POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

Duckrace: Iterative Learning Control for Autonomous Racing

Laurea Magistrale in Automation and Control Engineering

Author: Giulio Vaccari

Advisor: Prof. Simone Formentin

Co-advisor: Valentina Breschi, Riccardo Busetto

Academic year: 2021-2022

1. Introduction

The development of autonomous vehicles has become an increasingly important research focus in recent years. One crucial aspect of autonomous vehicle development is the development of efficient and effective racing algorithms that can navigate complex road networks and achieve optimal performance. Model Predictive Control (MPC) is a powerful control method that can be used to design racing algorithms for autonomous vehicles. However, traditional MPC algorithms are based on pre-computed trajectories that do not adapt to the environment.

In this thesis, we present a comprehensive study on the development of a Learning MPC (LMPC) algorithm for the Duckiebot, a miniature differential drive wheeled vehicle that navigates a race track using computer vision and control techniques. The objective of the LMPC algorithm is to learn the optimal trajectory for the Duckiebot based on feedback from previous laps.

To achieve this objective, we first designed and implemented a traditional MPC algorithm that could control the velocity and steering angle of the Duckiebot, with the objective of completing a lap of the Duckietown as quickly as possible but following a pre determined trajectory that we chose to be the center line of the road. We researched existing MPC algorithms and adapted them to fit the specific needs and constraints of the Duckiebot.

We then evolved the MPC algorithm into an LMPC algorithm by incorporating a method to efficiently decide the optimal reference for the MPC. The LMPC algorithm learns from the previous laps of the Duckiebot and adapts its trajectory to minimize the lap time.

During the course of this research, we established a Duckietown laboratory at Politecnico di Milano that is purpose-built for the task of racing. The laboratory is equipped with a variety of hardware and software tools, including multiple Duckiebots, cameras, and a dedicated track that is designed to challenge the capabilities of the Duckiebots. One of the secondary objectives of this thesis is to develop a cheap but effective localization technique that is specifically tailored for the challenging task of racing in our Duckietown environment.

Finally, we evaluated the performance of the LMPC algorithm in the Duckietown lab. We compared the performance of the LMPC algorithm with that of traditional MPC algorithms and observed that the LMPC algorithm achieved better performance in terms of lap time, with a time reduction of the 44% already from the first lap.

Overall, this thesis presents a comprehensive study on the development of a Learning MPC algorithm for the Duckiebot. The results of this thesis are expected to contribute to the development of more efficient and effective racing algorithms for autonomous vehicles, which will ultimately lead to safer and more reliable autonomous vehicles.

2. Problem statement and literature

2.1. Duckietown

Duckietown is an open-source platform for teaching and research in robotics and machine learning, consisting of miniature robotic vehicles called Duckiebots that navigate a urban road network. The Duckiebot are differential drive wheeled mobile robots equipped with a front facing camera and an IMU. Duckietown provides a low-cost and accessible platform for students and researchers to experiment with autonomous vehicle technologies. In addition to the physical Duckietown platform, there is also a simulation environment available on OpenAI Gym [1]. The simulator is easily customizable, allowing users to modify various aspects of the simulation, such as the road layout and eventual rewards. Duckietown has been used in a variety of educational and research settings, including university courses, hackathons, and robotics competitions, and the availability of the simulation environment has made it even more accessible to a wider range of users.

2.2. Optimal trajectory planning

Optimal trajectory planning is a crucial aspect of autonomous racing, the goal is to find the fastest and safest path for the vehicle to complete the race.
Optimal trajectory planning is a topic that has received significant attention in the field of autonomous racing, with a wide range of approaches and techniques proposed. However, there are still many challenges and open questions in this area, as many of these approaches have focused on finding the optimal trajectory offline and following it throughout the race, without considering the impact of variations in the trajectory dictated by the presence of other vehicles or obstacles on the track, or the handling of uncommon environments such as dirt roads and rallies. This approach may not always be optimal, as it does not allow for the real-time adaptation of the trajectory to changing conditions or the exploitation of opportunities to maximize the performance of a specific vehicle in a specific environment [2].
A different online approach has been developed by Rosolia and Borelli [3] [4]. In this approach a Learning Model Predictive Controller (LMPC) is used to learn and improve the trajectory at each lap, using the last iterations as baselines to improve. This approah allows to maximize the car performance online in a given track.
Duckietown represents a unique challenge in the development of an iterative MPC: it is difficult to build a model of the duckiebot that takes into account in its state an accurate distance from the track border, making it harder to set the constraint to stay inside the track. As we will see a solution to this problem will also open a new way to dynamically take into account fixed and moving obstacles in the track. On the other hand the Duckietown gym environment is ideal to virtually collect iterative high quality data at each lap, making it the perfect solution to test an iterative controller.

3. Experimental Setup

3.1. Duckiebot

Duckiebots are the mobile robots used in the Duckietown platform. Duckiebots are equipped with sensors and actuators, including a camera, ultrasonic sensor, IMU, motor controller, and motors. They are powered by a Nvidia Jetson Nano single-board computer, which provides powerful computing capabilities for computer vision and high computing tasks. Duckiebots are differential wheeled robots and they are designed to be controlled by giving the left and right wheel speed to a ROS node, with the control of the motors and the internal ROS connectivity being handled entirely by the robot out of the box.
Duckiebots require calibration, which consists in finding the optimal trim and gain parameters. To calibrate the wheels the robot runs along a straight line of known lengths, such as 2 meters. The trim and gain parameters are then adjusted until the robot is able to run straight along the

entire length of the line.
The Duckiebot model we used is (1), with u longitudinal speed, ω angular speed, V_r right wheel speed, V_l left wheel speed and parameters to be estimated: γ_1, γ_2, γ_3, ξ_1, ξ_2, ξ_3, a_r^u, a_l^u, a_r^ω, a_l^ω.

$$\begin{bmatrix}\dot{u}\\ \dot{\omega}\end{bmatrix} = \begin{bmatrix}-\gamma_1 u - \gamma_2\omega + \gamma_3\omega^2\\ -\xi_1\omega - \xi_2 u - \xi_3 u\omega\end{bmatrix} + \begin{bmatrix}a_r^u & a_l^u\\ a_r^\omega & a_l^\omega\end{bmatrix}\begin{bmatrix}V_r\\ V_l\end{bmatrix} \quad (1)$$

The αs are related to wheel dimensions and the distance between the wheels, but the other parameters do not have an immediate physical meaning. Duckietown provides the ideal parameters used for the simulator, while we estimated the vehicle parameters minimizing the following function with respect to γ_1, γ_2, γ_3, ξ_1, ξ_2, ξ_3, a_r^u, a_l^u, a_r^ω, a_l^ω:

$$E_x = F_x(\gamma_1, ..., a_l^\omega, x_{i+N-1}) - x_{i+N} \quad (2)$$

$$E_y = F_y(\gamma_1, ..., a_l^\omega, y_{i+N-1}) - y_{i+N} \quad (3)$$

$$\sum_{N=1}^{10}(E_x^2 + E_y^2) \qquad \forall_i \quad (4)$$

With 10 being the horizon in the MPC.

Parameters

	Ideal	Edtimated
γ_1	5	2.14148837
γ_2	0	0.12200042
γ_3	0	-0.28237442
ξ_1	4	1.3380637
ξ_2	0	0.40072379
ξ_3	0	1.30781483
a_r^u	1.5	1.30781483
a_l^u	1.5	1.03762896
a_r^ω	15	2.9650673
a_l^ω	15	2.89169198

Table 1: Estimated parameters.

To discretize the Runge-Kutta integration method has been used.

3.2. Connectivity: ROS and UDP

Duckietown is a system that is designed to be easily integrated with ROS. However, due to the limitations of ROS (which are addressed by ROS2), it is not possible to create a decentralized network. Each robot in Duckietown requires its own master node, and two masters cannot communicate with each other, making it difficult to establish communication between the devices. To address this issue and decrease messaging delay, a UDP multicast has been built on top of ROS messaging standards, allowing nodes in different masters to communicate with each other while still using standardized ROS libraries.

3.3. Map

Both in simulation and in the real environment the standard yet challenging Duckietown loop "ETH Large Loop" has been used. The physical track of the same shape has been named "Milano Duckar".

3.4. Watchtower

To accurately determine the position and orientation of the Duckiebot in the track during our experiments, we implemented a color extraction-based localization algorithm, which was based on the use of a Raspberry Pi computer equipped with a fisheye camera mounted to the ceiling of the track. The camera has been calibrated using ROS standard packages, and the rectification matrices have been stored for later use. We then placed colored paper on the Duckiebot, and used the camera and the algorithm to track the Duckiebot's position and orientation based on the colors of the paper. This allowed us to obtain accurate and reliable location information. We called this camera "watchtower", as it follows the original watchtower idea proposed by Duckietown, but following a very different implementation.

3.4.1 Localization strategy

The localization in the real track is performed by a ROS node in the watchtower with a frequency of around 20Hz. To be able to extract the Duckiebot location we followed a pipeline of image preprocessing and final localization based entirely on OpenCV:

1. Image rectification based on previously extracted parameters.
2. Origin definition: the origin is set on the top right to be coherent with the image standard.

3. Set resolution, the image is downscaled from 1296x972 pixel to 324x243, to allow lower processing times.
4. Automatic white balancing. To account for different lights during the day an algorithm based on the gray world assumption has been implemented.
5. Car localization

Car localization:
The following strategy has been adopted:

1. Pink pixels extraction and computation of their mean with respect to x and y axis, point P is found. The pink color is very well defined and can be used as reference.
2. Draw a bounding box around the pink pixels.
3. Look for the blue pixels inside the bounding box and compute their mean, point B is found.
4. The mean between the two points P and will be the car position.
5. Use P and B to compute the car orientation.

3.4.2 Trajectory extraction

As first trajectory reference for the Duckiebot we decided to use the central yellow line. To extract the line this are the steps that have been followed, using as input the top view of the track:

1. Filter the image by color to extract the yellow.
2. Extract the points using Hough Lines point extraction.
3. Sort the points by angle or by distance. The first is easier but less reliable in complex tracks where a single angle can refer to multiple points along the track. To sort by angle the mean of the x and y coordinates have been computed as track center, then we considered the angle formed by each point to the center point.
4. Interpolation of the points by angle or distance. Another advantage of using distance is that the interpolation can be more precise as the sampled points can have the same distance between one and the other. On the other hand, it is very difficult to translate the distance from a starting point to a position on the track. The best interpolation is quadratic, with some smoothing.
5. Finding the borders is only a matter of finding two points equally distant to each point along the central trajectory. This strategy has been adopted as it is the most lightweight and straight forward.

4. Control Algorithm

4.1. Trajectory following with MPC

The Model Predictive Control (MPC) is a widely utilized technique for advanced control that utilizes a mathematical model of the system being controlled and an optimization algorithm to determine the optimal control sequence. In our work CasaDI with IPOPT has been used for numerical optimization. The general MPC definition is the following:

$$\begin{aligned} \min \quad & J && (5)\\ \text{s.t.} \quad & X_{k+1} = F(X_k, U_k) && (6)\\ & -1 \leq U_k \leq 1 && (7)\\ & X_0 = [x_0, y_0, \theta_0, v_0, \omega_0]^T && (8) \end{aligned}$$

Both without and with preview MPC has been tested. In MPC without preview the reference is a constant point, thus $p_{kr} = p_r \forall k$. From an implementation prospective the next pkr is taken when the position of the car is close enough to the reference point and this distance is a tunable parameter. The MPC without preview is very good at following straight trajectories but it is late to the curves. To overcome these limitations the MPC with preview has been used. This is the formulation, with another parameter u_{max} introduced to maximize speed:

$$J = \sum_{k=0}^{N+1} \|p_k - p_{kr}\|^2_{Q_1} + \|\theta_k - \theta_{kr}\|^2_{Q_2} + \|u_k - u_{max}\|^2_{Q_3} + \|U\|^2_R \quad (9)$$

The formulation takes into account the reference for the orientation of the car. To compute the orientation the following approach was followed:

$$\theta_r = \arctan\left(\frac{y_{k+1} - y_k}{x_{k+1} - x_k}\right) \quad \forall k = 0, ..., N-1 \quad (10)$$

$$\theta_N = \arctan\left(\frac{y_1 - y_N}{x_0 - x_N}\right) \quad (11)$$

From a practical perspective at each step the closest point in the trajectory is found and its next N point are provided as reference for the MPC. To find the closest point quickly Voronoi regions are computed at the start of the process. In the MPC the tunable parameters are:

- N: prediction horizon. Higher means predictions of longer term but more computational time. N=5 is the best, higher does not provides significant improvements.
- Q_1: the weight on the distance from the reference in meters.
- Q_2: the weight on the distance from the angular reference in radians.
- Q_3: the weight on the difference between the current speed and the max speed.
- R: the weight on inputs.

To find the best combination of parameters a sensitivity analysis has been carried forward where R has been kept fixed to 1 and the ratio between the other parameters has been adjusted. The results showed that Q_1/Q_2 and Q_1/Q_3 should be bigger than 10^3, while Q_2/Q_3 should be smaller than 1.

4.2. Trajectory planning and following with LMPC

The Learning Model Predictive Controller is a recently developed technique [?] based on iterative learning control. The idea is similar to the MPC but this time the reference is not given but computed in real time based on the latest iteration. The aim of the LMPC is to minimize a parameter learning the optimal trajectory, in our case it needs to minimize the time per loop. The LMPC formulation, with a change from the original version to take into account the track margins, is the following.

$$J^{LMPC,j}_{t\to t+N}(x^j_t, z^j_t) = \min_{U^j_t, \lambda^j_t} J^{j-1}_l(z^j_t)\lambda^j_t \tag{12}$$

$$s.t. \quad x^j_{t|t} = x^j_t \tag{13}$$

$$\lambda \leq 0, \mathbb{1}\lambda = 1, \mathrm{D}^{\mathrm{j}-1}_{\mathrm{l}}(\mathrm{z}^{\mathrm{j}}_{\mathrm{t}})\lambda^{\mathrm{j}}_{\mathrm{t}} = \mathrm{x}^{\mathrm{j}}_{\mathrm{t+N|t}} \tag{14}$$

$$x^j_{k+1|t} = F(x^j_{k|t}, u^j_{k|t}) \tag{15}$$

$$x^j_{k|t} \subseteq X, u^j_{k|t} \subseteq U \tag{16}$$

$$\forall k = t, ..., t+N-1 \tag{17}$$

$$\lambda_{SS} \leq 0, \mathbb{1}\lambda_{\mathrm{SS}} = 1, \mathrm{T}(\mathrm{z}^{\mathrm{j}}_{\mathrm{t}})\lambda_{\mathrm{SS}} = \mathrm{x} \tag{18}$$

Where $F(x^j_{k|t}, u^j_{k|t})$ is the function with the Duckiebot model that returns the next state. Direct your attention on the convex constraint in (14):

$$\lambda \leq 0, \mathbb{1}\lambda = 1, \mathrm{D}^{\mathrm{j}-1}_{\mathrm{l}}(\mathrm{z}^{\mathrm{j}}_{\mathrm{t}})\lambda^{\mathrm{j}}_{\mathrm{t}} = \mathrm{x}^{\mathrm{j}}_{\mathrm{t+N|t}} \tag{19}$$

$$z^j_t = \begin{cases} x^{j-1}_N & \text{if } t = 0. \\ S^j_l(z^j_{t-1})\lambda^{j,*}_{t-1} & \text{else.} \end{cases} \tag{20}$$

The vector z^j_t represents a candidate terminal state for the planned trajectory of the LMPC at time t. S is the matrix which collects the evolution of the states stored in the columns of the matrix D:

$$S^j_l(x) = [x^l_{t^{l,*}_1+1}, ..., x^l_{t^{l,*}_k+1}, ..., x^j_{t^{j,*}_1+1}, ..., x^j_{t^{j,*}_k+1}] \tag{21}$$

$T(z^j_t)$ are a given number of points in the track margin that are close to the Duckiebot. the distance from the Duckiebot and the number of those points are tunable parameters. In our case there is only one input constraint in 16, $-1 \leq u \leq 1$.

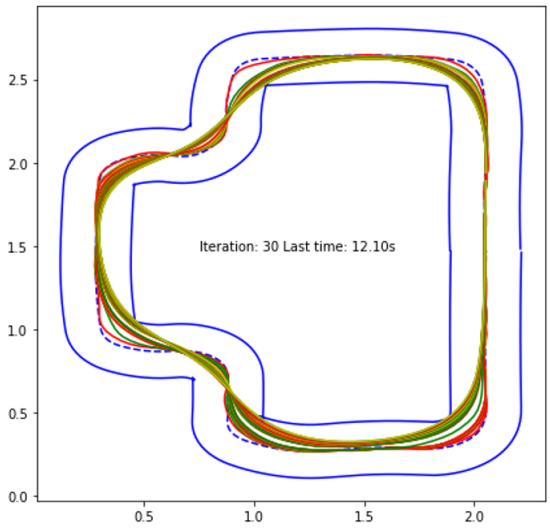

Figure 1: LMPC in Milano Duckar simulation.

4.2.1 Sensitivity analysis

The following hyperparameters had to be tuned:

1. N (integer): The horizon, the bigger the N the faster the convergence, but computationally it gets very heavy already with N=10. N=2 was found to be best;
2. K (integer): The number of nearest neighbours to consider for the convex hull. Bigger K means more exploration, sometimes it leads to cutting too much the curves. K=8 was found to be best;

3. i_j (integer): The number of past iterations to consider in the nearest neighbour. i_j = 4 was found to be best;
4. $Frame\_rate$ (integer): Default for Duckietown is 30Hz, 10Hz have been used in our experiments to guarantee a good accuracy of the model but also faster computational times.

5. Experimental results

5.1. MPC

The application of MPC without preview using real parameters proved to be ineffective both in simulation and on the actual track. Therefore, MPC with preview was utilized. In this study, the formulation used for MPC with preview is the one already defined, with the following parameters: $N = 10$, $Q_1 = 10^3$, $Q_2 = 10^{-2}$, $Q_3 = 0$ and $R = 10$.

5.2. LMPC with preview

The LMPC formulation we originally proposed is a MPC without preview and as explained in the section before it can not work. To overcome this limitation a new LMPC with preview has been designed. The idea is:

1. Compute the reference as a line between the current position and the target, where the target is the nearest point in the previous iteration at distance d, where d is a tunable parameter. For d we used $d = N * 0.03$, being 3cm the distance that the duckiebot covers in 0.1s;
2. Run the MPC with preview and a slack on the reference, to be able to generate a trajectory that stays inside the margins.

As visible in Figure 3 the LMPC is highly effective, already at the first lap it decreases the lap time by 44%.

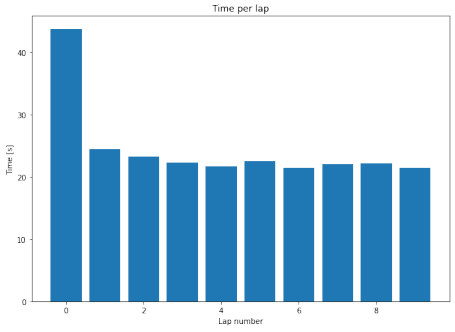

Figure 2: LMPC lap times in Milano Duckar.

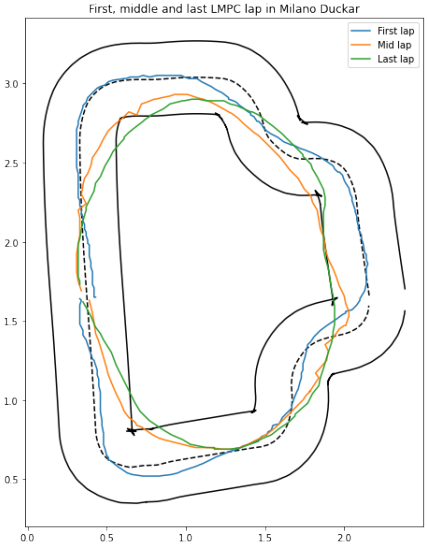

Figure 3: Significative LMPC laps in Milano Duckar.

6. Conclusions

Overall, this thesis provides compelling evidence that a learning MPC approach can be a valuable tool for autonomous navigation tasks and can potentially lead to improved performance and robustness in real-world applications. Further research is needed to explore the full potential of this approach and its integration with other control strategies and sensors.

References

[1] Maxime Chevalier-Boisvert, Florian Golemo, Yanjun Cao, Bhairav Mehta, and Liam Paull. Duckietown environments for openai gym. `https://github.com/duckietown/gym-duckietown`, 2018.

[2] Brian Paden, Michal Cáp, Sze Zheng Yong, Dmitry S. Yershov, and Emilio Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. *CoRR*, abs/1604.07446, 2016.

[3] Francesco Borelli Ugo Rosolia. Learning model predictive control for iterative tasks. a data-driven control framework. *IEEE Transactions on Automatic Control*, 2017.

[4] Francesco Borelli Ugo Rosolia. Learning how to autonomously race a car: A predictive control approach. *IEEE Transactions on Control Systems Technology, vol. 28, no. 6, pp. 2713-2719*, 2020.