Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnline CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 51-20 (2018) 279–286

# Real-Time Autonomous Car Motion Planning using NMPC with Approximated Problem Considering Traffic Environment

**Makoto Obayashi** * **Gaku Takano** *

* *DENSO IT Laboratory, Shibuya-ku, Tokyo, Japan (e-mail: {mohbayashi, gtakano}@d-itlab.co.jp).*

**Abstract:** Several studies have demonstrated that car dynamics computation is essential for autonomous car motion planning. One of the most promising techniques for motion planning with dynamics is model predictive control (MPC). Planned motion computed using MPC consists of solving an optimization problem with constraint equations representing car dynamics and environmental conditions. The disadvantage is that the optimization problem is complex when it is nonlinear. To overcome this, we have developed a highly efficient computing technique for autonomous car motion planning with nonlinear MPC (NMPC). It consists of an approximated problem and a sequential quadratic problem based on an active-set algorithm. We demonstrate the suitability of our approach by using a car dynamics simulator.



*Keywords:* Model Predictive Control, Optimization problems, Autonomous vehicles

## 1. INTRODUCTION

An autonomous car decision making architecture typically consists of several layers and function blocks (Gonzalez et al., (2016)). It enhances the independence of each function by clearly defining the interface of each layer and block so as to enable easy construction of the whole system by efficient division of labor. In such a system, the output of the decision making process become an abstractive behavior command (right/left turn, overtake, change lane), and the trajectory of a car is generated by a trajectory planning function. In this case, the decision making function does not consider the car dynamics. This means the trajectory planning function has to compute a solution that satisfies both the abstractive behavior command and the car dynamics. As a result, an unreasonable trajectory may be generated under certain circumstances. For example, it may cause rapid acceleration, vibratory steering, or near-collisions. To prevent this, it is necessary to smooth the trajectory and to ensure that the smoothing process operates with a sufficient margin in terms of both space and time. In the case of a crowded environment, it may be difficult to compute the appropriate trajectory. It has been pointed out that appropriate motion can be planned by explicitly considering car dynamics for decision making in an environment lacking any spatio-temporal margin (Polack et al., (2017), Yi et al., (2016)). A motion planning technique using model predictive control (MPC) utilizes car dynamics as constraint equations so as to simultaneously enable decision making and trajectory generation with a solution that satisfies the car dynamics constraints. Several papers have already been presented in the research domain of autonomous motion planning using MPC, including our own study on autonomous overtaking motion planning (Obayashi et al., (2016)). In autonomous car motion planning with MPC, MPC can be applied in one of two ways. The first involves accurately tracking to a predetermined trajectory. It is assumed that a trajectory is determined by another method and is outside the scope of this study. The other way involves generating the trajectory on which an autonomous car moves. Examples of proposed nonlinear MPC (NMPC) applications using this approach include automatically generating the motion of collision avoidance (Werling et al., (2012), Ziegler et al., (2014)), finding the optimal path, and generating overtaking motion (Obayashi et al., (2016)). In this approach, car state values that are real numbers need to be determined. This problem is essentially nonlinear in that the search space does not satisfy the unimodality because there are various desired motions. Naturally, since constraint equations also contain nonlinear functions, the optimization process becomes highly complex. One of the biggest challenges here is developing a real-time solver for nonlinear optimization with constraints. Most of the existing research in this area focuses on reducing the computation complexity through simplified dynamics and cost functions. In addition, as an approach to the optimization algorithm, speeding up is achieved by greatly limiting the number of iterations and setting strong assumptions on the existence of the solution. Each method provides a good solution for the intended purpose, but all are inappropriate for our problem on various points. In this study, we developed a motion planning system for autonomous cars that generates appropriate motions and responses to the behavior of other cars and the road structure on the basis of finite term prediction and car dynamics. Various formulations (cost functions, car dynamics equations, inequality constraints) used in this paper are similar to those in our previous work (Obayashi et al., (2016)).

Our contribution in this study is the development of a real-time computation technique for the nonlinear optimization of autonomous car motion planning. Our target of motion planning is appropriate decision making based on a long

2405-8963 © 2018, IFAC (International Federation of Automatic Control) Hosting by Elsevier Ltd. All rights reserved.
Peer review under responsibility of International Federation of Automatic Control.
10.1016/j.ifacol.2018.11.026

prediction horizon. We propose an approximated problem that maintains car dynamics and the traffic environment for a constrained nonlinear optimization to achieve accurate motion planning. The approximated problem can generate the car motion as a coarse solution, i.e., a nearly feasible solution, even in the case of a tight environment, and the nonlinear optimization can then reach fast convergence using the coarse solution from the approximated problem. In our approach, a motion is planned on the basis of the sampling time $\Delta T = 0.5$ and prediction horizon $H_p = 10$ [step] (5 [sec]). We use a feedback controller that is executed using an 8 [msec] cycle to track the planned motion precisely.

In section 2 of this paper, we discuss related work. In section 3, we provide the details of our problem and formulations for car dynamics, cost functions, and inequality constraints. We present the design of the proposed approximated problem for nonlinear optimization in NMPC in section 4, and in section 5, we describe the nonlinear optimization process. In section 6, we present the numerical experiments we performed, and the results are discussed in section 7. We conclude in section 8 with a brief summary.

## 2. RELATED WORK

There have been several studies on motion planning techniques using NMPC, but they tend to be limited to discussions on problem formulations while neglecting any mention of a suitable nonlinear optimization technique. Many techniques are in place for constrained nonlinear optimization. One important consideration in nonlinear optimization is real-time execution. To this end, we try to model the ego car and the other car state and environmental structure as one continuous system so as to generate accurate control signals. We have also constructed the cost functions to be continuous functions. We therefore adopt a continuous optimization technique based on Newton's method that iteratively solves a quadratic problem until it reaches convergence. In this case, the number of iterations varies greatly depending on the situation, which makes stable calculation and real-time execution difficult.

Ohtsuka (2004) proposed C/GMRES to reduce the required time for computing constrained nonlinear optimization in NMPC by assuming that the solution at time $t$ is located near the solution at time $t-1$. This method deforms the original problem into a system of linear equations, thus, avoiding iterative computation. Other works have shown that this approach is highly effective for some control problems. One of the most popular nonlinear optimization solvers for NMPC is the ACADO Toolkit (Diehl et al., (2005)). Similar to C/GMRES, it adopts the hypothesis to the location of the solution. On the basis of this hypothesis, the algorithm of ACADO can reduce the number of iterations in its nonlinear optimization solver.

Unfortunately, the above hypotheses cannot be introduced to our problem. Our aim is to develop autonomous car long-term motion planning consisting of control signals based on car dynamics, but substantial observation changes generate a large difference in motion (overtaking, collision avoidance, etc.) at time $t$ from the previous one at time $t-1$. As a result, the solution for the optimization is likely to be quite different at every time. The computational techniques utilized by previous approaches (ACADO Toolkit, C/GMRES) are thus inadequate for our aims. We have therefore developed a technique that can decrease the computational complexity of MPC.

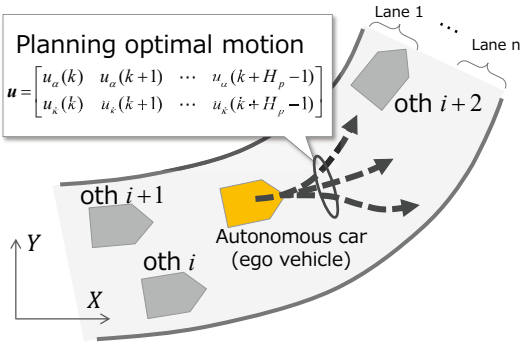


Fig. 1. Motion planning environment.

## 3. THE PROBLEM OF MOTION PLANNING USING NMPC

The schematic environment of the motion planning we are tackling is shown in Fig. 1. The road has an arbitrary curvature and features several lanes. There is no height change of the road. Other cars are also moving at different velocities in the environment. We assume that the ego has the exact environment information (including road structure) and can observe the state of the ego and other cars. The system we developed enables the ego to plan its motion such that it maintains its designated cruising velocity. The planned motion consists of real numbers representing continuous car motion that form a predictive horizon length time series of acceleration and curvature changing rate. Modeling the problem in continuous space requires a method for determining car motion that is suitable for real car dynamics. Our goal is to achieve a smoother solution than graph search-based approaches. The proposed motion planning system is shown in Fig. 2. We introduce the car dynamics, cost functions, and inequality constraints below.

### 3.1 Vehicle Dynamics

The vehicle dynamics in this study are shown in Fig. 3. The model is constructed on the basis of a bicycle model and control input signals $\boldsymbol{u}$ consist of acceleration $u_\alpha$ [m/s$^2$] and curvature change rate $u_\kappa$ [m$^{-1}$s$^{-1}$]. The state vector of the ego at $\boldsymbol{x}(k)$ is constructed as

$$\boldsymbol{x}(k) = [X(k), Y(k), \theta(k), V(k), \kappa(k)]^\top \quad (1)$$

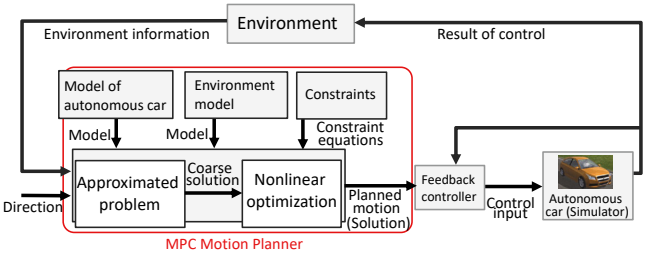


Fig. 2. Proposed architecture of Predictive Motion Planning using approximated problem.

where $X(k)$ [m], $Y(k)$ [m] means the position in the Cartesian coordinate and $\theta(k)$ [rad], $V(k)$ [m/s], and $\kappa(k)$ [$m^{-1}$] means yaw angle, velocity, and curvature, respectively. This is similar to our previous work (Obayashi et al., (2016)) but differs in that we adopt curvature $\kappa$ and curvature changing rate $u_{\dot{\kappa}}$ instead of the front wheel steering angle $\delta$ [rad] and steering changing rate $u_\delta$ [rad/s]. By using curvature and curvature changing rate, we are able to reduce nonlinear terms from the vehicle dynamics equations. We assume that equation $\kappa(k) = V(k) \cdot \tan\delta(k)/\text{wheelbase}$ holds between $\kappa$ and $\delta$. On the basis of the above, we define the discrete time car dynamics equations in accordance with each car state variable. Note that $\Delta T$ is the execution cycle of MPC.

$$\begin{aligned} X(k+1) &= X(k) + V(k)\cos\theta(k)\Delta T + \frac{1}{2}u_\alpha \cos\theta(k)\Delta T^2 \\ &- \frac{1}{2}V(k)^2\kappa(k)\sin\theta(k)\Delta T^2 - \frac{1}{2}u_\alpha(k)V(k)\kappa(k)\sin\theta(k)\Delta T^3 \\ &- \frac{1}{6}u_\kappa(k)V(k)^2\sin\theta(k)\Delta T^3 - \frac{1}{6}V(k)^3\kappa(k)^2\cos\theta(k)\Delta T^3 \end{aligned} \tag{2}$$

$$\begin{aligned} Y(k+1) &= Y(k) + V(k)\sin(k)\Delta T + \frac{1}{2}u_\alpha(k)\sin\theta(k)\Delta T^2 \\ &+ \frac{1}{2}V(k)^2\kappa(k)\cos\theta(k)\Delta T^2 + \frac{1}{2}u_\alpha(k)V(k)\kappa(k)\cos\Delta T^3 \\ &+ \frac{1}{6}u_\kappa(k)V(k)^2\cos(k)\Delta T^3 - \frac{1}{6}V(k)^3\kappa(k)^2\sin\theta(k)\Delta T^3 \end{aligned} \tag{3}$$

$$\begin{aligned} \theta(k+1) &= \theta(k) + V(k)\kappa(k)\Delta T + \frac{1}{2}u_{\dot{\kappa}}(k)V(k)\Delta T^2 \\ &+ \frac{1}{2}u_\alpha(k)\kappa(k)\Delta T^2 \end{aligned} \tag{4}$$

$$V(k+1) = V(k) + u_\alpha(k)\Delta T \tag{5}$$

$$\kappa(k+1) = \kappa(k) + u_{\dot{\kappa}}(k)\Delta T \tag{6}$$

We use the road coordinate (Fig. 4) as well as the Cartesian coordinate in our formulation of situations and assume that the ego has the road structure information including road base points ($P_i$ in Fig. 4). The ego computes the longitudinal position $S$ and the lateral position $L$ from the line segment composed of adjacent point $P_i$.

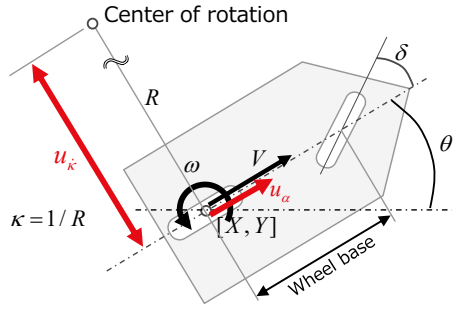


Fig. 3. Vehicle dynamics model based on bicycle model.

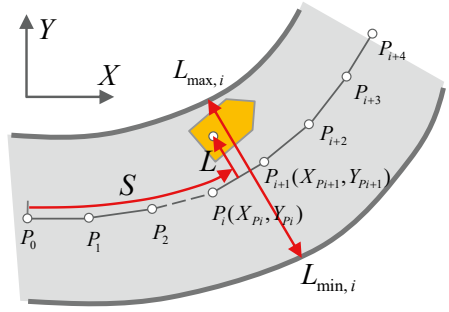


Fig. 4. Road coordinate axis.

### 3.2 Cost Functions

The cost function represents desirable vehicle state and motion. It is generally formulated as

$$\mathcal{J} = \phi(\boldsymbol{x}(H_p)) + \sum_{k=1}^{H_p} \mathcal{L}_*(\boldsymbol{x}(k), \boldsymbol{u}(k), k) \tag{7}$$

$\phi$ is the cost at the termination time of prediction length, and it is not used in this study. $\mathcal{L}_*(\boldsymbol{x}(k), \boldsymbol{u}(k), k)$ consists of the sum of weighted functions, as described below.

In general, if a road has several lanes, it is desirable for a car to drive in the middle of a lane. Moreover, a car should not keep moving on the lane boundaries for very long. We therefore adopt a function that encourages the ego to keep moving in the middle of each lane, as

$$\mathcal{L}_{\text{lat}}(k) = \prod_{i=1}^{N_{\text{lane}}} C_g \left(1 - \exp\left(-\frac{L(k) - \varepsilon_i}{l_g^2}\right)\right) \tag{8}$$

where $C_g$ and $l_g$ are the constant values that determine the shape of the function. We set these values as $C_g = 1.0$ and $l_g = 0.5$. $\varepsilon_i$ means the lateral position of the middle of lane $i$.

The cost of distance between cars $\mathcal{L}_{\text{dist}}$ is formulated using the time to collision (TTC) and time headway (THW) criteria (Kawabe et al., (2004)).

$$\mathcal{L}_{\text{dist}}(k) = \sum_{i=1}^{N_{\text{oth}}} \zeta\left(L_i(k) - L(k)\right) \cdot \left(w_{\text{ttc}} \cdot \mathcal{L}_{\text{ttc},i}(k) + w_{\text{thw}} \cdot \mathcal{L}_{\text{thw},i}(k)\right) \tag{9}$$

In the above equation, $N_{\text{oth}}$ means the number of other cars that need to be considered. The definitions of $\mathcal{L}_{\text{ttc}}$ and $\mathcal{L}_{\text{thw}}$ are

$$\mathcal{L}_{\text{ttc}}(k) = \frac{1}{2}\frac{V_{\text{f}}(k) - V_\ell(k)}{S_\ell(k) - S_{\text{f}}(k)} \tag{10}$$

$$\mathcal{L}_{\text{thw}}(k) = \frac{1}{2}\frac{V_{\text{f}}(k)}{S_\ell(k) - S_{\text{f}}(k)} \tag{11}$$

In the above two equations, $S_\ell$ and $V_\ell$ refer to the leading car's longitudinal position and velocity, respectively, and $S_{\text{f}}$ and $V_{\text{f}}$ refer to the following car's longitudinal position and velocity, respectively. The positional relationship, leading or following, is determined from the longitudinal relationship between the ego and an other car in the road coordinate. If there is relative lateral distance between the ego and the other car, the distance cost should be reduced. In the case of the other car moving on a different lane from the ego, the distance cost becomes nearly zero and these cars can pass by each other. We utilize a sigmoid function as a means to achieve varying cost based on relative lateral distance, as

$$\zeta(L_{\text{rel}}) = \frac{1}{1 + \exp\left(-s_a(L_{\text{rel}} + \frac{W_{\text{ego}}}{2})\right)} \cdot \frac{1}{1 + \exp\left(-s_a(-L_{\text{rel}} + \frac{W_{\text{ego}}}{2})\right)} \tag{12}$$

where $L_{\text{rel}}$ means the relative lateral distance between the ego and another car, $L_{\text{rel}} = L_i(k) - L(k)$, $(i = 1, \ldots, N_{\text{oth}})$, $W_{\text{ego}}$ means the width of the ego, and $s_a$ is the constant value for adjusting the shape of the function. We set these variables as $W_{\text{ego}} = 1.7$ [m] and $s_a = 8$, respectively.

The cost of cruising velocity is represented as

$$\mathcal{L}_{\text{vel}}(k) = \frac{1}{2}\left(V(k) - V_{\text{ref}}\right)^2 \tag{13}$$

We also introduce the cost of curvature and yaw angle to the cost functions. These are used for encouraging smooth driving along the road.

$$\mathcal{L}_{\rm cur}(k) = \frac{1}{2}\left(\kappa(k) - \kappa_{\rm ref}\right)^2 \tag{14}$$

$$\mathcal{L}_{\rm yaw}(k) = \frac{1}{2}\left(\theta(k) - \theta_{\rm ref}\right)^2 \tag{15}$$

The ideal curvature $\kappa_{\rm ref}$ and ideal yaw angle $\theta_{\rm ref}$ are derived from the road structure information that the ego has. We adopt the following equations to prevent rapid acceleration and curvature changing.

$$\mathcal{L}_{\rm acc}(k) = \frac{1}{2}u_\alpha(k)^2 \tag{16}$$

$$\mathcal{L}_{\dot\kappa}(k) = \frac{1}{2}u_{\dot\kappa}(k)^2 \tag{17}$$

### *3.3 Inequality Constraints and Environment*

We formulate inequality and use it as an inequality constraint for the nonlinear optimization to prevent catastrophic situations such as collision and overstepping the road boundaries. To enforce these collision constraints, we approximate the shape of the ego by using several circles having a wider diameter than the vehicle, similar to a related work (Ziegler et al., (2014)). The collision constraints are formulated as the L2 norm between the center of circles that belong to different cars in the Cartesian coordinate. In this study, the sampling time is relatively long ($\Delta T = 0.5$ [sec]), so we construct collision constraints at shorter time intervals than $\Delta T$. We introduce positive integer $T_p$ for time interpolation, and then our collision avoidance constraint checks the collision at $\Delta\tau = \Delta T/T_p$ time intervals. We adopt $T_p = 5$. Based on above, the collision avoidance constraint is as follows:

$$\|\boldsymbol{p}_{{\rm oth}i}(k) - \boldsymbol{p}_{{\rm ego}j}(k)\|_2^2 \geq M^2, \quad (i \in N_{\rm oth},\ j \in N_{\rm ego}) \tag{18}$$

where $\boldsymbol{p}_{\rm oth}$, $\boldsymbol{p}_{\rm ego}$ means the coordinate of the center of a circle. In our study, the center of a circle is located at the center of the rear and front axles. The variable $M$ means the sum of radiuses of each circle.

We use the lateral position in the road coordinate $L$ for formulating inequality constraints for road boundaries. The variable $L(k)$ is obtained from the road base line segment $aX + bY + c = 0$ that is located nearest to the ego positions $X(k)$ and $Y(k)$. As shown in Fig. 4, the road boundary constraint is designed to guarantee that the ego lateral position stays within road boundaries $L_{\rm max}$ and $L_{\rm min}$.

$$L_{min} \leq \frac{aX(k)^2 + bY(k)^2 + c}{\sqrt{a^2 + b^2}} \leq L_{max} \tag{19}$$

To restrain extremely rapid motion, control signals (acceleration $u_\alpha$ and curvature change rate $u_{\dot\kappa}$) are constrained by maximum and minimum values, as follows:

$$u_{\alpha,\rm min} \leq u_\alpha(k) \leq u_{\alpha,\rm max} \tag{20}$$

$$u_{\dot\kappa,\rm min} \leq u_{\dot\kappa}(k) \leq u_{\dot\kappa,\rm max} \tag{21}$$

The number of inequality constraint equations becomes over 1200 in the case of $\Delta T = 0.5$ [sec], $H_p = 10$ [step] (5 [sec]), and the number of other cars $N_{\rm oth} = 6$.

## 4. APPROXIMATED PROBLEM OF NONLINEAR OPTIMIZATION

In an optimization problem, optimization variables are control inputs and a vector composed of state variables of the ego car over a prediction horizon.

$$\boldsymbol{z} = [\boldsymbol{u}(0), \boldsymbol{x}(1), \ldots, \boldsymbol{u}(H_p - 1), \boldsymbol{x}(H_p)]^\top \tag{22}$$

Therefore, considering the cost function, car dynamics, and constraints mentioned in the previous section, our problem in this research becomes nonlinear optimization with nonlinear constraints. We adopt an optimization technique based on sequential quadratic programming (SQP). Therefore, the quality of the solution and the required time to reach the local optimal solution largely depend on the location of the initial point. To solve this problem, we insert an approximated problem using the problem structure of motion planning before SQP. The flow of the entire process is shown in Algorithm 1.

**Algorithm 1** Motion Planning Optimization

**Input :** $\boldsymbol{x}(0)$, $\boldsymbol{x}_{{\rm oth}i}(0)$ $i \in N_{\rm oth}$
**Output :** $\boldsymbol{z}^*$

1: $\boldsymbol{z}_{\rm app} \Leftarrow$ ApproximatedProblem($\boldsymbol{x}_{\rm ego}(0)$, $\boldsymbol{x}_{\rm oth}(0)$)
2: $\boldsymbol{z}^* \Leftarrow$ Optimization($\boldsymbol{z}_{\rm app}$, $\boldsymbol{x}_{\rm oth}(0)$)

We assume that the time when the motion planning is launched is zero ($k = 0$). We input the current state of the ego $\boldsymbol{x}(0)$ and the state of another vehicle $\boldsymbol{x}_{{\rm oth}i}(0), (i \in N_{\rm oth})$ to the motion planning process. We also assume that the ego has the road structure data. Finally, the planned motion of the ego $\boldsymbol{z}^*$ is calculated by nonlinear optimization using the coarse solution $\boldsymbol{z}_{\rm app}$ generated by the approximated problem.

### *4.1 Approximated Problem of Motion Planning*

In the approximated problem, the objective is to quickly obtain a coarse solution that is close to a feasible point of the original nonlinear optimization problem. Its main approaches are as follows.

- The ego acceleration is assumed to be constant in the approximated problem and we fix the acceleration $u_\alpha$ over all horizons to a fixed value. As a result, the time series of the control input values to be obtained is only the curvature change rate $u_{\dot\kappa}(k)$, $k = 0 \ldots H_p$. The other state variables of the ego are automatically determined from the current value $\boldsymbol{x}(0)$ and the control input based on the vehicle motion model of the equations (2)-(6).
- Control inputs that can shift the two states of the yaw and the curvature to arbitrary values are available. To travel smoothly on the road, it is desirable that the ego's yaw angle $\theta$ and curvature $\kappa$ be nearly equal to the road direction and curvature, respectively. The target yaw angle and curvature can be determined on the basis of the road structure data the ego has.
- We decompose the approximated problem into several subproblems. Three-step control inputs $u_{\dot\kappa}(k)$, $u_{\dot\kappa}(k+1)$, and $u_{\dot\kappa}(k+2)$ are sufficient for shifting the arbitrary ego state yaw and curvature to the arbitrary target states $\theta_{\rm ref}$ and $\kappa_{\rm ref}$. Therefore, the

minimum configuration of a subproblem is a 3-step ($3\Delta T$) control input determination problem.

The flow of the approximated problem is shown in Algorithm 2. The original problem is decomposed into several subproblems (line 1).

**Algorithm 2** Approximated Problem Algorithm

**Input:** $\boldsymbol{x}_{\mathrm{ego}}(0)$, $\boldsymbol{x}_{\mathrm{oth}}(0)$
**Output :** $\boldsymbol{z}_{\mathrm{app}}$

1: subProblems = divideOriginalProblem($H_p$)
2: $\mathcal{Z}_0 \leftarrow \boldsymbol{x}_{\mathrm{ego}}(0)$
3: $N_p \leftarrow$ length(subProblems) ▷ $N_p$: The number of sub problems
4: **for** $i = 1$ to $N_p$ **do**
5: $N_m \leftarrow \mathrm{length}(\mathcal{Z}_{i-1})$
6: **for** $j = 1$ to $N_m$ **do**
7: $\mathcal{S} \leftarrow$ solveSubProblem(subProblem(i-1), $\mathcal{Z}_{i-1}(j)$, $\boldsymbol{x}_{\mathrm{oth}}$)
8: $\mathcal{Z}_i \leftarrow \mathcal{Z}_i \cup \mathcal{S}$
9: **end for**
10: **end for**
11: $\boldsymbol{z}_{\mathrm{app}} \leftarrow$ selectMotion($\mathcal{Z}_{N_p}$)

Subproblems that are time divided from an original problem are computed in time order (line 4), as shown in. Fig. 5. In the case of $H_p = 10$, the approximated problem was divided into three subproblems. Control input signals and state transition are computed from the subproblems. $\mathcal{Z}_i, (i = 0, \ldots, N_p)$ means the set of solutions in each subproblem $i$. All subproblems adopt a solution generated by a previous time subproblem (line 7) and compute a solution independently. Figure 6 shows the correspondence relationship between control input computation and road structure. In the figure, vehicle state $\boldsymbol{x}(k) = [\ X(k),\ Y(k),\ \theta(k),\ V(k),\ \kappa(k)]^\top$ is already determined. We have to compute control input signals $u_{\dot{\kappa}}(k), u_{\dot{\kappa}}(k+1), u_{\dot{\kappa}}(k+2)$ over $3\Delta T$. The state $\boldsymbol{x}(k+3)$ is the final state in a subproblem. In our approach, we make the state $\boldsymbol{x}(k+3)$ correspond to the road curvature and the road direction. The ego position in $k+3$ can be computed because the acceleration is assumed to be constant in the approximated problem. Since the ego has the road structure information, it can determine the ideal state of yaw angle $\theta(k+3)$ and curvature $\kappa(k+3)$. We assign the set of curvatures $\kappa_i(k+1)$, $i = 1, \ldots, N_\kappa$ to search for the appropriate $\kappa(k+1)$. On the basis of the above, $\kappa_i(k+2)$ that corresponds to $\kappa_i(k+1)$ is uniquely determined by

$$\kappa_i(k+2) = \frac{1}{\Delta T}\left(\frac{1}{V}\theta(k+3) - \frac{1}{2}\kappa(k)\Delta T - \kappa_i(k+1)\Delta T - \frac{1}{2}\kappa(k+3)\Delta T\right) \tag{23}$$

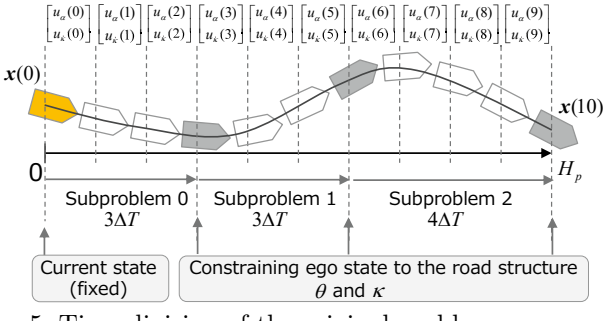


Fig. 5. Time division of the original problem

We assign the number of $N_\kappa$ curvatures to $\kappa_i(k+1)$. These curvatures are determined using the ideal curvature of the road structure. We set $N_\kappa = 21$ difference curvatures every $0.015[\mathrm{m}^{-1}]$ with the ideal curvature as the center. As a result, the number of $N_\kappa$ motions is generated. In a subproblem, motions that satisfy two inequality constraints become solutions. These inequality constraints are collision constraints (18) and road boundary constraints (19). Solutions that are generated from a subproblem becomes the initial values for other subproblems the next time. Since the solutions generated from the approximated problem satisfy vehicle dynamics, it is consistent with the equality constraint equations in the nonlinear optimization process after the approximated problem. Figure 6 shows the case of $3\Delta T$ ($\Delta T$ – $\Delta T$ – $\Delta T$), but it can easily be expanded to the case $4\Delta T$ ($\Delta T$ – $2\Delta T$ – $\Delta T$). The approximated problem has no iteration based on convergence, unlike Newton-based optimization, so the required time can easily be estimated and has less fluctuation.

The approximated problem generates several solutions. In our case, up to $N_\kappa^3$ coarse solutions are potentially generated. However, these can usually be kept to a certain amount because inequality constraints substantially reduce them. We select the one that seems most appropriate for initial values $z_{\mathrm{ini}}$. In our approach, we adopt the one that has the smallest maximum curvature change rate. In most cases, a single fixed acceleration vector $u_\alpha(k) = 0$, $(k = 0, \ldots, H_p - 1)$ is enough to find feasible solutions in the approximated problem. In addition, calculating the approximated problem with the setting of multiple fixed acceleration vectors will make it easier to obtain a feasible solution in many use cases.

## 5. NONLINEAR OPTIMIZATION FOR MOTION PLANNING

In this section, we describe the nonlinear optimization in detail. We use variables generated by the approximated problem as the initial point for the nonlinear optimization process. The solution is obtained by using a sequential quadratic problem (SQP). The nonlinear function should be approximated to be a quadratic problem (QP) with constraints. We adopt an active-set algorithm (Goldfarb et al., (1983)) to solve the QP in every SQP iteration. With the active-set algorithm, we only need to consider active ones for all constraints. If the variables $\boldsymbol{z}$ obtained from the approximated problem are nearly feasible, most of the inequality constraints are discarded from consideration.

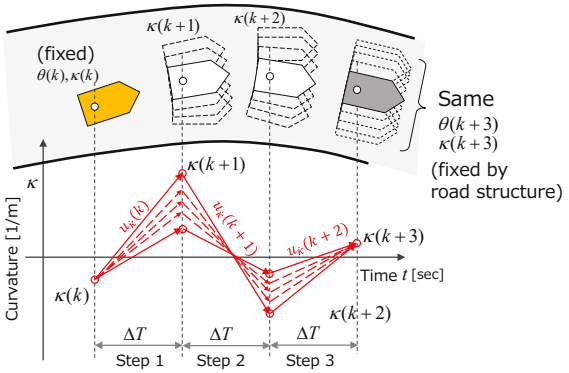


Fig. 6. Relation between curvature change rate and road structure in the approximated problem

This should help to decrease the computational complexity of the optimization process.

### 5.1 Linearization of Constraint Equations

In quadratic programming, all constraint equations should be linear. We used the following approach to linearize the vehicle dynamics around point $\boldsymbol{z}_i$. Note that $i$ means the number of iterations in the SQP process. The variable $\bar{\boldsymbol{z}}_{i+1}(k)$ at a certain time $k$ is expressed as $\bar{\boldsymbol{z}}_{i+1}(k) = \bar{\boldsymbol{z}}_i(k) + \Delta\boldsymbol{z}_i(k)$. In this equation, $\bar{\boldsymbol{z}}_i(k)$ means the coarse solution of the $i$th SQP iteration, and $\Delta\boldsymbol{z}_i(k)$ means the change amount that is obtained from the $i$th SQP computation. From these operations, linearized vehicle dynamics according to $X$ in the Cartesian axis are derived as follows.

$$\begin{aligned}
&\bar{X}_i(k+1) + \Delta X_i(k+1) = \bar{X}_i(k) + \Delta X_i(k) \\
&\quad + \left(\bar{V}_i(k) + \Delta V_i(k)\right)\cos\left(\bar{\theta}_i(k) + \Delta\theta_i(k)\right)\Delta T \\
&\quad + \frac{1}{2}\left(\bar{u}_{\alpha_i}(k) + \Delta u_{\alpha i}(k)\right)\cos\left(\bar{\theta}_i(k) + \Delta\theta_i(k)\right)\Delta T^2 \\
&\quad - \frac{1}{2}\left(\bar{V}_i(k) + \Delta V_i(k)\right)^2\left(\bar{\kappa}_i(k) + \Delta\kappa_i(k)\right)\sin\left(\bar{\theta}_i(k) + \Delta\theta_i(k)\right)\Delta T^2 \\
&\quad - \frac{1}{2}\left(\bar{u}_{\alpha i}(k) + \Delta u_{\alpha i}(k)\right)\left(\bar{V}_i(k) + \Delta V_i(k)\right)\left(\bar{\kappa}_i(k) + \Delta\kappa_i(k)\right) \\
&\qquad\qquad \sin\left(\bar{\theta}_i(k) + \Delta\theta_i(k)\right)\Delta T^3 \\
&\quad - \frac{1}{6}\left(\bar{u}_{\dot{\kappa}i}(k) + \Delta u_{\dot{\kappa}i}\right)\left(\bar{V}_i(k) + \Delta V_i(k)\right)^2\sin\left(\bar{\theta}_i(k) + \Delta\theta_i(k)\right)\Delta T^3 \\
&\quad - \frac{1}{6}\left(\bar{V}_i(k) + \Delta V_i(k)\right)^3\left(\bar{\kappa}_i(k) + \Delta\kappa_i(k)\right)^2\cos\left(\bar{\theta}_i(k) + \Delta\theta_i(k)\right)\Delta T^3
\end{aligned} \tag{24}$$

In equation (24), optimization variables are $\Delta X_i(k+1)$, $\Delta X_i(k)$, $\Delta\theta_i(k)$, $\Delta V_i(k)$, $\Delta\kappa_i(k)$, $\Delta u_{\alpha i}(k)$, $\Delta u_{\dot{\kappa}i}(k)$. We can obtain the following equations by expanding the equation and deleting nonlinear terms.

$$\boldsymbol{a}_i(k)\cdot\Delta\boldsymbol{z}_i(k) - b_i(k) = 0 \tag{25}$$

$$\begin{aligned}
\Delta\boldsymbol{z}_i(k) = [&\Delta u_{\alpha i}(k), \Delta u_{\dot{\kappa}i}(k), \Delta X_i(k), \Delta Y_i(k), \\
&\Delta\theta_i(k), \Delta V_i(k), \Delta\kappa_i(k), \Delta X_i(k+1)]^\top
\end{aligned} \tag{26}$$

Elements in the vector $\boldsymbol{a}_i$ and the constant $b_i$ in the above equation can be written as follows.

$$a_{1i} = \frac{1}{2}\cos\bar{\theta}_i(k)\Delta T^2 - \frac{1}{2}\bar{V}_i(k)\bar{\kappa}_i(k)\sin\bar{\theta}_i(k)\Delta T^3 \tag{27}$$

$$a_{2i} = -\frac{1}{6}\bar{V}_i(k)^2\sin\bar{\theta}_i(k)\Delta T^3 \tag{28}$$

$$a_{3i} = 1 \tag{29}$$

$$a_{4i} = 0 \tag{30}$$

$$\begin{aligned}
a_{5i} = &-\bar{V}_i(k)\sin\bar{\theta}(k)\Delta T - \frac{1}{2}\bar{u}_{\alpha i}(k)\sin\bar{\theta}(k)\Delta T^2 \\
&- \frac{1}{2}\bar{V}_i(k)^2\bar{\kappa}_i(k)\cos\bar{\theta}_i(k)\Delta T^2 - \frac{1}{2}\bar{u}_{\alpha i}(k)\bar{V}_i(k)\bar{\kappa}_i(k)\cos\bar{\theta}_i(k)\Delta T^3 \\
&- \frac{1}{6}\bar{u}_{\dot{\kappa}i}(k)\bar{V}_i(k)^2\cos\bar{\theta}_i(k)\Delta T^3 + \frac{1}{6}\bar{V}_i(k)^3\bar{\kappa}_i(k)^2\sin\bar{\theta}_i(k)\Delta T^3
\end{aligned} \tag{31}$$

$$\begin{aligned}
a_{6i} = &\cos\bar{\theta}_i(k)\Delta T - \bar{V}_i(k)\bar{\kappa}_i(k)\sin\bar{\theta}_i(k)\Delta T^2 \\
&- \frac{1}{2}\bar{u}_{\alpha i}\bar{\kappa}_i(k)\sin\bar{\theta}_i(k)\Delta T^3 - \frac{1}{3}\bar{u}_{\kappa i}\bar{V}_i(k)\sin\bar{\theta}_i(k)\Delta T^3 \\
&- \frac{1}{2}\bar{V}_i(k)^2\bar{\kappa}_i(k)^2\cos\bar{\theta}_i(k)\Delta T^3
\end{aligned} \tag{32}$$

$$\begin{aligned}
a_{7i} = &-\frac{1}{2}\bar{V}_i(k)^2\sin\bar{\theta}_i(k)\Delta T^2 - \frac{1}{2}\bar{u}_{\alpha i}(k)\bar{V}_i(k)\sin\bar{\theta}_i(k)\Delta T^3 \\
&- \frac{1}{3}\bar{V}(k)^3\bar{\kappa}_i(k)\cos\bar{\theta}_i(k)\Delta T^3
\end{aligned} \tag{33}$$

$$a_{8i} = -1 \tag{34}$$

$$\begin{aligned}
b_i(k) = &-\bar{X}_i(k+1) + \bar{X}_i(k) + \bar{V}_i(k)\cos\bar{\theta}_i(k)\Delta T \\
&+ \frac{1}{2}\bar{u}_{\alpha i}(k)\cos\bar{\theta}_i(k)\Delta T^2 - \frac{1}{2}\bar{V}_i(k)^2\bar{\kappa}_i(k)\sin\bar{\theta}_i(k)\Delta T^2 \\
&- \frac{1}{2}\bar{u}_{\alpha i}(k)\bar{V}_i(k)\bar{\kappa}_i(k)\sin\bar{\theta}_i(k)\Delta T^3 - \frac{1}{6}\bar{u}_{\dot{\kappa}i}(k)\bar{V}_i(k)^2\sin\bar{\theta}\Delta T^3 \\
&- \frac{1}{6}\bar{V}_i(k)^3\bar{\kappa}_i(k)^2\cos\bar{\theta}_i(k)\Delta T^3
\end{aligned} \tag{35}$$

In the same way, we can obtain the linear equation (36) by linearizing equation (3) – equation(6).

$$\boldsymbol{A}_i\cdot\Delta\boldsymbol{z}_i - \boldsymbol{b}_i = \boldsymbol{0} \tag{36}$$

The number of rows of the matrix $\boldsymbol{A}_i$ and the vector $\boldsymbol{b}_i$ become the product of prediction horizon $H_p$ and the length of the ego state vector $\boldsymbol{x}$. In the case of $H_p = 10$, the number of rows become 70.

### 5.2 State Vector Update

To obtain a solution, the linear search is also executed in accordance with the search direction $\Delta\boldsymbol{z}$ computed by the active-set solver. In our approach, we use the naive linear search to reduce the computational complexity. It determines the appropriate value to decrease the cost in the region of feasible. We use the modified BFGS method as the approximation of the Hessian matrix.

## 6. NUMERICAL EXPERIMENTS

We evaluate the suitability and efficiency of our proposed method for autonomous car motion planning. Evaluation items are as follows.

- **Required time for optimization**
  It is essential that the nonlinear optimization process be finished within a strictly defined time boundary that corresponds to the sampling time of MPC. We examine the characteristics of the required time by means of two use cases. We use three methods for comparison: (A) our proposed method, (B) nonlinear optimization without the approximated problem, and (C) fmincon, which is the constrained nonlinear optimization solver in MATLAB. We use the primal-dual interrior-point method (PDIPM) in the fmincon.
- **Suitability for autonomous car motion planning**
  Our approach delivers an efficient motion planning performance for an autonomous car. We use the car dynamics simulator CarSim (2017) for this evaluation. Since CarSim adopts highly accurate dynamics models, it is possible to evaluate the dynamical performance.

We use a 4.2 GHz Intel Core i7 running Windows 10 64 bit and MATLAB/Simulink2017a. Our optimization solver (active-set) and the approximated problem are written in C++.

### 6.1 Numerical Evaluation Settings

The required time for the optimization process depends on the cars and the road structure. We therefore construct two situations (Figs. 7 and 8) for our evaluation. The

biggest difference between the two situations is the number of other cars. In both cases, the ego has an initial velocity and a fixed cruising velocity. The ego has to plan the appropriate motions to overtake and to avoid collisions. In Fig. 7, the initial ego velocity and the directed cruising velocity $V_{\mathrm{ref}}$ are 20 [m/s]. The ego has to overtake other cars operating in front of it. Since the other cars (oth1 and oth2) have different velocities, longitudinal distance between the other cars increases with time. The ego needs to navigate through the space between oth1 and oth2. In Fig. 8, the initial ego velocity is 16 [m/s] and directed cruising velocity $V_{\mathrm{ref}}$ is set to 18 [m/s]. The ego is encouraged to accelerate and reach cruising velocity, but the front space is occupied by three cars. Furthermore, the longitudinal relative distance between the ego and oth2 decreases with time due to the relative velocity and the three cars (oth1, oth3, oth4) occupying the rear space. The ego has to plan a motion that prevents collision with oth2. This motion require accurate planning of acceleration $u_\alpha$ and curvature change rate $u_{\dot{\kappa}}$.

Our motion planning is launched with the period of $\Delta T = 0.5$ [sec] and the predictive horizon $H_p$ is 10 [step]. The control horizon is the same as the predictive horizon. Thus, the motion of the ego is planned through a 5 [sec] length prediction. We limited the maximum number of iterations of SQP to 30 times in the non-congested case (Fig.7) and 10 times in the congested case (Fig.8).

## 7. DISCUSSION

The evaluation results for required time are listed in Table 1. Our approach has a better performance than fmincon in both use cases. Furthermore, the difference between the average time and the worst case time is smaller with our approach. This is because the approximated problem generates variables near the feasible region, which is an important characteristic for real-time motion planning.

Figures 9, 10, and 11 show the results of the motion planning and dynamics evaluation. In both cases, the ego was able to control acceleration and curvature change ratio to avoid collision and overtake properly. In the figures, the horizontal axis shows elapsed time and the two vertical axes show the ego lateral position and longitudinal relative distance between each car. Red plots represent the results of motion planning. Ten-step motion is planned in every control period $\Delta T = 0.5$ [sec], so a red line is generated and plotted every $\Delta T$. Each red line consists of 11 markers, with a square marker indicating the ego state when the motion planner launched and a red circle indicating the first step of a planned motion, i.e., the target ego state transition after $\Delta T$ [sec]. The ego in CarSim is controlled to track the planned motion (red line), but the planned motion is modified by replanning after $\Delta T$. As shown, when the old planned motion is aborted, the ego executes a new planned motion. The blue line represents the ego transition simulated by CarSim. The green line shows the relative longitudinal distance between the ego and an other car.

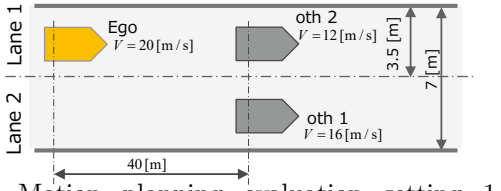


Fig. 7. Motion planning evaluation setting 1 (non-congested case).

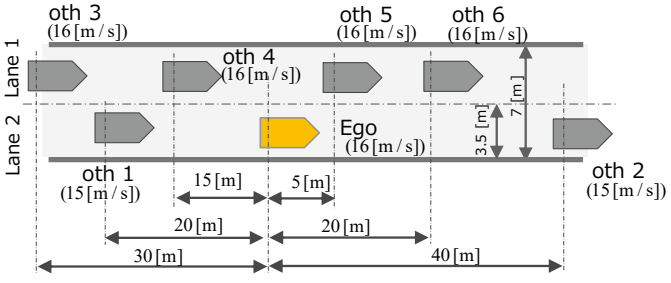


Fig. 8. Motion planning evaluation setting 2 (congested case).

These graphs reveal several interesting characteristics. Every red circle accurately corresponds to a blue line, thanks to the car dynamics equations. This feature provides a definite advantage when it comes to controlling a car precisely. As shown in these plots, the ego motion is strictly planned within the road boundary. Moreover, the ego lateral position stays around the centers of each lane ($\pm 1.75$[m]), except during lane changes. This is obviously caused by the cost function (8) and inequality constraints (19). In Fig. 9, when the ego changes lanes at around 6 [sec], its lateral position becomes under 2 [m], possibly due to the cost function for $u_{\dot{\kappa}}$, which encourages a decrease in rapid curvature change. However, in all cases, the ego lateral position did not move beyond the road boundaries (the ego width is set to 1.7 [m]). Frequently, some planned motion (red line) changed significantly from the previous one. This means that the previous planned motion should not be used for optimizing the initial variables.

An interesting motion emerged in Figs. 10 and 11. The ego slightly accelerated and moved into the space between oth5 and oth6 to avoid a collision with oth2, and when oth2 moved to the rear of the ego (20 [sec]–33 [sec]), the ego changed lanes quickly and accelerated to the directed velocity $V_{\mathrm{ref}} = 18$ [m/s]. During this motion, the ego planned acceleration appropriately. This demonstrates that the ego motion is appropriately generated by our proposed approach.

## 8. CONCLUSION

We have developed a highly efficient nonlinear optimization method for autonomous car motion planning. The autonomous car motion planning problem in MPC essentially

Table 1. Evaluation Result : Required Time.
(A) our method, (B) without the approximated problem, (C) fmincon (PDIPM)
Max iteration is restricted in (A) and (B) (non-congested case: 30, congested case : 10).

| Use Case | Method | Average [sec] | Worst case [sec] | Note |
|---|---|---|---|---|
| Non-congested | (A) | 0.32 | 0.37 | |
| | (B) | 0.19 | 0.72 | Including infeasible |
| | (C) | 9.2 | 12.7 | |
| Congested | (A) | 0.16 | 0.47 | |
| | (B) | 0.21 | 0.81 | Including infeasible |
| | (C) | 9.0 | 20 | |

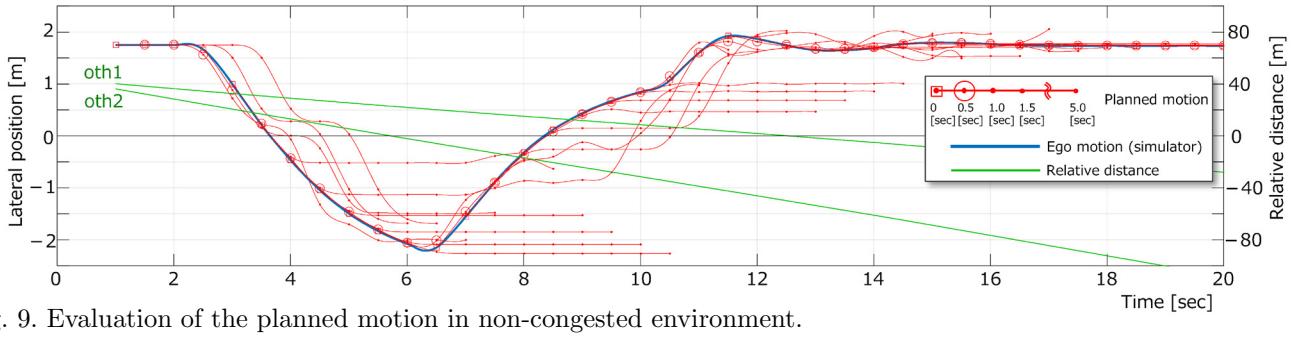


Fig. 9. Evaluation of the planned motion in non-congested environment.

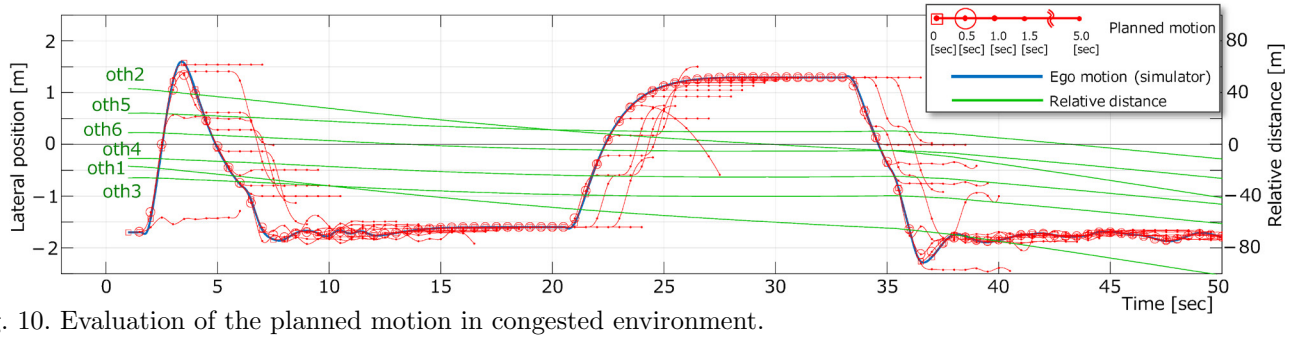


Fig. 10. Evaluation of the planned motion in congested environment.

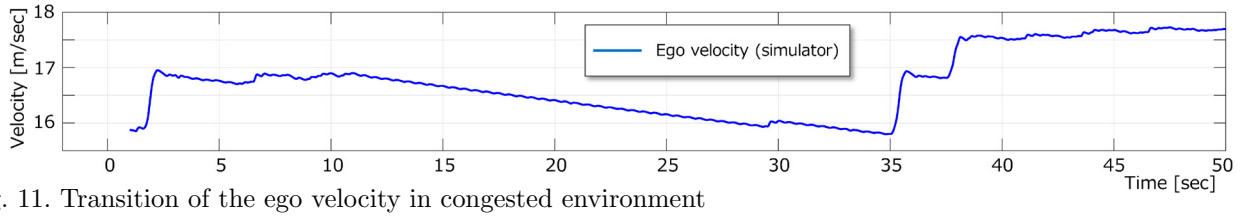


Fig. 11. Transition of the ego velocity in congested environment

contains nonlinear optimization and nonlinear constraints, which makes it highly computationally complex. To deal with this problem, we adopt a two-step problem solving method that combines the approximated problem and nonlinear optimization. The approximated problem can reduce the number of variables to be searched, and it can quickly generate a coarse solution that is close to a feasible point of the original problem with no iteration. The nonlinear optimization can then quickly reach convergence by using the coarse solution from the approximated problems. The advantage of our method is not only computation speed but also its ability to maintain nonlinear characteristics in the motion planning problem and deal with rapid changes of appropriate plan. We evaluated our approach using a car dynamics simulator and found its performance to be sufficient.

## REFERENCES

Diehl, M., Bock, G. H., and Scheloder, P. J. (2005). *A Real-Time Iteration Scheme for Nonlinear Optimization in Optical Feedback Control.* SIAM Journal on Control and Optimization, vol. 43, no.5, pp.1714–1736.

Goldfarb, D., and Idnami, A. (1983). *A Numerically Stable Dual Method for Solving Strictly Convex Quadratic Programs.* Mathematical Programming, vol. 27, Issue 1, pp. 1–33.

Gonzalez, D., Prez, J., Mialnes, V., and Nashashibi, F. (2016). *A Review of Motion Planning Techniques for Automated Vehicles.* IEEE Transactions of Intelligent Transportation Systems, vol. 17, no. 4.

Kawabe, H., Nishira, H., and Ohtsuka, T. (2004). *An optimal path generator using a receding horizon control sheme for intelligent automobiles.* IEEE International Conference on Control Applications.

Mechanical Simulation Corporation. (2017). *CarSim Math Models.* https://www.carsim.com/.

Obayashi, M., Uto, K., and Takano G. (2016). *Appropriate Overtaking Motion Generating Method Using Predictive Control with Suitable Car Dynamics.* IEEE Conference on Decision and Control (CDC).

Ohtsuka, T. (2004). *A Continuation/GMRES Method for Fast Computation of Nonlinear Receding Horizon Control.* Automatica, vol. 40, no.4, pp.563–574.

Polack, P., Altche, F., d'Andrea-Novel, B., and Fortelle, L., A. (2017). *The Kinematic Bicycle Model: a Consistent Model for Planning Feasible Trajectories for Autonomous Vehicles?.* IEEE Intelligent Vehicles Symposium.

Werling, M., and Liccardo, D. (2012). *Automatic Collision Avoidance Using Model-predictive Online Optimization.* IEEE Conference on Decision and Control (CDC).

Yi, B., Gottschling, S., Ferdinand, J., Simm, N., Bonarens, F., and Stiller, C. (2016). *Real Time Integrated Vehicle Dynamics Control and Trajectory Planning with MPC for Critical Maneuvers.* IEEE Intelligent Vehicles Symposium (IV).

Ziegler, J., Bender, P., Dang, T., and Stiller, C. (2014). *Trajectory planning for Bertha - A local, continuous method.* Intelligent Vehicles Symposium Proceedings.