2015 IEEE Intelligent Vehicles Symposium (IV)
June 28 - July 1, 2015. COEX, Seoul, Korea

The Combinatorial Aspect of Motion Planning: Maneuver Variants in Structured Environments

Philipp Bender[1], Ömer Şahin Taş[1], Julius Ziegler[2] and Christoph Stiller[1]

***Abstract*— Motion planning plays a key role in autonomous driving. In this work, we introduce the *combinatorial aspect* of motion planning which tackles the fact that there are usually many possible and locally optimal solutions to accomplish a given task. Those options we call *maneuver variants*. We argue that by partitioning the trajectory space into discrete solution classes, such that local optimization methods yield an optimum within each discrete class, we can improve the chance of finding the global optimum as the optimum trajectory among the manuever variants. This work provides methods to enumerate the maneuver variants as well as constraints to enforce them. The return of the effort put into the problem modification as suggested is gaining assuredness in the convergency behaviour of the optimization algorithm. We show an experiment where we identify three local optima that would not have been found with local optimization methods.**

I. Introduction

Motion planned for a vehicle must on one hand lead the vehicle to its destination and on the other hand obey the constraints set on motion. These constraints typically arise due to the physical limits of the vehicle and the existing traffic regulations. While such constraints generally aim to maintain a feasible ride, measures of, for example, comfort and agility define the optimality of the planned motion. As optimality is of great interest, factors effecting the arbitrarily chosen optimality criteria must be analyzed.

Fundamentally, two issues play a decisive role on the optimality of a planned motion. First, and yet probably the most addressed one is the accuracy of the calculation. Great extent of methods applied for autonomous vehicle motion planning can be grouped under either discretized, or under sampling-based methods [1], [2], [3], [4]. These sacrifice optimality against real time performance. A remedy is to refer to continuous methods. An approach among continuous methods that found practical application in autonomous driving is to model the motion planning problem as a constrained optimization problem [5].

The second issue is the combinatorial aspect of motion planning. If there are obstacles in the environment, which may be either static or dynamic, distinct maneuver variants emerge. Within the context of this work, dynamic obstacles in structured environments are of special interest. A trivial example for a dynamic obstacle occluded structured environment is illustrated in Figure 1. Here, we can identify three possibilities for the white car to traverse the scene. Regarding the costs of the maneuver, we intuitivelly expect each of the variants to have its own local minimum, where one of them corresponds to the global minimum. Therefore, it should be underlined that the global minimum is only attainable when the most favorable is chosen.

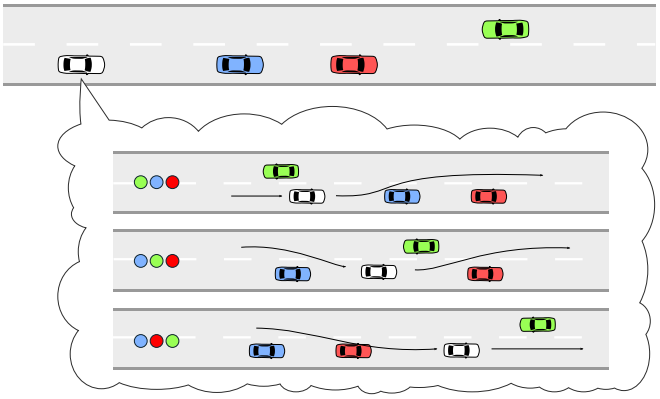


Fig. 1: The combinatorial aspect. In this scene, the white car has three options to overtake the red and blue car. The colored dots denote the order in which the white car passes the other vehicles.

Sampling-based methods intrinsically can evaluate alternative maneuvers in presence of static obstacles [6]. Depending on the applied heuristics, discretized methods can also be extended to handle the alternatives. However, even when these methods choose the most favorable maneuver variant, they will yield suboptimal solutions. The stated continuous methods on the other hand employ local optimization methods for real time tractability. This comes with the drawback that they will converge against the minimum that lies in the vicinity of their initialization point. Clearly, the obtained minimum does not neccessarily correspond to the global optimum. The chosen maneuver variant in Fig. 1 is fully determined by the initialization. This clearly reflects an undesirable case.

Contribution of the Paper

The work at hand will first provide proper abstractions to deal with the combinatorial aspect, showing how to discretize and enumerate the possible maneuver variants in structured environments. Then, we focus on continuous methods and discuss the possibilities to modify the planning problem in order to enforce a particular maneuver variant and thus to evaluate discrete solution classes.

The argumentation in this paper assumes and relies on *perfect information* about the environment. In that respect, as

[1] Philipp Bender, Ömer Şahin Taş and Christoph Stiller are with FZI Research Center for Information Technology, Mobile Perception Systems, Karlsruhe, Germany {pbender,tas,stiller}@fzi.de

[2] Julius Ziegler is with Atlatec, Karlsruhe, Germany ziegler@atlatec.de

well as position, we assume velocity and acceleration profiles of other traffic participants to be known in advance. As a further simplifying assumption, we neglect the influence of the ego vehicle on the motion of other traffic participants.

II. Related Work

Dealing with dynamically moving objects with known trajectory has been faced with the *path velocity decomposition* (PVD) [7]. The idea behind it is to first plan a *path* which is never touched again (relaxed e.g. in [8]), and then plan a velocity profile along this path, hence obtaining a complete *trajectory*. Figure 2 shows this idea. The grey boxes indicate obstacles, and three example trajectories are depicted. It is also noticeable that PVD states a good example on how to abstract a planning problem in a way that exposes the combinatorial aspect – here, each trajectory could either run above or below the individual obstacle boxes.

Recently, efficient planners working in the path×time domain have been introduced, using graph techniques to compute time-optimal trajectories [9], [10]. Those contributions follow a pre-defined path and the planners are shown to be very efficient. However, the problem with this is twofold. First, the resulting trajectories are not necessarily optimal, assuming quality factors including lateral forces and jerk. If the reference path is planned with a high velocity, resorting to lower velocities could preserve dynamical feasibility (slowing down usually does not make things worse, such as friction between tyre and road), but then the trajectory is more conservative than necessary, and maybe behind the driver's expectations. Secondly, and this is the main difference to the scope of this work, in certain maneuvers like overtaking or a lane change, *there is no path* to plan in advance. The velocity profile and the (collision free) path directly depend on each other and are inseparable, see Figure 3 [11], [12].

These drawbacks may be overcome using techniques like spatiotemporal space lattices [13]. Those methods solve the combinatorial problem, but with the disadvantage that the search space quickly grows in dimensions if time and higher derivatives are considered, and each dimension grows with required accuracy.

The most important inspiration for this work is the planning scheme introduced by Ziegler *et al.* [5] which describes a local and continuous method. When we speak of *optimality* of a trajectory, we refer to the optimality criteria defined there. The work presumed that the combinatorial aspects in a moving, structured traffic would be of minor importance. In contrast, it concluded by stating that the integration of combinatorial reasoning into the continuous methods is needed for optimal motion planning [14].

Another study perfomed by Kohlhaas *et al.* suggested a semantic way for representing maneuver variants in structured environments, which fundamentally could serve as a basis for evaluating alternative combinations [15]. However, the study is limited to the semantics and does not comprise any kinematic aspects.

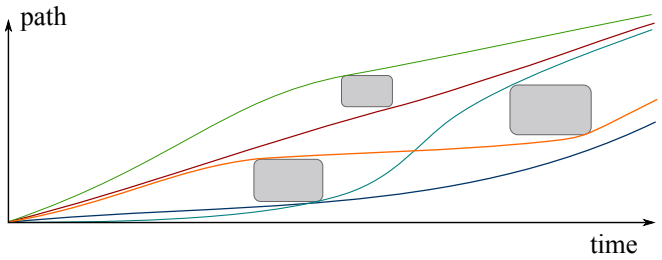


Fig. 2: The *path velocity decomposition* (PVD). Assuming a fixed and pre-planned path, PVD seeks for a velocity profile along this path.

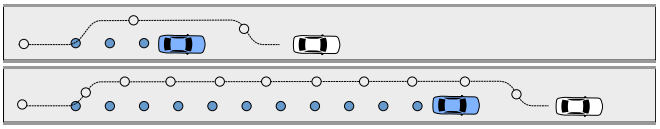


Fig. 3: An example of an overtaking maneuver, a fast one (top figure) and a slower one (bottom). Note that not only the *velocity profiles* (indicated by the colored dots, representing equidistant sampling times), but also the *paths* themselves differ.

It should be remarked that we refrain from using the term *combinatorial motion planning*, as this has previously been used either to define path planning methods that are built on combinatorial search structure [16], or for the multiple depot travelling salesman problem [17]. Therefore, we adopt the terms *combinatorial aspect* and *maneuver variants*.

III. Combinatorics in Motion Planning

To phase in the combinatorial aspect, we will proceed in three steps: First, we consider only static obstacles to get familiar with common problems and to introduce our notation scheme. Then, we will consider *moving* or *dynamic* obstacles, which is slightly more challenging. With the knowledge gained from those steps, we will devote our attention to *multiple dynamic objects* and additional constraints posed on the problem.

A. Maneuver Variants and Where They Come From

1) Static objects: Facing static objects which need to be passed to travel along the road, the vehicle basically has two options. The first option is to pass the vehicle on the left side, the other option is to pass it on the right. Our notation will be ◑ for the white (ego) vehicle passing a red obstacle on the left, or ◐ vice-versa. Intuitivelly we assume that for both options – we say *maneuver variants* – we will find one *best* trajectory.

However, as can be seen from Figure 4, the state space of possible trajectories becomes non-convex due to the obstacle. Hence, the resulting trajectory found by most local optimization techniques depends on the initialization. Sometimes the initialization is near the desired optimum and the local, gradient based method leads us to the global minimum. But

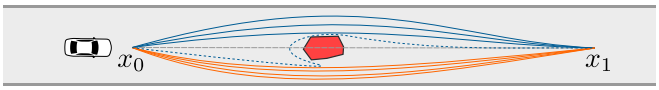


Fig. 4: A static obstacle (red) and the white ego vehicle with an initialization (grey, dashed) and some sample trajectories. The blue ones pass the obstacle on the left side, hence they belong to the maneuver variant ◑. The orange ones are representants of ◐. The dashed blue trajectory represents a slightly degenerated example of ◑.

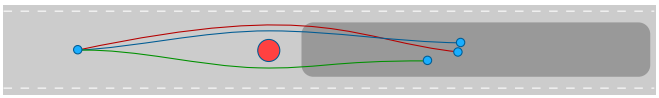


Fig. 5: Homotopy relative to the start point (on the left), and an *area* (instead of a fixed end point). Here, the end points are allowed to be moved inside the darker rounded rectangle, and it is easy to imagine that red and blue belong to the same class, while green does not.

in other cases, when the initialization is not in the vicinity of the desired, we may get stuck at any other local minimum. To avoid this, we seek for a way to *enforce* a certain variant using the instrument of constraints. But first, we need a way to reliably distinguish the two variants.

In the discipline of *topology*, there exists the notion of a *homotopy relative to a subspace* ([18], *Homotopy*). Vividly speaking, the existence of such a homotopy means that two functions can be transformed into each other, only by stretching and compressing, and especially without dissecting. In Figure 4 this would not be possible for a blue and orange one, due to the obstacle. In our context, the subspace is defined by the start and end point of the trajectory. To show the existence of a homotopy of two trajectories relative to the start and end points, the trajectories are interpreted as the mappings $\gamma_{0,1} : [0, T] \to X$, where X is the configuration space, in this example $X = \{x \mid (x \in \mathbb{R}^{x \times y}) \bigwedge (x \notin \mathbb{O})\}$. Here, $\mathbb{O}$ is the space occupied by the obstacle, and T is the planning horizon. Two trajectories are called *homotopic* relative to $\{x_0, x_1\}$ if there is a continuous function $H : [0, T] \times [0, 1] \to X$ so that

$$H(t, 0) = \gamma_0(t) \qquad H(0, s) = x_0$$
$$H(t, 1) = \gamma_1(t) \qquad H(T, s) = x_1$$

However, these definitions are not directly applicable to a trajectory planning problem, since trajectories are usually not required to be equal in their end points, except for the time coordinate. Also, due to limited computing capacities and limited knowledge of the environment, trajectories are not planned ultimately until the destination is reached, but only up to a fixed time horizon. Looking at Figure 4, we can easily imagine one of the orange trajectories to degenerate to a blue one, as soon as we remove the assumption of the fixed end point x_1. What we need here is a heuristic, which helps us to determine which class is best supported by a given

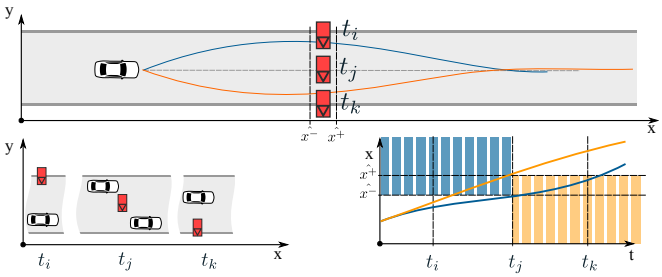


Fig. 6: Obstacle crossing the lane. The top row shows the position of the (red) obstacle at different times $t_{i,j,k}$. The bottom left picture shows the y position for a fixed x over time t, the bottom right shows the resulting constrained time path diagram for the obstacle. For the blue maneuver variant, ◑, the blue area is forbiden, and for the orange variant, ◐, the orange area is excluded from the configuration space.

trajectory, even if the end points are not fixed. A possible solution would be to relax the demand for a fixed end point, like shown in Fig. 5, but the problem of practical infeasibility to use this as an constraint still remains.

For now, we content with the simple assumption that traveling back in the longitudinal direction x is not possible. Then, we can easily exclude points that lie on the left (or right) side of the obstacle.

2) Moving (dynamic) objects: Objects moving only longitudinally are trivial to handle, once a way to handle static objects is found. Instead of restricting all trajectory samples to a certain area, we restrict the area for each sample individually.

More interesting situations arise when lateral movement is considered as well. For example consider an obstacle crossing the road from left to right as shown in Figure 6. The top row shows *the same obstacle* for *different times*. As explained in the section before, we find two maneuver variants to examine, ◑ and ◐. In addition to the spatial implications we had from the static obstacles, we also have to care about *time* and therefore *spatio-temporal* implications. For ◑ to succeed, it is necessary that the ego vehicle waits at $\hat{x}^-$ until the gap between the left bound of the lane and the obstacle is big enough for the vehicle to pass, and vice versa, for ◐ the vehicle has to pass the gap (be further than $\hat{x}^+$) before it is too tight. In the figure, both events (opening and blocking of the gaps) should happen at t_j. For the white vehicle this means that certain areas in the path time plane are not allowed, this is what the bottom left diagram in Figure 6 shows.

3) Multiple dynamic objects: This is the most interesting case. The special attention arises due to the fact that objects *pass each other*, hence blocking certain lane segments. To illustrate this, we look at Figure 1 and imagine that the white vehicle wants to overtake the blue one. Up to now, the blue vehicle is only a moving obstacle, and we already presented an approach to deal with this. The difference is the green vehicle which also acts as a dynamic obstacle, and in combination with the blue car we find the road to be

blocked at the time and place they meet each other. Similar to the case with the obstacle crossing the lane, we have the possibility to pass the blue one *before* the blockade starts, or wait until the cars passed each other.

The maneuver variants in this case are determined by the sequence the other cars are passed by the ego vehicle as well as the side on which the vehicles are passed. For a moment let us enumerate the maneuver variants in Figure 1, ignoring the side the ego vehicle wants to pass them. The white car can either

- wait for the green car to pass, then overtake the blue and red one (we will refer to this as (●,●,●), (*green*, *blue*, *red*), according to the colored dots in the figure),
- overtake blue, wait for green to pass and then proceed, possibly overtaking red, (●,●,●), or
- overtake blue and red before green approaches, (●,●,●).

Please note that the first maneuver variant covers the scenario to simply stay behind red as well.

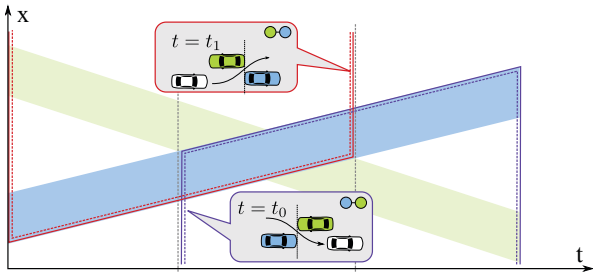
Actually, all possible permutations of the three colored dots are candidates for maneuver variants, but some of them can be discarded in advance. The blue car for example is known to always stay behind the red one, so (●,●,●) is rendered impossible.

In addition to the variants stated above, also the decision to pass left or right needs to be made. The set $\{◐,◑\} \times \{◐,◑\} \times \{◐,◑\}$ as well as each permutation of the elements will enumerate all maneuver variants. Without proper heuristics the number of maneuver variants will explode very quickly, in the case considered here we have 2^3 left/right combinations, with three permutation each, hence 24 trajectories to plan. A simple heuristic is to pass the vehicles on the side with the bigger gap (the blue one on the left side, denoted by ◑, and so on), and use the second maneuver variant which we refer to as (◑,◐,◑). The constraints to maintain the correct left/right decisions will be {◑,◐,◑}.

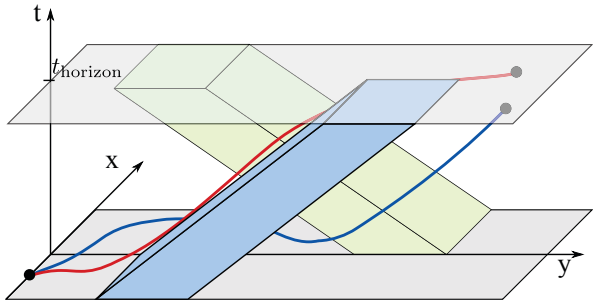
For every subsequent pair of passed vehicles in a maneuver variant, there exists a constraint which comes with this pair. Our notation for such a constraint (yet to be derived) will be ●-● for the constraint which arises with the decision to *first* pass blue, *then* green. To maintain the desired passing order, we derive the constraints {●-●,●-●}, hence our maneuver variant is described by the constraints {●-●,●-●,◑,◐,◑}.

We will use Figure 7a to explain where the constraints arise. In the center, we show the $x \times t$ plane, and as an example, only the green and blue vehicles are considered. The green vehicle is oncoming, therefore the green bar has a negative slope. The blue car is moving in the same direction as the ego vehicle, hence the positive slope. This configuration leaves two maneuver variants, (●,●) and (●,●), and hence the constraints ●-● and ●-●.

The constraint ●-● leads the ego vehicle to first pass the blue vehicle, then the green one. This means that the ego vehicle has to be *in front* of the blue car after $t = t_0$. Otherwise, the maneuver variant will not longer be possible. The obvious reason is that, if green and blue meet with white behind blue, there is no other alternative for the white car to first passing green, and then blue, which is obviously



(a) Temporal constraints from passing order decisions. At $t = t_0$, the ego vehicle has to have finished overtaking blue. Otherwise it is not possible to maintain the desired order (●,●). So the constraint ●-● restricts all samples after t_0 from the area outlined in purple. Similar considerations hold for ●-● .



(b) Two trajectories in the 3-dimensional configuration space. Both trajectories end in the plane at t_{horizon}.

Fig. 7: Constraints for two vehicles.

contrary to the desired order. Compare for the grey speech bubbles: the top one shows the earliest possible time for the white to overtake blue while still maintaining (●,●), the bottom one shows the latest possible time to maintain (●,●).

In a figurative sense, in the $x \times t$ plane, these constraints *punch out* a certain area.

B. Enforcing Maneuver Variants with Constraints

So far, we identified the need for two types of *constraints*. The ◑ type for *geometric decisions* of left/right maneuver variants, and the ○-● type *temporal decisions* to maintain a desired order.

As mentioned earlier, the ○-● type restricts samples to certain intervals in the $x \times t$ domain. Merging two or more of these contraints is rather trivial, see Figure 8. Basically, the constraints restrict intervals either with one of the endpoints being infinite or zero. Assuming only motion forward along the x axis, you can easily verify that every trajectory drawn in the yellow area crosses the colored bars in the desired order.

If an interval with an infinite upper endpoint intersects with an interval with a zero lower endpoint, the maneuver variant becomes infeasible. In the figure, this would mean that for example at t_i the green bar vanishes.

For the ◑ type, depending on the x coordinate, restrictions are set on the y coordinate. This is illustrated in Figure 9, which depicts the positions of two vehicles at a certain

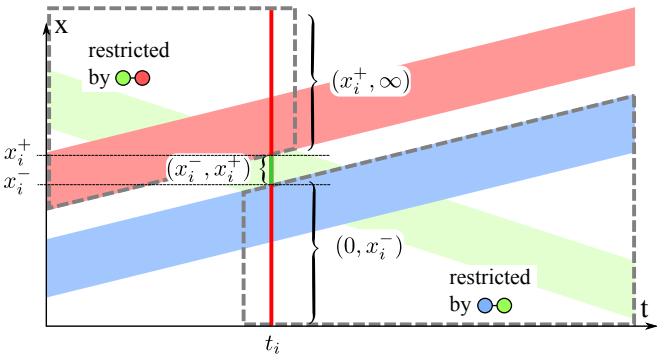


Fig. 8: For the desired order (●, ●, ●) the constraints {●●, ●●} have to be combined, effectively restricting the areas outlined in grey. The sample for t_i is in the x domain constrained to the interval (x_i^-, x_i^+).

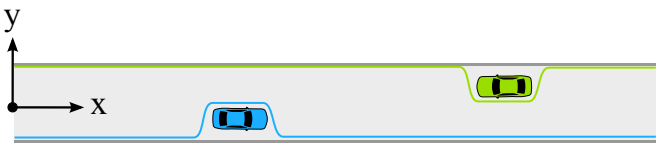


Fig. 9: To stay out of obstacles and to maintain the correct left/right decision, the y coordinate is restricted depending on the x coordinate for a given sample. The plot shows the positions of the vehicles for this given sample, as well as the resulting bounds.

time. To maintain the correct geometric decision and stay collision free, the planned trajectory must be above the blue line and below the green line. As motivated in [5], it is essential to have smooth, continuously differentiable functions, therefore we approximate the remaining space with sigmoidal functions. We use

$$\begin{aligned} y(x) &= a + b(\mathrm{sig}(s\,(x - x_0)) - \\ &\quad \mathrm{sig}(s\,(x - x_1))), \text{where} \\ \mathrm{sig}(x) &= \frac{1}{1 + \exp(-x)} \end{aligned}$$

and a, b, s are parameters to determine offset, height of the convexity as well as the sharpness. x_0 and x_1 define the location of the convexity. If a ◑ type constraint is completely dominated by a ●● one, it can be safely discarded. It means that, a vehicle is outside the feasible x range for a certain time (and therefore sample).

To summarize our findings, we recapitulate the contraints we pose on our trajectory planning problem:

For each sample, we have

- a feasible x range, determined by all ●● type constraints (most solvers will refer to this as *box constraint*), and
- for each vehicle a nonlinear inequality constraint to keep us above or below a sigmoid which determines the occupied space by a vehicle at the corresponding time.

It is worth to mention that the method proposed so far only relies on perfect knowledge, and not on specific shapes of the spatio temporal obstacles. The assumption of constant velocities is only considered to keep the figures and calculations simple. However, we rely on the fact that other cars pass each oher only one time at maximum, this assumption ignores alternately overtaking maneuvers and back-overtaking of other cars for example.

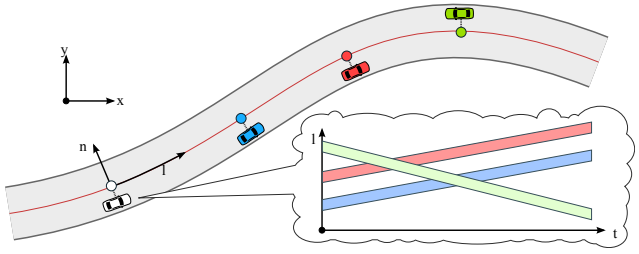


Fig. 10: The coordinate system used for combinatorial analysis. The main direction of motion is along the l coordinate, lateral deviations are expressed with the n coordinate. In the bottom right part the scene abstraction in the $t \times l$ domain is shown from the perspective of the white vehicle. The red line shows the center line of the lane, and the dots represent the mapping of the individual cars onto the centerline.

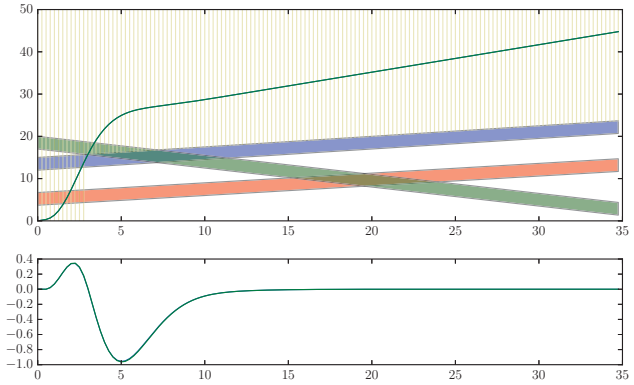
C. Extensions for Curved Roads

Until now, for sense of simplicity, we introduced our formulations for a straight road and defined the longitudinal direction with x. For a curved road, trajectories still need to be optimized in Cartesian coordinates, while the combinatorial planning scheme relies on a coordinate system bound to the lane. A transformation between both worlds is required. In [19] we introduced the *lanelet* concept to fuse topological and geometrical map representations required for high level planning. In order to deal with roads of any arbitrary shape, we proposed a method to derive twice continuous distance metrics and exploited it at a coordinate system built by polygonal line strips. Such a line strip, which we call *reference line*, is depicted in Figure 10. Now, using this reference line as a backbone, we can define the longitudinal direction of any road with l. Through the utilization of the distance metrics, we can seamlessly transform the coordinates and employ the combinatorial planning scheme for curved roads.

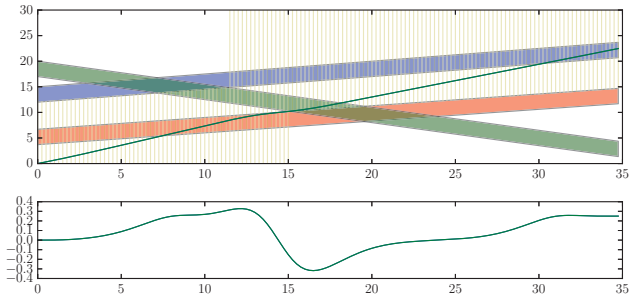
IV. Preliminary Experiments

To demonstrate the effect of our proposed planning scheme, we planned trajectories for the scenario depicted in Figure 1 which was the motivating example for this paper. We discretized time in 0.25 s steps, and covered 35 s which leads to 140 samples, of which the first three are kept fixed (since they define initial position, velocity and acceleration).

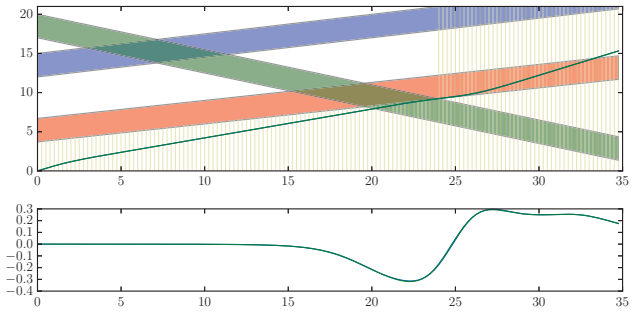
In the first run, we optimized a trajectory with the ●● constraints enabled. Then, we used the result as an initialization for a second run, this time *without* those constraints (only the ◑ type constraints were left). What we expected to see was that the result of the second pass will converge



(a) Maneuver variant (◑, ◑, ◑)



(b) Maneuver variant (◑, ◑, ◑)



(c) Maneuver variant (◑, ◑, ◑)

Fig. 11: Experimental setup with three vehicles (the green one is oncoming).

in the same maneuver variant as the first pass. Ideally, they should be identical, but since the ◑ type constraints map the exact vehicle geometry, while the ○● type constraints only approximate the occupied road segments, they may differ slightly. This experiment clearly indicates the presence of at least three distinct minima, and it depends on the initialization, which of them is found. Neglecting the combinatorial aspect, there is no guarantee that the best solution is found. Aggravating, each of the three solutions is *perfectly plausible* from a driver's point of view.

V. Conclusion

In this work, we introduced a way to systematically enumerate maneuver variants in structured environments. The experiment highlights the hypothesis that each maneuver variant introduces a local optimum, which is hard to find for local, continuous methods, if the initialization is done outside the catchment area of this optimum. However, complete trajectory planning for each variant quickly becomes infeasible due to the exponentially growing number of variants. Therefore, we strongly indicate that a reduced planning problem could serve as a heuristic to select a maneuver variant and provide an initialization for the original problem, so that it is guaranteed to converge against the desired local optimum.

Variant	(◑, ◑, ◑)	(◑, ◑, ◑)	(◑, ◑, ◑)
Solver iterations	133	84	98
Final cost	53444	215.7	521.4
(Time [ms])	(126)	(74)	(109)

TABLE I: Optimizer results for 137 samples (274 parameters).

Acknowledgement

The authors thank Dr. Gabi Breuel and Dr. Thao Dang of Daimler AG for the fruitful collaboration and the support for this work.

References

[1] M. Rufli and R. Siegwart, "On the design of deformable input-/state-lattice graphs," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 3071–3077.

[2] M. Werling, S. Kammel, J. Ziegler, and L. Gröll, "Optimal trajectories for time-critical street scenarios using discretized terminal manifolds," *The International Journal of Robotics Research*, 2011.

[3] Y. Kuwata, S. Karaman, J. Teo, E. Frazzoli, J. P. How, and G. Fiore, "Real-time motion planning with applications to autonomous urban driving," *Control Systems Technology, IEEE Transactions on*, vol. 17, no. 5, pp. 1105–1118, 2009.

[4] K. Chu, M. Lee, and M. Sunwoo, "Local path planning for off-road autonomous driving with avoidance of static obstacles," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 13, no. 4, pp. 1599–1616, 2012.

[5] J. Ziegler, P. Bender, T. Dang, and C. Stiller, "Trajectory planning for Bertha — a local, continuous method," in *Intelligent Vehicles Symposium (IV)*. IEEE, 2014, pp. 450–457.

[6] S. Karaman and E. Frazzoli, "Optimal kinodynamic motion planning using incremental sampling-based methods," in *Decision and Control (CDC), 2010 49th IEEE Conference on*. IEEE, 2010, pp. 7681–7687.

[7] K. Kant and S. W. Zucker, "Toward efficient trajectory planning: The path-velocity decomposition," *The International Journal of Robotics Research*, vol. 5, no. 3, pp. 72–89, 1986.

[8] T. Fraichard and C. Laugier, "Path-velocity decomposition revisited and applied to dynamic trajectory planning," in *Robotics and Automation (ICRA)*. IEEE, 1993, pp. 40–45.

[9] J. Johnson and K. Hauser, "Optimal acceleration-bounded trajectory planning in dynamic environments along a specified path," in *Robotics and Automation (ICRA)*. IEEE, 2012, pp. 2035–2041.

[10] ——, "Optimal longitudinal control planning with moving obstacles," in *Intelligent Vehicles Symposium (IV)*. IEEE, 2013, pp. 605–611.

[11] T. Shamir, "How should an autonomous vehicle overtake a slower moving vehicle: Design and analysis of an optimal trajectory," *Automatic Control, IEEE Transactions on*, vol. 49, no. 4, pp. 607–610, 2004.

[12] J. E. Naranjo, C. Gonzalez, R. Garcia, and T. de Pedro, "Lane-change fuzzy control in autonomous vehicles for the overtaking maneuver," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 9, no. 3, pp. 438–450, 2008.

[13] J. Ziegler and C. Stiller, "Spatiotemporal state lattices for fast trajectory planning in dynamic on-road driving scenarios," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*. IEEE, 2009, pp. 1879–1884.
[14] Ö. Ş. Taş, "Integrating combinatorial reasoning and continuous methods for optimal motion planning of autonomous vehicles," Master's thesis, Karlsruhe Institute of Technology, Germany, September 2014.
[15] R. Kohlhaas, T. Bittner, T. Schamm, and J. M. Zollner, "Semantic state space for high-level maneuver planning in structured traffic scenes," in *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on*. IEEE, 2014, pp. 1060–1065.
[16] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.
[17] W. Malik, S. Rathinam, S. Darbha, and D. Jeffcoat, "Combinatorial motion planning of multiple vehicle systems," in *Decision and Control, 2006 45th IEEE Conference on*. IEEE, 2006, pp. 5299–5304.
[18] "Encyclopaedia of mathematics," (2015, Jan). [Online]. Available: http://www.encyclopediaofmath.org/index.php/Homotopy
[19] P. Bender, J. Ziegler, and C. Stiller, "Lanelets: Efficient map representation for autonomous driving," in *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*. IEEE, 2014, pp. 420–425.
[20] S. Agarwal, K. Mierle, and Others, "Ceres solver," http://ceres-solver.org.

APPENDIX
NOTES ON THE CONTINUOUS PLANNER

We will briefly outline the continuous planner. The equations are mostly taken from [5], where we also provide an in-depth discussion on the subject.

A dynamically feasbile and comfortable optimal trajectory must minimize the integral

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

where $\mathbf{x}(t)$ defines the position of the vehicle in Cartesian coordinates and the integrand L is given by

$$L = w_{\mathrm{offs}} j_{\mathrm{offs}}^2 + w_{\mathrm{vel}} j_{\mathrm{vel}}^2 + w_{\mathrm{acc}} j_{\mathrm{acc}}^2 + w_{\mathrm{jerk}} j_{\mathrm{jerk}}^2. \tag{2}$$

The individual summands of the integrand penalize lateral offset from the road centerline, deviation from the reference velocity, excessive acceleration, jerk and yaw rate respectively. The influence of the summands can be tuned using the weighting factors $w_{\mathrm{offs}}, w_{\mathrm{vel}}$ etc.

The offset term can be formulated as

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

where d_{left} and d_{right} are the signed distance functions towards the bounds of the driving corridor. The velocity, acceleration and jerk terms can be formulated as

$$j_{\mathrm{vel}}(\mathbf{x}(t)) = w_{\mathrm{vel}} \left| \mathbf{v}_{\mathrm{des}}(\mathbf{x}(t)) - \dot{\mathbf{x}}(t) \right|^2, \tag{3}$$

$$j_{\mathrm{acc}}(\mathbf{x}(t)) = w_{\mathrm{acc}} \left| \ddot{\mathbf{x}}(t) \right|^2, \tag{4}$$

$$j_{\mathrm{jerk}}(\mathbf{x}(t)) = w_{\mathrm{jerk}} \left| \dddot{\mathbf{x}}(t) \right|^2, \tag{5}$$

respectively.

The integral is approximated by a sum, and the derivatives are converted to finite differences:

$$\dot{\mathbf{x}}(t_i) \approx \dot{\mathbf{x}}_{\mathrm{d}} = \frac{\mathbf{x}_{i+1} - \mathbf{x}_i}{\Delta}, \tag{6a}$$

$$\ddot{\mathbf{x}}(t_i) \approx \ddot{\mathbf{x}}_{\mathrm{d}} = \frac{\mathbf{x}_{i-1} - 2\mathbf{x}_i + \mathbf{x}_{i+1}}{\Delta^2}, \tag{6b}$$

$$\dddot{\mathbf{x}}(t_i) \approx \dddot{\mathbf{x}}_{\mathrm{d}} = \frac{-\mathbf{x}_i + 3\mathbf{x}_{i+1} - 3\mathbf{x}_{i+2} + \mathbf{x}_{i+3}}{\Delta^3}. \tag{6c}$$

Δ is the temporal distance between two samples. Since all individual terms contributing to the sum are given as squares, we slightly re-formulate (2):

$$\underset{\mathbf{x}_4,\ldots,\mathbf{x}_N}{\arg\min} \left(\mathbf{W}\mathbf{r}\right)^\top \mathbf{r} \tag{7}$$

where $\mathbf{r}$ is a set of residuals and $\mathbf{W}$ is a diagonal matrix with the corresponding weights. So, what we need to solve is a *non-linear least squares problem*, and Ceres [20] is an efficient solver for this type of problems.

Note that $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ are not free variables, but bound to constant values to initially set position, velocity and acceleration.

For N trajectory sample points, the residuals are given as

$$\begin{aligned} \mathbf{r} = (&r_{\mathrm{offs},1}, \ldots, r_{\mathrm{offs},N}, \\ &r_{\mathrm{vel},1}, \ldots, r_{\mathrm{vel},N-1}, \\ &r_{\mathrm{acc},1}, \ldots, r_{\mathrm{acc},N-2}, \\ &r_{\mathrm{jerk},1}, \ldots, r_{\mathrm{jerk},N-3})^\top \end{aligned} \tag{8}$$

Since Ceres does not allow for constraints (except for bound constraints on the parameters), we approximate them with appropriate cost terms. So for *each* sample and *each* constraint we introduce a residual which is 0 if the constraint is untouched, and smoothly increases, proportional to the constraint violation. Weighted appropriately, Ceres handles this nicely.