Constrained Iterative LQR for On-Road Autonomous Driving Motion Planning

Jianyu Chen, Wei Zhan and Masayoshi Tomizuka

***Abstract*—There exist a lot of challenges in trajectory planning for autonomous driving: 1) Needs of both spatial and temporal planning for highly dynamic environments; 2) Nonlinear vehicle models and non-convex collision avoidance constraints. 3) High computational efficiency for real-time implementation. Iterative Linear Quadratic Regulator (ILQR) is an algorithm which solves predictive optimal control problem with nonlinear system very efficiently. However, it can not deal with constraints. In this paper, the Constrained Iterative LQR (CILQR) is proposed to handle the constraints in ILQR. Then an on road driving problem is formulated. Simulation case studies show the capability of the CILQR algorithm to solve the on road driving motion planning problem.**

I. Introduction

Motion Planning is one of the most important modules in autonomous driving. A typical motion planner for autonomous vehicles takes into account the information from the environment (e.g. motions of the host vehicle and the surrounding vehicles), and plans a trajectory for the vehicle to execute. Although motion planning has been investigated for decades, it is still challenging to design a motion planner for autonomous vehicles satisfying the following requirements simultaneously: (1) The computation needs to be in real time to interact with the dynamic environment and deal with emergencies. (2) The planning needs to be spatiotemporal to deal with dynamic obstacles. (3) The planner needs to be able to deal with complex constraints such as collision avoidance.

The autonomous driving motion planning techniques can be roughly divided into three categories: Search-based, sampling-based and optimization-based methods [1], [2]. Search-based methods often refer to A* [3] search or D* [4] search. Their planning results are globally optimal, but the algorithms are inefficient when dealing with moving obstacles and are difficult to do spatiotemporal planning. They are also not suitable for on-road driving motion planning because it is hard to set a goal and find a good heuristic function, which greatly impacts the computation time of A* search and therefore its capability to be real-time. The sampling-based methods are mainly represented by Rapidly-exploring Random Tree (RRT) [5]. The main disadvantage of RRT is its inefficiency when doing collision check. Besides, its planning results are suboptimal. Both the search-based and sampling-based methods are planned in a discretized space (either in state space or in action space), which leads to non-smooth trajectories. And the resolution of the discretized space will largely influence the computational efficiency due to the "curse of dimensionality".

The optimization-based method formulates motion planning as a mathematical optimization problem. The planning results are continuous, optimal, and spatiotemporal. However, since the vehicle kinematics model is nonlinear, and the collision avoidance constraints are non-convex, the problem contains nonlinear equality constraints and non-convex inequality constraints, making it very difficult and inefficient to solve the optimization problem. The state-of-the-art work at this field is done by Mercedes-Benz [8], which utilized Sequential Quadratic Programming (SQP) to solve the nonlinear and non-convex problem, but the computation time is around 0.5s which should be shortened. Some works decouple the spatial and temporal planning [6], [7], which are computational efficient. However, the temporal planning are only locally allowed based on the spatial planning, which is not real spatiotemporal.

Iterative Linear Quadratic Regulator(ILQR) [9], [10] is an algorithm based on Linear Quadratic Regulator(LQR) that can solve the unconstrained optimal control problems subject to nonlinear system dynamics. Its most important advantage is time efficiency, which makes it a potential method to solve the real time issue in complicated optimization problems. However, very few literatures consider constraints in ILQR, while constraints are inevitable in autonomous driving motion planning. The control-limited differential dynamic programming (DDP) was proposed in [11], which solves the nonlinear optimal control problem under constraints, but the form of constraint is limited to a lower and upper bound of the control input. Extended LQR [12], [13] achieves collision avoidance, but it only deals with circle obstacles by penalizing the distance from the center of the host robot to the center of the obstacle. In autonomous driving, however, the shape of obstacles can be more complex. To the best of our knowledge, no work has studied how to handle a general form of constraints using ILQR. Therefore, no one has used ILQR to deal with the general on-road autonomous driving motion planning with complex constraints.

In this paper, the Constrained Iterative LQR (CILQR) is proposed to handle the general form of constraints for ILQR. LQR is solved by Dynamic Programming (DP) which is very

*J.Chen, W. Zhan and M. Tomizuka are with the Department of Mechanical Engineering, University of California, Berkeley, CA 94720 USA (e-mail: jianyuchen@berkeley.edu, wzhan@berkeley.edu, tomizuka@berkeley.edu).

978-1-5386-1526-3/17/$31.00 ©2017 IEEE

efficient, but it can only deal with unconstrained problems. To take advantage of the efficiency of DP, the constraints are transformed to the cost function by adding a barrier function. Nonlinear and non-convex constraints will be linearized. The barrier function is then quadratized and added to the original cost function. Then the CILQR algorithm is applied to a general on road autonomous driving motion planning problem, where a new general representation of the obstacles is developed using ellipse. The centerline is approximated using polynomial, enabling a continuously differentiable distance function for better convergence.

The remainder of the paper is organized as follows. Section II introduces the Constrained Iterative LQR algorithm. Section III shows how the algorithm is applied to a general on road autonomous driving motion planning problem. Section IV gives simulation results of some typical scenarios. Finally, section V concludes the paper.

II. Constrained Iterative LQR

A. Problem Statement

Problem 1 (A General Motion Planning Problem). A constrained receding horizon motion planning problem can be formulated as:

$$\min_{\substack{u_0,\dots,u_{N-1}\\ x_0,\dots,x_N}} J = c_N(x_N) + \sum_{k=0}^{N-1} c_k^u(x_k) + c_k^u(u_k)$$

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

$$x_0 = x^{start} \tag{1b}$$

$$f_k^x(x_k) \le 0, \quad k = 0, 1, ..., N \tag{1c}$$

$$f_k^u(u_k) \le 0, \quad k = 0, 1, ..., N-1 \tag{1d}$$

where x_k is the state, u_k is the control input, and the lower subscript represents the time step. N is the preview horizon. J is the cost function, where c_N is the final stage cost and c_k is the cost at time step k. (1a) is the system dynamic equation which can be nonlinear, (1b) is the initial state, (1c) and (1d) are the state and control constraints, they can be nonlinear and non-convex. Here we assume that the constraint function is decoupled for x and u at each time step.

This general formulation covers a large class of problems in motion planning. One of the state-of-the-art techniques that can solve the above nonlinear and non-convex optimization problem is SQP, which divides the original problem into a sequence of Quadratic Programming (QP) and solve it iteratively. However, SQP is designed for a more general purpose that can solve generic optimizations, which makes it not efficient enough for real time motion planning. For the motion planning problem shown above, with the system dynamics as the only equality constraints, the special structure can be utilized to improve the computation efficiency using ILQR.

B. Iterative LQR (ILQR)

Problem 2 (The Standard ILQR Problem) The Iterative LQR algorithm can deal with the following nonlinear system motion planning problem

$$\min_{\substack{u_0,\dots,u_{N-1}\\ x_0,\dots,x_N}} J = \tfrac{1}{2}x_N^T Q_N x_N + x_N^T p_N + q_N + \sum_{k=0}^{N-1} \left(\tfrac{1}{2}x_k^T Q x_k + x_k^T p + \tfrac{1}{2}u_k^T R u_k + u_k^T r + q\right) \tag{2}$$

$$s.t. \quad x_{k+1} = f(x_k, u_k), \quad k = 0, 1, ..., N-1 \tag{3a}$$

$$x_0 = x^{start} \tag{3b}$$

where Q_N, p_N, q_N, Q, p, R, r, q are parameters describing the quadratic cost function. The dynamics equation (3a) can be nonlinear. Start with an initial control input sequence, such as $u_k = 0$, the algorithm iterates with two steps:

Step 1 (Forward Simulation). The control sequence at the current iteration u_k is passed to the real system dynamics (3a) and get the state sequence x_k for this iteration.

Step 2 (Backward Optimization). First the nonlinear system (3a) is linearized around x_k and u_k:

$$\delta x_{k+1} = A_k \delta x_k + B_k \delta u_k \tag{4}$$

where $A_k = D_x f(x_k, u_k)$ and $B_k = D_u f(x_k, u_k)$ are the first derivatives of the original dynamic function. Now under this iteration and with the current linearized system, the problem becomes a standard LQR problem:

Problem 3 (The Standard LQR Problem)

$$\min_{\substack{u_0,\dots,u_{N-1}\\ x_0,\dots,x_N}} J = \tfrac{1}{2}\delta x_N^T Q_N \delta x_N + \delta x_N^T p_N + q_N + \sum_{k=0}^{N-1} \left(\tfrac{1}{2}\delta x_k^T Q_k \delta x_k + \delta x_k^T p_k + \tfrac{1}{2}\delta u_k^T R_k u_k + \delta u_k^T r_k + q_k\right) \tag{5}$$

$$s.t. \quad \delta x_{k+1} = A_k \delta x_k + B_k \delta u_k, \quad k = 0, 1, ..., N-1 \tag{6a}$$

$$x_0 = x^{start} \tag{6b}$$

This problem can be solved using the standard LQR algorithm. Note that we use the differential form of the state and control, δx_k and δu_k, in order to stay consistent with the symbols in ILQR. They should be considered as normal state and control when solving LQR.

The optimal solution δu_k will be used to update the current control sequence: $u_k \leftarrow u_k + \delta u_k$.

Details of ILQR algorithm can be found in [9], [10]. As mentioned in section I, ILQR is a very efficient algorithm to solve motion planning problems with nonlinear dynamic constraint. However, a big drawback of ILQR is that it cannot deal with complex constraints, which is essential in most robotic motion planning scenarios. Taking autonomous driving for example, there are various kinds of constraints for collision avoidance, acceleration, and traffic rules such

as speed limit. Without the ability to deal with constraints, the power of ILQR is significantly limited.

C. Constrained Iterative LQR (CILQR)

The CILQR algorithm aims to solve Problem 1 by extending the ILQR algorithm. There are two additional issues to extend from Problem 2 to Problem 1. First, there are state and control constraints in Problem 1. Second, the cost function in Problem 1 is in a general form, while in Problem 2 the cost function is quadratic. CILQR uses the same framework of ILQR, but at the backward optimization step, it transforms the constraints and costs into quadratic cost terms, so that the problem is in the same form as Problem 3 and the standard solution applies. The following 4 steps describe the transformation process based on the current nominal state and control x_k, u_k. Since x_k and u_k are symmetric in our problem, we will only discuss c_k^x and f_k^x, and then the same method applies to c_k^u and f_k^u.

step 1 (Cost Function Quadratization). If the cost function c_k^x is not in quadratic form, then it needs to be quadratized around x_k. The quadratization process is to compute the second-order Taylor series approximation of the function:

$$\begin{aligned} c_k^x\left(x_k+\delta x_k\right) \approx & \left(\delta x_k\right)^T \nabla^2 c_k^x\left(x_k\right) \delta x_k \\ & +\left(\delta x_k\right)^T \nabla c_k^x\left(x_k\right)+c_k^x\left(x_k\right) \end{aligned} \tag{7}$$

where $\nabla c_k^x\left(x_k\right)$ and $\nabla^2 c_k^x\left(x_k\right)$ are the first and second order derivatives of c_k^x on x_k

step 2 (Constraint Function Linearization). If the constraint $f_k^x \leq 0$ is not linear, it needs to be linearized. The linearization process is to replace f_k^x with its first order Taylor series approximation:

$$f_k^x\left(x_k+\delta x_k\right) \approx \left(\delta x_k\right)^T \nabla c_k^x\left(x_k\right)+c_k^x\left(x_k\right) \tag{8}$$

where $\nabla c_k^x\left(x_k\right)$ is the first order derivative of f_k^x on x_k.

step 3 (Barrier Function Shaping). The linearized constraint function f_k^x will be shaped by a barrier function:

$$b_k^x=q_1 \exp\left(q_2 f_k^x\right) \tag{9}$$

where q_1 and q_2 are parameters to tune the shape of the barrier function. The use of the barrier function is to approximate the indicator function, which transforms the constrained problem into an unconstrained problem. The exponential barrier function (9) is twice differentiable and its derivatives are easy to compute.

step 4 (Constraint Function Quadratization). The shaped constraint function will be quadratized:

$$b_k^x\left(x_k+\delta x_k\right) \approx \delta x_k^T \nabla^2 b_k^x\left(x_k\right) \delta x_k+\delta x_k^T \nabla b_k^x\left(x_k\right)+b_k^x\left(x_k\right) \tag{10}$$

By applying the chain rule, we have:

$$\nabla b_k^x\left(x_k\right)=q_1 q_2 \exp\left(q_2 c_k^x\left(x_k\right)\right) \nabla c_k^x\left(x_k\right) \tag{11}$$

and

$$\begin{aligned} & \nabla^2 b_k^x\left(x_k\right)= \\ & \quad q_1 q_2^2 \exp\left(q_2 c_k^x\left(x_k\right)\right) \nabla c_k^x\left(x_k\right)\left(\nabla c_k^x\left(x_k\right)\right)^T \end{aligned} \tag{12}$$

Finally, the resulting (7), (10) and their counterparts for δu_k are added to the cost function (5) of Problem 3.

Finite difference is a popular method to compute the derivatives. However, we suggest analytically deduce the derivative and hessian because the time efficiency of finite difference is low. Actually, one of the reasons why we use this 4-step scheme is to make it easy to get the analytical derivative functions.

III. On Road Autonomous Driving Motion Planning using Constrained Iterative LQR

A. Problem Statement

The on road autonomous driving motion planning problem can be formulated as following:

Problem 4 (On Road Autonomous Driving Motion Planning Problem)

$$\begin{aligned} & \min_{\substack{u_0,\ldots,u_{N-1} \\ x_0,\ldots,x_N}} J=\tfrac{1}{2} x_N^T Q_N x_N+x_N^T p_N+q_N+ \\ & \sum_{k=0}^{N-1}\left\{\tfrac{1}{2} x_k^T Q x_k+x_k^T p_k+q_k+\tfrac{1}{2} u_k^T R u_k+u_k^T r_k\right\} \end{aligned}$$

$$s.t. \quad x_{k+1}=f\left(x_k, u_k\right) \tag{13a}$$

$$x_0=x^{start} \tag{13b}$$

$$x_k \in C / \cup O_j\left(k\right), \quad k=1, ..., N \tag{13c}$$

$$u_k \in \Omega, \quad k=0, 1, ..., N-1 \tag{13d}$$

where (13a) is the system dynamic equation, which is nonlinear. The collision avoidance constraints are the complimentary set of the space occupied by the obstacles, denoted as $x_k \in C / \cup O_j\left(k\right)$, where C is the whole space, $O_j\left(k\right)$ is the space occupied by obstacle j at time step k. Since $O_j\left(k\right)$ is usually convex, (13c) is usually a non-convex inequality constraint.

Problem 4 is a subproblem of problem 1. The CILQR algorithm is applied to solve it. The solution will be a sequence of optimal control commands $u_0, u_1, ..., u_{N-1}$, and a planned trajectory $x_0, x_1, ..., x_N$.

B. Vehicle Model

The vehicle kinematic model shown in Fig. 1 is used in this paper. The state is $x=\left[\begin{array}{cccc} p^x & p^y & v & \theta \end{array}\right]^T$, where p^x and p^y represent the two-dimensional position of the vehicle, v is the vehicle velocity and θ represents the yaw angle. The vehicle dynamics is:

$$\dot{x} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{v} \\ \dot{\theta} \end{bmatrix} = g(x,u) \quad (14)$$

Note that there are nonlinear terms in the dynamic equations and that $\dot{\theta} = \frac{v}{L}\tan\delta$ where δ is the steering angle of the front wheels.

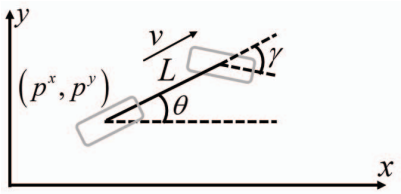

Fig. 1: Vehicle Kinematic Model

Since we are discussing in the discrete-time domain, this model needs to be transformed to discrete-time form. Once x_k and u_k is determined, we have for $t \in [T_s k, T_s(k+1)]$, $x(T_s k) = x_k = \begin{bmatrix} p_k^x & p_k^y & v_k & \theta_k \end{bmatrix}^T, u_k = \begin{bmatrix} \dot{v}_k & \dot{\theta}_k \end{bmatrix}^T$. So $\dot{x}(t) = \begin{bmatrix} v(t)\cos\theta(t) & v(t)\sin\theta(t) & \dot{v}_k & \dot{\theta}_k \end{bmatrix}^T$, where $v(t) = v_k + \dot{v}_k(t - T_s k)$ and $\theta(t) = \theta_k + \dot{\theta}_k(t - T_s k)$. Substituting $v(t)$ and $\theta(t)$ we get:

$$\begin{aligned} \dot{p}^x(t) &= (v_k + \dot{v}_k(t - T_s k))\cos(\theta_k + \theta_k(t - T_s k)) \\ \dot{p}^y(t) &= (v_k + \dot{v}_k(t - T_s k))\sin(\theta_k + \theta_k(t - T_s k)) \end{aligned} \quad (15)$$

After integrating we have:

$$\begin{aligned} p^x(t) &= p_k^x + \int_{T_s k}^{t} \dot{p}^x(t)\, d\tau \\ p^y(t) &= p_k^y + \int_{T_s k}^{t} \dot{p}^y(t)\, d\tau \end{aligned} \quad (16)$$

So the states in the next time step are $p_{k+1}^x = p^x((k+1)T_s)$ and $p_{k+1}^y = p^y((k+1)T_s)$ when $\dot{\theta}_k \neq 0$. For $\dot{\theta}_k = 0$ we have:

$$\begin{aligned} p_{k+1}^x &= p_k^x + \cos\theta_k \left(v_k T_s + \tfrac{\dot{v}_k}{2} T_s^2\right) \\ p_{k+1}^y &= p_k^y + \sin\theta_k \left(v_k T_s + \tfrac{\dot{v}_k}{2} T_s^2\right) \end{aligned} \quad (17)$$

and:

$$\begin{aligned} v_{k+1} &= v_k + \dot{v}_k T_s \\ \theta_{k+1} &= \theta_k + \dot{\theta}_k T_s \end{aligned} \quad (18)$$

(17) and (18) can be packaged to the discrete system dynamic model:

$$x_{k+1} = \begin{bmatrix} p_{k+1}^x & p_{k+1}^y & v_{k+1} & \theta_{k+1} \end{bmatrix}^T = f(x_k, u_k) \quad (19)$$

C. Objective Function

The objective function can be decoupled into four terms:

$$J = \sum_{k=0}^{N} c_k^{acc} + c_k^{yawr} + c_k^{off} + c_k^{vel} \quad (20)$$

where the four terms represent penalties for acceleration, yaw rate, offset from reference and velocity difference. We will describe below the details of how they are computed:

1) acceleration: The term:

$$c_k^{acc} = w_{acc} u_k^T \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} u_k \quad (21)$$

is the penalty for acceleration, which is highly related to passenger comfort. A lager acceleration than $1m/s^2$ will cause significant uncomfort [14]. Adding this term to the objective function can improve passenger comfort and tuning the weight w_{acc} can change the level of comfort.

2) Yaw Rate: The term:

$$c_k^{yawr} = w_{yawr} u_k^T \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} u_k \quad (22)$$

is the penalty for yaw rate. It penalizes the rapid changes in vehicle direction and the lateral acceleration which is related to driving safety and comfort. Tuning the weight w_{yawr} can change how much we concern the above issues.

3) Offset to Reference: The term c_k^{off} penalizes the distance to the reference. It is worth thinking about how we define the reference and the distance. A widely used method is to give a sequence of reference points $x^r = [x_0^r, x_1^r, ..., x_N^r]$ where the lower subscript represents the time stamp. The term can be then formulated as

$$c_k^{off} = (x_k - x_k^r)^T Q_k^r (x_k - x_k^r) \quad (23)$$

However, this formulation may result in some problematic scenarios. For example, as shown in Fig. 2, the vehicle is on a curve road and it needs to stop before the line A. The reference x_i^r are points on the road centerline. If we set the cost as (23), the planned trajectory will be similar to x_i in the figure. Note x_i, x_{i+1} and x_{i+2} significantly deviate from the centerline because they are "pulled" towards the corresponding reference points x_i^r, x_{i+1}^r and x_{i+2}^r.

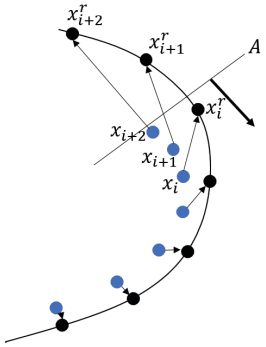

Fig. 2: Problematic Scenario on Curve Road

Besides the problematic scenarios, (23) requires us to build a powerful upper layer planner to give an adequate reference. The ideal cost is to penalize the distance from a specific point in the planned trajectory to the whole reference line, instead of a reference point. In this case we do not need to provide a sequence of reference points, but just a simple reference line such as the centerline of the road. Benz [8] tried to solve this problem by creating a distance field. However, the field is not continuously differentiable, introducing risk of nonconvergence.

To address the above problems, we use polynomial to represnet the reference line and penalize the distance from the points in planned trajectory to the polynomial. As shown in Fig. 3, a 3-order polynomial S is used as the reference trajectory, and x_k is one point in the planned trajectory. To find the distance from x_k to the curve S, we need to find a point s_k on S such that $\overline{x_k s_k} \perp l_k$, where l_k is the tangent line passing through s_k. Then the distance from x_k to curve S is $d(x_k, S) = \overline{x_k s_k}$. Its first order approximation is $d(x_k + \delta x_k, S) \approx d(x_k + \delta x_k, l_k)$, and its square forms a quadratic term, which will be set as c_k^{off}.

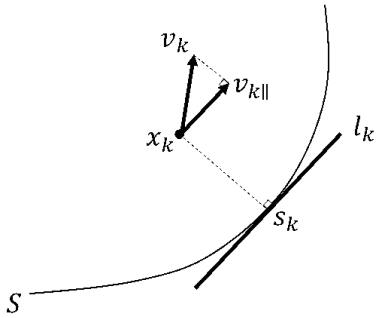

Fig. 3: The Distance Function To The Polynomial Curve

There are two main advantages to use this method. First, the polynomial is continuously differentiable and its tangent line is easy to compute. Second, in practice the commercial sensors often fit the lanes as polynomials, e.g, mobileye fits the lanes as 3rd order polynomials.

4) Velocity Difference: c_k^{vel} is the cost that helps us maintain a relatively high speed to achieve efficiency. As shown in Fig. 3, we set $c_k^{vel} = w_{vel}\left(\left\|v_{k\|}\right\| - v_{des}\right)^2$, where $v_{k\|}$ is the velocity along the direction of the tangent l_k, and v_{des} is the desired speed which is a scalar.

D. Constraints

Besides some objectives to optimize, there are also some constraints to obey when driving. Here four kinds of constraints are considered: the acceleration constraint, yaw rate constraint, boundary constraint and obstacle avoidance constraint.

1) Acceleration Constraint: The acceleration needs to be bounded according to the limit of engine force and braking force. This term is:

$$a_{low} \leq u_k^T \begin{bmatrix} 1 \\ 0 \end{bmatrix} \leq a_{high} \tag{24}$$

where $a_{high} > 0$ is the highest acceleration the engine can provide, and $a_{low} < 0$ is the largest deceleration the brake can provide.

2) Yaw Rate Constraint: The yaw rate needs to be bounded according to the steering wheel limit:

$$-\bar{s} \leq u_k^T \begin{bmatrix} 0 \\ 1 \end{bmatrix} \leq \bar{s} \tag{25}$$

where $\bar{s}$ is the largest yaw rate, which can be computed using the largest steering wheel angle with $\dot{\theta} = \frac{v}{L}\tan\delta$.

3) Boundary Constraint: The boundary constraint considers boundary on the longitudinal direction, which can be regarded as a linear constraint that is vertical to the road direction. Lots of scenarios contain boundary constraints. As shown in Fig. 4, the stop sign scenario can be represented as a fixed boundary on the crossing road, and the car following scenario can be represented as a moving boundary behind the front vehicle.

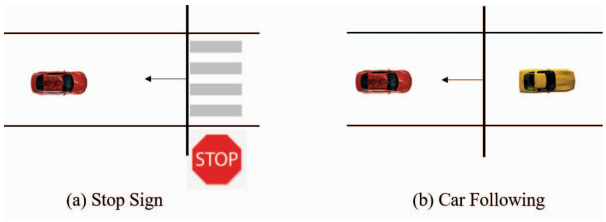

Fig. 4: The Boundary Constraints

The above constraints are linear and can follow from step 3 in the CILQR algorithm.

4) Obstacle Avoidance Constraint: The obstacle avoidance constraint considers the moving and static obstacles on the road. For example, how to overtake the front vehicle safely, and how to avoid static obstacles such as parking vehicles on the roadside. It is difficult to deal with because the constraints are non-convex, as the feasible state space is the space outside the convex obstacle (such as a rectangle representing a car). Here we ignore the vertical axis and only consider the planar problem.

In this paper, we model all the obstacles as ellipse, as shown in Fig. 5. The long-axis and short-axis of the ellipse can be set according to the obstacle's shape (e.g. the vehicle's length and width) and it's velocity. For example, for a car, the long-axis of the ellipse may be set as $a = l + vt_{safe} + s_{safe}$, and the short-axis be $b = w + s_{safe}$, where l is the vehicle length, w is the vehicle width, v is the vehicle velocity, t_{safe} is the safe headway and s_{safe} is a safety margin. Then combining with the yaw angle of the vehicle, we can write the ellipse analytically:

$$(x - x_O)^T P (x - x_O) = 1 \tag{26}$$

where x_O is the center of the ellipse.

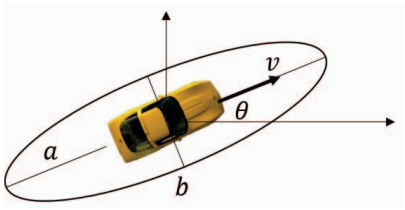

Fig. 5: The Ellipse Bound of The Obstacles

The safety constraint according to the ellipse (26) is $(x - x_0)^T P (x - x_0) > 1$. Let x_k be the point we would like to check its safety, and set the safety margin to be r. Then the safe boundary will be the trajectory of the center of a circle with radius r, moving around the boundary of the ellipse (26). As shown in Fig. 6, the red trajectory represents for the safe boundary. This constraint is very difficult to be represented analytically, thus we build another ellipse to approximate it. This ellipse has long-axis with $a + r$ and short axis with $b + r$, which is represented by $(x - x_0)^T P' (x - x_0) = 1$. The bigger black ellipse in Fig. 6 is the approximate ellipse, which is a little smaller than the true safe boundary, but their difference can be ignored.

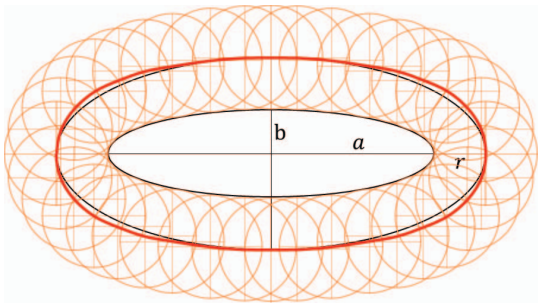

Fig. 6: The Approximation Ellipse of the Safety Boundary

Now consider the new ellipse and the point x_k we want to check, we have the following constraint:

$$f_k^x (x_k) = 1 - (x_k - x_0)^T P' (x_k - x_0) < 0 \qquad (27)$$

This is a non-convex constraint function and needs to follow the step 2 of CILQR algorithm.

Since the vehicle cannot be seen as a simple point, here we use two circles to represent the car, as shown in Fig. 7. One circle is at the center of the rear-axis, another one is at the symmetric position in the front of the vehicle. When checking the safety, both the two centers of the circles need to be checked with the safety margin being their radius. When we consider the safety of the front center, we need to note that its position is a nonlinear function of the state: $p_{front} = \begin{bmatrix} p^x + \cos\theta\Delta l & p^y + \sin\theta\Delta l \end{bmatrix}$ and a linearization process is needed.

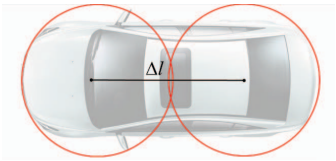

Fig. 7: Circle Representation of the Vehicle Shape

There are several reasons why we choose ellipse to model the obstacle constraints. First, ellipse can represent the moving obstacles better because of its adjustable long-axis and short-axis. Some works use circles to represent the obstacles, which is not appropriate in on-road driving scenarios as the longitudinal motion usually dominates the lateral motion. Second, ellipses are smooth. One may think about using rectangles to better approximate the shape of vehicles, but the rectangle will cause non-continuous derivatives at its corners. Third, as Gaussian is equal to ellipse, when we consider stochastic motion of surrounding vehicles, it is easily extended from the current ellipse representation.

IV. Case Studies

Three on road driving scenarios are studied to show the capability of CILQR. The simulation environment is shown in Fig. 8. The host vehicle is represented by the blue box and the surrounding vehicles are represented by red boxes. The yellow ellipse around each surrounding vehicle represents the safety margin for collision avoidance. The blue line represents the reference trajectory that the host vehicle should track. The time step information is represented by the depth of color: the deeper color represents the later time step. And in the following cases, the speed of surrounding vehicle is assumed to be constant in the preview horizon.

The sampling time of the simulation is $T_s = 0.25s$, and the preview horizon is $N = 40$. Thus the preview time is $T = 10s$, which is enough to do a long-term planning. The simulation is written in Matlab and run on a laptop with 2.6GHz Intel Core i7-6600U. Note that in the following cases, the computation can finish within 0.2 second, which is almost real-time. When the algorithm is implemented in a more efficient programming language such as C++, the computation is expected to be at least 10 times faster and the real-time requirement can be easily satisfied.

A. Case 1: Static Obstacle Avoidance

In this case there are several static obstacles along the road which are represented by ellipses. The host vehicle needs to pass by these obstacles safely. In Fig. 8, the planned trajectory, which is represented by a sequence of blue boxes with the increasing depth of color, shows that the host vehicle avoids the obstacles smoothly. The runtime of this case is 0.12s.

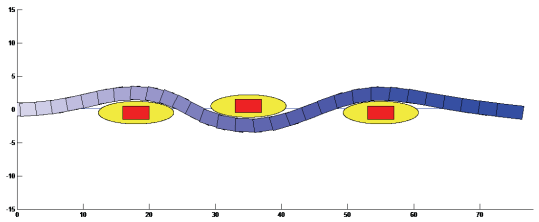

Fig. 8: Static Obstacle Avoidance

B. Case 2: Lane Change and Car Following

Lane changing is one of the most common maneuver in both urban driving and highway driving. Usually there will be a front vehicle in the target lane and the vehicle needs to do adaptive cruise control (ACC) after finishing the lane change. One dangerous situation when executing lane changing is when there exists a vehicle besides the host vehicle in the target lane. We created this dangerous scenario. As shown in Fig. 9, O_1^0 is the first obstacle at $t = 0$, O_1^T is the first obstacle at $t = T$, and O_2^0 is the second obstacle at $t = 0$, O_2^T is the second obstacle at $t = T$.

In the planned trajectory of the host vehicle, the vehicle first accelerates and passes around the first obstacle to change the lane. After finishing lane changing, it decelerates to do ACC to its front vehicle. The runtime of this case is 0.19s.

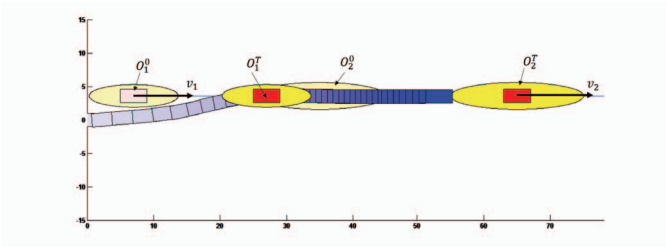

Fig. 9: The Lane Change Scenario

C. Case 3: A Mixed Scenario

In this scenario, the goal of the host vehicle is to overtake the slow front vehicle O_1 while avoiding the coming vehicle O_2, and then stop at position $x = 50m$. As usual, the subscript represents the index of the obstacle and the superscript represents the time. Fig. 10 shows the planned trajectory of the host vehicle. From the trajectory, we can see that the host vehicle first accelerates to overtake the front vehicle before encountering the coming vehicle, and then decelerates to stop at the stop line. The runtime of this case is 0.2s.

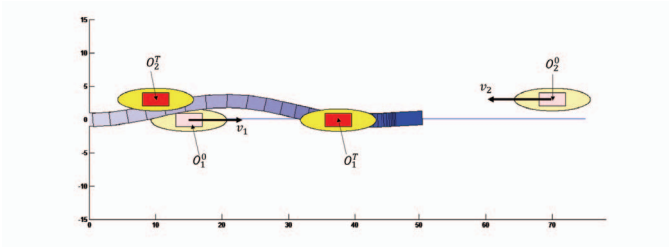

Fig. 10: A Mixed Scenario

V. Conclusion

In this paper, the constrained ILQR algorithm is proposed to handle the constraints for ILQR. The constraints are linearized, shaped by barrier function, and then quadratized to be placed into the cost function of ILQR. A general on road driving motion planning problem is then solved by CILQR algorithm, where we considered the distance to the polynomial reference line and the ellipse representation of obstacles. Case studies show our algorithm works properly in several on road driving scenarios.

References

[1] Gonzlez, David, et al. *A review of motion planning techniques for automated vehicles*. IEEE Transactions on Intelligent Transportation Systems 17.4 (2016): 1135-1145.

[2] Paden, Brian, et al. *A Survey of Motion Planning and Control Techniques for Self-driving Urban Vehicles*. IEEE Transactions on Intelligent Vehicles 1.1 (2016): 33-55.

[3] Dolgov, Dmitri, et al. *Practical search techniques in path planning for autonomous driving*. Ann Arbor 1001 (2008): 48105.

[4] Ferguson, Dave, and Anthony Stentz. *Field D*: An interpolation-based path planner and replanner*. Robotics research. Springer Berlin Heidelberg, 2007. 239-253.

[5] Kuffner, James J., and Steven M. LaValle. *RRT-connect: An efficient approach to single-query path planning*. Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on. Vol. 2. IEEE, 2000.

[6] Gu, Tianyu, et al. *Tunable and stable real-time trajectory planning for urban autonomous driving*. Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on. IEEE, 2015.

[7] Gu, Tianyu, John M. Dolan, and Jin-Woo Lee. *Runtime-bounded tunable motion planning for autonomous driving*. Intelligent Vehicles Symposium (IV), 2016 IEEE. IEEE, 2016.

[8] Ziegler, Julius, et al. *Trajectory planning for Bertha - A local, continuous method*. Intelligent Vehicles Symposium Proceedings, 2014 IEEE.

[9] Li, Weiwei, and Emanuel Todorov. *Iterative linear quadratic regulator design for nonlinear biological movement systems*. ICINCO (1). 2004.

[10] Todorov, Emanuel, 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.. IEEE, 2005.

[11] Tassa, Yuval, Nicolas Mansard, and Emo Todorov. *Control-limited differential dynamic programming*. 2014 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2014.

[12] van den Berg, Jur. *Extended LQR: locally-optimal feedback control for systems with non-linear dynamics and non-quadratic cost*. Robotics Research. Springer International Publishing, 2016. 39-56.

[13] van den Berg, Jur. *Iterated LQR smoothing for locally-optimal feedback control of systems with non-linear dynamics and non-quadratic cost*. 2014 American Control Conference. IEEE, 2014.

[14] Gonzlez, David, et al. *Speed profile generation based on quintic Bzier curves for enhanced passenger comfort*. Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on. IEEE, 2016.