Towards the implementation of a MPC-based planner on an autonomous All-Terrain Vehicle

Luca Bascetta, Davide Cucci, Gianantonio Magnani, Matteo Matteucci,
Dinko Osmanković, Adnan Tahirović

Abstract— **Planning and control for a wheeled mobile robot are challenging problems when poorly traversable terrains, including dynamic obstacles, are considered. To accomplish a mission, the control system should firstly guarantee the vehicle integrity, for example with respect to possible roll-over/tip-over phenomena. A fundamental contribution to achieve this goal, however, comes from the planner as well. In fact, computing a path that takes into account the terrain traversability, the kinematic and dynamic vehicle constraints, and the presence of dynamic obstacles, is a first and crucial step towards ensuring the vehicle integrity.**
The present paper addresses some of the aforementioned issues, describing the hardware/software architecture of the planning and control system of an autonomous All-Terrain Mobile Robot and the implementation of a real-time path planner.

I. INTRODUCTION

The popularity of the research on wheeled mobile robots has been recently increasing, due to their possible use in different outdoor environments. Planetary explorations, search and rescue missions in hazardous areas [1], surveillance, humanitarian de-mining [2], as well as agriculture works such as pruning vine and fruit trees, represent possible applications for autonomous vehicles in natural environments. Differently from the case of indoor mobile robotics, where only flat terrains are considered, outdoor robotics deals with all possible natural terrains. The unstructured environment and the terrain roughness, including dynamic obstacles [3], and poorly traversable terrains, make the development of an autonomous vehicle a challenging problem.

The aim of our research is to develop an All-Terrain Mobile Robot (ATMR), based on a commercial All-Terrain Vehicle (ATV), that is suitable for a wide range of different outdoor operations. The ATMR should be able to operate in any natural environment with a high level of autonomy. The advantage of using ATVs is represented by their good traversability potential for poorly traversable terrains and by the short time spent for reaching the goal, as well as by the possibility to operate in unsafe environments. On the other hand, the main disadvantage of ATVs is their low stability

margin due to dynamic constraints, roll-over and excessive side slip [4].

ATVs are highly unstable, especially during fast turns and uphill/downhill riding, and a roll/tip-over can often occur. To overcome those problems the development of some active control systems [5], and in particular an Anti-Roll-over System [6], would certainly enhance their drivability. Moreover, it becomes necessary once the vehicle is teleoperated or autonomous. The design and development of an All-Terrain Mobile Robot is thus a challenging task, especially when a high level of autonomy is required. Indeed, due to the complex tasks the robot is supposed to perform, the design of the entire control architecture is anything but trivial [7]: different kind of requirements come from software engineering (e.g. modularity or maintainability), control theory (e.g. stability, robustness, hard real-time-ness) and mobile robotics (e.g. path planning, obstacle avoidance). The hardware/software architecture should fulfil them all in the simplest way.

A natural way to achieve those requirements is to design a multi-layered software architecture, in order to map higher levels of algorithmic abstraction to the top layers of the architecture. The control level that will act as an interface from these high level tasks (action planning, goal prioritisations, etc.) and the vehicle itself will be called "virtual rider". The aim of the virtual rider is to interpret commands from planner and execute them avoiding dangerous manoeuvres that could result in instability. Together with the virtual rider algorithm, a low level control software will be necessary in order to execute simple commands such as steering or braking.

All the aforementioned issues, crucial to ensure the vehicle integrity, can be addressed at two different levels. On one side, the virtual raider should operate in real-time to keep, as much as possible, the vehicle in a safe condition, or to recover it from dangerous situations. On the other, the planner plays a crucial role in computing a safe path, that a priori avoids dangerous manoeuvres.

The present paper describes the implementation and preliminary validation of a MPC-based planner that allows to compute in real-time a path from a starting to a goal position, taking into account obstacles, terrain characteristics and vehicle dynamic and kinematic constraints. The planner is implemented using an optimal control software (ACADO) to solve an initial value optimal control problem in receding horizon manner. Performance function and cost-to-go term are based on the terrain roughness. We locally interpolate roughness data at each time horizon with differentiable functions making it possible to use optimal control techniques

provided by ACADO.

II. THE ATMR

The vehicle considered in this research (see Figs. 1 and 2) is a YAMAHA GRIZZLY 700, a commercial fuel powered All-Terrain Vehicle (ATV) equipped with an electric power steering (EPS).

The GRIZZLY 700 is a utility ATV and is thus specifically designed for agriculture work. As a result it has a total load capacity of 130 Kg, and it is equipped with a rear tow hook. The main characteristics of the vehicle are listed in Table I.

Fig. 1. The Yamaha Grizzly 700 ATV

Fig. 2. The vehicle with the new cover

For the purposes of the project, the original vehicle cover has been removed and substituted with an aluminium cover, that allows to easily accommodate for the control hardware and the sensors (Fig. 2).

III. CONTROL SYSTEM ARCHITECTURE

In order to make the vehicle teleoperated, or even autonomous, an on-board hardware/software control platform

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)

TABLE I

VEHICLE CHARACTERISTICS

has to be added. While the implementation of the whole architecture is still under development, a functional diagram that shows the main components of the control system and their relationships is shown in Fig. 3.

The architecture can be divided into three different layers. The top level is a high level planner responsible for the task acquisition and for the medium-long range navigation and planning functionalities. The virtual rider is an intermediate level and is responsible for short range navigation, planning and vehicle stabilisation. It has to ensure vehicle integrity with respect to roll-over/tip-over instabilities, obstacles and terrain traps, etc., replacing the typical low-level riding skills of a human. The lower level represents an interface between the vehicle commands and the virtual rider. Such level interacts with the vehicle measuring the steering angle, throttle ratio, vehicle speed, etc., and acting on the steering column, the throttle leverage and/or the brake pedal through suitable sensors and actuation systems (see [8] for further details).

To implement such a complex architecture that includes high level and low level tasks, the former characterised by an heavy computational load but slower sampling frequencies, the latter being simpler but needing a faster time response, a multi-layered and multiprocessor hardware/software architecture is required. In this way, one can separate complex (localisation and navigation on rough terrains, obstacle avoidance, sensor fusion, etc.) from simple tasks (motion control and servo actuation) and faster from slower ones.

The hardware/software architecture should be as modular as possible, in order to be simply reconfigurable and upgradeable. Indeed, the different computational complexity of the tasks calls for different layers of the control system, thus a multiprocessor architecture is an obvious choice. On the other hand, the navigation control system requires the complete knowledge of the state of the vehicle (in terms of what the sensors are perceiving) to take the best decision autonomously. Thus a very large amount of data must be shared between the system's layers.

The selected hardware architecture (Fig. 4) consists of:

- a **low level** CPU (PLC) with several I/O modules to perform the control of the steering angle, the throttle position, the pressure of the hydraulic braking circuit, etc. An industrial PLC provided by B&R AUTOMATION (X20 CPU: Celeron 650, 64 MB DRAM, 1 MB SRAM, maximum bus frequency 2 *kHz*) was selected for its
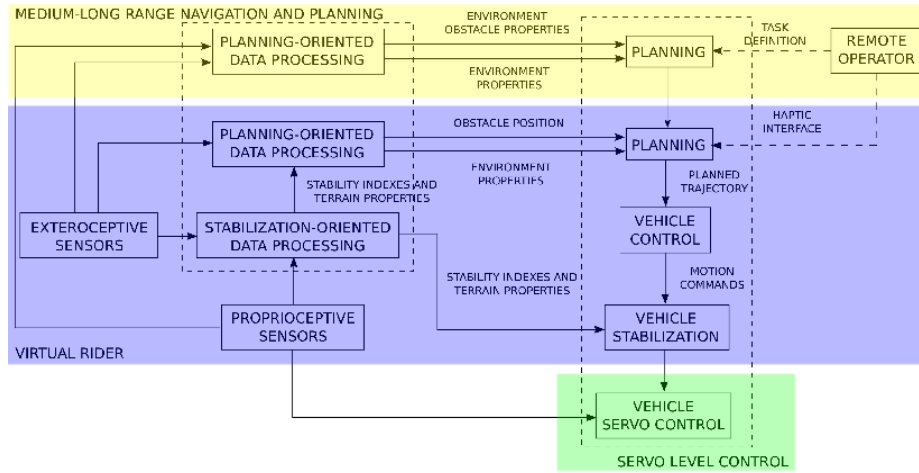
Fig. 3. Functional diagram of the controller architecture

dependability and robustness. Indeed, the choice of a PLC is a good compromise between the hard real-time requirement and the possibility of high level programming.

- a **high level** PC to implement the high level algorithms: the so called "virtual rider" (vision, terrain perception, localisation and mapping, obstacle and rollover avoidance, etc.), the medium-long range navigation an planning, etc. For this purpose, an Industrial PC (2.16 GHz Intel Core Duo T7400, 2 MB L2 cache, 1024 MB DDR2 RAM, 5 PCI slots) provided by B&R AUTOMATION was selected.

A standard Ethernet communication link was selected to connect the two CPUs.

IV. SOFTWARE ARCHITECTURE

Fig. 5 shows the main modules of the software architecture implemented on the ATMR (as previously introduced, the functional architecture presented in Section III has been only partially implemented).

In the upper part there are the modules running on the PC, while in the lower part the tasks running on the PLC. On the PC two different middle-wares have been used to implement the overall system: ROS [9] and OROCOS [10]. While the former provides useful functionalities out of the box (e.g., laser sensor acquisition, mapping and planning) and thus it helps in speeding up the development, the latter has been used for critical control tasks having hard real-time requirements.

As already stated, the PLC runs the low-level actuator control loops and the sensor acquisition functionalities (e.g., speed, steering angle, stability indexes, etc.). The set points are sent to the low-level control loops (i.e., *speed*, *steer*, and *brake*) by the OROCOS task named MULTIPLEXER, which has the role of deciding whether the ATMR should be teleoperated, i.e. guided by a wireless JOYPAD, or a REMOTE CONTROL STATION (RCS), or autonomous, i.e. the CONTROLLER is in charge of trajectory following.

A simple trajectory follower has been implemented, decoupling the geometrical path following, that is accomplished acting on the steering angle, from the speed control. Following this idea, two independent PID control loops have been realised: one controlling the steering angle on the basis of the vehicle alignment and distance error [11], computed by the SEQUENCER module, the other one regulating the vehicle speed.

The ATMR position is estimated by an EXTENDED KALMAN FILTER (EKF) that uses the Ackerman kinematic model and integrates speed and steer measurements from the ATV sensors together with the position provided by a RTK-GPS with external correction (up to few centimetres accuracy). At the present stage, the magnetometer measurements of an inertial measurement unit are used to initialise the EKF heading estimate, but we plan to integrate them in the EKF once a proper dynamic model of the vehicle is developed.

The pose computed by the EKF module is also provided to the modules implemented under ROS, and it is used to align the point clouds acquired by a Sick LD-MRS laser range finder with the map of the environment.

This map in turn is used by a Model Predictive Control (MPC) based planner to generate the desired trajectory for the ATMR. This task is performed by a set of ROS nodes since most of the routines where already available under that middle-ware and the loose real-time requirements of planning were satisfied by ROS scheduling. It should be noticed that planning is a critical aspect when moving in rough terrains since, by carefully taking into account the constraints of the vehicle, safe trajectory can be planned. The result of this planning activity is then fed to the SEQUENCER module to be executed under real-time conditions.

When navigating an unknown environment unexpected, or unmapped, obstacles might appear; in this case the map need to be updated and the planning activity re-executed to take into account the new information. In our case the MPC based planner is re-executed continuously so this map update is managed in a natural way. However, MPC planning might
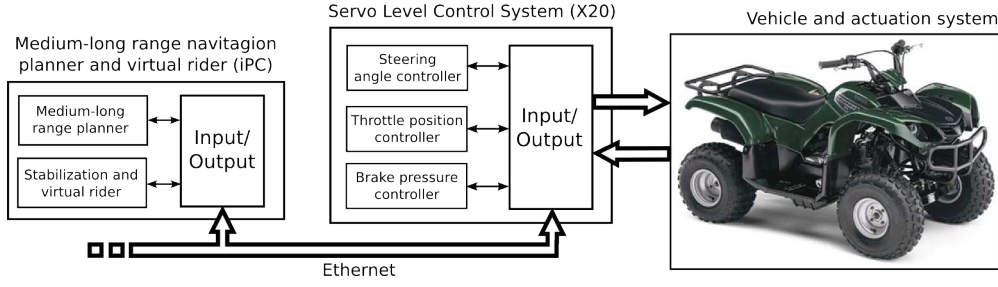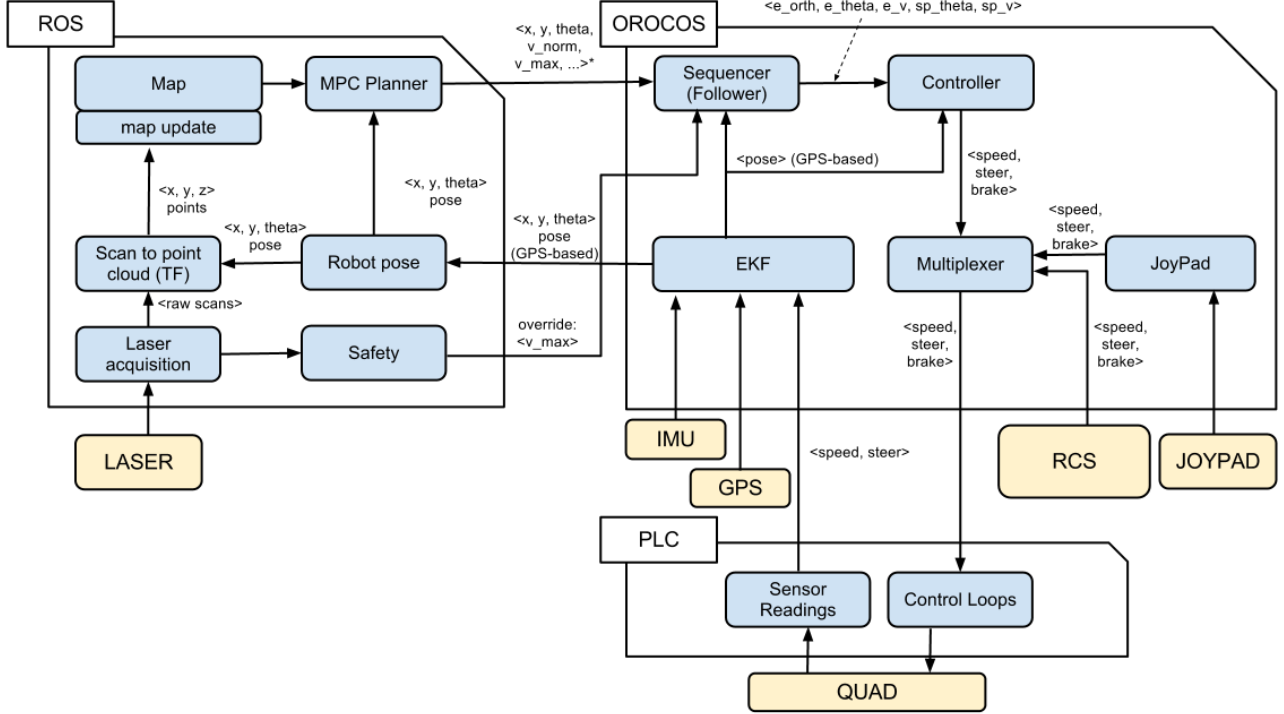
Fig. 4.   Hardware architecture

Fig. 5.   Software architecture

take some time to compute a new plan or the computation can even fail. To cope with this possibility, a SAFETY module that, observing the point cloud generated by the sensor, overrides the maximum speed allowed for trajectory following has been introduced.

V. A MPC-BASED PLANNER

Including the vehicle model into the motion planning stage provides a planner which generates trajectories that can be easily followed by a mobile robot. This especially comes to the fore when a vehicle moves with high speed and operates on rough terrains. Using a simpler planner that does not take into account the mobile vehicle model might cause a fatal error due to the difference between the planned and executed trajectories. For this reason, the gradient based algorithms such as the navigation function or a variant of the D^* [12], [13], [14], in our case are not considered an

acceptable solution.

Finding an optimal path on rough terrains, given a vehicle model and all information about the terrain, can be expressed as a two point boundary value optimal control problem (OCP). Including the terrain shape into an objective function for the OCP might result into a problem difficult to solve. Namely, the OCP softwares, including ACADO [15], the software used in this work, require a differentiable objective function. To overcome this problem, a kind of interpolation of the terrain shape must be applied. However, such an interpolation might be computationally intensive even for medium size terrains, and finding the best path solving an OCP might be impractical for real-time implementation.

The approaches [16], [17], [18], [19], [20], all consider the vehicle model to find the final path from an initial to the goal position. They use an appropriately selected state-space sampling technique, in which a planner propagates the

vehicle model over these states toward the goal position. However, these approaches might easily miss some key state spaces yielding a solution being far from optimal. If the vehicle discovers different information during the execution, these approaches re-plan from scratch finding a new path to the goal position. In case of uncertain terrains, a frequent complete replanning makes it difficult to use the approach for real-time implementation.

In this work, we use an adapted real-time Model Predictive Control (MPC) based motion planner, introduced in [21] and [22]. At each time sample, the planner finds the best local trajectory (within the sensor range) given the current vehicle state and terrain information. Such an "on-line" optimisation during the task execution is in accordance with the MPC approach, hence the name. The MPC based motion planner easily accommodates for a vehicle model and any form of constraints into the optimisation set-up. In [21], the optimization has been performed using genetic algorithms in order to cover the control space of the vehicle model and to find the best solution at each time sample. In this paper, the objective function and the cost-to-go term are based on the terrain roughness. We locally interpolate roughness data (within the vehicle sensor range) at each time horizon into differentiable functions making it possible to use optimal control techniques provided by ACADO.

The MPC optimization problem can be expressed as an initial value OCP problem with an end-free position (eqs. 1-5). The task of this optimization is to find the input $\mathbf{u}$ of the vehicle (velocity and steering angle momentum for kinematic model) along the optimization horizon $t \in (t_0, t_0 + T)$, that is over all potential candidate paths, by minimizing the cost function $J(\mathbf{u})$ given in (1). The integrand $\gamma(\mathbf{x}, \mathbf{u})$ represents the local roughness estimated by the vehicle within the sensor range. We use the roughness-based navigation RbNF, which represents a cost-to-go map, to extract a cost-to-go term Γ required by the MPC optimization. The RbNF might be computed as an optimal or approximated cost-to-go map [23]. The former gives better results, but is computationally expensive for large scale terrains. Since in our work we experiment with a small-scale terrain, the computational issue is not addressed. When the vehicle senses new information during the task execution, the RbNF can be updated similarly to [13]. Eqs. (2-5) represent optimization constraints including the differential constraint related to the vehicle model (2), control constraints (3), the safe stopping constraint (4) and the constraint which ensures the decrease of the Γ in order to guarantee that the plan reaches the goal position (5).

$$J(\mathbf{u}) = \int_{t_0}^{t_0+T} \gamma(\mathbf{x}, \mathbf{u})dt + \Gamma(t_0 + T) \qquad (1)$$

$$\frac{d}{dt}\mathbf{x} = f(\mathbf{x}) + g(\mathbf{x})\mathbf{u} \qquad (2)$$

$$\mathbf{u}(t) \leq \mathbf{u}_{max} \qquad (3)$$

$$v(t_0 + T) = 0 \qquad (4)$$

$$\Gamma(\mathbf{r}(t_0+T)) < \Gamma(\mathbf{r}(t_0+T_1)) < \Gamma(\mathbf{r}(t_0)) \qquad (5)$$

In some rare cases when ACADO fails, bringing back an infeasible solution or no solution, we use a backup strategy to guide the vehicle forward. In those cases, a planner selects a close way-point which is located along the steepest descent of the RbNF and solves for a two point boundary value OCP problem.

In the sequel, the aforementioned advantages of an MPC motion planner are summarised. An MPC based motion planner can easily accommodate for a vehicle model with all the required constraints. The planner might be near optimal (giving the current state information) due to "the optimality principle" since the RbNF is a near optimal estimator of the cost-to-go optimisation term. Since the MPC horizon can be arbitrarily chosen, a terrain shape interpolation required to get a differentiable objective function can be locally applied as in [24]. Having a differentiable objective function allows for using an OCP software. Using a software to solve a local OCP problem, like ACADO, covers much of the control and state space comparing to [16], [17], [18], [19], [20]. Finally, instead of repeating the complete path planning procedure from scratch when the vehicle senses new information, the RbNF can be easily updated similarly to [13].

VI. SIMULATION RESULTS

Fig. 6 illustrates a path generated by an MPC based planner. The path is drawn over a contour plot of the terrain roughness map. The terrain roughness map is computed by using terrain heights as in [25] and [22].
The example shows that the generated path avoids obstacles, follows less roughness regions (blue regions in Fig. 6) and reaches the goal position (start and goal positions are marked with a red and a pink disk, respectively).
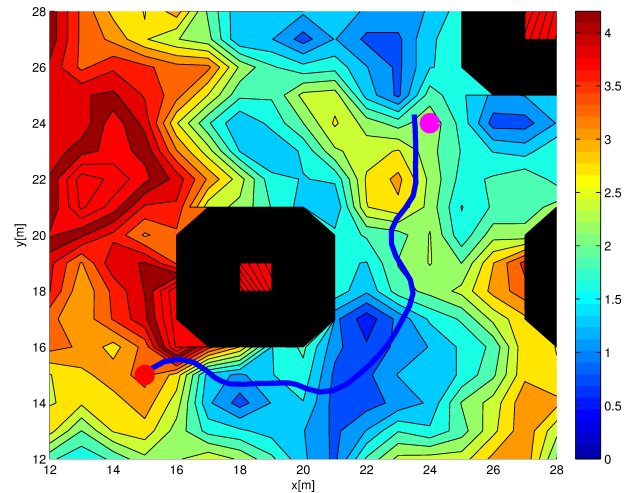

Fig. 6. An example of an MPC based solution

Finding a trajectory from an initial to the goal position using an OCP software is hardly feasible in real-time, especially for a large-scale terrain. The OCP solution finds the control inputs (velocity and steering angle for a kinematic model which is used in the simulations) that minimise

traversed roughness, taking into account the required constraints. Some of the possible constraints might include: avoiding obstacles, velocity and steering limitations, vehicle stability and the RbNF decreasing to guarantee reaching the goal (see, e.g. [22]).

In some cases where the terrain is small-scale, it is possible to compute a solution in a reasonable time by an OCP software such as ACADO. For this reason, we have used a small terrain 50m x 50m to compare an optimal and a MPC based solutions exploring the MPC sub-optimality. Fig. 7 depicts 10 simulations in which the same rough terrain and different vehicle initial positions are used. The average sub-optimality of the MPC based path planner can be computed as

$$\alpha = \frac{1}{N} \sum \frac{roughness^{OCP}}{roughness^{MPC}} = 0.43$$

where N is the number of simulations. One might see that in the 9th and 10th simulations, ACADO did not find a feasible solution for the OCP problem (depicted by 0 in the picture).
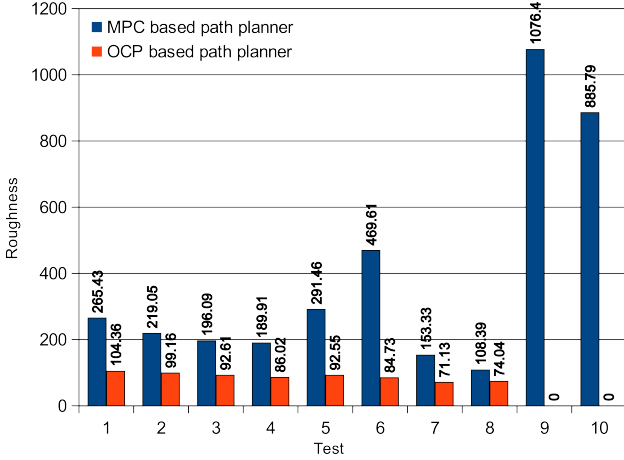

Fig. 7. I: Small-scale terrain. MPC and OCP solutions.

Fig. 8 depicts another example with 10 simulations on the same terrain with the same vehicle initial position and roughness shape, but with different obstacles. There are some examples where a MPC based solution has given a better result. This can be explained by the fact that an OCP software parametrises the control space in order to find the best solution. This might produce a solution that is not necessary the optimal one. In this example, the sub-optimality of the MPC path planner is much higher ($\alpha = 0.93$).

A two boundary value problem is difficult to solve in a feasible time on a large-scale terrain. For this reason, we use three different planners for a 500m x 500m terrain, a MPC based planner, a gradient based planner and a smooth gradient based planner. The gradient based planner is generated by the steepest descent of the RbNF. As already discussed, the gradient based planner is not considered as an acceptable solution in our work, since it does not take the vehicle model into account, and it is hard to predict how
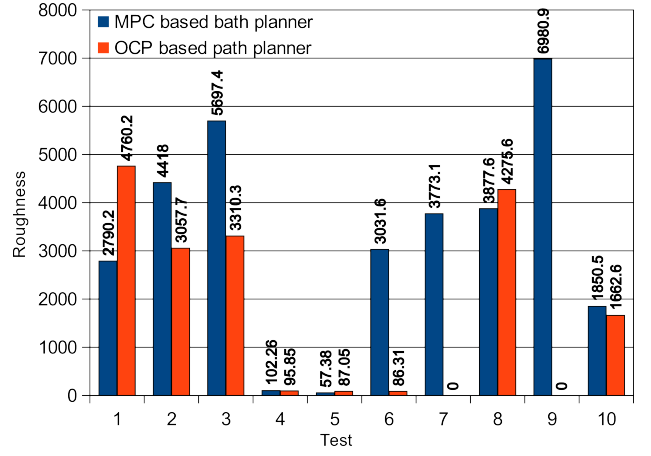

Fig. 8. II: Small-scale terrain. MPC and OCP solutions.

well the vehicle will follow such path. However, in order to validate the MPC based path planner, we introduce a smooth gradient based path planner which picks a point on the path obtained by the gradient based path planner and solves for a two boundary problem. Then, it repeats the procedure going towards the goal position. Fig. 9 compares the two planners on 10 different rough terrains. The sub-optimality of the MPC based path planner is $\alpha = 1.8$, which means that the MPC based planner performs better than the smooth gradient path planner. Again, this can be explained by the fact that the smooth gradient based path planner does take the vehicle model into account but only to follow the gradient based path planner.
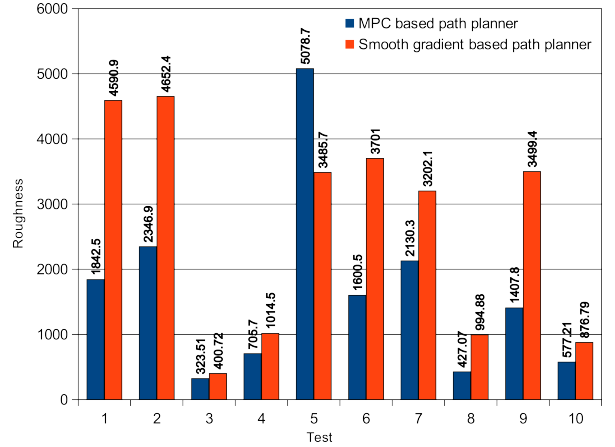

Fig. 9. Large-scale terrain. MPC and smooth gradient based solutions.

VII. CONCLUSIONS

This paper describes part of the work devoted to the development of an All-Terrain Mobile Robot, based on a commercial All-Terrain Vehicle, for high speed riding on difficult terrains.

Among the huge number of functionalities required to autonomously take the vehicle from a start to a goal position

through a safe path, accounting for terrain traversability, obstacles and vehicle constraints, the paper is focused on the hardware/software architecture and, above all, on the real-time implementation of a MPC-based planner. The issues involved in the implementation of the planner, using the open-source solver ACADO, are thoroughly discussed.

The simulation results show the effectiveness of the planner, and compare the paths computed by the MPC planner with those computed using a different approach.

An experimental validation of the MPC planning software, using the vehicle described in Section II, is ongoing. The results will be published soon.

REFERENCES

[1] A. Garcia Cerezo, A. Mandow, J. Martinez, J. Gomez de Gabriel, J. Morales, A. Cruz, A. Reina, and J. Seron, "Development of ALACRANE: a mobile robotic assistance for exploration and rescue missions," in *IEEE International Workshop on Safety, Security and Rescue Robotics*, 2007.

[2] P. Debenest, E. Fukushima, and S. Hirose, "Proposal for automation of humanitarian demining with buggy robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1, 2003, pp. 329–334.

[3] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *IEEE International Conference on Robotics and Automation*, vol. 2, 1985, pp. 500–505.

[4] R. Lenain, B. Thuilot, C. Cariou, and P. Martinet, "Backstepping observer dedicated to tire cornering stiffness estimation: application to an All Terrain Vehicle and a farm tractor," in *IEEE/RSJ International Conference on Intelligent Robotics Systems*, 2007, pp. 1763–1768.

[5] J. van der Burg and P. Blazevic, "Anti-lock braking and traction control concept for all-terrain robotic vehicles," in *IEEE International Conference on Robotics and Automation*, vol. 2, 1997, pp. 1400–1405.

[6] N. Bouton, R. Lenain, B. Thuilot, and J. Fauroux, "A rollover indicator based on the prediction of the load transfer in presence of sliding: application to an All Terrain Vehicle," in *IEEE International Conference on Robotics and Automation*, 2007, pp. 1158–1163.

[7] H. Utz, S. Sablatnög, S. Enderle, and G. Kraetzschmar, "Miro-middleware for mobile robot applications," *IEEE Transaction on Robotics and Automation*, vol. 18, no. 4, pp. 493–497, 2002.

[8] L. Bascetta, G. Magnani, P. Rocco, and A. Zanchettin, "Design and implementation of the low-level control system of an All-Terrain Mobile Robot," in *International Conference on Advanced Robotics*, 2009.

[9] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Ng, "ROS: an open-source Robot Operating System," in *IEEE International Conference on Robotics and Automation - Workshop on Open Source Software*, 2009.

[10] H. Bruyninckx, "Open robot control software: the OROCOS project," in *IEEE International Conference on Robotics and Automation*, 2001, pp. 2523–2528.

[11] M. Linderoth, K. Soltesz, and R. Murray, "Nonlinear lateral control strategy for nonholonomic vehicles," in *American Control Conference*, 2008, pp. 3219–3224.

[12] A. Stenz, "Optimal and efficient path planning for partially-known environments," in *Proc. of the IEEE International Conference on Robotics and Automation*, 1994, pp. 3310 – 3317.

[13] A. Stentz, "The focussed D* algorithm for real-time replanning," in *Proc. of the International Joint Conference on Artificial Intelligence*, 1995, pp. 1652–1659.

[14] S. Koenig and M. Likhachev, "Fast replanning for navigation in unknown terrain," *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 354 – 363, 2005.

[15] B. Houska and H. Ferreau, "ACADO toolkit - Automatic Control and Dynamic Optimization," http://acadotoolkit.org.

[16] C. J. Green and A. Kelly, "Toward optimal sampling in the space of paths," in *Proc. of the International Symposium of Robotics Research*, 2007, pp. 171 – 180.

[17] M. Pivtoraiko, R. A. Knepper, and A. Kelly, "Differentially constrained mobile robot motion planning in state lattices," *J. Field Robot.*, vol. 26, no. 3, pp. 308–333, 2009.

[18] T. M. Howard and A. Kelly, "Optimal rough terrain trajectory generation for wheeled mobile robots," *Int. J. Rob. Res.*, vol. 26, no. 2, pp. 141–166, 2007.

[19] M. Pivtoraiko and A. Kelly, "Efficient constrained path planning via search in state lattices," in *Proc. The 8th International Symposium on Artificial Intelligence, Robotics and Automation in Space*, September 2005.

[20] T. M. Howard, C. J. Green, A. Kelly, and D. Ferguson, "State space sampling of feasible motions for high-performance mobile robot navigation in complex environments," *Journal of Field Robotics*, vol. 25, no. 10, pp. 325–345, 2008.

[21] A. Tahirovic and G. Magnani, "General framework for mobile robot navigation using passivity-based MPC," *IEEE Transactions on Automatic Control*, vol. 56, no. 1, 2011.

[22] ——, "Passivity-based model predictive control for mobile robot navigation planning in rough terrains," in *Proc. The 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, October 2010.

[23] ——, "A roughness-based rrt for mobile robot navigation planning," in *Proc. the 18th IFAC World Congress*, September 2011.

[24] F. J. Aguilar, F. Agüera, M. A. Aguilar, and F. Carvajal, "Effects of terrain morphology, sampling density, and interpolation methods on grid dem accuracy," *Photogrammetric Engineering & Remote Sensing*, vol. 71, no. 7, 2005.

[25] K. Iagnemma and S. Dubowsky, *Mobile Robots in Rough Terrain: Estimation, Motion Planning and Control with Application to Planetary Rovers*. New York: Springer Berlin / Heidelberg, 2004.