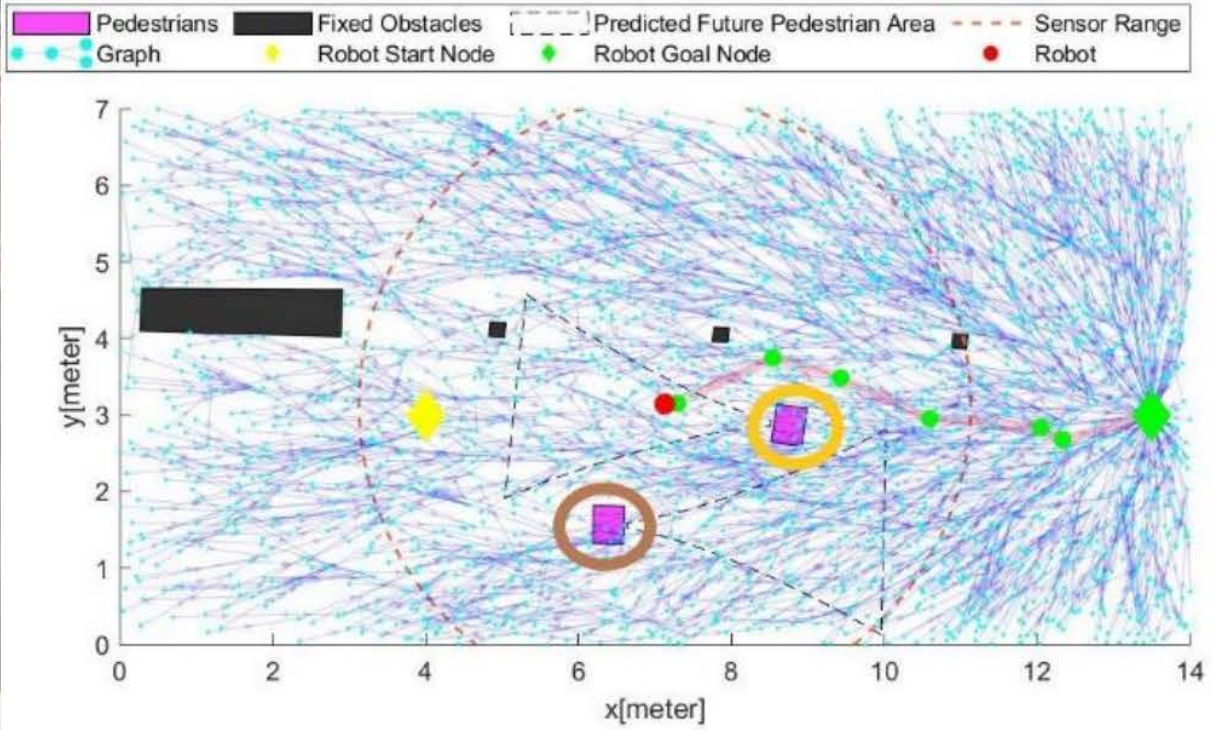
## *Simulation results*

# Autonomous navigation of a personal mobility device in human crowded scenarios

## *Simulation results*



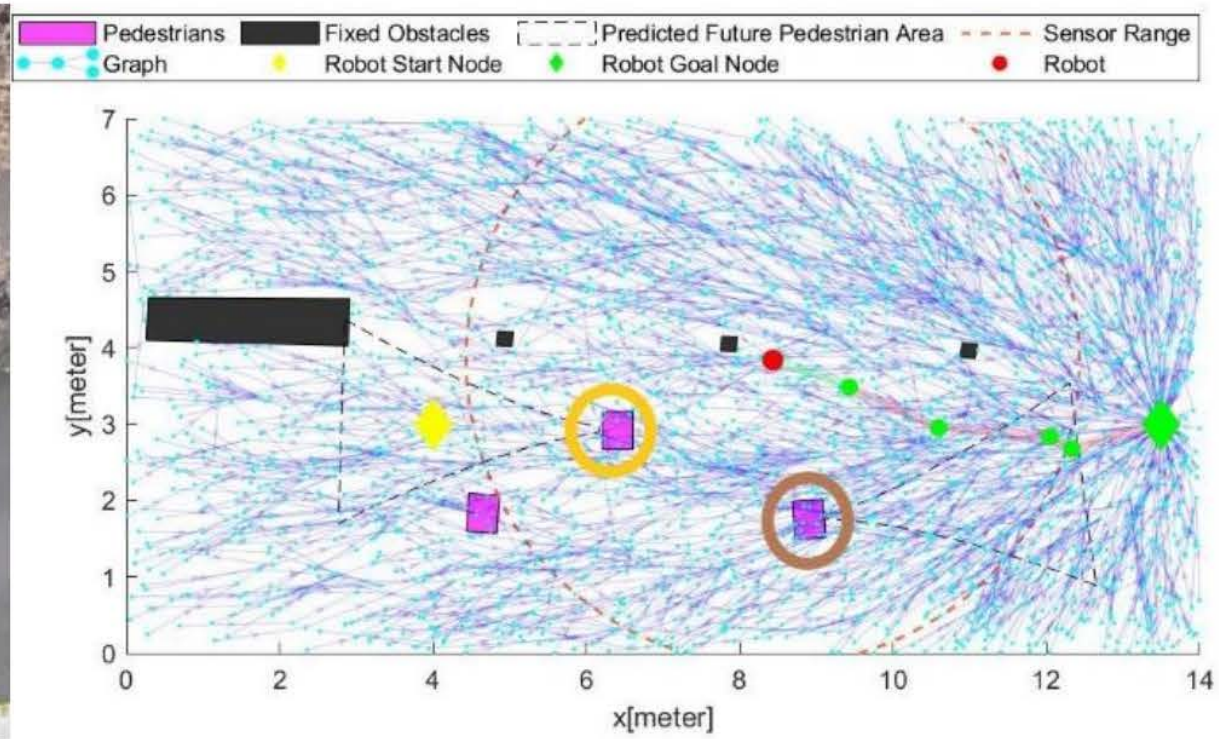




# Autonomous navigation of a personal mobility device in human crowded scenarios

## *Simulation results*





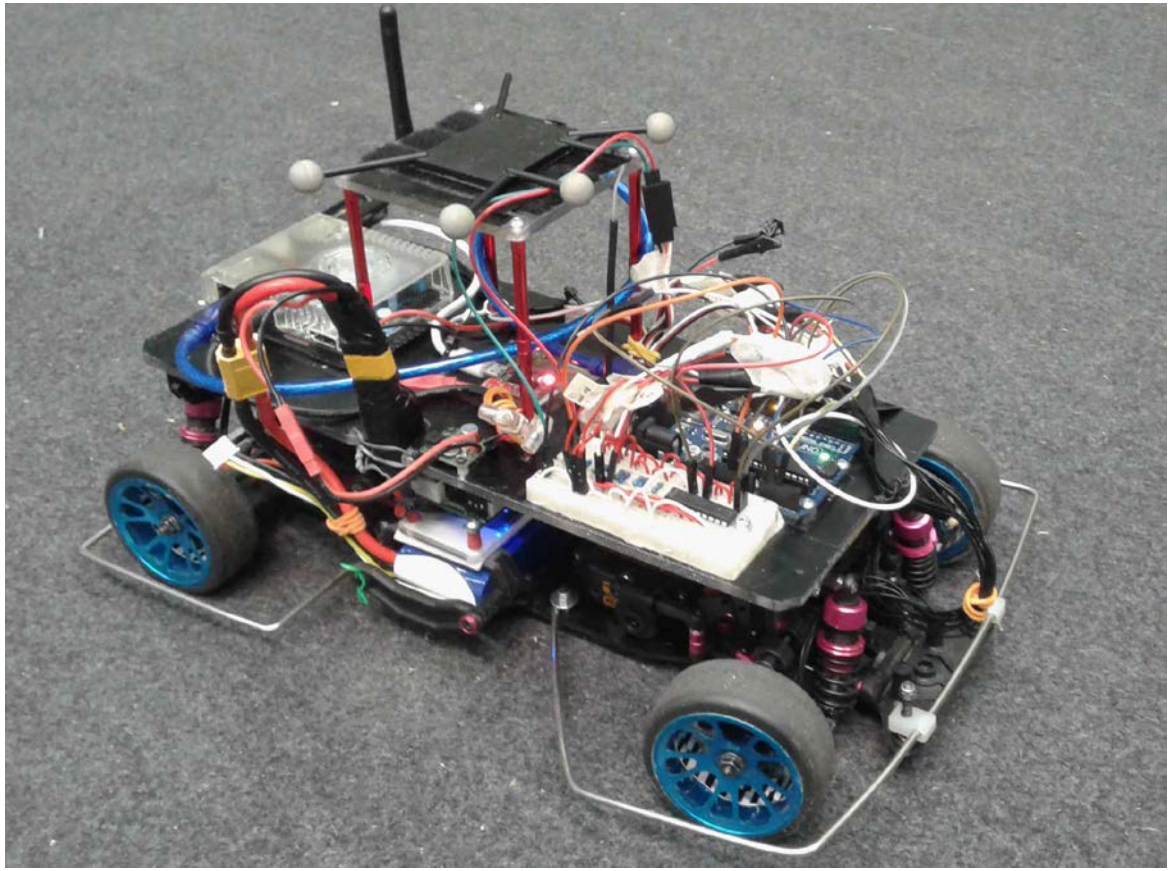
On the topic of autonomous navigation in human crowded scenarios there are still open research issues:

- human motion modelling for online prediction
- enforcing social constraints in navigation (planning and control) algorithms
- dealing with uncertainty in navigation (planning and control) algorithms
- efficient and optimal kinodynamic planning for dynamic environments

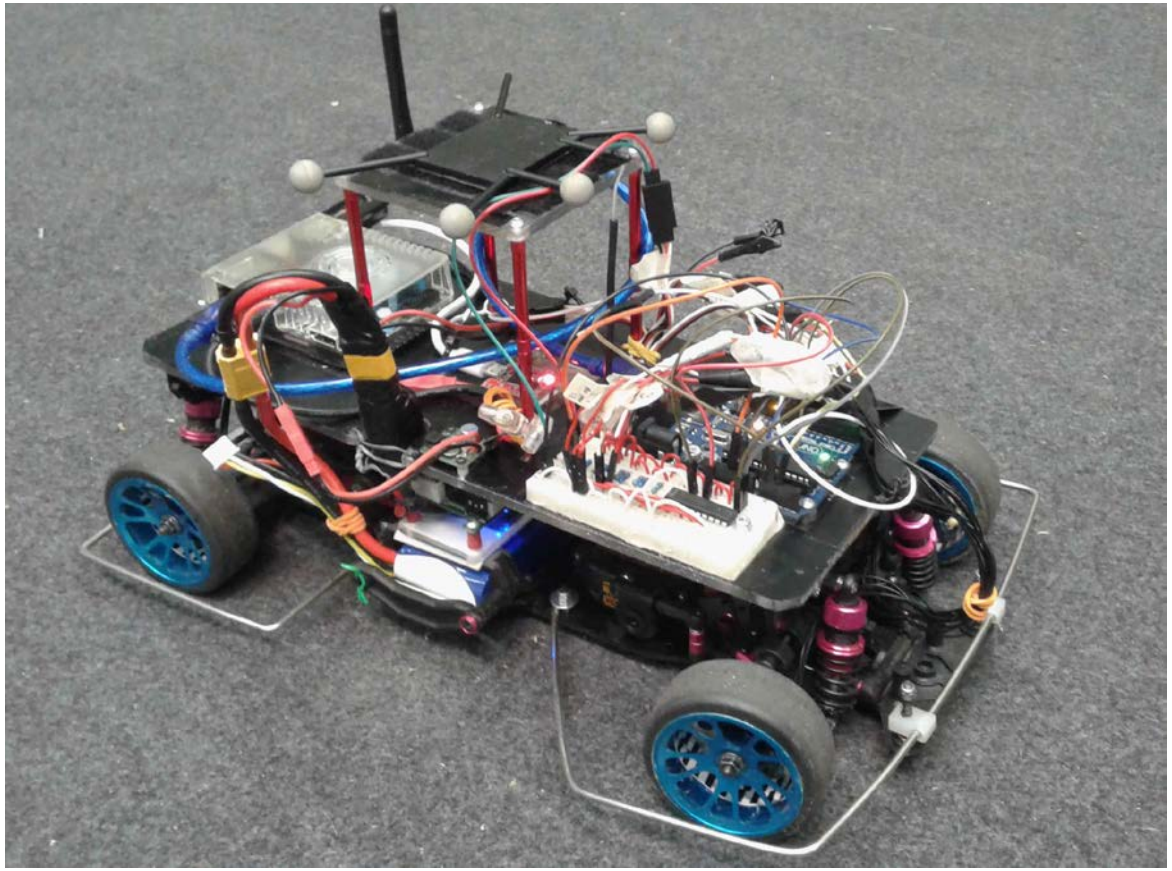
The aim is to devise innovative estimation and control techniques to support maneuvers at the limits of handling for an autonomous driving car.

These control techniques can be exploited to:

- increase performance of autonomous racing cars
- develop advanced ADAS systems to increase safety of commercial cars



- a 1:10 scale RWD car, actuated by a current / velocity controlled brushless motor and equipped with a steering servo
- an Odroid XU4 onboard PC, running ROS and executing the control algorithms
- an Arduino UNO, providing a bidirectional communication with steering and propulsion
- an IMU
- a 12-camera OptiTrack motion tracking system running at 100 Hz



The first step to set up the control system is the selection of the model that is used:

- to develop the control system
- to simulate the car for testing purposes

Driving at the limits of handling, due to large forces involved, calls for the adoption of a dynamic model.

Assuming that:

- roll and pitch motion
- lateral load transfer

are all negligible, we can adopt a single-track model.

We have also to be accurate in describing the tire-road interaction, as at the limits of handling the nonlinearity cannot be neglected.

It is convenient to use as state variables the longitudinal and lateral velocities, and the yaw rate

$$\dot{V}_x = \frac{F_x^R - F_y^F \sin(\delta)}{m} + rV_y$$

$$\dot{V}_y = \frac{F_y^F \cos(\delta) + F_y^R)}{m} - rV_x$$

$$\dot{r} = \frac{aF_y^F \cos(\delta) - bF_y^R}{J_z}$$

Front and rear lateral tire forces are modelled using a modified version of the Fiala model

$$F_y = \begin{cases} -C_\alpha z + \eta C_\alpha |z| z - \frac{1}{3}\eta^2 C_\alpha z^3 & |z| < \tan(\alpha_{sl}) \\ -\xi \mu F_z \mathrm{sgn}(\alpha) & |z| \geq \tan(\alpha_{sl}) \end{cases}$$

where

$$\alpha_{sl} = \arctan\left(\frac{1}{\eta}\right) \qquad z = \tan(\alpha) \qquad \eta = \frac{C_\alpha}{3\xi\mu F_z}$$

and

$$\alpha_F = \arctan\left(\frac{V_y + ar}{V_x}\right) - \delta$$

$$\alpha_R = \arctan\left(\frac{V_y - br}{V_x}\right)$$

Derating factor

$$\xi = \frac{\sqrt{(\mu F_z)^2 - F_x^2}}{\mu F_z}$$

Finally, the normal load on each tire can be computed as

$$F_z^F = mg\frac{b}{a+b} \qquad F_z^R = mg\frac{a}{a+b}$$

In order to set up a model-based design approach, we need to first identify model parameters.

The steering servo dynamics has been identified mounting the IMU on one of the front wheels and recording the yaw rate in response to a step of the angular servo position.

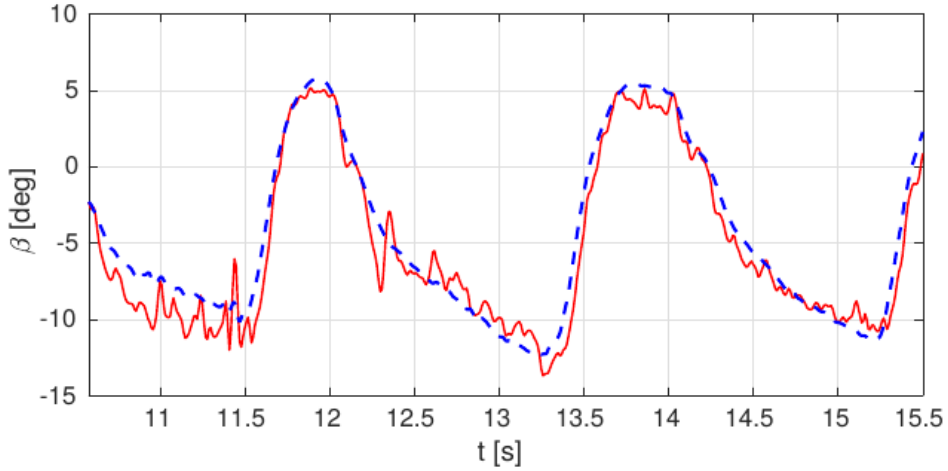
The actuator is described by a first order low-pass filter with

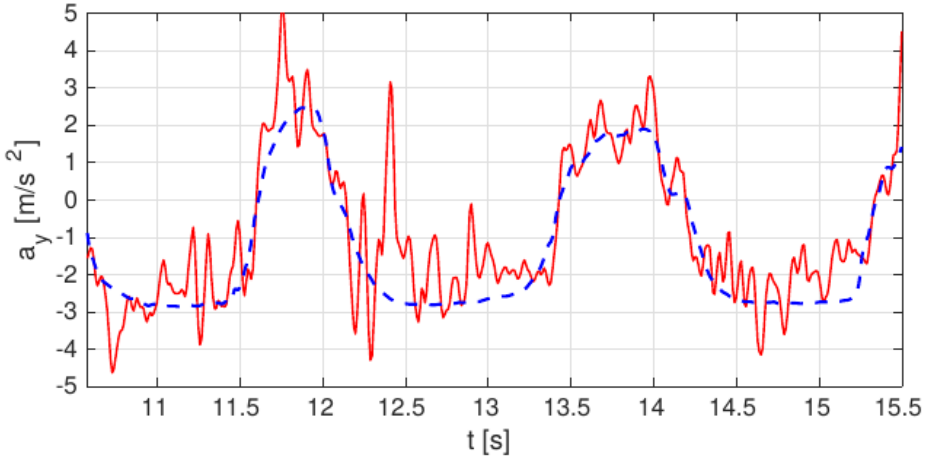
- a cut-off frequency of $50\ \mathrm{rad/s}$
- a delay of $0.04 - 0.06\ \mathrm{s}$

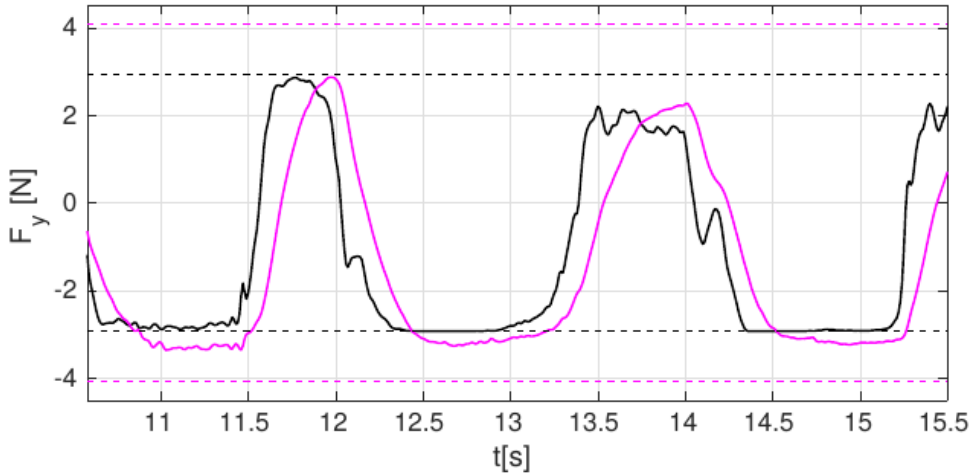
Vehicle mass and COG position have been measured with a weight balance.

The other parameters are estimated minimizing the error between measured yaw rate and lateral velocity, and the ones obtained by forward simulating vehicle dynamic model.

Car parameters have been identified using an eight-shaped trajectory.


10
5
0
-5
-10
-15
β [deg]
11
11.5
12
12.5
13
13.5
14
14.5
15
15.5
t [s]


5
4
3
2
1
0
-1
-2
-3
-4
-5
a_y [m/s^2]
11
11.5
12
12.5
13
13.5
14
14.5
15
15.5
t [s]


4
2
0
-2
-4
F_y [N]
11
11.5
12
12.5
13
13.5
14
14.5
15
15.5
t[s]

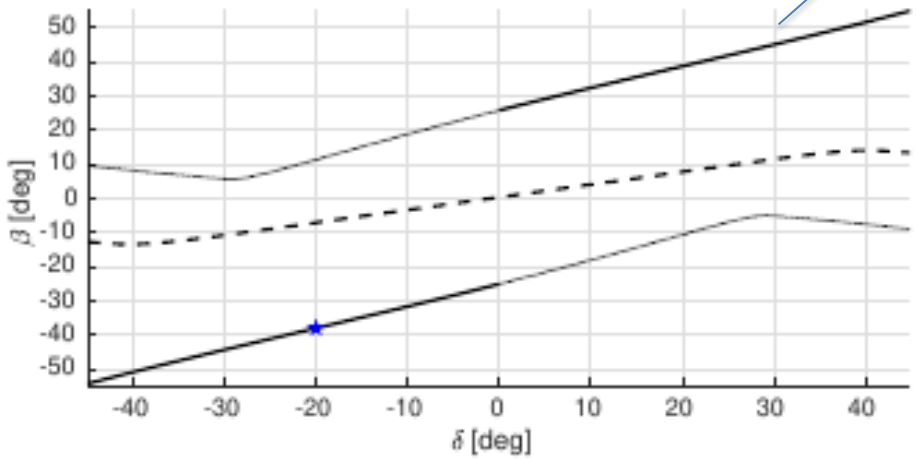
The aim of the controller is to stabilize the system around a drifting equilibrium point.

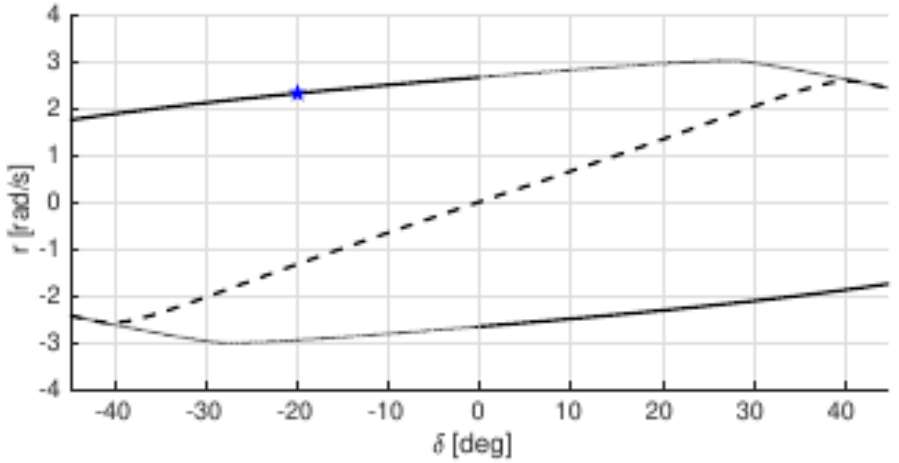
We thus start computing the equilibrium points of the single-track model assuming that

- longitudinal speed
- steering angle

are arbitrarily selected

Drifting equilibria



We select the star equilibrium point to linearize the system, and we set up an LQR problem with

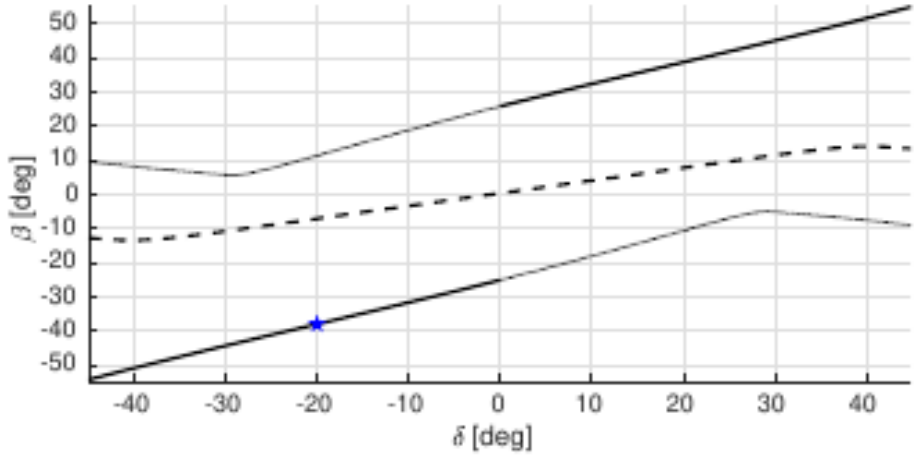
$$J = \int_0^{\infty} \left( \Delta \mathbf{x}^T Q \Delta \mathbf{x} + \Delta \mathbf{u}^T R \Delta \mathbf{u} \right) \mathrm{d}t$$

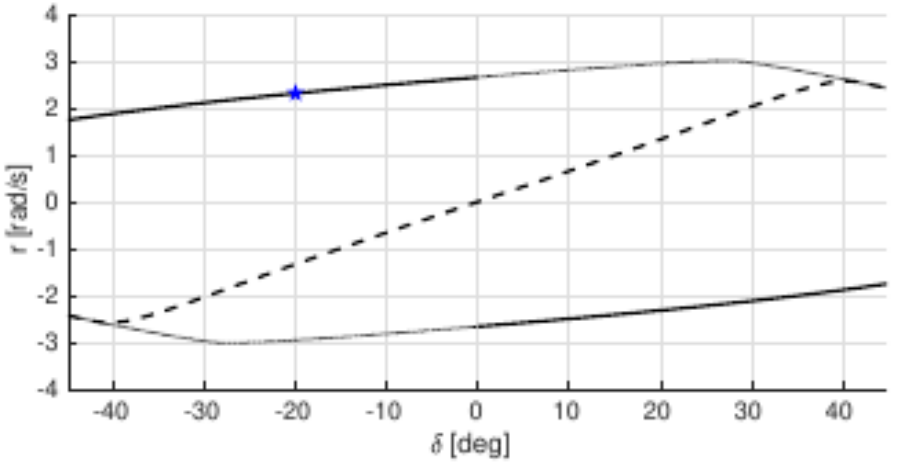
where

$$\Delta \mathbf{x} = \begin{bmatrix} \Delta V_x & \Delta V_y & \Delta r \end{bmatrix}^T \qquad \Delta \mathbf{u} = \begin{bmatrix} \Delta \delta & \Delta F_x^R \end{bmatrix}^T$$

and

$$Q = \mathrm{diag}\left( \frac{w_{\Delta x_i}}{\Delta x_{i,max}^2} \right) \qquad R = \mathrm{diag}\left( \frac{w_{\Delta u_j}}{\Delta u_{j,max}^2} \right)$$

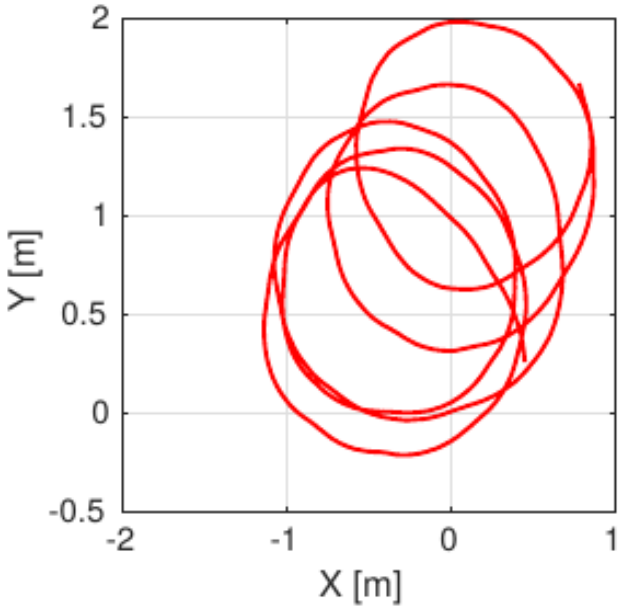


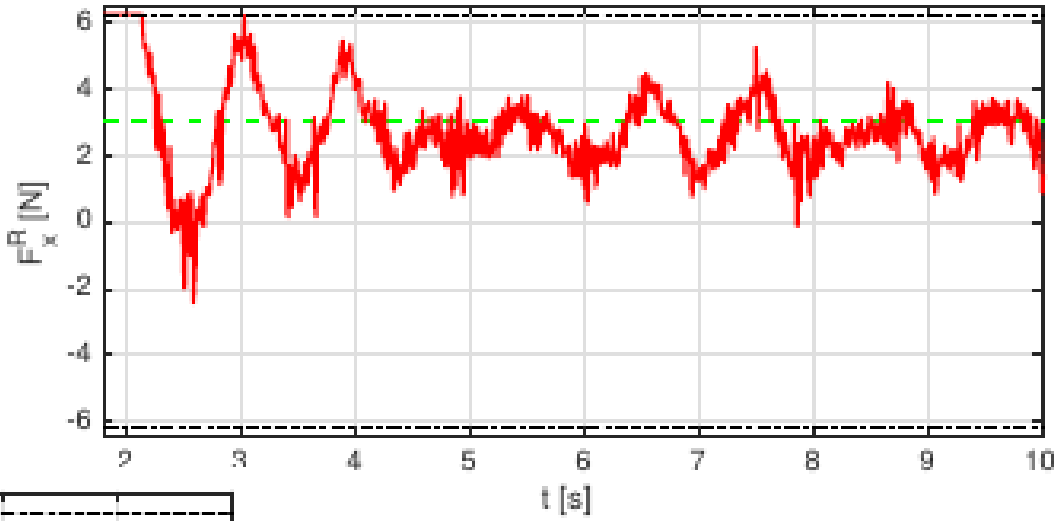



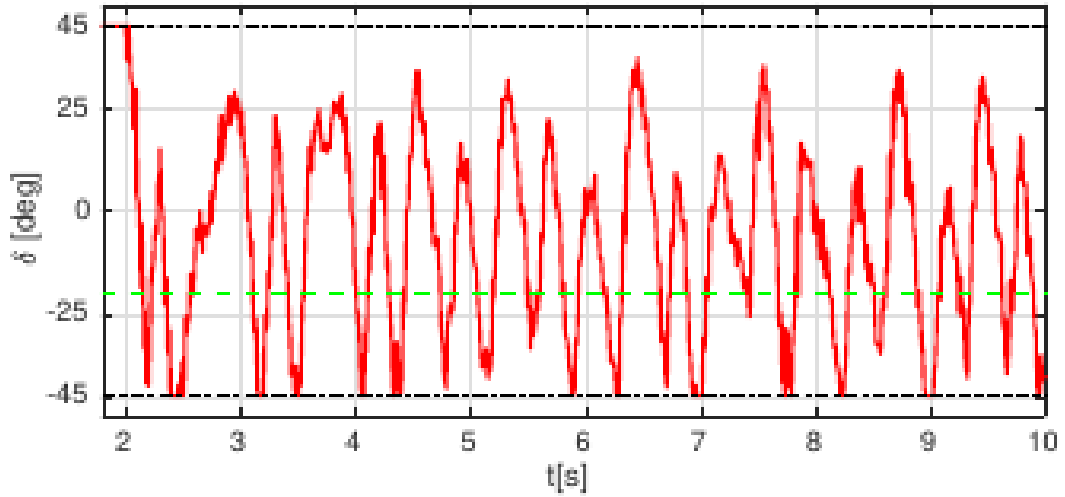
# Autonomous driving ad the limits of handling

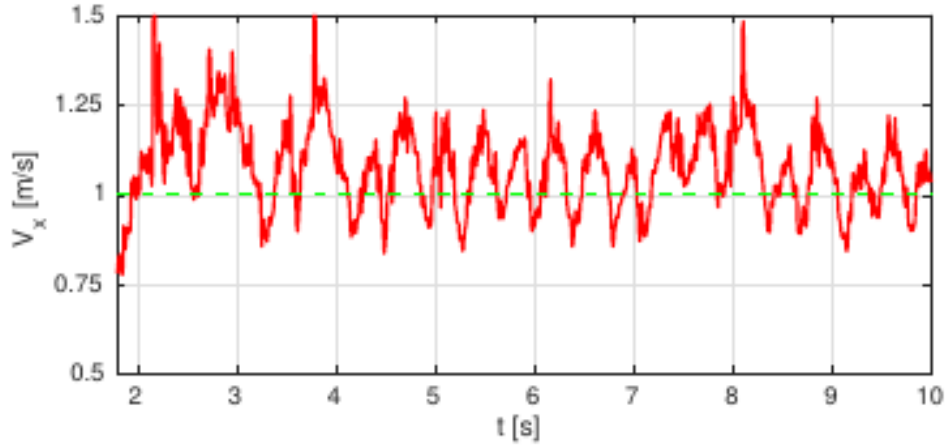
## *Experimental results*

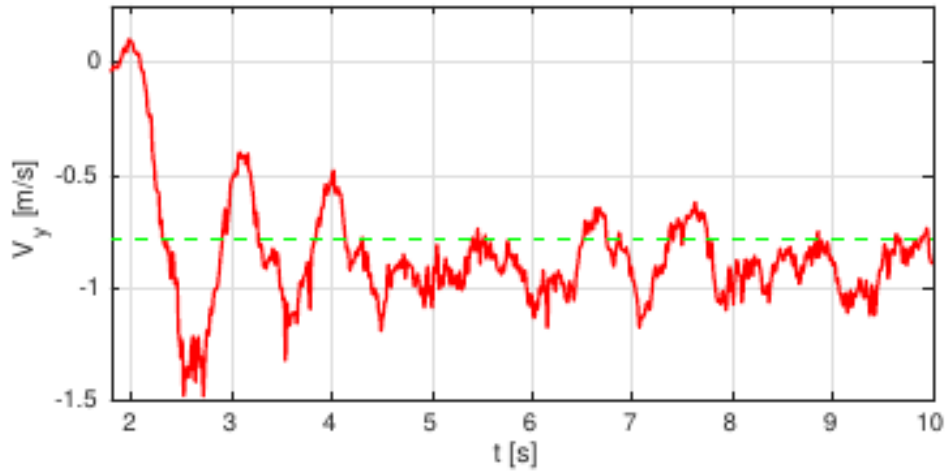


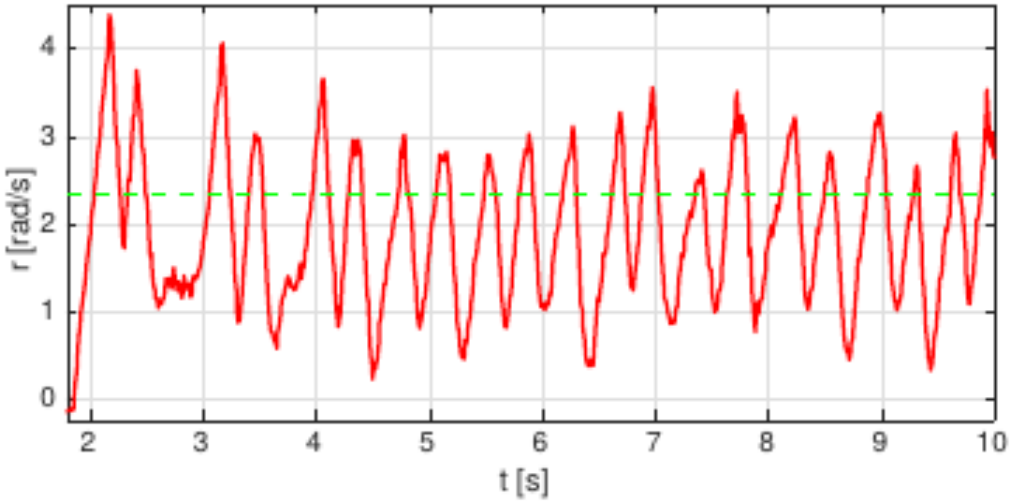



1.5
1.25
1
0.75
0.5
$V_x$ [m/s]
2
3
4
5
6
7
8
9
10
t [s]


0
-0.5
-1
-1.5
$V_y$ [m/s]
2
3
4
5
6
7
8
9
10
t [s]


4
3
2
1
0
r [rad/s]
2
3
4
5
6
7
8
9
10
t [s]

On the topic of autonomous driving at the limits of handling there are still open research issues:

- planning the trajectory of a racing car taking drifting manoeuvers into account
- performing trajectory tracking together with drifting stabilization
- online estimation / adaption of the tire-ground model
- increasing the performance of trajectory tracking using learning techniques

The aim is to develop a modular and flexible control framework allowing to control an indoor mobile manipulator as a whole robotic system for

- autonomous navigation
  - trajectory tracking
  - obstacle avoidance
  - tip/roll-over prevention
  - human safety and comfort
- execution of specific industrial tasks
  - control with exteroceptive sensors
  - priorities on the control of the different DOFs

- a Kuka Youbot composed of
  - an omnidirectional platform
  - a 5-DOF manipulator

In order to address different requirements coming from different aims (i.e., autonomous navigation, task execution) we should be able to change the way the controller computes the control action.

A solution is to adopt a MPC approach, where

- the cost function can be easily changed, changing the aim of the controller
- the constraints can be easily changed as well, allowing to represent the different requirements connected to the different behaviors

As usual we would like to setup a linear MPC problem, but either the kinematic/dynamic model of the manipulator or of the platform are (in general) nonlinear. In this specific case the platform is omnidirectional and it is thus described by a linear system.

In the design of the controller we enforce the following assumption

- the device is provided with low level current/velocity control loops

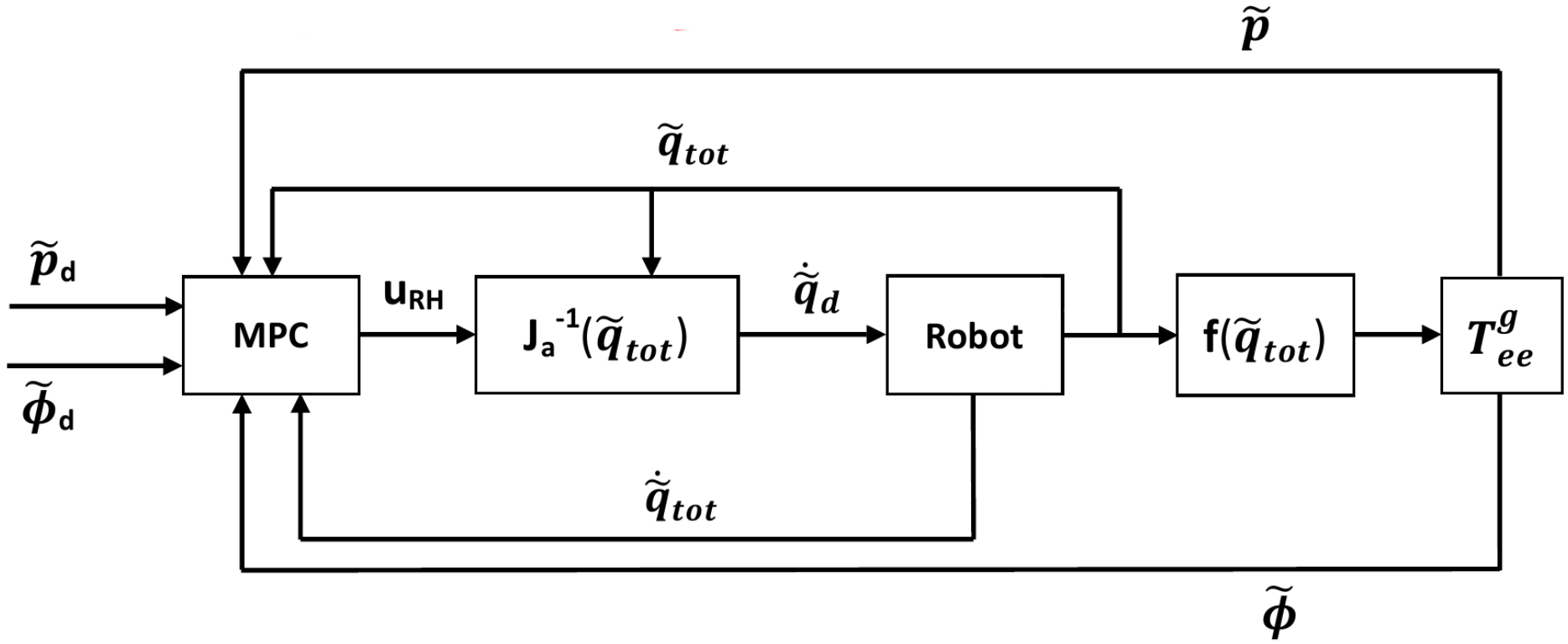
As a consequence

- we can adopt a cascaded control architecture where the MPC is the higher level whose outputs are the platform and joint velocities
- we can consider the kinematic models of the platform and the manipulator

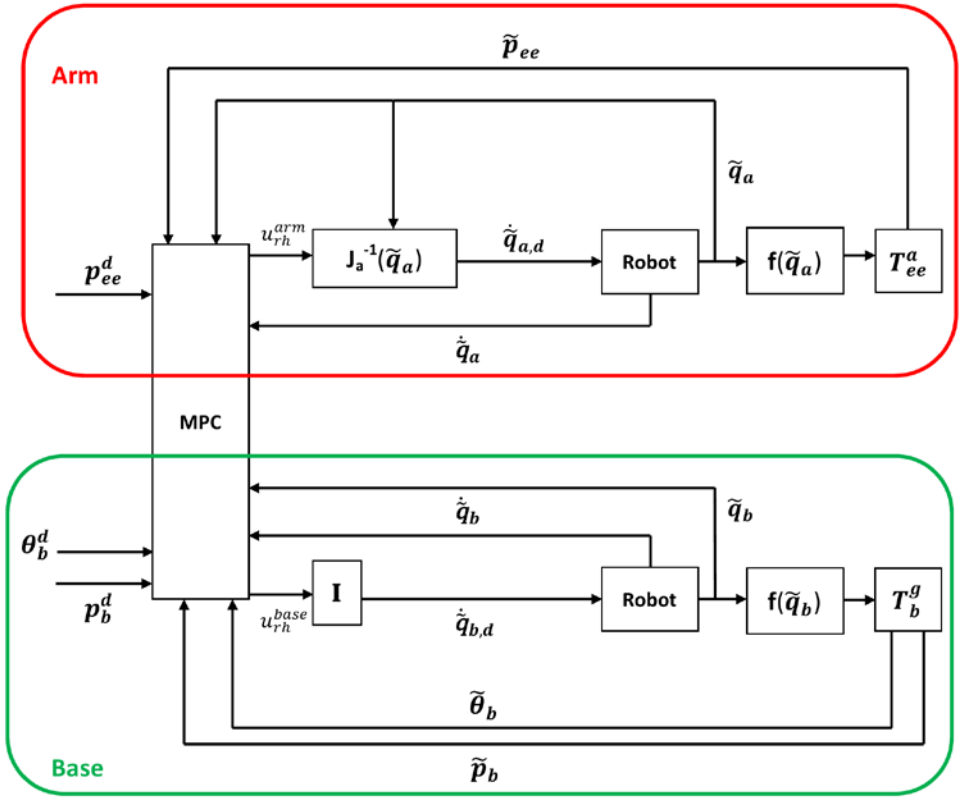
We further assume that

- during task execution mode, the platform and the arm are considered as a coupled system and the controlled variable is the end-effector pose
- during navigation mode, the platform and the arm are considered as decoupled systems and the controlled variables are the position of the end-effector and of the platform

Architecture of the task execution (coupled control) modality



Architecture of the navigation (decoupled control) modality



The cost function is

$$J(\mathbf{U}_t; \boldsymbol{\xi}_t, \boldsymbol{\xi}_{goal}) = \sum_{i=k}^{k+N-1} \left( \|\boldsymbol{\xi}_{i|t} - \boldsymbol{\xi}_{goal}\|^2_{\mathbf{Q}} + \|\mathbf{u}_{i|t}\|^2_{\mathbf{R}} \right) + \|\boldsymbol{\xi}_{k+N|t} - \boldsymbol{\xi}_{goal}\|^2_{\mathbf{S}}$$

where $\xi$ is the end-effector pose or the end-effector and base positions.

The cost function can be modified in order to consider a trajectory tracking problem and extended including the maximization of a manipulability index.

Constraint can be introduce for

- base and arm obstacle avoidance
- maximum joint position/velocity/acceleration constraints
- wheel maximum velocities
- platform stability