Real-Time Capable Path Planning for Energy Management Systems in Future Vehicle Architectures

Jonathan Brembeck and Christoph Winter

***Abstract*—In this paper an energy optimal path planning and velocity profile generation for our highly maneuverable Robotic Electric Vehicle research platform ROboMObil is presented. The ROMO [1] is a development of the German Aerospace Center's Robotics and Mechatronics Center to cope with several research topics, like energy efficient, autonomous or remote controlled driving for future (electro-) mobility applications. The main task of the proposed algorithms is to calculate an energy optimal trajectory in a real-time capable way. It is designed to incorporate data from actual traffic situations (e.g. oncoming traffic) or changed conditions (e.g. snowy conditions). The resulting trajectory is then fed forward to a lower level time independent path following control [2] that calculates the motion demands for our energy optimal control allocation. This in turn distributes the demand to the actuators of the over-actuated vehicle. We show a numerical reliable way to formulate the energy optimal path planning optimization objective, which is able to provide a consistent replanning feature considering the actual vehicle states. Besides this, different types of optimization methods are evaluated for their real-time capabilities. The velocity profile will be calculated afterwards and the generation of the profile is also enabled to handle dynamic replanning. Finally, we show several experimental results, using a virtual road definition and tests on a commercial real-time platform.**

I. Introduction

The Energy Management (EM) Strategy of efficient future vehicle architectures will be one of the key technologies for sustainable individual mobility. Besides many strategies to optimize the energy consumption of auxiliary devices or optimization of the energy flow in (serial) hybrid architectures, we focus in this work on the potential of energy management of the motion layer of the vehicle [3]. For our research we use DLRs robotic research platform ROboMObil [1], which represents an electro-mobility concept with four independent *Wheel Robots*. Through its full *X-by-Wire* central control architecture it is possible to command all motion actuators (steering, traction, braking) separately. In this way we are able to develop new control strategies without any restriction like couplings of the driver's input wish. In this publication we complete the EM scheme proposed in [3] and discuss the top level "Global Energy Optimization", (Fig. 1) and its connection to the lower level block "Optimal Actuator Control". In this high level planner stage we propose a real-time capable path planning and velocity profile generation scheme and its connection to the energy optimal *Control Allocation* (CA) algorithms [3] in the underlying level. Experimental results, in combination with the mentioned previous work [3], will show the capabilities and performance of the here proposed EM framework.

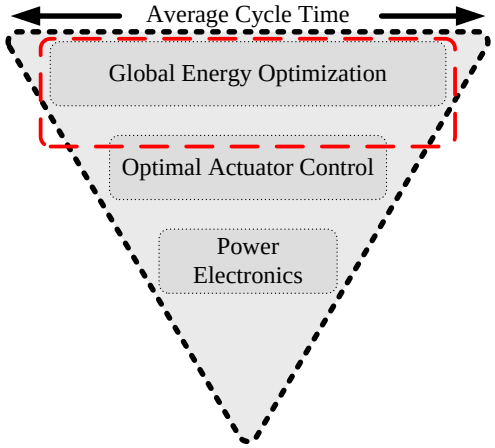

Fig. 1: ROMOs Energy Management strategy – the scope of this work is marked red (Graphic is developed in imitation of [6])

The paper is organized as follows: In section II we sketch the development of our proposed energy optimal path planning optimization objective and its difference to existing approaches. For its solution we show an adapted nonlinear optimization method to achieve efficient real-time capabilities. Furthermore, we extended the approach to generate an associated velocity profile generation. Section III shows experimental results in our Software In the Loop (SIL) virtual road simulator using a commercial real-time platform and the OpenDrive standard [5] as road definition. Finally we give a short overview of on-going investigations, as well as a summary of the scientific contribution in this publication. This paper is part of these investigations which are also published in [7] and not cited separately.

II. Real-Time Capable Path Planning

In this chapter we sketch an optimization problem formulation for a real-time capable path planning algorithm that combines the benefits of two approaches (explained in sec. II.A), which are a small scale optimization problem and inequality constraints which represent the environment more accurately. For doing this we assume a situation as shown in

Manuscript received January 10, 2014.
Jonathan Brembeck is research assistant and PhD student at the DLR Robotics and Mechatronics Center (RMC), Oberpfaffenhofen, Germany. (E-mail: Jonathan.Brembeck@dlr.de).
Christoph Winter has recently finished his MSc. at the Technical University of Munich and will now work as a research assistant at the DLR RMC (E-mail: Christoph.Winter@dlr.de).

Fig. 2. A vehicle is traveling along a predefined road (here we do not deal with the navigation in a road network) and it is able to use the available lateral space w_{res} on the street to minimize a criteria like the path curvature.

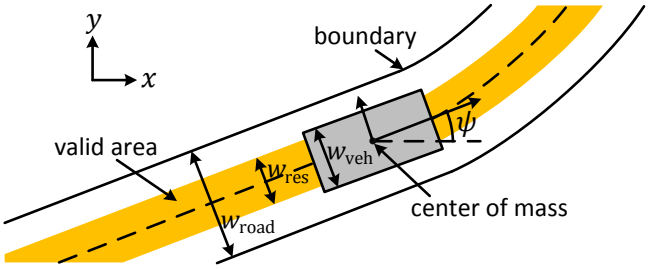

Fig. 2: The vehicle model with width of w_{veh} and orientation ψ defines the valid area of the road (graphic is based on [8]).

Furthermore, we introduce an additional constraint to enable real-time path replanning capabilities that incorporate the actual vehicle state. In a last step a velocity profile along the optimized path is calculated that considers the resistant forces and the physical limitations of the vehicle.

A. State of the art in vehicle path planning

As a preliminary work we examined two promising approaches of the publications [8] and [9]. The first approach in [8] is based on the idea that every road boundary as well as the vehicle path itself can be represented as two splines in the direction of x and y (Fig. 2):

$$\boldsymbol{o}_i(p) = \boldsymbol{a}_i(p-p_i)^3 + \boldsymbol{b}_i(p-p_i)^2 + \boldsymbol{c}_i(p-p_i) + \boldsymbol{d}_i \quad (1)$$

Each spline consists of $n-1$ polynomial functions $o_i(p)$ between n interpolation points p_i. The spline parameter p is defined by the linear relation $p_i = i-1$ with $i = 1 \dots n$. The authors derived a linearized Quadratic Problem (QP) with inequality constraints that are based on a polynomial coefficient comparison of the boundaries and the optimized path. As the minimization criteria they defined a linearized function $\tilde{E}$ of the curvature along the path:

$$\tilde{E} = \sum_{i=1}^{n-1} \sum_{q\in\{x,y\}} h_i\left(b_{i,q}^2 + b_{i,q}b_{i+1,q} + b_{i+1,q}^2\right) \quad (2)$$

with $h_i = p_{i+1} - p_i$. The resulting QP enables a good real-time capability due to the availability of powerful solvers e.g. *QL* [10]. Unfortunately we experienced several problems in implementation with this approach. If the path changes a quadrant it is necessary to redefine the inequality constraints, to be consistent with the left and right street boundary. Therefore, it is necessary to place an extra interpolation point at the transition of the quadrants, which could lead to a poor weighting of the spline segments. Due to the choice of the polynomial coefficients as optimization variables and its weak coupling between each other, the quadratic matrix of the QP gets sparse and thus the numeric error large, what may cause inaccurate results. This effect is strengthened through the linearization of the curvature objective, which is only valid for very small distances. In an attempt to handle this with weighting factors in the objective we were not able to guarantee a robust solution of the approach.

The second algorithm we have evaluated [9] simplifies the optimization approach drastically, since only the scalars $\alpha_i \in [0,1]$ at the junction points of the polynomial sections can be tuned by the optimizer. These scalars α_i define the point on the connection line between the right and the left street boundary at the point $p = p_i$ (see Fig. 3), where $\alpha_i = 0$ means the path is located on the left and $\alpha_i = 1$ means it is located on the right. In this way the optimization problem has a dense formulation and numerically reliable results are achievable. Nevertheless with this approach it cannot be guaranteed that the calculated path will stay within the boundaries between two control points.
In the following sub-section we will provide a sketch of our new approach to path planning that tries to combine the advantages of these both algorithms.

B. The new combined approach

The aim of our approach is to find a path representation leading to a small scale optimization problem and being capable to formulate more representative inequality constraints than in [9], i.e. also between the interpolation points.
Our problem formulation uses polynomials as the path representation (like in [8]), but with interpolation points fixed on a line between the boundary knots like in [9]. The benefit of this representation is that we are able to reduce the eight optimization variables (compare eq. (1), [8]) per spline segment to only three.

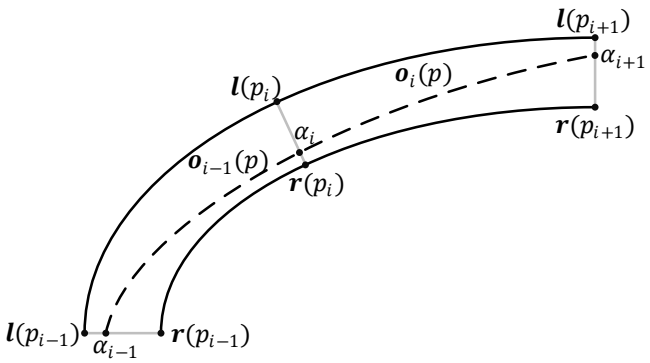

Fig. 3: Notation of spline representation.

The reduced set of variables for one spline segment $\boldsymbol{o}_i(p)$ are the polynomial coefficients $\boldsymbol{b}_i = \left(b_{i,x}\ b_{i,y}\right)^T$ and a scalar scaling factor α_i. All other spline parameters can be expressed in dependency on $\boldsymbol{b}_i$ and α_i. This representation can be derived from the constraint that a $\mathcal{C}^2$ continuous transition between the spline segments is guaranteed. This demand can be expressed in a set of equality constraints: The position and its first and second derivative of two spline segments have to be the same at each junction point:

$$h_i^3\boldsymbol{a}_i + h_i^2\boldsymbol{b}_i + h_i\boldsymbol{c}_i + \boldsymbol{d}_i = \boldsymbol{d}_{i+1} \quad (3)$$

$$3h_i^2\boldsymbol{a}_i + 2h_i\boldsymbol{b}_i + \boldsymbol{c}_i = \boldsymbol{c}_{i+1} \quad (4)$$

$$3h_i\boldsymbol{a}_i + \boldsymbol{b}_i = \boldsymbol{b}_{i+1} \quad (5)$$

with $i = 1 \dots n-2$. Equations (3) and (5) contain four unknown $\boldsymbol{a}_i, \boldsymbol{c}_i, \boldsymbol{d}_i$ and $\boldsymbol{d}_{i+1}$. To obtain a determined system of linear equations an additional one (6) is added to the set to

describe the shifting of the interpolation points between the left $\boldsymbol{l}_i$ and the right $\boldsymbol{r}_i$ side of the street boundaries via α:

$$\boldsymbol{d}_i = \boldsymbol{l}_i + (\boldsymbol{r}_i - \boldsymbol{l}_i) \cdot \alpha_i \quad \forall i = 1 \dots n \tag{6}$$

Solving the system of linear equations (3), (5) and (6) and substituting the results into (1) yields to the compact path representation with the $3n$ variables $\boldsymbol{b}_i$ and α_i.

The representation is used for our new equality constraint, which results from substituting the derived dependencies from $\boldsymbol{a}_i, \boldsymbol{c}_i$ and $\boldsymbol{d}_i$ on $\boldsymbol{b}_i$ and α_i into the remaining original constraint (4). Equation (7) shows the reduced equality constraint for $h_i = 1$ and $i = 2 \dots n-1$

$$\begin{aligned} &\boldsymbol{b}_{i-1} + 4\boldsymbol{b}_i + \boldsymbol{b}_{i+1} = \\ &3(\boldsymbol{l}_{i-1} + (\boldsymbol{r}_{i-1} - \boldsymbol{l}_{i-1})\alpha_{i-1}) - 6(\boldsymbol{l}_i + (\boldsymbol{r}_i - \boldsymbol{l}_i)\alpha_i) \\ &\quad + 3(\boldsymbol{l}_{i+1} + (\boldsymbol{r}_{i+1} - \boldsymbol{l}_{i+1})\alpha_{i+1}) \,. \end{aligned} \tag{7}$$

Consequently we can decrease the number of equality constraints, since they are partly contained in the new path representation.

The inequality constraints can be simply described with the introduced optimization variable α_i as in [9]:

$$0 \le \alpha_i \le 1; \quad i = 1 \dots n \,. \tag{8}$$

In the next section we show that the optimized path lies always in the constrained area. To achieve a better coverage, the interpolated points could be placed closer together, with the major disadvantage of a larger optimization problem. Therefore we suggest here a further development of orientation independent inequality constraints without the need of additional optimization variables. The same principle as for α is used at intermediate points between the spline sampling points. The i-th optimized path segment $\boldsymbol{o}_i(p)$ evaluated at k linearly spaced points $\rho_j = p_i + j \cdot \frac{h_i}{k+1}$ for $j = 1 \dots k$ must be located between the left boundary $\boldsymbol{l}_i(p)$ and the right boundary $\boldsymbol{r}_i(p)$ at the same points. The new formulated inequality constraints of (9) are visualized in Fig. 4.

$$0 \le \frac{\boldsymbol{r}_i(\rho_j) - \boldsymbol{l}_i(\rho_j)}{\left\|\boldsymbol{r}_i(\rho_j) - \boldsymbol{l}_i(\rho_j)\right\|^2} \left(\boldsymbol{o}_i(\rho_j) - \boldsymbol{l}_i(\rho_j)\right) \le 1 \tag{9}$$

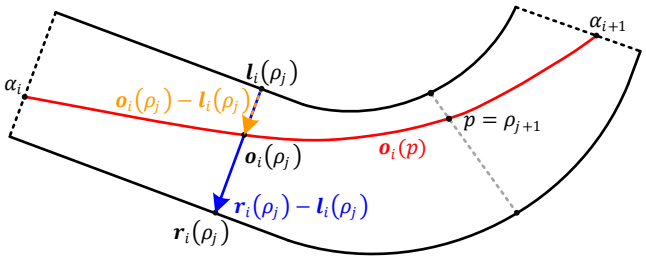

Fig. 4: Additional inequality constraints at $p = \rho_j$ between sampling points.

The vector between the left boundary and the optimized path $\boldsymbol{o}_i(\rho_j) - \boldsymbol{l}_i(\rho_j)$ in direction of the vector between the left and right boundary $\boldsymbol{r}_i(\rho_j) - \boldsymbol{l}_i(\rho_j)$ must be longer than zero and shorter than $\boldsymbol{r}_i(\rho_j) - \boldsymbol{l}_i(\rho_j)$. The accuracy of these constraints can be defined independently of the number of interpolated points. There is the possibility that despite valid inequality constraints the optimized path lies outside of the boundaries, but due to the curvature minimization it is very unlikely that a violating path is computed (e.g. loops between to control points may cause a large value in the curvature cost function).

In summary we propose an optimization problem with a nonlinear cost function and linear constraints. As cost function the path curvature is chosen as introduced in [11].

With a smaller curvature of the path, a turn can be passed with a higher velocity according to $v \le \sqrt{a_{\text{lat,max}}/\kappa}$ and thus with less energy consumption due to less acceleration of the vehicle. The optimization problem with the spline coefficients $\boldsymbol{x} = \left(b_{1,x}, b_{1,y}, \alpha_1, \dots, b_{n,x}, b_{n,y}, \alpha_n\right)^T$ is given as:

$$\begin{aligned} \boldsymbol{x}^* = \underset{\boldsymbol{x}}{\text{argmin}} \quad & \int_0^{p_n} \kappa^2(\boldsymbol{x}, p)\, \mathrm{d}p \\ \text{s.t.} \quad & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ & \boldsymbol{C}\boldsymbol{x} \le \boldsymbol{d} \,. \end{aligned} \tag{10}$$

Thereby all derivatives of the polynomials in the integral over the quadratic path curvature (11) can be easily calculated analytically

$$\kappa(p) = \frac{o_y''(p)o_x'(p) - o_x''(p)o_y'(p)}{\left(o_x'^2(p) + o_y'^2(p)\right)^{\frac{3}{2}}} \,. \tag{11}$$

The dimensions of the constraint matrixes are $\boldsymbol{A} \in \mathbb{R}^{[2(n-2)+4]\times[3n]}$ and $\boldsymbol{C} \in \mathbb{R}^{[2k\cdot(n-1)+2n]\times[3n]}$ compared to $\boldsymbol{A} \in \mathbb{R}^{[6(n-2)]\times[8(n-1)]}$ and $\boldsymbol{C} \in \mathbb{R}^{[16(n-1)]\times[8(n-1)]}$ in [7]. That means we achieve a reduction of the number of optimization parameters and an increase of non-zero entries within the minimization objective if the QP is applied.

To add a dynamic replanning feature in a real-world application (i.e. information of upcoming obstacles), we needed to modify the spline formulation. To have a continuous junction between a previously planned path and a new planned path segment the spline formulation must be capable to ascertain the starting position with its first derivative and the curvature as boundary conditions. With more than two boundary conditions the system of linear equations of the spline representation would be overdetermined [12]. To cope with this problem we introduce an extra point (red cross on spline without blue box in Fig. 5) whose position cannot be tuned by the optimizer. Thus, the system gets determined when introducing the third boundary condition and we can guarantee continuity of the replanned path.

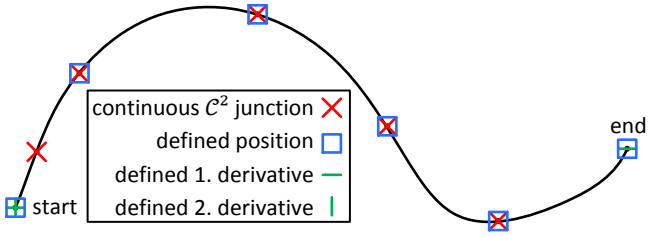

Fig. 5: All points except the second point have a defined position. The inner points have a $\mathcal{C}^2$ continuous junction. A defined slope is applied to the start and end points and a defined second derivative to the starting point.

In summary we have achieved a new optimized problem formulation and its solution based on the ideas for path planning in [8], [9]. We introduced a nonlinear objective that can handle, by the use of analytic derivatives, a precise

curvature evaluation (referring to (11)). Furthermore, we could reduce the dimension of the optimization variables from $8(n-1)$ to only $3n-1$ which reduces the computational effort. Through the reduction of degrees of freedom in the set of linear equations we enabled the introduction of an additional boundary condition to have consistent restart conditions for replanning the path. All constraints are kept linear and the inequality constraints are orientation independent through the pointwise proof. Through the representation with the variables $b_{x,i}$, $b_{y,i}$ and α_i their number could be reduced also. The number of equality constraints has been reduced to one third compared to [8]. The number of inequality constraints could be reduced to one eighth if no intermediate points are evaluated, which is equivalent to the constraints in [9]. It remains for further investigations to determine which distance and distribution between evaluated sampling and intermediate points are sufficient.

C. Structure exploiting optimization

To achieve our goal to make the optimization of the problem formulation (10) real-time capable, we decided to choose an optimization method that can exploit the problem structure. While testing different approaches (compare TABLE 1 and the explanation in the latter) we choose a nonlinear gradient method (NG) similar to [13] as optimizer of choice. The reason for this can be explained as follows. First, it is quite simple to derive the gradient function after approximating the curvature integral with a Riemann sum:

$$E_i = \int_{p_i}^{p_{i+1}} \kappa^2(\boldsymbol{x}_i, p)\,\mathrm{d}p \approx \hat{E}_i = \sum_j \kappa_i^2\left(\boldsymbol{x}_i, p_j\right) \cdot \Delta p_j \qquad (12)$$

with a small constant interval size Δp_j. This approximation is necessary since no analytic integral solution of the quadratic curvature term (11) is known. Using this, the gradient computation can easily be divided into sub problems through the piecewise polynomial representation of the spline. Second, the linear equality and inequality constraints enable projection methods to determine a feasible search direction. The negative gradient $-\nabla f(\boldsymbol{x})$, i.e. the search direction, is projected onto the equality constraints $\boldsymbol{A}$ by computing a Moore-Penrose pseudoinverse $\boldsymbol{P}$, resulting in the projected direction $\boldsymbol{r}$ as described in [13] and shown in (13)

$$\boldsymbol{r} = -\underbrace{\left(\boldsymbol{I} - \boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{A}^T)^{-1}\boldsymbol{A}\right)}_{\hat{\boldsymbol{P}}} \nabla f(\boldsymbol{x})\,. \qquad (13)$$

The included matrix inversion has to be performed only once, since the equality constraints do not change during the iterations. For the inequality constraints a more complex Gram-Schmidt projection is chosen, which projects the search direction into the null space of the active inequality constraints. This avoids computational expensive matrix inversions if the set of active inequality constraints changes, which is an advantage especially with large scale problems like they can occur in our case on long planned paths.

Nevertheless we want to give a short overview regarding the performance of our NG technique in comparison to other approaches: (linearized) Quadratic Program, `fmincon` (the built-in solver from MATLAB) and Sequential Quadratic Program (SQP).

TABLE 1
COMPARISON OF OPTIMIZERS

Algorithm	Time	Iterations	Cost Function c_i	Relative Increase $(c_i - c_{\min})/c_{\min}$
QP	0.09 s	-	$3.490 \cdot 10^{-2}$	38.7 %
NG	2.67 s	73	$2.549 \cdot 10^{-2}$	1.31 %
fmincon	7.13 s	201	$2.568 \cdot 10^{-2}$	2.07 %
SQP	38.6 s	482	$2.516 \cdot 10^{-2}$	-

TABLE 1 shows the comparison between different implemented optimization algorithms with a 3 km long track. On the one hand, the quadratic approximated cost function results in a significant higher curvature in contrast to the nonlinear cost criterion in (10). On the other hand, all algorithms using the nonlinear criterion yield to a similar optimized path. Regarding the computation time the NG is always faster than the SQP algorithm due to no Hessian matrix is required. The NG method is chosen for the real-time implementation of the online planning process, because of its simplicity and fast computation time without sacrificing the ability to find a reasonable solution.

D. Velocity Profile Generation

After we have gathered the spatial information of the path via optimization in the preceding sub-sections it is now necessary to generate the time information along the path. The velocity profile generation is based on [4] and consists of three main steps. First, a desired velocity profile is defined considering the maximum vehicle velocity (constraint by technical limits) and the road speed limits. In the second step this desired profile is modified to satisfy a lateral acceleration limit with the given path curvature regarding the following equation: $v \leq \sqrt{a_{\mathrm{lat,max}}/\kappa}$. This lateral acceleration limit $a_{\mathrm{lat,max}}$ is mainly based on comfort requirements. The last step is a forward and backward filtering of the velocity profile to fulfill the longitudinal acceleration limits. These limits result from all acting forces on the vehicle, which are the motor torque τ and the counteracting driving resistance as depicted in Fig. 6.

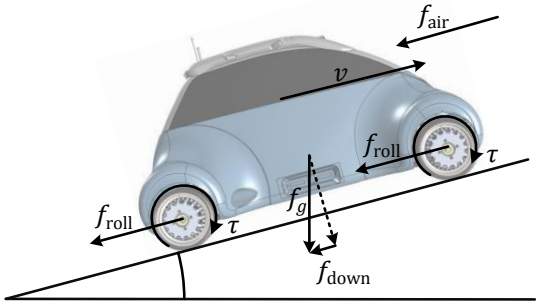

Fig. 6: Picture of the ROMO with all resulting forces.

The driving resistances considered here are f_{air}, f_{roll} and f_{down} of the air resistance, rolling resistance and downhill force, respectively.

III. REAL-TIME IMPLEMENTATION & EXPERIMENTS

This section summarizes our online path planning approach and explains the cooperation of the previously introduced

parts. Fig. 7 gives an overview of the signal flow of the real-time implementation with its main path planning part executed at a longer sampling time T_{slow} and the interface to the *Vehicle Dynamics Control* (VDC) executed at a shorter sampling time T_{fast}. Starting with the query of the road definition (by the use of the OpenDrive [5] representation), the road boundary points and therefore the inequality constraints can be determined. Additionally, the boundary conditions, i.e. start and end values of the spline and their derivatives, are required for the optimization. The start conditions have to match with the end of the previous planned path segment, provided by the unit delay blocks. The resulting optimization variables $\boldsymbol{x}^*$ are then evaluated to derive the spline representation of the optimized path. The curvature of the path defines the velocity profile $\boldsymbol{v}$ beside the vehicle dynamics limitations of the controlled vehicle. The interface to the vehicle dynamics control [2] is a time independent motion demand $\boldsymbol{\lambda}(s)$. It consists of the five quantities: absolute position x and y of the reference path, the corresponding path orientation ψ, its curvature κ and the generated velocity v. The feedback from the VDC is the state s describing the currently required arc length of the reference path.

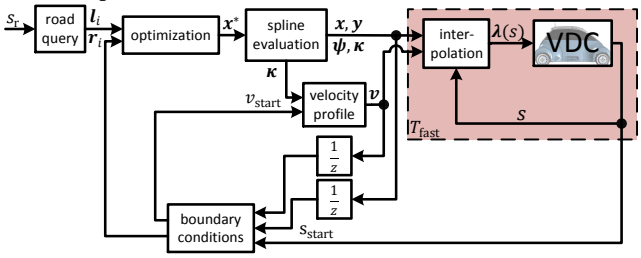

Fig. 7: Overview of the online planning approach with interface to VDC.

The path planning is implemented in MATLAB/Simulink and in a separate Modelica [15] model the vehicle dynamics model and the vehicle dynamics control are implemented [2]. Both models are cross-compiled for the dSpace real-time system ScalexIO [14] with the motion demand $\boldsymbol{\lambda}(s)$ and the arc length s defined above as interface to each other.

A. Experimental test track

The evaluation of the planning approach is performed with the road description illustrated in Fig. 8. This test road is a 2 km long part of track number 2716 of the *SmartDBRuaralOnly* OpenDrive [5] representation.

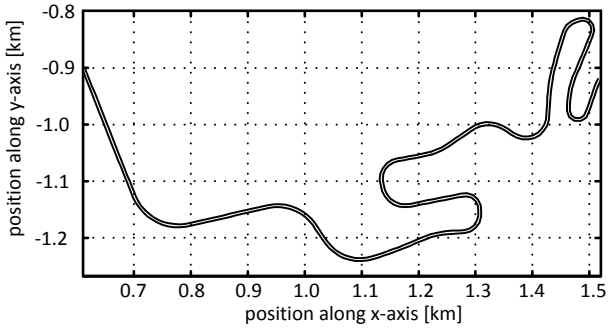

Fig. 8: First 2 km long part of the OpenDrive [5] test track.

B. Experimental results

The non-optimized curvature of the road is compared with the curvature achieved by the online planning in Fig. 9. The quadratic characteristic of the cost criterion causes a higher weighting on high curvatures. Thus, the peak curvature of sharp turns is reduced more than the curvature of wide turns. Additionally, smoother transitions between the turns are realized by the optimization. The depicted curvature reduction in Fig. 9 results in a path which requires less energy, as explained in II.B and according to [7], and a higher driving comfort. Fig. 10 shows the optimized path of our test track, which always stays within the bounds.

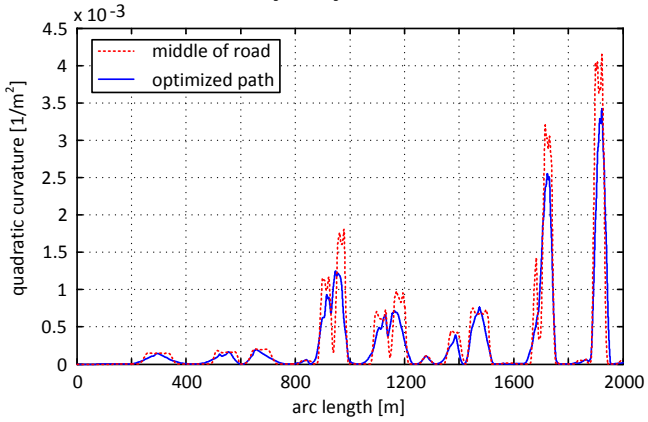

Fig. 9: Comparison of the quadratic path curvature of the test track and the online optimized path with a horizon of $\Delta s = 200$ m.

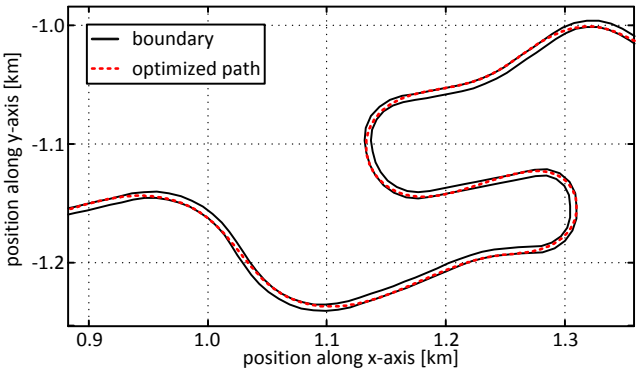

Fig. 10: Detail of global optimized test track.

A quantitative comparison and an evaluation of the influence of the optimization horizon length are given in the following table.

TABLE 2
COST FUNCTION EVALUATIONS

Horizon Size	Cost Function	Cost Reduction
$\Delta s = 0$ m (no opt.)	$626.8 \cdot 10^{-3}$	-
$\Delta s = 100$ m	$554.4 \cdot 10^{-3}$	11.5 %
$\Delta s = 200$ m	$510.9 \cdot 10^{-3}$	18.5 %
$\Delta s = 2$ km (global opt.)	$509.2 \cdot 10^{-3}$	18.8 %

The comparison shows that the global optimization achieves a saving of almost 19 % on this track. The cost function reduction is not drastic, since the width of 7 m of the test road is a very strict boundary specification. On the other hand, TABLE 2 depicts the dependency of the cost function on the chosen horizon size. The longer the optimization horizon is the more the cost function can be reduced. With a length of $\Delta s = 200$ m no major cost function difference

compared to the global optimization is realized, thus this length represents a sufficient choice of the horizon in our scenario.

As last part of the results section we want to show the real-time capability of our approach. An implementation is real-time capable if an upper bound of the execution time can be determined which is never exceeded and if the implementation produces reliable results within this time. All required computations, except the optimization, like the spline evaluation or the velocity profile generation are bounded by the considered track length. The optimization is also bounded by the maximum number of iterations, which can be applied as stop criterion beside a cost based criterion. Thus a global upper bound of the execution time of the online path planning approach can be determined.

Fig. 11 shows the execution time of the real-time system with a cost based stop criterion of the optimization. If the path within a turn is optimized more iterations are required to reach the stop criterion, which results in a longer execution time. In a straight segment the cost criterion is already very low at the beginning of the optimization process and less iterations are necessary.

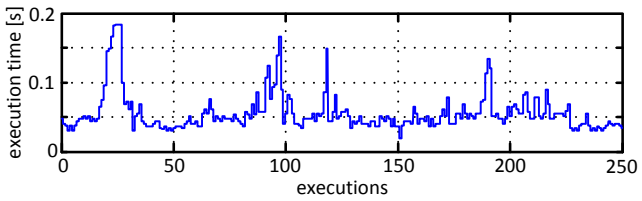

Fig. 11: Execution time on the real-time system without a fixed number of iterations and a horizon size of $\Delta s = 200$ m.

A fixed and sufficiently high number of iterations is chosen, yielding an average execution time of less than 0.22 s. With this execution time the sampling time T_{slow} of 0.4 s is utilized by 55 % on the real-time platform. The sampling time T_{slow} is chosen regarding to the tradeoff between fast path updates and the capability of handling also longer horizons.

IV. Discussion and Outlook

The first simulative results show the potential of online path optimization for saving energy to travel a given route on a street network using our virtual model of the ROboMObil. The algorithms are real-time capable and could be successfully connected to the underlying Optimal Actuator Control (see Fig. 1). Ongoing research will cope with an extension to the velocity profile as part of the optimization problem. It is also planned to enable replanning capabilities with time variant obstacles along the path.

Acknowledgment

The authors express their gratitude to Peter Ritzer for supporting the investigations, especially for providing a time independent path following control to connect the top and the middle control layer, Bernhard Thiele for his support with cross-compilation to real-time target hardware and Tilman Bünte for his prototype implementation of OpenDrive interface in Modelica [15]. Furthermore, we would like to thank the ROboMObil team and the head of the institute Johann Bals for their support.

V. References

[1] J. Brembeck et al., "ROMO – THE ROBOTIC ELECTRIC VEHICLE," in *22nd IAVSD International Symposium on Dynamics of Vehicle on Roads and Tracks*, Manchester, 2011.

[2] P. Ritzer, J. Brembeck, and R. Kennel, "Model Based Vehicle Dynamics Control for Modern Vehicle Architectures," Institute for Electrical Drive Systems and Power Electronics, TUM, Munich, Masters Thesis http://elib.dlr.de/87120/, 2013.

[3] J. Brembeck and P. Ritzer, "Energy optimal control of an over actuated Robotic Electric Vehicle using enhanced control allocation approaches," , June 2012, pp. 322-327.

[4] T. Bünte and E. Chrisofakis, "A Driver Model for Virtual Drivetrain Endurance Testing," in *Proceedings of 8th Modelica Conference*, 2011, pp. 180-188.

[5] M. Dupuis. (2010) OpenDRIVE Format Specification. [Online]. http://www.opendrive.org/docs/OpenDRIVEFormatSpecRev1.3D.pdf

[6] L. Rosario and P.C.K. Luk, "Applying Management Methodology to Electric Vehicles with Multiple Energy Storage Systems," in *International Conference on Machine Learning and Cybernetics*, vol. 7, Hong Kong, 2007, pp. 4223-4230.

[7] C. Winter, J. Brembeck, and R. Kennel, "Online Energy Optimal Path Planner for Advanced Electric Vehicles," Institute for Electrical Drive Systems and Power Electronics, TUM, Munich, Masters Thesis http://elib.dlr.de/87121/, 2013.

[8] J. Daniel, A. Birouche, J.-P. Lauffenburger, and M. Basset, "Navigation-based constrained trajectory generation for advanced driver assistance systems," *International Journal of Vehicle Autonomous Systems*, vol. 9, no. 3-4, pp. 269-296, 2011.

[9] F Braghin, F Cheli, S Melzi, and E Sabbioni, "Race Driver Model," *Computers and Structures*, vol. 86, no. 13-14, pp. 1503-1516, 2008.

[10] K. Schittkowski. (2005) QL: A Fortran Code for Convex Quadratic Programming - User's Guide. [Online]. http://www.ai7.uni-bayreuth.de/ql_rep.htm

[11] B. K. P. Horn, "The Curve of Least Energy," *ACM Transactions on Mathematical Software*, vol. 9, no. 4, pp. 441-460, Dezember 1983.

[12] D.S.G. Pollock,.: Academic Press, 1999, ch. Smoothing with cubic Splines, pp. 293-322.

[13] J. Rosen, "The Gradient Projection Method for Nonlinear Programming. Part I. Linear Constraints," *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, no. 1, pp. 181-217, 1960.

[14] dSpace, "SCALEXIO new technologies for HIL simulation," 2013. [Online]. http://www.dspace.com/shared/data/pdf/2013/dspace_productbrochure2013_SCALEXIO_en.pdf

[15] Modelica Association. (2012) Modelica - A Unified Object-Oriented Language for Systems Modeling Language Specification Version 3.3. [Online]. http://www.modelica.org