2014 IEEE Intelligent Vehicles Symposium (IV)
June 8-11, 2014. Dearborn, Michigan, USA

Trajectory Planning for BERTHA - a Local, Continuous Method

Julius Ziegler[1], Philipp Bender[1], Thao Dang[2] and Christoph Stiller[3]

***Abstract*—In this paper, we present the strategy for trajectory planning that was used on-board the vehicle that completed the 103 km of the Bertha-Benz-Memorial-Route fully autonomously. We suggest a local, continuous method that is derived from a variational formulation. The solution trajectory is the constrained extremum of an objective function that is designed to express dynamic feasibility and comfort. Static and dynamic obstacle constraints are incorporated in the form of polygons. The constraints are carefully designed to ensure that the solution converges to a single, global optimum.**

I. INTRODUCTION

In August 2013, the modified Mercedes-Benz S-Class S500 INTELLIGENT DRIVE ("BERTHA") completed the Bertha-Benz-Memorial-Route (BBMR) fully autonomously. The Bertha-Benz-Memorial-Route is a historic, 100 km long route connecting the cities of Mannheim and Pforzheim, and passing through 25 towns.

In this paper, we describe the aspects of trajectory planning that were used for this challenge, *i.e.* the part of the system that takes a sensor- and map based representation of the situation as input, and generates a trajectory that the vehicle is supposed to follow. To cope with moving obstacles, the trajectory is time parametrized.

We pose the trajectory planning problem as a non linear optimization problem with non linear inequality constraints. The objective function is essentially quadratic. We use a Newton type method to solve for the optimal trajectory. We show that the Hessian of the constrained optimization problem has a specific structure that can be exploited to improve performance of the optimization process, both in terms of the number of iterations required for convergence, and of complexity required for a single iteration.

Special attention was paid to the setup of the constraint system. Because we use a local optimization scheme that starts from an initial guess of the trajectory, we must make sure that the constraints are setup in such a way that the optimization problem has no stationary points other than the global optimum. To this end, the sensor information is reduced to simple polygons, from which differentiable constraint functions are derived. Through a decision process, the polygons are modified in a way to eliminate stationary points that are not the global optimum.

[1]J. Ziegler and P. Bender are with the FZI Research Center for Information Technology, Dep. on Mobile Perception Systems, 76131 Karlsruhe, Germany {ziegler|pbender}@fzi.de

[2]T. Dang is with Daimler AG, Research and Technology, 71063 Sindelfingen, Germany thao.dang@daimler.com

[3]C. Stiller is with Karlsruhe Institute of Technology (KIT), Department of Measurement and Control Systems (MRT), 76131 Karlsruhe, Germany stiller@kit.edu

II. RELATED WORK

A large part of vehicular trajectory- and path planning research is devoted to the solution of problems which show a strong combinatorial characteristic. This applies to most methods which have been designed for parking and maneuvering in narrow, unstructured spaces, where it is often necessary to maneuver backwards and forwards. Not surprisingly, this field is dominated by methods of combinatorial nature, *i.e.* a search tree is spanned which can be either random [1] or deterministic [2], [3], [4]. The search can be based on a graph which is either defined implicitly with an infinite amount of nodes as in [3], [4], or explicitly with a finite number of nodes as in [2]. We would like to focus our review of related work on methods which, like the work at hand, have been designed for driving in moving traffic within an environment which is structured by roads. Here, combinatorial aspects are often of minor importance, while smoothness, dynamics, optimality and the ability to treat moving obstacles become relevant.

Nevertheless, combinatorial methods have been applied to dynamic street scenarios [5], [6]. From our own experience in [5], we conclude that this bears some difficulties in practice. The requirements call for augmenting the search space with a time axis and with higher derivatives of the system state. If high accuracy is required, the resulting discrete search space, and with it the search time, can become prohibitively large. Hence, in practice, accuracy is limited by the necessarily low sampling density of the state space.

In Werling et al's method [7], the best trajectory is selected from a finite set. The set of candidate trajectories consists of s-shaped swerve trajectories, which align the vehicle on a path parallel to the road centerline after a finite time. This method can be considered as a simplified combinatorial one, where the graph is reduced to a star, and all paths have length 1. This limits the generality of maneuvers that can be planned in an anticipatory way, but drastically reduces the combinatorial complexity. A similar strategy was used in [8].

In this work, we abandon the combinatorial approach and resort to a continuous optimization scheme where the work space does not have to be discretized. Hence, there is no inherent inaccuracy due to the discretization limit, and the method does not suffer from a complexity which rises exponentially with the dimensionality of the state space.

III. METHODOLOGY

A. Preliminaries

The trajectory is planned in a global coordinate system. Hence, a precise, video based localization is required which

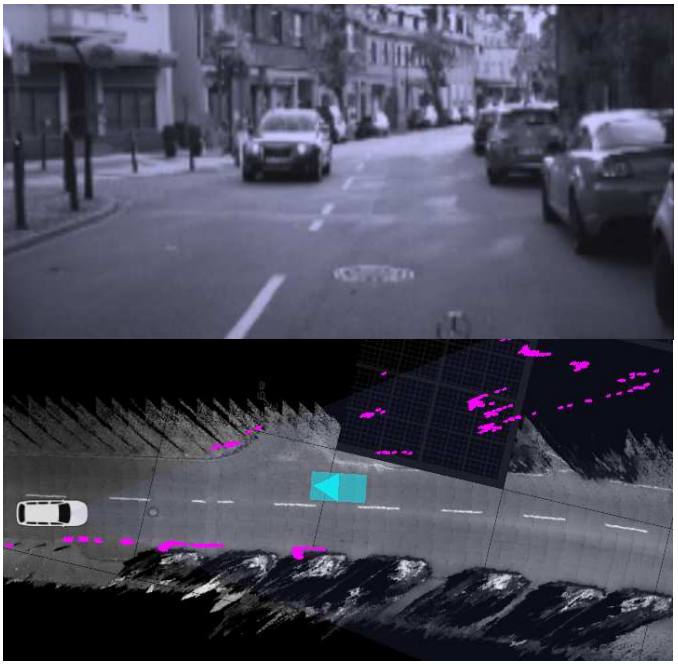


Figure 1: Obstacle- and object data (bottom) with matching video frame (top).

has been provided by [9], [10], [11]. Since the trajectory is referenced with respect to some global coordinate system, we can make use of rich maps that have been created offline. For trajectory planning, the main information provided by these map is the position of a left and a right bound, which together specify a driving corridor. A detailed description of the map contents and the mapping process can be found in [12]. As soon as a trajectory is available, a feedback controller stabilizes the vehicle along it. For this, the pose from the localization system is fed back to minimize the lateral and longitudinal offset towards the trajectory. The control strategy employed has been described in [7].

Obstacle- and object data is provided by stereo vision and radar sensors. Franke et al. give an account on BERTHAs perception system in [13]. Stereo- and radar-data is preprocessed by a data fusion- and validation stage and arrives at trajectory planning in two supplementary representations: Static *obstacles* are provided as a set of stixels [14], which are the result of scanning the disparity image for vertical structures. For every image column, the position of the closest such structure is provided. Moving objects are provided as a list of rectangular *objects* which also have a velocity vector assigned. Fig. 1 shows an example of the perception system. Static obstacles are displayed in magenta, the oncoming object as a box in cyan blue. BERTHA is represented in white.

B. Objective function

The trajectory planner computes an optimal trajectory $\mathbf{x}(t) = (x(t), y(t))^{\mathrm{T}}$ for the rear axle center point of the vehicle. The tangent angle $\psi(t)$ and the curvature $\kappa(t)$ of the trajectory are defined as

$$\begin{aligned} \psi(t) &= \arctan \frac{\dot{y}(t)}{\dot{x}(t)} \\ \kappa(t) &= \frac{\dot{x}(t)\ddot{y}(t) - \dot{y}(t)\ddot{x}(t)}{\sqrt[3]{\dot{x}^2 + \dot{y}^2}}. \end{aligned}$$

The optimal trajectory is defined as the one that minimizes the integral

$$J[\mathbf{x}(t)] = \int_{t_0}^{t_0+T} L(\mathbf{x}, \dot{\mathbf{x}}, \ddot{\mathbf{x}}, \dddot{\mathbf{x}}) \mathrm{d}t, \tag{1}$$

with

$$L = j_{\mathrm{offs}} + j_{\mathrm{vel}} + j_{\mathrm{acc}} + j_{\mathrm{jerk}} + j_{\mathrm{yawr}}.$$

The time t_0 is the current time and T is the preview horizon, which ideally approaches ∞. In practice, it is replaced by a finite preview horizon which is chosen as large as computation time allows. We now discuss the individual summands of the integrand L. All summands contain a weighting factor w_{offs}, w_{vel} etc.

$$j_{\mathrm{offs}}(\mathbf{x}(t)) = w_{\mathrm{offs}} \left| \frac{1}{2}(d_{\mathrm{left}}(\mathbf{x}(t)) + d_{\mathrm{right}}(\mathbf{x}(t)) \right|^2$$

is the term to make the trajectory pass in the center between the left and right bounds of the driving corridor. The functions d_{left} and d_{right} are the signed distance functions towards the bounds of the driving corridor, the distance being positive for all points left of the bound, and negative for all the points to the right. We will address the concrete implementation of these distance functions later in section III-E. The term

$$j_{\mathrm{vel}}(\mathbf{x}(t)) = w_{\mathrm{vel}} \left| \mathbf{v}_{\mathrm{des}}(\mathbf{x}(t)) - \dot{\mathbf{x}}(t) \right|^2$$

contains the quadratic error of the velocity vector of the trajectory compared to a reference velocity vector $\mathbf{v}_{\mathrm{des}}$. The absolute value v_{des} of $\mathbf{v}_{\mathrm{des}}$ is determined by the behavior generation and corresponds to the speed-limit extracted from the map. The direction of the velocity vector is orthogonal to the gradient of the distance functions of the corridor, such that the target direction is parallel to the bounds of the corridor . It is

$$\mathbf{v}_{\mathrm{des}}(\mathbf{x}) = v_{\mathrm{des}} \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \frac{1}{2}(\nabla d_{\mathrm{left}}(\mathbf{x}) + \nabla d_{\mathrm{right}}(\mathbf{x})).$$

The two terms described so far specify the desired behavior of the trajectory: it should run in the middle of the driving corridor and make progress along it at a specified velocity. They act against the following smoothness terms, which are motivated by driving dynamics an comfort. The term

$$j_{\mathrm{acc}}(\mathbf{x}(t)) = w_{\mathrm{acc}} \left| \ddot{\mathbf{x}}(t) \right|^2$$

suppresses strong acceleration in the transverse and longitudinal directions, and thus the forces acting on the passengers. The jerk term

$$j_{\mathrm{jerk}}(\mathbf{x}(t)) = w_{\mathrm{jerk}} \left| \dddot{\mathbf{x}}(t) \right|^2 \tag{2}$$

further smoothness the trajectory by dampening rapid changes in acceleration. The suppression of acceleration and jerk alone will not prevent rapid changes of direction that occur when driving along the trajectory. For this purpose, we introduced a term into the functional which attenuates high yaw rates:

$$j_{\mathrm{yawr}}(\mathbf{x}(t)) = w_{\mathrm{yawr}}\dot{\psi}(t)^2. \tag{3}$$

The objective *functional* (1) is transformed to a *function* by using the method of finite differences, as we will demonstrate now. For the sake of clarity, we will assume $w_{\mathrm{jerk}} = 0$, such that L is no longer dependent on the third derivative of $\mathbf{x}(t)$. The trajectory $\mathbf{x}(t)$ is approximated by N points $\mathbf{x}_i = \mathbf{x}(t_i)$, which are sampled at equidistant times

$$t_i = t_0 + ih, 0 \leq i < N, \tag{4}$$

with a sampling step width of h. The time derivatives of $\mathbf{x}$ are approximated by differences of the sampling points,

$$\dot{\mathbf{x}}(t_i) \approx \mathbf{x}_{\mathrm{d}} = \frac{\mathbf{x}_{i+1} - \mathbf{x}_{i-1}}{2h}, \tag{5}$$

$$\ddot{\mathbf{x}}(t_i) \approx \mathbf{x}_{\mathrm{dd}} = \frac{\mathbf{x}_{i-1} - 2\mathbf{x}_i + \mathbf{x}_{i+1}}{h^2}. \tag{6}$$

Now, an approximation of the integral $J[\mathbf{x}(t)]$ can be expressed as the finite sum

$$J_{\mathrm{d}}(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{N-1}) = \sum_{i=1}^{N-2} L(t_i, \mathbf{x}_i, \mathbf{x}_{\mathrm{d}}, \mathbf{x}_{\mathrm{dd}})h. \tag{7}$$

The variational problem of minimizing (1) has now been converted to the ordinary extremum problem of minimizing (7), where $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{N-1}$ are the free variables. We will refer to the $\mathbf{x}_i = (x_i, y_i)^\intercal$ as the *trajectory support points*. By slight abuse of notation, we will occasionally refer to J_{d} as a function

$$J_{\mathrm{d}}(\mathbf{X}) : \mathbb{R}^{2N} \longrightarrow \mathbb{R}, \tag{8}$$

where the argument vector $\mathbf{X} = (x_0, y_0, x_1, y_1, \ldots)^\intercal$ has been stacked from the single support point coordinates.

Insights from the field of linear quadratic (LQ) optimal control [15] can be used to chose the weights w_{offs}, w_{vel}, w_{acc} and w_{jerk} in such a way that overshoot or oscillation are avoided in the result trajectory.

C. Constraint functions

The optimal trajectory must minimize the energy functional (1), but, at the same time, obey a set of constraints. Constraints can be separated into two classes, internal and external constraints.

Internal constraints result from limits of the vehicle kinematics and dynamics. At low speeds, the curvature of the trajectory is limited by the steering geometry of the vehicle, so

$$-\kappa_{\max} \leq \kappa(t) \leq \kappa_{\max}. \tag{9}$$

At higher velocity, driving dynamics usually become bound by the friction limit of the tires. This limit can be thought of as a circle of forces [16], and at any time,

$$\|\ddot{\mathbf{x}}(t)\|^2 \leq a_{\max}^2 \tag{10}$$

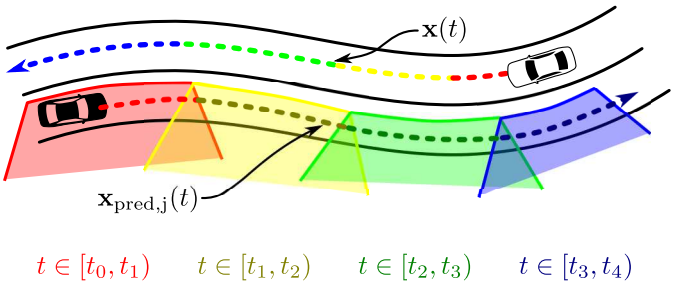


Figure 2: Polygonal constraints for moving objects. Only a part of the trajectory $\mathbf{x}(t)$ is constrained by each polygon.

must hold. Both equations (9) and (10) are implemented by using the approximate derivatives (5), (6) at all t_i, hence, we yield $2n$ inequations for (9) and n for (10).

External constraints are imposed by the driving corridor (which is taken from the map) and by obstacles and objects (which are detected at runtime by sensors). All external constraints are formed as polygons. For setting up a constraint equation from a polygon p, we require a distance function $d(p, \mathbf{x})$ which gives the distance of a point $\mathbf{x}$ to the polygon boundary. The distance function is signed, *i.e.* it is negative for all points inside p and positive for all points outside of it. Later in section III-E, we will demonstrate how the distance function was implemented concretely. If objects are moving, their position is predicted into the future and they will form a different polygon for every discrete time interval. Fig. 2 illustrates schematically how polygons are created for moving objects, where $\mathbf{x}_{\mathrm{pred},j}$ is the predicted trajectory of the jth moving object. All polygons that belong to the time interval $[t_i, t_{i+1})$ are put into the set P_i. If a polygon represents a static object, it will be in all sets P_i, $0 \leq i < N$.

To assert freedom of collision, the vehicles shape must be considered. We decompose the vehicle shape into k circles of radius r_{veh}, which are laid out equidistantly along its longitudinal axis as depicted in Fig. 3a. The reference point for the trajectory is the rear axis center point, which is indicated as a broken line in the figure. The rear-most circle center is $l_{\min}$ behind the rear axle, the foremost is $l_{\max}$ in front of it. Fig. 3b illustrates how the center points of the circles are stretched out to cover a single, infinitesimal segment of the trajectory, $\overline{\mathbf{x}_i\mathbf{x}_{i+1}}$. Fig.3c shows the approximation of the complete maneuver area for an example trajectory. Altogether, the maneuver area is composed from $(N-1)k$ circles. For later reference, we will denote the set of the circle center points as C_{maneuver}.

Further constraint inequations are now created from the sets P_i of all constraint polygons, $0 \leq i < N$, by combining them with all circles,

$$d(p, \mathbf{x}_{\mathrm{c}}) \geq r_{\mathrm{veh}}, p \in P_i, \mathbf{x}_{\mathrm{c}} \in C_{\mathrm{maneuver}}. \tag{11}$$

This creates further $(N-1)ko$ inequations, where o is the number of obstacles. The overall number of constraints is $M = (N-1)ko + 3N$.

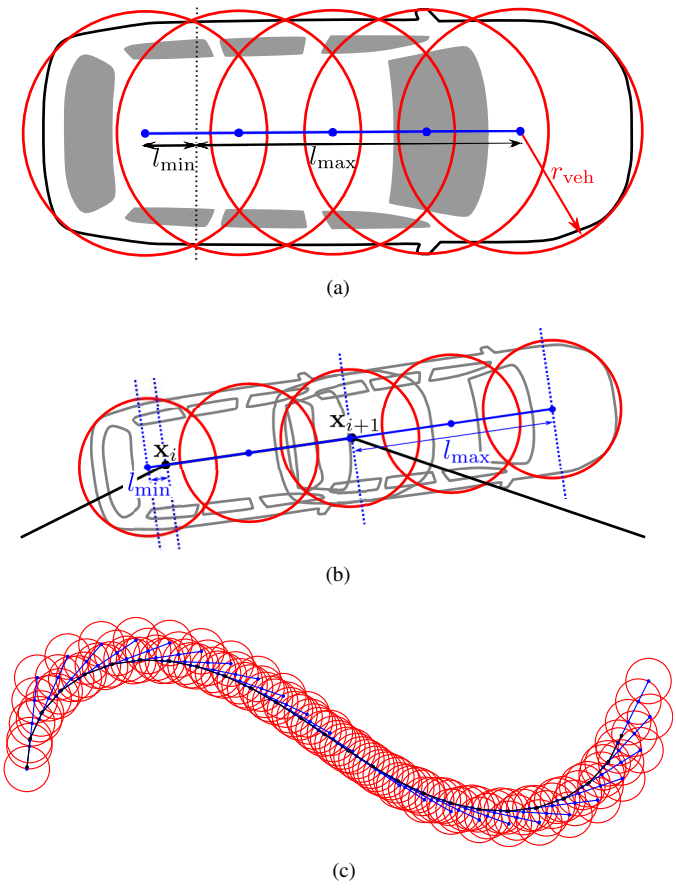


Figure 3: Circle decomposition of vehicle shape and maneuver area

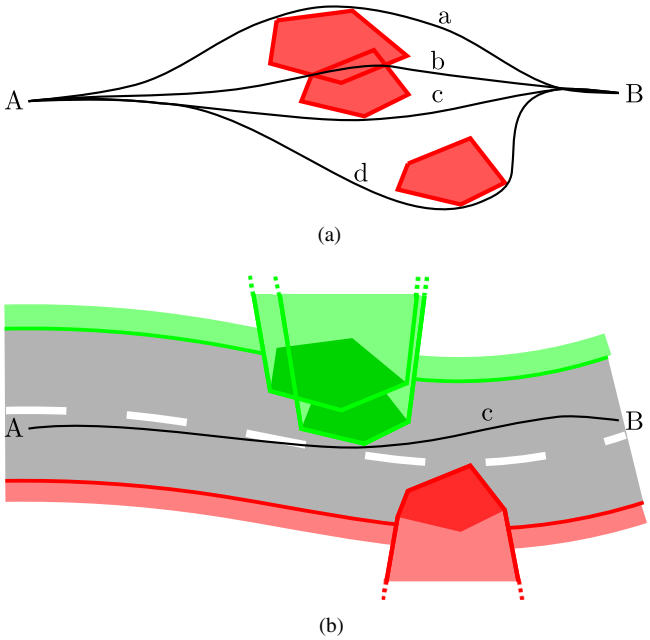


Figure 4: Stationary points of trajectory optimization (a) without (b) with decision process.

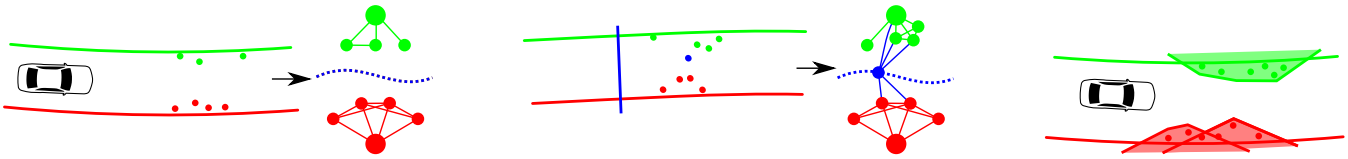
For convenience of notation, we rearrange the M individual constraint inequations (9), (10), (11) such that the right hand side becomes 0, and that the relation becomes "$\geq$". We will consistently refer to the left hand sides of these M inequations as the *constraint functions* $f_j(\mathbf{x}_0, \ldots, \mathbf{x}_{N-1})$, $0 \leq j < M$. Occasionally, we will refer to the vector valued constraint function $f(\mathbf{X}) : \mathbb{R}^{2N} \longrightarrow \mathbb{R}^M$ which is created by stacking all f_j, and by stacking the support points into a $2N$-vector $\mathbf{X} = (x_0, y_0, x_1, y_1 \ldots)^\intercal$.

D. *Building constraint polygons from sensor data*

As depicted earlier, constraints are brought into trajectory planning in the form of polygons. This section illustrates how the polygons are generated from the raw sensor measurements.

We will use a local optimization scheme to find the optimal trajectory. Hence, convergence to a global optimum cannot be guaranteed per se. The objective function (7) is essentially quadratic and thus, has one well defined stationary point which is the local minimum. However, multiple stationary points can be induced by the obstacle constraints. Consider Fig. 4a, which exemplifies the problem schematically. A trajectory that optimizes some kind of smoothness criterion is to be planned from A to B, and the free space is constrained by three polygons (red). The trajectories a,b,c,d are the stationary points of this problem. Especially interesting is trajectory b, which is not valid, because it clearly violates two constraints. However, locally, none of the constraint violations can be diminished without making the other constraint violation worse. Hence, b is a stationary point of the constrained optimization problem. Fig. 4b shows the same situation but it has been decided prior to optimization which obstacles are to be passed on the left (red) and which ones on the right hand side (green). The decision is greatly simplified by inspecting the run of the road, which has been provided by the map-based driving corridor, *cf.* sec. III-A. The left/right decision has been incorporated into the shapes of the polygons, by extending them infinitely to one side of the workspace. The extensions are tapered slightly towards the side that the trajectory is supposed to pass on. Note that the modified problem has exactly one stationary point, which is also the global optimum. Trajectory optimization will converge to the optimum, independent of where it has been initialized.

To build the constraint polygons, first a decision is made on the basis of single obstacle points. A minimum vertex graph cut is used for this. The structure of the graph to be cut is illustrated schematically in figures 5a and 5b. Each individual stixel correspond to a node in the graph. The two larger nodes represent the left and right bound of the driving corridor. Two nodes are connected if it is geometrically infeasible to pass between the corresponding stixels, or between the corresponding stixels and the respective corridor bound, without collision. The graph will now be cut into two sections. This is done in a minimal way, *i.e.* by removing the smallest possible amount of nodes required to separate the left and and right corridor bounds. In Figure 5a, the cut set is empty, because the corridor



(a) (b) (c)

Figure 5: Building constraint polygons from sensor data.

bounds were not connected initially. In figure 5b, the graph is cut by removing one node (blue). If the cut set is not empty, passage at this point is not possible, and a stopping maneuver is induced by putting a stop line constraint (blue).

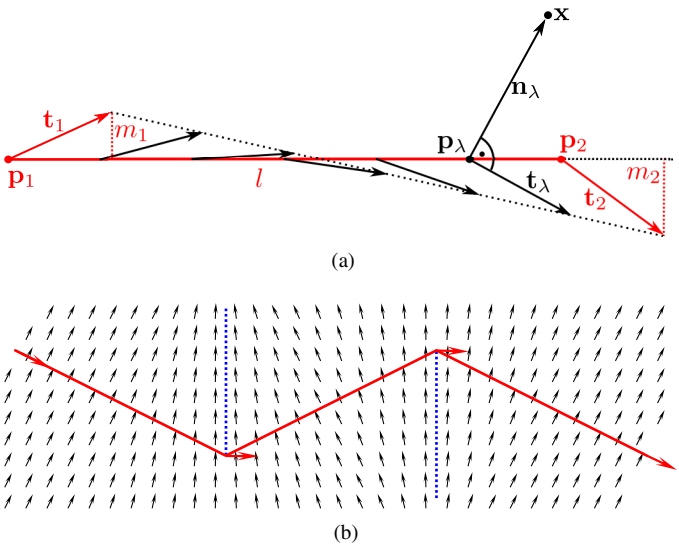
After the decision has been made for every single stixel, polygons are created by enveloping the stixels inside polygonal hulls, *cf.* Fig. 5c. Construction of the polygonal hulls asserts that all single polygons are convex (non-convex polygons can contribute to an indefinite Lagrangian). Enveloping stage is also used to add a safety distance to the obstacles.

For moving obstacles, left-right decision and geometric processing are different. Firstly, left-right decision is much simpler, because the direction of motion gives a good indication of whether a vehicle is to be passed on the left or right side. Based on their direction of motion, all vehicles are classified as either oncoming or as potentially overtakeable (*overtakees*). All oncommers are passed on the right, all overtakees are passed on the left. As already indicated in section III-C and Fig. 2, building constraint polygons for moving objects involves predicting their future poses. In our application and sensor setup, the best cue for this is the run of the road, which we can extract from the map. For trajectory prediction, a simple routing in the map is performed for every moving obstacle, to obtain a set of corridors that the other object will potentially use for traveling. A prediction is then done under the assumption that the other vehicle maintains its speed and its offset from the right corridor bound.

E. Distance function

For the constraint inequations (11), the signed distance to a polygon must be determined. Since we will use a Newton-type optimization method, it is desirable that all constraint functions are continuously differentiable. For the EUCLIDEAN distance to a non convex polygon, this can not be guaranteed. In this section, we will introduce a *pseudo distance* that has the desired properties. We define a single linear segment of a polygon as the tuple $G = (\overline{\mathbf{p_1 p_2}}, \mathbf{t_1}, \mathbf{t_2})$, where $\mathbf{p}_1$ and $\mathbf{p}_2$ are two adjacent corner points of the polygon, and $\mathbf{t}_1$,$\mathbf{t}_2$ are predefined tangent vectors at the corner points. The tangent vectors can, *e.g.*, be chosen as the difference of neighboring points, just like in (5). A pseudo tangent vector is now created by interpolating the corner tangents linearly along the length l of the segment,

$$\mathbf{t}_\lambda = \lambda \mathbf{t}_2 + (1-\lambda)\mathbf{t}_1$$



(a)

(b)

Figure 6: Pseudo distance and pseudo gradient field.

which is the pseudo tangent vector at the point

$$\mathbf{p}_\lambda = \lambda \mathbf{p}_2 + (1-\lambda)\mathbf{p}_1,$$

for $\lambda \in [0,1]$. For projecting a point $\mathbf{x} = (x,y)^\intercal$ onto G, we determine λ such that the pseudo tangent is perpendicular to to the pseudo normal vector $\mathbf{n}_\lambda = \mathbf{x} - \mathbf{p}_\lambda$, thus

$$\mathbf{n}_\lambda \mathbf{t}_\lambda = 0, \tag{12}$$

compare Fig. 6a. Without loss of generality, we assume that $\mathbf{p}_1 = (0,0)^\intercal$ and $\mathbf{p}_2 = (l,0)^\intercal$, as is the case in Fig. 6a. In this case, the corner tangents can be expressed in terms of their slopes m_1 and m_2 thus $\mathbf{t}_1 = (1,m_1)^\intercal$ and $\mathbf{t}_2 = (1,m_2)^\intercal$. In this case, λ can be easily solved for,

$$\lambda = (m_1 y + x)/(m_1 - m_2)y + l.$$

The signed pseudo distance d to the segment is $\|\mathbf{n}_\lambda\|$ if $y > 0$ and $-\|\mathbf{n}_\lambda\|$ else. For the sake of simplicity, instead of using the true gradient of d, we use the vector $\frac{\mathbf{n}_\lambda}{|d|}$ as the pseudo gradient of d. To determine the distance to a complete polygon, we compute the distance to every linear segment in the polygon and pick the smallest one. Fig. 6b depicts the field of pseudo gradients (black) for an exemplary poly line (red). The dashed blue lines are the points where the true, Euclidean, distance to the poly line would have a discontinuous gradient.

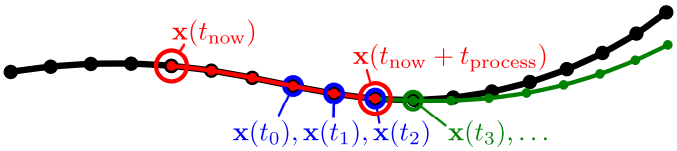


Figure 7: Continuous re-planning with C_2 continuity.

F. Re-planning scheme

As the vehicle drives along, more and more information becomes available through sensors. Hence, the trajectory must be continuously re-planned to adapt to the new sensor data. During re-planning, it is important that the trajectory is kept smooth over the transition points, so that no steps are generated on the steering input of the vehicle. The steering wheel angle is related directly to the curvature, and hence the second derivative of the trajectory. Thus, we strive for third level parametric continuity (C_2-continuity) of the trajectory.

In Fig. 7, the trajectory of the last iteration is displayed in black. It has become available just now, at time t_{now}. For the sake of clarity, we will assume that t_{now} and other timestamps in this section match up with one of the times t_i of the finite trajectory support, as introduced in equation (4). From t_{now} on, the vehicle will be guided by the trajectory controller along the red part of the trajectory, until a new trajectory becomes available. Planning a new trajectory will take up some processing time, and we will assume that this time is guaranteed to be less than t_{process}. We must expect that the old trajectory stays valid as control reference until time $t_{\text{now}} + t_{\text{process}}$. To guarantee continuous feed forward control, this part of the trajectory must never be changed. Thus, re-planning can only alter the part of the trajectory that is displayed in green. We know that the green trajectory will be available at latest at $t_{\text{now}} + t_{\text{process}}$.

It must be asserted that the green and the red part of the trajectory are connected smoothly, without a step in acceleration. This can be achieved by binding the first three support points of the newly planned trajectory, $\mathbf{x}(t_0), \mathbf{x}(t_1), \mathbf{x}(t_2)$, to constant values that are consistent with the red part of the trajectory. These *boundary points* are colored blue in Fig. 7, and fixing them is actually equivalent to fixing an initial value for $\mathbf{x}(t_0), \dot{\mathbf{x}}(t_0)$ and $\ddot{\mathbf{x}}(t_0)$ in the original, not discretized, variational formulation from equation (1).

During re-planning, an initial guess for local trajectory optimization is generated from the result of the last optimization.

G. Constrained optimization

The problem of finding the optimal trajectory has now been posed as the constrained optimization problem of finding trajectory support points

$$\underset{\mathbf{x}_3,\ldots,\mathbf{x}_{N-1}}{\arg\min}\ J_{\text{d}}(\mathbf{x}_0, \ldots, \mathbf{x}_{N-1}) \tag{13}$$

such that

$$f_j(\mathbf{x}_0, \ldots, \mathbf{x}_{N-1}) \geq 0, \quad 0 \leq j < M. \tag{14}$$

Note that in (13) and (14), $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2$ are *not* free variables, but bound to constant values as explained in the preceding section.

In a solution or solution candidate $\mathbf{X} = (x_0, y_0, \ldots, x_{N-1}, y_{N-1})^\intercal$ to the optimization problem, the jth constraint is called *active*, if $f_j(\mathbf{X}) = 0$ and *inactive* if $f_j(\mathbf{X}) > 0$.

The problem is solved by means of a sequential quadratic programming (SQP) method [17]. This method builds on the method of Lagrange multipliers [18] which are a vector $\mathbf{l} = (l_0, \ldots, l_{M-1})^\intercal$ of M auxiliary variables, one for every constraint. The function

$$L(\mathbf{x}_3, \ldots, \mathbf{x}_{N-1}, l_0, \ldots, l_{M-1}) = J_{\text{d}}(\mathbf{x}_0, \ldots, \mathbf{x}_{N-1}) + \mathbf{l}^\intercal f(\mathbf{X}) \tag{15}$$

is called the *Lagrangian* of the constrained optimization problem. The Lagrange multipliers of all inactive constraints are set to 0. Every solution to the optimization problem must be a stationary point of L. In SQP, L is locally approximated by a quadratic function through its Hessian and gradient, while f is approximated by a linear function. Instead of using a quasi-Newton method where the Hessian of the Lagrangian is replaced by an iteratively computed approximation, we suggest to compute the true Hessian instead. The rationale for this is twofold and aims at runtime performance.

Firstly, the objective function J_{d} is actually almost quadratic, the only non-quadratic term being that for yaw rate, (3) (which nevertheless approaches linearity when the solution approaches a parametrically smooth trajectory). Hence, when all constraints are inactive, and neglecting the yaw rate term, a true Newton method will find the extremum immediately, after a single iteration. Hence, by using the true Hessian, we expect to reduce the number of iterations required for convergence significantly.

Secondly, in the problem at hand, the Hessian of the Lagrangian has a sparse, banded structure. This can be seen from inspecting the single summands of J_{d}, which only depend on a small neighborhood of support point coordinates. The same applies to all constraint functions f_j and consequently, to the Lagrangian (15). This locality property translates to a sparsity property of the Hessian of L, where only values in a band around the diagonal are $\neq 0$. In the problem at hand, the bandwidth is 7. The sparsity of the Hessian can be exploited for performance. Firstly, only $7(N + M)$ entries of the Lagrangian have to be computed in the first place, instead of $(N + M)^2$. Secondly, in a single iteration of SQP, the Lagrangian forms the left hand side of a linear system of equations that has to be solved. For a banded matrix, this requires $\mathcal{O}(N)$ operations instead of $\mathcal{O}(N^3)$. Thus, by using the true Hessian, the cost of a single iteration of SQP can be reduced significantly.

IV. Experiments

In August of 2013, the proposed trajectory planner was used on-board the Bertha vehicle to complete the 100 km of the Bertha-Benz-Memorial-Route in multiple sections. The

experiment was conducted in public traffic. The amount of support points was $N = 30$, sampled at a time step of $h = \frac{1}{3}$s, yielding a preview horizon of $T = 10$s. Iteration of the optimizer was terminated after $t_{\text{process}} = 0.5$s, 20 iterations or when no progress was made towards the optimum, whichever event occurred first. Fig. 8 shows some example trajectories computed by the method. Note that these are not tracks of the vehicle position, but the anticipatory trajectories planned at a single time instant. Fig. 8a shows a fast passage through a roundabout. It gives a good impression on the dynamic capabilities of the method. The apexes at entry, midpoint and exit of the roundabout are clipped tightly, which minimizes lateral accelerations. Fig. 8b, 8c and 8d show passages through the same right turn, with obstacles placed at different positions. The blue crosses are positions of single stixels, accumulated over 5 frames, and the blue polygons are their polygonal envelopes (*cf.* Sec. III-D). A smooth trajectory is found, even with the little free space available. Fig. 8e shows a swerve maneuver towards a slower, oncoming vehicle. The predicted trajectory of the oncoming object is depicted in dark blue, and the resulting obstacle polygons in light blue. The obstacle polygons are time dependent, as has been previously illustrated in Fig. 2. The circle indicates the moment where the two vehicles encounter.

V. Discussion, Conclusions and Outlook

The trajectory planner proposed here has proven very capable of tackling the various challenges encountered along the BBMR. The essential design decisions where the following: Careful design of a single objective function, careful arrangement of polygonal constraints, and attention to detail in the numerical optimization scheme. The resulting system is very universal, and implements many maneuvers naturally. Without any outside heuristics, the vehicle slows down in tight bends or when turning, and cuts through corners smoothly. The driving style has been described as pleasant and natural by passengers. Nevertheless, we would like to point out some directions for improvement.

We started from the proposition that combinatorial aspects are of minor importance in on-road driving. However, during the BBMR project, we learned that situations that require simple combinatorial skills do occur in everyday driving. Consider, *e.g.*, the schematical Fig. 9. The road is partially blocked by a black obstacle. We are planning a trajectory for the red vehicle traveling from left to right. The black vehicle is approaching from the right and will form a critical section (CS) with the obstacle. It will block the passage from time t_{enter} through t_{exit}. Assume that in our local method, the trajectory is initialized to the black curve. The positions $\mathbf{x}(t_{\text{enter}})$ and $\mathbf{x}(t_{\text{exit}})$ are highlighted by green and blue circles on the trajectory. Two discrete classes of solutions can be identified for this situation. Either, we will pass the CS before the black vehicle enters it, or we wait in front of the CS until the black vehicle exits it. The local method presented in this paper is not guaranteed to converge in the correct solution class. It is also likely to get trapped in an invalid state, similar to trajectory

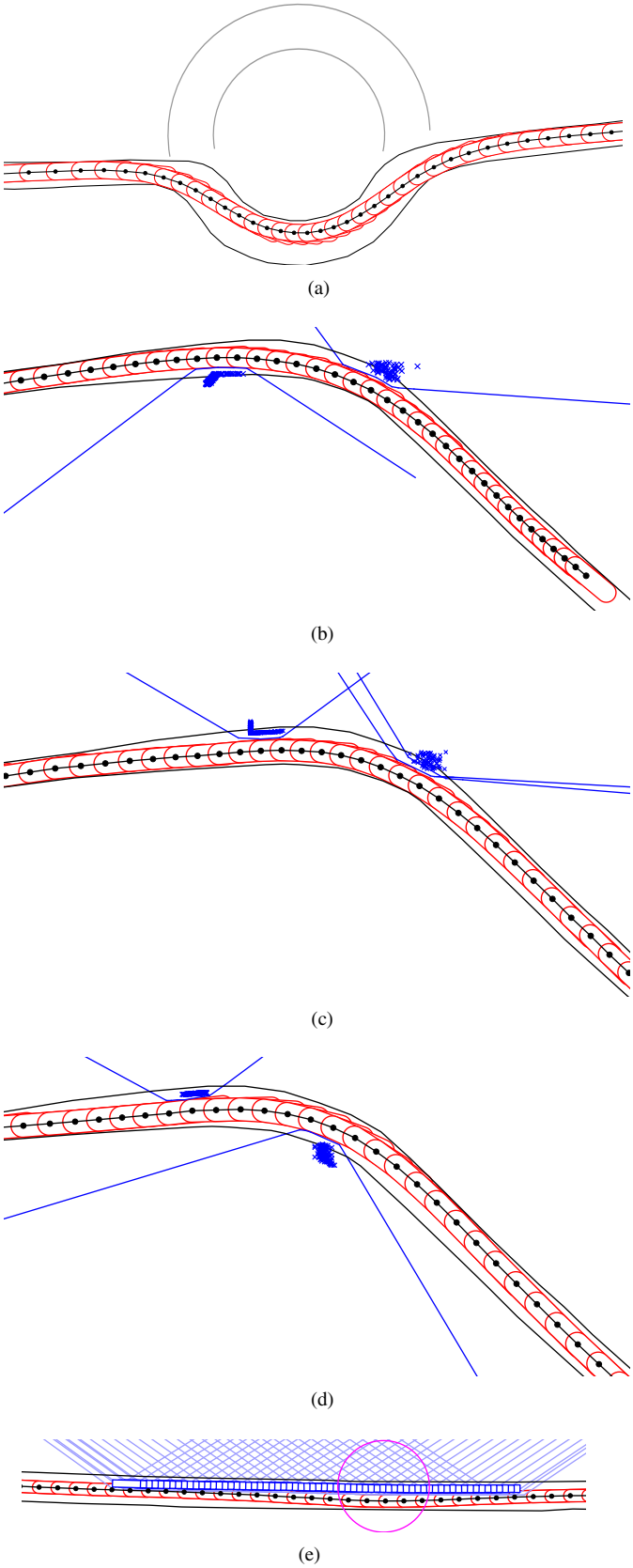


Figure 8: Example trajectories. Traveling direction is from left to right. Maneuver area of the vehicle in red, obstacles and constraint polygons in blue, boundary constraints in black.

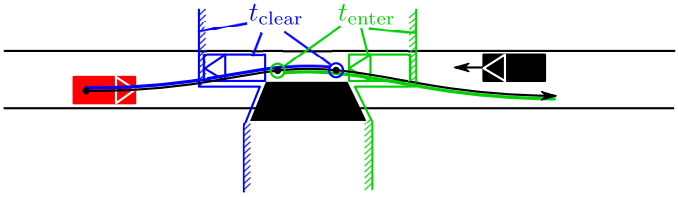


Figure 9: A critical section.

"b" in Fig. 4a. For the BBMR challenge, this situation was remedied by deciding ad hoc for one of the solution classes, based on the distance and speed of the oncomer towards the obstacle. Based on the decision, the constraint system was altered: Either the blue part of the trajectory in Fig. 9 is constrained to be in front of the hatched blue line, *or* the green part is constrained to be behind the hatched green line. Similar ad hoc decisions had to be implemented for merge maneuvers. In principle, this scheme can be extended to an optimal procedure, by enumerating the constraint systems for all discrete solution classes and trying the local solver on all of them. Then, the smallest cost solution is picked. This scheme is computationally expensive, however. A future direction of research will be a more systematic approach to detect and deal with situations that call for such discrete decisions.

think of all possible situations?

In summary, it can be said that a continuous, local method *must* be complemented by a combinatorial one if is meant to be fully universal. However, the by far largest part of everyday driving situations does *not* require combinatorial reasoning at all, and in these situations, a purely local method produces excellent results.

References

[1] M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," tech. rep., Computer Science Dept., Iowa State University, 1998.

[2] M. Pitvoraiko and A. Kelly, "Efficient constrained path planning via search in state lattices," in *International Symposium on Artificial Intelligence, Robotics, and Automation in Space*, 2005.

[3] J. Ziegler, M. Werling, and J. Schröder, "Navigating car-like vehicles in unstructured environment," in *Intelligent Vehicles Symposium*, IEEE.

[4] J. Barraquand and J.-C. Latombe, "Nonholonomic multibody mobile robots: Controllability and motion planning in the presence of obstacles," *Algorithmica*, vol. 10, pp. 121–155, 1993.

[5] J. Ziegler and C. Stiller, "Spatiotemporal state lattices for fast trajectory planning in dynamic on-road driving scenarios," in *International Conference on Intelligent Robots and Systems*, IEEE, 2009.

[6] M. McNaughton, C. Urmson, J. M. Dolan, and J.-W. Lee, "Motion planning for autonomous driving with a conformal spatiotemporal lattice.," in *ICRA*, pp. 4889–4895, IEEE, 2011.

[7] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, "Optimal trajectory generation for dynamic street scenarios in a frenet frame," in *IEEE International Conference on Robotics and Automation*, pp. 987–993, 2010.

[8] 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, pp. 325–345, 2008.

[9] H. Lategahn, M. Schreiber, J. Ziegler, and C. Stiller, "Urban localization with camera and inertial measurement unit," in *Intelligent Vehicles Symposium*, pp. 719–724, IEEE, 2013.

[10] M. Schreiber, C. Knöppel, and U. Franke, "LaneLoc: Lane marking based localization using highly accurate maps.," in *Intelligent Vehicles Symposium*, pp. 449–454, IEEE, IEEE, 2013.

[11] J. Ziegler, H. Lategahn, M. Schreiber, C. G. Keller, C. Knöppel, J. Hipp, M. Haueis, and C. Stiller, "Video based localization for Bertha," in *Intelligent Vehicles Symposium*, (Dearborn, Michigan, USA), IEEE, 2014.

[12] P. Bender, J. Ziegler, and C. Stiller, "Lanelets: Efficient map representation for autonomous driving," in *Intelligent Vehicles Symposium*, (Dearborn, Michigan, USA), IEEE, 2014.

[13] U. Franke, D. Pfeiffer, C. Rabe, C. Knöppel, M. Enzweiler, F. Stein, and R. G. Herrtwich, "Making Bertha see," in *IEEE ICCV Workshop Computer Vision for Autounomous Vehicles*, (Sydney, Australia), pp. 1–10, December 2013.

[14] D. Pfeiffer and U. Franke, "Towards a global optimal multi-layer stixel representation of dense 3D data," *BMVC, Dundee, Scotland. BMVA Press (August 2011)*, 2011.

[15] B. D. O. Anderson and J. B. Moore, *Optimal Control: Linear Quadratic Methods*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1990.

[16] H. B. Pacejka, *Tire and Vehicle Dynamics*. Society of Automotive Engineers, Inc., 2nd ed., 2006.

[17] M. Bartholomew-Biggs, "Constrained minimization using recursive quadratic programming," in *Numerical Methods for Nonlinear Optimization* (F. Lootsma, ed.), pp. 411–428, Academic Press, 1972.

[18] L. Lasdon, *Optimization Theory for Large Systems*. Dover Books on Mathematics Series, Dover Publications, 1970.