2014 IEEE International Conference on Robotics and Automation (ICRA 2014)

May 31 – June 7, 2014 Hong Kong, China

POLITECNICO DI MILANO

A Planner for All Terrain Vehicles on Unknown Rough Terrains based on the MPC Paradigm and D*-like Algorithm

Adnan Tahirović, Mehmed Brkić, Gianantonio Magnani and Luca Bascetta

Outline

- The All Terrain Robot
  - From the commercial ATV to the ATR
  - Control system components
  - Software & hardware architectures
- An MPC-based planner
- Simulation results
- Conclusion

From the commercial ATV to the ATR

From the commercial ATV to the ATR

Yamaha Grizzly 700

Main characteristics of the vehicle

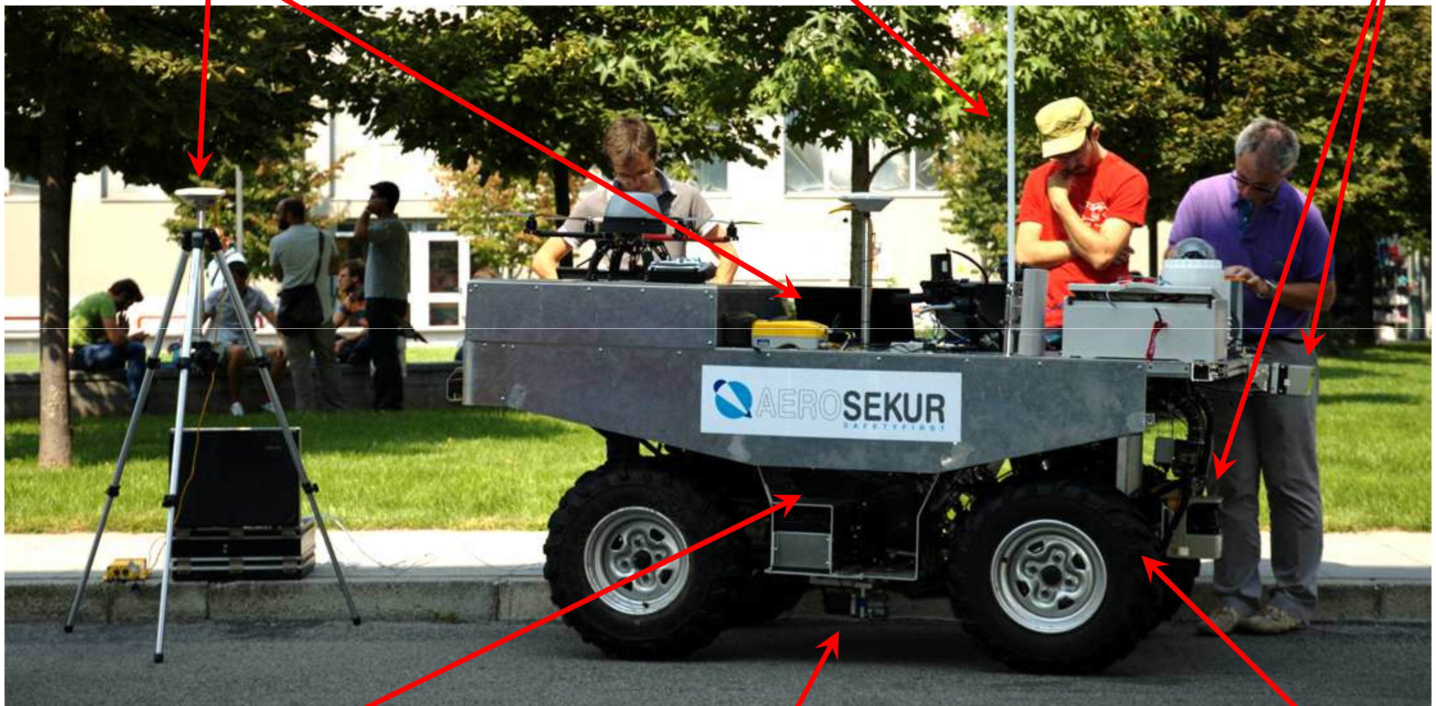
Engine type	686cc, 4-stroke, liquid-cooled, 4 valves
Drive train	2WD, 4WD, locked 4WD
Transmission	V-belt with all-wheel engine braking
Brakes	dual hydraulic disc (both f/r)
Suspensions	independent double wishbone (both f/r)
Steering System	Ackermann
Dimensions (LxWxH)	2.065 x 1.180 x 1.240 m
Weight	296 Kg (empty tank)

Sensors and low-level servosystems

Localization system

Wireless link

Obstacle detection

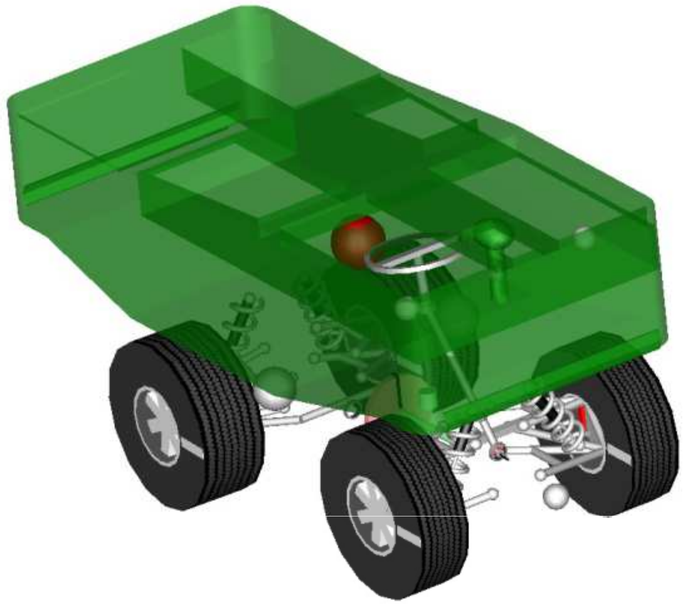


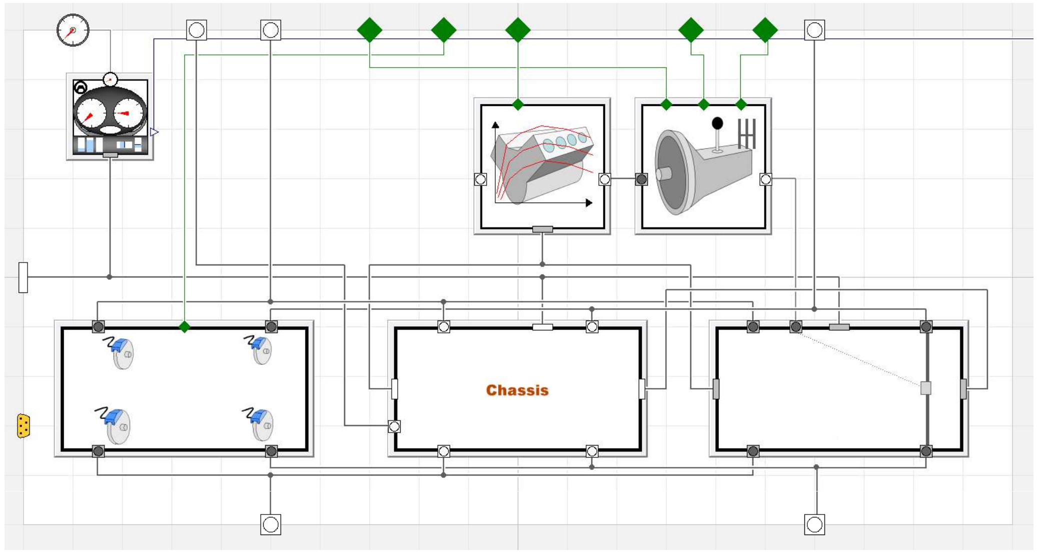
Brake control

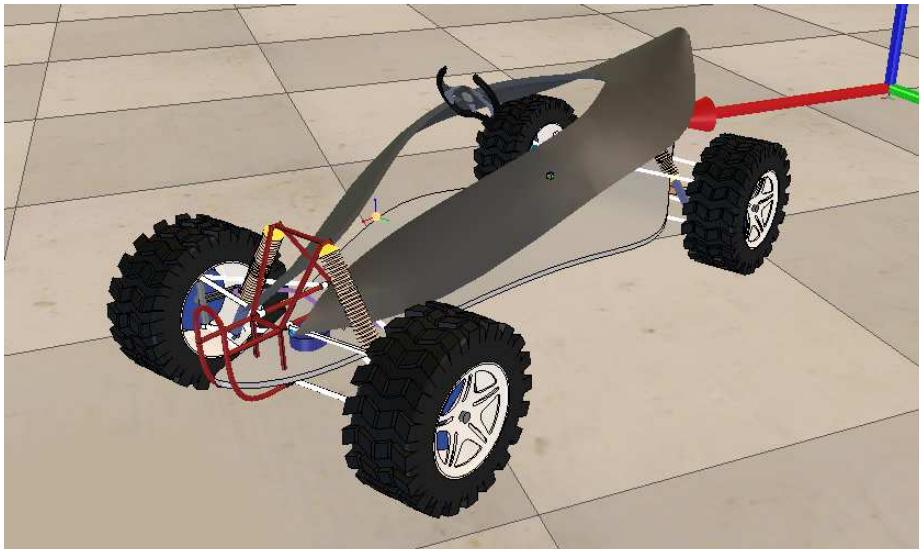
Throttle control

Steering control

ATR modelling and simulation





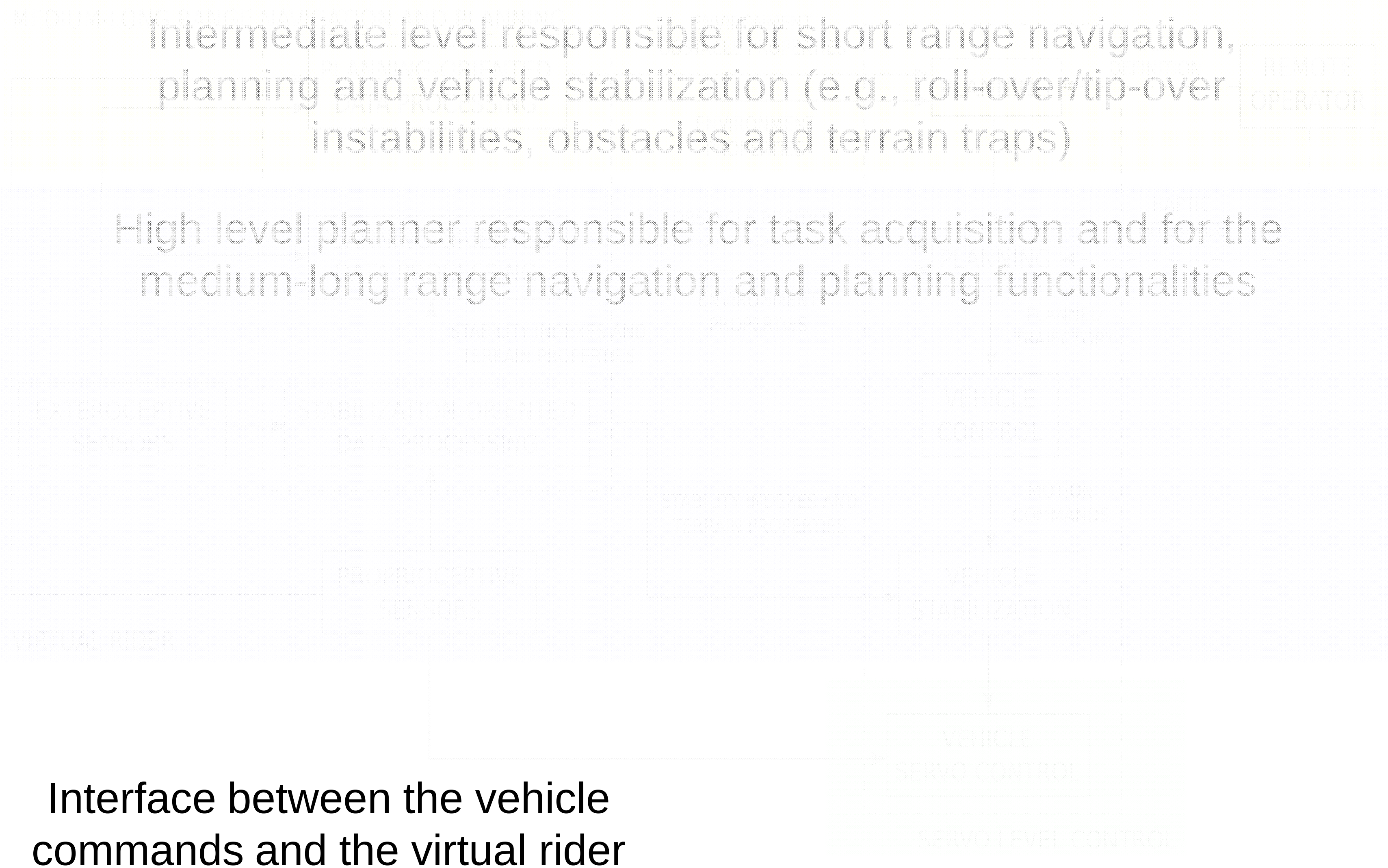


- Development of multi-body dynamic simulators for control system design and testing
- Development of simplified simulators for perception algorithm validation and software testing

Control system functional architecture

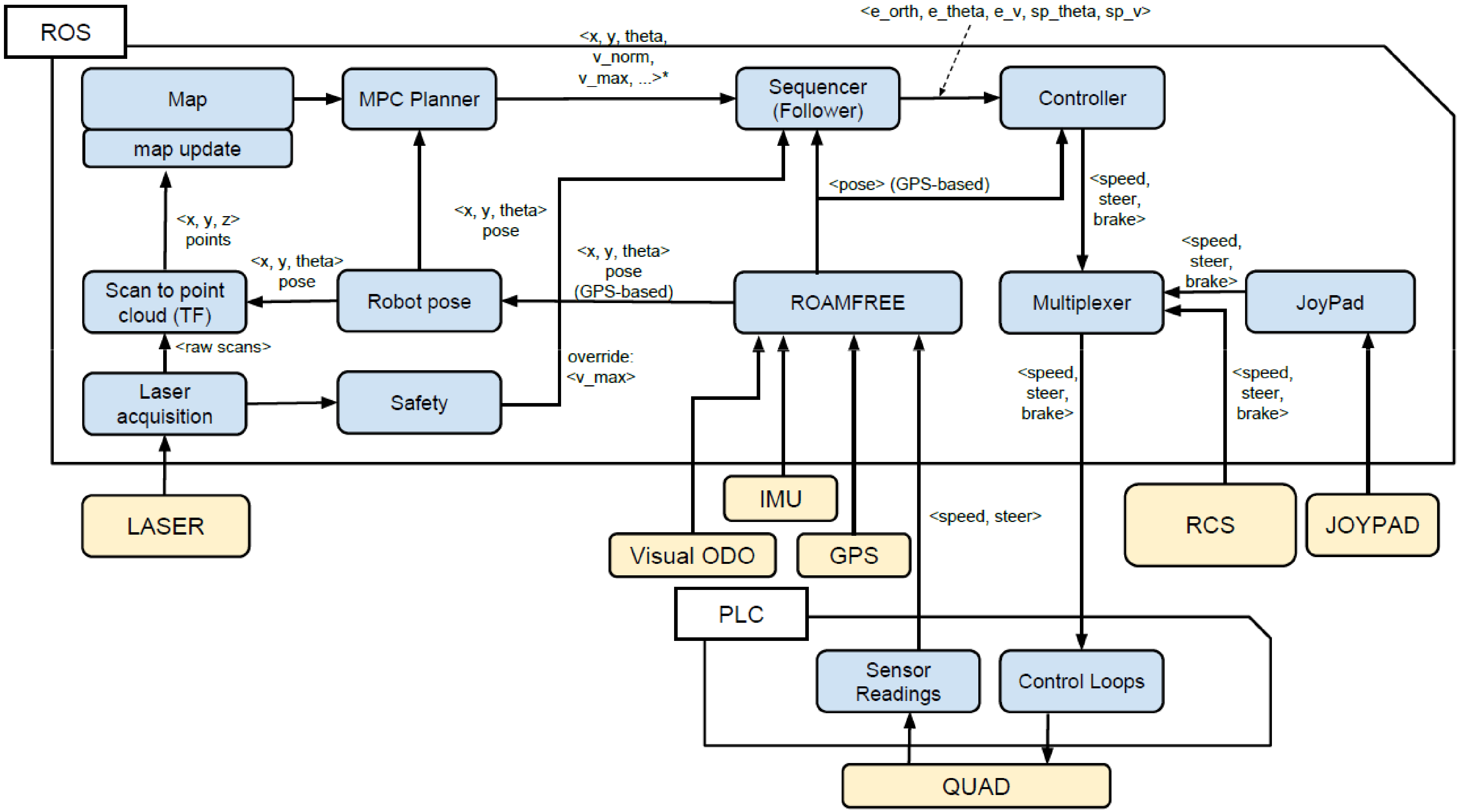
Intermediate level responsible for short range navigation, planning and vehicle stabilization (e.g., roll-over/tip-over instabilities, obstacles and terrain traps)

High level planner responsible for task acquisition and for the medium-long range navigation and planning functionalities

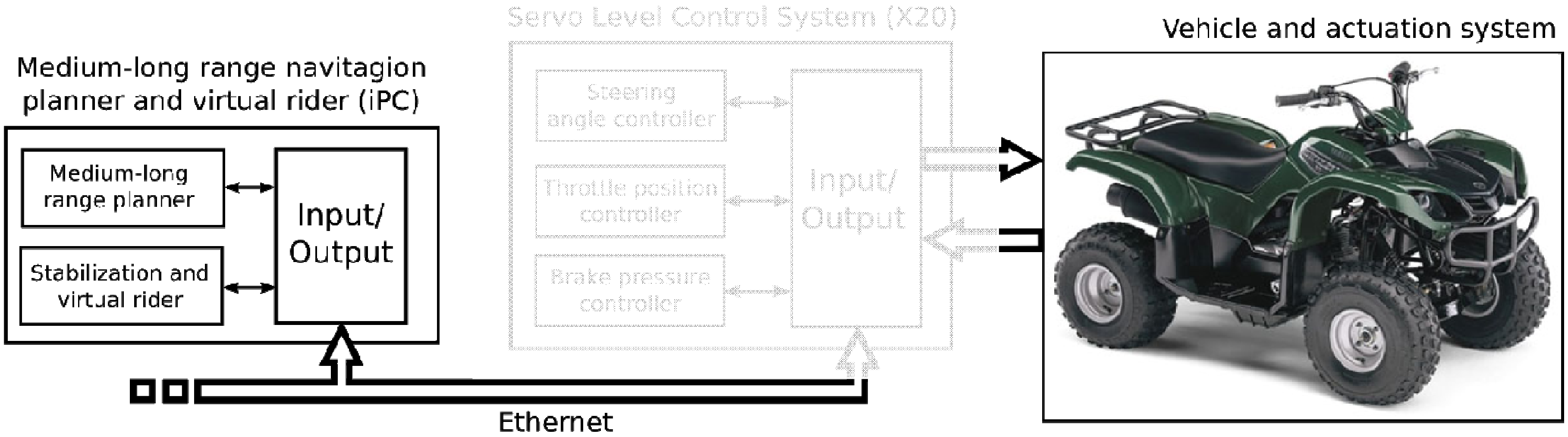


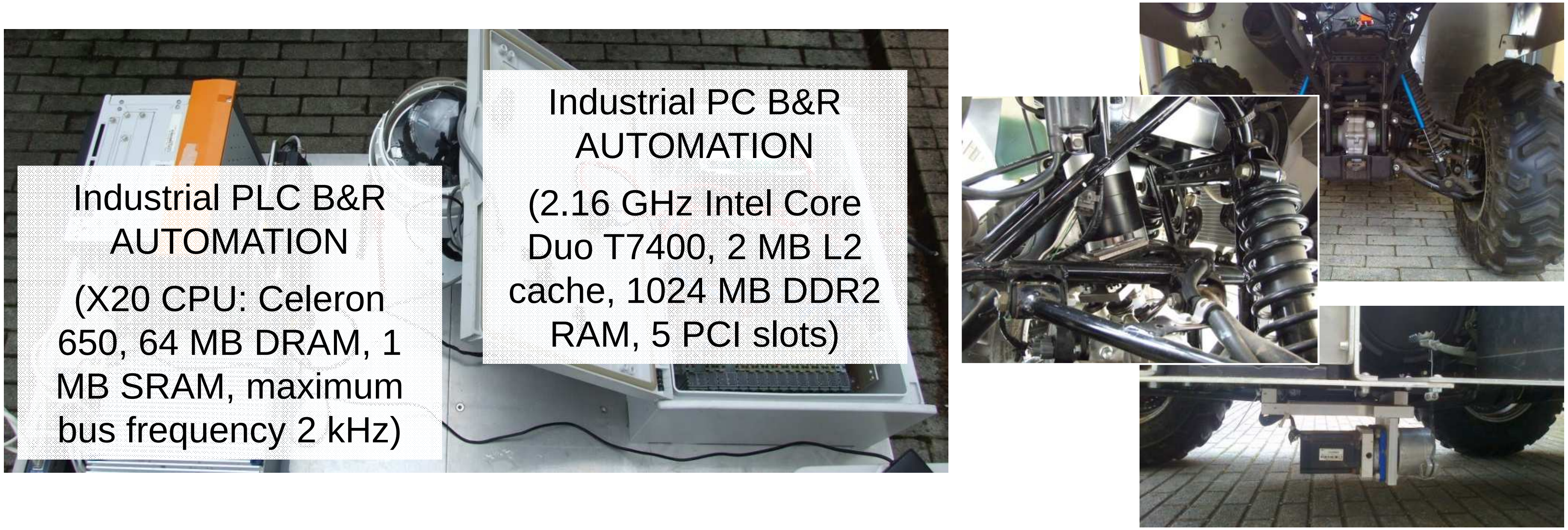
Interface between the vehicle commands and the virtual rider

Control system software architecture



Control system hardware architecture





Localization module: the ROAMFREE library

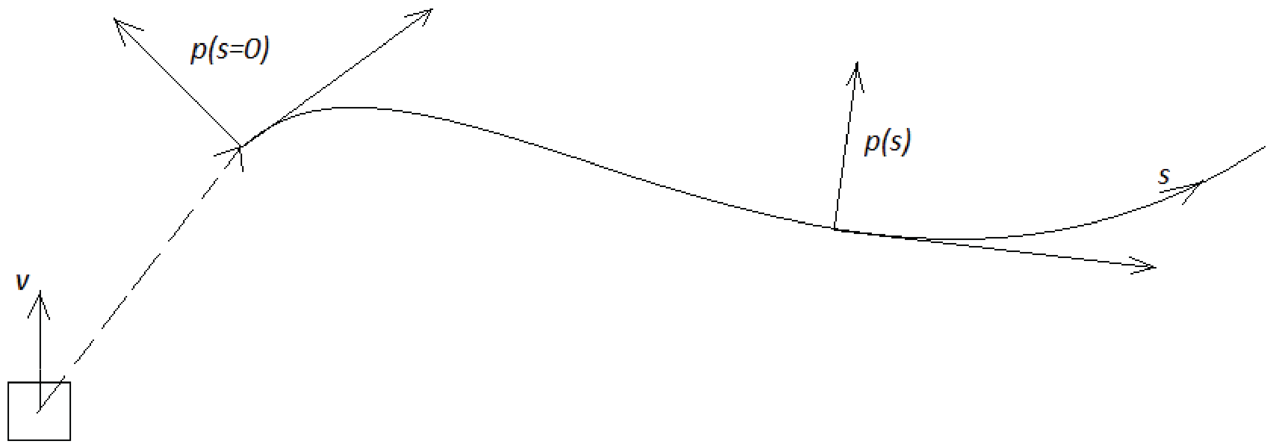
ROAMFREE sensor fusion library

- Ready to use library of sensors and kinematics
- 6-DOF accurate and robust pose tracking module
- Calibration suite for intrinsic sensor parameters
  (e.g.: sensors gains, biases, displacements, misalignments)

Core concepts

- Independence from physical hardware and robotic platform
- Turn-on-and go but flexible and extensible
- Optimized C++ core libraries / Python bindings
- ROS.org node available

Planar geometric path



Nonlinear control law taking into account the longitudinal velocity and the steering angle

$$v = \gamma e \quad \gamma > 0$$

$$\psi = \mathrm{atan}\left(\mathrm{L}\left(\frac{\sin\alpha}{\mathrm{e}} + h\frac{\theta}{e}\frac{\sin\alpha}{\alpha} + \beta\frac{\alpha}{e} \right)\right)$$

Autonomous navigation on rough terrains

... but the positioning is estimated by RTK-GPS ...

Why using MPC for path planning?

Planning for an all-terrain mobile robot to:

- move at high speed on rough terrains
- being safe w.r.t. obstacles
- ensuring vehicle stability

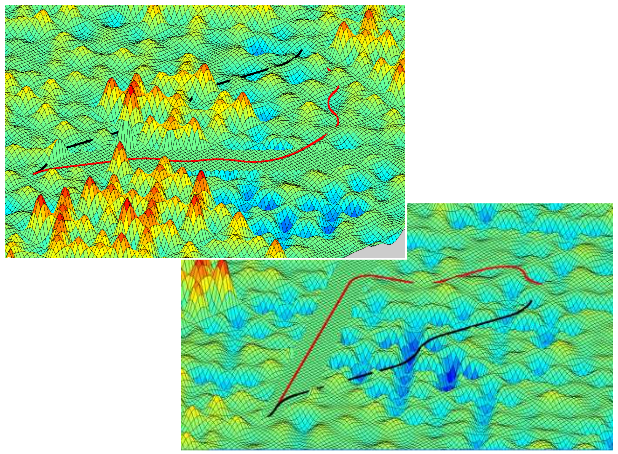
"A planner that does not take into account the vehicle model, might turn in a plan that fails due to the difference between real and planned paths"

Planning as an optimal control problem (OCP)

- impractical for real-time implementations

Planning using sampling-based techniques

- needs re-planning if something happens
- might be suboptimal



Why using MPC for path planning?

- to plan paths that are feasible for a real vehicle
- to generate plans that are near optimal when the cost-to-go function approaches the optimal cost-to-go map
- to develop a near optimal planner for the current information on “world state”
- to use any nonlinear vehicle model
- to impose any constraint (slipping, rolling, velocity, acceleration...)
- to deal with known and unknown terrains
- to use any planning technique which deals with unknown terrains to compute the cost-to-go function (here we use D*)

An MPC-based motion planner

At each time step, the planner finds the ***best*** local trajectory (within the sensor range) given the current vehicle state and terrain information

estimated local roughness

$$J(\mathbf{u}) = \int_{t_0}^{t_0+T} \gamma(\mathbf{x}, \mathbf{u})\,dt + \Gamma(t_0+T)$$

cost-to-go (roughness)

$$\frac{d}{dt}\mathbf{x} = f(\mathbf{x}) + g(\mathbf{x})\mathbf{u} \longleftarrow \text{vehicle kinematics/dynamics}$$

$$\mathbf{u}(t) \leq \mathbf{u}_{max} \longleftarrow \text{control constraints}$$

$$v(t_0+T) = 0 \longleftarrow \text{safe stopping criterion}$$

$$\Gamma(\mathbf{r}(t_0+T)) < \Gamma(\mathbf{r}(t_0+T_1)) < \Gamma(\mathbf{r}(t_0)) \longleftarrow \text{monotonicity constrain}$$

From known to unknown terrains

Previous works:

- known terrain
- Roughness based Navigation Function as a cost-to-go function for large scale terrains
- ACADO as an OCP solver for real-time implementation

Current work:

- partially and unknown terrain
- D* as a cost-to-go function for unknown terrains
- GPOPS as an OCP solver for real-time implementation

Computing the local roughness

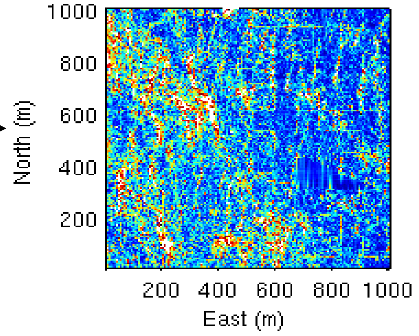
- A terrain map grid of regular square patches is created
- For each terrain patch a traversability measure is computed based on patch terrain elevation

$$\hat{\gamma}_{i,j}(p_{ij}) = \alpha(p_{ij}) \frac{\sqrt{var(z(\mathcal{R}_{ij}))}}{d}$$

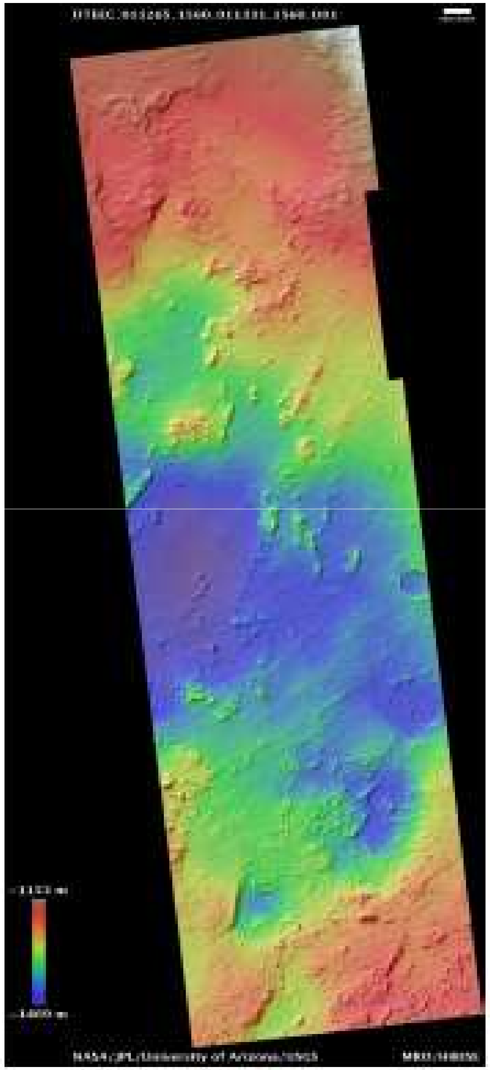
- For each terrain patch a roughness value is computed based on traversability measure, terrain friction, and obstacles (roughness map)

Hazard & Elevation Map → Roughness map → **Optimal roughness-to-go map obtained with D* algorithm**

Roughness map

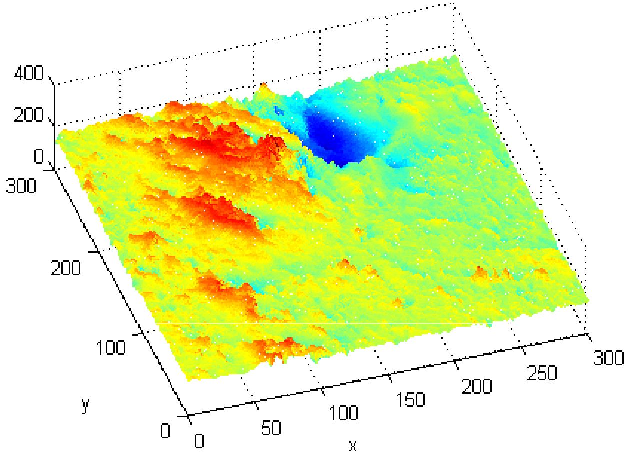


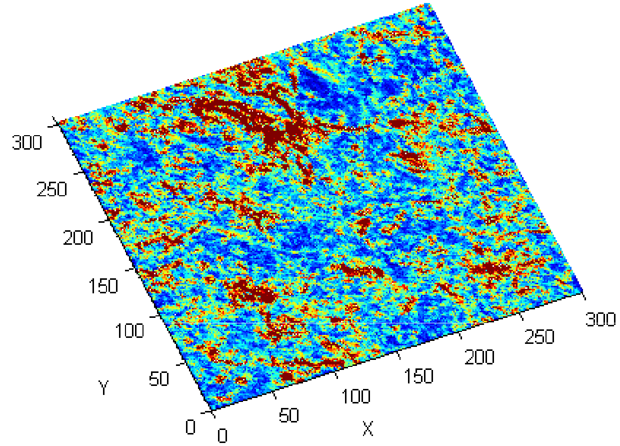
Optimal roughness-to-go map (I)

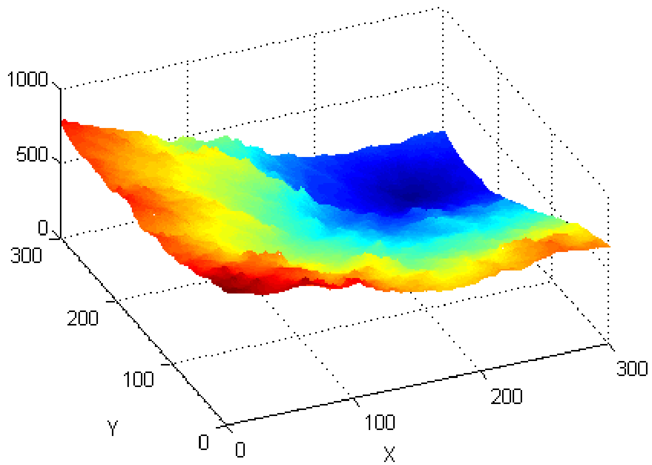


Martian Digital Terrain Map

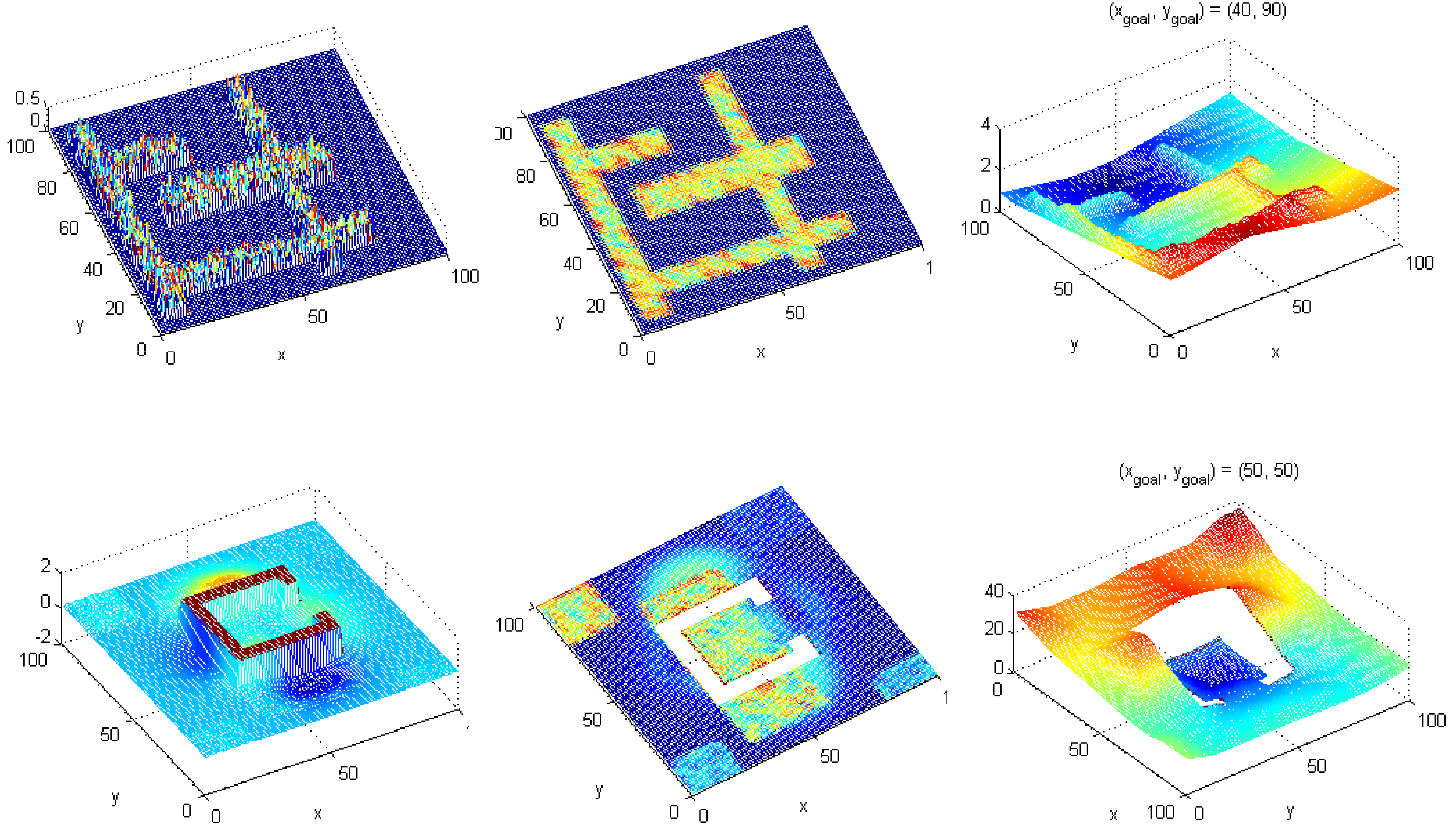
Example: roughness map taken from a portion of a Martian DTM and its optimal roughness-to-go map



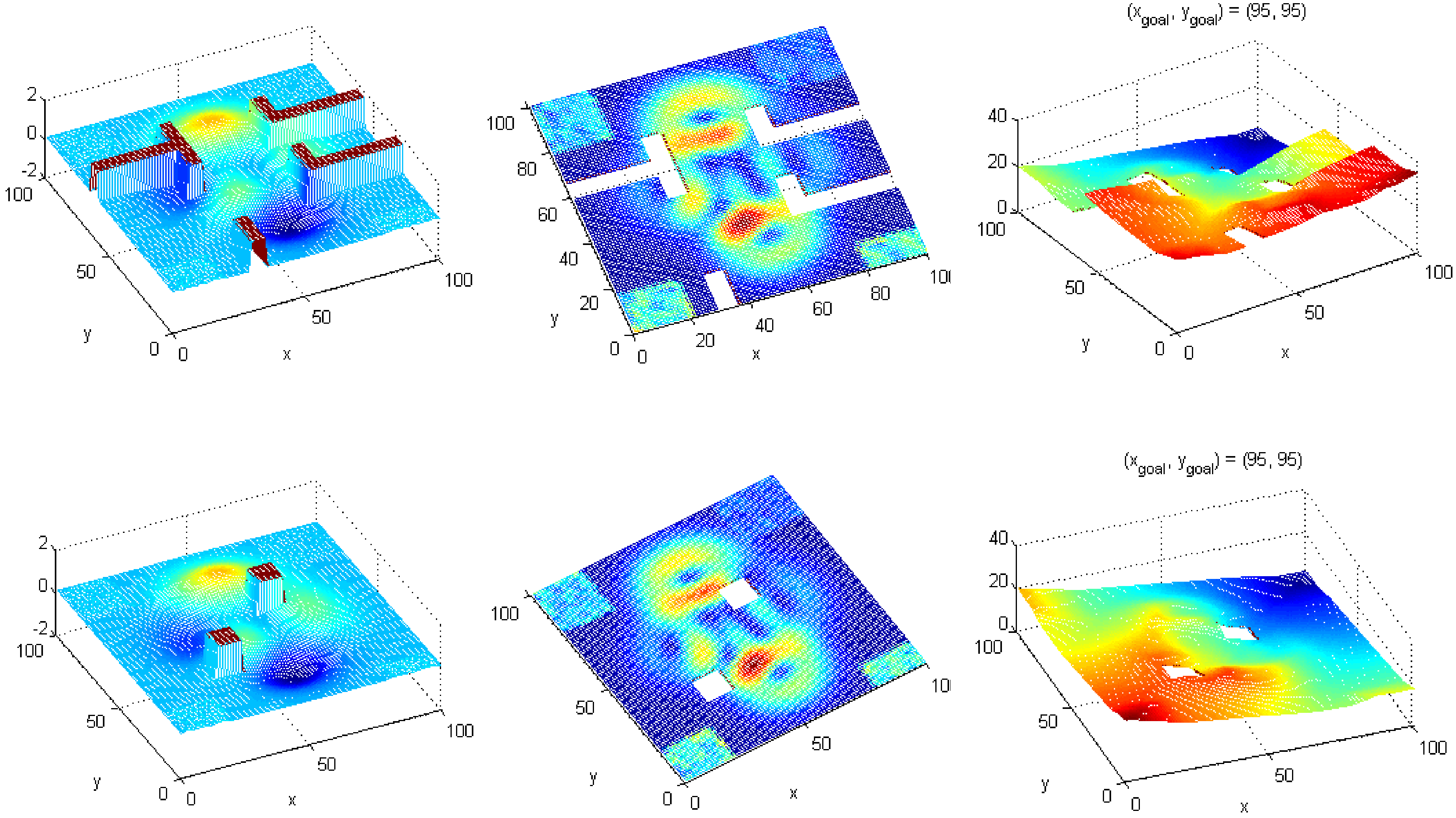




Optimal roughness-to-go map (II)

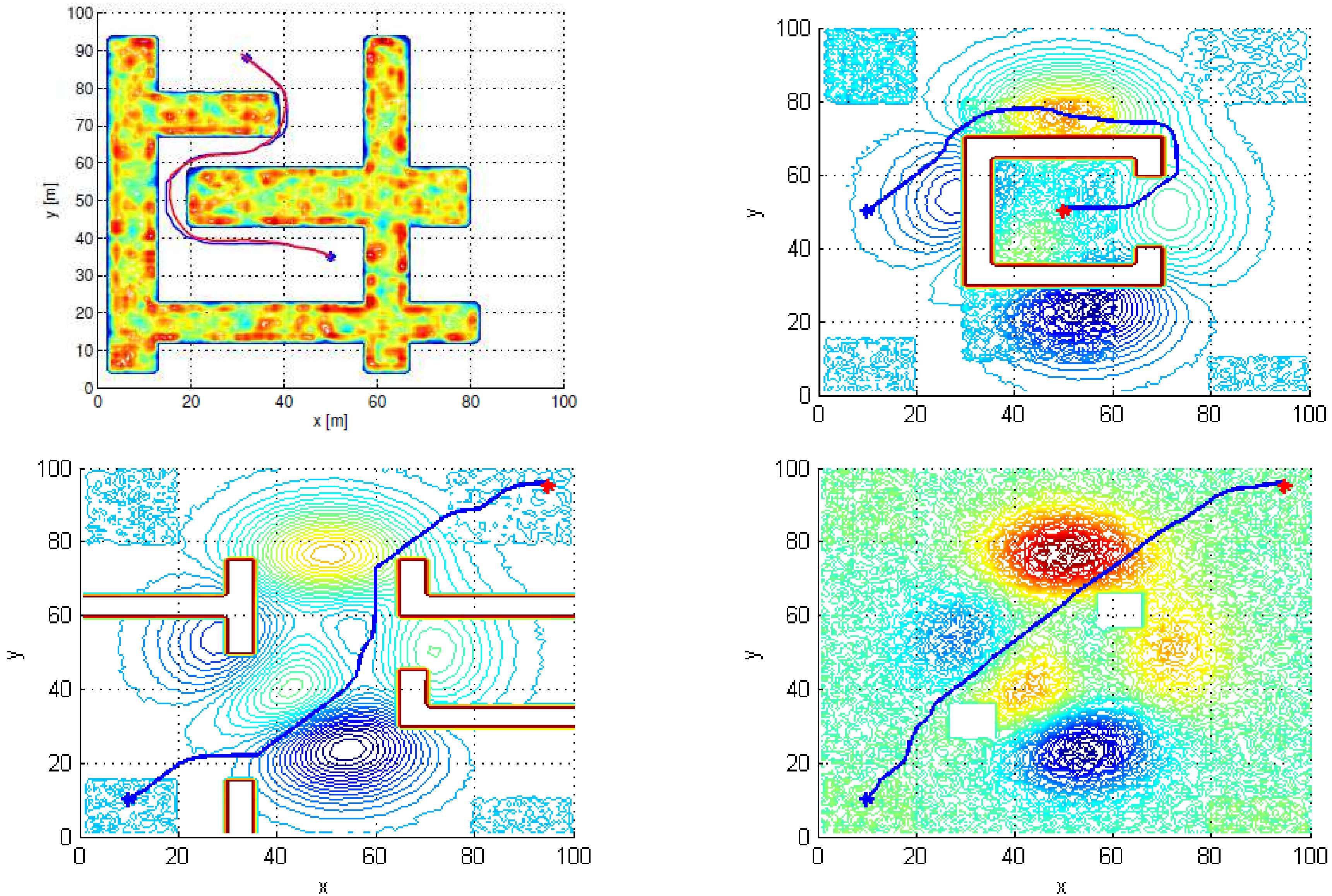


Optimal roughness-to-go map (III)



MPC-based planner on unwknown terrains

Simulation examples



Conclusion

Planning for an all-terrain mobile robot is a challenging task

- moving at high speed on rough terrains
- being safe w.r.t. obstacles
- ensuring vehicle stability

Doing it in real time is a more challenging task

- MPC by using GPOPS (Pseudospectral methods to solve an OCP)
- using the cost-to-go function by an interpolation of the cost-to-go map obtained by D*

Conclusion

The MPC-based planner has the following peculiarities:

- generates paths that are feasible for a real ATV vehicle
- can become near optimal when the cost-to-go function approaches the optimal cost-to-go map
- is a near optimal planner for the current information on “world state”
- allows to consider any nonlinear vehicle model
- allows to impose any constraint (slipping, rolling, velocity, acceleration, etc.)
- deals with known/unknown terrains
- can use any planning technique which deals with unknown terrains to compute the cost-to-go-function (here we use D*)