参赛队员姓名：时沐朗

中学：浙江省杭州第九中学

省份：浙江省

国家/地区：中国

指导教师姓名：杨涛

指导教师单位：浙江省杭州第九中学

论文题目：Kinematics-Based Vehicle Trajectory Optimization for Obstacle Avoidance and Goal Satisfaction

Kinematics-Based Vehicle Trajectory Optimization for Obstacle Avoidance and Goal Satisfaction

Mulang Shi

Abstract

The electric bicycle is one of the most important means of transportation in China, and it has a huge number of electric bicycle users. However, there are a lot of deaths caused by electric bicycle traffic accidents every year. In this paper, we design a driver-assisted electric bicycle and motion planning based on the iLQR algorithm that can avoid obstacles. The study provides a solution for the future generation of electric bicycles that can respond to emergencies and reduce accidents to improve travel safety.

Key Words: Trajectory Optimization, Motion Planning, Autunomous Driving

Contents

1 Introduction

The electric bicycle is one of the most frequently used transportation methods in China. In urban areas, people commonly use the electric bicycle to commute to work or travel short distances from home to metro and bus stations. According to China Statistical Yearbook [1], the number of electric bicycles produced in China reached 29.7 million in 2020, and the whole country has a total number of 300 million electric bicycles in use. However, due to the smaller size and fewer safety protections compared to other on-road vehicles, traffic accidents are more likely to cause severe safety issues for electric bicycles riders. An overwhelming amount of accidents related to electric bicycles have been recorded. In 2019, traffic police in Zhejiang Province handled over 6.1 million cases of non-motorized vehicle violations on the roads. Moreover, in the first quarter of 2020, traffic accidents involving electric bicycles in Zhejiang province accounted for 29.1% of deaths due to traffic accidents involving electric bicycles. As a common means of transportation, it is important to ensure the safety of electric bicycle driving. This project aims at developing safe and reliable motion planning algorithms for autonomous electric bicycle driving. The algorithm can be applied to a variety of autonomous driving or assisted driving systems to improve travel safety and reduce accidents.

A reliable and efficient motion planning algorithm is the core for autonomous driving, which can solve the problem of finding feasible moving trajectories by considering vehicle kinematic models while simultaneously allowing the vehicle to safely traverse around obstacles from the initial state to the target state. A typical planning module receives the dynamic external environment structures including obstacles, lanes, pedestrians, traffic lights, and other objects. Then it generates a trajectory with the ideal driving quality under desired conditions that satisfy safety and feasibility constraints. Researchers have studied motion planning for several decades, but designing an ideal trajectory for autonomous vehicles can still be a challenging task. The outcome trajectory should simultaneously meet the following requirements:

- Achieve real-time computing and interaction with the dynamic environment, as well as rapid response to emergencies.
- Have the ability to handle complex conditional constraints and vehicle kinematic models.
- Generate plans in a spatiotemporal domain to handling static and dynamic obstacles.

Current motion planning algorithms are traditionally grouped into three types: graph-search-based methods, sampling-based methods, and optimization-based methods [2]. The graph-search-based methods usually refer to A* [3] search, Dijkstra [4] search, and D* [5] search. In graph-search-based planners, the main idea is to search and traverse in the state space, beginning with the starting point A and ending with the target point B. The algorithms of graph-search based for accessing the different states in the state space provide a globally optimal solution, but have high requirements for memory and computational speed, and are inefficient in dealing with dynamic obstacles. The sampling-based methods are typically used the Rapid-Exploring Random Tree (RRT) [6] and the Probabilistic Roadmap

Method (PRM) [7]. This method can search the high-dimensional space quickly and efficiently, and find a planning path from the starting point to the target point by directing the search to a blank area through random sampling points in the state space. However, the disadvantage of sampling-based methods is that if the parameters of the planner are not set reasonably (e.g., too few search limits, too few sampling points, etc.), the solution may not be found. In addition, graph-search-based and sampling-based methods could potentially lead to non-smooth trajectories and also suffer from computational efficiency due to a large discrete space.

The numerical optimization-based methods aim at maximizing or minimizing a function that is constrained by practical conditions. But the nonconvex constraints make the problem complex and super challenging to solve. In the existing literature, people have developed mature and practical optimization algorithms to solve motion planning problems in practice. First, Linear Quadratic Regulator (LQR) can gain the optimal control law with state linear feedback, making it easy to constitute closed-loop optimal control. To take the nonlinear dynamics and nonconvex cost function, the Iterative Linear Quadratic Regulator (iLQR) [8] [9] is employed to effectively deal with optimal control problems of nonlinear dynamics and nonconvex cost functions. Due to its high efficiency, iLQR is an ideal candidate to solve real-time motion planning problems with low latency.

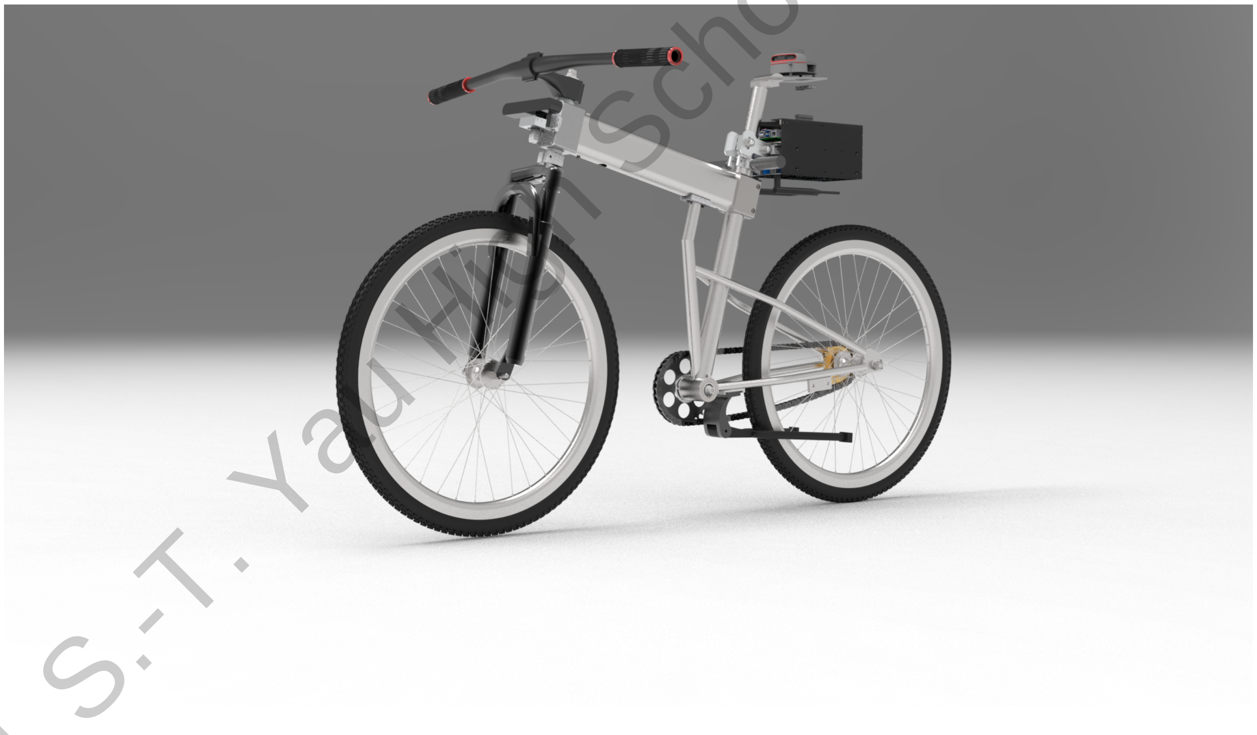
Figure 1: Electric bicycle with autonomous driving and self-balancing function.

In this paper, we implement an iLQR-based motion planning algorithm for autonomous vehicles and tested this motion planning on our designed self-balancing autonomous bicycle in Figure (1), which can increase the safety of vehicles by avoiding obstacles and following desired driving behaviors. The developed iLQR motion planning algorithm can produce safe and reliable trajectories with locally optimal convergence certificates and deal with complicated scenarios with high efficiency. The use of this method in assisted driving systems is expected to reduce traffic accidents while saving people's driving effort and improving social efficiency.

2 Existing Methods for Motion Planning

Motion planning involves a series of control decisions, and is usually divided into high-level routing and low-level trajectory optimization [10]. In high-level routing, plans are often calculated by offline maps before the vehicles start. During the movement of the vehicle, the low-level trajectories are obtained by sensing the environment around and generating the local maps thus allowing the vehicles to deal with the dynamic obstacles. Overall, motion planning methods can be divided into three types: graph-search-based planners, sampling-based planners, and optimization-based planners.

2.1 Graph-Search-Based Planners

The essence of graph-search-based planners is to find how to get from the starting point A to the target point B in state space. In this space, the motion planner relies on the known environment map and obstacle information to construct the trajectories from starting point to the target point, which has two methods: depth-first and breadth-first.

2.1.1 Dijkstra Algorithm

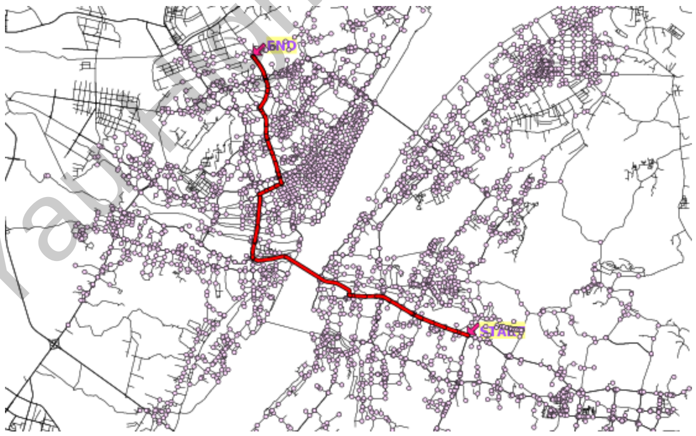

Figure 2: Dijkstra algorithm is applied for global path planning in order to optimize taxi drivers' driving routes in [11], so as to improve operational efficiency.

In Dijkstra algorithm, the total movement cost of each point from the initial point needs to be calculated. The priority queue structure is also needed for this algorithm which stores all the nodes to be traversed while ranking them according to their cost. During the operation of the algorithm, the next node to traverse is selected by the node with the lowest cost in the priority queue. Until the target point is reached.

Dijkstra algorithm uses a greedy strategy [12], creating an array range, the i-th range denotes the distance from the starting point to point i. In the beginning, all points are initialized to positive infinity, except for the points directly adjacent to the starting point. Create a variable s. The points in s indicate the points for which the shortest path has been found. At first, there are only starting points in s. After finding the minimum value of the point in the range matrix that has not been added to s. and adds that point to s. Then we need to examine which path is shorter between the path through the newly added point to the remaining points and the original path, and if the new path is shorter, we need to update the value of the range array. Then find the minimum value of the points in the range matrix that have not been added to s. Repeat the above operation until s contains all the points in the graph. Dijkstra algorithm is applied for global path planning in order to optimize cab drivers' driving routes in [11] as shown in Figure (2), so as to improve operational efficiency.

2.1.2 A-Star Algorithm (A*)

In order to solve the search efficiency problem of the Dijkstra algorithm, the A* algorithm guides the search process with the help of a heuristic function.

Specifically, the basic idea of A* means $g(x)$ is the cost of the initial point to point x, and $h(x)$ is the heuristic function part, which implements the calculation of the expected cost from the target point to the point x. The A* algorithm evaluates the initial point to the target point. The main loop checks the node x with the smallest $f(x)$ at each execution, where $f(x) = g(x) + h(x)$.

The A* algorithm is often applied in classical motion planning. Its heuristic search feature can reduce the computation time, but its results are usually discontinuous and the heuristic rules are not easily obtained directly. Figure (3) shows the A* algorithm adopted by Leedy and others, in which a local path is built for the autonomous car "Rockey" for the 2005 DARPA Urban Challenge.

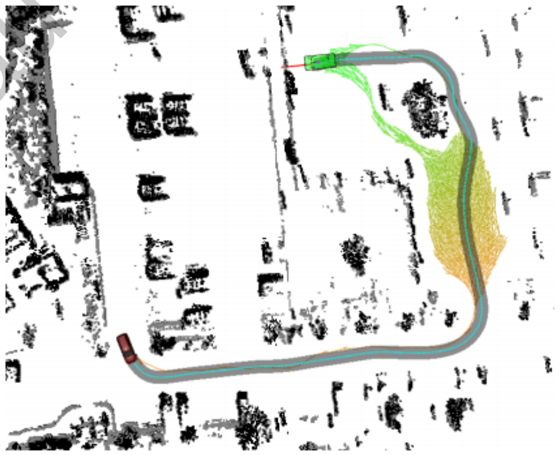

Figure 3: A* algorithm adopted by Leedy and others, in which a local path is built for the autonomous car "Rockey" for the 2005 DARPA Urban Challenge. [13]

2.2 Sampling-Based Planners

The sampling-based planner can quickly and efficiently explore the high-dimensional space by randomly sampling the points in the state space and directing the search to the empty area to produce a trajectory from the initial point to the target point. However, the quality of the paths obtained by this algorithm is mediocre, and they are usually not optimal.

The Rapid-Exploring Random Tree (RRT) algorithm is often adopted to handle high-dimensional motion planning problems with high efficiency and accuracy, and is often applied to generate robot arm trajectories or aircraft motion planning.

The basic idea of the RRT algorithm is to rapidly expand a group of tree-like paths to explore most of the space, waiting for an opportunity to find a feasible path. An initial point is chosen as the root of the tree, and child nodes are randomly generated and distances are recorded. The random nodes are usually distributed and generated equally in all directions when no obstacles are encountered. When the generated children nodes encounter the target point, a feasible path to the target point is obtained [14].

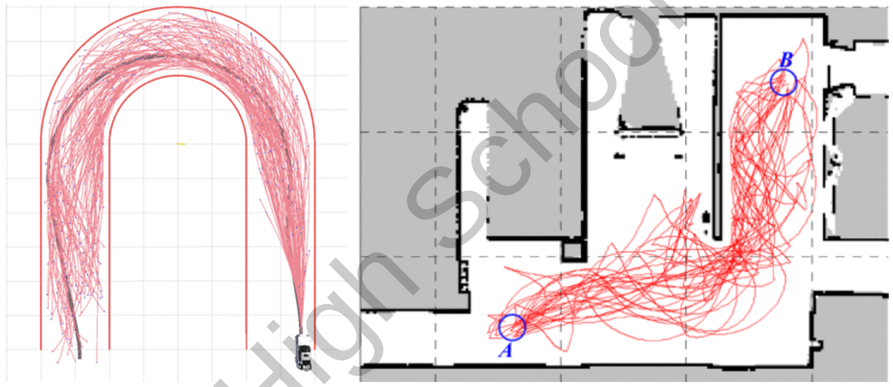

Figure 4: Find a feasible route from the starting point to the target point by using the RRT algorithm has shown in [15] and [16].

3 Iterative Linear Quadratic Regulator Algorithm

Motion planning is essentially a model-based optimal control problem, which guarantees the solution satisfies vehicle kinematics and is critical to real-time control in real time. Linear quadratic regulator (LQR) is very efficient to solve those problems, but strictly relies on linear dynamics and quadratic cost functions. However, many real-life engineering problems have nonlinear dynamics or cost functions. Therefore, LQR cannot be directly applied to tackle those nonlinear problems. To address the challenge of nonlinearity in the optimal control problems, Iterative linear quadratic regulator (iLQR) has been proposed to achieve a local solution in high efficiency.

3.1 Discrete-Time Dynamics

The system with discrete-time dynamics by function $\mathbf{f}$

$$\mathbf{x}_{t+1} = \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t), \tag{1}$$

describes the dynamics transition from time t to $t+1$ given the control at time t. The state $\mathbf{x}_t$ and the control $\mathbf{u}_t$ means the corresponding state and control values at time t. Let the state sequence $\mathbf{X} \equiv \{\mathbf{x}_0, \mathbf{x}_1 ..., \mathbf{x}_N\}$ and the control sequence $\mathbf{U} \equiv \{\mathbf{u}_0, \mathbf{u}_1 ..., \mathbf{u}_{N-1}\}$ satisfy Equation (1) for a fixed period of time horizon.

3.2 Cost Functions

The running costs c and the final step cost c_f is the sum of the total cost $C(\mathbf{x}_0, \mathbf{U})$, incurred from the starting state $\mathbf{x}_0$ and the control sequence $\mathbf{U}$ for a fixed time horizon:

$$C(\mathbf{x}_0, \mathbf{U}) = \sum_{t=0}^{N-1} c(\mathbf{x}_t, \mathbf{u}_t) + c_f(\mathbf{x}_N). \tag{2}$$

Similar to the reward in reinforcement learning (RL), the optimal control problem defines the cost function with the same goal. RL maximizes the cumulative reward, while optimal control minimizes the cumulative losses.

$$\mathbf{U}^*(\mathbf{x}) \equiv \arg\min_{\mathbf{U}} C(\mathbf{x}, \mathbf{U}), \tag{3}$$

$$s.t.\ \mathbf{x_0} = \mathbf{x}_{\text{init}}. \tag{4}$$

Let the partial sum of costs from t to N be the cost-to-go C_t at time t,

$$C_t(\mathbf{x}, \mathbf{U}_t) = \sum_{j=t}^{N-1} c(\mathbf{x}_j, \mathbf{u}_j) + c_f(\mathbf{x}_N). \tag{5}$$

The value function at time t is the cost-to-go function C_t given the optimal control sequence

$$V(\mathbf{x}, t) \equiv \min_{\mathbf{X},\mathbf{U}}\ C_t(\mathbf{X}, \mathbf{U}), \tag{6}$$

where $V(\mathbf{x}, N) = c_f(\mathbf{x}_N)$. The dynamic programming principle reduces the entire minimized control sequence to a single controlled minimization sequence that proceeds backward in time

$$V(\mathbf{x}, t) = \min_{\mathbf{u}} \left[c(\mathbf{x}, \mathbf{u}) + V'(\mathbf{f}(\mathbf{x}, \mathbf{u}), t+1) \right]. \tag{7}$$

3.3 Backward Path

In iLQR the discrete-time dynamics are expanded to the first order and cost is expanded until the second order. Differential dynamic programming (DDP) is a very similar technique, and the only difference is DDP expands the discrete-time dynamics to the second order. In this case, the integration of Equation (1) is performed for a given U. Next, a second-order Taylor expansion is performed on function Q and the local solution of Equation (7) is calculated. Define the argument in Equation (7) as a function of perturbations of the t-th nominal $(\mathbf{x}, \mathbf{u})$ pair:

$$Q(\delta\mathbf{x}, \delta\mathbf{u}) = c(\mathbf{x}+\delta\mathbf{x}, \mathbf{u}+\delta\mathbf{u}, t) - c(\mathbf{x}, \mathbf{u}, t) + V(\mathbf{f}(\mathbf{x}+\delta\mathbf{x}, \mathbf{u}+\delta\mathbf{u}), t+1) - V(\mathbf{f}(\mathbf{x}, \mathbf{u}), t+1) \tag{8}$$

and expand it to the second order

$$\approx \frac{1}{2}\begin{bmatrix} 1 \\ \delta\mathbf{x} \\ \delta\mathbf{u} \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} 0 & Q_{\mathbf{x}}^{\mathrm{T}} & Q_{\mathbf{u}}^{\mathrm{T}} \\ Q_{\mathbf{x}} & Q_{\mathbf{xx}} & Q_{\mathbf{xu}} \\ Q_{\mathbf{u}} & Q_{\mathbf{ux}} & Q_{\mathbf{uu}} \end{bmatrix} \begin{bmatrix} 1 \\ \delta\mathbf{x} \\ \delta\mathbf{u} \end{bmatrix}. \tag{9}$$

The expansion coefficients are

$$Q_{\mathbf{x}} = \mathbf{c}_{\mathbf{x}} + \mathbf{f}_{\mathbf{x}}^{\mathrm{T}} V'_{\mathbf{x}}, \tag{10}$$
$$Q_{\mathbf{u}} = \mathbf{c}_{\mathbf{u}} + \mathbf{f}_{\mathbf{u}}^{\mathrm{T}} V'_{\mathbf{x}}, \tag{11}$$
$$Q_{\mathbf{xx}} = \mathbf{c}_{\mathbf{xx}} + \mathbf{f}_{\mathbf{x}}^{\mathrm{T}} V'_{\mathbf{xx}} \mathbf{f}_{\mathbf{x}} + V'_{\mathbf{x}} \cdot \mathbf{f}_{\mathbf{xx}}, \tag{12}$$
$$Q_{\mathbf{uu}} = \mathbf{c}_{\mathbf{uu}} + \mathbf{f}_{\mathbf{u}}^{\mathrm{T}} V'_{\mathbf{xx}} \mathbf{f}_{\mathbf{u}} + V'_{\mathbf{x}} \cdot \mathbf{f}_{\mathbf{uu}}, \tag{13}$$
$$Q_{\mathbf{ux}} = \mathbf{c}_{\mathbf{ux}} + \mathbf{f}_{\mathbf{u}}^{\mathrm{T}} V'_{\mathbf{xx}} \mathbf{f}_{\mathbf{x}} + V'_{\mathbf{x}} \cdot \mathbf{f}_{\mathbf{ux}}. \tag{14}$$

As for the second half of the second-order derivative of $\mathbf{f}$, it is not present in iLQR, but it is present in DDP. From this point, we can also see that the DDP update is a full Newton step, while the iLQR is a Gauss-Newton step.

Comparing the minimization of Equation (9) with $\delta\mathbf{u}$, we obtain

$$\delta\mathbf{u}^* = \arg\min_{\delta\mathbf{u}} Q(\delta\mathbf{x}, \delta\mathbf{u}) = -Q_{\mathbf{uu}}^{-1}(Q_{\mathbf{u}} + Q_{\mathbf{ux}}\delta\mathbf{x}) = \mathbf{K}\delta\mathbf{x} + \mathbf{k}. \tag{15}$$

This is a local linear feedback policy with

$$\mathbf{K} = -Q_{\mathbf{uu}}^{-1} Q_{\mathbf{ux}}, \tag{16}$$
$$\mathbf{k} = -Q_{\mathbf{uu}}^{-1} Q_{\mathbf{u}}. \tag{17}$$

Substituting this policy into Equation (9), we can obtain a quadratic model of the value of time t.

$$\Delta V(t) = -\frac{1}{2} Q_{\mathbf{u}} Q_{\mathbf{uu}}^{-1} Q_{\mathbf{u}}, \tag{18}$$
$$V_{\mathbf{x}}(t) = Q_{\mathbf{x}} - Q_{\mathbf{u}} Q_{\mathbf{uu}}^{-1} Q_{\mathbf{ux}}, \tag{19}$$
$$V_{\mathbf{xx}}(t) = Q_{\mathbf{xx}} - Q_{\mathbf{xu}} Q_{\mathbf{uu}}^{-1} Q_{\mathbf{ux}}. \tag{20}$$

3.3.1 Regularization Term

The Newton iteration has two drawbacks. First, the Hessian matrix of the objective function at a point is non-positive definite, resulting in the Hessian matrix not being invertible or the direction of the computed update being the direction that increases the objective function. So the Newton iteration may not converge or even fail due to numerical issues when the Hessian matrix is not positive definite. Secondly, when the linearization of dynamics is far from the lowest point or the second-order approximation of cost function is not accurate enough, it is also likely to have convergence issues. The gradient descent method could be another candidate. For one thing, there is no need to compute the Hessian matrix for the gradient descent method. For another, it does not matter even if it has a distance from the lowest point. But the disadvantage of gradient descent is that it is computationally expensive, and might require orders of magnitudes more iterations.

Therefore, the Levenberg-Marquardt (LM) [17] algorithm is proposed, which combines the advantages of both by adjusting the LM parameter μ larger or smaller so that the algorithm is biased towards gradient descent or Newton's method. The first term is weakened when the LM parameter is large, and reverts to the original Newton's iteration when the LM parameter $\mu = 0$.

$$Q_{\mathbf{uu}} = \mathbf{c}_{\mathbf{uu}} + \mathbf{f}_{\mathbf{u}}^{\mathrm{T}} \left(V'_{\mathbf{xx}} + \mu \mathbf{I}_n\right) \mathbf{f}_{\mathbf{u}} + V_{\mathbf{x}} \cdot \mathbf{f}_{\mathbf{uu}}, \tag{21}$$
$$Q_{\mathbf{ux}} = \mathbf{c}_{\mathbf{ux}} + \mathbf{f}_{\mathbf{u}}^{\mathrm{T}} \left(V'_{\mathbf{xx}} + \mu \mathbf{I}_n\right) \mathbf{f}_{\mathbf{x}} + V_{\mathbf{x}} - \mathbf{f}_{\mathbf{ux}}. \tag{22}$$

In order to decrease t , all equations of the backword path need to be iterated. If encountering a non-PD $Q_{\mathbf{uu}}$, μ is increased and the backward path should be restarted. μ is decreased if successful.

3.4 Forward Path

After running backward pass on state $\delta \mathbf{x}_t = \mathbf{x}_t - \hat{\mathbf{x}}_t$ and control $\delta \mathbf{u}_t = \mathbf{u}_t - \hat{\mathbf{u}}_t$, where $\hat{\mathbf{x}}$ is the trajectory obtained from the previous iteration. Then run forward pass with real nonlinear dynamics and $\mathbf{u}_t = \mathbf{K}_t \left(\mathbf{x}_t - \hat{\mathbf{x}}_t\right) + \mathbf{k}_t + \hat{\mathbf{u}}_t$. Keep update $\hat{\mathbf{x}}_t$ and $\hat{\mathbf{u}}_t$ based on states and controls in the forward pass. It should be noted that the forward is interacted with the real nonlinear system to get the real trajectory, not with the linearized model.

3.4.1 Line Search

The Newton method iteration process might have overshoots shown in Figure (5). Our current iteration locates at point A, and the function in red is our second-order approximation to the function. The next step of the Newton iteration is to go to the lowest value under the current approximation, which is the location of point C, but this location deviates from the true lowest point B, then it overshoots. However, this position deviates from the true minimum at point B, and it overshoot. The cost is likely to rise instead of fall and does not converge when the second-order approximation deviates too much from the real situation. The common solution is to perform a line search, adding a linear factor α in front of the updated amount and reducing updated the step size that makes the true function value reduced by a certain amount.

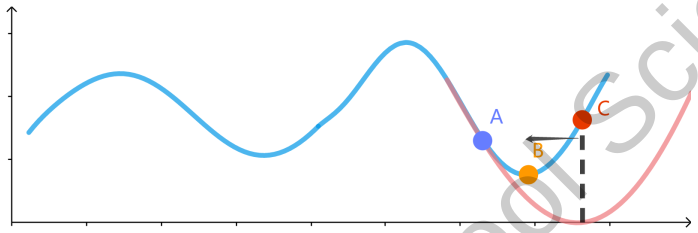

Figure 5: Using the Newton iteration method may lead to overshoot. The position obtained by the current iteration is point A. The red function is our second-order approximation to the blue function, and the Newton iteration goes next to the position of the lowest value under the current approximation, which is the position of point C. But this position is off from the true lowest point B.

In iLQR, the forward path is crucial, and the number of updates is related to the speed of convergence. As in Equation (23) by adding a linear search factor α for backtracking to the control law term $\mathbf{k}$.

$$\hat{\mathbf{u}} = \mathbf{u} + \alpha\mathbf{k} + \mathbf{K}(\hat{\mathbf{x}} - \mathbf{x}). \tag{23}$$

The α decreases from 1 to a smaller value close to 0. When $\alpha = 0$ is not updated, the trajectory does not change.

4 Motion Planning for Obstacle Avoidance and Driving Rule Satisfaction

We utilize iLQR algorithm to enable obstacle avoidance and driving rules satisfaction features. This motion planning algorithm strictly follows vehicle dynamics, and is able to handle complex scenarios with specific safety requirements.

4.1 Discrete-Time Dynamics Function

In our vehicle trajectory optimization problem, we have 8 states $(x, y, \theta, v, \phi, a_t, a_n, s)$ and 2 controls $(dcurvdt, datdt)$. State x and state y are the x-axis and y-axis coordinates of the vehicle, state θ is the yaw angle of the vehicle, and state v is the speed of the vehicle. State ϕ is the curvature, which is the derivative of θ with respect to s ($\frac{\partial\theta}{\partial s}$). In addition, state s is arc length, which calculates the length of traveled distance. a_t is the tangential acceleration and the a_n means the normal acceleration. The control $dcurvdt$ is the derivative of the curvature over time ($\frac{\partial curv}{\partial t}$). The control $datdt$ is also called the jerk, which is the derivative of tangential acceleration over time ($\frac{\partial a_t}{\partial t}$). Specifically, given state and control at time t, the next state at time $t+1$ follow the following dynamics equations:

$$\begin{aligned} x_{t+1} &= x_t + v_t \cos\theta_t \mathrm{dt}, \\ y_{t+1} &= y_t + v_t \sin\theta_t \mathrm{dt}, \\ \theta_{t+1} &= \theta_t + v_t\phi_t \mathrm{dt}, \\ v_{t+1} &= v_t + a_{t,t}\mathrm{dt}, \\ \phi_{t+1} &= \phi_t + dcurvdt_t * \mathrm{dt}, \\ a_{t,t+1} &= a_{t,t} + datdt_t * \mathrm{dt}, \\ a_{n,t+1} &= v_{t+1}^2\phi_{t+1}, \\ s_{t+1} &= s_t + v_t\mathrm{dt}. \end{aligned} \tag{24}$$

4.2 Cost Function

The cost function represents a parametrized driving goals. In order to achieve specific driving conditions such as avoiding obstacles, following lane centers, and driving comforts, we design the corresponding cost function terms.

4.2.1 Obstacle Cost

To avoid obstacles in the lane, we design the obstacle cost function. The function obtains the obstacle cost by multiplying the cost weight with the square of the distance between the vehicle and the obstacles,

$$\text{obstacle cost} = \text{weight} * [\text{distance}(\text{vehicle}, \text{obstacle})]^2. \tag{25}$$

4.2.2 Lane Center Cost

The vehicle is preferred to drive near the lane center if that is safe. The lane center cost will be penalized if the vehicle is away from the lane centerline,

$$\text{lane center cost} = \text{weight} * [\text{distance}(\text{vehicle}, \text{lane center})]^2. \tag{26}$$

4.2.3 Progress Cost

In order to reach the target location, we need the vehicle to move towards the target position. We will get higher cost if we are far away from the target pose,

$$\text{progress cost} = \text{weight} * \left[(\text{x} - \text{goal}_x)^2 + (\text{y} - \text{goal}_y)^2\right]. \tag{27}$$

4.2.4 Control Cost

The control cost handles the driving comfort. We want to avoid high control values that could result in flaky behaviors,

$$\text{control cost} = \text{weight} * dcurvdt^2 + \text{weight} * datdt^2. \tag{28}$$

4.2.5 Bound Violation Cost

Driving on public roads requires following traffic regulations. The bounds violation cost regulates the vehicles to comply with the specified hard bounds while driving, such as speed limits, and will apply a cost if the vehicle violates desired operation bounds.

$$\text{bounds violation cost} = \text{weight} * [\text{bounds violation values}]^2. \tag{29}$$

In the iLQR algorithm, in order to get the optimal path while satisfying the driving conditions mentioned above, we minimize the summation of all costs above.

4.3 Numerical Example: Motion Planning for Obstacle Avoidance and Goal Satisfaction

To illustrate the performance of our motion planning algorithm, we want our autonomous bicycle to drive at simulated a road with obstacles and lane centerline. The goal of the vehicle is to travel from the left starting position to a goal position along the road center if possible, and get around the obstacles safely with comfort. The obstacle may block part of the lane, so the planned trajectory needs to get around the obstacle with a safe buffer distance, while keeping the vehicle along the lane center as much as possible. As shown in Figure (6), the blue and green regions are obstacles, and the gray regions illustrated the safe buffer distance we want to keep away from the obstacle. We implement the motion planning algorithm in TensorFlow and obtained the desired trajectory as below.

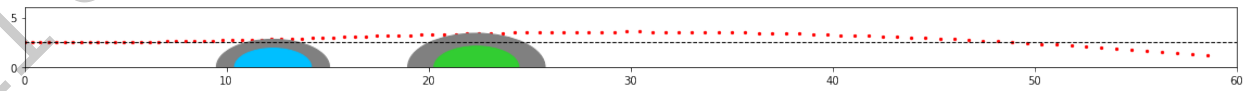

Figure 6: Test motion planning by simulating the actual environment.

The final result is shown in Figure (6), the x-axis is the longitudinal direction of the road and the y-axis is the lateral direction of the road. The dotted line in the center of the road is the lane center, and the red dotted trajectory is the optimal path calculated by our motion planning algorithm. We successfully drive around the obstacles with comfort and following closely to the lane center if possible.

5 Broader Impact

Besides implementation the motion planning algorithm, we have been building an autunomous electric bicycle product. In January of 2021, we spent six months designing the model of an electric bicycle with autonomous driving and the self-balancing function shown in Figure (7).

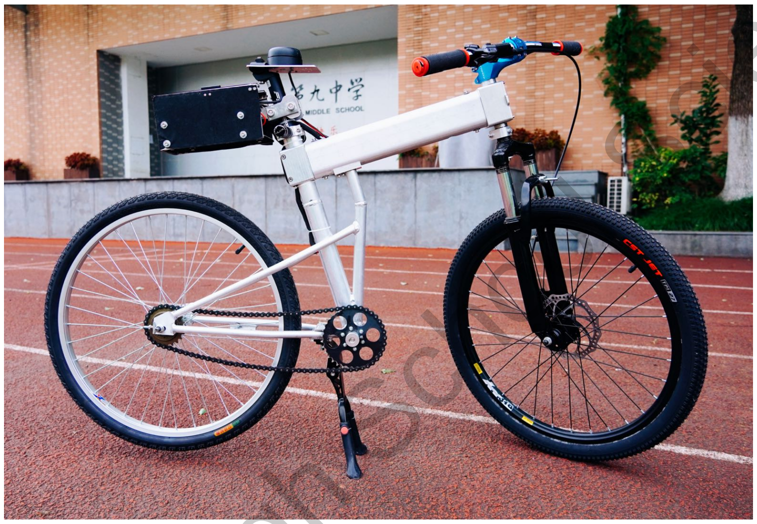

Figure 7: The autonomous self-balance electric bicycle.

The electric bicycle is controlled by the Nvidia-Jetson-Xavier-NX development board together with the STM32F103RCT6 development board. The Xavier development board provides the future trajectory of the vehicle, and the STM32 development board controls the bicycle balance in real-time through a serial-level PID algorithm while receiving and executing the trajectory and control sequences sent from the Xavier development board. The depth camera in front of the bike and the LIDAR above the bicycle collect information about the surrounding environment and send it to the Xavier development board through the serial port. After receiving the trajectory and control sequences from Xavier, the STM32 development board will substitute the data into the serial PID calculation to ensure the bicycle's balance and drives the bicycle to the target point by controlling the handlebar and throttle motors. In Figure (8), the handlebars and the motor controlling the throttle are connected to the STM32 development board in the control center at the back of the bike via blue wires; the depth camera at the front of the bike and the radar above are connected to the Xavier development board via yellow wires. Below the board is a 24V5AH Li-ion battery to provide power for the bike drive and computing. The communication between Xavier and the STM32 development board is done through the red wire using CAN protocol.

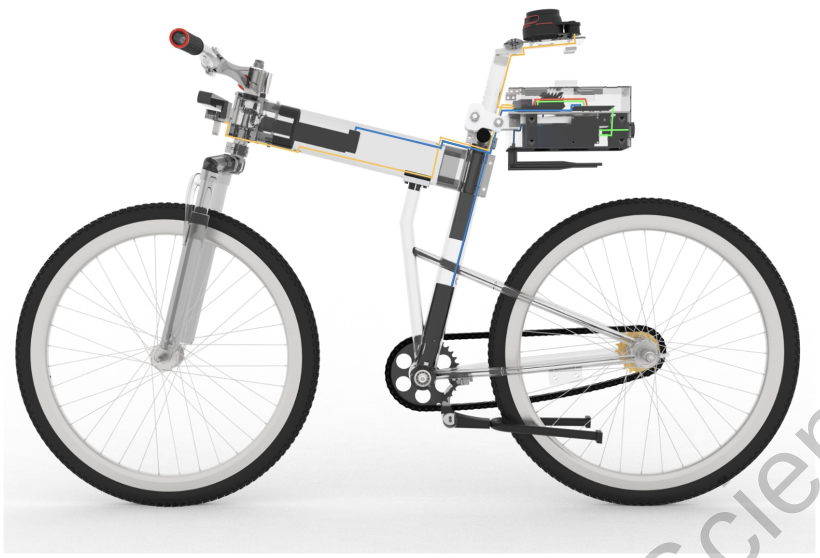

Figure 8: The structure of autonomous electric bicycle.

In the future, we will adopt the control systems of the autonomous self-balance bike to the common electric bicycle in China. The assisted driving system helps drivers to reduce emergency accidents and improve driving comfort at the same time.

6 Conclusion

In this paper, we successfully utilize iLQR motion planning for autonomous electric vehicle driving. The developed iLQR motion planning algorithm can produce safe and reliable trajectories with locally optimal convergence certificates and deal with complicated scenarios with high efficiency. Due to these features, iLQR is an ideal candidate to solve real-time motion planning problems. The use of this method in assisted driving systems of electric bicycles or other vehicles is expected to reduce traffic accidents while saving people's driving effort and improving social efficiency.

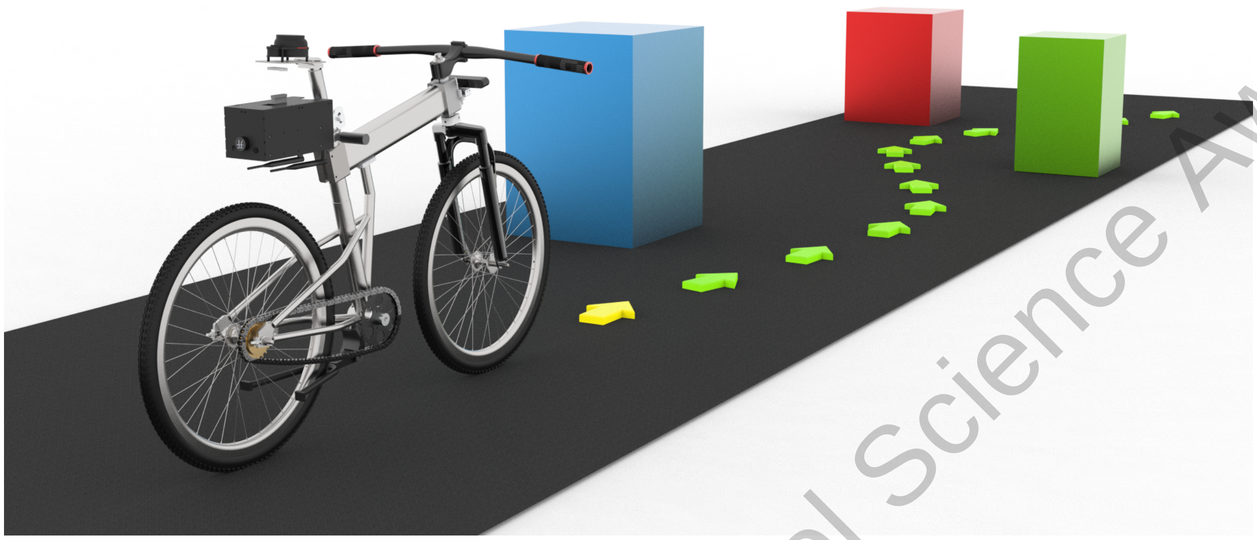

Figure 9: The demo of bicycle using iLQR motion planning.

References

[1] China statistical yearbook. *Statistical Theory and Practice*, (1):24–5, 2020.

[2] David González, Joshué Pérez, Vicente Milanés, and Fawzi Nashashibi. A review of motion planning techniques for automated vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 17(4):1135–1145, 2016.

[3] Thomas Howard, Mihail Pivtoraiko, Ross A. Knepper, and Alonzo Kelly. Model-predictive motion planning: Several key developments for autonomous mobile robots. *IEEE Robotics Automation Magazine*, 21(1):64–73, 2014.

[4] Joo Young Hwang, Jun Song Kim, Sang Seok Lim, and Kyu Ho Park. A fast path planning by path graph optimization. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 33(1):121–129, 2003.

[5] Dave Ferguson and Anthony Stentz. Field d*: An interpolation-based path planner and replanner. In Sebastian Thrun, Rodney Brooks, and Hugh Durrant-Whyte, editors, *Robotics Research*, pages 239–253, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[6] S.M. LaValle and J.J. Kuffner. Randomized kinodynamic planning. In *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, volume 1, pages 473–479 vol.1, 1999.

[7] L.E. Kavraki, P. Svestka, J.-C. Latombe, and M.H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.

[8] Weiwei Li and Emanuel Todorov. Iterative linear quadratic regulator design for nonlinear biological movement systems. volume 1, pages 222–229, 01 2004.

[9] E. Todorov and Weiwei Li. A generalized iterative lqg method for locally-optimal feedback control of constrained nonlinear stochastic systems. In *Proceedings of the 2005, American Control Conference, 2005.*, pages 300–306 vol. 1, 2005.

[10] K.H. Sedighi, K. Ashenayi, T.W. Manikas, R.L. Wainwright, and Heng-Ming Tai. Autonomous local path planning for a mobile robot using a genetic algorithm. In *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753)*, volume 2, pages 1338–1345 Vol.2, 2004.

[11] Qingquan Li, Zhe Zeng, Bisheng Yang, and T. Zhang. Hierarchical route planning based on taxi gps-trajectories. *2009 17th International Conference on Geoinformatics*, pages 1–5, 2009.

[12] A. Stentz. Optimal and efficient path planning for partially-known environments. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pages 3310–3317 vol.4, 1994.

[13] Bauman C. Cacciola S. Michael Webster J. Reinholtz C.F. Leedy B.M., Putney J.S. Virginia tech's twin contenders: A comparative study of reactive and deliberative navigation. *The 2005 DARPA Grand Challenge. Springer Tracts in Advanced Robotics*, 36, 2007.

[14] Steven M. LaValle and Jr. James J. Kuffner. Randomized kinodynamic planning. *The International Journal of Robotics Research*, 20(5):378–400, 2001.

[15] Jeong hwan Jeon, Raghvendra V. Cowlagi, Steven C. Peters, Sertac Karaman, Emilio Frazzoli, Panagiotis Tsiotras, and Karl Iagnemma. Optimal motion planning with the half-car dynamical model for autonomous high-speed driving. In *2013 American Control Conference*, pages 188–193, 2013.

[16] Chang-bae Moon and Woojin Chung. Kinodynamic planner dual-tree rrt (dt-rrt) for two-wheeled mobile robots using the rapidly exploring random tree. *IEEE Transactions on Industrial Electronics*, 62(2):1080–1090, 2015.

[17] Donald W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2):431–441, 1963.

7 Acknowledgement

In 2016, I was greatly blown away by the brain-like chip Tianji developed by a robotics team at Tsinghua University that I read about in the journal of Nature. Among them, the self-balancing bicycle equipped with the Tianji chip made by the Tsinghua team was deeply attractive to me, which was the inspiration for this self-driving bicycle, and decided to make an autonomous driving bicycle when I had the ability to do so, and even surpass the demo of Tsinghua team.

In the beginning, I chose to design this autonomous driving bicycle to challenge myself and achieve my dream in junior high school.

During my junior high and high school, I have been training for OI at Xuejun High School, and I am grateful to my parents for providing me with the opportunity to observe a computer science course at Zhejiang University. Meanwhile, I would like to thank Hangzhou Ninth High School for providing me with the experimental site and equipment.

I have designed a bicycle model using Solidworks independently since January this year. In March, I sent the model to a company that supports CNC cutting to create the frame of the bike. At the same time, I bought a development board with STM32-F103RCT6 chip, motor, FOC, and other devices to support my bike.

In a chance encounter, I learned that many people have died in accidents due to improper electric bicycle driving practices. I found that I could design a type of electric bicycle with assisted driving function based on my autonomous driving bicycle, which could reduce accidents by using assisted driving technology to make emergency avoidance when the system detects an imminent emergency situation.

Meanwhile, I would like to thank Xueming Shao from the School of Aeronautics and Astronautics of Zhejiang University, for his help in microcontroller programming and some physics modeling problems. In the process of developing and experimenting with motion planning using the Xavier development board, I would like to thank Yuanxun Shao from the Georgia Institute of Technology for giving me ideas on the iLQR algorithm. In writing my paper, thanks to Tao Yang and Yuanxun Shao gave me advice on the standardization of the paper.