Proceedings of the ASME 2020
Dynamic Systems and Control Conference
DSCC2020
October 5-7, 2020, Virtual, Online

DSCC2020-3138

MOTION PLANNING FOR AUTONOMOUS DRIVING WITH EXTENDED CONSTRAINED ITERATIVE LQR

Yutaka Shimizu*
Graduate School of
Information Science and Technology
University of Tokyo
7-3-1 Hongo Bunkyo-ku
Tokyo 113-0033
Japan.
Email: yutaka.shimizu@pf.is.s.u-tokyo.ac.jp

Wei Zhan, Liting Sun, Jianyu Chen
Department of Mechanical Engineering
University of California
Berkeley, CA 94720
Email: wzhan, litingsun, jianyuchen@berkeley.edu

Shinpei Kato †
Graduate School of
Information Science and Technology
University of Tokyo
7-3-1 Hongo Bunkyo-ku
Tokyo 113-0033
Japan.
Email: shinpei@is.s.u-tokyo.ac.jp

Masayoshi Tomizuka

Department of Mechanical Engineering
University of California
Berkeley, CA 94720
Email: tomizuka@berkeley.edu

ABSTRACT

Autonomous driving planning is a challenging problem when the environment is complicated. It is difficult for the planner to find a good trajectory that navigates autonomous cars safely with crowded surrounding vehicles. To solve this complicated problem, a fast algorithm that generates a high-quality, safe trajectory is necessary. Constrained Iterative Linear Quadratic Regulator (CILQR) is appropriate for this problem, and it successfully generates the required trajectory in real-time. However, CILQR has some deficiencies. Firstly, CILQR uses logarithmic barrier functions for hard constraints, which will cause numerical problems when the initial trajectory is infeasible. Secondly, the convergence speed is slowed with a bad initial trajectory, which might violate the real-time requirements. To address these problems, we propose the extended CILQR by adding two new features. The first one is using relaxed logarithmic barrier functions instead of the standard logarithmic barrier function to prevent numerical issues. The other one is adding an efficient initial trajectory creator to generate a good initial trajectory. Moreover, this initial trajectory helps CILQR to converge to a desired local optimum. These new features extend CILQR's usage to more practical autonomous driving applications. Simulation results show that our algorithm is effective in challenging driving environments.

1 INTRODUCTION

Autonomous Vehicles (AVs) have a great potential to improve our society and life. Since the 2007 DARPA Urban Challenge, autonomous driving has gained lots of attention from both academia and industry. However, there are still many unsolved problems in the field of autonomous driving. One of the challenging problems is path planning, which requires generating a real-time safe, smooth and efficient trajectory to navigate the au-

*This work was conducted during a visit to the Mechanical Systems Control Laboratory at University of California, Berkeley.

†TierIV Inc., 3-22-5 Hongo, Apt. 13F, Bunkyo-ku, TOKYO, 113-0033

tonomous vehicle in highly complicated environments. To address this challenging task, a considerable amount of works have been reported.

Optimization-based approach is one of the effective ways to solve this problem. It has gained much more attention since Mercedes-Benz showed its effectiveness. They formulate autonomous driving motion planning to a nonlinear optimization problem and solve it with Sequential Quadratic Programming (SQP) [9] and successfully ran the autonomous vehicle for 100 km without human intervention. Though the algorithm can deal with many challenging situations, its calculation time needs to be further reduced for emergency situations.

Researchers have proposed alternative optimization-based motion planners. Constrained Iterative Linear Quadratic Regulator (CILQR) [1, 2] is a newly proposed approach to this problem. It is based on Iterative Linear Quadratic Regulator (ILQR), which is an efficient algorithm to solve nonlinear optimal control problems. However, ILQR cannot handle both state and input constraints. Due to various state and input constraints in robot planning problems, it is not easy to apply ILQR to practical robotics problems. CILQR iteratively uses a shaped barrier function to embed the constraints into the objective function and solve the resulting unconstrained planning problem using ILQR. This approach shows faster convergence than SQP. However, there are some problems to CILQR to real-world autonomous driving, including numerical stability and convergence rate. The former problem comes from the barrier function, which will fail when the initial trajectory is infeasible. The latter comes from the quality of the initial trajectory since the convergence rate is slowed with a bad initial trajectory.

In this paper, we propose the extended CILQR algorithm that improves numerical stability and convergence rate. First, we design a new barrier function that relaxes the standard logarithm barrier function to solve the numerical stability problem. Then, we develop a new efficient initial trajectory creator that generates better initial trajectories faster. Finally, we test our proposed algorithm on simulated driving environments and verify that extended CILQR can handle some complicated driving situations when the original CILQR fails to solve. Furthermore, the computational time of the proposed method is fast enough to be used in a practical application.

The main contribution of this paper can be summarized as follows.

Relaxed logarithmic barrier function:
We propose a relaxed logarithmic function that solves the numerical stability problem when initial trajectory is infeasible.
Fast initial trajectory creator:
We develop a new fast initial trajectory creator that generates high quality trajectory.
Simulation in a practical environment We simulate our new algorithm in challenging situations where the original CILQR fails. Furthermore, we confirm that our new algorithm has fast computation speed.

The paper is organized as follows. In Section 2, we introduce some related works that has been developed. Section 3 explains the basic idea of ILQR, Differential Dynamic Programming (DDP) and CILQR. Section 4 introduces the relaxed logarithmic function and Section 5 gives details of the initial trajectory creator. Finally, Section 6 shows the result of the simulation and Section 7 concludes this paper and discusses the future work.

2 Related Works

Learning-based motion planning methods, such as imitation learning [3] and inverse reinforcement learning [4], are capable of learning the driving preference of human drivers, but lack the theoretical guarantee for safety and feasibility. On the other hand, conventional planning approaches without learning, such as search-based [5], sampling-based method and the optimization-based methods are able to theoretically guarantee the generated trajectories to be collision-free and feasible. The latter two approaches will be introduced in this section.

Sampling-based method has been developed for many years. It can be divided into two categories, random sampling-based and deterministic sampling-based. Random sampling-based method samples the destination points randomly in a configuration space. Typical methods in random sampling includes Rapidly-exploring Random Tree (RRT) [11] and its variant RRT* [35]. Random-based approach is very efficient when the problem state space has high dimensions, but the sampling resolution has a significant effect on computation time and trajectory quality. Deterministic sampling-based method samples the primitives in a deterministic way [28, 29], or adaptively [6]. Though deterministic sampling method is very efficient, it often results in sub-optimal trajectories.

Optimization-based method uses numerical optimization to obtain the optimal trajectory. As mentioned in Section 1, Mercedes-Benz formulates the autonomous driving planning problem into a non-convex optimization problem and then use SQP to solve it. Qian et al incorporates integer constraints into the planning problem to address logical constraints [30]. However, these approaches still require substantial computation time. To make the planning algorithm more efficient, Liu et al propose the convex feasible set (CFS) algorithm [31] to solve a specific non-convex optimization problem efficiently. Furthermore, they apply this algorithm to various planning problems for autonomous driving [32–34]. However, CFS requires inverse dynamics to get the optimal trajectory, thus the resulting trajectory is possibly infeasible. Another approach is to use the indirect shooting method that explicitly enforces system dynamics. This method is related to optimal control theory. LQR, ILQR and DDP [21] [22] [24] are typical methods in this field. However, these approaches have significant difficulty to incorporate the state and control constraints which are essential for robot control. To address this problem, some papers [1, 2, 24] incorporate constraints into ILQR/DDP. [24] incorporates input constraints, but they did not consider the state constraint. CILQR [1] [2] has incorporated the input and state constrains but they still have some problems. The biggest issue is that they need a feasible initial trajectory to solve the problem. [1] and [2] assume that the controller already has a feasible initial trajectory. However, this assumption does not necessarily hold. Moreover, it is not easy

to generate a feasible trajectory in real time. To the best of our knowledge, there are no prior works to include initial trajectory feeder to constrained ILQR or DDP.

3 Review of CILQR

3.1 Naive ILQR and DDP

First, we give a short review of ILQR and DDP. Let $x \in R^n$ be the state vector and $u \in R^m$ be the input vector. The path planning problem can be formulated in the following form.

$$x^*, u^* = \arg\min_{x,u} \{\phi(x_N) + \sum_{n=1}^{N-1} L^k(x_k, u_k)\} \tag{1a}$$

$$s.t. \;\; x_{k+1} = f^k(x_k, u_k), \quad k = 0, 1, 2, , , , , N-1 \tag{1b}$$

$$x_0 = x_{start} \tag{1c}$$

$$g(x, u) < 0 \tag{1d}$$

Note that (1a) is a cost function where $\phi(x_N)$ is the final stage cost and $L^k(x_k, u_k)$ is the stage cost at time step k. In addition, (1b) is the system dynamics equation and (1c) is the initial condition for state. Also $g(x, u)$ descries state constraint and input constraint.

ILQR/DDP uses dynamic programming to solve the unconstrained optimal control problem with non-linear cost function and system dynamics. The unconstrained optimal control problem takes the following form:

$$x^*, u^* = \arg\min_{x,u} \{\phi(x_N) + \sum_{n=1}^{N-1} L^k(x_k, u_k)\} \tag{2a}$$

$$s.t. \;\; x_{k+1} = f^k(x_k, u_k), \quad k = 0, 1, 2, , , , , N-1 \tag{2b}$$

$$x_0 = x_{start} \tag{2c}$$

ILQR/DDP is based on dynamic programming, specifically, the following Bellman equation.

$$V^k(x_k) = \min_{u_k} \left[L^k(x_k, u_k) + V^{k+1}(f^k(x_k, u_k)) \right] \tag{3}$$

where $V^k(x_k)$ is the cost-to-go at x_k. ILQR/DDP iterates with the following three steps until convergence:

(Step 1) **Nominal Trajectory Definition:** If at the first iteration, we need to define an initial control sequence $\overline{u}$ and propagate the system with equation (2b) and get a nominal trajectory $(\overline{x}, \overline{u})$. For later iterations, the nominal trajectory is calculated by the forward pass (step 3).

(Step 2) **Backward Pass:** In the second step, we have the final step cost-to-go $V^N(x_N) = \phi(x_N)$. Then from time step $N-1$ we start to calculate the following backward pass:

$$\begin{aligned} P^k(\delta x_k, \delta u_k) = L^k(\overline{x_k} + \delta x_k, \overline{u_k} + \delta u_k) - L^k(\overline{x_k}, \overline{u_k}) + \\ V^{k+1}(f^k(\overline{x_k} + \delta x_k, \overline{u_k} + \delta u_k))) - V^{k+1}(f^k(\overline{x_k}, \overline{u_k})) \end{aligned} \tag{4}$$

Approximating $P^k(\delta x_k, \delta u_k)$ by its second order Taylor expansion, we get:

$$P^k(\delta x_k, \delta u_k) = \frac{1}{2} \begin{bmatrix} 1 \\ \delta x_k \\ \delta u_k \end{bmatrix}^T \begin{bmatrix} 0 & (P_x^k)^T & (P_u^k)^T \\ P_x^k & P_{xx}^k & P_{xu}^k \\ P_u^k & P_{ux}^k & P_{uu}^k \end{bmatrix} \begin{bmatrix} 1 \\ \delta x_k \\ \delta u_k \end{bmatrix} \tag{5}$$

where

$$P_x^k = L_x^k + (f_x^k)^T V_x^{k+1} \tag{6a}$$

$$P_u^k = L_u^k + (f_u^k)^T V_x^{k+1} \tag{6b}$$

$$p_{xx}^k = L_{xx}^k + (f_x^k)^T V_{xx}^{k+1} f_x^k + V_x^{k+1} \bullet f_{xx}^k \tag{6c}$$

$$p_{uu}^k = L_{uu}^k + (f_u^k)^T V_{xx}^{k+1} f_u^k + V_x^{k+1} \bullet f_{uu}^k \tag{6d}$$

$$p_{ux}^k = L_{ux}^k + (f_u^k)^T V_{xx}^{k+1} f_x^k + V_x^{k+1} \bullet f_{ux}^k \tag{6e}$$

The third terms in (6c), (6d) and (6e) are the product of a vector with a tensor, which are ignored in ILQR. Then we can get the optimal control strategy by minimizing (5)

$$\delta u_k^* = \arg\min_{\delta u_k} P^k(x_k, u_k) = -(p_{uu}^k)^{-1}(P_u^k + P_{ux}^k \delta x_k) \tag{7}$$

The derivative of value function V_x^k and Hessian of value function V_{xx}^k are updated in the following formulation:

$$V_x^k = P_x^k - P_u^k (P_{uu}^k)^{-1} P_{ux}^k \tag{8a}$$

$$V_{xx}^k = P_{xx}^k - P_{xu}^k (P_{uu}^k)^{-1} P_{ux}^k \tag{8b}$$

This computation is repeated till reaching first time step $k = 0$.

(Step 3) **Forward Pass:** After getting the control strategy at each time step k, we compute the new trajectory from time step $k = 0$ to $k = N-1$.

$$x_0 = x_{start} \tag{9a}$$

$$u_k = \overline{u_k} + \delta u_k^* \tag{9b}$$

$$x_{k+1} = f^k(x_k, u_k) \tag{9c}$$

3.2 Constrained ILQR

Iterative LQR is an effective way to handle the optimal control problem with non-linear dynamics. However, it cannot handle state and control constraints. To cope with this problem, we use Constrained ILQR [1] [2]. CILQR transforms constraints into a certain form of penalties and add it to the cost function. [1] and [2] proposed two types of penalty functions. The first one is logarithmic barrier function and the second one is exponential barrier function. [1] further pointed out that logarithmic barrier function is better than exponential one because logarithmic barrier function can make hard constraint with fewer parameters. It also helps provide a theoretical proof the convergence. Therefore, we use logarithmic barrier function to penalize the constraints. It is expressed as:

$$b(g(x,u)) = -\frac{1}{t}\log(-g(x,u)) \tag{10}$$

where t is a parameter and $g(x,u)$ is a constraint function.

By using the barrier function described above, the objective function will be

$$J = \phi(x_N) + \sum_{n=1}^{N-1} L^k(x_k,u_k) + b(g(x,u)) \tag{11}$$

The detail of the algorithm and its properties can be found in [1] [2].

4 Relaxed Logarithmic Function

In this section we focus on the relaxed barrier function. The original logarithmic barrier function has a serious issue when calculating the trajectory, that is the numerical stability. This is because $\log(x)$ is undefined when $x \leq 0$. Especially, we will encounter this issue when the nominal trajectory collides with the obstacles. Fig. 1 depicts this situation. Red rectangles are obstacles and blue dot line is the initial nominal trajectory.

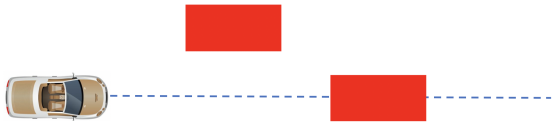

Figure 1. Failed pattern when calculating nominal trajectory

There are two options to solve this problem. The first one is to provide a strictly feasible initial trajectory, which means the trajectory is dynamically feasible and has no collision with surrounding objects. The other option is to modify the logarithmic barrier function. The first one seems effective, but it is not easy to generate a trajectory in real time that meets the requirements. Therefore, instead of providing a strictly feasible initial trajectory, we modify the standard barrier function to a relaxed barrier function, which is inspired by [37]. The relaxed barrier function is defined as below:

$$\beta_\delta(z) = \begin{cases} -\log(z) & (z > \delta) \\ \frac{k-1}{k}\left[\left(\frac{z-k\delta}{(k-1)\delta}\right)^k - 1\right] - \log\delta & (z \leq \delta) \end{cases} \tag{12}$$

where δ is a parameter in the range $0 < \delta \leq 1$. $k > 1$ is also a parameter and is an even integer. The authors in [37] suggest to use $k = 2$ and this makes the function continuously differentiable at $z = \delta$. A visualization of the function is described in Fig. 2.

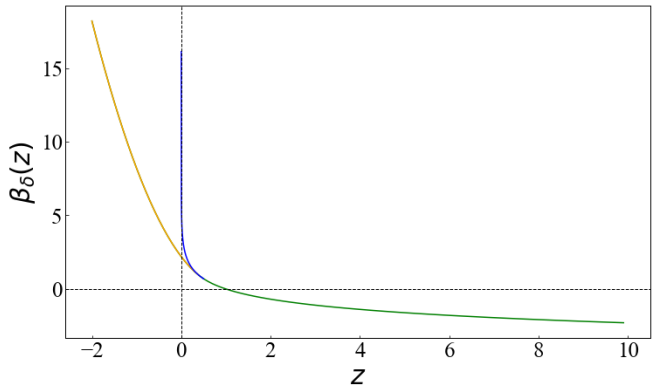

Figure 2. Relaxed Barrier Function

In Fig. 2, we set $\delta = 0.5$. Orange line describes the graph for $z \leq \delta$ and green line for $z > \delta$. The blue line represents original function in $0 < z \leq \delta$.

This function prevents the numerical stability problem because the new logarithmic function is well defined when when $x < 0$. This means the problem can be solved even when the nominal trajectory is infeasible or collides with the obstacles. It also benefits our initial trajectory creator, which is discussed in the next section.

5 Initial Trajectory Creator

In this section, we give the details of a fast initial trajectory creator for CILQR. As mentioned previously, CILQR itself has unstable convergence rate which is significantly influenced by the quality of the initial trajectory. To avoid this problem, we decide to add a new algorithm that provides a good initial trajectory for CILQR. However, we need to meet specific requirements when generating the initial trajectory. First, the initial trajectory should be in the vicinity of the global optimum trajectory. Second, its computation time should be acceptable to use. Although

there are many approaches to generate a trajectory in real time, most of them do not meet both requirements. For example, the widely used polynomial curve method will generate dynamically infeasible trajectories.

To satisfy the above requirements, we create a new initial trajectory generator based on [14]. However we cannot directly use the approach in [14] because the calculation time is high. The approach in [14] first samples destination points and draws a straight line to be a reference line. After that it smooths the trajectories using DDP and selects the best trajectory defined by cost functions. The approach can generate collision-free high quality trajectories, but smoothing all trajectories by DDP will cause unacceptable calculation time. To address this problem, we modify the method in [14] by utilizing the characteristics of the relaxed barrier function. The newly proposed algorithm has four elements as depicted in Fig. 3. We now introduce each component step by step.

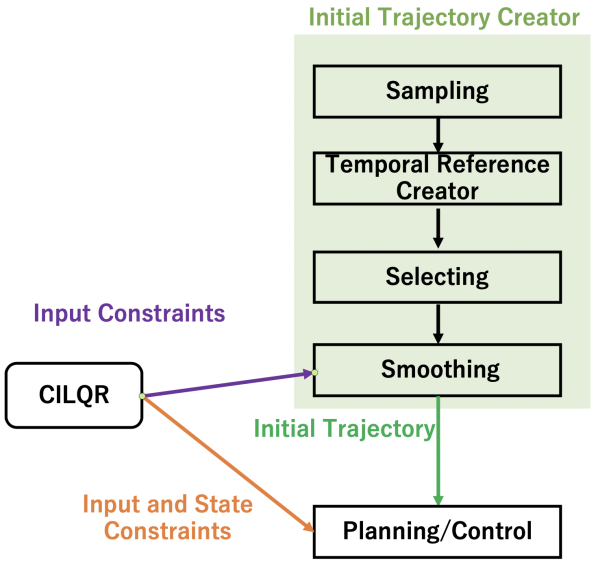

Figure 3. The structure of Initial Trajectory Creator

5.1 Sampling

The first step is to sample points around a destination point. Destination point is decided by a global reference trajectory. An example of the sampling process is shown in Fig.4. The blue dashed line denotes the reference trajectory and the blue point and red points denote the sampled destination points.

5.2 Creating temporal reference paths

After sampling the destination points, we generate temporal reference paths by linear interpolation. In Fig.5, the temporal reference paths are represented by the straight green lines.

5.3 Selection

In this step, we select an optimal temporal reference path among the paths generated in the previous step. The optimality is defined by minimizing a defined cost function. We use three types of cost as described in [16] [38], which are deviation cost J_d, static safe cost J_s and consistency cost J_c. They can be expressed in the following form.

$$J_d = \sum_{i=1}^{N}(x_i - x_{i,ref}) \tag{13a}$$

$$J_s = \sum_{k=1}^{N_s} r[k]g[i-k] \tag{13b}$$

$$J_c = \sum_{i=1}^{N}(x_i - x_{i,prev}) \tag{13c}$$

$$J = w_d J_d + w_s J_s + w_c J_c \tag{13d}$$

where x_i, $x_{i,ref}$ and $x_{i,prev}$ represents the ith point on the generated temporal reference path, global reference path and previous temporal reference path. Global reference path can be center line of the road or generated by the upper layer planer. w_d, w_s and w_c are the weights for each cost, which are hyper parameters. N_s is the number of the generated temporal reference paths, and N is the horizon. $r[k]$ and $g[i-k]$ are defined as follows:

$$r[k] = \begin{cases} 1.0 & (if\ k\ th\ path\ is\ passing\ the\ obstacle) \\ 0.0 & (otherwise) \end{cases} \tag{14}$$

$$g[i] = \frac{1}{\sqrt{2\pi}\sigma}\exp(-\frac{i^2}{2\sigma^2}) \tag{15}$$

σ represents the influence scope of the adjacent trajectory's value. After calculating the cost of each path, we select a path which has the minimum cost. Fig.6 illustrates the process. In the figure, the black rectangle represents an obstacle and red lines represent the paths collided with the obstacle. The yellow line represents the path that has minimal cost.

5.4 Smoothing

In this step we smooth the path. The path selected in the previous step is not dynamically feasible and the quality of the path does not satisfy our requirements. In [14], all straight paths are smoothed, which takes a lot of time. Furthermore, [14] uses DDP which means they need to calculate Hessian matrices, which costs further computation time. To prevent this issue we only smooth the selected path. This makes the calculation time much shorter than [14]. Therefore the smoothing problem can be de-

scribed as

$$x^*, u^* = \arg\min_{x,u}\{\phi(x_N) + \sum_{k=0}^{N-1} L^k(x_k, u_k, x_{ref,k})\} \tag{16a}$$

$$s.t.\ x_{k+1} = f^k(x_k, u_k),\ k = 0, 1, ..., N-1 \tag{16b}$$

$$x_0 = x_{start} \tag{16c}$$

$$g(u) < 0 \tag{16d}$$

where $x_{ref,k}$ is the point on the temporal reference straight path at time step k and $g(u)$ is the input constraint. Moreover, x_{start} indicates current vehicle's position. This problem can be efficiently solved by CILQR. The generated trajectory is dynamically feasible but might not be collision-free. Fig. 7 illustrates this process.

6 Case Studies

In this chapter, we test the proposed algorithm in various situations where naive CILQR fails. Furthermore, we confirm that our algorithm can be implemented in real time and is fast enough to be used in real vehicle driving scenarios. In our experiments, we use an open source platform called Autoware [39] [40], which is an open source autonomous driving platform based on Robot Operating System (ROS).

In our test scenario, we model the vehicle with a bicycle kinematics model, which is shown in Fig. 8. This vehicle model has four state variables. Position x, y, velocity v and heading angle θ. Furthermore, the vehicle has two control inputs, which are acceleration α and steering angle δ. Note that we can get the relationship between steering angle δ and curvature κ from the following equation

$$\kappa = \frac{\tan(\delta)}{L} \tag{17}$$

where L is the length of wheel base of the vehicle. We use this relationship to convert steering angle to curvature. Moreover, we assume the steering angle is maintained during a sampling time T_r and the vehicle will rotate around the instant center O with the radius of r. The distance the vehicle moves in one sampling time is

$$l = vT_r + \frac{1}{2}aT_r^2 \tag{18}$$

Figure 4. Sampling

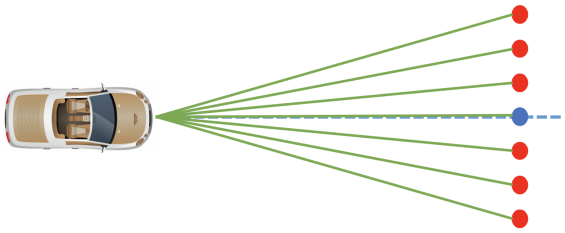

Figure 5. Straight Line Generator

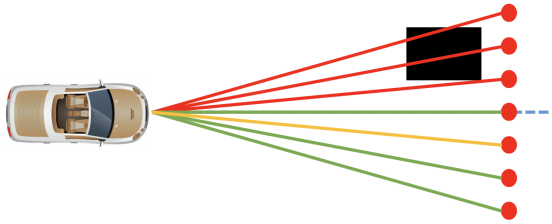

Figure 6. Cost calculation

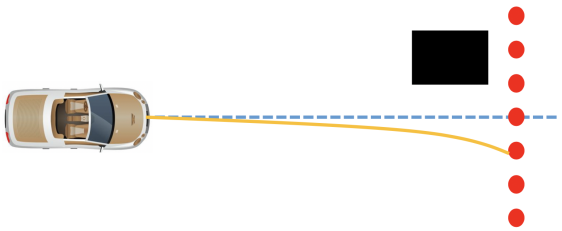

Figure 7. Smoothing

Therefore the vehicle state after one sampling time will be

$$v_{k+1} = v_k + \alpha T_r \tag{19a}$$

$$\theta_{k+1} = \theta_k + \int_0^l \kappa ds = \theta_k + \kappa l \tag{19b}$$

$$x_{k+1} = x_k + \int_0^l \cos(\theta_0 + \kappa s) ds = x_k + \frac{\sin(\theta_0 + \kappa l) - \sin\theta_0}{\kappa} \tag{19c}$$

$$y_{k+1} = y_k + \int_0^l \sin(\theta_0 + \kappa s) ds = y_k - \frac{\cos(\theta_0 + \kappa l) - \cos\theta_0}{\kappa} \tag{19d}$$

To summarize this equation, we transform the equation to the vector form.

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ v_{k+1} \\ \theta_{k+1} \end{bmatrix} = f\left(\begin{bmatrix} x_k \\ y_k \\ v_k \\ \theta_k \end{bmatrix}, \begin{bmatrix} \alpha \\ \kappa \end{bmatrix} \right) \tag{20}$$

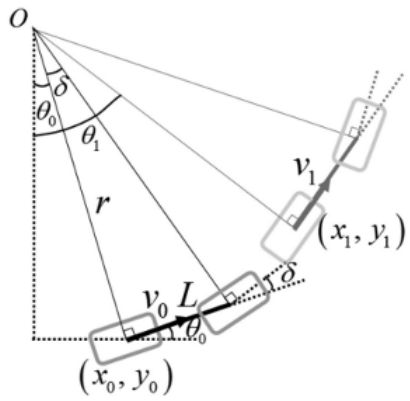

Figure 8. Bicycle Kinematics Model

After formulating the vehicle model, we design our cost functions. We have two types of cost functions, for CILQR in initial trajectory creator and for CILQR used in planning/control section. The first cost function J_1 can be written in the following form:

$$J_1 = (x_k - x_{k,ref})^T Q(x_k - x_{k,ref}) + u_k^T R u_k - \log(-g(u)) \tag{21}$$

where x_k and u_k represents vehicle state and control input. $x_{k,ref}$ is the reference trajectory generated in step B in section 5. $g(u)$ is the input constraint function which will be desribed later. On the other hand, the cost function J_2, which is used for CILQR in planning/control section, is described in the following form,

$$\begin{aligned} J_2 =& (x_k - x_{k,ref})^T Q(x1_k - x_{k,ref}) + \\ & (u_k - u_{k,ref})^T R(u_k - u_{k,ref}) - \log(-g(x,u)) \end{aligned} \tag{22}$$

where $x_{k,ref}$ is the reference trajectory that is defined by a global planner or waypoints collected from human driving experiences. $u_{k,ref}$ is the reference input generated by our initial trajectory creator. $g(x,u)$ is the state and input constraint function and can be split as $g(x,u) = g(x) + g(u)$.

The constraint function also has two kinds of forms. One is used in equation (21) that only constraints the control input and the other one is used in equation (22) that constraints both input and state. Input constraints can be written in the following form:

$$-\begin{bmatrix} a_{k,lim} \\ \kappa_{k,lim} \end{bmatrix} \leq u_k \leq \begin{bmatrix} a_{k,lim} \\ \kappa_{k,lim} \end{bmatrix} \tag{23}$$

where $a_{k,lim}$ and $\kappa_{k,lim}$ are acceleration and curvature constraint values at time step k. As a result, the input constraint function takes the following two forms:

$$g(u) = u_k - u_{k,lim} \tag{24a}$$
$$g(u) = -u_{k,lim} - u_k \tag{24b}$$

State constraints are for collision avoidance. Let d_{margin} be the safety distance between ego vehicle and an obstacle. Then the constraint can be written as

$$g(x) = d_{margin} - d(x_k, O_i) \tag{25}$$

where $d(x_k, O_i)$ is the Euclid distance between the ego vehicle position x_k and ith obstacle position O_i. The parameters details are given in Table 1.

Table 1. Parameters used in our simulation

Parameter	value
k_{lim}	0.25
$a_{lim}[\frac{m}{s^2}]$	1.0
Sampling time $[s]$	0.2
Preview horizon	70
Vehicle length$[m]$	4.5
Vehicle width$[m]$	1.7

The code is written in C++ and ROS Melodic. We also use Autoware version 1.13 and run on a laptop with 4.5GHz Intel Core i7-9750H

6.1 Initial Trajectories generated by the proposed algorithm

First of all, we show the result of initial trajectory generated by the proposed algorithm. Fig. 9 shows the result with an obstacle. The pink trajectory is the generated trajectory and other trajectories are reference straight paths. The red paths describe paths that will collide with the obstacle. Green trajectory

indicates global reference trajectory generated by a global planner. Moreover, we test the algorithm in a cluttered environment which is shown in Fig. 10. As this picture shows, the algorithm is effective even in a very complicated environment. We also find that the algorithm is fast enough to use in a real driving situation. Table. 2 shows the computation time of each environment. We measures 400 times and give the average time, max time, minimum time and the standard deviation. "Simple" indicates the environment with one obstacle, shown in Fig. 9 and "Cluttered" describes the environment with many obstacles, shown in Fig. 10.

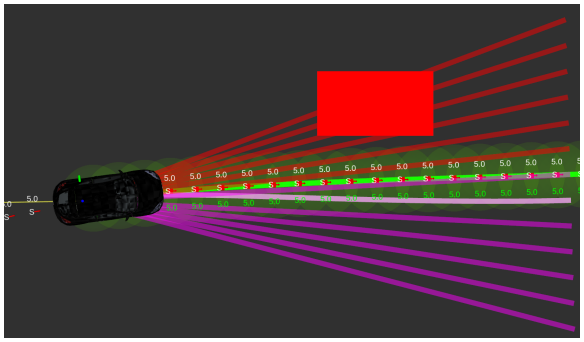

Figure 9. Initial Trajectory with one obstacle

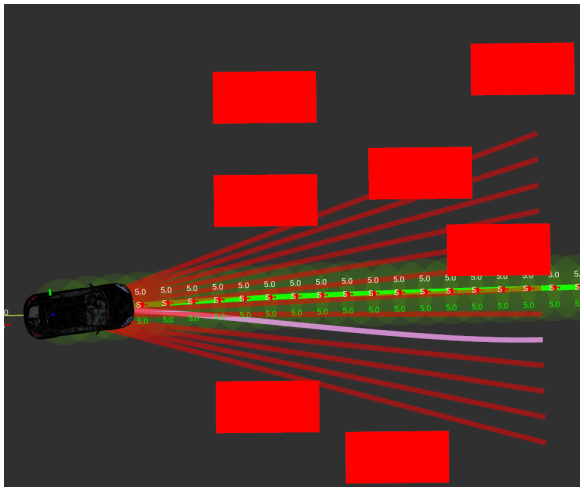

Figure 10. Initial Trajectory with many obstacles

6.2 CILQR with Initial Trajectory Creator

In this subsection, we give the result of the CILQR with Initial Trajectory Creator. Fig. 11 shows the trajectory generated by CILQR with Initial Trajectory Creator. In this situation, original CILQR will fail as its nominal trajectory will pass the obstacle which causes numerical stability problems. Fig. 12 gives the

Table 2. Calculation time of Initial Trajectory Creator

Time	mean $[ms]$	max $[ms]$	min $[ms]$	stddev $[ms]$
Simple	5.54	14.0	2.08	2.32
Cluttered	5.66	17.8	2.11	2.72

resulting trajectory in a complicated environment. This result verifies that CILQR with the initial trajectory can generate feasible trajectory in a cluttered environment. Also, we measure the calculation time in Table 3. The table shows that the average computational time is less than $10[ms]$, which suggests that the algorithm can be used in a real driving situation. In addition to these results, Fig. 13 shows the trajectories generated by Initial Trajectory Creator and CILQR, which are represented by pink and blue lines respectively. Note that although the trajectory generated by Initial Trajectory Creator is close to that generated by CILQR, the latter one has better state constraint satisfaction, which is the key point of using CILQR. Finally, we compare the calculation time with the calculation time of SQP. The code is written in C++ and use NLOPT to solve the nonlinear problem. We show the result in Table 4. You can find that extended CILQR is much faster than SQP.

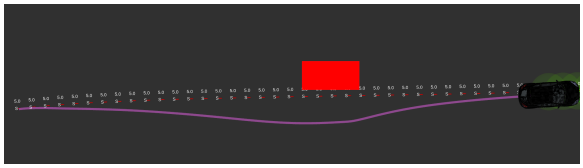

Figure 11. Initial Trajectory and CILQR with one obstacle

Figure 12. Initial Trajectory and CILQR with many obstacles

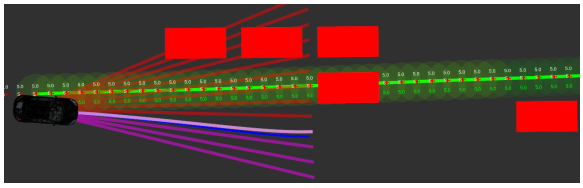

Figure 13. Initial Trajectory and CILQR Predicted Trajectory

Table 3. Calculation time of CILQR with Initial Trajectory Creator

Time	mean $[ms]$	max $[ms]$	min $[ms]$	stddev $[ms]$
Simple	16.5	43.7	6.06	7.01
Cluttered	29.8	65.0	9.06	13.9

Table 4. Comparison of calculation time between CILQR and NMPC

Algorithm	Horizon	Calculation Time $[ms]$
CILQR	70	29.8
SQP	30	213

7 Conclusion

In this paper, we modified the original CILQR and proposed a new algorithm. First, we designed a relaxed barrier function to solve the numerical problem. Second, we added an initial trajectory creator to provide high quality initial trajectories to help CILQR converge to a global trajectory faster. Simulations showed that our algorithm can solve some complicated problems well when the original CILQR algorithm cannot.

However, there are still some problems in the proposed approach. First, because of the relaxed barrier function, the resulted trajectory cannot be guaranteed to satisfy the hard constraint. Furthermore, we need to plan the vehicle's speed profile to deal with the dynamic obstacles . These problems will be addressed in the future works.

ACKNOWLEDGMENT

The authors want to thank Dr. Hiroyuki Okuda for his assistance on implementation of the base code of CILQR. In addition, Y.Shimizu was supported by JST CREST GrantNumber JPMJCR19F3,Japan.

REFERENCES

[1] J. Chen, W. Zhan and M. Tomizuka, "Autonomous Driving Motion Planning With Constrained Iterative LQR," in IEEE Transactions on Intelligent Vehicles, vol. 4, no. 2, pp. 244-254, June 2019.

[2] J. Chen, W. Zhan and M. Tomizuka, "Constrained iterative LQR for on-road autonomous driving motion planning," 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), Yokohama, 2017, pp. 1-7.

[3] L. Sun, C. Peng, W. Zhan, and M. Tomizuka, "A Fast Integrated Planning and Control Framework for Autonomous Driving via Imitation Learning," in ASME 2018 Dynamic Systems and Control Conference, Sep. 2018, pp. 1–11.

[4] L. Sun, W. Zhan, M. Tomizuka, and A. D. Dragan, "Courteous Autonomous Cars," in 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Oct. 2018, pp. 663–670.

[5] W. Zhan, J. Chen, C. Y. Chan, C. Liu, and M. Tomizuka, "Spatially-partitioned environmental representation and planning architecture for on-road autonomous driving," in 2017 IEEE Intelligent Vehicles Symposium (IV), Jun. 2017, pp. 632–639.

[6] Z. Li, W. Zhan, L. Sun, C.-Y. Chan, and M. Tomizuka, "Adaptive sampling-based motion planning with a non-conservatively defensive strategy for autonomous driving," in The 21st IFAC World Congress.

[7] D. González, J. Pérez, V. Milanés and F. Nashashibi, "A Review of Motion Planning Techniques for Automated Vehicles," in IEEE Transactions on Intelligent Transportation Systems, vol. 17, no. 4, pp. 1135-1145, April 2016.

[8] B. Paden, M. Čáp, S. Z. Yong, D. Yershov and E. Frazzoli, "A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles," in IEEE Transactions on Intelligent Vehicles, vol. 1, no. 1, pp. 33-55, March 2016.

[9] J. Ziegler, P. Bender, T. Dang and C. Stiller, "Trajectory planning for Bertha — A local, continuous method," 2014 IEEE Intelligent Vehicles Symposium Proceedings, Dearborn, MI, 2014, pp. 450-457.

[10] M. McNaughton, C. Urmson, J. M. Dolan and J. Lee, "Motion planning for autonomous driving with a conformal spatiotemporal lattice," 2011 IEEE International Conference on Robotics and Automation, Shanghai, 2011, pp. 4889-4895.

[11] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065), San Francisco, CA, USA, 2000, pp. 995-1001 vol.2.

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

[13] W. Moritz, Z. Julius, K. Sören and T. Sebastian. (2010). "Optimal Trajectory Generation for Dynamic Street Scenarios in a Frenet Frame." Proceedings - IEEE International Conference on Robotics and Automation. 987 - 993. 10.1109/ROBOT.2010.5509799.

[14] N. Wu, W. Huang, Z. Song, X. Wu, Q. Zhang and S. Yao, "Adaptive dynamic preview control for autonomous vehicle trajectory following with DDP based path planner," 2015 IEEE Intelligent Vehicles Symposium (IV), Seoul, 2015, pp. 1012-1017.

[15] M. Cibooglu, U. Karapinar and M. T. Söylemez, "Hybrid controller approach for an autonomous ground vehicle path tracking problem," 2017 25th Mediterranean Conference on Control and Automation (MED), Valletta, 2017, pp. 583-588.

[16] Y. Zhang et al., "Hybrid Trajectory Planning for Autonomous Driving in Highly Constrained Environments," in IEEE Access, vol. 6, pp. 32800-32819, 2018.

[17] L. Bai and S. Zhijiang. (2015). "Simultaneous dynamic optimization: A trajectory planning method for nonholonomic car-like robots," Advances in Engineering Software. 10.1016/j.advengsoft.2015.04.011.
[18] J. van den Berg, "Iterated LQR smoothing for locally-optimal feedback control of systems with non-linear dynamics and non-quadratic cost," 2014 American Control Conference, Portland, OR, 2014, pp. 1912-1918.
[19] D.Ferguson and A. Stentz, "Field d*: An interpolation-based path planner and replanner", in Robotics Research. Berlin, Germany: Srpinger-Verlag, 207, pp.239-253
[20] P. T. Boggs and J. W. Tolle, "Sequential quadratic programming", Acta Numer., vol. 4, pp. 1-51, 1995.
[21] D.H. Jacobson and D. Q. Mayne, "Differential Dynamic Programming," New York, NY, USA: Elsiver, 1970.
[22] W. Li and E. Todorov, "Iterative linear quadratic regulator design for non-linear biological movement systems," in Proc. 1stInt. Conf. Infrom. Control Autom. Robot, 2004, pp.222-229.
[23] E. Todorov and Weiwei Li, "A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems," Proceedings of the 2005, American Control Conference, 2005., Portland, OR, USA, 2005, pp. 300-306 vol. 1.
[24] Y. Tassa, N. Mansard and E. Todorov, "Control-limited differential dynamic programming," 2014 IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, 2014, pp. 1168-1175.
[25] D. H. Jacobson, "New second-order and first-order algorithms for determining optimal control: A differential dynamic programming approach," J. Optimi. thelory Appl., vol.2 no 6. pp. 411-440, 1968.
[26] D. Fassbender, B. C. Heinrich and H. Wuensche, "Motion planning for autonomous vehicles in highly constrained urban environments," 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Daejeon, 2016, pp. 4708-4713.
[27] Á. Domina and V. Tihanyi, "Comparison of path following controllers for autonomous vehicles," 2019 IEEE 17th World Symposium on Applied Machine Intelligence and Informatics (SAMI), Herlany, Slovakia, 2019, pp. 147-152.
[28] 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.
[29] J. Ziegler and C. Stiller, "Spatiotemporal state lattices for fast trajectory planning in dynamic on-road driving scenarios," 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, St. Louis, MO, 2009, pp. 1879-1884.
[30] X. Qian, F. Altché, P. Bender, C. Stiller and A. de La Fortelle, "Optimal trajectory planning for autonomous driving integrating logical constraints: An MIQP perspective," 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), Rio de Janeiro, 2016, pp. 205-210.
[31] C. Liu, C.-Y. Lin, and M. Tomizuka, "The convex feasible set algorithm for real time optimization in motion planning," SIAM J. Control Optim., vol. 56, no. 4 pp. 2712–2733, 2018.
[32] C. Liu, C. Lin, Y. Wang and M. Tomizuka, "Convex feasible set algorithm for constrained trajectory smoothing," 2017 American Control Conference (ACC), Seattle, WA, 2017, pp. 4177-4182.
[33] J. Chen, C. Liu and M. Tomizuka, "FOAD: Fast Optimization-based Autonomous Driving Motion Planner," 2018 Annual American Control Conference (ACC), Milwaukee, WI, 2018, pp. 4725-4732.
[34] C. Liu, W. Zhan and M. Tomizuka, "Speed profile planning in dynamic environments via temporal optimization," 2017 IEEE Intelligent Vehicles Symposium (IV), Los Angeles, CA, 2017, pp. 154-159.
[35] O. Arslan, K. Berntorp and P. Tsiotras, "Sampling-based algorithms for optimal motion planning using closed-loop prediction," 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 2017, pp. 4991-4996.
[36] C. Feller and C. Ebenbauer, "Relaxed Logarithmic Barrier Function Based Model Predictive Control of Linear Systems," in IEEE Transactions on Automatic Control, vol. 62, no. 3, pp. 1223-1238, March 2017.
[37] J. Hauser and A. Saccon, "A Barrier Function Method for the Optimization of Trajectory Functionals with Constraints," Proceedings of the 45th IEEE Conference on Decision and Control, San Diego, CA, 2006, pp. 864-869.
[38] Y. Meng, Y. Wu, Q. Gu and L. Liu, "A Decoupled Trajectory Planning Framework Based on the Integration of Lattice Searching and Convex Optimization," in IEEE Access, vol. 7, pp. 130530-130551, 2019.
[39] S. Kato, S. Tokunaga, Y. Maruyama, S. Maeda, M. Hirabayashi, Y. Kitsukawa, A. Monrroy, T. Ando, Y. Fujii, and T. Azumi,"Autoware on Board: Enabling Autonomous Vehicles with Embedded Systems," In Proceedings of the 9th ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS2018), pp. 287-296, 2018.
[40] S. Kato, E. Takeuchi, Y. Ishiguro, Y. Ninomiya, K. Takeda, and T. Hamada. "An Open Approach to Autonomous Vehicles," IEEE Micro, Vol. 35, No. 6, pp. 60-69, 2015.